Predictive Models: Visualisation, Exploration and Explanation

Przemyslaw Biecek and Tomasz Burzykowski

2018-09-14

Contents

1 Introduction **5**
- 1.1 Model Lifecycle . . . 5
- 1.2 Why do we need model explainers? . . . 5
- 1.3 Black-box models vs White-box models . . . 5
- 1.4 Model agnostic vs Model specific . . . 6
- 1.5 Glossary / Notation . . . 6
- 1.6 Thanks to . . . 7

Prediction level explanations **9**

2 Introduction **11**
- 2.1 Variable atribution vs What-if analysis . . . 11
- 2.2 When to use? . . . 11
- 2.3 A bit of philosophy: Three Laws for Prediction Level Explanations . . . 13
- 2.4 Example: Promoted or Fired? . . . 13

3 Break Down **15**
- 3.1 Introduction . . . 15
- 3.2 How to read Break Down Plots? . . . 15
- 3.3 How to construct Break Down Plots . . . 17
- 3.4 Interactions . . . 20
- 3.5 Pros and cons . . . 20
- 3.6 Code snippets for R . . . 21

4 Shapley Values **25**

5 LIME: Local Interpretable Model-Agnostic Explanations **27**

6 Ceteris Paribus Principle **29**
- 6.1 Introduction . . . 29
- 6.2 1D profiles . . . 29
- 6.3 Profile oscillations . . . 32
- 6.4 2D profiles . . . 34
- 6.5 Local model fidelity . . . 34
- 6.6 Pros and cons . . . 35
- 6.7 Code snippets for R . . . 37

Model level explanations **43**

7 Introduction **45**
- 7.1 Example: Price prediction . . . 45

8 Variable Importance **47**

9 Marginal Response **49**
9.1 Partial Dependency Plots 49
9.2 Merging Path Plots 49

10 Performance Diagnostic **51**

11 Residual Diagnostic **53**

12 Other topics **55**

Chapter 1

Introduction

Machine Learning (ML) models have a wide range of applications in classification or regression problems. Due to the increasing computational power of computers and complexity of data sources, ML models are becoming more and more sophisticated. Models created with the use of techniques such as boosting or bagging of neural networks are parametrized by thousands of coefficients. They are obscure; it is hard to trace the link between input variables and model outcomes - in fact they are treated as black boxes. They are used because of their elasticity and high performance, but their deficiency in interpretability is one of their weakest sides.

In many applications we need to know, understand or prove how the input variables are used in the model. We need to know the impact of particular variables on the final model predictions. Thus we need tools that extract useful information from thousands of model parameters.

Fully trained models

1.1 Model Lifecycle

Variable importance

Model response as a function of a variable

Model performance / diagnostic / validation

1.2 Why do we need model explainers?

AutoML

Feature Extraction

Model Improvement

1.3 Black-box models vs White-box models

(Biecek, 2018c)

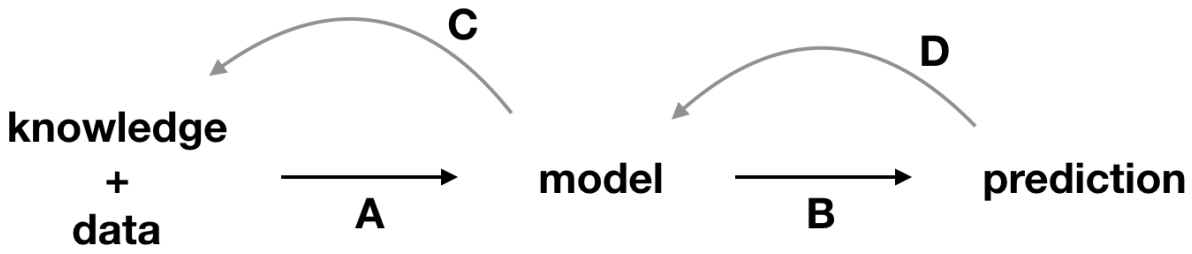


Figure 1.1: Workflow of a typical machine learning modeling. A) Modeling is a process in which domain knowledge and data are turned into models. B) Models are used to generate predictions. C) Understanding of a model structure may increase our knowledge, and in consequence it may lead to a better model. DALEX helps here. D) Understanding of drivers behind a particular model's predictions may help to correct wrong decisions, and in consequence it leads to a better model. DALEX helps here.

1.4 Model agnostic vs Model specific

1.5 Glossary / Notation

feature / variable

Let $f_M(x) : \mathcal{R}^d \to \mathcal{R}$ denote a predictive model, i.e. function that takes d dimensional vector and calculate numerical score. In section in which we work with larger number of models we use subscript M to index models. But to simplify notation, this subscript is omitted if profiles for only one model are considered.

Symbol $x \in \mathcal{R}^d$ refers to a point in the feature space. We use subscript x_i to refer to a different data points and superscript x^j to refer to specific dimensions. Additionally, let x^{-j} denote all coordinates except j-th and let $x|^j = z$ denote a data point x^* with all coordinates equal to x except coordinate j equal to value z. I.e. $\forall_{i \neq j} x^i = x^{*,i}$ and $x^j = z$. In other words $x|^j = z$ denote a x with jth coordinate changed to z.

Now we can define Ceteris Paribus Profile for model f, variable j and point x as

$$CP^{f,j,x}(z) := f(x|^j = z).$$

I.e. CP profile is a model response obtained for observations created based on x with j coordinated changes and all other coordinates kept unchanged.

It is convenient to use an alternative name for this plot: What-If Plots. CP profiles show what would happen if only a single variable is changed.

Figure 5.1 shows an example of Ceteris Paribus profile. The black dot stands for prediction for a single observation. Grey line show how the model response would change if in this single observation coordinate `surface` will be changes to selected value. From this profile one may read that the model response is non monotonic. If `construction.year` for this observation would be below 1935 the model response would be higher, but if construction year were between 1935 and 1995 the model response would be lower.

1.6 Thanks to

We are using the **bookdown** package (Xie, 2018) in this sample book

Chris Drake and Janusz Holyst

Prediction level explanations

Chapter 2

Introduction

Prediction level explainers help to understand how the model works for a single prediction. This is the main difference from the model level explainers that were focused on the model in general. Prediction level explainers are always in context of a single observation.

Think about following use-cases

- One wants to attribute effects of variables to a model predictions. Think about model for hart attack. Having a final score for a patient one wants to understand how much of this score come from smoking or age or gender.
- One wants to understand how the model response would change if some inputs are changed. Think about model for hart attack. How the model response would change if a patient cuts the number of smoked cigarettes by half.
- Model is not working correctly for a particular point and one wants to understand why predictions for this point are wrong.

2.1 Variable atribution vs What-if analysis

There are many different tools that may be used to explore model around a single data point and in following sections we will describe the most popular approaches. They can be divided into two classes.

- Analysis of the model curvature. Here we treat the model as a function and we are interested in the curvature of this function around the point of interest (see Figure 2.1). In section 5 we present the LIME method that approximates the black-box model in a point of interest while in section 6 we present Ceteris Paribus profiles that are more focused on conditional changes of model response given only one coordinate is modified.
- Analysis of the probabilistic behavior of the model. Here we are interested in decomposition of the model response to parts that can be attributed to particular features.

2.2 When to use?

There are several use-cases for such explainers. Think about following.

- Model improvement. If model works particular bad for a selected observation (the residual is very high) then investigation of model responses for miss fitted points may give some hints how to improve the model. For individual predictions it is easier to notice that selected variable should have different a effect.

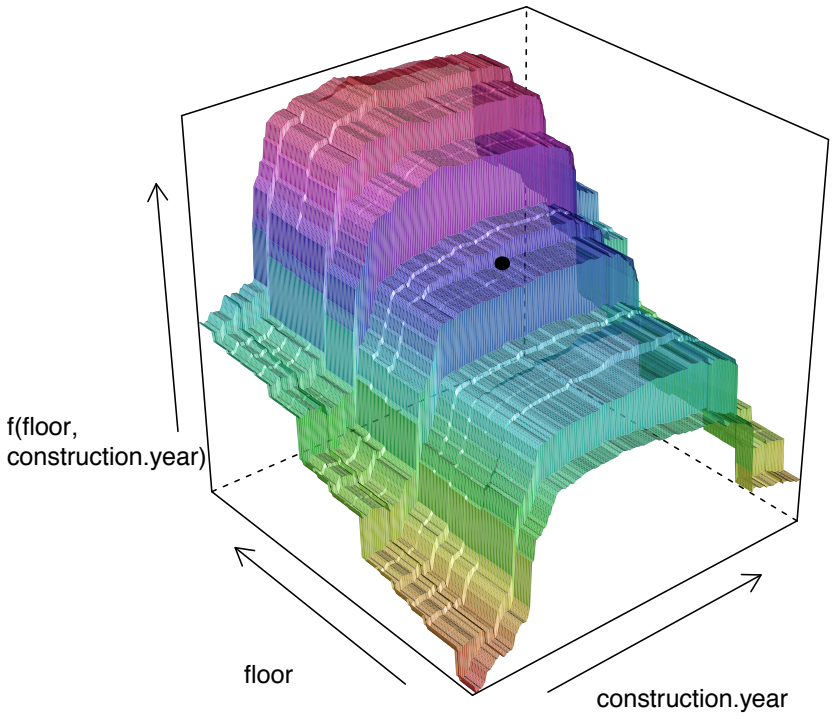


Figure 2.1: (fig:modelResponseCurve) Model response surface. We are interested in understanding the model behavior in a single point

- Additional domain specific validation. Understanding which factors are important for model predictions helps to be critical about model response. If model contributions are against domain knowledge then we may be more skeptical and willing to try another model. On the other hand, if the model response is aligned with domain knowledge we may trust more in these responses. Such trust is important in decisions that may lead to serious consequences like predictive models in medicine.
- Model selection. Having multiple candidate models one may select the final response based on model explanations. Even if one model is better in terms of global model performance it may happen that locally other model is better fitted. This moves us towards model consultations that identify different options and allow human to select one of them.

2.3 A bit of philosophy: Three Laws for Prediction Level Explanations

76 years ago Isaac Asimov devised Three Laws of Robotics: 1) a robot may not injure a human being, 2) a robot must obey the orders given it by human beings and 3) A robot must protect its own existence. These laws impact discussion around Ethics of AI. Today's robots, like cleaning robots, robotic pets or autonomous cars are far from being conscious enough to be under Asimov's ethics.

Today we are surrounded by complex predictive algorithms used for decision making. Machine learning models are used in health care, politics, education, judiciary and many other areas. Black box predictive models have far larger influence on our lives than physical robots. Yet, applications of such models are left unregulated despite many examples of their potential harmfulness. See *Weapons of Math Destruction* by Cathy O'Neil for an excellent overview of potential problems.

It's clear that we need to control algorithms that may affect us. Such control is in our civic rights. Here we propose three requirements that any predictive model should fulfill.

- **Prediction's justifications**. For every prediction of a model one should be able to understand which variables affect the prediction and how strongly. Variable attribution to final prediction.
- **Prediction's speculations**. For every prediction of a model one should be able to understand how the model prediction would change if input variables were changed. Hypothesizing about what-if scenarios.
- **Prediction's validations** For every prediction of a model one should be able to verify how strong are evidences that confirm this particular prediction.

There are two ways to comply with these requirements. One is to use only models that fulfill these conditions by design. White-box models like linear regression or decision trees. In many cases the price for transparency is lower performance. The other way is to use approximated explainers – techniques that find only approximated answers, but work for any black box model. Here we present such techniques.

2.4 Example: Promoted or Fired?

In this chapter we will use artificial dataset from Human Resources department in a call center to present pros and cons for different techniques of prediction level explainers. At the end of each section there is a collection of examples that shows how to use described techniques in R and Python.

The dataset is available in the `DALEX` package (Biecek, 2018c). Each row corresponds to a single employee of a call center. Features like gender, age, average number of working hours per week, grade from the last evaluation and level of salary are used as predictive features.

The problem here is to first build a model, that will determine when to fires and when to promote an employer, so it's a classification problem with three classes. But having a model we will use prediction level explainers to better understand how the model works for selected cases.

```
library("DALEX")
head(HR)
```

```
##   gender      age    hours evaluation salary   status
## 1   male 32.58267 41.88626          3      1    fired
## 2 female 41.21104 36.34339          2      5    fired
## 3   male 37.70516 36.81718          3      0    fired
## 4 female 30.06051 38.96032          3      2    fired
## 5   male 21.10283 62.15464          5      3 promoted
## 6   male 40.11812 69.53973          2      0    fired
```

In this book we are focused on model exploration rather than model building, thus for sake ok simplicity we will use two default models created with random forest (Breiman et al., 2018) and generalized linear model (**?**).

```
set.seed(59)
library("randomForest")
model_rf <- randomForest(status ~ gender + age + hours + evaluation + salary, data = HR)

library("nnet")
model_glm <- multinom(status ~ gender + age + hours + evaluation + salary, data = HR)
```

```
## # weights:  21 (12 variable)
## initial  value 8620.810629
## iter  10 value 7002.127738
## iter  20 value 6239.478146
## iter  20 value 6239.478126
## iter  20 value 6239.478124
## final  value 6239.478124
## converged
```

Chapter 3

Break Down

In this section we introduce tools based on variable relaxation principle. The main goal for these tools is to help understand how model output may be attributed to input variables or sets of variables. Attribution is in most cases additive, thus the model response is decomposed into parts that may be assigned to variables.

Presented explainers are linked with the first law introduced in Section 2.3, i.e. law for prediction's justifications. Note that there are more tools for variable attribution, some of them will be presented in next sections.

Think of following use cases:

- Think about a model for hart attack. A patient wants to know which factors have highest impact on the final heart risk score.
- Think about a model for apartment prices. An investor wants to know how much of the final price may be attributed to the location of an apartment.
- Think about a model for credit scoring. A customer wants to know if factors like gender, age or number of kids influence model decisions.

3.1 Introduction

The name *Break Down Plots* comes from the way how Break Down plots are working. The main goal is to decompose model predictions into parts that can be additively attributed to particular variables. See an example in Figure 3.1.

It is straightforward for linear (and more general: additive) models. But not that obvious for more complex models. In this section we present uniform, model agnostic approach to variable attribution.

Similar to the Shapley method introduced in the Section 4, Break Down Plots show additive decomposition of model output. As we will show later, the Shapely method may be perceived as an average from all possible Break Down Paths. In the last subsection we discuss pros and cons of this approach.

3.2 How to read Break Down Plots?

The intuition behind *Break Down* Plots is to show how model prediction would change if observations in general population will be made more similar to the observation of interest.

This idea is described in Figure **??** and let's trace it step by step. For a black-box model (here random forest model, but it's structure is not relevant) we want to decompose prediction for following point:

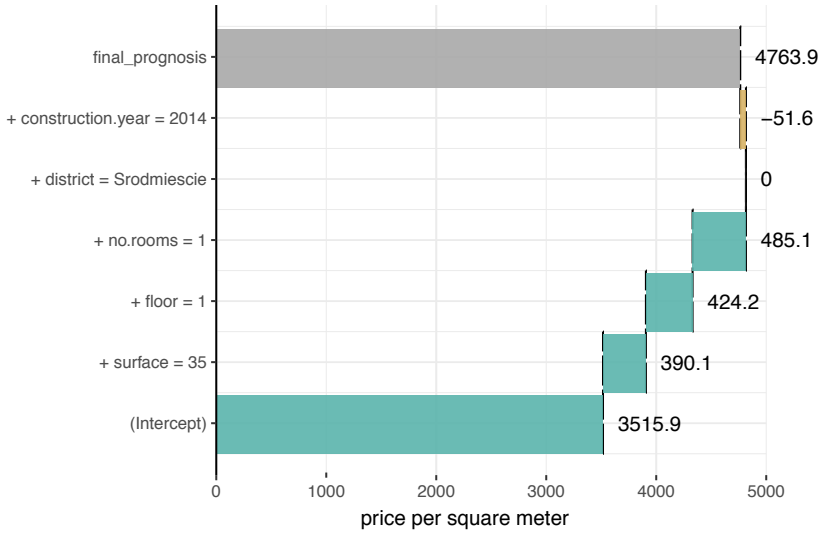


Figure 3.1: (fig:BDPrice1) An illustration of Break Down Plots. Model prediction for Random Forest models for a single observation (grey bar, 4763.9) is decomposed into parts that can be attributed to population average (the bottom bar, 3515.9) and effects of particular variables.

- surface: 35
- floor: 1
- no.rooms: 1
- district: Srodmiescie
- construction year: 2014

The model prediction is 4763.9. How to attribute following features to it?

Baseline for this decomposition is the distribution of model predictions for whole population. In the last row of panel A denoted by `all data` the violin plot shows distribution of model prediction for whole population (here calculated on the validation dataset). Red dot stand for the average, and the average model prediction, averaged over all points is 3515.9.

Then we check how model prediction would change, if every observation in the population would have variable `surface` equal to `35`. New distribution for these modified observation is presented in the second row `+ surface = 35`. Black intervals between these two rows show how individual prediction change from original model prediction to the prediction calculated with coordinate `surface` set to `35`. Again, the red dot stands for the average from model predictions for such modified population. This time the average is 3906.

In the next step we modify all observations in the population again, this time we set the `floor` variable to `1`. Again, model prediction will change and black intervals show how individual prediction change. Also the average from all these modified prediction will change. This time the new average is 4330.2.

Such steps are repeated for consecutive variables till all variables are set to the point of interest. In this last step all observations in the population will have coordinates changed to the observation of interests and all model predictions are equal 4763.9.

Tracing distributions is not easy, thus instead of distributions we are focused on averages. Panel B) shows how the average of distribution changes after each step. The average for whole population was 3515.9. After setting `surface` to `35` the average changed by `+390.1`. In the next step, when all observations were changed to `floor = 1` the average changed by `+424.1`.

Break Down Plots, presented in the panel C) shows only these differences. Positive values are presented with green bars while negative differences are marked with yellow bar. They sum up to final model prediction, which is denoted by a grey bar in this example.

NOTE

If the considered model is additive, then the order in which variables are added does not matter. If the model is not additive, then different orders will lead to different effects.

Simplest way to choose an order is to use a greedy procedure in which we first include variables with largest influence to the average.

Find more details about to this procedure in the next section.

3.3 How to construct Break Down Plots

Let us use the following notation: $x = (x_1, x_2, ..., x_d) \in \mathcal{R}^d$ is a vector with features. $f_M(x) : R^d \to R$ is a model under consideration. It may be a regression or classification model. Additionally X is a dataset with n observations, this dataset will be used to learn joint distribution of features.

For a single observation x the model prediction is equal to $f(x)$. Our goal is to attribute parts of this score to variables (dimensions) in the R^d space.

Model agnostic approach

The intuition behind this approach is to identify components of x that cannot be changed without a significant change in the prediction $f(x)$. In order to present this approach in a more formal way, we first need to introduce some definitions.

Relaxed model prediction

Let $f^{IndSet}(x^*)$ denotes an expected model prediction for x^{*new*} relaxed on the set of indexes $IndSet \subset \{1, \ldots, p\}$.

$$f^{IndSet}(x^*) = E[f(x)|x_{IndSet} = x^*_{IndSet}].$$

Thus $f^{IndSet}(x^*)$ is an expected value for model response conditioned on variables from set $IndSet$ in such a way, that $\forall_{i \in IndSet} x_i = x^*_i$.

The intuition behind relaxed prediction is that we are interested in an average model response for observations that are equal to x^* for features from $IndSet^C$ set and follow the population distribution for features from $IndSet$ set. Clearly, two extreme cases are

$$f^{\{1,\ldots,p\}}(x^*) = f(x^*),$$

which is the case of no relaxation, and

$$f^{\emptyset}(x^{new}) = E[f(x)].$$

which corresponds to full relaxation.

We will say that a variable was relaxed, when we do not fix its value and we let it follow the population distribution.

This will play a crucial part in the algorithm presented in this section.

Since we do not know the joint distribution of x, we will use its estimate instead.

$$\widehat{f^{IndSet}(x^*)} = \frac{1}{n} \sum_{i=1}^{n} f(x^i_{-IndSet}, x^*_{IndSet}).$$

We will omit the dashes to simplify the notation.

Distance to relaxed model prediction

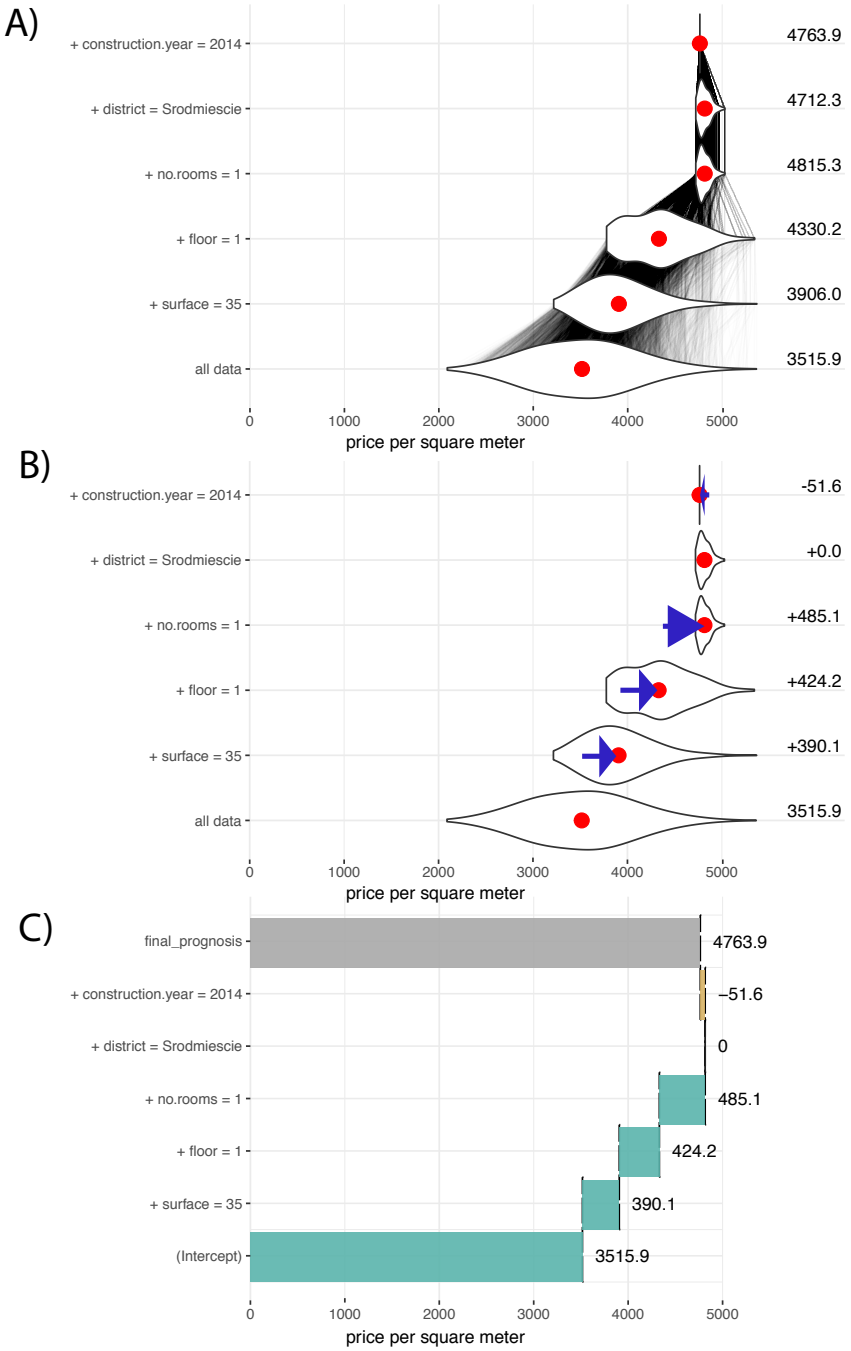


Figure 3.2: (fig:BDPrice4) Break Down Plots show how variables move the model prediction from population average to the model prognosis for a single observation. A) TODO B) TODO C) TODO An illustration of algorithm . Each row in this plot corresponds to a distribution of model scores $f(x)|x_IndSet = x^*_IndSet$ for different sets of $IndSet$ indexes. Initially $IndSet = 1, ..., p$ and in each step single variable is removed from this set. Labels on the left-hand side of the plots show which variable is removed in a given step. Red dots stand for conditional average - an estimate of relaxed predictions $f^{IndSet}(x^*)$. Violin plots summarize conditional distributions of scores while gray lines show how model predictions change for particular observations between consecutive relaxations.

Let us define the distance between model prediction and relaxed model prediction for a set of indexes $IndSet$.

$$d(x^*, IndSet) := |f^{IndSet}(x^*) - f(x^*)|.$$

It is the difference between model prediction for observation x^* and observation relaxed on features $indSet$. The smaller the difference, the less important are variables in the $indSet$ set.

Added feature contribution

For j-th feature we define its contribution relative to a set of indexes $IndSet$ (*added contribution*) as

$$\text{contribution}^{IndSet}(j) = f^{IndSet \cup \{j\}}(x^*) - f^{IndSet}(x^*).$$

It is the change in model prediction for x^* after relaxation on j.

The model agnostic feature contribution is based on distances to relaxed model predictions. In this approach we look for a series of variables that can be relaxed in such a way so as to move model prediction from $f(x^*)$ to a fully relaxed prediction $E[f(x)]$ (expected value for all model predictions). The order of features in this series is important. But here we use a greedy strategy in which we add features to the $indSet$ iteratively (one feature per iteration) and minimize locally the distance to relaxed model prediction.

The greedy search can start from a null set of indexes (then in each step a single feature is being relaxed) or it can start from a full set of relaxed features (then in each step a single feature is removed from the set). The above approaches are called *step-up* and *step-down*, respectively.

The algorithm **??** presents the procedure that generates a sequence of variables with increasing contributions. This sequence corresponds to variables that can be relaxed in such a way so as to minimize the distance to the original prediction. The resulting sequence of $Contributions$ and $Variables$ may be plotted with Break Down Plots. By relaxing consecutive variables one finds a path between single prediction and average model prediction.

Model agnostic break down of model predictions. The *step-down* approach.

1. \$p := \$ number of variables
2. $IndSet := \{1, \ldots, p\}$, i.e. set of indexes of all variables
3. FOR $i \in \{1, \ldots, p\}$
4. Find new variable that can be relaxed with small loss in relaxed distance to $f(x^*)$
5. FOR $jinIndSet$
6. Calculate relaxed distance with j removed
7. $dist(j) := d(x^*, IndSet \setminus \{j\})$
8. END FOR 5.
9. Find and remove j that minimizes loss
10. $j_{min} := \arg\min_j dist(j)$
11. $Contribution^{IndSet}(i) := f^{IndSet}(x^{new}) - f^{IndSet \setminus \{j_{min}\}}(x^*)$
12. $Variables(i) := j_{min}$
13. $IndSet := IndSet \setminus \{j_{min}\}$
14. END FOR 3.

Approach for linear models

For linear models (and also generalized linear models) the scoring function (e.g. link function) may be expressed as linear combination of feature vectors.

$$f(x^*) = (1, x^*)(\mu, \beta)^T = \mu + x_1^*\beta_1 + \ldots + x_p^*\beta_p.$$

In this case it is easy to attribute the impact of feature x_i to prediction $f(x^*)$. The most straightforward approach would be to use the $x_i^*\beta_i$ as the attribution. However, it is easier to interpret variable attributions

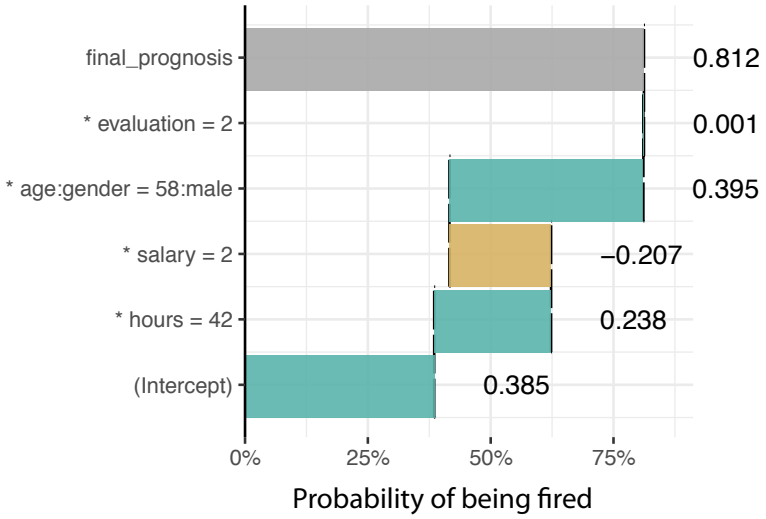


Figure 3.3: (fig:bdInter1) Break Down plot with interactions

if they are invariant to scale-location transformations of x_i, such as change of the unit or origin. This is why for linear models the variable attributions are defined as $(x_i^* - \bar{x}_i)\beta_i$.

Previous equation may be rewritten as follows:

$$f(x^*) = (1, x^*)(\mu, \beta)^T = baseline + (x_1^* - \bar{x}_1)\beta_1 + ... + (x_p^* - \bar{x}_p)\beta_p$$

where

$$baseline = \mu + \bar{x}_1\beta_1 + ... + \bar{x}_p\beta_p.$$

Components $(x_i^* - \bar{x}_i)\beta_i$ are all expressed in the same units.

3.4 Interactions

3.5 Pros and cons

Break Down Plots gives a uniform approach to decompose model prediction into parts that can be attributed additively to variables. Below we summarize key strengths and weaknesses of this approach.

Pros

- Graphical representation of Break Down plots is easy to understand.
- Break Down plots are compact, many variables may be presented in a small space.
- Break Down plots are model agnostic yet they reduce to intuitive interpretation for linear Gaussian and generalized models.
- Break Down plots are faster than other approaches to variable attribution.

Cons

- If the model is non-additive then showing only additive contributions may be misleading.
- For large number of variables the Break Down plot may be messy with many variables having small contributions.

- For non-additive models the Break Down plot will depend on the order of variables being relaxed.

3.6 Code snippets for R

In this section we present key features of the `breakDown` package for R (Biecek, 2018a). This package covers all features presented in this chapter. It is available on CRAN and GitHub. Find more examples at the website of this package `https://pbiecek.github.io/breakDown/`.

Model preparation

In this section we will present examples based on the `apartments` dataset. See section TODO for more details.

```
library("DALEX")
head(HR)
```

```
##   gender      age    hours evaluation salary   status
## 1   male 32.58267 41.88626          3      1    fired
## 2 female 41.21104 36.34339          2      5    fired
## 3   male 37.70516 36.81718          3      0    fired
## 4 female 30.06051 38.96032          3      2    fired
## 5   male 21.10283 62.15464          5      3 promoted
## 6   male 40.11812 69.53973          2      0    fired
```

The problem here is to predict average price for square meter for an apartment. Let's build a random forest model with `randomForest` package (Breiman et al., 2018).

```
library("randomForest")
model <- randomForest(status ~ gender + age + hours + evaluation + salary, data = HR)
model
```

```
##
## Call:
##  randomForest(formula = status ~ gender + age + hours + evaluation +     salary, data = HR)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 27.27%
## Confusion matrix:
##          fired   ok promoted class.error
## fired     2276  382      197   0.2028021
## ok         532 1253      436   0.4358397
## promoted   205  388     2178   0.2140022
```

Model exploration with `breakDown` package is performed in three steps.

1. Create an explainer - wrapper around model and validation data.

Since all other functions work in a model agnostic fashion, first we need to define a wrapper around the model. Here we are using the `explain()` function from `DALEX` package (Biecek, 2018c).

```
explainer_rf_fired <- explain(model,
                 data = HR,
                 y = HR$status == "fired",
                 predict_function = function(m,x) predict(m,x, type = "prob")[,1],
                 label = "fired")
```

2. Define point of interest.

Break Down Plots decompose model prediction around a single point.

```
new_observation <- data.frame(gender = factor("male", levels = c("male", "female")),
                      age = 57.7,
                      hours = 42.3,
                      evaluation = 2,
                      salary = 2)

predict(model, new_observation, type = "prob")
```

```
##   fired    ok promoted
## 1 0.778 0.218    0.004
## attr(,"class")
## [1] "matrix" "votes"
```

3. Calculate BD decomposition

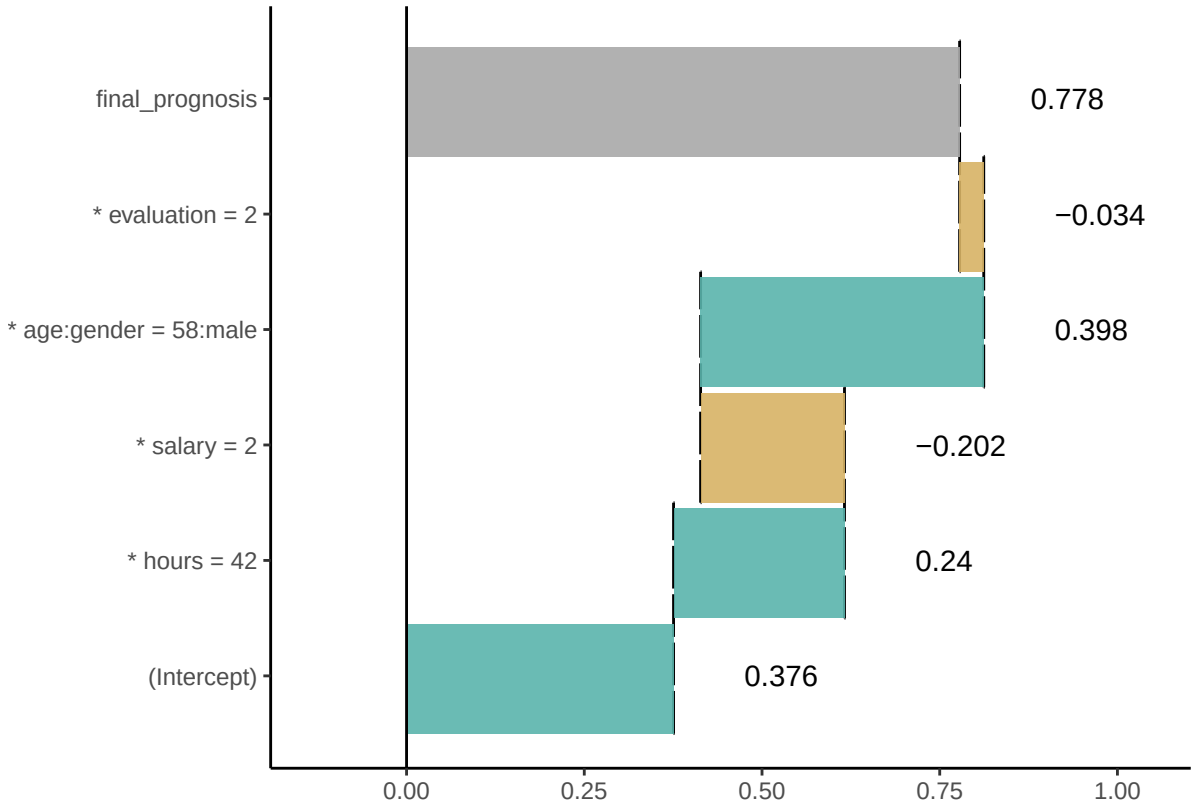
The `break_down()` function calculates BP decomposition for selected model around selected observation.

The result from `break_down()` function is a data frame with variable attributions.

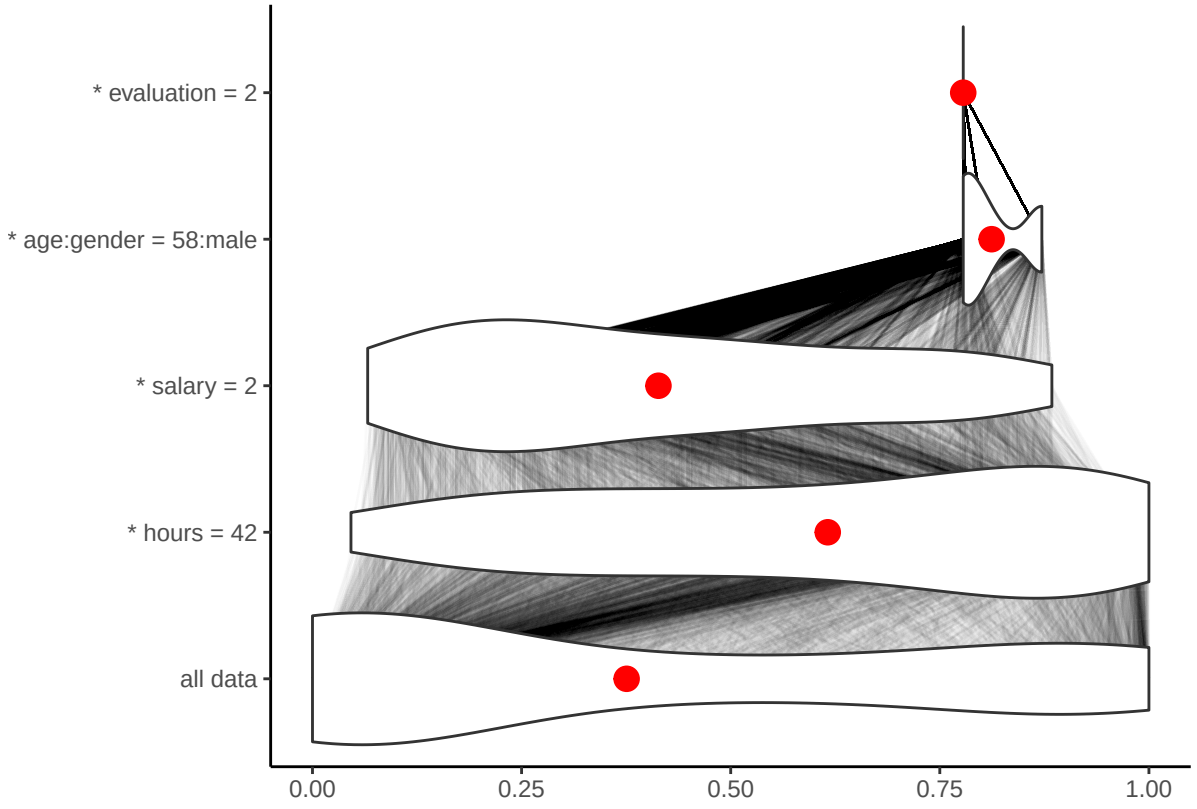
```
library("breakDown")
bd_rf <- break_down(explainer_rf_fired,
                new_observation,
                keep_distributions = TRUE)

bd_rf
```

```
##                          contribution
## (Intercept)                     0.376
## * hours = 42                    0.240
## * salary = 2                   -0.202
## * age:gender = 58:male          0.398
## * evaluation = 2               -0.034
## final_prognosis                 0.778
## baseline:  0
```

```
plot(bd_rf)
```




```
plot(bd_rf, plot_distributions = TRUE)
```



Chapter 4

Shapley Values

Variable Atribution

This approach can be seen as an approximation of Shapley values where feature contribution is linked with the average effect of a feature across all possible relaxations. These approaches are identical for additive models. For non-additive models the additive attribution is just an approximation in both cases, yet the greedy strategy produces explanations that are easier to interpret. It is worth noting that similar decomposition of predictions and measures of contribution for classifiers have been examined in **?**.

Chapter 5

LIME: Local Interpretable Model-Agnostic Explanations

(Pedersen and Benesty, 2018) (Staniak and Biecek, 2018)

```
library(lime)
library(live)
```

Sparse model approximation / variable selection / feature ranking

live: Local Interpretable (Model-Agnostic) Visual Explanations

Chapter 6

Ceteris Paribus Principle

In this section we introduce tools based on Ceteris Paribus principle. The main goal for these tools is to help understand how changes in model input affect changes in model output.

Presented explainers are linked with the second law introduced in Section 2.3, i.e. law for prediction's speculations. This is why these explainers are also known as *What-If model analysis* or *Individual Conditional EXpectations* (Goldstein et al., 2015). It turns out that it is easier to understand how blacx-box model is working if we can play with it by changing variable by variable.

Think of following usecases:

- Think about a model for hart attack. How the model response would change if a patient cuts the number of smoked cigarettes by half or increase physical activity.
- Think about a model for credit scoring. A customer gets a low score and is asking what he needs to change to increase this score to a certain level, to pass the bank criteria.
- Think about a model for apartment prices. An investor wants to know how much the price may increase if apartment standard is upgraded.

6.1 Introduction

Ceteris paribus is a Latin phrase meaning "other things held constant" or "all else unchanged". Using this principle we examine input variable per variable separatly, asumming that effects of all other variables are unchanged. See Figure 6.1

Similar to the LIME method introduced in the section 5, Ceteris Paribus profiles examine curvature of a model response function. The difference between these two methods that LIME approximates the model curvature with a simpler white-box model that is easier to present. Usually the LIME model is sparse, thus our attention may be limited to smaller number of dimensions. In contrary, the CP plots show conditional model response for every variable. In the last subsection we discuss pros and cons of this approach.

6.2 1D profiles

Let $f_M(x) : \mathcal{R}^d \to \mathcal{R}$ denote a predictive model, i.e. function that takes d dimensional vector and calculate numerical score. Symbol $x \in \mathcal{R}^d$ refers to a point in the feature space. We use subscript x_i to refer to a different data points and superscript x^j to refer to specific dimensions. Additionally, let x^{-j} denote all coordinates except j-th and let $x|^j = z$ denote a data point x^* with all coordinates equal to x except coordinate j equal to value z. I.e. $\forall_{i \neq j} x^i = x^{*,i}$ and $x^j = z$. In other words $x|^j = z$ denote a x with jth coordinate changed to z.

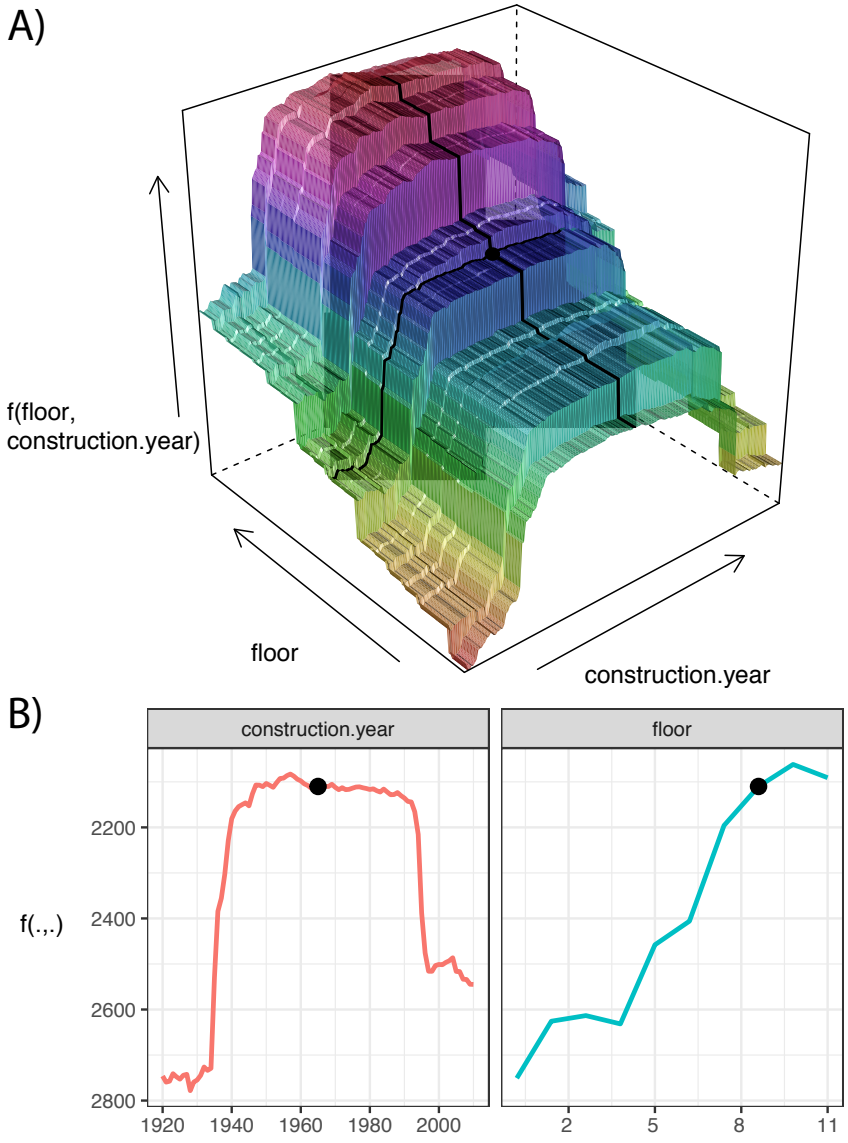


Figure 6.1: (fig:modelResponseCurveLine) A) Model response surface. Ceteris Paribus profiles marked with black curves helps to understand the curvature of the model response by updating only a single variable. B) CP profiles are individual conditional model responses

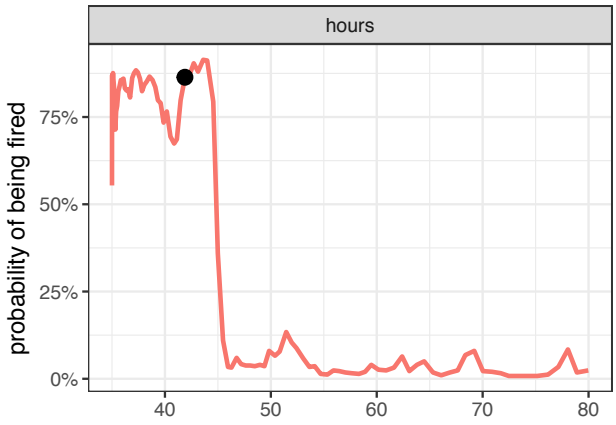


Figure 6.2: (fig:HRCPHiredHours) Ceteris Paribus profile for Random Forest model that assess the probability of being fired in call center as a function of average number of working hours

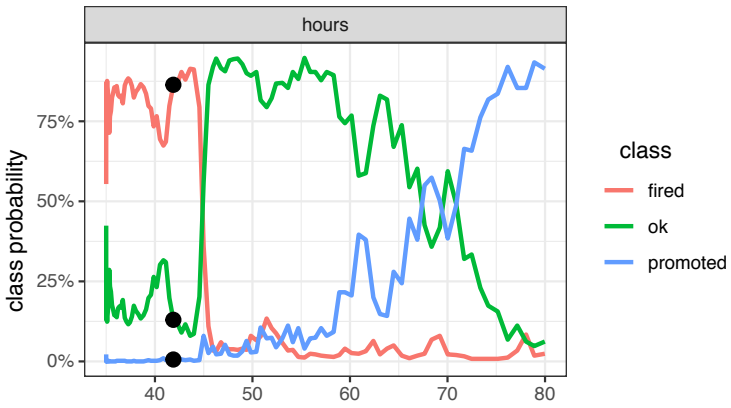


Figure 6.3: (fig:HRCPAllHours) Ceteris Paribus profiles for three classess predicted by the Random Forest model as a function of average number of working hours

Now we can define uni-dimensional Ceteris Paribus Profile for model f, variable j and point x as

$$CP^{f,j,x}(z) := f(x|^j = z).$$

I.e. CP profile is a model response obtained for observations created based on x with coordinate j changed and all other coordinates kept unchanged.

A natural way to visualise CP profiles is to use a profile plot as in Figure 6.2.

Figure 6.2 shows an example of Ceteris Paribus profile. The black dot stands for prediction for a single observation. Grey line show how the model response would change if in this single observation coordinate `hours` will be changed to selected value. One thing that we can read is that the model response is not smooth and there is some variability along the profile. Second thing is that for this particular observation the model response would drop significantly if the variable `hours` will be higher than 45.

Since in the example dataset we are struggling with model for three classes, one can plot CP profiles for each class in the same panel. See an example in the Figure 6.3.

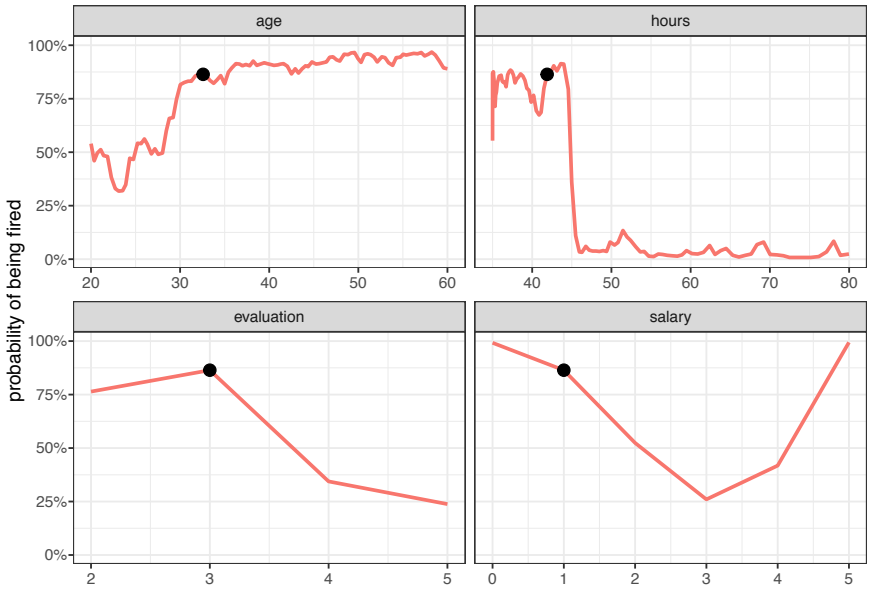


Figure 6.4: (fig:HRCPFiredAll) Ceteris Paribus profiles for all continuous variables

Usually model input consist many variables, then it is beneficial to show more variables at the same time. The easiest way to do so is to plot consecutive variables on separate panels. See an example in Figure 6.4.

6.3 Profile oscillations

Visual examination of variables is insightful, but for large number of variables we end up with large number of panels, most of which are flat. This is why we want to asses variable importance and show only profiles for important variables. The advantage of CP profiles is that they lead to a very natural and intuitive way of assessing the variable importance for a single prediction. The intuition is: the more important variable the larger are changes along the CP profile. If variable is not important then model response will barely change. If variable is important the CP profile change a lot for different values of a variable.

Let's write it down in a more formal way.

Let $vip_j^{CP}(x)$ denotes variable importance calculated based on CP profiles in point x for variable j.

$$vip_j^{CP}(x) = \int_{-\inf}^{inf} |CP^{f,j,x}(z) - f(x)|dz$$

So it's an absolute deviation from $f(x)$. Note that one can consider different modification of this coefficient:

1. Deviations can be calculated not as a distance from $f(x)$ but from average $\bar{CP}^{f,j,x}(z)$.
2. The integral may be weighted based on the density of variable x^j.
3. Instead of absolute deviations one may use root from average squares.

TODO: we need to verify which approach is better. Anna Kozak is working on this

The straightforward estimator for $vip_j^{CP}(x)$ is

$$\widehat{vip_j^{CP}}(x) = \frac{1}{n}\sum_{i=1}^{n} |CP^{f,j,x}(x_i) - f(x)|.$$

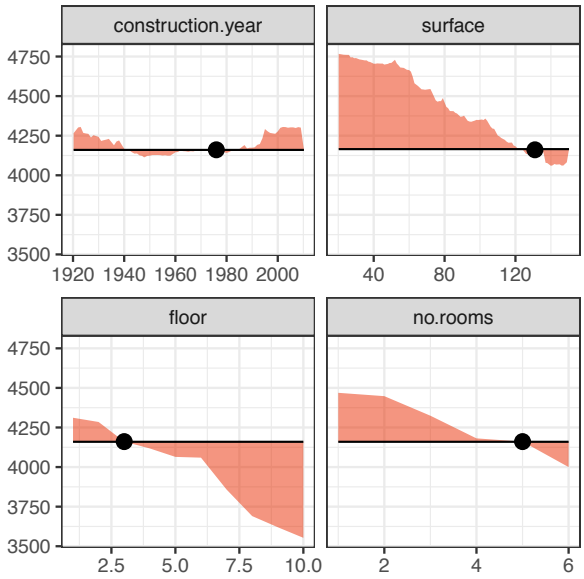


Figure 6.5: (fig:CPVIPprofiles) CP oscillations are average deviations between CP profiles and the model response

Figure 6.5 shows the idea behind measuring oscillations. The larger the highlighted area the more important is the variable.

Figure 6.6 summarizes variable oscillations. Such visuals help to quickly grasp how large are model oscillations around a specific point.

NOTE

Variable importance for single prediction may be very different than variable importance for the full model.

For example, consider a model

$$f(x_1, x_2) = x_1 * x_2$$

where variables x_1 and x_2 takes values in $[0, 1]$.

From the global perspective both variables are equally important.

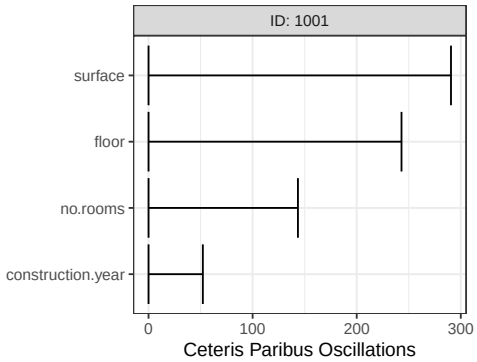


Figure 6.6: (fig:CPVIP1) Variable importance plots calculated for Ceteris Paribus profiles for observation ID: 1001

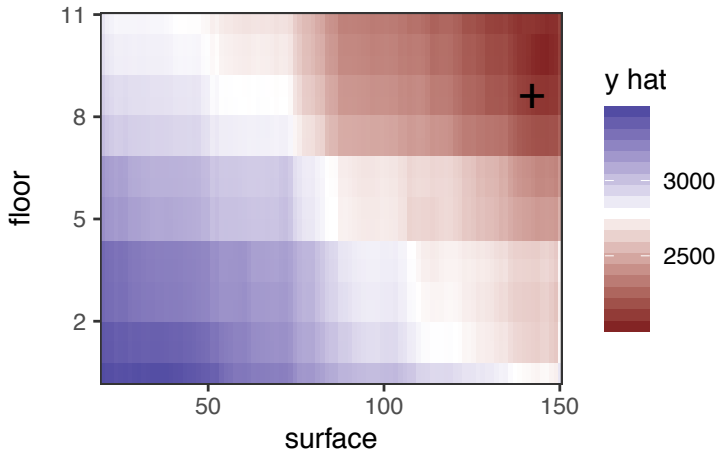


Figure 6.7: (fig:CP2Dsurflor) Ceteris Paribus plot for a pair of variales. Black cross marks coordinated for the observation of interest. Presented model estimates price of an appartment

But local variable importance is very different. Around point $x = (0, 1)$ the importance of x_1 is much larger than x_2. This is because profile for $f(z, 1)$ have larger oscillations than $f(0, z)$.

6.4 2D profiles

The definition of ceteris paribus profiles given in section 6.2 may be easily extended to two and more variables. Also definition of CP oscillations 6.3 have straight forward generalization for larger number of dimensions. Such generalisations are usefull when model is non additive. Presence of pairwise interactions may be detected with 2D Ceteris Paribus plots.

Let's define two-dimensional Ceteris Paribus Profile for model f, variables j and k and point x as

$$CP^{f,(j,k),x}(z_1, z_2) := f(x|^{(j,k)} = (z_1, z_2)).$$

I.e. CP profile is a model response obtained for observations created based on x with j and k coordinates changed to (z_1, z_2) and all other coordinates kept unchanged.

A natural way to visualise 2D CP profiles is to use a level plot as in Figure 6.7.

If number of variables is small or moderate thein it is possible to present all pairs of variables. See an example in Figure 6.8.

6.5 Local model fidelity

Ceteris Paribus profiles are also a useful tool to validate local model fidelity. It may happen that global performance of the model is good, while for some points the local fit is very bad. Local fidelity helps to understand how good is the model fit around point of interest.

How does it work?

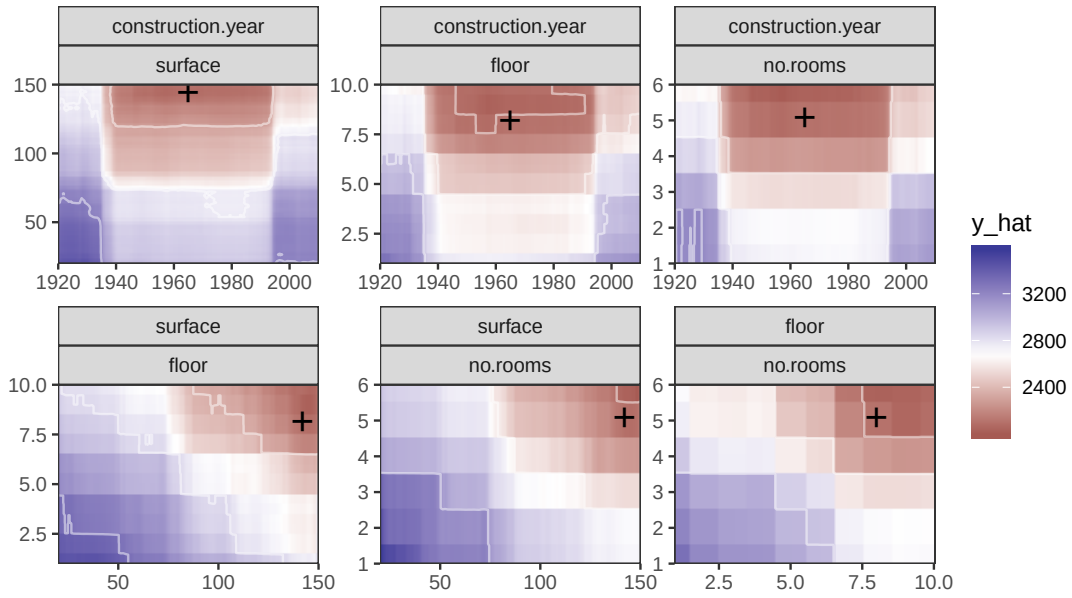


Figure 6.8: (fig:CP2Dall) Ceteris Paribus plot for all pairs of variales.

The idea behind fidelity plots is to select some number of points from the validation dataset that are closes to the point of interest. It's a similar approach as in k nearest neighbours. Then for these neighbours we may plot Ceteris Paribus Profiles and check how stable they are.

Also, if we know true taget values for points from the validation dataset we may plot residuals to show how large are residuals.

An example fidelity plot is presented in Figure 6.9. Black line shows the CP profiles for the point of interest, while grey lines show CP profiles for neihgbors. Red intervals stand for residuals and in this example it looks like residuals for neighbours are all negative. Thus maybe model is biased around the point of interest.

This observation may be confirmed by plots that compare distribution of all residuals against distribution of residuals for neighbors.

See Figure **??** for an example. Here residuals for neighbors are shifted towards highest values. This suggests that the model response is biased around the observation of interest.

6.6 Pros and cons

Ceteris Paribus principle gives a uniform and extendable approach to model exploration. Below we summarize key strengths and weaknesses of this approach.

Pros

- Graphical representation of Ceteris Paribus profile is easy to understand.
- Ceteris Paribus profiles are compact and it is easy to fit many models or many variables in a small space.
- Ceteris Paribus profiles helps to understand how model response would change and how stable it is
- Oscillations calculated for CP profiles helps to select the most important variables.
- 2D Ceteris Paribus profiles help to identify pairwise interactions between variables.

Cons

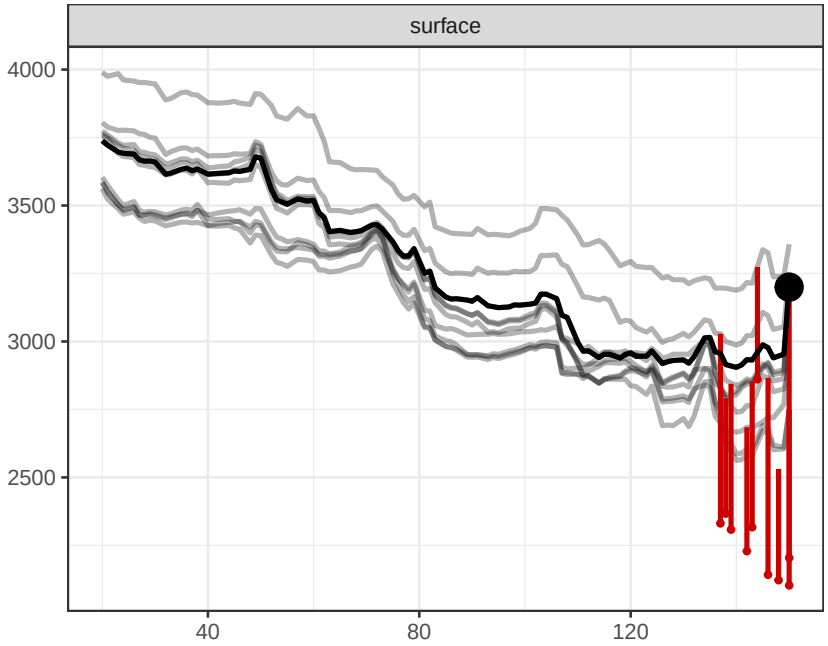


Figure 6.9: (fig:CPfidelity1) Local fidelity plots. Black line shows the CP profile for the point of interest. Grey lines show CP profiles for nearest neighbors. Red intervals correspond to residuals. Each red interval starts in a model prediction for a selected neighbor and ends in its true value of target variable.

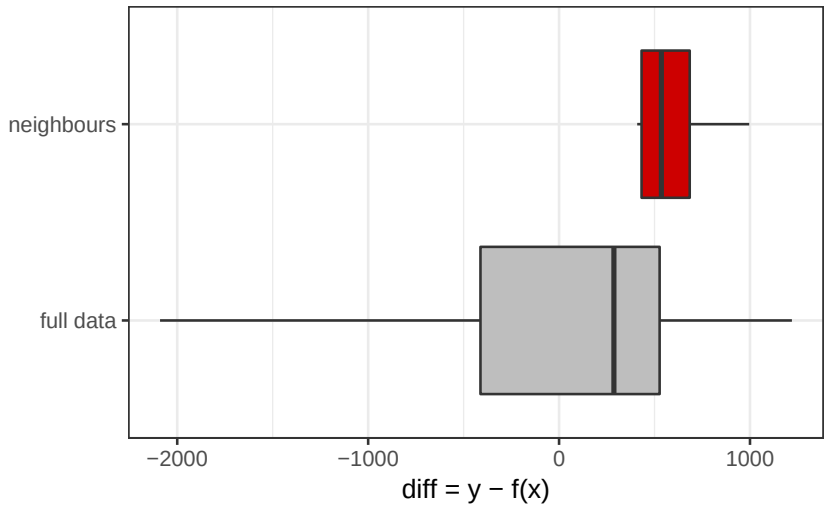


Figure 6.10: (fig:CPfidelityBoxplot) Distribution of residuals for whole validation data (grey boxplot) and for selected closes 15 neighbors (red boxplot).

- If variables are correlated (like surface and number of rooms) then the '*everything else kept unchanged*' approach leads to unrealistic settings.
- Interactions between variables are not visible in 1D plots.
- This tool is not suited for very wide data, like hundreds or thousands of variables.
- Visualization of categorical variables is non trivial.

6.7 Code snippets for R

In this section we present key features of the `ceterisParibus` package for R (Biecek, 2018b). This package covers all features presented in this chapter. It is available on CRAN and GitHub. Find more examples at the website of this package `https://pbiecek.github.io/ceterisParibus/`.

Model preparation

In this section we will present examples based on the `apartments` dataset. See section TODO for more details.

```
library("DALEX")
head(apartments)
```

```
##   m2.price construction.year surface floor no.rooms    district
## 1     5897              1953      25     3        1 Srodmiescie
## 2     1818              1992     143     9        5     Bielany
## 3     3643              1937      56     1        2       Praga
## 4     3517              1995      93     7        3      Ochota
## 5     3013              1992     144     6        5     Mokotow
## 6     5795              1926      61     6        2 Srodmiescie
```

The problem here is to predict average price for square meter for an apartment. Let's build a random forest model with `randomForest` package (Breiman et al., 2018).

```
library("randomForest")
rf_model <- randomForest(m2.price ~ construction.year + surface + floor +
      no.rooms, data = apartments)
rf_model
```

```
##
## Call:
##  randomForest(formula = m2.price ~ construction.year + surface +     floor + no.rooms, data = apartments)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 486660.5
##                     % Var explained: 40.74
```

Model exploration with `ceterisParibus` package is performed in four steps.

1. Create an explainer - wrapper around model and validation data.

Since all other functions work in a model agnostic fashion, first we need to define a wrapper around the model. Here we are using the `explain()` function from `DALEX` package (Biecek, 2018c).

```
library("DALEX")
explainer_rf <- explain(rf_model,
      data = apartmentsTest, y = apartmentsTest$m2.price)
explainer_rf
```

```
## Model label:  randomForest
## Model class:  randomForest.formula,randomForest
## Data head  :
##      m2.price construction.year surface floor no.rooms    district
## 1001     4644              1976     131     3        5 Srodmiescie
## 1002     3082              1978     112     9        4     Mokotow
```

2. Define point of interest.

Certeris Paribus profiles explore model around a single point.

```
new_apartment <- data.frame(construction.year = 1965, no.rooms = 5, surface = 142, floor = 8)
new_apartment
```

```
##   construction.year no.rooms surface floor
## 1              1965        5     142     8
```

```
predict(rf_model, new_apartment)
```

```
##        1
## 2311.108
```

3. Calculate CP profiles

The `ceteris_paribus()` function calculates CP profiles for selected model around selected observation.

By default CP profiles are calculated for all numerical variables. Use the `variables` argument to select subset of interesting variables. The result from `ceteris_paribus()`function is a data frame with model predictions for modified points around the point of interest.

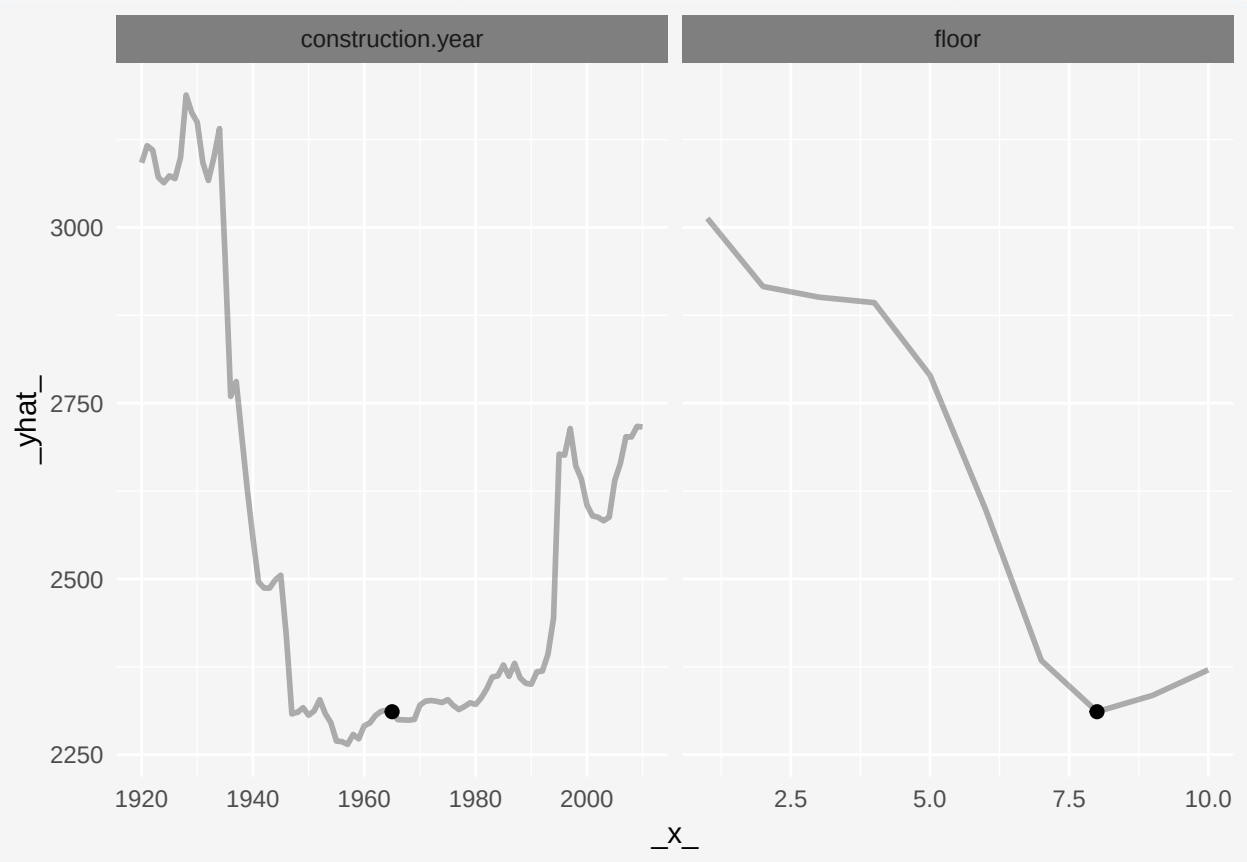
```
library("ceterisParibus")
cp_rf <- ceteris_paribus(explainer_rf, new_apartment,
                         variables = c("construction.year", "floor"))
cp_rf
```

```
## Top profiles    :
##     construction.year no.rooms surface floor   _yhat_           _vname_
## 1                1920        5     142     8 3091.884 construction.year
## 1.1              1921        5     142     8 3116.261 construction.year
## 1.2              1922        5     142     8 3110.009 construction.year
## 1.3              1923        5     142     8 3071.335 construction.year
## 1.4              1923        5     142     8 3071.335 construction.year
## 1.5              1924        5     142     8 3063.594 construction.year
##     _ids_      _label_
## 1       1 randomForest
## 1.1     1 randomForest
## 1.2     1 randomForest
## 1.3     1 randomForest
## 1.4     1 randomForest
## 1.5     1 randomForest
##
##
## Top observations:
##   construction.year no.rooms surface floor   _yhat_      _label_
## 1              1965        5     142     8 2311.108 randomForest
```

4. Plot CP profiles.

Generic `plot()` function plot CP profiles. It returns a `ggplot2` object that can be polished if needed. Use additional arguments of this function to select colors and sizes for elements visible in the plot.

```
plot(cp_rf)
```



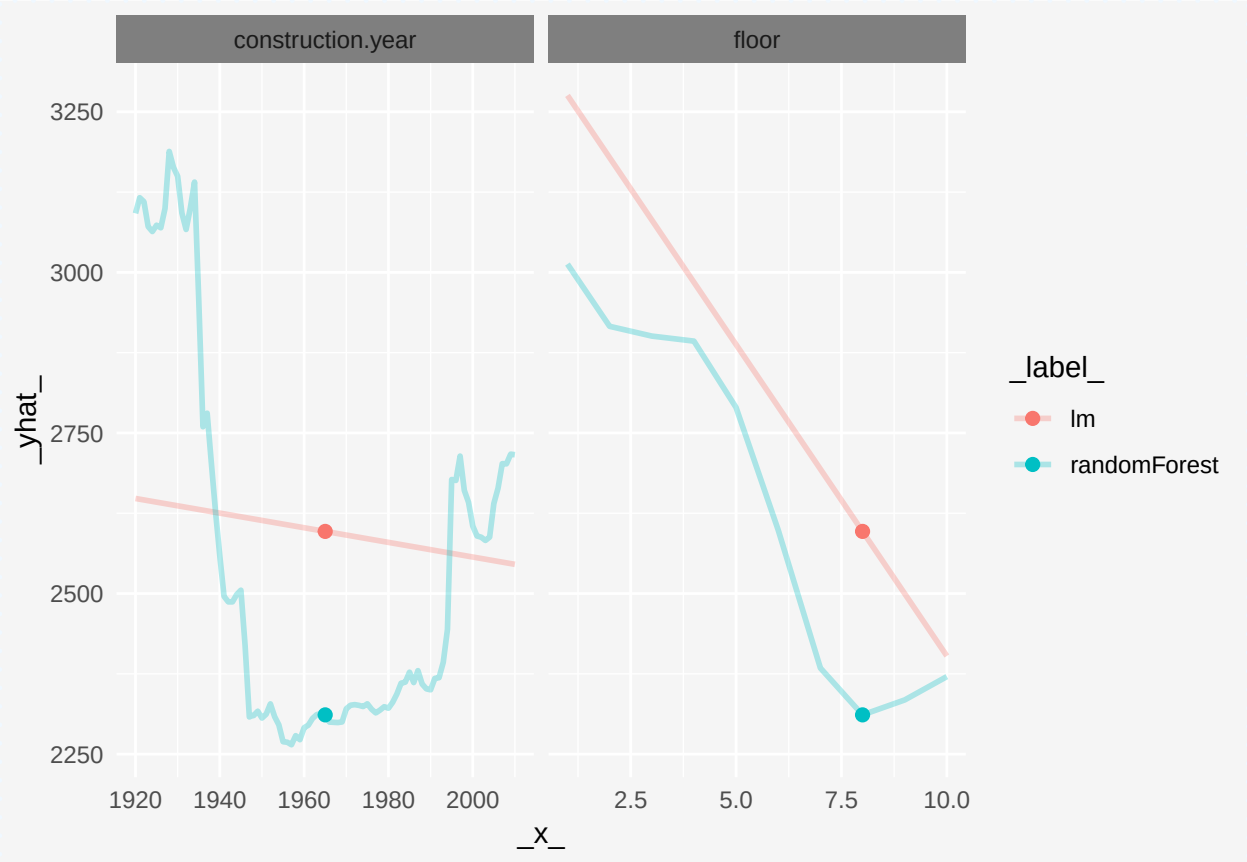
One of very useful features of `ceterisParibus` explainers is that profiles for two or more models may be superimposed in a single plot. This helps in model comparisons.

Let's create a linear model for this dataset and repeat steps 1-3 for the lm model.

```
lm_model <- lm(m2.price ~ construction.year + surface + floor +
      no.rooms, data = apartments)
explainer_lm <- explain(lm_model,
      data = apartmentsTest, y = apartmentsTest$m2.price)
cp_lm <- ceteris_paribus(explainer_lm, new_apartment,
                         variables = c("construction.year", "floor"))
```

Now we can use function `plot()` to compare both models in a single chart. Additional argument `color = "_label_"` set color as a key for model.

```
plot(cp_rf, cp_lm, color = "_label_")
```



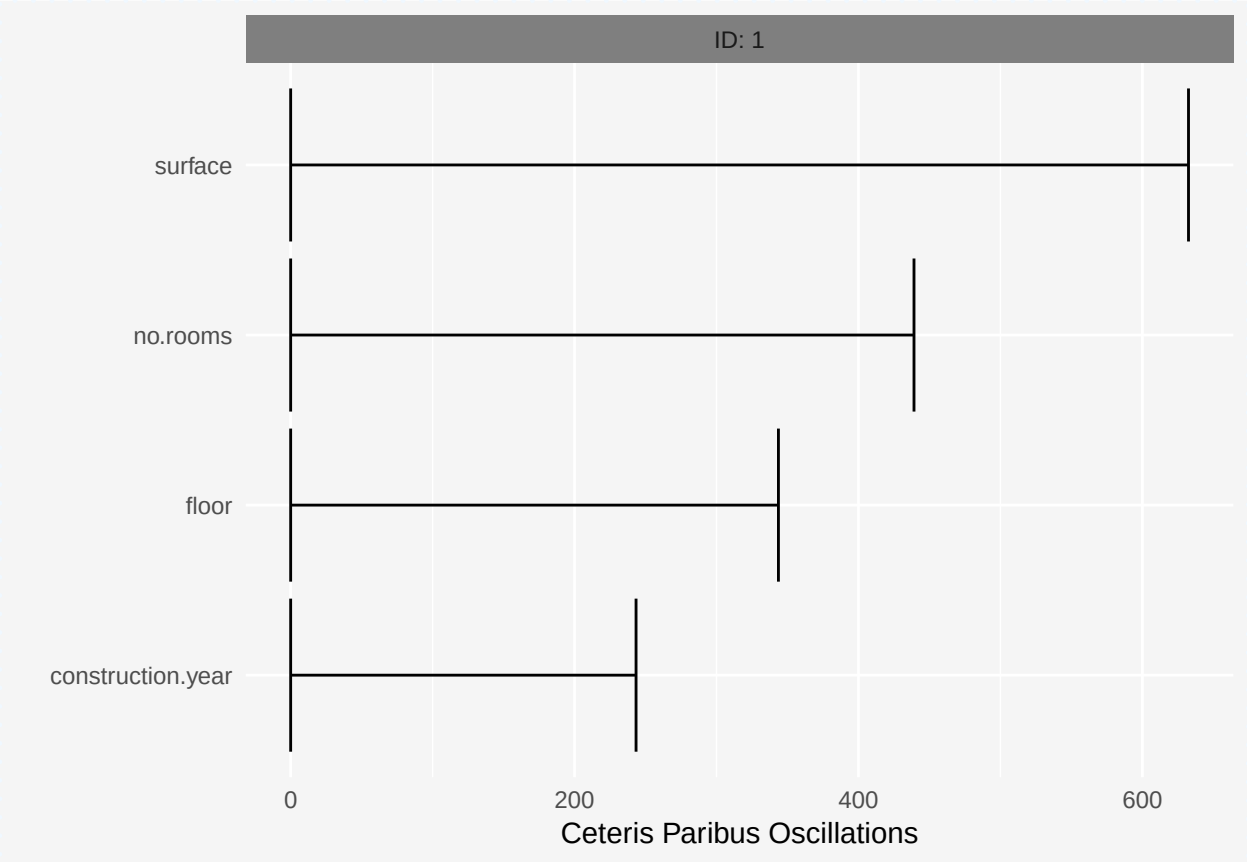
Oscillations

The `calculate_oscillations()` function calculates oscillations for CP profiles.

```
cp_rf_all <- ceteris_paribus(explainer_rf, new_apartment)
co_rf_all <- calculate_oscillations(cp_rf_all)
co_rf_all
```

```
##              _vname_ _ids_ oscillations
## 2            surface     1     632.5026
## 4          no.rooms      1     439.1318
## 3              floor     1     343.6137
## 1 construction.year      1     243.3204
```

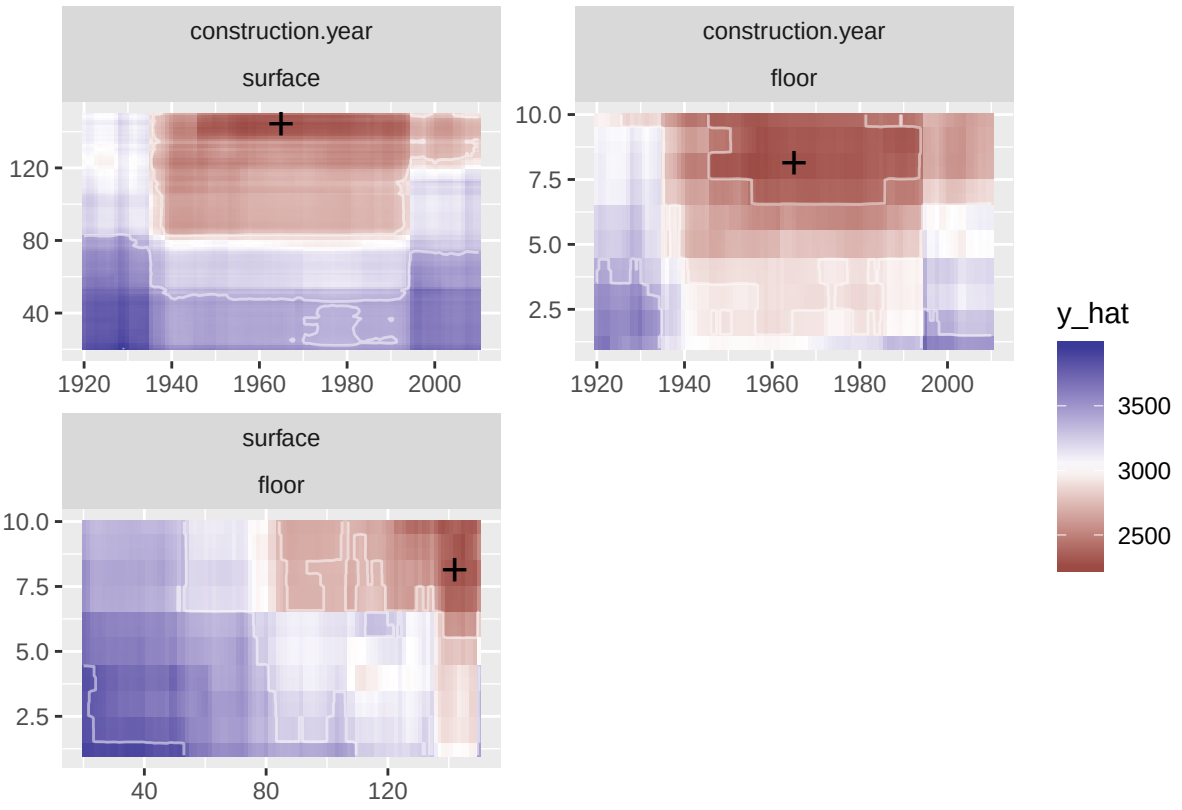
```
plot(co_rf_all)
```



2D Ceteris Paribus profiles

And the `what_if_2d()` function calculates 2D CP profiles.

```
wi_rf_2d <- what_if_2d(explainer_rf, observation = new_apartment,
                       selected_variables = c("surface","floor", "construction.year"))
plot(wi_rf_2d, split_ncol = 2)
```


construction.year
surface
120
80
40
1920
1940
1960
1980
2000
construction.year
floor
10.0
7.5
5.0
2.5
1920
1940
1960
1980
2000
surface
floor
10.0
7.5
5.0
2.5
40
80
120
y_hat
3500
3000
2500

Model level explanations

Chapter 7

Introduction

7.1 Example: Price prediction

(Biecek, 2018c)

In this chapter we show examples for three predictive models trained on `apartments` dataset from the `DALEX` package. Random Forest model (elastic but biased), Support Vector Machines model (large variance on boundaries) and Linear Model (stable but not very elastic). Presented examples are for regression (prediction of square meter price), but the CP profiles may be used in the same way for classification.

```
library("DALEX")
# Linear model trained on apartments data
model_lm <- lm(m2.price ~ construction.year + surface + floor +
                      no.rooms + district, data = apartments)

library("randomForest")
set.seed(59)
# Random Forest model trained on apartments data
model_rf <- randomForest(m2.price ~ construction.year + surface + floor +
                      no.rooms + district, data = apartments)

library("e1071")
# Support Vector Machinesr model trained on apartments data
model_svm <- svm(m2.price ~ construction.year + surface + floor +
                        no.rooms + district, data = apartments)
```

For these models we use `DALEX` explainers created with `explain()` function. There exapliners wrap models, predict functions and validation data.

```
explainer_lm <- explain(model_lm,
                        data = apartmentsTest[,2:6], y = apartmentsTest$m2.price)
explainer_rf <- explain(model_rf,
                        data = apartmentsTest[,2:6], y = apartmentsTest$m2.price)
explainer_svm <- explain(model_svm,
                        data = apartmentsTest[,2:6], y = apartmentsTest$m2.price)
```

Examples presented in this chapter are generated with the `ceterisParibus` package in version 0.3.1.

```
library("ceterisParibus")
```

Chapter 8

Variable Importance

Feature selection

Chapter 9

Marginal Response

Feature extraction

9.1 Partial Dependency Plots

Accumulated Local Effects (ALE) Plots

(?)

```
library(ALEPlot)
```

Interactions - extraction

9.2 Merging Path Plots

(?)

```
library(factorMerger)
```

Chapter 10

Performance Diagnostic

Model selection

Chapter 11

Residual Diagnostic

Model validation

Chapter 12

Other topics

Bibliography

Biecek, P. (2018a). *breakDown: Model Agnostic Explainers for Individual Predictions*. R package version 0.1.6.

Biecek, P. (2018b). *ceterisParibus: Ceteris Paribus Profiles*. R package version 0.3.1.

Biecek, P. (2018c). *DALEX: Descriptive mAchine Learning EXplanations*. R package version 0.2.4.

Breiman, L., Cutler, A., Liaw, A., and Wiener, M. (2018). *randomForest: Breiman and Cutler's Random Forests for Classification and Regression*. R package version 4.6-14.

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

Pedersen, T. L. and Benesty, M. (2018). *lime: Local Interpretable Model-Agnostic Explanations*. R package version 0.4.0.

Staniak, M. and Biecek, P. (2018). *live: Local Interpretable (Model-Agnostic) Visual Explanations*. R package version 1.5.7.

Xie, Y. (2018). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.7.