Explanatory Model Analysis

Explore, Explain and Examine Predictive Models

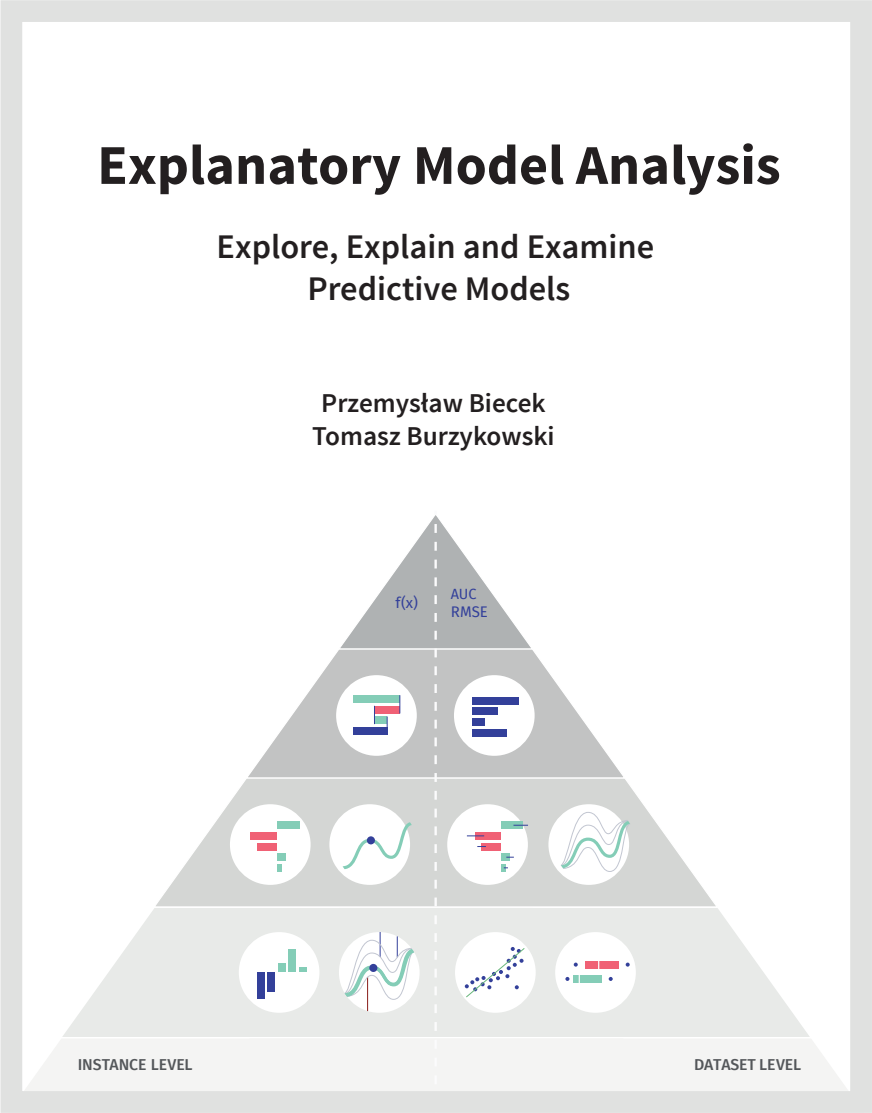
Przemyslaw Biecek and Tomasz Burzykowski

2020-03-08

Contents

List of Tables

List of Figures


Explanatory Model Analysis
Explore, Explain and Examine
Predictive Models
Przemysław Biecek
Tomasz Burzykowski
f(x)
AUC
RMSE
INSTANCE LEVEL
DATASET LEVEL

Why explore

0.1 Introduction

0.1.1 Notes to readers

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

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

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

0.1.2 The aim of the book

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

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

is used for various models that allow prediction of continuous variables. Prediction of nominal variables is called *classification*, and its beginning may be attributed to works of Ronald Fisher in 1936.

During the last century, many statistical models that can be used for predictive purposes have been developed. These include linear models, generalized linear models, regression and classification trees, rule-based models, and many others. Developments in mathematical foundations of predictive models were boosted by increasing computational power of personal computers and availability of large datasets in the era of „big data'' that we have entered.

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

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

(Diaz et al., 2018). These are examples of models and algorithms that led to serious violations of fairness and ethical principles. An example of situation when data drift led to deterioration in model performance is the Google Flu model, which gave worse predictions after two years than at baseline (Salzberg, 2014), (Lazer et al., 2014).

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

Figure 1 shows how the increase in the model complexity affects the relative importance of domain understanding vs. modeling vs. validation. Simplest models are usually built on top of a good understanding of the domain. Domain knowledge helps to create and select most important variables that can be transformed into predictive scores. Machine learning exploits the tradeoff between availability of data and domain knowledge. Flexible models can use massive data to learn good features and filter out bad ones. The effort is shifted from a deep understanding of the domain towards computationally heavy training of models. The validation part is of an increased importance because it creates a feedback loop with the modeling. Results from model validation lead to next decisions related to model training. This is different than in case of statistical hypothesis testing. Statistical hypotheses shall be stated in advance of data analysis and obtained p-values shall not interfere in the way how data or models were prepared.

What will be next? The increasing automation in the EDA (Exploratory Data Analysis) and modeling part of the process shift the focus towards the validation of models. The purpose of validation is not only to measure how good is the model but also what other risks are associated with models. Risks like concept drift, gender,

age or race bias. This book is about new methods that can be used for validation and justification.

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 mindset, key problems, and several examples of methods that can be used in model exploration.

0.1.3 A bit of philosophy: three laws of model explanation

In 1942, Isaac Asimov forumlated Three Laws of Robotics:

1) a robot may not injure a human being,
2) a robot must obey the orders given it by human beings, and
3) a robot must protect its own existence.

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

It's clear that we need to control the models and algorithms that may affect us. Thus, Asimov's laws are referred to in the context of

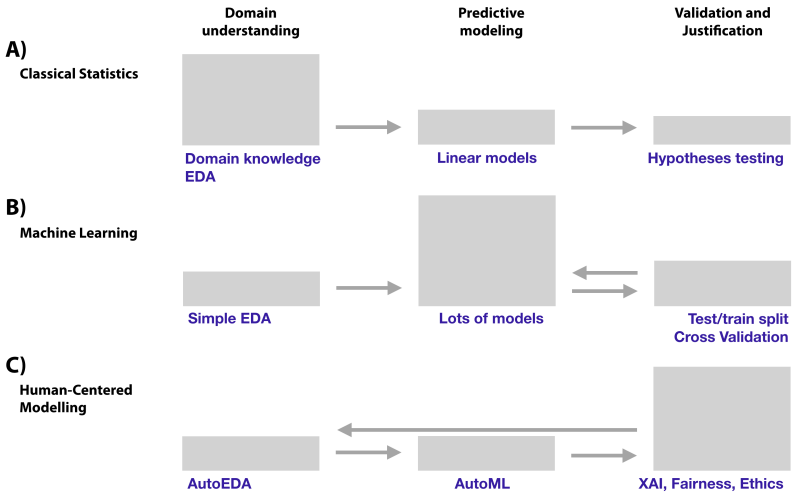


FIGURE 1 Shift in the relative importance and effort put in different phases of the data-driven modeling. (A) Statistical modeling is often based on deep understanding of the domain. Manual data exploration, consultations with domain experts, variable transformations lead to good models. Structures of models are often based on (generalized) linear models. Model verification is done through hypothesis testing. (B) Machine learning modeling is often based on elastic models fitted to large volumes of data. Domain exploration is often shallow while the focus is based on predictive performance. Lots of attention is put in cross validation and other strategies that deal with overfitting. (C) What will be next? Human-centered modeling? Better tools for auto EDA and auto ML will shift focus into the part related with validation against the domain knowledge like fairness, bias or new techniques for data exploration. Arrows show feedback loops in the modeling process. The feedback loop is even larger now, as the results from model validation are helping also in the domain understanding.

the discussion around Ethics of Artificial Intelligence. Initiatives to formulate principles for AI development have been undertaken, for instance, in the UK [Olhede & Wolfe, Significance 2018, 15: 6-7]. Following Asimov's approach, we propose three requirements that any predictive model should fulfill:

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

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

0.1.4 The structure of this book

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

Before embarking on the description of the methods, in Chapter 0.2, we provide a short introduction to the process of data exploration and model assembly along with notation and definition of key concepts that are used in consecutive chapters. In chapters

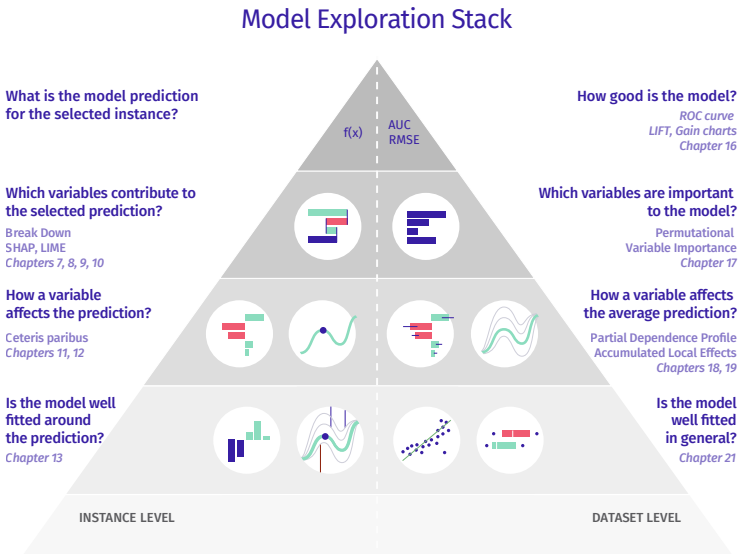


FIGURE 2 Stack with model exploration methods presented in this book. Left side is focused on instance-level explanation while the right side is focused on dataset-level explanation. Consecutive layers of the stack are linked with a deeper level of model exploration. These layers are linked with law's of model exploration introduced in Section 0.1.3

0.3 and 0.4, 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 0.5, we describe two datasets that are used throughout the book to illustrate the presented methods and tools.

Rest of the book is structured in Figure 2.

The **Instance-level** part of the book consists of Chapters **??**-**??**. Chapters **??**-**??** present methods to decompose model predictions into variable contributions. In particular, Chapter **??** introduces Break-down (BD) plots for models with additive effects. On the other hand, Chapter **??** presents a method that allows for interactions. Finally, Chapter **??** describes SHAP (Lundberg and Lee, 2017) an alternative method for decomposing model predictions that is closely linked with Shapley values (Shapley, 1953) devel-

oped originally for cooperative games. Chapter **??** presents a different approach to explanation of single-instance predictions. It is based on a local approximation of a black-box model by a simpler, glass-box one. In this chapter, we discuss the Local Interpretable Model-Agnostic Explanations (LIME) method (Ribeiro et al., 2016). These chapters corresponds to the second layer of the stack in Figure 2.

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

The **Dataset-level explainers** part of the book consists of Chapters **??**-**??**. These chapters present methods in the same order as appeared in the Model Exploration Stack in Figure 2. Chapter **??** shows selected measures for model benchmarking along with performance measures for classification and regression models. On top of these measures, the Chapter **??** 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 **??** and **??**. 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 **??** that summarises diagnostic techniques for model residuals.

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

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

0.1.5 Terminology

It is worth noting that, when it comes to predictive models, the same concepts have often been given different names in statistics and in machine learning. For instance, in the statistical-modelling literature, one refers to „explanatory variables,'' with „independent variables,'' „predictors,'' or „covariates'' as often-used equivalents. Explanatory variables are used in the model as means to explain (predict) the „dependent variable,'' also called „predicted'' variable or „response.'' In machine-learning terminology, „input variables'' or „features'' are used to predict the „output'' or „target'' variable. In statistical modelling, models are fit to the data that contain „observations'', whereas in the machine-learning world a dataset may contain „instances" or „cases''. When we talk about values that define a single instance of a model in statistical modelling we refer to model „coefficients" while in machine-learning it is more common to use phrase model „parameters''. In statistics it is common to say that model coefficients are „estimated" while in machine learning it is more common to say that parameters are „trained'' or are obtained in the process of „model training''.

To the extent possible, in our book we try to consistently use

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

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

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

0.1.6 Glass-box models vs. black-box models

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

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

The most common classes of glass-box models are decision or re-

gression trees, as an example in Figure 3, rules, or models with an explicit compact structure, like the following model for obesity based on the BMI index.

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

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

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

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

Note that some glass-box models, like the decision tree model presented in Figure 3 by design satisfies explainability laws introduced

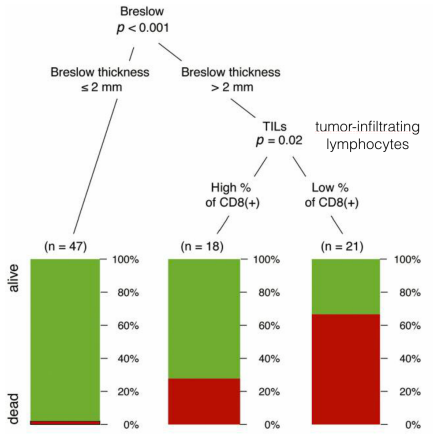


FIGURE 3 Example classification tree model for melanoma risk patients based on [@BILLCD8]. The model is based on two explanatory variables, Breslow thickness and Tumor infiltration lymphocytes. These two variables lead to three groups of paritents with different odds of survival.

in Section 0.1.3. For *Prediction's validation* we see in each node how many patients fall in a given category. For *Prediction's justification* we see which variables are used in every decision path. For *Prediction's speculation* we can trace how changes in particular variables will affect the model prediction. We can, of course, argue if the model is good or not, but obviously the model structure is transparent.

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

0.1.7 Model-agnostic vs. model-specific approach

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

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

Of course, the list of model classes with dedicated collections of

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

For these reasons, in our book we focus on model-agnostic techniques. In particular, we prefer not to assume anything about the model structure, as we may be dealing with a black-box model with an unspecified structure. Often we do not have access to model parameters just to a specified Application Programming Interface (API) that allows for querying remote models (for example in Microsoft Cognitive Services (Azure, 2019)). In that case, the only operation that we may be able to perform is the evaluation of a model for a specified data.

However, while we do not assume anything about the structure of the model, we will assume that the model operates on p-dimensional vector of variables/features and, for a single observation, it returns a single value (score/probability) which is a real number. This assumption holds for a broad range of models for data such as tabular data, images, text data, videos, etc. It may not be suitable for, e.g., models with memory like sequence-to-sequence models (Sutskever et al., 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.

0.1.8 What is in this book and what is not

The area of model exploration and explainability is quickly growing and is present in many different flavors. Instead of showing every existing method (is it really possible?) we rather selected a subset of consistent tools that are a good starting set for model exploration. Our focus was on the impact of the model exploration

and explanation tools rather than on selected methods. We believe that once we become aware of potential beyond visual model exploration, once we will learn a language of model explanation, we will improve our process of data modeling.

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

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

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

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

instance, in *Applied Predictive Modeling* by Max Kuhn and Kjell Johnson (Kuhn and Johnson, 2013).

0.1.9 Acknowledgements

This book has been prepared using the `bookdown` package (Xie, 2018), created thanks to the amazing work of Yihui Xie. Figures and tables are created in R language for statistical computing (R Core Team, 2018) with numerous libraries that support predictive modeling. Just to name few frequently used in this book `randomForest` (Liaw and Wiener, 2002), `ranger` (Wright and Ziegler, 2017), `rms` (Harrell Jr, 2018), `gbm` (Ridgeway, 2017) or `caret` (from 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.

0.2 Model Development

0.2.1 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.

0.2.2 The Process

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

The most known methodology for data science projects is CRISP-DM (Chapman et al., 1999), (Wikipedia, 2019) which is a tool agnostic procedure. The key component of CRISP-DM is the break down of the whole process into six phases, that are iterated: business understanding, data understanding, data preparation, modeling, evaluation and deployment. CRISP-DM is general, it was designed for any data science project. For predictive models some methodologies are introduced in (Grolemund and Wickham, 2019) and (Hall, 2019).Both are focused on iterative repetitions of some phases. Figure 4 presents a variant of iterative process divided into five steps. Data preparation is needed prior to any modeling. Better data is needed for better models. On the other hand, garbage-in garbage-out. Once the data is gathered, steps that are usually highlighted are Data understanding, Model assembly and Model audit.

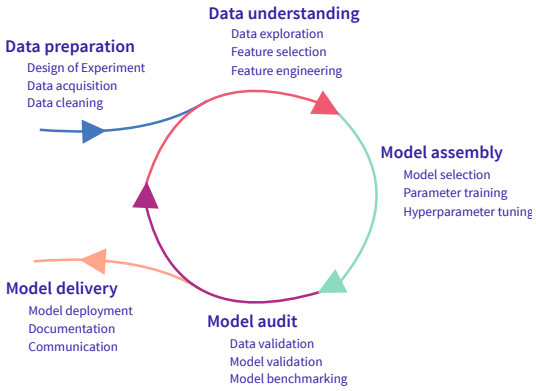


FIGURE 4 Lifecycle of predictive model can be decomposed into five tasks. First we need data that is poured into the model development cycle. The model development is highly iterative, learn something new about the data, assemble a new model based on current understanding, and validate the new model. Repeat these steps as long as needed to be satisfied with model performance. Once the model is created we can deliver the model to the production along with required tests and documentation.

This is the common thinking about model development. Repeat these steps until some convergence, e.g. repeat until best model is identified.

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 et al., 1999), (Boehm, 1988). One can think about MDP as an extension of process introduced in Figure 4. What is important is to notice that consecutive iterations are not identical. Our knowledge increases during the process and consecutive iterations are performed with different goals in mind.

This is why MDP is build as an untangled version of Figure 4. The MDP process is shown in Figure 5. Each vertical stripe is a single run of the cycle. First iterations are usually focused on *formulation of the problem*. Sometimes the problem is well stated, however it's a rare situation valid maybe only for kaggle competitions. In

FIGURE 5 Overview of the Model Development Process. Horizontal axis show how time passes from the problem formulation to the model decommissioning. Vertical axis shows tasks are performed in a given phase. Each vertical strip is a next iteration of cycle presented in Figure 4

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

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

As suggested in the title of this book, three primary applications are: exploration, explanation and debugging. *Exploration* refers to situations in which we better understand the data and the domain.

Presented techniques can be used to speed up the variable engineering or variable selection. *Explanation* refers to situations in which we are interested in decision paths beyond particular predictions. *Debugging* refers to situations in which we want to understand weak points of a model and correct them. These applications target phases Data understanding, Model assembly and Model audit.

In this book we present various examples based on three use cases. Two introduced in Chapter 0.5 (binary classification in surviving Titanic sinking and regression in apartments pricing) and one in Chapter **??** (estimation of soccer player value based on its skills). Due to space limitation we do not show the full life cycle of these problems, but we are focused on phases Crisp modeling and Fine tuning.

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

0.2.3 Notation

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

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

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

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

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

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

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

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

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

$$r_i = y_i - f(x_i) = y_i - \hat{y}_i. \tag{0.1}$$

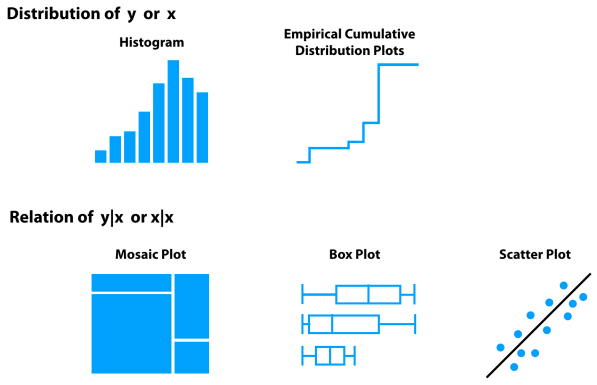


FIGURE 6 Basic methods for visual exploration. Histogram for distribution of continuous or categorical variables, empirical cumulative distribution for continuous variables. Mosaic plot for relation between two categorical variables, boxplots for relation between continuous and categorical variables or scatterplot for relation between two continuous variables.

0.2.4 Data exploration

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

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

In this book we will rely on five visual methods for data exploration presented in Figure 6. Two of them are used to present distribution of explanatory or target variables; three others are used to explore pairwise relations between variables.

Distribution of categorical variable is summarized with a barplot,

distribution of numerical variable is summarized with a histogram or empirical cumulative distribution function.

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

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

0.2.5 Model training

In predictive modeling, we are interested in a distribution of a dependent variable Y given vector x_*. The latter contains values of explanatory variables. In the ideal world, we would like to know the conditional distribution of Y given x_*. In practical applications, however, we usually do not predict the entire distribution, but just some of its characteristics like the expected (mean) value, a quantile, or variance. Without loss of generality we will assume that we model the conditional expected value $E_Y(Y|x_*)$.

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

"good" approximation of the conditional expected value, it should be reflected in its satisfactory predictive performance.

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

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 $\theta \in \Theta$ that maximize some loss function $L(y, f_\theta(X))$. For models with large parameter spaces it is common to add additional term $\lambda(\theta)$ that control the model complexity.

$$\hat{\theta} = \arg\min_{\theta \in \Theta} L(y, f_\theta(X)) + \lambda(\theta). \tag{0.2}$$

For statistical models it is common to assume some family of probability distributions for $y|x$. In such case the loss function L may be defined as a minus log-likelihood function for θ. Likelihood is probability of observing $y|x$ 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 (0.2) become

$$\hat{\theta} = \arg\min_{\theta \in \Theta} ||y - X\beta||_2 + \lambda(\beta). \tag{0.3}$$

For linear regression, the penalty term $\lambda(\beta)$ is equal to 0, and optimal parameters β in equation (0.3) have close analytical solution $\hat{\beta} = (X^T X)^{-1} X^T y$. In ridge regression the penalty $\lambda(\beta) = \lambda||\beta||_2$ and also (0.3) have analytical solution $\hat{\beta} = (X^T X + \lambda I)^{-1} X^T y$. For

LASSO regression the penalty $\lambda(\beta) = \lambda||\beta||_1$ and 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, 2002) or (Kuhn and Johnson, 2013).

0.2.6 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 **??** overviews measures for model performance. These measures are usually used to trace the progress in model development.
- assess the feature importance. Section **??** 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 **??** – **??** 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 **??** 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 **??** introduces

Ceteris Paribus profiles that helps in a what-if analysis for a model.
- validated local fit for a model. Section **??** 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 **??** – **??** show different techniques like SHAP, LIME or Break Down for local exploration of a model.

0.3 Do-it-yourself With R

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

0.3.1 What to install?

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

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

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

The most important one is the `DALEX` package in version 1.0 or newer. It is the entry point to solutions introduced in this book.

The package can be installed by executing the following command from the R command line:

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

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

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

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

0.3.2 How to work with DALEX?

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

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

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

around a model that makes it easier to explore and explain the model.

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

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

For an example, see Section 0.5.1.7.

0.3.3 How to work with `archivist`?

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

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

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

Once the package has been installed, function `aread()` can be used to retrieve R objects from any remote repository. For this book, we use a GitHub repository `models` hosted at https://github.com/

pbiecek/models. For instance, to download a model with the md5 hash `ceb40`, the following command has to be executed:

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

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

0.4 Do-it-yourself With Python

0.5 Data sets and models

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

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

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

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

FIGURE 7 Titanic sinking by Willy Stöwer

0.5.1 Sinking of the RMS Titanic

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

- *gender*, person's (passenger's or crew member's) gender, a factor (categorical variable) with two levels (categories) `male` (78%) and `female` (22%);
- *age*, person's age in years, a numerical variable; the age is given in (integer) years, range 0 – 74 years;
- *class*, the class in which the passenger travelled, or the duty class

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

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

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

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

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

0.5.1.1 Data exploration

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

Before exploring the data, we first do some pre-processing. In particular, the value of variables *age*, *country*, *sibsp*, *parch*, and *fare*

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

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

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

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

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

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

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

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

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

After imputing the missing values, we investigate the association between survival status and other variables. Most variables in the

Titanic dataset are categorical, except Age and Fare. In order to keep the exploration uniform we first transformed them into categorical variables. Figure 8 shows histograms for both variables. Age is discretized into 5 categories with cutoffs 5, 10, 20 and 30 while Fare is discretized with cutoffs 1, 10, 25, and 50.

Figures 9-12 present graphically the proportion non- and survivors for different levels of the other variables with the use of mosaic plots. The height of the bars (on the y-axis) reflects the marginal distribution (proportions) of the observed levels of the variable. On the other hand, the width of the bars (on the x-axis) provides the information about the proportion of non- and survivors. Note that, to construct the graphs for *age* and *fare*, we categorized the range of the observed values.

Figure 9 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 10, i.e., a higher proportion of survivors among those with 1-3 parents/children and 1-2 siblings/spouses aboard. Figure 11 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 12 does not suggest any noteworthy trends.

0.5.1.2 Logistic regression model

The dependent variable of interest, *survival*, is binary. Thus, a natural choice is to start the predictive modelling with logistic regression model. As there is no reason to expect a linear relationship between age and odds of survival, we use linear tail-restricted cubic

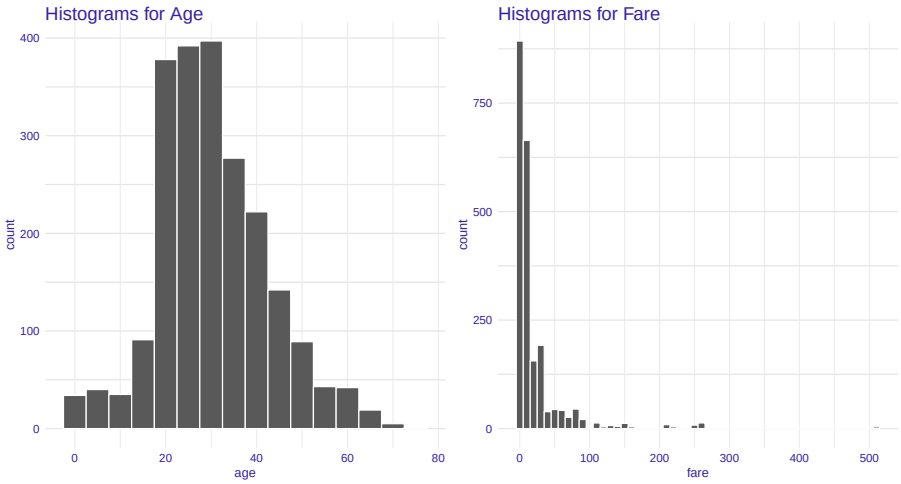


FIGURE 8 Histogram of Age and Fare for the Titanic data.

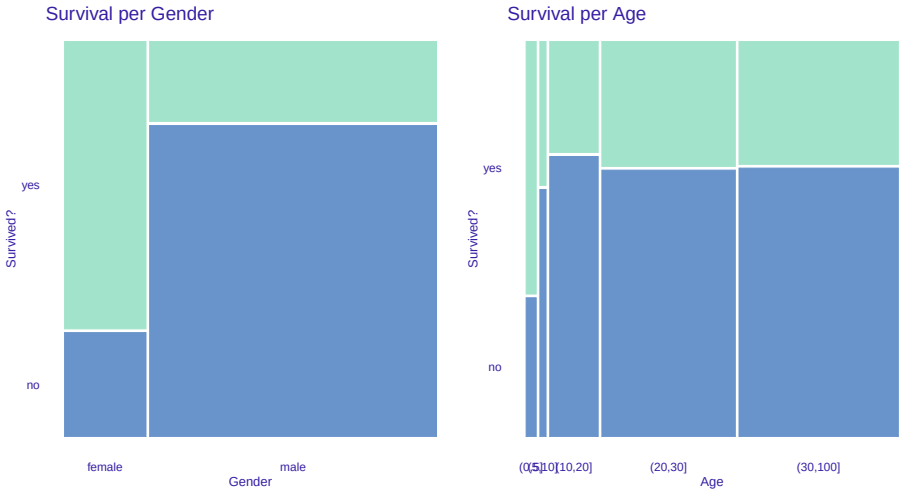


FIGURE 9 Survival status in groups defined be Gender and Age for the Titanic data.

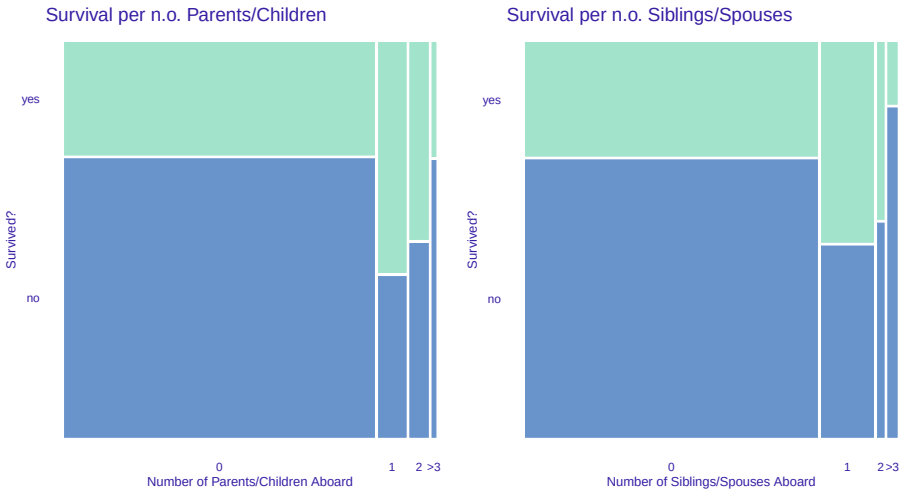


FIGURE 10 Survival according to the number of parents/children and siblings/spouses in the Titanic data.

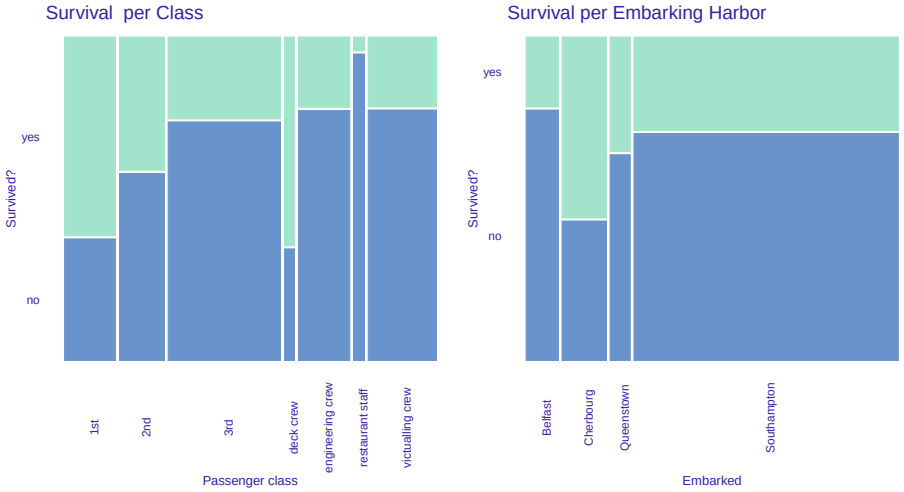


FIGURE 11 Survival according to the class and port of embarking in the Titanic data.

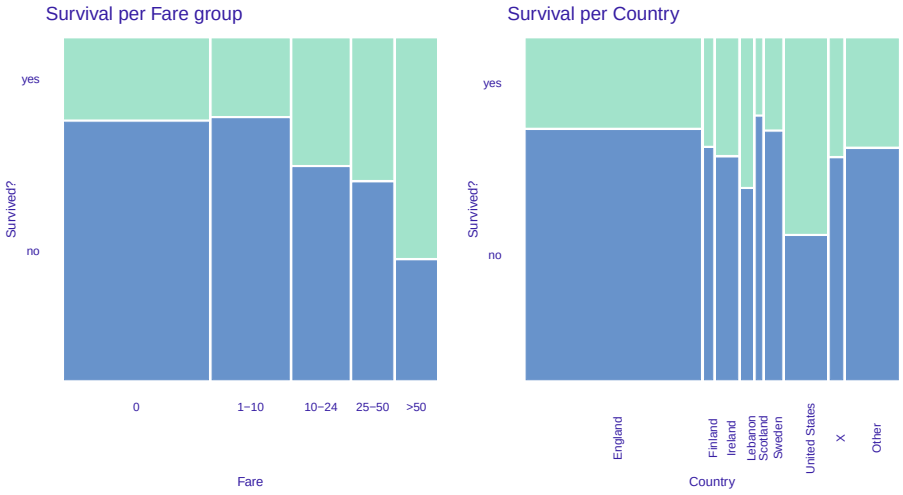


FIGURE 12 Survival according to fare and country in the Titanic data.

splines, available in the `rcs()` function of the `rms` package (Harrell Jr, 2018), to model the effect of age. We also do not expect linear relation for the `fare` variable, but because of it's skewness, we do not use splines for this variable. The results of the model are stored in model-object `titanic_lmr_v6`, which will be used in subsequent chapters.

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

```
## Logistic Regression Model
##
##  lrm(formula = survived == "yes" ~ gender + rcs(age) + class +
##      sibsp + parch + fare + embarked, data = titanic)
##
##                        Model Likelihood     Discrimination    Rank Discrim.
##                           Ratio Test           Indexes           Indexes
##  Obs           2207    LR chi2     752.06    R2       0.404    C       0.817
```

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

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

0.5.1.3 Random forest model

As a challenger to the logistic regression model, we consider a random forest model. Random forest is known for good predictive performance, is able to grasp low-order variable interactions, and is quite stable (Breiman, 2001). To fit the model, we apply the

`randomForest()` function, with default settings, from the package with the same name (Liaw and Wiener, 2002).

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

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

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

0.5.1.4 Gradient boosting model

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

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

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

0.5.1.5 Support Vector Machine model

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

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

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

0.5.1.6 Model predictions

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

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

```
johny_d <- data.frame(
            class = factor("1st", levels = c("1st", "2nd", "3rd", "deck crew", "engineering cr
            gender = factor("male", levels = c("female", "male")),
            age = 8,
            sibsp = 0,
            parch = 0,
```

```
            fare = 72,
            embarked = factor("Southampton", levels = c("Belfast","Cherbourg","Queenstown","So
)
```

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", "engineering cr
            gender = factor("male", levels = c("female", "male")),
            age = 47,
```

```
            sibsp = 0,
            parch = 0,
            fare = 25,
            embarked = factor("Cherbourg", levels = c("Belfast","Cherbourg","Queenstown","Sout
)
predict(titanic_lmr_v6, henry, type = "fitted")
```

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

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

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

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

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

0.5.1.7 Model adapters

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

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

Each adapter contains all elements needed to create a model explanation, i.e., a suitable `predict()` function, validation data set,

and the model object. Thus, in subsequent chapters we will use the explainers instead of the model objects to keep code snippets more concise.

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

0.5.1.8 List of objects for the `titanic` example

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

For the ease of reference, we summarize the models in Table 0.1. The binary model-objects can be downloaded by using the indicated `archivist` hooks (Biecek and Kosinski, 2017). By calling a function specified in the last column of the table, one can restore a selected model in its local R environment.

TABLE 0.1: Predictive models created for the `titanic` dataset.

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

Model name	Model generator	Variables	Archivist hooks
`titanic_svm_v6`	`e1071:: svm` 1.7-2	gender, age, class, sibsp, parch, fare, embarked	Get the model: `archivist:: aread("pbiecek/models/be26e")`. Get the explainer: `archivist:: aread("pbiecek/models/21966")`

Table 0.2 summarizes the data frames that will be used in examples in the subsequent chapters.

TABLE 0.2: 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")`

FIGURE 13 Warsaw skyscrapers by Artur Malinowski Flicker

Description	No. rows	Variables	Link to this object
`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")`

0.5.2 Apartment prices

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

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

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

- *m2.price*, apartments price per meter-squared (in EUR), a numerical variable range 1607 – 6595;
- *construction.year*, the year of construction of the block of flats in which the apartment is located, a numerical variable range 1920 – 2010;
- *surface*, apartment's total surface in square meters, a numerical variable range 20 – 150;
- *floor*, the floor at which the apartment is located (ground floor taken to be the first floor), a numerical integer variable with values from 1 to 10;
- *no.rooms*, the total number of rooms, a numerical variable with values from 1 to 6;
- *district*, a factor with 10 levels indicating the district of Warsaw where the apartment is located.

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

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

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

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

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

```
head(apartments_test)
```

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

0.5.2.1 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 **??** indicates that

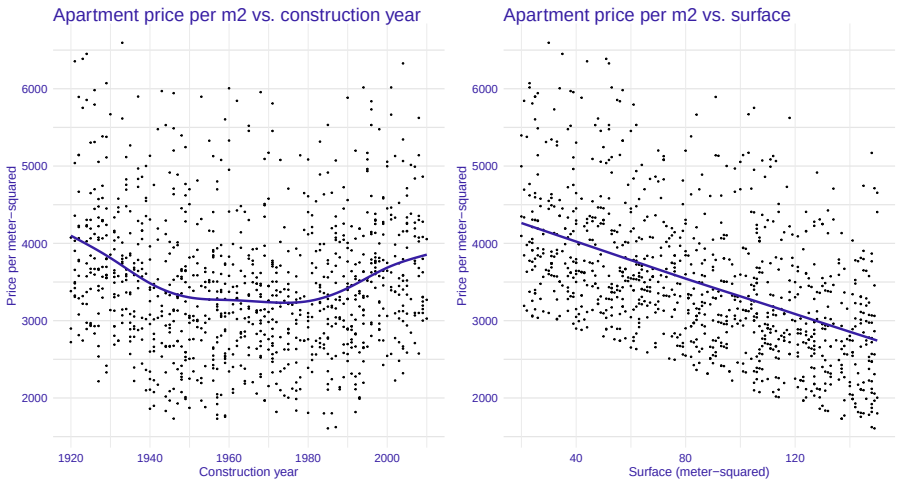


FIGURE 14 Left panel shows apartment price per m2 vs. year of construction, right panel shows price vs. square footage

the distribution of the variable is slightly skewed to the right.

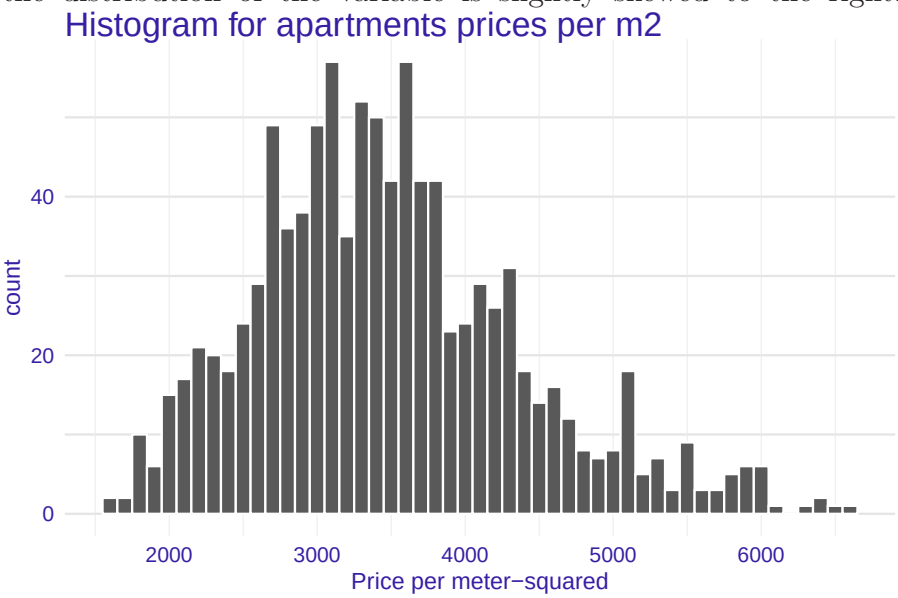


Figure 14 suggests (possibly) a nonlinear relation between *construction.year* and *m2.price* and a linear relation between *surface* and *m2.price*.

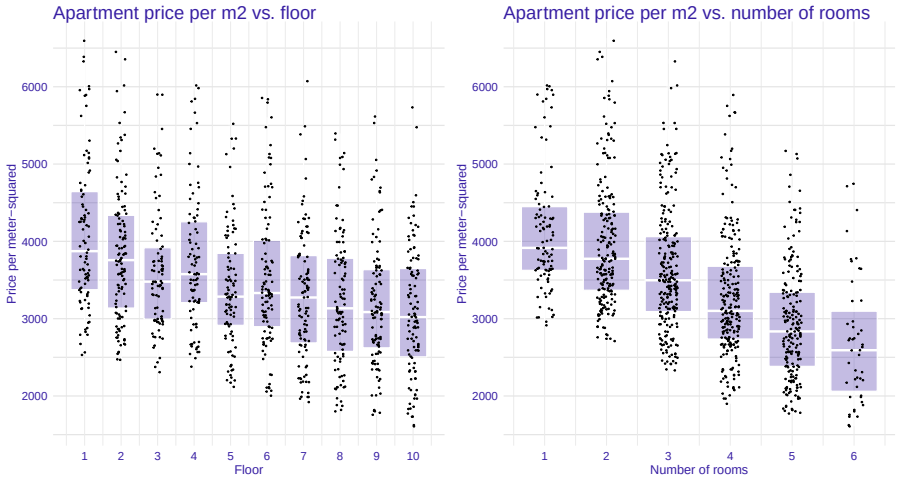


FIGURE 15 Price per meter-squared vs. floor and vs. number of rooms.

Relation between *floor* and *m2.price* is also close to linear, as well as relation between *no.rooms* and *m2.price* as seen in Figure 15.

Surface and number of rooms are correlated and prices depend on district. Boxplots plots in Figure 16 indicate that the highest prices per meter-squared are observed in Srodmiescie (Downtown).

0.5.2.2 Linear regression model

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

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

```
## Analysis of Variance Table
##
## Response: m2.price
##                    Df    Sum Sq   Mean Sq  F value    Pr(>F)
```

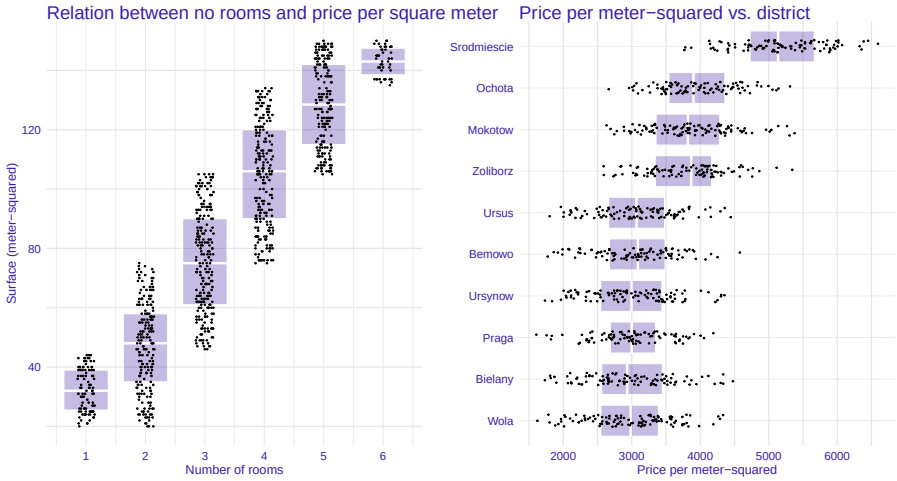


FIGURE 16 Left panel: surface vs. number of rooms. Right panel: price per meter-squared for different districts

```
## construction.year   1   2629802   2629802   33.233 1.093e-08 ***
## surface             1 207840733 207840733 2626.541 < 2.2e-16 ***
## floor               1  79823027  79823027 1008.746 < 2.2e-16 ***
## no.rooms            1    956996    956996   12.094  0.000528 ***
## district            9 451993980  50221553  634.664 < 2.2e-16 ***
## Residuals         986  78023123     79131
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

0.5.2.3 Random forest model

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

```
library("randomForest")
set.seed(72)
```

```
apartments_rf_v5 <- randomForest(m2.price ~ ., data = apartments)
apartments_rf_v5
```

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

0.5.2.4 Support vector model

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

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

0.5.2.5 Model predictions

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

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

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

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

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

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

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

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

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

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

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

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

0.5.2.6 Model adapters

In similar spirit to the Section 0.5.1.7 we will use explainers also for predictive models created for the `apartments` dataset.

```
explain_apartments_lm_v5 <- explain(model = apartments_lm_v5,
                                    data = apartments_test,
                                    y = apartments_test$m2.price,
                                    label = "Linear Regression")
```

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

0.5.2.7 List of objects for the `apartments` example

In Sections 0.5.2.2 and 0.5.2.3 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.

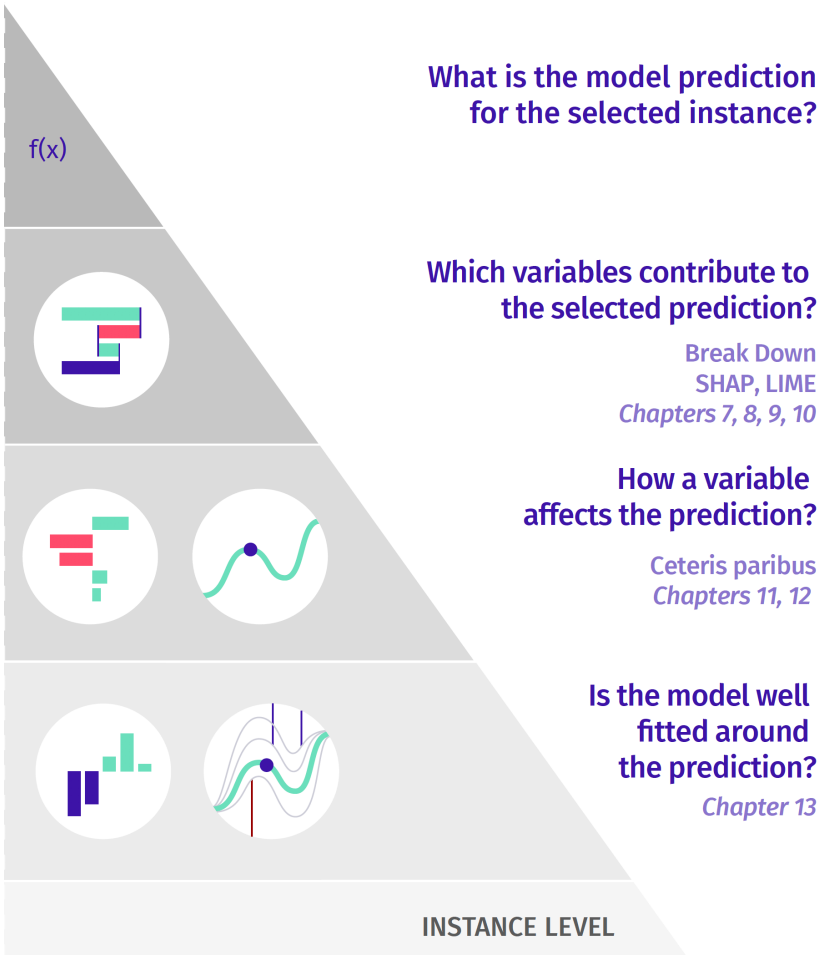
For the ease of reference, we summarize the models in Table 0.3. The binary model-objects can be downloaded by using the indicated `archivist` hooks (Biecek and Kosinski, 2017). By calling a function specified in the last column of the table, one can restore a selected model in a local R environment.

TABLE 0.3: Predictive models created for the `apartments` dataset.

Model name	Model generator	Variables	Archivist hooks
`apartments_lm_v5`	`stats:: lm` v.3.5.3	construction .year, surface, floor, no.rooms, district	Get the model: `archivist:: aread("pbiecek/models/55f19")`. Get the explainer: `archivist:: aread("pbiecek/models/78d4e")`

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

Instance Level



0.6 Introduction to Instance Level Exploration

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

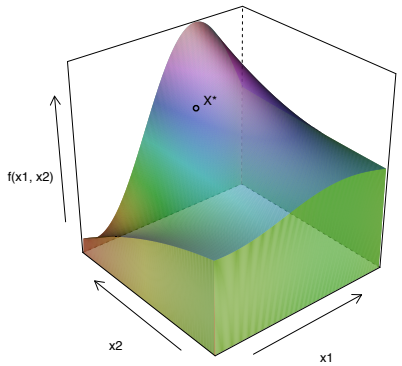
- We may want to evaluate the effects of explanatory variables on model predictions. For instance, we may be interested in predicting the risk of heart attack based on person's age, sex, and smoking habits. A model may be used to construct a score (for instance, a linear combination of the explanatory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

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

- One approach is to analyze how the model prediction for point x^* is different from the average model prediction and how the difference can be distributed among explanatory variables. It is often called the „variable attributions'' approach. An example is provided in panel A of Figure 17. Chapters **??-??** present various methods implementing this approach.

- Another approach is to analyze the curvature of the response surface around the point of interest x^*. Treating the model as a function, we are interested in the local behavior of this function around x^*. In case of a black-box model, we may approximate it with a simpler glass-box model around x^*. An example is provided in panel B of Figure 17. Chapter **??** presents the Local Interpretable Model-agnostic Explanations (LIME) method that exploits the concept of a „local model' '.
- Yet another approach is to investigate how the model prediction changes if the value of a single explanatory variable changes. The approach is useful in the so-called „What-If' ' analyses. In particular, we can construct plots presenting the change in model-based predictions induced by a change of a single variable. Such plots are usually called Ceteris-paribus (CP) profiles. An example is provided in panel C in Figure 17. Chapters **??**-**??** introduce the CP profiles and methods based on them.

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



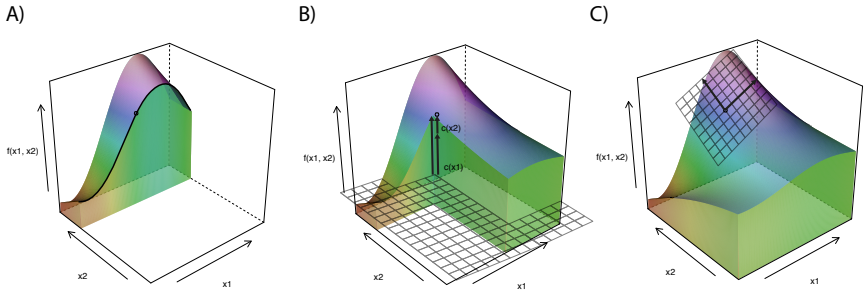


FIGURE 17 Response surface for a model that is a function of two variables. We are interested in understanding the response of a model in a single point x*. Illustration of different approaches to instance-level explanation. Panel A illustrates the concept of variable attributions like Break Down or SHAP. Additive effects of each variable show how the model response differs from the average. Panel B illustrates the concept of explanations through local models e.g. LIME. A simpler glass-box model is fitted around the point of interest. It describes the local behaviour of the black-box model. Panel C presents a What-If analysis with Ceteris-paribus profiles. The profiles show the model response as a function of a value of a single variable, while keeping the values of all other explanatory variables fixed.

Bibliography

Alber, M., Lapuschkin, S., Seegerer, P., Hägele, M., Schütt, K. T., Montavon, G., Samek, W., Müller, K.-R., Dähne, S., and Kindermans, P.-J. (2018). innvestigate neural networks!

Allaire, J. and Chollet, F. (2019). *keras: R Interface to 'Keras'*. R package version 2.2.4.1.

Azure (2019). Microsoft Cognitive Services.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. (2015). On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *Plos One*, 10(7):e0130140.

Biecek, P. (2018). *DALEX: Explainers for Complex Predictive Models in R*.

Biecek, P. (2019). Model Development Process. *CoRR*, abs/1907.04461.

Biecek, P. and Kosinski, M. (2017). archivist: An R package for managing, recording and restoring data analysis results. *Journal of Statistical Software*, 82(11):1–28.

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

Boehm, B. (1988). *A Spiral Model of Software Development and Enhancement*.

Breiman, L. (2001). Random forests. In *Machine Learning*, volume 45, pages 5–32.

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

Casey, B., Farhangi, A., and Vogl, R. (2018). Rethinking explainable machines: The gdpr's 'right to explanation' debate and the rise of algorithmic audits in enterprise. *Berkeley Technology Law Journal*.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., and Wirth, R. (1999). *The CRISP-DM 1.0 Step-by-step data mining guide*.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. In *Machine Learning*, pages 273–297.

Dastin, J. (2018). Amazon scraps secret ai recruiting tool that showed bias against women.

Diaz, M., Johnson, I., Lazar, A., Piper, A. M., and Gergle, D. (2018). Addressing age-related bias in sentiment analysis. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, Chi '18, pages 412:1–412:14, New York, NY, USA. Acm.

Duffy, C. (2019). Apple co-founder steve wozniak says apple card discriminated against his wife.

Efron, B. and Hastie, T. (2016). *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science*. Cambridge University Press, New York, NY, USA, 1st edition.

Faraway, J. (2002). *Practical Regression and ANOVA using R*.

Foster, D. (2017). *xgboostExplainer: An R package that makes xgboost models fully interpretable*. R package version 0.1.

Friedman, J. H. (2000). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232.

from Jed Wing, M. K. C., Weston, S., Williams, A., Keefer, C., Engelhardt, A., Cooper, T., Mayer, Z., Kenkel, B., the R Core Team, Benesty, M., Lescarbeau, R., Ziem, A., Scrucca, L., Tang, Y., and Candan., C. (2016). *caret: Classification and Regression Training*. R package version 6.0-64.

Galecki, A. and Burzykowski, T. (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.

Goodman, B. and Flaxman, S. (2016). European union regulations on algorithmic decision-making and a "right to explanation". *Arxiv*.

Grolemund, G. and Wickham, H. (2019). *R for Data Science*.

Hall, P. (2019). *On Explainable Machine Learning Misconceptions and A More Human-Centered Machine Learning*.

Harrell Jr, F. E. (2018). *rms: Regression Modeling Strategies*. R package version 5.1-2.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Hoover, B., Strobelt, H., and Gehrmann, S. (2019). exbert: A visual analysis tool to explore learned representations in transformers models.

Jacobson, I., Booch, G., and Rumbaugh, J. (1999). *The Unified Software Development Process*.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2014). *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.

Karbowiak, E. and Biecek, P. (2019). *EIX: Explain Interactions in Gradient Boosting Models*. R package version 1.0.

Kruchten, P. (1998). *The Rational Unified Process*.

Kuhn, M. and Johnson, K. (2013). *Applied predictive modeling*. Springer, New York, NY.

Kuhn, M. and Vaughan, D. (2019). *parsnip: A Common API to Modeling and Analysis Functions*. R package version 0.0.2.

Larson, J., Mattu, S., Kirchner, L., and Angwin, J. (2016). How we analyzed the compas recidivism algorithm.

Lazer, D., Kennedy, R., King, G., and Vespignani, A. (2014). The parable of google flu: Traps in big data analysis. *Science*, 343(6176):1203–1205.

LeDell, E., Gill, N., Aiello, S., Fu, A., Candel, A., Click, C., Kraljevic, T., Nykodym, T., Aboyoun, P., Kurka, M., and Malohlava, M. (2019). *h2o: R Interface for 'H2O'*. R package version 3.22.1.1.

Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R News*, 2(3):18–22.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., and Leisch, F. (2017). *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. R package version 1.6-8.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., and Leisch, F. (2019). *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. R package version 1.7-2.

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

Nolan, D. and Lang, D. T. (2015). *Data Science in R: A Case Studies Approach to Computational Reasoning and Problem Solving*. Chapman & Hall/CRC.

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

Paluszynska, A. and Biecek, P. (2017). *randomForestExplainer: A set of tools to understand what is happening inside a Random Forest*. R package version 0.9.

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

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). *Why Should I Trust You?: Explaining the Predictions of Any Classifier*, page 1135–1144. ACM Press.

Ridgeway, G. (2017). *gbm: Generalized Boosted Regression Models*. R package version 2.1.3.

Ross, C. and Swetliz, I. (2018). Ibm's watson supercomputer recommended 'unsafe and incorrect' cancer treatments, internal documents show.

Ruiz, J. (2018). Machine learning and the right to explanation in gdpr.

Salzberg, S. (2014). Why Google Flu Is A Failure.

Samek, W., Wiegand, T., and Müller, K.-R. (2017). Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models.

Shapley, L. S. (1953). A Value for n-Person Games. In Kuhn, H. W. and Tucker, A. W., editors, *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, Princeton.

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

Simonyan, K., Vedaldi, A., and Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

Tukey, J. W. (1977). *Exploratory Data Analysis*. Addison-Wesley.

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer, New York, fourth edition. Isbn 0-387-95457-0.

Wes, M. (2012). *Python for Data Analysis*. O'Reilly Media, Inc., 1 edition.

Wickham, H. (2009). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

Wickham, H. and Grolemund, G. (2017). *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. O'Reilly Media, Inc., 1st edition.

Wikipedia (2019). *CRISP DM: Cross-industry standard process for data mining*.

Wright, M. N. and Ziegler, A. (2017). *ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R*.

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