Chapter 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/PM_VEE/issues. Copyediting has not been done yet so read at your own risk.

We are primarily interested in the organization and consistency of the content, but any comments will be welcommed.

Thanks for taking the time to read this.

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

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 enitre 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 susspected 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, methods for verification, exploration, and explanation of predictive models are needed.

Figure 1.1 shows how the relative importance of domain understanding vs. modeling vs. validation is changing with model complexity. Simplest models are usually built on top of a good understanding of the domain. This allows identifying the most important variables that can be transformed into a predictive score. Today's machine learning is more focussed on the modeling. The effort is shifted from a deep understanding of the domain towards computationally heavy training of models. The validation part is of 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 in contrast with the goal of hypothesis testing. Hypotheses shall be stated in advance and obtained p-values shall not interfere in the way how data or models were prepared. Increasing automation in the EDA (Exploratory Data Analysis) and modeling part shift the focus towards the validation. 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. The feedback loop is even larger now, as the results from model validation are helping also in the domain understanding.

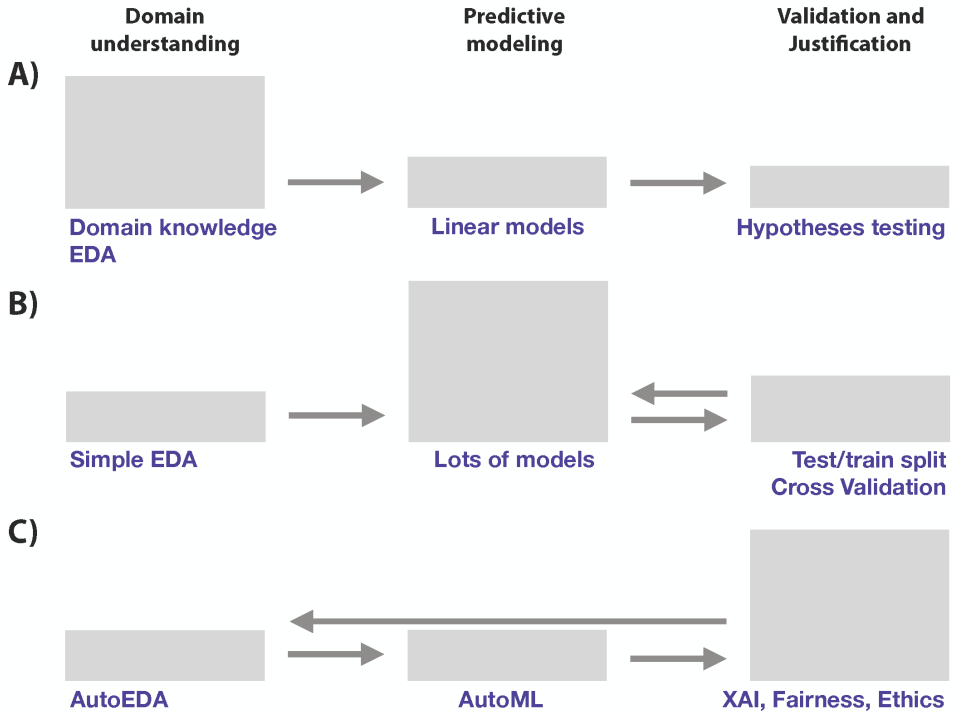


Figure 1.1: Shift in the relative importance and effort put in different phases of the data-driven modeling. (A) Statistical modeling is often based on deep understanding of the domain. Manual data exploration, consultations with domain experts, variable transformations lead to good models. Structures of models are in most cases simple 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.

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

Seventy-six years ago, 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 Artifical 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 like linear models, rule based models or classification trees with small number of parameters. However, the price for transparency may be a reduction in performance. Another way is to use tools that allow, perhaps by using approximations, to ,,explain" predictions for any model. In our book, we will focus on the latter approach.

1.4 The structure of the 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 *Global explainers*, we present techniques for exploration and explanation of model's performance for an entire dataset.

Before embarking on the description of the methods, in Chapter 2, we provide a short description of the process of data exploration and model assembly along with notation and definition of key concepts that areused 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.

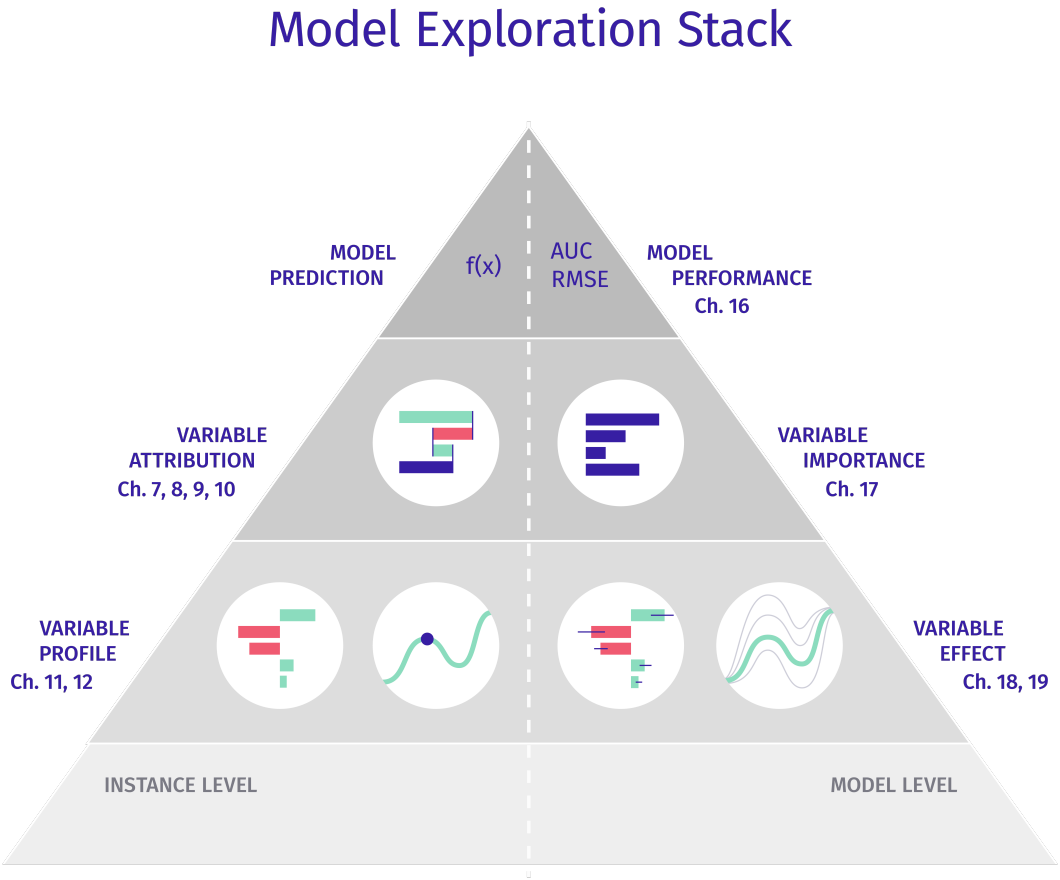


Figure 1.2: Stack with Model Explanation methods presented in this book. Left side is focused on instance level explanation while the right side is focused on model level explanation. Consecutive layers of the stack are linked with a deeper level of model exploration. These layers are linked with law's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

Figure 1.2 overviews the main part of the book. 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 for models including 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 paricular, in the chapter, the Local Interpretable Model-Agnostic Explanations (LIME) method (Ribeiro, Singh, and Guestrin 2016) is discussed. These chapters corresponds to the second layer of the stack in Figure 1.2.

In Chapters 11-13, methods based on Ceteris-paribus (CP) profiles are presented. The profiles show the change of model-based predictions induced by a change of a single variable; they are introduced in Chapter 11. Chapter 12 presents a CP-profile-based measure that summarizes the impact of a selected variable on model's predictions. The measure can be used to select the profiles that are worth plotting for a model with a large number of explanatory variables. Chapter 13 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 **Global explainers** part of the book consists of Chapters 15-20. These chapters present methods in the same order as appered in the Model Exploration Stack.

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 guideance 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, ,,global" 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 substract or divide these values.

In this book we focus on ,,black-box" models. We discuss them 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. 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 classess 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.

$$\text{BMI} = \frac{\text{mass}_{kg}}{{\text{height}_{m^2}}.$$

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.

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 cofficients.

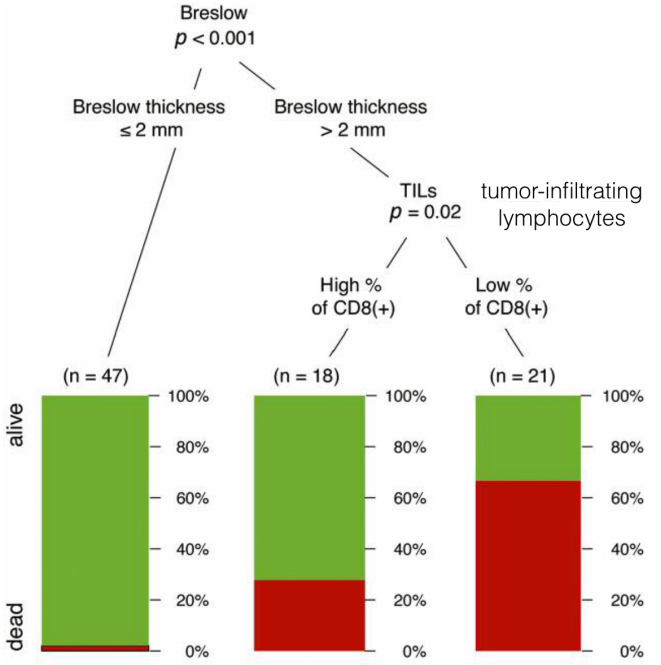


Figure 1.3: Example classification tree model for melanoma risk patients based on (Donizy et al. 2016). The model is based on two explanatory variables, Breslow thickness and Tumor infiltration lymphocytes. These two variables lead to three groups of paritents with different odds of survival.

Note that glass-box models, like the decision tree model presented in Figure 1.3 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 and 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 xgboost models (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 API that allow for quering remote models, like for aexmple in Microsoft Cognitive Services (Azure 2019). In that case, the only operation that we may be able to perform is 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 seq2seq 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 good starting pack 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-dependency 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 2013a).

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 2002a), `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. 2013a. “Applied Predictive Modeling.” New York, NY: Springer. 2013. http://www.amazon.com/Applied-Predictive-Modeling-Max-Kuhn/dp/1461468485/.

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. 2002a. “Classification and Regression by randomForest.” *R News* 2 (3): 18–22. https://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.

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.

Chapter 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 phase.

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 (Grolemund and Wickham 2019) and (Hall 2019). First is a very simple, focused on iteractions of three phases: data transformation, modeling and visualisation.

Figure 2.1 presents this iterative process divided into five steps. This is the common thinking about model development. Repeat until convergence, or repeat until best model is identified.

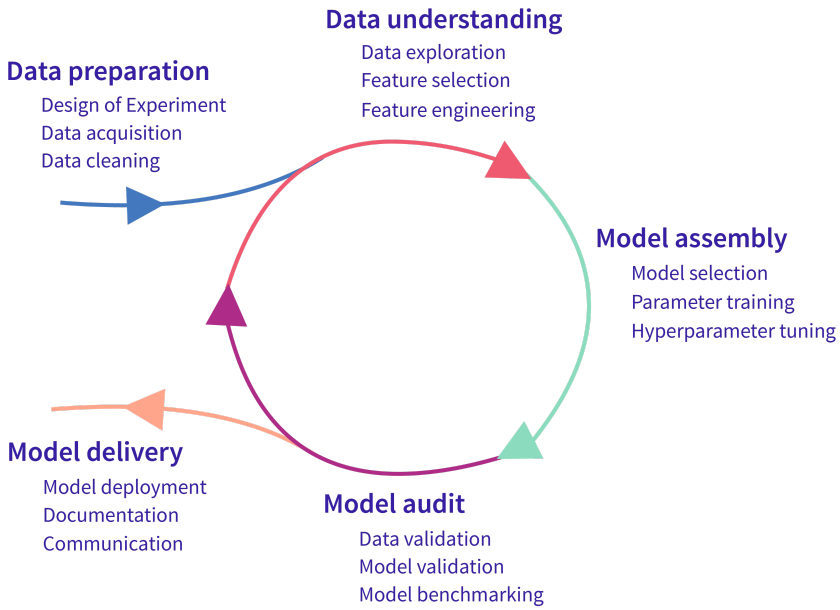


Figure 2.1: Lifecycle of predictive model can be decomposed into six tasks. First we need data that is poured into the model development. The model development is highly iterative, learn something new about the data, assemble a new model based on current understanding, and validate the new model. Repeat 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* introduced in (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 an 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 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, *crisp versions of models*. These initial versions of models are 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*.

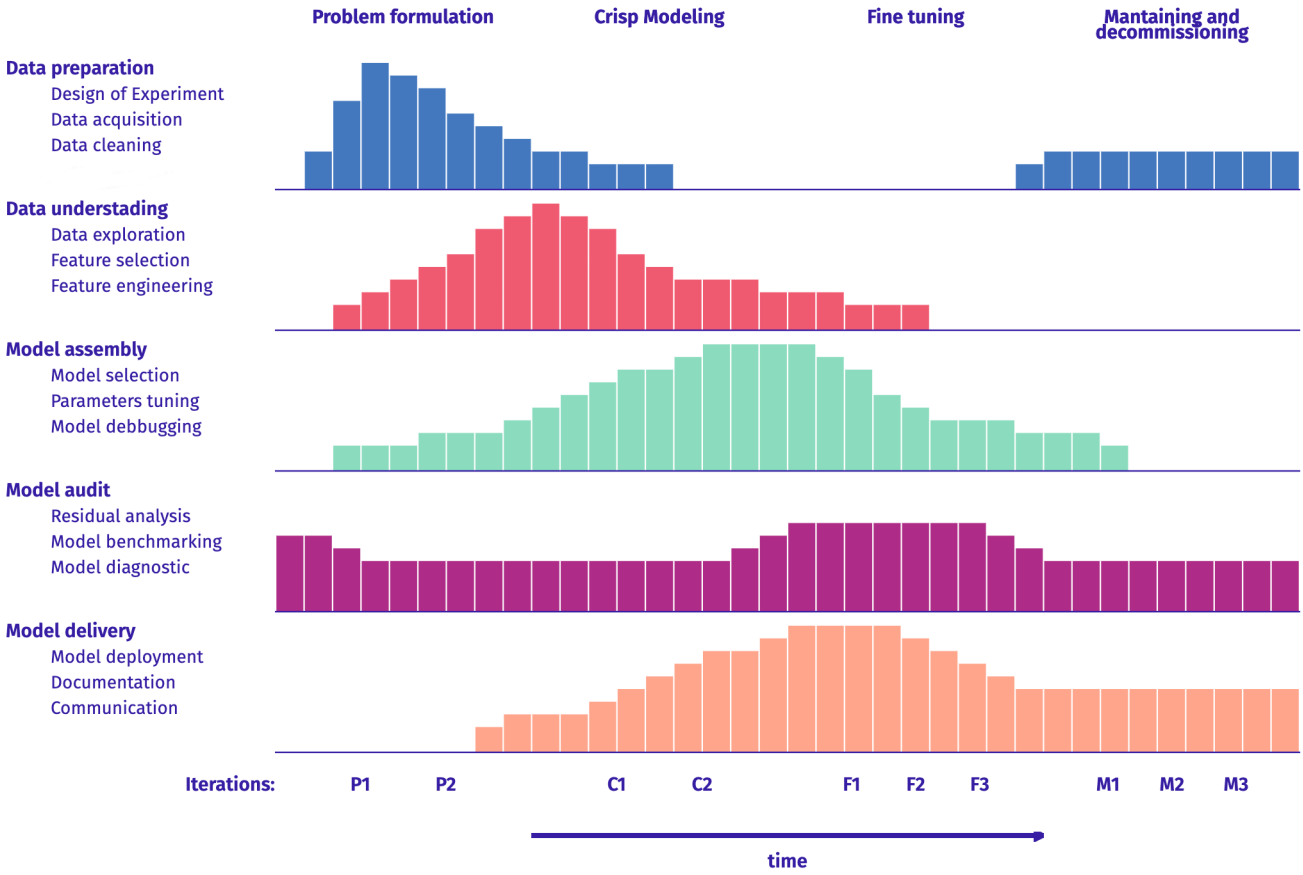


Figure 2.2: Overview of the Model Development Process. Horizontal axis show how time passes from the problem formulation to the model decommissioning. Vertical axis shows tasks are performed in a given phase. Each veritical strip is a next iteration of cycle presented in Figure @(fig:MDPwashmachine)

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 week 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 and one in Chapter 21. 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 entre book. In some cases this may result in formulae 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, $x_i \in \mathcal{X}$.

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 will refer to the j-th coordinate by using j in superscript. Thus, x_i^j deontes 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 $x^{j|=z} = (x^1, \ldots, x^{j-1}, z, x^{j+1}, \ldots, x^p)$.

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 multi-variate dependent variables; however, we use examples with uni-variate responses to simplify the notation. Typically, during the model development, we created many competing models. Formally we shall also index models to refer to a specific version of a trained model. But for the sake of simplicity we omit this index where it is not important.

We will use $r_i = y_i - f(x_i)$ we refer to the **model residual**, i.e., 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 observaton.

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 introduced new tools for data exploration, like for example boxplots for continuous variables. Availability of computational tools makes the process of data exploration easier and ore interactive. Find a good overview of techniques for data exploration in (Nolan and Lang 2015) or (*R for Data Science: Import, Tidy, Transform, Visualize, and Model Data* 2017).

In this book we will relay 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.

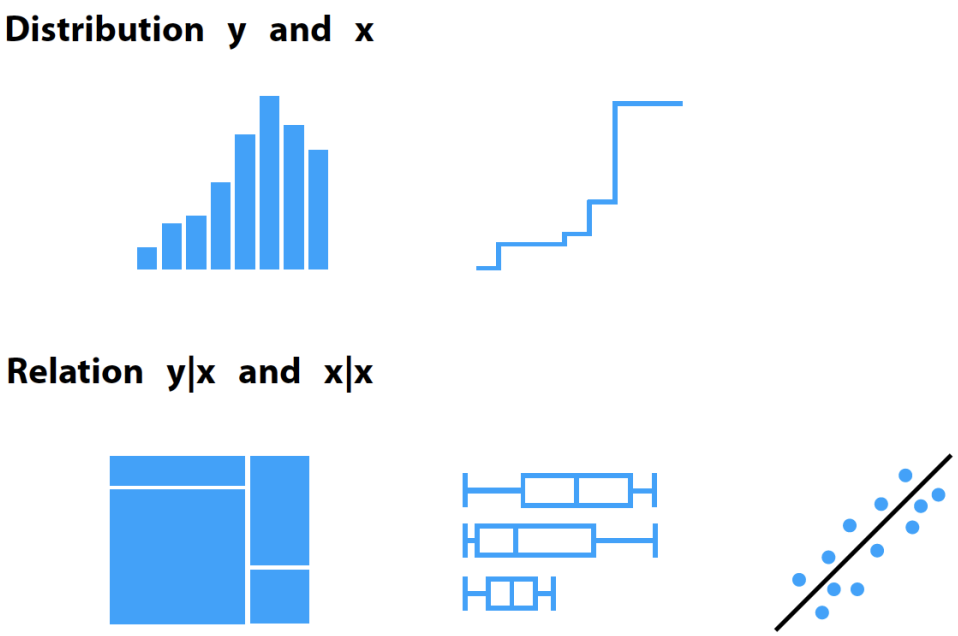


Figure 2.3: Basic methods for visual exploration. Histogram for distribution of continuous or categorical variables, empirical cumulative distribution for continuous variables. Mosaic plot for relation between two categorical variables, boxplots for relation between continuous and categorical variables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

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 (Kuhn and Johnson 2013b)). 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 Θ be a space for possible model parameters. Model training is a procedure of selection a θ that maximize some loss function L. For models with large parameter spaces it is common to add additional term $\lambda(\theta)$ that control the model complexity.

$$\hat{\theta} = \arg\min_{\theta \in \Theta} L(y, f_\theta(x)) + \lambda(\theta). \tag{2.1}$$

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 θ. Likelihood is probability of observing x as a function of parameter θ.

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 (2.1) become

$$\hat{\theta} = \arg\min_{\theta \in \Theta} ||y - x\beta||_2 + \lambda(\beta). \tag{2.2}$$

For linear regression, the penalty term $\lambda(\beta)$ is equal to 0, and optimal parameters β in equation (2.2) have close analytical solution. In ridge regression the penalty $\lambda(\beta) = ||\beta||_2$ and also (2.2) have analytical solution. For LASSO regression the penalty $\lambda(\beta) = ||\beta||_1$ and β 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 (Venables and Ripley 2010) or (Kuhn and Johnson 2013b).

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 Dependency 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: An Introduction*.

Kuhn, Max, and Kjell Johnson. 2013b. *Applied Predictive Modeling*. 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.

R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. 2017. 1st ed. O'Reilly Media, Inc.

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

Venables, W. 2010. *Modern Applied Statistics with S*. Springer Publishing Company, Incorporated.

Wikipedia. 2019. *CRISP Dm: Cross-Industry Standard Process for Data Mining*. https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining.

Chapter 3 Do-it-yourself With R

In our book we introduce different methods for instance-level and global explanation and exploration of predictive models. In each chapter, there is a section with code snippets for R that show how a particular method has been implemented. In this chapter we provide a short description of steps that will allow the reader to replicate the results presented for various methods.

3.1 What to install?

Obviously, R (R Core Team 2018) is needed. It is always better to use the newest version, but at least R in version 3.5 should be used. R can be downloaded from 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. 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 2002b), `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 constucting 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 `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 (Biecek 2018) 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 5.1.6.

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 R models and 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.

3.4 DrWhy Packages

Here we present list of arguments in explainers from `DrWhy` universe. All explainers use unified set of arguments. All of them are generic with two specific implementations `*.explainer` and `*.default`. The first one is working for objects created with `DALEX::explain()` function.

Common core arguments

- `x` a model to be explained, or an explainer created with function `DALEX::explain()`.
- `data` validation dataset. Used to determine univariate distributions, calculation of quantiles, correlations and so on. It will be extracted from `x` if it's an explainer.
- `predict_function` predict function that operates on the model `x`. Since the model is a black box, the `predict_function` is the only interface to access values from the model. It should be a function that takes at least a model `x` and `data` and returns vector of predictions. If model response has more than a single number (like multiclass

models) then this function should return a marix/data.frame of the size `m x d`, where `m` is the number of observations while `d` is the dimensionality of model response. It will be extracted from `x` if it's an explainer.

- `new_observation` an observation/observations to be explained. Required for local/instance level explainers. Columns in should correspond to columns in the `data` argument.
- `...` other parameters.
- `label` name of the model. By default it's extracted from the `class` attribute of the model

Function specific arguments

- `keep_distributions` if `TRUE`, then distributions of partial predictions is stored and can be plotted with the generic `plot()`.

References

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

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.

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. 2002b. "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.

Chapter 5 Data sets and models

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

- *Sinking of the RMS Titanic*
- *Apartment prices*

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 data are available in a dataset included in the `stablelearner` package. The dataset, after some data cleaning and variable transformations, is also avaliable 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);
- *age*, person's age in years, a numerical variable; for adults, the age is given in (integer) years; for children younger than one year, the age is given as $x/12$, where x is the number of months of child's age;
- *class*, the class in which the passenger travelled, or the duty class of a crew member; a factor with seven levels
- *embarked*, the harbor in which the person embarked on the ship, a factor with four levels;
- *country*, person's home country, a factor with 48 levels;
- *fare*, the price of the ticket (only available for passengers; 0 for crew members), a numerical variable;
- *sibsp*, the number of siblings/spouses aboard the ship, a numerical variable;
- *parch*, the number of parents/children aboard the ship, a numerical variable;
- *survived*, a factor with two levels 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
```

```
str(titanic)
```

```
## 'data.frame':    2207 obs. of  9 variables:
##  $ gender  : Factor w/ 2 levels "female","male": 2 2 2 1 1 2 2 1 2 2 ...
##  $ age     : num  42 13 16 39 16 25 30 28 27 20 ...
##  $ class   : Factor w/ 7 levels "1st","2nd","3rd",..: 3 3 3 3 3 3 2 2 3 3 ...
##  $ embarked: Factor w/ 4 levels "Belfast","Cherbourg",..: 4 4 4 4 4 4 2 2 2 4
##  $ country : Factor w/ 48 levels "Argentina","Australia",..: 44 44 44 15 30 4
##  $ fare    : num  7.11 20.05 20.05 20.05 7.13 ...
##  $ sibsp   : num  0 0 1 1 0 0 1 1 0 0 ...
##  $ parch   : num  0 2 1 1 0 0 0 0 0 0 ...
##  $ survived: Factor w/ 2 levels "no","yes": 1 1 1 2 2 2 1 2 2 2 ...
```

```
levels(titanic$class)
```

```
## [1] "1st"             "2nd"             "3rd"             "deck crew"
## [5] "engineering crew" "restaurant staff" "victualling crew"
```

```
levels(titanic$embarked)
```

```
## [1] "Belfast"     "Cherbourg"   "Queenstown"  "Southampton"
```

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 average (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 class average
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$class == "3rd"] = 13
# missing sibsp, parch are replaced by 0
titanic$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. Figures 5.2-5.5 present graphically the proportion non- and survivors for different levels of the other variables. 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.

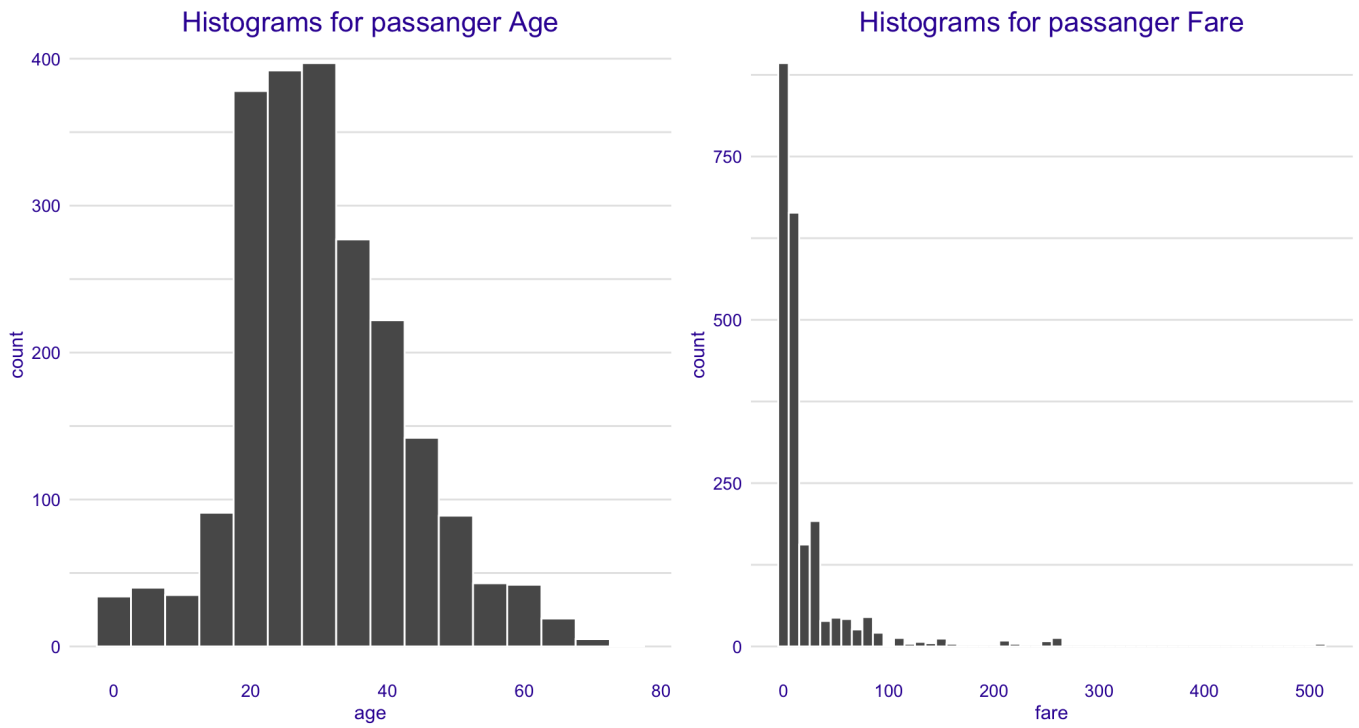


Figure 5.1: Histogram of Age and Fare for the Titanic data.

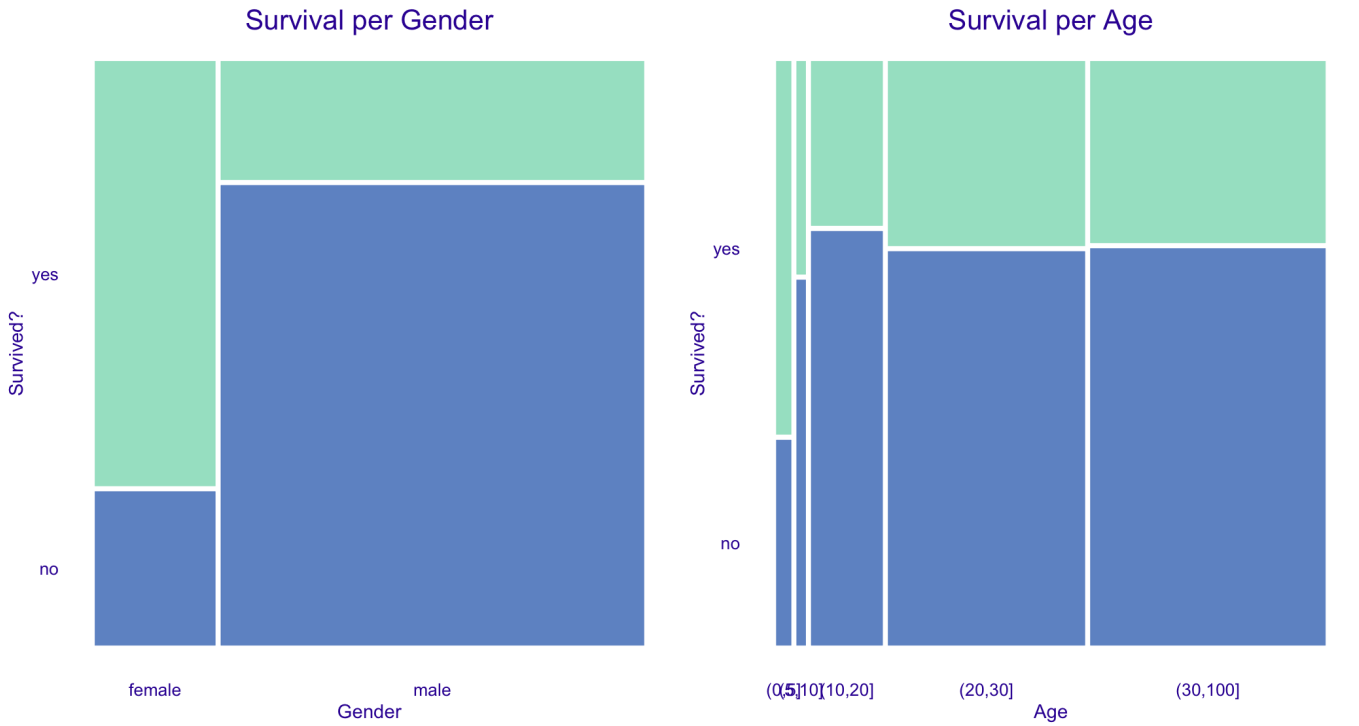


Figure 5.2: Survival status in group defined be Gender and Age for the Titanic data.

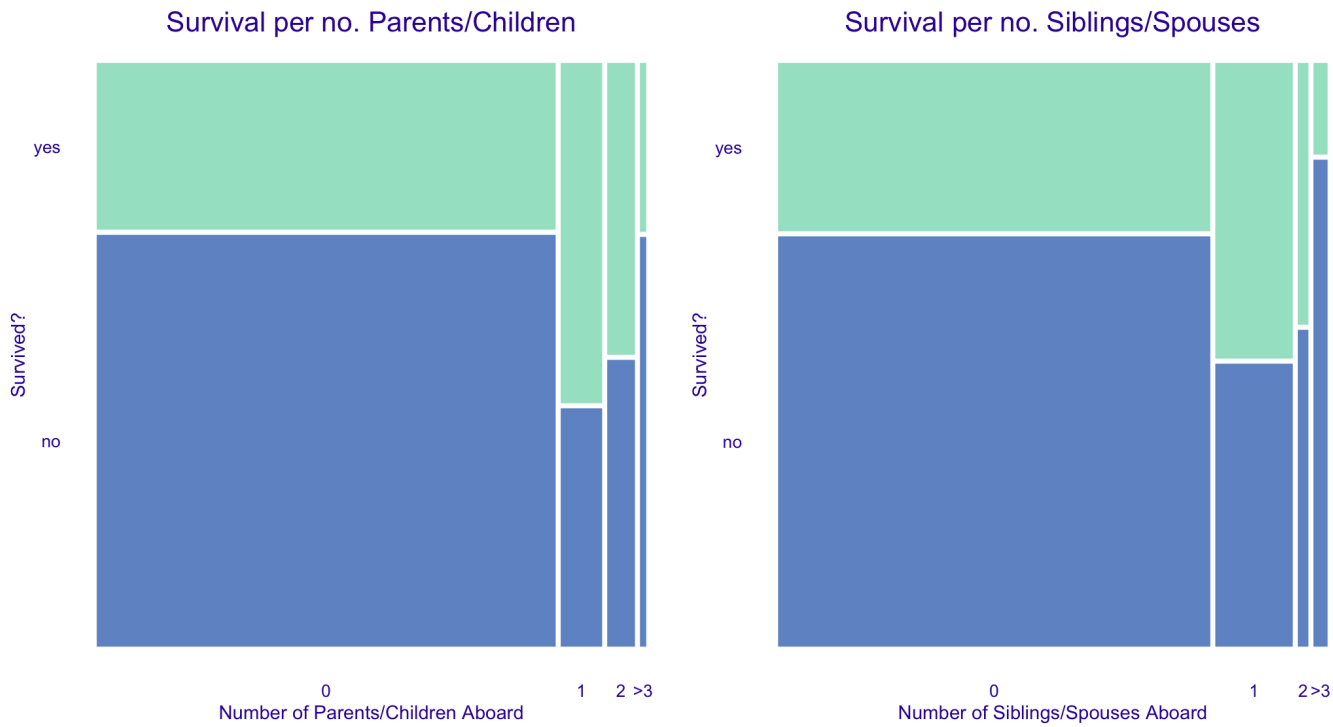


Figure 5.3: Survival according to the number of parents/children and siblings/spouses in the Titanic data.

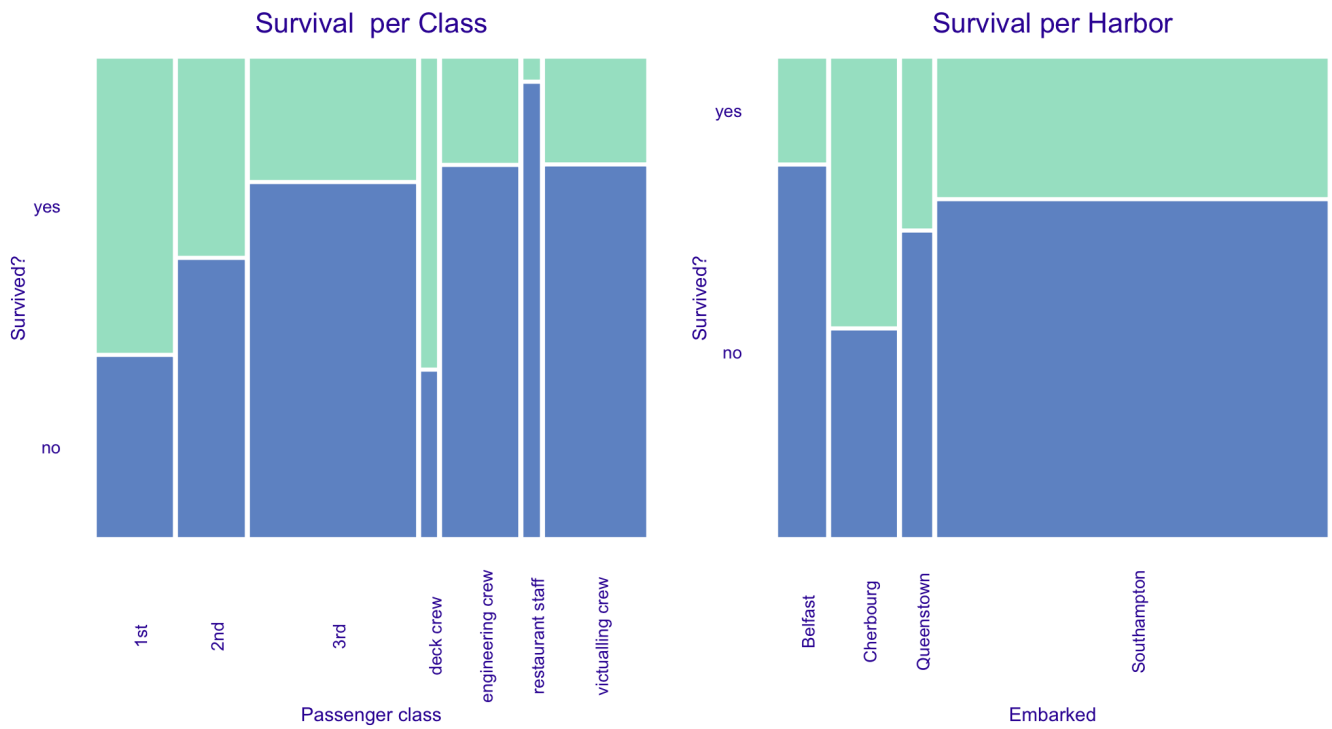


Figure 5.4: Survival according to the class and port of embarking in the Titanic data.

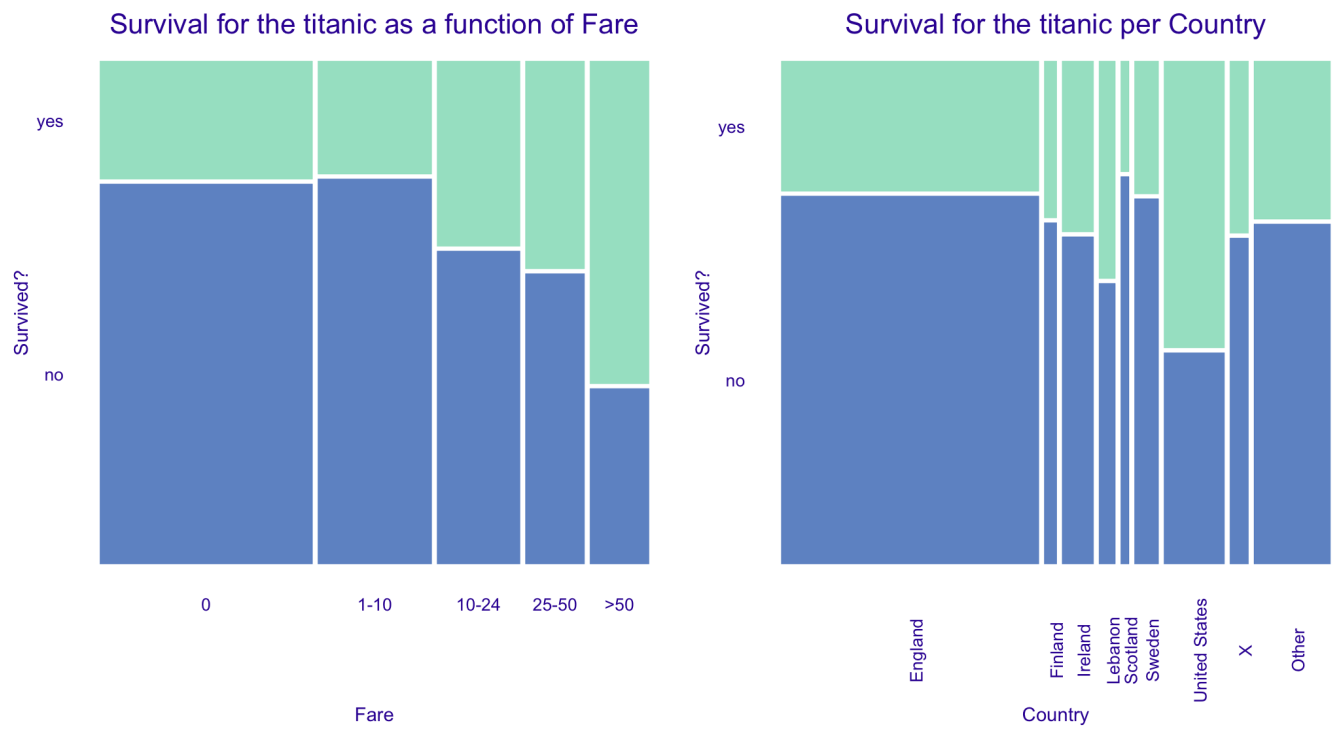


Figure 5.5: Survival according to fare and country in the Titanic data.

5.1.2 Logistic regression

The dependent variable of interest, *survival*, is binary. Thus, a natural choice to build a predictive model is logistic regression. We do not consider country as an explanatory variable. 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
##
```

5.1.3 Random forest

As an alternative to a logistic regression model, we consider a random forest model. Random forest is known for good predictive performance, is able to grasp low-level 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 2002a).

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 + f
                              data = titanic)
titanic_rf_v6
```

```
##
## Call:
##  randomForest(formula = survived ~ class + gender + age + sibsp +      parch -
##                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

Finally, we consider the gradient-boosting model. (Friedman 2000) The model is known for being able to accomodate higher-order interactions between variables. We use the same set of 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 +
                      data = titanic, n.trees = 15000)
```

```
## Distribution not specified, assuming bernoulli ...
```

```
titanic_gbm_v6
```

```
## gbm(formula = survived == "yes" ~ class + gender + age + sibsp +
##     parch + fare + embarked, 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 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 data frame `johny_d` that contains the data describing the passenger.

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

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

```
(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.

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

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

```
(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", "e
            gender = factor("male", levels = c("female", "male")),
            age = 47,
            sibsp = 0,
            parch = 0,
            fare = 25,
            embarked = factor("Cherbourg", levels = c("Belfast","Cherbourg","Quee
)
round(predict(titanic_lmr_v6, henry, type = "fitted"),2)
```

```
##    1
## 0.43
```

```
round(predict(titanic_rf_v6, henry, type = "prob")[1,2],2)
```

```
## [1] 0.25
```

```
round(predict(titanic_gbm_v6, henry, type = "response", n.trees = 15000),2)
```

```
## [1] 0.31
```

5.1.6 Explainers

Model-objects created with different libraries may have different internal structures. Thus, first, we have got to create a wrapper around the model. 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`, a function that returns prediction scores; if not specified, then it is extracted from the `class(model)`. In the example below we create explainers for the logistic regression, random forest, and gradient boosting models created for the Titanic data.

Each explainer wraps 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$survived == "yes",
                                  predict_function = function(m, x) predict(m, x,
                                  label = "Logistic Regression v6")
explain_titanic_rf_v6 <- explain(model = titanic_rf_v6,
                                  data = titanic[, -9],
                                  y = titanic$survived == "yes",
                                  label = "Random Forest v6")
explain_titanic_rf_v3 <- explain(model = titanic_rf_v3,
                                  data = titanic[, -9],
                                  y = titanic$survived == "yes",
                                  label = "Random Forest v3")
explain_titanic_gbm_v6 <- explain(model = titanic_gbm_v6,
                                  data = titanic[, -9],
                                  y = titanic$survived == "yes",
                                  predict_function = function(m, x) predict(m, x,
                                  label = "Generalized Boosted Regression v6")
```

5.1.7 List of objects for the `titanic` example

In the previous sections we have built several 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 recreate a selected model in a 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/ceb40")` . Get the explainer: `archivist::aread("pbiecek/models/51c50")`
`titanic_rf_v6`	`randomForest::randomForest` v.4.6.14	gender, age, class, sibsp, parch, fare, embarked	Get the model: `archivist::aread("pbiecek/models/1f938")` . Get the explainer: `archivist::aread("pbiecek/models/42d51")`
`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/0e5d2")`
`titanic_gbm_v6`	`gbm::gbm` v.2.1.5	gender, age, class, sibsp, parch, fare, embarked	Get the model: `archivist::aread("pbiecek/models/24e72")` . Get the explainer: `archivist::aread("pbiecek/models/3d514")`

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 Warszawa, 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*, apatments price per meter-squared (in EUR), a numerical variable;
- *construction.year*, the year of construction of the block of flats in which the apartment is located, a numerical variable;

- *surface*, apartment's total surface in squared meters, a numerical variable;
- *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;
- *distric*, a factor with 10 levels indicating tha distric of Warszawa 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
```

```
str(apartments)
```

```
## 'data.frame':    1000 obs. of  6 variables:
##  $ m2.price         : num  5897 1818 3643 3517 3013 ...
##  $ construction.year: num  1953 1992 1937 1995 1992 ...
##  $ surface          : num  25 143 56 93 144 61 127 105 145 112 ...
##  $ floor            : int  3 9 1 7 6 6 8 8 6 9 ...
##  $ no.rooms         : num  1 5 2 3 5 2 5 4 6 4 ...
##  $ district         : Factor w/ 10 levels "Bemowo","Bielany",..: 6 2 5 4 3 6 3
```

```
table(apartments$floor)
```

```
##
##   1   2   3   4   5   6   7   8   9  10
##  90 116  87  86  95 104 103 103 108 108
```

```
table(apartments$no.rooms)
```

```
##
##   1   2   3   4   5   6
##  99 202 231 223 198  47
```

```
levels(apartments$district)
```

```
##  [1] "Bemowo"      "Bielany"     "Mokotow"     "Ochota"      "Praga"
##  [6] "Srodmiescie" "Ursus"       "Ursynow"     "Wola"        "Zoliborz"
```

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`, also included in the `DALEX` package.

```
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.

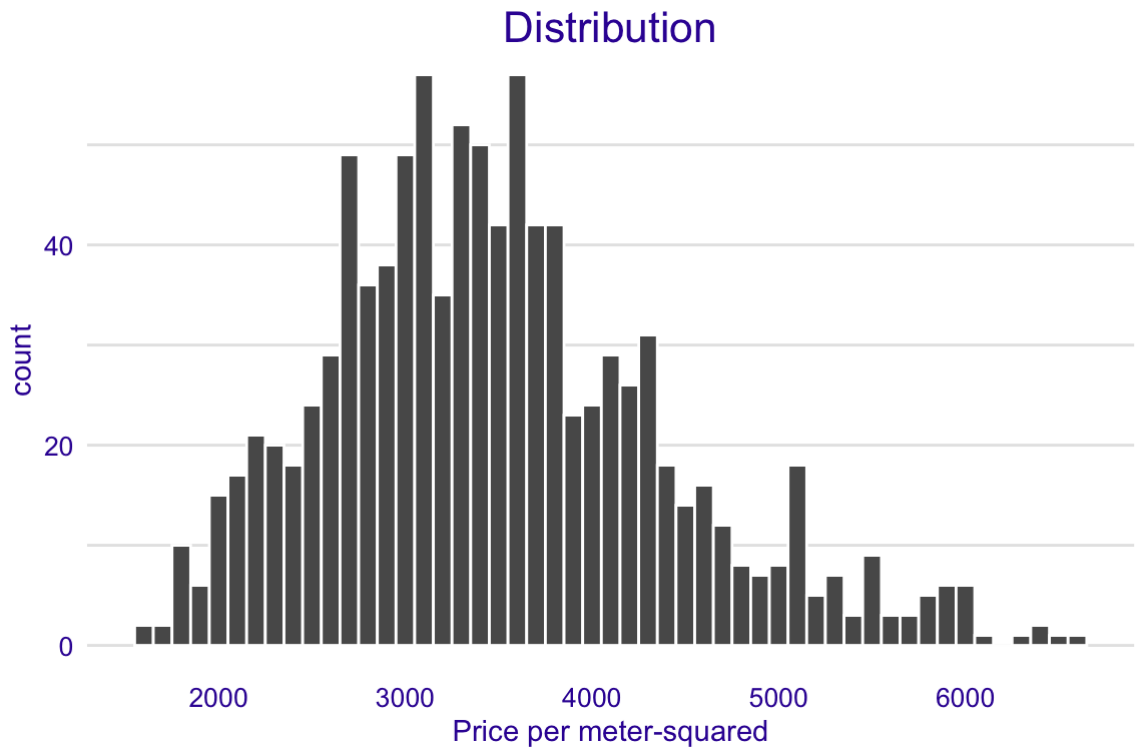


Figure 5.6: (fig:appartmentsExplorationMi2) Distribution of the price per meter-squared in the apartments data.

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

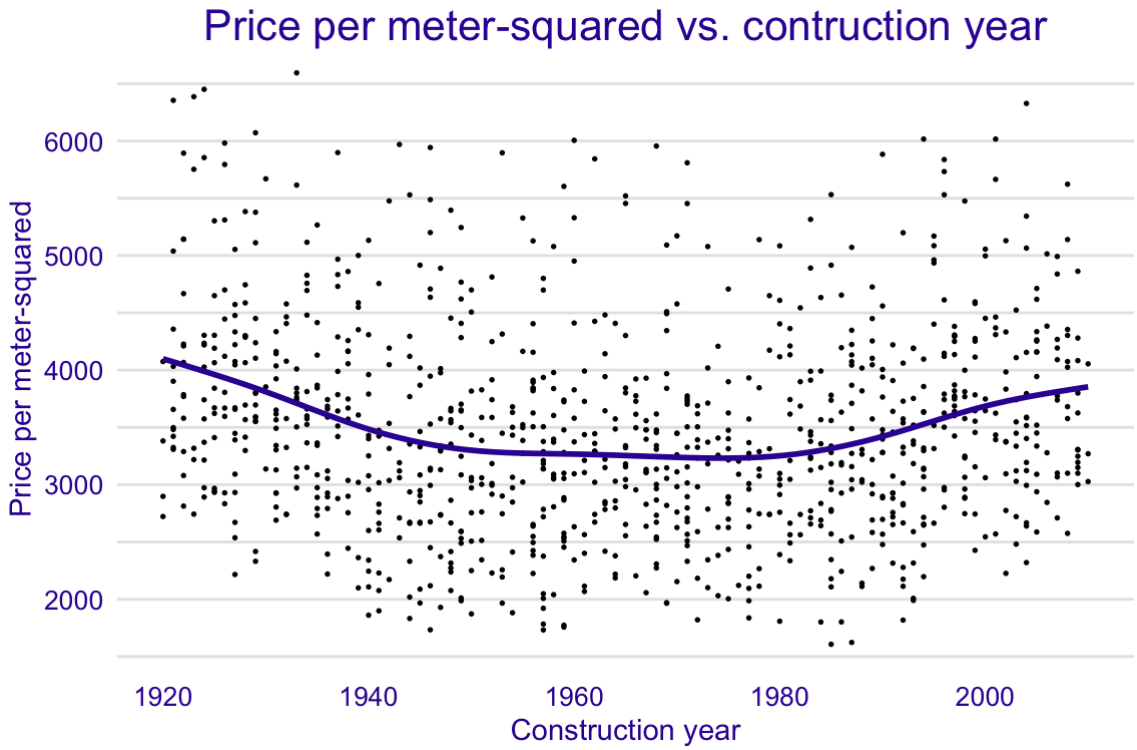


Figure 5.7: (fig:appartmentsMi2Construction) Price per meter-squared vs. construction year

Figure 5.8 indicates a linear relation between *surface* and *m2.price*.

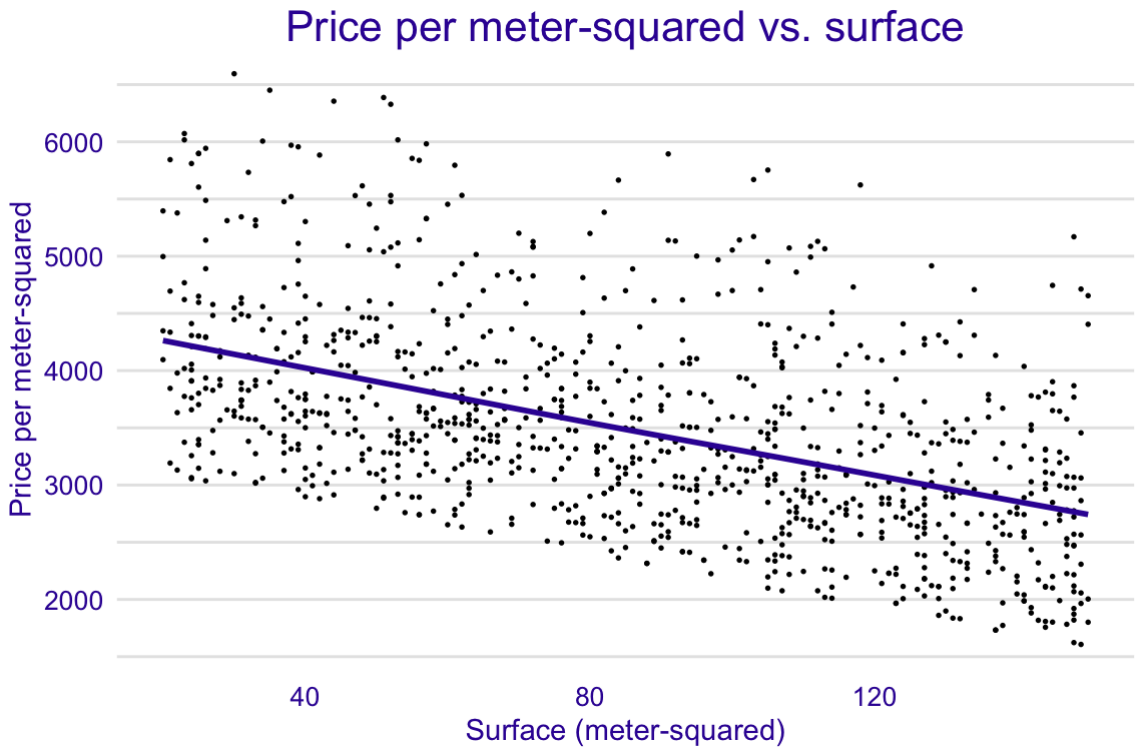


Figure 5.8: (fig:appartmentsMi2Surface) Price per meter-squared vs. surface

Relation between *floor* and *m2.price* is also close to linear, as seen in Figure 5.9.

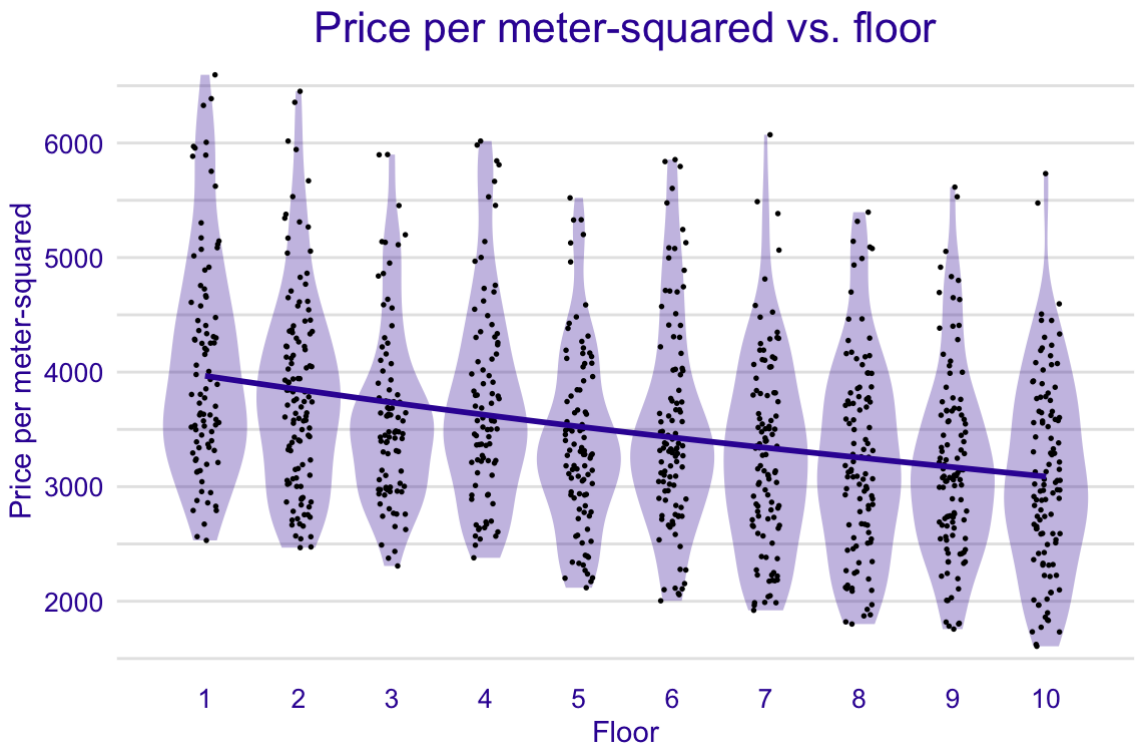


Figure 5.9: (fig:appartmentsMi2Floor) Price per meter-squared vs. floor

There is a close to linear relation between *no.rooms* and *m2.price*, as suggested by Figure 5.10. It is worth noting that, quite naturally, surface and number of rooms are correlated (see Figure 5.11).

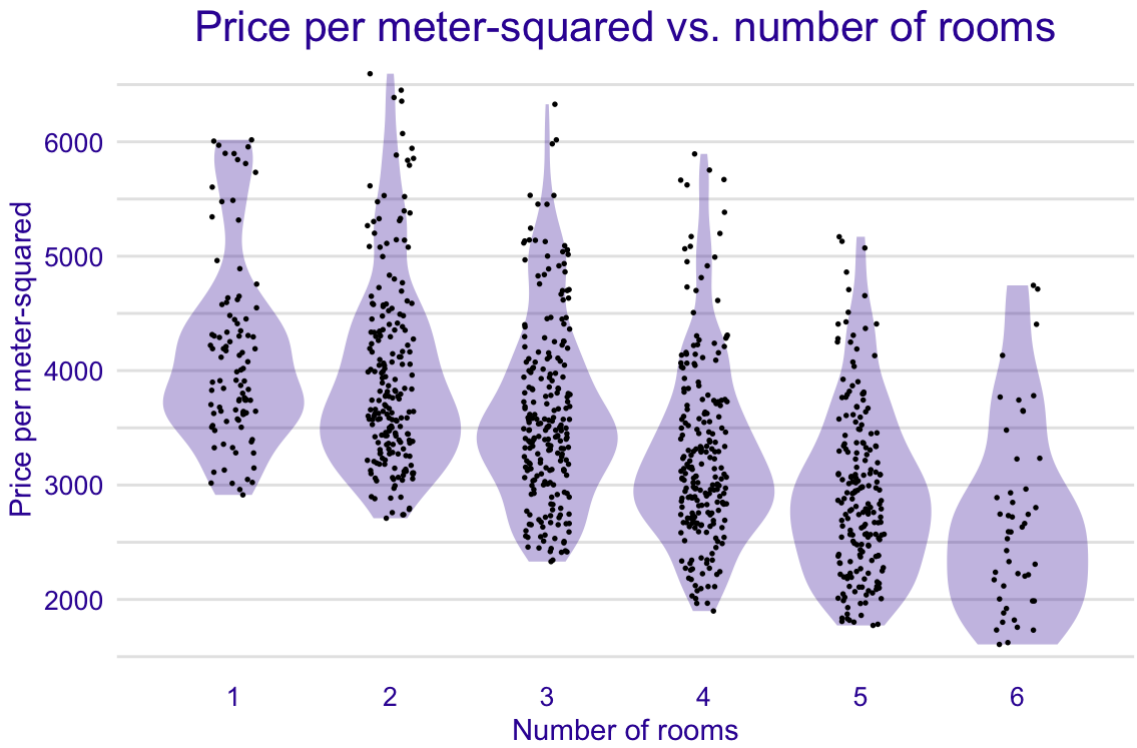


Figure 5.10: (fig:appartmentsMi2Norooms) Price per meter-squared vs. number of rooms

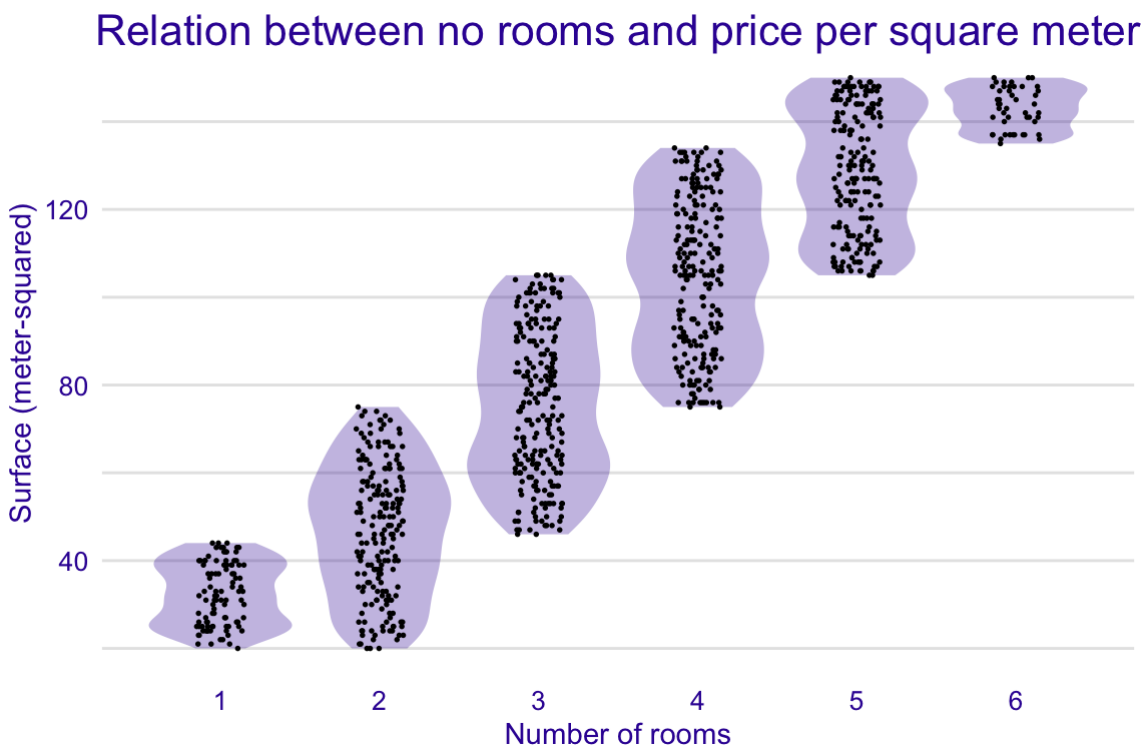


Figure 5.11: (fig:appartmentsSurfaceNorooms) Surface vs. number of rooms

Prices depend on district. Violin plots in Figure 5.12 indicate that the highest prices per meter-squared are observed in Srodmiescie (Downtown).

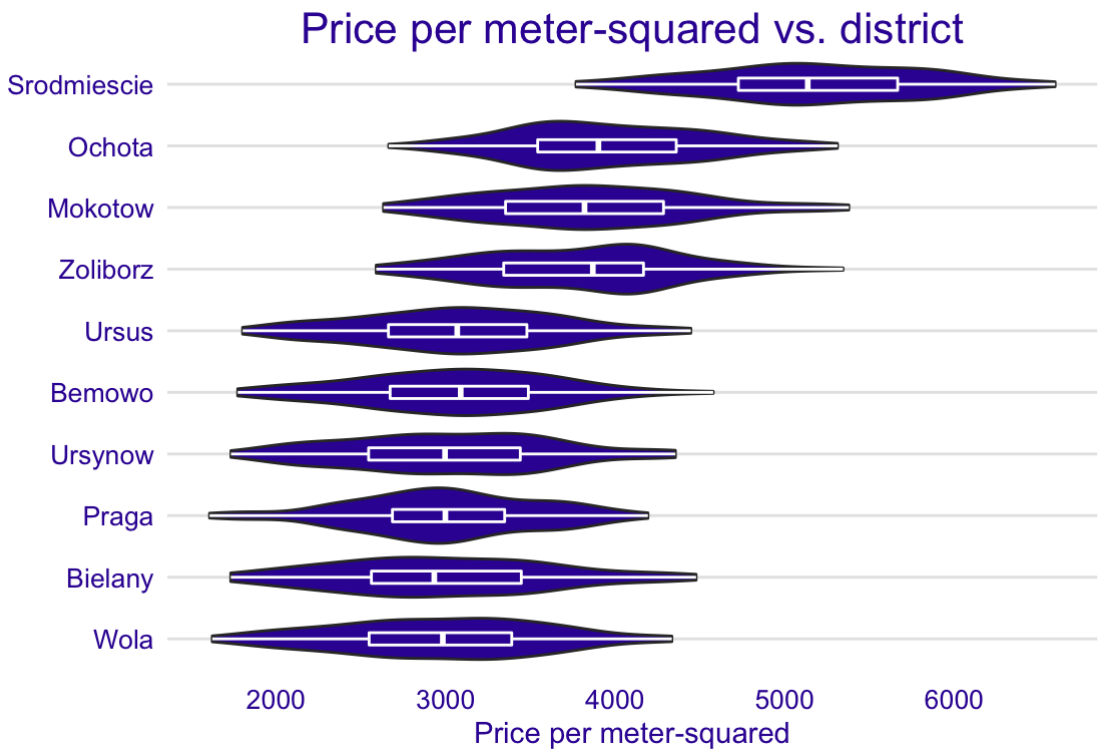


Figure 5.12: (fig:appartmentsMi2District) Price per meter-squared for different districts

5.2.2 Linear regression

The dependent variable of interest, *m2.price*, is continuous. Thus, a natural choice to build a predictive model is linear regression. We treat all the other variables in the `apartments` data frame 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)
apartments_lm_v5
```

```
##
## Call:
## lm(formula = m2.price ~ ., data = apartments)
##
## Coefficients:
##         (Intercept)    construction.year              surface
##            5003.248              -0.229             -10.238
##                floor             no.rooms      districtBielany
##             -99.482             -37.730              34.106
##       districtPraga      districtUrsynow       districtBemowo
##             -20.214              -1.974              16.891
##       districtUrsus     districtZoliborz      districtMokotow
##              46.833             906.865             935.271
##      districtOchota  districtSrodmiescie
##             943.145            2097.502
```

5.2.3 Random forest

As an alternative 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 2002a).
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 Model predictions

By aplying the `predict()` function to model-object `apartments_lm_v5` with `apartments_test` as the data frame for which predictions are to be computed, we obtain the predicted prices for the testing set of six apartments for the linear regression model. Subsequently, we compute the mean squared difference between the predicted and actual prices for the test apartments. We repeat the same steps for the random forest model.

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

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

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

```
## [1] 282.9519
```

For the random forest model, the square-root of the mean squared difference is equal to 283. It is only minimally smaller than the value of 283.1, obtained for the linear regression model. 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.

5.2.5 Explainers

In similar spirit to the Section 5.1.6 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$m2.price,
                                    label = "Linear Regression v5")
explain_apartments_rf_v5 <- explain(model = apartments_rf_v5,
                                    data = apartments_test,
                                    y = apartments_test$m2.price,
                                    label = "Random Forest v5")
```

5.2.6 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 recreate 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/f49ea")`
`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/569b0")`

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.

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. 2002a. “Classification and Regression by randomForest.” *R News* 2 (3): 18–22. https://CRAN.R-project.org/doc/Rnews/.

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

Chapter 6 Introduction to Instance Level Exploration

Instance-level explainers 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 patient's score?
- 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 experiences heart attack. What has driven that prediction?

A model is a function with a p-dimensional vector x as an argument. The plot of the value(s) of the function can be constructed in a $p+1$-dimensional space. An example with $p=2$ is presented in Figure 6.1. We will use it as an illustration of key ideas. The plot provides an information about the values of the function in the vicinity of point x^*.

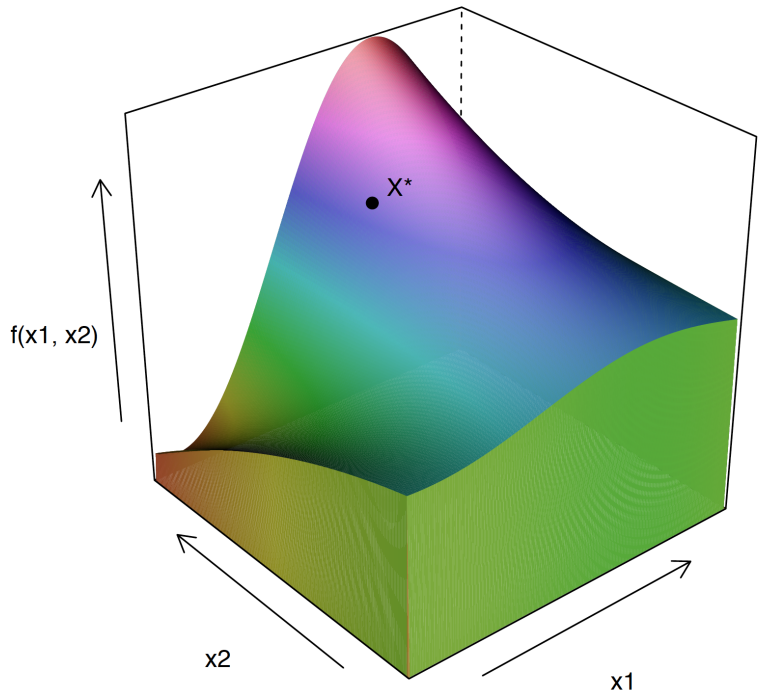


Figure 6.1: (fig:cutsSurfaceReady) Response surface for a model that is a function of two variables. We are interested in understanding the response of a model in a single point x*

There are many different tools that may be used to explore the predictions of the model around a single point x^*. In the following sections we will describe the most popular approaches. They can be divided into three classes.

- One 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 A of Figure 6.2. Chapters 11-13 introduce the CP profiles and methods based on them.
- Another 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 B of Figure 6.2. Chapters 7-9 present various methods implementing this approach.
- Yet another approach is to analyze the curvature of the response surface (see Figure 6.1) 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 C of Figure 6.2. Chapter 10 presents the Local Interpretable Model-agnostic Explanations (LIME) method that exploits the concept of a ,,local model."

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

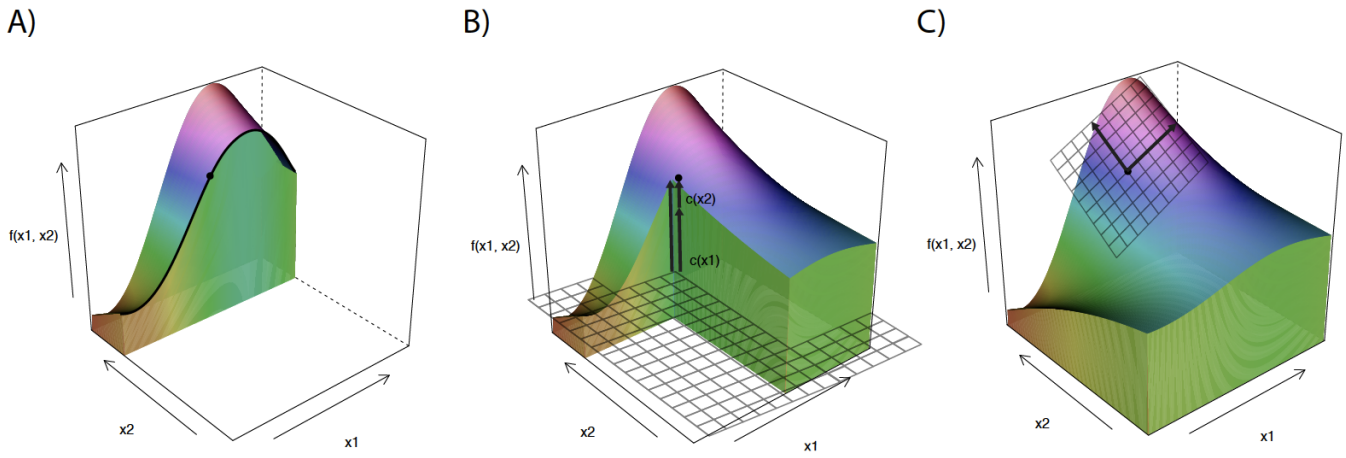


Figure 6.2: (fig:cutsTechnikiReady) Illustration of different approaches to instance-level explanation. Panel A presents a What-If analysis with Ceteris-paribus profiles. The profiles show the model response as a function of a value of a single variable, while keeping the values of all other explanatory variables fixed. Panel B illustrates the concept of variable attributions. Additive effects of each variable show how the model response differs from the average. Panel C illustrates the concept of local models. A simpler glass-box model is fitted around the point of interest. It describes the local behaviour of the black-box model.

Chapter 7 Break-down Plots for Additive Variable Attributions

In Chapter 12, we introduced a method for assessment of local variable-importance based on Ceteris-paribus (CP) profiles. The main disadvantage of this method is that the sum of the developed importance scores does not equal the final model prediction.

In this chapter we introduce Break-down (BD) plots, which offer a solution to this problem. BD plots show "variables attributions" i.e., the decomposition of the difference between the single-instance and the average model predictions among the different explanatory variables. Note that the method is similar to the `EXPLAIN` algorithm introduced in (Robnik-Šikonja and Kononenko 2008) and implemented in the `ExplainPrediction` package (Robnik-Šikonja 2018).

7.1 Intuition

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

The idea is illustrated in Figure 7.1. Consider the prediction for `johny_d` for the random-forest model (see Section 5.1.3) for the Titanic data. Panel A shows distribution of model predictions. The row `all data` shows the distribution 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.

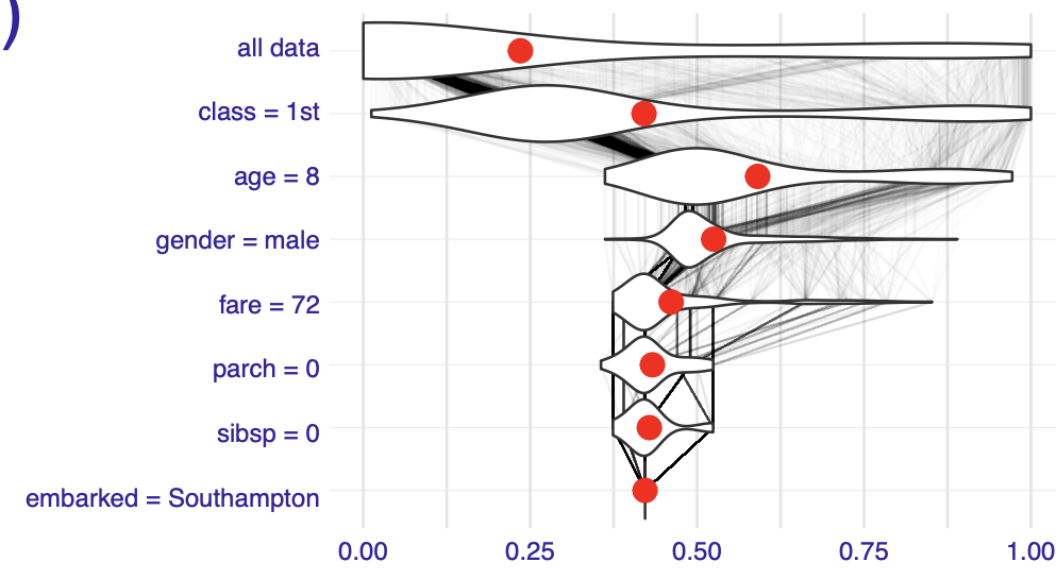
To evaluate the contribution of the explanatory variables to the particular instance prediction, we consider the predictions when fixing the values of the variables. For instance, the row `class=1st` in Panel A of Figure 7.1 presents the distribution of the predictions obtained when the value of the `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. The last row corresponds to the prediction for `model response for` johny_d`.

The black lines in Panel A show how the inidividual predictions change after the value of the j-th variable has been replaced by the value indicated in the name of the row.

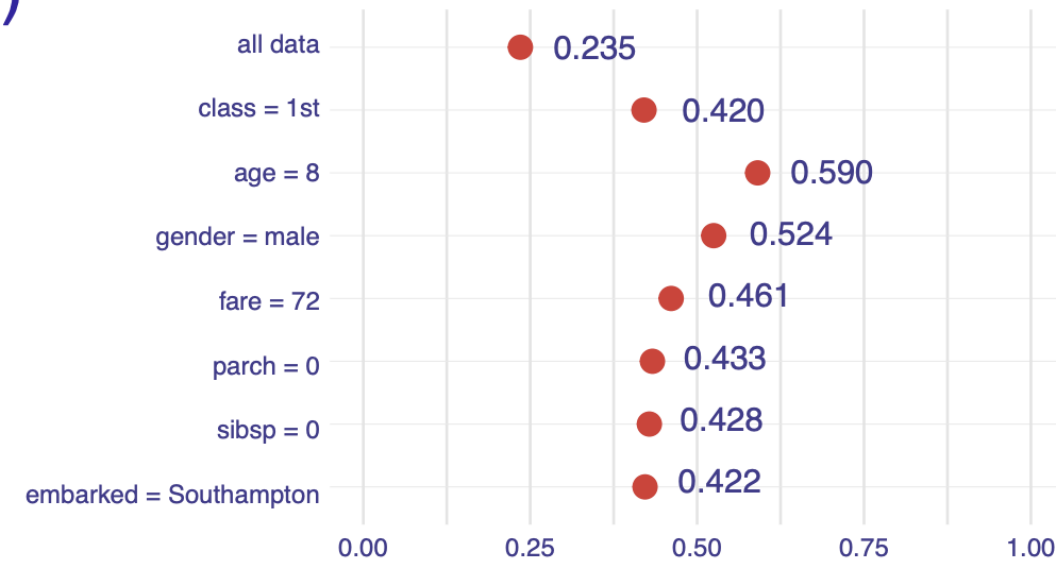
Eventually, however, we may be interested in the average predictions, as indicated in Panel B of Figure 7.1, or even only in the changes of the averages, as shown in Panel C. In Panel C, positive changes are presented with green bars, while negative differences are marked with red bar. The changes sum 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.

A)


all data
class = 1st
age = 8
gender = male
fare = 72
parch = 0
sibsp = 0
embarked = Southampton
0.00
0.25
0.50
0.75
1.00

B)


all data
0.235
class = 1st
0.420
age = 8
0.590
gender = male
0.524
fare = 72
0.461
parch = 0
0.433
sibsp = 0
0.428
embarked = Southampton
0.422
0.00
0.25
0.50
0.75
1.00

C)

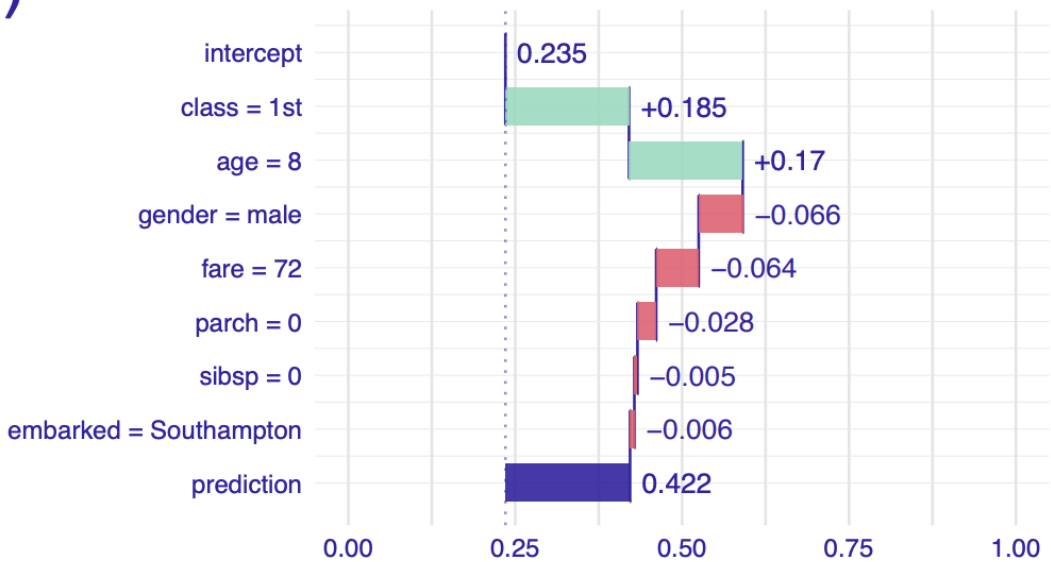

intercept
0.235
class = 1st
+0.185
age = 8
+0.17
gender = male
−0.066
fare = 72
−0.064
parch = 0
−0.028
sibsp = 0
−0.005
embarked = Southampton
−0.006
prediction
0.422
0.00
0.25
0.50
0.75
1.00

Figure 7.1: (fig:BDPrice4) 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 dirstribution and the average of the predictions when fixing values of subseqeunt explanatory variables. The last row shows the prediction for a particular instance of interest. B) Red dots indicate the average predictions from Panel B. C) The green and red bars indicate, resspectively, positive and negative changes in the average predictions (variable contributions).

7.2 Method

First, let's see how variable attribution works for linear models.

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 $\beta = (\beta^0, \beta^1, \ldots, \beta^p)$, where β^0 is the intercept. The prediction for Y at point $X = x = (x^1, x^2, \ldots, x^p)$ 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:

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

We are interested in the contribution of the i-th explanatory variable to model prediction $f(x_*)$ for a single observation described by x_*. In this case, the contribution is equal 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 is centered by subtracting a constant $\hat{x}^i$ (usually, the mean of x^i). This leads the following, intuitive formula for the variable attribution:

$$v(i, x_*) = \beta_i(x^i_* - \hat{x}^i).$$

We want to calculate $v(f, x_*, i)$, which 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 $v(f, x_*, i)$ 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, $v(f, x_*, i)$ can be expressed as follows:

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

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

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

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

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

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

Thus, the contributions of the explanatory variables are the differences between the model prediction for x_* and the average 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 dataset. 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 model coefficients and we work with an estimated model.

7.2.2 Break-down for general case

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 variable-importance measures for all explanatory variables to be equal to the instance prediction (property called *local accuracy*), so that

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

where v_0 denotes the average model response. If we re-write the equation above as follows:

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

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 $J \cap L = \varnothing$. 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 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

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

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 $J = \varnothing$, then

$$\Delta^{l|\varnothing}(x_*) = E_X[f(X) \,|\, X^l = x^l_*] - E_X[f(X)].$$

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.

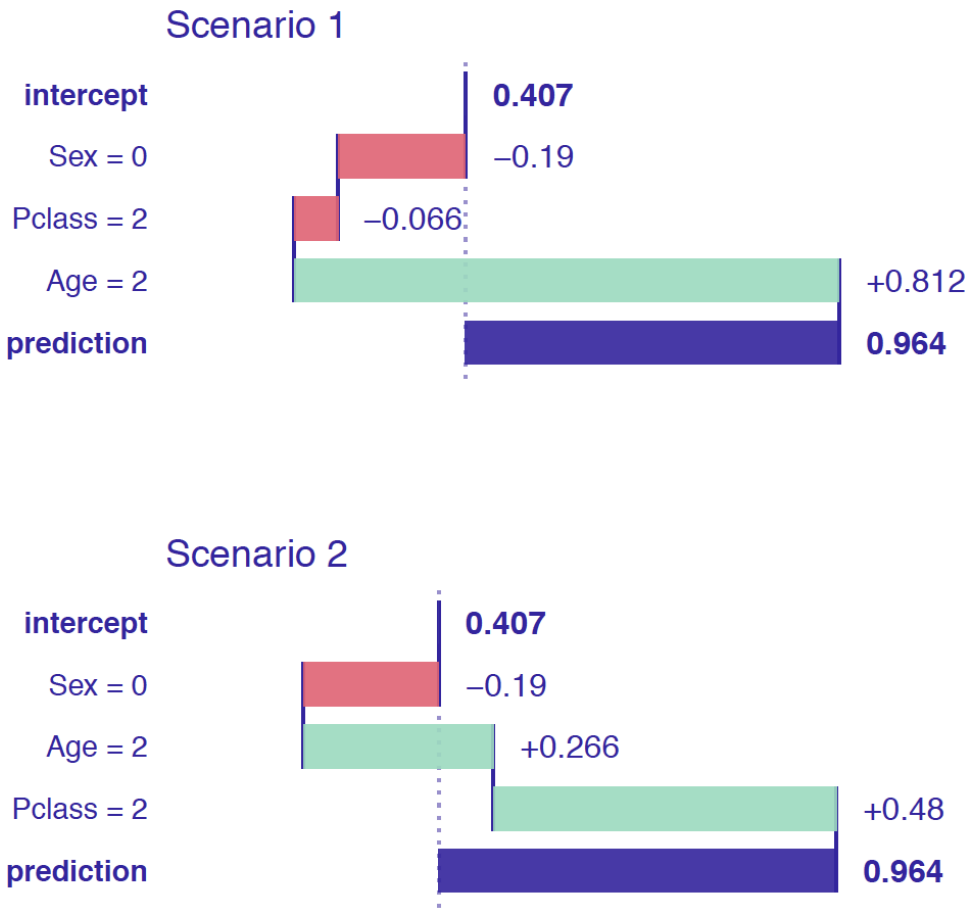


Figure 7.2: (fig:ordering) An illustration of the order-dependence of the variable-contribution values. Two *Break-down* explanations for the same observation from Titanic data set. The underlying model is a random forest. Scenarios differ due to the order of variables in *Break-down* algorithm. Blue 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 theeach 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 5.1.5) 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 $E[f(X) \mid X^j = x^j_*]$ and scores $|\Delta^{j|\varnothing}|$ 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*	$E[f(X) \mid X^j = x^j_*]$	$\lvert\Delta^{j\mid\varnothing}\rvert$
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 $|\Delta^{j|\varnothing}|$ 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*	$E[f(X) \mid X^{\{1,\ldots,j\}} = x^{\{1,\ldots,j\}}_*)]$	$\Delta^{j\mid\{1,\ldots,j\}}$
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.

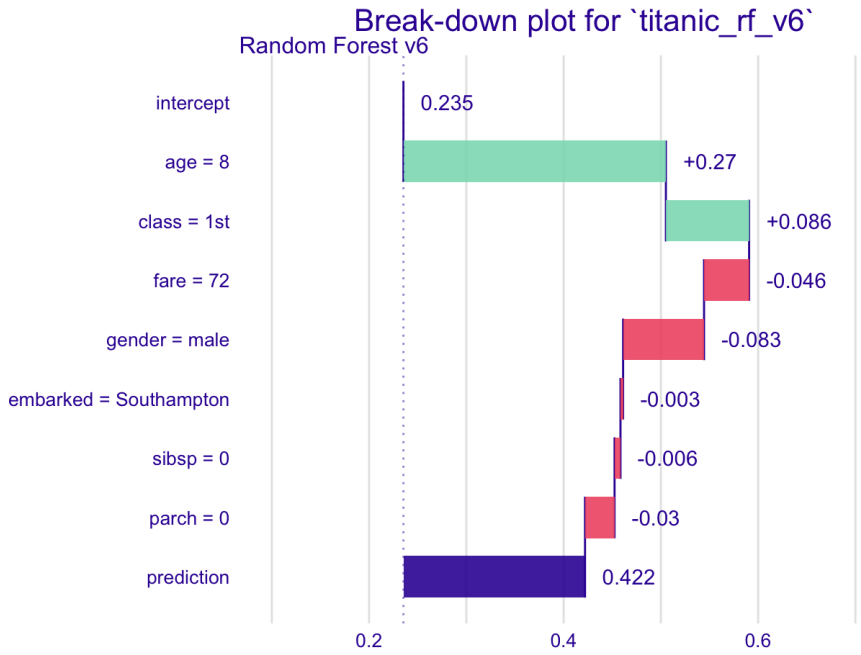


Figure 7.3: (fig:BDjohnyExample) Break-down plot for the `titanic_rf_v6` model and `johny_d` for the Titanic data.

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. 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 present key features of the `iBreakDown` R package (Gosiewska and Biecek 2019a) which is a part of the `DrWhy.AI` universe. The package covers all methods presented in this chapter. It is available on CRAN and GitHub. More details and examples can be found at https://modeloriented.github.io/iBreakDown/.

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 5.1.7.

```
library("randomForest")
explain_rf_v6 <- archivist::aread("pbiecek/models/9b971")

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

7.5.1 Basic use of the `break_down()` function

The `iBreakDown::break_down()` function calculates the variable-importance measures for a selected model and the instance of interest. The result of applying the `break_down()` function is a data frame containg the calculated measures. In the simplest call, the function requires only two arguments: the model explainers and the data frame for the instance of interest. The call below essentailly re-creates the variable-importance values ($\Delta^{j|\{1,\ldots,j\}}$) presented in Table 7.2.

```
library("iBreakDown")
bd_rf <- break_down(explain_rf_v6, johny_d)
bd_rf
```

```
##                                                contribution
## Random Forest v6: intercept                           0.235
## Random Forest v6: age = 8                             0.270
## Random Forest v6: class = 1st                         0.086
## Random Forest v6: fare = 72                          -0.046
## Random Forest v6: gender = male                      -0.083
## Random Forest v6: embarked = Southampton             -0.003
## Random Forest v6: sibsp = 0                          -0.006
## Random Forest v6: parch = 0                          -0.030
## Random Forest v6: prediction                          0.422
```

Applying the generic `plot()` function to the object resulting from the application of the `break_down()` function creates a BD plot. In this case, it is the plot from Figure 7.3. .

```
plot(bd_rf)
```

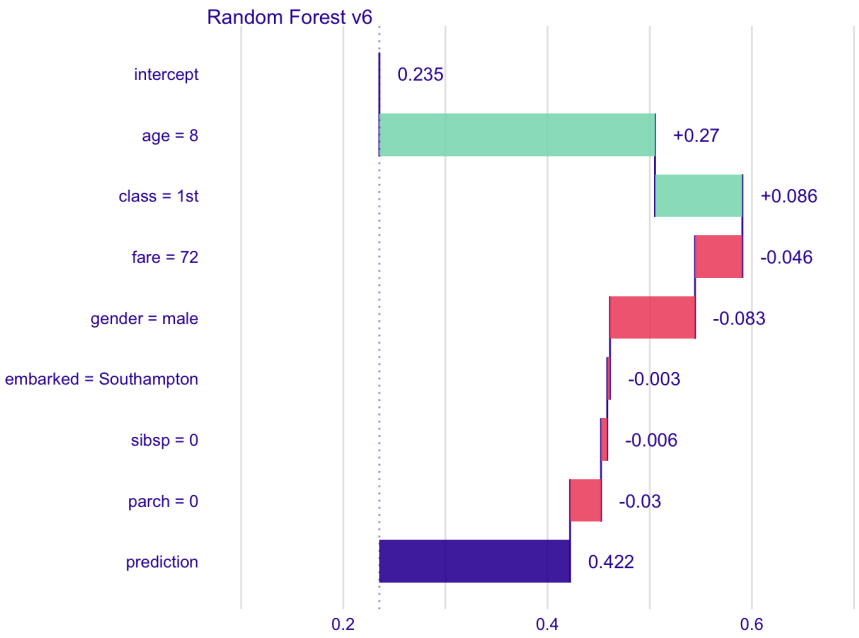


Figure 7.4: Generic plot() function for the break down method.

7.5.2 Advanced use of the `break_down()` function

The function `break_down()` 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.

```
library("iBreakDown")
bd_rf_order <- break_down(explain_rf_v6,
                 johny_d,
                 order = c("class", "age", "gender", "fare", "parch", "sibsp", "e
plot(bd_rf_order, max_features = 3)
```

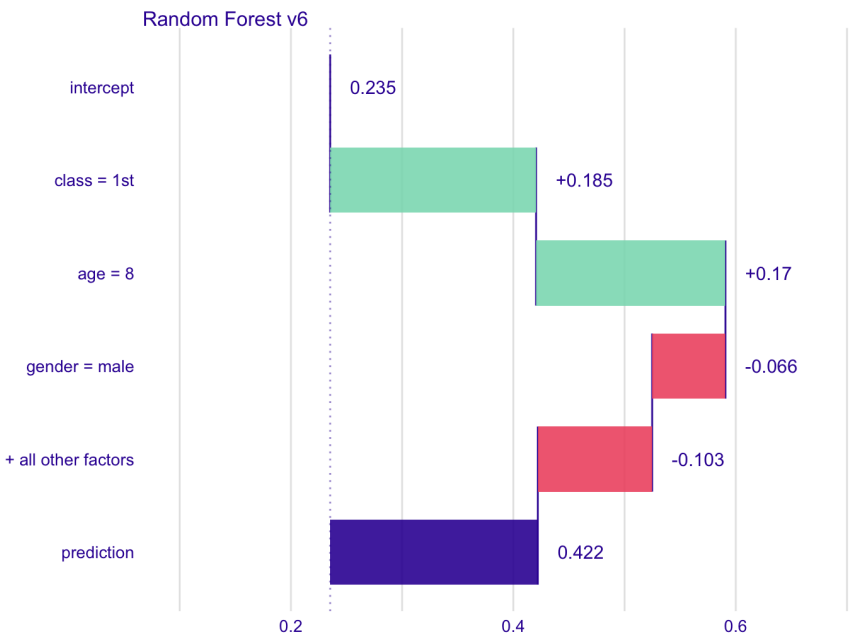


Figure 7.5: Break Down plot limited to top three features.

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 <- break_down(explain_rf_v6,
                 johny_d,
                 order = c("class", "age", "gender", "fare", "parch", "sibsp", "e
                 keep_distributions = TRUE)
plot(bd_rf_distr, plot_distributions = TRUE)
```

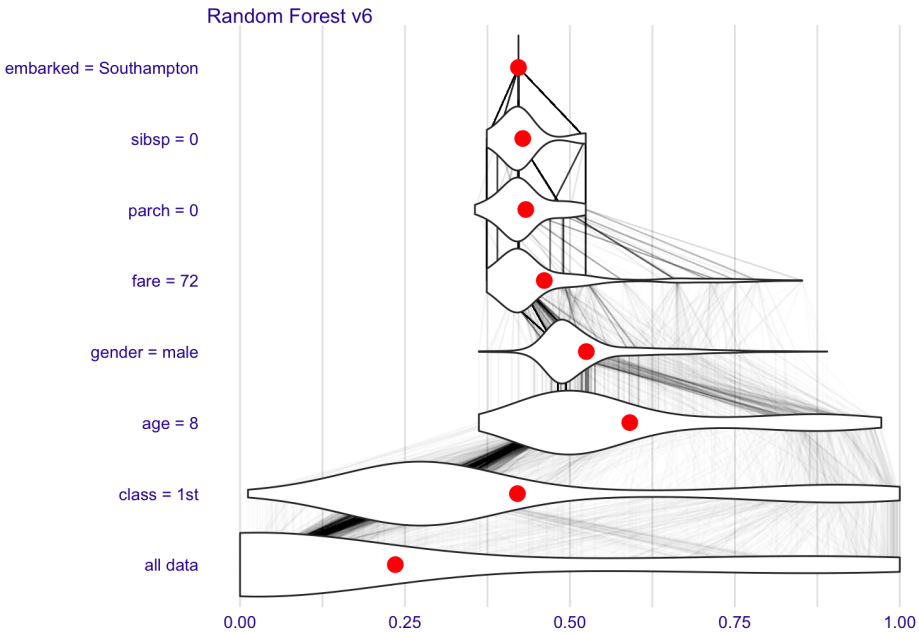


Figure 7.6: Distributions for conditional conditionings.

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.

Chapter 8 Break-down Plots for Models with 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 models with interactions, the estimated value of the variable-importance measure depends on the ordering of the explanatory varaibles 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

An interation 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 classess: 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 surviors 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.

- 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).

- 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.

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 intepret 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.

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%

The example illustrates that interactions complicate the evaluation of the importance of explanatory variables to model predictions. In what follows we present a simple alogrithm 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 $\Delta_I^{\{i,j\}}(x_*) \equiv \Delta^{\{i,j\}|\emptyset}(x_*) - \Delta^{i|\emptyset}(x_*) - \Delta^{j|\emptyset}(x_*)$.
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 $v(j, x_*) = \Delta^{j|\{1,...,j-1\}}(x_*)$ (see Section 7.2).

The numerical 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 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 5.1.5) 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 $\Delta^{\{i,j\}|\emptyset}(x_*)$, and interaction effects $\Delta_I^{\{i,j\}}(x_*)$ 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 $\Delta^{\{i,j\}|\emptyset}(x_*)$, and interaction effects $\Delta^{\{i,j\}}_I(x_*)$ for the random-forest model `titanic_rf_v6` and passenger `johny_d` in the Titanic data. 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 $\Delta^{\{i,j\}}_I(x_*)$.

Variable	$E_X[f(X)\|X^i = x^i_*, X^j = x^j_*]$	$\Delta^{\{i,j\}}(x_*)$	$\Delta^{\{i,j\}}_I(x_*)$
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	$E_X[f(X)\|X^i = x^i_*, X^j = x^j_*]$	$\Delta^{\{i,j\}}(x_*)$	$\Delta_I^{\{i,j\}}(x_*)$
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 $\Delta^{j|\{1,...,j\}}(x_*)$ 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	$\Delta^{j\|\{1,...,j\}}(x_*)$	$E_X[f(X)\|X^{\{1,...,j\}} = x_*^{\{1,...,j\}})]$
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. Given that the plots includes an interaction, the polot is called iBreak-down (iBD) plot.

```
## Preparation of a new explainer is initiated
##   -> model label       :  randomForest  (  default  )
##   -> data              :  2207  rows  10  cols
##   -> target variable   :  2207  values
##   -> predict function  :  yhat.randomForest  will be used (  default  )
##   -> predicted values  :  numerical, min =  0 , mean =  0.2353095 , max =  1
##   -> residual function :  difference between y and yhat (  default  )
##   -> residuals         :  numerical, min =  -0.892 , mean =  0.0868473 , max =
##   -> model_info        :  package randomForest , ver. 4.6.14 , task classifica
##   A new explainer has been created!
```

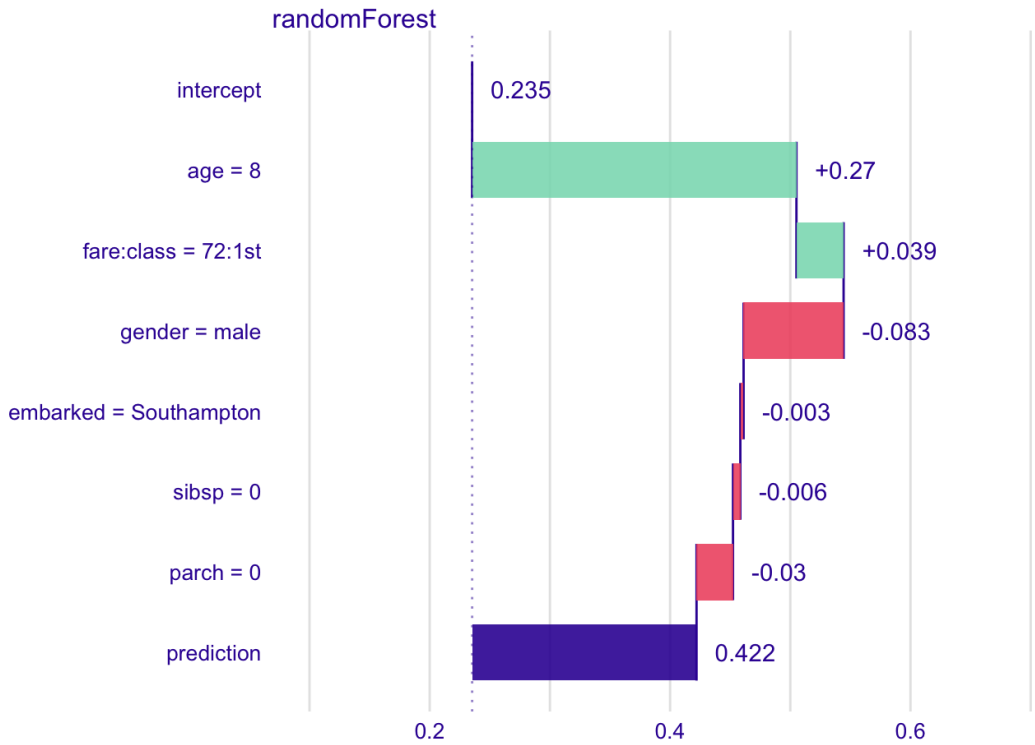


Figure 8.1: (fig:iBreakDownTitanicExamplePlot) Break-down plot with interactions for the `titanic_rf_v6` model and `johny_d` for the Titanic data.

8.4 Pros and cons

iBD plots share many features of BD plots for models without interactions. 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

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 5.1.7.

```
library("randomForest")
explain_rf_v6 <- archivist::aread("pbiecek/models/9b971")

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

The key function to construct iBD plots is the `iBreakDown::break_down()` function from the `iBreakDown` R package (Gosiewska and Biecek 2019a). The use of the function has already been explained in Section 7.5. The additional necessary argument is `interactions = TRUE` .

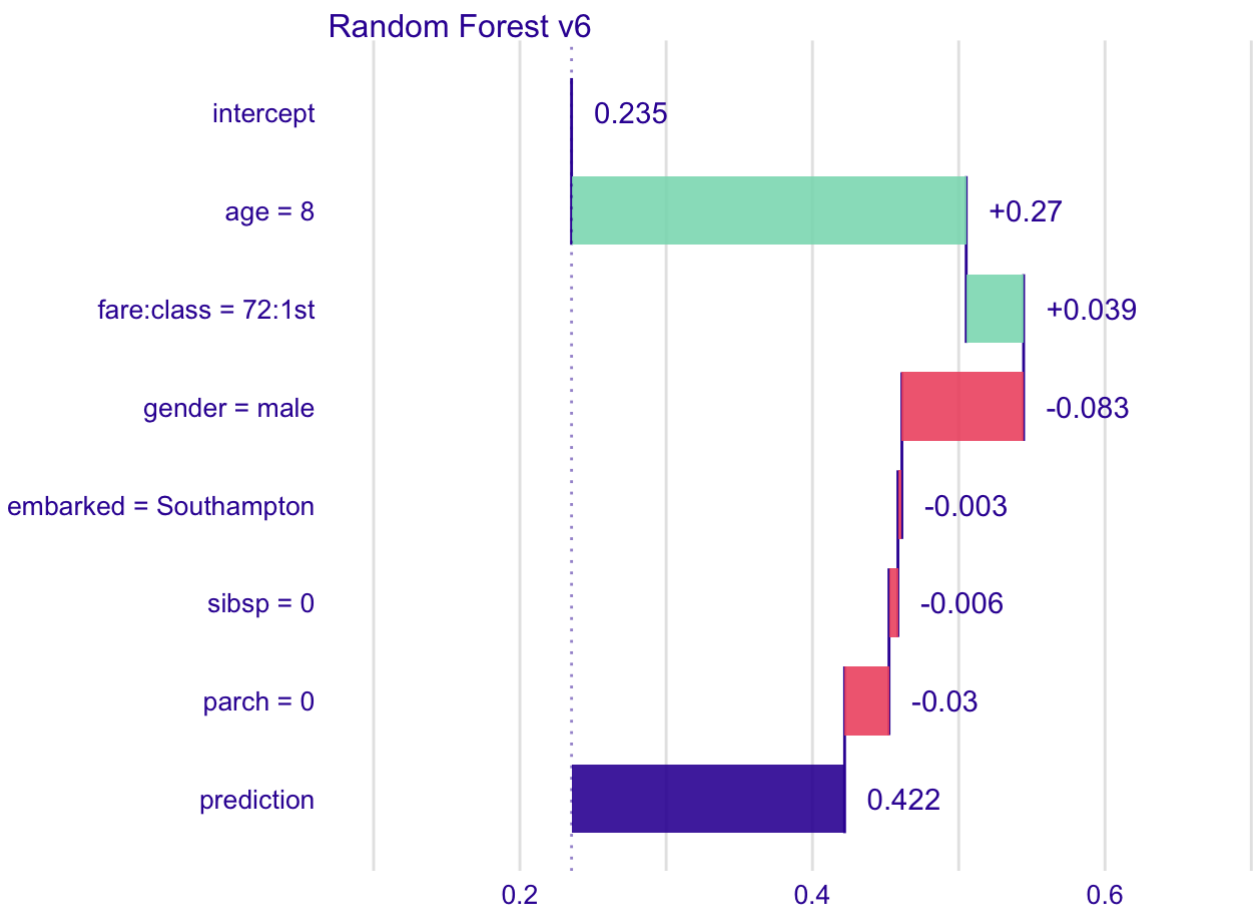
```
library("iBreakDown")
bd_rf <- break_down(explain_rf_v6, johny_d, interactions = TRUE)

bd_rf
```

```
##                                             contribution
## Random Forest v6: intercept                        0.235
## Random Forest v6: age = 8                          0.270
## Random Forest v6: fare:class = 72:1st              0.039
## Random Forest v6: gender = male                   -0.083
## Random Forest v6: embarked = Southampton          -0.003
## Random Forest v6: sibsp = 0                       -0.006
## Random Forest v6: parch = 0                       -0.030
## Random Forest v6: prediction                       0.422
```

Applying the generic `plot()` function to the object resulting from the application of the `break_down()` function creates an iBD plot. In this case, it is the plot from Figure 8.1.

```
plot(bd_rf)
```



References

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

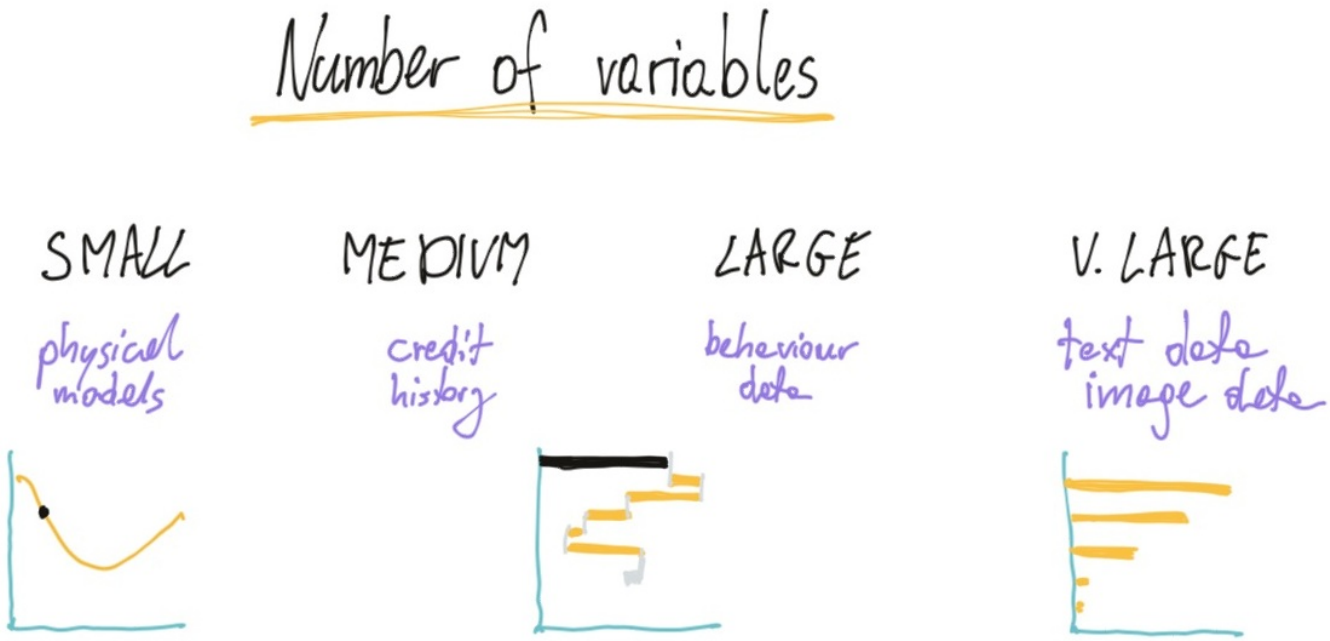
Chapter 14 Summary of Instance-level Explainers

In the first part of the book, we introduced a number of techniques for exploration and explanation of model predictions for instances of interest. In this chaper, we discuss their strengths and weaknesses taking into account different possible applications.

[TOMASZ: THIS WOULD LOOK MORE APPROPRIATE AS A FINAL SECTION OF A CHAPTER, IN WHICH THERE WOULD BE ONE OR TWO WORKED EXAMPLES ILLUSTRATING APPLICATION OF VARIOUS METHODS.]

14.1 Number of explanatory variables in the model

One of the most important criteria for selection of model exploration and explaination methods is the number of explanatory variables in the model.



figure/instanceExplainer.jpg

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 credit-scoring model [TOMASZ: SCORING IN WHICH SENSE? THE RISK OF PAYMENT OF INSTALLMENTS?] based on behavioral data that may include 100+ variables. [TOMASZ: WE HAVE NOT GOT ANY EXAMPLE.]

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 Cahpter 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

Most of the presented methods assumed that explanatory variables are independent. Obviously, this is not always the case. For instance, in the case of the data on aparemten prices (see Chapter 5.2), the number of rooms and surface of an apartment will most likely be positively associated.

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), * permute variables in blocks to preserve the correlation structure, as it was described in Chapter 10 .

14.3 Models with interactions

In models with initeractions, 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. [TOMASZ: WE HAVE NOT GOT SUCH A MODEL.] 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 investigate the effect of pairwise interactions, we can use 2D CP profiles, as introduced in @ref(). [TOMASZ: WE MISS A CHAPTER ON 2D PROFILES. WORTH RE-INTRODUCING? OR - WE SHOULD HAVE AN EXAMPLE.]

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 predcitions 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 evalaution 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 hypothethical 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 wll be rainy. However, investigation of the instance predictions (for individual days) may lead to a preference for one of them. [TOMASZ: NOT SURE IF HELPFUL, BUT WANTED TO MAKE THIS CASE MORE CONCRETE.]

Chapter 13 Local Diagnostics With Ceteris-paribus Profiles

13.1 Introduction

It may happen that, while the global predictive performance of the model is good, model predictions for some observations are very bad. In this chapter, we present two local-diagnostics techniques that can address this issue. In particular, we focus on fidelity plots: the plot of CP profiles for nearest neighbors and the local-fidelity plot.

The 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

One approach to local model diagnostics is to examine how the predictions vary for observations from the training dataset. Figure 13.1 presents CP profiles 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.

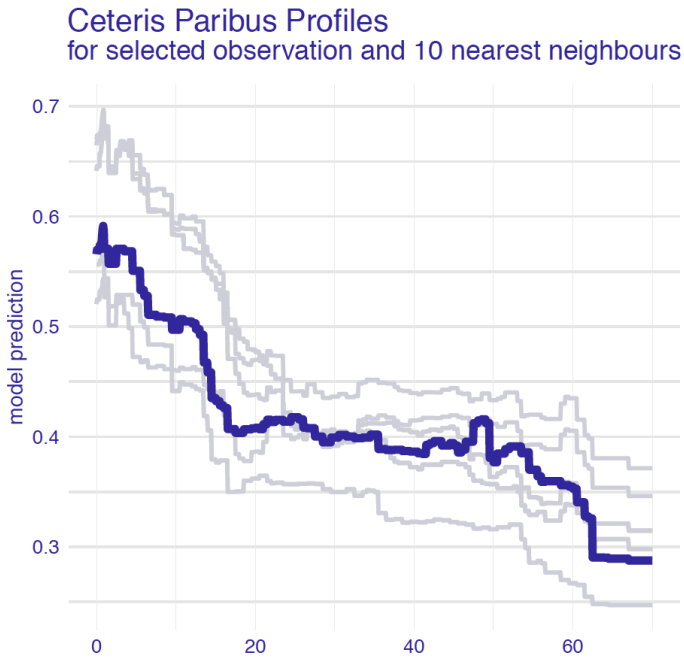


Figure 13.1: (fig:profileWith10NN) Ceteris-paribus profiles for a selected instance (dark violet line) and 10 nearest neighbors (light grey lines) for the `titanic_rf_b6` model. The profiles are almost parallel and close to each other what suggests the stability of the model.

Once we have selected the nearest neighbors, we can also look closer at the model fit around the point of interest. Figure 13.2 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 sugests that, on average, the model predictions are biased for the instance of interest.

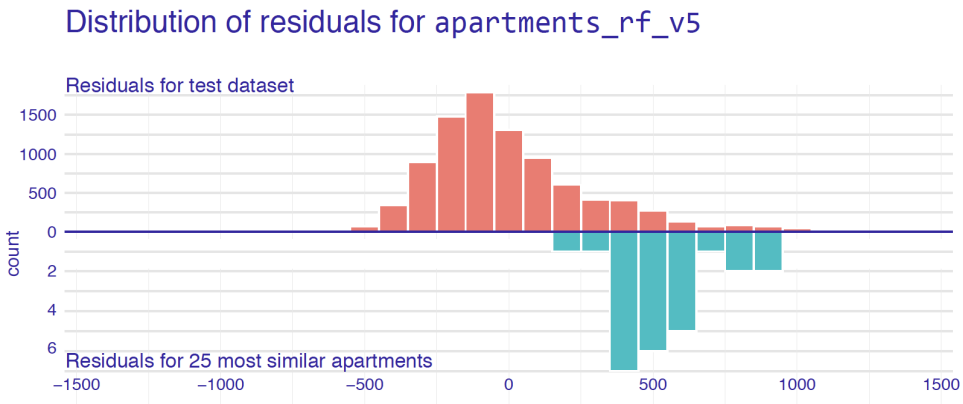


Figure 13.2: (fig:profileBack2BackHist) Histograms of residuals for the `apartments_rf_v5` model for the Apartments dataset. Upper panel: residuals calculated for all observations from the dataset. Bottom panel: residuals calculated for 25 nearest neighbors of the instance of interest.

13.3 Method

The proposed method is based on three elements:

- identification of nearest neighbors,
- calculation and visualization of CP profiles for the selected neighbors, and
- analysis of residuals for the 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, of course, *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.

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 Profiles 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 Chapter 12).

13.3.3 Local-fidelity plot

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, i.e.,

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

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 defintion also applies to categorical responses, as it is common to define, in such case, a binary "success" indicator and compute the predicted probaiblity 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.

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 5.1.5). 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 13.4 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 probabilityof 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.

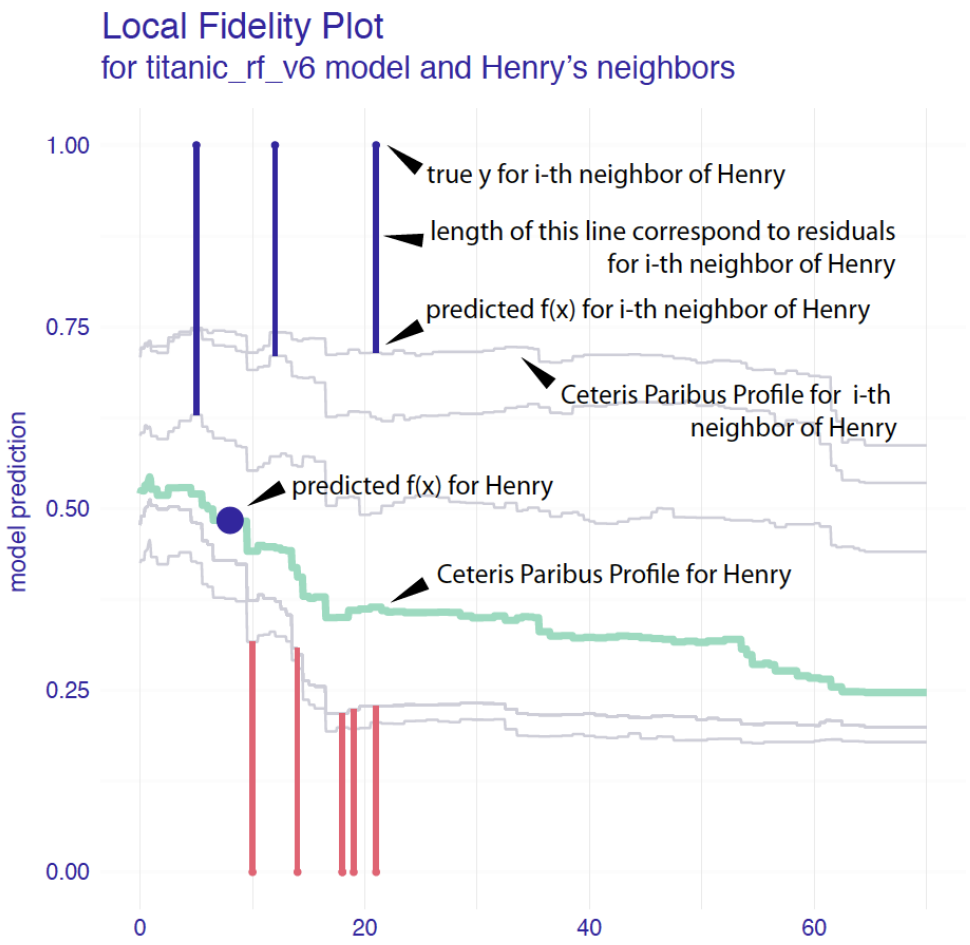


Figure 13.3: (fig:localFidelityPlots) Elements of a local-fidelity plot for a continuous explanatory variable. The green line shows the Ceteris-paribus profile for the instance of interest. Profiles of the nearest neighbors are marked with grey lines. The vertical intervals correspond to residuals; the 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.

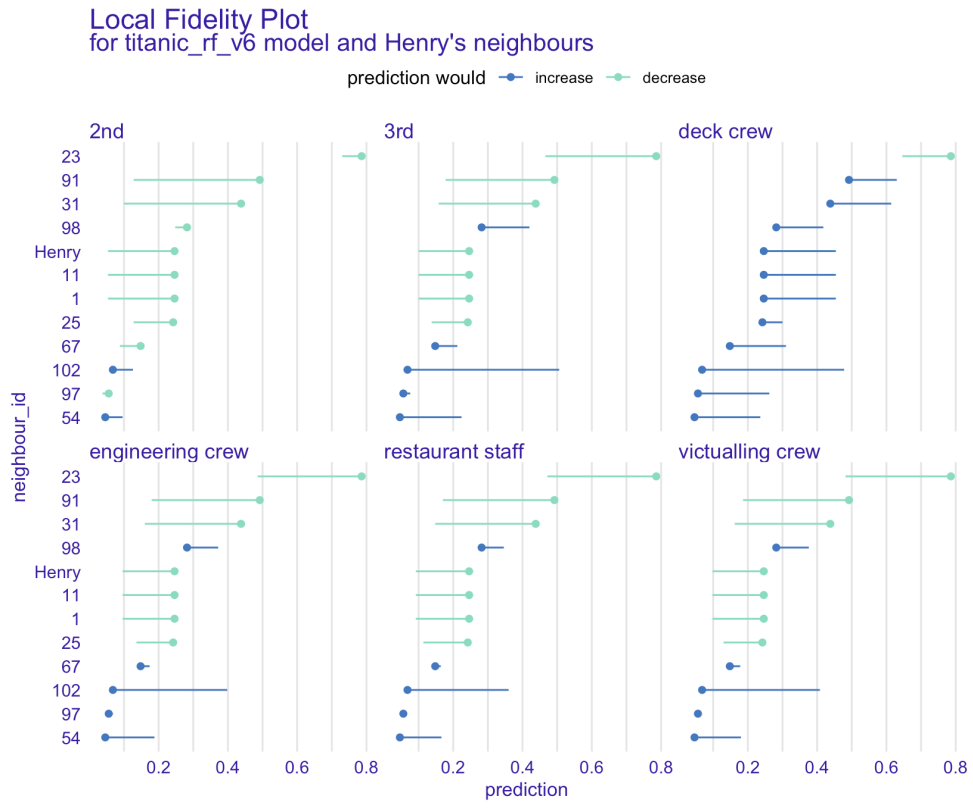


Figure 13.4: (fig:localFidelityPlots2) The local-fidelity plot for the categorical explanatory variable `class` in the random effects model for the Titanic data, Henry, and his 10 neighbors. Each panel indicates how the model prediction would change if the class changed from `1st` to another one. Dots 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 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 `ingredients` (Biecek et al. 2019) to construct local-fidelity plots. More details and examples can be found at `https://modeloriented.github.io/ingredients/` .

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 5.1.7.

```
library("randomForest")
explain_rf_v6 <- archivist::aread("pbiecek/models/9b971")

library("DALEX")
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 Figure 13.1. Toward this aim we need some number of passengers most similar to `henry`. To select the ‘’neighbors”, we use the `select_neighbours()` function from the `ingredients` package. It returns `n` observations (by default, 20) most similar to the observation of interest according to the `distance` measure (by default, the Gower distance is used).

In the example below, we select 10 nearest neighbors to `henry` find from the `titanic` dataset. Note that the similarity is based only on two explanatory variables, *gender*, and *class*, as indicated in the argument `variables`.

```
library("ingredients")
henry_neighbors <- select_neighbours(henry,
                          data = titanic,
                          n = 10,
                          variables = c("class", "gender"))
```

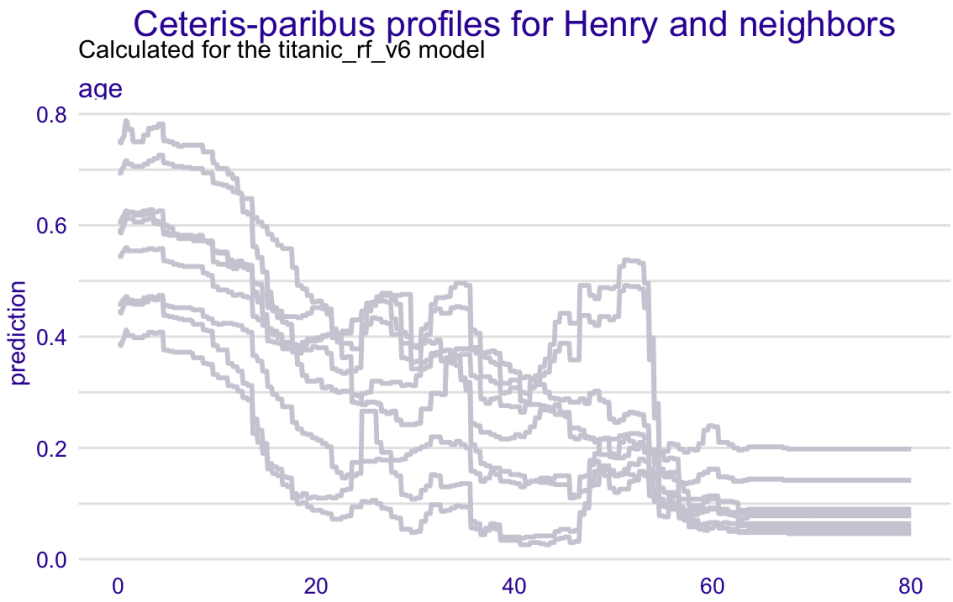
Now we are ready to plot profiles for Henry and his neighbors. First, we have got to calculate the corresponding profiles with `ceteris_paribus()` function introduced in Section 11.6.

```
cp_henry <- ceteris_paribus(explain_rf_v6,
                            henry,
                            variable_splits = list(age = seq(0,80,0.1)))
cp_henry_neighbors <- ceteris_paribus(explain_rf_v6,
                            henry_neighbors,
                            variable_splits = list(age = seq(0,80,0.1)))
```

Subsequently, we can plot the profiles. Note that, in the example below, we do this only for a single variable `age`.

```
library("ggplot2")

plot(cp_henry_neighbors, color = '#ceced9') +
#  show_profiles(cp_henry, size = 2)  +
  ggtitle("Ceteris-paribus profiles for Henry and neighbors", "Calculated for the
```

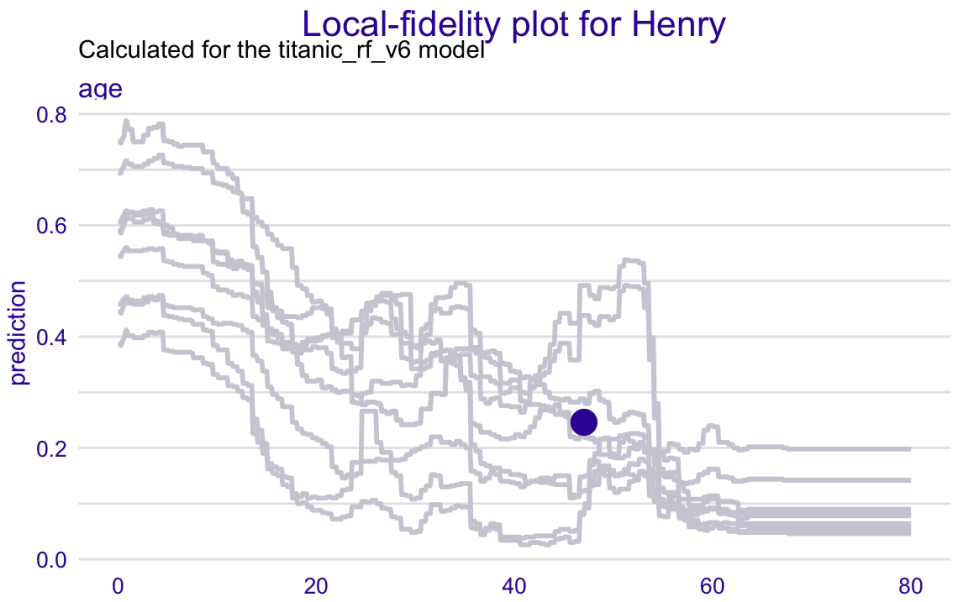


To construct the local-fidelity plot presented in Figure 13.3, we have got to add the information about the observed values of the dependent variable for the selected neighbors. Toward this aim, we use the `y` argument in the `ceteris_paribus()` function. The argument takes numerical values. Our binary dependent variable `survived` assumes values `yes/no`; to convert them to numerical values, we use the `survived == "yes"` expression.

```
cp_henry_neighbors <- ceteris_paribus(explain_rf_v6,
                        henry_neighbors,
                        y = henry_neighbors$survived == "yes",
                        variable_splits = list(age = seq(0,80,0.1)))
```

Finally, we add residuals to the plot by using the `show_residuals()` function. As a result, we obtain the local-fidelity plot for `henry`.

```
plot(cp_henry_neighbors, color = '#ceced9') +
#  show_profiles(cp_henry, size = 2) +
  show_observations(cp_henry, variables = "age", size = 5) +
#  show_residuals(cp_henry_neighbors, variables = "age") +
  ggtitle("Local-fidelity plot for Henry","Calculated for the titanic_rf_v6 mode
```



References

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

Chapter 12 Ceteris-paribus Oscillations and Local Variable-importance

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 is discussed in the Chapters 7 and 9.

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. Panle 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: (fig:CPVIPprofiles) 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 (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

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

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 equi-distant 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

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

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 `henry`.

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.

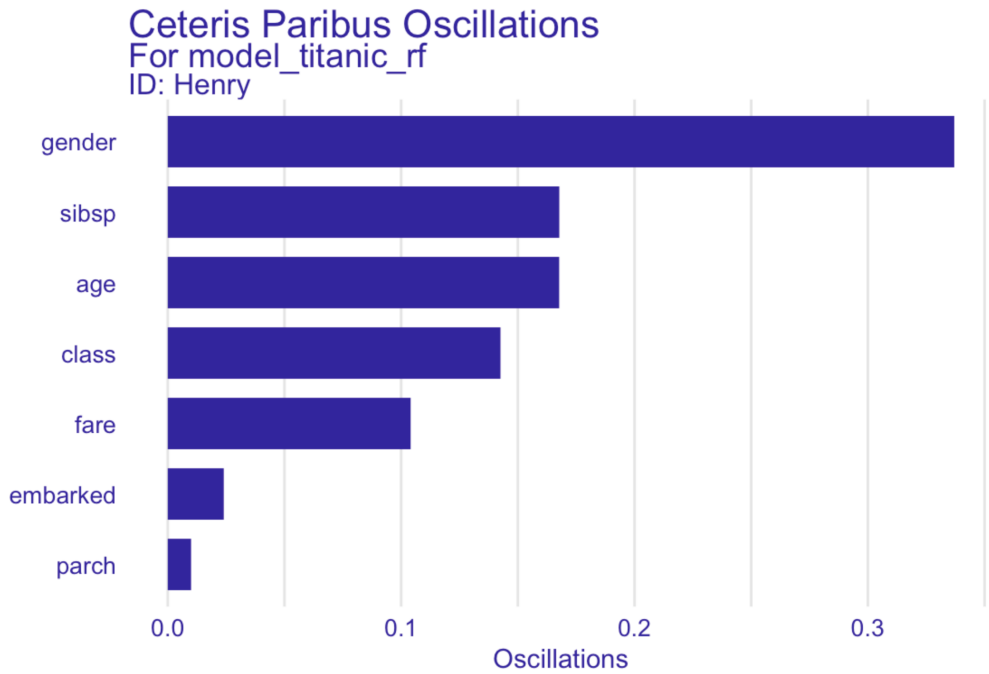


Figure 12.2: (fig:CPVIP1) Variable-importance measures calculated for Ceteris-paribus oscillations for `henry` based on the `titanic_rf_v6` model

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 local variable importance do not sum up to the instance prediction for which they are calculated. In Chapters 7 and 9, we will introduce measures that address this problem.

12.6 Code snippets for R

In this section, we present key features of R package `ingredients` which is a part of the `DrWhy.AI` universe and covers all methods presented in this chapter. More details and examples can be found at https://modeloriented.github.io/ingredients/.

For illustration purposes we use the random forest model `titanic_rf_v6` (see Section **??**). 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 5.1.7.

```
library("randomForest")
explain_rf_v6 <- archivist::aread("pbiecek/models/9b971")

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

12.6.1 Basic use of the `calculate_oscillations` 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 `ceteris_paribus()` function to the wrapper object.

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

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

The resulting object can subsequently be processed with the `calculate_oscillations()` function to calculate the oscillations and the estimated value of the variable-importance measure (12.1).

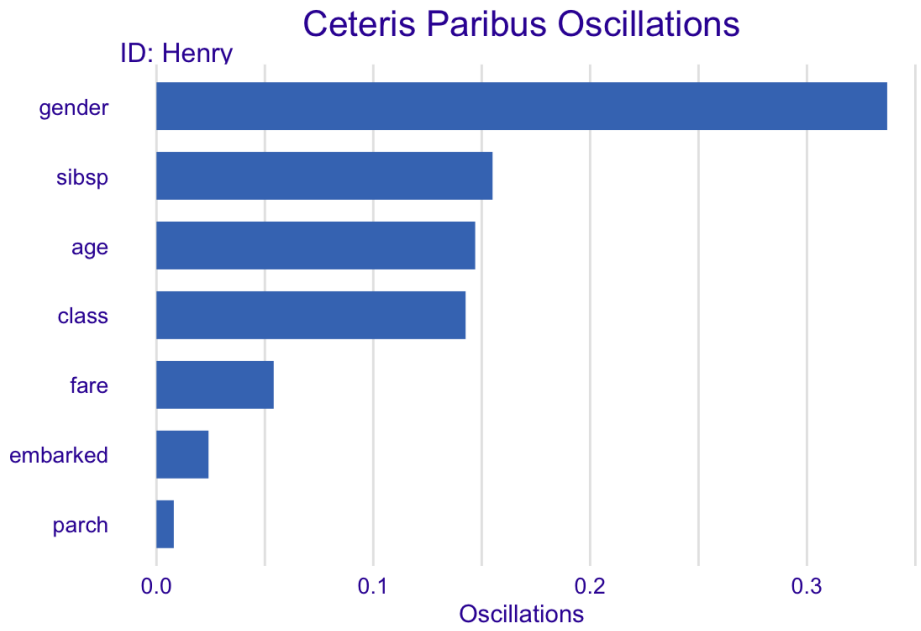
```
oscillations_titanic_rf <- calculate_oscillations(cp_titanic_rf)
oscillations_titanic_rf
```

```
##    _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, `calculate_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 `calculate_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 `calculate_oscillations` function

As mentioned in the previous section, `calculate_oscillations()` 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.

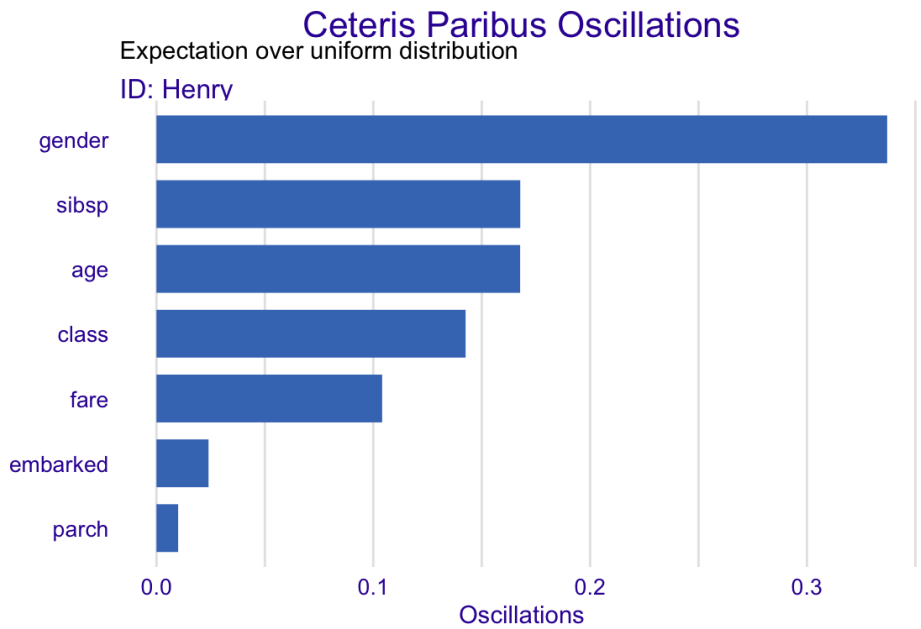
```
cp_titanic_rf_uniform <- ceteris_paribus(explain_rf_v6, henry,
             variable_splits = list(age = seq(0, 65, 0.1),
                                    fare = seq(0, 200, 0.1),
                                    sibsp = seq(0, 8, 0.1),
                                    parch = seq(0, 8, 0.1),
                                    gender = unique(titanic$gender),
                                    embarked = unique(titanic$embarked),
                                    class = unique(titanic$class)))
```

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

```
oscillations_uniform <- calculate_oscillations(cp_titanic_rf_uniform)
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", "Expectatior
```



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.

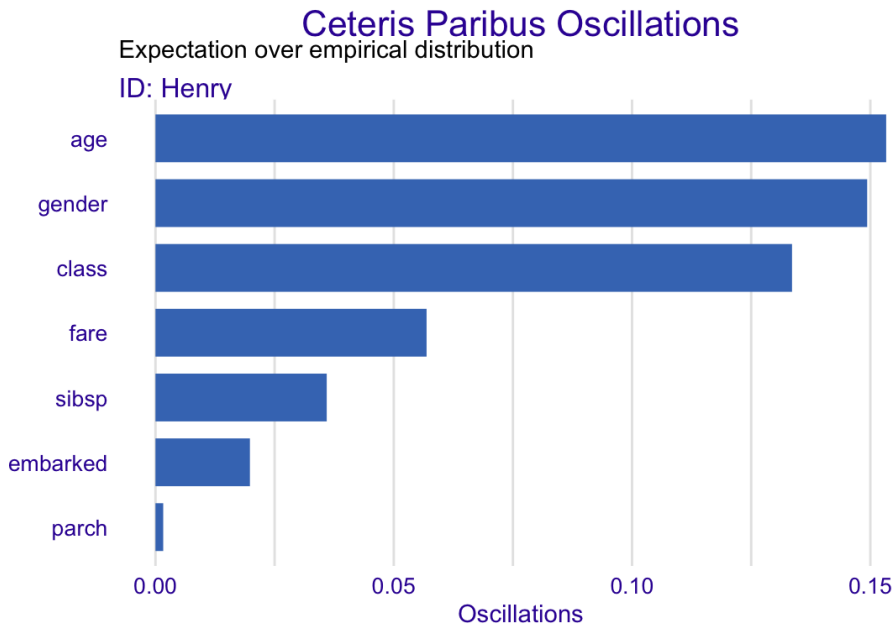
```
titanic <- na.omit(titanic)

cp_titanic_rf_empirical <- ceteris_paribus(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))


oscillations_empirical <- calculate_oscillations(cp_titanic_rf_empirical)
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


plot(oscillations_empirical) + ggtitle("Ceteris Paribus Oscillations", "Expectat
```


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

Chapter 11 Ceteris-paribus Profiles and What-If Analysis

11.1 Introduction

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

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 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.

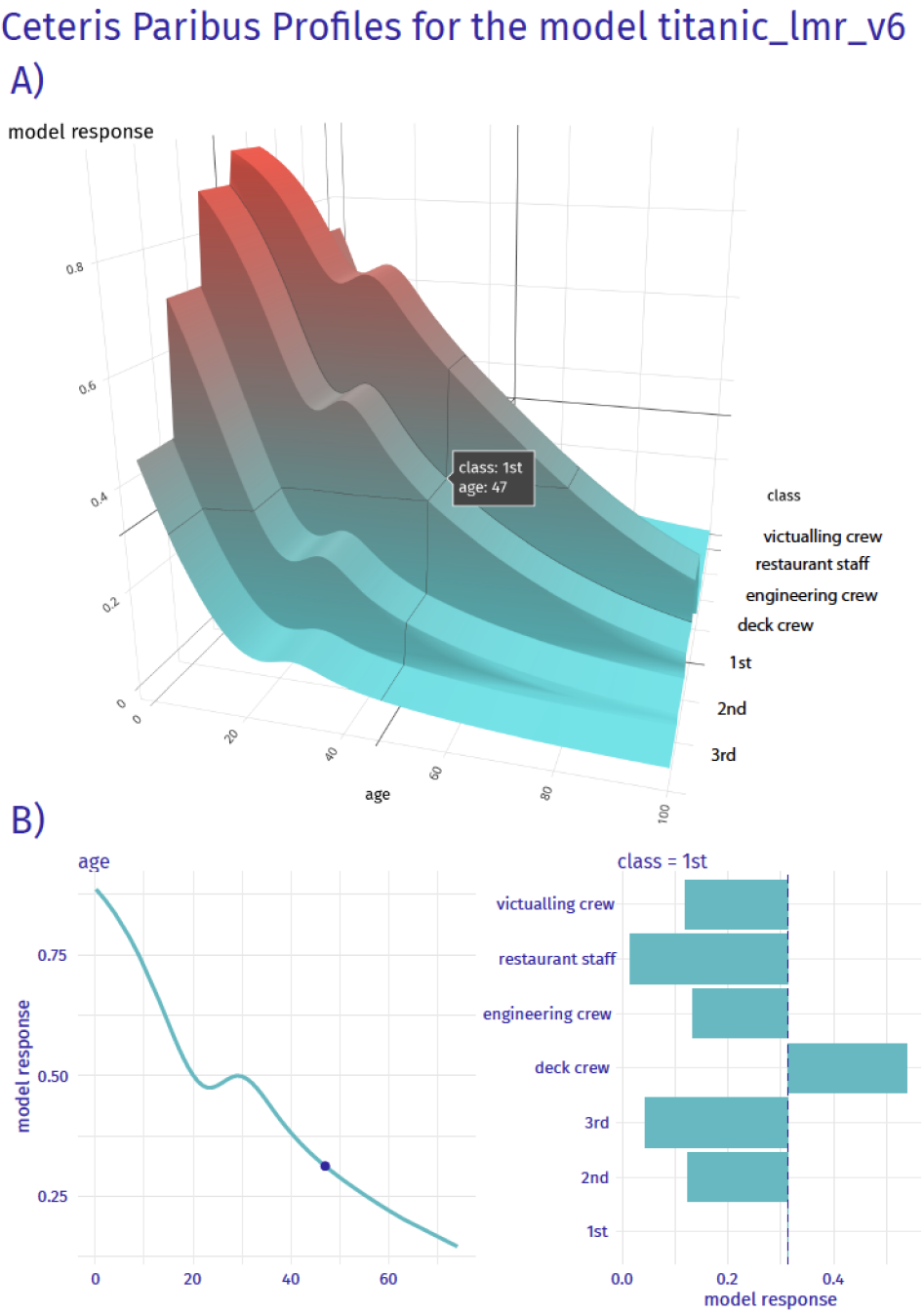


Figure 11.1: (fig:modelResponseCurveLine) A) Model response (prediction) surface. Ceteris-paribus (CP) profiles marked with black curves help to understand the curvature of the surface while changing only a single explanatory variable. B) CP profiles for individual variables, age (continuous) and class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

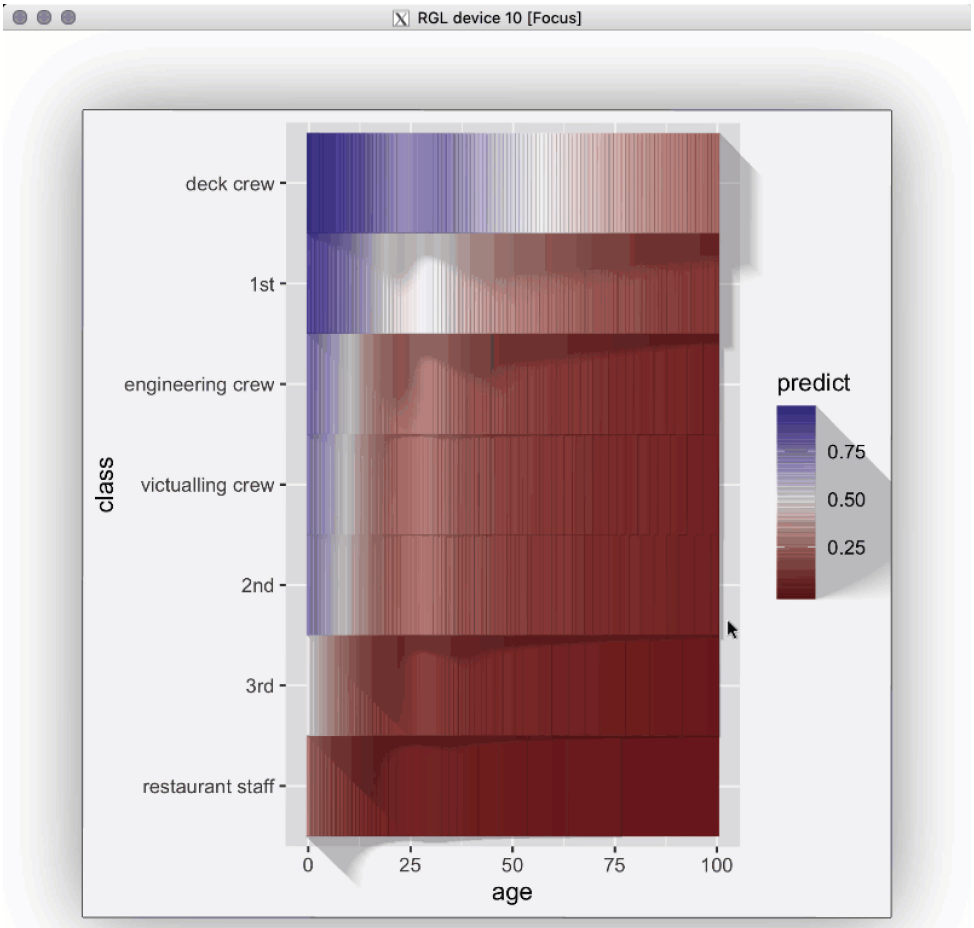


Figure 11.2: (fig:modelResponseCurveAnimation) Animated model response 2D surface as in (fig:modelResponseCurveLine).

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 every 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.

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 got 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.

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:

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

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 and random forest models 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 forrest model, a substantial increase would be obtained if *age* became lower than 13 or so.

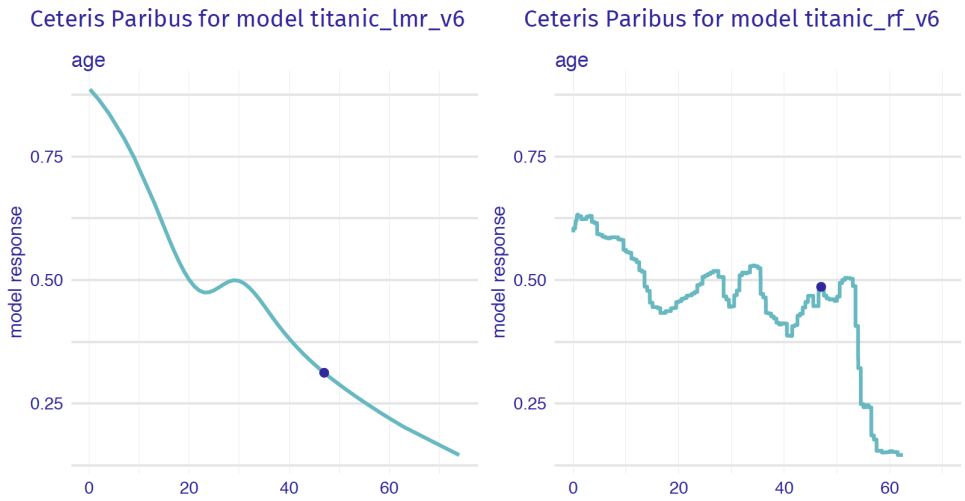


Figure 11.3: (fig:profileAgeRf) Ceteris-paribus profiles for variable `age` for the logistic regression (`titanic_lmr_v6`) and random forest (`titanic_rf_v6`) models that predict the probability of surviving based on the Titanic data

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”.

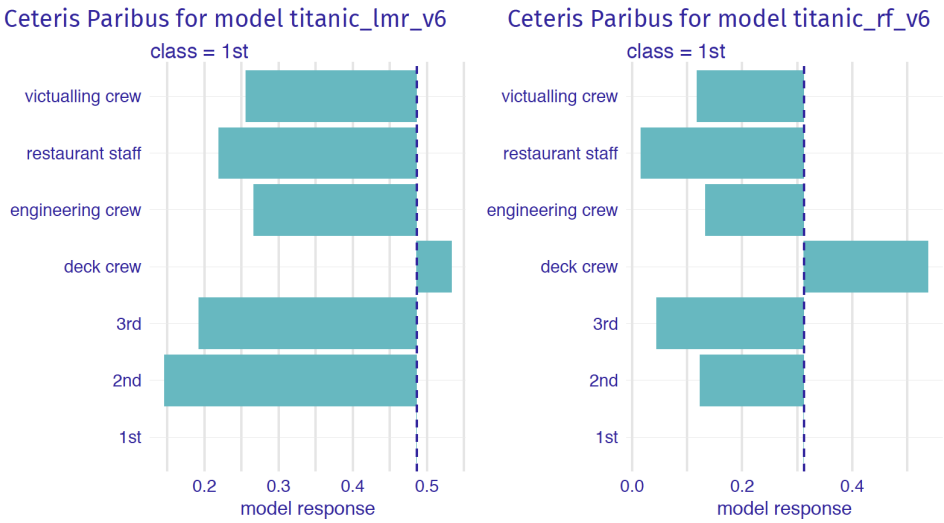


Figure 11.4: (fig:profileAgeRf2) Ceteris-paribus profiles for variable `class` for the logistic regression (`titanic_lmr_v6`) and random forest (`titanic_rf_v6`) models that predict the probability of surviving based on the Titanic data

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.

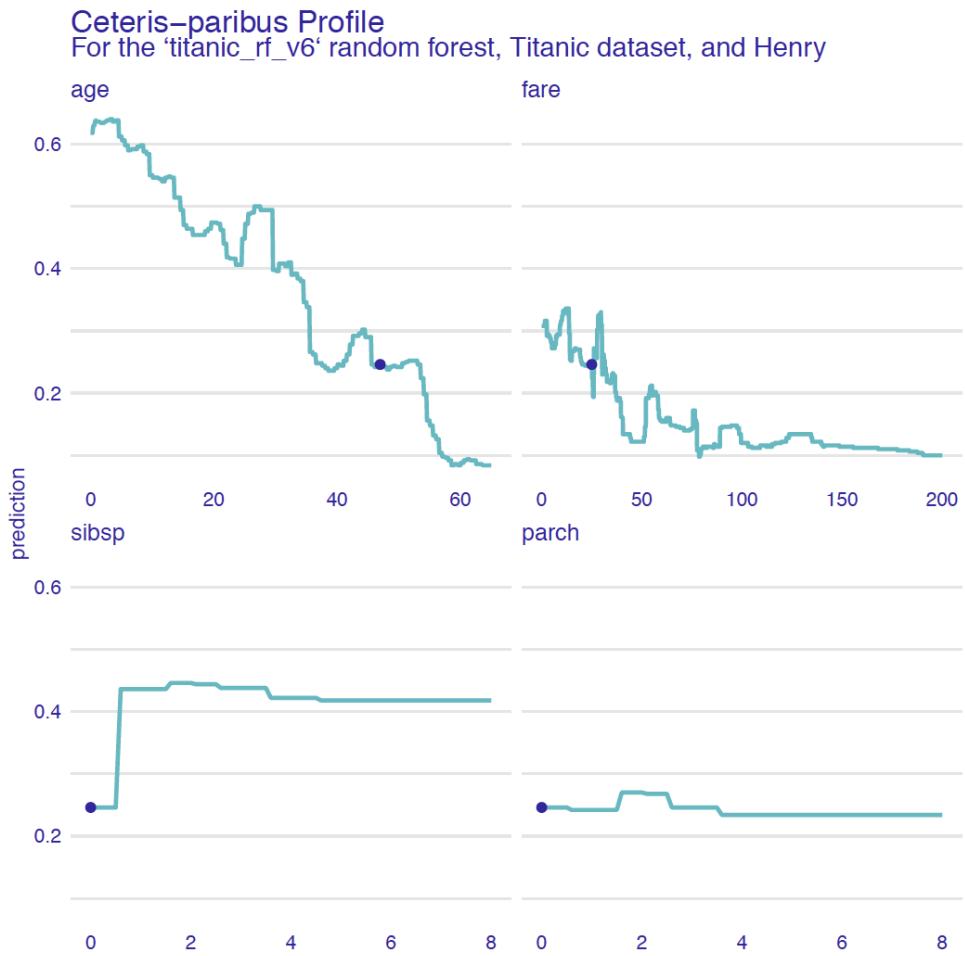


Figure 11.5: (fig:profileV4Rf) Ceteris-paribus profiles for all continuous explanatory variables for the random forest (`titanic_rf_v6`) model for the `titanic` dataset

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.

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 extreme 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 `ingredients` (Biecek et al. 2019) which is a part of `DrWhy.AI` universe and covers all methods presented in this chapter. More details and examples can be found at https://modeloriented.github.io/ingredients/.

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 5.1.7.

```
library("rms")
explain_lmr_v6 <- archivist::aread("pbiecek/models/2b9b6")

library("randomForest")
explain_rf_v6 <- archivist::aread("pbiecek/models/9b971")

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 `ceteris_paribus` function

The easiest way to create and plot CP profiles is to call `ceteris_paribus()` 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 `ceteris_paribus()` function requires the explainer-object and the instance data frame as arguments. As a result, the function yields an object od the class `ceteris_paribus_explainer`. It is a data frame with model predictions.

```
library("ingredients")
cp_titanic_rf <- ceteris_paribus(explain_rf_v6, henry)
cp_titanic_rf
```

```
## Top profiles    :
##                class gender age sibsp parch fare  embarked _yhat_ _vname_ _ids
## 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
##              _label_
## 1   Random Forest v6
## 1.1 Random Forest v6
## 1.2 Random Forest v6
## 1.3 Random Forest v6
## 1.4 Random Forest v6
## 1.5 Random Forest v6
##
##
## Top observations:
##   class gender age sibsp parch fare  embarked _yhat_          _label_ _ids_
## 1   1st   male  47     0     0   25 Cherbourg  0.246 Random Forest v6     1
```

To obtain a graphical representation of CP profiles, the generic `plot()` function can be applied to the data frame returned by the `ceteris_paribus()` 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")) +
  show_observations(cp_titanic_rf, variables = c("age", "fare")) +
  ggtitle("Ceteris Paribus Profiles", "For the random forest model and the Titan
```

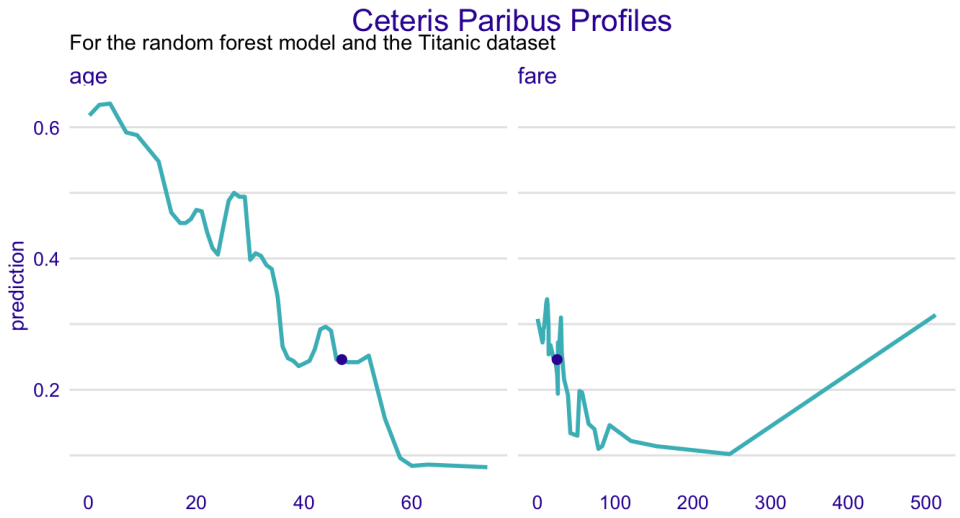


Figure 11.6: Ceteris-paribus profiles for `age` and `fare` variables and the `titanic_rf_v6` model.

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 = "categor
  ggtitle("Ceteris Paribus Profiles", "For the random forest model and the Titan
```

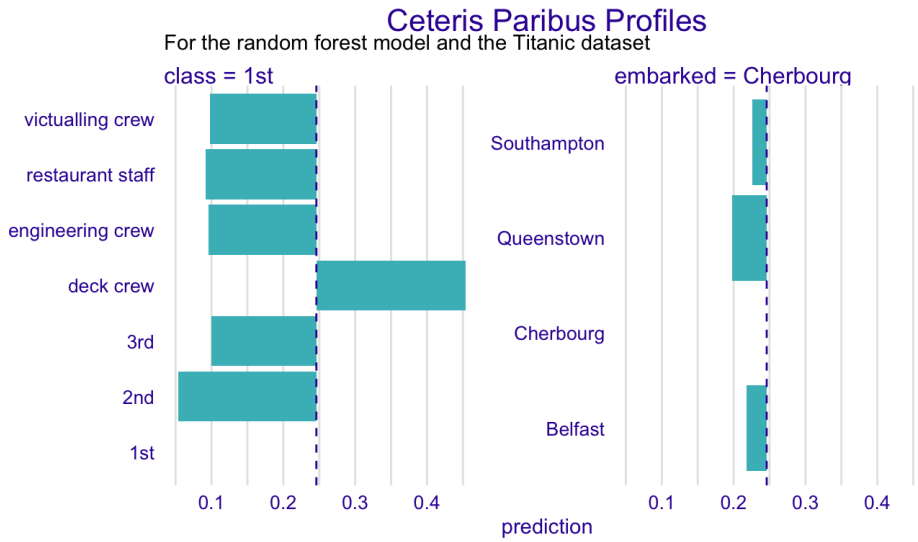


Figure 11.7: Ceteris-paribus profiles for `class` and `embarked` variables and the `titanic_rf_v6` model.

11.6.2 Advanced use of the `ceteris_paribus` function

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


plot(cp_titanic_rf) +
  show_observations(cp_titanic_rf, variables = c("age", "fare"), size = 5) +
  ylim(0, 1) +
  ggtitle("Ceteris Paribus Profiles", "For the random forest model and titanic da
```

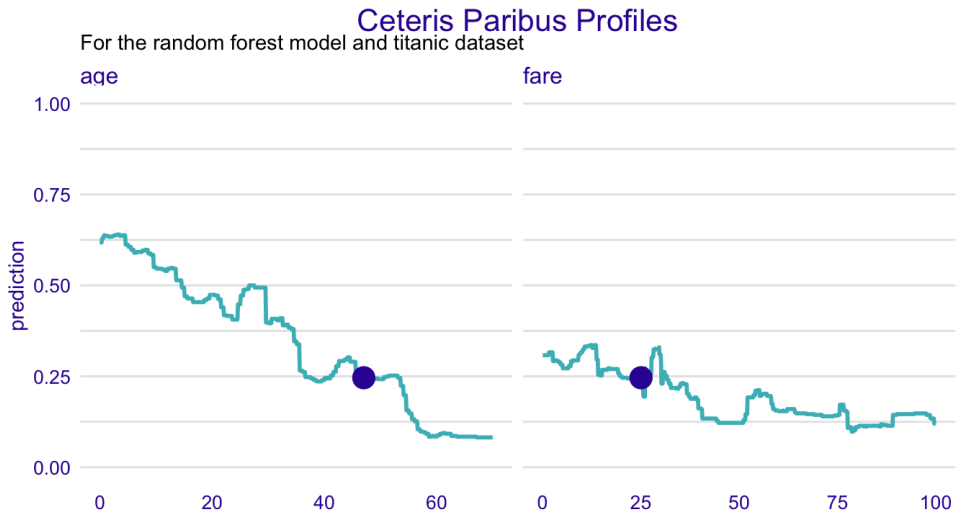


Figure 11.8: Ceteris-paribus profiles for `class` and `embarked` variables and the `titanic_rf_v6` model. Blue dot stands for `henry` .

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 superpose profiles for different models or observations.

In the example below, we present the code to create XO 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 <- ceteris_paribus(explain_rf_v6, rbind(henry, johny_d))


plot(cp_titanic_rf2, color = "_ids_") +
  show_observations(cp_titanic_rf2, size = 5, variables = c("age", "fare")) +
  show_rugs(cp_titanic_rf2, sides = "bl", variables = c("age", "fare")) +
  scale_color_manual(name = "Passenger:", breaks = 1:2, values = c("#4378bf", "#8
  ggtitle("Ceteris Paribus Profiles", "For the random forest model and the Titan
```

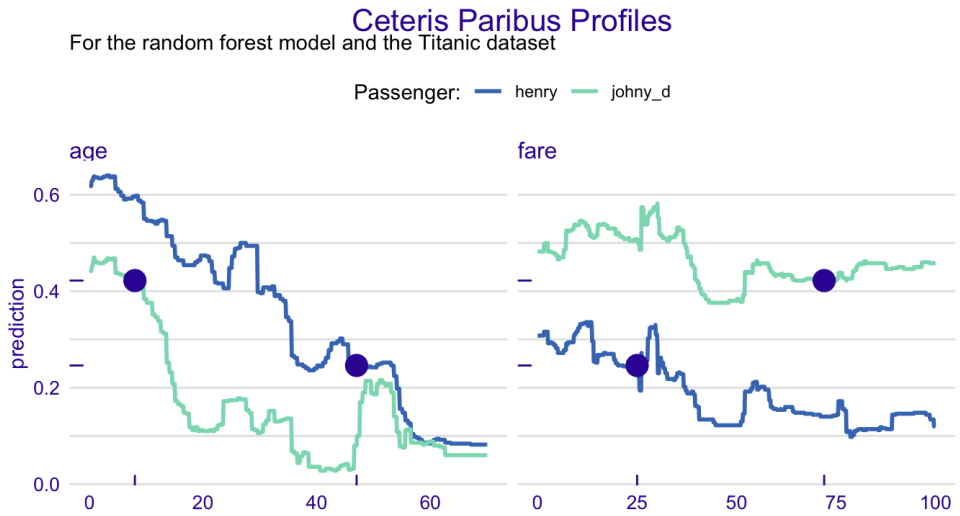


Figure 11.9: Ceteris-paribus profiles for the `titanic_rf_v6` model. Profiles for different passangers are color-coded.

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_") +
  show_observations(cp_titanic_rf, cp_titanic_lmr, color = "black", variables = c
  scale_color_discrete(name = "Selected models:") + ylim(0,1) +
  ggtitle("Ceteris Paribus Profiles for Henry")
```

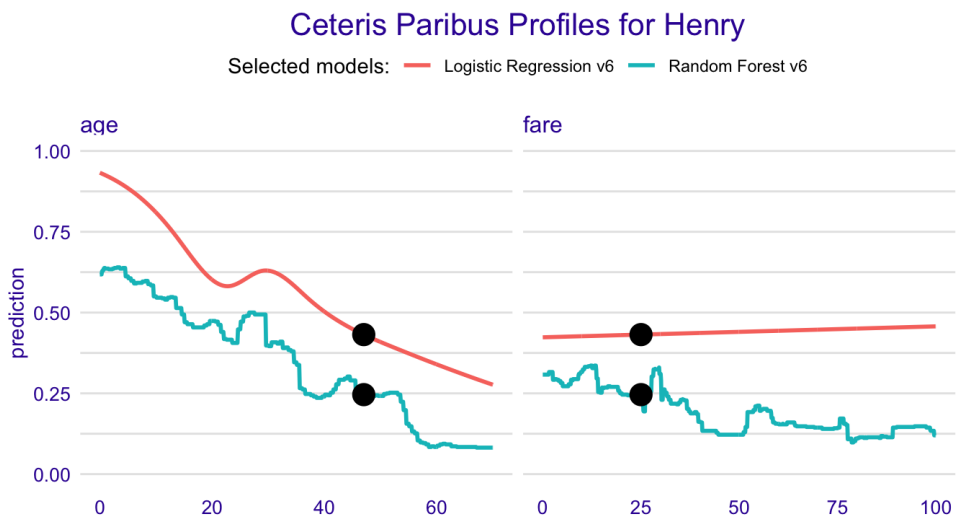


Figure 11.10: Champion-challenger comparison of the `titanic_lmr_v6` and `titanic_rf_v6` models. Profiles for different models are color-coded.

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.

Chapter 10 Local Interpretable Model-agnostic Explanations (LIME)

10.1 Introduction

Ceteris-paribus (CP) profiles, introduced in Chapter 11, are suitable for models with a small number of interpretable explanatory variables. In case of such models it makes sense to explore each variable separately and analyze how does it affect model predictions.

Break-down (BD) plots and plots of Shapley values, introduced in Chapters 7 and 9, respectively, are most suitable for modela with a small or moderate number of explanator variables. These plots do not offer as detailed information as the CP profiles, but can include more variables.

None of those approaches is well-suited for models with a large number of explanatory variables. Such models with even thousands of variables are not uncommon in, for instance, genomics. Also, if most of explanatory variables are binary, then CP profiles and BD plots are not very infromative. In such cases, sparse explainers offer a useful alternative. The most popular example of such explainers are Local Interpretable Model-agnostic Explanations (LIME) and their modifications.

The LIME method was originally proposed in (Ribeiro, Singh, and Guestrin 2016). The key idea behind this method is to locally approximate a black-box model by a simpler white-box model, which is easier to interpret. In this chapter, we describe the approach.

10.2 Intuition

The intuition behind LIME is explained in Figure 10.1. We want to understand the predictions of a complex black-box model around a single instance of interest. The model presented in Figure 10.1 is 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 white-box model that will locally approximate the predictions of the black-box model. The white-box model may then serve as a ‘’local explainer” for the more complex model.

We may select different classes of white-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.

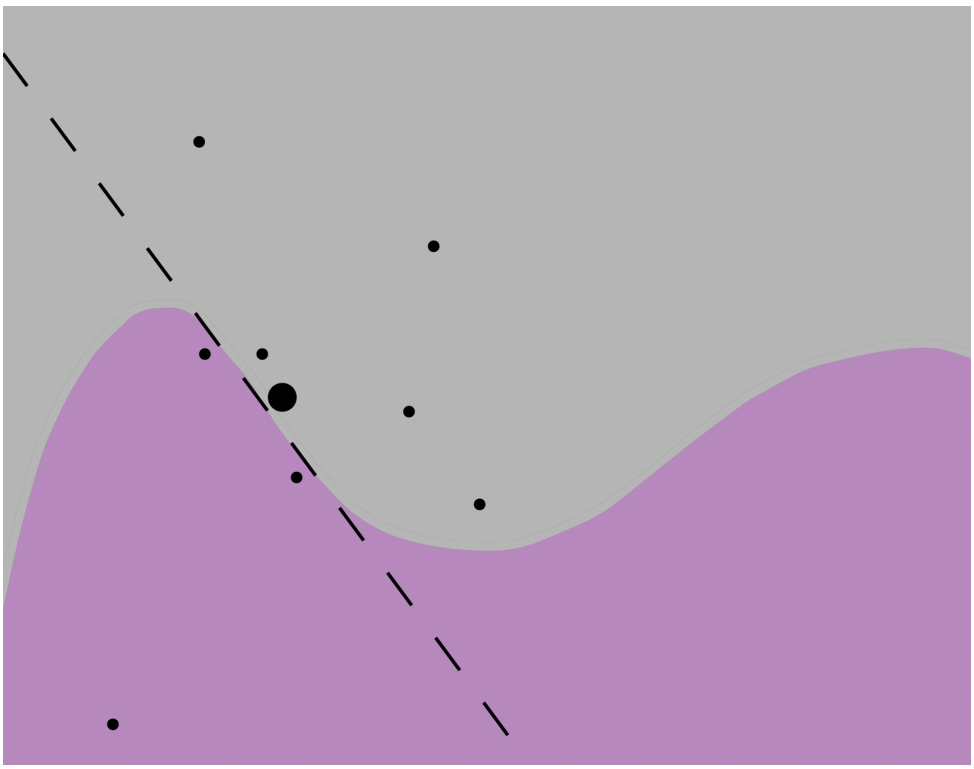


Figure 10.1: (fig:limeEx) The idea behind local model approximations. The axes represent the values of two continuous explanatory variables in a binary-classification mode. The colored areas for which combinations of the variables the model classifies the observation to one of the two classes. To ‘’explain” the prediction for the instance of interest (the large black dot), an artificial dataset around it is used to construct a simpler white-box model (here, a logistic-regression model, indicated by the dashed line) that locally approximates the predictions of the black-box model.

10.3 Method

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. To find the required approximation, We can can consider the following ''loss function,"

$$L(f, g, \Pi_{x_*}) + \Omega(g),$$

where model $g()$ belongs to class G, Π_{x_*} of x^* defines a neighborhood of x_* in which approximation is sought, $L()$ is a goodness-of-fit measure (e.g., the likelihood), 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 apply to different data spaces. The black-box model (function) $f(x) : \mathcal{X} \to \mathcal{R}$ is defined on the original, lare-dimensional space $\mathcal{X}$. The white-box model (function) $g : \mathcal{X}' \to \mathcal{R}$ applies to a lower-dmension, more interpretable space $\mathcal{X}'$. We will present some examples of $\mathcal{X}'$ in the next section. For now we will just assume that function $h()$ transforms $\mathcal{X}$ into $\mathcal{X}'$.

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

```
Input: x - observation to be explained
Input: N - sample size for the white-box model
Input: K - number of variables for the white-box model
Input: similarity - distance function between observations in the original input
1. Let x' = h(x) be an interpretable version of x
2. for i in 1...N {
3.   z'[i] <- sample_around(x')
     # prediction for a new observation.
     # z[i] is created based on 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 most important variables based on new dataset (y', x') with weights w'.

The practical implementation of the idea involves three important steps, which are discussed in the subsequent sub-sections.

10.3.1 Interpretable data representation

As it has been mentioned, the black-box model $f()$ and the white-box model $g()$ may apply to different data spaces. For example, let's consider a VGG16 neural network (Simonyan and Zisserman 2015) trained for ImageNet data. The model uses an image of the size of 244×244 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., it 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 white-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 categires is often used.

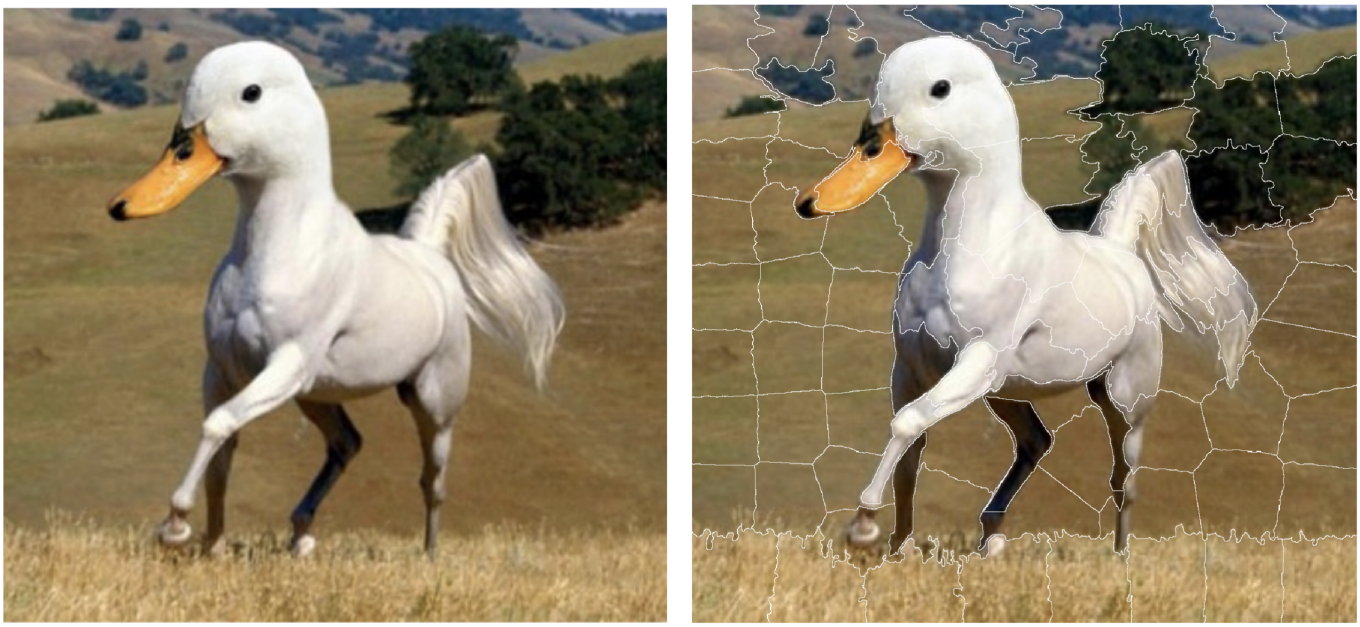


Figure 10.2: (fig:duckHorse06) The left panel shows an ambiguous picture, half-horse and half-duck. The right panel shows 100 superpixels identified for this figure. Source: www.rowsdowr.com

10.3.2 Sampling around the instance of interest

To develop the locally-approximation white-box model, new data points around the instance of interest are needed. 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. For this reason, the data for the development of the white-box model are often created by using perturbations of the instance of interest.

For a set of binary variables, the common choice is to change (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 (**???**) and (Molnar 2019) adds some Gaussian noise to continuous variables. In (Pedersen and Benesty 2019) continuous variables are discretized with the use of quintiles and the perturbations are don on discretized variables. In (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.

10.3.3 Developing the white-box model

Once the new data were sampled around the instance of interest, we may attempt to develop an interpretable white-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 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.

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 10.3 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 top K superpixels 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 colour are influential for the prediction ''standard poodle".

Figure 10.3: (fig:duckHorse04) LIME for two predictions (‘’standard poodle” and ‘’goose”) obtained by the VGG16 network with ImageNet weights for the half-duck, half-horse image. Source: https://twitter.com/finmaddison/status/352128550704398338

10.4 Example: Titanic data

Let us consider the random-forest model `titanic_rf_v6` (see Section 5.1.3 and passenger `johny_d` (see Section 5.1.5) as the instance of interest in the Titanic data.

Figure 10.4 presents explanations generated by the LIME method. In this example interpretable data representation is in the form of a binary vector. Each variable is dychotomized into two levels. For example `age` is transformed into a binary variable `<= / >` than 15.36, `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. Figure 10.4 shows coefficients estimated in the 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 orignially conitinuous variable is used. On the other hand, for class, a dichotomized version based on the combination of several original categories is used.

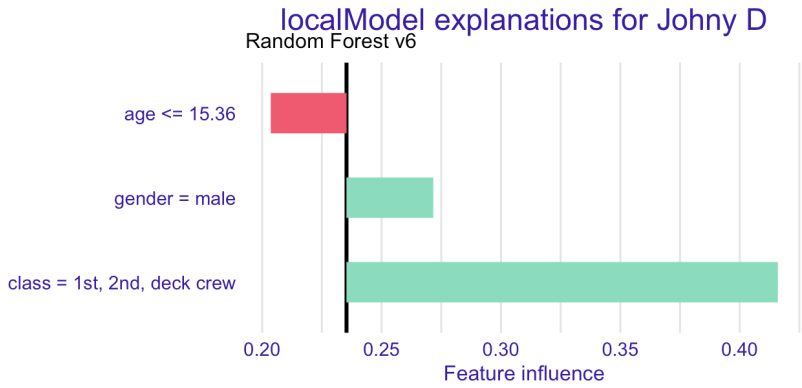


Figure 10.4: (fig:LIMEexample01) LIME method for the prediction for `johny_d` for the random-forest model `titanic_rf_v6` and the Titanic data. Presented values are beta coefficients in the K-LASSO model fitted locally to the response from the original model.

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 and change point method (Picard 1985) to find a local discretization. Different implementations of LIME differ in the way how the interpretable feature space is created.
Figure 10.5 illustrates how the two levels for age can be extracted from the Ceteris Paribus profile with the change point method.

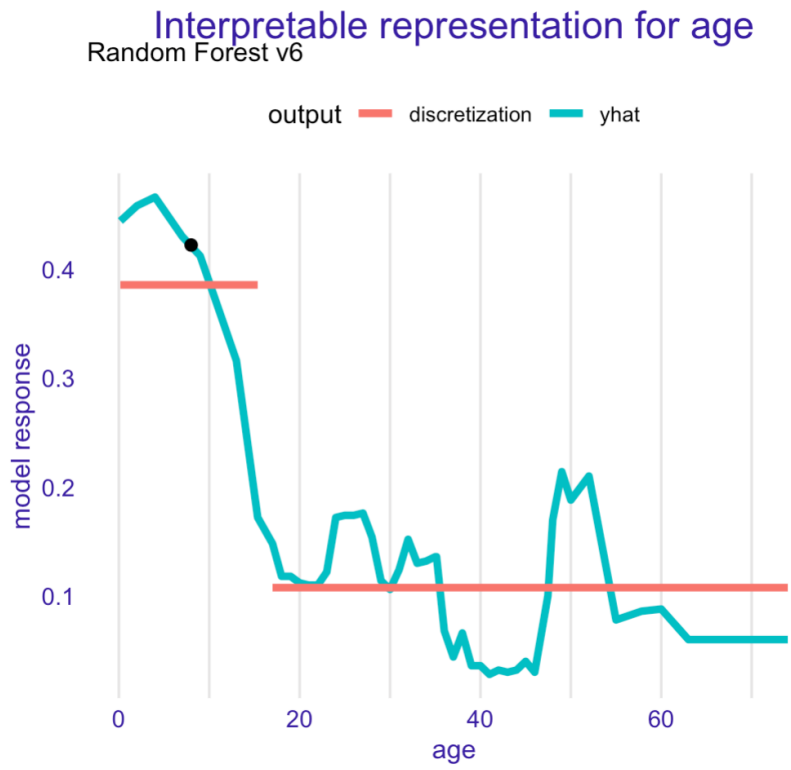


Figure 10.5: (fig:LIMEexample02) Interpretable variable generated for age. LIME method for the prediction for `johny_d` for the random-forest model `titanic_rf_v6` and the Titanic data.

10.5 Pros and cons

As mentioned by (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 tranformation 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. 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 variabes, predictions are easier to explain. The LIME method can be applied to complex, high-dimensional models.

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 white-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 white-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 straghtforward.

10.6 Code snippets for R

LIME and similar methods are implemented in various R and Python packages. For example, `lime` (Pedersen and Benesty 2018) 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.

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 5.1.7.

```
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 specifiy 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 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_pe
lime_expl

#      model_type case label label_prob  model_r2 model_intercept model_predictio
#1 classification    1    no      0.602 0.5806297       0.5365448        0.580593
#2 classification    1    no      0.602 0.5806297       0.5365448        0.580593
#3 classification    1    no      0.602 0.5806297       0.5365448        0.580593
#4 classification    1    no      0.602 0.5806297       0.5365448        0.580593
#  feature feature_value feature_weight  feature_desc                 data   prec
#1    fare            72     0.00640936  21.00 < fare 1, 2, 8, 0, 0, 72, 4 0.602,
#2  gender             2     0.30481181 gender = male 1, 2, 8, 0, 0, 72, 4 0.602,
#3   class             1    -0.16690730   class = 1st 1, 2, 8, 0, 0, 72, 4 0.602,
#4     age             8    -0.10026475     age <= 22 1, 2, 8, 0, 0, 72, 4 0.602,
```

Result is a table with coefficients for the K-LASSO method. 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. Fist 7 columns are duplicated and they all summaries how well the local surrogate K-LASSO model fit to the black-box model. The `feature_weight` columns shows the β 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

$$\hat{y} = 0.00640936 * 1_{fare>21} + 0.30481181 * 1_{gender=male} - 0.16690730 * 1_{class=1st} - 0.100264$$

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

Color correspond to the sign of the β coefficient while length of the bar corresponds to the absolute value of β coefficient in the K-LASSO method.

```
plot_features(lime_expl)
```

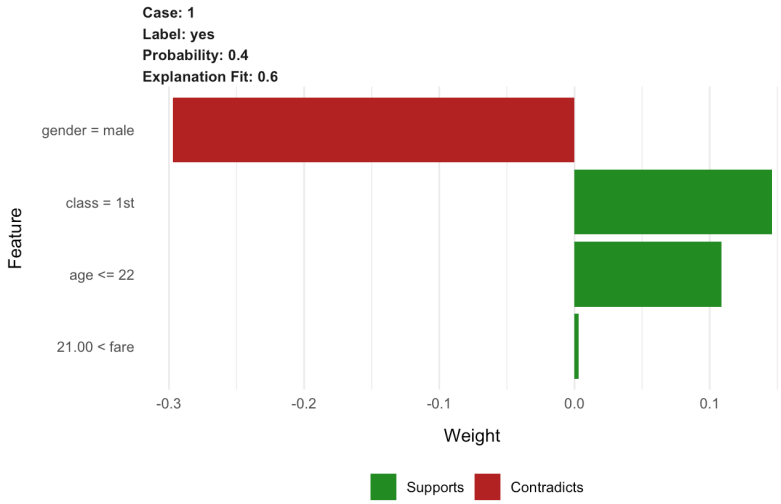


Figure 10.6: (fig:limeExplLIMETitanic) LIME-method results for the prediction for `johny_d` for the random-forest model `titanic_rf_v6` and the Titanic data, generated by the `lime` package.

10.6.2 The localModel package

The key elements of the `localModel` package are functions `DALEX::explain()`, which creates an explainer, and `individual_surrogate_model()`, which develops the local white-box model.

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

```
library("localModel")

localModel_rf <- DALEX::explain(model = titanic_rf_v6,
                     data = titanic[,colnames(johny_d)])
localModel_lok <- individual_surrogate_model(localModel_rf, johny_d,
                                       size = 1000, seed = 1313)
localModel_lok
#   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 `localModel_lok` one will find β 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`.

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 10.7. Bars correspond to β coefficients in the LASSO model.

```
plot(localModel_lok)
```

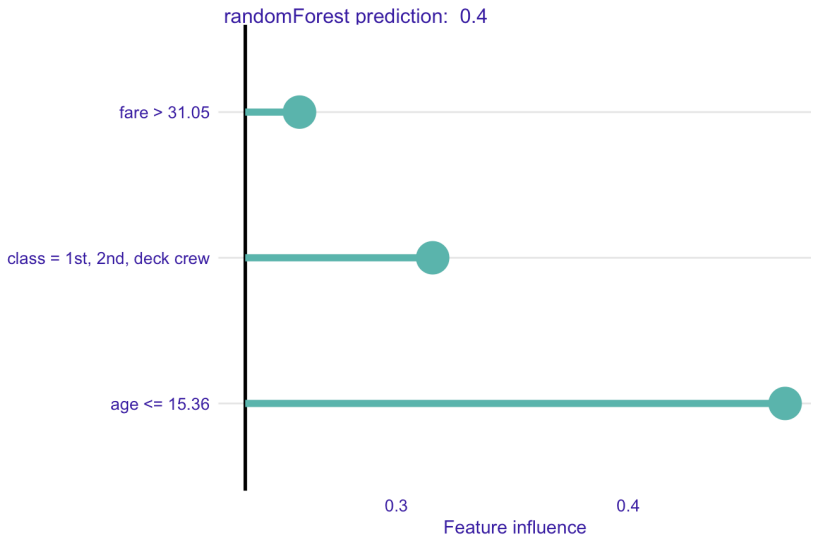


Figure 10.7: (fig:limeExplLocalModelTitanic) LIME-method results for the prediction for `johny_d` for the random-forest model `titanic_rf_v6` and the Titanic data, generated by the `localModel` package.

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 white-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 β 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 10.8. 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)
```

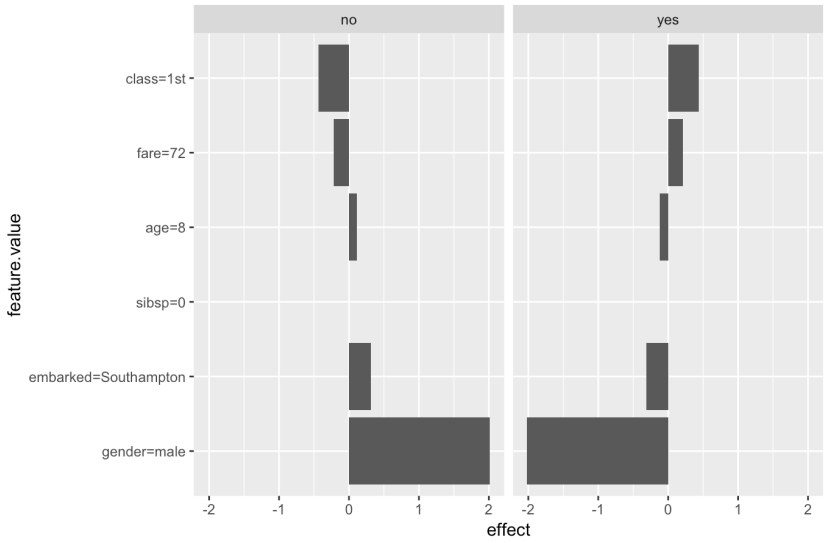


Figure 10.8: (fig:limeExplIMLTitanic) LIME-method results for the prediction for `johny_d` for the random-forest model `titanic_rf_v6` and the Titanic data, generated by the `iml` package.

References

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. 2018. *Lime: Local Interpretable Model-Agnostic Explanations*. https://CRAN.R-project.org/package=lime.

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.

Chapter 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 (Štrumbelj and Kononenko 2010) and (Štrumbelj and Kononenko 2014). It was widely adopted after the publication of the 2017 paper (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 were an unification of a collection of different commonly used techniques for model explanations.

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 5.1.5) 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.

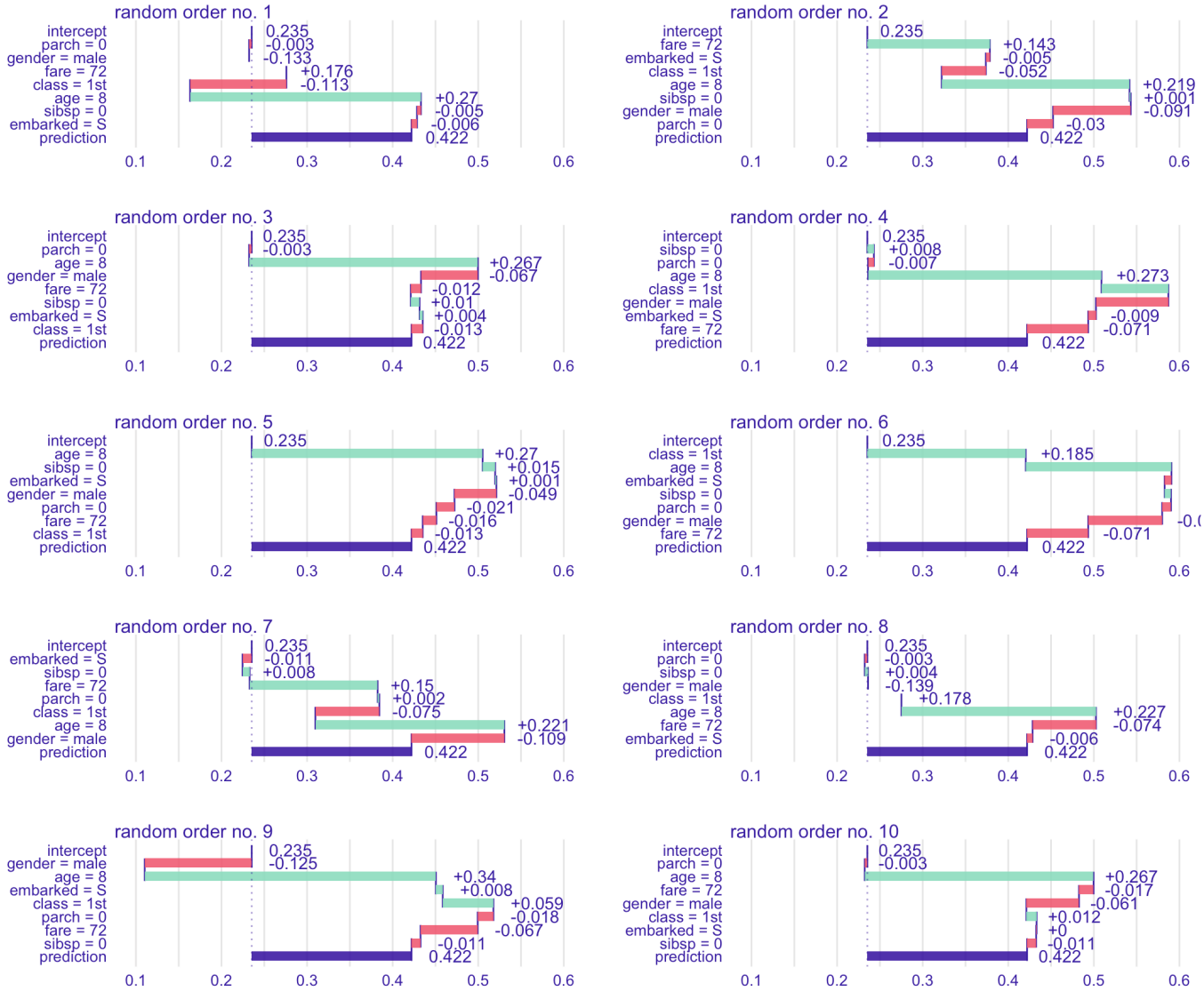


Figure 9.1: (fig:shap10orderings) Break-down plots for ten random orderings of explanatory variables for the prediction for `johny_d` for the random-forest model for the Titanic dataset. Each panel presents a single ordering, indicated by the order of the rows in the plot

To remove the influence of the ordring 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. The plot indicates that the most important variables, from the point of view of the prediction for `johny_d` , are `age` and `gender` .

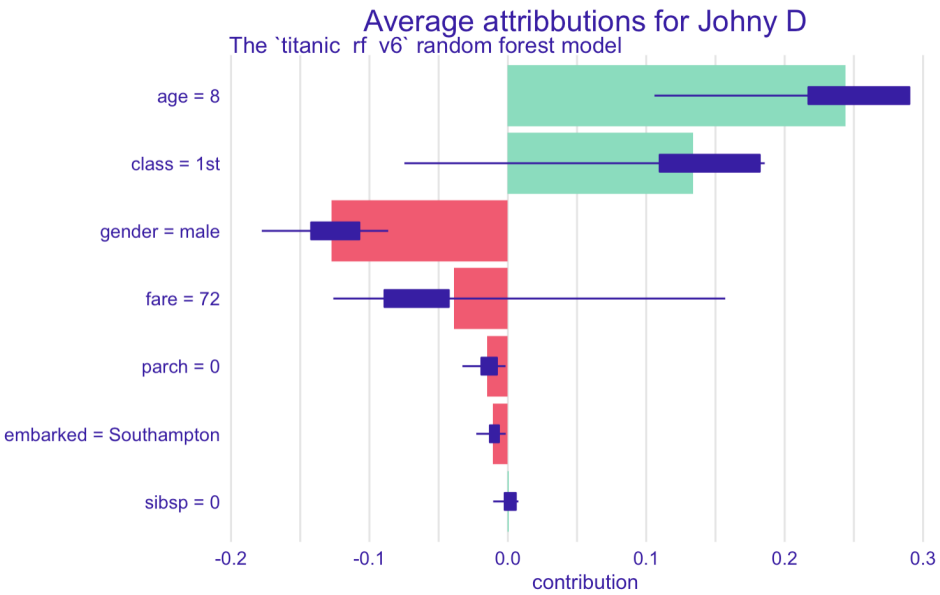


Figure 9.2: (fig:shapOrdering) Average contributions for ten random orderings. Red and green bars present the averages. Box-plots summarize the distribution of contributions for each explanatory variable across the orderings.

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 SHAP values when referring to values calculated with the SHAP method.

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 (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 SHAP 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 SHAP value from (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 (Štrumbelj and Kononenko 2014). An efficient implementation of computations of SHAP values was introduced in (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 predicition, that is,

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

```
## Preparation of a new explainer is initiated
##   -> model label       :  Random Forest v6
##   -> data              :  2207  rows  7  cols
##   -> target variable   :  2207  values
##   -> predict function  :  yhat.randomForest  will be used (  default  )
##   -> predicted values  :  numerical, min =  0 , mean =  0.2353095 , max =  1
##   -> residual function :  difference between y and yhat (  default  )
##   -> residuals         :  numerical, min =  -0.892 , mean =  0.0868473 , max =
##   -> model_info        :  package randomForest , ver. 4.6.14 , task classifica
##   A new explainer has been created!
```

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 5.1.5) 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 SHAP values across the orderings. It is clear that the young age of Johny D results in a positive contribution for all orderings. The SHAP value is equal to 0.2525. On the other hand, the effect of gender is in all cases negative, with the SHAP 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 SHAP 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.

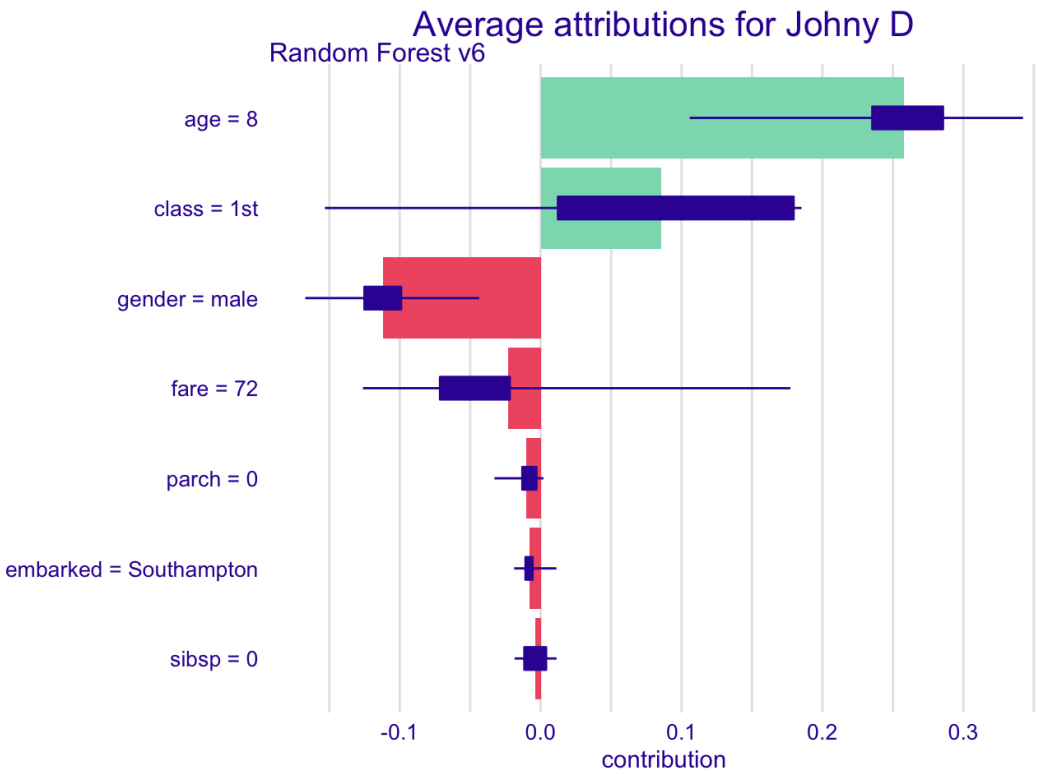


Figure 9.3: (fig:shappJohny02) Variable contributions for the prediction for `johny_d` for the random-forest model `titanic_rf_v6` and the Titanic data for 25 random orderings. Box-plots summarize the distribution of the contributions for each explanatory variable across the orderings. Red and green bars present the SHAP values.

In most applications the detailed information about the distribution of variable contributions across the considered orderings of explanatory variabled will not be necessary. Thus, one could simplify the plot by presenting only the SHAP values, as in Figure 9.4.

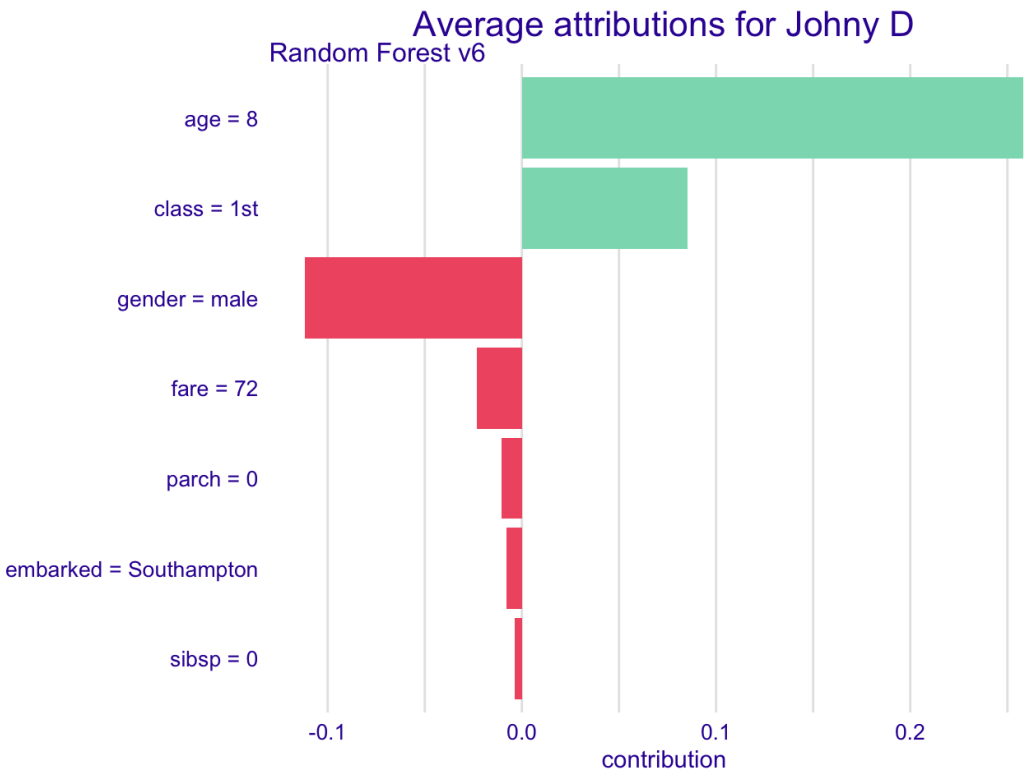


Figure 9.4: (fig:shappJohny01) SHAP values for the prediction for `johny_d` for the random-forest model `titanic_rf_v6` and the Titanic data for 25 random orderings.

Table 9.1 presents the SHAP values underlying the plot in Figure 9.4. Table: (#tab:shapOrderingTable) SHAP values for the prediction for `johny_d` for the random-forest model `titanic_rf_v6` and the Titanic data for 25 random orderings.

Variable	SHAP 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

SHAP values provide a uniform approach to decompose model predictions into parts that can be attributed additively to different explanatory variables. In (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 SHAP values is that they are based on the assumption of additivity of variable effects. If the model is not additive, then the SHAP 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 modelling 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 SHAP 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 SHAP values is time consuming. However, sub-sampling can be used to address the issue.

9.5 Code snippets for R

In this section, we present the key features of the `iBreakDown` R package (Gosiewska and Biecek 2019a) which is a part of the `DrWhy.AI` universe. The package covers all methods presented in this chapter. It is available on CRAN and GitHub. More details and examples can be found at https://modeloriented.github.io/iBreakDown/.

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: `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 5.1.7.

```
library("randomForest")
explain_rf_v6 <- archivist::aread("pbiecek/models/9b971")

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

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

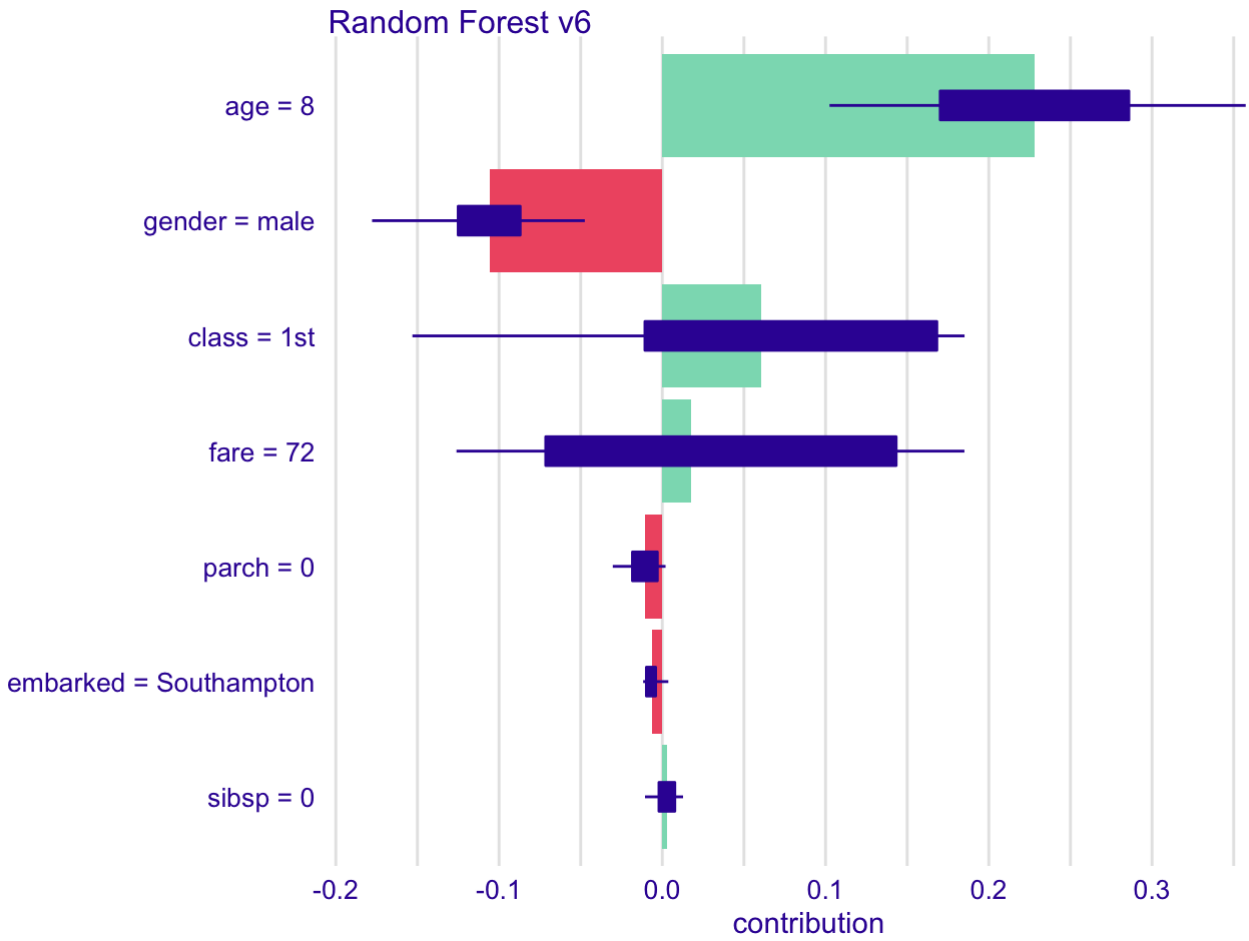
```
predict(explain_rf_v6, johny_d)
```

```
## [1] 0.422
```

With the help of function `shap()` from the `iBreakDown` we can re-create Figure 9.4. 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 whcih the SHAP 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 conctructs the plot that includes the SHAP values and the corresponding box-plots.

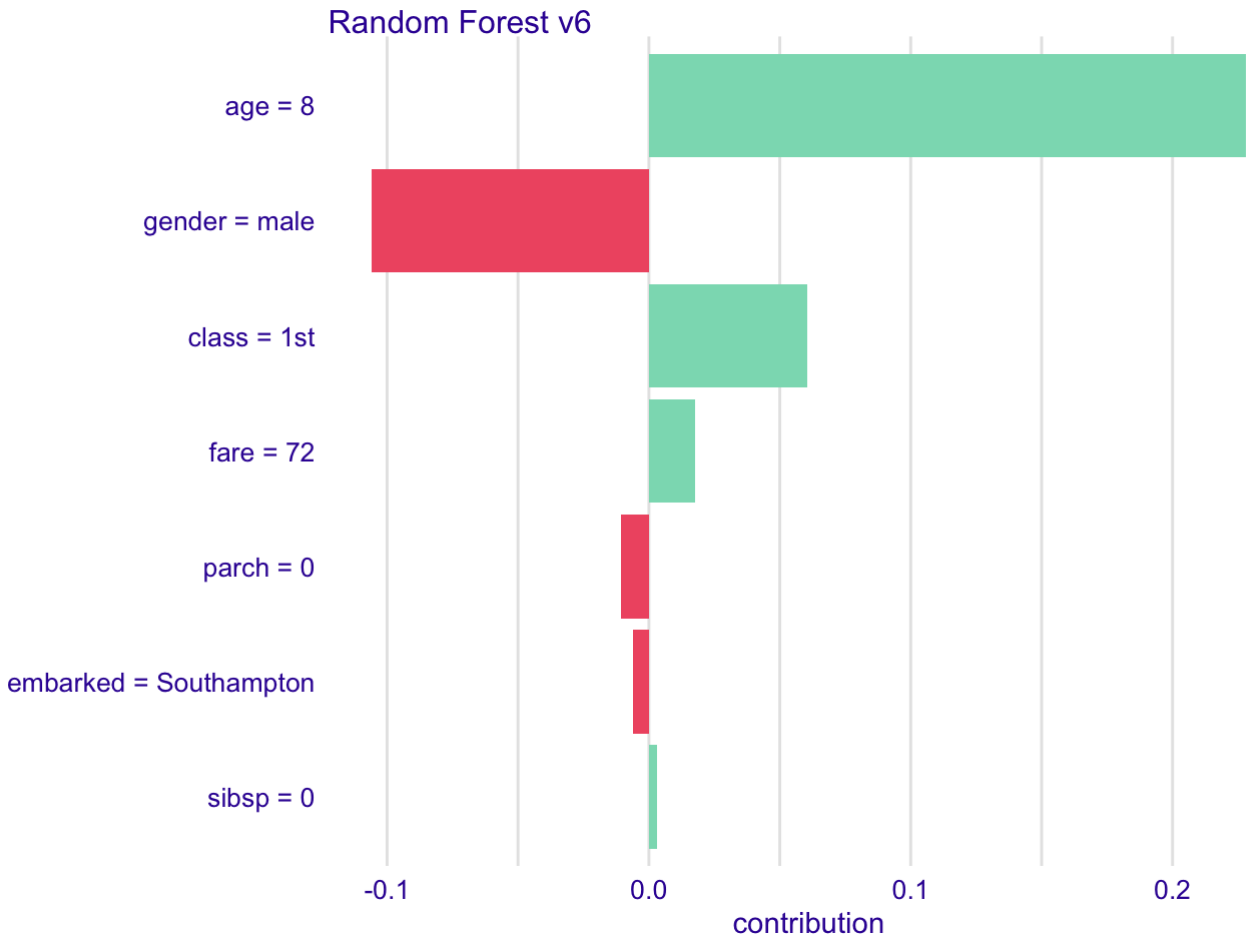
```
library("iBreakDown")

shap_johny <- shap(explain_rf_v6, johny_d, B = 25)
plot(shap_johny)
```



To obtain a plot with only SHAP values, as in Figure 9.3, we can use the `show_boxplots=FALSE` argument in the `plot()` function call.

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



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

```
shap_johny
```

```
##                                                      min           q1       median
## Random Forest v6: age = 8                     0.10243679  0.170255324  0.25260715
## Random Forest v6: class = 1st                -0.15308473 -0.010615541  0.06023416
## Random Forest v6: embarked = Southampton -0.01172814 -0.009733348 -0.00645672
## Random Forest v6: fare = 72                  -0.12608518 -0.071392841 -0.02088219
## Random Forest v6: gender = male              -0.17778251 -0.125022202 -0.10727820
## Random Forest v6: parch = 0                  -0.03033983 -0.018268237 -0.00697689
## Random Forest v6: sibsp = 0                  -0.01059538 -0.002020843  0.00516493
##                                                     mean           q3           ma
## Random Forest v6: age = 8                     0.227945700  0.285757816  0.35734753
## Random Forest v6: class = 1st                 0.060587041  0.168151790  0.18513547
## Random Forest v6: embarked = Southampton -0.006162356 -0.004091527  0.00369913
## Random Forest v6: fare = 72                   0.017683407  0.143228817  0.18506116
## Random Forest v6: gender = male              -0.105745029 -0.087078160 -0.04747349
## Random Forest v6: parch = 0                  -0.010581930 -0.003043951  0.00200996
## Random Forest v6: sibsp = 0                   0.002963697  0.007650204  0.01272224
```

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.