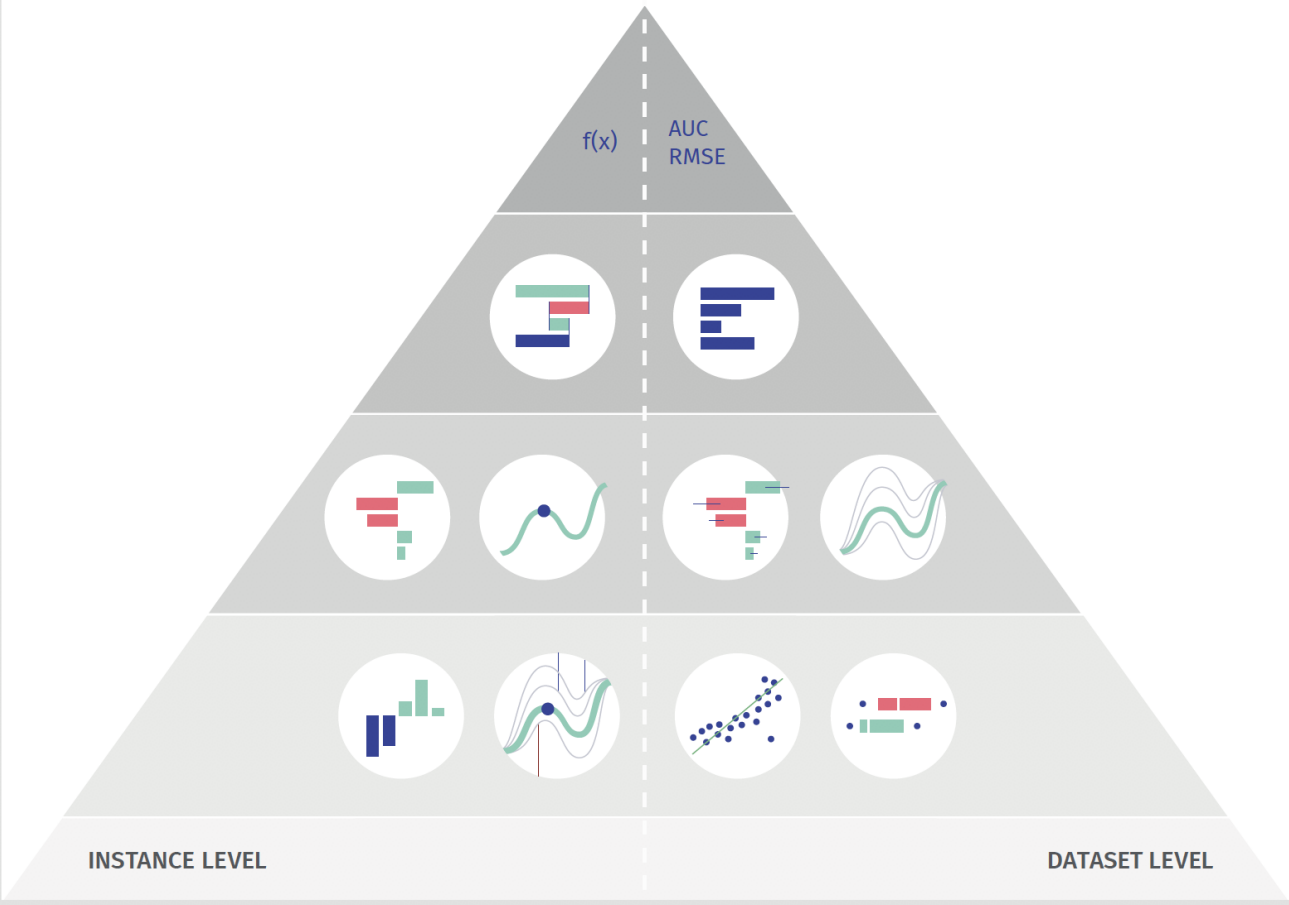
# Explanatory Model Analysis

## Explore, Explain and Examine Predictive Models

Przemysław Biecek
Tomasz Burzykowski



# 1 Introduction

## 1.1 Notes to readers

A note to readers: this text is a work in progress.

We've released this initial version to get more feedback. Feedback can be given at the GitHub repo https://github.com/pbiecek/ema/issues. We are primarily interested in the organization and consistency of the content, but any comments will be welcomed.

We'd like to thank everyone that contributed feedback, found typos, or ignited discussions while the book was being written, including GitHub contributors: agosiewska, Rees Morrison, kasiapekala, hbaniecki, AsiaHenzel, kozaka93, agilebean.

## 1.2 The aim of the book

Predictive models are used to guess (statisticians would say: predict) values of a variable of interest based on other variables. As an example, consider prediction of sales based on historical data, prediction of risk of heart disease based on patient characteristics, or prediction of political attitudes based on Facebook comments.

Predictive models have been constructed through the entire human history. Ancient Egyptians, for instance, used observations of the rising of Sirius to predict flooding of the Nile. A more rigorous approach to model construction may be attributed to the method of least squares, published more than two centuries ago by Legendre in 1805 and by Gauss in 1809. With time, the number of applications in economy, medicine, biology, and agriculture has grown. The term *regression* was coined by Francis Galton in 1886. Initially, it was referring to biological applications, while today it is used for various models that allow prediction of continuous variables. Prediction of nominal variables is called *classification*, and its beginning may be attributed to works of Ronald Fisher in 1936.

During the last century, many statistical models that can be used for predictive purposes have been developed. These include linear models, generalized linear models, regression and classification trees, rule-based models, and many others. Developments in mathematical

foundations of predictive models were boosted by increasing computational power of personal computers and availability of large datasets in the era of ,,big data" that we have entered.

With the increasing demand for predictive models, model features such as flexibility, ability to perform internally variable selection (feature engineering), and high precision of predictions are of interest. To obtain robust models, ensembles of models are used. Techniques like bagging, boosting, or model stacking combine hundreds or thousands of small models into a one super-model. Large deep neural models have over a billion parameters.

There is a cost of this progress. Complex models may seem to operate like ,,black boxes''. It may be difficult, or even impossible, to understand how thousands of coefficients affect the model prediction. At the same time, complex models may not work as well as we would like them to. An overview of real problems with massive-scale black-box models may be found in an excellent book of Cathy O'Neil (O'Neil 2016) or in her TED Talk ,,*The era of blind faith in big data must end*". There is a growing number of examples of predictive models with performance that deteriorated over time or became biased in some sense. For instance, IBM's Watson for Oncology was criticized by oncologists for delivering unsafe and inaccurate recommendations (Ross and Swetliz 2018). Amazon's system for CV screening was found to be biased against women (Dastin 2018). The COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) algorithm for predicting recidivism, developed by Northpointe (now Equivant), is accused to be biased against blacks (Larson et al. 2016). Algorithms beyond Apple Credit Card are accused to be gender-biased (Duffy 2019). Some tools for sentiment analysis are suspected to be age-biased (Diaz et al. 2018). These are examples of models and algorithms that led to serious violations of fairness and ethical principles. An example of situation when data drift led to deterioration in model performance is the Google Flu model, which gave worse predictions after two years than at baseline (Salzberg 2014), (Lazer et al. 2014).

A reaction to some of these examples and problems are new regulations, like the General Data Protection Regulation (Gdpr 2018). Also, new civic rights are being formulated (Goodman and Flaxman 2016), (Casey, Farhangi, and Vogl 2018), (Ruiz 2018). A noteworthy example is the *,,Right to Explanation"*, i.e., the right to be provided an explanation for an output of an automated algorithm (Goodman and Flaxman 2016). To exercise the right, we need new methods for verification, exploration, and explanation of predictive models.

Figure 1.1 shows how the increase in the model complexity affects the relative importance of domain understanding vs. modeling vs. validation. Simplest models are usually built on top of a good understanding of the domain. Domain knowledge helps to create and select most important variables that can be transformed into predictive scores. Machine learning exploits

the tradeoff between availability of data and domain knowledge. Flexible models can use massive data to learn good features and filter out bad ones. The effort is shifted from a deep understanding of the domain towards computationally heavy training of models. The validation part is of an increased importance because it creates a feedback loop with the modeling. Results from model validation lead to next decisions related to model training. This is different than in case of statistical hypothesis testing. Statistical hypotheses shall be stated in advance of data analysis and obtained p-values shall not interfere in the way how data or models were prepared.

What will be next? The increasing automation in the EDA (Exploratory Data Analysis) and modeling part of the process shift the focus towards the validation of models. The purpose of validation is not only to measure how good is the model but also what other risks are associated with models. Risks like concept drift, gender, age or race bias. This book is about new methods that can be used for validation and justification.

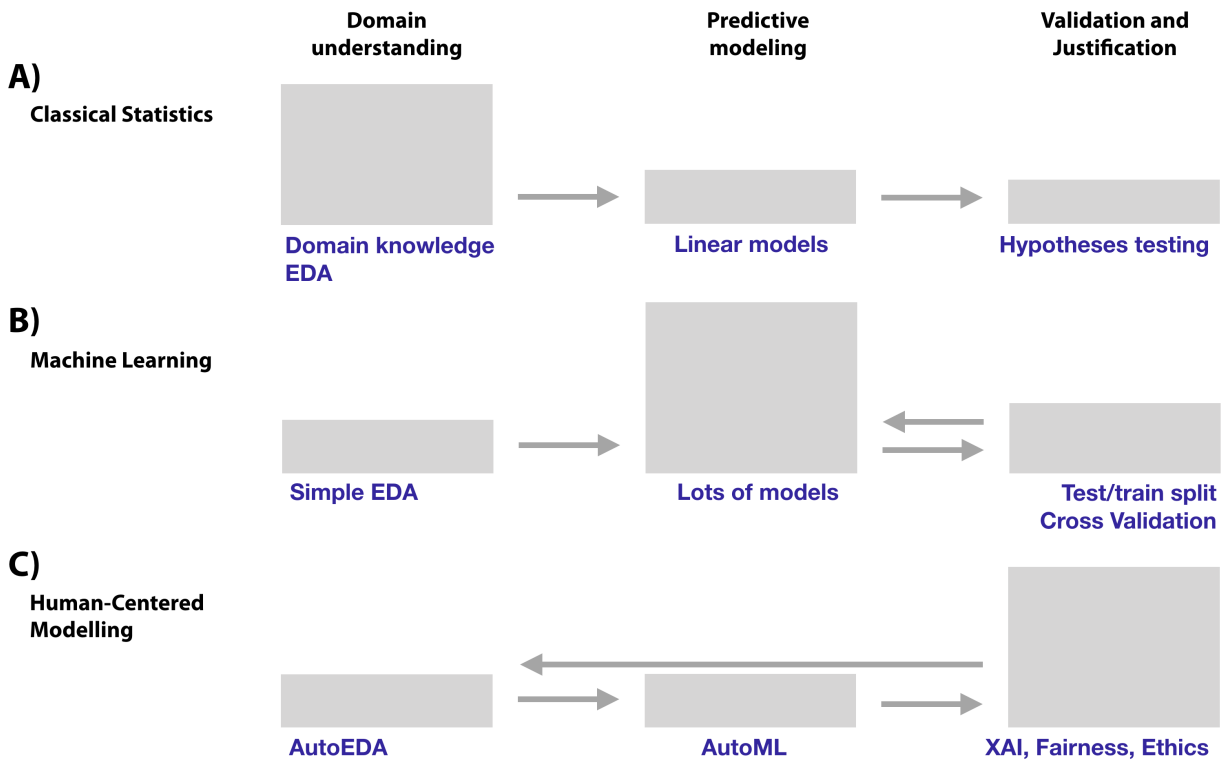



Figure 1.1: Shift in the relative importance and effort put in different phases of the data-driven modeling. (A) Statistical modeling is often based on deep understanding of the domain. Manual data exploration, consultations with domain experts, variable transformations lead to good models. Structures of models are often based on (generalized) linear models. Model verification is done through hypothesis testing. (B) Machine learning modeling is often based on elastic models fitted to large volumes of data. Domain exploration is often shallow while the focus is based on predictive performance. Lots of attention is put in cross validation and other strategies that deal with overfitting. (C) What will be next? Human-centered modeling?

Better tools for auto EDA and auto ML will shift focus into the part related with validation against the domain knowledge like fairness, bias or new techniques for data exploration. Arrows show feedback loops in the modeling process. The feedback loop is even larger now, as the results from model validation are helping also in the domain understanding.

Out of this we can conclude that, today, the true bottleneck in predictive modelling is not the lack of data, nor the lack of computational power, nor inadequate algorithms, nor the lack of flexible models. It is the lack of tools for model validation, model exploration, and explanation of model decisions. Thus, in this book, we present a collection of methods that may be used for this purpose. As development of such methods is a very active area of research and new methods become available almost on a continuous basis, we do not aim at being exhaustive. Rather, we present the mind-set, key problems, and several examples of methods that can be used in model exploration.

## 1.3 A bit of philosophy: three laws of model explanation

In 1942, Isaac Asimov forumlated 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.

Today's robots, like cleaning robots, robotic pets, or autonomous cars are far from being conscious enough to fall under Asimov's ethics. However, we are more and more surrounded by complex predictive models and algorithms used for decision making. Artificial Intelligence models are used in health care, politics, education, justice, and many other areas. The models and algorithms have a far larger influence on our lives than physical robots. Yet, applications of such models are left unregulated despite examples of their potential harmfulness. See *Weapons of Math Destruction* by Cathy O'Neil (O'Neil 2016) for an excellent overview of selected problems.

It's clear that we need to control the models and algorithms that may affect us. Thus, Asimov's laws are referred to in the context of the discussion around Ethics of Artificial Intelligence. Initiatives to formulate principles for AI development have been undertaken, for instance, in the UK [Olhede & Wolfe, Significance 2018, 15: 6-7]. Following Asimov's approach, we propose three requirements that any predictive model should fulfill:

- **Prediction's validation**. For every prediction of a model, one should be able to verify how strong is the evidence that confirms the prediction.
- **Prediction's justification**. For every prediction of a model, one should be able to understand which variables affect the prediction and to what extent.
- **Prediction's speculation**. For every prediction of a model, one should be able to understand how the model prediction would change if input variables changed.

We see two ways to comply with these requirements. One is to use only models that fulfill these conditions by design. There are so called interpretable by design models like linear models, rule based models or classification trees with small number of parameters (Molnar 2019). However, the price for transparency may be a reduction in performance. Another way is to use tools that allow, perhaps by using approximations or simplifications, to ,,explain" predictions for any model. In our book, we will focus on the latter approach.

## 1.4 The structure of this book

This book is split in two major parts. In the part *Instance-level explainers*, we present techniques for exploration and explanation of model predictions for a single observation. On the other hand, in the part *Dataset-level explainers*, we present techniques for exploration and explanation of a model for an entire dataset.

Before embarking on the description of the methods, in Chapter 2, we provide a short introduction to the process of data exploration and model assembly along with notation and definition of key concepts that are used in consecutive chapters. In Chapters 3 and 4, we provide a short description of R and Python tools and packages that are necessary to replicate the results presented in this book. In Chapter 5, we describe two datasets that are used throughout the book to illustrate the presented methods and tools.

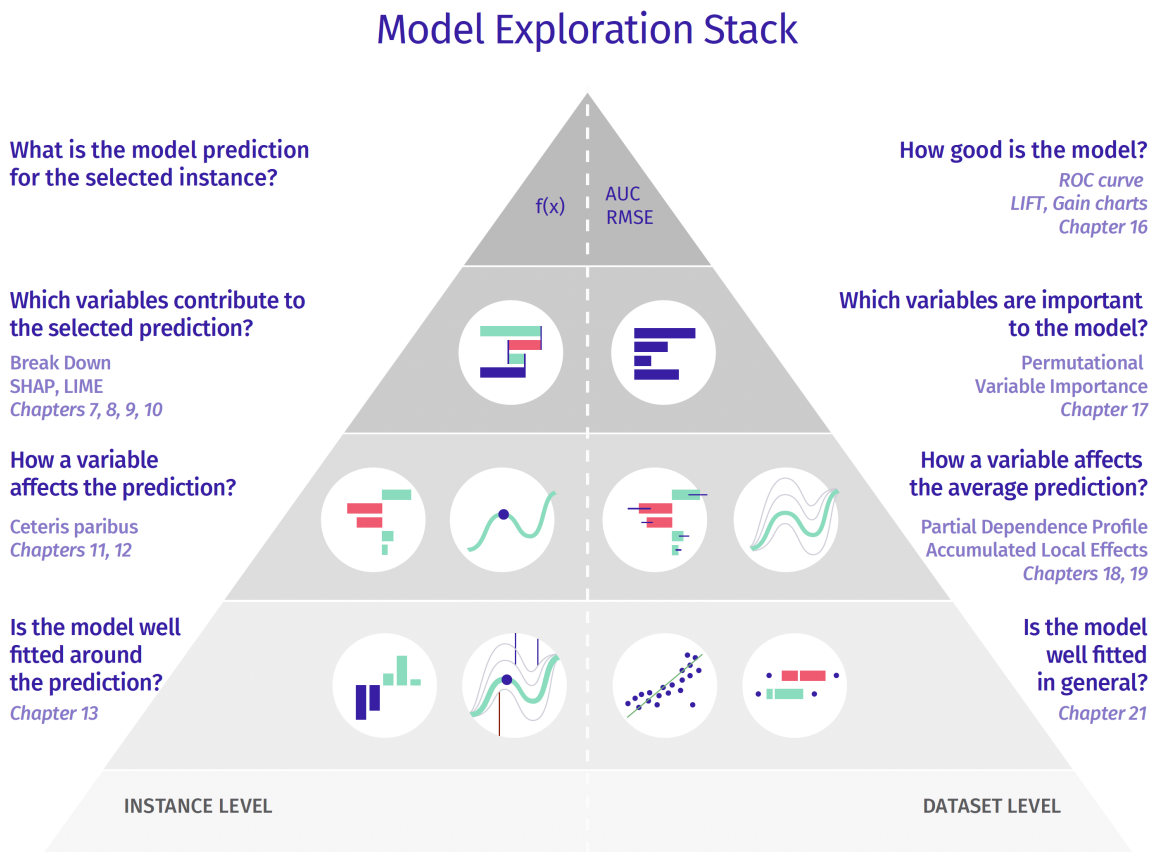



Figure 1.2: Stack with model exploration methods presented in this book. Left side is focused on instance-level explanation while the right side is focused on dataset-level explanation. Consecutive layers of the stack are linked with a deeper level of model exploration. These layers are linked with law's of model exploration introduced in Section 1.3

Rest of the book is structured in Figure 1.2.

The **Instance-level** part of the book consists of Chapters 7-14. Chapters 7-9 present methods to decompose model predictions into variable contributions. In particular, Chapter 7 introduces Break-down (BD) plots for models with additive effects. On the other hand, Chapter 8 presents a method that allows for interactions. Finally, Chapter 9 describes SHAP (Lundberg and Lee 2017) an alternative method for decomposing model predictions that is closely linked with Shapley values (Shapley 1953) developed originally for cooperative games. Chapter 10 presents a different approach to explanation of single-instance predictions. It is based on a local approximation of a black-box model by a simpler, glass-box one. In this chapter, we discuss the Local Interpretable Model-Agnostic Explanations (LIME) method (Ribeiro, Singh, and Guestrin 2016). These chapters corresponds to the second layer of the stack in Figure 1.2.

In Chapters 11-13 we present methods based on Ceteris-paribus (CP) profiles. The profiles show the change of model-based predictions induced by a change of a single variable. These profiles are introduced in Chapter 11 while Chapter 12 presents a CP-profile-based measure that summarizes the impact of a selected variable on model's predictions. This measure can be used to determine the order of variables in model exploration. It is particularly important for models with large numbers of explanatory variables. Chapter 13 is focused on model diagnostic. It describes local-fidelity plots that are useful to investigate the sources of a poor prediction for a particular single observation. The final chapter of the first part, Chapter 14 compares various instance-level explainers.

The **Dataset-level explainers** part of the book consists of Chapters 15-20. These chapters present methods in the same order as appeared in the Model Exploration Stack in Figure 1.2. Chapter 16 shows selected measures for model benchmarking along with performance measures for classification and regression models. On top of these measures, the Chapter 17 presented an algorithm for assessment of importance of variables based on selected performance measure. This method is model agnostic and can be used for cross models comparisons. Next layer of the Model Exploration Stack is presented in Chapters 18 and 19. Here we introduce Partial Dependency and Accumulated Dependency methods for univariate exploration of variable effects. This part of the book is closed with the Chapter 20 that summarises diagnostic techniques for model residuals.

To make the exploration of the book easier, in each chapter we introduce a single method and each chapter has the same structure:

- Section *Introduction* explains the goal of and the general idea behind the method.
- Section *Method* shows mathematical or computational details related to the method. This subsection can be skipped if you are not interested in the details.
- Section *Example* shows an exemplary application of the method with discussion of results.
- Section *Pros and cons* summarizes the advantages and disadvantages of the method. It also provides some guidance regarding when to use the method.
- Section *Code snippets* shows the implementation of the method in R and Python. This subsection can be skipped if you are not interested in the implementation.

## 1.5 Terminology

It is worth noting that, when it comes to predictive models, the same concepts have often been given different names in statistics and in machine learning. For instance, in the statistical-modelling literature, one refers to ,,explanatory variables," with ,,independent

variables," ,,predictors," or ,,covariates" as often-used equivalents. Explanatory variables are used in the model as means to explain (predict) the ,,dependent variable," also called ,,predicted" variable or ,,response." In machine-learning terminology, ,,input variables" or ,,features" are used to predict the ,,output" or ,,target" variable. In statistical modelling, models are fit to the data that contain ,,observations'', whereas in the machine-learning world a dataset may contain ,,instances" or ,,cases''. When we talk about values that define a single instance of a model in statistical modelling we refer to model ,,coefficients" while in machine-learning it is more common to use phrase model ,,parameters''. In statistics it is common to say that model coefficients are ,,estimated" while in machine learning it is more common to say that parameters are ,,trained" or are obtained in the process of ,,model training".

To the extent possible, in our book we try to consistently use the statistical-modelling terminology. However, the reader may find references to a ,,feature" here and there. Somewhat inconsistently, we also introduce the term ,,instance-level" explanation. Instance-level explanation methods are designed to extract information about the behavior of the model related to a specific observation (or instance). On the other hand, ,,dataset-level" explanation techniques allow obtaining information about the behavior of the model for an entire dataset.

We consider models for dependent variables that can be continuous or nominal/categorical. The values of a continuous variable can be represented by numbers with an ordering that makes some sense (zip codes or phone numbers are not considered as continuous variables while age, number of children are). A continuous variable does not have to be continuous in the mathematical sense; counts (number of floors, steps, etc.) will be treated as continuous variables as well. A nominal/categorical variable can assume only a finite set of values that are not numbers in the mathematical sense, i.e. it makes no sense to subtract or divide these values.

In this book we focus on ,,black-box" approach. We discuss this approach in a bit more detail in the next section.

## 1.6 Glass-box models vs. black-box models

Black-box models are models with a complex structure that is hard to understand by humans. Usually this refers to a large number of model coefficients or complex mathematical transformations. As people vary in their capacity to understand complex models, there is no strict threshold for the number of coefficients that makes a model a black-box. In practice, for most people this threshold is probably closer to 10 than to 100.

A ,,glass-box” (sometimes called white-box or transparent-box) model, which is opposite to a ,,black-box” one, is a model that is easy to understand (though maybe not by every person). It has a simple structure and a limited number of coefficients.

The most common classes of glass-box models are decision or regression trees, as an example in Figure 1.3, rules, or models with an explicit compact structure, like the following model for obesity based on the BMI index.

$$BMI = \frac{mass_{kg}}{height_{m^2}}.$$

In the model, two explanatory variables are used, mass in kilograms and height in meters. Based on them a BMI index is derived that commonly used for classification into *Underweight* (BMI < 18), *Normal* (18 < BMI < 25) or *Overweight* (BMI > 25) categories. Having the model in a compact form it is easy to understand how changes in one variable affect the model output.

The structure of a glass-box model is, in general, easy to understand. It may be difficult to collect the necessary data, build the model, fit it to the data, or perform model validation, but once the model has been developed its interpretation and mode of working is straightforward.

Why is it important to understand the model structure? There are several important advantages. If the model structure is clear, we can easily see which variables are included in the model and which are not. Hence, for instance, we may be able to, question the model when a particular explanatory variable was excluded from it. Also, in the case of a model with a clear structure and a limited number of coefficients, we can easily link changes in model predictions with changes in particular explanatory variables. This, in turn, may allow us to challenge the model against domain knowledge if, for instance, the effect of a particular variable on predictions is inconsistent with previously established results. Note that linking changes in model predictions with changes in particular explanatory variables may be difficult when there are many variables and/or coefficients in the model. For instance, a classification tree with hundreds of nodes is difficult to understand, as is a linear regression model with hundreds of coefficients.

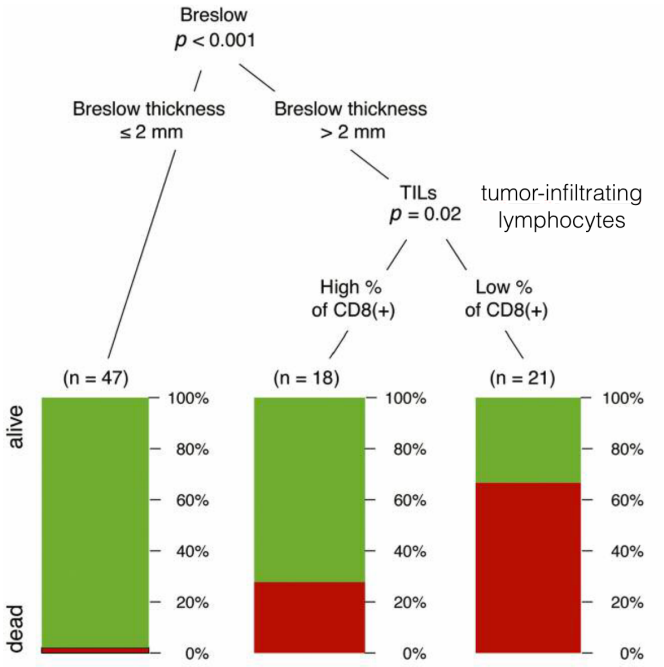



Figure 1.3: Example classification tree model for melanoma risk patients based on (Donizy et al. 2016). The model is based on two explanatory variables, Breslow thickness and Tumor infiltration lymphocytes. These two variables lead to three groups of paritents with different odds of survival.

Note that some glass-box models, like the decision tree model presented in Figure 1.3 by design satisfies explainability laws introduced in Section 1.3. For *Prediction's validation* we see in each node how many patients fall in a given category. For *Prediction's justification* we see which variables are used in every decision path. For *Prediction's speculation* we can trace how changes in particular variables will affect the model prediction. We can, of course, argue if the model is good or not, but obviously the model structure is transparent.

Comprehending the performance of a black-box models presents more challenges. The structure of a complex model, such as a neural-network model, may be far from transparent. Consequently, we may not understand which features influence the model decisions and by how much. Consequently, it may be difficult to decide whether the model is consistent with our domain knowledge. In our book we present tools that can help in extracting the information necessary for the evaluation of complex models.

## 1.7 Model-agnostic vs. model-specific approach

Interest in model interpretability is as old as the statistical modeling itself. Some classes of models have been developed for a long period of time or have attracted intensive research. Consequently, those classes of models are equipped with excellent tools for model exploration or visualisation. For example:

- There are many tools for diagnostics and evaluation of linear models, see for example (Galecki and Burzykowski 2013) or (Faraway 2002). Model assumptions are formally defined (normality, linear structure, homogenous variance) and can be checked by using normality tests or plots (normal qq-plot), diagnostic plots, tests for model structure, tools for identification of outliers, etc.
- For many more advanced models with an additive structure, like the proportional hazards model, many tools can be used for checking model assumptions, see for example (Harrell Jr 2018) or (Sheather 2009).
- Random-forest models are equipped with the out-of-bag method of evaluating performance and several tools for measuring variable importance (Breiman et al. 2018). Methods have been developed to extract information from the model structure about possible interactions (Paluszynska and Biecek 2017). Similar tools have been developed for other ensembles of trees, like boosting models (xgboost, gbm). See (Foster 2017) or (Karbowiak and Biecek 2019).
- Neural networks enjoy a large collection of dedicated model-explanation tools that use, for instance, the layer-wise relevance propagation technique (Bach et al. 2015), or saliency maps technique (Simonyan, Vedaldi, and Zisserman 2013), or a mixed approach. Broader summary is presented in (Samek, Wiegand, and Müller 2017) and (Alber et al. 2018).
- BERT family of models leads to high-performance models in Natural Language Processing. The exBERT method (Hoover, Strobelt, and Gehrmann 2019) is designed to visualize the activation of attention heads in this model.

Of course, the list of model classes with dedicated collections of model-explanation and/or diagnostics methods is much longer. This variety of model-specific approaches does lead to issues, though. For instance, one cannot easily compare explanations for two models with different structures. Also, every time a new architecture or a new ensemble of models is proposed, one needs to look for new methods of model exploration. Finally, for brand-new models no tools for model explanation or diagnostics may be immediately available.

For these reasons, in our book we focus on model-agnostic techniques. In particular, we prefer not to assume anything about the model structure, as we may be dealing with a black-box model with an unspecified structure. Often we do not have access to model parameters

just to a specified Application Programming Interface (API) that allows for querying remote models (for example in Microsoft Cognitive Services (Azure 2019)). In that case, the only operation that we may be able to perform is the evaluation of a model for a specified data.

However, while we do not assume anything about the structure of the model, we will assume that the model operates on $p$-dimensional vector of variables/features and, for a single observation, it returns a single value (score/probability) which is a real number. This assumption holds for a broad range of models for data such as tabular data, images, text data, videos, etc. It may not be suitable for, e.g., models with memory like sequence-to-sequence models (Sutskever, Vinyals, and Le 2014) or Long Short Term Memory models (Hochreiter and Schmidhuber 1997) in which the model output depends also on sequence of previous inputs or generative models that output text of images.

## 1.8 What is in this book and what is not

The area of model exploration and explainability is quickly growing and is present in many different flavors. Instead of showing every existing method (is it really possible?) we rather selected a subset of consistent tools that are a good starting set for model exploration. Our focus was on the impact of the model exploration and explanation tools rather than on selected methods. We believe that once we become aware of potential beyond visual model exploration, once we will learn a language of model explanation, we will improve our process of data modeling.

Taking this goal into account **in this book, we do show**

- how to determine features that affect model prediction for a single observation. In particular, we present the theory and examples of methods that can be used to explain prediction like Break Down plots, Ceteris Paribus profiles, local-model approximations, or Shapley values;
- techniques to examine fully-trained machine-learning models as a whole. In particular, we review the theory and examples of methods that can be used to explain model performance globally, like partial-dependence plots, variable-importance plots, and others;
- charts that can be used to present key information in a quick way;
- tools and methods for model comparison;
- code snippets for R and Python that explain how to use the described methods.

On the other hand, **in this book, we do not focus on**

- any specific model. The techniques presented are model agnostic and do not make any assumptions related to the model structure;
- data exploration. There are very good books on this topic, like *R for Data Science* by Garrett Grolemund and Hadley Wickham (Grolemund and Wickham 2019) or *Python for Data Analysis* (Wes 2012) by Wes McKinney or an excellent *Exploratory Data Analysis* by John Tukey (Tukey 1977);
- the process of model building. There are also very good books on this topic, see *Modern Applied Statistics with S* by W. Venables and B. Ripley (Venables and Ripley 2002), *An Introduction to Statistical Learning* by Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani (James et al. 2014) or *Computer Age Statistical Inference* by Bradley Efron and Trevor Hastie (Efron and Hastie 2016);
- any particular tools for model building. These are discussed, for instance, in *Applied Predictive Modeling* by Max Kuhn and Kjell Johnson (Kuhn and Johnson 2013).

## 1.9 Acknowledgements

This book has been prepared using the `bookdown` package (Xie 2018), created thanks to the amazing work of Yihui Xie. Figures and tables are created in R language for statistical computing (R Core Team 2018) with numerous libraries that support predictive modeling. Just to name few frequently used in this book `randomForest` (Liaw and Wiener 2002), `ranger` (Wright and Ziegler 2017), `rms` (Harrell Jr 2018), `gbm` (Ridgeway 2017) or `caret` (Jed Wing et al. 2016). For statistical graphics we used the `ggplot2` library (Wickham 2009) and for model governance we used `archivist` (Biecek and Kosinski 2017).

Przemek's work on interpretability started during research trips within the RENOIR (H2020 grant no. 691152) secondments to Nanyang Technological University (Singapour) and Davis University of California (USA). So he would like to thank Prof. Janusz Holyst for the chance to take part in this project. Przemek would also like to thank Prof. Chris Drake for her hospitality. This book would have never been created without perfect conditions that Przemek found at Chris's house in Woodland.

## References

Alber, Maximilian, Sebastian Lapuschkin, Philipp Seegerer, Miriam Hägele, Kristof T. Schütt, Grégoire Montavon, Wojciech Samek, Klaus-Robert Müller, Sven Dähne, and Pieter-Jan Kindermans. 2018. "INNvestigate Neural Networks!"

Azure. 2019. “Microsoft Cognitive Services.” https://azure.microsoft.com/en-en/services/cognitive-services/.

Bach, Sebastian, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. “On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation.” Edited by Oscar Deniz Suarez. *Plos One* 10 (7): e0130140. https://doi.org/10.1371/journal.pone.0130140.

Biecek, Przemyslaw, and Marcin Kosinski. 2017. “archivist: An R Package for Managing, Recording and Restoring Data Analysis Results.” *Journal of Statistical Software* 82 (11): 1–28. https://doi.org/10.18637/jss.v082.i11.

Breiman, Leo, Adele Cutler, Andy Liaw, and Matthew Wiener. 2018. *RandomForest: Breiman and Cutler’s Random Forests for Classification and Regression*. https://CRAN.R-project.org/package=randomForest.

Casey, Bryan, Ashkon Farhangi, and Roland Vogl. 2018. “Rethinking Explainable Machines: The Gdpr’s ’Right to Explanation’ Debate and the Rise of Algorithmic Audits in Enterprise.” *Berkeley Technology Law Journal*. https://ssrn.com/abstract=3143325.

Dastin, Jeffrey. 2018. “Amazon Scraps Secret Ai Recruiting Tool That Showed Bias Against Women.” *Reuters*. https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazonscraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G.

Diaz, Mark, Isaac Johnson, Amanda Lazar, Anne Marie Piper, and Darren Gergle. 2018. “Addressing Age-Related Bias in Sentiment Analysis.” In *Proceedings of the 2018 Chi Conference on Human Factors in Computing Systems*, 412:1–412:14. Chi ’18. New York, NY, USA: Acm. https://doi.org/10.1145/3173574.3173986.

Donizy, Piotr, Przemyslaw Biecek, Agnieszka Halon, and Rafal Matkowski. 2016. “BILLCD8 – a Multivariable Survival Model as a Simple and Clinically Useful Prognostic Tool to Identify High-Risk Cutaneous Melanoma Patients” 36 (September): 4739–48.

Duffy, Clare. 2019. “Apple Co-Founder Steve Wozniak Says Apple Card Discriminated Against His Wife.” *CNN Business*. https://edition.cnn.com/2019/11/10/business/goldman-sachs-apple-card-discrimination/index.html.

Efron, Bradley, and Trevor Hastie. 2016. *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science*. 1st ed. New York, NY, USA: Cambridge University Press.

Faraway, Julian. 2002. *Practical Regression and Anova Using R*.

Foster, David. 2017. *XgboostExplainer: An R Package That Makes Xgboost Models Fully Interpretable*. https://github.com/AppliedDataSciencePartners/xgboostExplainer/.

Galecki, Andrzej, and Tomasz Burzykowski. 2013. *Linear Mixed-Effects Models Using R: A Step-by-Step Approach*. Springer Publishing Company, Incorporated.

Gdpr. 2018. “The Eu General Data Protection Regulation (Gdpr) Is the Most Important Change in Data Privacy Regulation in 20 Years.” https://eugdpr.org/.

Goodman, Bryce, and Seth Flaxman. 2016. “European Union Regulations on Algorithmic Decision-Making and a "Right to Explanation".” *Arxiv*. https://arxiv.org/abs/1606.08813.

Grolemund, Garrett, and Hadley Wickham. 2019. *R for Data Science*. https://r4ds.had.co.nz/.

Harrell Jr, Frank E. 2018. *Rms: Regression Modeling Strategies*. https://CRAN.R-project.org/package=rms.

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. “Long Short-Term Memory.” *Neural Computation* 9 (8): 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735.

Hoover, Benjamin, Hendrik Strobelt, and Sebastian Gehrmann. 2019. “ExBERT: A Visual Analysis Tool to Explore Learned Representations in Transformers Models.”

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2014. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.

Jed Wing, Max Kuhn. Contributions from, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, et al. 2016. *Caret: Classification and Regression Training*. https://CRAN.R-project.org/package=caret.

Karbowiak, Ewelina, and Przemyslaw Biecek. 2019. *EIX: Explain Interactions in Gradient Boosting Models*. https://CRAN.R-project.org/package=EIX.

Kuhn, Max, and Kjell Johnson. 2013. *Applied predictive modeling*. New York, NY: Springer. http://appliedpredictivemodeling.com/.

Larson, Jeff, Surya Mattu, Lauren Kirchner, and Julia Angwin. 2016. “How We Analyzed the Compas Recidivism Algorithm.” *ProPublica*. https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm.

Lazer, David, Ryan Kennedy, Gary King, and Alessandro Vespignani. 2014. “The Parable of Google Flu: Traps in Big Data Analysis.” *Science* 343 (6176). American Association for the Advancement of Science: 1203–5. https://doi.org/10.1126/science.1248506.

Liaw, Andy, and Matthew Wiener. 2002. “Classification and Regression by randomForest.” *R News* 2 (3): 18–22. http://CRAN.R-project.org/doc/Rnews/.

Lundberg, Scott M, and Su-In Lee. 2017. “A Unified Approach to Interpreting Model Predictions.” In *Advances in Neural Information Processing Systems 30*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, 4765–74. Curran Associates, Inc. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

Molnar, Christoph. 2019. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*.

O’Neil, Cathy. 2016. *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. New York, NY, USA: Crown Publishing Group.

Paluszynska, Aleksandra, and Przemyslaw Biecek. 2017. *RandomForestExplainer: A Set of Tools to Understand What Is Happening Inside a Random Forest*. https://github.com/MI2DataLab/randomForestExplainer.

R Core Team. 2018. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. 2016. “Why Should I Trust You?: Explaining the Predictions of Any Classifier.” In, 1135–44. ACM Press. https://doi.org/10.1145/2939672.2939778.

Ridgeway, Greg. 2017. *Gbm: Generalized Boosted Regression Models*. https://CRAN.R-project.org/package=gbm.

Ross, Casey, and Ike Swetliz. 2018. “IBM’s Watson Supercomputer Recommended ‘Unsafe and Incorrect’ Cancer Treatments, Internal Documents Show.” *Statnews*. https://www.statnews.com/2018/07/25/ibm-watson-recommended-unsafe-incorrect-treatments/.

Ruiz, Javier. 2018. “Machine Learning and the Right to Explanation in Gdpr.” https://www.openrightsgroup.org/blog/2018/machine-learning-and-the-right-to-explanation-in-gdpr.

Salzberg, Steven. 2014. “Why Google Flu Is A Failure.” *Forbes*. https://www.forbes.com/sites/stevensalzberg/2014/03/23/why-google-flu-is-a-failure/.

Samek, Wojciech, Thomas Wiegand, and Klaus-Robert Müller. 2017. “Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models.”

Shapley, Lloyd S. 1953. “A Value for n-Person Games.” In *Contributions to the Theory of Games Ii*, edited by Harold W. Kuhn and Albert W. Tucker, 307–17. Princeton: Princeton University Press.

Sheather, Simon. 2009. *A Modern Approach to Regression with R*. Springer Texts in Statistics. Springer New York.

Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman. 2013. “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps.” *CoRR* abs/1312.6034. http://arxiv.org/abs/1312.6034.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. 2014. “Sequence to Sequence Learning with Neural Networks.” *CoRR* abs/1409.3215. http://arxiv.org/abs/1409.3215.

Tukey, John W. 1977. *Exploratory Data Analysis*. Addison-Wesley.

Venables, W. N., and B. D. Ripley. 2002. *Modern Applied Statistics with S*. Fourth. New York: Springer. http://www.stats.ox.ac.uk/pub/MASS4.

Wes, McKinney. 2012. *Python for Data Analysis*. 1st ed. O’Reilly Media, Inc.

Wickham, Hadley. 2009. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. http://ggplot2.org.

Wright, Marvin N., and Andreas Ziegler. 2017. *ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R*. *Journal of Statistical Software*. Vol. 77. https://doi.org/10.18637/jss.v077.i01.

Xie, Yihui. 2018. *bookdown: Authoring Books and Technical Documents with R Markdown*. https://CRAN.R-project.org/package=bookdown.

# 2 Model Development

## 2.1 Introduction

In this book we present methods that can be used for exploration and explanation of predictive models. But before we can explore a model, first we need to train one.

In this part of the book we overview the process of model development and introduce steps that lead to a model creation. It is not a comprehensive manual ,,how to train a model in 5 steps". The goal of this chapter is to show what needs to be performed before we can do any diagnostic or exploration of a trained model.

Predictive models are created for different purposes. Sometimes it is a team of data scientists that spend months on a single model that will be used for model scoring in a big financial company. Every detail is important for models that operate on large scale and have long-term consequences. Another time it is an in-house model trained for prediction of a demand for pizza. The model is developed by a single person in few hours. If model will not perform well it will be updated, replaced or removed.

Whatever it is a large model or small one, similar steps are to be taken during model development.

## 2.2 The Process

Several approaches are proposed in order to describe the process of model development. Their main goal is to standardize the process. And the standardisation is important because it helps to plan resources needed to develop and maintain the model and also to not miss any important step.

The most known methodology for data science projects is CRISP-DM (Chapman et al. 1999), (Wikipedia 2019) which is a tool agnostic procedure. The key component of CRISP-DM is the break down of the whole process into six phases, that are iterated: business understanding, data understanding, data preparation, modeling, evaluation and deployment. CRISP-DM is general, it was designed for any data science project. For predictive models some methodologies are introduced in ,,R for Data Science" (Grolemund and Wickham 2019) and

,,On XAI Misconceptions" (Hall 2019).Both are focused on iterative repetitions of some phases. Figure 2.1 presents a variant of iterative process divided into five steps. Data preparation is needed prior to any modeling. Better data is needed for better models. On the other hand, garbage-in garbage-out. Once the data is gathered, steps that are usually highlighted are Data understanding, Model assembly and Model audit. This is the common thinking about model development. Repeat these steps until some convergence, e.g. repeat until best model is identified.

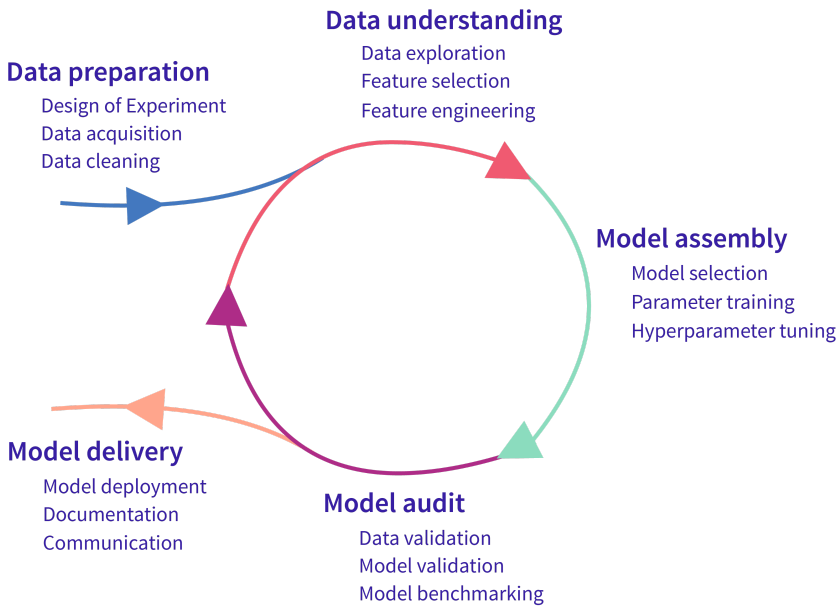



Figure 2.1: Lifecycle of predictive model can be decomposed into five tasks. First we need data that is poured into the model development cycle. The model development is highly iterative, learn something new about the data, assemble a new model based on current understanding, and validate the new model. Repeat these steps as long as needed to be satisfied with model performance. Once the model is created we can deliver the model to the production along with required tests and documentation.

In this book we use *Model Development Process* (Biecek 2019). It is motivated by Rational Unified Process for Software Development (Kruchten 1998), (Jacobson, Booch, and Rumbaugh 1999), (Boehm 1988). One can think about MDP as an extension of process introduced in Figure 2.1. What is important is to notice that consecutive iterations are not identical. Our knowledge increases during the process and consecutive iterations are performed with different goals in mind.

This is why MDP is build as an untangled version of Figure 2.1. The MDP process is shown in Figure 2.2. Each vertical stripe is a single run of the cycle. First iterations are usually focused on *formulation of the problem*. Sometimes the problem is well stated, however it's a rare

situation valid maybe only for kaggle competitions. In most real-life problems the problem formulation requires lots of discussions and experiments. Once the problem is defined we can start building first prototypes, first *crisp versions of models*. These initial versions of models are needed to verify if the problem can be solved and how far we are form the solution. Usually we gather more information and go for the next phase, the *fine tuning*. We repeat these iterations until a final version of a model is developed. Then we move to the last phase *maintenance and* (one day) *decommissioning*.

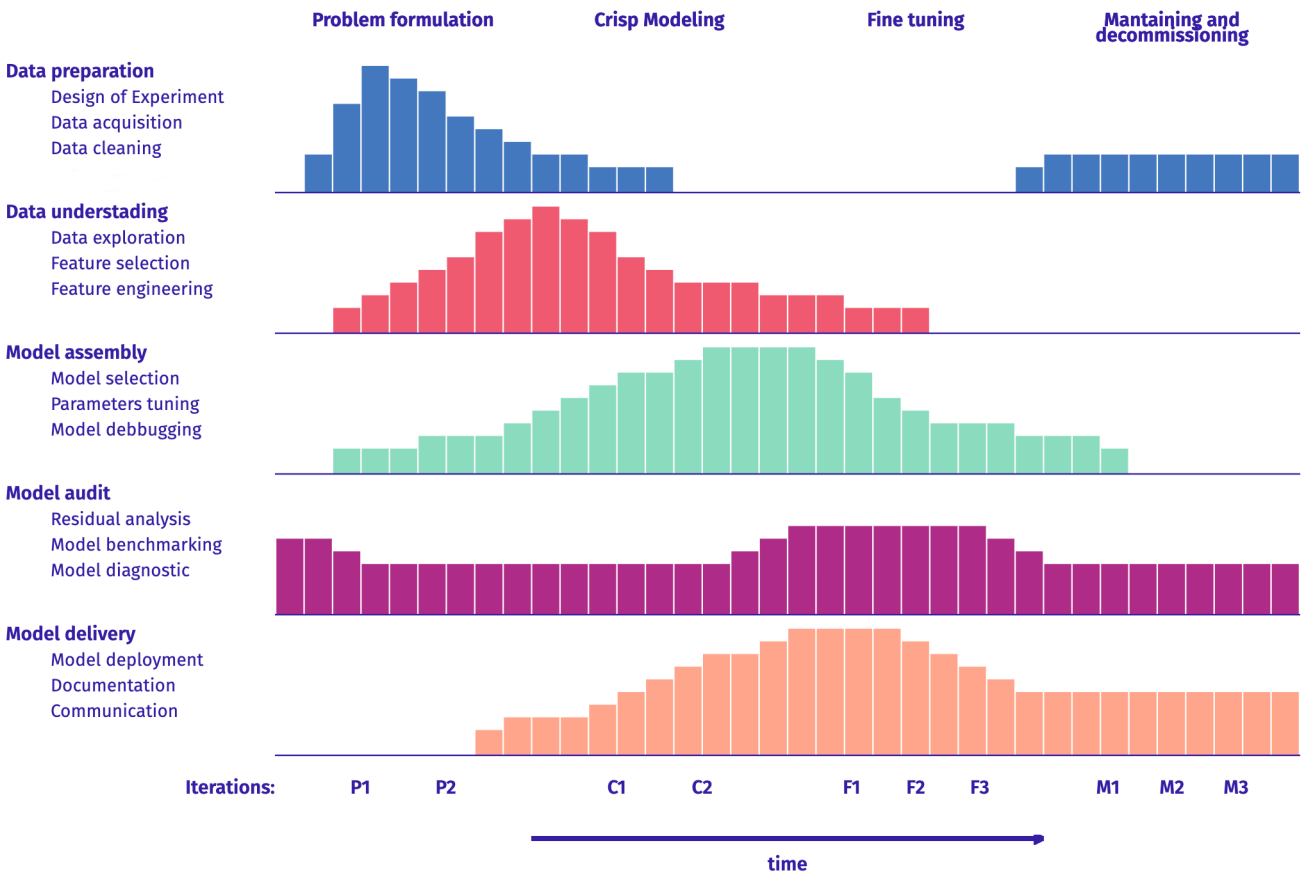



Figure 2.2: Overview of the Model Development Process. Horizontal axis show how time passes from the problem formulation to the model decommissioning. Vertical axis shows tasks are performed in a given phase. Each vertical strip is a next iteration of cycle presented in Figure 2.1

Having in mind the map of model development we can point places where one can use methods presented in this book.

As suggested in the title of this book, three primary applications are: exploration, explanation and debugging. *Exploration* refers to situations in which we better understand the data and the domain. Presented techniques can be used to speed up the variable engineering or variable selection. *Explanation* refers to situations in which we are interested in decision paths

beyond particular predictions. *Debugging* refers to situations in which we want to understand weak points of a model and correct them. These applications target phases Data understanding, Model assembly and Model audit.

In this book we present various examples based on three use cases. Two introduced in Chapter 5 (binary classification in surviving Titanic sinking and regression in apartments pricing) and one in Chapter 21 (estimation of soccer player value based on its skills). Due to space limitation we do not show the full life cycle of these problems, but we are focused on phases Crisp modeling and Fine tuning.

Rest of this chapter is focused on a brief overview of the notation and commonly used methods for data exploration, model training and model validation.

## 2.3 Notation

Methods described in this book were developed by different authors, who used different mathematical notations. We try to keep the mathematical notation consistent throughout the entire book. In some cases this may result in formulas with a fairly complex system of indices.

In this section, we provide a general overview of the notation we use. Whenever necessary, parts of the notation will be explained again in subsequent chapters.

We assume that the data consist $n$ observations/instances. Each observation is described by $p$ explanatory variables. Thus data is described as a set of points on a $p$-dimensional input space $\mathcal{X} \equiv \mathcal{R}^p$. By $x \in \mathcal{X}$ we will refer to a single point in this input space. By $x_i$ we refer to the $i$-th observation in this dataset. Of course, $x_i \in \mathcal{X}$. By $X$ we denote a matrix $n \times p$ with rows corresponding to consecutive observations.

Some methods of model exploration are constructed around an observation of interest which will be denoted by $x_*$. The observation may not necessarily belong to the analyzed dataset; hence, the use of the asterisk in the index. Of course, $x_* \in \mathcal{X}$.

Points in $\mathcal{X}$ are $p$ dimensional vectors. We refer to the $j$-th coordinate by using $j$ in superscript. Thus, $x_i^j$ denotes the $j$-th coordinate of the $i$-th observation from the analyzed dataset. If $\mathcal{J}$ denotes a subset of indices, then $x^{\mathcal{J}}$ denotes the elements of vector $x$ corresponding to the indices included in $\mathcal{J}$.

We will use the notation $x^{-j}$ to refer to a vector that results from removing the $j$-th coordinate from vector $x$. By $x^{j|=z}$, we denote a vector with the values at all coordinates equal to the values in $x$, except of the $j$-th coordinate, which is set equal to $z$. So, if $w = x^{j|=z}$, then $w^j = z$ and $\forall_{k \neq j} w^k = x^k$. In other words $x^{j|=z} = (x^1, \ldots, x^{j-1}, z, x^{j+1}, \ldots, x^p)$.

By $x^{*j}$ we denote a matrix with the values as in $x$ except the $j$th column which is permuted.

In this book, a model is a function $f : \mathcal{X} \to \mathcal{R}$ that transforms a point from $\mathcal{X}$ into a real number. In most cases, the presented methods can be used directly for multivariate dependent variables; however, we use examples with uni-variate responses to simplify the notation. Typically, during the model development, we create many competing models. Formally we shall index models to refer to a specific version of a trained model. But for the sake of simplicity we omit these indexes where they are not important.

Later in this book we will use the term **model residual** as the the difference between the observed value of the dependent variable $Y$ for the $i$-th observation from a particular dataset and the model prediction for the observation

$$r_i = y_i - f(x_i) = y_i - \hat{y}_i. \tag{2.1}$$

## 2.4 Data exploration

Before we start the modeling we need to understand the data. Visual, tabular and statistical tools for data exploration are used depending on the character of variables.

The most know introduction to data exploration is the famous book by John Tukey (Tukey 1977). It introduces new tools for data exploration, like for example boxplots or stem-and-leaf plots. Availability of computational tools makes the process of data exploration easier and more interactive. Find a good overview of techniques for data exploration in ,,Data Science in R" (Nolan and Lang 2015) or ,,R for Data Science" (Wickham and Grolemund 2017).

In this book we will rely on five visual methods for data exploration presented in Figure 2.3. Two of them are used to present distribution of explanatory or target variables; three others are used to explore pairwise relations between variables.

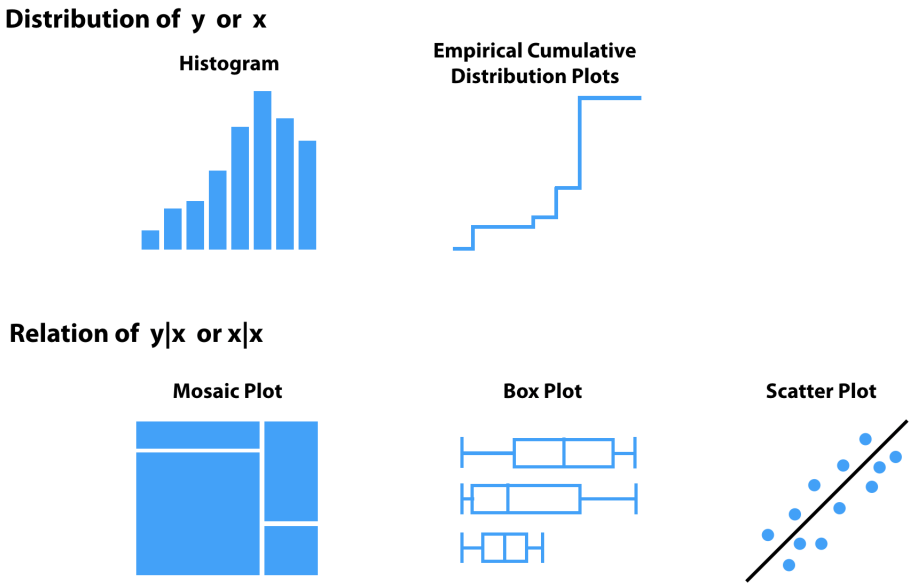



Figure 2.3: Basic methods for visual exploration. Histogram for distribution of continuous or categorical variables, empirical cumulative distribution for continuous variables. Mosaic plot for relation between two categorical variables, boxplots for relation between continuous and categorical variables or scatterplot for relation between two continuous variables.

Distribution of categorical variable is summarized with a barplot, distribution of numerical variable is summarized with a histogram or empirical cumulative distribution function.

Primary goal for exploration of target variable is to decide if some variable transformation is needed (e.g. if the variable is skewed or with fat tails) or to verify if target variable is balanced (because some methods are not working well with unbalanced data). Exploration of dependent variables is performed mainly to decide if any variable transformation is needed.

Relations between two variables, mostly between a single dependent variable and target variable, are visualized with mosaic plots (for two categorical variables), boxplots (for numerical and categorical variable) and scatter plots (for two numerical variables). Such exploration may provide some insights for variable selection/filtering (if the variable is not related with the target then variable may be removed from the model) or variable engineering (if from the exploration we gain information how a variable may be transformed).

## 2.5 Model training

In predictive modeling, we are interested in a distribution of a dependent variable $Y$ given vector $x_*$. The latter contains values of explanatory variables. In the ideal world, we would like to know the conditional distribution of $Y$ given $x_*$. In practical applications, however, we

usually do not predict the entire distribution, but just some of its characteristics like the expected (mean) value, a quantile, or variance. Without loss of generality we will assume that we model the conditional expected value $E_Y(Y|x_*)$.

Assume that we have got model $f()$, for which $f(x_*)$ is an approximation of $E_Y(Y|x_*)$, i.e., $E_Y(Y|x_*) \approx f(x_*)$. Note that we do not assume that it is a "good" model, nor that the approximation is precise. We simply assume that we have a model that is used to estimate the conditional expected value and to form predictions of the values of the dependent variable. Our interest lies in the evaluation of the quality of the predictions. If the model offers a "good" approximation of the conditional expected value, it should be reflected in its satisfactory predictive performance.

Usually the available data is split into two parts. One will be used for model training (estimation of model parameters), second will be used for model validation. The splitting may be repeated as in k-fold cross validation or repeated k-fold cross validation (see for example ,,Applied predictive modeling" (Kuhn and Johnson 2013)). We leave the topic of model validation for Chapter 16.

Training procedures are different for different models, but most of them can be written as an optimization problem. Let $\Theta$ be a space for possible model parameters. Model training is a procedure of selection a $\theta \in \Theta$ that maximize some loss function $L(y, f_\theta(X))$. For models with large parameter spaces it is common to add additional term $\lambda(\theta)$ that control the model complexity.

$$\hat{\theta} = \arg\min_{\theta \in \Theta} L(y, f_\theta(X)) + \lambda(\theta). \tag{2.2}$$

For statistical models it is common to assume some family of probability distributions for $y|x$. In such case the loss function $L$ may be defined as a minus log-likelihood function for $\theta$. Likelihood is probability of observing $y|x$ as a function of parameter $\theta$.

For example, in linear regression we assume that that observed vector of values $y$ follow a multidimensional Gaussian distribution

$$y \sim \mathcal{N}(X\beta, I\sigma^2),$$

where $\theta = (\beta, \sigma^2)$. In this case equation (2.2) become

$$\hat{\beta} = \arg\min_{\theta \in \Theta} ||y - X\beta||_2 + \lambda(\beta). \tag{2.3}$$

For linear regression, the penalty term $\lambda(\beta)$ is equal to $0$, and optimal parameters $\beta$ in equation (2.3) have close analytical solution $\hat{\beta} = (X^T X)^{-1} X^T y$. In ridge regression the penalty $\lambda(\beta) = \lambda||\beta||_2$ and also (2.3) have analytical solution $\hat{\beta} = (X^T X + \lambda I)^{-1} X^T y$. For

LASSO regression the penalty $\lambda(\beta) = \lambda||\beta||_1$ and $\beta$ are estimated through a numerical optimization.

For classification, the natural choice for distribution of $y$ is a Binomial distribution. This leads to logistic regression and logistic loss function. For multi label classification frequent choice is the cross-entropy loss function.

Apart from linear models for $y$ there is a large variety of predictive models. Find a good overview of different techniques for model development in ,,Modern Applied Statistics with S" (Venables and Ripley 2002) or ,,Applied predictive modeling" (Kuhn and Johnson 2013).

## 2.6 Model understanding

Usually the model development starts with some crisp early versions that are refined in consecutive iterations. In order to train a final model we need to try numerous candidate models that will be explored, examined and diagnosed. In this book we will introduce techniques that:

- summarise how good is the current version of a model. Section 16 overviews measures for model performance. These measures are usually used to trace the progress in model development.
- assess the feature importance. Section 17 shows how to assess influence of a single variable on model performance. Features that are not important are usually removed from a model during the model refinement.
- shows how a single feature affects the model response. Sections 18 – 19 present Partial Dependence Profiles, Accumulated Local Effects and Marginal Profiles. All these techniques help to understand how model consumes particular features.
- identifies potential problems with a model. Section 20 shows techniques for exploration of model residuals. Looking closer on residuals often help to improve the model. This is possible with tools for local model exploration which are presented in the fist part of the book.
- performs sensitivity analysis for a model. Section 11 introduces Ceteris Paribus profiles that helps in a what-if analysis for a model.
- validated local fit for a model. Section 13 introduces techniques for assessment if for a single observation the model support its prediction.
- decompose model predictions into pieces that can be attributed to particular variables. Sections 7 – 10 show different techniques like SHAP, LIME or Break Down for local exploration of a model.

# References


Biecek, Przemyslaw. 2019. “Model Development Process.” *CoRR* abs/1907.04461. http://arxiv.org/abs/1907.04461.

Boehm, Barry. 1988. *A Spiral Model of Software Development and Enhancement*. *IEEE Computer, IEEE, 21(5):61-72*.

Chapman, Pete, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rudiger Wirth. 1999. *The CRISP-DM 1.0 Step-by-step data mining guide*. ftp://ftp.software.ibm.com/software/analytics/spss/support/Modeler/Documentation/14/UserManual/CRISP-DM.pdf.

Grolemund, Garrett, and Hadley Wickham. 2019. *R for Data Science*. https://r4ds.had.co.nz/.

Hall, Patrick. 2019. *On Explainable Machine Learning Misconceptions and a More Human-Centered Machine Learning*. https://github.com/jphall663/xai_misconceptions/blob/master/xai_misconceptions.pdf.

Jacobson, Ivar, Grady Booch, and James Rumbaugh. 1999. *The Unified Software Development Process*.

Kruchten, Philippe. 1998. *The Rational Unified Process*.

Kuhn, Max, and Kjell Johnson. 2013. *Applied predictive modeling*. New York, NY: Springer. http://appliedpredictivemodeling.com/.

Nolan, Deborah, and Duncan Temple Lang. 2015. *Data Science in R: A Case Studies Approach to Computational Reasoning and Problem Solving*. Chapman & Hall/CRC.

Tukey, John W. 1977. *Exploratory Data Analysis*. Addison-Wesley.

Venables, W. N., and B. D. Ripley. 2002. *Modern Applied Statistics with S*. Fourth. New York: Springer. http://www.stats.ox.ac.uk/pub/MASS4.

Wickham, Hadley, and Garrett Grolemund. 2017. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. 1st ed. O’Reilly Media, Inc.

Wikipedia. 2019. *CRISP DM: Cross-industry standard process for data mining*. https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining.

# 3 Do-it-yourself with R

In this book we introduce various methods for instance-level and dataset-level explanation and exploration of predictive models. In each chapter, there is a section with code snippets for R and Python that shows how to use a particular method. In this chapter we provide a short description of steps that are needed to set-up the environment with required libraries.

## 3.1 What to install?

Obviously, the R software (R Core Team 2018) is needed. It is always a good idea to use the newest version. At least R in version 3.6 is recommended. R can be downloaded from the CRAN website https://cran.r-project.org/.

A good editor makes working with R much easier. There is a plenty of choices, but, especially for beginners, it is worth considering the RStudio editor, an open-source and enterprise-ready tool for R. It can be downloaded from https://www.rstudio.com/.

Once R and the editor are available, the required packages should be installed.

The most important one is the `DALEX` package in version 1.0 or newer. It is the entry point to solutions introduced in this book. The package can be installed by executing the following command from the R command line:

```
install.packages("DALEX")
```

Installation of `DALEX` will automatically take care about installation of other hard requirements (packages required by it), like the `ggplot2` package for data visualization.

To repeat all examples in this book, two additional packages are needed: `ingredients` and `iBreakDown`. The easiest way to get them, including other useful weak dependencies, is to execute the following command:

```
DALEX::install_dependencies()
```

## 3.2 How to work with `DALEX` ?

To conduct model exploration with `DALEX`, first, a model has to be created. Then the model has got to be prepared for exploration.

There are many packages in R that can be used to construct a model. Some packages are structure-specific, like `randomForest` for Random-Forest Classification and Regression models (Liaw and Wiener 2002), `gbm` for Generalized Boosted Regression Models (Ridgeway 2017), extensions for Generalized Linear Models (Harrell Jr 2018), or many others. There is also a number of packages that can be used for constructing models with different structures. These include the `h2o` package (LeDell et al. 2019), `caret` (Jed Wing et al. 2016) and its successor `parsnip` (Kuhn and Vaughan 2019), a very powerful and extensible framework `mlr` (Bischl et al. 2016), or `keras` that is a wrapper to Python library with the same name (Allaire and Chollet 2019).

While it is great to have such a large choice of tools for constructing models, the downside is that different packages have different interfaces and different arguments. Moreover, model-objects created with different packages may have different internal structures. The main goal of the `DALEX` package is to create a level of abstraction around a model that makes it easier to explore and explain the model.

Function `DALEX::explain` is THE function for model wrapping. The function requires five arguments:

- `model`, a model-object;
- `data`, a data frame with validation data;
- `y`, observed values of the dependent variable for the validation data; it is an optional argument, required for explainers focused on model validation and benchmarking.
- `predict_function`, a function that returns prediction scores; if not specified, then a default `predict()` function is used. Note that, for some models, the default `predict()` function returns classes; in such cases you should provide a function that will return numerical scores.
- `label`, a name of a model; if not specified, then it is extracted from the `class(model)`. This name will be presented in figures, so it is recommended to make the name informative.

For an example, see Section 5.1.7.

## 3.3 How to work with `archivist` ?

As we will focus on exploration of predictive models, we prefer not to waste space nor time on replication of the code necessary for model development. This is where the `archivist` packages helps.

The `archivist` package (Biecek and Kosinski 2017) is designed to store, share, and manage R objects. We will use it to easily access pretrained R models and precalculated explainers. To install the package, the following command should be executed in the R command line:

```
install.packages("archivist")
```

Once the package has been installed, function `aread()` can be used to retrieve R objects from any remote repository. For this book, we use a GitHub repository `models` hosted at https://github.com/pbiecek/models. For instance, to download a model with the md5 hash `ceb40`, the following command has to be executed:

```
archivist::aread("pbiecek/models/ceb40")
```

Since the md5 hash `ceb40` uniquely defines the model, referring to the repository object results in using exactly the same model and the same explanations. Thus, in the subsequent chapters, pre-constructed model explainers will be accessed with `archivist` hooks. In following sections we will also use `archivist` hooks in references to datasets.

## References


Allaire, JJ, and François Chollet. 2019. *Keras: R Interface to 'Keras'*. https://CRAN.R-project.org/package=keras.

Biecek, Przemyslaw, and Marcin Kosinski. 2017. "archivist: An R Package for Managing, Recording and Restoring Data Analysis Results." *Journal of Statistical Software* 82 (11): 1–28. https://doi.org/10.18637/jss.v082.i11.

Bischl, Bernd, Michel Lang, Lars Kotthoff, Julia Schiffner, Jakob Richter, Erich Studerus, Giuseppe Casalicchio, and Zachary M. Jones. 2016. "mlr: Machine Learning in R." *Journal of Machine Learning Research* 17 (170): 1–5. http://jmlr.org/papers/v17/15-066.html.

Harrell Jr, Frank E. 2018. *Rms: Regression Modeling Strategies*. https://CRAN.R-project.org/package=rms.

Jed Wing, Max Kuhn. Contributions from, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, et al. 2016. *Caret: Classification and Regression Training*. https://CRAN.R-project.org/package=caret.

Kuhn, Max, and Davis Vaughan. 2019. *Parsnip: A Common Api to Modeling and Analysis Functions*. https://CRAN.R-project.org/package=parsnip.

LeDell, Erin, Navdeep Gill, Spencer Aiello, Anqi Fu, Arno Candel, Cliff Click, Tom Kraljevic, et al. 2019. *H2o: R Interface for 'H2o'*. https://CRAN.R-project.org/package=h2o.

Liaw, Andy, and Matthew Wiener. 2002. "Classification and Regression by randomForest." *R News* 2 (3): 18–22. http://CRAN.R-project.org/doc/Rnews/.

R Core Team. 2018. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Ridgeway, Greg. 2017. *Gbm: Generalized Boosted Regression Models*. https://CRAN.R-project.org/package=gbm.

# 5 Data sets and models

We illustrate the methods presented in this book by using two datasets:

- Predicting odds of survival out of *Sinking of the RMS Titanic*
- Predicting prices for *Apartments in Warsaw*

The first dataset will be used to illustrate the application of the techniques in the case of a predictive model for a binary dependent variable. The second one will provide an example for models for a continuous variable.

In this chapter, we provide a short description of each of the datasets, together with results of exploratory analyses. We also introduce models that will be used for illustration purposes in subsequent chapters.

## 5.1 Sinking of the RMS Titanic

Titanic sinking by Willy Stöwer

Sinking of the RMS Titanic is one of the deadliest maritime disasters in history (during peacetime). Over 1500 people died as a consequence of collision with an iceberg. Projects like *Encyclopedia titanica* `https://www.encyclopedia-titanica.org/` are a source of rich and precise data about Titanic's passengers. The `stablelearner` package includes a data frame with some passenger characteristics. The dataset, after some data cleaning and variable transformations, is also available in the `DALEX` package. In particular, the `titanic` data frame contains 2207 observations (for 1317 passengers and 890 crew members) and nine variables:

- *gender*, person's (passenger's or crew member's) gender, a factor (categorical variable) with two levels (categories) `male` (78%) and `female` (22%);
- *age*, person's age in years, a numerical variable; the age is given in (integer) years, range 0 – 74 years;
- *class*, the class in which the passenger travelled, or the duty class of a crew member; a factor with seven levels: `1st` (14.7%), `2nd` (12.9%), `3rd` (32.1%), `deck crew` (3%), `engineering crew` (14.7%), `restaurant staff` (3.1%), `victualling crew` (19.5%);
- *embarked*, the harbor in which the person embarked on the ship, a factor with four levels, `Belfast` (8.9%), `Cherbourg` (12.3%), `Queenstown` (5.6%), `Southampton` (73.2%);
- *country*, person's home country, a factor with 48 levels, the most common are `England` (51%), `United States` (12%), `Ireland` (6.2%) and `Sweden` (4.8%);
- *fare*, the price of the ticket (only available for passengers; 0 for crew members), a numerical variable range 0 – 512;
- *sibsp*, the number of siblings/spouses aboard the ship, a numerical variable range 0 – 8;
- *parch*, the number of parents/children aboard the ship, a numerical variable range 0 – 9;
- *survived*, a factor with two levels `yes` (67.8%), `no` (32.2%), indicating whether the person survived or not.

The R code below provides more info about the contents of the dataset, values of the variables, etc.

```
library("DALEX")
head(titanic, 2)
```

```
##   gender age class    embarked       country  fare sibsp parch survived
## 1   male  42   3rd Southampton United States  7.11     0     0       no
## 2   male  13   3rd Southampton United States 20.05     0     2       no
```

Models considered for this dataset will use *survived* as the (binary) dependent variable.

### 5.1.1 Data exploration

It is always advisable to explore data before modelling. However, as this book is focused on model exploration, we will limit the data exploration part.

Before exploring the data, we first do some pre-processing. In particular, the value of variables *age*, *country*, *sibsp*, *parch*, and *fare* is missing for a limited number of observations (2, 81, 10, 10, and 26, respectively). Analyzing data with missing values is a topic on its own (Little and Rubin 1987; Schafer 1997; Molenberghs and Kenward 2007). An often-used approach is to impute the missing values. Toward this end, multiple imputation should be considered (Schafer 1997; Molenberghs and Kenward 2007; van Buuren 2012). However, given the limited number of missing values and the intended illustrative use of the dataset, we will limit ourselves to, admittedly inferior, single imputation. In particular, we replace the missing *age* values by the mean of the observed ones, i.e., 30. Missing *country* will be coded by "X". For *sibsp* and *parch*, we replace the missing values by the most frequently observed value, i.e., 0. Finally, for *fare*, we use the mean fare for a given *class*, i.e., 0 pounds for crew, 89 pounds for the 1st, 22 pounds for the 2nd, and 13 pounds for the 3rd class. The R code presented below implements the imputation steps.

- missing `age` is replaced by its average, that is 30

```
titanic$age[is.na(titanic$age)] = 30
```

- missing `country` is replaced by `"X"`

```
titanic$country <- as.character(titanic$country)
titanic$country[is.na(titanic$country)] = "X"
titanic$country <- factor(titanic$country)
```

- missing `fare` is replaced by within `class` average, that is 89, 22 and 13 correspondingly

```
titanic$fare[is.na(titanic$fare) & titanic$class == "1st"] = 89
titanic$fare[is.na(titanic$fare) & titanic$class == "2nd"] = 22
titanic$fare[is.na(titanic$fare) & titanic$class == "3rd"] = 13
```

- missing `sibsp` and `parch` are replaced by 0

```
titanic$sibsp[is.na(titanic$sibsp)] = 0
titanic$parch[is.na(titanic$parch)] = 0
```

After imputing the missing values, we investigate the association between survival status and other variables. Most variables in the Titanic dataset are categorical, except Age and Fare. In order to keep the exploration uniform we first transformed them into categorical variables. Figure 5.1 shows histograms for both variables. Age is discretized into 5 categories with cutoffs 5, 10, 20 and 30 while Fare is discretized with cutoffs 1, 10, 25, and 50.

Figures 5.2-5.5 present graphically the proportion non- and survivors for different levels of the other variables with the use of mosaic plots. The height of the bars (on the y-axis) reflects the marginal distribution (proportions) of the observed levels of the variable. On the other hand, the width of the bars (on the x-axis) provides the information about the proportion of non- and survivors. Note that, to construct the graphs for *age* and *fare*, we categorized the range of the observed values.

Figure 5.2 indicates that the proportion of survivors was larger for females and children below 5 years of age. This is most likely the result of the “women and children first” principle that is often evoked in situations that require evacuation of persons whose life is in danger. The principle can, perhaps, partially explain the trend seen in Figure 5.3, i.e., a higher proportion of survivors among those with 1-3 parents/children and 1-2 siblings/spouses aboard. Figure 5.4 indicates that passengers travelling in the first and second class had a higher chance of survival, perhaps due to the proximity of the location of their cabins to the deck. Interestingly, the proportion of survivors among crew deck was similar to the proportion of the first-class passengers. It also shows that the proportion of survivors increased with the fare, which is consistent with the fact that the proportion was higher for passengers travelling in the first and second class. Finally, Figure 5.5 does not suggest any noteworthy trends.

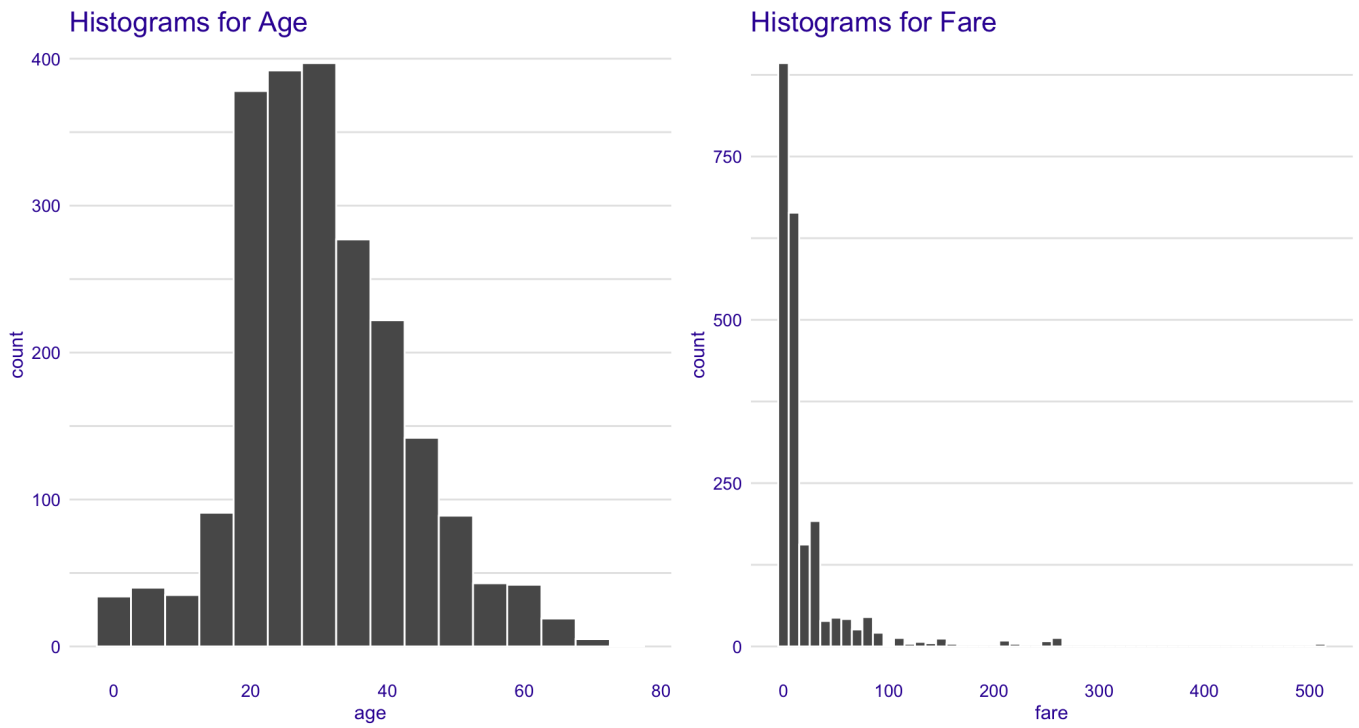



Figure 5.1: Histogram of Age and Fare for the Titanic data.

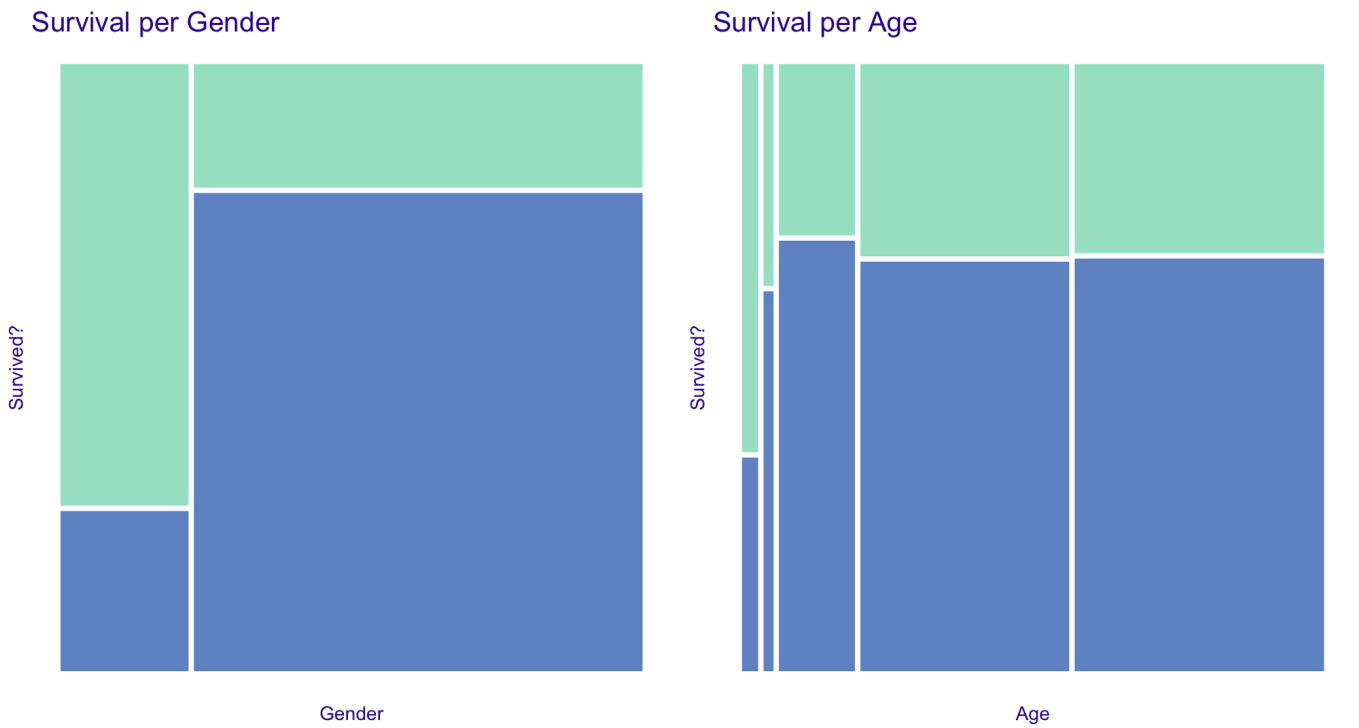



Figure 5.2: Survival status in groups defined be Gender and Age for the Titanic data.

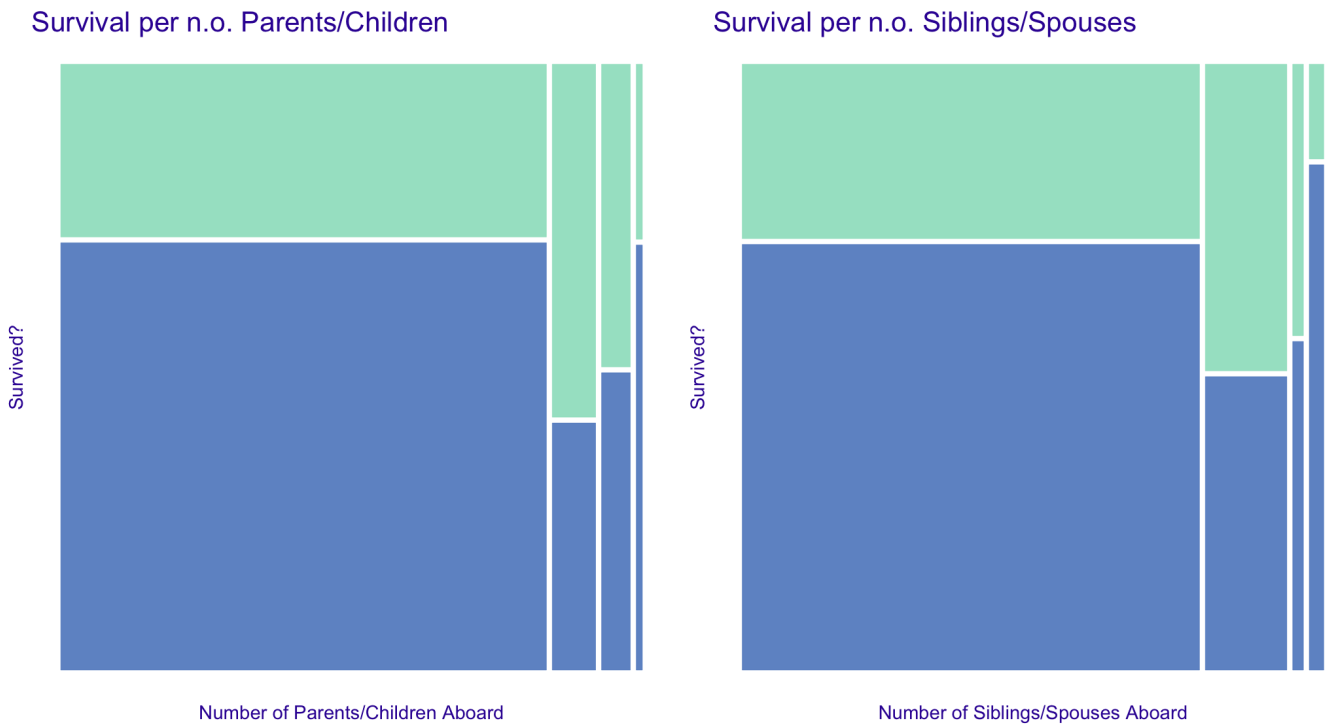



Figure 5.3: Survival according to the number of parents/children and siblings/spouses in the Titanic data.

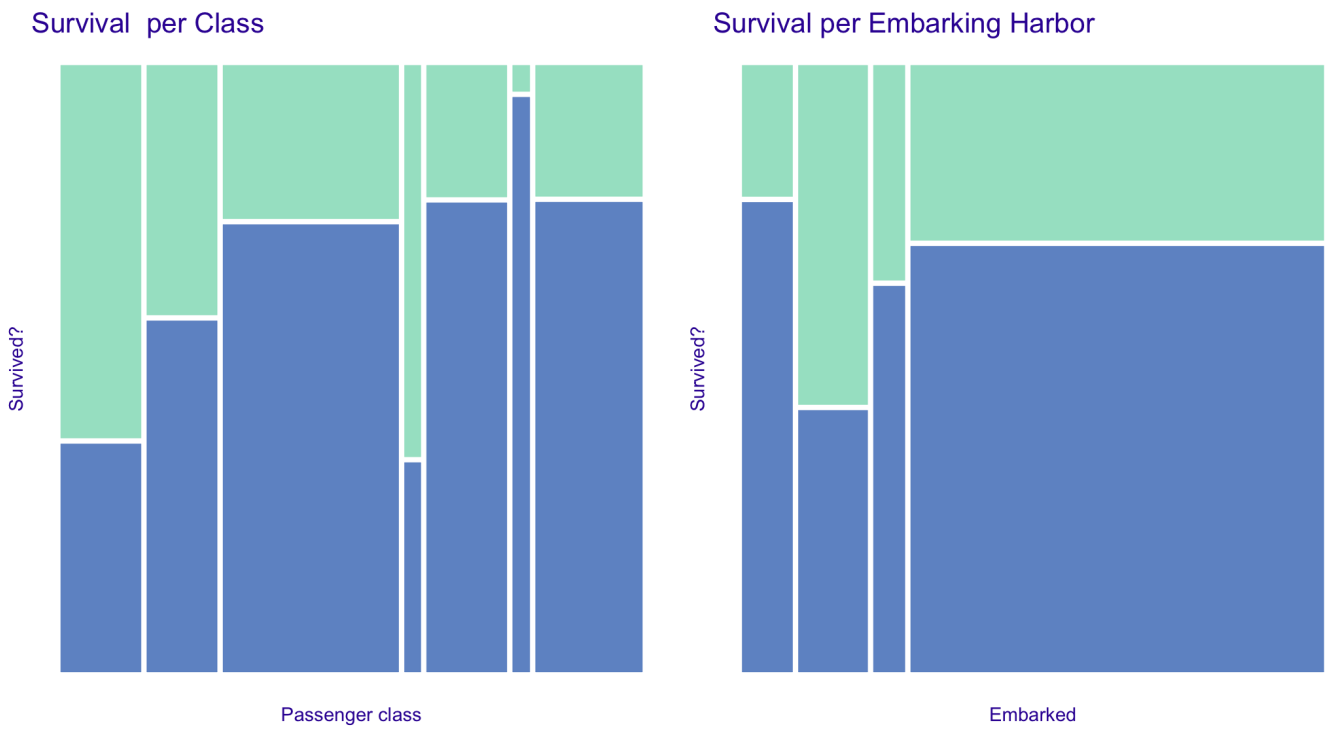



Figure 5.4: Survival according to the class and port of embarking in the Titanic data.

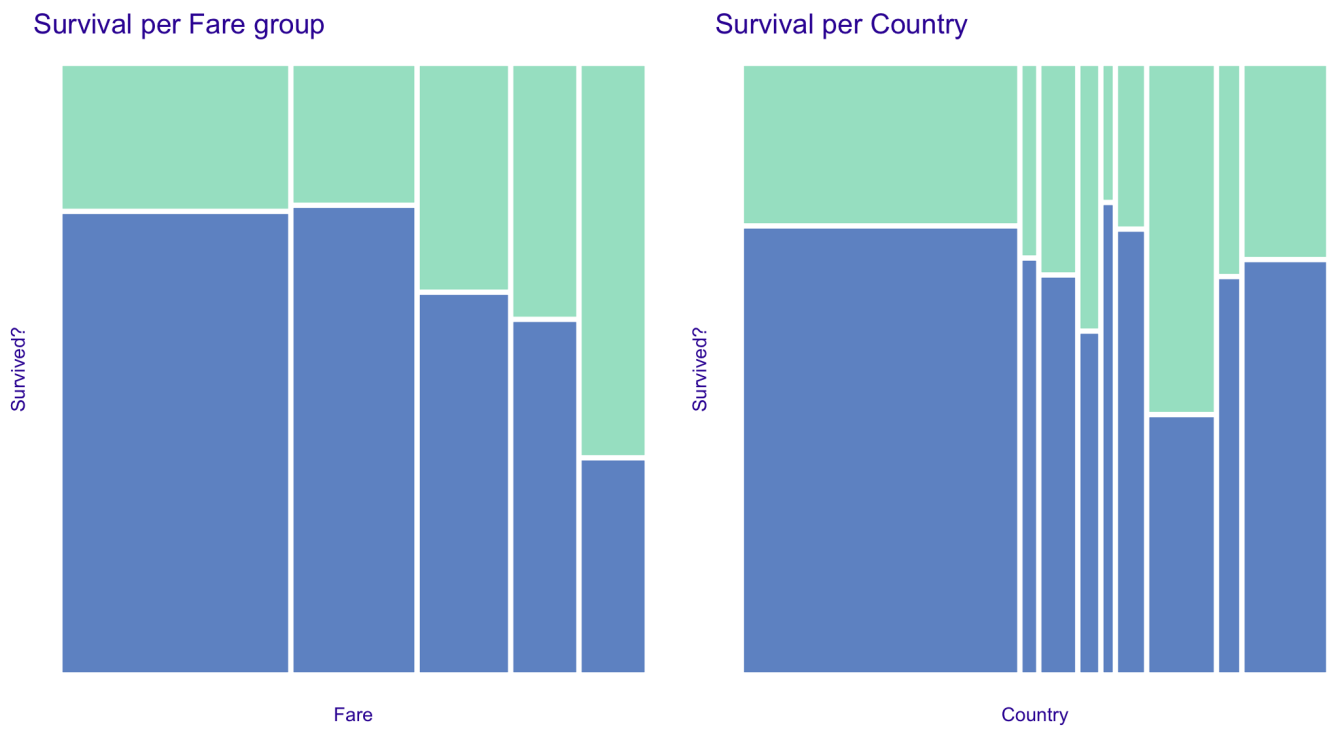



Figure 5.5: Survival according to fare and country in the Titanic data.

### 5.1.2 Logistic regression model

The dependent variable of interest, *survival*, is binary. Thus, a natural choice is to start the predictive modelling with logistic regression model. As there is no reason to expect a linear relationship between age and odds of survival, we use linear tail-restricted cubic splines, available in the `rcs()` function of the `rms` package (Harrell Jr 2018), to model the effect of age. We also do not expect linear relation for the `fare` variable, but because of it's skewness, we do not use splines for this variable. The results of the model are stored in model-object `titanic_lmr_v6` , which will be used in subsequent chapters.

```
library("rms")
set.seed(1313)
titanic_lmr_v6 <- lrm(survived == "yes" ~ gender + rcs(age) + class +
         sibsp + parch + fare + embarked, titanic)
titanic_lmr_v6
```

```
## Logistic Regression Model
##
##  lrm(formula = survived == "yes" ~ gender + rcs(age) + class +
##      sibsp + parch + fare + embarked, data = titanic)
##
##                        Model Likelihood     Discrimination    Rank Discrim.
##                           Ratio Test           Indexes           Indexes
##  Obs           2207    LR chi2     752.06    R2       0.404    C       0.817
##   FALSE        1496    d.f.            17    g        1.647    Dxy     0.635
##   TRUE          711    Pr(> chi2) <0.0001    gr       5.191    gamma   0.636
##  max |deriv| 0.0001                          gp       0.282    tau-a   0.277
##                                              Brier    0.146
##
##                         Coef    S.E.   Wald Z Pr(>|Z|)
##  Intercept               4.5746 0.5480   8.35 <0.0001
##  gender=male            -2.7687 0.1586 -17.45 <0.0001
##  age                    -0.1180 0.0221  -5.35 <0.0001
##  age'                    0.6313 0.1628   3.88 0.0001
##  age''                  -2.6583 0.7840  -3.39 0.0007
##  age'''                  2.8977 1.0130   2.86 0.0042
##  class=2nd              -1.1390 0.2501  -4.56 <0.0001
##  class=3rd              -2.0627 0.2490  -8.28 <0.0001
##  class=deck crew         1.0672 0.3498   3.05 0.0023
##  class=engineering crew -0.9702 0.2648  -3.66 0.0002
##  class=restaurant staff -3.1712 0.6583  -4.82 <0.0001
##  class=victualling crew -1.0877 0.2596  -4.19 <0.0001
##  sibsp                  -0.4504 0.1006  -4.48 <0.0001
##  parch                  -0.0871 0.0987  -0.88 0.3776
##  fare                    0.0014 0.0020   0.70 0.4842
##  embarked=Cherbourg      0.7881 0.2836   2.78 0.0055
##  embarked=Queenstown     0.2745 0.3409   0.80 0.4208
##  embarked=Southampton    0.2343 0.2119   1.11 0.2689
##
```

Note that our prime interest is not in the assessment of model performance, but rather in the understanding of model behavior. This is why we do not split the data into train/test subsets. The model is trained and will be explained on the whole dataset.

### 5.1.3 Random forest model

As a challenger to the logistic regression model, we consider a random forest model. Random forest is known for good predictive performance, is able to grasp low-order variable interactions, and is quite stable (Breiman 2001). To fit the model, we apply the `randomForest()` function, with default settings, from the package with the same name (Liaw and Wiener 2002).

In the first instance, we fit a model with the same set of explanatory variables as the logistic regression model. The results of the model are stored in model-object `titanic_rf_v6`.

```
library("randomForest")
set.seed(1313)
titanic_rf_v6 <- randomForest(survived ~ class + gender + age + sibsp +
         parch + fare + embarked, data = titanic)
titanic_rf_v6
```

```
##
## Call:
##  randomForest(formula = survived ~ class + gender + age + sibsp +      parch + fare +
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 18.62%
## Confusion matrix:
##       no yes class.error
## no  1393 103  0.06885027
## yes  308 403  0.43319269
```

For comparison purposes, we also consider a model with only three explanatory variables: *class*, *gender*, and *age*. The results of the model are stored in model-object `titanic_rf_v3`.

```
titanic_rf_v3 <- randomForest(survived ~ class + gender + age,
         data = titanic)
titanic_rf_v3
```

```
##
## Call:
##  randomForest(formula = survived ~ class + gender + age, data = titanic)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 21.02%
## Confusion matrix:
##       no yes class.error
## no  1367 129  0.08622995
## yes  335 376  0.47116737
```

### 5.1.4 Gradient boosting model

Let's consider an another challenger – the gradient-boosting model (Friedman 2000). The tree based boosting models are known for being able to accommodate higher-order interactions between variables. We use the same set of six explanatory variables as for the logistic regression model. To fit the gradient-boosting model, we use function `gbm()` from the `gbm` package (Ridgeway 2017). The results of the model are stored in model-object `titanic_gbm_v6` .

```
library("gbm")
set.seed(1313)
titanic_gbm_v6 <- gbm(survived == "yes" ~ class + gender + age + sibsp +
         parch + fare + embarked, data = titanic, n.trees = 15000,
         distribution = "bernoulli")
titanic_gbm_v6
```

```
## gbm(formula = survived == "yes" ~ class + gender + age + sibsp +
##     parch + fare + embarked, distribution = "bernoulli", data = titanic,
##     n.trees = 15000)
## A gradient boosted model with bernoulli loss function.
## 15000 iterations were performed.
## There were 7 predictors of which 7 had non-zero influence.
```

### 5.1.5 Support Vector Machine model

Finally, we consider also Support Vector Machine model (Cortes and Vapnik 1995). We use the C-classification mode. To fit the Support Vector Machine model, we use function `svm()` from the `e1071` package (Meyer et al. 2019). The results of the model are stored in model-object `titanic_svm_v6` .

```
library("e1071")
set.seed(1313)
titanic_svm_v6 <- svm(survived == "yes" ~ class + gender + age + sibsp +
            parch + fare + embarked, data = titanic,
            type = "C-classification", probability = TRUE)
titanic_svm_v6
```

```
##
## Call:
## svm(formula = survived == "yes" ~ class + gender + age + sibsp +
##     parch + fare + embarked, data = titanic, type = "C-classification",
##     probability = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  1030
```

### 5.1.6 Model predictions

Let us now compare predictions that are obtained from the three different models. In particular, we will compute the predicted probability of survival for an 8-year-old boy who embarked in Belfast and travelled in the 1-st class with no parents nor siblings and with a ticket costing 72 pounds.

First, we create a dataframe `johny_d` that contains the data describing the passenger.

```r
johny_d <- data.frame(
            class = factor("1st", levels = c("1st", "2nd", "3rd", "deck crew",
                        "engineering crew", "restaurant staff", "victualling crew")),
            gender = factor("male", levels = c("female", "male")),
            age = 8,
            sibsp = 0,
            parch = 0,
            fare = 72,
            embarked = factor("Southampton", levels = c("Belfast",
                        "Cherbourg","Queenstown","Southampton"))
)
```

Subsequently, we use the generic function `predict()` to get the predicted probability of survival for the logistic regression model.

```r
(pred_lmr <- predict(titanic_lmr_v6, johny_d, type = "fitted"))
```

```
##         1
## 0.7677036
```

The predicted probability is equal to 0.77.

We do the same for the random forest and gradient boosting models.

```r
(pred_rf <- predict(titanic_rf_v6, johny_d, type = "prob"))
```

```
##      no   yes
## 1 0.578 0.422
## attr(,"class")
## [1] "matrix" "votes"
```

```r
(pred_gbm <- predict(titanic_gbm_v6, johny_d, type = "response", n.trees = 15000))
```

```
## [1] 0.6632574
```

As a result, we obtain the predicted probabilities of 0.42 and 0.66, respectively.

The models lead to different probabilities. Thus, it might be of interest to understand the reason for the differences, as it could help us to decide which of the predictions we might want to trust.

Note that for some examples we will use another observation (instance) with lower chances of survival. Let's call this passenger Henry.

```
henry <- data.frame(
         class = factor("1st", levels = c("1st", "2nd", "3rd", "deck crew",
                        "engineering crew", "restaurant staff", "victualling crew")),
         gender = factor("male", levels = c("female", "male")),
         age = 47,
         sibsp = 0,
         parch = 0,
         fare = 25,
         embarked = factor("Cherbourg", levels = c("Belfast",
                           "Cherbourg","Queenstown","Southampton"))
)
predict(titanic_lmr_v6, henry, type = "fitted")
```

```
##         1
## 0.4318245
```

```
predict(titanic_rf_v6, henry, type = "prob")
```

```
##      no   yes
## 1 0.754 0.246
## attr(,"class")
## [1] "matrix" "votes"
```

```
predict(titanic_gbm_v6, henry, type = "response", n.trees = 15000)
```

```
## [1] 0.3073358
```

### 5.1.7 Model adapters

Model-objects created with different machine learning libraries may have different internal structures. Thus, first, we have got to create an adapter for the model that provides an uniform interface. Toward this end, we use the `explain()` function from the `DALEX` package (Biecek 2018). The function requires five arguments:

- `model`, a model-object;
- `data`, a validation data frame;
- `y`, observed values of the dependent variable for the validation data;
- `predict_function`, a function that returns prediction scores; if not specified, then a default `predict()` function is used;
- `label`, an unique name of the model; if not specified, then it is extracted from the `class(model)`.

Each adapter contains all elements needed to create a model explanation, i.e., a suitable `predict()` function, validation data set, and the model object. Thus, in subsequent chapters we will use the explainers instead of the model objects to keep code snippets more concise.

```
explain_titanic_lmr_v6 <- explain(model = titanic_lmr_v6,
                                  data = titanic[, -9],
                                  y = titanic$survived == "yes",
                                  label = "Logistic Regression")
explain_titanic_lmr_v6$model_info$type = "classification"
explain_titanic_rf_v6 <- explain(model = titanic_rf_v6,
                                  data = titanic[, -9],
                                  y = titanic$survived == "yes",
                                  label = "Random Forest")
explain_titanic_rf_v3 <- explain(model = titanic_rf_v3,
                                  data = titanic[, -9],
                                  y = titanic$survived == "yes",
                                  label = "Random Forest small")
explain_titanic_gbm_v6 <- explain(model = titanic_gbm_v6,
                                  data = titanic[, -9],
                                  y = titanic$survived == "yes",
                                  label = "Generalized Boosted Regression")
explain_titanic_svm_v6 <- explain(model = titanic_svm_v6,
                                  data = titanic[, -9],
                                  y = titanic$survived == "yes",
                                  label = "Support Vector Machine")
```

### 5.1.8 List of objects for the `titanic` example

In the previous sections we have built four predictive models for the `titanic` data set. The models will be used in the rest of the book to illustrate the model explanation methods and tools.

For the ease of reference, we summarize the models in Table 5.1. The binary model-objects can be downloaded by using the indicated `archivist` hooks (Biecek and Kosinski 2017). By calling a function specified in the last column of the table, one can restore a selected model in its local R environment.

Table 5.1: Predictive models created for the `titanic` dataset.

| Model name | Model generator | Variables | Archivist hooks |
|---|---|---|---|
| `titanic_lmr_v6` | `rms::lmr` v.5.1.3 | gender, age, class, sibsp, parch, fare, embarked | Get the model: `archivist::aread("pbiecek/models/56d8a")`. Get the explainer: `archivist::aread("pbiecek/models/ff1cd")` |
| `titanic_rf_v6` | `randomForest::randomForest` v.4.6.14 | gender, age, class, sibsp, parch, fare, embarked | Get the model: `archivist::aread("pbiecek/models/31570")`. Get the explainer: `archivist::aread("pbiecek/models/6ed54")` |
| `titanic_rf_v3` | `randomForest::randomForest` v.4.6.14 | gender, age, class | Get the model: `archivist::aread("pbiecek/models/855c1")`. Get the explainer: `archivist::aread("pbiecek/models/5b32a")` |
| `titanic_gbm_v6` | `gbm::gbm` v.2.1.5 | gender, age, class, sibsp, parch, fare, embarked | Get the model: `archivist::aread("pbiecek/models/08544")`. Get the explainer: `archivist::aread("pbiecek/models/87271")` |
| `titanic_svm_v6` | `e1071::svm` 1.7-2 | gender, age, class, sibsp, parch, fare, embarked | Get the model: `archivist::aread("pbiecek/models/be26e")`. Get the explainer: `archivist::aread("pbiecek/models/21966")` |

Table 5.2 summarizes the data frames that will be used in examples in the subsequent chapters.

Table 5.2: Data frames created for the `titanic` example.

| Description | No. rows | Variables | Link to this object |
|---|---|---|---|
| `titanic` dataset with imputed missing values | 2207 | gender, age, class, embarked, country, fare, sibsp, parch, survived | `archivist::aread("pbiecek/models/27e5c")` |
| `johny_d` 8-year-old boy that travelled in the 1st class without parents | 1 | class, gender, age, sibsp, parch, fare, embarked | `archivist::aread("pbiecek/models/e3596")` |
| `henry` 47-year-old male passenger from the 1st class, paid 25 pounds and embarked at Cherbourg | 1 | class, gender, age, sibsp, parch, fare, embarked | `archivist::aread("pbiecek/models/a6538")` |

## 5.2 Apartment prices

Warsaw skyscrapers by Artur Malinowski Flicker

Predicting house prices is a common exercise used in machine-learning courses. Various datasets for house prices are available at websites like Kaggle (https://www.kaggle.com) or UCI Machine Learning Repository (https://archive.ics.uci.edu).

In this book, we will work with an interesting variant of this problem. The `apartments` dataset is an artificial dataset created to match key characteristics of real apartments in Warsaw, the capital of Poland. However, the dataset is created in a way that two very different models, namely linear regression and random forest, have almost exactly the same accuracy. The natural question is then: which model should we choose? We will show that the model-explanation tools provide important insight into the key model characteristics and are helpful in model selection.

The dataset is available in the `DALEX` package (Biecek 2018). It contains 1000 observations (apartments) and six variables:

- *m2.price*, apartments price per meter-squared (in EUR), a numerical variable range 1607 – 6595;

- *construction.year*, the year of construction of the block of flats in which the apartment is located, a numerical variable range 1920 – 2010;
- *surface*, apartment's total surface in square meters, a numerical variable range 20 – 150;
- *floor*, the floor at which the apartment is located (ground floor taken to be the first floor), a numerical integer variable with values from 1 to 10;
- *no.rooms*, the total number of rooms, a numerical variable with values from 1 to 6;
- *district*, a factor with 10 levels indicating the district of Warsaw where the apartment is located.

The R code below provides more info about the contents of the dataset, values of the variables, etc.

```
library("DALEX")
head(apartments, 2)
```

```
##   m2.price construction.year surface floor no.rooms    district
## 1     5897              1953      25     3        1 Srodmiescie
## 2     1818              1992     143     9        5     Bielany
```

Models considered for this dataset will use *m2.price* as the (continuous) dependent variable.

Model predictions will be obtained for a set of six apartments included in data frame `apartments_test` .

```
head(apartments_test)
```

```
##      m2.price construction.year surface floor no.rooms    district
## 1001     4644              1976     131     3        5 Srodmiescie
## 1002     3082              1978     112     9        4     Mokotow
## 1003     2498              1958     100     7        4     Bielany
## 1004     2735              1951     112     3        5        Wola
## 1005     2781              1978     102     4        4      Bemowo
## 1006     2936              2001     116     7        4      Bemowo
```

## 5.2.1 Data exploration

Note that `apartments` is an artificial dataset created to illustrate and explain differences between random forest and linear regression. Hence, the structure of the data, the form and strength of association between variables, plausibility of distributional assumptions, etc., is better than in a real-life dataset. In fact, all these characteristics of the data are known. Nevertheless, we conduct some data exploration to illustrate the important aspects of the data.

The variable of interest is *m2.price*, the price per meter-squared. The histogram presented in Figure 5.6 indicates that the distribution of the variable is slightly skewed to the right.

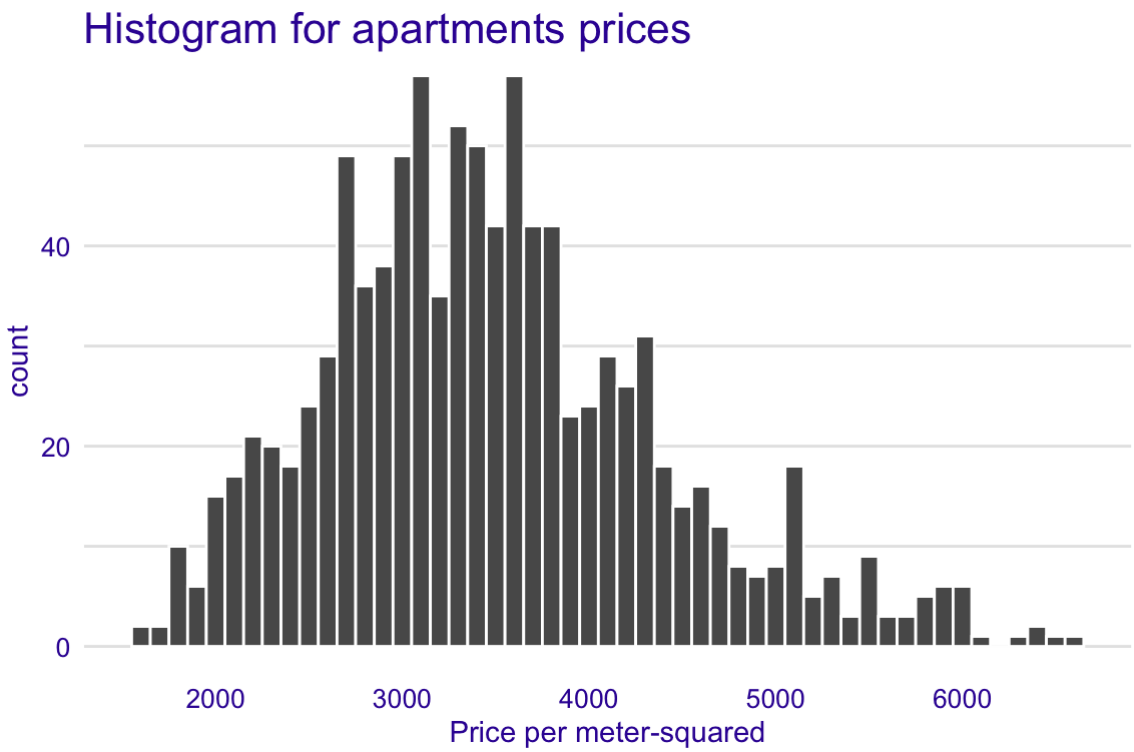



Figure 5.6: Distribution of the price per meter-squared in the apartments data.

Figure 5.7 suggests (possibly) a nonlinear relation between *construction.year* and *m2.price* and a linear relation between *surface* and *m2.price*.

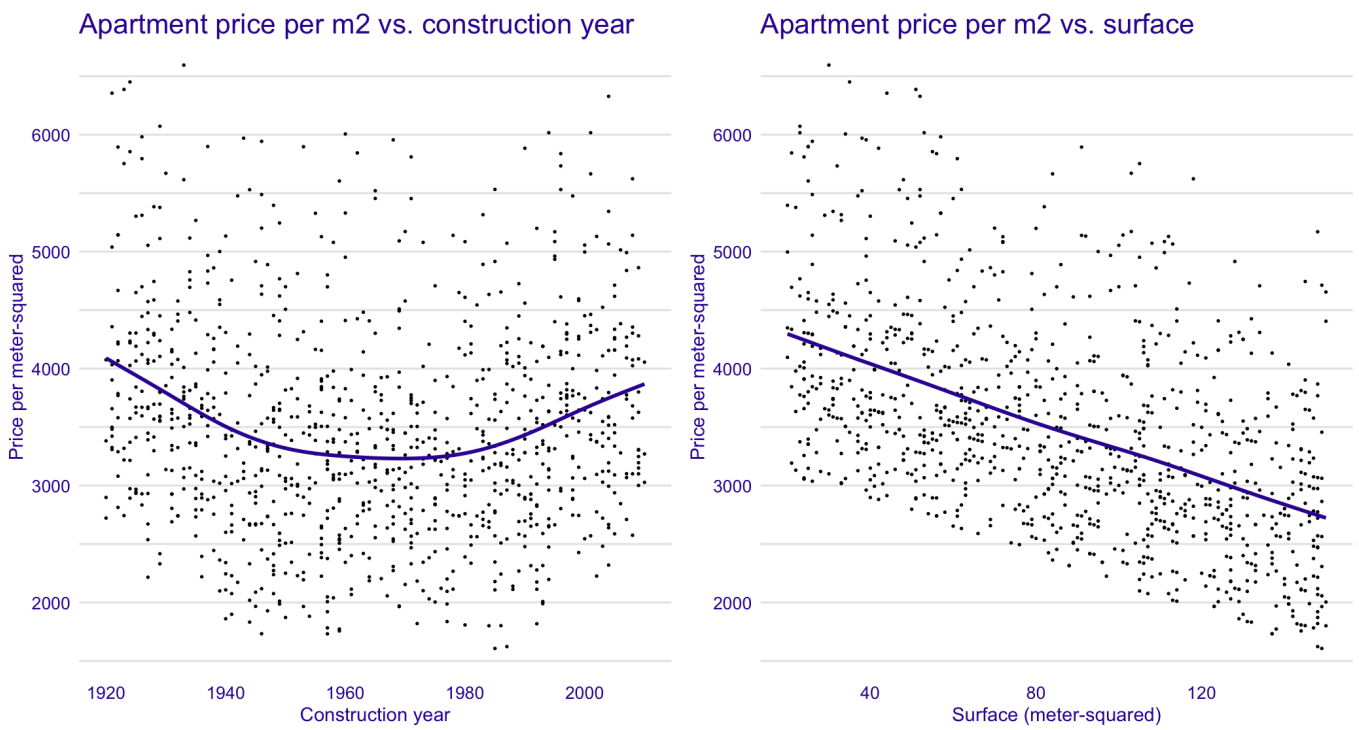



Figure 5.7: Left panel shows apartment price per m2 vs. year of construction, right panel shows price vs. square footage

Relation between *floor* and *m2.price* is also close to linear, as well as relation between *no.rooms* and *m2.price* as seen in Figure 5.8.

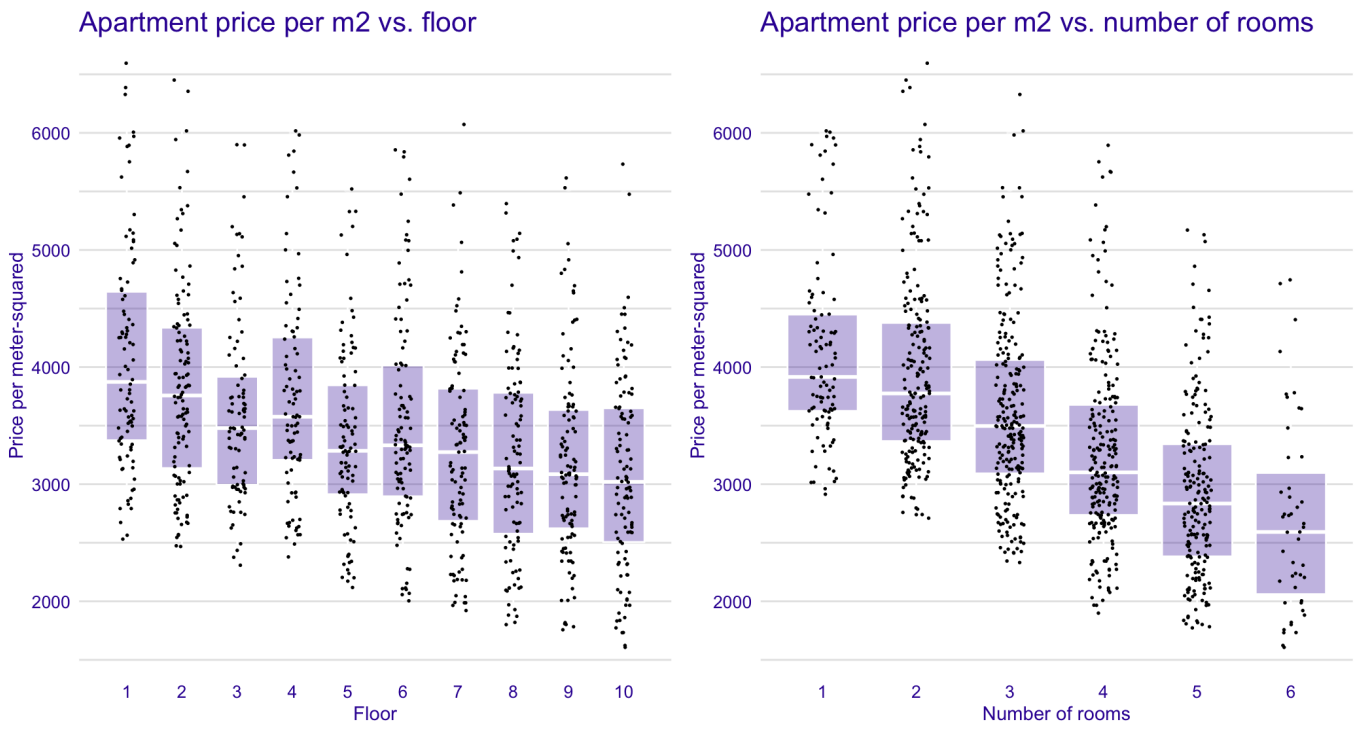


Figure 5.8: Price per meter-squared vs. floor and vs. number of rooms.

Surface and number of rooms are correlated and prices depend on district. Boxplots plots in Figure 5.9 indicate that the highest prices per meter-squared are observed in Srodmiescie (Downtown).

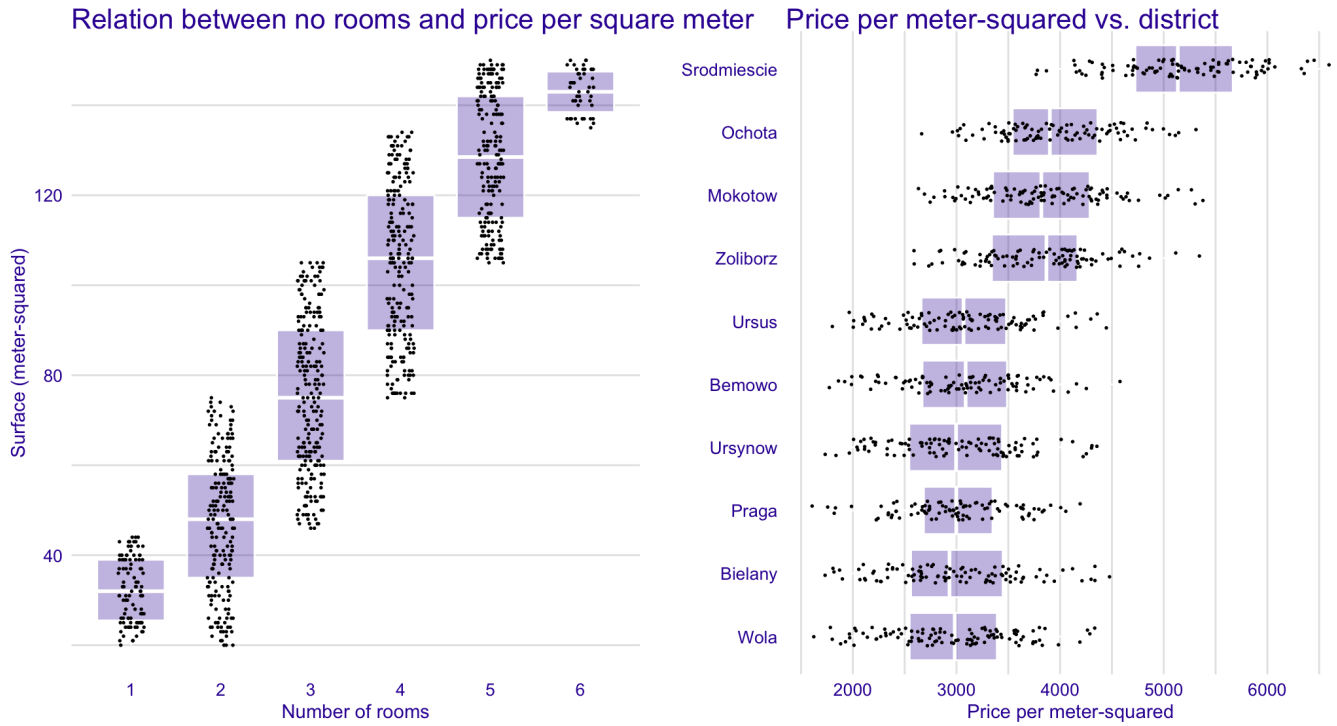



Figure 5.9: Left panel: surface vs. number of rooms. Right panel: price per meter-squared for different districts

## 5.2.2 Linear regression model

The dependent variable of interest, *m2.price*, is continuous. Thus, a natural choice to build a predictive model is the linear regression. We treat all the other variables in the `apartments` dataframe as explanatory and include them in the model. The results of the model are stored in model-object `apartments_lm_v5` .

```
apartments_lm_v5 <- lm(m2.price ~ ., data = apartments)
anova(apartments_lm_v5)
```

```
## Analysis of Variance Table
##
## Response: m2.price
##                    Df    Sum Sq   Mean Sq  F value    Pr(>F)
## construction.year   1   2629802   2629802   33.233 1.093e-08 ***
## surface             1 207840733 207840733 2626.541 < 2.2e-16 ***
## floor               1  79823027  79823027 1008.746 < 2.2e-16 ***
## no.rooms            1    956996    956996   12.094  0.000528 ***
## district            9 451993980  50221553  634.664 < 2.2e-16 ***
## Residuals         986  78023123     79131
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

### 5.2.3 Random forest model

As a challenger to linear regression, we consider a random forest model. To fit the model, we apply the `randomForest()` function, with default settings, from the package with the same name (Liaw and Wiener 2002).
The results of the model are stored in model-object `apartments_rf_v5` .

```
library("randomForest")
set.seed(72)
apartments_rf_v5 <- randomForest(m2.price ~ ., data = apartments)
apartments_rf_v5
```

```
##
## Call:
##  randomForest(formula = m2.price ~ ., data = apartments)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##           Mean of squared residuals: 79789.39
##                     % Var explained: 90.28
```

### 5.2.4 Support vector model

As an another challenger to the linear regression model, we consider a Support Vector Machines model. To fit the model, we use the `svm()` function, with default settings, from the package `e1071` (Meyer et al. 2017).
The results of the model are stored in model-object `apartments_svm_v5` .

```
library("e1071")
apartments_svm_v5 <- svm(m2.price ~ construction.year + surface + floor +
         no.rooms + district, data = apartments)
apartments_svm_v5
```

### 5.2.5 Model predictions

The `predict()` function calculates predictions for a specific model. In the example below we use model-object `apartments_lm_v5` to calculate predictions for prices for first six rows.

```
predict(apartments_lm_v5, apartments_test[1:6, ])
```

```
##     1001     1002     1003     1004     1005     1006
## 4820.009 3292.678 2717.910 2922.751 2974.086 2527.043
```

```
predict(apartments_rf_v5, apartments_test[1:6, ])
```

```
##     1001     1002     1003     1004     1005     1006
## 3399.854 2545.792 2695.787 2748.969 3682.974 3739.780
```

In the example below we calculate predictive performance for `apartments_lm_v5` and `apartments_rf_v5` as the square root of the average of squared errors (RMSE).

```
predicted_apartments_lm <- predict(apartments_lm_v5, apartments_test)
(rmsd_lm <- sqrt(mean((predicted_apartments_lm - apartments_test$m2.price)^2)))
```

```
## [1] 283.0865
```

```
predicted_apartments_rf <- predict(apartments_rf_v5, apartments_test)
(rmsd_rf <- sqrt(mean((predicted_apartments_rf - apartments_test$m2.price)^2)))
```

```
## [1] 795.2587
```

For the random forest model, the root-mean-square of the mean squared difference is equal to 795.3. It is almost identical as root-mean-square for the linear regression model 283.1. Thus, the question we may face is: should we choose the more complex, but flexible random-forest model, or the simpler and easier to interpret linear model? In the subsequent chapters we will try to provide an answer to this question.

As we will show in following chapters, a proper model exploration helps to understand which model we should choose. And even more, it helps to understand weak and strong sides of both models and in consequence we can create a new model better than these two.

### 5.2.6 Model adapters

In similar spirit to the Section 5.1.7 we will use explainers also for predictive models created for the `apartments` dataset.

```
explain_apartments_lm_v5 <- explain(model = apartments_lm_v5,
         data = apartments_test, y = apartments_test$m2.price,
         label = "Linear Regression")
explain_apartments_rf_v5 <- explain(model = apartments_rf_v5,
         data = apartments_test, y = apartments_test$m2.price,
         label = "Random Forest")
explain_apartments_svm_v5 <- explain(model = apartments_svm_v5,
         data = apartments_test, y = apartments_test$m2.price,
         label = "Support Vector Machines")
```

### 5.2.7 List of objects for the `apartments` example

In Sections 5.2.2 and 5.2.3 we have built two predictive models for the `apartments` data set. The models will be used in the rest of the book to illustrate the model explanation methods and tools.

For the ease of reference, we summarize the models in Table 5.3. The binary model-objects can be downloaded by using the indicated `archivist` hooks (Biecek and Kosinski 2017). By calling a function specified in the last column of the table, one can restore a selected model in a local R environment.

Table 5.3: Predictive models created for the `apartments` dataset.

| Model name | Model generator | Variables | Archivist hooks |
|---|---|---|---|
| `apartments_lm_v5` | `stats:: lm` v.3.5.3 | construction .year, surface, floor, no.rooms, district | Get the model: `archivist:: aread("pbiecek/models/55f19")` . Get the explainer: `archivist:: aread("pbiecek/models/78d4e")` |
| `apartments_rf_v5` | `randomForest:: randomForest` v.4.6.14 | construction .year, surface, floor, no.rooms, district | Get the model: `archivist:: aread("pbiecek/models/fe7a5")` . Get the explainer: `archivist:: aread("pbiecek/models/b1739")` |
| `apartments_svm_v5` | `e1071:: svm` v.1.7-2 | construction .year, surface, floor, no.rooms, district | Get the model: `archivist:: aread("pbiecek/models/545fa")` . Get the explainer: `archivist:: aread("pbiecek/models/16602")` |

# References

Biecek, Przemyslaw. 2018. *DALEX: Explainers for Complex Predictive Models in R*. *Journal of Machine Learning Research*. Vol. 19. http://jmlr.org/papers/v19/18-416.html.

Biecek, Przemyslaw, and Marcin Kosinski. 2017. “archivist: An R Package for Managing, Recording and Restoring Data Analysis Results.” *Journal of Statistical Software* 82 (11): 1–28. https://doi.org/10.18637/jss.v082.i11.

Breiman, Leo. 2001. “Random Forests.” In *Machine Learning*, 45:5–32. https://doi.org/10.1023/a:1010933404324.

Cortes, Corinna, and Vladimir Vapnik. 1995. “Support-Vector Networks.” In *Machine Learning*, 273–97.

Friedman, Jerome H. 2000. “Greedy Function Approximation: A Gradient Boosting Machine.” *Annals of Statistics* 29: 1189–1232.

Harrell Jr, Frank E. 2018. *Rms: Regression Modeling Strategies*. https://CRAN.R-project.org/package=rms.

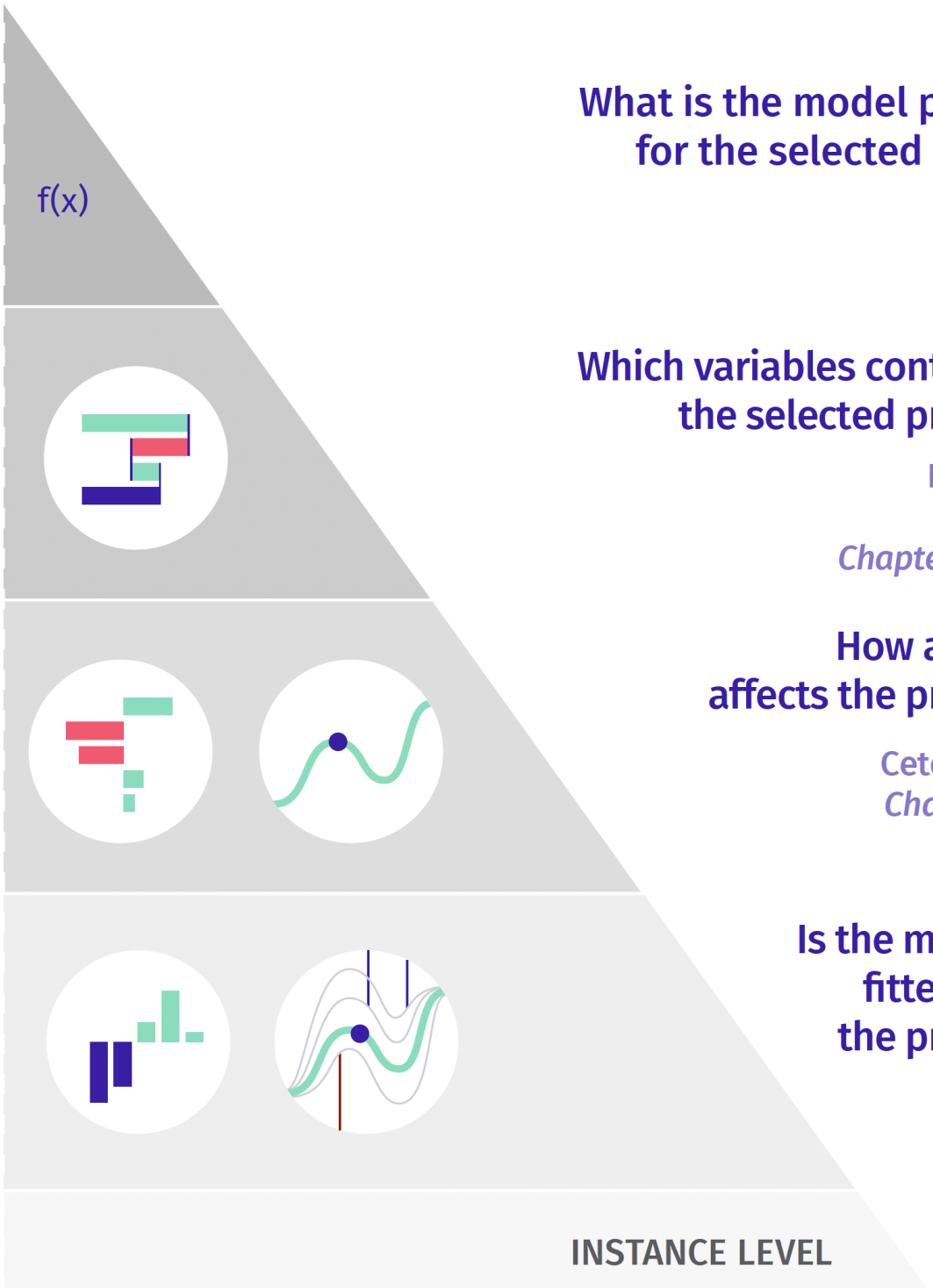
Liaw, Andy, and Matthew Wiener. 2002. “Classification and Regression by randomForest.” *R News* 2 (3): 18–22. http://CRAN.R-project.org/doc/Rnews/.

Meyer, David, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, and Friedrich Leisch. 2017. *E1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), Tu Wien*. https://CRAN.R-project.org/package=e1071.

Meyer, David, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, and Friedrich Leisch. 2019. *E1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), Tu Wien*. https://CRAN.R-project.org/package=e1071.

Ridgeway, Greg. 2017. *Gbm: Generalized Boosted Regression Models*. https://CRAN.R-project.org/package=gbm.

# Instance Level




**What is the model prediction for the selected instance?**

**Which variables contribute to the selected prediction?**

Break Down
SHAP, LIME
*Chapters 7, 8, 9, 10*

**How a variable affects the prediction?**

Ceteris paribus
*Chapters 11, 12*

**Is the model well fitted around the prediction?**

*Chapter 13*

# 6 Introduction to Instance Level Exploration

Instance-level methods help to understand how a model yields a prediction for a single observation. We can think about the following situations as examples:

- We may want to evaluate the effects of explanatory variables on model predictions. For instance, we may be interested in predicting the risk of heart attack based on person's age, sex, and smoking habits. A model may be used to construct a score (for instance, a linear combination of the explanatory variables representing age, sex, and smoking habits) that could be used for the purposes of prediction. For a particular patient we may want to learn how much the different variables contribute to the score of an individual patient?
- We may want to understand how models predictions would change if values of some of the explanatory variables changed. For instance, what would be the predicted risk of heart attack if the patient cut the number of cigarettes smoked per day by half?
- We may discover that the model is providing incorrect predictions and we may want to find the reason. For instance, a patient with a very low risk-score experienced a heart attack. What has driven the wrong prediction?

A model is a function with a $p$-dimensional vector $x$ as an argument. Instance level methods are designed to explore the model around a single point of interest $x^*$. In the following sections we will describe the most popular approaches to such exploration. They can be divided into three classes.

- One approach is to analyze how the model prediction for point $x^*$ is different from the average model prediction and how the difference can be distributed among explanatory variables. It is often called the ,,variable attributions" approach. An example is provided in panel A of Figure 6.1. Chapters 7-9 present various methods implementing this approach.
- Another approach is to analyze the curvature of the response surface around the point of interest $x^*$. Treating the model as a function, we are interested in the local behavior of this function around $x^*$. In case of a black-box model, we may approximate it with a simpler glass-box model around $x^*$. An example is provided in panel B of Figure 6.1. Chapter 10 presents the Local Interpretable Model-agnostic Explanations (LIME) method that exploits the concept of a ,,local model".

- Yet another approach is to investigate how the model prediction changes if the value of a single explanatory variable changes. The approach is useful in the so-called ,,What-If" analyses. In particular, we can construct plots presenting the change in model-based predictions induced by a change of a single variable. Such plots are usually called Ceteris-paribus (CP) profiles. An example is provided in panel C in Figure 6.1. Chapters 11-13 introduce the CP profiles and methods based on them.

Each method has its own merits and limitations. They are briefly discussed in the corresponding chapters. Chapter 14 offers a comparison of the methods.

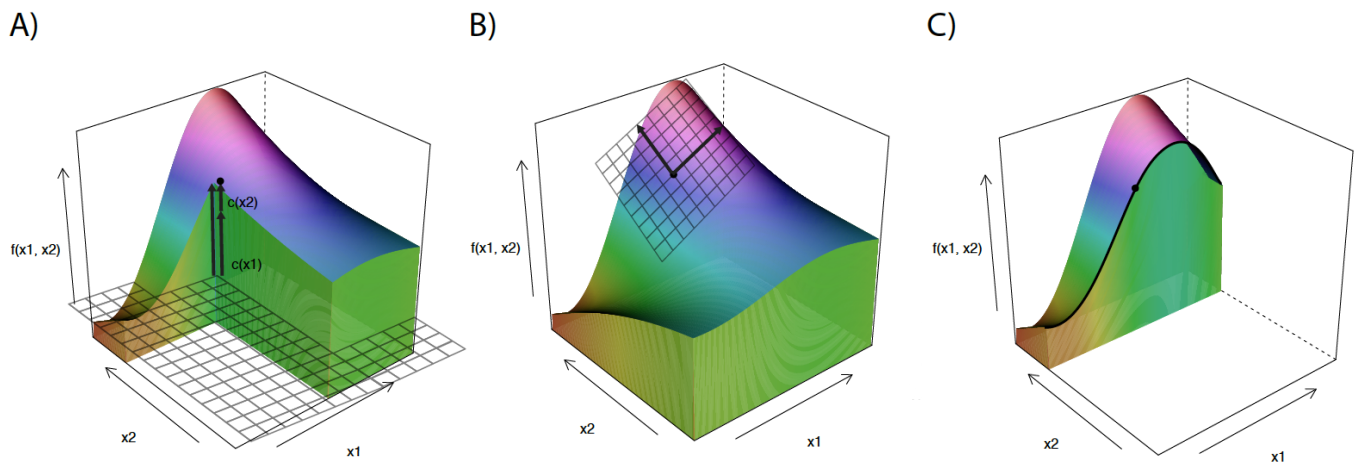



Figure 6.1: Response surface for a model that is a function of two variables. We are interested in understanding the response of a model in a single point x*. Illustration of different approaches to instance-level explanation. Panel A illustrates the concept of variable attributions like Break Down or SHAP. Additive effects of each variable show how the model response differs from the average. Panel B illustrates the concept of explanations through local models e.g. LIME. A simpler glass-box model is fitted around the point of interest. It describes the local behaviour of the black-box model. Panel C presents a What-If analysis with Ceteris-paribus profiles. The profiles show the model response as a function of a value of a single variable, while keeping the values of all other explanatory variables fixed.

# 7 Break-down Plots for Additive Attributions

Probably the most common question related to the explanation of model prediction for a single instance is: *which variables contributed to this result the most?*

Unfortunately, there is no silver bullet. Fortunately, there are some bullets. In this chapter we introduce Break-down (BD) plots, which offer a solution to this problem. Next two chapters are related to extensions of BD plot. Finally, Chapter 10 offer a different approach to this problem. The goal for BD plots is to show "variables attributions" i.e., the decomposition of the model prediction among explanatory variables.

## 7.1 Intuition

The underlying idea is to calculate contribution of an explanatory variable $x^i$ to model's prediction $f(x)$ as a shift in the expected model response after conditioning on other variables.

This idea is illustrated in Figure 7.1. Consider an example related to the prediction for the random-forest model `model_rf_v6` for Titanic data (see Section 5.1.3). We are interested in chances of survival for `johny_d` - an 8-years old passenger from first class. Panel A shows distribution of model predictions for all 2207 instances from dataset $X$. The row `all data` shows the vioplot of the predictions for the entire dataset. The red dot indicates the average and it is an estimate of the expected model prediction $E_X[f(X)]$ over the distribution of all explanatory variables. In this example the average model response is 23.5%.

To evaluate the contribution of the explanatory variables to the particular instance prediction, we trace changes in model predictions when fixing the values of consecutive variables. For instance, the row `class=1st` in Panel A of Figure 7.1 presents the distribution of the predictions obtained when the value of the `class` variable has been fixed to the `1st` class. Again, the red dot indicates the average of the predictions. The next row `age=8` shows the distribution and the average predictions with the value of variable `class` set to `1st` and

`age` set to `8`, and so on. With this procedure after $p$ steps every row in $X$ will be filled up with variable values of `johny_d`. All predictions for these rows will be equal, so the last row in the Figure corresponds to the prediction for `model response` for `johny_d`.

The thin black lines in Panel A show how the individual prediction for a single person changes after the value of the $j$-th variable has been replaced by the value indicated in the name of the row.

As we see from lines between first and the second row, the conditioning over `class=1st` has different effect on different instances. For some the model prediction has not changes (probably these passengers were already in the 1st class). For some the model prediction increase (probably they were in 2nd or 3rd class) while for other passenger the model prediction decreases (probably these were desk crew members).

Eventually, however, we may be interested in the average predictions, as indicated in Panel B of Figure 7.1, or even only in the changes of the averages, as shown in Panel C. In Panel C, positive changes are presented with green bars, while negative differences are marked with red bar. The changes sum up to the final prediction, which is illustrated by the violet bar at the bottom of Panel C.

What can be learned from Break-down plots? In this case we have concise summary of effects of particular variables on expected model response. First, we see that average model response is 23.5 percent. These are odds of survival averaged over all people on Titanic. Note that it is not the fraction of people that survived, but the average model response, so for different models one can get different averages. The model prediction for Johny D is 42.2 percent. It is much higher than an average prediction. Two variables that influence this prediction the most are class (=1st) and age (=8). Setting these two variables increase average model prediction by 33.5 percent points. Values in all other variables have rather negative effect. Low fare and being a male diminish odds of survival predicted by the model. Other variables do not change model predictions that much. Note that value of variable attribution depends on the value not only a variable itself. In this example the `embarked = Southampton` has small effect on average model prediction. It may be because the variable `embarked` is not important or it is possible that variable `embarked` is important but `Southampton` has an average effect out of all other possible values of the `embarked` variable.

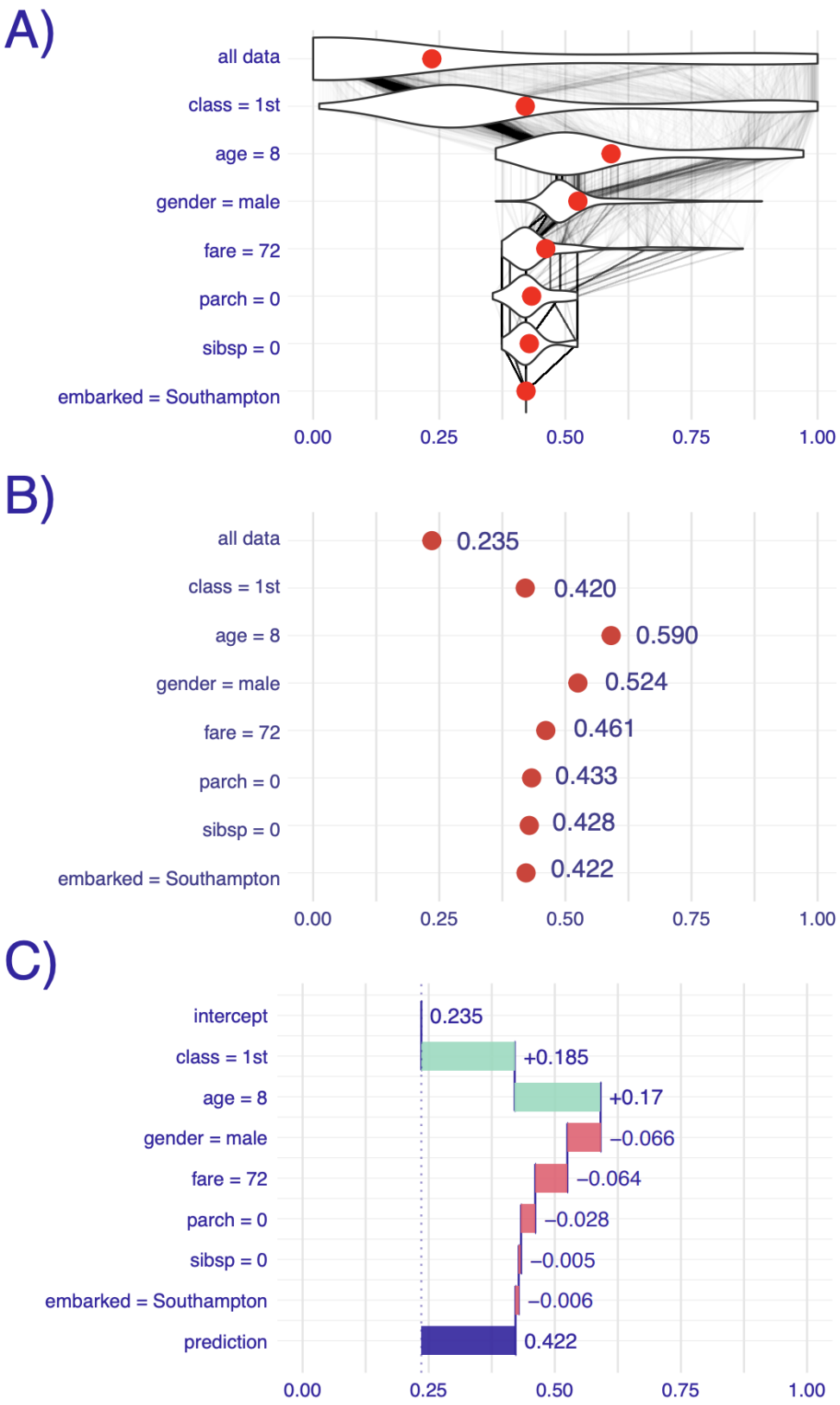



Figure 7.1: Break-down plots show how the contribution of individual explanatory variables change the average model prediction to the prediction for a single instance (observation). Panel A) The first row shows the distribution and the average (red dot) of model predictions for all data. The next rows show the distribution and the average of the predictions when fixing values of subsequent explanatory variables. The last row shows the prediction for a

particular instance of interest. B) Red dots indicate the average predictions from Panel A. C) The green and red bars indicate, respectively, positive and negative changes in the average predictions (variable contributions).

## 7.2 Method

First, let's see how variable attribution works for linear models. Because of the simple and additive structure of linear models it will be easier to build some intuitions.

### 7.2.1 Break-down for linear models

Assume a classical linear model for response $y$ with $p$ explanatory variables collected in the vector $X = (X^1, X^2, \ldots, X^p)$ and coefficients $\beta = (\beta^0, \beta^1, .., \beta^p)$, where $\beta^0$ is the intercept. The prediction for $y$ at point $x = (x^1, x^2, \ldots, x^p)$ is given by the expected value of $Y$ conditional on $X = x$. For a linear model, the expected value is given by the following linear combination:

$$E_Y(y|x) = f(x) = \beta^0 + x^1\beta^1 + \ldots + x^p\beta^p.$$

Now assume that we selected a single point from the input space $x_* \in \mathcal{R}^p$. We are interested in the contribution of the $i$-th explanatory variable to model prediction $f(x_*)$ for a single observation described by $x_*$. Because of additive structure of the linear model we expect that this contribution will be somehow linked to $x_*^i\beta^i$, because the $i$-th variable occurs only in this term. As it will become clear in the sequel, it is easier to interpret the variable's contribution if $x^i$ is centered by subtracting a constant $\hat{x}^i$ (usually, the mean of $x^i$). This leads the following, proposition for the variable attribution:

$$v(i, x_*) = \beta^i(x_*^i - \bar{x}^i). \tag{7.1}$$

Here $v(x_*, i)$ is the contribution of the $i$-th explanatory variable to the prediction of model $f()$ at point $x_*$. Assume that $E_Y(y|x_*) \approx f(x_*)$, where $f(x_*)$ is the value of the model at $x_*$. A possible approach to define $v(x_*, i)$ is to measure how much the expected model response changes after conditioning on $x_*^i$:

$$v(i, x_*) = E_Y(y|x_*) - E_{X^i}\{E_Y[y|(x_*^1, \ldots, x_*^{i-1}, X^i, x_*^{i+1}, x_*^p)]\} \approx f(x_*) - E_{X^i}[f(x_*^{-i})],$$

where $x_*^{-i}$ indicates that variable $X^i$ in vector $x_*$ is treated as random. For the classical linear model, if the explanatory variables are independent, $v(x_*, i)$ can be expressed as follows:

$$v(i, x_*) = f(x_*) - E_{X^i}[f(x_*^{-i})] = \beta^0 + x_*^1\beta^1 + \ldots + x_*^p\beta^p - E_{X^i}[\beta^0 + x_*^1\beta^1 + \ldots + \beta^i X^i \ldots$$

$$\ldots = \beta^i[x_*^i - E_{X^i}(X^i)].$$

In practice, given a dataset, the expected value of $X^i$ can be estimated by the sample mean $\bar{x}^i$. This leads to

$$v(i, x_*) = \beta^i(x_*^i - x^i).$$

Note that the linear-model-based prediction may be re-expressed in the following way:

$$f(x_*) = [\beta^0 + \bar{x}^1\beta^1 + \ldots + \bar{x}^p\beta^p] + [(x_*^1 - \bar{x}^1)\beta^1 + \ldots + (x_*^p - \bar{x}^p)\beta^p]$$

$$\equiv [average\ prediction] + \sum_{j=1}^{p} v(i, x_*). \tag{7.2}$$

Thus, the contributions of the explanatory variables $v(i, x_*)$ sum up to the difference between the model prediction for $x_*$ and the average model prediction.

**NOTE for careful readers**

Obviously, sample mean $\bar{x}^i$ is an estimator of the expected value $E_{X^i}(X^i)$, calculated using a training data. For the sake of simplicity we do not emphasize these differences in the notation. Also, we ignore the fact that, in practice, we never know the true model coefficients and we work with an estimated coefficients.

## 7.2.2 Break-down for a general case

Note that the method is similar to the `EXPLAIN` algorithm introduced in ,,Explaining Classifications for Individual Instances" (Robnik-Šikonja and Kononenko 2008) and implemented in the `ExplainPrediction` package (Robnik-Šikonja 2018).

Again, let $v(j, x_*)$ denote the variable-importance measure of the $j$-th variable and instance $x_*$, i.e., the contribution of the $j$-th variable to prediction at $x_*$.

We would like the sum of the $v(j, x_*)$ for all explanatory variables to be equal to the instance prediction (property called *local accuracy*), so that

$$f(x_*) = v_0 + \sum_{j=1}^{p} v(j, x_*), \tag{7.3}$$

where $v_0$ denotes the average model response. If we rewrite the equation above as follows:

$$E_X[f(X)|X^1 = x_*^1, \ldots, X^p = x_*^p] = E_X[f(X)] + \sum_{j=1}^{p} v(j, x_*),$$

then a natural proposal for $v(j, x_*)$ is

$$v(j, x_*) = E_X[f(X)|X^1 = x_*^1, \ldots, X^j = x_*^j] - E_X[f(X)|X^1 = x_*^1, \ldots, X^{j-1} = x_*^{j-1}]$$

In other words, the contribution of the $j$-th variable is the difference between the expected value of the prediction conditional on setting the values of the first $j$ variables equal to their values in $x_*$ and the expected value conditional on setting the values of the first $j-1$ variables equal to their values in $x_*$.

Note that the definition does imply the dependence of $v(j, x_*)$ on the order of the explanatory variables that is reflected in their indices.

To consider more general cases, let $J$ denote a subset of $K$ ($K \leq p$) indices from $\{1, 2, \ldots, p\}$, i.e., $J = \{j_1, j_2, \ldots, j_K\}$ where each $j_k \in \{1, 2, \ldots, p\}$. Furthermore, let $L$ denote another subset of $M$ ($M \leq p - K$) indices from $1, 2, \ldots, p$ distinct from $J$. That is, $L = \{l_1, l_2, \ldots, l_M\}$ where each $l_m \in \{1, 2, \ldots, p\}$ and $J \cap L = \emptyset$. Let us define now

$$\begin{aligned}\Delta^{L|J}(x_*) \equiv\ & E_X[f(X)|X^{l_1} = x_*^{l_1}, \ldots, X^{l_M} = x_*^{l_M}, X^{j_1} = x_*^{j_1}, \ldots, X^{j_K} = x_*^{j_K}] \\ -\ & E_X[f(X)|X^{j_1} = x_*^{j_1}, \ldots, X^{j_K} = x_*^{j_K}].\end{aligned}$$

In other words, $\Delta^{L|J}(x_*)$ is the change between the expected model prediction when setting the values of the explanatory variables with indices from the set $J \cup L$ equal to their values in $x_*$ and the expected prediction conditional on setting the values of the explanatory variables with indices from the set $J$ equal to their values in $x_*$.

In particular, for the $l$-th explanatory variable, let

$$\begin{aligned}\Delta^{l|J}(x_*) \equiv \Delta^{\{l\}|J}(x_*) =\ & E_X[f(X)|X^{j_1} = x_*^{j_1}, \ldots, X^{j_K} = x_*^{j_K}, X^l = x_*^l] \\ -\ & E_X[f(X)|X^{j_1} = x_*^{j_1}, \ldots, X^{j_K} = x_*^{j_K}].\end{aligned}$$

Thus, $\Delta^{l|J}$ is the change between the expected prediction when setting the values of the explanatory variables with indices from the set $J \cup \{l\}$ equal to their values in $x_*$ and the expected prediction conditional on setting the values of the explanatory variables with indices from the set $J$ equal to their values in $x_*$. Note that, if $J = \emptyset$, then

$$\Delta^{l|\emptyset}(x_*) = E_X[f(X)|X^l = x_*^l] - E_X[f(X)]. \tag{7.5}$$

It follows that

$$v(j, x_*) = \Delta^{j|\{1,\ldots,j-1\}}(x_*).$$

Unfortunately, for non-additive models (that include interactions), the value of so-defined variable-importance measure depends on the order, in which one sets the values of the explanatory variables. Figure 7.2 presents an example. We fit the random forest model to predict whether a passenger survived or not, then, we explain the model's prediction for a 2-year old boy that travels in the second class. The model predicts survival with a probability of $0.964$. We would like to explain this probability and understand which factors drive this prediction. Consider two explanations.

**Explanation 1:** The passenger is a boy, and this feature alone decreases the chances of survival. He traveled in the second class which also lower survival probability. Yet, he is very young, which makes odds higher. The reasoning behind such an explanation on this level is that most passengers in the second class are adults, therefore a kid from the second class has high chances of survival.

**Explanation 2:** The passenger is a boy, and this feature alone decreases survival probability. However, he is very young, therefore odds are higher than adult men. Explanation in the last step says that he traveled in the second class, which make odds of survival even more higher. The interpretation of this explanation is that most kids are from the third class and being a child in the second class should increase chances of survival.

Note that the effect of *the second class* is negative in explanations for scenario 1 but positive in explanations for scenario 2.

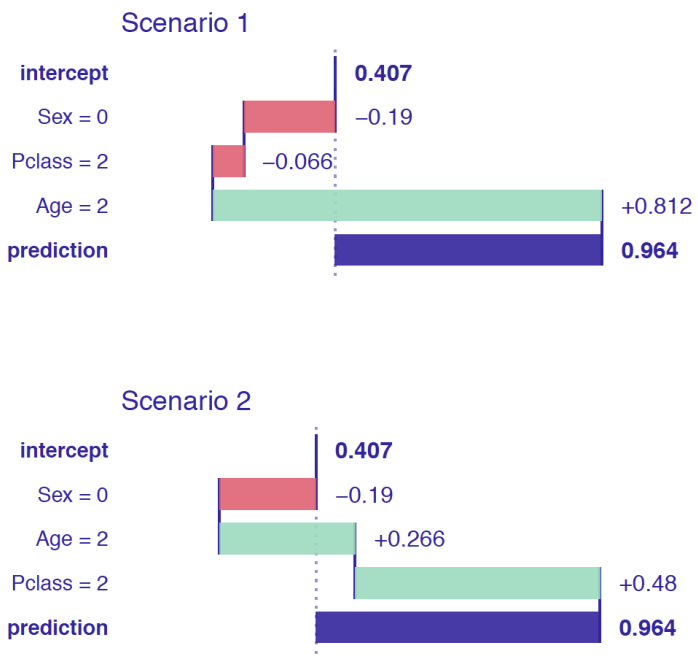



Figure 7.2: An illustration of the order-dependence of the variable-contribution values. Two *Break-down* explanations for the same observation from Titanic data set. The underlying model is a random forest. Scenarios differ due to the order of variables in *Break-down*

algorithm. Last bar indicates the difference between the model's prediction for a particular observation and an average model prediction. Other bars show contributions of variables. Red color means a negative effect on the survival probability, while green color means a positive effect. Order of variables on the y-axis corresponds to their sequence used in *Break-down* algorithm.

There are three approaches that can be used to address the issue of the dependence of $v(j, x_*)$ on the order, in which one sets the values of the explanatory variables.

In the first approach, one chooses an ordering according to which the variables with the largest contributions are selected first. In this chapter, we describe a heuristic behind this approach.

In the second approach, one identifies the interactions that cause a difference in variable-importance measure for different orderings and focuses on those interactions. This approach is discussed in Chapter 8.

Finally, one can calculate an average value of the variance-importance measure across all possible orderings. This approach is presented in Chapter 9.

To choose an ordering according to which the variables with the largest contributions are selected first, one can apply a two-step procedure. In the first step, the explanatory variables are ordered. In the second step, the conditioning is applied according to the chosen order of variables.

In the first step, the ordering is chosen based on the decreasing value of the scores equal to $|\Delta^{k|\emptyset}|$. Note that the absolute value is needed, because the variable contributions can be positive or negative. In the second step, the variable-importance measure for the $j$-th variable is calculated as

$$v(j, x_*) = \Delta^{j|J},$$

where

$$J = \{k : |\Delta^{k|\emptyset}| < |\Delta^{j|\emptyset}|\},$$

that is, $J$ is the set of indices of explanatory variables that have scores $|\Delta^{k|\emptyset}|$ smaller than the corresponding score for variable $j$.

The time complexity of each of the two steps of the procedure is $O(p)$, where $p$ is the number of explanatory variables.

## 7.3 Example: Titanic data

Let us consider the random-forest model `titanic_rf_v6` (see Section 5.1.3 and passenger `johny_d` (see Section 5.1.6) as the instance of interest in the Titanic data.

The average of model predictions for all passengers is equal to $v_0 = 0.2353095$. Table 7.1 presents the scores $|\Delta^{j|\emptyset}|$ and the expected values $E[f(X|X^j = x^j_*)]$. Note that $\Delta^{j|\emptyset} = E[f(X)|X^j = x^j_*] - v_0$ and, since for all variables $E[f(X)|X^j = x^j_*] > v_0$, we have got $E[f(X|X^j = x^j_*)] = |\Delta^{j|\emptyset}| + v_0$.

Table 7.1: Expected values $E[f(X)|X^j = x^j_*]$ and scores $|\Delta^{j|\emptyset}|$ for the random-forest model `titanic_rf_v6` for the Titanic data and `johny_d` . The scores are sorted in the decreasing order.

| **variable** $j$ | $E[f(X)\|X^j = x^j_*]$ | $\|\Delta^{j\|\emptyset}\|$ |
|---|---|---|
| age | 0.7407795 | 0.5051210 |
| class | 0.6561034 | 0.4204449 |
| fare | 0.6141968 | 0.3785383 |
| sibsp | 0.4786182 | 0.2429597 |
| parch | 0.4679240 | 0.2322655 |
| embarked | 0.4602620 | 0.2246035 |
| gender | 0.3459458 | 0.1102873 |

Based on the ordering defined by the scores $|\Delta^{j|\emptyset}|$ from Table 7.1, we can compute the variable-importance measures based on the sequential contributions $\Delta^{j|J}$. The computed values are presented in Table 7.2.

Table 7.2: Variable-importance measures $\Delta^{j|\{1,\ldots,j\}}$ for the random-forest model `titanic_rf_v6` for the Titanic data and `johny_d` computed by using the ordering of variables defined in Table 7.1.

| **variable** $j$ | $E[f(X)\|X^{\{1,\ldots,j\}} = x_*^{\{1,\ldots,j\}})]$ | $\Delta^{j\|\{1,\ldots,j\}}$ |
|---|---|---|
| intercept | 0.2353095 | 0.2353095 |
| age = 8 | 0.5051210 | 0.2698115 |
| class = 1st | 0.5906969 | 0.0855759 |
| fare = 72 | 0.5443561 | -0.0463407 |
| gender = male | 0.4611518 | -0.0832043 |
| embarked = Southampton | 0.4584422 | -0.0027096 |
| sibsp = 0 | 0.4523398 | -0.0061024 |
| parch = 0 | 0.4220000 | -0.0303398 |
| prediction | 0.4220000 | 0.4220000 |

Results from Table 7.2 are presented as a waterfall plot in Figure 7.3.

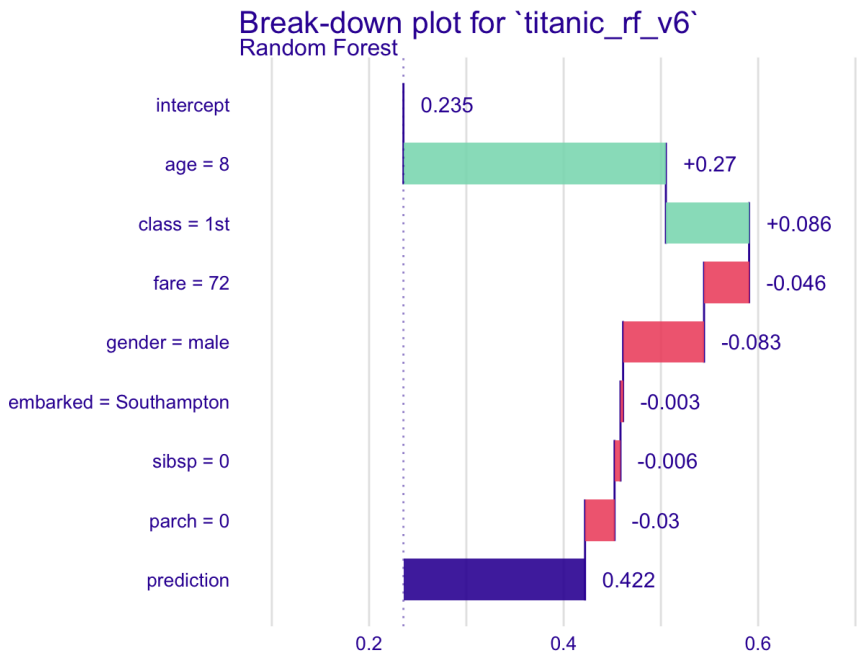



Figure 7.3: Break-down plot for the `titanic_rf_v6` model and `johny_d` for the Titanic data.

## 7.4 Pros and cons

Break-down plots offer a model-agnostic approach that can be applied to any predictive model that returns a single number for a single instance. The approach offers several advantages. The plots are easy to understand. They are compact; results for many variables may be presented in a small space. The approach reduces to an intuitive interpretation for the generalized-linear models. Numerical complexity of the Break-down algorithm is linear in the number of explanatory variables.

Break-down plots for non-additive models may be misleading, as they show only the additive contributions. An important issue is the choice of the ordering of the explanatory variables that is used in the calculation of the variable-importance measures. Also, for models with a large number of variables, the Break-down plot may be complex and include many variables with small contributions to the instance prediction.

## 7.5 Code snippets for R

In this section, we use an `DALEX::variable_attribution()` function which is a wrapper for `iBreakDown` R package (Gosiewska and Biecek 2019a). The package covers all methods presented in this chapter. It is available on CRAN and GitHub.

For illustration purposes, we use the `titanic_rf_v6` random-forest model for the Titanic data developed in Section 5.1.3. Recall that it is developed to predict the probability of survival from sinking of Titanic. Instance-level explanations are calculated for a single observation: `henry` - a 47-year-old passenger that travelled in the 1st class.

`DALEX` explainers for the model and the `henry` data are retrieved via `archivist` hooks as listed in Section 5.1.8.

```
library("randomForest")
explain_rf_v6 <- archivist::aread("pbiecek/models/6ed54")

library("DALEX")
henry <- archivist::aread("pbiecek/models/a6538")
henry
```

```
##   class gender age sibsp parch fare  embarked
## 1   1st   male  47     0     0   25 Cherbourg
```

### 7.5.1 Basic use of the `variable_attribution()` function

The `DALEX::variable_attribution()` function calculates the variable-importance measures for a selected model and the instance of interest. The result of applying the `variable_attribution()` function is a data frame containing the calculated measures. In the simplest call, the function requires only three arguments: the model explainer, the data frame for the instance of interest and the method for calculation of variable attribution, here `break_down`. The call below essentially re-creates the variable-importance values ($\Delta^{j|\{1,\ldots,j\}}$) presented in Table 7.2.

```
bd_rf <- variable_attribution(explain_rf_v6,
                 new_observation = henry,
                 type = "break_down")
```

Applying the generic `plot()` function to the object resulting from the application of the `variable_attribution()` function creates a BD plot. In this case, it is the plot from Figure 7.4.

```
plot(bd_rf)
```

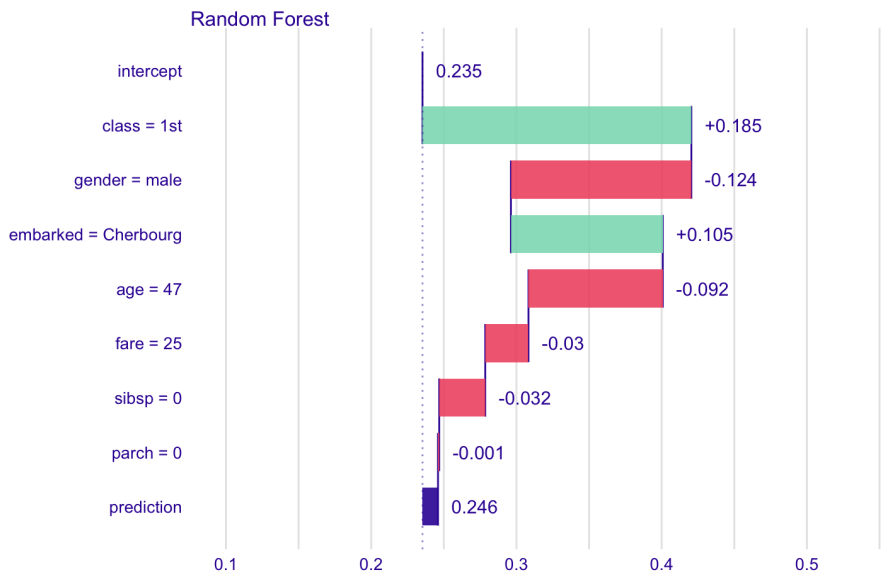



Figure 7.4: Generic plot() function for the BreakDown method calculated for `henry`.

Now we can compare contributions calculated for `johny_d` presented in Figure 7.3 with contributions calculated for `henry` presented in 7.4. Both explanations refer to the same model `model_rf_v6`. In both cases the `class=1st` increases chances of survival. For `johny_d` young age increases chances of survival while for `henry` the `age=47` decreases chances of survival.

### 7.5.2 Advanced use of the `variable_attribution()` function

The function `variable_attribution()` allows more arguments. The most commonly used are:

- `x` - a wrapper over a model created with function `DALEX::explain()`,
- `new_observation` - an observation to be explained is should be a data frame with structure that matches the training data,
- `order` - a vector of characters (column names) or integers (column indexes) that specify order of explanatory variables that is used for computing the variable-importance measures. If not specified (default), then a one-step heuristic is used to determine the order,
- `keep_distributions` - a logical value; if `TRUE`, then additional diagnostic information about conditional distributions is stored in the resulting object and can be plotted with the generic `plot()` function.

In what follows we illustrate the use of the arguments.

First, we will specify the ordering of the explanatory variables. Toward this end we can use integer indexes or variable names. The latter option is prerferable in most cases because of transparency. Additionally, to reduce clutter in the plot, we set `max_features = 3` argument in the `plot()` function.

```
bd_rf_order <- variable_attribution(explain_rf_v6,
         new_observation = henry, type = "break_down",
         order = c("class", "age", "gender", "fare", "parch",
                         "sibsp", "embarked"))

plot(bd_rf_order, max_features = 3)
```

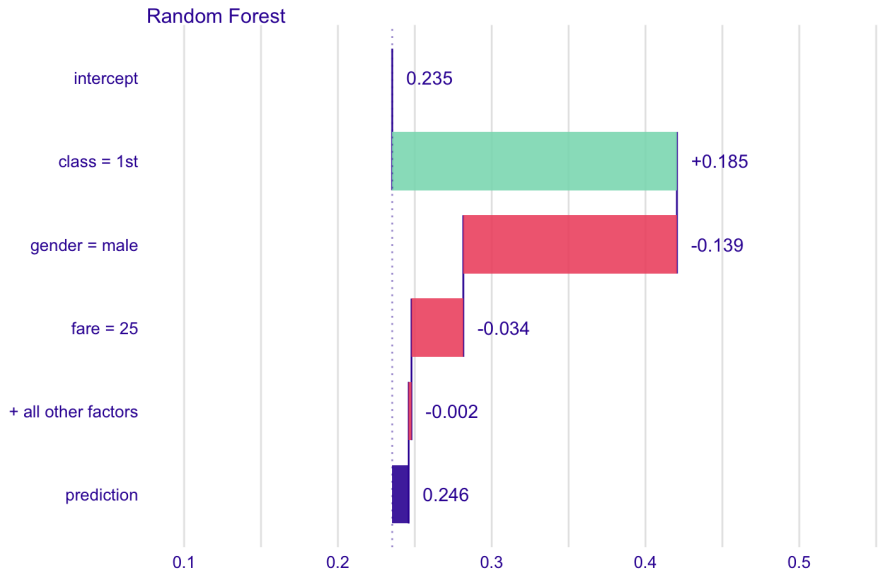



Figure 7.5: Break Down plot for top three variables.

We can use the `keep_distributions = TRUE` argument to enrich the resulting object with additional information about conditional distributions. Subsequently, we can apply the `plot_distributions = TRUE` argument in the `plot()` function to present the distributions as violin plots. Red dots in the plots indicate the average model predictions. Thin black lines between violin plots correspond to predictions for individual observations. They can be used to trace how model predictions change after consecutive conditionings.

```
bd_rf_distr <- variable_attribution(explain_rf_v6,
          new_observation = henry, type = "break_down",
          order = c("class", "age", "gender", "fare",
                          "parch", "sibsp", "embarked"),
          keep_distributions = TRUE)

plot(bd_rf_distr, plot_distributions = TRUE)
```

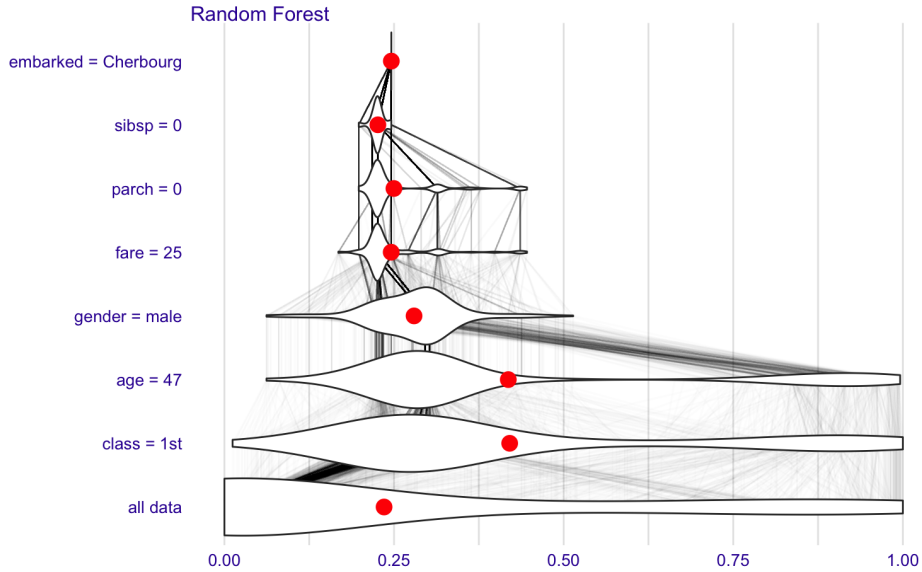



Figure 7.6: Break Down plot with distributions for a defined order of variables.

# References

Gosiewska, Alicja, and Przemyslaw Biecek. 2019a. “iBreakDown: Uncertainty of Model Explanations for Non-additive Predictive Models.” https://arxiv.org/abs/1903.11420v1.

Robnik-Šikonja, Marco, and Igor Kononenko. 2008. “Explaining Classifications for Individual Instances.” *IEEE Transactions on Knowledge and Data Engineering* 20 (5): 589–600. https://doi.org/10.1109/tkde.2007.190734.

Robnik-Šikonja, Marko. 2018. *ExplainPrediction: Explanation of Predictions for Classification and Regression Models*. https://CRAN.R-project.org/package=ExplainPrediction.

# 8 Break-down Plots for Interactions (iBreak-down Plots)

In Chapter 7, we presented a model-agnostic approach to evaluation of the importance of an explanatory variable for model predictions. An important issue is that, for some models, e.g. models with interactions, the estimated value of the variable-importance measure depends on the ordering of the explanatory variables that is used when computing the measure.

In this chapter, we present an algorithm that addresses the issue. In particular, the algorithm identifies interactions between pairs of variables and takes them into account when constructing Break-down (BD) plots. In our presentation we focus on interactions that involve pairs of explanatory variables, but the algorithm can be easily extended to interactions involving a larger number of variables.

## 8.1 Intuition

Lack of additivness means that the effect of an explanatory variable depends on the value(s) of other variable(s). To illustrate such a situation, we will consider the Titanic dataset (see Section 5.1). For the sake of simplicity, we consider only two variables, `age` and `class`. In the data `age` is a continuous variable, but we will use a dichotomized version of it, with two levels: boys (0-16 years old) and adults (17+ years old). Also, we will consider just two classes: the 2nd and "other".

Table 8.1 shows percentages of survivors for boys and adult men in the 2nd class and other classes on Titanic. Overall, the proportion of survivors among males is 20.5%. However, among boys in the 2nd class, the proportion is 91.7%. How do age and class contribute to this higher survival probability? Let us consider the following two decompositions.

- Decomposition 1: The overall probability of survival for males is 20.5%, but for the male passengers from the 2nd class the probability is even lower, i.e. 13.5%. Thus, the effect of the 2nd class is negative, as it decreases the probability of survival by 7%. Now, if, for male passengers of the 2nd class, we consider age, we see that the survival probability

for boys increases by 78.2%, from 13.5% (for a male in the 2nd class) to 91.7%. Thus, by considering first the effect of the class, and then the effect of age, we can conclude the effect of -7% for the 2nd class and +78.2% for age (being a boy).

- Decomposition 2: The overall probability of survival for males is 20.5%, but for boys the probability is higher, i.e., 40.7%. Thus, the effect of age (being a boy) is positive, as it increases the survival probability by 20.2%. On the other hand, for boys, travelling in the 2nd class increases the probability further, from 40.7% overall to 91.7%. Thus, by considering first the effect of age, and then the effect of class, we can conclude the effect of +20.2% for age (being a boy) and +51% for the 2nd class.

Table 8.1: Proportions of survivors for men on Titanic.

| Class | Boys (0-16) | Adults (>16) | Total |
|---|---|---|---|
| 2nd | 11/12 = 91.7% | 13/166 = 7.8% | 24/178 = 13.5% |
| other | 22/69 = 31.9% | 306/1469 = 20.8% | 328/1538 = 21.3% |
| Total | 33/81 = 40.7% | 319/1635 = 19.5% | 352/1716 = 20.5% |

By considering effects of class and age in different order, we get very different contributions. This is because there is an interaction: the effect of class depends on the age and `vice versa`. In particular, from Table 8.1 we could conclude that the overall effect of 2nd class is negative (-7%), as it decreases the probability of survival from 20.5% to 13.5%. On the other hand, the overall effect of age (being a boy) is positive (+20.2%), as it increases the probability of survival from 20.5% to 40.7%. Based on those effects, we would expect a probability of 20.5%-7%+20.2%=33.7% for a boy in the 2nd class. However, the actually observed proportion is much higher, 90.7%. The difference of 90.7%-33.7%=57% is the interaction effect. We can interpret it as an additional effect of the 2nd class specific for boys, or as an additional effect of age (being a boy) for the 2nd class male passengers.

The example illustrates that interactions complicate the evaluation of the importance of explanatory variables to model predictions. In what follows we present an algorithm to include interactions in the BD plots.

## 8.2 Method

Identification of interactions in the model is performed in three steps (Gosiewska and Biecek 2019a):

1. Calculate the variable-importance measure separately for each explanatory variable. In particular, for each variable, compute $\Delta^{j|\emptyset}(x_*)$ (see Section 7.2).
2. Calculate the measure for each pair of variables. Subtract the obtained value from the sum of the measures for the particular variables to obtain a contribution attributable to an interaction. In particular, for each pair of variables, compute $\Delta^{\{i,j\}|\emptyset}$ (see Section 7.2) and then

$$\Delta_I^{\{i,j\}}(x_*) \equiv \Delta^{\{i,j\}|\emptyset}(x_*) - \Delta^{i|\emptyset}(x_*) - \Delta^{j|\emptyset}(x_*). \tag{8.1}$$

3. Rank the so-obtained importance measures for the main" and interaction effects to determine the final ordering for computing the variable-importance measures. Using the ordering, compute variable-importance measures $v(j, x_*) = \Delta^{j|\{1,\ldots,j-1\}}(x_*)$ (see Section 7.2).

The time complexity of the first step is $O(p)$, where $p$ is the number of explanatory variables. For the second step, the complexity is $O(p^2)$, while for the third step it is $O(p)$. Thus, the time complexity of the entire procedure is $O(p^2)$.

## 8.3 Example: Titanic data

Let us consider the random-forest model `titanic_rf_v6` (see Section 5.1.3) and passenger `johny_d` (see Section 5.1.6) as the instance of interest in the Titanic data.

Table 8.2 presents the expected model predictions $E_X[f(X)|X^i = x_*^i, X^j = x_*^j]$, single-variable effects $\Delta^{\{i,j\}|\emptyset}(x_*)$ (see Equation (7.5)), and interaction effects $\Delta_I^{\{i,j\}}(x_*)$ (see Equation (8.1)) for each explanatory variable and each pair of variables. All the measures are calculated for `johny_d`, the instance of interest. The rows in the table are sorted according to the absolute value of the net impact of the variable or net impact of the interaction between two variables. For a single variable the net impact is simply measured by $\Delta^{\{i,j\}}(x_*)$ while for the pairs of variables the net impact is measured by $\Delta_I^{\{i,j\}}(x_*)$. This way if two variables are important but there is no interaction, then the net effect of interaction $\Delta_I^{\{i,j\}}(x_*)$ is smaller than additive effect of each variable and the interaction will be lower in the table, see `age` and `gender`. Contrary, is the interaction is important then its net effect will be higher than each variable separately, see `fare` and `class`.

Based on the ordering of the rows, the following sequence of variables is identified as informative:

- `age` because it has largest net effect $0.270$,
- then `fare:class` because the net effect of the interaction is $-0.231$,

- then `gender` because its net effect if $0.125$ and single variables like `class` or `fare` are already used in the interaction,
- then `embarked` because of its net effect $-0.011$,
- then `sibsp` , and `parch` as variables with lowest net effects but still larger than effect of their interaction.

Table 8.2: Expected model predictions $E_X[f(X)|X^i = x^i_*, X^j = x^j_*]$, single-variable effects $\Delta^{\{i,j\}|\emptyset}(x_*)$ (see Equation (7.5)), and interaction effects $\Delta_I^{\{i,j\}}(x_*)$ (see Equation (8.1)) for the random-forest model `titanic_rf_v6` and passenger `johny_d` in the Titanic data. The rows are sorted according to the absolute value of the net impact of the variable or net impact of the interaction between two variables. For a single variable the net impact is defined as $\Delta^{\{i,j\}}(x_*)$ while for the pairs of variables the net impact is equal to $\Delta_I^{\{i,j\}}(x_*)$.

| **Variable** | $E_X[f(X)\|X^i = x^i_*, X^j = x^j_*]$ | $\Delta^{\{i,j\}\|\emptyset}(x_*)$ | $\Delta_I^{\{i,j\}}(x_*)$ |
|---|---|---|---|
| age | 0.505 | 0.270 | |
| fare:class | 0.333 | 0.098 | -0.231 |
| class | 0.420 | 0.185 | |
| fare:age | 0.484 | 0.249 | -0.164 |
| fare | 0.379 | 0.143 | |
| gender | 0.110 | -0.125 | |
| age:class | 0.591 | 0.355 | -0.100 |
| age:gender | 0.451 | 0.215 | 0.070 |
| fare:gender | 0.280 | 0.045 | 0.027 |
| embarked | 0.225 | -0.011 | |
| embarked:age | 0.504 | 0.269 | 0.010 |
| parch:gender | 0.100 | -0.136 | -0.008 |
| sibsp | 0.243 | 0.008 | |
| sibsp:age | 0.520 | 0.284 | 0.007 |
| sibsp:class | 0.422 | 0.187 | -0.006 |
| embarked:fare | 0.374 | 0.138 | 0.006 |
| sibsp:gender | 0.113 | -0.123 | -0.005 |
| fare:parch | 0.380 | 0.145 | 0.005 |
| parch:sibsp | 0.236 | 0.001 | -0.004 |
| parch | 0.232 | -0.003 | |

| **Variable** | $E_X[f(X)\|X^i = x^i_*, X^j = x^j_*]$ | $\Delta^{\{i,j\}\|\emptyset}(x_*)$ | $\Delta_I^{\{i,j\}}(x_*)$ |
|---|---|---|---|
| parch:age | 0.500 | 0.264 | -0.002 |
| embarked:gender | 0.101 | -0.134 | 0.002 |
| embarked:parch | 0.223 | -0.012 | 0.001 |
| fare:sibsp | 0.387 | 0.152 | 0.001 |
| embarked:class | 0.409 | 0.173 | -0.001 |
| gender:class | 0.296 | 0.061 | 0.001 |
| embarked:sibsp | 0.233 | -0.002 | 0.001 |
| parch:class | 0.418 | 0.183 | 0.000 |

Table 8.3 presents the variable-importance measures computed by using the sequence of variables `age`, `fare:class`, `gender`, `embarked`, `sibsp`, and `parch`.

Table 8.3: Variable-importance measures $\Delta^{j|\{1,\ldots,j\}}(x_*)$ computed by using the sequence of variables `age`, `fare:class`, `gender`, `embarked`, `sibsp`, and `parch` for the random-forest model `titanic_rf_v6` for the Titanic data and `johny_d`.

| **Variable** | $\Delta^{j\|\{1,\ldots,j\}}(x_*)$ | $E_X[f(X)\|X^{\{1,\ldots,j\}} = x_*^{\{1,\ldots,j\}})]$ |
|---|---|---|
| intercept | | 0.235 |
| age = 8 | 0.269 | 0.505 |
| fare:class = 72:1st | 0.039 | 0.544 |
| gender = male | -0.083 | 0.461 |
| embarked = Southampton | -0.002 | 0.458 |
| sibsp = 0 | -0.006 | 0.452 |
| parch = 0 | -0.030 | 0.422 |

Figure 8.1 presents the BD plot corresponding to the results from Table 8.3.

As we see the interaction between `fare` and `class` is included in the plot as a single bar. As effects of these two variables cannot be disentangled, the plot shows combination of both variables as a single contribution. From Table 8.2 we can read that `class` alone would increase average prediction by $0.185$, `fare` would increase average prediction by $0.143$, but together they increase the average prediction only by $0.098$. It's because the `fare=72` is a

high value on average, but is below median when it comes for the 1st class passengers. So these two values combined `fare:class=72:1st` signal a cheaper version of the fist class, this is why its contribution to model prediction is smaller than contribution of `class` and `fare` separately.

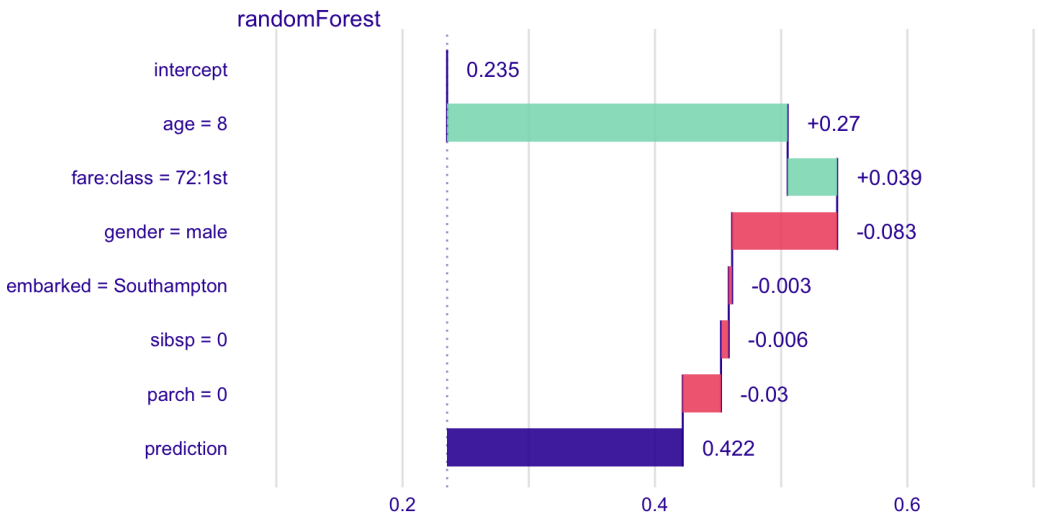



Figure 8.1: Break-down plot with interactions for the `titanic_rf_v6` model and `johny_d` for the Titanic data.

## 8.4 Pros and cons

iBD plots share many pros and cons of BD plots for models without interactions (see section 7.4). However, in case of interactions, the iBD plots provide more correct explanations.

Though the numerical complexity of the iBD procedure is quadratic, it may be time-consuming in case of models with a large number of explanatory variables. If $p$ stands for the number of variables, then we need to estimate $p * (p + 1)/2$ net effects for single variables and pair of variables. For datasets with small number of observations calculations of net effects will suffer from larger variance and therefore larger randomness in the ranking of effects. The identification of interactions in the presented procedure is not based on a formal statistical significance test. Thus, for small sample sizes, the procedure may be prone to errors.

## 8.5 Code snippets for R

In this section, we use an `DALEX()` package which is a wrapper for `iBreakDown` R package (Gosiewska and Biecek 2019a). The package covers all methods presented in this chapter. It is available on CRAN and GitHub.

For illustration purposes, we use the `titanic_rf_v6` random-forest model for the Titanic data developed in Section 5.1.3. Recall that it is developed to predict the probability of survival from sinking of Titanic. Instance-level explanations are calculated for a single observation: `henry` - an 47-years old passenger that travelled in the 1st class.

`DALEX` explainers for the model and the `henry` data are retrieved via `archivist` hooks as listed in Section 5.1.8.

```
library("randomForest")
explain_rf_v6 <- archivist::aread("pbiecek/models/6ed54")

johny_d <- archivist::aread("pbiecek/models/e3596")
henry
```

```
##   class gender age sibsp parch fare  embarked
## 1   1st   male  47     0     0   25 Cherbourg
```

The key function to construct iBD plots is the `DALEX::varaible_attribution()` function. The use of the function has already been explained in Section 7.5. In order to use Break-down plots the necessary argument is `type = "break_down_interactions"` .

```
library("DALEX")
bd_rf <- variable_attribution(explain_rf_v6,
                 new_observation = henry,
                 type = "break_down_interactions")
bd_rf
```

```
##                                                 contribution
## Random Forest: intercept                               0.235
## Random Forest: class = 1st                             0.185
## Random Forest: gender = male                          -0.124
## Random Forest: embarked:fare = Cherbourg:25            0.107
## Random Forest: age = 47                               -0.125
## Random Forest: sibsp = 0                              -0.032
## Random Forest: parch = 0                              -0.001
## Random Forest: prediction                              0.246
```

Now we can plot this object with the generic `plot()` function.

```
plot(bd_rf)
```

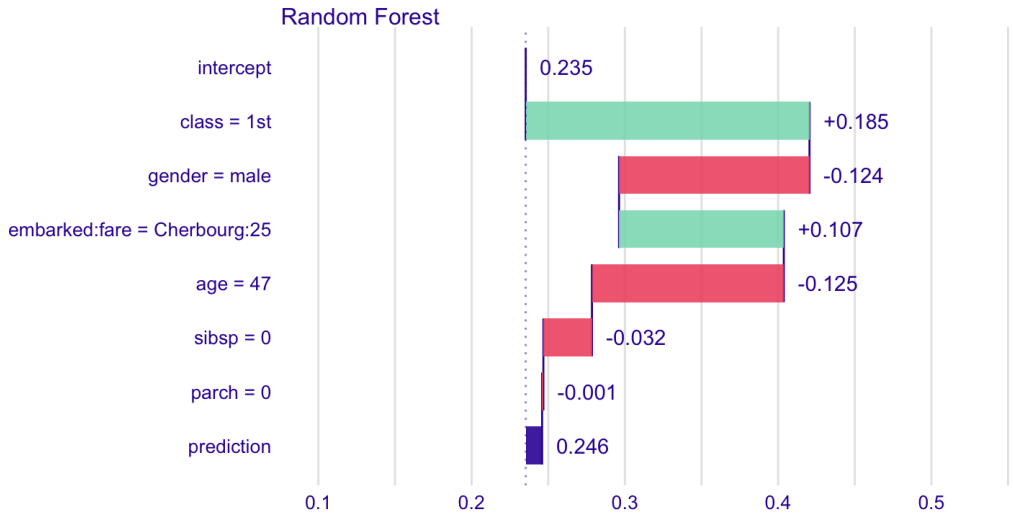



Figure 8.2: Generic plot() function for the iBreakDown method calculated for `henry`.

The Figure 8.2 shows iBD plot for `henry` while Figure 8.1 shows iBD plot for `johny_d`. In this case different variables were identified as an interaction. As `fare=25` for `henry` is much lower than `fare=72` for `johny_d` effect of `class` was not modified by `fare`.

# References

Gosiewska, Alicja, and Przemyslaw Biecek. 2019a. “iBreakDown: Uncertainty of Model Explanations for Non-additive Predictive Models.” https://arxiv.org/abs/1903.11420v1.

# 9 Shapley Additive Explanations (SHAP) and Average Variable Attributions

In Chapter 7, we introduced Break-down (BD) plots, a method of assessment of local variable-importance based on the contribution of an explanatory variable to model's prediction. We also indicated that, in the presence of interactions, the computed value of the contribution depends on the order of explanatory covariates that is used in calculations. One solution to the problem is to find an ordering in which the most important variables are placed at the beginning. Another solution, described in Chapter 8, is to identify interactions and explicitly present their contributions to the predictions.

In this chapter, we introduce yet another approach to address the ordering issue. It is based on the idea of averaging the value of a variable's contribution over all, or a large number of, possible orderings. The idea is closely linked to ,,Shapley values" (Shapley 1953), developed originally for cooperative games.

The approach was first introduced in ,,An Efficient Explanation of Individual Classifications Using Game Theory" (Štrumbelj and Kononenko 2010) and (Štrumbelj and Kononenko 2014). It was widely adopted after the publication of the NIPS paper ,,A Unified Approach to Interpreting Model Predictions" (Lundberg and Lee 2017) and Python's library SHAP (Lundberg 2019). The authors of SHAP (SHapley Additive exPlanations) introduced an efficient algorithm for tree-based models (Lundberg, Erion, and Lee 2018). They also showed that SHAP values can be presented an unification of a collection of different commonly used techniques for model explanations (Lundberg and Lee 2017).

## 9.1 Intuition

Figure 9.1 presents BD plots for ten random orderings (indicated by the order of the rows in each plot) of explanatory variables for the prediction for `johny_d` (see Section 5.1.6) for the random-forest model (see Section 5.1.3) for the Titanic dataset. The plots show clear differences in the contributions of various variables for different orderings. The most remarkable differences can be observed for variables `fare` and `class`, with contributions changing the sign depending on the ordering.

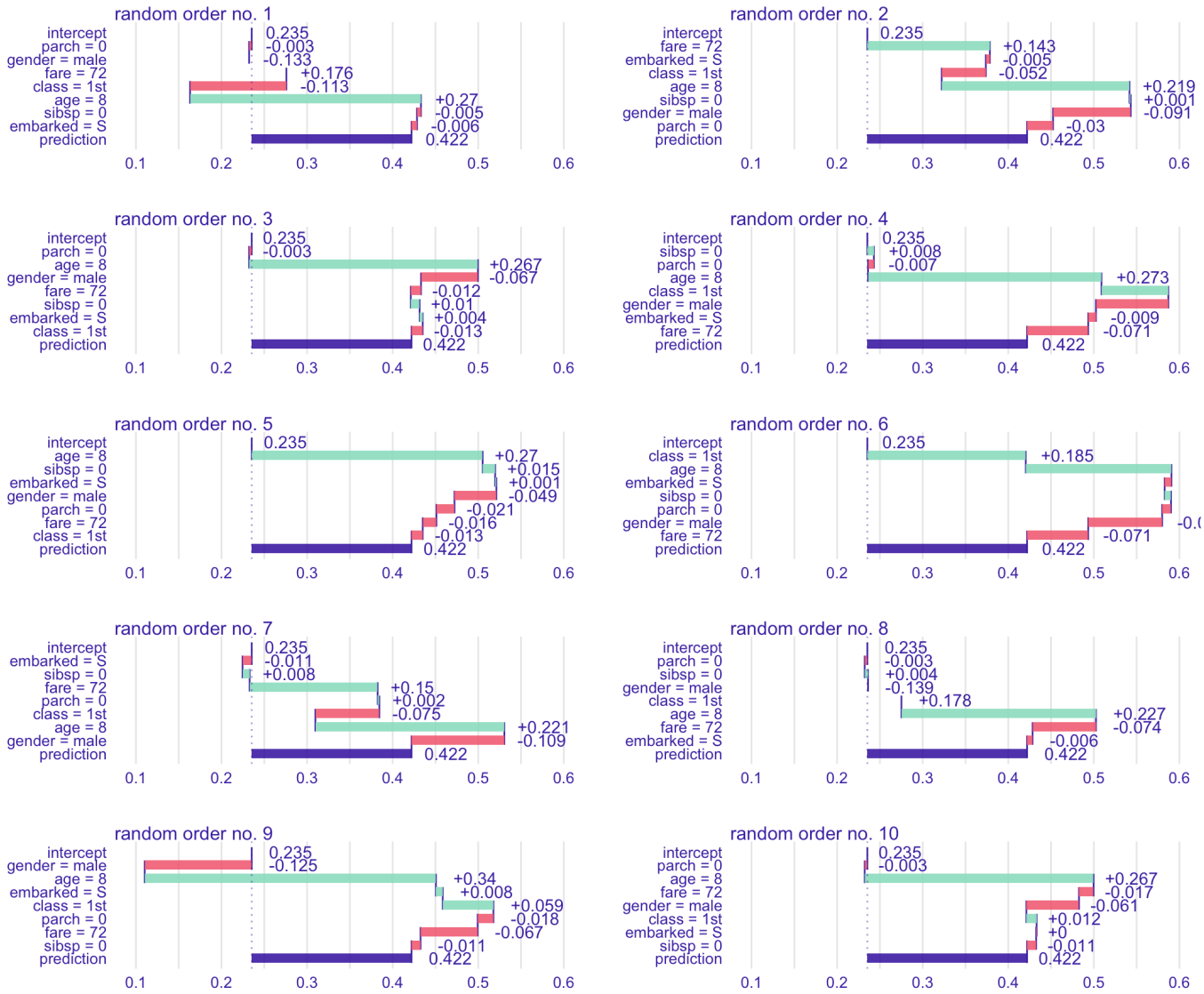



Figure 9.1: Break-down plots for ten random orderings of explanatory variables for the prediction for `johny_d` for the random-forest model for the Titanic dataset. Each panel presents a single ordering, indicated by the order of the rows in the plot.

To remove the influence of the ordering of the variables, we can compute an average value of the contributions. Figure 9.2 presents the average contributions, calculated over the ten orderings presented in Figure 9.1. Red and green bars present, respectively, the negative and positive averages. Violet box-plots summarize the distribution of the contributions for each explanatory variable across different orderings. The plot indicates that the most important variables, from the point of view of the prediction for `johny_d` are `age` and `gender`.

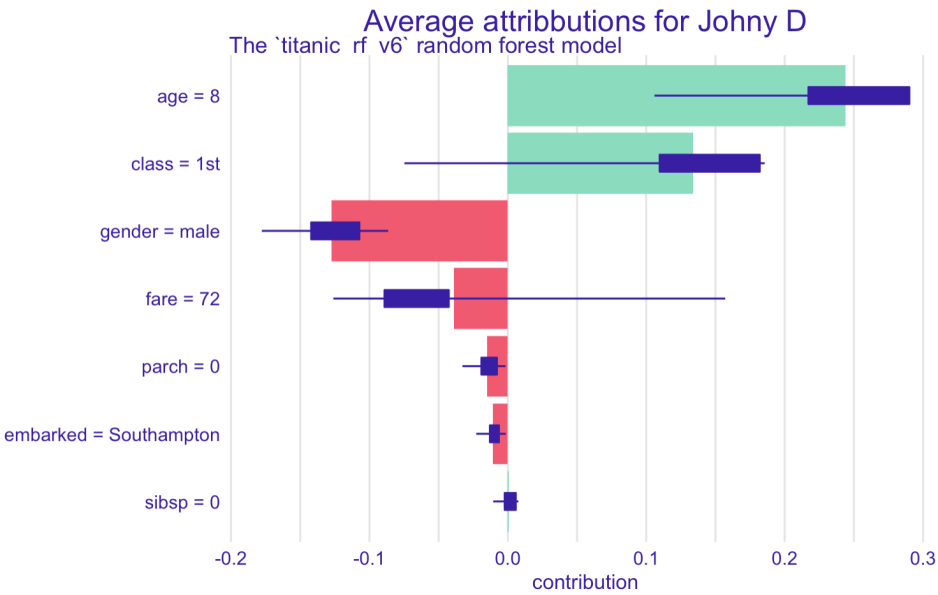



Figure 9.2: Average contributions for ten random orderings. Red and green bars present the averages. Box-plots summarize the distribution of contributions for each explanatory variable across the orderings.

## 9.2 Method

SHapley Additive exPlanations (SHAP) are based on ,,Shapley values," a concept in cooperative game theory developed by Lloyd Shapley (Shapley 1953). Note that the notation may be confusing at the first glance. Shapley values are introduced for cooperative games. SHAP is an acronym for a method designed for ML models. We will use the name Shapley values.

Consider the following problem. A coalition of players cooperates, and obtains a certain overall gain from the cooperation. Players are not identical, and different players may have different importance. Cooperation is beneficial, because it may bring more benefit than individual actions. The problem to solve is how to distribute the generated surplus among the players? The Shapley value provides one possible fair answer to this question (Shapley 1953).

Now let's translate this problem to the context of model predictions. Explanatory variables are the players, while model $f()$ plays the role of the coalition. The payoff from the coalition is the model prediction. The problem to solve is how to distribute the model prediction across particular variables?

The idea of using Shapley values for evaluation of local variable-importance was introduced in ,,An Efficient Explanation of Individual Classifications Using Game Theory" (Štrumbelj and Kononenko 2010). We define them here in the notation introduced in Section 7.2.

Let us consider a permutation $J$ of the set of indices $\{1, 2, \ldots, p\}$ corresponding to an ordering of $p$ explanatory variables included in model $f()$. Denote by $\pi(J, j)$ the set of the indices of the variables that are positioned in $J$ before the $j$-th variable. Note that, if the $j$-th variable is placed as the first, then $\pi(J, j) = \emptyset$. Consider the model prediction $f(x_*)$ for a particular instance of interest $x_*$. The Shapley value is defined as follows:

$$\varphi(x_*, j) = \frac{1}{p!} \sum_{J} \Delta^{j|\pi(J,j)}(x_*), \tag{9.1}$$

where the sum is taken over all $p!$ possible permutations (orderings of explanatory variables) and the variable-importance measure $\Delta^{j|J}(x_*)$ was defined in Section 7.2. Essentially, $\varphi(x_*, j)$ is the average of the variable-importance measures across all possible orderings of explanatory variables.

It is worth noting that the value of $\Delta^{j|\pi(J,j)}(x_*)$ is constant for all ordering $J$ that share with the same subset $\pi(J, j)$. It follows that equation (9.1) can be expressed in an alternative form:

$$\begin{aligned} \varphi(x_*, j) &= \frac{1}{p!} \sum_{s=0}^{p-1} \sum_{\substack{S \subseteq \{1,\ldots,p\} \setminus \{j\} \\ |S|=s}} \left[ s!(p-1-s)! \Delta^{j|S}(x_*) \right] \\ &= \frac{1}{p} \sum_{s=0}^{p-1} \sum_{\substack{S \subseteq \{1,\ldots,p\} \setminus \{j\} \\ |S|=s}} \left[ \binom{p-1}{s}^{-1} \Delta^{j|S}(x_*) \right], \end{aligned} \tag{9.2}$$

where $|S|$ denotes the cardinal number (size) of set $S$ and the second sum is taken over all subsets $S$ of explanatory variables, excluding the $j$-th one, of size $s$.

Note that the number of all subsets of sizes from 0 to $p-1$ is $2^p - 1$, i.e., it is much smaller than number of all permutations $p!$. Nevertheless, for a large $p$, it may not be feasible to compute the Shapley values from Equations (9.1) nor (9.2). In that case, an estimate based on a sample of permutations may be considered. A Monte Carlo estimator was introduced in ,,Explaining prediction models and individual predictions with feature contributions" (Štrumbelj and Kononenko 2014). An efficient implementation of computations of Shapley values was introduced in package SHAP (Lundberg and Lee 2017).

From the properties of Shapley values for cooperative games it follows that, in the context of predictive models, they enjoy the following properties:

- Symmetry: if two explanatory variables $j$ and $k$ are interchangeable, i.e., for any set of explanatory variables $S \subseteq \{1, \ldots, p\} \setminus \{j, k\}$ we have got

$$\Delta^{j|S}(x_*) = \Delta^{k|S}(x_*),$$

then their Shapley values are equal:

$$\varphi(x_*, j) = \varphi(x_*, k).$$

- Dummy feature: if an explanatory variable $j$ does not contribute to any prediction for any set of explanatory variables $S \subseteq \{1, \ldots, p\} \setminus \{j\}$, that is,

$$\Delta^{j|S}(x_*) = 0,$$

then its Shapley value is equal to 0:

$$\varphi(x_*, j) = 0.$$

- Additivity: if model $f()$ is a sum of two other models $g()$ and $h()$, then the Shapley value calculated for model $f()$ is a sum of Shapley values for models $g()$ and $h()$.

- Local accuracy: the sum of Shapley values is equal to the model prediction, that is,

$$f(x_*) - E_X[f(X)] = \sum_{j=1}^{p} \varphi(x_*, j).$$

## 9.3 Example: Titanic data

Let us consider the random-forest model `titanic_rf_v6` (see Section 5.1.3 and passenger `johny_d` (see Section 5.1.6) as the instance of interest in the Titanic data.

Box-plots in Figure 9.3 present the distribution of the contributions $\Delta^{j|\pi(J,j)}(x_*)$ for each explanatory variable of the model for 25 random orderings of the explanatory variables. Red and green bars represent, respectively, the negative and positive Shapley values across the orderings. It is clear that the young age of Johny D results in a positive contribution for all orderings. The Shapley value is equal to $0.2525$. On the other hand, the effect of gender is in all cases negative, with the Shapley value equal to $-0.0908$.

The picture for `fare` and `class` is more complex, as their contributions can even change the sign, depending on the ordering. While Figure 9.3 presents the Shapley values separately for each of the variables, it is worth noting that, by using the iBD plot in Section 8.3 the pair was identified as one for each an interaction effect was present. Hence, the effect of the variables should not be separated.

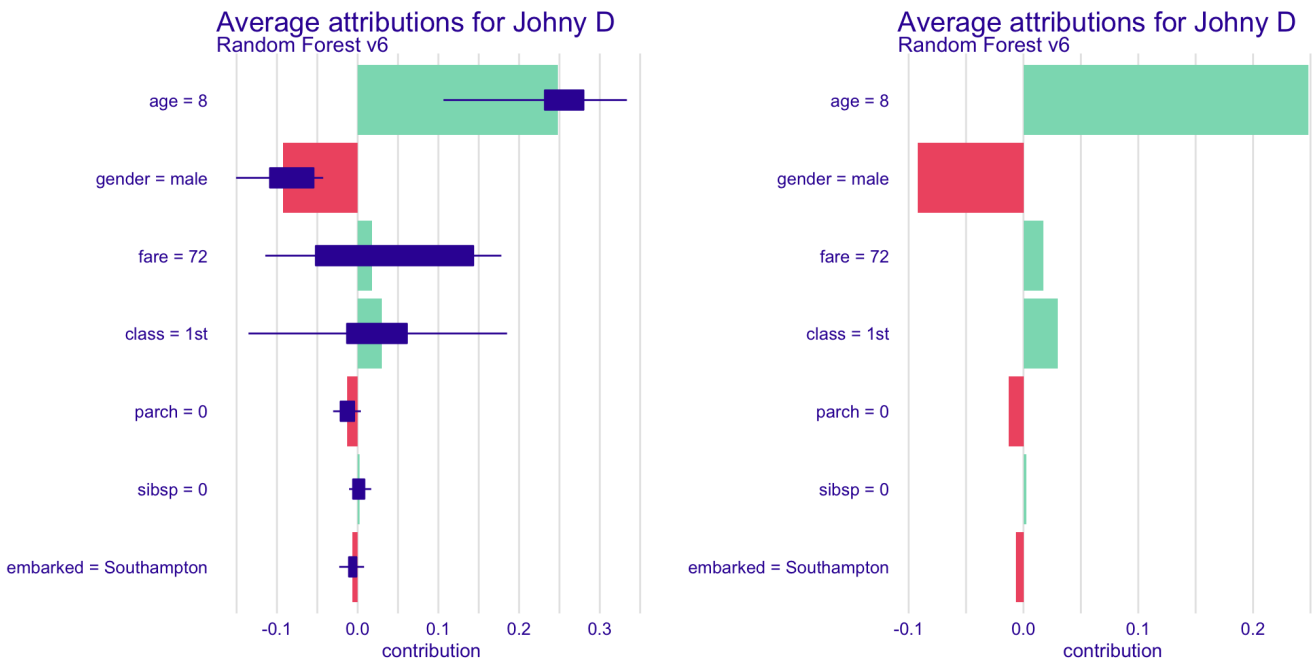



Figure 9.3: Variable contributions for the prediction for `johny_d` for the random-forest model `titanic_rf_v6` and the Titanic data for 25 random orderings. Left plot: Box-plots summarize the distribution of the contributions for each explanatory variable across the orderings. Red and green bars present the Shapley values. Right plot: Average attributions without boxplots.

In most applications the detailed information about the distribution of variable contributions across the considered orderings of explanatory variables will not be necessary. Thus, one could simplify the plot by presenting only the Shapley values, as in right panel in Figure 9.3. Table 9.1 presents the Shapley values underlying this plot.

Table 9.1: Shapley values for the prediction for `johny_d` for the random-forest model `titanic_rf_v6` and the Titanic data for 25 random orderings.

| Variable | Shapley value |
|---|---|
| age = 8 | 0.2525 |
| class = 1st | 0.0246 |
| embarked = Southampton | -0.0032 |
| fare = 72 | 0.0140 |
| gender = male | -0.0943 |
| parch = 0 | -0.0097 |
| sibsp = 0 | 0.0027 |

## 9.4 Pros and cons

Shapley values provide a uniform approach to decompose model predictions into parts that can be attributed additively to different explanatory variables. In ,,A Unified Approach to Interpreting Model Predictions" (Lundberg and Lee 2017) it is shown that the method unifies different approaches to additive features attribution, like DeepLIFT (Shrikumar, Greenside, and Kundaje 2017), Layer-Wise Relevance Propagation (Binder et al. 2016), or LIME (Ribeiro, Singh, and Guestrin 2016). The method has got a strong formal foundation derived from the cooperative games theory. It also enjoys an efficient implementation in Python, with ports or re-implementations in R.

An important drawback of the Shapley values is that they are additive attributions of variable effects. If the model is not additive, then the Shapley values may be misleading. This issue can be seen as arising from the fact that, in the cooperative games, the goal is to distribute the payoff among payers. However, in the predictive modeling context, we want to understand how do the players affect the payoff? Thus, we are not limited to independent payoff-splits for players.

It is worth noting that, for an additive model, the approaches presented in Chapters 7, 8, and in the current one lead to same variable contributions. It is because for additive models different orderings lead to same attributions. And since Shapley values can bee seen as an average across all ordering it's an average from identical values.

An important practical limitation of the method is that, for large models, the calculation of the Shapley values is time consuming. However, sub-sampling can be used to address the issue.

## 9.5 Code snippets for R

In this section, we use an `DALEX::variable_attribution()` function which is a wrapper for `iBreakDown` R package (Gosiewska and Biecek 2019a). The package covers all methods presented in this chapter. It is available on CRAN and GitHub. Note that there are also other R packages that offer similar functionality, like `shapper` (Gosiewska and Biecek 2019b), which is a wrapper for the Python library `SHAP` (Lundberg 2019), and `iml` (Molnar, Bischl, and Casalicchio 2018).

For illustration purposes, we use the `titanic_rf_v6` random-forest model for the Titanic data developed in Section 5.1.3. Recall that it is developed to predict the probability of survival from sinking of Titanic. Instance-level explanations are calculated for a single observation: `henry` - an 42-year-old passenger that travelled in the 1st class.

`DALEX` explainers for the model and the `jonhy_d` data are retrieved via `archivist` hooks as listed in Section 5.1.8.

```
library("randomForest")
explain_rf_v6 <- archivist::aread("pbiecek/models/6ed54")

library("DALEX")
henry <- archivist::aread("pbiecek/models/e3596")
henry
```

```
##   class gender age sibsp parch fare  embarked
## 1   1st   male  47     0     0   25 Cherbourg
```

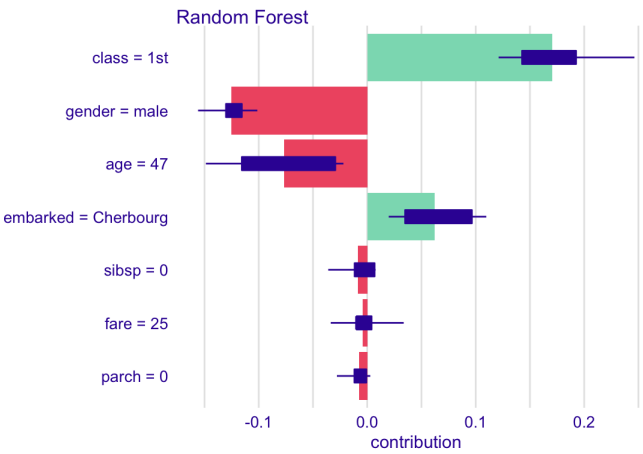
We obtain the model prediction for this instance with the help of the `predict()' function.

```
predict(explain_rf_v6, henry)
```

```
## [1] 0.246
```

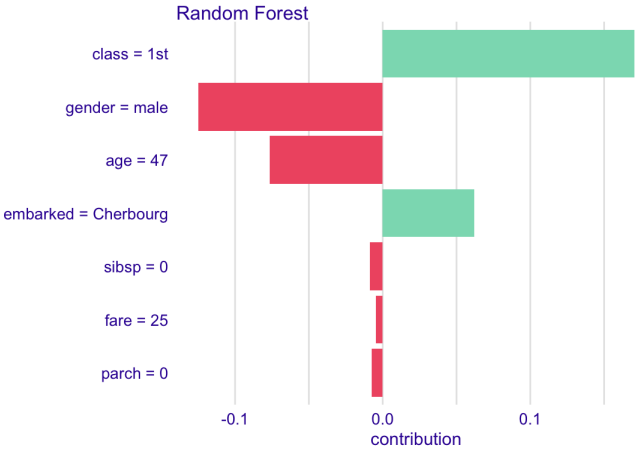
With the help of function `variable_attribution()` we can re-create Figure 9.3. The function is applied to the explainer, created with the `explain()` function from the `DALEX` package, and a data frame for the instance of interest. Additionally, in the `B=25` argument we indicate that we want to select 25 random orderings of explanatory variables for which the Shapley values are to be computed. The resulting object is a data frame with variable contributions computed for every ordering. Applying the generic function `plot()` to the object constructs the plot that includes the Shapley values and the corresponding box-plots.

```
shap_henry <- variable_attribution(explain_rf_v6,
                                   henry,
                                   type = "shap",
                                   B = 25)
plot(shap_henry)
```




To obtain a plot with only Shapley values, we can use the `show_boxplots=FALSE` argument in the `plot()` function call.

```
plot(shap_henry, show_boxplots = FALSE)
```



When we compare this plot with the `johny_d` in Figure 9.3 the largest difference is related to effect of `age`. Young `johny_d` has larger than average chances of survival, much larger than 47 years old `henry`.

The object obtained as a result of the application of function `shap()` allows to compute other summary statistics beyond the average.

```
shap_henry
```

```
##                                           min           q1       median
## Random Forest: age = 47             -0.14872225 -0.115577707 -0.077651998
## Random Forest: class = 1st           0.12112732  0.142754644  0.176491618
## Random Forest: embarked = Cherbourg  0.01981876  0.034941097  0.051973267
## Random Forest: fare = 25            -0.03364295 -0.009764386 -0.009519710
## Random Forest: gender = male        -0.15592478 -0.130189397 -0.125022202
## Random Forest: parch = 0            -0.02795650 -0.011817399 -0.005084730
## Random Forest: sibsp = 0            -0.03593203 -0.011618034 -0.006115541
##                                           mean            q3
## Random Forest: age = 47             -0.076695025 -0.0294435886
## Random Forest: class = 1st           0.170520888  0.1925099683
## Random Forest: embarked = Cherbourg  0.062216294  0.0964676031
## Random Forest: fare = 25            -0.004429615  0.0039927503
## Random Forest: gender = male        -0.125052324 -0.1155987766
## Random Forest: parch = 0            -0.007256366 -0.0008337109
## Random Forest: sibsp = 0            -0.008613321  0.0067818305
##                                            max
## Random Forest: age = 47             -0.021961033
## Random Forest: class = 1st           0.246304486
## Random Forest: embarked = Cherbourg  0.109760761
## Random Forest: fare = 25             0.033626643
## Random Forest: gender = male        -0.101295877
## Random Forest: parch = 0             0.002820118
## Random Forest: sibsp = 0             0.007650204
```

# References

Binder, Alexander, Grégoire Montavon, Sebastian Bach, Klaus-Robert Müller, and Wojciech Samek. 2016. “Layer-Wise Relevance Propagation for Neural Networks with Local Renormalization Layers.” *CoRR* abs/1604.00825. http://arxiv.org/abs/1604.00825.

Gosiewska, Alicja, and Przemyslaw Biecek. 2019a. “iBreakDown: Uncertainty of Model Explanations for Non-additive Predictive Models.” https://arxiv.org/abs/1903.11420v1.

Gosiewska, Alicja, and Przemyslaw Biecek. 2019b. *shapper: Wrapper of Python Library 'shap'*. https://github.com/ModelOriented/shapper.

Lundberg, Scott. 2019. *SHAP (SHapley Additive exPlanations)*. https://github.com/slundberg/shap.

Lundberg, Scott M., Gabriel G. Erion, and Su-In Lee. 2018. “Consistent Individualized Feature Attribution for Tree Ensembles.” *CoRR* abs/1802.03888. http://arxiv.org/abs/1802.03888.

Lundberg, Scott M, and Su-In Lee. 2017. “A Unified Approach to Interpreting Model Predictions.” In *Advances in Neural Information Processing Systems 30*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, 4765–74. Curran Associates, Inc. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

Molnar, Christoph, Bernd Bischl, and Giuseppe Casalicchio. 2018. “iml: An R package for Interpretable Machine Learning.” *Joss* 3 (26). Journal of Open Source Software: 786. https://doi.org/10.21105/joss.00786.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. 2016. “Why Should I Trust You?: Explaining the Predictions of Any Classifier.” In, 1135–44. ACM Press. https://doi.org/10.1145/2939672.2939778.

Shapley, Lloyd S. 1953. “A Value for n-Person Games.” In *Contributions to the Theory of Games Ii*, edited by Harold W. Kuhn and Albert W. Tucker, 307–17. Princeton: Princeton University Press.

Shrikumar, Avanti, Peyton Greenside, and Anshul Kundaje. 2017. “Learning Important Features Through Propagating Activation Differences.” *CoRR* abs/1704.02685. http://arxiv.org/abs/1704.02685.

Štrumbelj, Erik, and Igor Kononenko. 2010. “An Efficient Explanation of Individual Classifications Using Game Theory.” *Journal of Machine Learning Research* 11 (March). JMLR.org: 1–18. http://dl.acm.org/citation.cfm?id=1756006.1756007.

Štrumbelj, Erik, and Igor Kononenko. 2014. “Explaining prediction models and individual predictions with feature contributions.” *Knowledge and Information Systems* 41 (3): 647–65. https://doi.org/10.1007/s10115-013-0679-x.

# 10 Local Interpretable Model-agnostic Explanations (LIME)

## 10.1 Introduction

Break-down (BD) and Shapley plots, introduced in Chapters 7 and 9, respectively, are most suitable for models with a small or moderate number of explanatory variables.

None of those approaches is well-suited for models with a very large number of explanatory variables. In genomics or image recognition, models with hundreds of thousands or millions of input variables are not uncommon. In such cases, sparse explainers with small number of non zero effects offer a useful alternative. The most popular example of such sparse explainers are Local Interpretable Model-agnostic Explanations (LIME) and their modifications.

The LIME method was originally proposed in ,,Why Should I Trust You?: Explaining the Predictions of Any Classifier" (Ribeiro, Singh, and Guestrin 2016). The key idea behind this method is to locally approximate a black-box model by a simpler glass-box model, which is easier to interpret. In this chapter, we describe this approach.

## 10.2 Intuition

The intuition behind the LIME method is explained in Figure 10.1. We want to understand factors that influence a complex black-box model around a single instance of interest. Areas presented in Figure 10.1 correspond to decision regions for a binary classifier, i.e., it pertains to a binary dependent variable. The axes represent the values of two continuous explanatory variables. The colored areas correspond to the decision regions, i.e., they indicate for which combinations of the variables the model classifies the observation to one of the two classes. The instance of interest is marked with the large black dot. By using an artificial dataset around the instance of interest, we can use a simpler glass-box model that will locally approximate the predictions of the black-box model. The glass-box model may then serve as a ''local explainer" for the more complex model.

We may select different classes of glass-box models. The most typical choices are regularized linear models like LASSO regression (Tibshirani 1994) or decision trees (Hothorn, Hornik, and Zeileis 2006). The important point is to limit the complexity of the models, so that they are easier to explain.

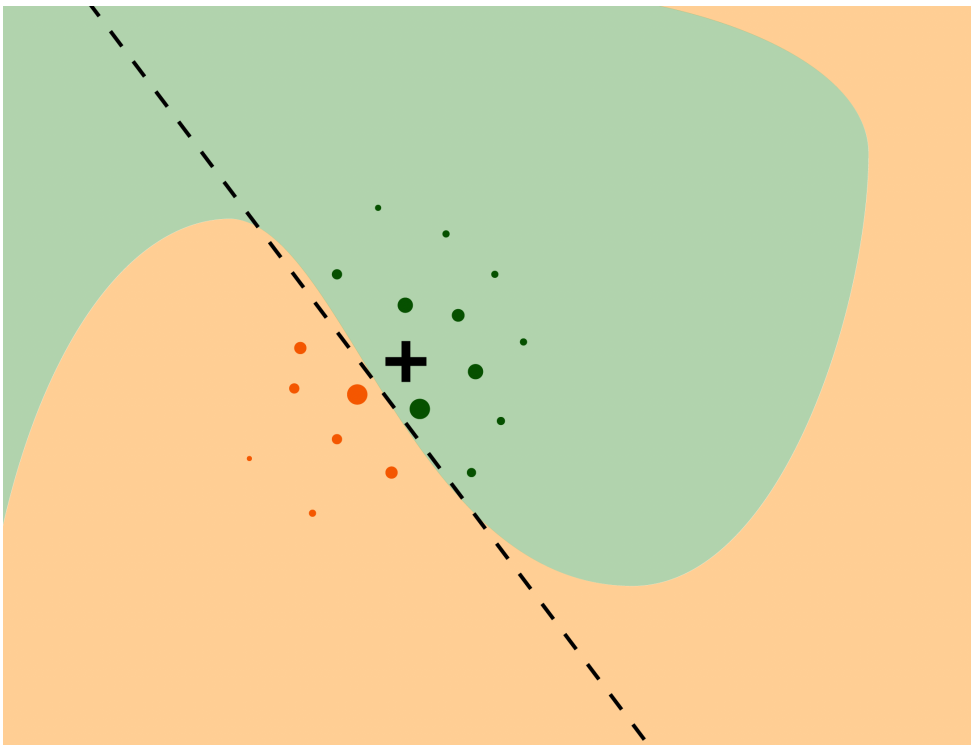


Figure 10.1: The idea behind LIME approximation with local glass-box model. The colored areas correspond to decision regions for a complex binary classification model. The black cross corresponds to the instance of interest x*. Small dots correspond to the generated new data. Size of dots corresponds to proximity to the instance of interest, i.e. to weights w'. Dashed line correspond to a simple linear model fitted for the artificial data. It approximates the black box model around the instance of interest. The simple linear model ,,explains" local behaviour of the black box model.

## 10.3 Method

As an explanation, we want to find a model that locally approximates a black-box model $f()$ around the instance of interest $x_*$. Consider class $G$ of interpretable models (linear models or decision trees). To find the required approximation, we consider the following ,,loss function"

$$\hat{g} = \arg\min_{g \in G} L(f, g, \Pi_{x_*}) + \Omega(g),$$

where model $g()$ belongs to class $G$, $\Pi_{x_*}$ defines a neighborhood of $x_*$ in which approximation is sought, $L()$ is a fidelity measure between models $f()$ and $g()$, and $\Omega(g)$ is a penalty for the complexity of model $g()$. The penalty is used to select simple models from class $G$.

Note that the models $f()$ and $g()$ may operate on different variable spaces. The black-box model (function) $f(x) : \mathcal{X} \to \mathcal{R}$ is defined on the original, large, p-dimensional space $\mathcal{X}$. The glass-box model (function) $g : \mathcal{X}' \to \mathcal{R}$ applies to a lower q-dimensional, interpretable space $\mathcal{X}'$, and usually $q << p$. We will present some examples of $\mathcal{X}'$ in the next section. For now we will just assume that some function $h()$ transforms $\mathcal{X}$ into $\mathcal{X}'$.

If we limit class $G$ to sparse linear models with $K$ non zero coefficients, the following algorithm may be used to find an interpretable glass-box model $g()$ that includes $K$ most important, interpretable explanatory variables:

```
Input: x* - observation to be explained
Input: N  - sample size for the glass-box model
Input: K  - complexity, number of variables for the glass-box model
Input: similarity - distance function in the original input space
1. Let x' = h(x*) be a version of x* in the interpretable space
2. for i in 1...N {
3.   z'[i] <- sample_around(x')
     # prediction for a new observation z'[i]
4.   y'[i] <- f(z[i])
5.   w'[i] <- similarity(x', z'[i])
6. }
7. return K-LASSO(y', x', w')
```

In Step 7, $K - LASSO(y', x', w')$ stands for a weighted LASSO linear-regression that selects $K$ variables based on new dataset $(y', x')$ with weights $w'$.

The practical implementation of this idea involves three important steps, which are discussed in the subsequent subsections.

### 10.3.1 Interpretable data representation

As it has been mentioned, the black-box model $f()$ and the glass-box model $g()$ operates on different data spaces. For example, let's consider a VGG16 neural network (Simonyan and Zisserman 2015) trained for ImageNet data (Deng et al. 2009). The model uses an image of

the size of $244 \times 244$ pixels as input and predicts to which of 1000 potential categories does the image belong to. The original data space is of dimension $3 \times 244 \times 244$ (three single-color channels *red, green, blue* for a single pixel $\times 244 \times 244$ pixels), i.e., the input space is 178,608-dimensional. Explaining predictions in such a high-dimensional space is difficult. Instead, the space can be transformed into superpixels, which are treated as binary features that can be turned on or off. Figure 10.2 presents an example of 100 superpixels created for an ambiguous picture. Thus, in this case the black-box model $f()$ operates in principle on data space $\mathcal{X} = R^{178,608}$, while the glass-box model $g()$ works on space $\mathcal{X}' = \{0,1\}^{100}$.

It is worth noting that superpixels are frequent choices for image data. For text data, words are frequently used as interpretable variables. To reduce to complexity of the data space, continuous variables are often discretized to obtain interpretable tabular data. In case of categorical variables, combination of categories is often used. We will present examples in the next section.

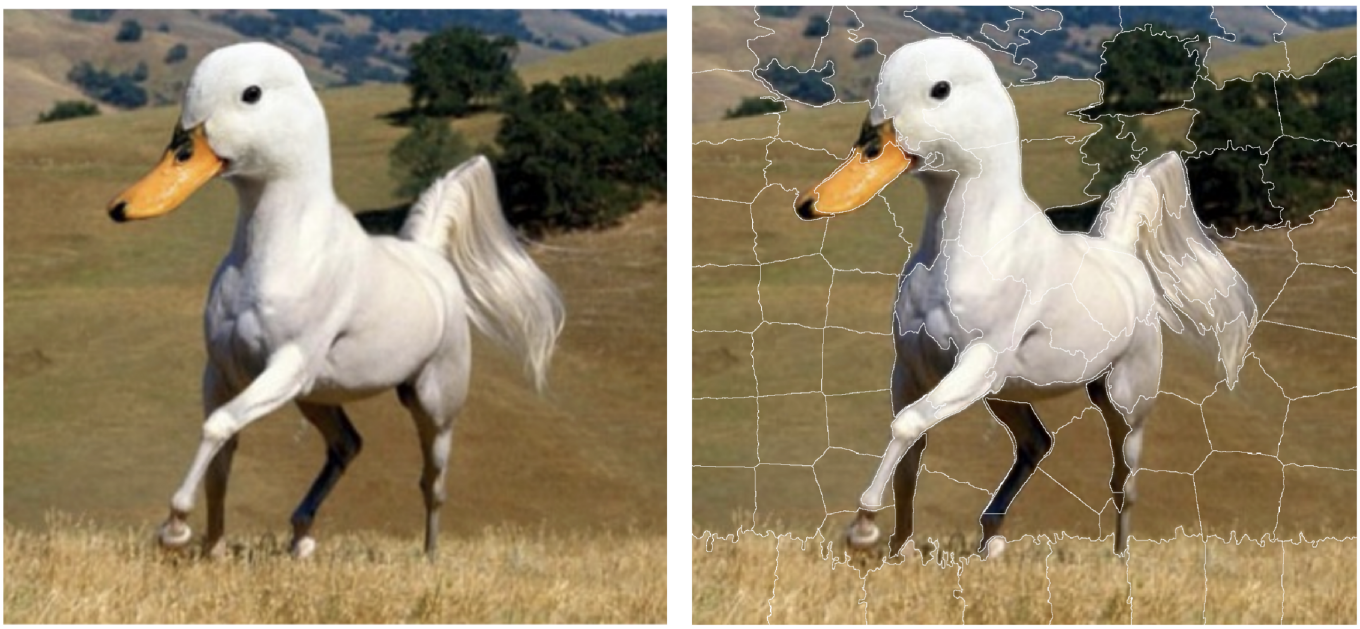


Figure 10.2: The left panel shows an ambiguous picture, half-horse and half-duck. The right panel shows 100 superpixels identified for this figure. Source: https://twitter.com/finmaddison/status/352128550704398338.

### 10.3.2 Sampling around the instance of interest

To develop the locally-approximation glass-box model, we need new data points in the interpretable space around the instance of interest. It may not be enough to sample points from the original dataset, because in a high-dimensional data space the data are usually very sparse and data points are ,,far" from each other. We need new artificial data points in the interpretable space. For this reason, the data for the development of the glass-box model are often created by using perturbations of the instance of interest.

For a set of binary variables in the interpretable space, the common choice is to flip (from 0 to 1 or from 1 to 0) the value of a randomly-selected number of variables describing the instance of interest.

For continuous variables, various proposals are introduced in different papers. For example ,,iml: An R package for Interpretable Machine Learning" (Molnar, Bischl, and Casalicchio 2018) and (Molnar 2019) adds some Gaussian noise to continuous variables. In ,,lime: Local Interpretable Model-Agnostic Explanations" (Pedersen and Benesty 2019) continuous variables are discretized with the use of quintiles and the perturbations are don on discretized variables. In ,,localModel: LIME-Based Explanations with Interpretable Inputs Based on Ceteris Paribus Profiles" (Staniak et al. 2019) continuous variables are discretized based on segmentation of local Ceteris Paribus profiles.

In the example of the duck-horse in Figure 10.2, the perturbations of the image would be created by randomly including or excluding some of the superpixels. See an example in Figure 10.3.

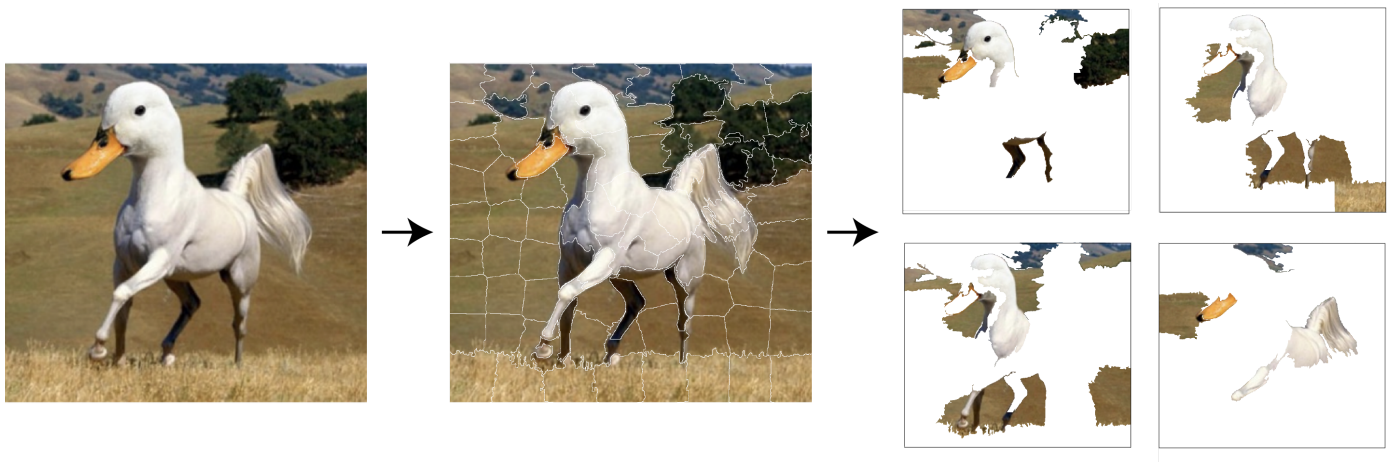


Figure 10.3: In the original input space image is described by RGB colors for each pixel (left panel). The image is transformed into the interpretable input space with 100 super pixels (central panel). The artificial data is a subset of superpixels (right panel).

### 10.3.3 Developing the glass-box model

Once the new data were sampled around the instance of interest, we may attempt to develop an interpretable glass-box model $g()$ from class $G$.

The most common choices for $G$ are generalized linear models. To get sparse models, i.e., models with a limited number of variables, LASSO (Tibshirani 1994) or similar regularization-modelling techniques are used. For instance, in the algorithm presented in Section 10.3, the K-LASSO method has been mentioned. An alternative choice are classification-and-regression trees (Breiman et al. 1984).

The VGG16 network for each picture predicts 1000 probabilities that corresponds to the 1000 classes used for training. For the duck-horse picture the two most likely classes are *'standard poodle'* and *'goose'*. Figure 10.4 presents LIME explanations for these top two classes. The explanations were obtained with the K-LASSO method which selected $K$ superpixels that were the most influential from the model-prediction point of view. Here we show results for $K = 15$. For each of the selected two classes, the $K$ superpixels with non-zero coefficients are highlighted. It is interesting to observe that the superpixel which contains the beak is influential for the prediction *'goose'*, while the superpixels linked with the white colour are influential for the prediction *'standard poodle'*. This is aligned with the intention thus such additional validation increases trust in model prediction.

Figure 10.4: LIME for two predictions ('standard poodle' and 'goose') obtained by the VGG16 network with ImageNet weights for the half-duck, half-horse image.

## 10.4 Example: Titanic data

Most examples of LIME method are related to the text or image data. Here we present examples for tabular data to facilitate comparisons between methods introduced in different chapters. Let us consider the random-forest model `titanic_rf_v6` (see Section 5.1.3) and passenger `johny_d` (see Section 5.1.6) as the instance of interest in the Titanic data.

First, we need to define an interpretable input space. One option would be to gather similar variables into larger constructs corresponding to concepts. For example `class` and `fare` variables can be combined into a concept `wealth`, `age` and `gender` into a concept `demography` and so on. In this example we have relatively small number of variables so we will use a simpler interpretable data representation in the form of a binary vector. Each variable is dychotomized into two levels. For example `age` is transformed into a binary variable `<=` / `>` than 15, `class` is transformed into a binary variable `1st` / `2nd` / `deck crew` and so on. The LIME algorithm is applied to this interpretable feature space and the K-LASSO method with $K = 3$ is used to identify 3 most important variables that will be transformed into an explanation.

Once the interpretable variable space is defined, we need to transform `johny_d` to this space and generate a new dataset that will be used for K-LASSO approximations of random forest model. Figure 10.5 shows coefficients estimated in this K-LASSO model.

The three variables that are identified as the most influential are: `age`, `gender`, and `class`. Note that, for age, a dichotomized version of the originally continuous variable is used. On the other hand, for class, a dichotomized version based on the combination of several original categories is used.

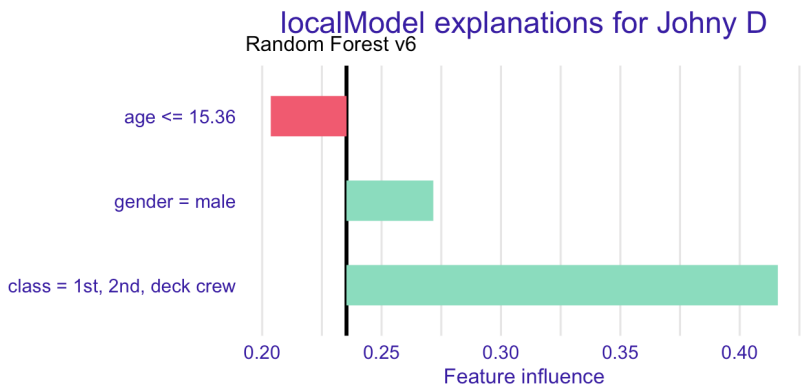



Figure 10.5: LIME method for the prediction for `johny_d` for the random-forest model `titanic_rf_v6` and the Titanic data. Presented values are beta coefficients in the K-LASSO model fitted locally to the response from the original model.

The interpretable features can be defined in a many different ways. One idea would to be use quartiles for the feature of interest. Another idea is to use Ceteris Paribus profiles (see Chapter 11 and change-point method (Picard 1985) to find a instance specific discretization. Different implementations of LIME differ in the way how the interpretable feature space is created.

## 10.5 Pros and cons

As mentioned by ,,Why Should I Trust You?: Explaining the Predictions of Any Classifier" (Ribeiro, Singh, and Guestrin 2016), the LIME method

- is *model-agnostic*, as it does not imply any assumptions on the black-box model structure,
- offers an *interpretable representation*, because the original data space is transformed into a more interpretable lower-dimension space (like transformation from individual pixels to super pixels for image data),
- provides *local fidelity*, i.e., the explanations are locally well-fitted to the black-box model.

The method has been widely adopted in text and image analysis, in part due to the interpretable data representation. Also, explanations are delivered as a subset of an image/text and our brain is good in the justification of such explanations. The underlying intuition for the method is easy to understand: a simpler model is used to approximate a more

complex one. By using a simpler model, with a smaller number of interpretable explanatory variables, predictions are easier to explain. The LIME method can be applied to complex, high-dimensional models.

But there are several important limitations. For instance, despite several proposals, the issue of finding interpretable representations for continuous and categorical variables is not solved yet. Also, because the glass-box model is selected to approximate the black-box model, and the data themselves, the method does not control the quality of the local fit of the glass-box model to the data. Thus, the latter model may be misleading.

Finally, in high-dimensional data, data points are sparse. Defining a ‘’local neighborhood” of the instance of interest may not be straightforward. Importance of the local neighbourhood is presented for example in the article ,,On the Robustness of Interpretability Methods” (Alvarez-Melis and Jaakkola 2018). Sometimes even slight changes in the neighbourhood affects strongly obtained explanations.

To summarise, the most useful applications of LIME are limited to high dimensional data for which one can defined a low-dimensional interpretable data representation, as in image analysis, text analysis or genomics.

## 10.6 Code snippets for R

LIME and similar methods are implemented in various R and Python packages. For example, `lime` (Pedersen and Benesty 2019) is a port of the LIME Python library (Lundberg 2019), while `live` (Staniak and Biecek 2018), `localModel` (Staniak et al. 2019), and `iml` (Molnar, Bischl, and Casalicchio 2018) are separate R packages that implements this method from scratch.

Different implementations of LIME offer different algorithms for extraction of interpretable features, different methods for sampling, and different methods of weighting. For instance, regarding transformation of continuous variables into interpretable features, `lime` performs global discretization using quartiles, `localModel` performs local discretization using CP profiles, while `live` and `iml` work directly on continuous variables. Due to these differences, the packages yield different results (explanations).

In what follows, for illustration purposes, we use the `titanic_rf_v6` random-forest model for the Titanic data developed in Section 5.1.3. Recall that it is developed to predict the probability of survival from sinking of Titanic. Instance-level explanations are calculated for a

single observation: `johny_d` - an 8-year-old passenger that travelled in the 1st class. `DALEX` explainers for the model and the `jonhy_d` data are retrieved via `archivist` hooks as listed in Section 5.1.8.

```
library("DALEX")
library("randomForest")

titanic <- archivist::aread("pbiecek/models/27e5c")
titanic_rf_v6 <- archivist::aread("pbiecek/models/31570")
johny_d <- archivist::aread("pbiecek/models/e3596")
```

## 10.6.1 The lime package

The key elements of the `lime` package are functions `lime()` , which creates an explainer, and `explain()` , which evaluates explanations.

The detailed results for the `titanic_rf_v6` random-forest model and `johny_d` are presented below. First we need to specify that we will work with a model for classification.

```
library("lime")
model_type.randomForest <- function(x, ...) "classification"
```

Second we need to create an explainer - an object with all elements needed for calculation of explanations. This can be done with the `lime` function, the dataset and the model.

```
lime_rf <- lime(titanic[,colnames(johny_d)], titanic_rf_v6)
```

In the last step we generate an explanation. The `n_features` set the K for K-LASSO method. Here we ask for explanations not larger than 4 variables. The `n_permutations` argument defines how many points are to be sampled for a local model approximation. Here we use a set of 1000 artificial points for this.

```
lime_expl <- lime::explain(johny_d, lime_rf, labels = "yes",
                           n_features = 4, n_permutations = 1000)
lime_expl
```

```
#      model_type case label label_prob  model_r2 model_intercept model_prediction
#1 classification    1    no      0.602 0.5806297       0.5365448        0.5805939
#2 classification    1    no      0.602 0.5806297       0.5365448        0.5805939
#3 classification    1    no      0.602 0.5806297       0.5365448        0.5805939
#4 classification    1    no      0.602 0.5806297       0.5365448        0.5805939
#  feature feature_value feature_weight  feature_desc                 data   prediction
#1    fare            72     0.00640936  21.00 < fare 1, 2, 8, 0, 0, 72, 4 0.602, 0.398
#2  gender             2     0.30481181 gender = male 1, 2, 8, 0, 0, 72, 4 0.602, 0.398
#3   class             1    -0.16690730   class = 1st 1, 2, 8, 0, 0, 72, 4 0.602, 0.398
#4     age             8    -0.10026475     age <= 22 1, 2, 8, 0, 0, 72, 4 0.602, 0.398
```

In this table the `feature_weight` column has coefficients for the K-LASSO method in the explanation. In the column `case` one will find an index of observation for which the explanation is calculated. Here it's 1 since we asked for explanation for only one observation. The `feature_weight` columns shows the $\beta$ coefficients in the K-LASSO model, `feature` column points out which variables have non zero coefficients in the K-LASSO method. The `feature_value` column denotes values for the selected features for the observation of interest. The `feature_description` column shows how the original feature was transformed into a interpretable feature.

This implementation of the LIME method dichotomizes continuous variables by using quartiles. Hence, in the output we get a binary variable `age <= 22` .

The corresponding local white box model is

$$\hat{y} = 0.00640936 * 1_{fare>21} + 0.30481181 * 1_{gender=male} - 0.16690730 * 1_{class=1st} - 0.10026$$

Figure 10.6 shows the graphical presentation of the results, obtained by applying the generic `plot()` function.

Color corresponds to the sign of the $\beta$ coefficient while length of the bar corresponds to the absolute value of $\beta$ coefficient in the K-LASSO method.

```
plot_features(lime_expl)
```

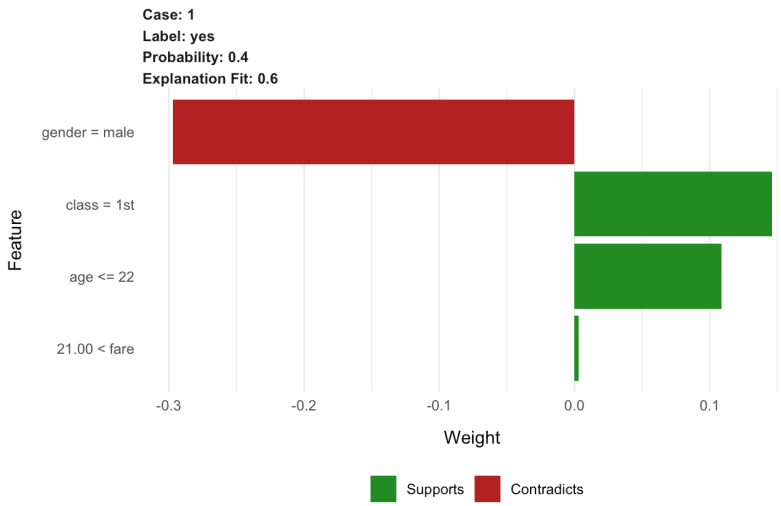



Figure 10.6: LIME-method results for the prediction for `johny_d` for the random-forest model `titanic_rf_v6` and the Titanic data, generated by the `lime` package.

## 10.6.2 The localModel package

The `localModel` package operates on `DALEX::explain()` object. The main function in this package is `individual_surrogate_model()` which trains the local glass-box model.

The detailed results for the `titanic_rf_v6` random-forest model and `johny_d` are presented below.

```
library("localModel")

explainer_titanic_rf <- DALEX::explain(model = titanic_rf_v6,
            data = titanic[,colnames(johny_d)])
local_model_rf <- individual_surrogate_model(explainer_titanic_rf,
            johny_d, size = 1000, seed = 1313)
local_model_rf
#   estimated                    variable dev_ratio response
#1 0.23479837                (Model mean) 0.6521442
#2 0.14483341                 (Intercept) 0.6521442
#3 0.08081853 class = 1st, 2nd, deck crew 0.6521442
#4 0.00000000     gender = female, NA, NA 0.6521442
#5 0.23282293                 age <= 15.36 0.6521442
#6 0.02338929                 fare > 31.05 0.6521442
```

In the column `estimated` one will find $\beta$ coefficients for LASSO logistic regression while in the `variable` column one will find corresponding values.

The implemented version of LIME dichotomizes continuous variables by using CP profiles. The CP profile for `johny_d`, presented in Figure 11.9 in Chapter 11, indicated that, for age, the largest drop in the predicted probability of survival was observed for the age increasing beyond 15 years. Hence, in the output of the `individual_surrogate_model()`, we see a binary variable `age < 15.36`.

Figure 10.7 illustrates how the two levels for age can be extracted from the Ceteris Paribus profile.

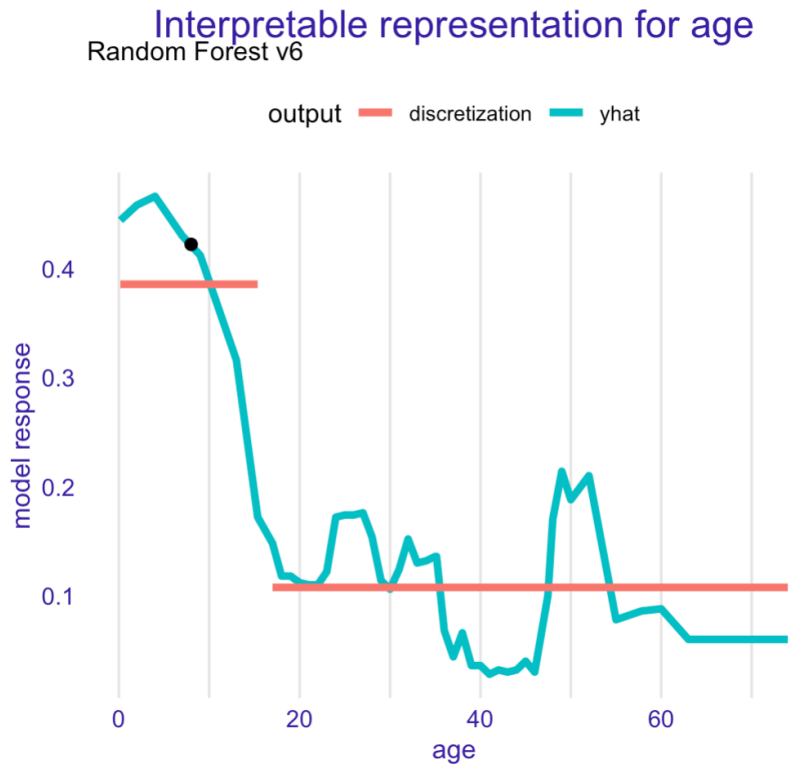



Figure 10.7: Interpretable instance-level discretisation of age variable. Based on the Ceteris Paribus profiles we may estimate an optimal change-point as 15 years.

The graphical presentation of the results, obtained by applying the generic `plot()` function is provided in Figure 10.8. Bars correspond to $\beta$ coefficients in the LASSO model.

```
plot(local_model_rf)
```

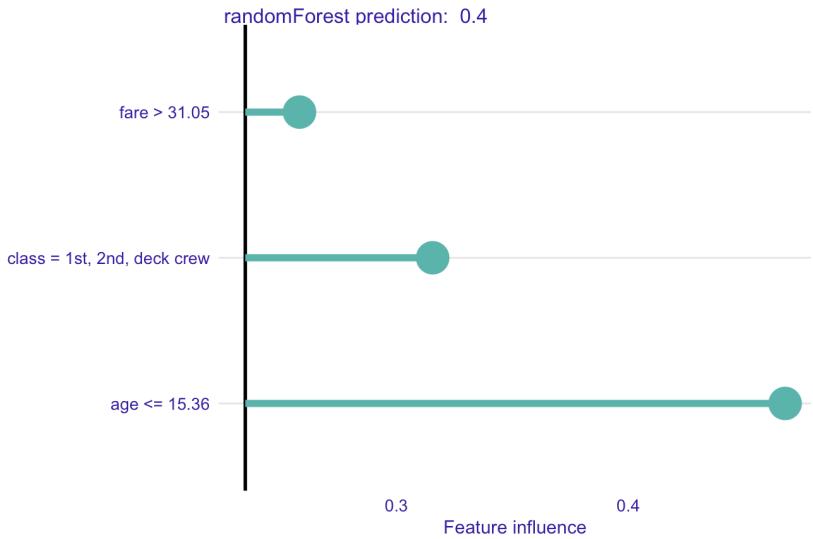



Figure 10.8: LIME-method results for the prediction for `johny_d` for the random-forest model `titanic_rf_v6` and the Titanic data, generated by the `localModel` package.

### 10.6.3 The iml package

The key elements of the `iml` package are functions `Predictor$new()`, which creates an explainer, and `LocalModel$new()`, which develops the local glass-box model.

The detailed results for the `titanic_rf_v6` random-forest model and `johny_d` are presented below.

```
library("iml")
iml_rf = Predictor$new(titanic_rf_v6, data = titanic[,colnames(johny_d)])
iml_glass_box = LocalModel$new(iml_rf, x.interest = johny_d, k = 6)
iml_glass_box
#Interpretation method:  LocalModel
#
#Analysed predictor:
#Prediction task: unknown
#
#Analysed data:
#Sampling from data.frame with 2207 rows and 7 columns.
#
#Head of results:
#          beta x.recoded     effect  x.original              feature
#1 -0.158368701         1 -0.1583687         1st            class=1st
#2  1.739826204         1  1.7398262        male          gender=male
#3  0.018515945         0  0.0000000           0                sibsp
#4 -0.001484918        72 -0.1069141          72                 fare
#5  0.131819869         1  0.1318199 Southampton embarked=Southampton
#6  0.158368701         1  0.1583687         1st            class=1st
```

In the `effect` column on can read $\beta$ coefficients for the LASSO method.

The implemented version of LIME does not transform continuous variables. The CP profile for `johny_d`, presented in Figure 11.9 in Chapter 11, indicated that, for boys younger than 15-year-old, the predicted probability of survival did not change very much. Hence, in the printed output, age does not appear as an important variable.

The graphical presentation of the results, obtained by applying the generic `plot()` function to the object resulting from the application of the `explain()` function, is provided in Figure 10.9. Note that only first 6 rows are listed in the table above. The whole table has 12 coefficients that corresponds to bars in the plot.

```
plot(iml_glass_box)
```

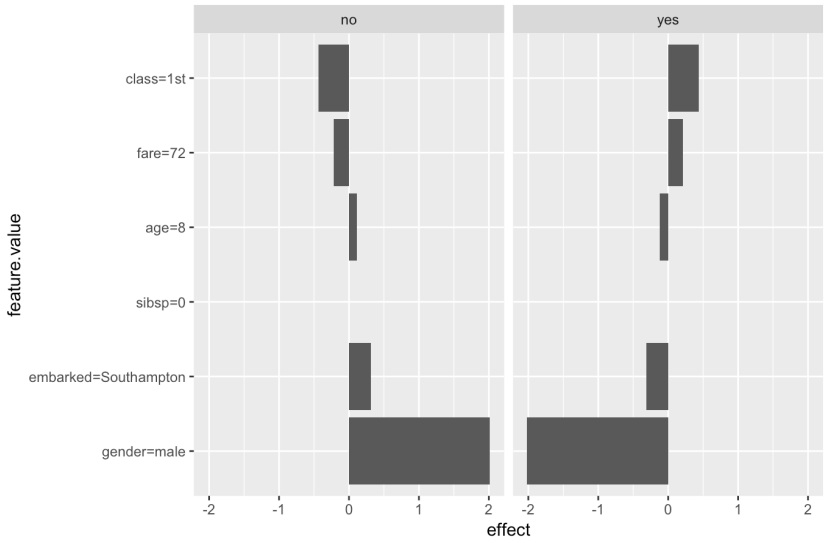



Figure 10.9: LIME-method results for the prediction for `johny_d` for the random-forest model `titanic_rf_v6` and the Titanic data, generated by the `iml` package.

# References

Alvarez-Melis, David, and Tommi S. Jaakkola. 2018. “On the Robustness of Interpretability Methods.” *arXiv E-Prints*, June, arXiv:1806.08049.

Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification and Regression Trees*. Monterey, CA: Wadsworth; Brooks.

Deng, J., W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. 2009. “ImageNet: A large-scale hierarchical image database.” In *2009 Ieee Conference on Computer Vision and Pattern Recognition*, 248–55. https://doi.org/10.1109/cvpr.2009.5206848.

Hothorn, Torsten, Kurt Hornik, and Achim Zeileis. 2006. “Unbiased Recursive Partitioning: A Conditional Inference Framework.” *Journal of Computational and Graphical Statistics* 15 (3): 651–74.

Lundberg, Scott. 2019. *SHAP (SHapley Additive exPlanations)*. https://github.com/slundberg/shap.

Molnar, Christoph. 2019. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*.

Molnar, Christoph, Bernd Bischl, and Giuseppe Casalicchio. 2018. “iml: An R package for Interpretable Machine Learning.” *Joss* 3 (26). Journal of Open Source Software: 786. https://doi.org/10.21105/joss.00786.

Pedersen, Thomas Lin, and Michaël Benesty. 2019. *lime: Local Interpretable Model-Agnostic Explanations*. https://CRAN.R-project.org/package=lime.

Picard, Dominique. 1985. “Testing and estimating change-points in time series.” *Advances in Applied Probability* 17 (4). Cambridge University Press: 841–67. https://doi.org/10.2307/1427090.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. 2016. “Why Should I Trust You?: Explaining the Predictions of Any Classifier.” In, 1135–44. ACM Press. https://doi.org/10.1145/2939672.2939778.

Simonyan, Karen, and Andrew Zisserman. 2015. “Very Deep Convolutional Networks for Large-Scale Image Recognition.” In *International Conference on Learning Representations*.

Staniak, Mateusz, Przemyslaw Biecek, Krystian Igras, and Alicja Gosiewska. 2019. *localModel: LIME-Based Explanations with Interpretable Inputs Based on Ceteris Paribus Profiles*. https://CRAN.R-project.org/package=localModel.

Staniak, Mateusz, and Przemysław Biecek. 2018. *Live: Local Interpretable (Model-Agnostic) Visual Explanations*. https://CRAN.R-project.org/package=live.

Tibshirani, Robert. 1994. “Regression Shrinkage and Selection Via the Lasso.” *Journal of the Royal Statistical Society, Series B* 58: 267–88.

# 11 Ceteris-paribus Profiles

## 11.1 Introduction

Chapters 7 – 10 are related to the decomposition of a prediction $f(x)$ into parts linked with particular variables. In this chapter we focus on methods that analyse an effect of selected variables on model response. These techniques may be used for sensitivity analysis, how stable is the model response, or to what-if analysis, how model response would change if input is changes. It is important to remember that the what-if analysis is performed not in the sense of causal modeling, but in the sense of model exploration. We need causal model to do causal inference for the real-world phenomena. Here we focus on explanatory analysis of the model behaviour. To show the difference between these two things, think about a model for survival for lung-cancer patients based on some treatment parameters. We need causal model to say how the survival would change if the treatment is changed. Techniques presented in this chapter will explore how the model result will change if the treatment is changed.

*Ceteris paribus* is a Latin phrase meaning "other things held constant" or "all else unchanged." In this chapter, we introduce a technique for model exploration based on the *Ceteris paribus* principle. In particular, we examine the influence of each explanatory variable, assuming that effects of all other variables are unchanged. The main goal is to understand how changes in a single explanatory variable affects model predictions.

Explanation tools (explainers) presented in this chapter are linked to the second law introduced in Section 1.3, i.e. the law of "Prediction's speculation." This is why the tools are also known as *What-If model analysis* or *Individual Conditional Expectations* (Goldstein et al. 2015). It appears that it is easier to understand how a black-box model is working if we can explore the model by investigating the influence of explanatory variables separately, changing one at a time.

## 11.2 Intuition

Ceteris-paribus profiles show how the model response would change if a single variable is changed. For example, panel A of Figure 11.1 presents response (prediction) surface for the `titanic_lmr_v6` model for two explanatory variables, *age* and *class*, from the *titanic* dataset (see Section 5.1). We are interested in the change of the model prediction induced by each of the variables. Toward this end, we may want to explore the curvature of the response surface around a single point with *age* equal to 47 and *class* equal to "1st," indicated in the plot. Ceteris-paribus (CP) profiles are one-dimensional profiles that examine the curvature across each dimension, i.e., for each variable. Panel B of Figure 11.1 presents the CP profiles corresponding to *age* and *class*. Note that, in the CP profile for *age*, the point of interest is indicated by the black dot. In essence, a CP profile shows a conditional expectation of the dependent variable (response) for the particular explanatory variable.

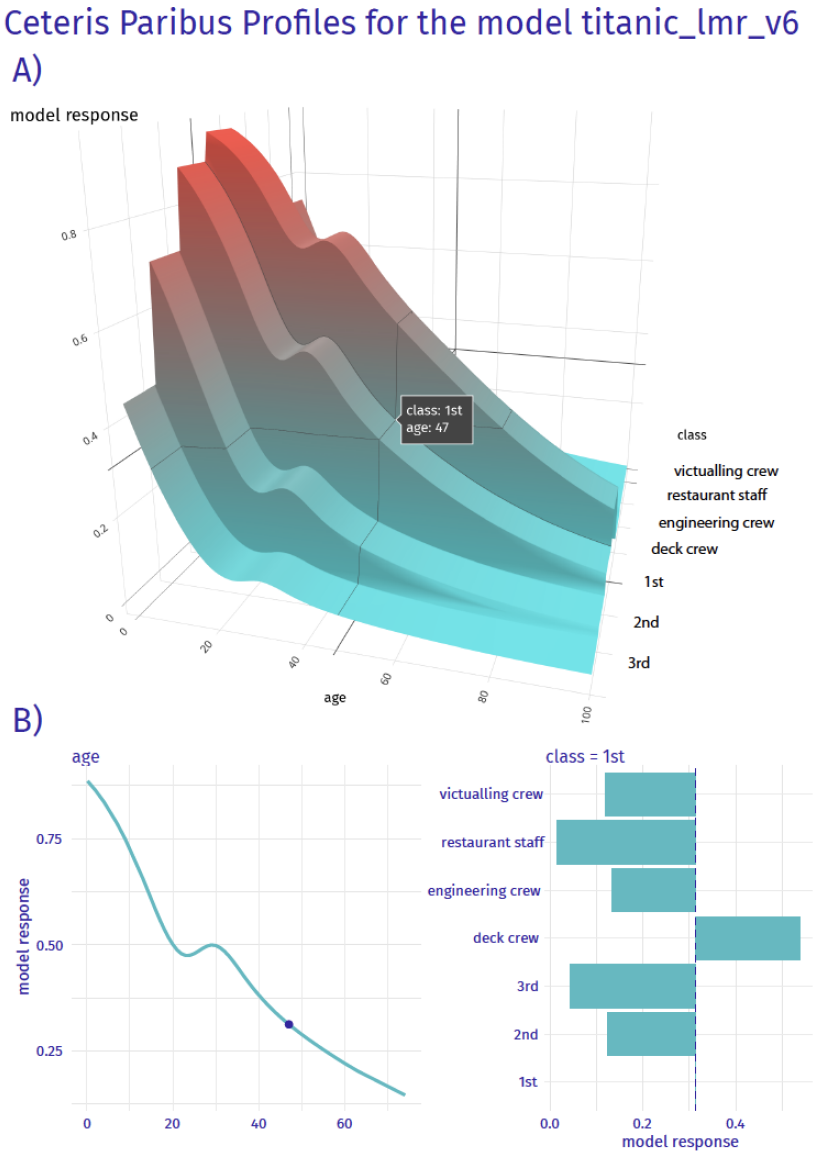



Figure 11.1: Panel A) Model response (prediction) surface. Ceteris-paribus (CP) profiles marked with black curves help to understand the curvature of the surface while changing only a single explanatory variable. Panel B) CP profiles for individual variables, age (continuous) and class (categorical).

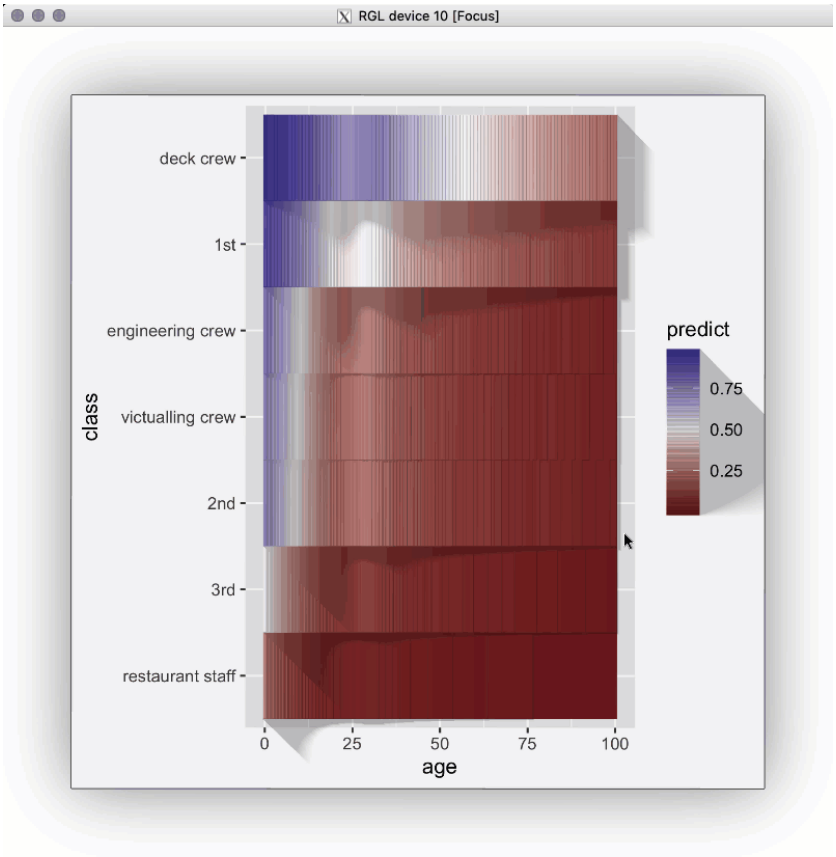



Figure 11.2: Animated model response for 2D surface as in 11.1.

CP belongs to the class of techniques that examine local curvature of the model response surface. Other very popular technique from this class called LIME is presented in Chapter 10. The difference between these two methods lies in the fact that LIME approximates the model of interest locally with a simpler glass-box model. Usually, the LIME model is sparse, i.e., contains fewer explanatory variables. Thus, one needs to investigate a plot across a smaller number of dimensions. On the other hand, the CP profiles present conditional predictions for a single variable and, in most cases, are easier to interpret. More detailed comparison of these techniques is presented in the Chapter 14.

## 11.3 Method

In this section, we introduce more formally one-dimensional CP profiles.

Recall (see Section 2.3) that we use $x_i$ to refer to the vector corresponding to the $i$-th observation in a dataset. Let $x_*^j$ denote the $j$-th element of $x_*$, i.e., the $j$-th explanatory variable. We use $x_*^{-j}$ to refer to a vector resulting from removing the $j$-th element from $x_*$. By

$x_*^{j|=z}$, we denote a vector resulting from changing the value of the $j$-th element of $x_*$ to (a scalar) $z$.

We define a one-dimensional CP profile $h()$ for model $f()$, the $j$-th explanatory variable, and point $x_*$ as follows:

$$h_{x_*}^{f,j}(z) := f(x_*^{j|=z}). \tag{11.1}$$

CP profile is a function that provides the dependence of the approximated expected value (prediction) of $Y$ on the value $z$ of the $j$-th explanatory variable. Note that, in practice, $z$ is taken to go through the entire range of values typical for the variable, while values of all other explanatory variables are kept fixed at the values specified by $x_*$.

Note that in the situation when only a single model is considered, we will skip the model index and we will denote the CP profile for the $j$-th explanatory variable and the point of interest $x_*$ by $h_{x_*}^{j}(z)$.

## 11.4 Example: Titanic

For continuous explanatory variables, a natural way to represent the CP function is to use a profile plot similar to the ones presented in Figure 11.3. In the figure, the dot on the curves marks an instance prediction, i.e., prediction $f(x_*)$ for a single observation $x_*$. The curve itself shows how the prediction would change if the value of a particular explanatory variable changed.

Figure 11.3 presents CP profiles for the *age* variable in the logistic regression `titanic_lmr_v6` and random forest model `titanic_rf_v6` for the Titanic dataset (see Sections 5.1.2 and 5.1.3, respectively). It is worth observing that the profile for the logistic regression model is smooth, while the one for the random forest model shows more variability. For this instance (observation), the prediction for the logistic regression model would increase substantially if the value of *age* became lower than 20. For the random forest model, a substantial increase would be obtained if *age* became lower than 13 or so.

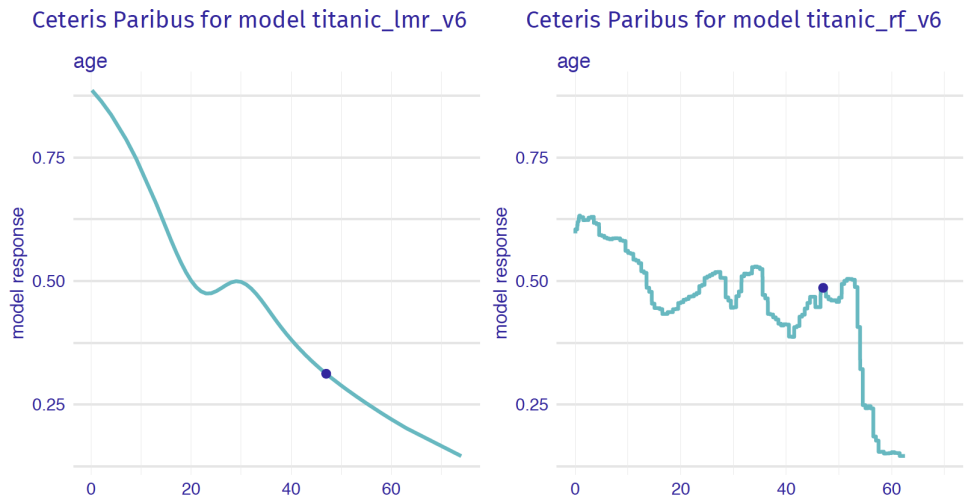



Figure 11.3: Ceteris-paribus profiles for variable `age` for the logistic regression ( `titanic_lmr_v6` ) and random forest ( `titanic_rf_v6` ) models that predict the probability of surviving based on the Titanic data. Black dot corresponds to the passenger `johny_d` .

For a categorical explanatory variable, a natural way to represent the CP function is to use a barplot similar to the ones presented in Figure 11.4. The barplots in Figure 11.3 present CP profiles for the *class* variable in the logistic regression and random forest models for the Titanic dataset (see Sections 5.1.2 and 5.1.3, respectively). For this instance (observation), the predicted probability for the logistic regression model would decrease substantially if the value of *class* changed to “2nd”. On the other hand, for the random forest model, the largest change would be marked if *class* changed to “restaurant staff”.

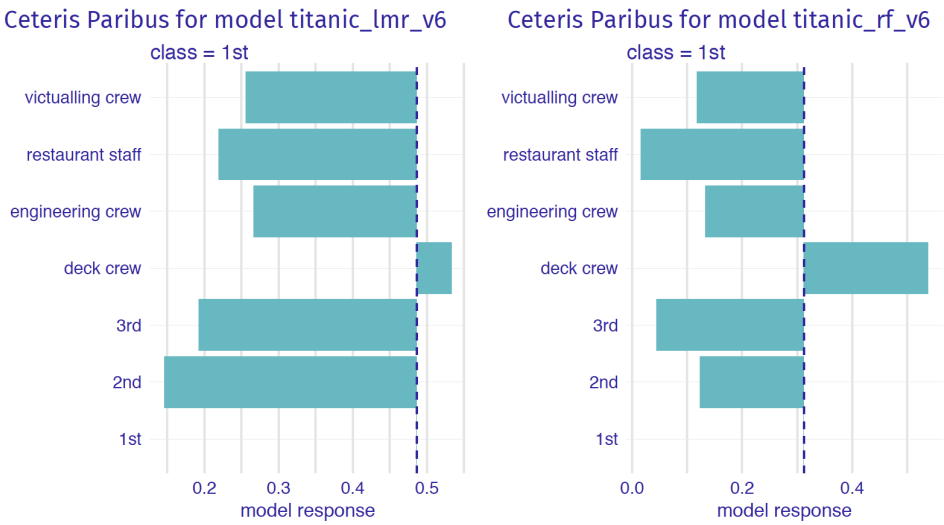



Figure 11.4: Ceteris-paribus profiles for variable `class` for the logistic regression ( `titanic_lmr_v6` ) and random forest ( `titanic_rf_v6` ) models that predict the probability of surviving based on the Titanic data.

Usually, black-box models contain a large number of explanatory variables. However, CP profiles are legible even for tiny subplots, created with techniques like sparklines or small multiples (Tufte 1986). In this way we can display a large number of profiles at the same time keeping profiles for consecutive variables in separate panels, as shown in Figure 11.5 for the random forest model for the Titanic dataset. It helps if these panels are ordered so that the most important profiles are listed first. We discuss a method to assess the importance of CP profiles in the next chapter.

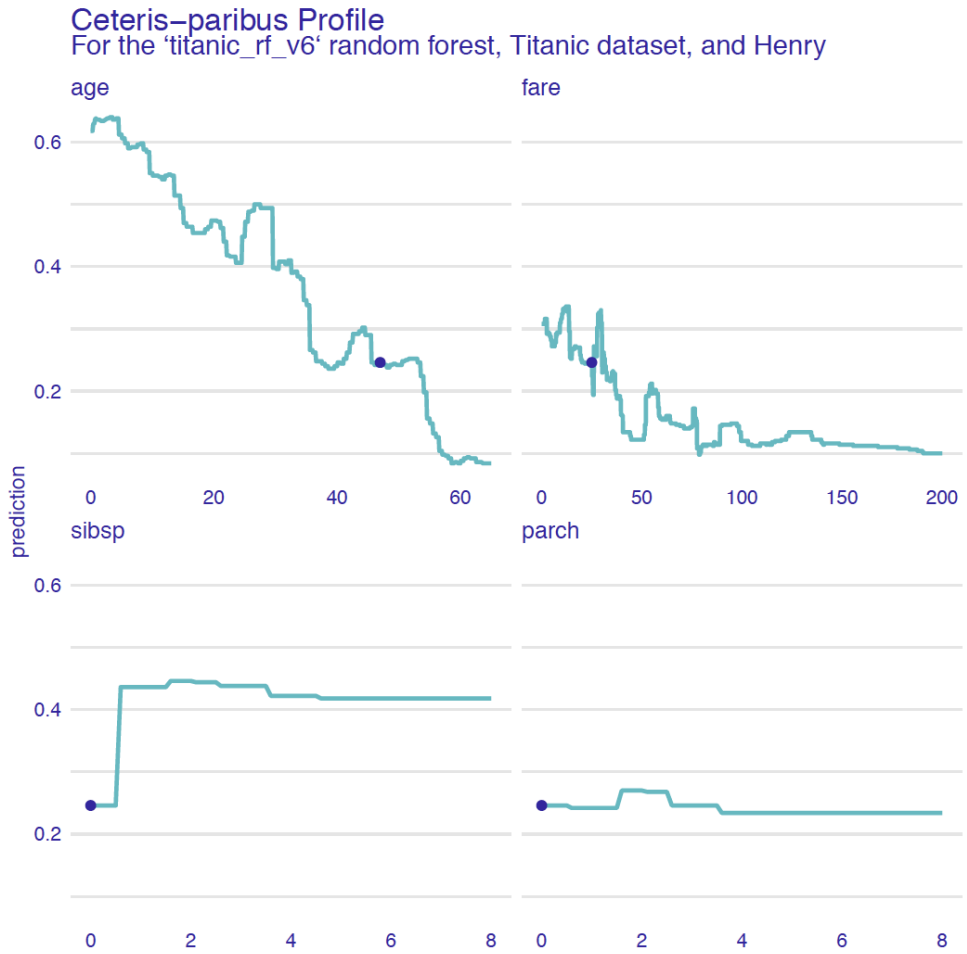



Figure 11.5: Ceteris-paribus profiles for all continuous explanatory variables for the random forest ( `titanic_rf_v6` ) model for the `titanic` dataset.

## 11.5 Pros and cons

One-dimensional CP profiles, as presented in this chapter, offer a uniform, easy to communicate and extendable approach to model exploration. Their graphical representation is easy to understand and explain. It is possible to show profiles for many variables or models in a single plot. CP profiles are easy to compare, thus we can juxtapose two or more models to better understand differences between models. We can also compare two or more instances to better understand model stability. CP profiles are also a useful tool for sensitivity analysis.

But. There are several issues related to the use of the CP profiles. If explanatory variables are correlated, then changing one variable implies a change in the other. In such case, the application of the *Ceteris paribus* principle may lead to unrealistic settings, as it is not possible to keep one variable fixed while varying the other one. For example, apartment's price prediction features like surface and number of rooms are correlated thus it is unrealistic to consider very small apartments with extremely large number of rooms. Special cases are interactions, which require the use of two-dimensional CP profiles that are more complex than one-dimensional ones. Also, in case of a model with hundreds or thousands of variables, the number of plots to inspect may be daunting. Finally, while barplots allow visualization of CP profiles for factors (categorical explanatory variables), their use becomes less trivial in case of factors with many nominal (unordered) categories (like, for example, a ZIP-code).

## 11.6 Code snippets for R

In this section, we present key features of the R package `DALEX` which is a part of `DrWhy.AI` universe and covers all methods presented in this chapter. Note that presented functions in fact are wrappers to package `ingredients` (Biecek et al. 2019).

Note that there are also other R packages that offer similar functionality, like `condvis` (O'Connell, Hurley, and Domijan 2017), `pdp` (Greenwell 2017), `ICEbox` (Goldstein et al. 2015), `ALEPlot` (Apley 2018), `iml` (Molnar, Bischl, and Casalicchio 2018).

For illustration, we use two classification models developed in Chapter 5.1, namely the logistic regression model `titanic_lmr_v6` (Section 5.1.2) and the random forest model `titanic_rf_v6` (Section 5.1.3). They are developed to predict the probability of survival after sinking of Titanic. Instance-level explanations are calculated for a single observation `henry` - a 47 years old male passenger that travelled in the 1st class.

`DALEX` explainers for both models and the `henry` data frame are retrieved via the `archivist` hooks as listed in Section 5.1.8.

```
library("rms")
explain_lmr_v6 <- archivist::aread("pbiecek/models/34e19")

library("randomForest")
explain_rf_v6 <- archivist::aread("pbiecek/models/6ed54")

library("DALEX")
henry <- archivist::aread("pbiecek/models/a6538")
henry
```

```
##   class gender age sibsp parch fare  embarked
## 1   1st   male  47     0     0   25 Cherbourg
```

### 11.6.1 Basic use of the `individual_profile` function

The easiest way to create and plot CP profiles is to call `individual_profile()` function and then the generic `plot()` function. By default, profiles for all variables are being calculated and all numeric features are being plotted. One can limit the number of variables that should be considered with the `variables` argument.

To obtain CP profiles, the `individual_profile()` function requires the explainer-object and the instance data frame as arguments. As a result, the function yields an object of the class `ceteris_paribus_explainer`. It is a data frame with model predictions.

```
library("DALEX")
cp_titanic_rf <- individual_profile(explain_rf_v6,
                                    new_observation = henry)
cp_titanic_rf
```

```
## Top profiles    :
##                class gender age sibsp parch fare  embarked _yhat_ _vname_
## 1                1st   male  47     0     0   25 Cherbourg  0.246   class
## 1.1              2nd   male  47     0     0   25 Cherbourg  0.054   class
## 1.2              3rd   male  47     0     0   25 Cherbourg  0.100   class
## 1.3        deck crew   male  47     0     0   25 Cherbourg  0.454   class
## 1.4 engineering crew   male  47     0     0   25 Cherbourg  0.096   class
## 1.5 restaurant staff   male  47     0     0   25 Cherbourg  0.092   class
##     _ids_       _label_
## 1       1 Random Forest
## 1.1     1 Random Forest
## 1.2     1 Random Forest
## 1.3     1 Random Forest
## 1.4     1 Random Forest
## 1.5     1 Random Forest
##
##
## Top observations:
##   class gender age sibsp parch fare  embarked _yhat_       _label_ _ids_
## 1   1st   male  47     0     0   25 Cherbourg  0.246 Random Forest     1
```

To obtain a graphical representation of CP profiles, the generic `plot()` function can be applied to the data frame returned by the `individual_profile()` function. It returns a `ggplot2` object that can be processed further if needed. In the examples below, we use the `ggplot2` functions, like `ggtitle()` or `ylim()`, to modify plot's title or the range of the Y-axis.

The resulting plot can be enriched with additional data by applying functions `ingredients::show_rugs` (adds rugs for the selected points), `ingredients::show_observations` (adds dots that shows observations), or `ingredients::show_aggreagated_profiles`. All these functions can take additional arguments to modify size, color, or linetype.

Below we show an R snippet that can be used to replicate plots presented in the upper part of Figure 11.5.

```
library("ggplot2")
plot(cp_titanic_rf, variables = c("age", "fare")) +
  ggtitle("Ceteris Paribus Profile",
          "For the random forest model and the titanic dataset")
```

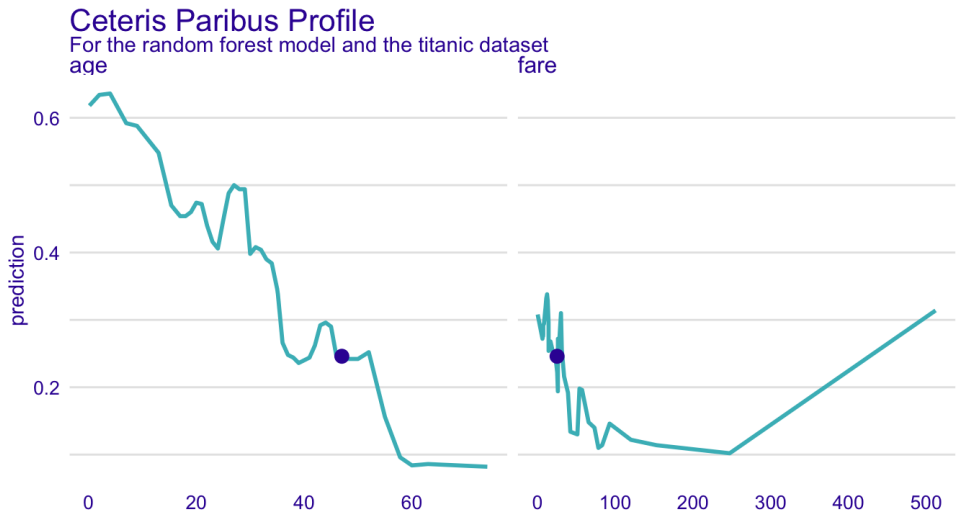



Figure 11.6: Ceteris-paribus profiles for `age` and `fare` variables and the `titanic_rf_v6` model.

By default, all numerical variables are plotted. To plot CP profiles for categorical variables, we have got to add the `variable_type = "categorical"` argument to the `plot()` function. The code below an be used to recreate the right-hand-side plot from Figure 11.4.

```
plot(cp_titanic_rf, variables = c("class", "embarked"),
     variable_type = "categorical") +
  ggtitle("Ceteris Paribus profile",
          "For the random forest model and the titanic dataset")
```

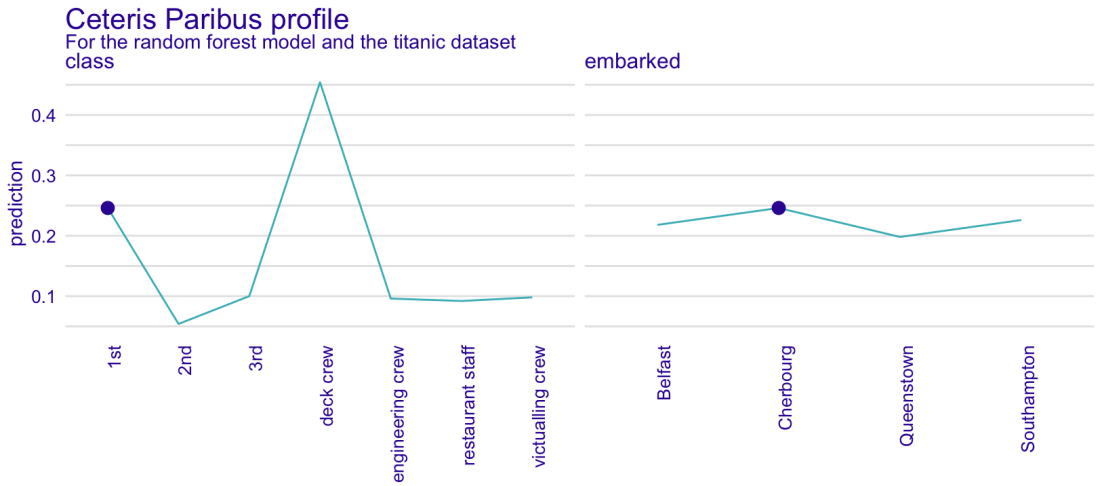



Figure 11.7: Ceteris-paribus profiles for `class` and `embarked` variables and the `titanic_rf_v6` model.

## 11.6.2 Advanced use of the `individual_profile` function

The `individual_profile()` is a very flexible function. To better understand how it can be used, we briefly review its arguments.

- `x`, `data`, `predict_function`, `label` - information about a model. If `x` is created with the `DALEX::explain` function, then other arguments are extracted from `x`; this is how we use the function in this chapter. Otherwise, we have got to specify directly the model, the validation data, the predict function, and the model label.
- `new_observation` - instance (one or more), for which we want to calculate CP profiles. It should be a data frame with same variables as in the validation data.
- `y` - observed value of the dependent variable for `new_observation`. The use of this argument is illustrated in Section 13.1.
- `variables` - names of explanatory variables, for which CP profiles are to be calculated. By default, the profiles will be constructed for all variables, which may be time consuming.
- `variable_splits` - a list of values for which CP profiles are to be calculated. By default, these are all values for categorical variables. For continuous variables, uniformly-placed values are selected; one can specify the number of the values with the `grid_points` argument (the default is 101).

The code below allows to obtain the plots in the upper part of Figure 11.5. The argument `variable_splits` specifies the variables ( `age` and `fare` ) for which CP profiles are to be calculated, together with the list of values at which the profiles are to be evaluated.

```
cp_titanic_rf <- individual_profile(explain_rf_v6, henry,
              variable_splits = list(age = seq(0, 70, 0.1),
                                     fare = seq(0, 100, 0.1)))


plot(cp_titanic_rf, variables = c("age", "fare")) +
  ylim(0, 1) +
  ggtitle("Ceteris Paribus profile",
          "For the random forest model and the titanic dataset")
```

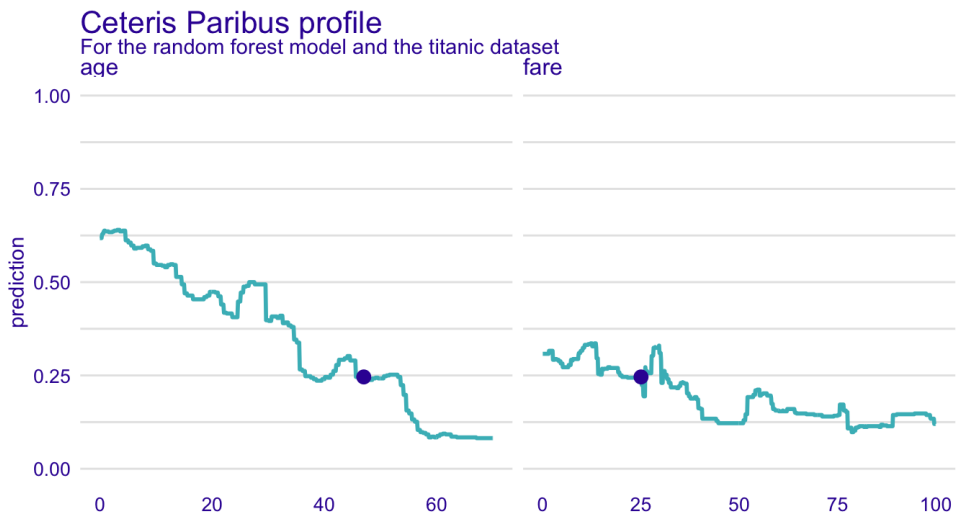



Figure 11.8: Ceteris-paribus profiles for `class` and `embarked` variables and the `titanic_rf_v6` model. Blue dot stands for `henry` .

To enhance the plot, additional functions can be used. The generic `plot()` function creates a `ggplot2` object with a single `geom_line` layer. Function `show_observations` adds `geom_point` layer, `show_rugs` adds `geom_rugs` , while `show_profiles` adds another `geom_line` . All these functions take, as the first argument, an object created with the `ceteris_paribus` function. They can be combined freely to superimpose profiles for different models or observations.

In the example below, we present the code to create CP profiles for two passengers, `henry` and `johny_d`. Their profiles are included in a plot presented in Figure 11.9. We use the `scale_color_manual` function to add names of passengers to the plot, and to control colors and positions.

```
johny_d <- archivist::aread("pbiecek/models/e3596")
cp_titanic_rf2 <- variable_profile(explain_rf_v6, rbind(henry, johny_d))


library(ingredients)
plot(cp_titanic_rf2, color = "_ids_", variables = c("age", "fare")) +
  scale_color_manual(name = "Passenger:", breaks = 1:2,
            values = c("#4378bf", "#8bdcbe"),
            labels = c("henry" , "johny_d")) +
  ggtitle("Ceteris Paribus profile",
            "For the random forest model and the titanic dataset")
```

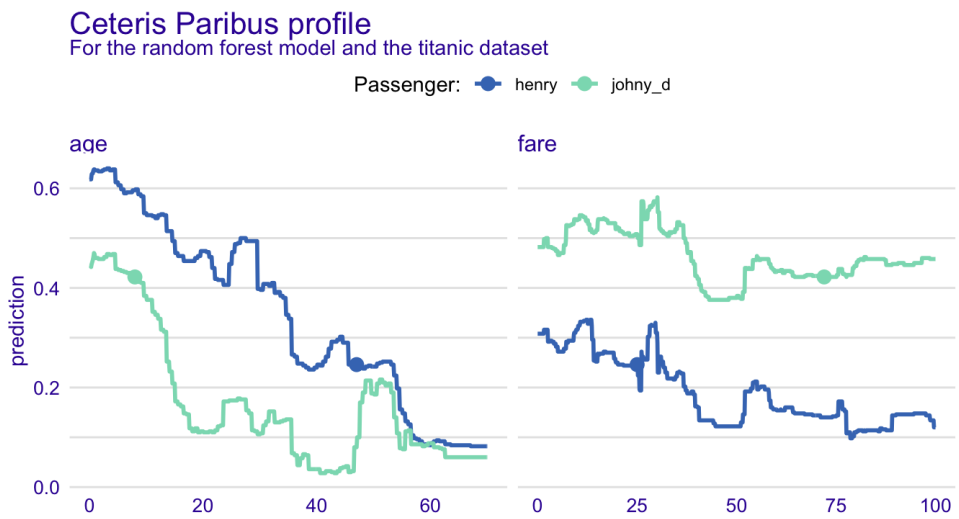



Figure 11.9: Ceteris-paribus profiles for the `titanic_rf_v6` model. Profiles for different passengers are color-coded.

## 11.6.3 Champion-challenger analysis

One of the most interesting uses of the explainers is comparison of CP profiles for two or more of models.

To illustrate this possibility, first, we have go to construct profiles for the models. In our illustration, for the sake of clarity, we limit ourselves just to two models: the logistic regression and random forest models for the Titanic data. Moreover, we only consider the `age` and `fare` variables. We use `henry` as the instance, for which predictions are of interest.

```
cp_titanic_rf <- ceteris_paribus(explain_rf_v6, henry)
cp_titanic_lmr <- ceteris_paribus(explain_lmr_v6, henry)
```

Subsequently, we construct the plot. The result is shown in Figure 11.10. Predictions for `henry` are slightly different, logistic regression returns in this case higher predictions then random forest. For `age` variable profiles of both models are similar, in both models we see decreasing dependency. While for `fare` the logistic regression model is slightly positive while random forest is negative. The larger the `fare` the larger is difference between these models. Such analysis helps us to which degree different models agree on what if scenarios.

Note that every `plot` and `show_*` function can take a collection of explainers as arguments. Profiles for different models are included in a single plot. In the presented R snippet, models are color-coded with the help of the argument `color = "_label_"` , where `_label_` refers to the name of the column in the CP explainer that contains the model label.

```
plot(cp_titanic_rf, cp_titanic_lmr, color = "_label_",
     variables = c("age", "fare")) +
     ggtitle("Ceteris Paribus Profiles for Henry")
```

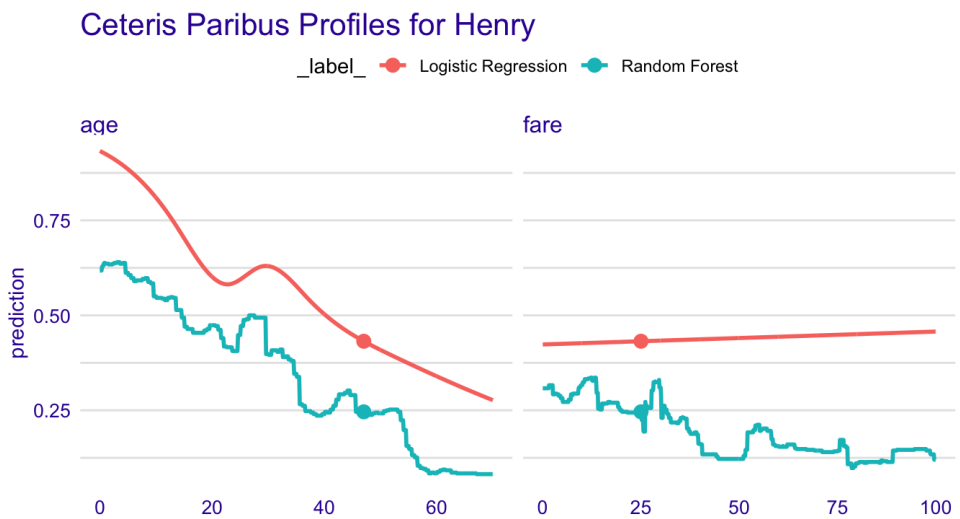



Figure 11.10: Champion-challenger comparison of the `titanic_lmr_v6` and `titanic_rf_v6` models. Profiles for different models are color-coded.

# References


Apley, Dan. 2018. *ALEPlot: Accumulated Local Effects (Ale) Plots and Partial Dependence (Pd) Plots*. https://CRAN.R-project.org/package=ALEPlot.

Biecek, Przemyslaw, Hubert Baniecki, Adam Izdebski, and Katarzyna Pekala. 2019. *ingredients: Effects and Importances of Model Ingredients*.

Goldstein, Alex, Adam Kapelner, Justin Bleich, and Emil Pitkin. 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. https://doi.org/10.1080/10618600.2014.907095.

Greenwell, Brandon M. 2017. "pdp: An R Package for Constructing Partial Dependence Plots." *The R Journal* 9 (1): 421–36. https://journal.r-project.org/archive/2017/RJ-2017-016/index.html.

Molnar, Christoph, Bernd Bischl, and Giuseppe Casalicchio. 2018. "iml: An R package for Interpretable Machine Learning." *Joss* 3 (26). Journal of Open Source Software: 786. https://doi.org/10.21105/joss.00786.

O'Connell, Mark, Catherine Hurley, and Katarina Domijan. 2017. "Conditional Visualization for Statistical Models: An Introduction to the Condvis Package in R." *Journal of Statistical Software, Articles* 81 (5): 1–20. https://doi.org/10.18637/jss.v081.i05.

Tufte, Edward R. 1986. *The Visual Display of Quantitative Information*. Cheshire, CT, USA: Graphics Press.

# 12 Ceteris-paribus Oscillations

## 12.1 Introduction

Visual examination of Ceteris-paribus (CP) profiles is insightful, but for a model with a large number of explanatory variables we may end up with a large number of plots which may be overwhelming. To prioritize between the profiles we need a measure that would summarize the impact of a selected variable on model's predictions. In this chapter we describe a solution closely linked with CP profiles. An alternative instance-level variable importance is discussed in the Chapters 7 (Break Down), 9 (SHAP) and 10 (LIME).

## 12.2 Intuition

To assign importance to CP profiles, we can use the concept of profile oscillations. In particular, the larger influence of an explanatory variable on prediction at a particular instance, the larger the fluctuations along the corresponding CP profile. For a variable that exercises little or no influence on model prediction, the profile will be flat or will barely change. In other words, the values of the CP profile should be close to the value of the model prediction for the particular instance. Consequently, the sum of differences between the profile and the value of the prediction, take across all possible values of the explanatory variable, should be close to zero. The sum can be graphically depicted by the area between the profile and the horizontal line representing the instance prediction. On the other hand, for an explanatory variable with a large influence on the prediction, the area should be large. Figure 12.1 illustrates the concept. Panel A of the Figure corresponds to the CP profiles presented in Figure 11.5. The larger the highlighted area in Figure 12.1, the more important is the variable for the particular prediction.


Figure 12.1: The value of the colored area summarizes the Ceteris-paribus-profile oscillations and provides the mean of the absolute deviations between the CP profile and the instance prediction. Panel A shows plots for continuous explanatory variables, while panel B shows plots for categorical variables in the `titanic_rf_v6` model.

## 12.3 Method

Let us formalize this concept now. Denote by $g^j(z)$ the probability density function of the distribution of the $j$-th explanatory variable. The summary measure of the variable's importance for model prediction at point $x_*$, $vip^j_{CP}(x_*)$, computed based on the variable's CP profile, is defined as follows:

$$vip^j_{CP}(x_*) = \int_{\mathcal{R}} |h^j_{x_*}(z) - f(x_*)| g^j(z) dz = E_{X^j}\left[|h^j_{x_*}(X^j) - f(x_*)|\right]. \qquad (12.1)$$

Thus, $vip^j_{CP}(x_*)$ is the expected absolute deviation of the CP profile from the model prediction for $x_*$ over the distribution $g^j(z)$ for the $j$-th explanatory variable.

The true distribution of $j$-th explanatory variable is, in most cases, unknown. Thus, there are several options how to calculate Equation (12.1).

One is to calculate just the area under the CP curve, i.e., to assume that $g^j(z)$ is a uniform distribution for the range of variable $x^j$. It folows then that a straightforward estimator of $vip_{CP}^{j,uni}(x_*)$ is

$$\widehat{vip}_{CP}^{j,uni}(x_*) = \frac{1}{k}\sum_{l=1}^{k}|h_{x_*}^j(z_l) - f(x_*)|, \tag{12.2}$$

where $z_l$ ($l = 1, \ldots, k$) are the selected values of the $j$-th explanatory variable. For instance, one can select use all unique values of $x^j$ in the considered dataset. Alternatively, for a continuous variable, one can use an equidistant grid of values.

Another approach is to use the empirical distribution for $x^j$. This leads to the estimator of $vip_{CP}^{j,emp}(x_*)$ defined as

$$\widehat{vip}_{CP}^{j,emp}(x_*) = \frac{1}{n}\sum_{i=1}^{n}|h_{x_*}^j(x_i^j) - f(x_*)|, \tag{12.3}$$

where index $i$ goes through all observations in a dataset.

The use of of $\widehat{vip}_{CP}^{j,emp}(x_*)$ is preferred when there are enough data to accurately estimate the empirical distribution and when the distribution is not uniform. On the other hand, $\widehat{vip}_{CP}^{j,uni}(x_*)$ is in most cases quicker to compute and, therefore, it is preferred if we look for fast approximations.

It is worth noting that the importance of an explanatory variable for instance prediction may be very different for different points $x_*$. For example, consider model

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

where $x_1$ and $x_2$ take values in $[0, 1]$. Consider prediction for an observation described by vector $x_* = (0, 1)$. In that case, the importance of $X_1$ is larger than $X_2$. This is because the CP profile for the first variable, given by the values of function $f(z, 1) = z$, will have oscillations. On the other hand, the profile for the second variable will show no oscillations, because the profile is given by function $f(0, z) = 0$. Obviously, the situation is reversed for $x_* = (1, 0)$.

## 12.4 Example: Titanic

Figure 12.2 provides a barplot of variable importance measures for different continuous explanatory variables for the random forest model `titanic_rf_v6` for `johny_d` .

The longer the bar, the larger the CP-profile oscillations for a particular explanatory variable. Thus, Figure 12.2 indicates that the most important variable for prediction for the selected observation are `gender` and `sibsp`, followed by `age`.

From the Ceteris Paribus one can read that if Henry were older, this would significantly lower the chance of survival. One the other hand, were Henry not travelling alone, this would increase the chance.

From the oscillation's plot one can only read which features are important but one cannot read how they influence the prediction. This is why profile oscillations shall be accompanied by Ceteris Paribus profiles.

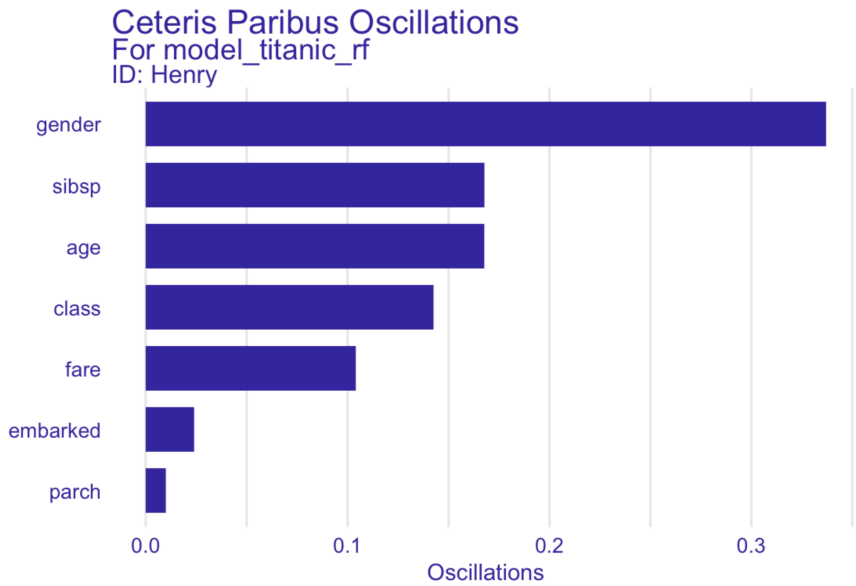



Figure 12.2: Variable-importance measures calculated for Ceteris-paribus oscillations for `johny_d` based on the `titanic_rf_v6` model.

## 12.5 Pros and cons

Oscillations of CP profiles are easy to interpret and understand. By using the average of oscillations, it is possible to select the most important variables for an instance prediction. This method can easily be extended to two or more variables. In such cases one needs to integrate the equation (12.2) over larger number of variables.

There are several issues related to the use of the CP oscillations. For example, the oscillations may not be of help in situations when the use of CP profiles may itself be problematic (e.g., in the case of correlated explanatory variables or interactions - see Section 11.5). An important

issue is that the CP based local variable importance do not sum up to the instance prediction for which they are calculated, opposite to Break Down (Chapter 7) and Shapley values (Chapter 9).

## 12.6 Code snippets for R

In this section, we present key features of R package `DALEX` which is a part of the `DrWhy.AI` universe and covers all methods presented in this chapter.

For illustration purposes we use the random forest model `titanic_rf_v6` (see Section 5.1.3). Recall that it is developed to predict the probability of survival from sinking of Titanic. Instance-level explanations are calculated for a single observation: `henry` - a 47-year-old passenger that travelled in the 1st class.

`DALEX` explainers for both models and the Henry data are retrieved via `archivist` hooks as listed in Section 5.1.8.

```
library("randomForest")
explain_rf_v6 <- archivist::aread("pbiecek/models/6ed54")

library("DALEX")
henry <- archivist::aread("pbiecek/models/a6538")
henry
```

### 12.6.1 Basic use of the `variable_attribution` function

To calculate CP oscillations, we have got to calculate CP profiles for the selected observation. We use `henry` as the instance prediction of interest.

CP profiles are calculated by applying the `variable_attribution()` function to the `explainer` object to calculate the oscillations and the estimated value of the variable-importance measure as in Equation (12.1).

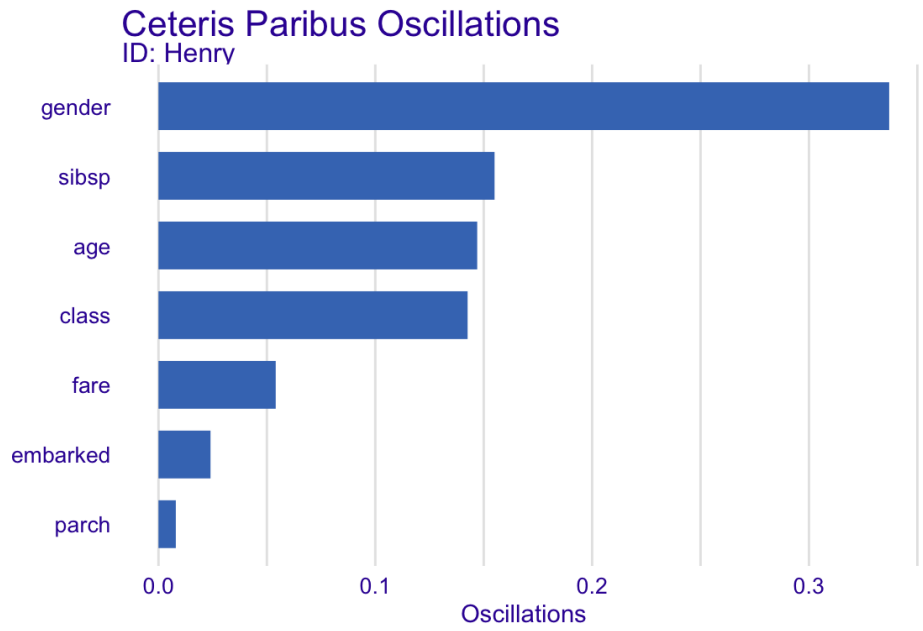
```
library("ingredients")
library("ggplot2")

(oscillations_titanic_rf <- variable_attribution(explain_rf_v6,
                                henry, type = "oscillations"))
```

```
##    _vname_ _ids_ oscillations
## 2   gender     1   0.33700000
## 4    sibsp     1   0.15500000
## 3      age     1   0.14700000
## 1    class     1   0.14257143
## 6     fare     1   0.05407273
## 7 embarked     1   0.02400000
## 5    parch     1   0.00800000
```

Note that, by default, `variable_attribution(type = "oscillations")` estimates $vip^j_{CP}(x_*)$ by $\widehat{vip}^{j,uni}_{CP}(x_*)$, given in (12.2), using all unique values of the explanatory variable as the grid points.

The `variable_attribution(type = "oscillations")` function returns an object of class `ceteris_paribus_oscillations`, which has a form of a data frame, but has also an overloaded `plot()` function. We can use the latter function to plot the local variable-importance measures for the instance of interest.

```
oscillations_titanic_rf$`_ids_` <- "Henry"
plot(oscillations_titanic_rf) + ggtitle("Ceteris Paribus Oscillations")
```




## 12.6.2 Advanced use of the `variable_attribution` function

As mentioned in the previous section, `variable_attribution()` estimates $vip^{j}_{CP}(x_*)$ by $\widehat{vip}^{j,uni}_{CP}(x_*)$ using all unique values of the explanatory variable as the grid points. However, other approaches are also possible.

One is to use $\widehat{vip}^{j,uni}_{CP}(x_*)$, but assuming an equi-distant grid of values for a continuous explanatory variable. Toward this aim, we have got to explicitly specify a dense uniform grid of values for such a variable. The `variable_splits` argument can be used for this purpose.

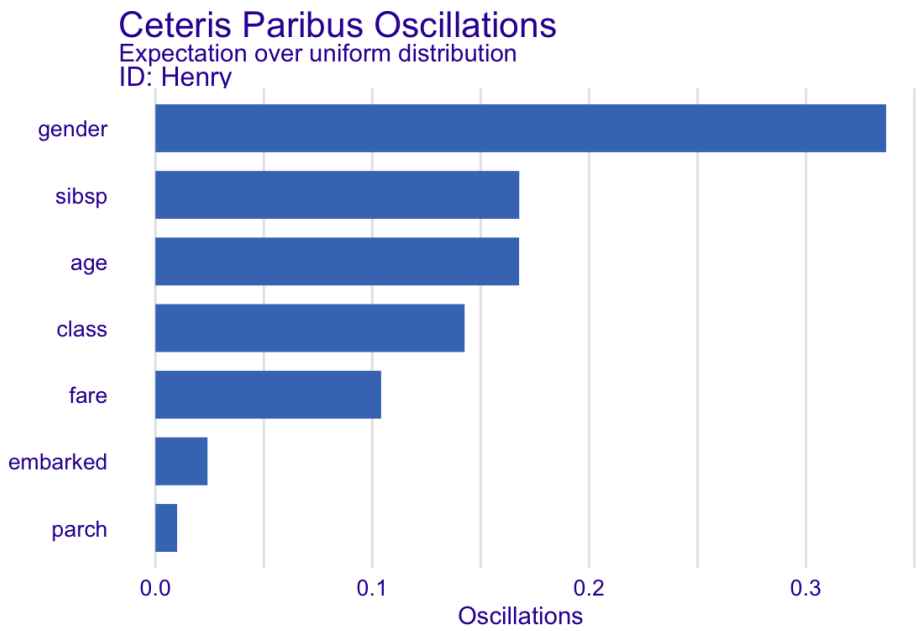
```
oscillations_uniform <- variable_attribution(explain_rf_v6, henry,
              variable_splits = list(age = seq(0, 65, 0.1),
                                     fare = seq(0, 200, 0.1),
                                     sibsp = seq(0, 8, 0.1),
                                     parch = seq(0, 8, 0.1),
                                     gender = unique(titanic$gender),
                                     embarked = unique(titanic$embarked),
                                     class = unique(titanic$class)),
              type = "oscillations")
```

Subsequently, we apply the `calculate_oscillations()` function to compute the oscillations and the variable-importance measures.

```
oscillations_uniform$`_ids_` <- "Henry"
oscillations_uniform
```

```
##    _vname_ _ids_ oscillations
## 5   gender Henry    0.3370000
## 3    sibsp Henry    0.1677778
## 1      age Henry    0.1677235
## 7    class Henry    0.1425714
## 2     fare Henry    0.1040790
## 6 embarked Henry    0.0240000
## 4    parch Henry    0.0100000
```

```
plot(oscillations_uniform) +
    ggtitle("Ceteris Paribus Oscillations",
            "Expectation over uniform distribution")
```

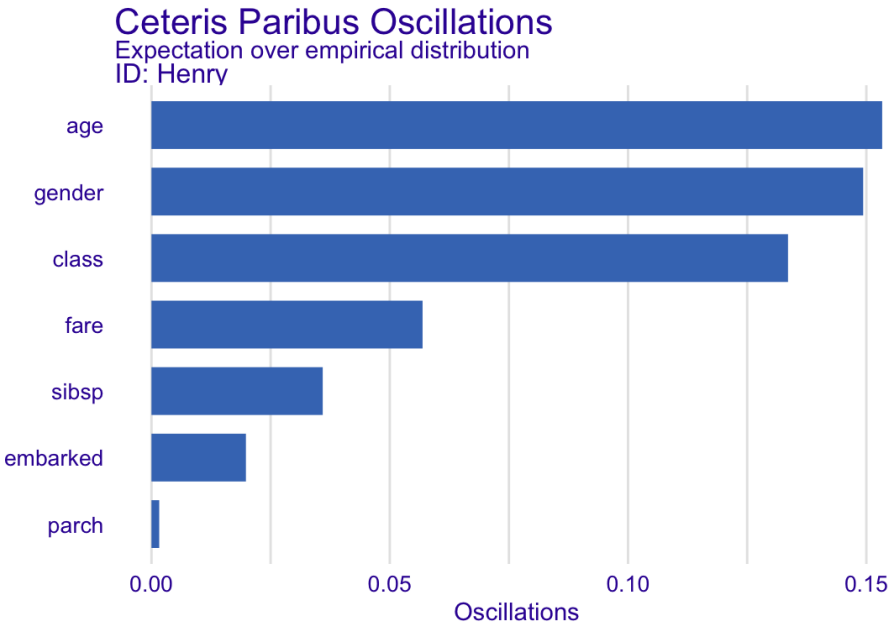



Another approach is to calculate the expectation (12.1) over the empirical distribution of a variable, i..e, to use $\widehat{vip}_{CP}^{j,emp}(x_*)$, given in (12.3). Toward this aim, we use the `variable_splits` argument to explicitly specify the validation-data sample to define the grid of values.

```r
titanic <- na.omit(titanic)

oscillations_empirical <- variable_attribution(explain_rf_v6, henry,
              variable_splits = list(age = titanic$age,
                                     fare = titanic$fare,
                                     sibsp = titanic$sibsp,
                                     parch = titanic$parch,
                                     gender = titanic$gender,
                                     embarked = titanic$embarked,
                                     class = titanic$class),
              type = "oscillations")


oscillations_empirical$`_ids_` <- "Henry"
oscillations_empirical
```

```
##    _vname_ _ids_ oscillations
## 1      age Henry  0.153323969
## 5   gender Henry  0.149336656
## 7    class Henry  0.133567739
## 2     fare Henry  0.056883552
## 3    sibsp Henry  0.035932034
## 6 embarked Henry  0.019818758
## 4    parch Henry  0.001623924
```

```r
plot(oscillations_empirical) +
    ggtitle("Ceteris Paribus Oscillations",
              "Expectation over empirical distribution")
```


Ceteris Paribus Oscillations
Expectation over empirical distribution
ID: Henry
age
gender
class
fare
sibsp
embarked
parch
0.00
0.05
0.10
0.15
Oscillations

# 13 Local Diagnostics Plots

## 13.1 Introduction

It may happen that, while the global predictive performance of a model is good, the model predictions for some observations are very misfitted. We often say that the model does not cover well some areas of the input space.

For example, a model calibrated for typical patients in a certain hospital may not do well with exceptionally young patients. Or a model calibrated for the credit risk of spring holiday consumer loans may not work well on a group of autumn loans for Christmas holiday gifts. For this reason, we should not be satisfied with global measures of model performance. For important decisions, it is good to check how the model behaves for observations similar to the instance of interest.

In this chapter, we present two local-diagnostics techniques that address this issue, namely, *local fidelity plots* that show local performance around observation of interest and *local stability plots* that show the local stability around observation of interest.

The general idea behind fidelity plots is to select a number of observations ("neighbors") from the validation dataset that are closest to the instance (observation) of interest. Then, for the selected observations, we plot CP profiles and check how stable they are. Additionally, if we know true values of the dependent variable for the selected neighbors, we may add residuals to the plot to evaluate the local fit of the model.

## 13.2 Intuition

Assume that we have identified a set of observations from the training data similar in terms of dependent variables to the observation of interest. The basic idea behind local fidelity plots is to compare distribution of residuals for these similar cases against distribution of all residuals.

Figure 13.1 presents histograms of residuals for the entire dataset and the selected neighbors for the random forest model for the Apartments dataset (Section 5.2.3). The distribution of residuals for the entire dataset is rather symmetric and centered around 0, suggesting a reasonable average performance of the model. On the other hand, the residuals for the

selected neighbors are centered around the value of 500. This suggests that for the apartment of interest around this apartment the model is biased towards values smaller than observed (residuals are positive, so on average $y$ is higher than $\hat{y}$, see (2.1)).

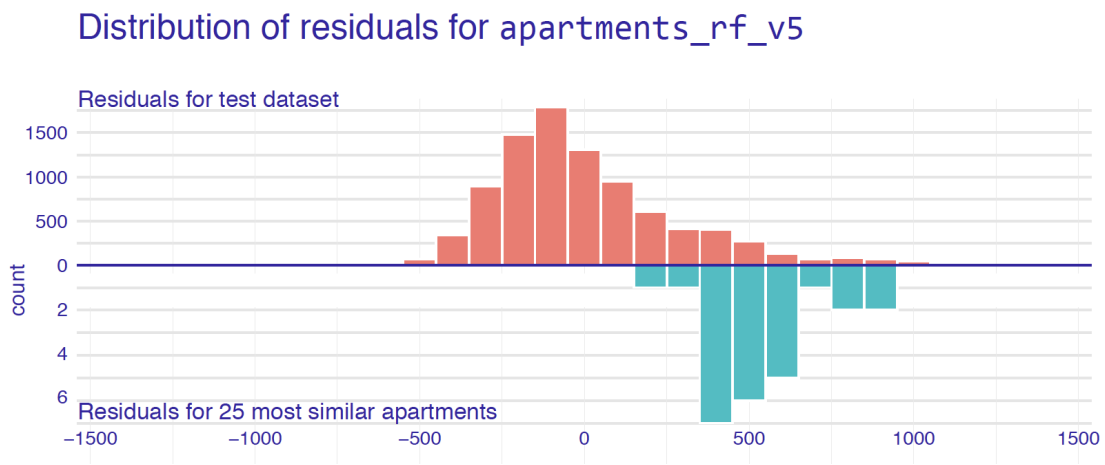



Figure 13.1: Histograms of residuals for the `apartments_rf_v5` model for the Apartments dataset. Upper panel: residuals calculated for all observations from the dataset. Bottom panel: residuals calculated for 25 nearest neighbors of the instance of interest.

Another approach to local model diagnostics is to examine how stable is model behaviour around the observation of interest. Figure 13.2 presents CP profiles for variable `age` for the instance of interest and its 10 nearest neighbors for the random forest model for the Titanic dataset (Section 5.1.3). The profiles are almost parallel and very close to each other. This suggests that model predictions are stable around the instance of interest, because small changes in the explanatory variables (represented by the nearest neighbors) have not got much influence on the predictions.

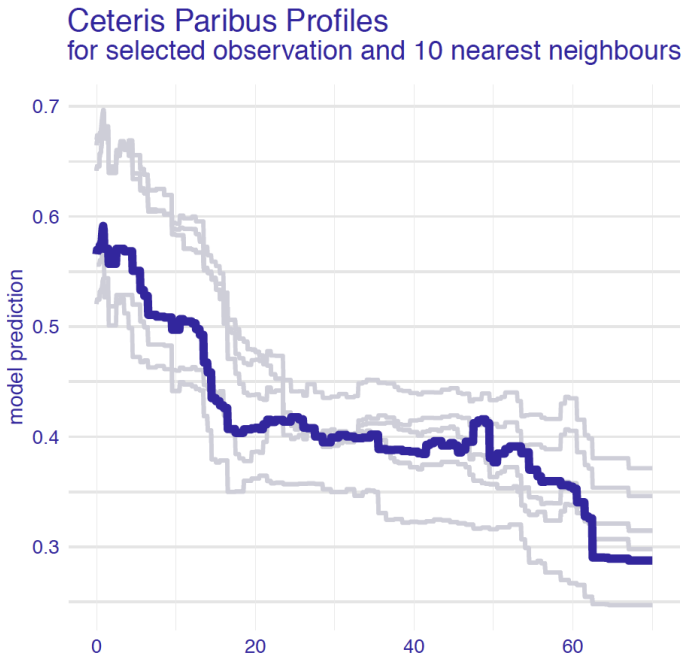



Figure 13.2: Ceteris-paribus profiles for a selected instance (dark violet line) and 10 nearest neighbors (light grey lines) for the `titanic_rf_b6` model. The profiles are almost parallel and close to each other what suggests the stability of the model.

Of course CP profiles for different variables may be very different so a natural question arises which variables shall we examine. The most natural choice is to explore the most important variables according to results from the Break Down, SHAP, LIME od CP Oscillations methods.

## 13.3 Method

The proposed method is based on three steps:

- first, we need to select observations nearest to the observation of interest,
- for fidelity analysis we need to calculate and compare residuals for the neighbors.
- for stability analysis we need to calculate and visualize CP profiles for the selected neighbors.

In what follows we discuss each of the elements in more detail.

### 13.3.1 Nearest neighbors

There are two important questions related to the selection of the neighbors “nearest” to the instance (observation) of interest:

- How many neighbors should we choose?
- What metric should be used to measure the “proximity” of observations?

The answer to both questions is *it depends*.

- The smaller the number of neighbors, the more local is the analysis. However, a very small number will lead to a larger variability of the results. In many cases we found that 20 neighbors works fine. However, one should always take into account computational time (smaller number of neighbors results in quicker calculations) and the size of the dataset (for a small dataset, smaller sets of neighbors may be preferred).
- The metric is very important. The more explanatory variables, the more important is the choice. In particular, the metric should be capable of accommodating variables of different nature (categorical, continuous). Our default choice is the Gower similarity measure:

$$d_{gower}(x_i, x_j) = \frac{1}{p}\sum_{k=1}^{p} d^k(x_i^k, x_j^k),$$

where $x_i$ is a $p$-dimensional vector of explanatory covariates for the $i$-th observation and $d^k(x_i^k, x_j^k)$ is the distance between values of the $k$-th variable for the $i$-th and $j$-th observations. Note that $d^k()$ depends on the nature of the variable. For instance, for a continuous variable it is equal to $|x_i^k - x_j^k|/\{max(x_1^k, \ldots, x_n^k) - min(x_1^k, \ldots, x_n^k)\}$, i.e., the absolute difference scaled by the observed range of the variable. On the other hand, for a categorical variable, it is simply $I(x_i^k = x_j^k)$, where $I()$ is the indicator function. Note that $p$ may be equal to the number of all explanatory variables included in the model, or only a subset of them. An advantage of Gower similarity measure is that it “deals” with heterogeneous vectors with both categorical and continuous variables. The disadvantage of Gower similarity measure is that it does not take into account neither variable correlation nor variable importance. For high dimensional setting an interesting alternative would be the proximity measure in Random Forest (Breiman 2001). It takes into account variable importance but requires a fitted Random Forest model.

Once we have decided on the number of neighbors, we can use the chosen metric to select the required number observations “closest” to the one of interest.

## 13.3.2 Local-fidelity plot

Figure 13.1 illustrates two distribution of residuals, for the whole dataset and for neighbours of the observation of interest.

For a typical observation these two distributions shall be similar. An alarming situation would be if the residuals for neighbours will be shifted towards the extremely positive of negative values.

Apart from visual examination we may also use some statistical tests that compares these two distributions. Since we cannot assume any distribution for residuals we can use a nonparametric test like Wilcoxon test or Kolmogorov-Smirnov test.

[TODO: maybe we need a better test for the stochastic dominance, or it is enough to have a test for location parameter?]

### 13.3.3 Local-stability plot for neighbors

Once nearest neighbors have been identified, we can graphically compare CP profiles for selected (or all) variables.

For a model with a large number of variables, we may end up with a large number of plots. In such a case a better strategy is to focus only on $K$ most important variables, selected by using the variable-importance measure (see for example Chapter 12).

CP profiles are helpful to assess the model stability. In addition, we can enhance the plot by adding residuals to it to allow evaluation of the local model fit. For model $f()$ and observation $i$ described by the vector of explanatory variables $x_i$, the residual is the difference between the observed and predicted value of the dependent variable $Y_i$. Let us recall the definition (2.1):

$$r_i = y_i - f(x_i).$$

Note that, for a binary variable, the residual is the difference between the value of 0 or 1, depending on how we code “success,” and the value of the predicted probability of “success.” This definition also applies to categorical responses, as it is common to define, in such case, a binary “success” indicator and compute the predicted probability of “success” for each category separately.

The plot that includes CP profiles for the nearest neighbors and the corresponding residuals is called a local-fidelity plot. See an example in Figure 13.2.

## 13.4 Example: Titanic

As an example, we will use the predictions for the random forest model for the Titanic data (see Section 5.1.3).

Figure 13.3 presents a detailed explanation of the elements of a local-fidelity plot for *age*, a continuous explanatory variable. The plot includes eight nearest neighbors of Henry (see Section 5.1.6). Profiles are quite apart from each other, which indicates potential instability of model predictions. However, the residuals included in the plots are positive and negative, indicating that, on average, the instance prediction should not be biased.

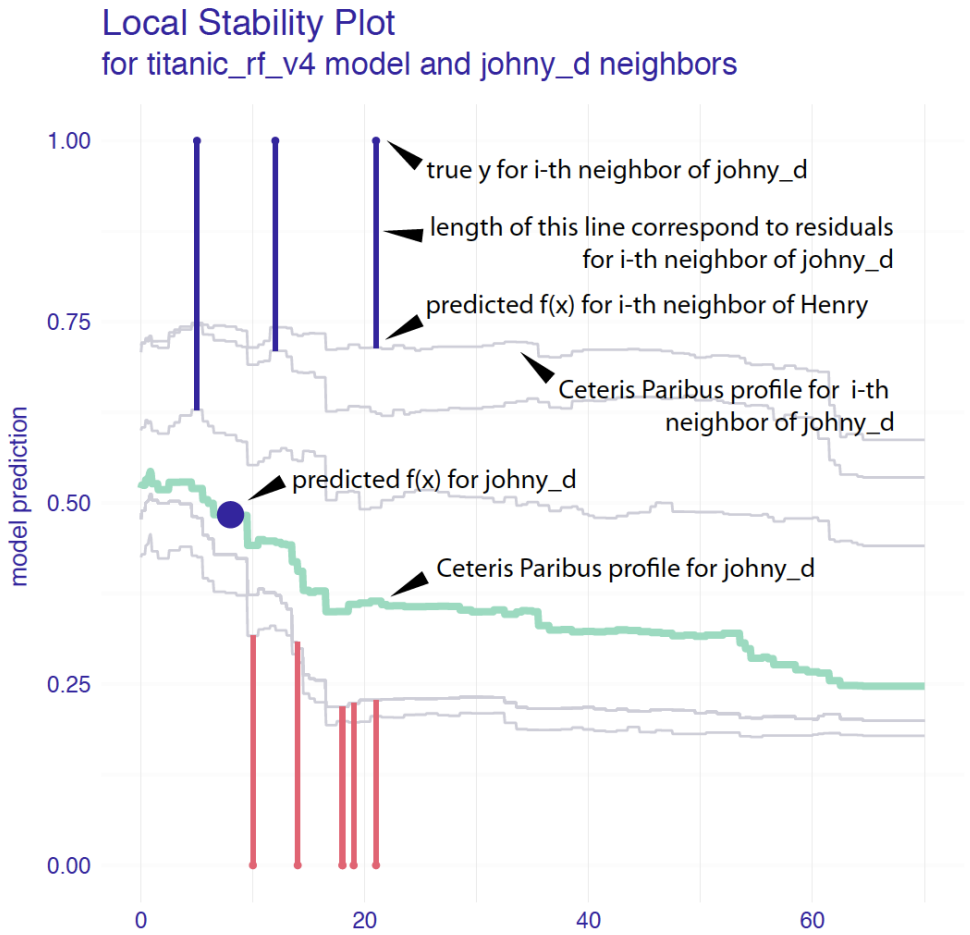



Figure 13.3: Elements of a local-stability plot for a continuous explanatory variable. The green line shows the Ceteris-paribus profile for the instance of interest. Profiles of the nearest neighbors are marked with grey lines. The vertical intervals correspond to residuals; the shorter the interval, the smaller the residual and the more accurate prediction of the model. Blue intervals correspond to positive residuals, red intervals to negative intervals. Stable model will have profiles close to each other; additive model will have parallel lines.

Figure 13.4 presents a local-fidelity plot for the categorical explanatory variable `class`. Henry and his neighbors traveled in the `1st` class. In different panels we see how the predicted probability of survival changes if the `1st` class is replaced, for instance, by the

`2nd` (in most cases, they probability will be reduced) or the `deck crew` (in most cases, the probability will increase). Such plots can help to detect interactions, as we see that the same change (let's say, from the `1st` to the `3rd` class) results in a different change of the model prediction.

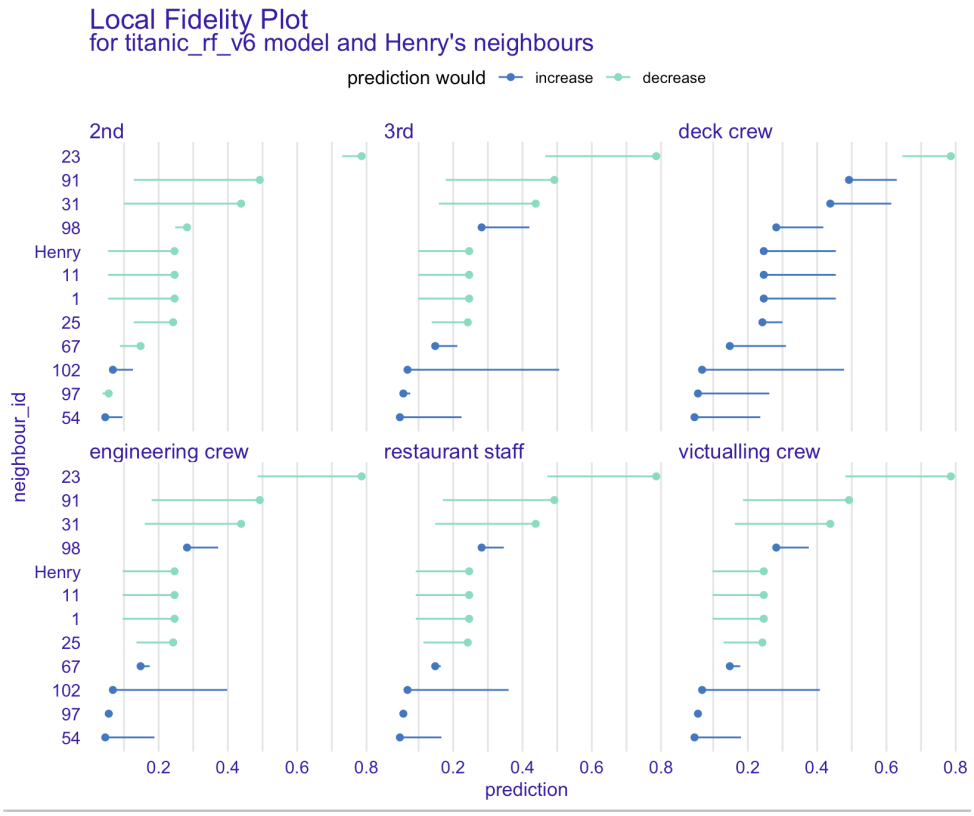



Figure 13.4: The local-stability plot for the categorical explanatory variable `class` in the random effects model for the Titanic data, `johny_d` , and his 10 neighbors. Each panel indicates how the model prediction would change if the class changed from `1st` to another one. Dots indicate original model predictions for the neighbors; the end of the interval corresponds to model prediction after changing the class. The top-lef panel indicates that, for the majority of the neighbors, the change from the `1st` to the `2nd` class reduces the predicted value of the probability of survival. On the other hand, the top-right panel indicates that changing the lass to `deck crew` members increases the predicted probability.

## 13.5 Pros and cons

Local fidelity and stability plots may be very helpful to check if

- the model is locally additive, as for such models the CP profiles should be parallel;
- the model is locally stable, as in that case the CP profiles should be close to each other;

- the model fit for the instance of interest is good, as in that case the residuals should be small and their distribution should be balanced around 0.

The drawback is that such plots are quite complex and lack objective measures of the quality of the model fit. Thus, they are mainly suitable for an exploratory analysis.

## 13.6 Code snippets for R

In this section, we show how to use the R package `DALEX` (Biecek 2018) to construct local-fidelity plots.

We use the random forest model `titanic_rf_v6` developed for the Titanic dataset (see Section 5.1.3 ) as the example. Recall that we try to address a classification problem for a binary dependent variable - we want to predict the probability of survival for a selected passenger.

`DALEX` explainers for the model and the `henry` data frame are retrieved via `archivist` hooks, as listed in Section 5.1.8.

```
library("randomForest")
library("DALEX")
explain_rf_v6 <- archivist::aread("pbiecek/models/6ed54")
henry <- archivist::aread("pbiecek/models/a6538")
henry
```

```
##   class gender age sibsp parch fare  embarked
## 1   1st   male  47     0     0   25 Cherbourg
```

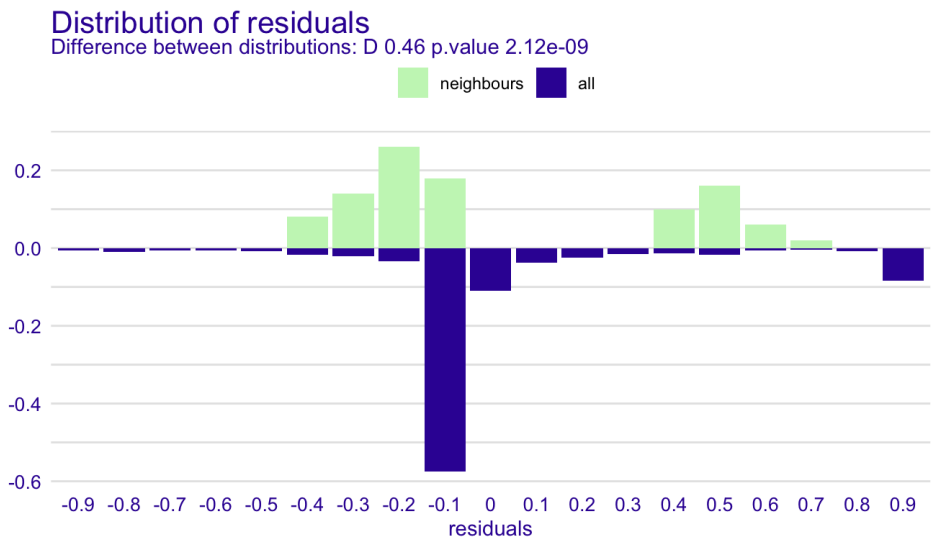
We will show how to construct fidelity plot as in Figure 13.2. Toward this aim we need some number of passengers most similar to `henry` . Here we are using `individual_diagnostics` function from the `DALEX` package. First argument is an explainer, second the instance of interest, optional arguments are `neighbours` (number of neighbours) and `distance` (by default, the Gower distance is used).

This function needs to calculate residuals, so explainer shall be created with the `y` argument and also the `residual_function` argument.

```
id_rf_v6 <- individual_diagnostics(explain_rf_v6,
                           henry,
                           neighbours = 100)
id_rf_v6
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  residuals_all and residuals_sel
## D = 0.45973, p-value = 2.116e-09
## alternative hypothesis: two-sided
```

```
plot(id_rf_v6)
```



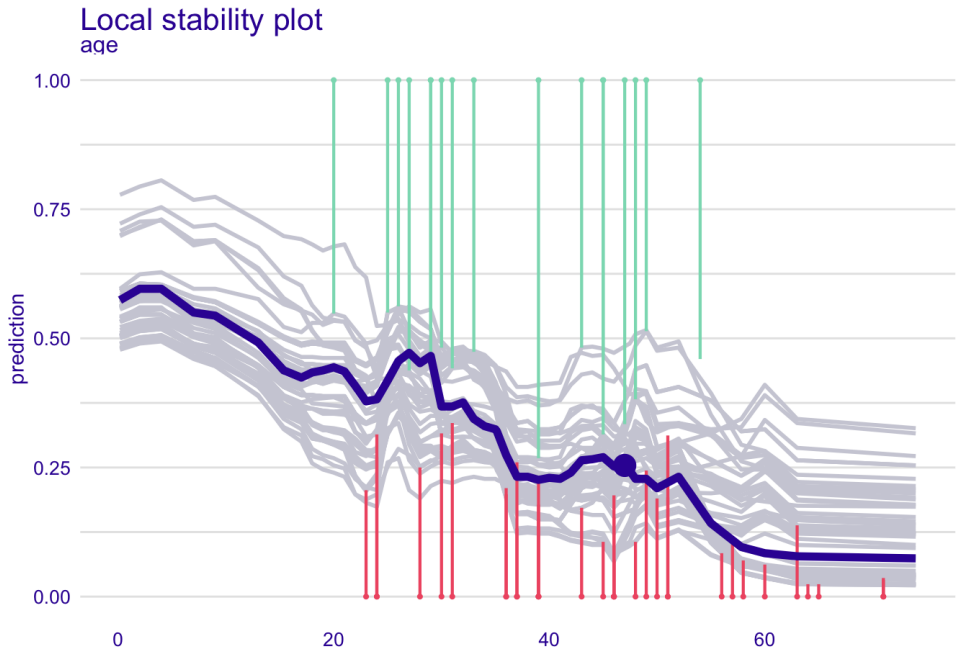

The function `individual_diagnostics()` can be also used for a local stability plot as in Figure 13.3 and **??**. To do this we need to also specify an `variables` argument.

Toward this aim, we use the `y` argument in the `individual_profile()` function. The argument takes numerical values. Our binary dependent variable `survived` assumes values `yes/no` ; to convert them to numerical values, we use the `survived == "yes"` expression.

```r
id_rf_v6 <- individual_diagnostics(explain_rf_v6,
                           henry,
                         neighbours = 10,
                         variables = "age")
id_rf_v6
```

```
## Top profiles    :
##     class gender        age sibsp parch fare  embarked _yhat_ _vname_
## 1     1st   male  0.1666667     0     0   25 Cherbourg  0.574     age
## 1.1   1st   male  2.0000000     0     0   25 Cherbourg  0.596     age
## 1.2   1st   male  4.0000000     0     0   25 Cherbourg  0.596     age
## 1.3   1st   male  7.0000000     0     0   25 Cherbourg  0.550     age
## 1.4   1st   male  9.0000000     0     0   25 Cherbourg  0.544     age
## 1.5   1st   male 13.0000000     0     0   25 Cherbourg  0.492     age
##     _ids_       _label_
## 1       1 Random Forest
## 1.1     1 Random Forest
## 1.2     1 Random Forest
## 1.3     1 Random Forest
## 1.4     1 Random Forest
## 1.5     1 Random Forest
##
##
## Top observations:
##   class gender age sibsp parch fare  embarked _yhat_       _label_ _ids_
## 1   1st   male  47     0     0   25 Cherbourg  0.254 Random Forest     1
```

```r
plot(id_rf_v6)
```

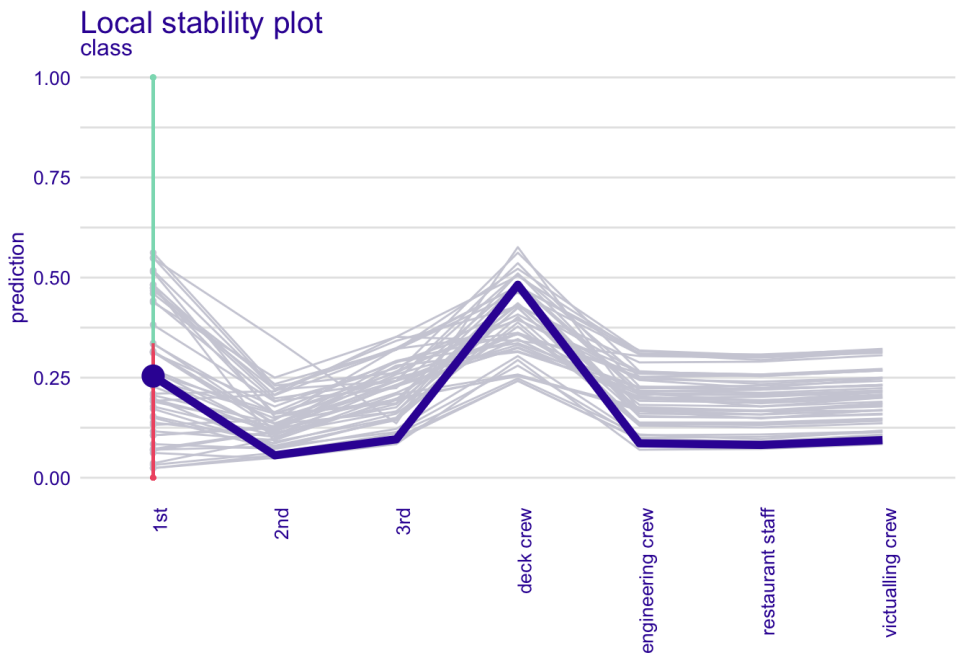



As we see the 10 passengers closest to `henry` are all from the `1st` class with age span between 20 and 60. Profiles for both `age` and `class` looks stable.

```
id_rf_v6 <- individual_diagnostics(explain_rf_v6,
                                   henry,
                                neighbours = 10,
                                variables = "class")
id_rf_v6
```

```
## Top profiles    :
##                class gender age sibsp parch fare  embarked _yhat_ _vname_
## 1                1st   male  47     0     0   25 Cherbourg  0.254   class
## 1.1              2nd   male  47     0     0   25 Cherbourg  0.056   class
## 1.2              3rd   male  47     0     0   25 Cherbourg  0.096   class
## 1.3        deck crew   male  47     0     0   25 Cherbourg  0.482   class
## 1.4 engineering crew   male  47     0     0   25 Cherbourg  0.086   class
## 1.5 restaurant staff   male  47     0     0   25 Cherbourg  0.082   class
##     _ids_       _label_
## 1       1 Random Forest
## 1.1     1 Random Forest
## 1.2     1 Random Forest
## 1.3     1 Random Forest
## 1.4     1 Random Forest
## 1.5     1 Random Forest
##
##
## Top observations:
##   class gender age sibsp parch fare  embarked _yhat_       _label_ _ids_
## 1   1st   male  47     0     0   25 Cherbourg  0.254 Random Forest     1
```

```
plot(id_rf_v6)
```



# References

Biecek, Przemyslaw. 2018. *DALEX: Explainers for Complex Predictive Models in R*. *Journal of Machine Learning Research*. Vol. 19. http://jmlr.org/papers/v19/18-416.html.

Breiman, Leo. 2001. “Random Forests.” In *Machine Learning*, 45:5–32. https://doi.org/10.1023/a:1010933404324.

# 14 Summary of Instance-level Explainers

In the first part of the book, we introduced a number of techniques for exploration and explanation of model predictions for individual instances. In each chapter we introduced and presented a single technique. But in practice these techniques rarely shall be used separately. It's more informative to combine different views offered by each technique into a more holistic overview.

See an example in Figure 14.1. Four different approaches to the explanation of the random forest model are used. First row shows results from variable attribution methods like LIME, SHAP and Break Down. All these method agree that the most important variables for `johny_d` are his `age`, `gender`, `class` and `fare`. Since `fare` and `class` are correlated and possibly `age` is in the interaction with `gender` then the additive decomposition is ambiguous. Second row shows Ceteris Paribus profiles for these four most important variables. We see that higher age or being in the `2nd` class in he `restaurant staff` would decrease the model response while lower fare (which is counter intuitive), being a female or in the `deck crew` would increase the model response. Third row show univariate distributions of particular variables. We see that `fare=72` is very high as for a ticket and that only small fraction of people on the titanic were children (no kids in the crew). Combination of different perspectives supplement each other.

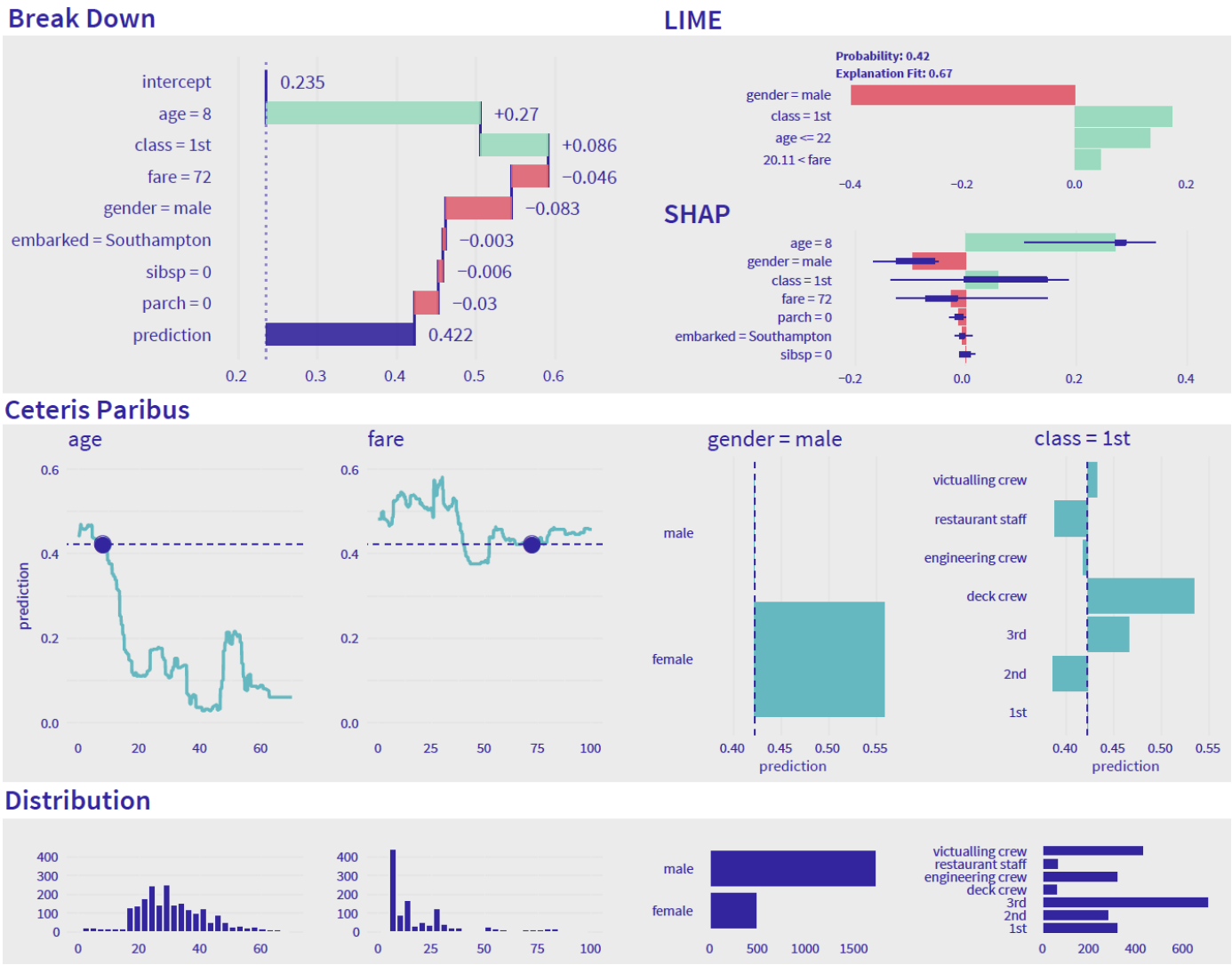



Figure 14.1: Instance-level explanations of the fandom forest model for the titanic data and `johny_d` an 8-years old boy that travels in the 1ts class.

In the Chapter 21 we show an example how instance level explanations may be combined with dataset level explanations on a new use-case related to FIFA 19 data.

On one hand it is good to supplement different techniques for explanation with each other, but on another hand these techniques are different and may be more or less suitable for some selected problems. Below we discuss some differences.

# 14.1 Number of explanatory variables in the model

One of the most important criteria for selection of model exploration and explanation methods is the number of explanatory variables in the model.

### 14.1.1 Low to medium number of explanatory variables

A low number of variables usually implies that the particular variables have a very concrete meaning and interpretation. An example are models for the Titanic data presented in Sections 5.1.2-5.1.4.

In such a situation, the most detailed information about the influence of the variables on the model predictions is provided by the CP profiles. In particular, the variables that are most influential for model predictions are selected by considering CP-profile oscillations (see Chapter 12) and then illustrated graphically with the help of individual-variable CP profiles (see Chapter 11).

### 14.1.2 Medium to large number of explanatory variables

In models with a medium or large number of variables, it is still possible that most (or all) of them are interpretable. An example of such a model is a car-insurance pricing model in which we need to estimate the value of an insurance based on behavioral data that includes 100+ variables about characteristics of the driver and characteristics of the car.

When the number of explanatory variables increases, it becomes harder to show CP profile for each individual variable. In such situation, the most common approach is to use BD plots, presented in Chapter 7, or plots of Shapley values, discussed in Chapter 9). They allow a quick evaluation whether a particular variable has got a positive or negative effect on model's prediction; we can also judge the size of the effect. If necessary, it is possible to limit the plots only to the variables with the largest effects.

### 14.1.3 Very large number of explanatory variables

When the number of explanatory variables is very large, it may be difficult to interpret the role of each single variable. An example of such situation are models for processing of images or texts. In that case, explanatory variables may be individual pixels in image processing or individual characters in text analysis. As such, their individual interpretation is limited. Due to additional issues with computational complexity, it is not feasible to use CP profiles, BD plots,

nor Shapley values to evaluate influence of individual values on model's predictions. Instead, the most common approach is to use LIME, presented in Chapter 10, which works on context-relevant groups of variables.

## 14.2 Correlated explanatory variables

When we derived some properties for presented methods we assumed that explanatory variables are independent. Obviously, this is not always the case. For instance, in the case of the data on apartment prices (see Chapter 5.2), the number of rooms and surface of an apartment will most likely be positively associated same is true for the class variable and fare for titanic data.

Of course all presented methods can be applied for correlated features, however sometimes it may be harder to analyze these features independently from each other.

To address the issue, the two most common approaches are: * to create new features that are independent (sometimes it is possible due to domain knowledge; sometimes it can e achieve by using principal components analysis or a similar technique), * construct two-dimensional extensions for CP plots (model response is plotted as a 2d surface) or permute variables in blocks to preserve the correlation structure of variables.

## 14.3 Models with interactions

In models with interactions, the effect of one explanatory variable may depend on values of other variables. For example, the probability of survival on Titanic may decrease with age, but the effect may be different for different classes of passengers. In such a case, to explore and explain model's predictions, we have got to consider not individual variables, but sets of variables included in interactions. To identify interactions, we can use BD plots as described in Chapter 8. To show effects of an interaction we may use a set of CP profiles. For the Titanic example we may use CP profiles for age with to instances that differ only in gender. The less parallel are such profiles the higher the effect of an interaction.

## 14.4 Sparse explanations

Predictive models may use hundreds of explanatory variables to yield a prediction for a particular instance. However, for a meaningful interpretation and illustration, most of human beings can handle only a very limited (say, less than 10) number of variables. Thus, sparse explanations are of interest. The most common method that is used to construct such explanations is LIME (Chapter 10). However, constructing a sparse explanation for a complex model is not trivial and may be misleading. Hence, care is needed when applying LIME to very complex models.

## 14.5 Additional uses of model exploration and explanation

In the previous chapters we focused on the application of the presented methods to exploration and explanation of predictive models. However, the methods can also be used to other aims:

- Model improvement. If a model prediction is particularly bad for a selected observation, then the investigation of the reasons for such a bad performance may provide some hints about how to improve the model. In case of instance predictions it is easier to note that a selected explanatory variable should have a different effect than the observed one.
- Additional domain-specific validation. Understanding which factors are important for model predictions helps in evaluation of the plausibility of the model. If the effects of some variables on the predictions are inconsistent with the domain knowledge, then this may provide a ground for criticising the model and, eventually, replacing it by another one. On the other hand, if the influence of the variables on model predictions is consistent with prior expectations, the user may become more confident with the model. Such a confidence is fundamental when the model predictions are used as a support for taking decisions that may lead to serious consequences, like in the case of, for example, predictive models in medicine.
- Model selection. In case of multiple candidate models, one may use results of the model explanation techniques to select one of the candidates. It is possible that, even if two models are similar in terms of a global model fit, the fit of one of them is locally much better. Consider the following, highly hypothetical example. Assume that a model is sought to predict whether it will rain on a particular day in a region where it rains on a half of the days. Two models are considered: one which simply predicts that it will rain every other day, and another that predicts that it will rain every day since October till March.

Arguably, both models are rather unsophisticated (to say the least), but they both predict that, on average, half of the days will be rainy. However, investigation of the instance predictions (for individual days) may lead to a preference for one of them.

## 14.6 Champion Challenger analysis

The techniques for explaining and exploring models have many applications. One of them is the opportunity to compare models.

Why compare models? One scenario is the Champion-Challenger analysis. Let's assume that some institution uses a predictive model but wants to know if they could get a better model using other modeling techniques. For example, the risk department in a bank uses logistical regression to assess credit risk. The model has some efficiency and is the so-called champion - the best model considered in the class of logistic regression models. However, it is worth checking whether using more complex models, so called challengers, e.g. boosting or random trees, will not be more effective. And if they are more effective, the question will arise as to how these challengers differ from the champion.

Another reason why we want to compare models is because of the iterativness of the modeling process itself (see 2.2). During the modeling process many versions of the models are created, often with different structures, sometimes with very similar efficiency. Comparative analysis allows for better understanding how these models differ from each other.

Below is an example of champion-challenger analysis for Random Forest model `model_titanic_rf` , logistic regression model `model_titanic_lmr` , boosting model of `model_titanic_gbm` and support-vector machines (SVM) model of `model_titanic_svm` .

Each of these models has a different way of functioning. Random forest and boosting models are on trees, so the response curves will be stepped one. The logistic regression and booster models have continuous and smooth response curves.

Figure 14.2 shows the Shapley values for the four models built in chapter 5.1 using the example of `john_d` . For three models, namely random forest, boosting and logistic regression, similar variables are indicated as important: `class` , `age` and `gender` . For the SVM model the most important variable is `gender` , followed by `age` and `parch` .

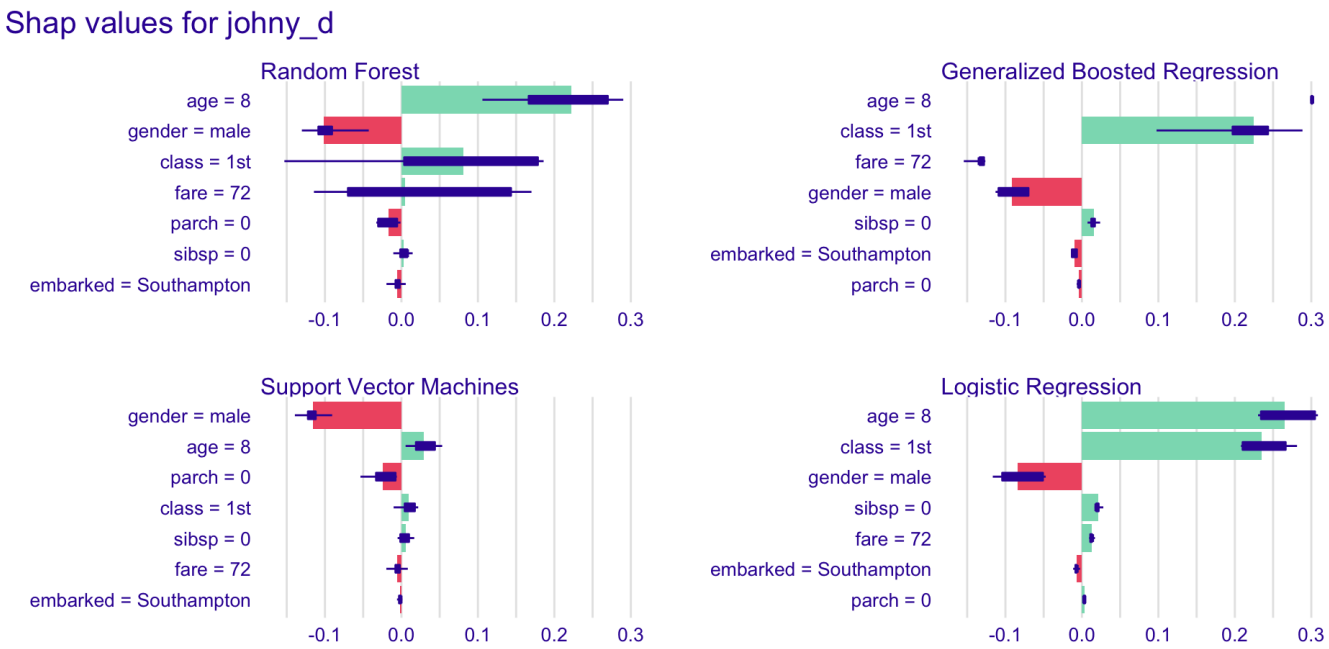



Figure 14.2: SHAP plots for four different models for the Titanic data.

Shapley values show an additive distribution for model predictions. In the chapter 8 we discussed what to do if the add-on attribute may not reflect the exact behaviour of the model. Figure 14.3 compares Break Down plots with interactions for the four models under consideration.

Each of these models obviously has a different estimate for the chances of survival for `johny_d`. The highest estimate has the logistic regression model $0.764$ while the lowest estimate has the random forest model $0.441$. For the SVM model, the most important variable is `gender` and for the other models, `age` and `class`. The Random Forest model included interactions of `fare:class` and the SVM model included interactions of `fare:age`.

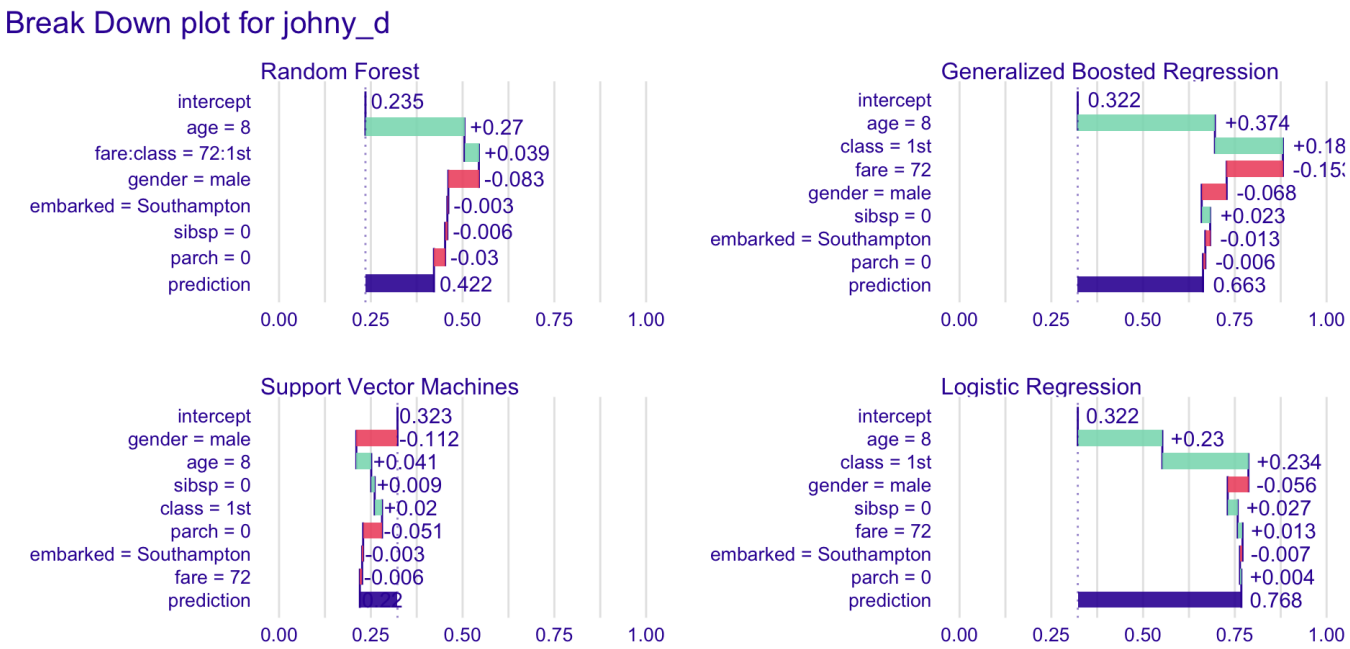



Figure 14.3: Break Down plots for four different models for the Titanic data.

Figure 14.4 shows Ceteris Paribus profiles for the four models considered for the `age` and `fare` variables. The logistic regression and GBM models behave in a similar way. Random forest and SVM models are much less sensitive to the `age` variable.

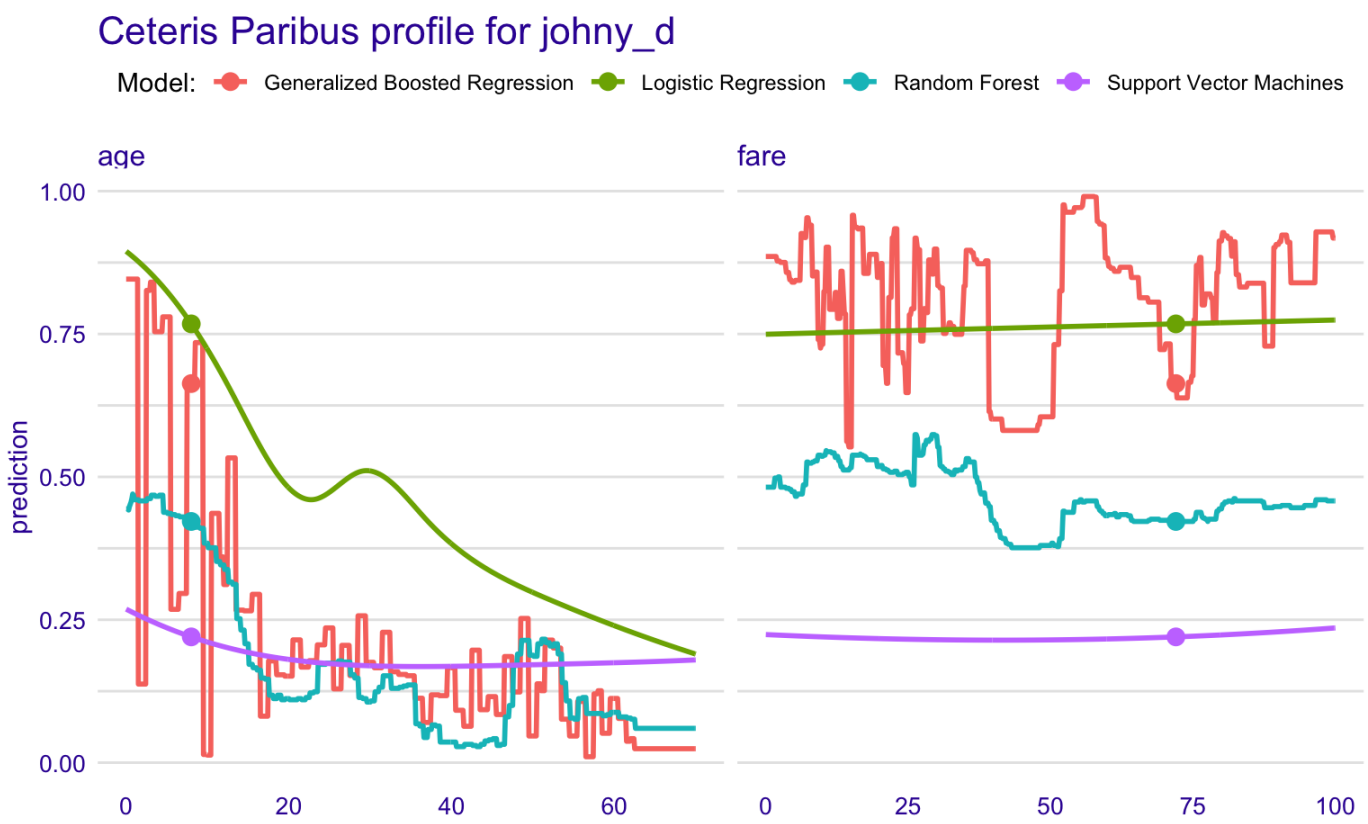
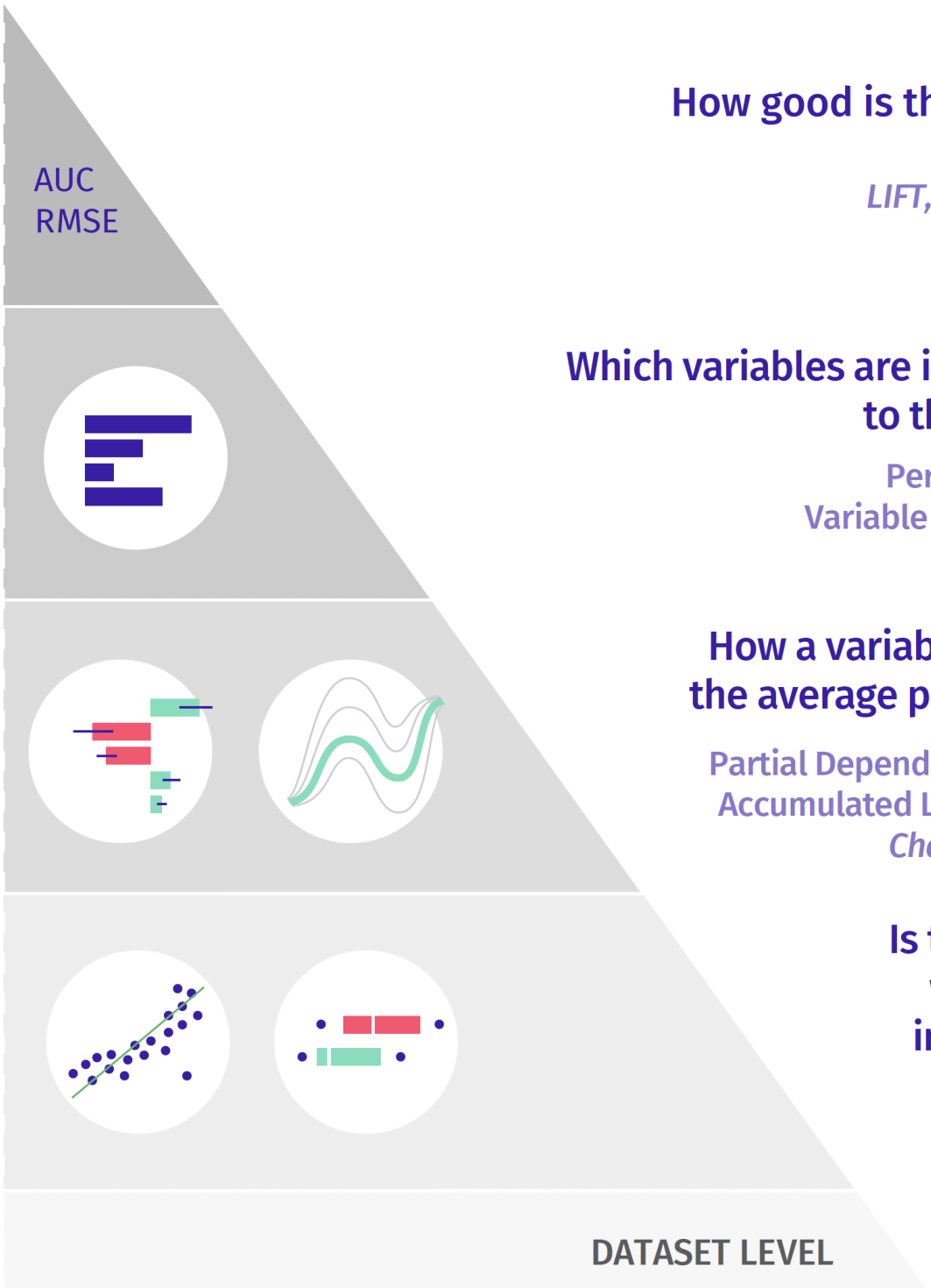


Figure 14.4: Ceteris Paribus profiles for four different models for the Titanic data.

Each of the four models under consideration has a different structure and each of them is for some reason a complex model. Random forest and boosting models are complex due to the large number of trees used for prediction. The SVM model is complex due to the non-linear function of the kernel and the logistic regression model due to spline transformations.

The compilation of the operating profile of the models side-by-side allows for a better understanding of the similarities and differences in the signals that these models have learned.

# Dataset Level




## How good is the model?

*ROC curve*
*LIFT, Gain charts*
*Chapter 16*

## Which variables are important to the model?

Permutational
Variable Importance
*Chapter 17*

## How a variable affects the average prediction?

Partial Dependence Profile
Accumulated Local Effects
*Chapters 18, 19*

## Is the model well fitted in general?

*Chapter 21*

# 15 Model-level exploration

In Part I, we focused on instance-level explainers, which help to understand how a model yields a prediction for a single observation (instance).

In Part II, we concentrate on model-level explainers, which help to understand how model's predictions perform overall, for a set of observations. Assuming that the observations form a representative sample from a general population, model-level explainers can provide an information about the quality of predictions for the population.

The following examples illustrate situations in which model-level explainers may be useful:

- We may want to learn which variables are ''important" in the model. For instance, we may be interested in predicting the risk of heart attack by using explanatory variables that are obtained based on results of some medical examinations. If some of the variables do not influence model's predictions, we could simplify the model by removing the variables.
- We may want to understand how a selected variable influences model's predictions. For instance, we may be interested in predicting prices of apartments. Apartment's location is an important factor, but we may want to know which locations lead to higher prices?
- We may want to discover whether there are any observations, for which the model yields wrong predictions. For instance, for a model predicting the probability of survival after a risky treatment, we might know whether there are patients for whom the model predictions are extremely wrong. Identifying such a group of patients might point to, for instance, an incorrect form of a explanatory variable or even a missed variable.

Model-level explainers focus on four main aspects of a model:

- Variable's importance: which explanatory variables are ''important", and which are not?
- Variable's effect: how does a variable influence average model's predictions?
- Model's performance: how ''good" is the model? Is one model ''better" than another?
- Model's fit: which observations are misfitted by the model, where residual are the largest?

In all cases, measures capturing a particular aspect of the model have to be defined. We will discuss them in subsequent chapters. In particular, in Chapter 16, we discuss measures that are useful for the evaluation of the overall predictive model performance. In Chapter 17, we focus on methods that allow evaluation of a variable's effect on model's predictions. Chapter 18 and Chapter 19 focus on exploration of the effect of selected variables on model response.

Chapter 20 presents an overview of the classical residual-diagnostics tools. Finally, in Chapter 21, we present an example of an analysis that illustrates the use of the model-level explainers introduced in the previous chapters.

# 16 Model Performance Measures

## 16.1 Introduction

In this chapter, we present measures that are useful for the evaluation of the overall performance of a predictive model. They may be applied for several purposes:

- model evaluation: we may want to know how good is the model, i.e., how reliable are the model predictions (how frequent and how large errors we may expect);
- model comparison: we may want to compare two or more models in order to choose between them;
- out-of-sample and out-of-time comparisons: we may want to check model's performance when applied to new data to evaluate if the performance has not worsened.

Depending of the nature of the dependent variable (continuous, binary, categorical, count, etc.), different model performance measures may be used. Moreover, the list of useful measures is growing as new applications emerge. In this chapter, we focus on a selected set of measures that are used in model-level exploration techniques that are introduced in subsequent chapters.

## 16.2 Intuition

Most model performance measures are based on comparison of the model predictions with the (known) values of the dependent variable in a dataset. For an ideal model, the predictions and the dependent-variable values should be equal. In practice, it is never the case, and we want to quantify the disagreement.

In applications, we can weigh differently the situation when the prediction is, for instance, larger than the true value, as compared to the case when it is smaller. Depending on the decision how to weigh different types of disagreement, we may need different performance measures.

When assessing model's overall performance, it is important to take into account the risk of overestimation of the quality of the performance when considering the data that were used for developing of the model. To mitigate the risk, various assessment strategies, such as cross-

validation, have been proposed (see (**???**)). In what follows, we consider the simple train-test-split strategy, i.e., we assume that the available data are split into a training set and a testing set. Model is created on the training set, and the testing set is used to assess the model's performance.

In the best possible scenario we can specify a single model performance measure before the model is created and then we optimize model for this measure. But in practice the more common scenario is to have few performance measures that are often selected after the model is created.

## 16.3 Method

Assume that we have got a testing dataset with $n$ observations on $p$ explanatory variables and on a dependent variable $Y$. Let $x_i$ denote the (column) vector of values of the explanatory variables for the $i$-th observation, and $y_i$ the corresponding value of the dependent variable. Denote by $\hat{y}_i = f(x_i)$ model's $f()$ prediction corresponding to $y_i$. Let $X = (x_1', \ldots, x_n')$ denote the matrix of explanatory variables for all $n$ observations, and $y = (y_1, \ldots, y_n)'$ denote the (column) vector of the values of the dependent variable.

### 16.3.1 Continuous dependent variable

The most popular model performance measure for models for a continuous dependent variable is the mean squared-error, defined as

$$MSE(f, X, y) = \frac{1}{n}\sum_{i}^{n}(f(x_i) - y_i)^2 = \frac{1}{n}\sum_{i}^{n} r_i^2, \tag{16.1}$$

where $r_i = f(x_i) - y_i$ is the residual for the $i$-th observation. Thus, MSE can be seen as a sum of squared residuals. MSE is a convex differentiable function, which is important from an optimization point of view. As the measure weighs all differences equally, large residuals have got a high impact on MSE. Thus, the measure is sensitive to outliers. For a ''perfect" predictive model, which predicts all $y_i$ exactly, $MSE = 0$.

Note that MSE is constructed on a different scale than the dependent variable. Thus, a more interpretable variant of this measure is the root-mean-squared-error (RMSE), defined as

$$RMSE(f, X, y) = \sqrt{MSE(f, X, y)}. \tag{16.2}$$

A popular variant of RMSE is its normalized version, $R^2$, defined as

$$R^2(f, X, y) = 1 - \frac{MSE(f, X, y)}{MSE(f_0, X, y)}. \tag{16.3}$$

In (16.3), $f_0()$ denotes a ‘’baseline” model. For instance, in the case of the classical linear regression, $f_0()$ is the model that includes only the intercept, which implies the use of the average value of $Y$ as a prediction for all observations. $R^2$ is normalized in the sense that the ‘’perfect” predictive model leads to $R^2 = 1$, while $R^2 = 0$ means that we are not doing better than the baseline model. In the context of the classical linear regression, $R^2$ is the familiar coefficient of determination and can be interpreted as the fraction of the total variance of $Y$ explained by model $f()$.

Given sensitivity of MSE to outliers, sometimes the median absolute-deviation (MAD) is considered as a model performance measure:

$$MAD(f, X, y) = median(|r_1|, \ldots, |r_n|). \tag{16.4}$$

MAD is more robust to outliers than MSE. A disadvantage of MAD are its less favorable mathematical properties.

### 16.3.2 Binary dependent variable

To introduce model performance measures, we, somewhat arbitrarily, label the two possible values of the dependent variable as ‘’success” and ‘’failure‘‘. (Of course, in a particular application, the meaning of the’‘success” outcome does not have to be positive nor optimistic; in diagnostic tests ‘’success” often means detection of a diseases.) We also assume that model prediction $f(x_i)$ takes the form of the predicted probability of success.

If, additionally, we assign the value of 1 to success and 0 to failure, it is possible to use MSE, RMSE, and MAE, as defined in (16.1), (16.2), (16.4), respectively, as a model performance measure. In practice, however, those summary measures are not often used. One of the main reasons is that they penalize too mildly for wrong predictions. In fact, the maximum penalty for an individual prediction is equal to 1 (if, for instance, the model yields zero probability for an actual success).

To address this issue, the log-likelihood function based on the Bernoulli distribution can be used:

$$l(f, X, y) = -\sum_{i=1}^{n} [y_i \ln\{f(x_i)\} + (1 - y_i) \ln\{1 - f(x_i)\}]. \tag{16.5}$$

Note that, in the machine-learning world, often $l(f, X, y)/n$ is considered (sometimes also with $\ln$ replaced by $\log_2$) and termed ‘’logloss” or ‘’cross-entropy’‘. The log-likelihood heavily’‘penalizes” the cases when the model-predicted probability of success $f(x_i)$ is high for an actual failure ($y_i = 0$) and low for an actual success ($y_i = 1$).

In many situations, however, a consequence of a prediction error depends on the form of the error. For this reason, performance measures based on the (estimated values of) probability of correct/wrong prediction are more often used. To introduce some of those measures, we assume that, for each observation from the testing dataset, the predicted probability of success $f(x_i)$ is compared to a fixed cut-off threshold, $C$ say. If the probability is larger than $C$, then we assume that the model predicts success; otherwise, we assume that it predicts failure. As a result of such a procedure, the comparison of the observed and predicted values of the dependent variable for the $n$ observations in the testing dataset can be summarized in the following table:

| | **True value:** `success` | **True value:** `failure` | **Total** |
|---|---|---|---|
| $f(x) > C$, predicted: `success` | True Positive: $TP_C$ | False Positive (type I error): $FP_C$ | $P_C$ |
| $f(x) \leq C$, predicted: `failure` | False Negative (type II error): $FN_C$ | True Negative: $TN_C$ | $N_C$ |
| Total | S | F | $n$ |

In machine-learning world, the table is often referred to as the ‘’confusion table” or ‘’confusion matrix’‘. In statistics, it is often called the’‘decision table”. The counts $TP_C$ and $TN_C$ on the diagonal of the table correspond to the cases when the predicted and observed value of the dependent variable $Y$ coincide. $FP_C$ is the number of cases in which failure is predicted as success. These are false-positive, or type I error, cases. On the other hand, $FN_C$ is the count of false-negative, or type II error, cases, in which success is predicted as failure. Marginally, there are $P_C$ predicted successes and $N_C$ predicted failures, with $P_C + N_C = n$. In the testing dataset, there are $S$ observed successes and $F$ observed failures, with $S + N = n$.

The simplest measure of model performance is **Accuracy**, defined as

$$ACC_C = \frac{TP_C + TN_C}{n}.$$

It is the fraction of correct predictions in the entire testing dataset. Accuracy is of interest if true positives and true negatives are more important than their false counterparts. However, accuracy may not be very informative when one of the binary categories is much more prevalent. For example, if the testing data contain 90% of successes, a model that would always predict a success would reach accuracy of 0.9, although one could argue that this is not a very useful model.

There may be situations when false positives and/or false negatives may be of more concern. In that case, one might want to keep their number low. Hence, other measures, focused on the false results, might be of interest.

In the machine-learning world, two other measures are often considered: **Precision** and **Recall**. Precision is defined as

$$Precision_C = \frac{TP_C}{TP_C + FP_C} = \frac{TP_C}{P_C}.$$

Precision is also referred to as the positive predictive value. It is the fraction of correct predictions among the predicted successes. Precision is high if the number of false positives is low. Thus, it is a useful measure when the penalty for committing the type I error (false positive) is high. For instance, consider the use of a genetic test in cancer diagnostics, with a positive result of the test taken as an indication of an increased risk of developing a cancer. A false positive result of a genetic test might mean that a person would have to unnecessarily cope with emotions and, possibly, medical procedures, related to the fact of being evaluated as having a high risk of developing a cancer. We might want to avoid this situation more than the false negative case. The latter would mean that the genetic test gives a negative result for a person that, actually, might be at an increased risk of developing a cancer. However, an increased risk does not mean that the person will develop cancer. And even so, we could hope that we could detect it in due time.

Recall is defined as

$$Recall_C = \frac{TP_C}{TP_C + FN_C} = \frac{TP_C}{S_C}.$$

Recall is also referred to as sensitivity or true positive rate. It is the fraction of correct predictions among the true successes. Recall is high if the number of false negatives is low. Thus, it is a useful measure when the penalty for committing the type II error (false negative) is high. For instance, consider the use of an algorithm that predicts whether a bank transaction is fraudulent. A false negative result means that the algorithm accepts a fraudulent transaction as a legitimate one. Such a decision may have immediate and unpleasant consequences for the bank, because it may imply a non-recoverable loss of money. On the

other hand, a false positive result means that a legitimate transaction is considered as fraudulent one and is blocked. However, upon further checking, the legitimate nature of the transaction can be confirmed with, perhaps, annoyed client as the only consequence for the bank.

The harmonic mean of these two measures defines the **F1 score**:

$$F1\ score_C = \frac{2}{\frac{1}{Precision_C} + \frac{1}{Recall_C}} = 2 \cdot \frac{Precision_C \cdot Recall_C}{Precision_C + Recall_C}.$$

F1 score tends to give a low value if either precision or recall is low, and a high value if both precision and recall are high. For instance, if precision is 0, F1 score will also be 0 irrespectively of the value of recall. Thus, it is a useful measure if we have got to seek a balance between precision and recall.

In statistics, and especially in applications in medicine, the popular measures are **Sensitivity** and **Specificity**. Sensitivity is simply another name for recall. Specificity is defined as

$$Specificity_C = \frac{TN_C}{TN_C + FP_C} = \frac{TN_C}{F_C}.$$

Specificity is also referred to as true negative rate. It is the fraction of correct predictions among the true failures. Specificity is high if the number of false positives is low. Thus, as precision, it is a useful measure when the penalty for committing the type I error (false positive) is high.

The reason why sensitivity and specificity may be more often used outside the machine-learning world is related to the fact that their values do not depend on the proportion $S/n$ (sometimes termed ‘’prevalence”) of true successes. This means that, once estimated in a sample obtained from a population, they may be applied to other populations, in which the prevalence may be different. This is not true for precision, because one can write

$$Precision_C = \frac{Sensitivity_C \cdot \frac{S}{n}}{Sensitivity_C \cdot \frac{S}{n} + Specificity_C \cdot \left(1 - \frac{S}{n}\right)}.$$

All the measures depend on the choice of cut-off $C$. To assess the form and the strength of dependence, a common approach is to construct the Receiver Operating Characteristic (ROC) curve. The curve plots the $Sensitivity_C$ in function of $1 - Specificity_C$ for all possible, ordered values of $C$. Figure 16.1 presents the ROC curve for the random-forest model for the Titanic dataset (see Section 5.1.3). Note that the curve indicates an inverse relationship between sensitivity and specificity: by increasing one measure, the other is decreased.

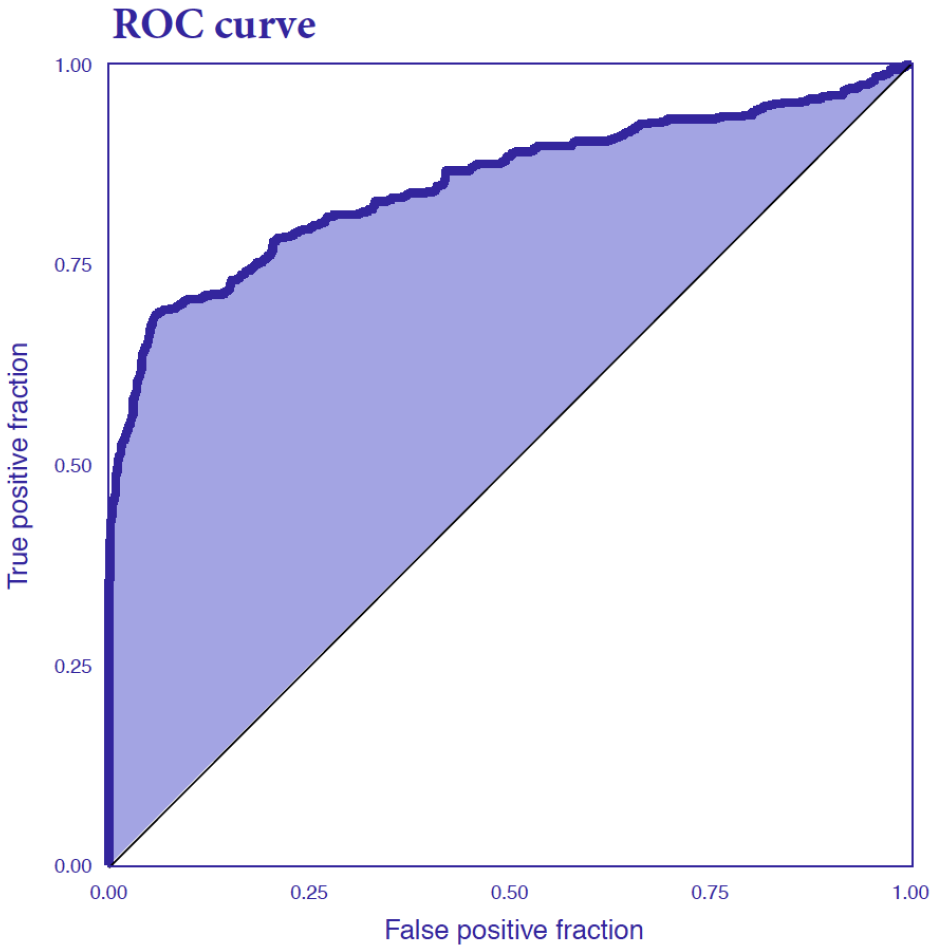



Figure 16.1: ROC curve for the random-forest model for the Titanic dataset. The Gini coefficient can be calculated as 2 x area between the ROC curve and the diagonal (this area is highlighted).

The ROC curve is very informative. For a model that predicts successes and failures at random, the corresponding ROC curve will be equal to the diagonal line. On the other hand, for a model that yields perfect predictions, the ROC curve reduces to a two intervals that connect points (0,0), (0,1), and (1,1).

Often, there is a need to summarize the ROC curve and, hence, model's performance. A popular measure that is used toward this aim is the area under the curve (AUC). For a model that predicts successes and failures at random, AUC is the area under the diagonal line, i.e., it is equal to 0.5. For a model that yields perfect predictions, AUC is equal to 1.

Another ROC-curve-based measure that is often used is the Gini coefficient $G$. It is closely related to AUC; in fact, it can be calculated a $G = 2 \times AUC - 1$. For a model that predicts successes and failures at random, $G = 0$; for a perfect-prediction model, $G = 1$.

The value of Gini's coefficient or, equivalently, of $AUC - 0.5$ allow a comparison of the model-based predictions with random guessing. A measure that explicitly compares a prediction model with a baseline (or null) model is the **Lift**. Commonly, random guessing is considered as the baseline model. In that case,

$$Lift_C = \frac{\frac{TP_C}{P_C}}{\frac{S}{n}} = \frac{nPrecision_C}{S}.$$

Note that $S/n$ can be seen as the estimated probability of a correct prediction of a success for random guessing. On the other hand, $TP_C/P_C$ is the estimated probability of a correct prediction a success given that the model predicts a success. Hence, informally speaking, the lift indicates how many more (or less) times the model does better in predicting success than random guessing. As other measures, the lift depends on the choice of cut-off $C$. The plot of the lift as a function of $P_C$ is called the lift chart.

There are many more measures aimed at measuring performance of a predictive model for a linearly dependent variable. An overview can be found in, e.g., (Berrar D. Performance Measures for Binary Classification. Encyclopedia of Bioinformatics and Computational Biology Volume 1, 2019, Pages 546-560). [TOMASZ: INCLUDE IN THE REFERENCE LIST.]

### 16.3.3 Categorical dependent variable

To introduce model performance measures for a categorical dependent variable, we assume that $y_i$ is now a vector of $K$ elements. Each element $y_{ik}$ ($k = 1, \ldots, K$) is a binary variable indicating whether the $k$-th category was observed for the $i$-th observation. We assume that for each observation only one category can be observed. Thus, all elements of $y_i$ are equal to 0 except of one that is equal to 1. Furthermore, We assume that model prediction $f(x_i)$ takes the form of a vector of the predicted probabilities for each of the $K$ categories. The predicted category is the one with the highest predicted probability.

The log-likelihood function (16.5) can be adapted to the categorical dependent variable case as follows:

$$l(f, X, y) = -\sum_{i=1}^{n}\sum_{k=1}^{K} y_{ik} \ln\{f(x_i)_k\}. \tag{16.6}$$

It is essentially the log-likelihood function based on a multinomial distribution.

It is also possible to extend the performance measures like accuracy, precision, etc., introduced in Section 16.3.2. Toward this end, first, a confusion table is created for each category $k$, treating the category as ''success" and all other categories as ''failure". Let us denote the counts in the table by $TP_k$, $FP_k$, $TN_k$, and $FN_k$. Based on the counts, we can compute the average accuracy across all classes as follows:

$$\overline{ACC_C} = \frac{1}{K}\sum_{k=1}^{K}\frac{TP_{C,k}+TN_{C,k}}{n}. \tag{16.7}$$

Similarly, one could compute the average precision, average sensitivity, etc. In machine-learning world, this approach is often termed ''macro-averaging". The averages computed in that way treat all classes equally.

An alternative approach is to sum the appropriate counts from the confusion tables for all classes, and then form a measure based on the so-computed cumulative counts. For instance, for precision, this would lead to

$$\overline{Precision_{C\mu}} = \frac{\sum_{k=1}^{K} TP_{C,k}}{\sum_{k=1}^{K}(TP_{C,k}+FP_{C,k})}. \tag{16.8}$$

In machine-learning world, this approach is often termed ''micro-averaging" (hence subscript $\mu$ for ''micro" in Equation $Precision_\mu$ in (16.8)). Note that, for accuracy, this computation still leads to Equation (16.7). The measures computed in that way favor classes with larger numbers of observations.

### 16.3.4 Count dependent variable

In case of counts, one could consider using any of the measures for a continuous dependent variable mentioned in Section 16.3.1. However, a particular feature of a count dependent variable is that, often, its variance depends on the mean value. Consequently, weighing all contributions to MSE equally, as in Equation (16.1), is not appropriate, because the same residual value $r_i$ indicates a larger discrepancy for a smaller count $y_i$ than for a larger one. Therefore, a popular measure is of performance of a predictive model for counts is Pearson's statistic:

$$\chi^2(f,X,y) = \sum_i^n \left\{\frac{f(x_i)-y_i}{\sqrt{f(x_i)}}\right\}^2 = \sum_i^n \left\{\frac{r_i}{\sqrt{f(x_i)}}\right\}^2. \tag{16.9}$$

From Equation (16.9) it is clear that, if the same residual value is obtained for two different observed counts, it is assigned a larger weight for the count for which the predicted value is smaller.

## 16.4 Example

### 16.4.1 Apartment prices

Let us consider the linear regression model `apartments_lm_v5` (see Section 5.2.2) and the random-forest model `apartments_rf_v5` (see Section 5.2.3) for the data on the apartment prices (see Section 5.2). Recall that, for these data, the dependent variable, the price, is continuous. Hence, we can use the performance measures presented in Section 16.3.1. In particular, we consider MSE and RMSE The values of the two measures for the two models are presented below.

```
## Model label:  Linear Regression v5
##          score name
## mse  80137.98   mse
## rmse 283.0865  rmse

## Model label:  Random Forest v5
##          score name
## mse  80061.77   mse
## rmse   282.952 rmse
```

Both MSE and MAE indicate that, overall, the random-forest model performs better than the linear regression model.

### 16.4.2 Titanic data

Let us consider the random-forest model `titanic_rf_v6` (see Section 5.1.3 and the logistic regression model `titanic_lmr_v6` (see Section 5.1.2) for the Titanic data (see Section 5.1). Recall that, for these data, the dependent variable is binary, with success defined as survival of the passenger.

First, we will take a look at the accuracy, F1 score, and AUC for the models.

```
## Model label:  Logistic Regression v6
##         score name
## auc 0.8196991  auc
## f1  0.6589018   f1
## acc 0.8046689  acc

## Model label:  Random Forest v6
##         score name
## auc 0.8566304  auc
## f1  0.7289880   f1
## acc 0.8494521  acc
```

Overall, the random-forest model is performing better, as indicated by the larger values of all the measures.

Figure 16.2 presents ROC curves for both models. The curve for the random-forest model lies above the one for the logistic regression model for the majority of the cut-offs $C$, except for the very high values.

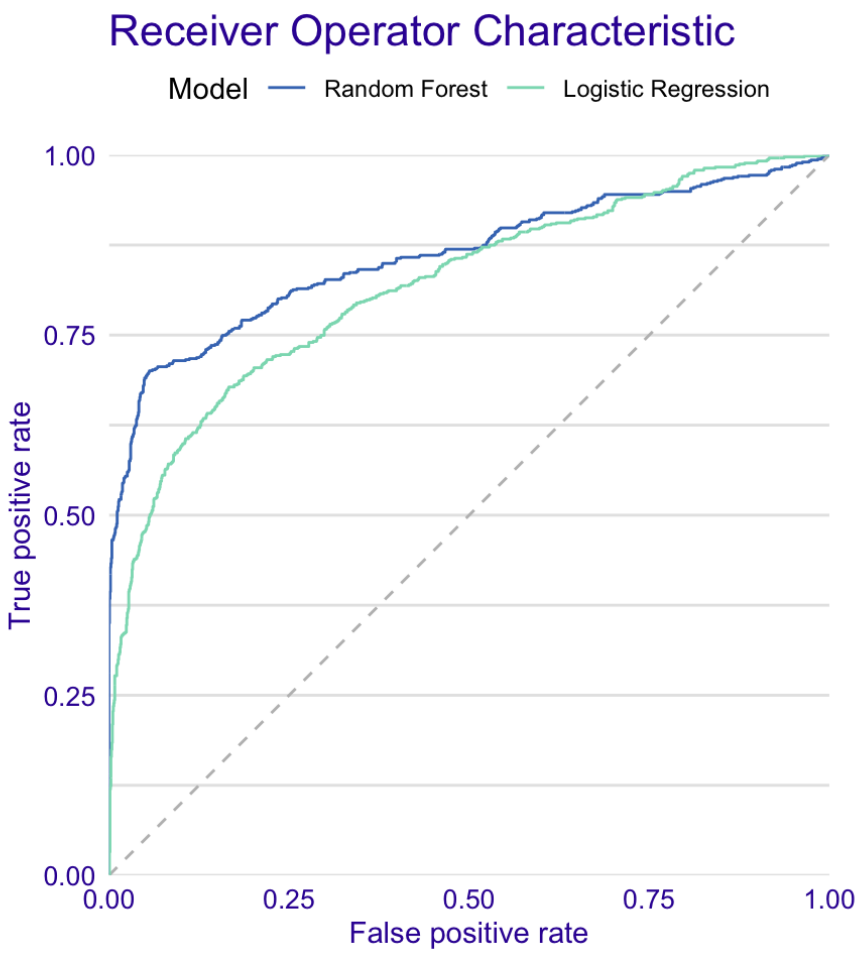


Figure 16.2: ROC curves for the random-forest model and the logistic regression model for the Titanic dataset.

Figure 16.3 presents lift curves for both models. Also in this case the curve for the random-forest suggests a better performance than for the logistic regression model, except for the very high values of cut-off $C$.

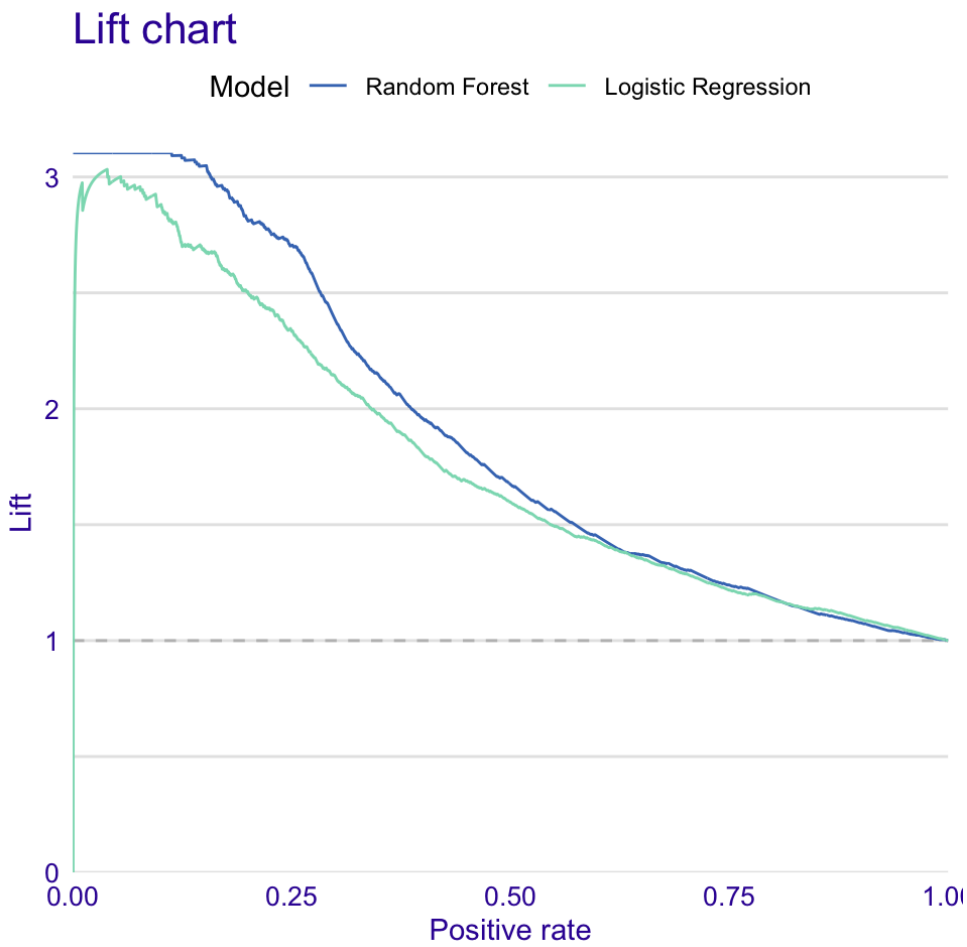



Figure 16.3: Lift curves for the Random forest model and the logistic regression model for the Titanic dataset.

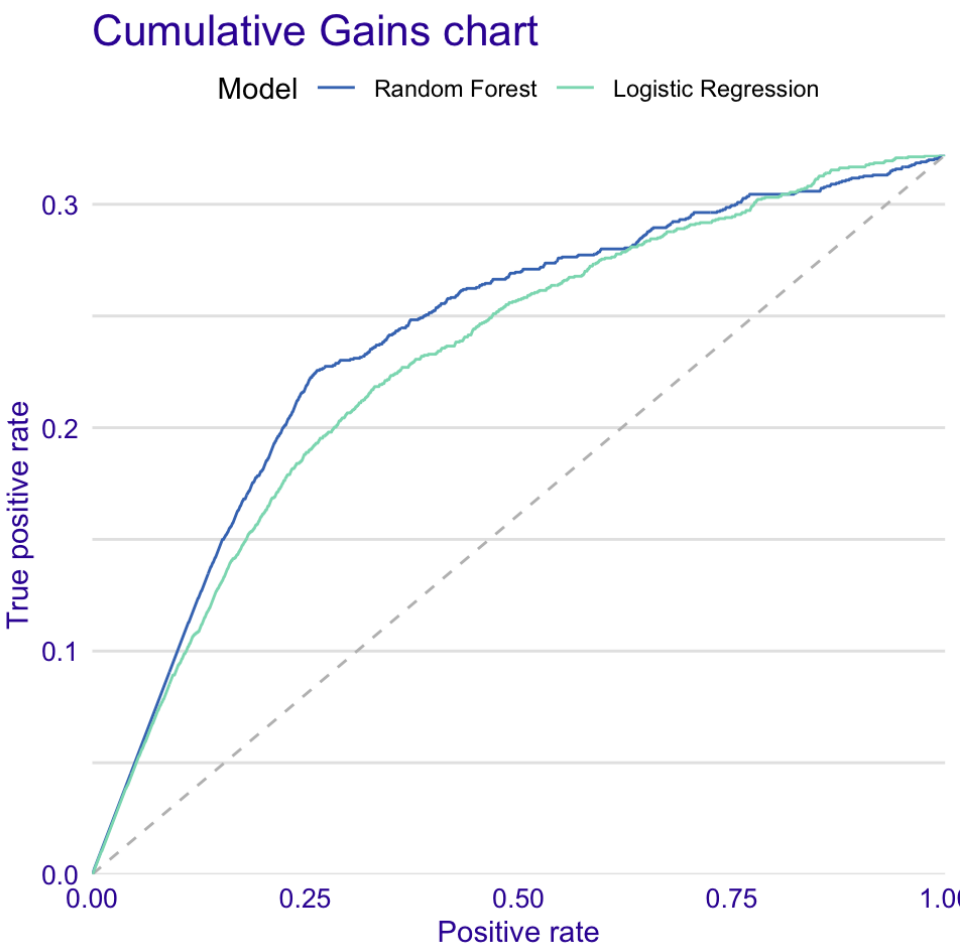



Figure 16.4: Cumulative gain chart for the Random forest model and the logistic regression model for the Titanic dataset.

## 16.5 Pros and cons

All model performance measures presented in this chapter face some limitations. For that reason, many measures are available, as the limitations of a particular measure were addressed by developing an alternative. For instance, RMSE is frequently used and reported for linear regression models. However, as it is sensitive to outliers, MAE was proposed. In case of predictive models for a binary dependent variable, the measures like accuracy, F1 score, sensitivity, and specificity, are often considered depending on the consequences of correct/incorrect predictions in a particular application. However, the value of those measures depends on the cut-off value used for creating the predictions. For this reason, ROC curve and AUC have been developed and have become very popular. They are not easily extended to the case of a categorical dependent variable, though.

Given the advantages and disadvantages of various measures, and the fact that each may reflect a different aspect of the predictive performance of a model, it is customary to report and compare several of them when evaluating a model's performance.

## 16.6 Code snippets for R

In this section, we present the key features of the `DALEX` R package which is a part of the DrWhy.AI universe. The package covers all methods presented in this chapter. Please note that more advanced measures of performance are available in the `auditor` R package (Gosiewska and Biecek 2018). Note that there are also other R packages that offer similar functionality. These include, for instance, packages `mlr` (Bischl et al. 2016), `caret` (Jed Wing et al. 2016), `tidymodels` (Max and Wickham 2018), and `ROCR` (Sing et al. 2005).

For illustration purposes, we use the random-forest model `titanic_rf_v6` (see Section 5.1.3 and the logistic regression model `titanic_lmr_v6` (see Section 5.1.2) and the random-forest model `titanic_rf_v6` (see Section 5.1.3) for the Titanic data (see Section 5.1). Consequently, the functions from the `DALEX` package are applied in the context of a binary classification problem. However, the same functions can be used for, e.g., linear regression problems.

To illustrate the use of the functions, we first load explainers for both models.

```
library("DALEX")
library("randomForest")

explainer_titanic_rf <- archivist:: aread("pbiecek/models/6ed54")
explain_titanic_lmr <- archivist:: aread("pbiecek/models/ff1cd")

DALEX::model_performance(explainer_titanic_rf)
DALEX::model_performance(explain_titanic_lmr)
```

Function `DALEX::model_performance()` calculates selected model performance measures. By default a set of selected performance measures are calculated. The argument `type` in the `explain()` function is used to determine if we deal with classification or regression model. The `data` argument serves for specification of the test dataset, for which the selected measures are to be computed. Note that, by default, the data are extracted from the explainer object. Finally, it is possible to use the `cutoff` argument to specify the cut-off value to obtained cut-off-dependent measures like F1 score or accuracy.

```
model_performance(explain_titanic_rf)
```

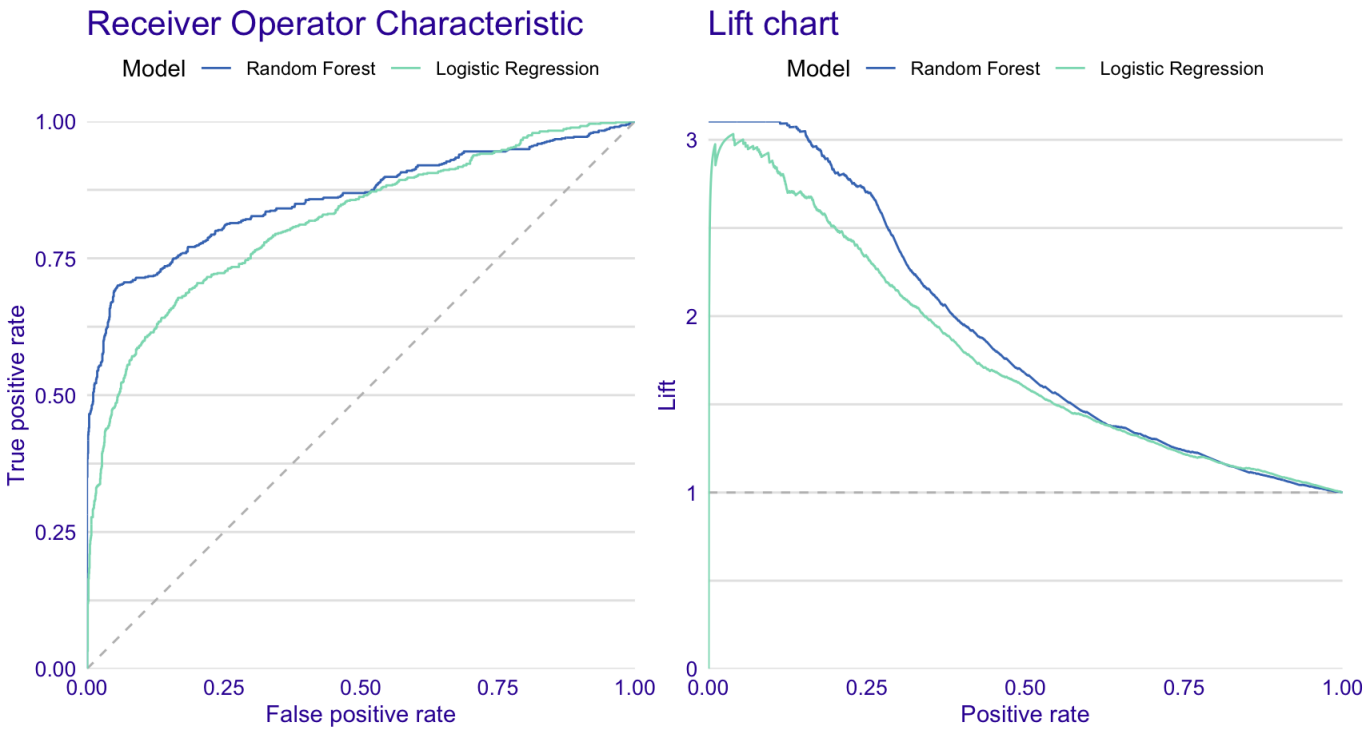
```
## Measures for:  classification
## recall    : 0.6385373
## precision: 0.8832685
## f1        : 0.7412245
## accuracy : 0.8563661
## auc       : 0.8595467
##
## Residuals:
##      0%     10%     20%     30%     40%     50%     60%     70%     80%
## -0.8920 -0.1140 -0.0240 -0.0080 -0.0040  0.0000  0.0000  0.0100  0.1400
##     90%    100%
##  0.5892  1.0000
```

```
model_performance(explain_titanic_lmr)
```

```
## Measures for:  regression
## mse       : 0.1459437
## rmse      : 0.3820258
## r2        : 0.3316733
## mad       : 0.2119129
##
## Residuals:
##          0%         10%         20%         30%         40%         50%
## -0.98457244 -0.31904861 -0.23408037 -0.20311483 -0.15200813 -0.10318060
##         60%         70%         80%         90%        100%
## -0.06933478  0.05858024  0.29306442  0.73666519  0.97151255
```

ROC or lift curves can be constructed with the `plot()` function. The argument `geom` specifies what type of visual model performance summary shall be plotted. Use `geom = "lift"` for lift charts, `geom = "roc"` for ROC charts, `geom = "histogram"` for historam of residuals. In all cases the plot functions return `ggplot2` objects and can take one or more explainer objects as arguments. In the latter case, the profiles for each explainer are superimposed on one plot.

```
eva_rf <- DALEX::model_performance(explain_titanic_rf)
eva_lr <- DALEX::model_performance(explain_titanic_lmr)

p1 <- plot(eva_rf, eva_lr, geom = "roc")
p2 <- plot(eva_rf, eva_lr, geom = "lift")

library("patchwork")
p1 + p2
```




The resulting plots are shown in Figures 16.2 and 16.3. Both plots can be supplemented with boxplots for residuals. Toward this end, the residuals have got to be computed and added to the explainer object with the help of the `model_performance()` function. Subsequently, the `plot()` can be applied to the resulting object.

```
plot(eva_rf, eva_lr, geom = "boxplot")
```

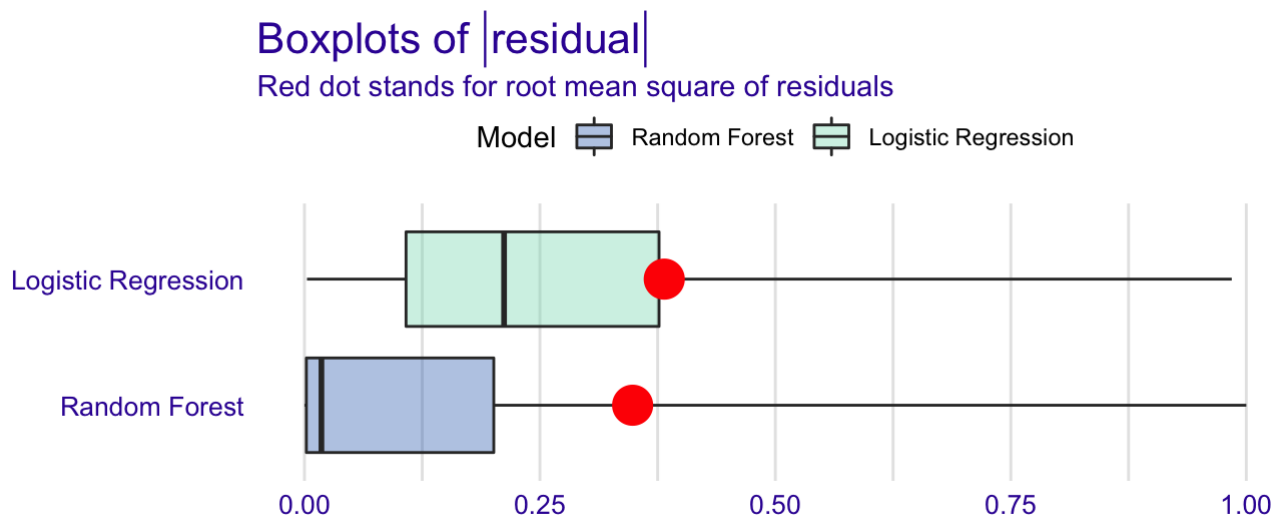



Figure 16.5: Boxplots for residuals for two models on Titanic dataset.

## References

Bischl, Bernd, Michel Lang, Lars Kotthoff, Julia Schiffner, Jakob Richter, Erich Studerus, Giuseppe Casalicchio, and Zachary M. Jones. 2016. “mlr: Machine Learning in R.” *Journal of Machine Learning Research* 17 (170): 1–5. http://jmlr.org/papers/v17/15-066.html.

Gosiewska, Alicja, and Przemyslaw Biecek. 2018. *Auditor: Model Audit - Verification, Validation, and Error Analysis*. https://CRAN.R-project.org/package=auditor.

Jed Wing, Max Kuhn. Contributions from, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, et al. 2016. *Caret: Classification and Regression Training*. https://CRAN.R-project.org/package=caret.

Max, Kuhn, and Hadley Wickham. 2018. *Tidymodels: Easily Install and Load the ’Tidymodels’ Packages*. https://CRAN.R-project.org/package=tidymodels.

Sing, T., O. Sander, N. Beerenwinkel, and T. Lengauer. 2005. “ROCR: visualizing classifier performance in R.” *Bioinformatics* 21 (20): 7881. http://rocr.bioinf.mpi-sb.mpg.de.

# 17 Variable's Importance

## 17.1 Introduction

In this chapter, we present methods that are useful for the evaluation of an explanatory variable importance. The methods may be applied for several purposes.

- Model simplification: variables that do not influence model's predictions may be excluded from the model.
- Model exploration: comparison of a variable's importance in different models may help in discovering interrelations between the variables.Also, ordering of variables in function of their importance is helpful in deciding in what order should we perform further model exploration.
- Domain-knowledge-based model validation: identification of the most important variables may be helpful in assessing the validity of the model based on the domain knowledge.
- Knowledge generation: identification of the most important variables may lead to discovery of new factors involved in a particular mechanism.

The methods for assessment of variable importance can be divided, in general, into two groups: model-specific and model-agnostic.

For models like linear models, random forest, and many others, there are methods of assessing of variable importance that exploit particular elements of the structure of the model. These are model-specific methods. For instance, for linear models, one can use the value of the normalized regression coefficient or its corresponding p-value as the variable-importance measure. For tree-based ensembles, such a measure may be based on the use of a particular variable in particular trees (see, e.g., `XgboostExplainer` (Foster 2017) for gradient boosting and `RandomForestExplainer` (Paluszynska and Biecek 2017) for random forest).

In this book we focus on model-agnostic methods. These methods do not assume anything about the model structure. Therefore, they can be applied to any predictive model or ensemble of models. Moreover, and perhaps even more importantly, they allow comparing variable importance between models with different structures.

## 17.2 Intuition

We focus on the method described in more detail in (Fisher, Rudin, and Dominici 2018). The main idea is to measure how much the model fit decreases if the effect of a selected explanatory variable or of a group of variables is removed. The effect is removed by means of perturbations like resampling from an empirical distribution of just permutation of the values of the variable.

The idea is in some sense borrowed from variable important measure proposed by @ref{randomForestBreiman} for random forest. If a variable is important, then after permutation of this variable we expect that the model performance will be lower. The larger drop in the performance, the more important is the variable.

Despite the simplicity of definition, the permutation variable importance is a very powerful model agnostic tool for model exploration. Values of permutation variable importance may be compared between different structures of models. This property is discussed in detail in the section *Pros and Cons*.

## 17.3 Method

Consider a set of $n$ observations for a set of $p$ explanatory variables. Denote by $\tilde{y} = (f(x_1), \ldots, f(x_n))$ the vector of predictions for model $f()$ for all the observations. Let $y$ denote the vector of observed values of the dependent variable $Y$.

Let $\mathcal{L}(\tilde{y}, y)$ be a loss function that quantifies goodness of fit of model $f()$ based on $\tilde{y}$ and $y$. For instance, $\mathcal{L}$ may be the value of likelihood. Consider the following algorithm:

1. Compute $L = \mathcal{L}(\tilde{y}, y)$, i.e., the value of the loss function for the original data.
2. For each explanatory variable $X^j$ included in the model, do steps 3-6.
3. Replace vector $x^j$ of observed values of $X^j$ by vector $x^{*j}$ of resampled or permuted values.
4. Calculate model predictions $\tilde{y}^{*j}$ for the modified data, $\tilde{y}^{*j} = f(x^{*j})$.
5. Calculate the value of the model performance for the modified data:

$$L^{*j} = \mathcal{L}(\tilde{y}^{*j}, y)$$

6. Quantify the importance of explanatory variable $x^j$ by calculating $vip_{Diff}(x^j) = L^{*j} - L$ or $vip_{Ratio}(x^j) = L^{*j}/L$, where $L$ is the value of the loss function for the original data.

Note that the use of resampling or permuting data in Step 3 involves randomness. Thus, the results of the procedure may depend on the actual configuration of resampled/permuted values. Hence, it is advisable to repeat the procedure several times. In this way, the uncertainty related to the calculated variable-importance values can be assessed.

The calculations in Step 6 normalize" the value of the variable importance measure with respect to $L$. However, given that $L$ is a constant, the normalization has no effect on the ranking of variables according to $vip_{Diff}(x^j)$ or $vip_{Ratio}(x^j)$. Thus, in practice, often the values of $L^{*j}$ are simply used to quantify variable's importance.

## 17.4 Example: Titanic data

In this section, we illustrate the use of the permutation-based variable-importance method by applying it to the random forest model for the Titanic data (see Section 5.1.3). Recall that the goal is to predict survival probability of passengers based on their sex, age, cityplace of embarkment, class in which they travelled, fare, and the number of persons they travelled with.

Figure 17.1 shows the values of loss function measured as $1 - AUC^{*j}$ after permuting, in turn, each of the variables included in the model. Additionally, the plot indicates the value of $L$ by the vertical dashed line at the left-hand-side of the plot. Length of the bar span between $L$ and $L^{*j} = 1 - AUC^{*j}$ and correspond to the variable importance.

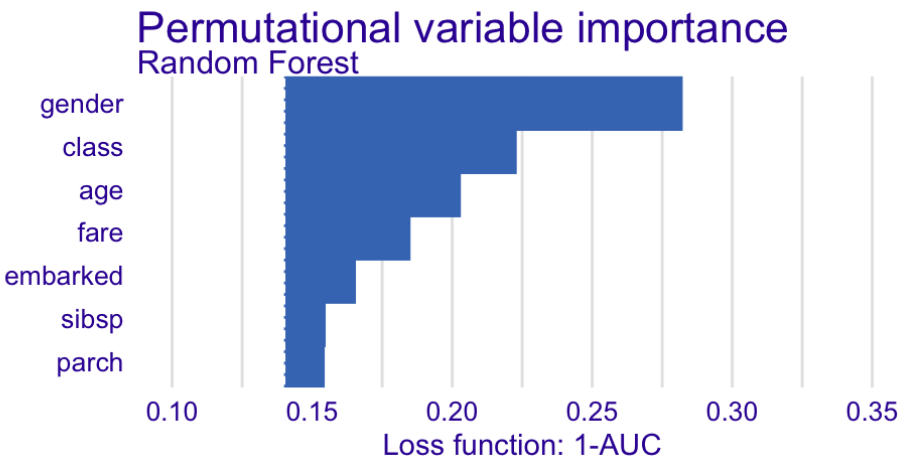



Figure 17.1: Each interval presents the difference between the loss function for the original data (vertical dashed line at the left) and for the data with permuted observation for a particular variable.

The plot in Figure 17.1 suggests that the most important variable in the model is gender. This agrees with the conclusions drawn in the exploratory analysis presented in Section 5.1.1. The next three important variables are class of the travel (first-class patients had a higher chance of survival), age (children had a higher chance of survival), and fare (owners of more expensive tickets had a higher chance of survival).

To take into account the uncertainty related to the use of permutations, we can consider computing the average values of $L^{*j}$ over a set of, say, 10 permutations. The plot in Figure 17.2 presents the average values. The only remarkable difference, as compared to Figure 17.1, is the change in the ordering of the `sibsp` and `parch` variables.

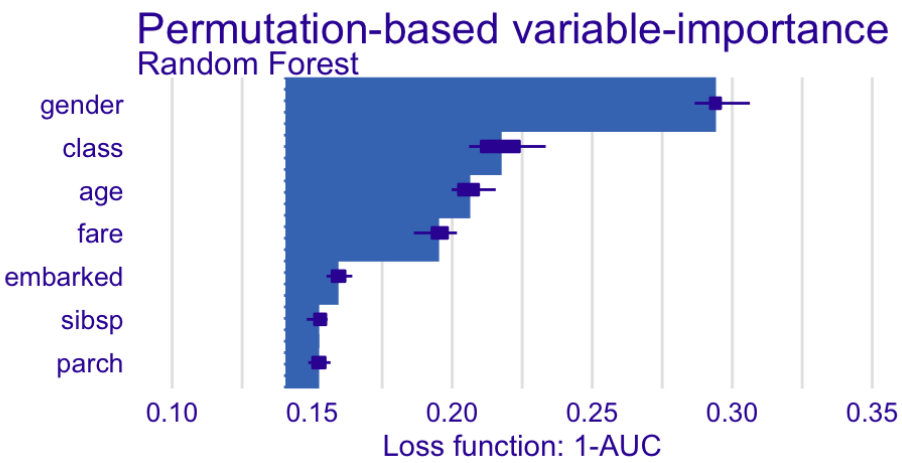



Figure 17.2: Average variable importance based on 10 permutations.

The plots similar to those presented in Figures 17.1 and 17.2 are useful for comparisons of variable importance for different models. Figure 17.3 presents the single-permutation results for the random forest, gradient boosting (see Section 5.1.4), and logistic regression (see Section 5.1.2) models. The best result, in terms of the smallest value of the goodness-of-fit function $L$, are obtained for the random forest model. Note, however, that this model includes more variables than the other two. For instance, variable `fare`, which is highly correlated with the travel class, is not important neither in the gradient boosting nor in the logistic regression model, but is important in the random forest model.

The plots in Figure 17.3 indicate that `gender` is the most important variable in all three models, followed by `class`.


Figure 17.3: Variable importance for the random forest, gradient boosting, and logistic regression models for the Titanic data.

## 17.5 Pros and cons

Permutation variable importance offer a model-agnostic approach to assessment of the influence of each variable on model performance. The approach offers several advantages. The plots are easy to understand. They are compact, all most important variables are presented in a single plot.

Permutation variable importance is expressed in a terms of model performance and can be compared between models. In different models the same variable may have different importance scores and comparison of such scores may lead to interesting insights. For

example if variables are correlated then models like random forest are expected to spread importance across every variable while in regularized regression models coefficients for one correlated variable may dominate over coefficients for other variables.

The same approach can be used to measure importance of a single explanatory variable or a group of variables. The latter is useful for aspects - groups of variables that are complementary or are related to a similar concept. For example in the Titanic example the `fare` and `class` variables are linked with wealth of a passenger. Instead of calculation of effects of each variable independently we may calculate effect of both variables by permutation of both.

The disadvantage of this measure comes from the randomness behind permutations. For different permutations we may get different results. Also different choices of model performance measure, like Precision, Accuracy, AUC, lead to different numeric values of variable importance. And last disadvantage is related with the data used for assessment of model performance. Different importance values may be obtained on training and testing data.

## 17.6 Code snippets for R

For illustration, We will use the random forest model for the apartment prices data (see Section 5.2.3).

Let's recover a regression model for prediction of apartment prices.

A popular loss function for regression model is the root mean square loss.

$$L(y, \tilde{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2}$$

It is implemented in the `DALEX` package in the function `loss_root_mean_square`. The initial loss function $L$ for this model is

```
loss_root_mean_square(
  predict(model_rf, apartmentsTest),
  apartmentsTest$m2.price
)
```

```
## [1] 792.8346
```

Let's calculate variable importance for root mean square loss with the `model_parts` function.

```
vip <- model_parts(explainer_rf,
            loss_function = loss_root_mean_square)
vip
```

```
##             variable mean_dropout_loss        label
## 1       _full_model_          796.0100 randomForest
## 2           no.rooms          828.4179 randomForest
## 3 construction.year          842.2287 randomForest
## 4           district          850.4096 randomForest
## 5              floor          858.8663 randomForest
## 6            surface          875.2063 randomForest
## 7         _baseline_         1118.4724 randomForest
```

On a diagnostic plot is useful to present variable importance with boxplots that show results for different permutations.

```
library("ggplot2")
plot(vip) +
  ggtitle("Permutation variable importance", "")
```

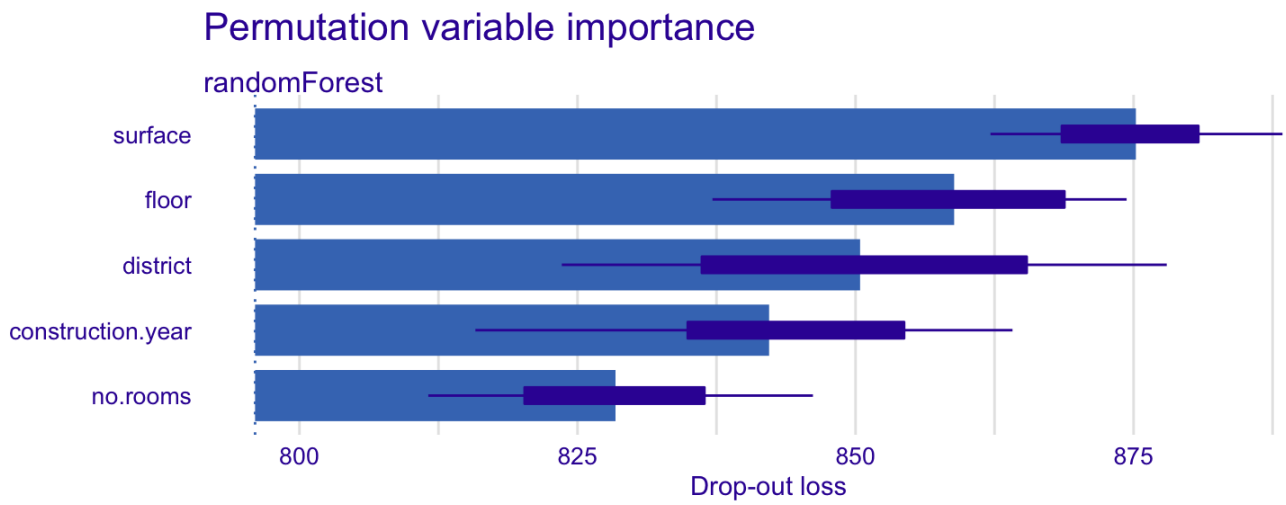


Figure 17.4: Permutation variable importance calculated as root mean square loss for random forest model for apartments data.

### 17.6.1 Models comparison

Variable importance plots are very useful tool for model comparison. In the section 5.2 we have trained three models on `apartments` dataset. These were models with different structures to make the comparison more interesting. Random Forest model (Breiman et al. 2018) (elastic but biased), Support Vector Machines model (Meyer et al. 2017) (large variance on boundaries) and Linear Model (stable but not very elastic).

Let's calculate permutation variable importance with root mean square error for these three models.

```
vip_lm <- variable_importance(explainer_lm,
            loss_function = loss_root_mean_square)
vip_lm
```

```
##            variable mean_dropout_loss label
## 1      _full_model_          281.8345    lm
## 2 construction.year          281.7864    lm
## 3          no.rooms          293.7945    lm
## 4             floor          486.0535    lm
## 5           surface          614.4047    lm
## 6          district         1018.8827    lm
## 7        _baseline_         1262.6592    lm
```

```
vip_rf <- variable_importance(explainer_rf,
            loss_function = loss_root_mean_square)
vip_rf
```

```
##            variable mean_dropout_loss        label
## 1      _full_model_          802.9422 randomForest
## 2          no.rooms          834.9660 randomForest
## 3 construction.year          851.9975 randomForest
## 4          district          852.5380 randomForest
## 5             floor          874.3987 randomForest
## 6           surface          880.9620 randomForest
## 7        _baseline_         1110.6190 randomForest
```

```
vip_svm <- variable_importance(explainer_svm,
            loss_function = loss_root_mean_square)
vip_svm
```

```
##            variable mean_dropout_loss label
## 1      _full_model_          984.9034   svm
## 2          district          950.4622   svm
## 3          no.rooms          980.3698   svm
## 4 construction.year         1041.9925   svm
## 5             floor         1072.9481   svm
## 6           surface         1096.7851   svm
## 7        _baseline_         1237.6861   svm
```

Now we can plot variable importance for all three models on a single plot. Intervals start in a different values, thus we can read that loss for SVM model is the lowest.

When we compare other variables it looks like in all models the `district` is the most important feature followed by `surface` and `floor`.

```
library("ggplot2")
plot(vip_rf, vip_svm, vip_lm) +
  ggtitle("Permutation variable importance", "")
```

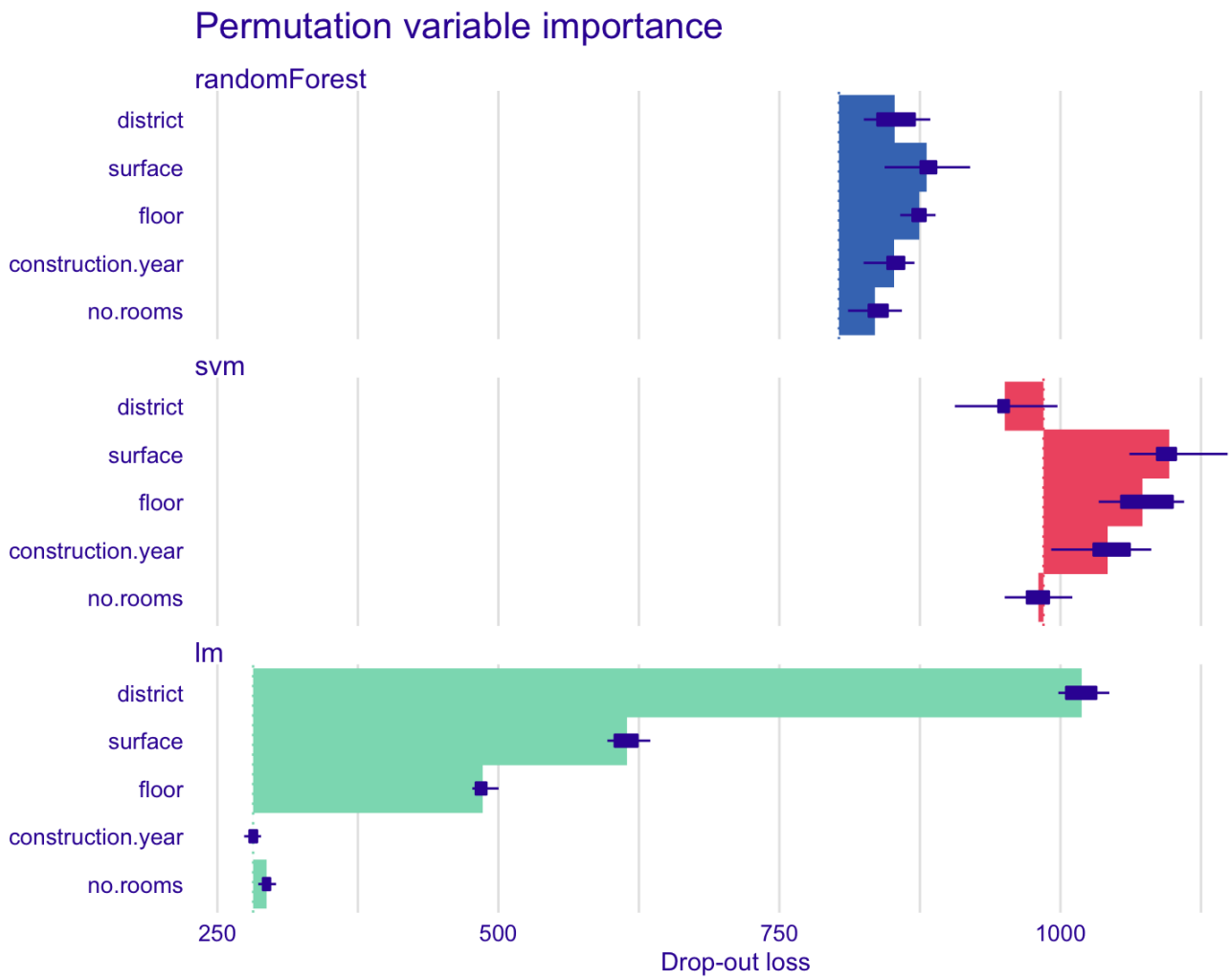



Figure 17.5: Permutation variable importance on apartments data for Random forest, Support vector model and Linear model.

There is interesting difference between linear model and others in the way how important is the `construction.year`. For linear model this variable is not importance, while for remaining two models there is some importance.

In the next chapter we will see how this is possible.

## References

Breiman, Leo, Adele Cutler, Andy Liaw, and Matthew Wiener. 2018. *RandomForest: Breiman and Cutler's Random Forests for Classification and Regression*. https://CRAN.R-project.org/package=randomForest.

Fisher, Aaron, Cynthia Rudin, and Francesca Dominici. 2018. “Model Class Reliance: Variable Importance Measures for Any Machine Learning Model Class, from the ’Rashomon’ Perspective.” *Journal of Computational and Graphical Statistics*. http://arxiv.org/abs/1801.01489.

Foster, David. 2017. *XgboostExplainer: An R Package That Makes Xgboost Models Fully Interpretable*. https://github.com/AppliedDataSciencePartners/xgboostExplainer/.

Meyer, David, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, and Friedrich Leisch. 2017. *E1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), Tu Wien*. https://CRAN.R-project.org/package=e1071.

Paluszynska, Aleksandra, and Przemyslaw Biecek. 2017. *RandomForestExplainer: A Set of Tools to Understand What Is Happening Inside a Random Forest*. https://github.com/MI2DataLab/randomForestExplainer.

# 18 Partial dependence profiles

## 18.1 Introduction

In this chapter we focus on partial dependence (PD) plots, sometimes also called PD profiles. They were introduced by Friedman in a paper devoted to Gradient Boosting Machines (GBM) (Friedman 2000). For many years PD profiles went unnoticed in the shadow of GBM. However, in recent years, the profiles have become very popular and are available in many data-science-oriented software packages like `DALEX`, `iml` (Molnar, Bischl, and Casalicchio 2018), `pdp` (Greenwell 2017).

The general idea underlying the construction of PD profiles is to show how the expected value of model prediction behaves as a function of a selected explanatory variable. For a single model, one can construct an overall PD profile by using all observations from a dataset, or several profiles for sub-groups of the observations. Comparison of sub-group-specific PD profiles may provide important insight into, for instance, stability of the model predictions. PD profiles are also useful for comparisons of different models:

- *Agreement between profiles for different models is reassuring.* Some models are more flexible than others. If PD profiles for models from the two classes are similar, we can treat it as a evidence that the more flexible model is not over-fitting.
- *Disagreement between profiles may suggest a way to improve a model.* If a PD profile of a simpler, more interpretable model disagrees with a profile of a flexible model, this may suggest a variable transformation that can be used to improve the interpretable model. For example, if a random-forest model indicates a non-linear relationship between the dependent variable and an explanatory variable, then a suitable transformation of the explanatory variable may improve the fit or performance of a linear regression model.
- *Evaluation of model performance at boundaries.* Models are known to have a different behavior at the boundaries of dependent variables, i.e., for the largest or the lowest values. For instance, random-forest models are known to shrink predictions towards the average, whereas support-vector machines are known to have larger variance at edges. Comparison of PD profiles may help to understand the differences in models' behavior at boundaries.

## 18.2 Intuition

The general idea underlying the construction of PD profiles is to show how the expected value of model prediction behaves as a function of a selected explanatory variable. Toward this aim, the average of a set of individual Ceteris-paribus (CP) profiles is used. Recall that a CP profile (see Chapter 11) shows the dependence of an instance-level prediction for an explanatory variable. A PD profile is estimated by the average of the CP profiles for all instances (observations) from a dataset.

Note that, for additive models, CP profiles are parallel. In particular, they have got the same shape. Consequently, the average retains the shape, while offering a more precise estimate. However, for models that, for instance, include interactions, CP profiles may not be parallel. In that case, the average may not necessarily correspond to the shape of any particular profile. Nevertheless, it can still offer a summary of how (in general) the model predictions depend on changes in a given explanatory variable.

The left-hand-side panel of Figure 18.1 presents CP profiles for the explanatory variable age for the random-forest model `titanic_rf_v6` (see Section **??**) for 25 randomly selected instances (observations) from the Titanic dataset (see Section **??**). Note that the profiles are not parallel, indicating non-additive effects of explanatory variables. The right-hand-side panel show the average of the CP profiles, which offers an estimate of the PD profile. Clearly, the shape of the PD profile does not capture, for instance, the shape of the three CP profiles shown at the top of the panel. Nevertheless, it does seem to reflect the fact that the majority of CP profiles suggest a substantial drop in the predicted probability of survival for the ages between 2 and 18.

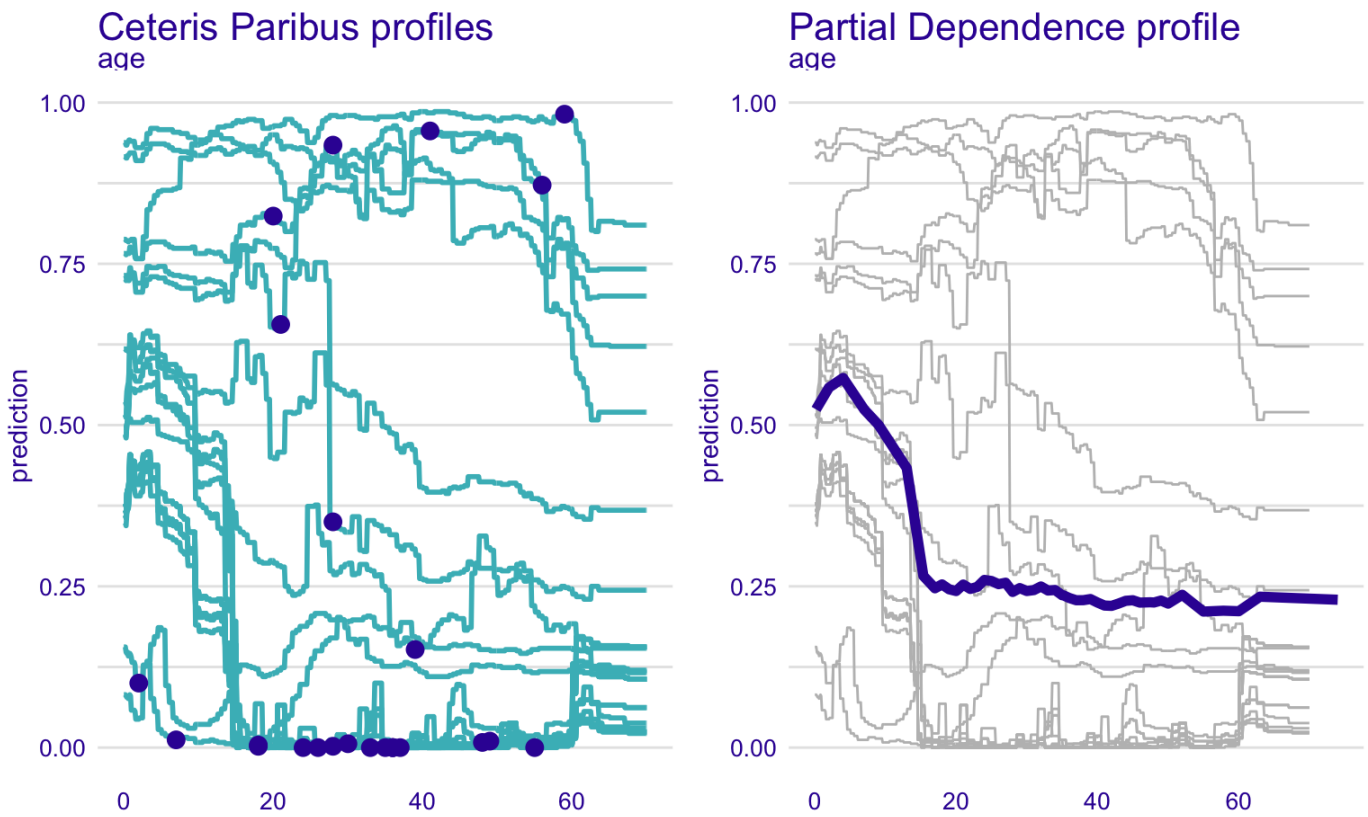



Figure 18.1: Ceteris-paribus and partial-dependence profiles for the random-forest model for 25 randomly selected observations from the Titanic dataset. Left: CP profiles for age; blue dots indicate the age and corresponding prediction for the selected observations. Right: CP profiles (grey lines) and the corresponding partial-dependence profile (blue line)

## 18.3 Method

### 18.3.1 Partial dependence profiles

The value of a PD profile for model $f()$ and explanatory variable $X^j$ at $z$ is defined as follows:

$$g_{PD}^{f,j}(z) = E_{X^{-j}}[f(X^{j|=z})]. \tag{18.1}$$

Thus, it is the expected value of the model predictions when $X^j$ is fixed at $z$ over the (marginal) distribution of $X^{-j}$, i.e., over the joint distribution of all explanatory variables other than $X^j$. Or, in other words, it is the expected value of the CP profile ntoduced in Equation (11.1) for $X^j$ over the (marginal) distribution of $X^{-j}$.

Usually, we do not know the true distribution of $X^{-j}$. We can estimate it, however, by the empirical distribution of $N$, say, observations available in a training dataset. This leads to the use of the average of CP profiles for $X^j$ as an estimator of the PD profile:

$$\hat{g}^{f,j}_{PD}(z) = \frac{1}{N}\sum_{i=1}^{N} f(x_i^{j|=z}). \tag{18.2}$$

### 18.3.2 Clustered partial dependence profiles

As it has been already mentioned, the average of CP profiles is a good summary if the profiles are parallel. If they are not parallel, the average may not adequately represent the shape of a subset of profiles. To deal with this issue, one can consider clustering the profiles and calculate the average separately for each cluster. To cluster the CP profiles, one may use standard methods like K-means or hierarchical clustering. The similarities between observations can be calculated based on the Euclidean distance between CP profiles.

Figure 18.2 illustrates an application of that approach to the random-forest model `titanic_rf_v6` (see Section 5.1.3) for 100 randomly selected instances (observations) from the Titanic dataset. The CP profiles for age are marked in grey. It can be noted that they could be split into three clusters based on the `hclust` method: one for a group of passengers with a substantial drop in the predicted survival probability for ages below 18 (with the average represented by the red line), one with an almost linear decrease of the probability over the age (with the average represented by the green line), and one with almost constant predicted probability (with the average represented by the blue line). The plot itself does not allow to identify the variables that may be linked with these clusters, but additional exploratory analysis could be performed for this purpose.

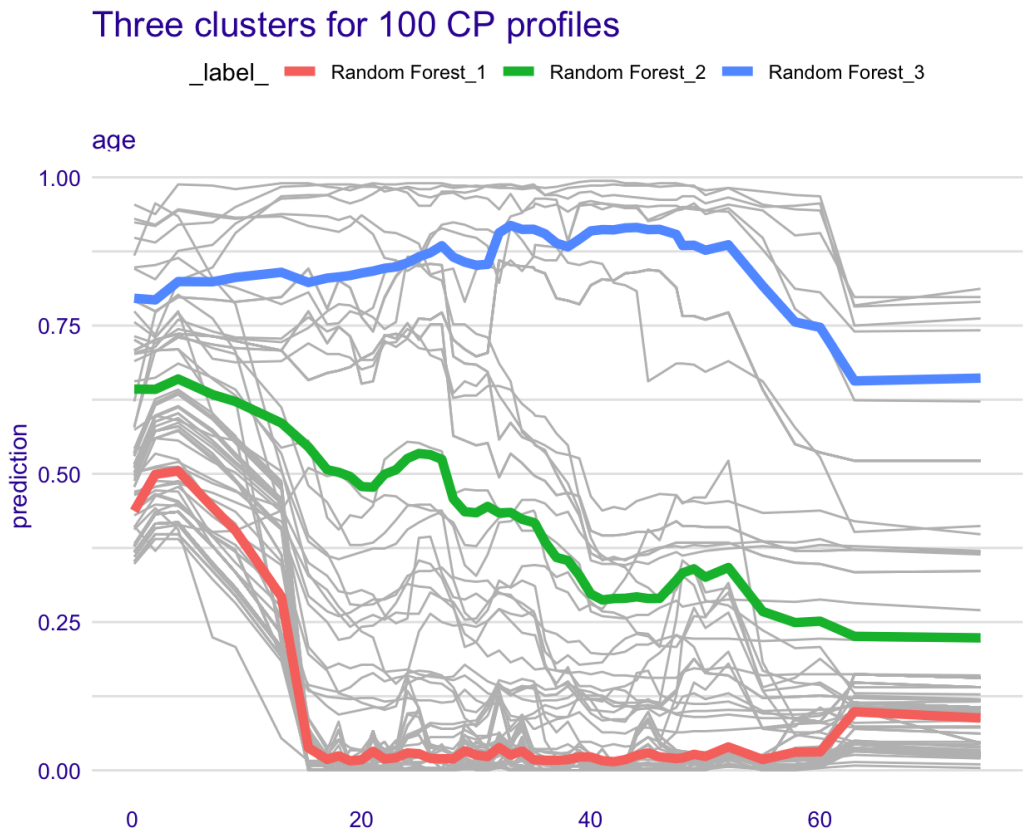



Figure 18.2: Clustered partial-dependence profiles for the random-forest model for 100 randomly selected observations from the Titanic dataset. Grey lines indicate Ceteris-paribus profiles that are clustered into 3 groups with the average profiles indicated by the blue, green, and red lines.

### 18.3.3 Grouped partial dependence profiles

It may happen that we can identify an explanatory variable that can influence the shape of CP profiles for the explanatory variable of interest. The most obvious situation is when a model includes an interaction between the variable of interest and another one. In that case, a natural approach is to investigate the PD profiles for the variable of interest corresponding to the groups of observations defined by the variable involved in the interaction. Figure 18.3 illustrates an application of the approach to the random-forest model `titanic_rf_v6` (see Section 5.1.3) for 100 randomly selected instances (observations) from the Titanic dataset. The CP profiles for age are marked in grey. The red and blue lines present the PD profiles for females and males, respectively. The latter have different shapes: the predicted survival probability for females is more stable across different ages, as compared to males. Thus, the PD profiles clearly indicate an interaction between age and gender.

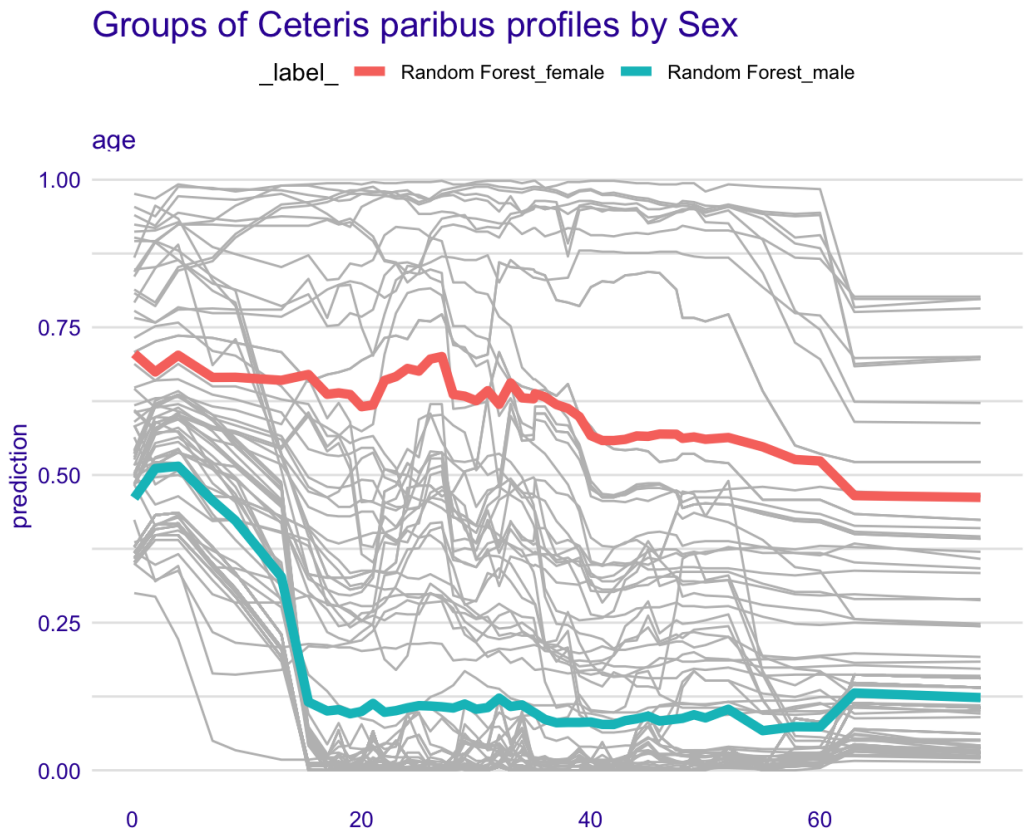



Figure 18.3: Partial-dependence profiles for two genders for the random-forest model for 100 randomly selected observations from the Titanic dataset. Grey lines indicate ceteris-paribus profiles for age.

## 18.3.4 Contrastive partial dependence profiles

Comparison of clustered or grouped PD profiles for a single model may provide important insight into, for instance, stability of the model predictions. PD profiles can also be compared between different models.

Figure 18.4 presents PD profiles for age for the random-forest model and the logistic regression model with splines for the Titanic data (see Section 5.1.3). The profiles are similar with respect to a general relation between age and the predicted probability of survival (the younger the passenger, the better chance of survival). However, the profile for the random-forest model is flatter. The difference between both models is the largest at the edges of the age scale. This pattern can be treated as expected, because random-forest models, in general, shrink predictions towards the average and they are not very good for extrapolation outside the range of values observed in the training dataset.

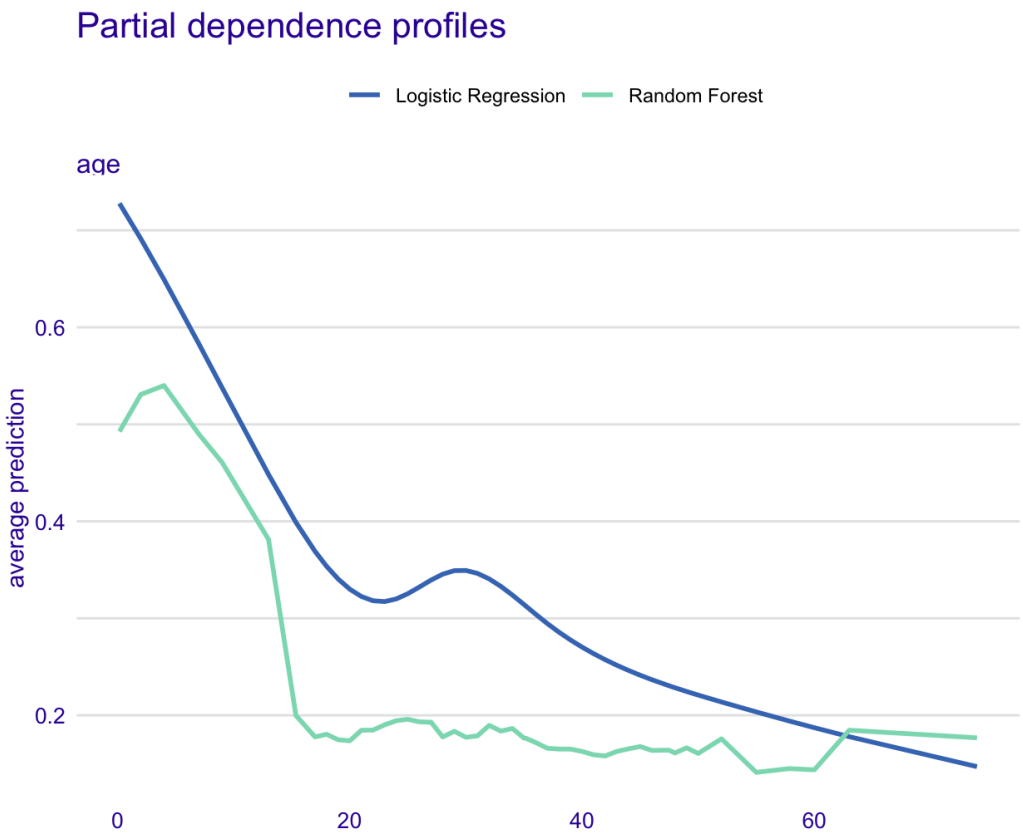



Figure 18.4: Partial-dependence profiles for age for the random-forest (green line) and logistic-regression (blue line) models for the Titanic dataset.

## 18.4 Example: Apartments data

In this section, we use PD profiles to evaluate performance of the random-forest model `apartments_rf_v5` (see Section 5.2.3) for the Apartments dataset (see Section @ref()). Recall that the goal is to predict the price per square-meter of an apartment. In our illustration we focus on two explanatory variables, surface and construction year.

### 18.4.1 Partial dependence profiles

Figure 18.5 presents CP profiles (green lines) for 25 randomly-selected apartments together with the estimated PD profile (blue line) for surface and construction year.

PD profile for surface suggest an approximately linear relationship between the explanatory variable and the predicted price. On the other hand, PD profile for construction year is U-shaped: the predicted price is the highest for the very new and very old apartments. While the

data were simulated, they were generated to reflect the effect of a lower quality of building materials used in housing construction after the II World War.

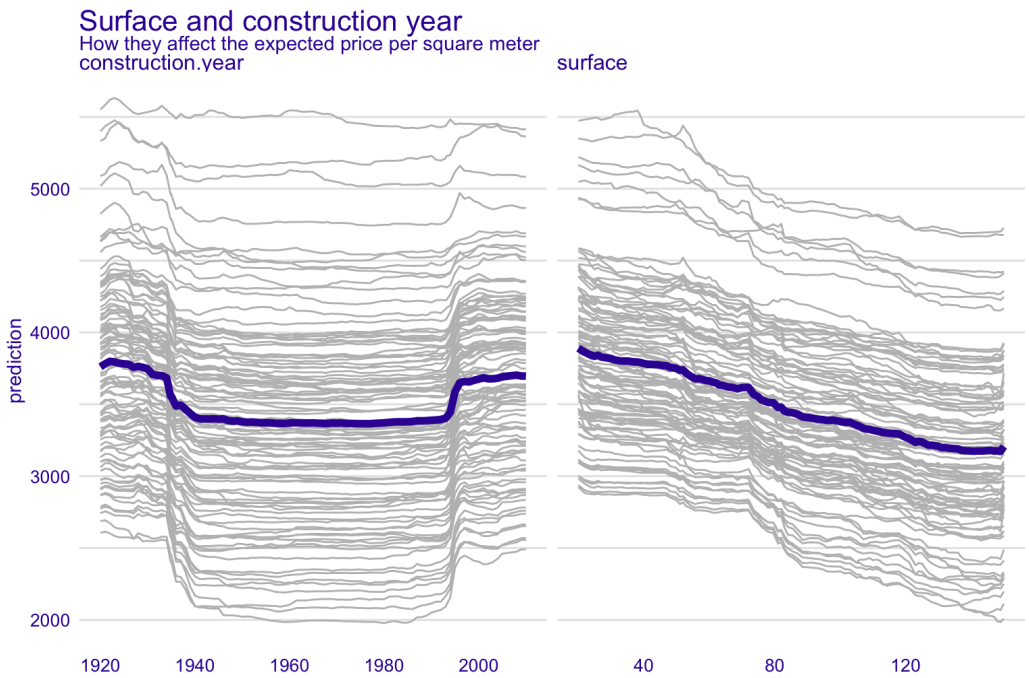



Figure 18.5: Ceteris-paribus and partial-dependence profiles for 100 randomly-selected apartments for the Random forest model for the Apartments dataset.

## 18.4.2 Clustered partial dependence profiles

All CP profiles for construction year, presented in Figure 18.5, seem to be U-shaped. The same shape is observed for the PD profile. One might want to confirm that the shape is, indeed, common for all the observations. The left-hand-side panel of Figure 18.6 presents clustered PD profiles for construction year for three clusters derived from the CP profiles presented in Figure 18.5. The three PD profiles differ slightly in the size of the oscillations at the edges, but they all are U-shaped. Thus, we could conclude that the overall PD profile adequately captures the shape of the CP profiles. Or, put differently, there is little evidence that there might be any strong interaction between construction year and any other variable in the model. Similar conclusions can be drawn for the CP and PD profiles for surface, presented in the right-hand-side panel of Figure 18.6.

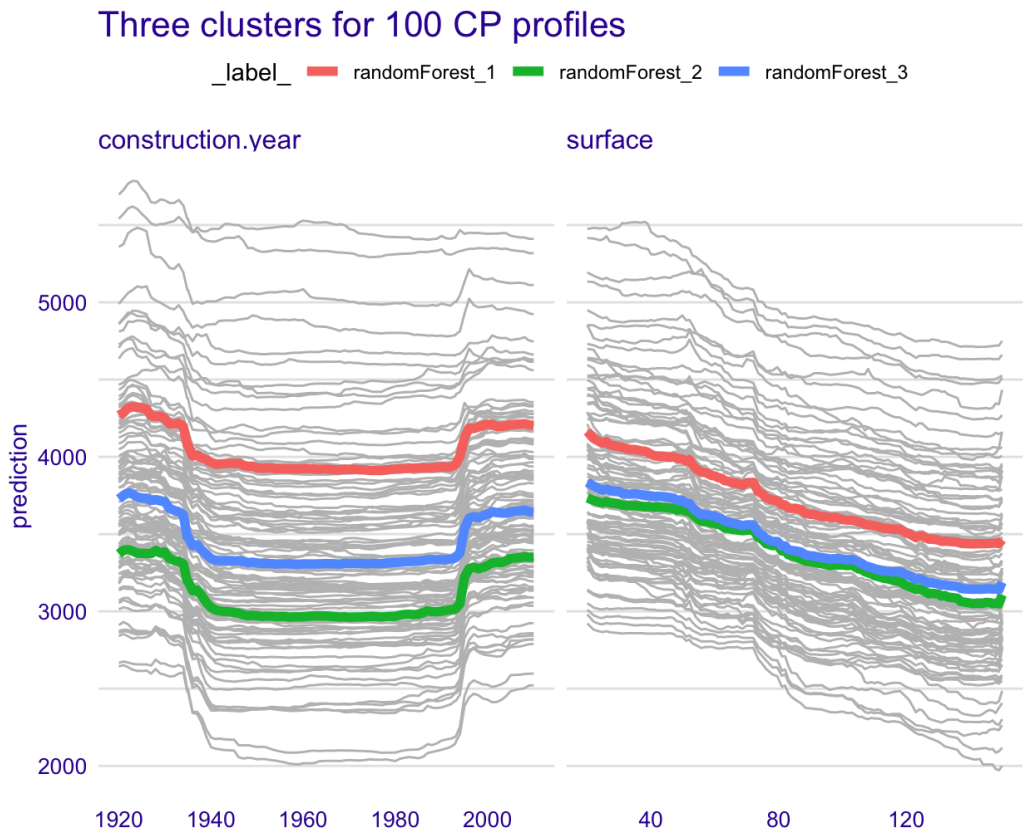



Figure 18.6: Ceteris-paribus (grey lines) and partial-dependence profiles (red, green and blue lines) for three clusters for 100 randomly-selected apartments for the random-forest model for the Apartments dataset. Left: profiles for construction year. Right: profiles for surface.

### 18.4.3 Grouped partial dependence profiles

One of the categorical explanatory variables in the Apartments dataset is district. We may want to investigate whether the relationship between the model predictions and construction year and surface is similar for all districts. Toward this aim, we can use grouped PD profiles, for groups of apartments defined by districts.

Figure 18.7 shows PD profiles for construction year (left-hand-side panel) and surface (right-hand-side panel) for each district. Several observations are worth making. First, profiles for apartments in ‘’Srodmiescie” (Downtown) are clearly much higher than for other districts. Second, the profiles are roughly parallel, indicating that the effects of construction year and surface are similar in each district. Third, the profiles appear to form three clusters, i.e., ‘’Srodmiescie” (Downtown), three districts close to ‘’Srodmiescie” (namely ‘’Mokotow’‘,’‘Ochota’‘, and’‘Ursynow”), and the six remaining districts.

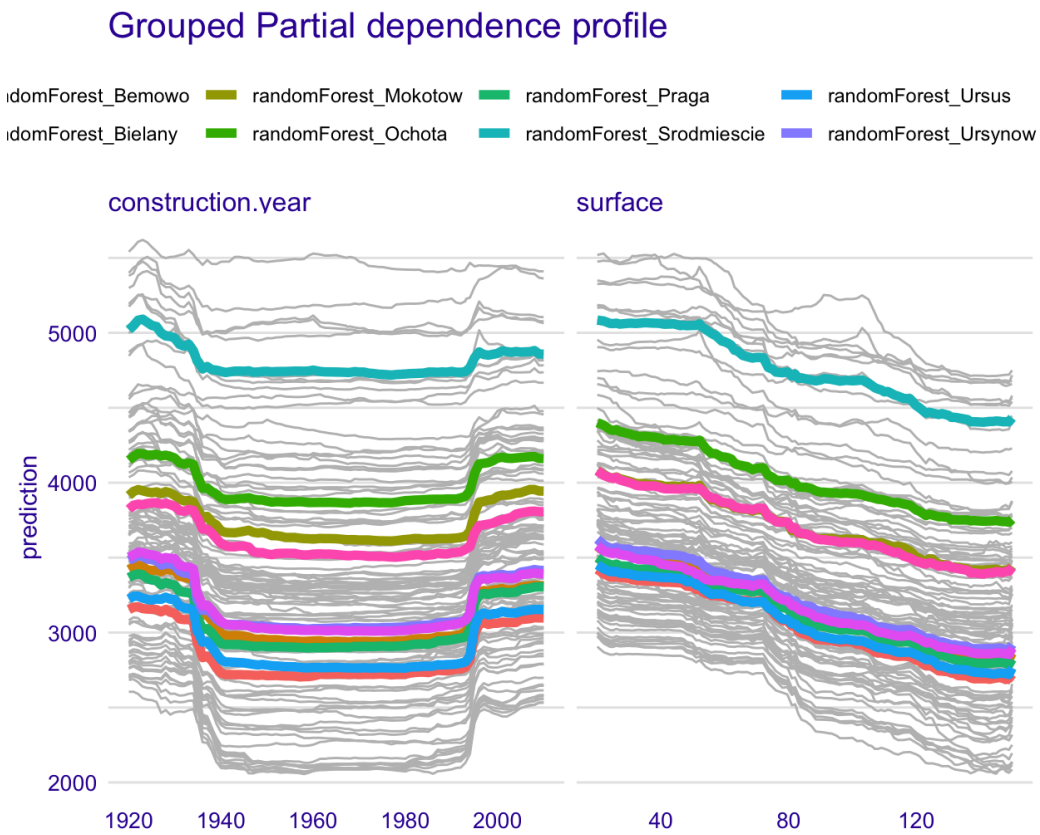



Figure 18.7: Partial-dependence profiles for separate districts for the random-forest model for the Apartments dataset. Left: profiles for construction year. Right: profiles for surface.

### 18.4.4 Contrastive partial dependence profiles

One of the main challenges in predictive modelling is to avoid over-fitting. The issue is particularly important for flexible models, such as random-forest models.

Figure 18.8 presents PD profiles for construction year (left-hand-side panel) and surface (right-hand-side panel) for the linear regression model (see Section @ref()) and the random-forest model. Several observations are worth making. The linear model cannot, of course, accommodate the non-monotonic relationship between the construction year and the price per square-meter. However, for surface, both models support a linear relationship, though the slope of the line resulting from the linear regression is steeper. This may be seen as an expected difference, given that random-forest models yield predictions that are shrunk towards the mean.

Thus, the profiles in Figure 18.8 suggest that both models miss some aspects of the data. In particular, the linear regression model does not capture the U-shaped relationship between the construction year and the apartment price. On the other hand, the effect of the surface on the apartment price seems to be underestimated by the random-forest model. Hence, one could conclude that, by addressing the issues, one could improve either of the models, possibly with an improvement in predictive performance.

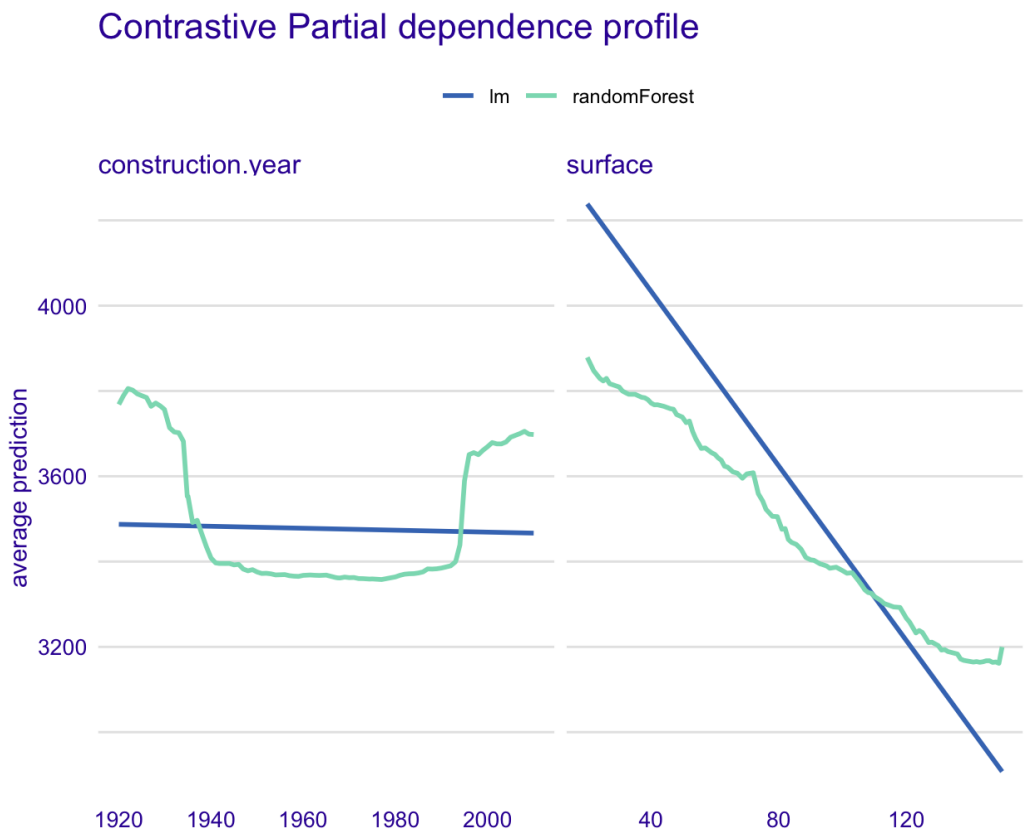



Figure 18.8: Partial-dependence profiles for the linear regression and random-forest models for the Apartments dataset. Left: profiles for construction year. Right: profiles for surface.

## 18.5 Pros and cons

PD profiles, presented in this chapter, offer a simple way to summarize the effect of a particular explanatory variable on the dependent variable. They are easy to explain and intuitive. They can be obtained for sub-groups of observations and compared across different models. For these reasons, they have gained in popularity and have been implemented in various software packages, including R and Python.

Given that the PD profiles are averages of CP profiles, they inherit the limitations of the latter. In particular, as CP profiles are problematic for correlated features, PD profiles are also not suitable for that case. (An approach to deal with this issue will be discussed in the next chapter.) For models including interactions, the averages of CP profiles may offer a crude and potentially misleading summarization.

## 18.6 Code snippets for R

Here we show partial dependence profiles calculated with `DALEX` package which wrap functions from `ingredients` package (Biecek et al. 2019). You will also find similar functions in the `pdp` package (Greenwell 2017), `ALEPlots` package (Apley 2018) or `iml` (Molnar, Bischl, and Casalicchio 2018) package.

The easiest way to calculate PD profiles is to use the function `DALEX::model_profile` . The only required argument is the explainer and by default PD profiles are calculated for all variables. The only required argument is the model explainer. By default, PD profiles are calculated for all explanatory variables. In the code below we use the `variables` argument to limit the list of variables for which PD profiles are calculated. We store the computed PD profile in object `pdp_rf` . Subsequently, we apply the `plot()` function to the object to generate the plot of the PD profile.

For illustration purposes, we use the random-forest model `titanic_rf_v6` (see Section **??**) for the Titanic data. Recall that it is developed to predict the probability of survival from sinking of Titanic. Below we use `variables` argument to limit list of variables for which PD profiles are calculated. Here we need profiles only for the `age` variable.

```
pdp_rf <- model_profile(explain_titanic_rf, variables = "age")
plot(pdp_rf) +
  ggtitle("Partial dependence profile for age")
```

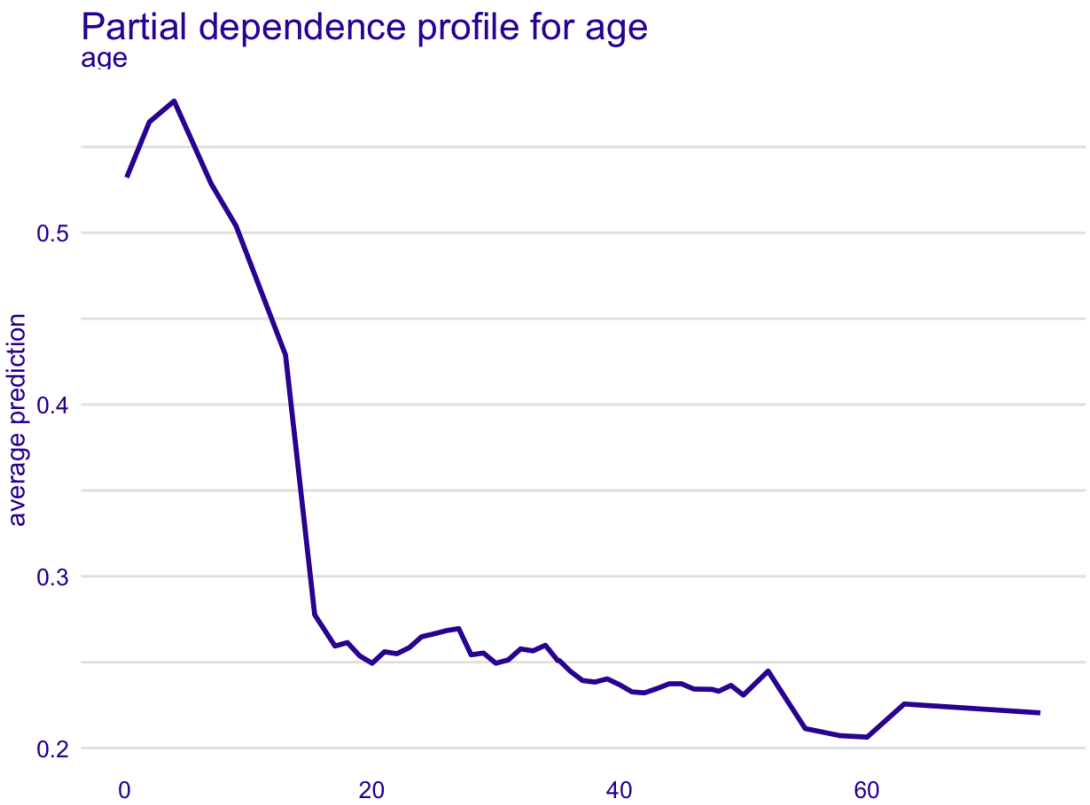



Figure 18.9: Partial dependence profile for age.

PD profiles can be plotted on top of CP profiles. This is a very useful feature if we want to learn how similar are the CP profiles to the average. Toward this aim, we first have got to compute and store the CP profiles with the help of the `model_profile()` function. The argument `N` set the number of randomly-selected instances used for calculation of partial dependence. By default its 100.

The argument `geom = "profiles"` in the `plot()` function results in Partial dependence profile plotted on top of Ceteris paribus profiles. In the example below we only select the profiles for age.

```
pdp_rf <- model_profile(explain_titanic_rf, variables = "age")
plot(pdp_rf, geom = "profiles") +
  ggtitle("Ceteris Paribus and Partial dependence profiles for age")
```

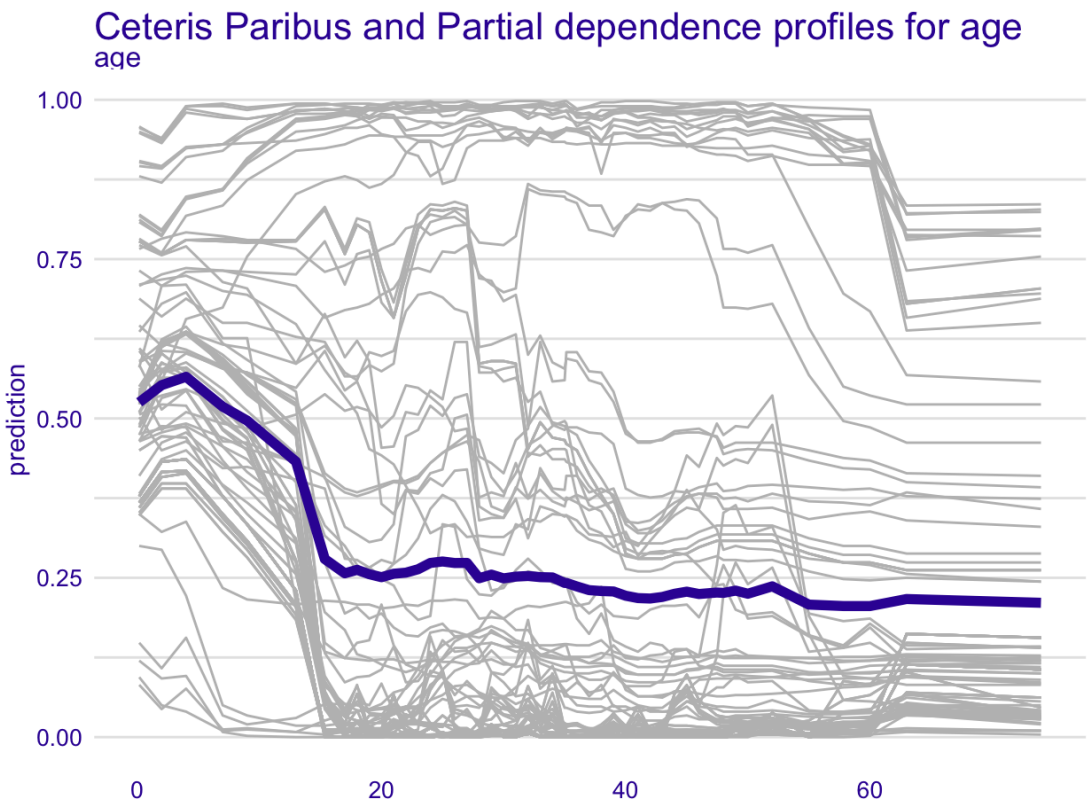



Figure 18.10: Ceteris-paribus and partial-dependence profiles for age.

### 18.6.1 Clustered partial dependence profiles

To calculate clustered PD profiles, first we have to calculate and store the CP profiles and the use the `hclust` clustering to the profiles. This can be done with the `model_profile()` function. The number of clusters is specified with the help of argument `k` . Additional arguments of the function include `center` (a logical argument indicating if the profiles should be centered before calculation of distances between them) and `variables` (a list with the names of the explanatory variables for which the profiles are to be clustered, with the default value `NULL` indicating all the available variables).

The clustered PD profiles can be plotted on top of the CP profiles by setting the `geom = "profiles"` argument to the `plot()` function. Note that in the R code below we perform the calculations only for a randomly-selected set of 100 observations from the `titanic` data frame. Also, we only select the plots for the profiles for `age` .

```
pdp_rf <- model_profile(explain_titanic_rf, variables = "age", k = 3)
plot(pdp_rf, geom = "profiles") +
  ggtitle("Clustered Partial dependence profiles")
```

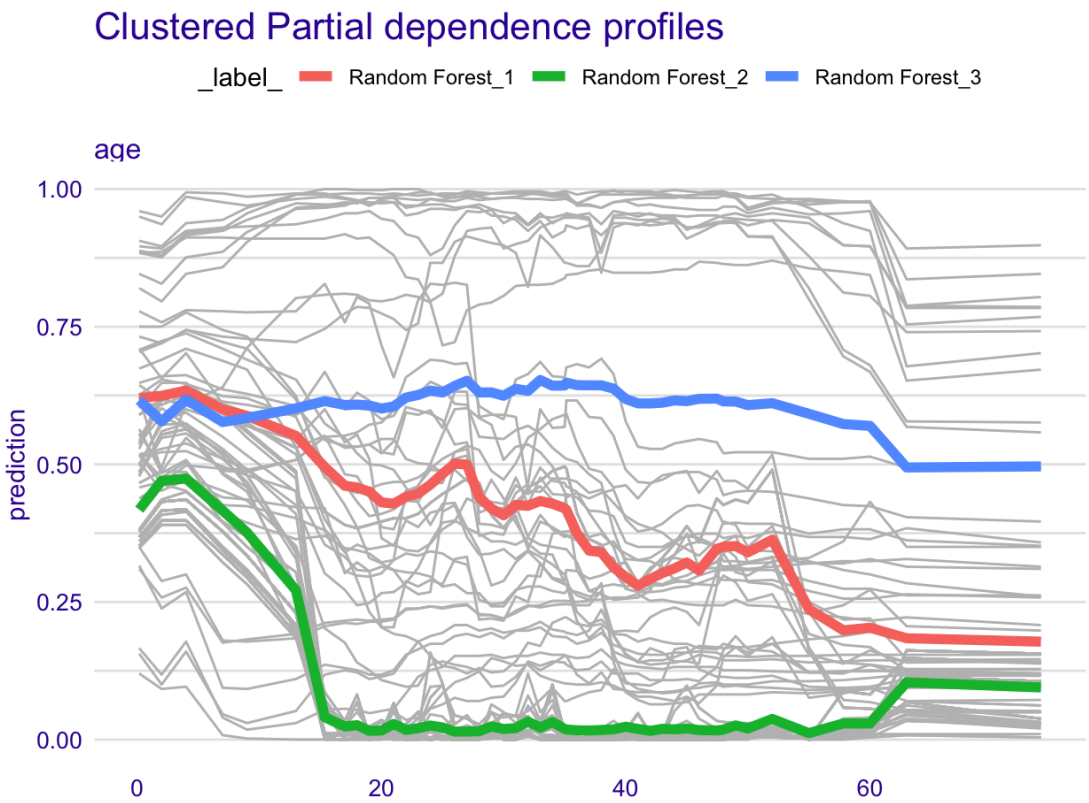



Figure 18.11: Clustered Partial dependence profiles.

## 18.6.2 Grouped partial dependence profiles

The `model_profile()` function admits the `groups` argument. If the argument is set to the name of a categorical explanatory variable, PD profiles are constructed for the groups of observations defined by the levels of the variable. In the example below, the argument is applied to obtain PD profiles for `age` grouped by `gender`. Subsequently, the profiles are plotted on top of the CP profiles for 100 randomly-selected observations from the `titanic` data frame (stored in object `pdp_sex_rf`).

```
pdp_sex_rf <- model_profile(explain_titanic_rf, variables = "age", groups = "gender")
plot(pdp_sex_rf, geom = "profiles") +
  ggtitle("Grouped Partial dependence profiles")
```

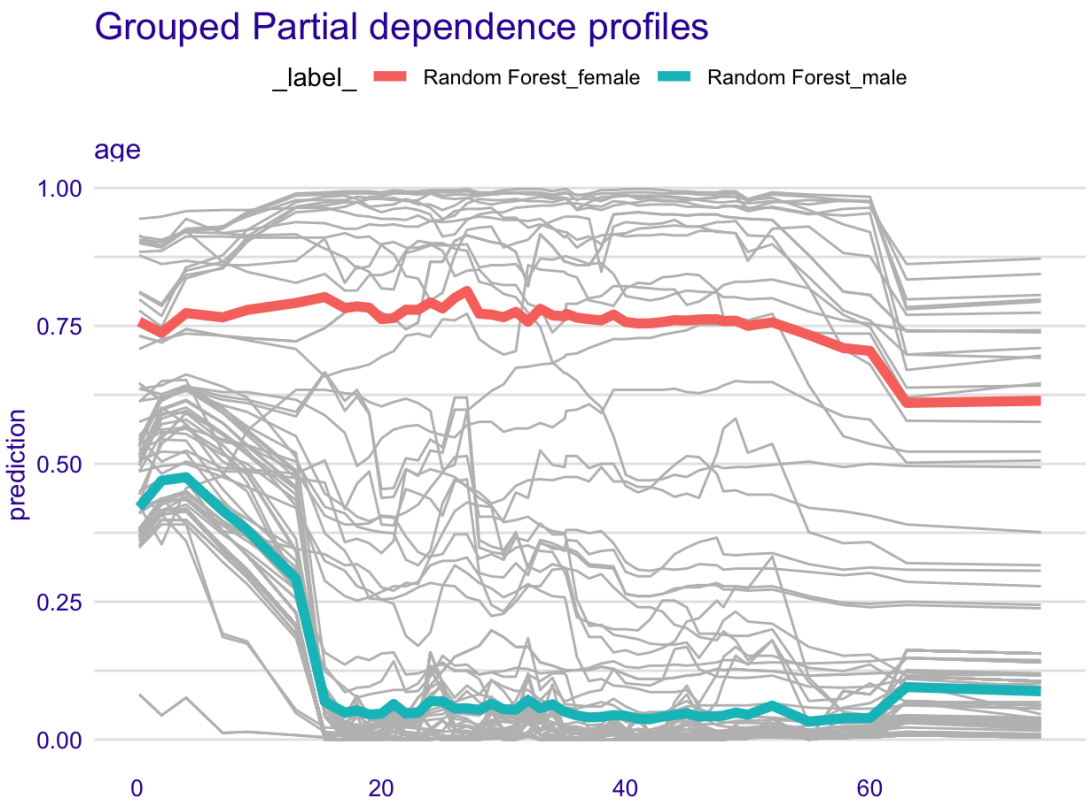



Figure 18.12: Grouped Partial dependence profiles.

## 18.6.3 Contrastive partial dependence profiles

To overlay PD profiles for two or more models in a single plot, one can use the generic `plot()` function. In the code below, we create PD profiles for `age` for the random-forest (see Section **??**) and logistic regression (see Section **??**) models, stored in the explainer-objects `explain_titanic_rf` and `explain_titanic_lmr`, respectively. Subsequently, we apply the `plot()` function to plot the two PD profiles together in a single plot.

```
pdp_rf <- model_profile(explain_titanic_rf, variables = "age")
pdp_lmr <- model_profile(explain_titanic_lmr, variables = "age")

plot(pdp_rf$agr_profiles, pdp_lmr$agr_profiles) +
  ggtitle("Contrastive Partial dependence profiles")
```

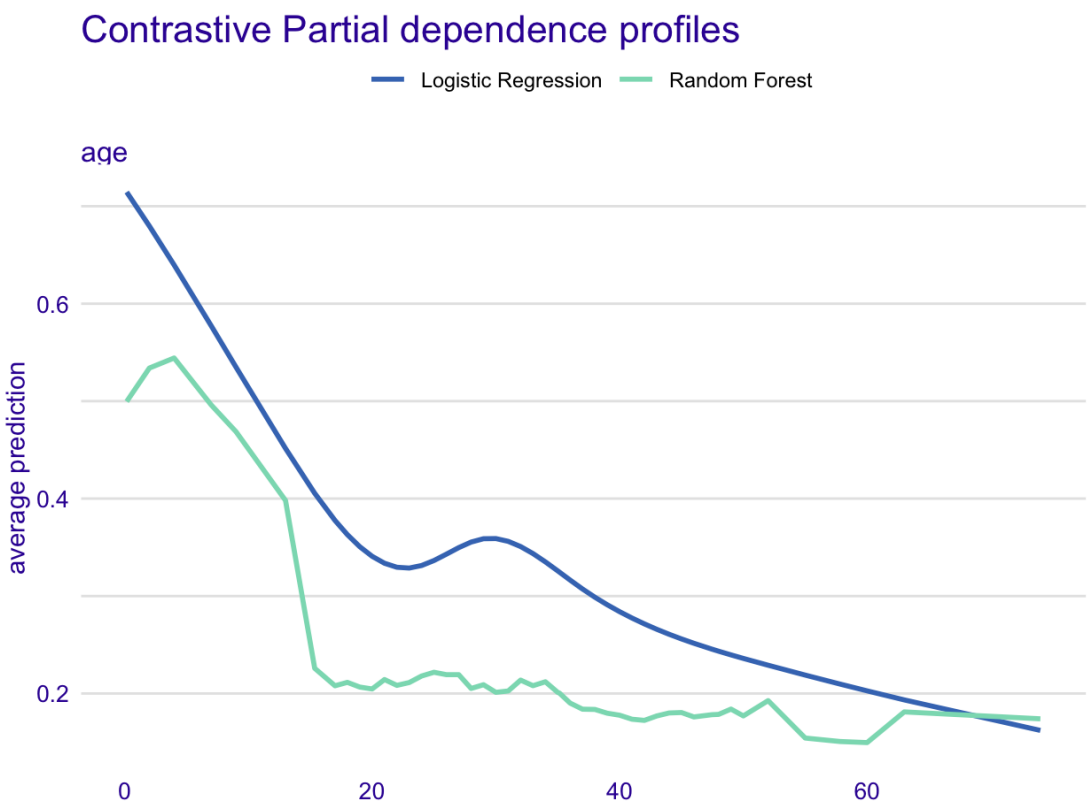



Figure 18.13: Contrastive Partial dependence profiles.

# References


Apley, Dan. 2018. *ALEPlot: Accumulated Local Effects (Ale) Plots and Partial Dependence (Pd) Plots*. https://CRAN.R-project.org/package=ALEPlot.

Biecek, Przemyslaw, Hubert Baniecki, Adam Izdebski, and Katarzyna Pekala. 2019. *ingredients: Effects and Importances of Model Ingredients*.

Friedman, Jerome H. 2000. “Greedy Function Approximation: A Gradient Boosting Machine.” *Annals of Statistics* 29: 1189–1232.

Greenwell, Brandon M. 2017. “pdp: An R Package for Constructing Partial Dependence Plots.” *The R Journal* 9 (1): 421–36. https://journal.r-project.org/archive/2017/RJ-2017-016/index.html.

Molnar, Christoph, Bernd Bischl, and Giuseppe Casalicchio. 2018. “iml: An R package for Interpretable Machine Learning.” *Joss* 3 (26). Journal of Open Source Software: 786. https://doi.org/10.21105/joss.00786.

# 19 Local-dependence and Accumulated Local Profiles

## 19.1 Introduction

Partial-dependence (PD) profiles, introduced in the previous chapter, are easy to explain and interpret, especially given their estimation as an average of Ceteris-paribus (CP) profiles. However, as it was mentioned in Section 18.5, the profiles may be misleading if the explanatory variables are correlated. In many applications, this is the case. For example, in the Apartments dataset (see Section 5.2), one can expect that variables ‘’surface” and ‘’number of rooms” may be positively correlated, because apartments with larger number of rooms usually also have a larger surface. Thus, it is not realistic to consider, for instance, an apartment with 5 rooms and 20 square meters. Similarly, in the Titanic dataset, a positive correlation can be expected for the values of variables ‘’fare” and ‘’passenger class”, as tickets in the higher classes are more expensive than in the lower classes.

In this chapter, we present accumulated local (AL) profiles that address this issue. As they are related to local-dependence (LD) profiles, we introduce the latter first. Both approaches were proposed in ,,ALEPlot: Accumulated Local Effects (ALE) Plots and Partial Dependence (PD) Plots” (Apley 2018).

## 19.2 Intuition

Let us consider the following, simple nonlinear model with for two explanatory variables:

$$f(x_1, x_2) = (x_1 + 1) \cdot x_2. \tag{19.1}$$

Moreover, assume that explanatory variables $X^1$ and $X^2$ are uniformly distributed over the interval $[-1, 1]$ and perfectly correlated, i.e., $X^2 = X^1$. Suppose that we have got the following dataset with 8 observations:

| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $X^1$ | -1 | -0.71 | -0.43 | -0.14 | 0.14 | 0.43 | 0.71 | 1 |
| $X^2$ | -1 | -0.71 | -0.43 | -0.14 | 0.14 | 0.43 | 0.71 | 1 |
| $y$ | 0 | -0.2059 | -0.2451 | -0.1204 | 0.1596 | 0.6149 | 1.2141 | 2 |

Note that, for both $X^1$ and $X^2$, the sum of all observed values is equal to 0.

The top part of Panel A of Figure 19.1 shows CP profiles for $X^1$ for model (19.1) calculated for the eight observations. The bottom part of the panel presents the corresponding estimate of the PD profile for $X^1$, i.e., the average of the CP profiles. The profile suggests no effect of variable $X^1$, which is clearly a misleading conclusion.

To understand the reason, let us explicitly express the CP profile for $X^1$ for model (19.1):

$$h_{x_1,x_2}^{(x_1+1)\cdot x_2,1}(z) = f(z, x_2) = (z+1) \cdot x_2. \tag{19.2}$$

By allowing $z$ to take any value in the interval $[-1, 1]$, we get the CP profiles as straight lines with the slope equal to the value of variable $X^2$. Hence, for instance, the CP profile for observation $(-1, -1)$ is a straight line with the slope equal to -1.

Recall that the PD profile for $X^j$, defined in equation (18.1), is the expected value, over the joint distribution of all explanatory variables other than $X^j$, of the model predictions when $X^j$ is set to $z$. This leads to the estimation of the profile by taking the average of CP profiles for $X^j$ (see equation (18.2)).

In our case, this implies that the PD profile for $X^1$ is the expected value of the model predictions over the distribution of $X^2$, i.e., over the uniform distribution on the interval $[-1, 1]$. Thus, the PD profile is estimated by taking the average of the CP profiles (see (19.2)) at each value of $z$ in $[-1, 1]$:

$$\hat{g}_{PD}^{(x_1+1)\cdot x_2,1}(z) = \frac{1}{8}\sum_{i=1}^{8}(z+1)\cdot x_{2,i} = \frac{z+1}{8}\sum_{i=1}^{8} x_{2,i} = 0. \tag{19.3}$$

As a result, the PD profile for $X^1$ is estimated as a horizontal line at 0, as seen in the bottom part of Panel A of Figure 19.1.

The calculations in equation (19.3) ignore the fact that, given our assumptions, one cannot change $z$ *freely* for a particular value of $X^2$, because $X^1$ and $X^2$ are assumed to be perfectly correlated. In fact, in this case, the CP profile for the $i$-th observation should actually be undefined for any values of $z$ different from $x_i^2$. As a consequence, the sum used in the calculation of the PD profile in equation (19.3) would involve undefined terms for any $z$.

Thus, the issue is related to the fact of using the marginal distribution of $X^2$, which disregards the value of $X^1$, in the definition of the PD profile. This observation suggests a modification: instead of the marginal distribution, one might consider the conditional distribution of $X^2|X^1$. The modification leads to the definition of the LD profile.

For in our example, the conditional distribution of $X^2$, given $X^1 = z$, is just a probability mass of 1 at $z$. Consequently, the Local-dependence (LD) profile, for any $z \in [-1, 1]$, is given by

$$g_{LD}^{(x_1+1)\cdot x_2,1}(z) = z \cdot (z + 1). \tag{19.4}$$

It turns out, however, that the modification does not fully address the issue of correlated explanatory variables.

Looking at equation (19.1) one can see that from the perspective of $X^1$ the effect of the variable $x_1$ is similar to the function $f(x_1) = -x_1 - 1$ for values close to $(x_1, x_2) = (-1, -1)$ and similar to the function $f(x_1) = x_1 + 1$ for values close to $(x_1, x_2) = (1, 1)$. So first it decreases at the same rate as it increases later. The function in equation (19.4) does not show this.

In the (Apley 2018), author proposed Accumulated Local Effects (AL) profiles, where the effect of the $X^1$ variable is defined as the cumulative sum of local derivatives due to the $X^1$ variable

$$g_{AL}^{(x_1+1)\cdot x_2,1}(z) = \int_{-1}^{z} E\left[\frac{\partial f(x_1, x_2)}{\partial x_1}|X^1 = v\right] dv = \int_{-1}^{z} E\left[X^2|X^1 = v\right] dv = \int_{-1}^{z} v dv$$

This formula better reflects the behavior of function $f(x_1, x_2)$ due to $X^1$ removing the effect of changes in variable $X^2$.

# 19.3 Method

## 19.3.1 Local-dependence profile

LD profile for model $f(x)$ and variable $X^j$ is defined as follows:

$$g_{LD}^{f,j}(z) = E_{X^{-j}|X^j=z}\left[f\left(X^{j|=z}\right)\right]. \tag{19.6}$$

Thus, it is the expected value of the model predictions over the conditional distribution of $X^{-j}$ given $X^j = z$, i.e., over the joint distribution of all explanatory variables other than $X^j$ conditional on the value of the latter variable set to $z$. Or, in other words, it is the expected

value of the CP profiles in equation (11.1) for $X^j$ over the conditional distribution of $X^{-j}|X^j$.

For example, consider $X = (X^1, X^2)$ with $X^1$ uniformly distributed over $[-1, 1]$ and $X^2 = X^1$. Then the conditional distribution of $X^{-1} \equiv X^2$, given $X^1 = z$, is the point mass of 1 at $z$. Consequently, for model (19.1) and variable $X^1$, we get

$$g_{LD}^{(x_1+1)\cdot x_2,1}(z) = E_{X^2|X^1=z}[f(z, X^2)] = E_{X^2|X^1=z}[(z+1)\cdot X^2] = (z+1)\cdot E_{X^2|X^1=z}(X^2) = ($$

as given in Equation (19.4).

As proposed in ,,ALEPlot: Accumulated Local Effects Plots and Partial Dependence Plots" (Apley 2018), LD profile can be estimated as follows:

$$\hat{g}_{LD}^{f,j}(z) = \frac{1}{|N_j|} \sum_{k \in N_j} f\left(x_k^{j|=z}\right), \tag{19.7}$$

where $N_j$ is the set of observations with the value of $X^j$ ''close" to $z$ that is used to estimate the conditional distribution of $X^{-j}|X^j = z$.

Note that, in general, the estimator given in formula (19.7) is neither smooth nor continuous at boundaries between subsets $N_i$. A smooth estimator for $g_{LD}^{f,j}(z)$ can be defined as follows:

$$\tilde{g}_{LD}^{f,j}(z) = \frac{1}{\sum_k w_k(z)} \sum_{i=1}^{N} w_i(z) f(x_i^{j|=z}), \tag{19.8}$$

where weights $w_i(z)$ capture the distance between $z$ and $x_i^j$. In particular, for a categorical variable, we may just use the indicator function $w_i(z) = 1_{z=x_i^j}$, while for a continuous variable we may use the Gaussian kernel:

$$w_i(z) = \phi(z - x_i^j, 0, s), \tag{19.9}$$

where $\phi(y, 0, s)$ is the density of a normal distribution with mean 0 and standard deviation $s$. Note that $s$ plays the role of a smoothing factor.

As argued in ,,Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models" (Apley and Zhu 2019), if an explanatory variable is correlated with some other variables, the LD profile for the variable will capture the effect of all of the variables. This is because the profile is obtained by marginalizing (in fact, ignoring) over the remaining variables in the model. Thus, in this respect, LD profiles share the same limitation as PD profiles. To address the limitation, AD profiles can be used. We present them in the next section.

## 19.3.2 Accumulated local profile

Consider model $f(x)$ and define

$$h^j(u) = \left[\frac{\partial f(x)}{\partial x^j}\right]_{x=u}.$$

The AL profile for model $f(x)$ and variable $X^j$ is defined as follows:

$$g_{AL}^{f,j}(z) = \int_{z_0}^{z} \left\{E_{X^{-j}|X^j=u}\left[h^j(X^{j|=u})\right]\right\} du + c, \tag{19.10}$$

where $z_0$ is a value close to the lower bound of the effective support of the distribution of $X^j$ and $c$ is a constant, usually selected so that $E_{X^j}\left[g_{AL}^{f,j}(X^j)\right] = E_X f(x)$.

To interpret equation (19.10) note that $h^j(u)$ describes the local effect (change) of the model due to $X^j$. Or, to put it in other words, $h^j(u)$ describes how much the CP profile for $X^j$ changes at $u$. This effect (change) is averaged over the ''relevant" (according to the conditional distribution of $X^{-j}|X^j$) values of $X^{-j}$ and, subsequently, accumulated (integrated) over values of $u$ up to $z$. As argued by (Apley and Zhu 2019), the averaging of the local effects allows avoiding the issue, present in the PD and LD profiles, of capturing the effect of other variables in the profile for a particular variable. To see this, one can consider the approximation

$$f(u^{j|=u+du}) - f(u^{j|=u}) \approx h^j(u)du,$$

and note that the difference $f(u^{j|=u+du}) - f(u^{j|=u})$, for a model without interaction, effectively removes the effect of all variables other than $X^j$. For example, consider model $f(x_1, x_2) = x_1 + x_2$, with $f_1(x_1, x_2) = 1$. Then

$$f(u + du, x_2) - f(u, x_2) = (u + du + x_2) - (u + x_2) = du = f_1(u, x_2)du.$$

As another example, consider model (19.1). In this case, $f_1(x_1, x_2) = x_2$ and the effect of variable $X^2$ will still be present in the AL profile for $X^1$.

Continuing with model in formula (19.1), assume that $X = (X^1, X^2)$ with $X^1$ uniformly distributed over $[-1, 1]$ and $X^2 = X^1$. In this case, the conditional distribution of $X^2|X^1 = z$ is the point mass of 1 at $z$. Then

$$g_{AL}^{(x_1+1)\cdot x_2,1}(z) = \int_{-1}^{z} \left[E_{X^2|X^1=u}\left(X^2\right)\right] du + c = \int_{-1}^{z} u du + c = \frac{z^2 - 1}{2} + c. \tag{19.5}$$

Since $E_{X^1}\left[(X^1)^2\right] = 1/3$, then, upon taking $c = -1/3$, we get $E_{X^1}\left[g_{AL}^{(x_1+1)\cdot x_2,1}(X^1)\right] = 0$.

To estimate AL profile, one replaces the integral in equation (19.10) by a summation and the derivative with a finite difference (Apley 2018). In particular, consider a partition of the range of observed values $x_i^j$ of variable $X^j$ into $K$ intervals $N_j(k) = (z_{k-1}^j, z_k^j]$ $(k = 1, \ldots, K)$. Note that $z_0^j$ can be chosen just below $\min(x_1^j, \ldots, x_N^j)$ and $z_K^j = \max(x_1^j, \ldots, x_N^j)$. Let $n_j(k)$ denote the number of observations $x_i^j$ falling into $N_j(k)$, with $\sum_{k=1}^K n_j(k) = N$. An estimator of AL profile for variable $X^j$ can then be constructed as follows:

$$\hat{g}_{AL}^{f,j}(z) = \sum_{k=1}^{k_j(z)} \frac{1}{n_j(k)} \sum_{i:x_i^j \in N_j(k)} \left[ f(x_i^{j|=z_k^j}) - f(x_i^{j|=z_{k-1}^j}) \right] - \hat{c}, \tag{19.11}$$

where $k_j(z)$ is the index of interval $N_j(k)$ in which $z$ falls, i.e., $z \in N_j[k_j(z)]$, and $\hat{c}$ is selected so that $\sum_{i=1}^n \hat{g}_{AL}^{f,j}(x_i^j) = 0$. To interpret formula (19.11) note that difference $f\left(x_i^{j|=z_k^j}\right) - f\left(x_i^{j|=z_{k-1}^j}\right)$ corresponds to the difference of the CP profile for the $i$-th observation at the limits of interval $N_j(k)$. This differences for are then averaged across all observations falling into the interval and accumulated.

Note that, in general, $\hat{g}_{AL}^{f,j}(z)$ is not smooth at the boundaries of intervals $N_j(k)$. A smooth estimate can obtained as follows:

$$\tilde{g}_{AL}^{f,j}(z) = \sum_{k=1}^{K} \frac{1}{\sum_l w_l(z_k)} \sum_{i=1}^{N} w_i(z_k) \left[ f(x_i^{j|=z_k}) - f(x_i^{j|=z_k-\Delta}) \right] - c, \tag{19.12}$$

where points $z_k$ $(k = 0, \ldots, K)$ form a uniform grid covering the interval $(z_0, z)$ with step $\Delta = (z - z_0)/K$, and weight $w_i(z_k)$ captures the distance between point $z_k$ and observation $x_i^j$. In particular, we may use similar weights as in case of equation (19.8).

### 19.3.3 An illustrative example

Let us consider model (19.1) and explanatory-variable vector $X = (X^1, X^2)$ with $X^1$ uniformly distributed over $[-1, 1]$ and $X^2 = X^1$. Hence, $X^2$ is perfectly correlated with $X^1$. Moreover, consider the eight observations for $X$ from Section 19.2.

The top part of Panel A of Figure 19.1 shows CP profiles for $X^1$ for the eight observations, as computed in formula (19.2). The bottom part of the panel shows the estimated PD profile obtained by using the average of the CP profiles. As indicated by formula (19.3), the PD profile is estimated at 0. The estimate is correct, as

$$g_{PD}^{(x_1+1)\cdot x_{2,1}}(z) = E_{X^2}[(z+1) \cdot X^2] = (z+1) \cdot E_{X^2}(X^2) = 0,$$

given that $X^2$ is uniformly-distributed over $[-1, 1]$. It is, however, misleading, as there clearly is an effect of $X^1$.

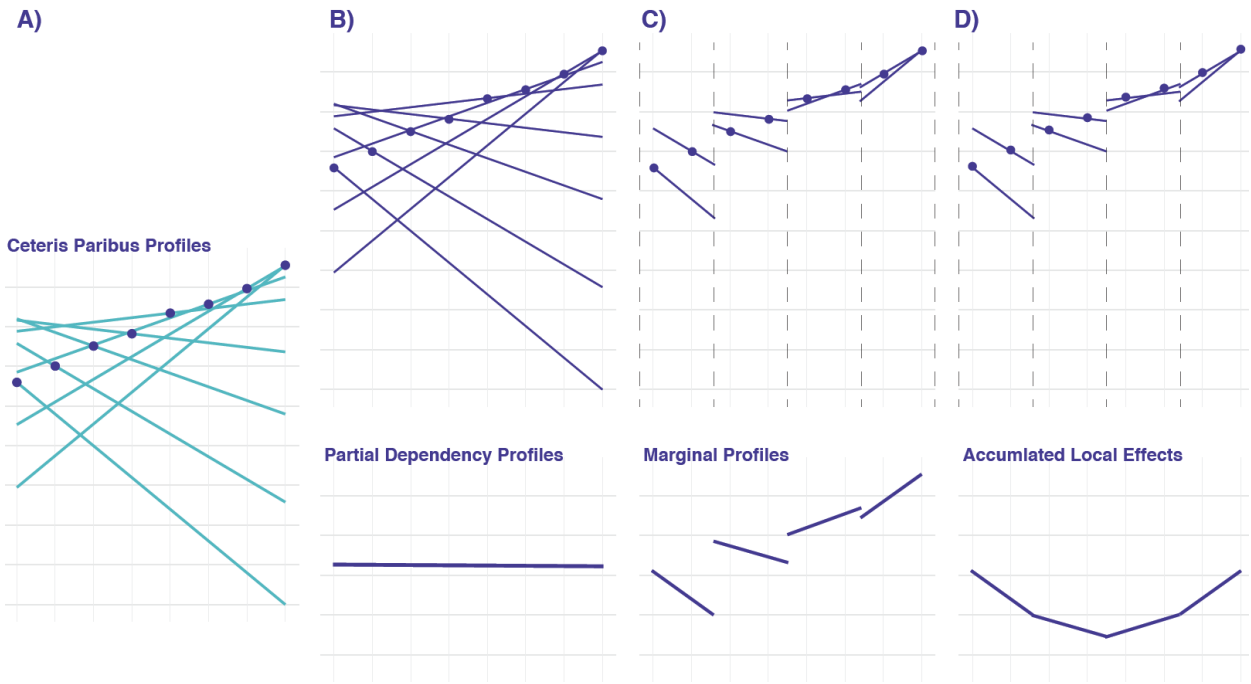



Figure 19.1: Partial-dependence, local-dependence, and accumulated local profiles. Panel A: Ceteris-paribus profiles (top plot) and the corresponding partial-dependence profile. Panel B: local-dependence profile. Panel C: accumulated local profile.

The LD profile for model (19.1), as computed in (19.4), is

$$g_{LD}^{(x_1+1)\cdot x_2,1}(z) = (z+1)\cdot z.$$

By using estimator (19.7), with the data split into four intervals each containing two observations, we obtain the estimated LD profile shown at the bottom of Panel B of Figure 19.1.

As shown in (19.5), the AL profile for model presented in formula (19.1) is up to the constant equal

$$g_{AL}^{(x_1+1)\cdot x_2,1}(z) = \frac{z^2-1}{2} + c.$$

By using estimator (19.11), with the data split into four intervals each containing two observations, we obtain the estimated AL profile shown at the bottom of Panel C of Figure 19.1.

## 19.4 Example: Apartments data

In this section, we use PD, LD, and AL profiles to evaluate performance of the random-forest model `apartments_rf_v5` (see Section 5.2.3) for the Apartments dataset (see Section 5.2). Recall that the goal is to predict the price per square-meter of an apartment. In our illustration we focus on two explanatory variables, surface and the number of rooms, as they are correlated (see Figure 5.9).

Figure 19.2 shows the three types of profiles for both variables estimated according to formulas (18.2), (19.8) and (19.12).

Number of rooms and surface are two correlated variables, moreover both have some effect on the price per square meter. As we see profiles calculated with different methods are different.

The green curve corresponds to the PD profile, which is the ordinary average of CP profiles. The red curve corresponds to the LD profile. It is steeper because the effect of the `surface` variable is additionally overlapped by the effects of variables correlated with `surface`, e.g. number of rooms. The blue curve corresponds to the AL profile. The estimator for AL profiles eliminates the effect of correlated variables. Since the AL and PD profiles are parallel to each other, it suggests that the model is additive for these two variables. In other words the effect of the `surface` variable in the model does not depend on the value of other variables.

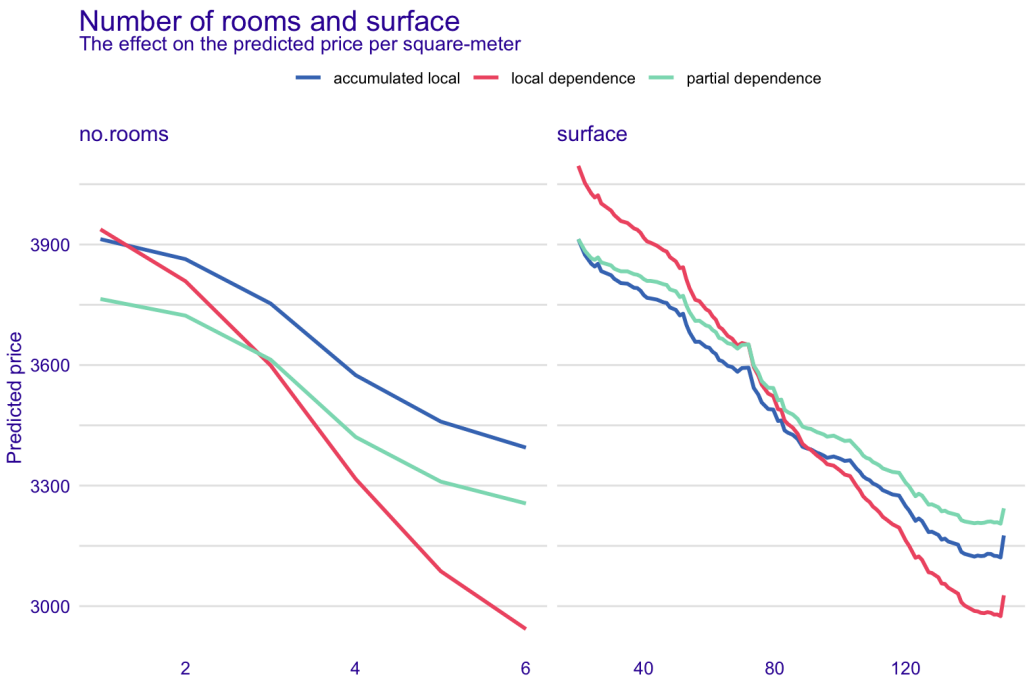



Figure 19.2: Partial dependence, local dependence, and accumulated local profiles for the random forest model for the Apartments dataset.

## 19.5 Pros and cons

In this chapter we introduced tools for exploration of the relation between model response and model inputs. These tools are useful to summarize how ,,in general" model responds to the input of interest. All presented approaches are based on Ceteris-paribus profiles introduced in Chapter 11 but they differ in a way how individual profiles are merged into a global model response.

When the variables in the data are independent and there are no interactions in the model, the Ceteris-paribus profiles are parallel and their average, i.e. Partial Dependence, summarizes them all well.

When there are interactions in the model, the Ceteris-paribus profiles are not parallel. Additionally, if variables in the data are correlated, averaging entire Ceteris-paribus profiles may distort the relationship described by the model.

Comparison of PD, LD and AL profiles allows to identify if there are any interactions in the model and if data are significantly correlated. When there are interactions, it may be helpful to explore these relations with generalization of PD profiles for two or more dependent variables.

## 19.6 Code snippets for R

In this section, we present key features of R package `DALEX` which is a wrapper over package `ingredients` (Biecek et al. 2019). Similar functionalities can be found in package `ALEPlots` (Apley 2018) or `iml` (Molnar, Bischl, and Casalicchio 2018).

For illustration purposes, we use the random-forest model `apartments_rf_v5` (see Section 5.2.3) for the Apartments dataset (see Section @ref()). Recall that the goal is to predict the price per square-meter of an apartment. In our illustration we focus on two explanatory variables, surface and the number of rooms.

LD profiles are calculated by applying function `DALEX::variable_profile` to the model-explainer object. By default, profiles are calculated for all explanatory variables. To limit the calculation to selected variables, one can pass the names of the variables to the `variables`

argument. A plot of the calculated profiles can be obtained by applying the generic `plot()` function, as shown in the code below. The resulting profiles correspond to those shown in Figure 19.2.

```
explain_apartments_rf <- explain(model_apartments_rf,
                                 data = apartments,
                                 verbose = FALSE)

pd_rf <- variable_profile(explain_apartments_rf,
                          type = "partial",
                          variables = c("no.rooms", "surface"))
plot(pd_rf) +
  ggtitle("Partial dependence for surface and number of rooms")
```

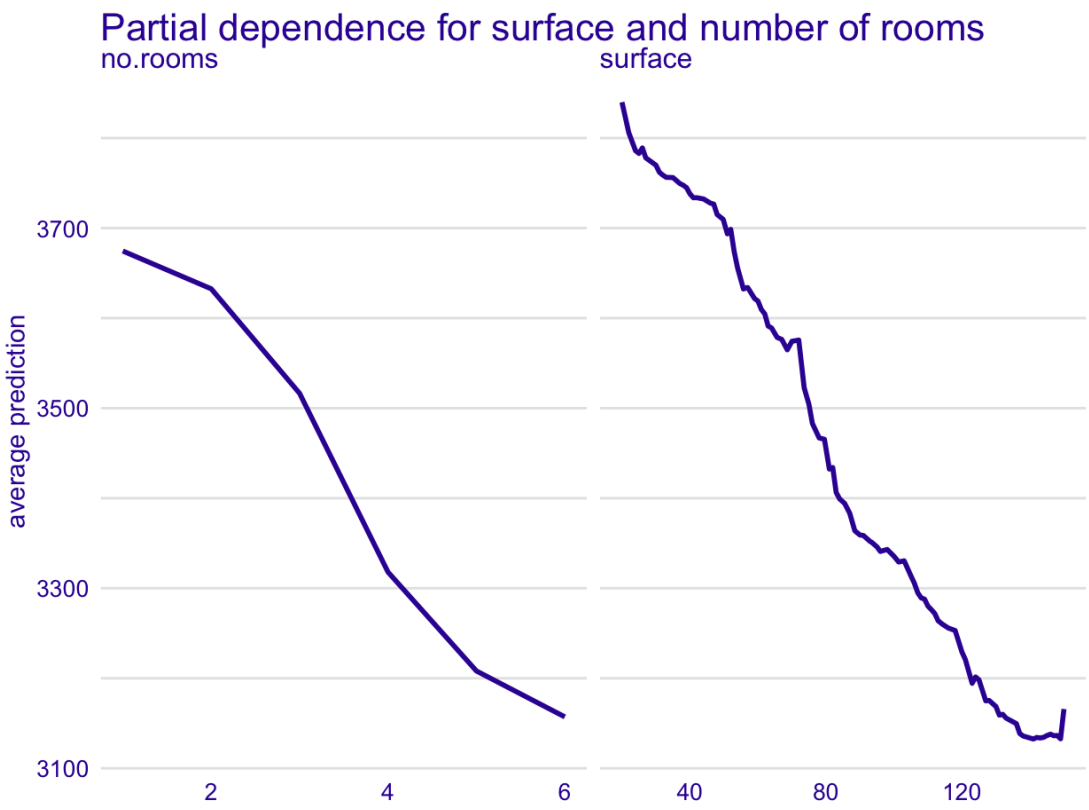



Figure 19.3: Partial Dependence profile for surface and number of rooms.

LD profiles are calculated by applying function `DALEX::variable_profile` with additional argument `type = "conditional"` .

```
cd_rf <- variable_profile(explain_apartments_rf,
                          type = "conditional",
                          variables = c("no.rooms", "surface"))
plot(cd_rf) +
  ggtitle("Local dependence for surface and number of rooms")
```

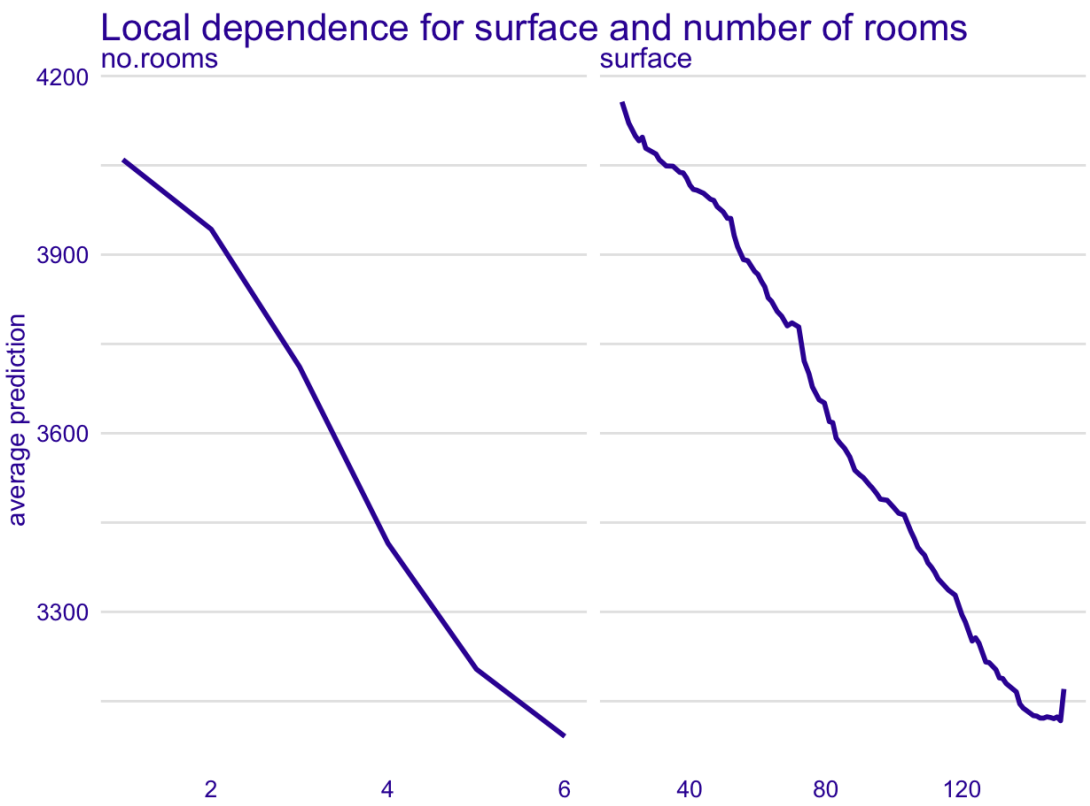



Figure 19.4: Local dependence profiles for the number of rooms and surface.

LD profiles are calculated by applying function `DALEX::variable_profile` with additional argument `type = "accumulated"` .

```
ac_rf <- variable_profile(explain_apartments_rf,
                          type = "accumulated",
                          variables = c("no.rooms", "surface"))
plot(ac_rf) +
  ggtitle("Accumulated local profiles for surface and number of rooms")
```

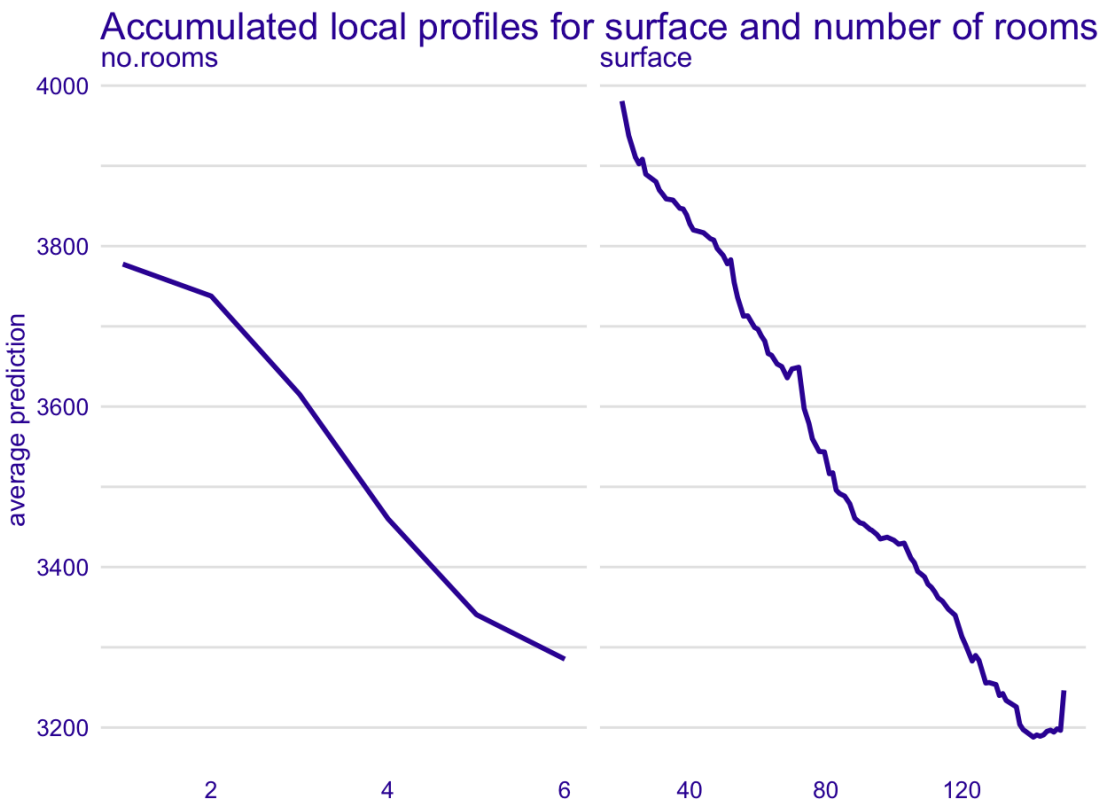



Figure 19.5: Accumulated local profiles for the number of rooms and surface.

One can also use the `plot()` function to juxtapose different types of profiles in a single plot.

```
pd_rf$agr_profiles$`_label_` = "Partial Dependence"
cd_rf$agr_profiles$`_label_` = "Local Dependence"
ac_rf$agr_profiles$`_label_` = "Accumulated Local"
plot(pd_rf$agr_profiles, cd_rf$agr_profiles, ac_rf$agr_profiles)
```

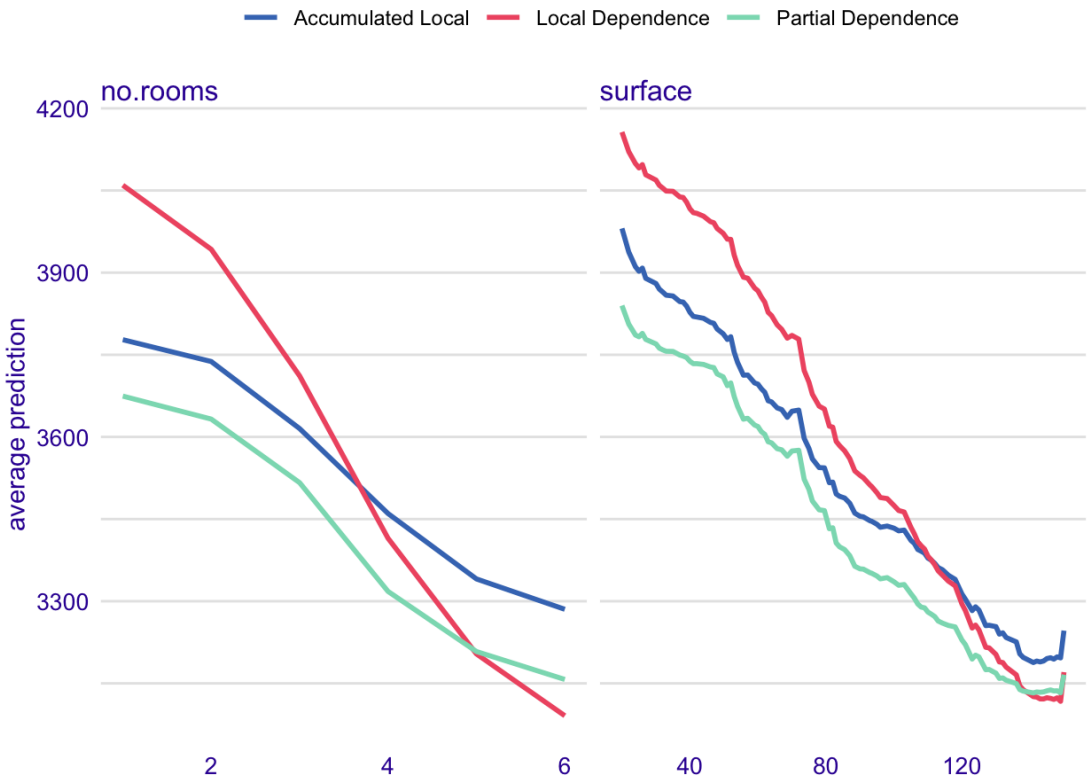



Figure 19.6: Different types of profiles for the number of rooms and surface.

# References

Apley, Dan. 2018. *ALEPlot: Accumulated Local Effects (Ale) Plots and Partial Dependence (Pd) Plots*. https://CRAN.R-project.org/package=ALEPlot.

Apley, Daniel W., and Jingyu Zhu. 2019. “Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models.” *CoRR* abs/1612.08468. http://arxiv.org/abs/1612.08468.

Biecek, Przemyslaw, Hubert Baniecki, Adam Izdebski, and Katarzyna Pekala. 2019. *ingredients: Effects and Importances of Model Ingredients*.

Molnar, Christoph, Bernd Bischl, and Giuseppe Casalicchio. 2018. “iml: An R package for Interpretable Machine Learning.” *Joss* 3 (26). Journal of Open Source Software: 786. https://doi.org/10.21105/joss.00786.

# 20 Residual Diagnostics

## 20.1 Introduction

In this chapter, we present methods that are useful for detailed examination of both overall and instance-specific model performance. In particular, we focus on graphical methods that use residuals. The methods may be used for several purposes:

- In the first part of the book, we discussed tools for single-instance examination. Residuals can be used to identify potentially problematic instances. The single-instance explainers can then be used in the problematic cases to understand, for instance, which factors contribute most to the errors in prediction.
- For most models, residuals should express a random behavior with certain properties (like, e.g., being concentrated around 0). If we find any systematic deviations from the expected behavior, they may signal an issue with a model (like, e.g., an omitted explanatory variable or wrong functional form of a variable included in the model).
- In Chapter 16 we discussed measures to evaluate the overall performance of a predictive model. Sometimes, however, we may be more interested in cases with the largest errors of prediction, which can be identified with the help of residuals.

Residual diagnostics is a classical topic related to statistical modeling. Literature on the topic is vast – essentially every book on statistical modeling includes some discussion about residuals. Thus, in this chapter, we are not aiming at being exhaustive. Rather, our goal is to present selected concepts that underlie the use of residuals.

## 20.2 Intuition

As we mentioned in Section 2.5, we primarily focus on models describing the expected value of the dependent as a function of explanatory variables. In such case, for a perfect predictive model, the predicted value of the dependent variable should be exactly equal to the actual value of the variable for every observation. Perfect prediction is rarely, if ever, expected. In practice, we want the predictions to be reasonably close to the actual values. This suggests

that we can use the difference between the predicted and the actual value of the dependent variable to quantify the quality of predictions obtained from a model. The difference is called a residual.

For a single observation, residual will almost always be different from zero. While a large (absolute) value of a residual may indicate a problem with a prediction for a particular observation, it does not mean that the quality of predictions obtained from a model is unsatisfactory in general. To evaluate the quality, we should investigate the ‘’behavior” of residuals for a group of observations. In other words, we should look at the distribution of the values of residuals.

For a ‘’good” model, residuals should deviate from zero randomly, i.e., not systematically. Thus, their distribution should be symmetric around zero, implying that their mean (or median) value should be zero. Also, residuals should be close to zero themselves, i.e., they should show low variability.

Usually, to verify these properties, graphical methods are used. For instance, a histogram of can be used to check the symmetry and location of the distribution of the residuals. For linear regression models, a plot of residuals against a continuous covariate can be checked for absence of any patterns that would suggest any systematic error in the predictions obtained for a specific range of the values of the covariate.

Note that a model may imply a concrete distribution for residuals. For instance, in the case of the classical linear regression model, standardized residuals should be normally distributed with mean zero and a constant variance. In such a case, a the distributional assumption can be verified by using a suitable graphical method like, for instance, a quantile-quantile plot. If the assumption is found to be violated, one might want to be careful regarding to predictions obtained from the model.

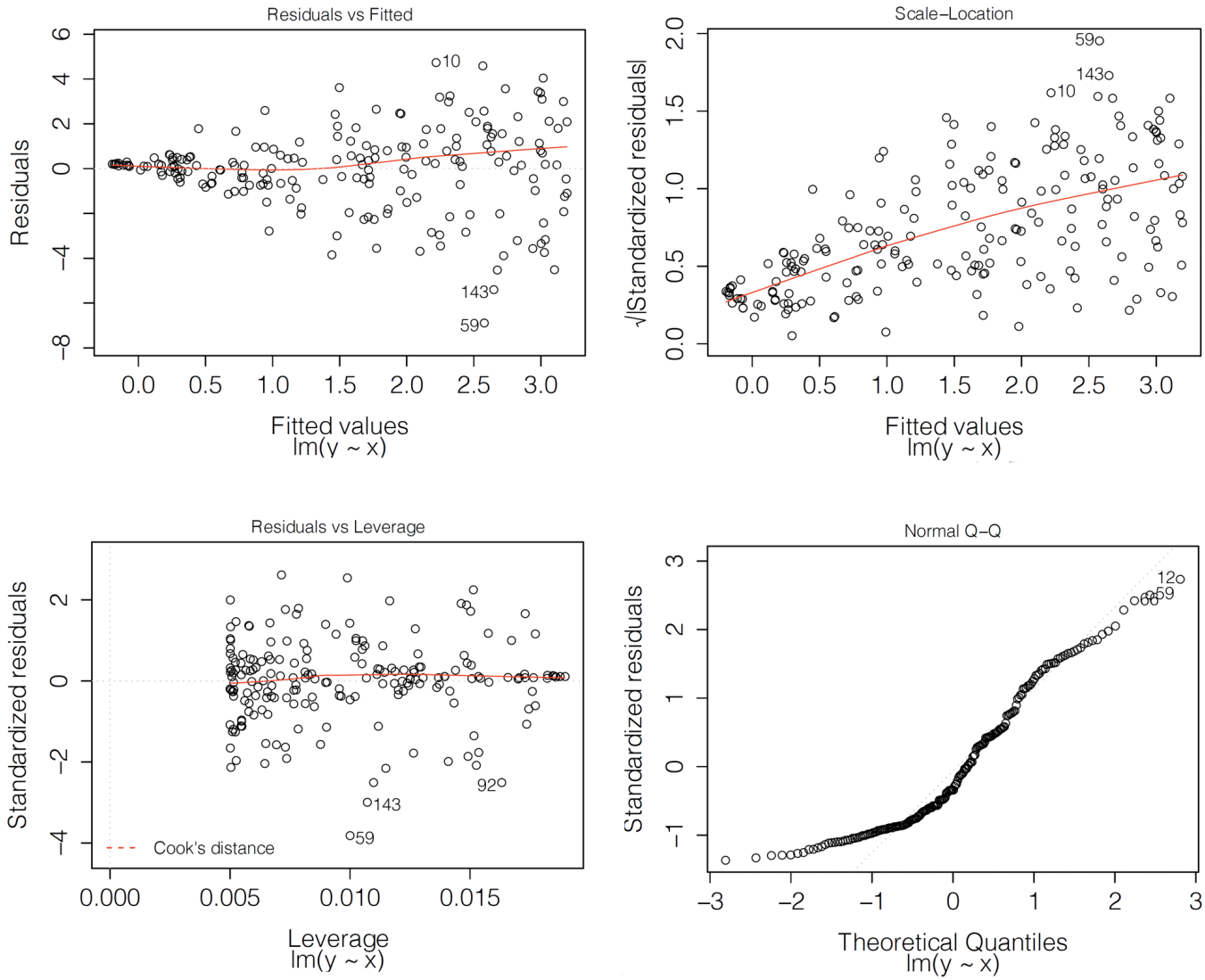



Figure 20.1: Diagnostic plots for linear models. Consecutive panels present residuals as a function of fitted values, standardized residuals as a function of fitted values, leverage plot and qq-plot.

## 20.3 Method

As it was already mentioned in Chapter 2, for a continuous dependent variable (or a count), residual $r_i$ for the $i$-th observation in a dataset is the difference between the model prediction and the corresponding value of the variable:

$$r_i = y_i - f(x_i). \tag{20.1}$$

A histogram of the estimated residuals can be used to check the symmetry and location of their distribution. An index plot of residuals, i.e., the plot of residuals against the corresponding observation number, may be used to identify observations with large residuals.

For diagnostic purposes the *standardized residuals* are defined as

$$\tilde{r}_i = \frac{r_i}{\sqrt{\mathrm{Var}(r_i)}}, \tag{20.2}$$

where $\mathrm{Var}(r_i)$ is the variance of the residual $r_i$. Of course, in practice, the variance of $r_i$ is usually unknown. Hence, in Equation (20.2), the estimated value of the variance is used. Residuals defined in this way are often called the *Pearson residuals*.

For a gaussian linear model the $\mathrm{Var}(r_i)$ can be estimated from design matrix and the distribution of $\tilde{r}_i$ is approximately standard normal. In general case, for complicated models, it is hard to estimate the variance $\mathrm{Var}(r_i)$ for a single instance so it is often approximated by a constant.

Definition (20.2) can also be applied to a binary dependent variable if the model prediction $f(x_i)$ is the probability observing $y_i$ and upon coding the two possible values of the variable as 0 and 1. However, in this case, the range of possible values of $r_i$ is restricted to $(-1, 1)$, which limits the usefulness of the residuals. For this reason, more often the Pearson residuals are used. Note that, if the values of the explanatory-variable vectors $x_i$ lead to different predicted values $f(x_i)$ for different observations in a dataset, the distribution of the Pearson residuals will not be approximated by the standard normal one. Nevertheless, the index plot may still be useful to detect observations with large residuals. The standard-normal approximation is more likely to apply in the situation when vectors $x_i$ split data into (a few, perhaps) groups sharing the same predicted value $f(x_i)$. In that case, one one can consider averaging residuals $r_i$ per group and standardizing them by $\sqrt{f(x_i)\{1 - f(x_i)/k\}}$, where $k$ is the number of observations in a particular group.

For categorical data, residuals can only be defined in terms of residuals of the binary variables indicating the category observed for the $i$-th observation.

## 20.4 Example: Apartments data

In this section, we use the linear-regression model `apartments_lm_v5` (Section 5.2.2) and the random-forest model `apartments_rf_v5` (Section 5.2.3) for the apartment-prices data (Section 5.2). Recall that the dependent variable of interest, the price per square-meter, is

continuous. Thus, we can use residuals $r_i$, as defined in equation (20.1). It is worth noting that, as it was mentioned in Section 16.4.1, RMSE for both models is very similar. Thus, overall, the two models could be seen as performing similarly.

Figures 20.2 and 20.3 summarize the distribution of residuals for both models. In particular, Figure 20.2 presents histogram of residuals, while Figure 20.3 shows box-and-whisker plots for the absolute value of the residuals.

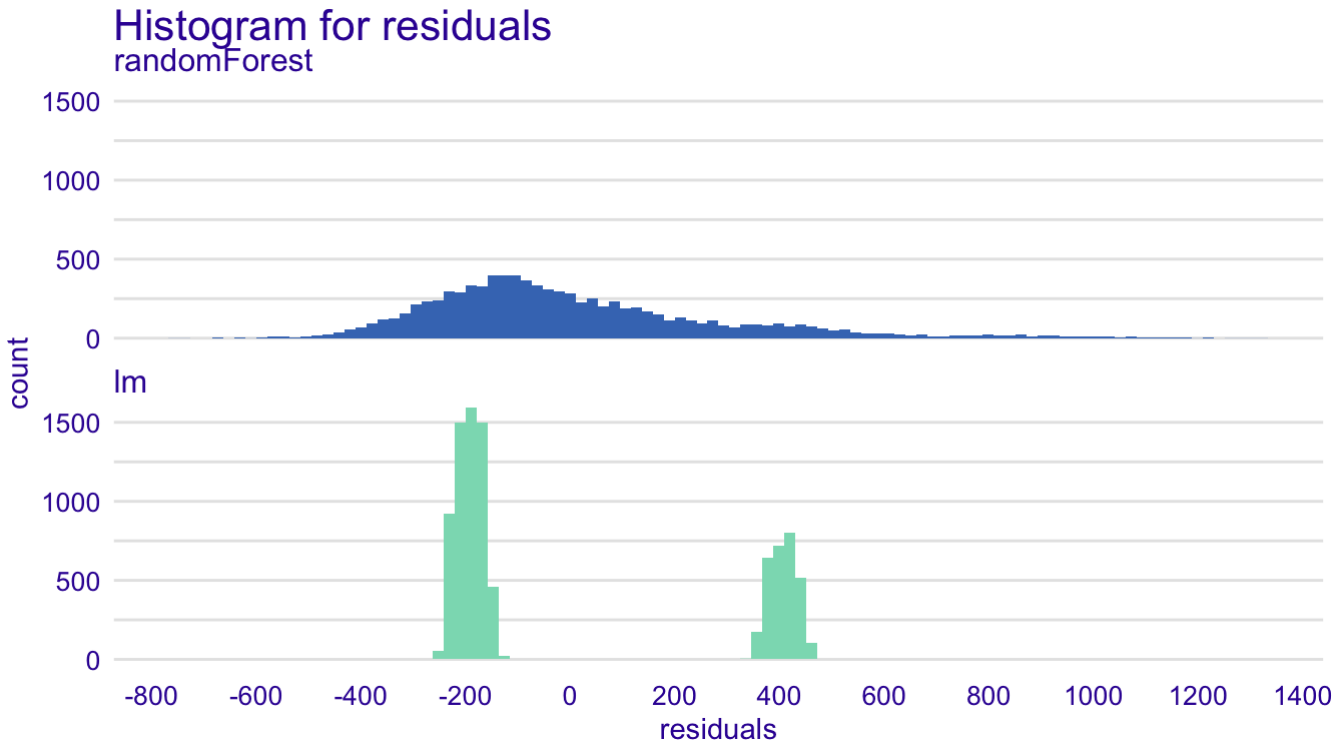



Figure 20.2: Histogram of residuals the linear-regression model `apartments_lm_v5` and the random-forest model `apartments_rf_v5` for the `apartments` data.

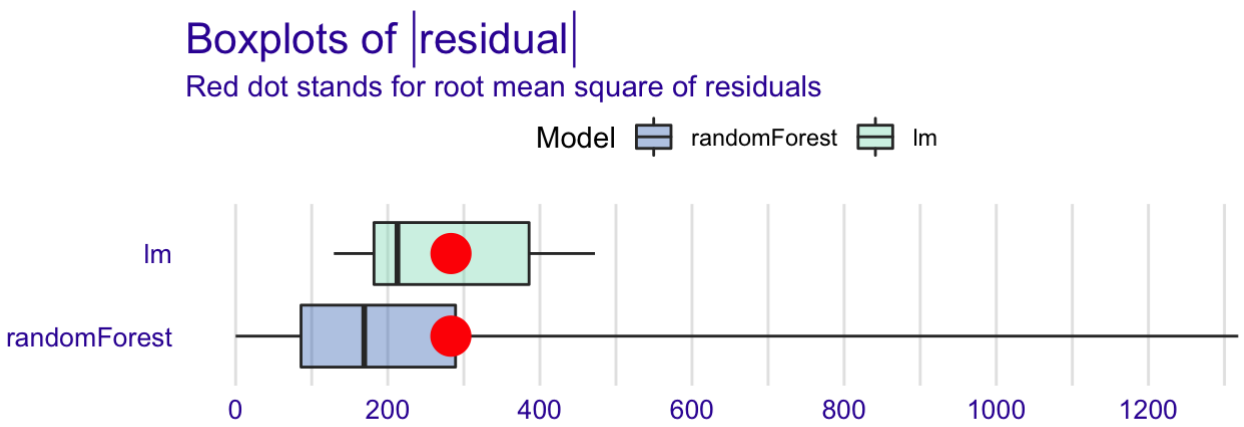


Figure 20.3: Box-and-whisker plots of the absolute values of the residuals of the linear-regression model `apartments_lm_v5` and the random-forest model `apartments_rf_v5` for the `apartments` data. The crosses indicate the average value that corresponds to RMSE.

Despite the similar value of RMSE, the distribution of residuals for both models is different. In particular, Figure 20.2 indicates that the distribution for the linear-regression model is, in fact, split into two separate, normal-like parts, which may suggest omission of a binary explanatory variable in the model. The two components are located around the values of about -200 and 400. As mentioned in the previous chapters, the reason for this behavior of the residuals is the fact that the model does not capture the non-linear relationship between the price and the year of construction.

As seen from Figure 20.2, the distribution for the random-forest model is skewed to the right and multimodal. It seems to be centered at a value closer to zero than the distribution for the linear-regression model, but it shows a larger variation. These conclusions are confirmed by the box-and-whisker plots in Figure 20.3.

The two plots suggest that the residuals for the random-forest model are more frequently smaller than the residuals for the linear-regression model. However, a fraction of the random-forest-model residuals are very large and these large residuals result in the RMSE being comparable for the two models.

In the remainder of the section, we focus on the random-forest model.

Figure 20.4 shows a scatterplot of residuals (y-axis) in function of the observed (x-axis) values of the dependent variable. For a perfect predictive model, we would expect the horizontal line at zero. For a ‘’good” model, we would like to see a symmetric scatter of points around the horizontal line at zero, indicating random deviations of predictions from the observed values. The plot in Figure 20.4 shows that, for the large observed values of the dependent variable, the residuals are positive, while for small values they are negative. Thus, the plot suggests that the predictions are shifted (biased) towards the average.

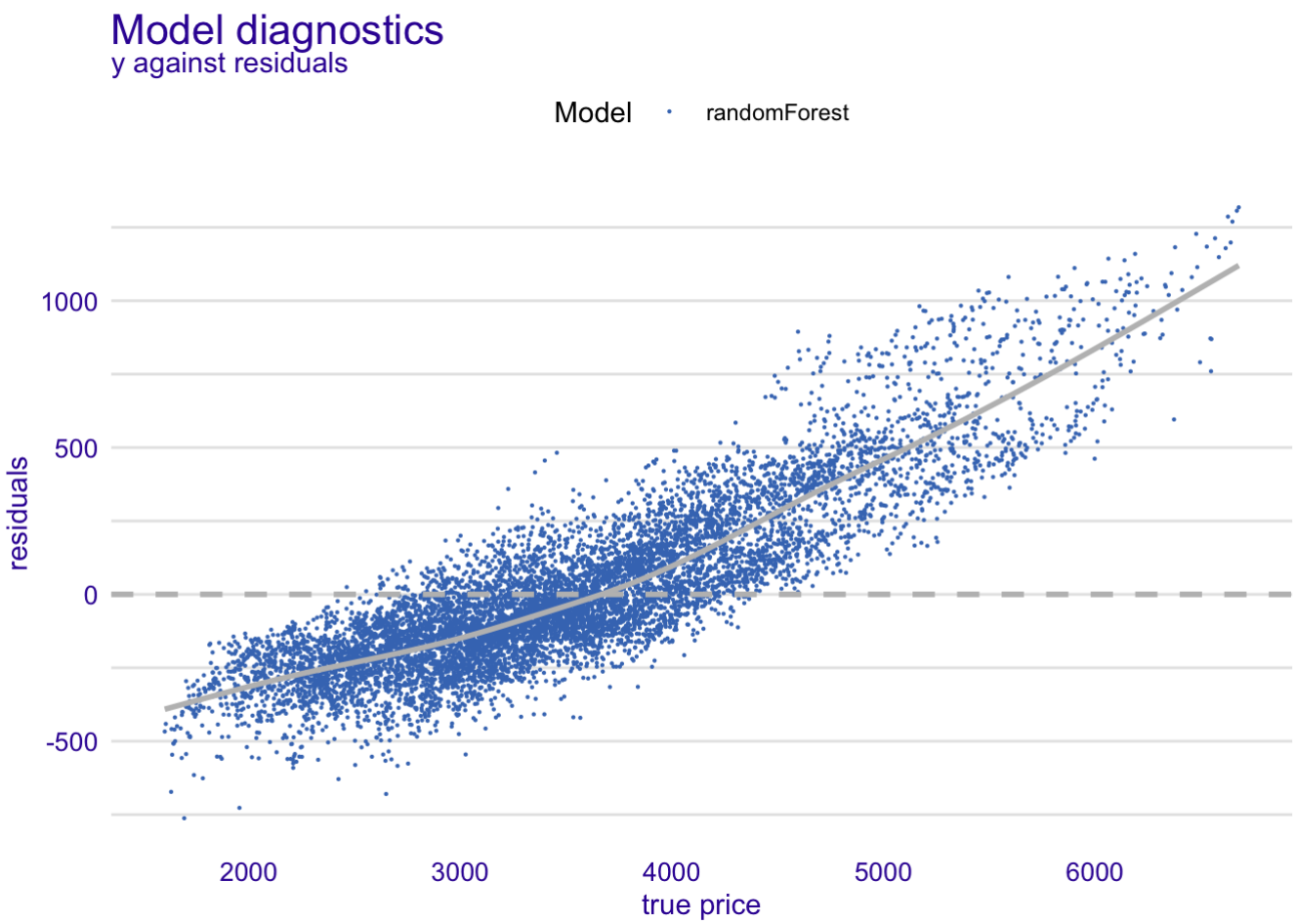



Figure 20.4: Residuals and observed values of the dependent variable for the random-forest model `apartments_rf_v5` for the `apartments` data.

The shift towards the average can also be seen from Figure 20.5 that shows a scatterplot of the predicted (y-axis) and observed (x-axis) values of the dependent variable. For a perfect predictive model we would expect a diagonal line (marked in red). The plot shows that, for large observed values of the dependent variable, the predictions are smaller than the observed values, with an opposite trend for the small observed values of the dependent variable.

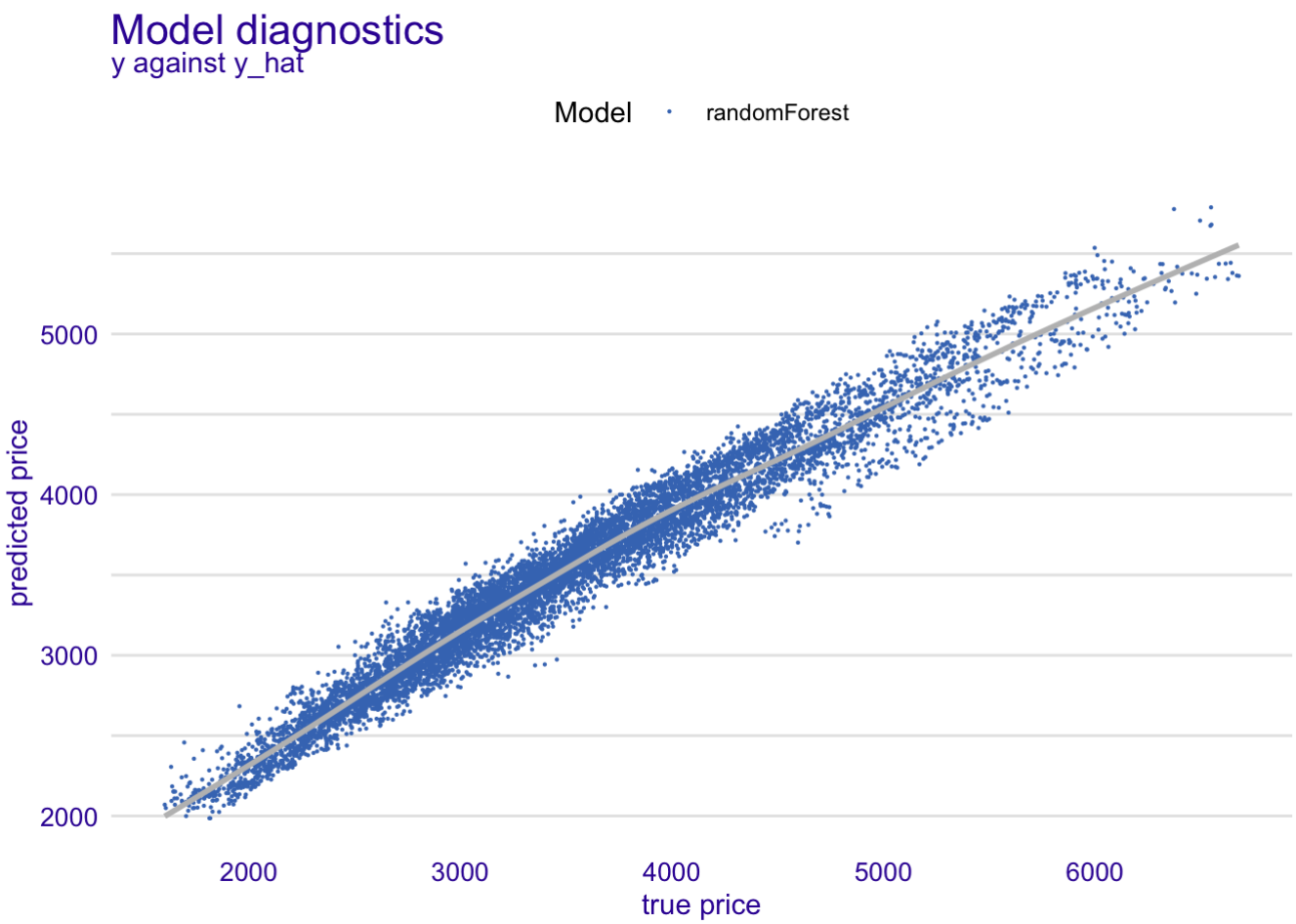



Figure 20.5: Predicted and observed values of the dependent variable for the random-forest model `apartments_rf_v5` for the `apartments` data. Red line is the diagonal.

Figure 20.6 shows an index plot of residuals, i.e., their scatterplot in function of an (arbitrary) id-number of the observation (x-axis). The plot indicates an asymmetric distribution of residuals around zero, as there is an excess of large positive (larger than 500) residuals without a corresponding fraction of negative values. This can be linked to the right-skewed distribution seen in Figures 20.2 and 20.3 for the random-forest model.

```
plot(md_rf, variable = "ids", yvariable = "residuals") +
    xlab("observation id") + ylab("residuals")
```

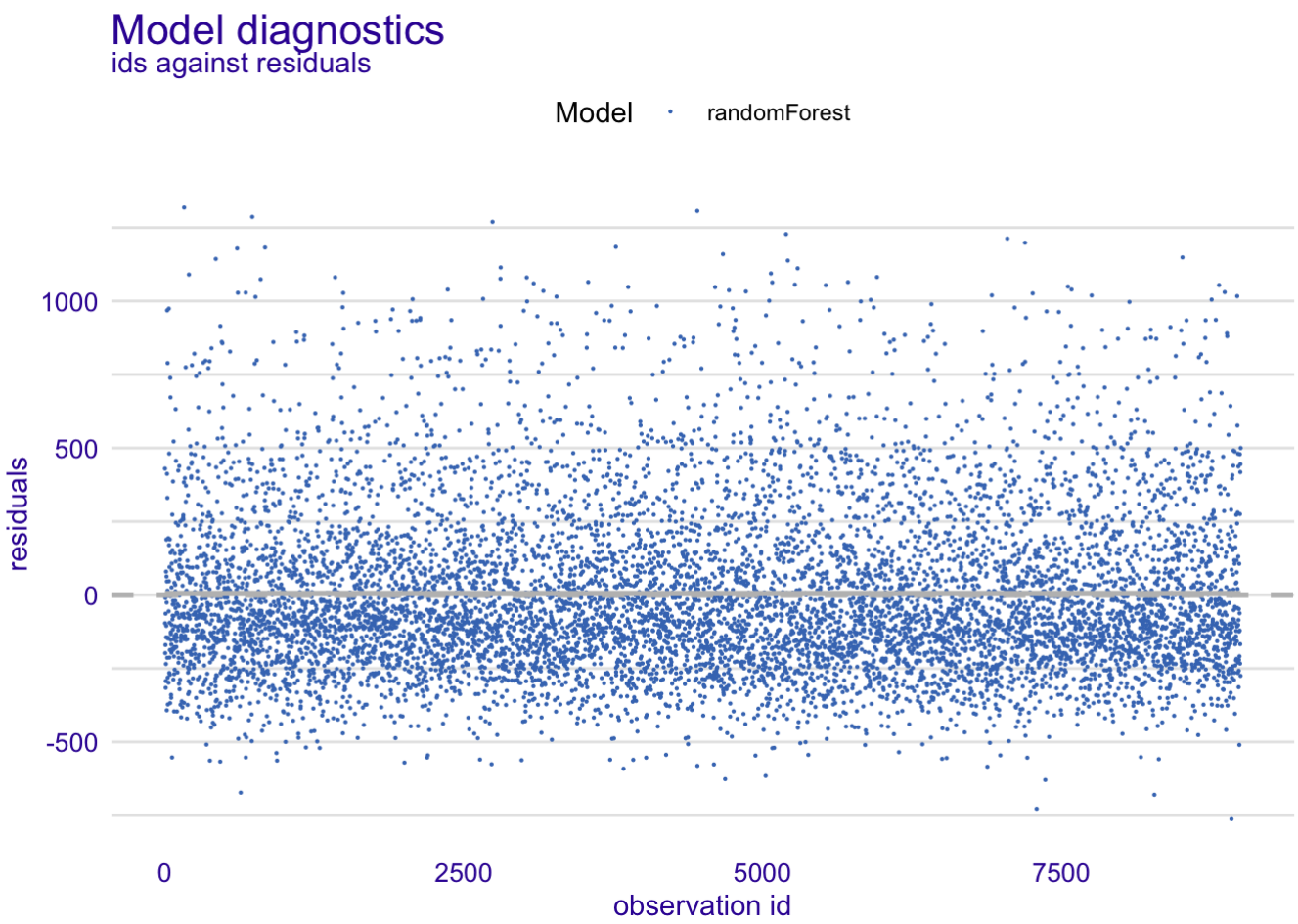



Figure 20.6: An index plot of residuals for the random-forest model `apartments_rf_v5` for the `apartments` data.

Figure 20.7 shows a scatterplot of residuals (y-axis) in function of the predicted (x-axis) value of the dependent variable. For a ‘’good” model, we would like to see a symmetric scatter of points around the horizontal line at zero. The plot in Figure 20.7, as the one in Figure 20.4, the plot suggests that the predictions are shifted (biased) towards the average.

```
plot(md_rf, variable = "y_hat", yvariable = "residuals") +
    xlab("predicted price")
```

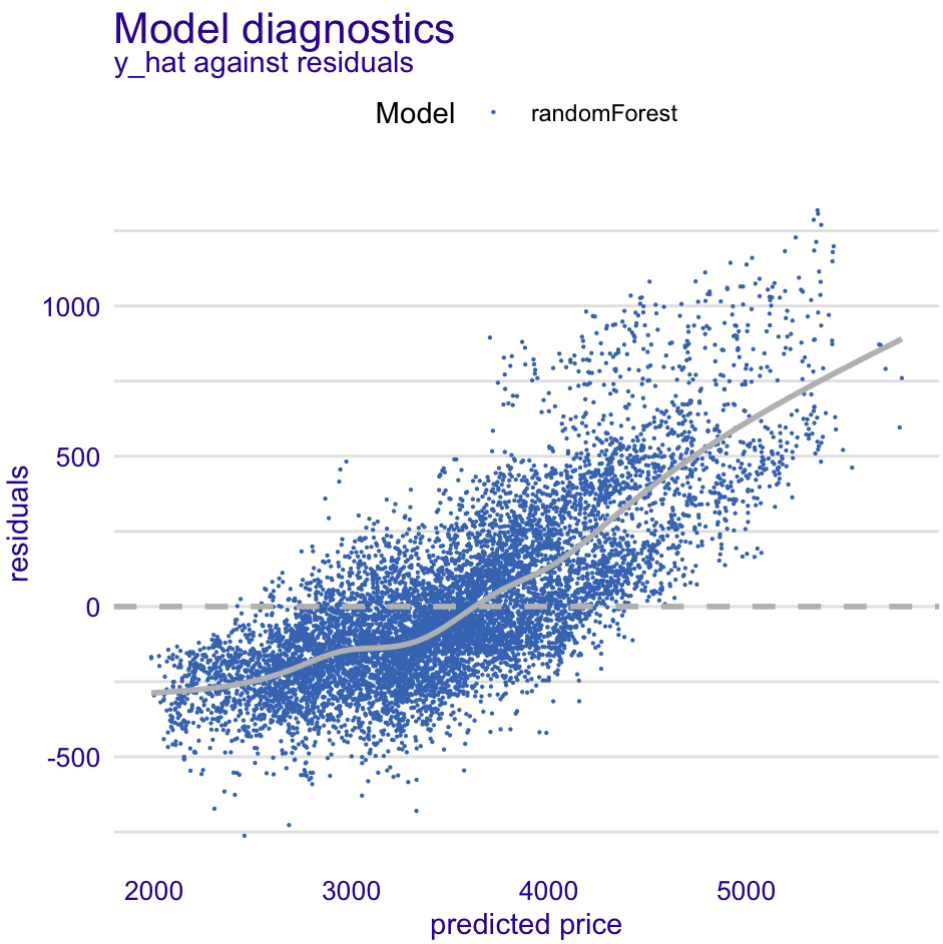



Figure 20.7: Residuals and predicted values of the dependent variable for the random-forest model `apartments_rf_v5` for the `apartments` data.

The random-forest model, as the linear-regression model, assumes that residuals should be homoscedastic, i.e., that they should have a constant variance. Figure 20.8 presents the scale-location plot of residuals, i.e., a scatterplot of the absolute value of residuals in function of the predicted values of the dependent variable. The plot includes a smoothed line capturing the average trend. For homoscedastic residuals, we would expect a symmetric scatter around a horizontal line, for which the smoothed trend should be also horizontal. The plot in Figure 20.8 deviates from the expected pattern and indicates that the variability of the residuals depends on the (predicted) value of the dependent variable.

```
plot(md_rf, variable = "y_hat", yvariable = "abs_residuals") +
    xlab("predicted price") + ylab("|residuals|")
```

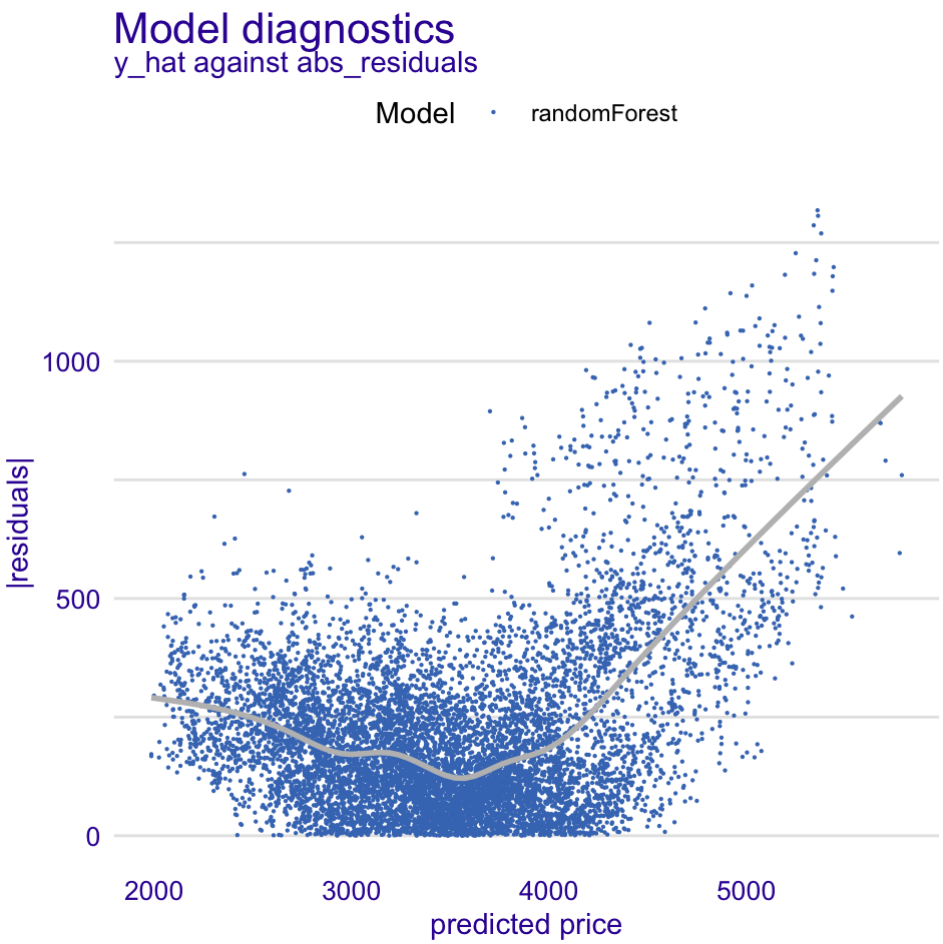



Figure 20.8: The scale-location plot of residuals for the random-forest model `apartments_rf_v5` for the `apartments` data. The square-roots of the absolute values of standardized residuals (y-axis) are plotted in function of the predicted values of the dependent variable (x-axis).

## 20.5 Pros and cons

Diagnostics of the residuals is a very important stage of model exploration. Properly performed diagnostics allows to identify many different types of problems such as:

- Bias in predictions for instances with extremely high values of the target variable.
- The heterogeneous variance of the residuals, suggesting perhaps incorrect specification of the model.
- High values of residual for some ranges of a variable suggesting an incorrect model specification for some subgroup of observations.

However, the problem with diagnostics is that there is lots of diagnostic charts to review. And without quantitative measures of meeting assumption, we can only rely on the organoleptic review of the graph after the graph.

## 20.6 Code snippets for R

In this section, we present the key features of the `DALEX` package which is a wrapper for functions from `auditor` package (Gosiewska and Biecek 2018). This package covers all methods presented in this chapter.

First, we load explainers for the linear-regression model `apartments_lm_v5` and the random-forest model `apartments_rf_v5` created in Section 5.2.6 for the `apartments` data.

```
library("DALEX")
library("randomForest")

explainer_apartments_lr <- archivist:: aread("pbiecek/models/f49ea")
explainer_apartments_rf <- archivist:: aread("pbiecek/models/569b0")
```

There are two functions that will be used for exploration of residuals. The `DALEX::model_performance()` function is useful for exploration of distribution of residuals while `DALEX::model_diagnostics()` function is useful for looking for relation between residuals and other variables.

Let's start with distributions of residuals. This can be done with the `model_performance()` function. The residuals are stored in separate objects that can be used for construction of various plots and summaries.

```
mr_lr <- DALEX::model_performance(explainer_apartments_lr)
mr_rf <- DALEX::model_performance(explainer_apartments_rf)
```

The generic `plot()` function shows different statistics based on specified `geom` argument. In particular, Figure 20.2 can be constructed by using the following simple code:

```
plot(mr_lr, mr_rf, geom = "histogram")
```

Note that, by including the two objects containing residuals for the linear-regression model and the random-forest model in the function call, we automatically get an overlay of the plots of the histogram of residuals for the two models.

The box-and-whisker plots of the residuals for the two models, shown in Figure 20.3, can be constructed by using the following simple call with `geom = "boxplot"` .

```
plot(mr_lr, mr_rf, geom = "boxplot")
```

Function `model_diagnostics()` calculates residuals for various scatter plots of residuals against some other variables

```
md_lr <- model_diagnostics(explainer_apartments_lr)
md_rf <- model_diagnostics(explainer_apartments_rf)
```

The generic `plot()` function produces a scatterplot of residuals (y-axis) in function of the observed (x-axis) values of the dependent variable, as in Figure 20.4. By using arguments `variable` and `yvariable` , one specify which variables will be plotted on OX and OY axis. Apart of variables names, one can use following constants:

- `y` for true target values,
- `y_hat` for predicted target values,
- `obs` for ids of an observation,
- `residuals` for calculated residual,
- `abs_residuals` for absolute values of residual.

For example, to reproduce Figure 20.4 one needs to plot target variable on the OX axis and residuals on OY.

```
plot(md_rf, variable = "y", yvariable = "residuals")
```

To produce Figure 20.5 we need to plot predicted target values on OY axis. This can be done with `yvariable = "y_hat"` argument.

```
plot(md_rf, variable = "y", yvariable = "y_hat")
```

The Figure 20.6 has indexes of observations on OX axis. This can be achieved with `variable = "ids"` argument.

```
plot(md_rf, variable = "ids", yvariable = "residuals")
```

In the Figure 20.8 on OY scale we plotted absolute residuals. This can be done with `yvariable = "abs_residuals"` argument.

```
plot(md_rf, variable = "y_hat", yvariable = "abs_residuals")
```

## References

Gosiewska, Alicja, and Przemyslaw Biecek. 2018. *Auditor: Model Audit - Verification, Validation, and Error Analysis*. https://CRAN.R-project.org/package=auditor.

# 21 FIFA 19

In previous chapters we introduced a number of methods for instance level exploration of predictive models. In consecutive chapter we showed how to use Ceteris Paribus profiles, SHAP values, LIME or Break Down plots for models created on the dataset `titanic`. These examples we introduced and discussed separately as each of them was focused on a single method described in a given chapter.

In this chapter we present an example of full process for model development along the process introduces in Chapter 2. We will use a new dataset for FIFA 19 soccer game. Based on it we will tour through the process of data preparation, model assembly and model understanding. In each phase we show how to combine results from different methods of exploration.

The main goal of this chapter is to show how different techniques complement each other. Some phases, like data preparation, are simplified in order to leave space for the method for visual exploration and explanation of predictive models.

## 21.1 Introduction

The story is following. The `https://sofifa.com/` portal is a reliable website for FIFA ratings of football players. Data from this website was scrapped and make available at the Kaggle webpage `https://www.kaggle.com/karangadiya/fifa19`.

We will use this data to build a predictive model for assessment of player value. Once the model will be created we will use methods for exploration and explanation to better understand how it is working and also to better understand which factors and how influence the player value.

## 21.2 Data preparation

The scraped data contains 89 columns, and various information about players along with photo, club, nationality and others. Here we will focus on 40 players statistics and the way how they influence model predictions.

The data set contains statistics for 16924 players. First, let's see distribution of selected variables from this dataset.

Players values are heavily skewed. Half of players have estimated value between 0.3 and 2.2 millions of Euro. But few players have estimated values higher than 100 millions of Euro. Figure 21.1 presents empirical cumulative distribution function and histogram with log transformation of the OX axis.

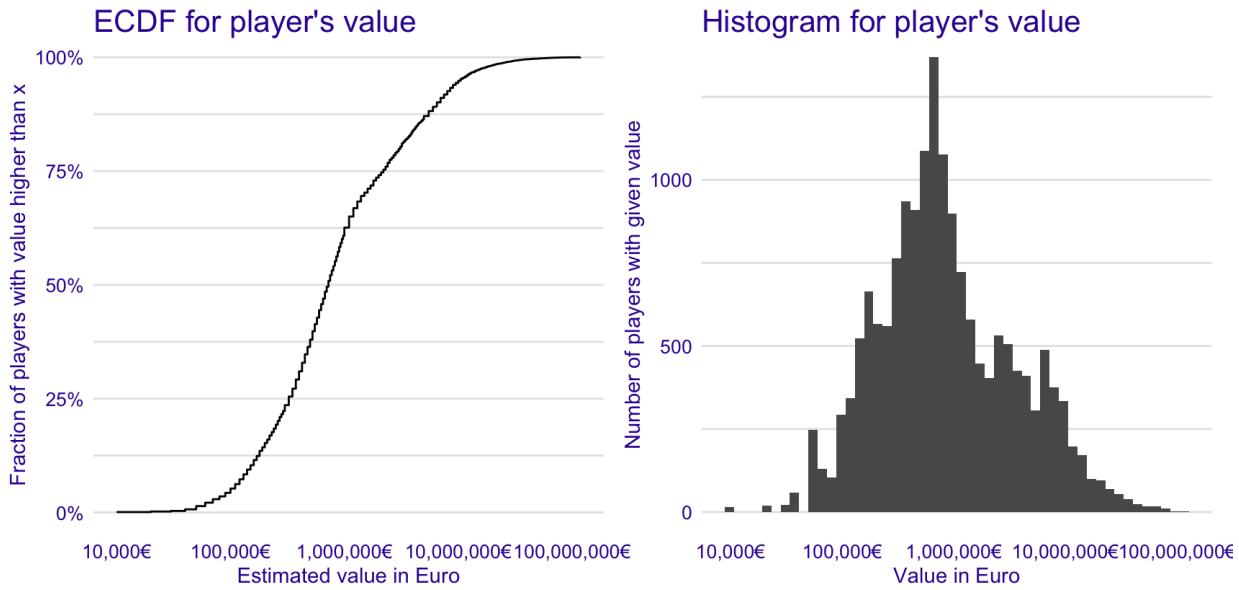



Figure 21.1: Empirical cumulative distribution function and histogram for values of players. The OX axis is in the log10 transformation.

Due to a large number of player characteristics we are not going to explore all of them but rather we will focus on four that will be discussed later in this chapter, namely: `Age`, `Reactions`, `BallControl` and `ShortPassing`.

Figure 21.2 presents distributions for these variables. For `Age` we see that most players are between 20 and 30 years old. What is interesting in `BallControl` and `ShortPassing` is that they have bimodal distribution. The reason for that is that these characteristics are very low for goalkeepers but higher for other players. The variable `Reactions` has Gaussian shaped distribution with average 62 and standard deviation 9.

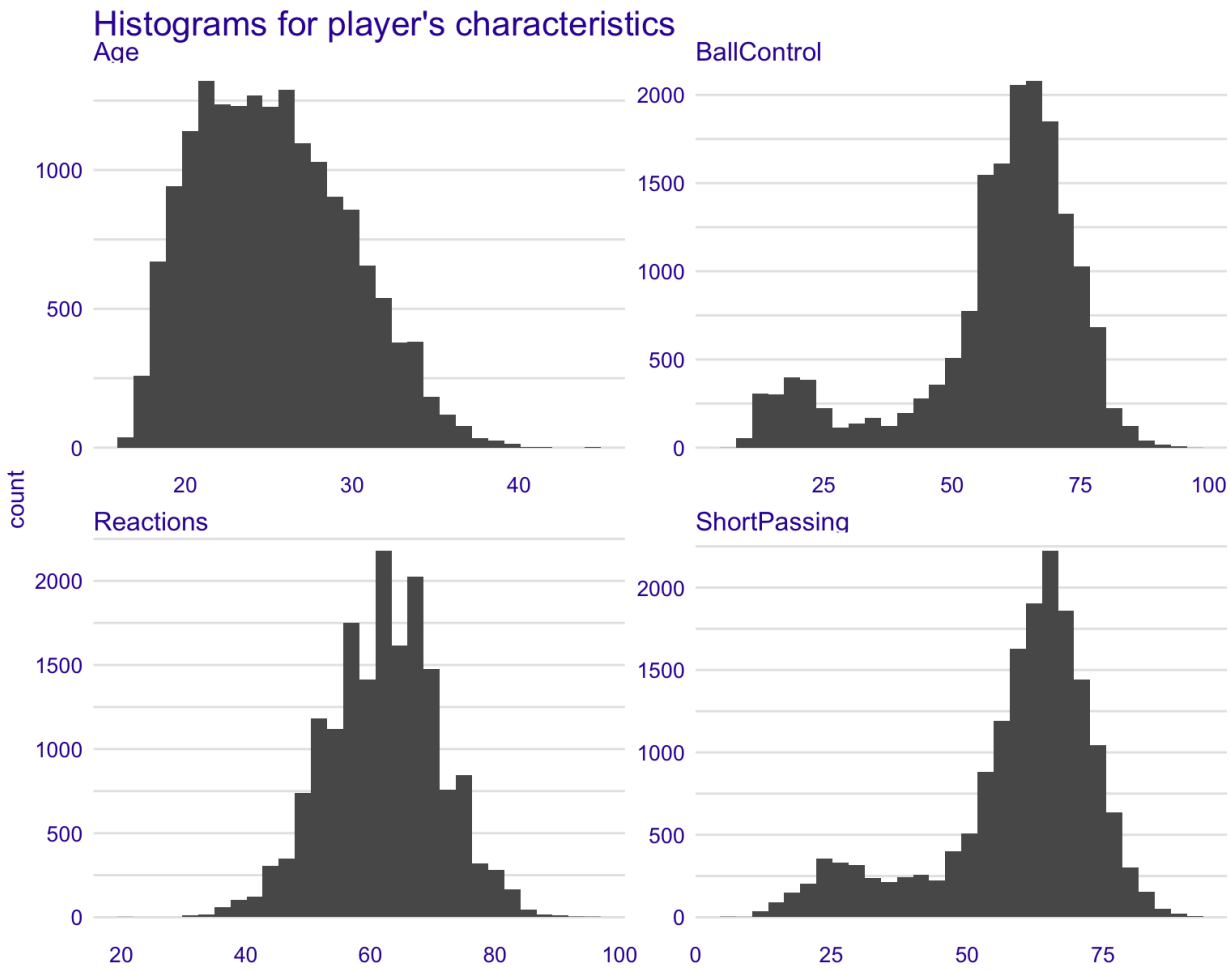



Figure 21.2: Histograms for selected characteristics of players. Note that `BallControl` and `ShortPassing` have bimodal distributions.

## 21.3 Data understanding

Time to see how these variables are linked with player's value. Figure 21.3 shows scatterplots for selected four characteristics. Because of the skewness of player's value the OY value is presented after log transformation.

For `Age` it looks like the relation is not monotonic, there is some optimal age in which players value is the highest, between 24 and 28 years. Value of youngest players are on average 10x lower. Same with olders players.

For variables `BallControl` and `ShortPassing` the relation is not monotonic. In general the larger value of these coefficients the higher value of a player and most expensive are players with top characteristics. But among players with very low scores in `BallControl` and

`ShortPassing` some are very expensive too. As it was suggested earlier, these players are probably goalkeepers.

For variable `Reactions` the link with player's value is monotonic. As expected, the higher `Reactions` the higher player's value.

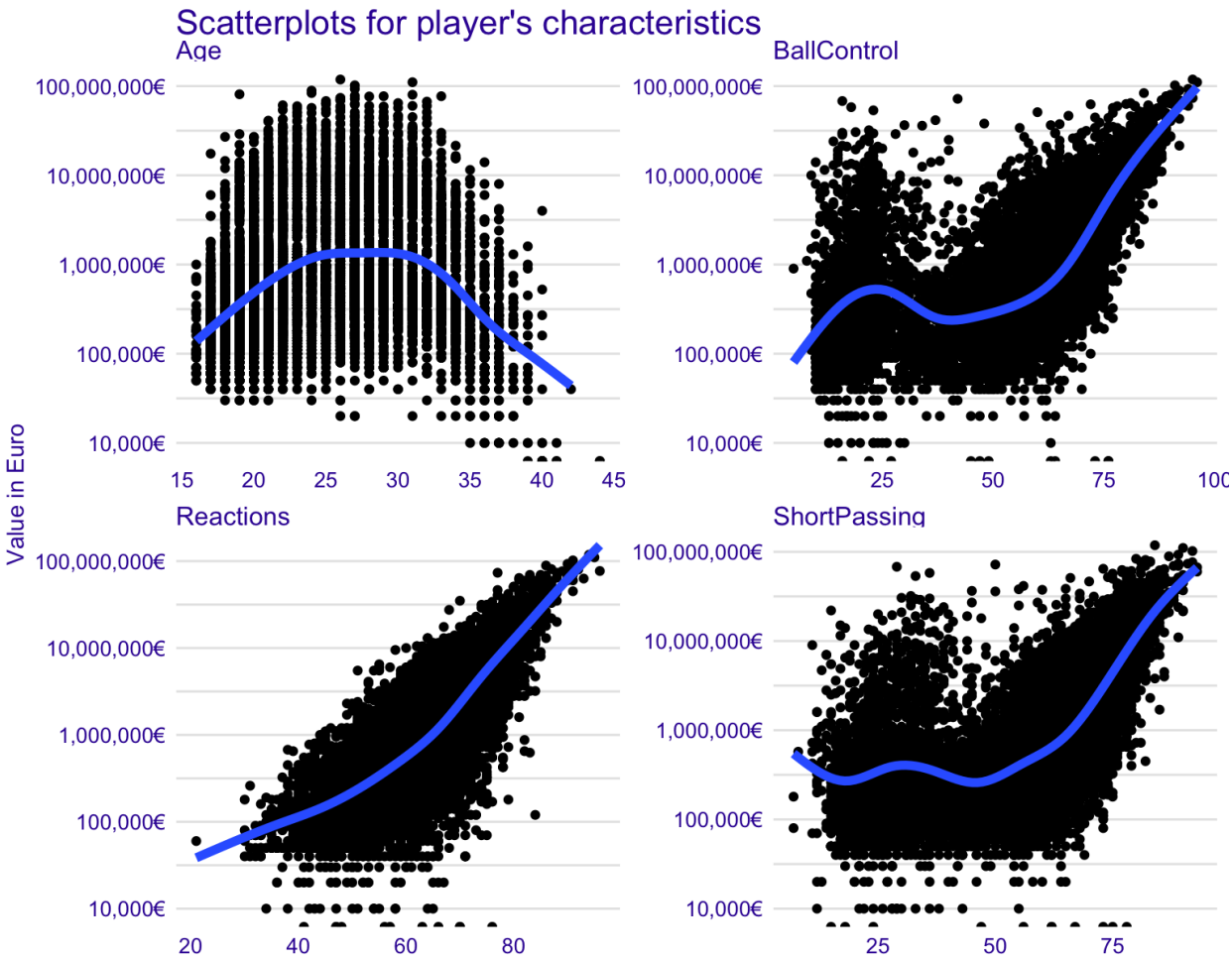



Figure 21.3: Scatterplot for relation between selected four players characteristics and values of players.

Figure 21.4 shows pairwise scatterplots for dependent variables. Three observations are clear from these scatterplots. One is that `Age` has positive correlation with other variables. On average older players have higher skills. Second is that skills are positively correlated, the correlation between `BallControl` and `ShortPassing` is higher than 0.9. Third is that goalkeepers' characteristics are different than rest of players.

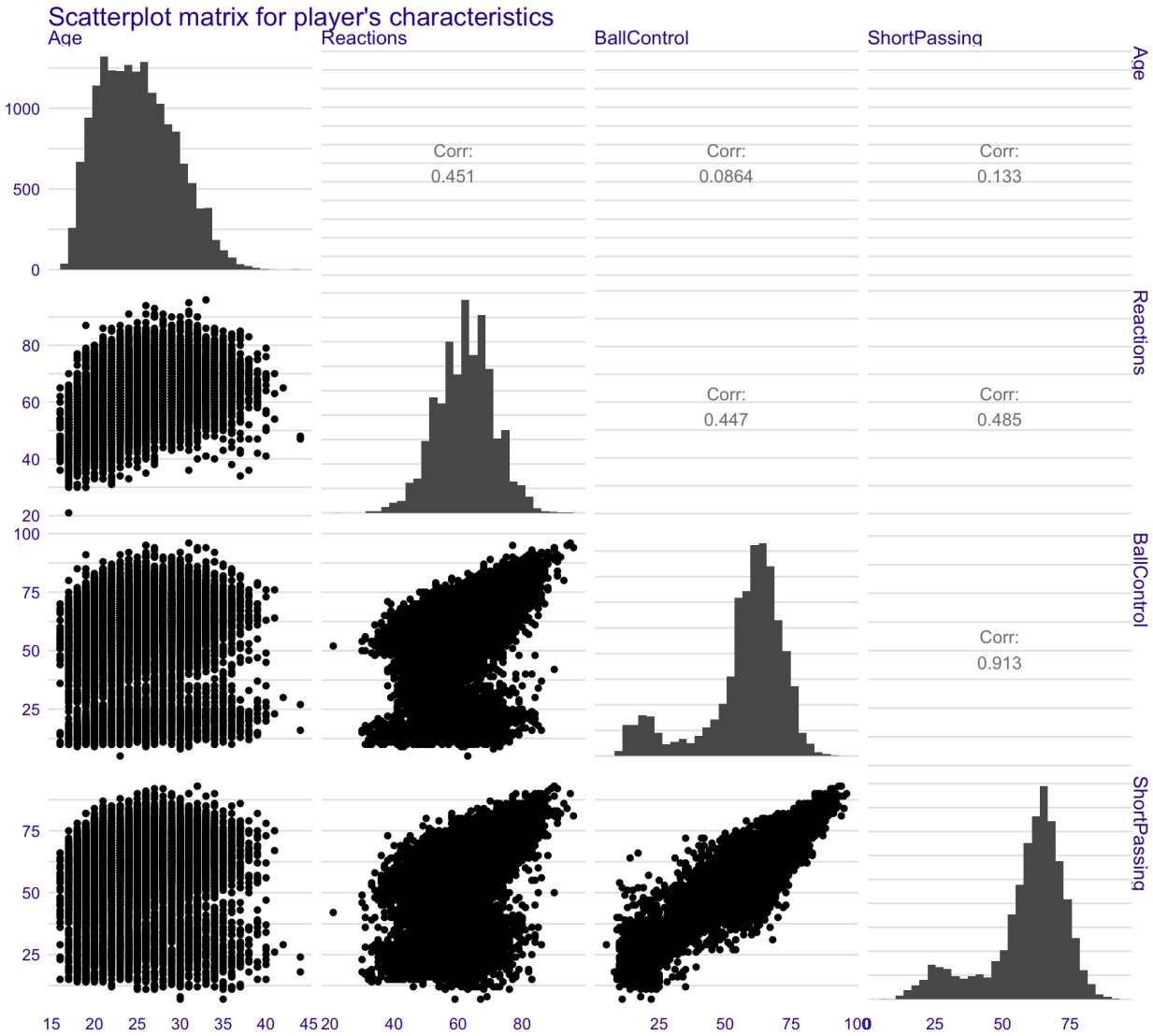



Figure 21.4: Scatterplot for relation between selected four players characteristics and values of players.

Let's compare results from this data exploration with exploration of predictive models that will be fitted on this data.

# 21.4 Model assembly

Time to build a predictive model for players' value based on selected characteristics. We will use a trained elastic model to explore the relation between players' characteristics and players' values.

Having clean data then model assembly is easy. For FIFA 19 data we will try four models with different structures that are able to catch different types of relations. One model would be enough, but we will try four different models to see if they catch similar relations.

Considered models are:

- boosting model with 250 trees 1 level depth as implemented in package `gbm` (Ridgeway 2017),
- boosting model with 250 trees 4 levels depth, this model shall be able to catch interactions between features,
- linear model with spline transformation of dependent variables implemented in package `rms` (Harrell Jr 2018),
- random forest model with 250 trees as implemented in package `ranger` (Wright and Ziegler 2017).

```
library("gbm")
fifa_gbm_shallow <- gbm(LogValue~., data = fifa19small, n.trees = 250,
        interaction.depth = 1, distribution = "gaussian")

fifa_gbm_deep <- gbm(LogValue~., data = fifa19small, n.trees = 250,
        interaction.depth = 4, distribution = "gaussian")

library("ranger")
fifa_rf <- ranger(LogValue~., data = fifa19small, num.trees = 250)

library("rms")
fifa_ols <- ols(LogValue ~ rcs(Age) + rcs(International.Reputation) +
        rcs(Skill.Moves) + rcs(Crossing) + rcs(Finishing) +
        rcs(HeadingAccuracy) + rcs(ShortPassing) + rcs(Volleys) +
        rcs(Dribbling) + rcs(Curve) + rcs(FKAccuracy) +
        rcs(LongPassing) + rcs(BallControl) + rcs(Acceleration) +
        rcs(SprintSpeed) + rcs(Agility) + rcs(Reactions) +
        rcs(Balance) + rcs(ShotPower) + rcs(Jumping) + rcs(Stamina) +
        rcs(Strength) + rcs(LongShots) + rcs(Aggression) +
        rcs(Interceptions) + rcs(Positioning) + rcs(Vision) +
        rcs(Penalties) + rcs(Composure) + rcs(Marking) +
        rcs(StandingTackle) + rcs(SlidingTackle) + rcs(GKDiving) +
        rcs(GKHandling) + rcs(GKKicking) + rcs(GKPositioning) +
        rcs(GKReflexes), data = fifa19small)
```

Before we can explore model behavior we need to create explainers with the `DALEX::explain` function. These explainers will be later used to assess model performance.

Note that models are trained on logarithm of the value, but it will be much more natural to operate on values in Euro. This is why in explainers we specified a user defined predict function that transforms log value to the value in Euro.

Each explainer got also a unique `label` and corresponding `data` and `y` arguments.

```
library("DALEX")
fifa_gbm_exp_deep <- explain(fifa_gbm_deep,
        data = fifa19small, y = 10^fifa19small$LogValue,
        predict_function = function(m,x) 10^predict(m, x, n.trees = 250),
        label = "GBM deep")

fifa_gbm_exp_shallow <- explain(fifa_gbm_shallow,
        data = fifa19small, y = 10^fifa19small$LogValue,
        predict_function = function(m,x) 10^predict(m, x, n.trees = 250),
        label = "GBM shallow")

fifa_rf_exp <- explain(fifa_rf,
        data = fifa19small, y = 10^fifa19small$LogValue,
        predict_function = function(m,x) 10^predict(m, x)$predictions,
        label = "RF")

fifa_rms_exp <- explain(fifa_ols,
        data = fifa19small, y = 10^fifa19small$LogValue,
        predict_function = function(m,x) 10^predict(m, x),
        label = "RMS")
```

## 21.5 Model audit

We have created four models. Let's see which model is better. Figure 21.5 compares distributions of absolute model residuals. Crosses corresponds to average, which correspond to RMSE. On average, smallest residuals are for the Random Forest model.

```r
library("DALEX")
(fifa_mr_gbm_shallow <- model_performance(fifa_gbm_exp_shallow))
```

```
## Measures for:  regression
## mse      : 8.793857e+12
## rmse     : 2965444
## r2       : 0.7359532
## mad      : 184494.3
##
## Residuals:
##            0%           10%           20%           30%           40%
## -36235118.885   -551904.801   -190134.326    -91880.898    -41545.399
##           50%           60%           70%           80%           90%
##      7545.811     78107.824    179592.760    385908.869   1242757.255
##          100%
##  84138902.525
```

```r
(fifa_mr_gbm_deep <- model_performance(fifa_gbm_exp_deep))
```

```
## Measures for:  regression
## mse      : 2.258819e+12
## rmse     : 1502937
## r2       : 0.9321761
## mad      : 118742.1
##
## Residuals:
##           0%          10%          20%          30%          40%
## -19116967.31   -436856.06   -148437.56    -57458.30    -18811.11
##          50%          60%          70%          80%          90%
##     10420.76     46392.59    100655.11    217644.71    714571.14
##         100%
##  41410615.28
```

```r
(fifa_mr_gbm_rf <- model_performance(fifa_rf_exp))
```

```
## Measures for:  regression
## mse      : 1.127823e+12
## rmse     : 1061990
## r2       : 0.9661357
## mad      : 51320.23
##
## Residuals:
##           0%          10%          20%          30%          40%
## -4610822.779   -97148.598   -41372.043   -20398.749    -6968.849
##          50%          60%          70%          80%          90%
##     5584.176    25714.410    65998.232   181294.901   589440.628
##         100%
## 29046254.806
```

```
(fifa_mr_gbm_rms <- model_performance(fifa_rms_exp))
```

```
## Measures for:  regression
## mse      : 2.191297e+13
## rmse     : 4681129
## r2       : 0.342035
## mad      : 148187.1
##
## Residuals:
##            0%           10%           20%           30%           40%
## -2.722515e+08 -4.983836e+05 -1.746193e+05 -7.411303e+04 -2.777345e+04
##           50%           60%           70%           80%           90%
##  7.872114e+03  5.199682e+04  1.301219e+05  3.007244e+05  1.071466e+06
##          100%
##  6.091332e+07
```

```
plot(fifa_mr_gbm_shallow, fifa_mr_gbm_deep, fifa_mr_gbm_rf, fifa_mr_gbm_rms, geom = "box
  scale_y_continuous("Absolute residuals in Euro", trans = "log10", labels = dollar_form
  ggtitle("Distributions of model absolute residuals")
```

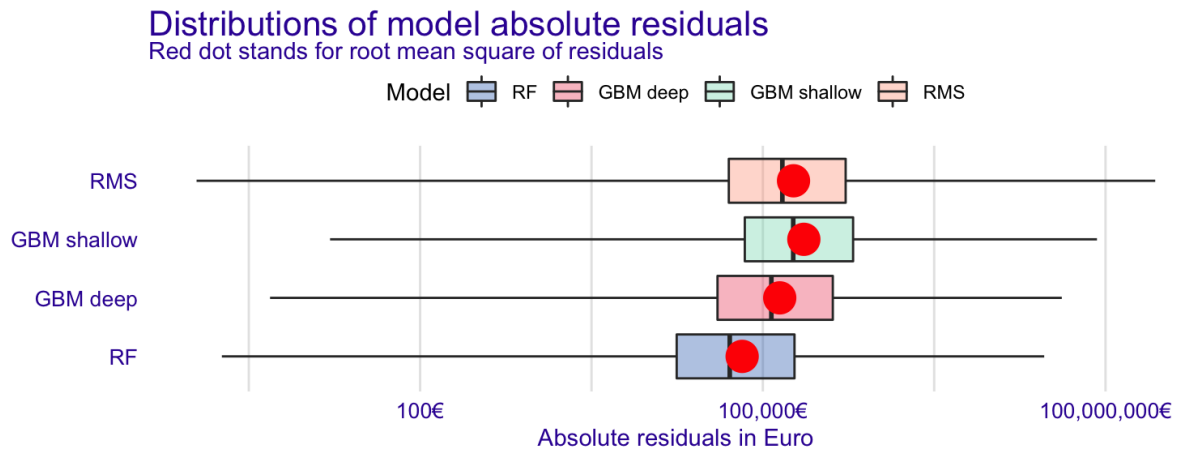



Figure 21.5: Distribution of absolute values of residuals. The smaller are values the better is the model. Dots stand for averages.

But performance is not everything. Figure 21.6 show diagnostics plots for every model. Each scatterplot shows true target variable against model predictions. The random forest model has predictions closest to the true target values.

Extreme predictions (lowest and highest) are biased towards the mean, what is typical for such type of models. This means that Random Forest models learned factors that influence players' values, but for the most expensive players these values will be underestimated.

```
fifa_md_gbm_shallow <- model_diagnostics(fifa_gbm_exp_shallow)
fifa_md_gbm_deep <- model_diagnostics(fifa_gbm_exp_deep)
fifa_md_gbm_rf <- model_diagnostics(fifa_rf_exp)
fifa_md_gbm_rms <- model_diagnostics(fifa_rms_exp)

plot(fifa_md_gbm_shallow, fifa_md_gbm_deep,
                fifa_md_gbm_rf, fifa_md_gbm_rms,
     variable = "y", yvariable = "y_hat") +
  scale_x_continuous("Value in Euro", trans = "log10", labels = dollar_format(suffix = "
  scale_y_continuous("Estimated value in Euro", trans = "log10", labels = dollar_format(
  facet_wrap(~label) +
  geom_abline(slope = 1) + theme(legend.position = "none") +
  ggtitle("Diagnostics plot Predicted vs True target values", "")
```

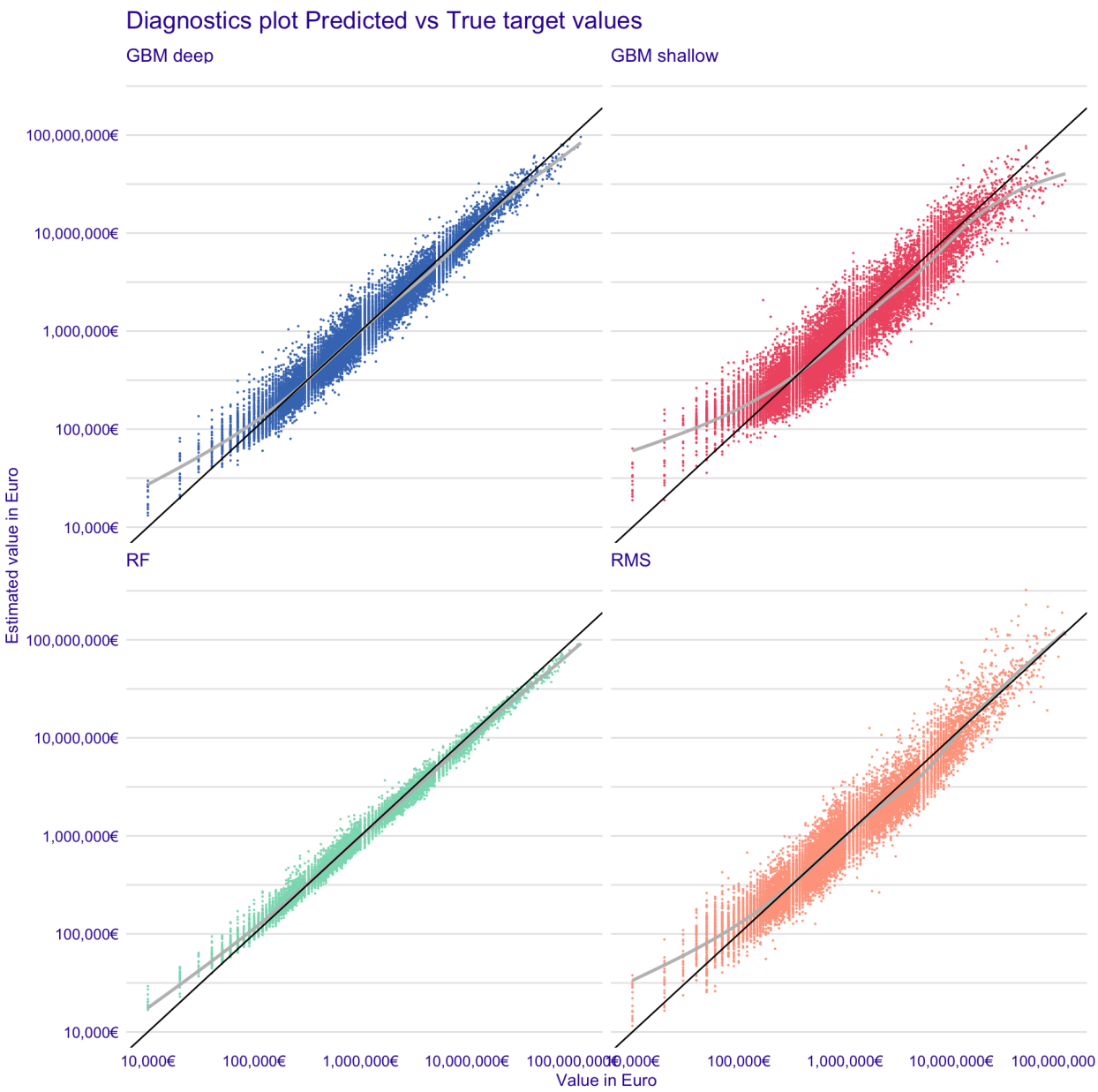



Figure 21.6: Diagnostics plots Predicted vs. True target values. Points correspond to particular players. The closer to the diagonal the better is the model.

# 21.6 Model understanding

Figure 21.7 shows variable importance plots for four selected models. Only 12 most important variables in each model are presented.

Some variables are important for all models, like `Reactions` or `BallControl`. Importance of other variables may be very different. All models except random forest are using some characteristics of goalkeepers.

```r
fifa_mp_gbm_shallow <- model_parts(fifa_gbm_exp_shallow)
fifa_mp_gbm_deep <- model_parts(fifa_gbm_exp_deep)
fifa_mp_rf <- model_parts(fifa_rf_exp)
fifa_mp_rms <- model_parts(fifa_rms_exp)

plot(fifa_mp_gbm_shallow, fifa_mp_gbm_deep,
     fifa_mp_rf, fifa_mp_rms,
     max_vars = 20, bar_width = 4, show_boxplots = FALSE)
```

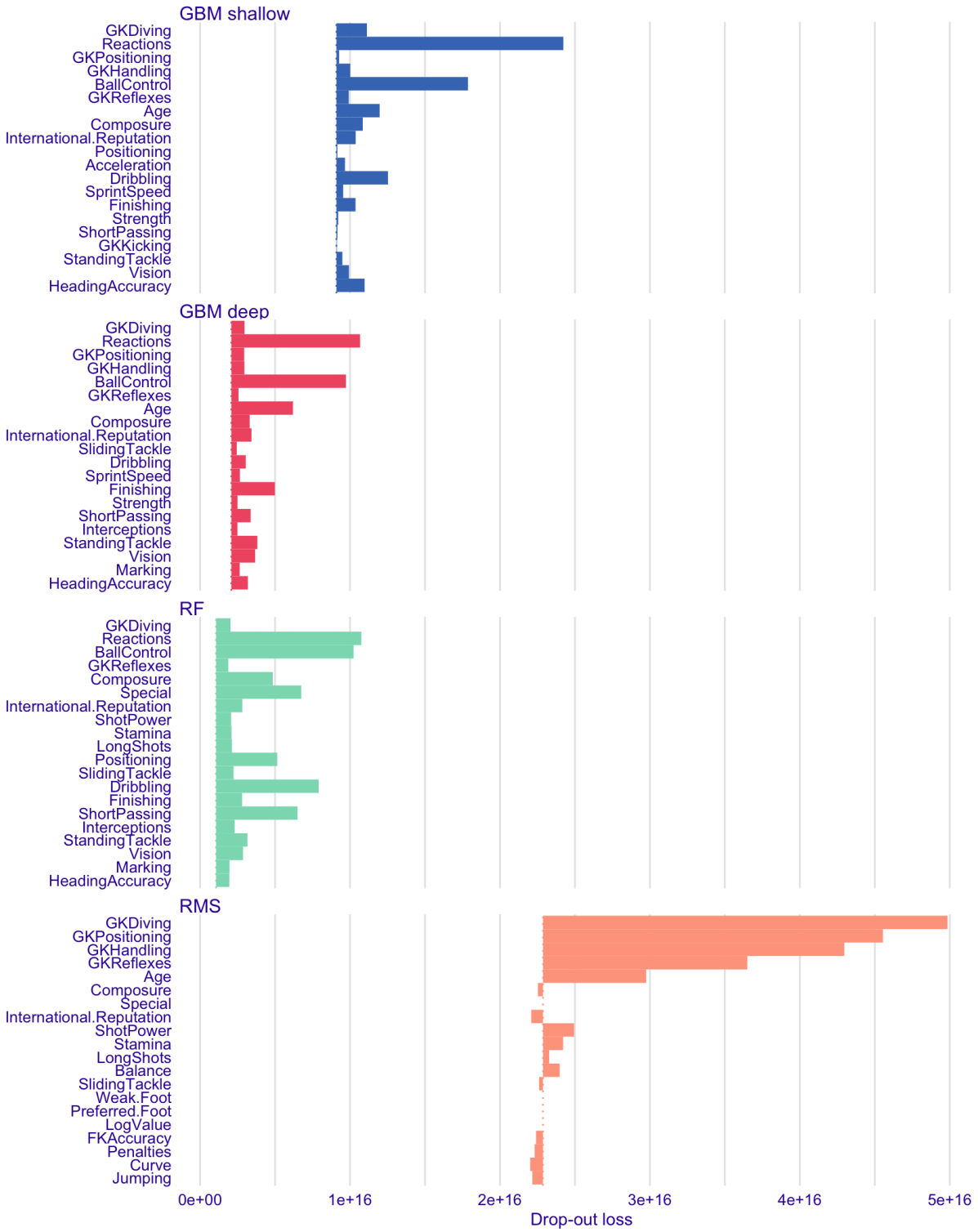



Figure 21.7: Variable importance plots for four considered models. Each bar starts in a RMSE for the model and ends in a RMSE calculated for data with permuted single variable.

Figure 21.8 shows Partial Dependence profiles for the most important variables. They show average relation between particular variable and players value.

The general direction of relation in all models is the same. The larger the player characteristic the higher is the price. With a single exception – variable Age.

Random forest model has smallest range of average model responses. All tree-based models stabilize average predictions at the ends of variables ranges.

The most interesting difference between Exploratory Data Analysis presented in Figure 21.3 and Exploratory Model Analysis presented in Figure 21.8 is related with variable `Age`. In Figure 21.3 the relation was non-monotonic while in Figure 21.8 its monotonically decreasing. How we can explain this difference? One explanation is following: Youngest players have lower values not because of their age but because of lower skills that are correlated with Age. The EDA analysis cannot entangle these effects, thus for youngest players we see lover values also because their lower skills. But models learned that once we take skills into account, the effect of age is only decreasing.

This example also shows, that proper *exploration of models may be more insightful than exploration of raw data*. Variable `Age` is correlated with other confounding variables. This entangle was visible in the EDA analysis. But models learned to disentangle these effects.

```
selected_variables <- c("Age", "Reactions","BallControl", "ShortPassing")

fifa19_pd_shallow <- model_profile(fifa_gbm_exp_shallow, variables = selected_variables)
fifa19_pd_deep <- model_profile(fifa_gbm_exp_deep, variables = selected_variables)$agr_p
fifa19_pd_rf <- model_profile(fifa_rf_exp, variables = selected_variables)$agr_profiles
fifa19_pd_rms <- model_profile(fifa_rms_exp, variables = selected_variables)$agr_profile

plot(fifa19_pd_shallow, fifa19_pd_deep, fifa19_pd_rf, fifa19_pd_rms) +
  scale_y_continuous("Estimated value in Euro", trans = "log10", labels = dollar_format(
  ggtitle("Partial Dependence profiles for selected variables")
```

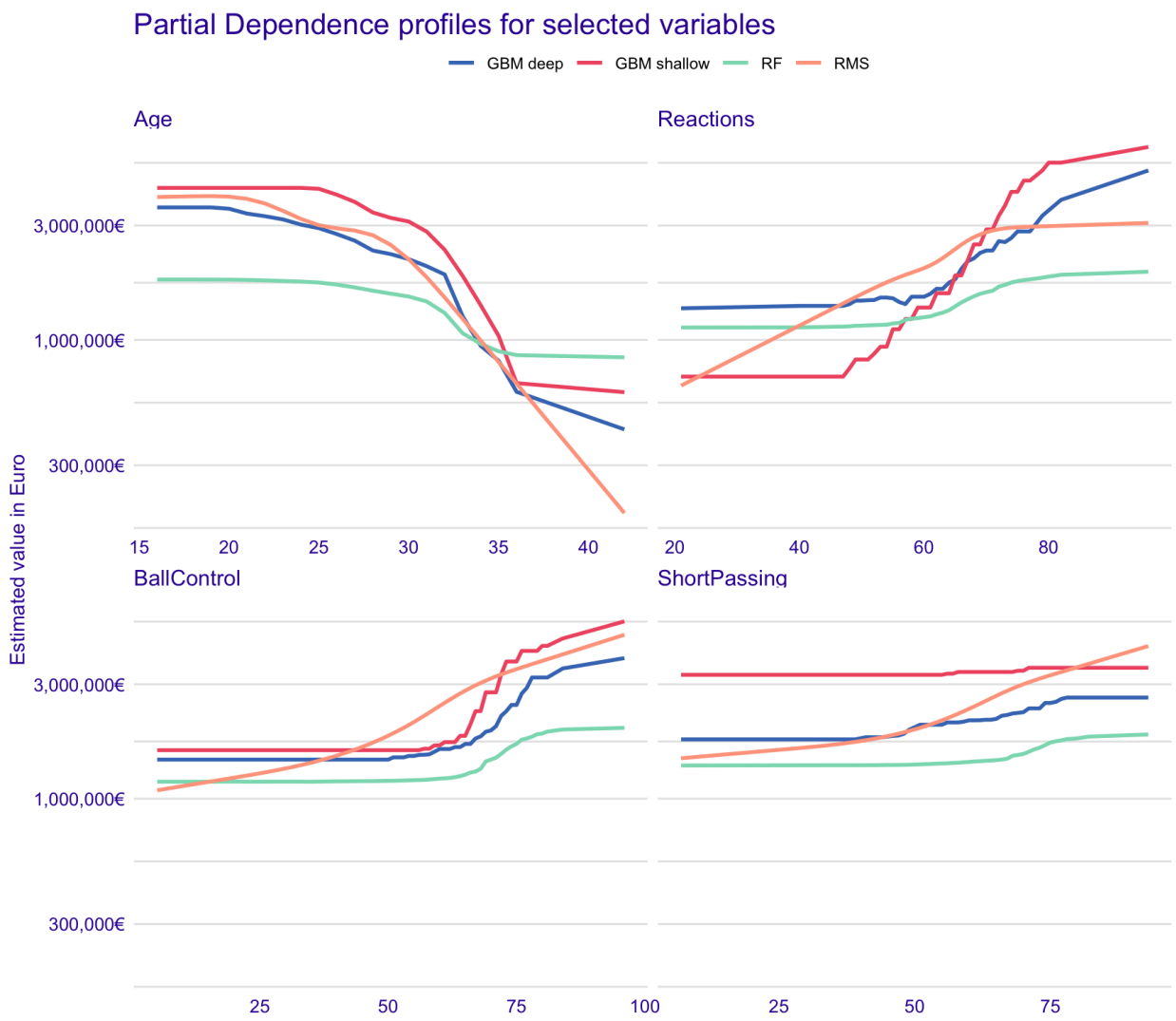



Figure 21.8: Partial Dependence profiles for four selected variables and four considered models.

## 21.7 Instance understanding

Time to see how the model behaves for a single observation / player This can be done for any player, but for this example we will use *Robert Lewandowski*, the most valuable polish football player.

Here are his characteristics in the FIFA 19 database.

```
fifa19small["R. Lewandowski",]
```

```
##                Age Special Preferred.Foot International.Reputation
## R. Lewandowski  29    2152          Right                        4
##                Weak.Foot Skill.Moves Crossing Finishing HeadingAccuracy
## R. Lewandowski         4           4       62        91              85
##                ShortPassing Volleys Dribbling Curve FKAccuracy LongPassing
## R. Lewandowski           83      89        85    77         86          65
##                BallControl Acceleration SprintSpeed Agility Reactions
## R. Lewandowski          89           77          78      78        90
##                Balance ShotPower Jumping Stamina Strength LongShots
## R. Lewandowski      78        88      84      78       84        84
##                Aggression Interceptions Positioning Vision Penalties
## R. Lewandowski         80            39          91     77        88
##                Composure Marking StandingTackle SlidingTackle GKDiving
## R. Lewandowski        86      34             42            19       15
##                GKHandling GKKicking GKPositioning GKReflexes LogValue
## R. Lewandowski          6        12             8         10 7.886491
```

In the chapter 7 we showed a Break Down plots for presentation of variable attributions. In the Figure 21.9 we show Break Down plots for Robert Lewandowski predictions.

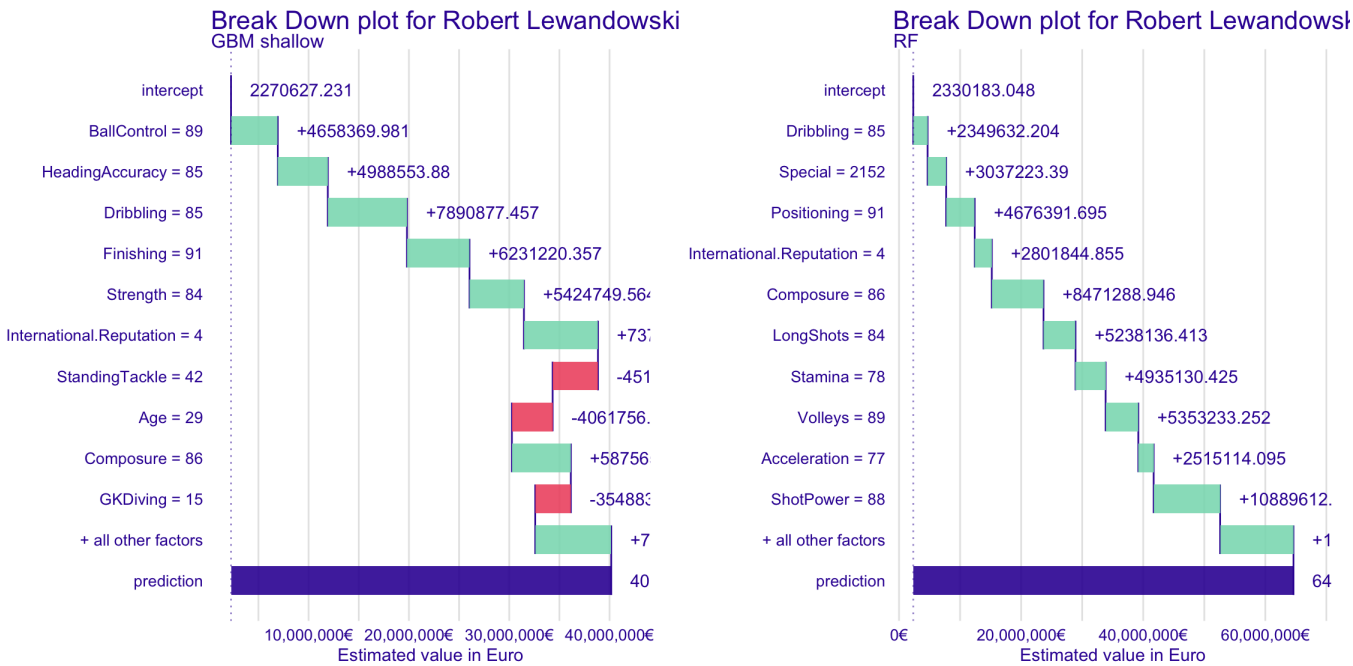



Figure 21.9: Break down plot for Robert Lewandowski. Results for GBM and RF model.

In the chapter 9 we showed a SHAP values for presentation of variable attributions. In the Figure 21.10 we show SHAP plots for Robert Lewandowski predictions. As it was expected, these explanations are consistent.

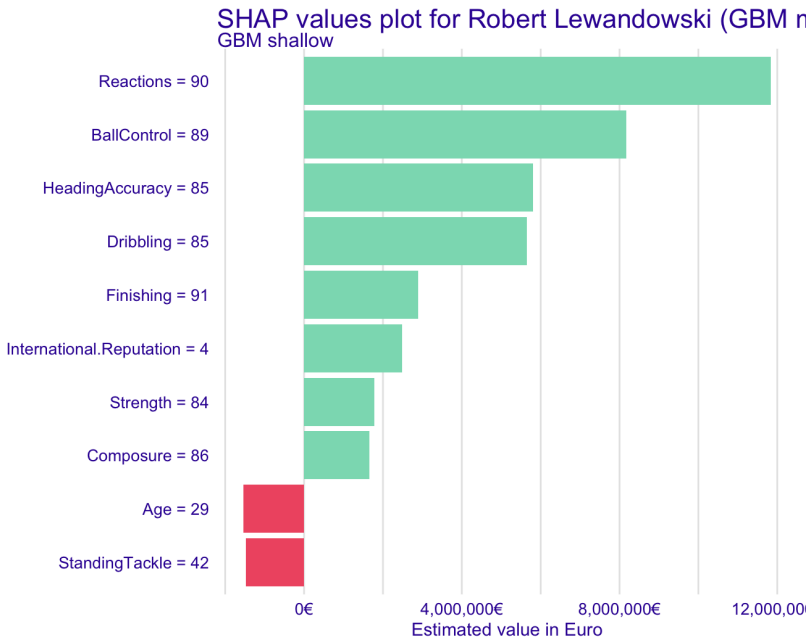



Figure 21.10: SHAP values for GBM model.

Robert Lewandowski is a striker. It makes sense that his most valuable characteristics are `Reactions` and `BallControl`.

How these plots will look like for goalkeepers? Figure 21.11 show Break Down plots for Wojciech Szczęsny - most valuable polish goalkeeper. As we see the most important coefficients make sense, most of them are linked with properties of goalkeepers.

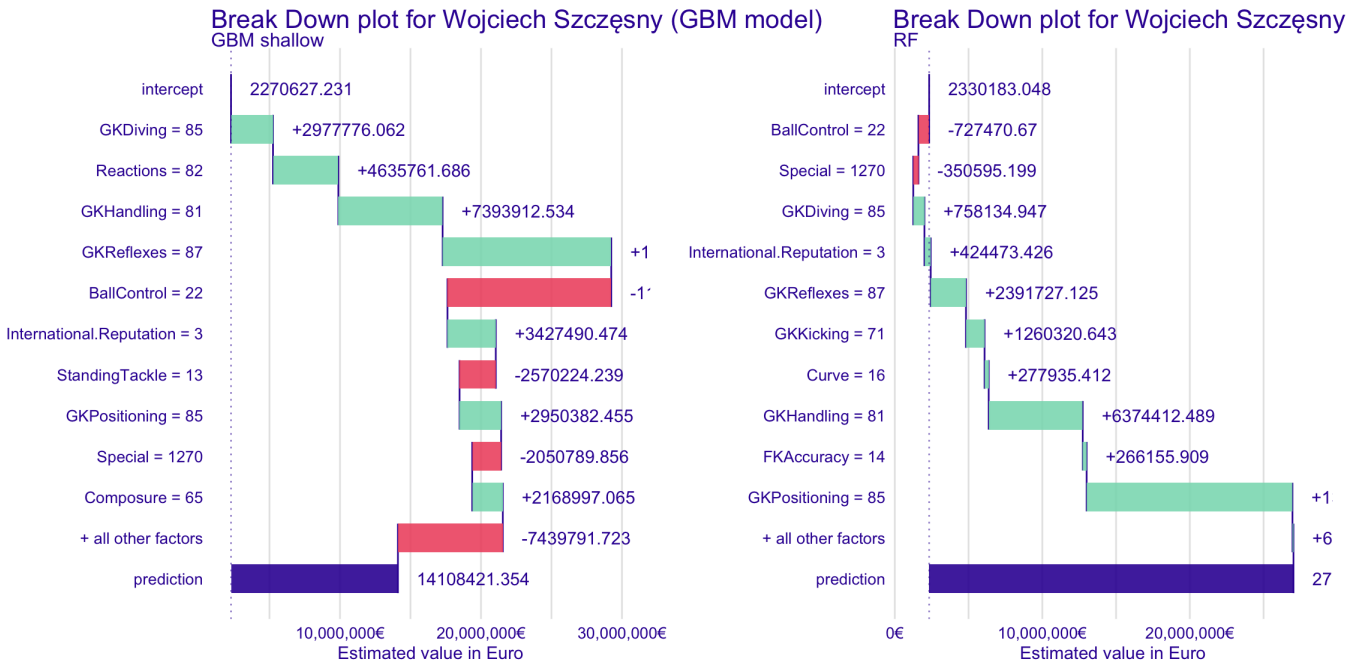



Figure 21.11: Break down plot for Wojciech Szczęsny. Results for GBM and RF model.

In Chapter 11 we introduced Ceteris Paribus profiles. These are more details steps of the model exploration. Based on an example of Robert Lewandowski. Figure 21.12 show how change in one characteristic affects model value.

Tree based models are flat on borders, and for them Robert value is the highest for variables `Reactions`, `BallControl` or `Dribling`. When it comes to Age we see that the predicted value is just before a larger drop in value prediction.

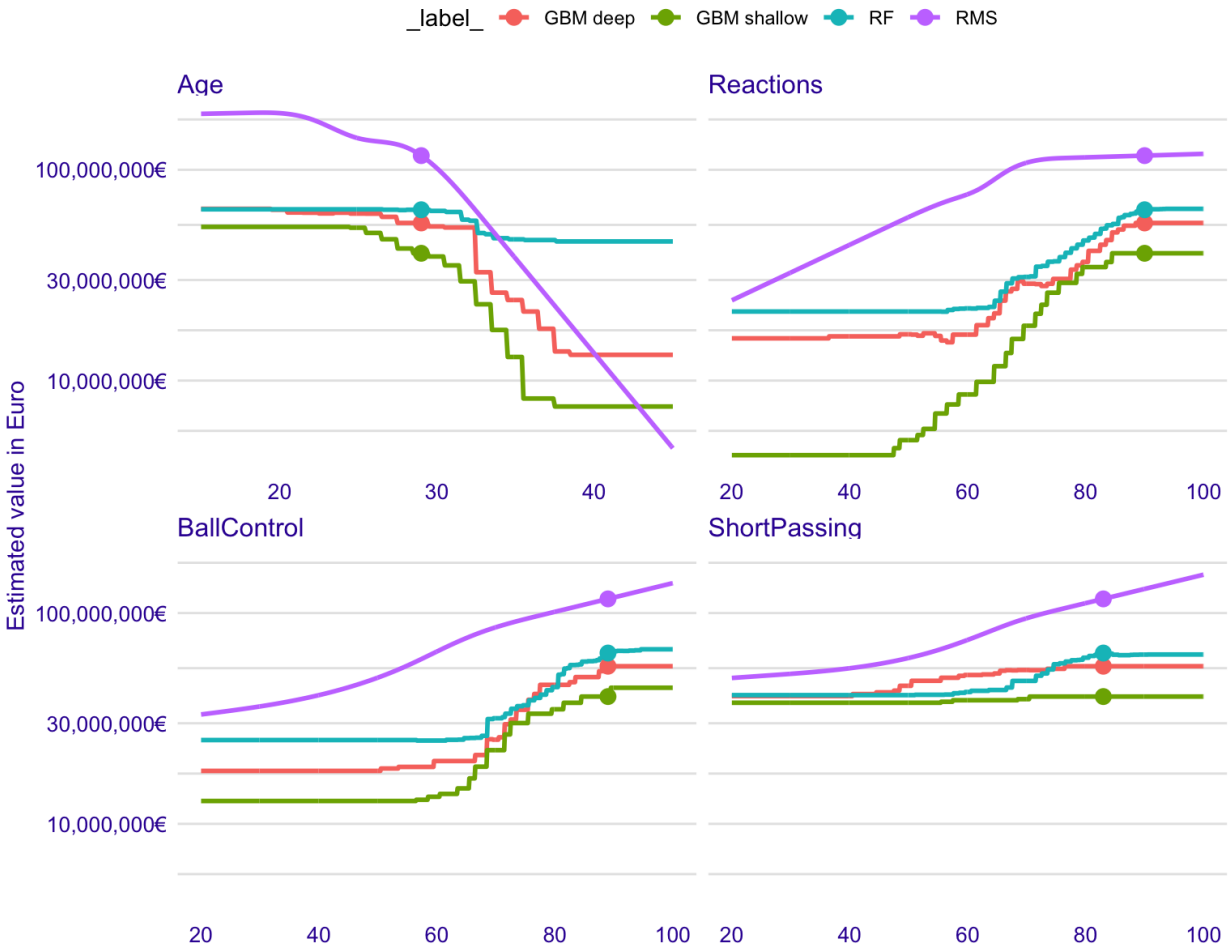



Figure 21.12: Ceteris Paribus profiles for Robert Lewandowski for four selected variables.

Figure 21.13 shows residuals for all observations against residuals for 50 closest neighbours of Robert Lewandowski. Clearly, among neighbours he has the most expensive players and therefore their residuals are much higher than average residuals. This was also visible in the right top corner of each panel in Figure 21.6.

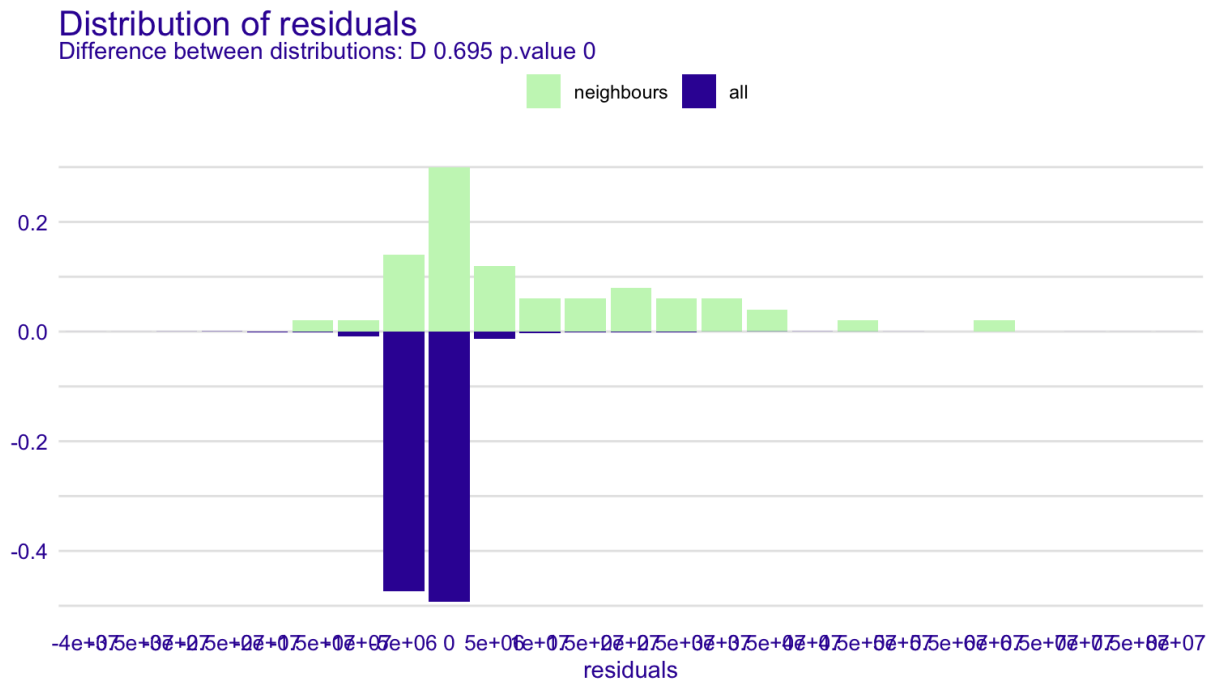



Figure 21.13: Distribution of residuals for all players and neighbours of Robert Lewandowski.

Figure 21.14 shows local-fidelity plot for 15 neighbours of Robert Lewandowski. Model behaviour for these neighbours are similar. We also see that the most expensive players are undervalued by the model.

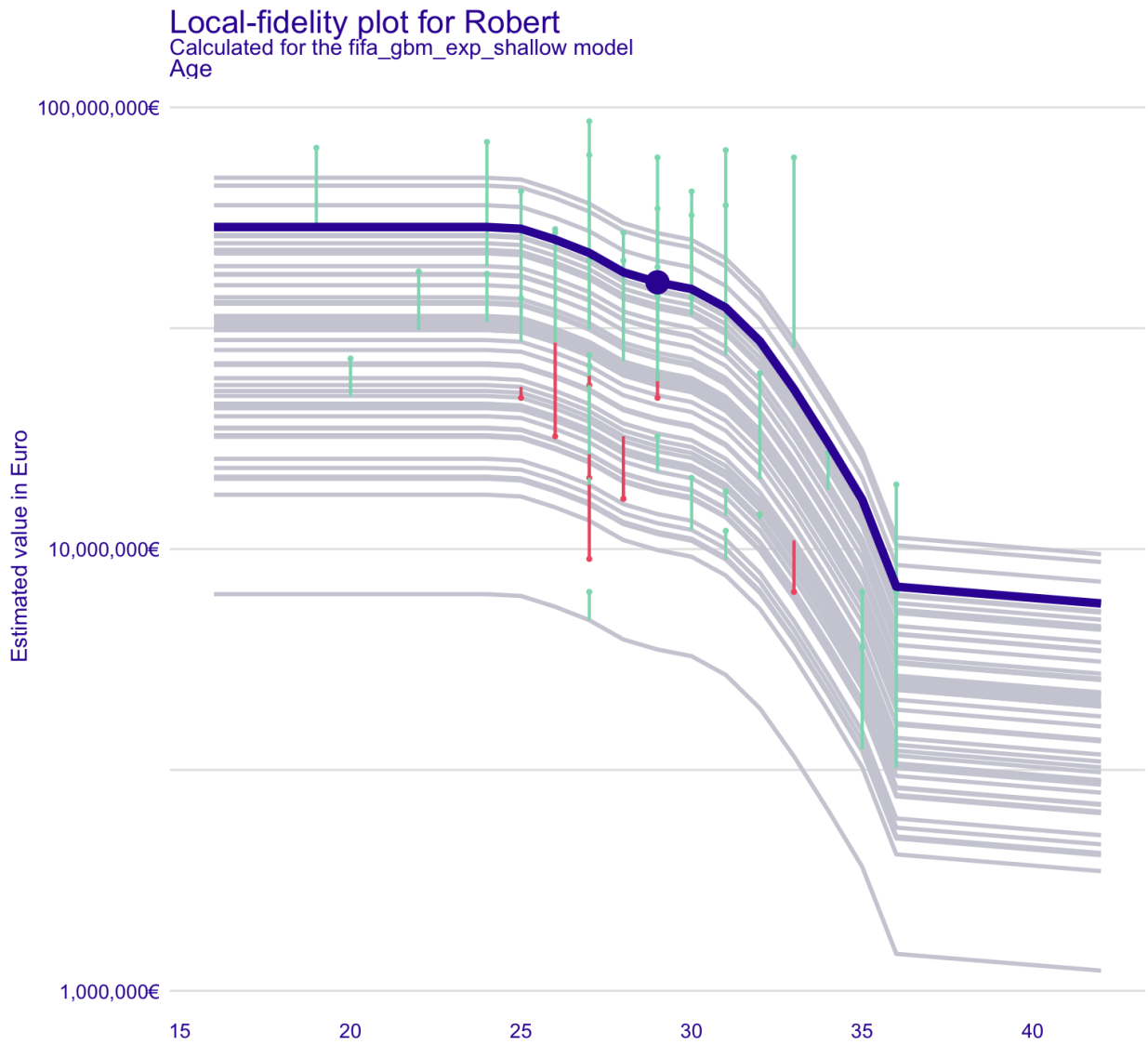



Figure 21.14: Ceteris Paribus profiles for 15 neighbours of Robert Lewandowski.

## 21.8 CR7

In this section we present model explanations for Cristiano Ronaldo (CR7). Here are his characteristics in the FIFA 19 database.

```
fifa19small["Cristiano Ronaldo",]
```

```
##                    Age Special Preferred.Foot International.Reputation
## Cristiano Ronaldo   33    2228          Right                        5
##                    Weak.Foot Skill.Moves Crossing Finishing HeadingAccuracy
## Cristiano Ronaldo          4           5       84        94              89
##                    ShortPassing Volleys Dribbling Curve FKAccuracy
## Cristiano Ronaldo           81      87        88    81         76
##                    LongPassing BallControl Acceleration SprintSpeed Agility
## Cristiano Ronaldo          77          94           89          91      87
##                    Reactions Balance ShotPower Jumping Stamina Strength
## Cristiano Ronaldo         96      70        95      95      88       79
##                    LongShots Aggression Interceptions Positioning Vision
## Cristiano Ronaldo         93         63            29          95     82
##                    Penalties Composure Marking StandingTackle SlidingTackle
## Cristiano Ronaldo         85        95      28             31            23
##                    GKDiving GKHandling GKKicking GKPositioning GKReflexes
## Cristiano Ronaldo         7         11        15            14          11
##                    LogValue
## Cristiano Ronaldo 7.886491
```

Let's start with Break Down plots for variable attributions.

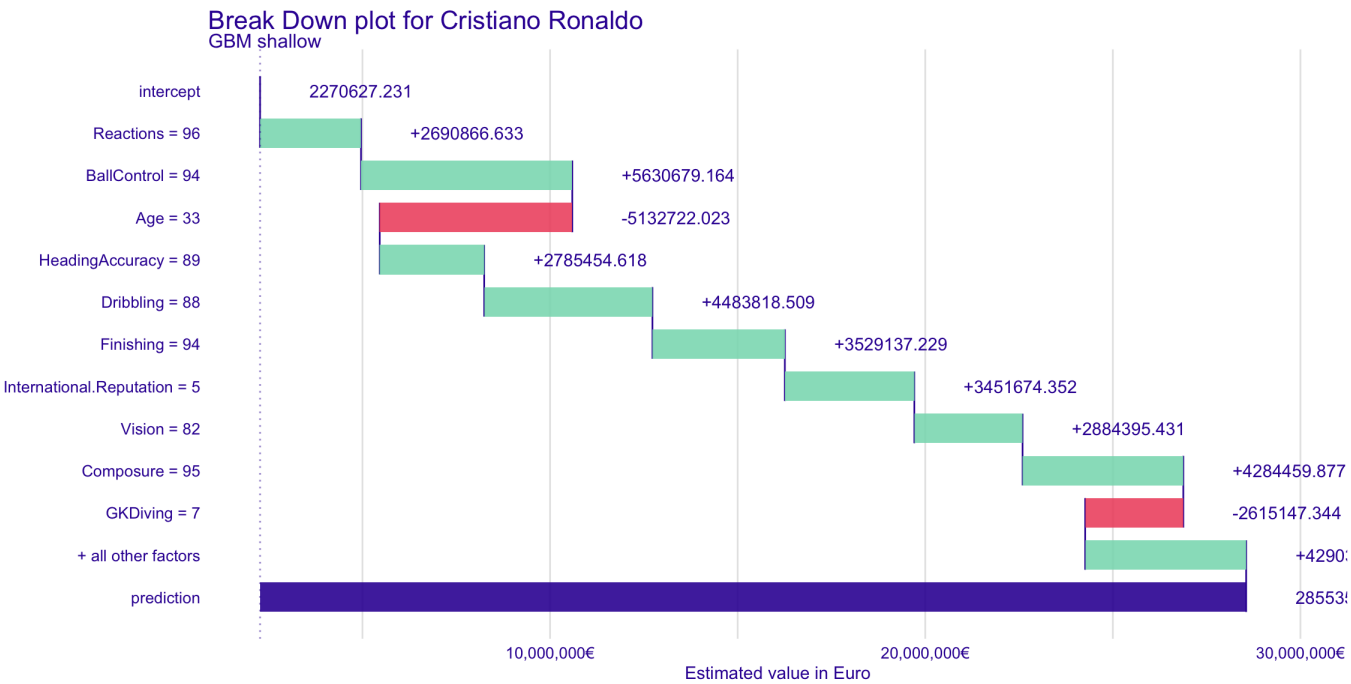



Figure 21.15: Break down plot for Cristiano Ronaldo.

Cristiano Ronaldo is a striker. It makes sense that his most valuable characteristics are `Reactions` and `BallControl`. Let's see Ceteris Paribus profiles for him.

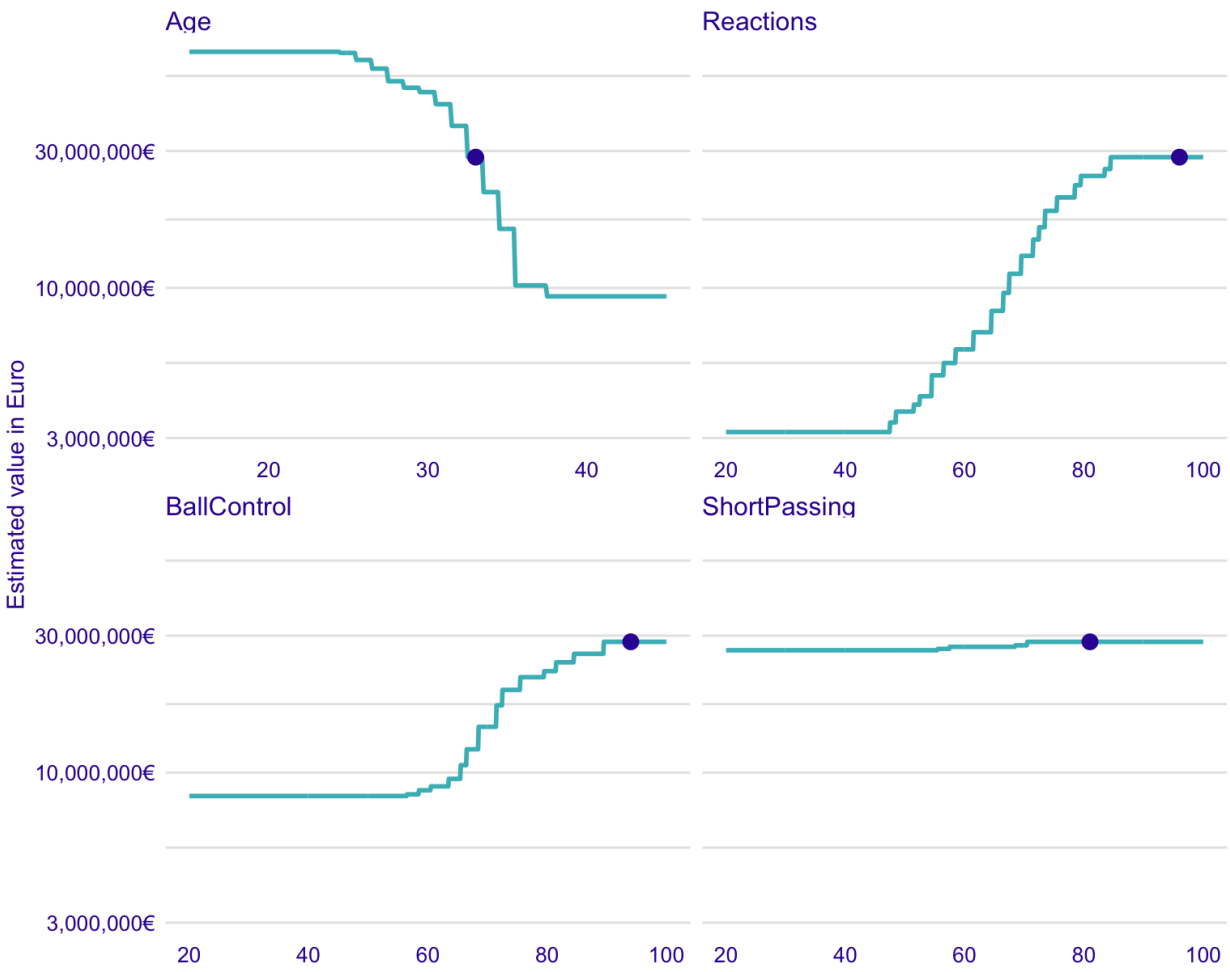



Figure 21.16: Ceteris Paribus profiles for Cristiano Ronaldo for four selected variables.

# References


Harrell Jr, Frank E. 2018. *Rms: Regression Modeling Strategies*. https://CRAN.R-project.org/package=rms.

Ridgeway, Greg. 2017. *Gbm: Generalized Boosted Regression Models*. https://CRAN.R-project.org/package=gbm.

Wright, Marvin N., and Andreas Ziegler. 2017. *ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R*. *Journal of Statistical Software*. Vol. 77. https://doi.org/10.18637/jss.v077.i01.

# References


Alber, Maximilian, Sebastian Lapuschkin, Philipp Seegerer, Miriam Hägele, Kristof T. Schütt, Grégoire Montavon, Wojciech Samek, Klaus-Robert Müller, Sven Dähne, and Pieter-Jan Kindermans. 2018. “INNvestigate Neural Networks!”

Allaire, JJ, and François Chollet. 2019. *Keras: R Interface to 'Keras'*. https://CRAN.R-project.org/package=keras.

Alvarez-Melis, David, and Tommi S. Jaakkola. 2018. “On the Robustness of Interpretability Methods.” *arXiv E-Prints*, June, arXiv:1806.08049.

Apley, Dan. 2018. *ALEPlot: Accumulated Local Effects (Ale) Plots and Partial Dependence (Pd) Plots*. https://CRAN.R-project.org/package=ALEPlot.

Apley, Daniel W., and Jingyu Zhu. 2019. “Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models.” *CoRR* abs/1612.08468. http://arxiv.org/abs/1612.08468.

Azure. 2019. “Microsoft Cognitive Services.” https://azure.microsoft.com/en-en/services/cognitive-services/.

Bach, Sebastian, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. “On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation.” Edited by Oscar Deniz Suarez. *Plos One* 10 (7): e0130140. https://doi.org/10.1371/journal.pone.0130140.

Biecek, Przemyslaw. 2018. *DALEX: Explainers for Complex Predictive Models in R*. *Journal of Machine Learning Research*. Vol. 19. http://jmlr.org/papers/v19/18-416.html.

———. 2019. “Model Development Process.” *CoRR* abs/1907.04461. http://arxiv.org/abs/1907.04461.

Biecek, Przemyslaw, Hubert Baniecki, Adam Izdebski, and Katarzyna Pekala. 2019. *ingredients: Effects and Importances of Model Ingredients*.

Biecek, Przemyslaw, and Marcin Kosinski. 2017. “archivist: An R Package for Managing, Recording and Restoring Data Analysis Results.” *Journal of Statistical Software* 82 (11): 1–28. https://doi.org/10.18637/jss.v082.i11.

Binder, Alexander, Grégoire Montavon, Sebastian Bach, Klaus-Robert Müller, and Wojciech Samek. 2016. “Layer-Wise Relevance Propagation for Neural Networks with Local Renormalization Layers.” *CoRR* abs/1604.00825. http://arxiv.org/abs/1604.00825.

Bischl, Bernd, Michel Lang, Lars Kotthoff, Julia Schiffner, Jakob Richter, Erich Studerus, Giuseppe Casalicchio, and Zachary M. Jones. 2016. “mlr: Machine Learning in R.” *Journal of Machine Learning Research* 17 (170): 1–5. http://jmlr.org/papers/v17/15-066.html.

Boehm, Barry. 1988. *A Spiral Model of Software Development and Enhancement*. *IEEE Computer, IEEE, 21(5):61-72*.

Breiman, Leo. 2001. “Random Forests.” In *Machine Learning*, 45:5–32. https://doi.org/10.1023/a:1010933404324.

Breiman, Leo, Adele Cutler, Andy Liaw, and Matthew Wiener. 2018. *RandomForest: Breiman and Cutler’s Random Forests for Classification and Regression*. https://CRAN.R-project.org/package=randomForest.

Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification and Regression Trees*. Monterey, CA: Wadsworth; Brooks.

Casey, Bryan, Ashkon Farhangi, and Roland Vogl. 2018. “Rethinking Explainable Machines: The Gdpr’s ’Right to Explanation’ Debate and the Rise of Algorithmic Audits in Enterprise.” *Berkeley Technology Law Journal*. https://ssrn.com/abstract=3143325.

Chapman, Pete, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rudiger Wirth. 1999. *The CRISP-DM 1.0 Step-by-step data mining guide*. ftp://ftp.software.ibm.com/software/analytics/spss/support/Modeler/Documentation/14/UserManual/CRISP-DM.pdf.

Cortes, Corinna, and Vladimir Vapnik. 1995. “Support-Vector Networks.” In *Machine Learning*, 273–97.

Dastin, Jeffrey. 2018. “Amazon Scraps Secret Ai Recruiting Tool That Showed Bias Against Women.” *Reuters*. https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazonscraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G.

Deng, J., W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. 2009. “ImageNet: A large-scale hierarchical image database.” In *2009 Ieee Conference on Computer Vision and Pattern Recognition*, 248–55. https://doi.org/10.1109/cvpr.2009.5206848.

Diaz, Mark, Isaac Johnson, Amanda Lazar, Anne Marie Piper, and Darren Gergle. 2018. “Addressing Age-Related Bias in Sentiment Analysis.” In *Proceedings of the 2018 Chi Conference on Human Factors in Computing Systems*, 412:1–412:14. Chi ’18. New York, NY, USA: Acm. https://doi.org/10.1145/3173574.3173986.

Donizy, Piotr, Przemyslaw Biecek, Agnieszka Halon, and Rafal Matkowski. 2016. “BILLCD8 – a Multivariable Survival Model as a Simple and Clinically Useful Prognostic Tool to Identify High-Risk Cutaneous Melanoma Patients” 36 (September): 4739–48.

Duffy, Clare. 2019. “Apple Co-Founder Steve Wozniak Says Apple Card Discriminated Against His Wife.” *CNN Business*. https://edition.cnn.com/2019/11/10/business/goldman-sachs-apple-card-discrimination/index.html.

Efron, Bradley, and Trevor Hastie. 2016. *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science*. 1st ed. New York, NY, USA: Cambridge University Press.

Faraway, Julian. 2002. *Practical Regression and Anova Using R*.

Fisher, Aaron, Cynthia Rudin, and Francesca Dominici. 2018. “Model Class Reliance: Variable Importance Measures for Any Machine Learning Model Class, from the ’Rashomon’ Perspective.” *Journal of Computational and Graphical Statistics*. http://arxiv.org/abs/1801.01489.

Foster, David. 2017. *XgboostExplainer: An R Package That Makes Xgboost Models Fully Interpretable*. https://github.com/AppliedDataSciencePartners/xgboostExplainer/.

Friedman, Jerome H. 2000. “Greedy Function Approximation: A Gradient Boosting Machine.” *Annals of Statistics* 29: 1189–1232.

Galecki, Andrzej, and Tomasz Burzykowski. 2013. *Linear Mixed-Effects Models Using R: A Step-by-Step Approach*. Springer Publishing Company, Incorporated.

Gdpr. 2018. “The Eu General Data Protection Regulation (Gdpr) Is the Most Important Change in Data Privacy Regulation in 20 Years.” https://eugdpr.org/.

Goldstein, Alex, Adam Kapelner, Justin Bleich, and Emil Pitkin. 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. https://doi.org/10.1080/10618600.2014.907095.

Goodman, Bryce, and Seth Flaxman. 2016. “European Union Regulations on Algorithmic Decision-Making and a "Right to Explanation".” *Arxiv*. https://arxiv.org/abs/1606.08813.

Gosiewska, Alicja, and Przemyslaw Biecek. 2018. *Auditor: Model Audit - Verification, Validation, and Error Analysis*. https://CRAN.R-project.org/package=auditor.

———. 2019a. “iBreakDown: Uncertainty of Model Explanations for Non-additive Predictive Models.” https://arxiv.org/abs/1903.11420v1.

———. 2019b. *shapper: Wrapper of Python Library ’shap’*. https://github.com/ModelOriented/shapper.

Greenwell, Brandon M. 2017. “pdp: An R Package for Constructing Partial Dependence Plots.” *The R Journal* 9 (1): 421–36. https://journal.r-project.org/archive/2017/RJ-2017-016/index.html.

Grolemund, Garrett, and Hadley Wickham. 2019. *R for Data Science*. https://r4ds.had.co.nz/.

Hall, Patrick. 2019. *On Explainable Machine Learning Misconceptions and a More Human-Centered Machine Learning*. https://github.com/jphall663/xai_misconceptions/blob/master/xai_misconceptions.pdf.

Harrell Jr, Frank E. 2018. *Rms: Regression Modeling Strategies*. https://CRAN.R-project.org/package=rms.

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. “Long Short-Term Memory.” *Neural Computation* 9 (8): 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735.

Hoover, Benjamin, Hendrik Strobelt, and Sebastian Gehrmann. 2019. “ExBERT: A Visual Analysis Tool to Explore Learned Representations in Transformers Models.”

Hothorn, Torsten, Kurt Hornik, and Achim Zeileis. 2006. “Unbiased Recursive Partitioning: A Conditional Inference Framework.” *Journal of Computational and Graphical Statistics* 15 (3): 651–74.

Jacobson, Ivar, Grady Booch, and James Rumbaugh. 1999. *The Unified Software Development Process*.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2014. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.

Jed Wing, Max Kuhn. Contributions from, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, et al. 2016. *Caret: Classification and Regression Training*. https://CRAN.R-project.org/package=caret.

Karbowiak, Ewelina, and Przemyslaw Biecek. 2019. *EIX: Explain Interactions in Gradient Boosting Models*. https://CRAN.R-project.org/package=EIX.

Kruchten, Philippe. 1998. *The Rational Unified Process*.

Kuhn, Max, and Kjell Johnson. 2013. *Applied predictive modeling*. New York, NY: Springer. http://appliedpredictivemodeling.com/.

Kuhn, Max, and Davis Vaughan. 2019. *Parsnip: A Common Api to Modeling and Analysis Functions*. https://CRAN.R-project.org/package=parsnip.

Larson, Jeff, Surya Mattu, Lauren Kirchner, and Julia Angwin. 2016. "How We Analyzed the Compas Recidivism Algorithm." *ProPublica*. https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm.

Lazer, David, Ryan Kennedy, Gary King, and Alessandro Vespignani. 2014. "The Parable of Google Flu: Traps in Big Data Analysis." *Science* 343 (6176). American Association for the Advancement of Science: 1203–5. https://doi.org/10.1126/science.1248506.

LeDell, Erin, Navdeep Gill, Spencer Aiello, Anqi Fu, Arno Candel, Cliff Click, Tom Kraljevic, et al. 2019. *H2o: R Interface for 'H2o'*. https://CRAN.R-project.org/package=h2o.

Liaw, Andy, and Matthew Wiener. 2002. "Classification and Regression by randomForest." *R News* 2 (3): 18–22. http://CRAN.R-project.org/doc/Rnews/.

Lundberg, Scott. 2019. *SHAP (SHapley Additive exPlanations)*. https://github.com/slundberg/shap.

Lundberg, Scott M., Gabriel G. Erion, and Su-In Lee. 2018. "Consistent Individualized Feature Attribution for Tree Ensembles." *CoRR* abs/1802.03888. http://arxiv.org/abs/1802.03888.

Lundberg, Scott M, and Su-In Lee. 2017. "A Unified Approach to Interpreting Model Predictions." In *Advances in Neural Information Processing Systems 30*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, 4765–74. Curran Associates, Inc. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

Max, Kuhn, and Hadley Wickham. 2018. *Tidymodels: Easily Install and Load the 'Tidymodels' Packages*. https://CRAN.R-project.org/package=tidymodels.

Meyer, David, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, and Friedrich Leisch. 2017. *E1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), Tu Wien*. https://CRAN.R-project.org/package=e1071.

———. 2019. *E1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), Tu Wien*. https://CRAN.R-project.org/package=e1071.

Molnar, Christoph. 2019. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*.

Molnar, Christoph, Bernd Bischl, and Giuseppe Casalicchio. 2018. “iml: An R package for Interpretable Machine Learning.” *Joss* 3 (26). Journal of Open Source Software: 786. https://doi.org/10.21105/joss.00786.

Nolan, Deborah, and Duncan Temple Lang. 2015. *Data Science in R: A Case Studies Approach to Computational Reasoning and Problem Solving*. Chapman & Hall/CRC.

O’Connell, Mark, Catherine Hurley, and Katarina Domijan. 2017. “Conditional Visualization for Statistical Models: An Introduction to the Condvis Package in R.” *Journal of Statistical Software, Articles* 81 (5): 1–20. https://doi.org/10.18637/jss.v081.i05.

O’Neil, Cathy. 2016. *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. New York, NY, USA: Crown Publishing Group.

Paluszynska, Aleksandra, and Przemyslaw Biecek. 2017. *RandomForestExplainer: A Set of Tools to Understand What Is Happening Inside a Random Forest*. https://github.com/MI2DataLab/randomForestExplainer.

Pedersen, Thomas Lin, and Michaël Benesty. 2019. *lime: Local Interpretable Model-Agnostic Explanations*. https://CRAN.R-project.org/package=lime.

Picard, Dominique. 1985. “Testing and estimating change-points in time series.” *Advances in Applied Probability* 17 (4). Cambridge University Press: 841–67. https://doi.org/10.2307/1427090.

R Core Team. 2018. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. 2016. “Why Should I Trust You?: Explaining the Predictions of Any Classifier.” In, 1135–44. ACM Press. https://doi.org/10.1145/2939672.2939778.

Ridgeway, Greg. 2017. *Gbm: Generalized Boosted Regression Models*. https://CRAN.R-project.org/package=gbm.

Robnik-Šikonja, Marco, and Igor Kononenko. 2008. “Explaining Classifications for Individual Instances.” *IEEE Transactions on Knowledge and Data Engineering* 20 (5): 589–600. https://doi.org/10.1109/tkde.2007.190734.

Robnik-Šikonja, Marko. 2018. *ExplainPrediction: Explanation of Predictions for Classification and Regression Models*. https://CRAN.R-project.org/package=ExplainPrediction.

Ross, Casey, and Ike Swetliz. 2018. “IBM’s Watson Supercomputer Recommended ‘Unsafe and Incorrect’ Cancer Treatments, Internal Documents Show.” *Statnews*. https://www.statnews.com/2018/07/25/ibm-watson-recommended-unsafe-incorrect-treatments/.

Ruiz, Javier. 2018. “Machine Learning and the Right to Explanation in Gdpr.” https://www.openrightsgroup.org/blog/2018/machine-learning-and-the-right-to-explanation-in-gdpr.

Salzberg, Steven. 2014. “Why Google Flu Is A Failure.” *Forbes*. https://www.forbes.com/sites/stevensalzberg/2014/03/23/why-google-flu-is-a-failure/.

Samek, Wojciech, Thomas Wiegand, and Klaus-Robert Müller. 2017. “Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models.”

Shapley, Lloyd S. 1953. “A Value for n-Person Games.” In *Contributions to the Theory of Games Ii*, edited by Harold W. Kuhn and Albert W. Tucker, 307–17. Princeton: Princeton University Press.

Sheather, Simon. 2009. *A Modern Approach to Regression with R*. Springer Texts in Statistics. Springer New York.

Shrikumar, Avanti, Peyton Greenside, and Anshul Kundaje. 2017. “Learning Important Features Through Propagating Activation Differences.” *CoRR* abs/1704.02685. http://arxiv.org/abs/1704.02685.

Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman. 2013. “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps.” *CoRR* abs/1312.6034. http://arxiv.org/abs/1312.6034.

Simonyan, Karen, and Andrew Zisserman. 2015. “Very Deep Convolutional Networks for Large-Scale Image Recognition.” In *International Conference on Learning Representations*.

Sing, T., O. Sander, N. Beerenwinkel, and T. Lengauer. 2005. “ROCR: visualizing classifier performance in R.” *Bioinformatics* 21 (20): 7881. http://rocr.bioinf.mpi-sb.mpg.de.

Staniak, Mateusz, Przemyslaw Biecek, Krystian Igras, and Alicja Gosiewska. 2019. *localModel: LIME-Based Explanations with Interpretable Inputs Based on Ceteris Paribus Profiles*. https://CRAN.R-project.org/package=localModel.

Staniak, Mateusz, and Przemysław Biecek. 2018. *Live: Local Interpretable (Model-Agnostic) Visual Explanations*. https://CRAN.R-project.org/package=live.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. 2014. “Sequence to Sequence Learning with Neural Networks.” *CoRR* abs/1409.3215. http://arxiv.org/abs/1409.3215.

Štrumbelj, Erik, and Igor Kononenko. 2010. “An Efficient Explanation of Individual Classifications Using Game Theory.” *Journal of Machine Learning Research* 11 (March). JMLR.org: 1–18. http://dl.acm.org/citation.cfm?id=1756006.1756007.

Štrumbelj, Erik, and Igor Kononenko. 2014. “Explaining prediction models and individual predictions with feature contributions.” *Knowledge and Information Systems* 41 (3): 647–65. https://doi.org/10.1007/s10115-013-0679-x.

Tibshirani, Robert. 1994. “Regression Shrinkage and Selection Via the Lasso.” *Journal of the Royal Statistical Society, Series B* 58: 267–88.

Tufte, Edward R. 1986. *The Visual Display of Quantitative Information*. Cheshire, CT, USA: Graphics Press.

Tukey, John W. 1977. *Exploratory Data Analysis*. Addison-Wesley.

Venables, W. N., and B. D. Ripley. 2002. *Modern Applied Statistics with S*. Fourth. New York: Springer. http://www.stats.ox.ac.uk/pub/MASS4.

Wes, McKinney. 2012. *Python for Data Analysis*. 1st ed. O’Reilly Media, Inc.

Wickham, Hadley. 2009. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. http://ggplot2.org.

Wickham, Hadley, and Garrett Grolemund. 2017. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. 1st ed. O’Reilly Media, Inc.

Wikipedia. 2019. *CRISP DM: Cross-industry standard process for data mining*. https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining.

Wright, Marvin N., and Andreas Ziegler. 2017. *ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R*. *Journal of Statistical Software*. Vol. 77. https://doi.org/10.18637/jss.v077.i01.

Xie, Yihui. 2018. *bookdown: Authoring Books and Technical Documents with R Markdown*. https://CRAN.R-project.org/package=bookdown.