University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

Rafał Cylwa
Student no. 332078

Local Conditional Expectations: visualising feature variable dependence in supervised machine learning models

Master's thesis
in APPLIED MATHEMATICS

Supervisor:
dr hab. Przemysław Biecek
Faculty of Mathematics, Informatics and Mechanics
University of Warsaw

December 2018

Supervisor's statement

Hereby I confirm that the presented thesis was prepared under my supervision and that it fulfils the requirements for the degree of Master of Computer Science.

Date Supervisor's signature

Author's statement

Hereby I declare that the presented thesis was prepared by me and none of its contents was obtained by means that are against the law.

The thesis has never before been a subject of any procedure of obtaining an academic degree.

Moreover, I declare that the present version of the thesis is identical to the attached electronic version.

Date Author's signature

Abstract

In this master thesis we discuss methods of visualising feature variable dependence in supervised machine learning models. First chapter is devoted to theoretical description of existing methods and in the final section Local Conditional Expectations we propose a new method for visualising what-if scenarios. This method extends and improves Ceteris Paribus profiles by taking into account correlations between feature variables and adjusting their values according to a linear model fitted to the value of plotted variable in a what-if scenario. Second chapter covers use cases of methods described in chapter one using implementations in R language from the DALEX package family which are provided together with this master thesis.

Keywords

supervised learning, feature importance, linear regression, XAI, visualization

Thesis domain (Socrates-Erasmus subject area codes)

11.2 Statistics
11.3 Informatics, Computer Science

Subject classification

62 Statistics
62J Linear inference, regression
62J99 Others
68 Computer Science
68Q Theory of Computing
68Q32 Computational learning theory

Tytuł pracy w języku polskim

Local Conditional Expectations: wizualizacja zależności od zmiennych objaśniających w modelach uczenia nadzorem

Contents

Chapter 1

Motivation and background

In recent years we have been witnessing a renaissance of artificial intelligence research, mostly in the field of machine learning. Thanks to great algorithmic advances as well as rapid rise and ease of availability of compute power, machine learning is gaining increasing interest both in scientific and business community. Chairman of the World Economic Forum, Klaus Schwab proposes in his book [Schwab, 2017] that we are on the verge of so-called Fourth Industrial Revolution. Machine learning systems are now or will be soon responsible for driving cars, detecting cancer in radiology practice, pricing and trading financial instruments, deciding on granting a loan and many, many more. However, that great effectiveness of these systems comes at a price. Very often the most accurate algorithms are also the most complicated and sophisticated ones and are usually referred to as black boxes. If these methods are supposed to play a crucial role in making automated but important decisions or assisting in human-made decisions we need to make sure we understand how these black boxes work, how do they come to their conclusions and hence we can trust them.

The need for explainability is in fact driven by many different factors. Verification of the system is one of them e.g. domain experts like radiologist without machine learning knowledge need to be able to interpret and verify the model results using their own expertise. Another factor is learning from the system. Famous Google's DeepMind AplhaGo system from [Silver et al., 2017] achieved super-human level in playing the Go game and invented some new moves, never explored by humans. By extracting the knowledge from the system one could find interesting facts about the researched domain.

Nowadays, explainability became so important that it is now enforced by EU regulation called General Data Protection Regulation. The "right to explanation" is stated in Article 22, Paragraph 3 of GDPR (see [Goodman and Flaxman, 2016]). Given that, companies that e.g. do algorithmic trading or automated loan granting need to provide *explanations* for how do their systems work.

All the reasons stated above gave birth to a new subfield of machine learning research called XAI (Explainable Artificial Intelligence) that aims to provide a better transparency of all machine learning models. Throughout this thesis we will discuss some particular topics that aim towards this goal.

Chapter 2

PDP, ICE, ALE, Ceteris Paribus and Local Conditional Expectations

In this chapter we establish framework with notation and present existing methods for measuring and visualising effects of feature variables in supervised black box models. In the last section we propose new method for visualising feature variable dependence around a single observation.

In the supervised learning setting we are working with a set observations

$$\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$$

called a training set where each observation consist of a vector of feature variables $x_i \in \mathbb{R}^p$ and a response variable $y \in \mathcal{Y}$. We claim each x_i and y_i is a realisation of a random variable X and Y respectively and we assume that there exists some functional relationship between X and Y meaning we can write that

$$Y = f(X) + \epsilon \tag{2.1}$$

where f is some unknown function and ϵ is an independent random error term with mean equal to zero. Depending on whether $\mathcal{Y} = \mathbb{R}$ or $\mathcal{Y} = \{G_1, G_2, ..., G_k\}$ we are dealing with either regression or classification task respectively (in the second case we say that G_i is a label of i-th class) but the task is always to find an estimate $\hat{f}$ of function f based on the training data.
There are two main reasons to estimate f:

- prediction - when we predict the value of y for previously unseen feature vector x. In this setting we often treat a model as a "black box" and care mostly about the accuracy of a prediction and not the internal structure of estimated $\hat{f}$

- inference - when we want to investigate the details of the dependence between x and y e.g. finding which features in the feature vector are "important" in the estimated model or what is the relationship between individual features and the response y.

One of the most popular supervised learning techniques, widely used in many fields of application, is linear regression. It assumes a linear relationship between the response variable and features of the form:

$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p \tag{2.2}$$

while beta coefficients are estimated to best fit the training data. This is an example of a *parametric* model where we assume upfront a certain type of functional relationship. Hence its easy to interpret and make inference based on such fitted model. However, despite great interpretability, when it comes to prediction accuracy these methods often loose with some less restrictive *non-parametric* approaches. On the other hand, these more flexible methods are oftentimes very hard to interpret, meaning it is difficult to determine e.g. which factors are important how does the response depend on individual features unless you don't know all the technical details about how the model is being trained etc. and even then, due to their internal complexity, these models are considered black boxes.

Fortunately, even if we treat supervised machine learning models as black boxes returning outputs and not assuming anything about their internal structure there exist some model-agnostic methods of unveiling this structure. We will now discuss those that relate to model response for individual features.

2.1. Partial Dependence Plots

Partial dependence plots, described in [Friedman et al., 2001], are designed to show the dependence of model response and a single feature by averaging over remaining features.

Let $\hat{f}(x) : \mathbb{R}^p \to \mathbb{R}$ be a trained black box model function i.e. an estimate of the real, unknown f and let $x_{-i} = \{x_1, x_2, ..., x_p\} \setminus \{x_i\}$. We define Partial Dependence Function for feature x_i and function f by

$$PD(x_i) = \mathbb{E}_{x_{-i}}[f(x_i, x_{-i})] = \int f(x_i, x_{-i}) dP(x_{-i}) \tag{2.3}$$

where $dP(x_{-i})$ is the marginal probability distribution of x_{-i}. This can be estimated from

the training data by

$$\hat{PD}(x_i) = \frac{1}{n}\sum_{j=1}^{n} \hat{f}(x_i, x^j_{-i}) \tag{2.4}$$

Of course one can define partial dependence function for more than just one feature at a time (two dimensions make also sense for visualisation purposes) but we will limit ourselves to one dimension (given that methods described later require that).

Partial Dependence Plots are a global method, meaning that they describe a global structure of a model (in contrary to local methods that focus on individual observations or sets of observations). They:

- are simple to explain to a layperson
- are easy to implement
- represent how a selected feature on average influences the response of a model

However one must be careful since they can be misleading when feature variables are correlated. In formula 2.3 we average over a marginal distribution so that if there are some highly correlated variables we will include into the average some very "unlikely" data points i.e. with a very "unlikely" combination of feature values. The issue of correlation is addressed by ALE plots in [Apley, 2016] by using conditional distribution rather than a marginal one. The second problem with PDP can occur when there are interaction effects between feature variables. Averaging in 2.3 can hide these potential interactions leading to an unreliable plot. This issue is addressed by Individual Conditional Expectation plots in [Goldstein et al., 2015] and we will discuss them in the next section.

Nevertheless, Partial Dependence Plots oftentimes serve as a good summary of feature dependence but must be used with caution and some more detailed inspection of the model and data itself.

2.2. Individual Conditional Expectation plots

Oftentimes PDPs can be a usefull summary of the relationship of a model and an individual feature variable but we need to use them with causion since they can be misleading e.g. in case of interactions of the presented variable with some other feature variable. We can investigate if this is the case using so-called Individual Conditional Expectation plots introduced in [Goldstein et al., 2015], which are "desintegrated" PDP's meaning that instead of one aggregated curve (PDP) we plot N individual curves, each corresponding to single observation. More precisely, for a specific observation $x^* = (x_1^*, x_2^*, ..., x_p^*)$ from the training dataset the corresponding ICE curve for feature x_i is simply $\hat{f}(x_i, x_{-i}*)$ plotted against x_i (all features apart x_i are frozen at x^*).

$$ICE_{\hat{f}}(x_i, x*) = \hat{f}(x_i, x^*_{-i}) \tag{2.5}$$

They have the following property that if x_i has no interactions with other variables then all curves in the ICE plot will be parallel. Such situation is presented in Figure 2.1.

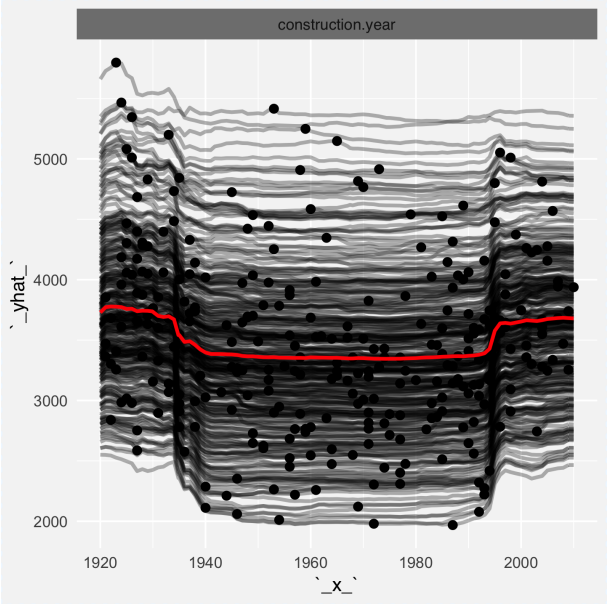


Figure 2.1: Black curves represent Individual Conditional Expectation with black dots representing individual observations. Red curve represents a partial dependence function i.e. an average of black curves.

Figure 2.1 reassures us that there are no interactions between presented feature variable and some other variable, hence a corresponding PDP can be a good summary of the relationship e.g. for presentation purposes.

Sometimes, like in Figure 2.2, ICE curves are not parallel and a corresponding PDP gives a false picture of how the dependency actually looks like. In this case there is and interaction of presented x variable with some other variable. By isolating ICE curves into two groups

corresponding to different values of this "interacting" variable (see Figure 2.3) one can see that the relationship is strictly positive or strictly negative, depending on the group. PDP averages out this effects leading to a flat PDP (red line in Figure 2.2).

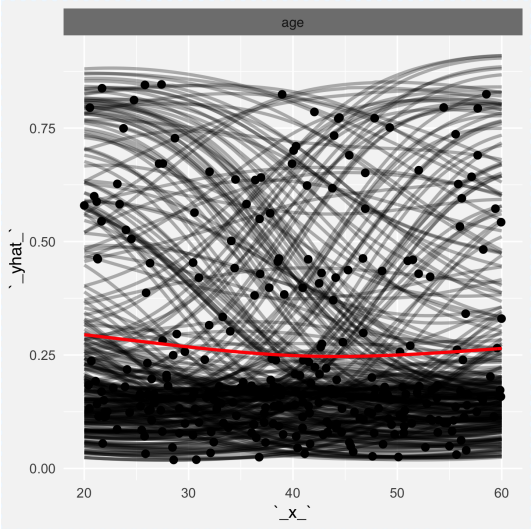


Figure 2.2: Black curves represent Individual Conditional Expectation with black dots representing individual observations. Red curve represents a partial dependence function i.e. an average of black curves.

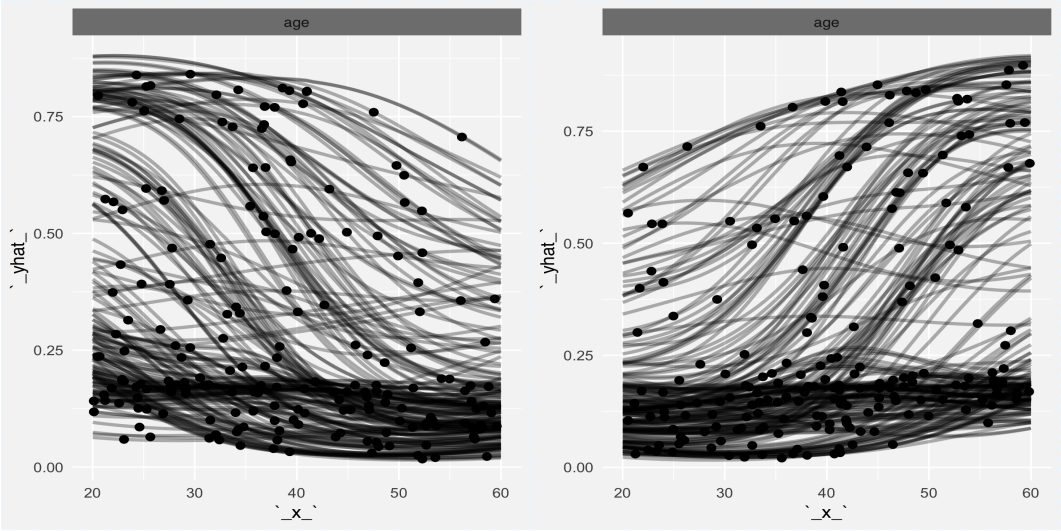


Figure 2.3: ICE curves from from Figure 2.2 separated into two groups corresponding to two different levels of a variable interacting with presented variable x.

This example indicates that PDPs should be always accompaigned with ICE plots to justify their correctness.

2.3. Accumulated Local Effects plots

As we have mentioned previously there is an issue with PDP properly representing global dependence of a model when a particular feature is strongly correlated with another feature. Firstly, the averaging in 2.4 can be calculated in some very "unlikely" points. E.g. imagine one of the variables is apartment surface and other in the number of rooms. This procedure can lead to calculating the value of $\hat{f}$ in data points like having 8 rooms and $20m^2$, which is very unrealistic. To cope with that, [Apley, 2016] proposes using conditional distribution rather than marginal ones resulting in so-called M-plot:

$$P_M(x_i) = \mathbb{E}_{x_{-i}|x_i}[f(x_i, x_{-i})] = \int f(x_i, x_{-i}) dP(x_{-i}|x_i) \tag{2.6}$$

But even if we narrow down ourselves to conditional distribution, eliminating the risk of "unlikely" data points, we estimate a combined effects of all variables rather than just the one of interest (by conditioning in one variable we also unintentionally narrow down the range of a correlated variable). To eliminate this, instead of calculating averages, we calculate differences in prediction on the boundary of a small neighbourhood of x_i and accumulate them over the grid defining the neighbourhood (see Figure 2.4a).

Formally they are defined as:

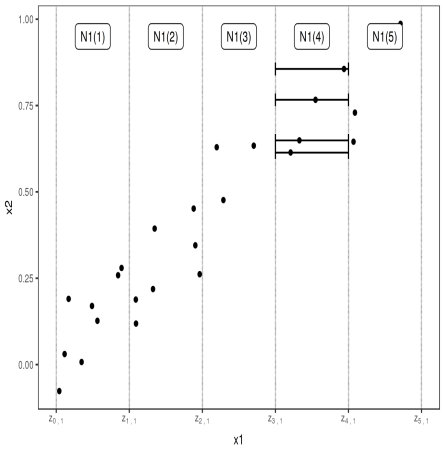
$$P_{ALE}(x_i) = \int_{z_0,1}^{x_i} \int f^i(z_i, x_{-i}) dP(x_{-i}|z_i) dz_i - const \tag{2.7}$$

where

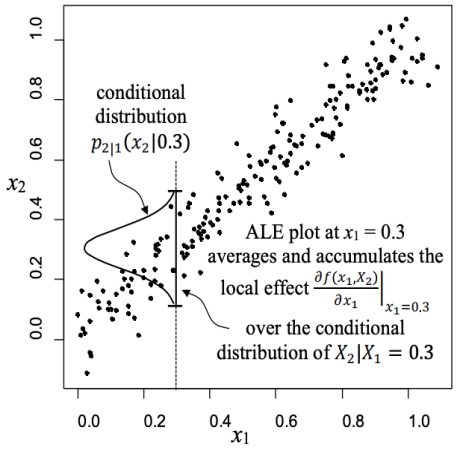
$$f^i(x_i, x_{-i}) = \frac{\partial f(x_i, x_{-i})}{\partial x_i} \tag{2.8}$$

In implementation, as stated before, the derivatives are replaced by differences over intervals (see Figure 2.4a) conditioned on x_i. This derivatives (or differences over intervals) isolate the effect of the feature of interest and block the effect of correlated features. The whole procedure is intuitively depicted in Figure 2.4b.

Although ALE plots can present some advantages over Partial Dependence plots they are much more less intuitive and complicated to describe to a layperson. This makes them probably better suited as a tool for and expert data scientist rather than general purpose method. They are a relatively new method and so far only implemented in [Apley, 2018].



(a) Interval grid for calculating differences of predictions in 2.7



(b) Illustration of the computation of ALE plot at $x_1 = 0.3$

Figure 2.4

2.4. Ceteris Paribus profiles

So far we were discussing methods that describe a global structure of a model i.e. what is the general pattern of model response to a individual features. One might also want to ask some questions about local structure, meaning how does the model behave near a certain observation. Let us imagine that an insurance company has a scoring model to assess the creditworthiness of their potential customers based on their "features". A customer that got a low score might be interested in improving it by asking a question: "How would my score change if I change variable x_k (e.g my wage)?" Or let us take a real estate brokerage company that has a model for estimating prices of apartments. A client interested in an apartment in a certain district, on the first floor etc. but would like to see how the price would change if we were looking for something with more rooms. In other words we are asking how does the model response changes if we change the value of one feature, having all other features unchanged (lat. ceteris paribus).

To answer that we can use so called Ceteris Paribus plots from [Biecek, 2018a], which are defined exactly the same as ICE curves but in this case we don't plot all of them at the same time but rather only one curve linked with a particular observation (like a particular credit client or an individual apartment).

Ceteris Paribus profile of variable x_i for a model $\hat{f}$ and an observation x^* is defined as:

$$CP_{\hat{f}}(x_i, x*) = \hat{f}(x_i, x^*_{-i}) \tag{2.9}$$

They can be used to conduct "what-if" scenario analysis like in the examples with real estate prices and credit scoring. They can be used in regression setting (see Figure 2.5) or probabilistic classification i.e. when an algorithm outputs a probability of belonging to a certain class. In the second case the y-axis of Figure 2.5 represents probability.

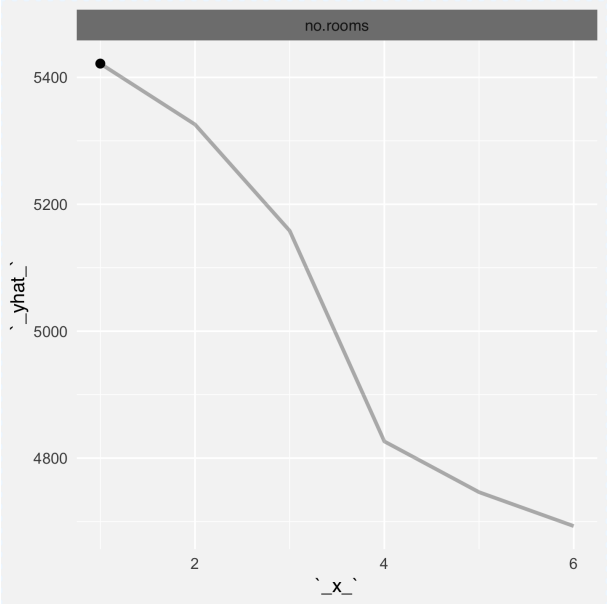


Figure 2.5: Ceteris paribus profile for individual observation (black dot).

Although they can be useful to analyze local structure of a model they suffer similar problem as PDP i.e. they don't take into account correlations. As per example with apartment price when conducting a "what-if" analysis for a given observation, when increasing solely the number of rooms one should also increase the surface. Ceteris Paribus principle should not be used for such highly correlated variables.

2.5. Local Conditional Expectation profiles

As we have mentioned in previous chapter, PDP's and ICE plots can be useful in unveiling the global structure of the model, but when want to move to local structure i.e. conduct and visualise a "what-if" scenario for a single observation the Ceteris Paribus profiles may be misleading due to their reliance on potentially "unlikely" data.

We propose a method that, instead of keeping all the features unchanged, adjusts the values of features to the changing value of the feature that is being plotted.

Let $x^* = (x_1^*, x_2^*, ..., x_p^*)$ be an observation for which we want to calculate a what-if scenario and let $\hat{f} : \mathbb{R}^p \to \mathbb{R}$ be a trained "black box" machine learning model. We define the Local Conditional Expectation function for i-th feature as:

$$LCE_{\hat{f},x^*}(x_i) := \begin{cases} \hat{f}(x_1^*, ..., x_i, ..., x_p^*) & when\ x_i = x_i^* \\ \hat{f}(E(x_1|x_i, x^*), ..., x_i, .., E(x_p|x_i, x^*)) & elswhere \end{cases} \tag{2.10}$$

To define $E(x_j|x_k, x^*)$, we first simple linear pairwise relationship between x_j and x_k i.e. $x_j \approx \alpha x_k + \beta$ to capture the general trend of this relationship. After finding estimates $\hat{\alpha}$

and $\hat{\beta}$ we furthermore adjust $\hat{\beta}$ so that the regression line passes (see Figure 2.6) through the (x_j^*, x_k^*) point i.e.

$$\hat{\beta}^* := x_j^* - \alpha x_k^* \tag{2.11}$$

and finally we define

$$E(x_j|x_k, x^*) := \hat{\alpha} x_k + \beta^* \tag{2.12}$$

This shifting of $E(x_j|x_k, x^*)$ to pass through x^* provides continuity of the LCE profile.

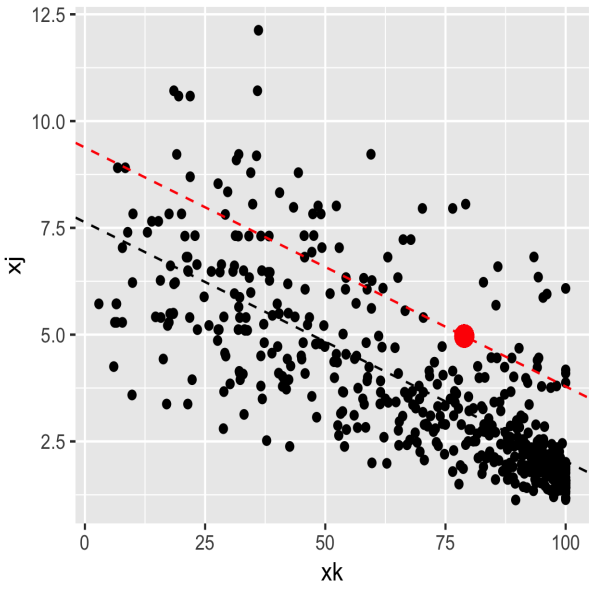


Figure 2.6: Black line is a fitted linear regression line between x_j and x_k. Red line represents $E(x_j|x_k, x^*)$, i.e. shifted black line to pass through observation x^*

The advantages of LCE profiles over Ceteris Paribus profiles as well as the justification for their construction are discussed in the next chapter.

Chapter 3

Use cases

This chapter is devoted to practical use cases of methods described in Chapter 2. All methods are implemented in R language packages. We provide R code for using existing methods and an implementation with how-to-use of Local Conditional Expectation profiles.

The following datasets are used in all examples:

- **Apartments** is an artificial dataset and it is attached to DALEX package. Structure of the dataset is copied from real dataset from PBImisc package. The regression task is to predict price per m^2 of an apartment in Warsaw using construction year, surface, floor, number of rooms and district.
- **Dragons** is an artificial dataset and is attached to DALEX2 package. We will work with classification task i.e. classify longevity of a dragon based on year of birth, year of discovery, height, weight, number of scars, colour and number of lost teeth.

3.1. PDP and ICE plots

We fit Support Vector Machine and Random Forest models to the apartments data and create Partial Dependence plots using DALEX ([Biecek, 2018b]) package framework (PDP are also implemented in *pdp* package [Greenwell, 2017]).
Libraries used:

```
library("DALEX")                # data and explainers
library("ceterisParibus")       # profile calculation
library("randomForest")         # Random Forest model
library("e1071")                # SVM model
```

Fitting models:

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

Create explainer objects from DALEX package:

```
explainer_rf <- explain(rf_model,
                        data = apartments[,2:6],
```

```
                          y = apartments$m2.price)

explainer_svm <- explain(svm_model,
                         data = apartments[, 2:6],
                         y = apartments$m2.price)
```

Calculating ceteris paribus profiles:

```
cp_rf <- ceteris_paribus(explainer_rf,
                         apartments,
                         y = apartments$m2.price)

cp_svm <- ceteris_paribus(explainer_svm,
                         apartments,
                         y = apartments$m2.price)
```

Plotting aggregated ceteris paribus profiles i.e. PDP for variables *surface* and *construction.year*:

```
pdp_surface <- plot(cp_rf, cp_svm,
                    selected_variables = "surface",
                    aggregate_profiles = mean,
                    show_observations = FALSE,
                    color = "_label_")

pdp_construction_year <- plot(cp_rf, cp_svm,
                              selected_variables = "construction.year",
                              aggregate_profiles = mean,
                              show_observations = FALSE,
                              color = "_label_")
```

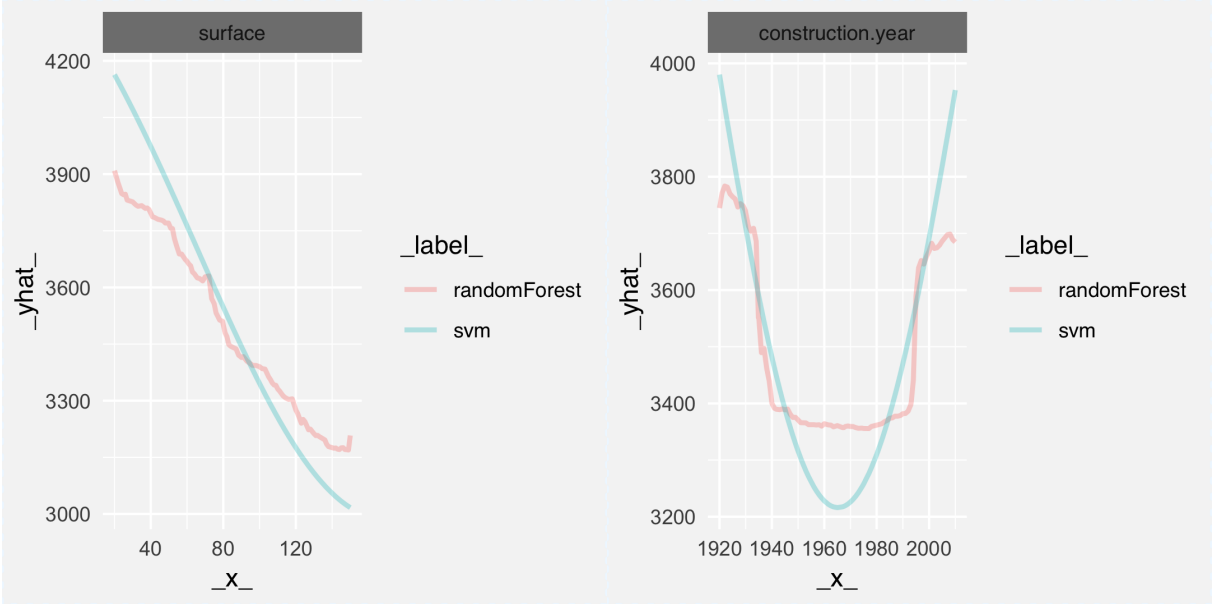


Figure 3.1: PDP plots for *surface* and *construction.year* variables comparing SVM and Random Forest models. The y-axis is apartment's price per m^2.

Looking at Figure 3.1 one can infer that both models have similar global structure i.e. price per m^2 decreases with increasing *surface* of the whole apartment. When it comes to

construction.year the cheapest apartments are those from 1970s era with both very new or very old being the most expensive ones.

In order to produce ICE plots for the same example let us choose SVM model (since ICE plots for two models in one plot would not be readable), remove aggregate_profiles line from the PDP construction code and add a layer to the plot representing a PDP.

```
ice_surface_svm <- plot(cp_svm,
                        selected_variables = "surface",
                        show_observations = TRUE) +
                   ceteris_paribus_layer(cp_svm,
                        selected_variables = "surface",
                        aggregate_profiles = mean,
                        color = "red",
                        show_observations = FALSE,
                        alpha = 0.9)

ice_construction_svm <- plot(cp_svm,
                             selected_variables = "construction.year",
                             show_observations = TRUE) +
                        ceteris_paribus_layer(cp_svm,
                             selected_variables = "construction.year",
                             aggregate_profiles = mean,
                             color = "red",
                             show_observations = FALSE,
                             alpha = 0.9)
gridExtra::grid.arrange(ice_surface_svm, ice_construction_svm, ncol = 2)
```

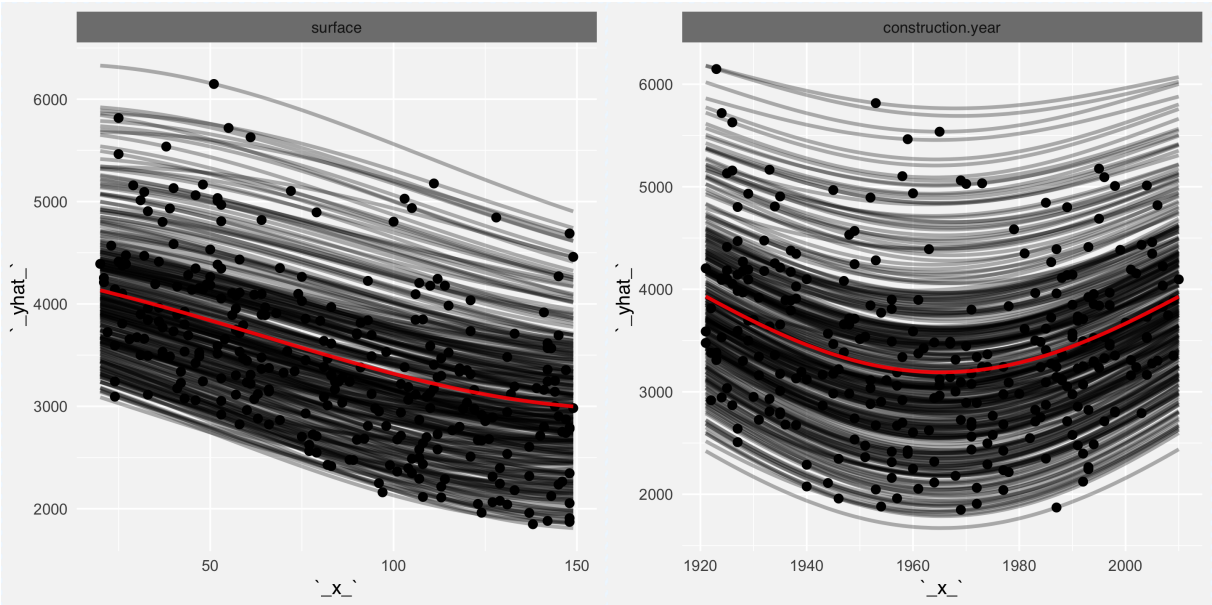


Figure 3.2: ICE plots for *surface* and *construction.year* variables and SVM model. The y-axis is apartment's price per m^2. Black dots stand for individual observations. Red PDP curve is added using ceteris_paribus_layer function.

Since ICE curves in Figure 3.2 are all parallel to each other, PDP is a good choice for visualising this dependence.

3.2. LCE profiles

We will now demonstrate the usage of Local Conditional Expectation profiles in visualising "what-if" scenarios with a direct comparison to Ceteris Paribus profiles using a ceterisParibus R package ([Biecek, 2018a]) implementation in both regression and classification tasks. We examine their behaviour on a wide variety of models form simple linear regression to neural networks.

3.2.1. Regression

We will use the Apartments dataset from DALEX package. Apart from previously fit Random Forest and SVM we add Linear Regression:

```
lm_model <- lm(m2.price ~ construction.year + surface +
                          floor + district + no.rooms, data = apartments)
```

Construct explainer objects:

```
explainer_lm <- explain(lm_model,
                        data = apartments[, 2:6],
                        y = apartments$m2.price)
```

Create a single observation for "what-if" analysis:

```
new_obs <- data.frame(construction.year = 1954,
                      surface = 145,
                      floor = 7L,
                      no.rooms = 6,
                      district = factor("Zoliborz", levels = aplevels))
```

And calculate both Ceteris Paribus and Linear Conditional Expectation profiles for all three models:

```
cp_rf <- ceteris_paribus(explainer_rf, new_obs)
cp_rf_lce <- local_conditional_expectations(explainer_rf, new_obs)

cp_svm <- ceteris_paribus(explainer_svm, new_obs)
cp_svm_lce <- local_conditional_expectations(explainer_svm, new_obs)

cp_lm <- ceteris_paribus(explainer_lm, new_obs)
cp_lm_lce <- local_conditional_expectations(explainer_lm, new_obs)
```

Using the functionality of the Ceteris Paribus R package we can plot Random Forest profiles for each variable in a grid (see Figure 3.3).

```
lce_rf_apartments <- plot(cp_rf,
                          selected_variables = selected_variables) +
                     ceteris_paribus_layer(cp_rf_lce,
                        selected_variables = selected_variables,
                        color = "red")
```

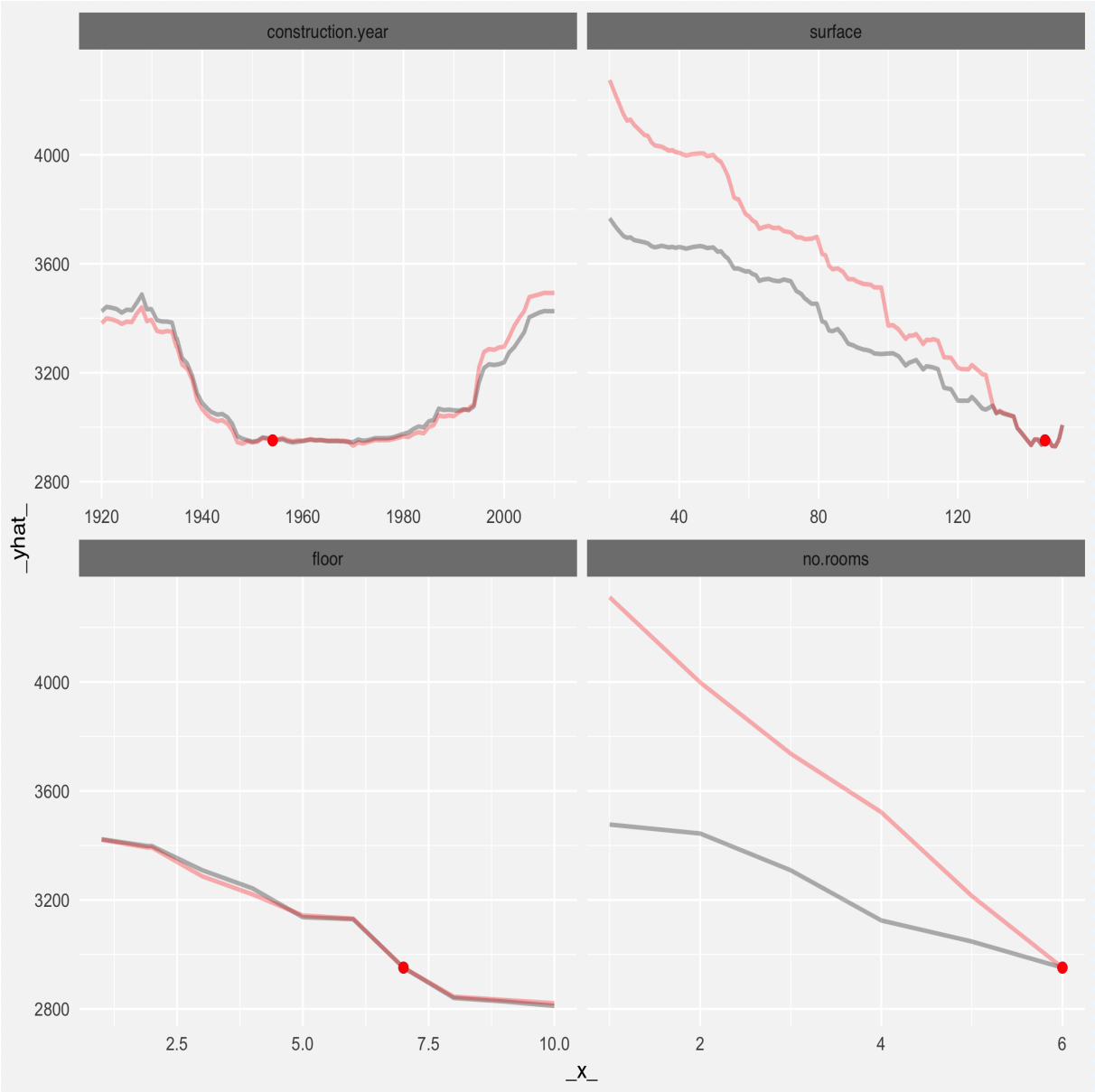


Figure 3.3: Ceteris paribus (black line) and Local Conditional Expectation (red line) profiles for Random Forest model fitted to apartments data. Red dot represents an apartment chosen for analysis.

By superimposing CP and LCE profiles we observe that they are almost the same for *construction.year* and *floor* but differ for both *surface* and *no.rooms*. Intuition suggests that both *construction.year* and *no.rooms* are not dependent on any other feature variable, hence LCE profiles should be the same as Ceteris Paribus profiles. On the other hand, there is an obvious correleation between *surface* and *no.rooms*, see correlations table 3.1.

	no.rooms	surface	construction.year	floor
no.rooms	1.00	0.91	0.05	0.04
surface	0.91	1.00	0.07	0.02
construction.year	0.05	0.07	1.00	-0.02
floor	0.04	0.02	-0.02	1.00

Table 3.1: Correlation matrix for Apartments feature variables

To further analyze the behaviour of LCE let us calculate and plot model-agnostic feature importance from [Fisher et al., 2018]. It measures feature's importance by calculating the increase of the model's prediction error after permuting the feature values. The more such permutation increases prediction error the more important the feature is. It can be calculated using DALEX framework:

```
rf_importance_plot <- explainer_rf %>% variable_importance() %>% plot()

svm_importance_plot <- explainer_svm %>% variable_importance() %>% plot()

lm_importance_plot <- explainer_lm %>% variable_importance() %>% plot()
```

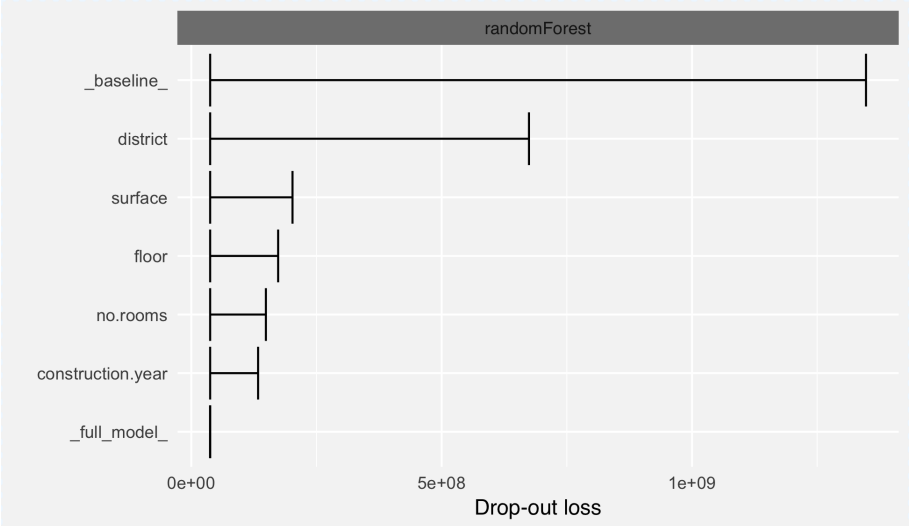


Figure 3.4: "Drop-out" feature importance plot for apartments data and Random Forest model

The fact that both *surface* and *no.rooms* variables show differences in CP versus LCE profiles is a manifestation of a property of random forests seen in Figure 3.4 i.e. treating correlated variables with similar importance (we will see that is not the case for other models). Nevertheless this difference is a desired outcome meaning LCE represents a better what-if scenario than Ceteris Paribus by taking into account the correlation between *surface* and *no.rooms*.

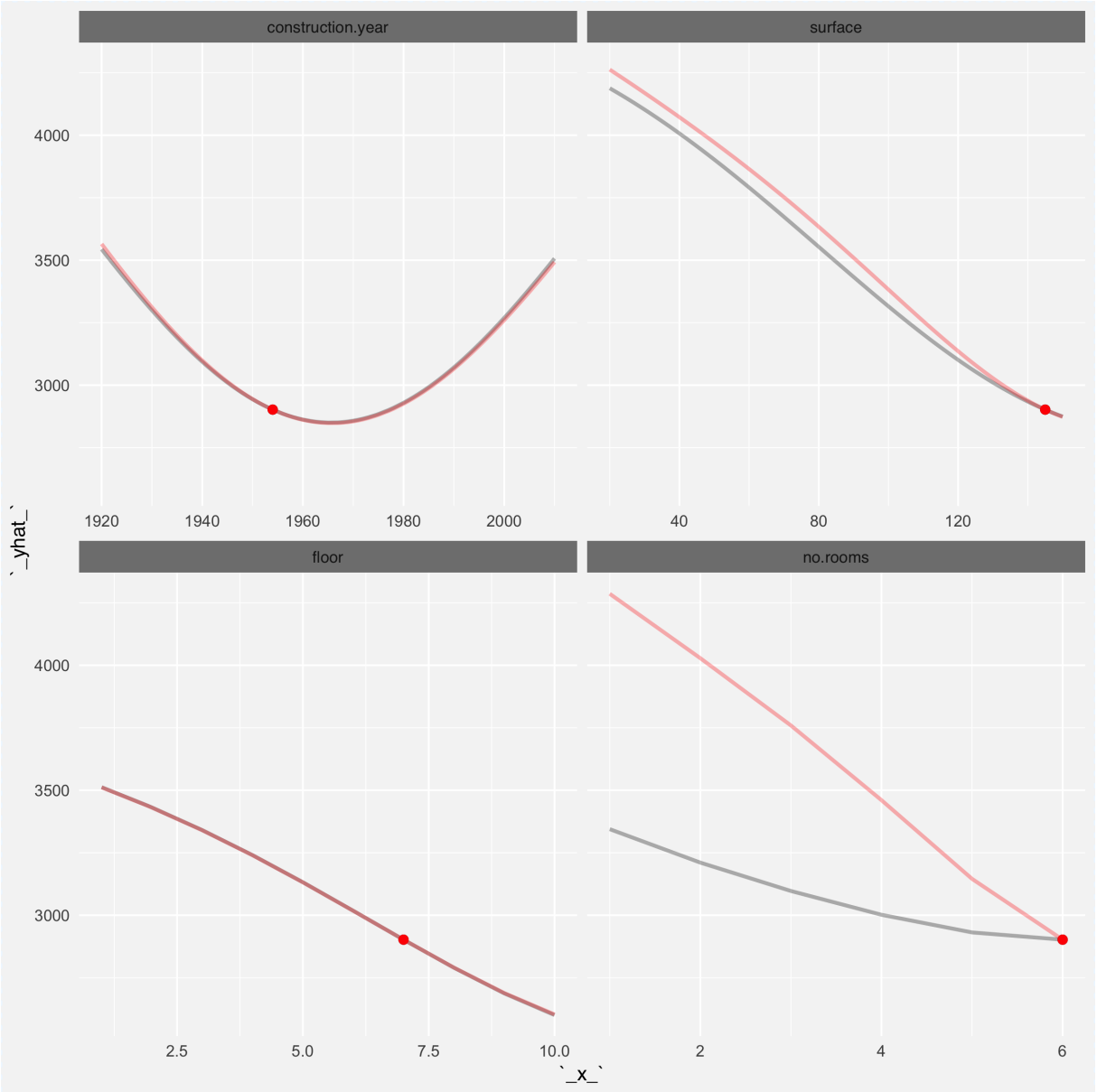


Figure 3.5: Ceteris paribus (black line) and Local Conditional Expectation (red line) profiles for SVM model fitted to apartments data. Red dot represents an apartment chosen for analysis.

Next we create profiles for SVM and linear model:

```
lce_svm_apartments <- plot(cp_svm,
                           selected_variables = selected_variables) +
  ceteris_paribus_layer(cp_svm_lce,
                        selected_variables = selected_variables),
                        color = "red")

lce_lm_apartments <- plot(cp_lm,
                          selected_variables = selected_variables) +
                     ceteris_paribus_layer(cp_lm_lce,
                          selected_variables = selected_variables,
                          color = "red")
```

The advantage of LCE over CP profiles is even more visible in the SVM model. Here, the difference is present only on the *no.rooms* plot (see Figure 3.5). In this case variable *no.rooms* with low importance (see Figure 3.6) is correlated with high importance *surface* variable leading to a difference in CP versus LCE plots and a better representation of the what-if scenario by the LCE plot. On the other hand *surface* variable plots show no difference because of SVM neglecting (meaning low importance) *no.rooms* variable. LCE's behaviour for Linear regression is analogous to the one of SVM and presented in Figure 3.7.

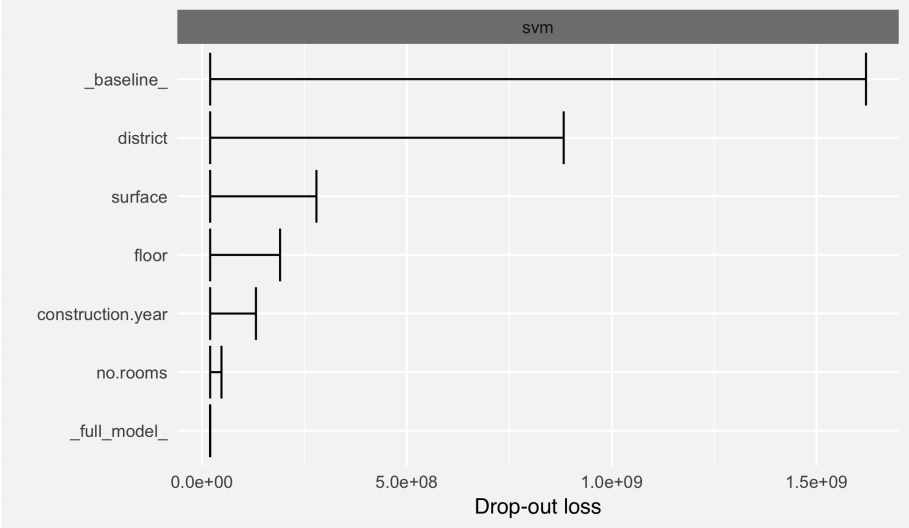


Figure 3.6: "Drop-out" feature importance plot for apartments data and SVM model

DALEX and ceterisParibus packages support models fit using *mlr* framework (see [Bischl et al., 2016]). We will use it to fit Gradient Boosting Machine and single-layer neural network. When using *mlr* one need to slightly modify predict function to extract proper predicted values.

```
set.seed(1234)
regr_task <- makeRegrTask(id = "ap", data = apartments, target = "m2.price")
regr_lrn_gbm <- makeLearner("regr.gbm", par.vals = list(n.trees = 500))
regr_gbm <- train(regr_lrn_gbm, regr_task)
custom_predict <- function(object, newdata) {
  pred <- predict(object, newdata = newdata)
  response <- pred$data$response
  return(response)
}
```

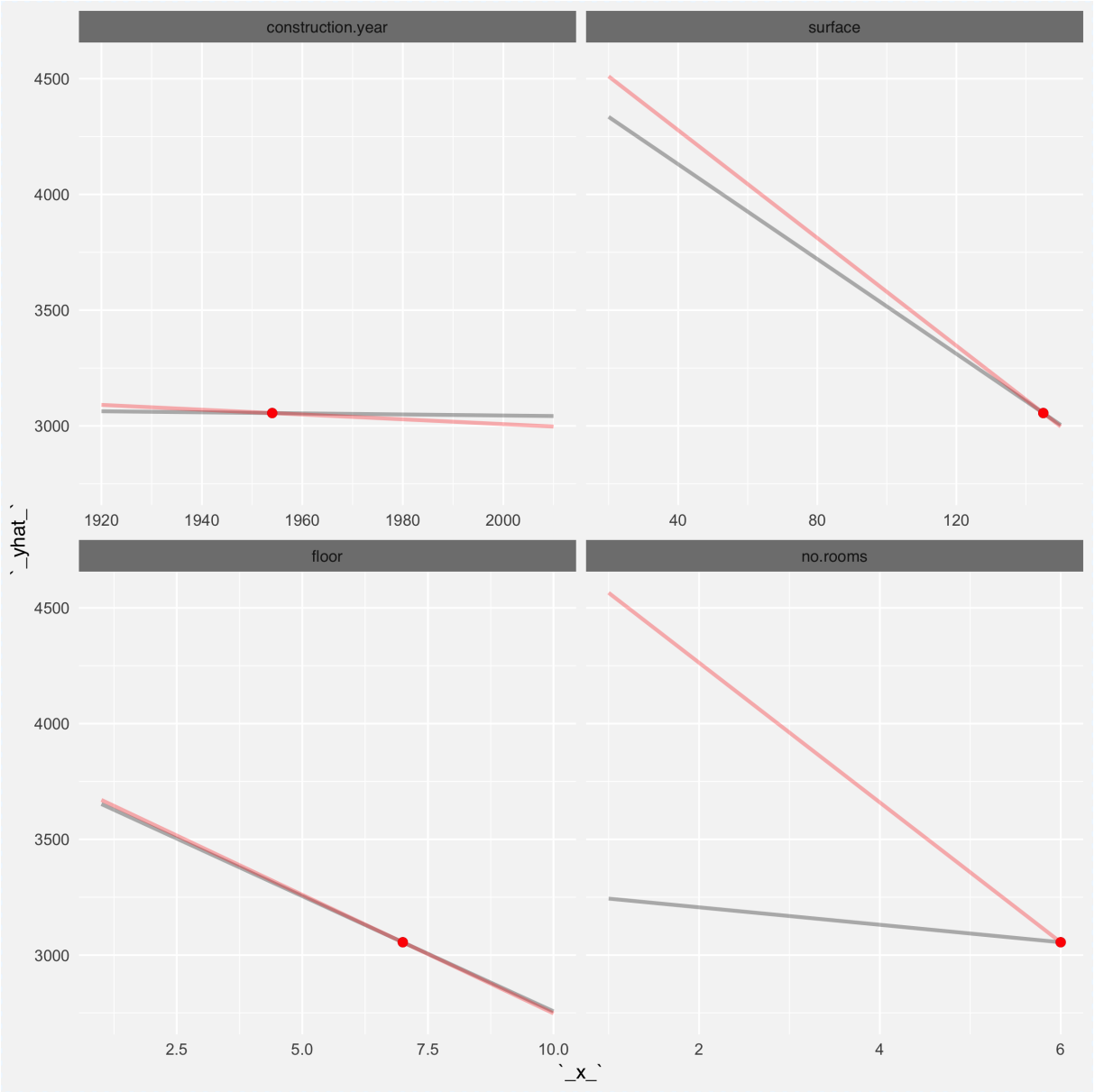


Figure 3.7: Ceteris paribus (black line) and Local Conditional Expectation (red line) profiles for Linear Regression model fitted to apartments data. Red dot represents an apartment chosen for analysis.

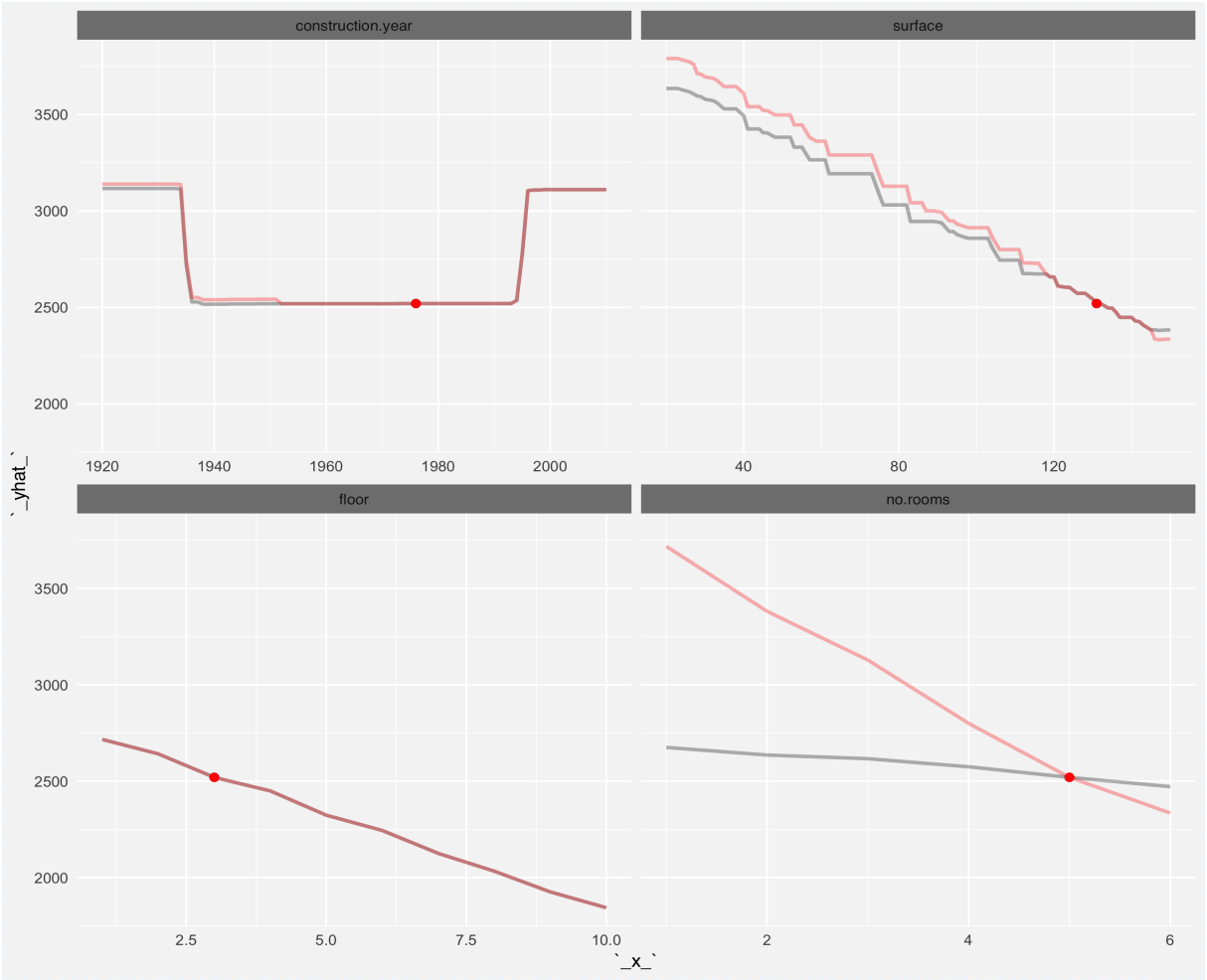


Figure 3.8: Ceteris paribus (black line) and Local Conditional Expectation (red line) profiles for Gradient Boosting Machine model fitted to apartments data. Red dot represents an apartment chosen for analysis.

```
explainer_gbm <- explain(regr_gbm,
                         data = apartmentsTest,
                         y=apartmentsTest$m2.price,
                         predict_function = custom_predict,
                         label="gbm")

cp_gbm <- ceteris_paribus(explainer_regr_gbm,
                          observations = apartmentsTest[1, ])
lce_gbm <- local_conditional_expectations(explainer_regr_gbm,
                                          observations = apartmentsTest[1, ])

selected_variables <- c("surface", "no.rooms", "construction.year", "floor")
lce_gbm_apartments <- plot(cp_gbm, selected_variables = selected_variables) +
   ceteris_paribus_layer(lce_gbm, color = "red",
                         selected_variables = selected_variables)
```

Again, LCE makes an adjustment in no.rooms to match surface variable, similarily as in SVM and linear model (Figure 3.8)

Finally we fit a neural network with *size* and *decay* parameters tuning.

```
set.seed(1234)
regr_task <- makeRegrTask(id = "ap", data = apartments, target = "m2.price")
regr_lrn_nn <- makeLearner("regr.nnet")

ps = makeParamSet(
  makeDiscreteParam("size", values = seq(1, 10, by=1)),
  makeDiscreteParam("decay", values = seq(0, 0.1, by=0.005))
)
ctrl = makeTuneControlGrid(resolution = 2L)
rdesc = makeResampleDesc("CV", iters = 2L)
res = tuneParams("regr.nnet", regr_task, rdesc, par.set = ps, control = ctrl)
print(res)

regr_lrn_nn <- setHyperPars(regr_lrn_nn,
                 par.vals = list(maxit=500, size=10, decay=0.095))
regr_lrn_nn <- makePreprocWrapperCaret(regr_lrn_nn,
                 ppc.scale=TRUE, ppc.center=TRUE)

regr_nn <- train(regr_lrn_nn, regr_task)

custom_predict <- function(object, newdata) {
    pred <- predict(object, newdata=newdata)
    response <- pred$data$response
    return(response)
}

explainer_regr_nn <- explain(regr_nn,
                             data=apartments[, 2:6],
                             y = apartments$m2.price,
                             predict_function = custom_predict,
                             label="nn")

cp_nn <- ceteris_paribus(explainer_regr_nn,
                         observations = apartments[3, ])
lce_nn <- local_conditional_expectations(explainer_regr_nn,
                                         observations = apartments[3, ])

selected_variables <- c("surface", "no.rooms", "construction.year", "floor")
lce_nn_apartments <- plot(cp_nn, selected_variables = selected_variables) +
  ceteris_paribus_layer(lce_nn, color = "red",
                        selected_variables = selected_variables)
```

As expected the differences in CP versus LCE profiles (Figure 3.9) are only present in *no.rooms* and *surface*. This time the model gets even more "confused" with *no.rooms* variable in the ceteris paribus profiles leading to a totally opposite relationship relationship to the desired one. Simple correction that LCE makes solves this problem.

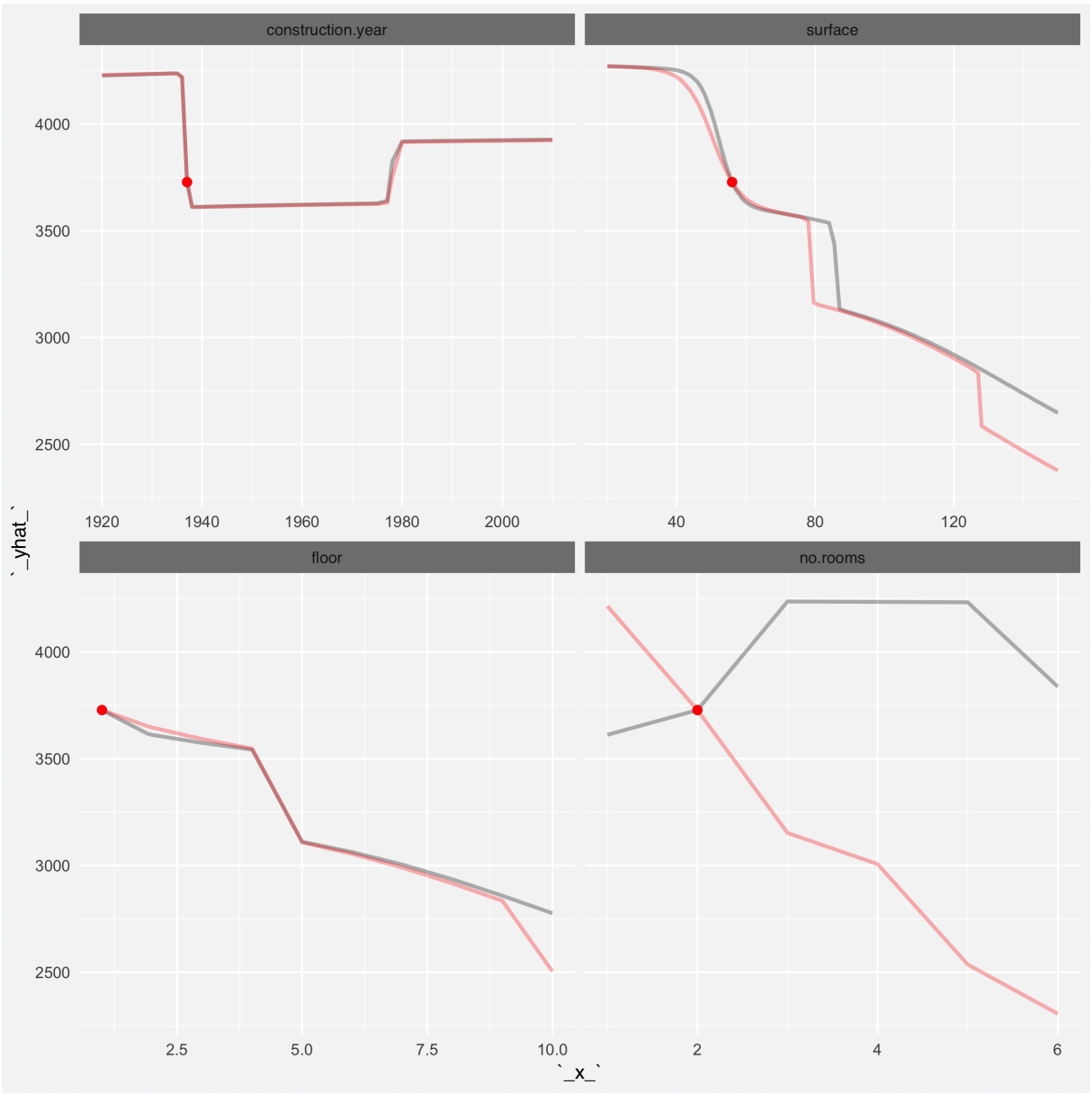


Figure 3.9: Ceteris paribus (black line) and Local Conditional Expectation (red line) profiles for single layer Neural Network model fitted to apartments data. Red dot represents an apartment chosen for analysis.

3.2.2. Classification

Local Conditional Expectation plots can be used along with probabilistic classifiers i.e. classifiers that output a probability distribution over the set of classes rather than the most likely class for a given observation. In this case on the *y-axis* of the LCE plot we will have a probability value rather than output variable value.

For demonstration we will use an artificial *dragons* dataset and try to predict longevity of dragons i.e. classify them as either "short living" or "long living". To do that we discretize life length variable into two categories and fit a Random Forest model.

```
set.seed(1234)
dragons_discrete <- dragons %>%
  mutate(life_length_dis = cut(life_length, 2,
                               labels = c("short_living", "long_living")))
rf_dragons <- randomForest(life_length_dis ~ year_of_birth + height +
                           weight + scars + colour + year_of_discovery +
                           number_of_lost_teeth, dragons_discrete)
```

To use ceterisParibus package in classification setting we need to additionally define functions returning probabilities rather than classes and fit two separate explainer objects.

```
pred1 <- function(m, x)   predict(m, x, type = "prob")[,1]
pred2 <- function(m, x)   predict(m, x, type = "prob")[,2]

rf_dragons_explainer_short <- explain(rf_dragons,
                                      data = dragons_discrete[, 1:7],
                                      predict_function = pred1,
                                      label = "young")
rf_dragons_explainer_long <- explain(rf_dragons,
                                     data = dragons_discrete[, 1:7],
                                     predict_function = pred2,
                                     label = "old")
```

Define a dragon for conducting a what-if scenario and calculate corresponding ceteris paribus and LCE profiles.

```
new_dragon <- data.frame(year_of_birth = 1796,
                         height = 56.45376,
                         weight = 14.24896,
                         scars = 29,
                         colour = factor("red", levels = clevels),
                         year_of_discovery = 1700,
                         number_of_lost_teeth = 25)
cp_short <- ceteris_paribus(rf_dragons_explainer_short,
                            observations = new_dragon)
cp_long <- ceteris_paribus(rf_dragons_explainer_long,
                           observations = new_dragon)
lce_short <- local_conditional_expectations(rf_dragons_explainer_short,
                                            observations = new_dragon)
lce_long <- local_conditional_expectations(rf_dragons_explainer_long,
                                           observations = new_dragon)
sel_vars <- c("height", "weight",
              "number_of_lost_teeth", "year_of_discovery")
lce_dragons_rf <- plot(cp_short, selected_variables = sel_vars) +
  ceteris_paribus_layer(lce_short, selected_variables = sel_vars,
  color = "red")
```

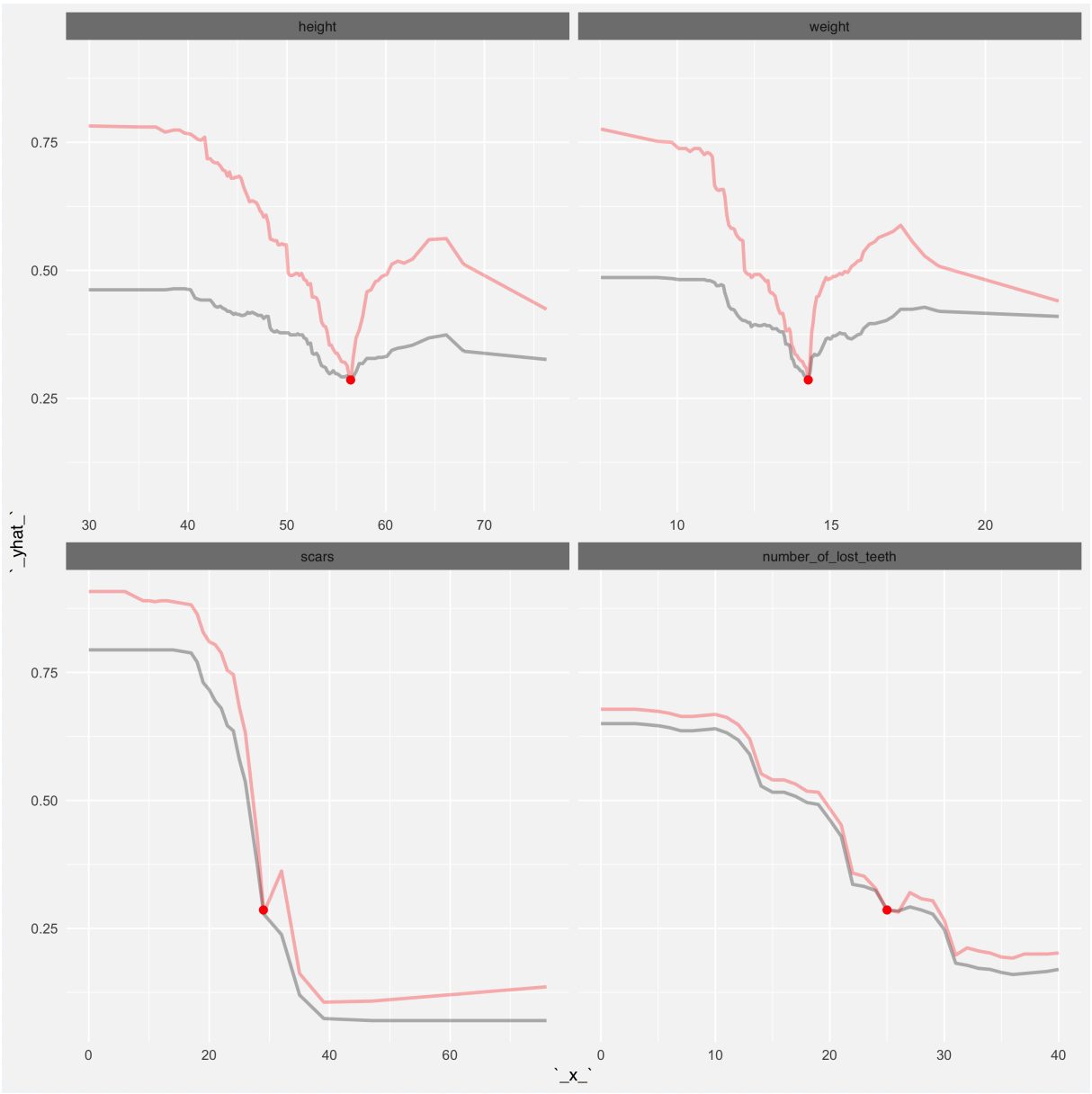


Figure 3.10: Ceteris paribus (black line) and Local Conditional Expectation (red line) profiles for Random Forest classification model fitted to dragons data. Red dot represents a dragon chosen for analysis.

We can do the same for SVM model:

```
svm_dragons <- svm(life_length_dis ~ year_of_birth + height +
                           weight + scars + colour + year_of_discovery +
                           number_of_lost_teeth, dragons_discrete,
                           probability = TRUE)
pred1 <- function(m, x)
  attr(predict(m, x, probability = TRUE), "probabilities")[,1]
pred2 <- function(m, x)
  attr(predict(m, x, probability = TRUE), "probabilities")[,2]

svm_dragons_explainer_short <- explain(svm_dragons,
                                       data = dragons_discrete[, 1:7],
                                       predict_function = pred1,
                                       label = "short_living")
svm_dragons_explainer_long <- explain(svm_dragons,
                                      data = dragons_discrete[, 1:7],
                                      predict_function = pred2,
                                      label = "long_living")
cp_short <- ceteris_paribus(rf_dragons_explainer_short,
                            observations = new_dragon)
cp_long <- ceteris_paribus(rf_dragons_explainer_long,
                           observations = new_dragon)

lce_short <- local_conditional_expectations(rf_dragons_explainer_short,
                                            observations = new_dragon)
lce_long <- local_conditional_expectations(rf_dragons_explainer_long,
                                           observations = new_dragon)
lce_dragons_svm <- plot(cp_short, selected_variables = sel_vars) +
  ceteris_paribus_layer(lce_short, selected_variables = sel_vars,
  color = "red")
```

In this example we have two highly correlated variables *height* and *weight*. When conducting a what-if scanario for a particular observation it is very natural to adjust e.g. *weight* when changing *height* value. Applying LCE to Random Forest (Figure 3.10) and SVM (Figure 3.11) results in differences between ceteris Paribus and LCE scenarios. Similarly to the apartments regression example the differences for both variables are more visible in Random Forest (thanks to equal treating of correlated variables), while in SVM the adjustment is more evident in the *weight* variable. Nevertheles, in both cases LCE leads to a more realistic "what-if" scenario plot.

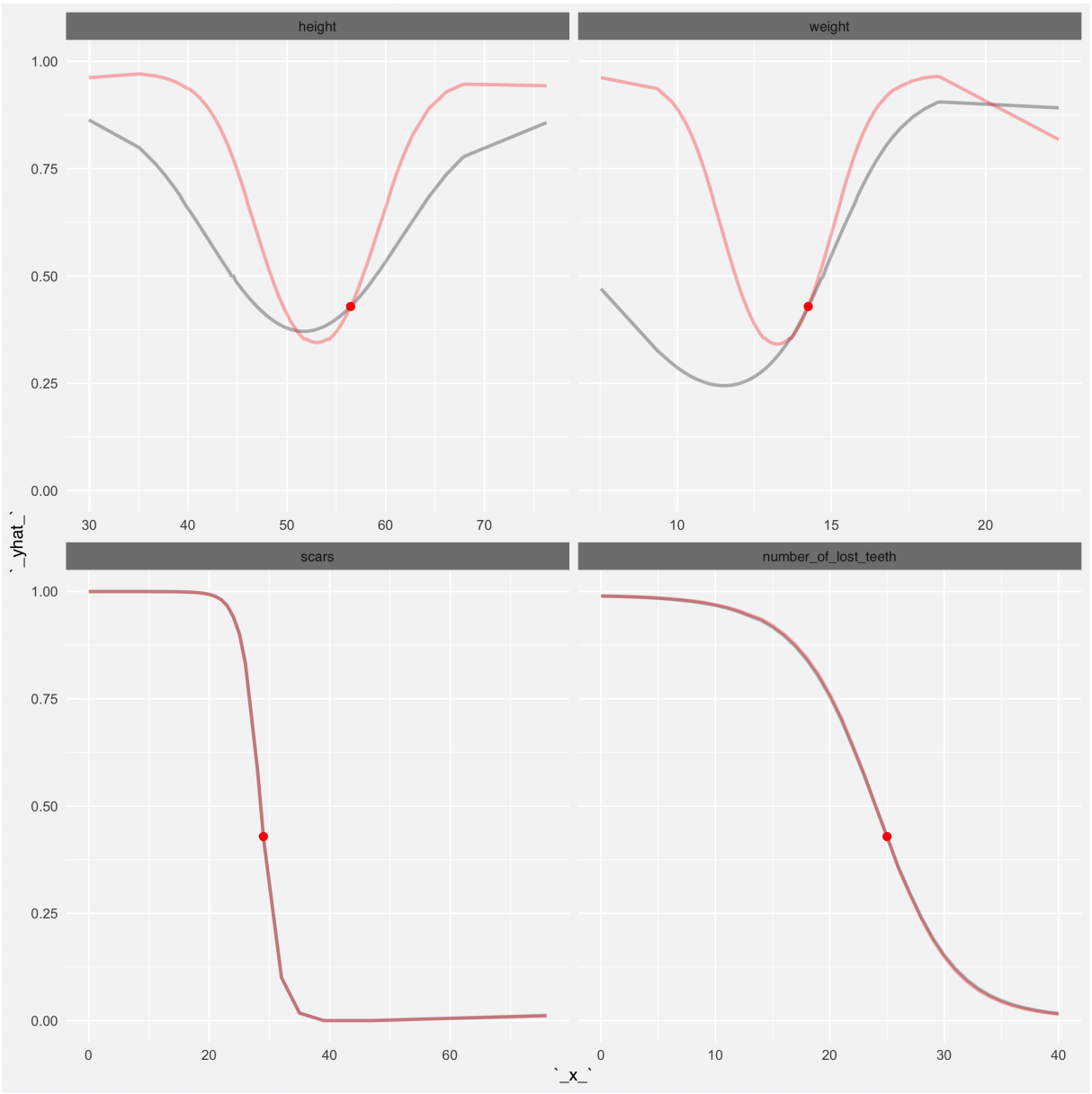


Figure 3.11: Ceteris paribus (black line) and Local Conditional Expectation (red line) profiles for SVM classification model fitted to dragons data. Red dot represents a dragon chosen for analysis.

Chapter 4

Summary

In this thesis we described different model-agnostic methods of visualizing feature dependence in black box supervised learning models and provided real use cases in R language. Methods were split into two groups:

- global - visualising the general relationship between a feature variable and model response
- local - visualising relationship between a feature variable and model response when starting from a certain observation (also called what-if scenarios)

Global methods can be summarised as:

- Partial Dependence Plots - averaging out all features but the plotted one by the marginal distribution. They assume independece of feature variables and should be used as a summary after closer investigation e.g. using ICE.
- Individual Conditional Expectations - supplementing PDPs by *disintegrating* them into individual curves (per observation) and showcasing potential interactions between features.
- Accumulated Local Effects plots - sophisticated alternative to PDPs. With the use of conditional distribution (instead of marginal) and averaging the accumulated differences they are able to isolate the pure effect of a feature on the prediction (even when correlations between features are present).

Local methods can be summarized as:

- Ceteris Paribus - for a given observation, showing how would the model response change if change the value of one feature keeping all other features unchanged
- Local Conditional Expectations - proposed method serving the same purpose as Ceteris Paribus but also addressing the fact that when conducting a "what-if" analysis highly correlated features should not be kept unchanged but they should be adjusted to the value of analyzed variable. These adjustments are made according to simple pairwise linear relationships. In case of no correlation this procedure leads to the same results as Ceteris Paribus profiles but produces different plots when the correlations are present.

As a part of this master thesis, we include implementation of Local Conditional Expectations in both ceterisParibus and ceterisParibus2 R packages, being a part of DALEX and DALEX2 frameworks for explainable machine learning to be used by the data science community.

Bibliography

[Apley, 2018] Apley, D. (2018). *ALEPlot: Accumulated Local Effects (ALE) Plots and Partial Dependence (PD) Plots*. R package version 1.1.

[Apley, 2016] Apley, D. W. (2016). Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models. *arXiv preprint arXiv:1612.08468*.

[Biecek, 2018a] Biecek, P. (2018a). *ceterisParibus: Ceteris Paribus Profiles*. R package version 0.3.0.

[Biecek, 2018b] Biecek, P. (2018b). Dalex: explainers for complex predictive models. *arXiv preprint arXiv:1806.08915*.

[Bischl et al., 2016] Bischl, B., Lang, M., Kotthoff, L., Schiffner, J., Richter, J., Studerus, E., Casalicchio, G., and Jones, Z. M. (2016). mlr: Machine learning in r. *Journal of Machine Learning Research*, 17(170):1–5.

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

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

[Fisher et al., 2018] 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*.

[Fisher, 1936] Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188.

[Friedman et al., 2001] Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The Elements of Statistical Learning*, volume 1. Springer series in statistics New York, NY, USA:.

[Goldstein et al., 2015] 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.

[Goodman and Flaxman, 2016] Goodman, B. and Flaxman, S. (2016). European union regulations on algorithmic decision-making and a" right to explanation". *arXiv preprint arXiv:1606.08813*.

[Greenwell, 2017] Greenwell, B. M. (2017). pdp: An r package for constructing partial dependence plots. *R Journal*, 9(1).

[R Core Team, 2018] R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

[Schwab, 2017] Schwab, K. (2017). *The fourth industrial revolution.* World Economic Forum.

[Silver et al., 2017] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *Nature*, 550(7676):354.

[Wickham, 2016] Wickham, H. (2016). *ggplot2: elegant graphics for data analysis.* Springer.