Limitations of Interpretable Machine Learning Methods

Contents

List of Tables

List of Figures

Preface

This project explains the limitations of current approaches in interpretable machine learning, such as partial dependence plots (PDP, Accumulated Local Effects (ALE), permutation feature importance, leave-one-covariate out (LOCO) and local interpretable model-agnostic explanations (LIME). All of those methods can be used to explain the behavior and predictions of trained machine learning models. The interpretation methods might not work well in the following cases:

- if a model models interactions (e.g. when a random forest is used)
- if features strongly correlate with each other
- if the model does not correctly model causal relationships
- if parameters of the interpretation method are not set correctly

FIGURE 1: Creative Commons License

‘‘‘

Structure of the book

TODO

[1] http://creativecommons.org/licenses/by-nc-sa/4.0/

1

Introduction

TODO

2

Introduction to Partial Dependence Plots (PDP) and Individual Conditional Expectation (ICE)

2.1 Partial Dependence Plots (PDP)

The Partial Dependence Plot (PDP) is a rather intuitive and easy-to-understand visualization of the features' impact on the predicted outcome. It maps the marginal effect of the selected variable(s) and can reveal the nature of dependence structure between target and individual feature variable.(**?**)

The underlying function can be described as follows:

Let x_S be the set of features of interest for the PDP and x_C the complement set which contains all other features. While the general model function $f(x) = f(x_S, x_C)$ depends on all input variables, the partial dependence function marginalizes over the feature distribution in set C (**?**):

$$f_{x_S}(x_S) = \mathbb{E}_{x_C}[f(x_S, x_C)]$$

The partial dependence function can be estimated by averaging the actual feature values of x_C in the training data at given values of x_S or, in other words, it computes the marginal effect of x_S on the prediction. In order to derive realistic results, a major assumption of the PDP is that the features in x_S and x_C are independent and thus uncorrelated.(**?**)

$$\hat{f}_{x_S}(x_S) = \frac{1}{n}\sum_{i=1}^{n} f(x_S, x_C^{(i)})$$

In classification problems with probability outputs, the partial dependence function is modeled separately for all of the K different classes, i.e. it shows the probability for each respective class at given feature values of x_S.(**?**)

Advantages and Limitations of Partial Dependence Plots

Partial Dependence Plots are easy to compute and a poular way to explain

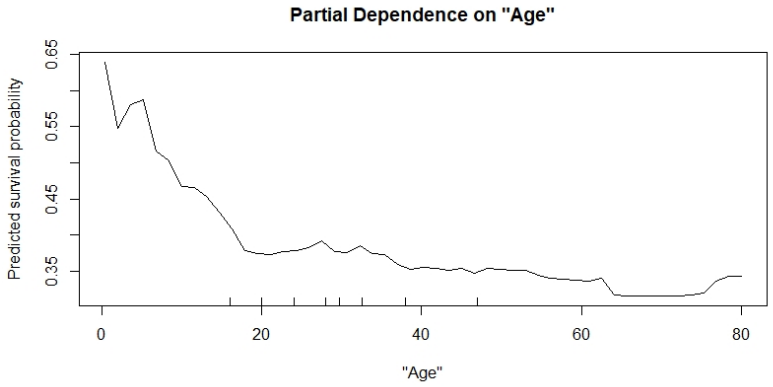


FIGURE 2.1: The PDP shows that the survival probability is sharply dropping until age 18 and more moderately afterwards.

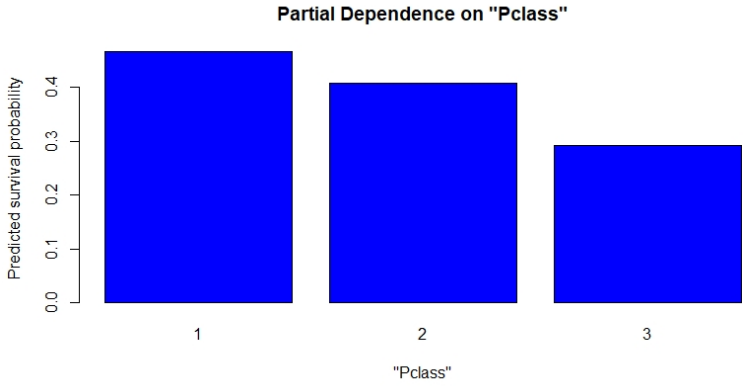


FIGURE 2.2: The classification PDP reveals that passengers in lower classes had a lower probability to survive than those in a higher class.

insights from black box Machine Learning models. With their intuitive character, PDPs perfectly qualify for the communication to non-technical audience. However, due to limited visualization techniques and the restriction of human perception to a maximum of three dimensions, only one or two features can reasonably be displayed in one PDP.(**?**)

Drawing a PDP with one or two feature variables allows a straight-forward interpretation of the marginal effects. This holds true as long as the features are not correlated. Should this assumption be violated, the partial dependence function will produce unrealistic data points. Furthermore, opposite effects of

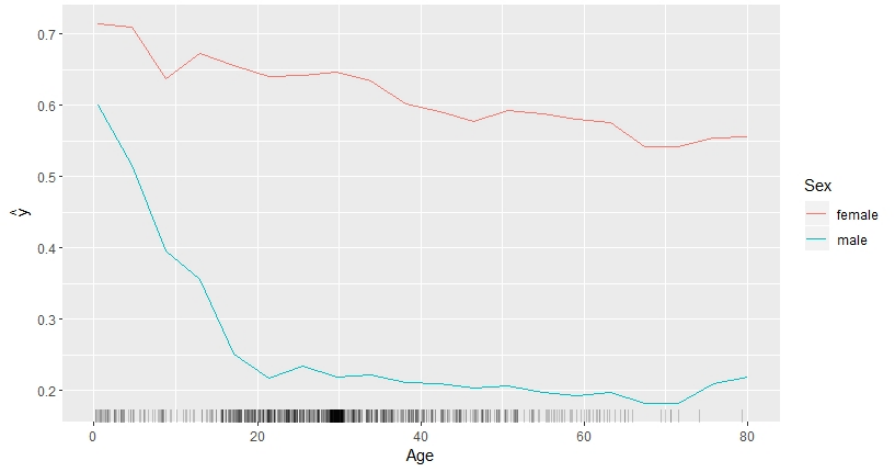


FIGURE 2.3: The two-dimensional PDP for the numerical feature Age and the categorical feature Sex shows that while the survival probability for both genders declines as age increases, that there is a difference between genders an that the decrease is much steeper for males.

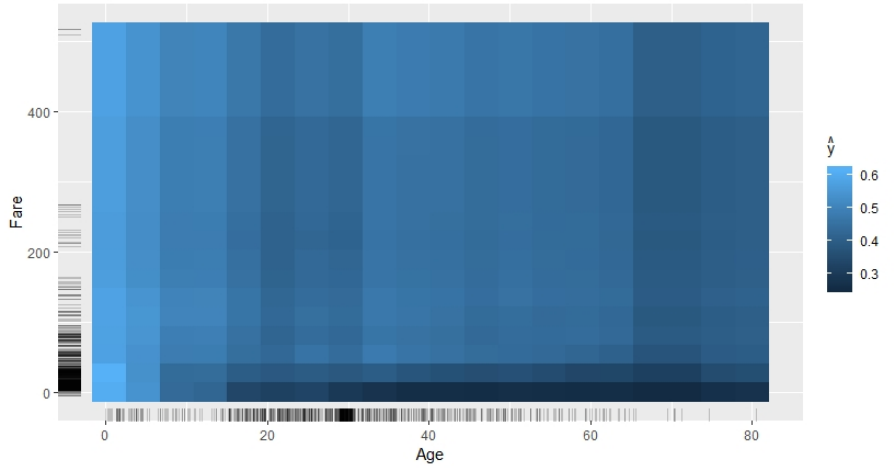


FIGURE 2.4: The two-dimensional PDP for the numerical features Age and Fare illustrates that survival probability of younger passengers is fairly uniform, whilke from age 20 onwards, passengers travelling at a lower fare also had a much lower probability to survive than those that paid a high fare.

heterogeneous subgroups might remain hidden through averaging the marginal effects, which could lead to wrong conclusions.(**?**)

2.2 Individual Conditional Expectation Curves

While the partial dependence plots provide the average effect of a feature, the Individual Conditional Expectation (ICE) plots disaggregate this average and plot the functional relationship between the predicted response and the feature for individual instances. Thus, a PDP is the average of the lines of an ICE plot.(**?**)

A formal definition: consider the response function $\hat{f}$, for each instance in $(x_S^{(i)}, x_C^{(i)})_{i=1}^N$, the curve $\hat{f}_S^{(i)}$ is plotted against the observed values of $x_S^{(i)}$, while $x_C^{(i)}$ remains fixed.(**?**)(**?**)

In ICE plots, each line represents separately one instance and shows what would happen to the model's prediction if the feature of a particular instance varied, holding all other features the same (c.p.). An ICE plot can highlight the variation in the fitted values across the range of a feature. This suggests where and to what extent heterogeneities might exist.

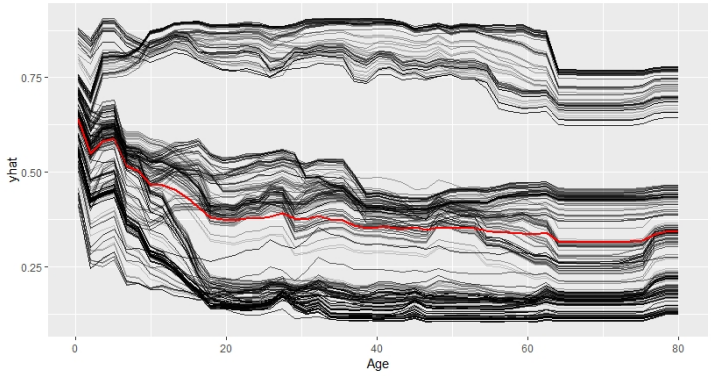


FIGURE 2.5: The ICE plot indicates that there is underlying heterogeneity in the complement set.

###Centered ICE Plot### Heterogeneity in the model can be difficult to distinguish when the curves have a wide range of intercepts and "stacked" on each other. The so called centered ICE plot (c-ICE) is a simple solution which removes level effects. The curves are centered at a certain point in the feature and display only the difference in the prediction to this point. (**?**) After

anchoring a location x^a in the range of x_s and connecting all prediction lines at that point, the new curves are defined as:

$$\hat{f}_{cent}^{(i)} = \hat{f^{(i)}} - \mathbf{1}\hat{f}(x^a, x_C^{(i)})$$

It is recommended that the most interpretable plots occur when the minimum or the maximum observed value is chosen.

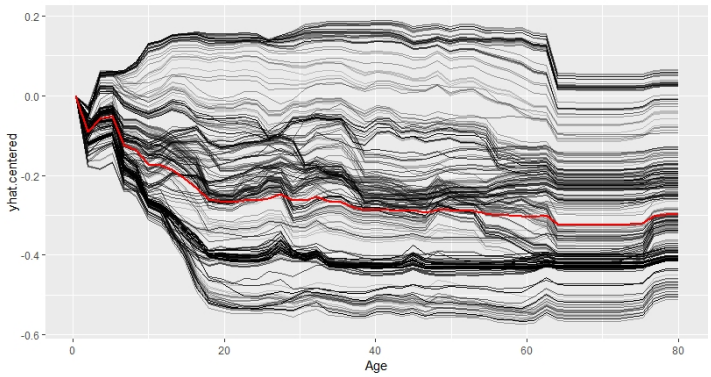


FIGURE 2.6: Centered ICE plot.

###Derivative ICE Plot### Another way to explore the heterogeneity is to show plots of the partial derivative of $\hat{f}$ with respect to x_s. Assume that x_s does not interact with the other predictors in the fitted model, the prediction function can be written as:

$$\hat{f}(x) = \hat{f}(x_s, x_C) = g(x_s) + h(x_C),$$

so that

$$\frac{\partial \hat{f}(\mathbf{x})}{\partial x_s} = g'(x_s)$$

When no interactions are present in the fitted model, all curves in the d-ICE plot are equivalent, and the plot shows a single line. When interactions do exist, the derivative lines will be heterogeneous. As it can be difficult to visually assess derivatives from ICE plots, it is useful to plot an estimate of the partial derivative directly.(**?**)

###Advantages and Limitations of ICE Plots### **Advantages** ICE plots are more intuitive than PDPs and enable data scientists to drill much deeper to explore individual differences and identify subgroups and interactions between model inputs.

Disadvantages Firstly, only one feature can be plotted in an ICE plot meaningfully. Otherwise, there will be a problem of overplotting and you would

see nothing. Secondly, ICE plots have the same problem as PDPs that some data points in the lines might be invalid. Finally, it might be difficult to see to average in ICE plots.(**?**)

3

PDP and Correlated Features

Author: firstname lastname

4

PDP and Feature Interactions

Author: firstname lastname

5

PDP and Causal Interpretation

Author: firstname lastname

6

Introduction to Accumulated Local Effects (ALE)

6.1 Motivation

As seen in section 2 PDPs don't work well as soon as two or more features are correlated. This gives rise to the definition of ALEs. Although their definition makes sense for high dimensional feature spaces including categorical features, within this section we only treat a space with two continous features.

6.2 The Theoretical Formula

The uncentered ALE with respect to a starting point $z_{0,j}$ is defined by (**?**) as

$$\widetilde{ALE}_{\hat{f},\ j}(x) = \hat{f}_{x_j,ALE}(x) = \int_{z_{0,\ j}}^{x} E_{X_c|X_j}[\hat{f}^j(X_j,\ X_c) \mid X_j = z_j]\ dz_j,$$

where $\hat{f}$ is an arbitrary prediction function on the featurespace $X = (X_j, X_c)$ (with feature of interest X_j and other features X_c) as well as its j-th partial derivative $\hat{f}^j(*,*)$.

6.2.1 Centering

The ALE (centered ALE) is defined as

$$ALE_{\hat{f},\ j}(x) = \widetilde{ALE}_{\hat{f},\ j}(x) - E_{X_j}[\widetilde{ALE}_{\hat{f},\ j}(X_j)]$$

The centering makes sense as it helps interpreting the ALE in a reasonable way. This will be explained in section 3.4..

6.3 Estimation Formula

Since this theoretical formula is of no use for a blackbox model with unknown or even non existing gradients, an approximative approach will be used. The uncentered ALE can be approximated by the formula

$$\widetilde{ALE}_{\hat{f},\ j}(x) = \int_{z_{0,\ j}}^{x} \sum_{k=1}^{K} 1_{]z_{k-1,\ j},\ z_{k,\ j}]}(x_j)\ \frac{1}{n_j(k)} \sum_{i:x_j^{(i)} \in N_j(k)} \frac{[\hat{f}(z_{k,\ j}, x_j^{(i)}) - \hat{f}(z_{k-1,\ j}, x_j^{(i)})]}{z_{k,\ j} - z_{k-1,\ j}}\ dx_j\ .$$

In a first step the relevant dimension of the feature space is divided into K intervals beginning with the starting point $z_{0,j}$. As it is not clear how to exactly divide the feature space, section 3.x deals with that question. The upper boundary of the k-th interval is denoted by $z_{k,\ j}$ as well as the lower boundary by $z_{k-1,\ j}$. $N_j(k)$ denotes the k-th interval, i.e. $]z_{k-1,\ j}, z_{k,\ j}]$ and $n_j(k)$ the total number of observations having the j-value within this interval. $x_j^{(i)}$ is the j-value of the i-th observation and correspondingly $x_j^{(i)}$ the values of the other features. The term on the right approximates the expected partial derivative within each interval. Therefore each instance within an interval is shifted to the upper and lower limit of the interval and the total difference of the prediction is calculated. Devided by the length of the interval this is a reasonable approximation for the "local" effect on the prediction, if the feature of interest changes (cet. par.). By averaging these approximations over all observations within the k-th interval, we recieve a rough estimator for the term $E_{X_c|X_j}[\hat{f}^j(X_j,\ X_c) \mid X_j \in N_j(k)]$, which we take as constant effect for the k-th interval. By integrating over this step function, which represents the locally estimated derivatives, the (local) changes are accumulated. Thats why the name Accumulated Local Effects is quite reasonable. The approximative formula for the centered ALE follows directly as

$$\widehat{ALE}_{\hat{f},\ j}(x) = \widetilde{ALE}_{\hat{f},\ j}(x) - \frac{1}{n}\sum_{i=1}^{n} \widetilde{ALE}_{\hat{f},\ j}(x_j^{(i)})\ .$$

6.3.1 Implementation Formula

As both the centered and the uncentered ALE estimations are piecewise linear functions (integration over a step function), one can first calculate the ALE at the interval boundaries and interpolate in a second step. Therefore the folowing formula proposed by (**?**, page 11) , with slightly changed notation will be useful. The definitions of its components are as above. Additionally $k_j(x)$ is defined as the number of the interval that contains x, i.e. $x \in\ ]z_{k_j(x)-1,\ j},\ z_{k_j(x),\ j}]$.

$$\widehat{\widetilde{ALE}}_{steps,\ \hat{f},\ j}(x) = \sum_{k=1}^{k_j(x)} \frac{1}{n_j(k)} \sum_{i:\ x_j^{(i)} \in N_j(k)} [\hat{f}(z_{k,\ j}, x_j^{(i)}) - \hat{f}(z_{k-1,\ j},\ x_j^{(i)})].$$

This formula returns a step function. The values in each interval are the accumulated values of the averaged total differences in each interval. To transfer this formula into the correct estimator of the uncentered ALE one has to linearely interpolate the points $(z_{k-1,\ j},\ \widehat{\widetilde{ALE}}_{steps,\ \hat{f},\ j}(z_{k-1,\ j}))$ with $(z_{k,\ j}, \widehat{\widetilde{ALE}}_{steps,\ \hat{f},\ j}(z_{k,\ j}))$ for $k \in \{1, ..., K\}$ and $\widehat{\widetilde{ALE}}_{steps,\hat{f},j}(z_{0,\ j}) = 0$.
Since in this formula there is no integral, it is easier to implement.

6.4 Intuition and Interpretation

As the former sections introduced the theoretical basics for the ALE, this section shall provide an intuition as well for the calculation method as for the interpretation. As described above, the local behaviour of the model with respect to the variable of interest is estimated by moving the existing data points to the boundaries of their interval and evaluating the total difference of the prediction for the "new" data points. Figure 6.1 first offered by (**?**) gives a good intuition for this procedure.

First one splits the total range of the variable of interest (in this case x_1) to intervals of suitable size.
For each interval the contained data points are moved to the interval boundaries. One gets twice as much "simulated" new data points as originally contained in each interval. The prediction function is now evaluated at this simulated points and the total difference of the prediction (for the given interval) is estimated as the mean change. Divided by the length of the interval one gets an estimation for the partial derivative within this interval. Theoretically one recieves the uncentered ALE by integration over this step function. Technically in a first step the total change per interval is accumulated. In a second step linear interpolation at the interval boundaries simulates a constant change within each interval. Both variants lead to the same result.

As the evaluation is ideally done on relatively small intervals, on the one hand the local behaviour of the model is estimated. On the other hand the covariance structure of the features is taken into account, as only "realistic" data points are simulated. This is in accordance with sampling from the conditional distribution.

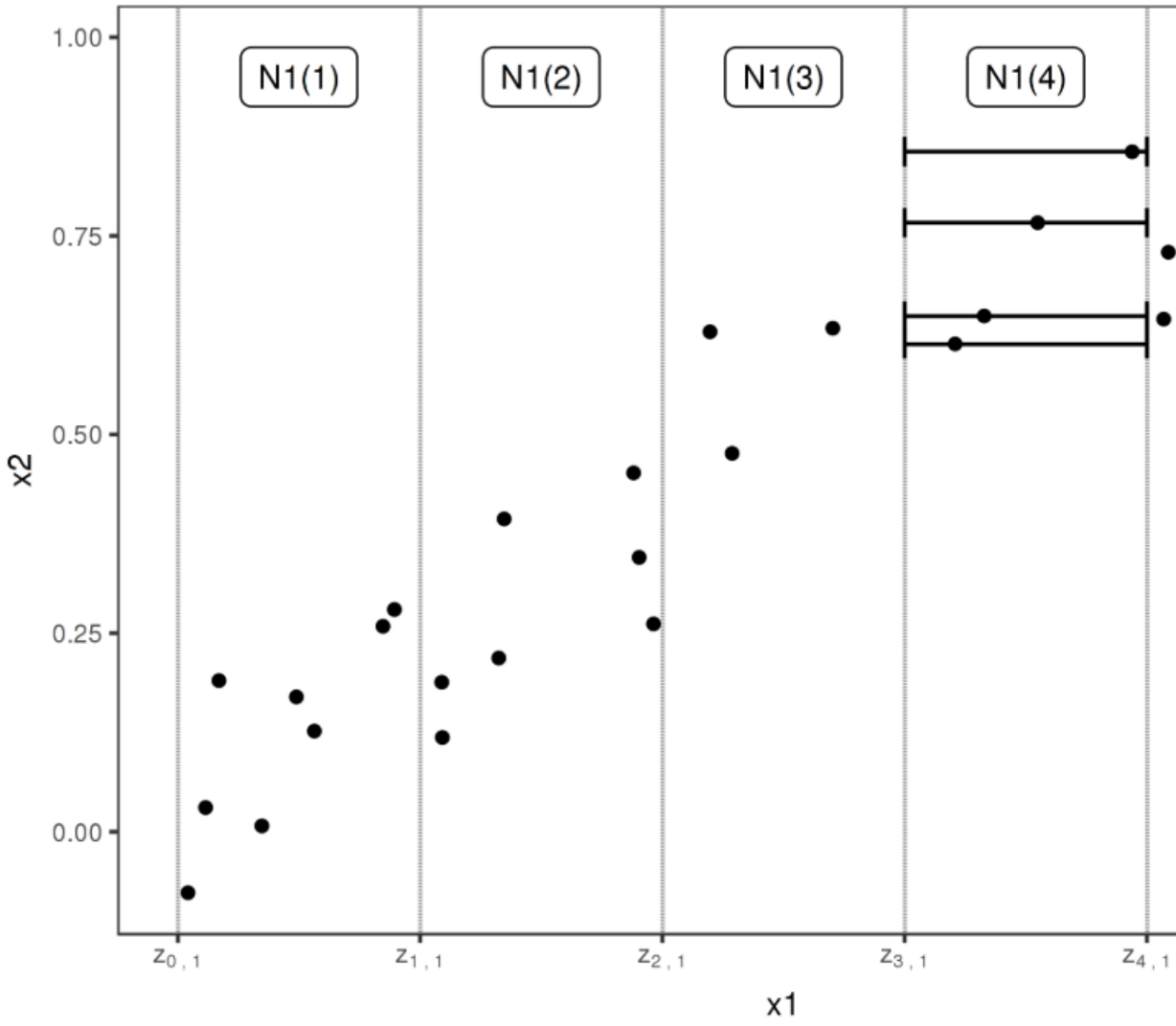


FIGURE 6.1: The data points within the 4-th interval are shifted to the interval boundaries $z_{3,\ 1}$ and $z_{4,\ 1}$.

In a last step the uncentered ALE is centered, i.e. shifted by a constant such that the expectation of the centered ALE is zero.

Figure 6.2 shows an example ALE which could match the data situtaion of Figure 6.1 :

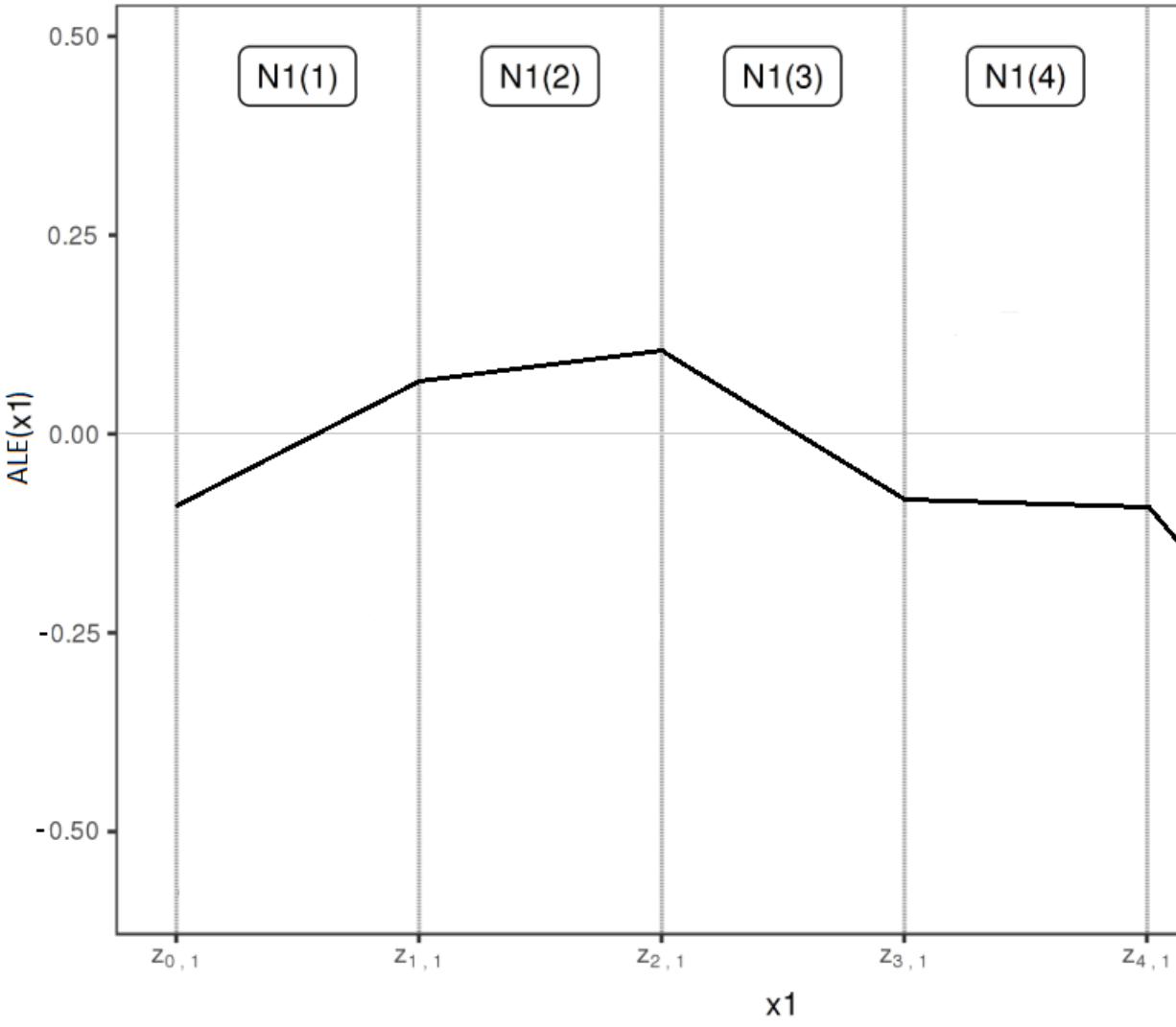


FIGURE 6.2: ALE on basis of 5 intervals

To understand the interpretation of the ALE it can be usefull to first have a look on the intuition behind the uncentered ALE. If the value of the uncen-

tered ALE at $x_1 = 2$ equals 1, this means that if one samples a data point from the joint distribution of both features but only knows that $x_1 = 2$, one would expect the prediction to be 1 higher than the average prediction for realistic data points at $x_1 = z_{0,1}$ (i.e. data points sampled from the conditional distribution at $x_1 = z_{0,1}$). This expectation strongly depends on the reference point $z_{0,1}$, which per definition is smaller than the smallest x_1-value of the data. By substracting the expectation of the uncentered ALE - which is the mean difference of the prediction of a data point from the joint distibution to the prediction of a realistic data point(i.e. from the conditional distribution) at $x_1 = z_{0,1}$ - the interpretation becomes a lot easier. If the value of the (centered) ALE at $x_1 = 2$ equals for example 2, this means that, if one samples a data point from the joint distribution of both features and x_1 equals 2, one would expect the prediction to be 2 higher than the average prediction for an average data point of the joint distribution.

So far only the case of 2-dimensional feature spaces with one feature of interest was taken into account. In the following chapters methods and interpretation for ALE with two numeric features (second order effects) or one categorical feature will be in the focus. Furthermore we will have a look on the size of the intervals the data is evaluated on, which can be crucial for the expressiveness of the ALE.

7

Comparison of ALE and PDP

Author: firstname lastname

8

ALE Intervals, Piece-Wise Constant Models and Categorical Features

Author: firstname lastname

9

Introduction to Feature Importance

As in previous chapters already discussed, there exist a variety of methods that enable a better understanding of the relationship between features and the outcome variables, especially for complex machine learning models. For instance, Partial Dependence (PD) plots visualize the feature effects on a global, aggregated level, whereas Individual Conditional Expectation (ICE) plots unravel the average feature effect by analyzing individual observations, allowing to detect, if existing, any heterogeneous relationships. Yet, these methods do not provide any insights to what extent a feature contributes to the predictive power of a model - in the following defined as feature importance. This perspective becomes interesting when recalling that so far black-box machine learning models aim for predictive accuracy rather than for inference, and hence, it is persuasive to also establish agnostic-methods that focus on the performance dimension. In the following, the two most common approaches, Permutation Feature Importance (PFI) by **?** and Leave-One-Covariate-Out (LOCO) by **?**, for calculating and visualizing a Feature Importance metric, are introduced. At this point, however, it is worth to clarify that the concepts of feature effects and feature importance can by no means be ranked but rather should be considered as mutual complements that enable the interpretability from different angles. After introducing the concepts of PFI and LOCO, a brief discussion of their interpretability but also its non-negligible limitations will follow.

9.1 Permutation Feature Importance (PFI)

The concept of Permutation Feature Importance was first introduced by **?** and applied on a random forest model. The main principle is rather straightforward and easily implemented. The idea is as follows: When permuting the values of feature j, its explanatory power mitigates, as it breaks the association to the outcome variable y. Therefore, if the model relied on the feature j, the prediction error $e = L(y, f(X))$ of the model f should increase when predicting with the "permuted feature" dataset X_{perm} instead of with the "initial feature" dataset X. The importance of feature j is then evaluated by the increase of the prediction error which can be either determined by taking

the difference $e_{perm} - e_{orig}$ or taking the ratio e_{perm}/e_{orig}. Note, taking the ratio can be favorable when comparing the result across different models. A feature is considered less important, if the increase in the prediction error was comparably small and the opposite if the increase was large. Thereby, it is important to note that when calculating the prediction error based on the permuted features there is no need to retrain the model f. This property constitutes computational advantages, especially in case of complex models and large feature spaces. Below, a respective PFI algorithm based on **?** is outlined. Note however, that their original algorithm has a slightly different specification and was adjusted here for general purposes.

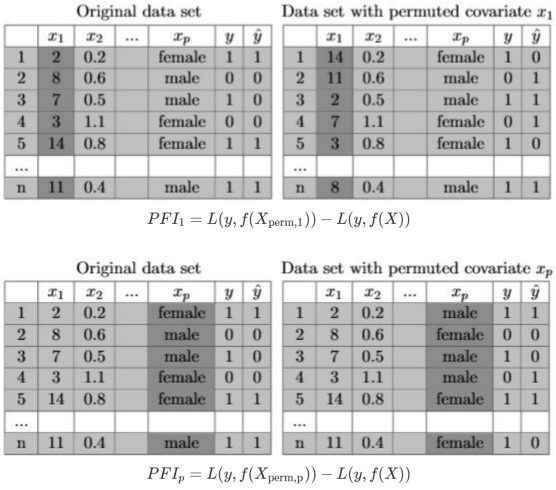
The Permutation Feature Importance algorithm based on Fisher, Rudin, and Dominici (2018):

Input: Trained model f, feature matrix X, target vector y, error measure $L(y, f(X))$

1. Estimate the original model error $e_{orig} = L(y, f(X))$ (e.g. mean squared error)
2. For each feature $j = 1, ..., p$ do:
   - Generate feature matrix X_{perm} by permuting feature j in the data X
   - Estimate error $e_{perm} = L(y, f(X_{perm}))$ based on the predictions of the permuted data
   - Calculate permutation feature importance $PFI_j = e_{perm}/e_{orig}$. Alternatively, the difference can be used: $PFI_j = e_{perm} - e_{orig}$
3. Sort features by descending FI.

In Figure 9.1 it is illustrated, by a fictional example, how the permutation algorithm alters the original dataset. For each of the p features, the respectively permuted dataset is then used to first predict the outcomes and then calculate the prediction error.

To show, how the PFI for all features of a model can be visualized and thereby more conveniently compared, the PFI algorithm with a random forest model is applied on the dataset "Boston" (see Figure 9.2), which is available in R via the `MASS` package. To predict the house price, seven variables are included, whereby as the results show, the PFIs vary substantially across the variables. The depicted points correspond to the median PFIs over all shuffling iterations of one feature and the boundaries of the bands illustrate the 0.05- and 0.95-quantiles, respectively (see `iml` package).



Original data set

	x_1	x_2	...	x_p	y	$\hat{y}$
1	2	0.2		female	1	1
2	8	0.6		male	0	0
3	7	0.5		male	1	0
4	3	1.1		female	0	0
5	14	0.8		female	1	1
...						
n	11	0.4		male	1	1

Data set with permuted covariate x_1

	x_1	x_2	...	x_p	y	$\hat{y}$
1	14	0.2		female	1	0
2	11	0.6		male	0	1
3	2	0.5		male	1	1
4	7	1.1		female	0	1
5	3	0.8		female	1	0
...						
n	8	0.4		male	1	1

$$PFI_1 = L(y, f(X_{\text{perm},1})) - L(y, f(X))$$

Original data set

	x_1	x_2	...	x_p	y	$\hat{y}$
1	2	0.2		female	1	1
2	8	0.6		male	0	0
3	7	0.5		male	1	0
4	3	1.1		female	0	0
5	14	0.8		female	1	1
...						
n	11	0.4		male	1	1

Data set with permuted covariate x_p

	x_1	x_2	...	x_p	y	$\hat{y}$
1	2	0.2		male	1	1
2	8	0.6		female	0	0
3	7	0.5		male	1	0
4	3	1.1		male	0	1
5	14	0.8		female	1	1
...						
n	11	0.4		female	1	0

$$PFI_p = L(y, f(X_{\text{perm},p})) - L(y, f(X))$$

FIGURE 9.1: Example for Permutation Feature Importance

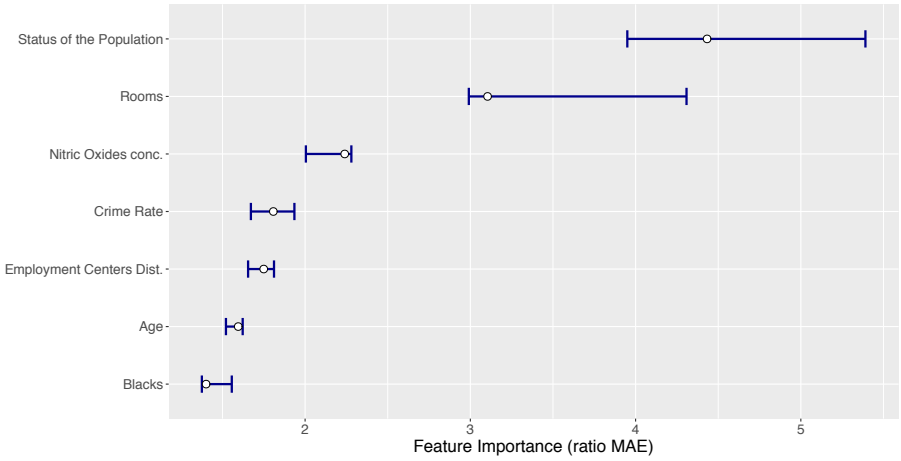


FIGURE 9.2: Visualization of Permutation Feature Importance with a random forest applied on Boston dataset

9.2 Leave-One-Covariate-Out (LOCO)

The concept of Leave-One-Covariate-Out (LOCO) is as already mentioned a different approach to PFI with the same objective, to gain insights on the importance of a specific feature for the prediction performance of a model. Although applications of LOCO exist, where comparable to PFI, the initial values of feature j are replaced by its mean, median or zero (see **?**), and hence, circumvent the disadvantage of re-training the model f, the common approach follows the idea to simply leave the respective feature out. The overall prediction error of the re-trained model f_{-j} is then compared to the prediction error resulted from the baseline model. However, re-training the model results in higher computational costs, becoming more severe with an increasing feature space. Besides, as one is actually interested in assessing the Feature Importance of a fixed model f, comparing the performance of a fixed model with the performance of a model f_{-j} fitted merely with a subset of the data, might raise plausible concerns (see **?**). The pseudo-code shown below, illustrates the algorithm for the common case where the feature is left out (see **?**).

The Leave-One-Covariate-Out algorithm based on Lei et al. (2018):

Input: Trained model f, feature matrix X, target vector y, error measure $L(y, f(X))$

1. Estimate the original model error $e_{orig} = L(y, f(X))$ (e.g. mean squared error)
2. For each feature $j = 1, ..., p$ do:
   - Generate feature matrix X_{-j} by removing feature j in the data X
   - Refit model f_{-j} with data X_{-j}
   - Estimate error $e_{-j} = L(y, f_{-j}(X_{-j}))$ based on the predictions of the reduced data
   - Calculate LOCO Feature Importance $FI_j = e_{-j}/e_{orig}$. Alternatively, the difference can be used: $FI_j = e_{-j} - e_{orig}$
3. Sort features by descending FI.

In Figure 9.3 it is shown, how the LOCO algorithm alters the original dataset, whereby it always differs, depending on the respective feature that is left out. Note, that the qualitative and quantitative interpretations correspond the ones from the PFI method. So do the visualization tools and therefore at this point it is refrained from providing the reader with an additional real data example.

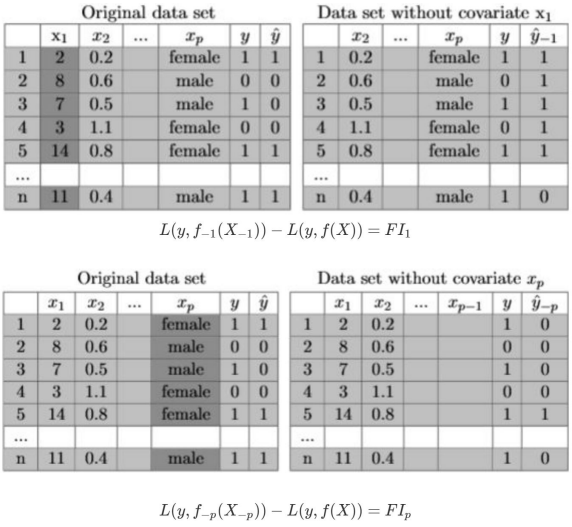


FIGURE 9.3: Example for Leave-One-Covariate-Out Feature Importance

9.3 Interpretability of Feature Importance and its Limitations

After both methods are discussed, the question to what extent these agnostic-methods can contribute to a more comprehensive interpretability of machine learning models will be now touched on. By doing so, it is necessary to also reflect upon the limitations of the model regarding the interpretability of the results. The latter will constitute the main focus in the following chapters. Conveniently, both methods are highly adaptable on whether using classification or regression models, as they are non-rigid towards the prediction error metric (e.g. Accuracy, Precision, Recall, AUC, Average Log Loss, Mean Absolute Error, Mean Squared Error etc.). This allows to assess Feature Importance based on different performance measures. Besides, the interpretation can be conducted on a high-level, as both concepts do consider neither the shape of the relationship between the feature and outcome variable nor the direction of the feature effect. However, as illustrated in Figure 9.2, PFI and LOCO only return for each feature a single number and thereby neglect possible variations between subgroups in the data. Chapter XX will focus on how this limitation can be, at least for PFI, circumvented and introduces the concepts of Partial Importance (PI) and Individual Conditional Importance (ICI) which both avail themselves on the conceptual ideas of PD and ICE (see **?**). Besides, two general limitations appear when some features in the feature space are corre-

lated. First, correlation makes an isolated analysis of the explanatory power of a feature complicated which results in an erroneous ranking in Feature Importance and hence, in incorrect conclusions. Second, if correlation exists and only in case of applying the PFI method, permuting a feature can result in unrealistic data instances so that the model performance is evaluated based on data which is never observed in reality. This makes comparisons of prediction errors complicated and therefore it should always be checked for this problem, if applying the PFI method. Chapter XX will focus on this limitations by comparing the performance of PFI and LOCO for different models and different levels of correlation in the data. Beyond these limitations, it is evident to also question whether these agnostic-methods should be computed on training or test data. As answering that depends highly on the research question and data, it is refrained from going into more detail at this point but will be examined and further discussed in chapter XX.

10

PFI, LOCO and Correlated Features

Author: firstname lastname

11

Partial and Individual Permutation Feature Importance

Author: firstname lastname

12

PFI: Training vs. Test Data

Author: firstname lastname

13

Introduction to Local Interpretable Model-Agnostic Explanations (LIME)

When doing machine learning we always build models. Models are simplifications of reality. Even if the predictive power of a model may be very strong, it will still only be a model. However, models with high predictive capacity do most of the time not seem simple to a human as seen throughout this book. In order to simplify a complex model we could use another model. These simplifying models are referred to as surrogate models. They imitate the black box prediction behaviour of a machine learning model subject to a specific and important constraint: surrogate models are interpretable. For example, we may use a neural network to solve a classification task. While a neural network is anything but interpretable, we may find that some of the decision boundaries are explained reasonably well by a logistic regression which in fact yields interpretable coefficients.

In general, there are two kinds of surrogate models: global and local surrogate models. In this chapter, we will focus on the latter ones.

13.1 Local Surrogate Models and LIME

The concept of local surrogate models is heavily tied to **?**, who propose local interpretable model-agnostic explanations (LIME). Different from global surrogate models, local ones aim to rather explain single predictions by interpretable models than the whole black box model at once. These surrogate models, also referred to as explainers, need to be easily interpretable (like linear regression or decision trees) and thus may of course not have the adaptability and flexibility of the original black box model which they aim to explain. However, we actually don't care about a **global** fit in this case. We only want to have a very **local** fit of the surrogate model in the proximity of the instance whose prediction is explained.

A LIME explanation could be retrieved by the following algorithm:

1. Get instance x out of the data space for which we desire an explanation for its predicted target value.
2. *Perturb* your dataset X and receive a perturbed data set Z of increased size.
3. Retrieve predictions $\hat{y}_Z$ for Z using the black box model f.
4. Weight Z w.r.t. the proximity/neighbourhood to x.
5. Train an explainable weighted model g on Z and the associated predictions $\hat{y}_Z$.

Return: An explanation for the interpretable model g.

The visualisation below nicely depicts the described algorithm for a two-dimensional classification problem based on simulated data. We start only with our data split into two classes: 1 and 0. Then, we fit a model that can perfectly distinguish between the two classes. This is indicated by the sinus-shaped function drawn as a black curve. We do not perturb the data in this case. (However, we may argue that our perturbation strategy is to use the original data. We will more formally discuss perturbation later on.) Now, we choose the data point, which we want an explanation for. It is coloured in yellow. With respect to this point, we weight our data by giving close observations higher weights. We illustrate this by the size of data points. Afterwards, we fit a classification model based on these weighted instances. This yields an interpretable linear decision boundary – depicted by the purple line. As we can see, this is indeed locally very similar to the black box decision boundary and seems to be a reasonable result.

This way we receive a single explanation. This one explanation can only help to understand and validate the corresponding prediction. However, the model as a whole can be examined and validated by multiple (representative) LIME explanations.

13.2 How LIME works in detail

So far so good. However, the previous outline was not very specific and leaves (at least) three questions. First, what does neighbourhood refer to? Second, what properties should suitable explainers have? Third, what data do we use, why and how do we perturb this data?

To better assess these open questions it may be helpful to study the math-

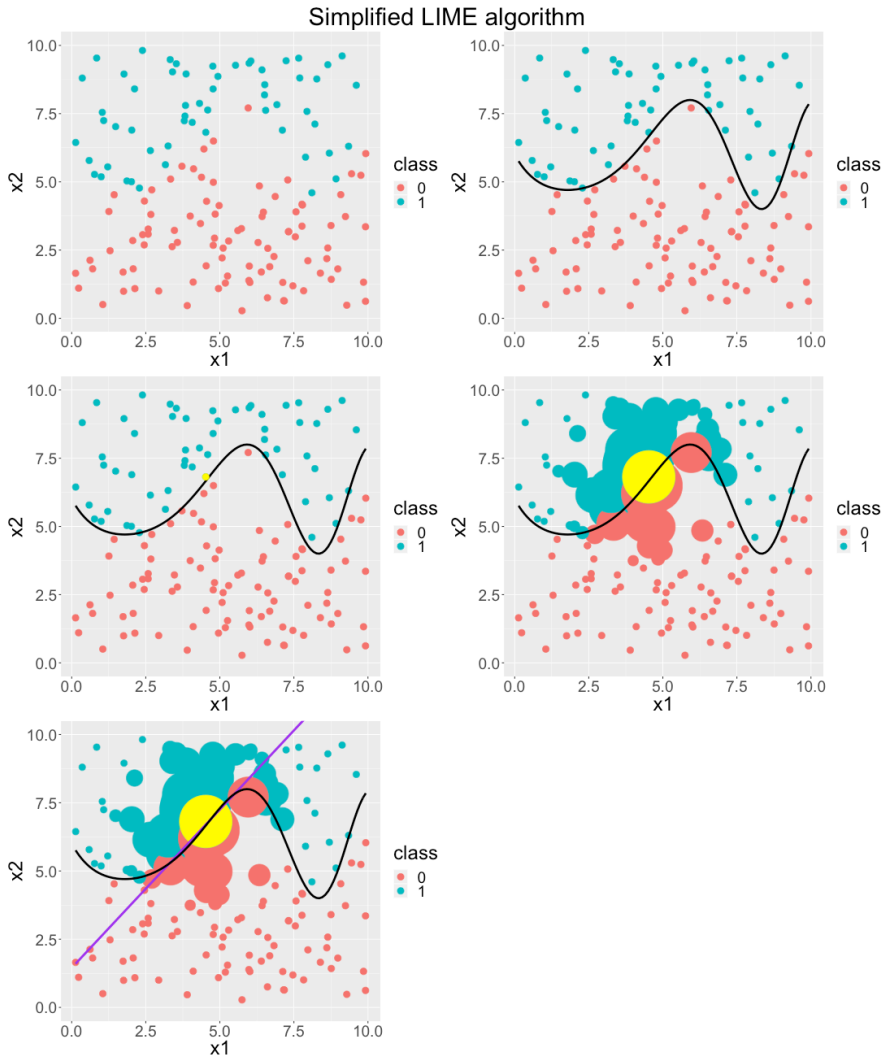


FIGURE 13.1: Simplified graphical representaion of the LIME algorithm. Each single panel represents one step of the described algorithm. It reads from left to right.

ematical definition of $LIME$. The explanation for a datapoint x, which we aim to interpret, can be represented by the following formula:

$$explanation\,(x) = arg\,min_{g \epsilon G}\,\mathcal{L}\,(f, g, \pi_x) + \Omega\,(g)$$

Let's decompose this compact, yet precise definition:

x can be an instance that is entirely new to us as long as it can be represented in the same way as the training data of the black box model. The final explanation for x results from the maximisation of the loss-like fidelity term $\mathcal{L}\,(f, g, \pi_x)$ and a complexity term $\Omega\,(g)$. f refers to the black box model we want to explain and g to the explainer. G represents the complete hypothesis space of a given interpretable learner. The explanation has to deal with two trade-off terms when minimising: The first term $\mathcal{L}\,(f, g, \pi_x)$ is responsible to deliver the optimal fit of g to the model f while a low *loss* is desirable indicating high (local) *fidelity*. The optimal fit is only found with respect to a proximity measure $\pi_x(z)$ in the neighbourhood of x.

13.2.1 Neighbourhood

This leads us to the first open question: What does neighbourhood refer to? Neighbourhood is a very vague term. This is for good reason because a priori it is not clear how to specify a neighbourhood properly. Technically, there are many different options to deal with this issue. Weighting the observations w.r.t. their distance to the observation being explained seems like a good idea. This may possibly be implemented as an arbitrarily parametrised kernel. However, this leaves in total many scientific degrees of freedom which makes the neighbourhood definition somewhat problematic. This neighbourhood issue will be discussed in more detail in the next chapter.

13.2.2 What makes a good explainer?

We already answered the second open question – what properties suitable explainers should have – in parts. We mentioned the interpretability property and outlined generalised linear models or decision trees as examples. However, we did not discuss further desired properties of these models. Since they have strong assumptions, it is unlikely that they are capable of maintaining an optimal fit to the original black box model. Recall our formula. As we want local optimal fit subjected to a certain (low) degree of explainer complexity – in order to allow interpretation – our formula needs to facilitate this aspect. $\Omega\,(g)$ is our complexity measure and responsible to choose the model with the lowest complexity. For example, for decision trees, tree depth may describe the complexity. In the case of linear regression, the L_1 norm may indicate how

simple the interpretation has to be. The resulting LASSO model allows us to focus only on the most important features.

13.2.3 Sampling and perturbation

Having answered the first two open question we still have the last question related to the data and the perturbation unresolved. Besides the tabular data case, we can also interpret models trained on more complex data, like text data or image data. However, some data representations (e.g. word embeddings) are not human-interpretable and must be replaced by interpretable variants (e.g. one-hot-encoded word vectors) for LIME to yield interpretable results. The function modeled by the black box model operates in the complete feature space. It can even yield predictions for instances not seen in the training data. This means that the original data does not sufficiently explore the feature space. Hence, we want to create a more complete *grid* of the data and fill the feature space with new observations so that we can better study the behaviour of the black box model. Still, the data for the explainer should be related to the original data. Otherwise the explainer may be ill-placed in space having nothing in common with the original problem anymore. This is why we perturb the original data. But how does perturbation work? This is a priori not clear at all. For categorical features, perturbation may be realised by randomly changing the categories of a random amount of features, or even recombining all possible levels of these. Numerical features may be drawn from a properly parametrised (normal) distribution. The perturbed data set, which is used to train the explainer, should be much larger than the original one and supposed to better represent the (possible) feature space, giving the surrogate model more anchor points – especially in sparse areas. Further details on this topic will be studied in chapter X.

13.3 Example

A fully implemented example of LIME can be seen in the following code block with its resulting plot. In the latter we can observe how much each feature contributes to the surrogate model's prediction and to what extend this prediction offers a good fit on the black box model ('Explanation Fit' between 0 and 1).

```
required_packages <- c("lime", "mlr")
install_these <-
  required_packages[!(required_packages %in% installed.packages())]
```

```
if (length(install_these) > 0) install.packages(install_these)
library(lime)
library(mlr)

# separate data point we want to explain
to_explain  = iris[ 1, 1:4]
train_set   = iris[-1, ]

# create task and calculate black box model
task_iris   = makeClassifTask(data = train_set, target = "Species")
learner     = makeLearner("classif.randomForest", ntree = 200, predict.type = "prob")
black_box   = train(learner, task_iris)

# use lime to explain new data point
explainer   = lime(train_set[, 1:4], black_box)
explanation = explain(to_explain,
                      explainer,
                      n_labels = 1,
                      n_features = 4)

plot_features(explanation)
```

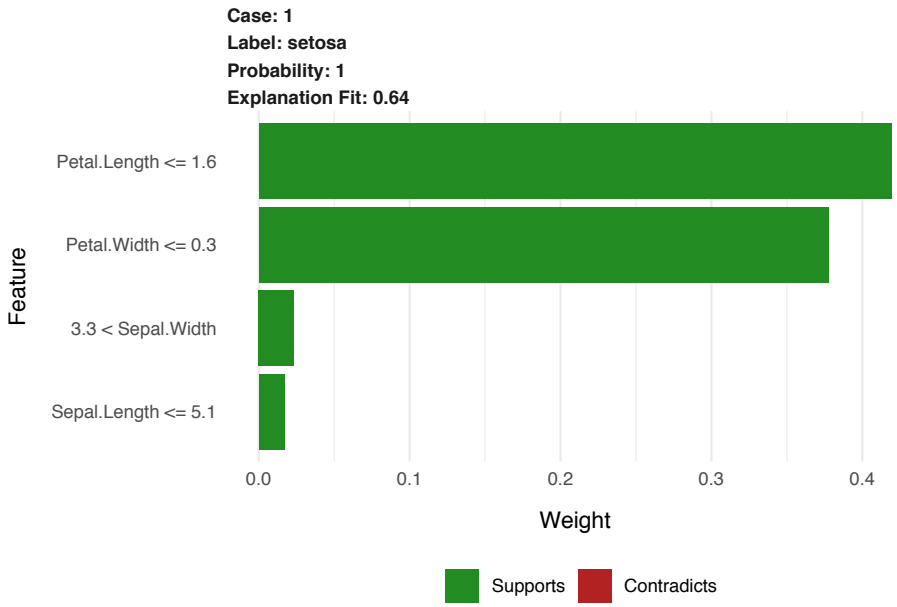


FIGURE 13.2: Basic example of a LIME application. We create a black box model on the iris dataset without the first data point and then explain the prediction of this point with LIME.

13.4 Outlook

The definition of LIME still seems after all very rough and vague. This leaves us many scientific degrees of freedom when implementing it – for the good and for the bad. For example, we see that the model f can be any machine learning model that exists. This gives us the opportunity to drastically change the underlying predictive model while keeping the same explainer g with the same complexity constraints.

On the other hand, LIME being a very generic approach also means that many "hyperparameters", like the neighbourhood definition or the sampling/perturbation strategy, are arbitrary. Hence, it is likely that in some use cases LIME explanations heavily depend on changing the hyperparameters. In these cases, the explanations can hardly be trusted and should be treated with great care.

The following two chapters will focus on two very significant hyperparameters: the neighbourhood definition and the sampling strategy. They will investigate how these affect the results of the method and their interpretability. We will emphasise the coefficient stability of LIME explainers in order to illustrate the trustworthiness of the results.

14

LIME and Neighbourhood

Author: Philipp Kopper

How much neighbourhood may be an issue can be illustrated in a setting very different to LIME: Descriptive statistics or in particular kernel density estimations. The figure below illustrates kernel densities from a standard normal distribution. One can easily see that the second panel seems to be appropriate while the first one is too granular and the third one does not feature any information. The proper definition of the neighbourhood is in this case very crucial. However, with no prior information this definition is arbitrary. We can only judge on the proper definition of the neighbourhood from our experience and our expectations. This may work in low dimensional problems and descriptive statistics. However, machine learning models operate in multivariate space and mostly tackle complex associations. Thus, it seems much harder to argue on the proper neighbourhood definition when working with LIME.

This chapter reviews the neighbourhood issue w.r.t. the LIME algorithm critically. The objective of this chapter is more to outline this particular issue and not to suggest solutions for it. First of all, it describes the neighbourhood defintion abstractly (section 1). Then, it reviews possible implementations with special emphasis on the implementation in Python and R (section 2). Section 3 discusses inherent problems and potential issues of the current implementations. In section 4 we examine the coefficient stability of LIME explanations w.r.t to altering the neighbourhood definition within an experiment in R. We use both, simulated and real data. We make use of the stability selection algorithm which is explained in section 4.1. Finally, section 5 concludes reviewing the take-aways of this chapter and proposing an improved algorithm w.r.t. the neighbourhood issue.

14.1 The Neighbourhood in LIME

When obtaining explanations with LIME the neighbourhood of an instance is determined when fitting the model by applying weights to instances w.r.t. proximity to the instance of interest. However, it is important to note that

this step is already arbitrary. **?** show that increasing the density of observations around the instance of interest is very helpful to achieve model fidelity. This could be obtained in many other ways than weighting observations as in LIME. One possible alternative might be to combine steps 2 (sampling) and 4 (weighting) of the LIME algorithm. This way we would increase the density around the instance already by proper sampling. In fact, **?** claim that this way should be preferred over the way LIME samples. In this chapter, however, we focus on the explicit implementation in LIME and analyse how the weighting strategy *ceteris paribus* affects surrogate model stability.

In LIME the weighting of instances is performed using a kernel function over the distances of instances to the instance of interest. This leaves us two *arbitrary* (in fact they may not be that arbitrary) choices: the distance and the kernel function. Typical distance functions applicable to statistical data analysis are based on the L0, L1 and L2 norms. For numerical features one tends to use either manhattan distance (L1) or euclidean distance (L2). For categorical features one would classically apply hamming distance (L0). Mixed data (data with both categorical and numerical features) usually one combines distances for numerical and categorical features. So does Gower's distance **?**

$$d_G(x_i, x_j) = \sum_{p=1}^{P} \frac{d_{euc}(x_{p,i}, x_{p,j})}{range(x_p)}$$

or a bit more general the distance proposed by **?**

$$d_H(x_i, x_j) = d_{euc}(x_i, x_j) + \lambda d_{ham}(x_i, x_j)$$

with d_{euc} refering to the euclidean distance and d_{ham} to the hamming distance. λ steers the importance of categorical features relative to numerical ones. However, it is important to note that despite these mesasures it may be challenging to properly determine distances for mixed data properly. **?** recommend using cosine distance for text, euclidean distance for images.

For the kernel function itself there are two hyperparameters to be set. First of all the type of the kernel. Second, the kernel width. By default the R implementation implements an exponential kernel where the kernel width equls the square root of the number of features.

As the choice of the distance measure seems least arbitrary and the choice of the kernel function is not expected to have crucial impact on the neighbourhood definition, we focus on the kernel width in our experimental study. However, it is very important to notice that practically the distance measure may interact with the kernel width. This is because as of the current implementation in R Gower distance does not apply any kernel as it already returns scaled distances. We observed that this property may have some undesired properties as global models seem to be preferred over local models. We

suggest to implement a kernel function also for the Gower distance. We are currently working on this issue in collaboration with the package owner.

14.2 The Problem in a one-dimensional setting

How crucial the kernel width is can be illustrated by a very simple example. We simulate data with one target and two features. One feature is purely noise and the other one has a non-linear sinus-like effect on the target. If we plot the influential feature on the x-axis and the target on the y-axis we can observe this pattern in figure 1.

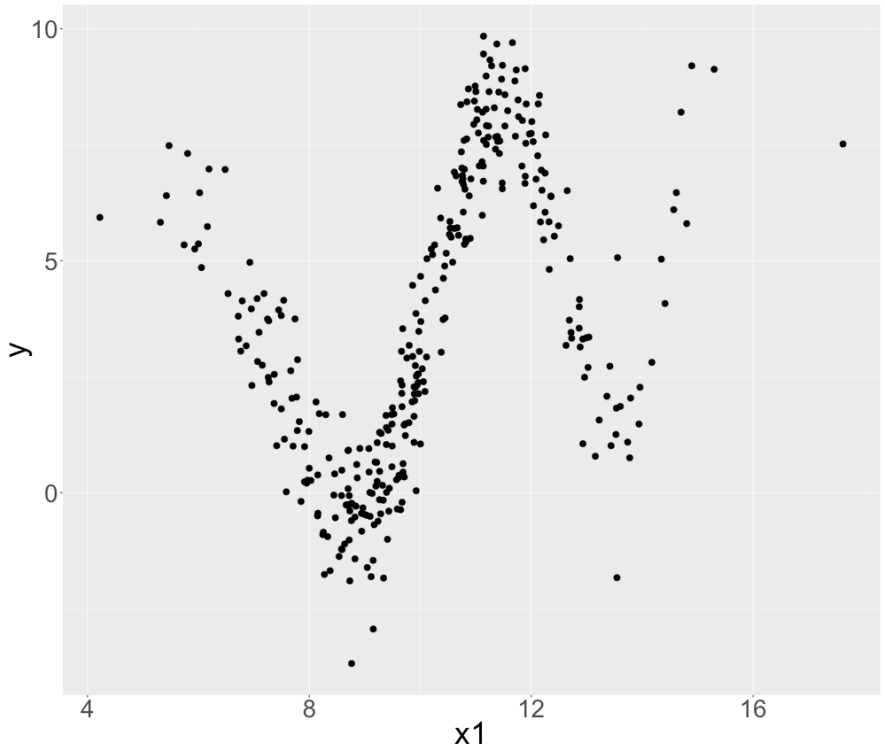


FIGURE 14.1: Simulated data: Non-linear univariate relationship.

Now we fit a random forest on this problem which should be able to detect the non-linearity and incorporate it into its predictive surface. In fact, we observe that the predictions of the random forest look very accurate in figure 2.

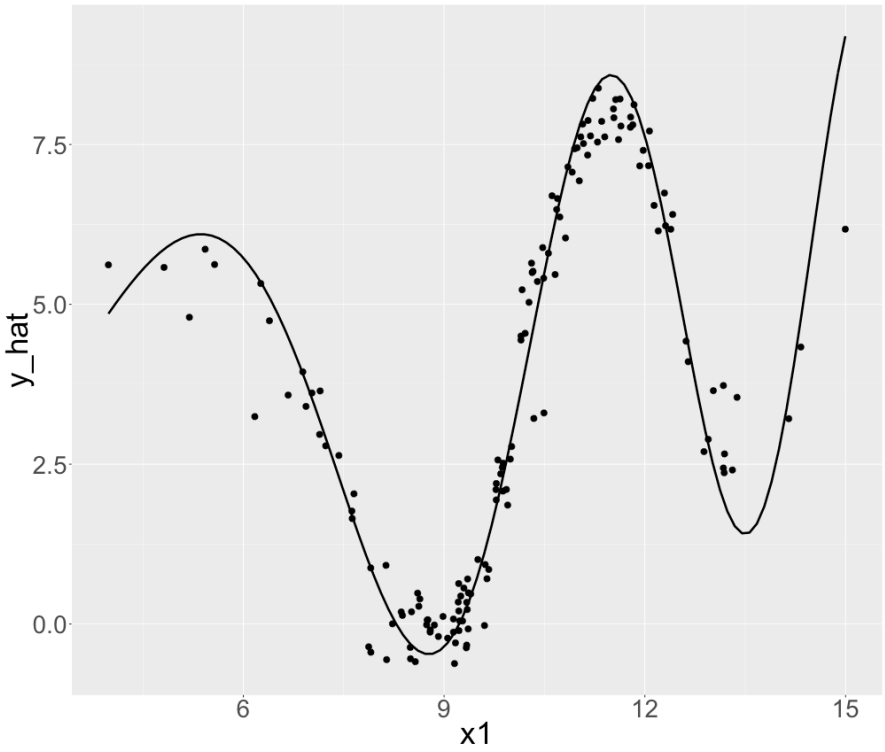


FIGURE 14.2: Simulated data: Random Forest Predictions.

LIME could now be used to explain this random forest locally. "Good" local models would look very different w.r.t. the value of the feature. We could describe the data locally well by piece-wise linear models. This is depicted in the figure 3.

LIME should be able to find these good local explanations given the right kernel size. Let's select one instance that is close to the predictive surface. We indicate this by the green point in figure 4. This particular instance can be linearly described by a linear regression with approximately intercept 60 and slope -4.5. If we set the kernel width to 0.08, we actually fit this local model (on average). This is indicated by the red line in figure 4. However, if we increased the kernel width to 2 the coefficients change to -2.84 (intercept) and 0.64 (slope) (on average) which seems drastically distorted as observed by the yellow line in figure 4. The second line does not seem to fit a local linear model but rather a global one.

More systematically we review explanations resulting from altering the kernel size in figure 5. We average over many different models to achieve more robust local models. We do that because we observe some coefficent variation result-

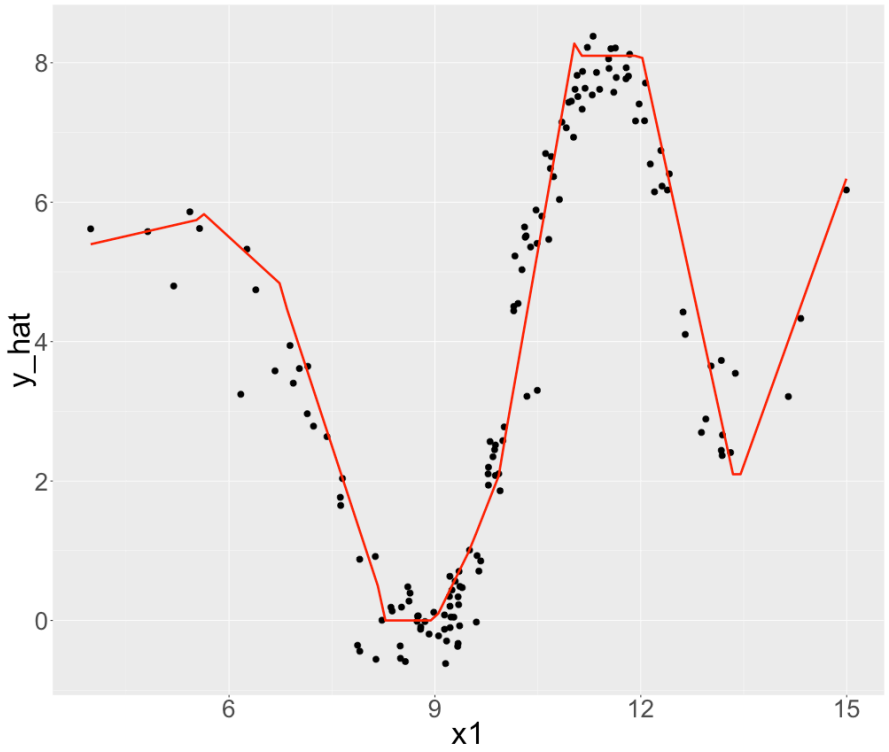


FIGURE 14.3: Simulated data: Non-linear univariate relationship linearly interpolated.

ing from the (random) sampling. In figure 5 we see these averaged models for 8 different kernel sizes. We observe that the larger we set the kernel size, the more we converge to a linear model that operates globally. In fact, the largest three kernel sizes (0.5, 1 and 2) appear very global while 0.005, 0.05 and 0.1 seem to fit good local models. 0.25 and 0.3 are neither global nor very local. This is very intuitive and complies with the idea of a weighted local regression.

Additionally, we analyse the same alteration of the kernel size for an observation where a good local approximation would be a linear model with positive slope in figure 6. We observe a similar behaviour.

This behaviour is not necessarily a problem but only a property of the LIME. However, it is problematic that the appropriate kernel size is not a priori clear. Additionally, there is no straight forward way to determine a good kernel width for a given observation to be explained. The only generic goodness-of-fit criterion of LIME, model fidelity, is endogenous w.r.t. the kernel size. If we set the kernel size very small there will be many models with extremely good local fit as local refers only to a single observation. In our examples it looks

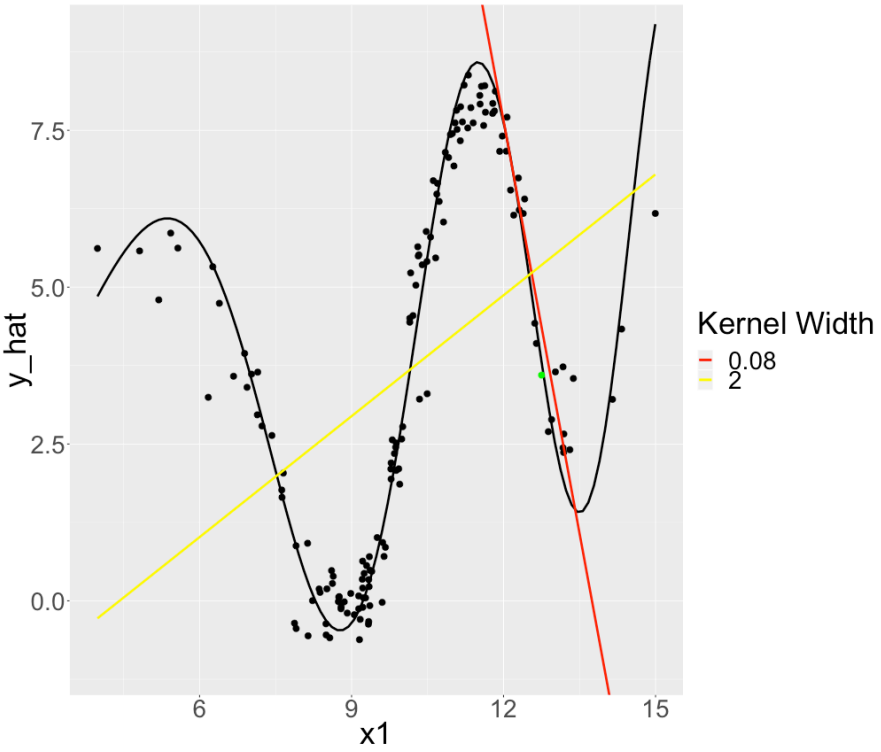


FIGURE 14.4: Simulated data: Local models for univariately non-linear data.

as if a very small kernel sizes should be prefered. A small kernel width indeed grants local fit. But what a small kernel width is also strongly depends on the dimensionality and complexity of the problem.

14.3 The problem in more complex settings

The previous setting was trivial for LIME. We could visualise the predicitve surface. This means that interpretability was largely given in the first place. We will study our problem in more complex settings to show that the problem persists. We will do so by examining simulated and real data.

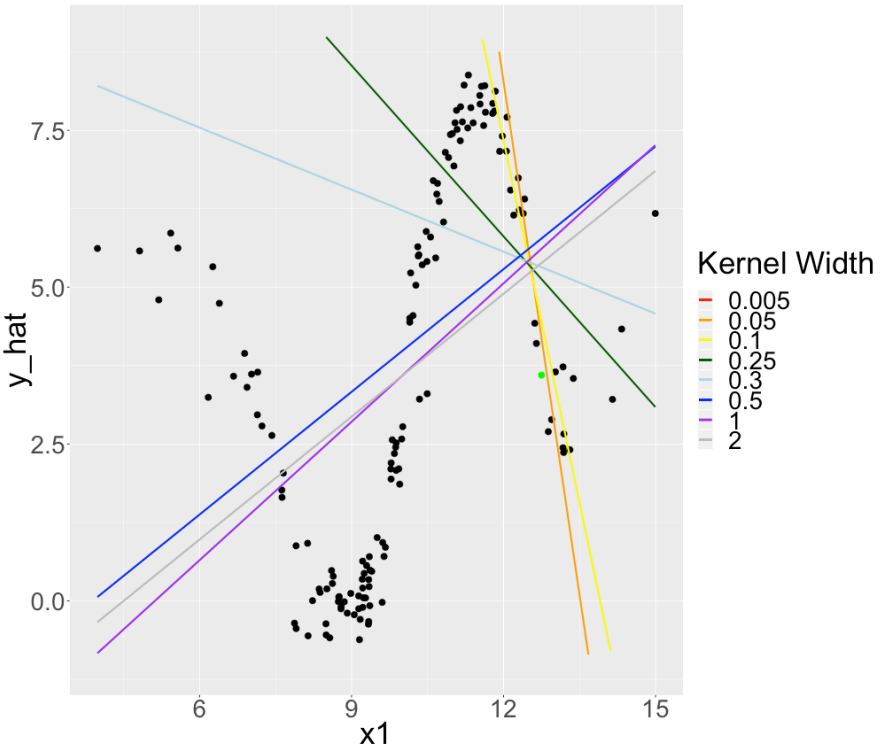


FIGURE 14.5: Simulated data: Local models for univariately non-linear data with different kernel sizes.

14.3.1 Simulated data

We simulate data with multiple numeric features and a numeric target. We assume the features to originate from a multivariate normal distribution where all features are moderately correlated. Note that the choice of the normal distribution to be consistent with the assumptions within LIME. We simulate three different data sets. In the first one the true associations are linear. In the second one we exchange one of the linear associations with a non-linear one. In the third one the true assocations are linear but only affect the target within a sub interval of the feature domain. This should examine LIME's ability to determin local feature importance. For all three data sets we expect the kernel width to have crucial impact on the resulting explainer. However, for the global linear relationships we expect the weakest dependency because the true local model and the true global model are identical.

We refrain from simulating more complex data sets as we strongly favour that the true marginal predicitive surface is human interpretable.

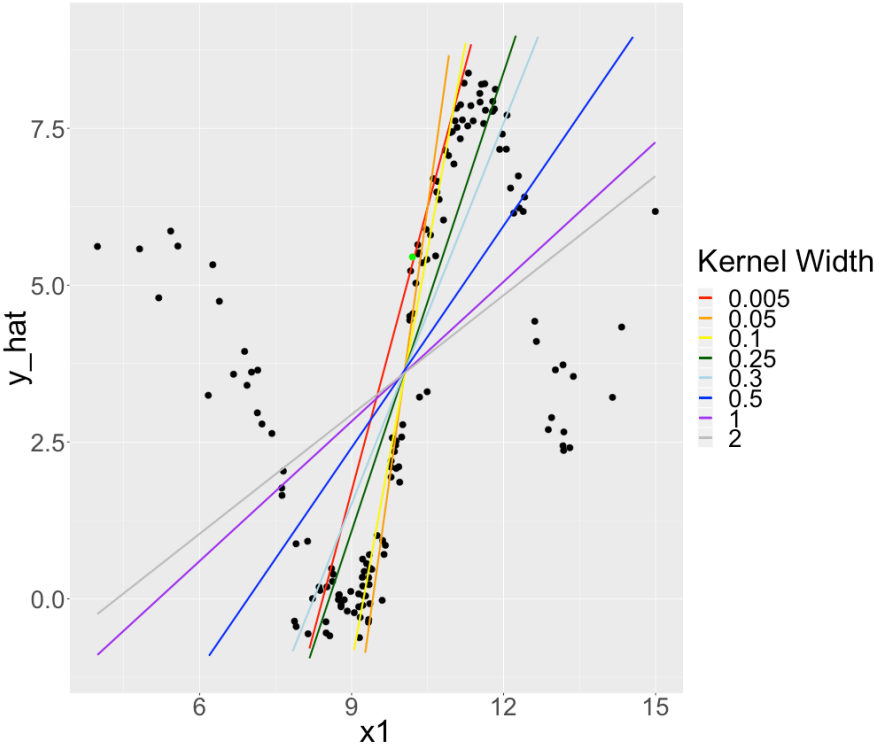


FIGURE 14.6: Simulated data: Local models for univariately non-linear data with different kernel sizes.

14.3.1.1 Global Linear Relationships

We simulate data where the true predicitve surface is a hyperplane. Good machine learning models should be able to approximate the hyperplane. This case is somewhat trivial because LIME. The most suitable model for this data would be linear regression which is interpretable in the first place. However, LIME can be easily tested in this controlled environment. We know the true local coefficients as they equal to the global ones. Thus, we can evaluate the goodness of the kernel width appropriatley.

The simulated data looks as follows: The feature space consists of three features (x_1, x_2, x_3). All originate from a multivariate Gaussian distribution with mean μ and covariance Σ. μ is set to be 5 for all features and Σ incorporates moderate correlation. The true relationship of the features on the target y is described by:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon$$

We set the true coefficients to be $\beta_1 = 4$, $\beta_2 = -3$, $\beta_1 = 5$.

We use a random forest as black box model. Below we indicate the test set performance of the model in figure 7. It seems as if the random forest largely succeeds in finding the true association between the features and the target.

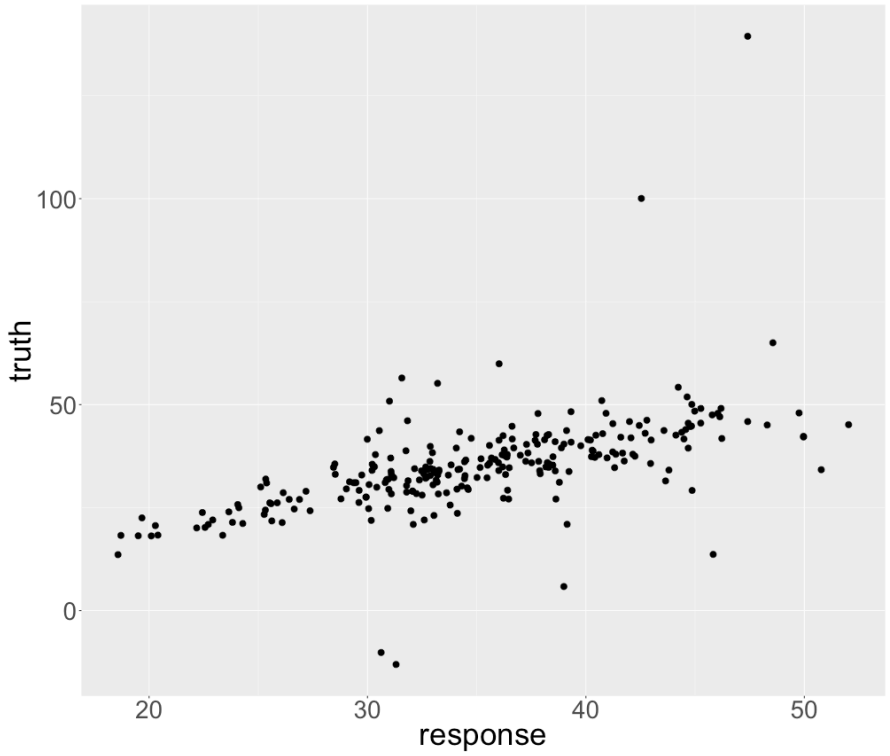


FIGURE 14.7: Simulated data: Model evaluation for random forest.

We choose random observations and compute the local LIME model for each one w.r.t. different kernel sizes. In this case we assume that the kernel size may be infinitely large as the global model should equal good local models. However, if the kernel width is set too small we may fit too much noise. Hence, in this case we may find no good local models.

The figures below (figure 8) indicate the local parameters for one of the selected observations for different kernel sizes which have been determined by LIME. Note that this is representative for all observations.

We observe that too small kernel widths are not able to reproduce the global predictive surface at all. However, moderately all kinds of kernel widths from moderate size to very large kernels fit very similar models which are all very close to the *true* model where **qualitative** model explanations remain identi-

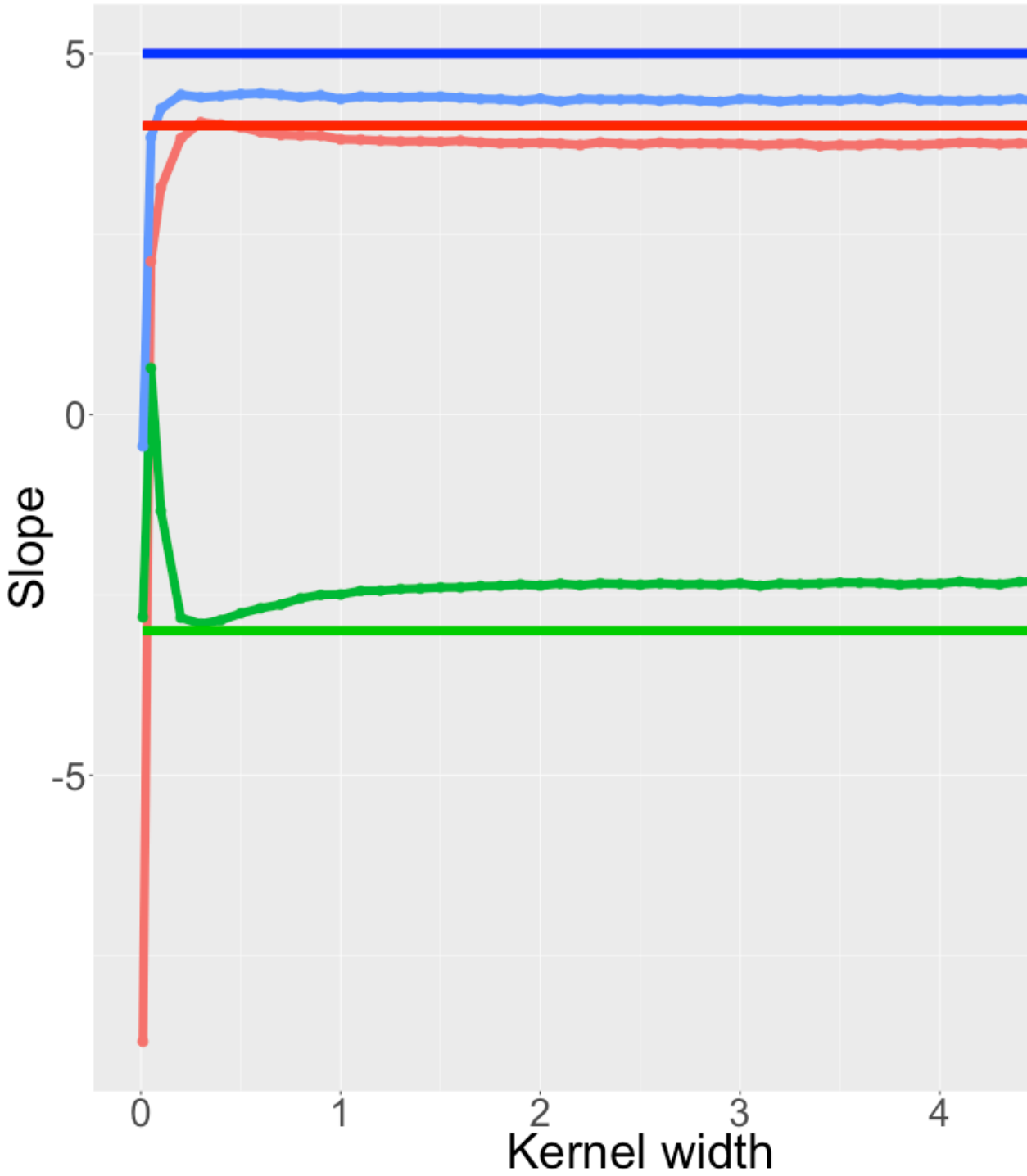


FIGURE 14.8: Simulated data: Coefficients for different kernel widhts.

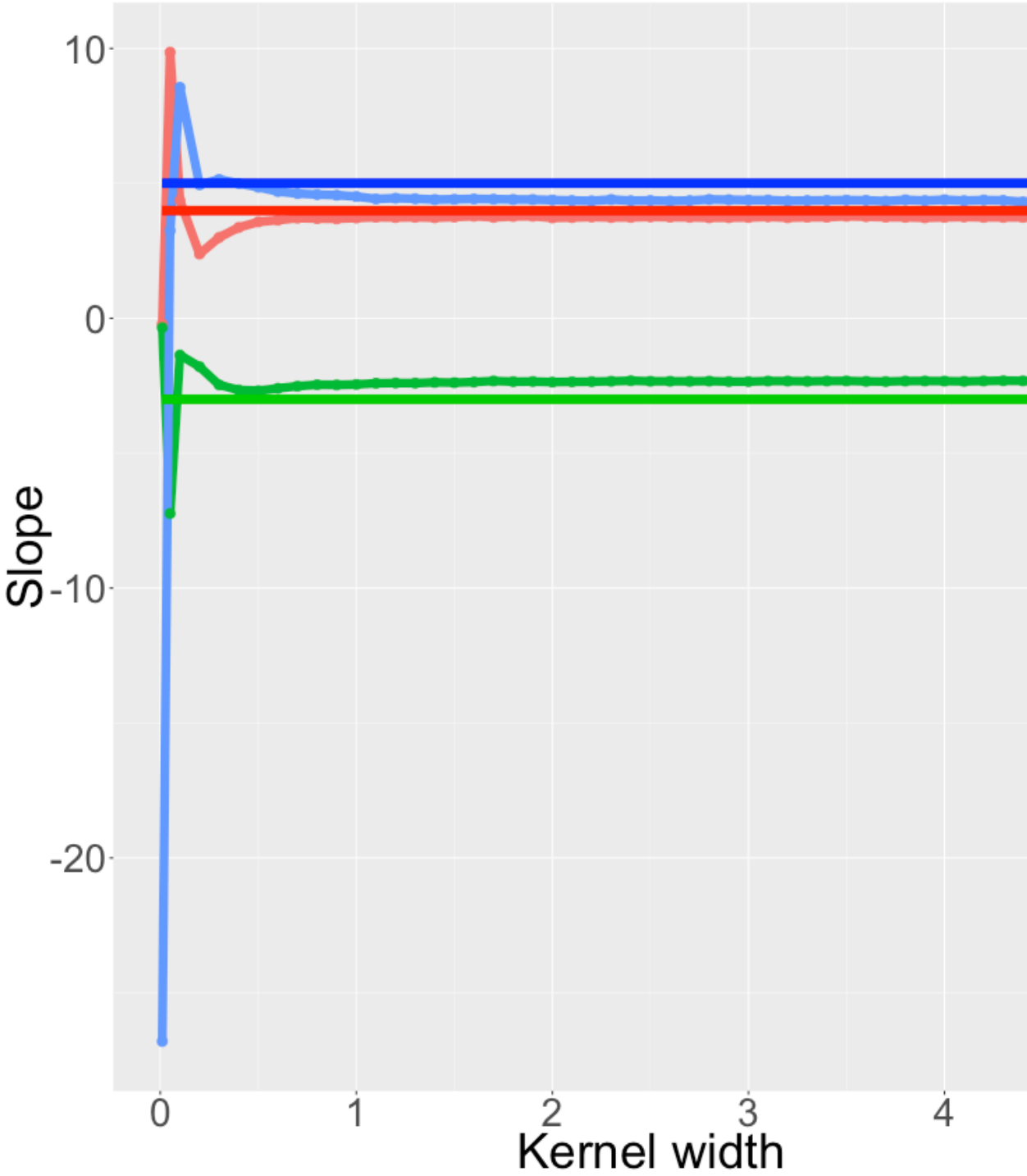


FIGURE 14.9: Simulated data: Coefficients for different kernel widhts.

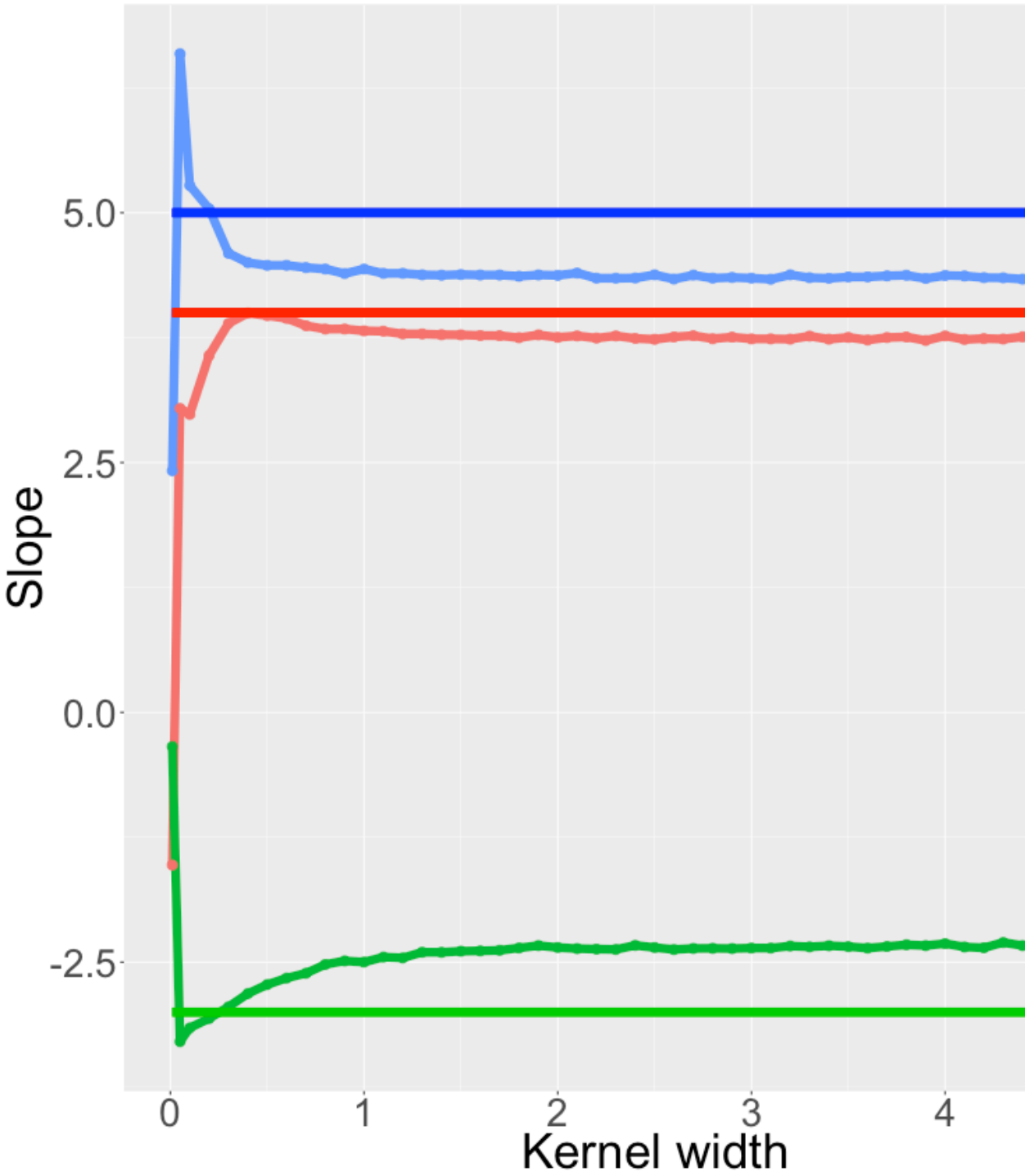


FIGURE 14.10: Simulated data: Coefficients for different kernel widhts.

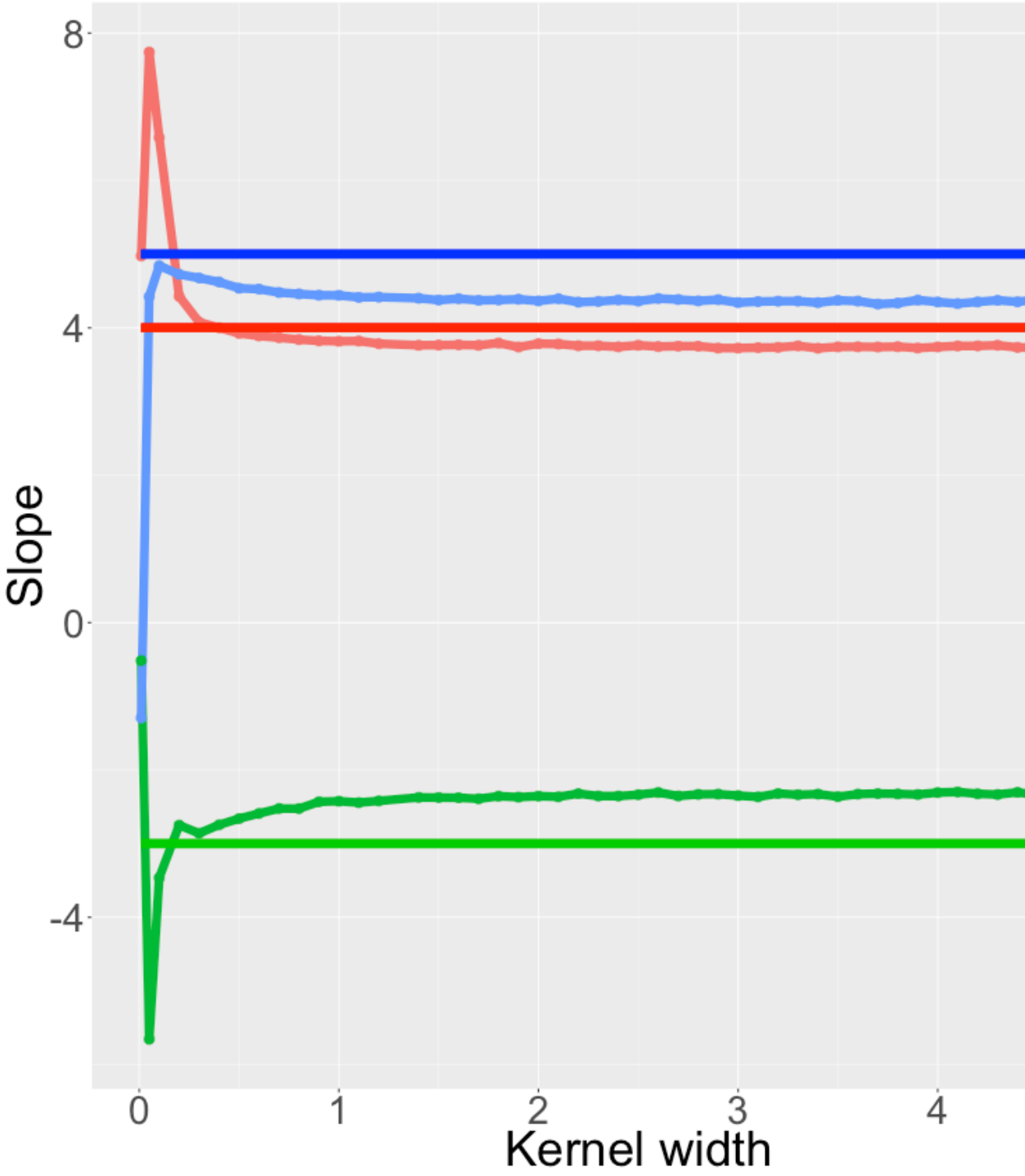


FIGURE 14.11: Simulated data: Coefficients for different kernel widhts.

cal. We further notice that LIME does not converge to the true global model but a shrunk version of it. This can be reasoned by the regularisation applied in LIME (?).

These results allow to conclude that for predicitive surfaces corresponding to linear relationships, the kernel width is a non-critical parameter. However, this case may be seen as trivial for most users of LIME.

14.3.1.2 Local Linear Relationships

For non-linear relationships we have already seen that the kernel width is more crucial. Thus, we aim to study the behaviour of the explanations w.r.t. the kernel size where the true associations are non-linear or *local*.

We may induce non-linearity by different means. However, first of all it seems interesting to study how the kernel width affects LIME explanations in a very simple setting of non-linearity: The features only affect the target locally linearly, as expressed by:

$$y = \beta_0 + \beta_1 x_1 1_{x_1 < c_1} + \beta_2 x_2 1_{x_2 > c_2} + \beta_3 x_3 + \epsilon$$

where x_1 and x_2 only affect y within the given interval. We again fit a random forest which can deal with this property. LIME should be able to detect this as well – given an appropriate kernel. To illustrate this we investigate *representative* observations, i.e. one belonging to each *bin* of the predicitve surface.

…

We outlined earlier that with an increasing dimensionality the same kernel width will correspond to a different locality. If we add new features, we should observe this. The kernel width needs to be larger for higher dimensionality in order to fit appropriate local models.

14.3.1.3 Global Non-Linearity

14.3.2 Real data

The real data set is… EXPLANATIONS WHAT IT IS First of all, we will show that the same problems as for simulated data persist for real data. When LIME is used in practise on real data, we aim that it produces consistent, or in other words stable, results. This means that if the input slighlty changes, we wish the explanation not to drastically change as well. We examine the stability of the explanations by coefficient and so-called stability paths. Stability paths are explained in detail below. We study how drastically explanations differ at

different kernel sizes if we explain the nearest neighbours of the instance to be explained are used instead.

15

LIME and Sampling

Author: Firstname Lastname

Bibliography

Apley, D. W. (2016). *Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models*.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Casalicchio, G., Molnar, C., and Bischl, B. (2018). Visualizing the feature importance for black box models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 655–670. Springer.

Fisher, A., Rudin, C., and Dominici, F. (2018). Model class reliance: Variable importance measures for any machine learning model class, from the" rashomon" perspective. *arXiv preprint arXiv:1801.01489*.

Goldstein, A., Kapelner, A., Bleich, J., and Pitkin, E. (2013). Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics*, 24.

Hall, P., Phan, W., and Ambati, S. S. (2017). Ideas on interpreting machine learning.

Hastie, T., Tibshirani, R., and Friedman, J. (2013). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer New York.

Lei, J., G'Sell, M., Rinaldo, A., Tibshirani, R. J., and Wasserman, L. (2018). Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113(523):1094–1111.

Molnar, C. (2019). *Interpretable Machine Learning*. https://christophm.github.io/interpretable-ml-book/.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM.