Landscape of R packages for eXplainable Artificial Intelligence

by Szymon Maksymiuk, Alicja Gosiewska, Przemysław Biecek

Abstract The growing availability of data and computing power is fueling the development of predictive models. In order to ensure the safe and effective functioning of such models, we need methods for exploration, debugging, and validation. New methods and tools for this purpose are being developed within the eXplainable Artificial Intelligence (XAI) subdomain of machine learning. In this work, (1) we present the taxonomy of methods for model explanations, (2) we identify and compare 27 packages available in R to perform XAI analysis, (3) we present an example of an application of particular packages, (4) we acknowledge trends in recent developments. The article is primarily devoted to the tools available in R, but since it is easy to integrate the Python code, we will also show examples for the most popular libraries from Python.

Importance of eXplainable Artificial Intelligence

The growing demand for fast and automated development of predictive models has contributed to the popularity of machine learning frameworks such as **caret** (from Jed Wing et al., 2019), **mlr** (Bischl et al., 2016), **mlr3** (Lang et al., 2019), **tidymodels** (Kuhn and Wickham, 2020), **h2o** (H2O.ai, 2015), **scikit-learn** (Pedregosa et al., 2011), **keras** (Chollet et al., 2015), **pytorch** (Paszke et al., 2019), and many others. These tools allow us to quickly test models of very different structures and choose the best one based on a selected performance measure. However, it soon became apparent that such a process leads to treating models like black-boxes. This, in turn, makes it difficult to detect certain problems early enough. Insufficiently tested models quickly lose their effectiveness, lead to unfair decisions, discriminate, are deferred by users, and do not provide the option to appeal (Gill et al., 2020). To overcome these problems methods and tools for analysis of black-box predictive models are essential. In this work, we will present tools that can be used for this purpose.

There are various situations where we need tools for in-depth model analysis. For example:

- The model makes incorrect decisions on certain observations. We want to understand what the cause is of these incorrect decisions, in hopes to improve the model. In some sense, we want to debug the model by looking for the source of its ineffectiveness.
- Model predictions are used by inquisitive parties. In order to build their trust and confidence, we need to present additional arguments or reasoning behind particular predictions.
- Sometimes we expect that the model will automatically discover some relationships hidden in the data. By analyzing the model, we want to extract and understand these relationships in order to increase our domain knowledge.
- It is becoming increasingly common to expect not only decisions but also reasons, arguments, and explanations for a decision to be made in an automated way. Sometimes such explanations are required by law, see, for example, GDPR regulations (Goodman and Flaxman, 2017).
- Often the key factor is the question of responsibility. If model developers responsibly recommend the use of a model, they often need to understand the way the model works. Thus, we cannot rely on the black-box. We need a deeper understanding of the model.

In this work, we present tools that can be used in eXplainable Artificial Intelligence (XAI) modeling, which can help to explore predictive models. In recent years, plenty of interesting software has been developed. We hope that presenting XAI tools in this paper will make them easier to apply and will, therefore, lead to building better and safer predictive models. The main contributions of this paper are as follows.

1. We introduced two taxonomies of methods for model comparisons. The first one concerns a model as a subject of analysis, and the second one concerns the domain of the explanation.
2. We conducted a comprehensive review and identified 27 popular R packages that implement XAI methods.
3. We compared the capabilities of recognized R packages, taking into account the variety of implemented methods, interoperability, and time of operation.
4. We prepared knitr reports with illustrative examples for each of the considered packages.

This paper is focused on the tools available to R users rather than on an overview of mathematical details for particular methods. Those interested in a more detailed analysis of the methods may want to familiarize themselves with a work of Chatzimparmpas et al. (2020) with a meta-analysis of 18 survey papers that refer to the explainability of machine learning models. Some of them are related to model visualization, such as Liu et al. (2017), predictive visual analytics (Lu et al., 2017a,b), interaction with models (Amershi et al., 2014; Dudley and Kristensson, 2018), deep learning (Choo and Liu, 2018; Garcia et al., 2018; Hohman et al., 2019), or dimensionality reduction (Sacha et al., 2016). Algorithmics details for XAI methods can also be found in books *Interpretable Machine Learning* (Molnar, 2019) or *Explanatory Model Analysis* (Biecek and Burzykowski, 2021). Chatzimparmpas et al. (2020) emphasized the importance of regularly keeping up with new surveys, both due to the covering of more articles as well as the different perspectives. To the best of our knowledge, none of these surveys aims at a comprehensive analysis of software tools for eXplainable Artificial Intelligence. In this paper, we conduct an extensive review of the R packages.

It should be noted that work in this area is being carried out in various camps of modelers. The contributions of the statisticians are intertwined with those made by practitioners in machine learning or deep learning. This sometimes leads to redundancy in the nomenclature. The name *eXplainable Artificial Intelligence* was popularized by the DARPA program (Gunning, 2017), emphasizing the question of how much the algorithm can explain the reasons for a recommended decision. The name *Interpretable Machine Learning* was popularized by a book with the same title (Molnar, 2019), which emphasizes a model's interpretability. The name *Explanatory Model Analysis* (Biecek and Burzykowski, 2021) refers to the main objective, which is to explore model behavior. These threads are also known as *Veridical Data Science* (Yu and Kumbier, 2020) or *Responsible Machine Learning* (Gill et al., 2020). In this paper, we will use the term XAI, but note that in most cases these names can be used interchangeably.

Taxonomy of XAI methods

The literature presents several taxonomies of XAI methods categorization (Gilpin et al., 2018; Biran and Cotton, 2017; Molnar, 2019; Biecek and Burzykowski, 2021). Figure 1 summarizes the most frequent grouping of these methods.

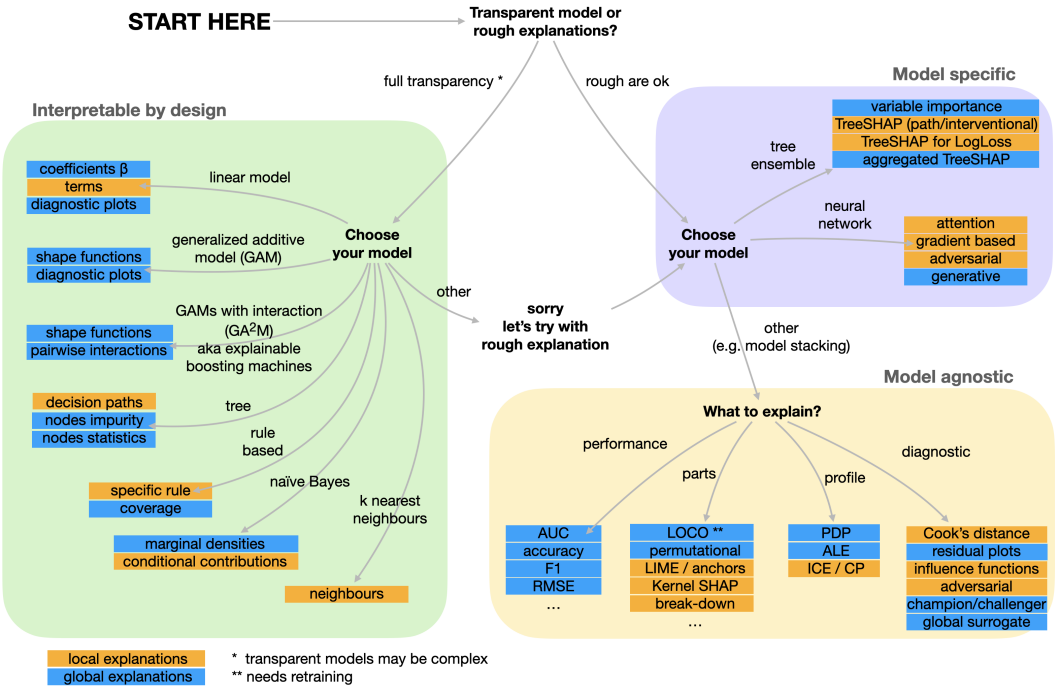


Figure 1: Model oriented taxonomy for XAI methods.

The introduced taxonomy puts the model structure in the center. From this point of view, we are dealing with three groups of methods.

- Methods for models with interpretable structure (**interpretable by design**), such as linear models, decision trees (Hothorn et al., 2006), decision rules (Hahsler et al., 2011), k-nearest neighbors (Venables and Ripley, 2002; Schliep and Hechenbichler, 2016), and others. The

architecture of such predictive models allows us to directly explain a certain aspect of the model. Such explanations are accurate, i.e. they are based on model coefficients and describe the model in a complete way. Of course, such models can be difficult to understand. A tree can contain hundreds of decision leaves, making it challenging to grasp the whole. But, the model structure is, in general, directly interpretable.

- **Model-specific methods**. There are situations in which the model structure is so complex that it cannot be interpreted by looking at individual parameters. However, there are methods for knowledge extraction designed for specific classes of models. For tree ensembles, it is possible to analyze the tree structures in order to extract from them some relationships between variables. For neural networks, it is possible to trace gradients. This group of methods assumes full access to the model structure. We expect that the explanation of the operation is a rough approximation of a more complex process.
- **Model-agnostic methods**. The most general class of methods are those that facilitate analysis of the model without any knowledge of the internal structure. The analysis is usually carried out on the basis of a number of model evaluations on properly prepared perturbed input data. Such an analysis can be carried out on any predictive model.

In the last chapter of this paper, we will present examples of methods for each of these groups, but the main emphasis is on model-agnostic methods. The reason for this is that we often compare models with different structures when looking for a final model, and, sometimes, we even have to compare models created in different frameworks. The main advantage of that approach is flexibility. It allows users of such an algorithm to compare explanations of different machine learning models, for instance, the forest type model with a boosting tree or even neural network, with common metrics between them. The disadvantage is that explanations acquired with that approach may be less accurate. They only approximate the behavior of the predictive model ignoring information that comes from the structure.

Another possible taxonomy refers to the task to which the explanation is used. The most common are two situations when the explanation concerns one observation or the whole dataset. If the explanation concerns a single observation, it is called **local** or individual. If an explanation refers to the whole dataset, it is called **global** or dataset-level. In applications, we sometimes fall into a gray zone where we are interested in the behavior of the model for a group of observations. Usually, both local and global methods can be used in such cases.

Table 1 introduces the second taxonomy of methods according to the purpose of model explanation, whether the goal is to understand the importance of the variables, profile the variables, or to analyze the performance of a model. Later on in the article, we will present examples of comparisons of models based on this taxonomy.

Global	*Local*
Model Parts • Permutational variable importance • Leave-One-Covariate-Out (LOCO) • Surrogate models • Aggreagated SHapley Additive exPlanations	**Predict Parts** • BreakDown (BD) • SHapley Additive exPlanations (SHAP) • Local Interpretable Model-agnostic Explanations (LIME)
Model Profile • Partial Dependence Profiles (PDP) • Accumulated Local Effects (ALE)	**Predict Profile** • Ceteris Paribus (CP) / Individal Conditional Expectations (ICE)
Model Diagnostics • Residual plots • Variable vs. prediction plots • Demographic parity	**Predict Diagnostics** • Local residual density plot

Table 1: XAI taxonomy for model-agnostic and model-specific methods along with examples. Names in *italics* at the top denote the explanation domain. **Bolded** names refer to the explanation task, while itemized entries are examples of explanation methods that comply with the given task.

Below we discuss the objectives for each task and give examples of methods.

- **Model Parts** focuses on the importance of variables or groups of variables. It considers a global level approach. The Model Parts task can be realized by computing the impact of a single variable or group of variables have on the model performance using various methods like permutation variable importance, Leave One Covariate Out (LOCO) (Lei et al., 2018), or surrogate tree models. It seems the most popular is the permutation variable importance (Fisher et al., 2018) which assesses the importance of a variable by the change in the model performance after masking the effect of one or a group of variables. The effect of masking is achieved by permutations or resampling values for selected variables.
- **Model Profile** aims at presenting how a single variable or a group of variables affects model response. It shows the profile of model prediction as a function of dependent variables. Model Profile represents the global domain of explanations, where the whole dataset is taken into consideration. The most popular examples of such methods are Partial Dependence Profiles (PDP) and Accumulated Local Effects (ALE). PDP (Friedman, 2000) is a method of profiling the global behavior of models in the context of one or a pair of variables. Its purpose is to show how the expected prediction of the model change based on changes to dependent variables. In other words, it estimates model response as a function of a specific variable. ALE (Apley and Zhu, 2016) is to some extent, similar to PDP, but it takes into account the conditional distribution of a variable, instead of a marginal one. Thus, it is more suited in cases where variables are correlated.
- **Model Diagnostics** focuses on methods that can be utilized to evaluate the performance of the model, improve the quality of predictions or present the structure of the model. It is a very wide category without distinctive representatives. Examples of explanation methods that comply with a definition of Model Diagnostics are residual density plots, variable vs prediction plots, and many other figures that present responses or residuals versus different variables.
- The **Predict Parts** explanation task analyses and presents the impact of components and from the perspective of a single observation. The goal of that task is to measure the contributions each observation has to the final prediction. It belongs to the local explanation domain. Examples of the Predict Parts Explanation Method are LIME (Ribeiro et al., 2016), SHAP (Lundberg and Lee, 2017) or BreakDown (Gosiewska and Biecek, 2019b). Probably the most popular method in this group is the SHapley Additive exPlanations (SHAP) (Lundberg and Lee, 2017) method. It is based on the Shapley values concept, which derives from game theory. They are the solution for the problem in cooperative games where non-identical players contribute, to different extents, to the outcome. Shapley values allow us to calculate how the surplus should be distributed. Translating it into the language of models, players now are dependent variables while the model prediction is a surplus to divide. Current implementations usually approximate Shapley values in general case or calculate exact values for tree ensembles.
- **Predict Profile** sometimes called sensitivity analysis, is a task similar to Model Profile. The only difference lies in the explanation subject. For this task, there is no aggregation over the whole dataset. Instead, the effects of variables are explained from the perspective of a single observation. An instance of an explanation method that complies with the Predict Profile task is Individual Conditional Expectations (Goldstein et al., 2015) also known as the Ceteris Paribus Profile (Biecek and Burzykowski, 2021).
- **Predict Diagnostics** follows the same principia as the Model Diagnostics task. What differs between the two is the way they look at the model. For Predict Diagnostics, a single observation or its neighbors are taken as perspective instead of the whole dataset. One of the examples of the Predict Diagnostics methods is a comparison of residual distribution for k neighbors of a single observation versus distribution over the whole dataset.

Comparison of packages for XAI analysis

Selection criteria for XAI packages

The number of tools that can be used to analyze predictive models is growing rapidly, and, at the same time, there is no established definition of an XAI tool. Therefore, it would not be possible to reliably identify and present all existing XAI packages.

For a reliable analysis[1], we have adopted the following criteria. For the initial set of packages, we have used the list *Awesome Machine Learning Interpretability*, maintained by Patrick Hall (Hall, 2020).

[1]The authors have made every effort to maintain an objective view when selecting and analyzing packages. However, a potential conflict of interest should be noted, we are authors or co-authors of some of the described packages, which belong to the `DrWhy.ai` family.

	Package	License	Date of last update	Last version	GitHub stars	CRAN downloads	Date of first release
R	ALEPlot	GPL-2	2018-05-24	1.1	-	36,831	2017-11-13
	auditor	GPL-2/GPL-3	2020-05-28	1.3.0	54	26,757	2018-05-11
	DALEX	GPL-2/GPL-3	2021-03-20	2.2.0	796	99,688	2018-02-28
	DALEXtra	GPL-2/GPL-3	2020-09-07	2.0	40	13,554	2019-09-19
	EIX	GPL-2	2021-03-23	1.2.0	12	10,491	2019-05-31
	ExplainPrediction	GPL-3	2018-01-07	1.3.0	-	27,589	2015-09-07
	fairness	MIT + addons	2020-11-19	1.2.0	21	10,369	2019-09-27
	fastshap	GPL-2/GPL-3	2020-02-02	0.0.5	46	26,820	2019-11-22
	flashlight	GPL-2/GPL-3	2021-02-13	0.7.5	9	14,312	2019-08-25
	forestmodel	GPL-2	2020-07-19	0.6.2	22	36,568	2015-11-26
	fscaret	GPL-2/GPL-3	2018-05-08	0.9.4.4	-	44,857	2013-06-13
	iBreakDown	GPL-3	2020-07-29	1.3.1	57	55,192	2019-04-04
	ICEbox	GPL-2/GPL-3	2017-07-13	1.1.2	32	38,117	2013-10-18
	iml	MIT + addons	2020-09-24	0.10.1	389	126,471	2018-03-13
	ingredients	GPL-3	2021-02-05	2.0.1	31	58,454	2019-04-09
	lime	MIT + addons	2021-02-24	0.5.2	438	110,090	2017-09-15
	live	MIT + addons	2020-01-15	1.5.13	34	18,009	2018-04-03
	mcr	GPL-3	2014-02-12	1.2.1	-	39,120	2012-07-24
	modelDown	Apache License 2.0	2020-04-15	1.1	102	9,624	2019-06-15
	modelStudio	GPL-3	2021-01-07	2.1.1	163	15,943	2019-09-03
	pdp	GPL-2/GPL-3	2018-08-27	0.7.0	71	188,853	2016-09-02
	randomForestExplainer	GPL-2/GPL-3	2020-07-11	0.10.1	175	39,944	2017-07-15
	shapper	GPL-2/GPL-3	2020-08-28	0.1.3	47	15,615	2019-03-02
	smbinning	GPL-2/GPL-3	2019-04-01	0.9	-	275,329	2015-02-15
	survxai	GPL-2/GPL-3	2020-08-28	0.2.2	9	10,694	2018-08-24
	vip	GPL-2/GPL-3	2020-12-17	0.3.2	129	118,346	2018-06-15
	vivo	GPL-2	2020-09-07	0.2.1	14	9,292	2019-06-17
Python	aix360	Apache 2.0	2020-10-28	0.2.1	787	-	2019-08-08
	eli5	MIT	2021-01-23	0.11.0	2323	-	2016-09-15
	interpret	MIT	2021-01-20	0.2.4	3572	-	2019-05-04
	lime	BSD	2020-06-26	0.2.0.1	8552	-	2016-03-24
	shap	MIT	2021-03-03	0.39.0	12083	-	2016-12-01
	skater	MIT	2018-09-21	1.1.2	973	-	2017-05-23

Table 2: List of packages that will be compared. The type of license date of the last CRAN release, number of GitHub stars, CRAN downloads, and date of the first release are also presented. A dash means that information was not accessible, meaning either the package does not have a GitHub repository or is not available on CRAN (it is a Python package). Downloads were acquired using **deepdep** (Rafacz et al., 2020) package. Access 2021-03-23.

We chose this list because of the pioneer works related to the synthesis of approaches to XAI described in *An Introduction to Machine Learning Interpretability* (Hall and Gill, 2018; Hall, 2018). We restricted this list to packages published on CRAN with at least 5000 downloads. The list is in Table 2.

In addition, this list was extended by packages available on CRAN, which have in the DESCRIPTION file some keywords used in XAI analysis. We checked 15,993 packages (access 08.07.2020) looking for keywords like *xai*, *iml*, *explain* and *interpretability*. This way, we encountered a great deal of false-positive entries (for instance, packages that explain the meaning of HTTP codes or indeed use XAI methods but to solve other, well-defined problems, not to actually explain the predictive models). This list was then manually cleaned and we ended up with 27 packages.

We would like to point out that **mcr** (Manuilova et al., 2014) and **smbinning** (Jopia, 2019) may look like they do not actually belong to the XAI area. However, they were included in Patrick Hall's original list, and we would consider it biased to arbitrarily remove packages despite meeting all the requirements of being included. Therefore, we did our best to extract the XAI related functionalities of those packages.

Note, the purpose of this paper is to compare packages that can be used for models created in R. However, we have decided to add also some Python libraries for context.

Available methods in XAI packages

Table 3 summarizes methods available in the XAI packages. Some packages are focused only on a single aspect of a model explanation, while others implement a larger number of methods. In Section .1 *Example gallery for XAI packages*, we present an explanation generated by each package.

Most of the compared packages implement Model Parts or Predict Parts explanations. This means that the primary focus is on inspection of how consecutive variables contribute to model prediction either locally or globally. Nine out of all compared R packages implement both Model Parts and Predict Parts methods. Two packages implement exactly those two types of methods, while seven provide other types of methods as well. On the other hand, Predict Diagnostics is the least represented type of explanation as only one package support it. However, Model Diagnostics methods are available in nine out of checked R packages, and, for three of them, this is the only supported type of explanation. Only five packages allow the variables to be locally profiled, while nine provide an opportunity to do it globally. Eight of the compared packages provide access to three or more different types of explanations, but only five to at least four types. Taking into consideration the compared Python packages, they give access to similar features as the R packages do.

		Global Explanations			Local Explanations		
	Package	Model parts	Model profile	Model diagnostics	Predict parts	Predict profile	Predict diagnostics
R	ALEPlot	-	✓	-	-	-	-
	auditor	-	-	✓	-	-	-
	DALEX/DALEXtra	✓	✓	✓	✓	✓	✓
	EIX	✓	-	✓	✓	-	-
	ExplainPrediction	✓	-	-	✓	-	-
	fairness	-	-	✓	-	-	-
	fastshap	✓	✓	-	✓	-	-
	flashlight	✓	✓	✓	✓	✓	-
	forestmodel	✓	-	-	-	-	-
	fscaret	✓	-	-	-	-	-
	ICEbox	-	✓	-	-	-	-
	iml	✓	✓	-	✓	✓	-
	lime	-	-	-	✓	-	-
	live	-	-	-	✓	-	-
	mcr	-	-	✓	-	-	-
	modelDown	✓	✓	✓	-	-	-
	modelStudio	✓	✓	-	✓	✓	-
	pdp	✓	✓	-	-	-	-
	randomForestExplainer	✓	-	-	-	-	-
	shapper	-	-	-	✓	-	-
	smbinning	✓	-	✓	-	-	-
	survxai	✓	-	✓	✓	✓	-
	vip	✓	-	-	-	-	-
	vivo	✓	-	-	✓	-	-
Python	aix360	✓	✓	✓	✓	-	-
	eli5	✓	-	-	✓	-	-
	interpret	✓	✓	-	✓	-	-
	lime	-	-	-	✓	-	-
	shap	✓	✓	-	✓	-	-
	skater	✓	✓	-	✓	-	-

Table 3: Groups of methods (see Table 1) implemented in considered XAI packages. ✓means that selected package implements at least one method that belongs to a given explanation task. As **DALEXtra** depends on **DALEX**, it inherits every functionality **DALEX** has. Package **ingredients** and **iBreakDown** were not included in this table since they are imported by **DALEX**, **modelStudio** and **modelDown**. Access 2020-08-15.

Models comparisons and the Rashomon effect

The Rashomon effect (Wikipedia, 2020) refers to a situation in which an event has contradictory interpretations by different spectators. Breiman (2001) has introduced this concept to machine learning modeling. Two or more models of similar performance may correspond to a different relation between variables.

The juxtaposition of explanations for different models allows for a more complete analysis. However, not every package facilitates easy model comparison, and not every package was designed with model comparison in mind. Packages **auditor** (Gosiewska and Biecek, 2019a), **DALEX**, **DALEXtra** (Maksymiuk and Biecek, 2020), **flashlight** (Mayer, 2020b), **ingredients** (Biecek et al., 2019), **modelDown** (Romaszko et al., 2019), **modelStudio** (Baniecki and Biecek, 2019), and **vivo** (Kozak and Biecek, 2020) are designed for plot explanations of two or more models next to each other in a single chart.

Interoperability XAI frameworks

Most of the packages we discuss in this article implement model-agnostic explanation methods. Yet, not all of them work smoothly with the different frameworks used to train predictive models. In Table 4, we summarize which packages work with popular machine learning frameworks. Such analysis is important because several models (for example random forest) are available in different frameworks (**mlr**, **scikit-learn**, and others) and although algorithms should be the same, the models differ in terms of names of parameters or interfaces to making predictions. Therefore, to explain models created with various ML frameworks it may be necessary to put additional effort into obtaining the prediction of the model in the form understandable for the XAI framework.

Interoperability with the framework is not binary, and it may require a different amount of work. The packages can support the given framework out of the box, meaning they do not require any additional work to generate an explanation except providing a model or loading any additional library. Packages may also allow for relatively easy use of frameworks, for example, bypassing their own user-defined function that accesses model predictions. If the explanations and model are in two different programming languages, the application of one to another may require even more work, e.g. using the **reticulate** (Ushey et al., 2019) package.

Overall, we compared 27 R packages and six Python libraries in terms of their interoperability with various machine learning frameworks. Four of those frameworks are implemented in R, two of them in Python and one in Java, which is accessible via official R and Python wrappers. Four compared XAI packages, **EIX** (Karbowiak and Biecek, 2020), **forestmodel** (Kennedy, 2020), **randomForestExplainer** (Paluszynska et al., 2020) and **smbinning**, present a model-specific approach and, are designed to work only with a particular group of models. One R package, **mcr**, does not allow model input at all. Instead, it fits and explains its own algorithms. Only one package (i.e. **fscaret** (Szlek et al., 2013)) supports one framework despite implementing model-agnostic methods, which is due to the fact that it is an extension for the **caret** ML framework. To the best of our knowledge, two of the compared packages, **DALEXtra** and **modelStudio** provide support for all types of models taken into consideration. Moreover, there is no universal ML framework that would be supported by all XAI packages. The **caret** package is supported by the highest number of packages, while, on the other hand, **mlr3** by the least, which is probably related to the time of development of these frameworks.

Time of operation

The time of operation is an important aspect of software and varies between different XAI packages. Some explanations are generated in near real-time, while others take a long time to produce results. In Table 5, we have an overview of computation time for different XAI packages. We performed this benchmark to enhance the comparison of R packages. The benchmark was performed on a standard laptop used for everyday work. We computed the time of evaluation of each chunk from Markdown files linked in Section .1.

		R				Python		Java
	Package	mlr	mlr3	parsnip	caret	keras	scikit-learn	h2o
R	ALEPlot	⋆	⋆	⋆	⋆	●	●	⋆
	auditor	⋆	⋆	✓	✓	●	●	⋆
	DALEX	⋆	⋆	✓	✓	●	●	⋆
	DALEXtra	✓	✓	✓	✓	✓	✓	✓
	EIX[2]	-	-	-	-	-	-	-
	ExplainPrediction	⋆	⋆	⋆	⋆	●	●	⋆
	fairness	⋆	⋆	⋆	⋆	●	●	⋆
	fastshap	⋆	⋆	⋆	⋆	●	●	⋆
	flashlight	⋆	⋆	⋆	⋆	●	●	⋆
	forestmodel[3]	-	-	-	-	-	-	-
	fscaret	-	-	-	✓	-	-	-
	iBreakDown	⋆	⋆	✓	✓	●	●	⋆
	ICEbox	⋆	⋆	⋆	⋆	●	●	⋆
	iml	✓	⋆	⋆	✓	●	●	⋆
	ingredients	⋆	⋆	✓	✓	●	●	⋆
	lime	✓	⋆	✓	✓	●	●	✓
	live	⋆	⋆	⋆	⋆	●	●	⋆
	mcr[4]	-	-	-	-	-	-	-
	modelDown	⋆	⋆	✓	✓	●	●	⋆
	modelStudio	✓	✓	✓	✓	✓	✓	✓
	pdp	⋆	⋆	⋆	⋆	●	●	⋆
	randomForestExplainer[5]	-	-	-	-	-	-	-
	shapper	⋆	⋆	⋆	⋆	●	●	⋆
	smbinning[6]	-	-	-	-	-	-	-
	survxai	⋆	⋆	⋆	-	-	-	-
	vip	⋆	⋆	✓	✓	●	●	⋆
	vivo	⋆	⋆	✓	✓	●	●	⋆
Python	aix360[7]	-	-	-	-	-	✓	-
	eli5	-	-	-	-	✓	✓	-
	interpret[8]	-	-	-	-	-	✓	-
	lime	-	-	-	-	✓	✓	⋆
	shap	-	-	-	-	✓	✓	⋆
	skater	●	●	●	●	✓	✓	⋆

Table 4: Interoperability of XAI toolkits. ✓stands for the support that the XAI framework gives for the modeling framework. Support means that computing explanations do not require any additional steps besides providing a model. ⋆ stands for partial support, which means that it is necessary to pass additional user-defined functions that extracts models' predictions. For instance, R package **flashlight**, requires passing two-argument function `function(model, newdata)` that returns prediction vector. • mark symbolizes that obtaining predictions requires greater effort, for instance, manual configuration of R-Python connection via **reticulate**. Keep in mind that marks in the h2o column refer to interoperability with R or Python h2o ports.

Wherever it was possible, we tried to be consistent with the parameters of functions from different packages. For example, the number of samples for Shapley-based method $= 50$. However, given that markdowns contain only sample codes, the default parameters between functions in different packages may cause differences in code evaluation times. Therefore, the benchmark should not be considered the final comparison of the speed of the packages, but rather the support in developing an intuition for the overall performance time.

The packages **DALEXtra**, **modelStudio**, **modelDown**, and **randomForestExplainer** have long computation times, however, they are designed to compute standalone reports that can be later viewed without any additional computations. XAI frameworks, such as, **DALEX**, **flashlight**, and

[2] **EIX** is a package that provide model-specific explanations of packages created with xgboost or **LightGBM** packages.

[3] **forestmodel** is a package that provide model-specific explanations of linear models created with stats package

[4] **mcr** package build its own model and present explanation for them.

[5] **randomForestExplainer** is a package that provide model-specific explanations of forest models created with **randomForest** package

[6] **smbinning** is a package that provide model-specific explanations of scoring linear models created among others with stats package

[7] **aix360** package build its own model and present explanation for them.

[8] **interpret** package build its own model and present explanation for them.

Package	Model parts	Model profile	Model diagnostics	Predict parts	Predict profile	Predict diagnostics	Report	Sum	N
ALEplot	-	0.29	-	-	-	-	-	0.58	2
auditor	-	-	0.18	-	-	-	-	65.88	9
DALEX	6.01	0.52	0.22	1.81	0.49	0.41	-	36.18	18
DALEXtra	-	-	0.42	-	-	-	9.15	11.44	4
EIX	5.15	-	5.14	26.58	-	-	-	42.02	4
ExplainPrediction	38.25	-	-	38.87	-	-	-	77.12	2
fairness	-	-	0.1	-	-	-	-	0.20	2
fastshap	107.92	106.3	-	107.93	-	-	-	430.09	4
flashlight	7.44	0.17	-	0.8	0.17	-	-	50.62	19
fscaret	74.06	-	-	-	-	-	-	74.06	1
iBreakDown	-	-	-	3.55	-	-	-	25.65	3
ICEbox	-	31.36	-	-	-	-	-	91.54	3
iml	42.3	0.27	-	0.42	2.54	-	-	115.02	9
ingredients	9.8	1.35	-	0.25	0.06	-	-	142.21	10
lime	-	-	-	0.4	-	-	-	0.80	2
live	-	-	-	2.5	-	-	-	5.00	2
mcr	-	-	0.48	-	-	-	-	2.18	6
modelDown	-	-	-	-	-	-	62.98	125.95	2
modelStudio	-	-	-	-	-	-	183.09	183.09	1
pdp	0.22	0.06	-	-	-	-	-	0.45	4
randomForest-Explainer	44.47	-	-	-	-	-	478.8	1027.16	7
shapper	-	-	-	4.95	-	-	-	9.90	2
smbinning	-	-	1.41	-	-	-	-	3.85	3
survxai	-	0.85	0.03	-	0.13	-	-	1.86	4
vip	1.69	-	-	-	-	-	-	310.58	5
vivo	1.34	-	-	1.22	-	-	-	2.56	2

Table 5: Summarized times of computations (measured in seconds) for functions used in R markdowns linked in Section .1. Columns that correspond to types of explanations contain median time over chunks. Column Report contains a median time of report generation. Column Sum contains the overall time of computation of all chunks. Column N contains the number of chunks that were evaluated. Dashes mean that the package does not implement such an explanation type. There might be some differences between this Table and Table 3 that comes from the design of packages; for example, **modelStudio** provides Model Profile explanations, yet only inside a report.

iml (Molnar et al., 2018) have a wide range of computation times because of the several different methods they implement. It is worth noting that we calculated all times for the models fitted to the titanic data set that has 2,207 observations, which is a relatively small amount of data. For large data sets, some explanations may take a long time or be impossible to compute.

Recommendations for a gentle introduction to XAI

In this Section, we provide recommendations on how to begin explaining models. We covered the most popular methods, such as permutational variable importance, SHAP explanation, Partial Dependence Profiles, ICE profile, and fairness check. We briefly describe the methods and their important parameters. Moreover, we provide code snippets with examples from the frameworks that we believe are best for getting started with XAI, i.e. **DALEX**, **flashlight**, and **iml**. All examples in this Section are based on a **ranger** (Wright and Ziegler, 2017) model trained on the titanic data set from the **DALEX** package.

```
data(titanic_imputed, package = "DALEX")
ranger_model <- ranger::ranger(survived~., data = titanic_imputed,
                               classification = TRUE, probability = TRUE)
```

Different libraries in R generate models whose predict function returns data in different formats. Model explanation tools need a unified interface for calculating predictions. The following function is such a unifying API. It takes a ranger model and data, then returns a score vector as the result. For a `ranger` object, this requires pulling a second column from the `predictions` slot.

```
flashlight_predict <- function(X.model, new_data)
        predict(X.model, new_data)$predictions[,2]
iml_predict <- function(X.model, newdata)
        predict(X.model, newdata)$predictions[,2]
```

The commonly used XAI technique is the model agnostic assessment of the importance of variables. Permutational variable importance (Fisher et al., 2018) is a popular choice since it works in a model agnostic fashion. We repeatedly permute the values of a variable and examine how the performance of a model changes on average. The more the model's performance decreases, the more important the variable is.

The two most important parameters of this method are the number of iterations and the loss function. More iterations increase the stability of estimation, at the cost of longer execution time. Another important parameter for this method is the performance measure. Note that sometimes a loss function is used for this purpose.

In each of packages **DALEX**, **flashlight**, and **iml** it is possible to specify both the number of permutations (parameters `B`, `m_repetitions`, and `n.repetitions`, respectively) and the loss function (`loss_function`, `metrics`, and `loss`). In each case one can specify also a custom loss function or a function from another package, like **MetricsWeighted** (Mayer, 2020a).

Figure 2 shows the permutational variable importance generated by each of the packages. To make the results as comparable as possible, we tried to keep the same values of parameters for each implementation.

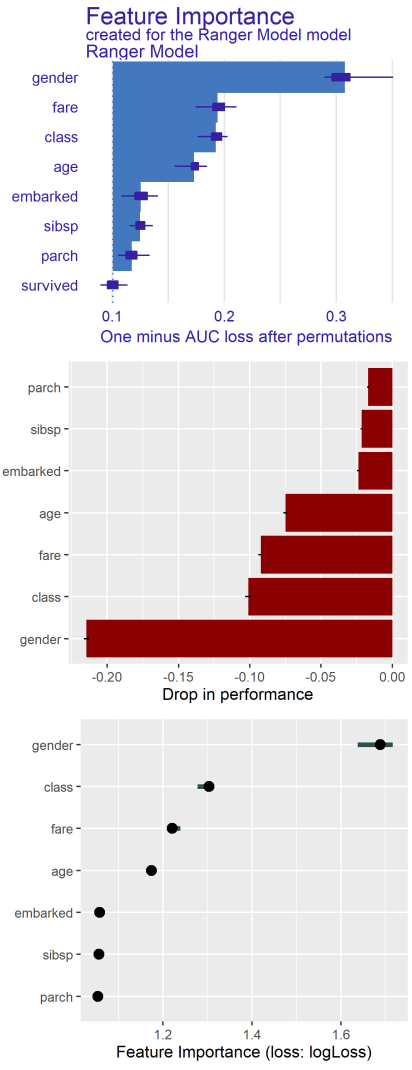


Figure 2: Permutational variable importance examples for different libraries. From the top there are **DALEX**, **flashlight** and **iml**. The computation times of the plots are 4.59s for **DALEX**, 5.64s for **flashlight**, 8.86s for **iml**.

```
library("DALEX")
exp_dalex <-
    explain(ranger_model,
    data  = titanic_imputed,
    y     = titanic_imputed$survived,
    label = "Ranger Model")

fi_dalex <- model_parts(exp_dalex, B = 10,
    loss_function = loss_one_minus_auc)
plot(fi_dalex)
```

```
library("flashlight")
library("MetricsWeighted")
exp_flashlight <-
    flashlight(model = ranger_model,
    data    = titanic_imputed,
    y       = "survived",
    label   = "Titanic Ranger",
   metrics = list(auc = MetricsWeighted::AUC),
   predict_function = flashlight_predict)

fi_flashlight <- light_importance(exp_flashlight,
    m_repetitions = 10)
plot(fi_flashlight, fill = "darkred")
```

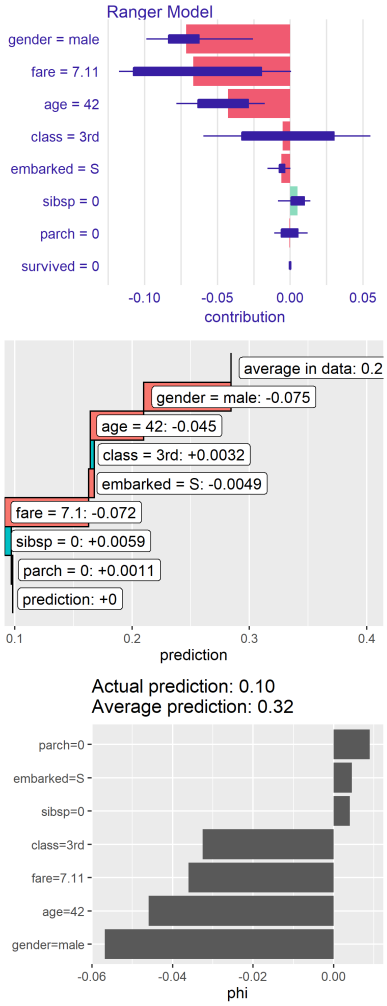
```
library("iml")
X <- titanic_imputed[
which(names(titanic_imputed) != "survived")
]
exp_iml <-
    Predictor$new(ranger_model,
    data = X,
    y = titanic_imputed$survived,
    predict.function = iml_predict)

fi_iml <- FeatureImp$new(exp_iml,
    loss = "logLoss",
    n.repetitions = 10)
plot(fi_iml)
```

For the local prediction level exploration, the most popular XAI method for variable attribution is SHAP (Lundberg and Lee, 2017). The method uses Shapley value concept from game theory to estimate variables' contributions to a model's prediction for one observation.

Shapley values are approximated with Monte-Carlo sampling. The approximation is based on a number of random variables' orderings. The higher the number, the more stable is the approximation. Of course, the higher number of steps, the longer computations are.

In each of the packages **DALEX**, **flashlight**, and **iml** one can specify the number of orderings (parameters `B`, `n_perm`, and `sample.size`, respectively). Additionally for **DALEX** and **flashlight** number of samples taken into consideration during computing the explanation is also an important parameter as it significantly affects computation time (`N` and `n_max` accordingly). This does not apply to **iml** due to the difference in the method chosen to calculate Shapley values.

Figure 3 shows examples of SHAP explanation, note that once the number of orderings was set to the same value across all three packages, then the results are very similar. In this example **DALEX** and **flashlight** are slower than **iml**, in part due to the storing of partial results from individual permutations which makes it possible to determine error boxplots and in part because in each permutation **iml** takes into account a single observation per variable while **DALEX** and **flashlight** takes into account `N / n_max` observations. This is why results for **DALEX** and **flashlight** are in this example more stable than **iml** for the same number of permutations. Note that the fastest implementation of SHAP values in R is available in the **treeshap** (Komisarczyk et al., 2020) package.




```
# DALEX
shap_dalex <-
    predict_parts(exp_dalex,
    new_observation = titanic_imputed[1,],
      type = "shap",
      N    = 50,
      B    = 50)
plot(shap_dalex)
```

```
# flashlight
shap_flashlight <-
    light_breakdown(exp_flashlight,
      new_obs = titanic_imputed[1, ],
      n_max   = 50,
      n_perm  = 50,
      visit_strategy = "permutation"
   )
plot(shap_flashlight)
```

```
# iml
shap_iml <-
    Shapley$new(exp_iml,
      x.interest  = X[1, ],
      sample.size = 50)
plot(shap_iml)
```

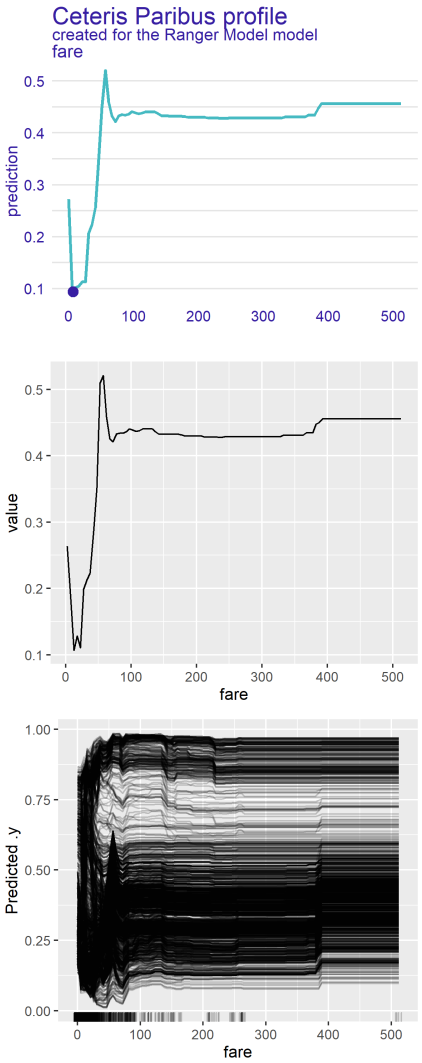
Figure 3: SHAP examples for different libraries. From the top there are **DALEX**, **flashlight** and **iml**. The computation times of the plots are 10.23s for **DALEX**, 7.11s for **flashlight**, 0.33s for **iml**.

Ceteris Paribus (CP) Profiles, known also as ICE curves (Goldstein et al., 2015), are profiles of the variables from the perspective of a single observation. CP profile shows how a model's prediction would change if the value of a single variable changed and the other variables are fixed.

The most important parameter of this method is the number of grid points (different values of a variable) based on which the profile is constructed. It is also important for which observation we are calculating CP, since each observation has a different profile.

In **DALEX**, **flashlight**, and **iml** it is possible to specify both the number of grid points (`grid_points`, `n_bins`, and `grid.size` respectively). **DALEX** and **flashlight** allow to specify one observation for which profile shall be calculated (`new_observation` and `indices` accordingly) while in **iml** packages it is not possible and requires change of the reference data set.

Figure 4 shows an example of Ceteris Paribus profiles calculated using **DALEX**, **flashlight** and **iml** packages. To make results as comparable as possible, we tried to keep the same values of parameters for each implementation. The longer running time for **iml** is due to the fact that it calculates curves for each observation.




```
# DALEX
cp_dalex <-
    predict_profile(exp_dalex,
      new_observation = titanic_imputed[1,],
      variables       = "fare",
      grid_points     = 101)
plot(cp_dalex, variables = "fare")
```

```
# flashlight
cp_flashlight <-
    light_ice(exp_flashlight,
      v       = "fare",
      indices = 1,
      n_bins  = 101)
plot(cp_flashlight)
```

```
# iml
cp_iml <-
    FeatureEffect$new(exp_iml,
      feature   = "fare",
      method    = "ice",
      grid.size = 101)
plot(cp_iml)
```

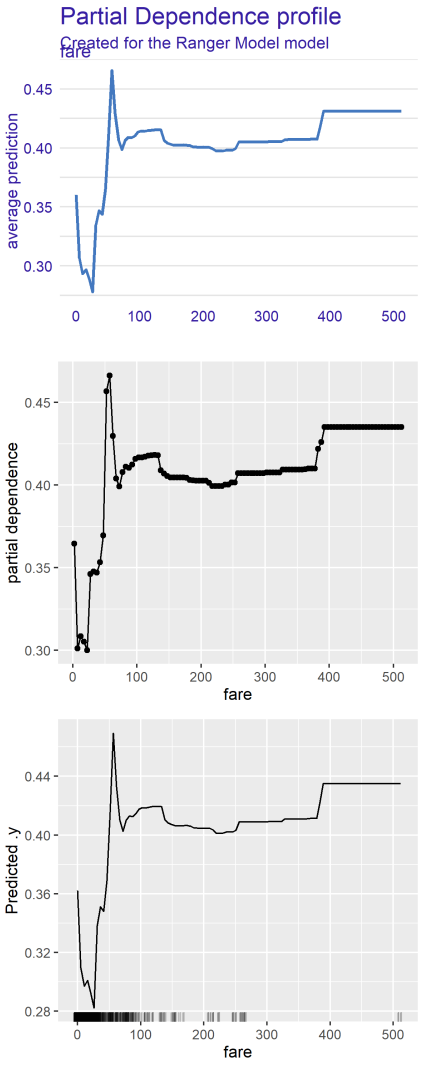
Figure 4: Ceteris Paribus Profiles examples for different libraries. From the top there are **DALEX**, **flashlight** and **iml**. The computation times of the plots are 0.05s for **DALEX**, 0.03s for **flashlight**, 10.02s for **iml**.

Partial Dependence Profile (PDP) is a common XAI technique that is useful to examine variables from a model perspective. It shows the global relationship between the dependent variable and model response by averaging Ceteris Paribus profiles for a chosen variable. Therefore, PDP is the profile of mean prediction that is a function of the independent variable.

First out of the two most important parameters is the number of grid points (values of variables) based on which the profile is constructed. The second is the distribution of the before-mentioned grid points. The PDP will be different, depending on whether we distribute the points uniformly or according to the empirical distribution of the variable.

Grid number of grid points can be changed in all of the three packages (`grid_points` for **DALEX**, `n_bins` for **flashlight** and `grid.size` for **iml**). Yet, the grid distribution can be changed only in the **DALEX** and **flashlight** packages (parameter `variable_splits_type` and `cut_type`). Moreover, only **DALEX** and **flashlight** allow to set the number of observations used to construct PDP (`N` and `pd_n_max` accordingly).

Examples of PDP curves in different packages are in Figure 5. Although we tried to keep the parameters of the method the same, small differences are caused by different observation samples and grid distribution.




```
# DALEX
pdp_dalex <-
    model_profile(
      exp_dalex,
      variables   = "fare",
      type        = "partial",
      N           = 1000,
      grid_points = 101,
      variable_splits_type = "uniform")
plot(pdp_dalex)
```

```
# flashlight
pdp_flashlight <-
    light_profile(exp_flashlight,
      v        = "fare",
      type     = "partial dependence",
      pd_n_max = 1000,
      n_bins   = 101,
      cut_type = "equal")
plot(pdp_flashlight)
```

```
# iml
pdp_iml <-
    FeatureEffect$new(exp_iml,
      feature   = "fare",
      method    = "pdp",
      grid.size = 101)
plot(pdp_iml)
```

Figure 5: Partial Dependence Profiles examples for different libraries. From the top there are **DALEX**, **flashlight** and **iml**. The computation times of the plots are 2.99s for **DALEX**, 3.18s for **flashlight**, 11.28 s for **iml**.

Unique features of selected XAI packages

In the previous Section, we have covered popular and basic XAI techniques with examples from the frameworks, such as **DALEX**, **flashlight**, and **iml**. In this Section, we present general recommendations for those three packages based on our experience in XAI. For key functionalities, each of these packages offers similar methods that can be used in similar ways. However, these packages differ in additional specialized functionality.

The **DALEX** package provides a unified wrapper for machine learning models that can be later utilized by other XAI packages. A rich system of additional packages allows many non-standard applications. For example, the **modelDown** package allows to automatically convert an explainer into static html documentation based on `pkgdown` templates, which can be versioned or shared with other users. The xai2shiny (Adam et al., 2021) package allows to automatically convert an explainer with just one line of code into a shiny application which allows for dynamic model exploration. The **modelStudio** and **arenar** (Piątyszek and Biecek, 2020) packages allow you to build, with a single command, an interactive java script based tool for model exploration using the IEMA approach. The **fairmodels** (Wiśniewski and Biecek, 2020) package does fairness analysis of the model for similar behavior in subpopulations. The **triplot** (Pekala and Biecek, 2020) package allows you to calculate the importance of not only individual variables but also whole aspects of variables. The **auditor** package allows very detailed analysis of model residuals while the **drifter** package allows analysis of drift in the data and in the model structure. In addition, the design of the **DALEX** assumes that each explanation can be presented in Rashomon's perspective. This means that any number of explainers can be placed on a single graph, e.g., for cross-comparison of different models. This helps to see their advantages and disadvantages.

The **flashlight** package, similarly to **DALEX**, allows to combine explanations for various models and plot them together. It also provides a surrogate tree models' explanation. However, the biggest advantage of the **flashlight** package, not implemented in any other XAI framework covered in this paper, is the possibility to put weights on observations while computing the explanations. Thus, it is great for applications where observations have sampling weights.

The most distinctive attribute of the **iml** package is implementation on the basis of R6 classes. Additionally, due to the differences in the method of calculation of SHAP values, it computes that explanation faster than **DALEX** and **flashlight**. The **iml** just like **flashlight** provides surrogate tree model explanation and model level interactions based on Friedman H-statistics.

Discussion

The article compared 27 different R packages with methods for XAI analysis of predictive models. The selection criteria (Section 2.3.1) limited the pool of packages that were analyzed. However, we believe that the constraints resulted in the fact that all considered libraries were mature and have a group of everyday users. Moreover, the fact that the package is published on CRAN proves that it is operational, has been tested, and is being maintained.

The first observation refers to explanation methods. Analysis of explanation tasks that are covered by various R packages shows that the overall distribution of those tasks over libraries is changing in time. Back in the day, explanations of predictive models were focused on a global level, with popular profiling methods, such as Partial Dependence, such as (**pdp**, September 2016) or Accumulated Local Effect plots (**ALEplot**, November 2017). The Model Parts task also fit that trend with the **vip** package (June 2018) and variable importance related function in **pdp**. Nowadays, the Predict Parts Task seems to be more and more popular, especially methods related to Shapley values. On top of previously published packages, such as **shapper** (March 2019) and **fastshap** (November 2019), there are new, recently created tools. **shapr** (Sellereite and Jullum, 2019), **SHAPforxgboost** (Liu and Just, 2020) and **treeshap**, which still awaits publication on CRAN, are examples of such new packages. Not to mention other Predict Parts explanation methods like BreakDown. Another new trend in eXplainable Artificial Intelligence is fairness. There was only one package dedicated to fairness analysis in our comparison, but a number of recently published packages still await their reception. There are, for instance, **fairmodels**, **fairml** (Scutari, 2020) and **aif360** (de Queiroz et al., 2020) packages. Their number, the short time from publication, and the fact that **fairness** was first published in September 2019, support the statement that fairness is a new, popular trend in R XAI that cannot be overseen.

The packages that were compared were designed to serve as eXplainable Artificial Intelligence support for model creators and users. However, different tools can be used in the various stages of the model development process. It is easy to explain with the help of an example from our list. The main target group of **flashlight** users is different than the target of **modelStudio** or **modelDown**. The first one is a tool that serves mostly model developers, so they can fit models based on XAI experience, which is an important part of the model life cycle. The second package is dedicated to end-users;

those who get a ready model and want to explore its behavior, rather than fitting a new one. That second group of packages is an important new trend in the XAI toolkits world. They are a gateway to model exploration for those who lack expert knowledge about XAI and modeling itself but want to familiarize themselves with the behavior of the model they are using. This trend is also followed by recently developed packages such as **arenar** or **xai2cloud** (Rydelek, 2020).

One more aspect that distinguishes the compared packages is their complexity. On the one hand, some packages focus on one method like **pdp** or **ALEPlot**, while, on the other hand, there are packages, such as **iml** or **DALEX**, that provide a wide range of different explanation method. There are also tools that present a unique approach to certain areas of XAI. We find it necessary to acknowledge those.

- **DALEXtra** - is designed to bridge XAI packages with frameworks for ML model developments including those coming from different programming languages like Python **scikit-learn**. It also helps in finding a solution to the Rashomon effect problem, by providing an interface to compare model explanations and performance.
- **flashlight** - this is one of the tools that provides a wide range of explanation methods for complex model analysis. What distinguishes it from similar **DALEX** and **iml** packages is weight use cases. **flashlight** allows users to specify the weight to every observation and consider them while computing explanations.
- **modelStudio** and **arenar** - both of these packages provide an easy to use interface to model explanations. Such analysis requires no expert knowledge, since, with a few lines of code, a stand-alone HTML document with precalculated explanations (**modelStudio**), or an online browser application (**arenar**) can be created. Moreover, **arenar** is capable of presenting explanations for more than one model at once and comparing them (See Rashomon Effect in Section 2.3.3).
- **treeshap** - this is an example of a new, yet-to-be published package that could be an important tool in the future. It uses the additive nature of Shapley values and allows them to be directly computed for ensemble tree-like models. In addition, the implementation in C++ makes the computations fast.
- **triplot** implements explanations that take into account the correlation of variables. This tool, instead of considering a single variable, explains a model using aspects. An aspect is a group of variables (usually correlated ones) that is considered as one variable.
- **vip** - this is a package oriented around variable importance. It proposes an interesting concept of mixing up model-specific and model-agnostic explanation types. For some types of models, a natural variable importance measure can be extracted, while, at the same time, there is an option to access that measure in a model-agnostic way. It provides three different ways of computing model-agnostics variable importance, including permutational variable importance, Shapley-based variable importance, and variance-based variable importance.

Summary

In this article, we presented R packages dedicated to eXplainable Artificial Intelligence. We started demonstrating the importance of XAI in today's world (see Section 2.1), and we discussed the issue of previously presented taxonomies. We proposed a taxonomy dedicated to predictive model explanations (see Section 2.2).

Research of the literature has shown that there are plenty of R packages dedicated to XAI. Each of them is extensive to a different degree (Section 2.3.2), providing a wide range of various opportunities for explaining models using a number of tools. On the one hand, there are packages such as **ALEplot** (Apley, 2018) and **lime** (Pedersen and Benesty, 2019) that implement one specific method, or are oriented around one explanation task. On the other hand, packages such as **flashlight** or **DALEX** implement diverse methods belonging to different explanation tasks. Explanation of predictive models can also be utilized in the process of finding a model that is better for a given use-case (Section 2.3.3). This process is simplified by the interfaces provided by some of the frameworks. Section 2.3.4 shows that R packages for XAI are flexible in terms of interoperability with frameworks used for model development. This means that XAI methods can be used for any type of predictive model.

We believe that examples presented in Section .1 and in the `xai-tools.drwhy.ai` webpage will encourage more frequent use of XAI tools during the modeling process. The large range of diversity among R packages that provide explanations of machine learning models means that everyone should be able to find a method convenient to use.

Acknowledgements

We would like to thank the whole MI^{2}DataLab team, especially Hubert Baniecki and Anna Kozak for the discussions and support. Last, but not least, we would like to thank Patrick Hall for his valuable comments. Work on this paper was financially supported by the NCN Opus grant 2017/27/B/ST6/0130.

Bibliography

R. Adam, M. Polakowski, and P. Biecek. *xai2shiny: Creates Shiny Application From A DALEX Explainer*, 2021. R package version 1.1.0. [p14]

S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza. Power to the People: The Role of Humans in Interactive Machine Learning. *AI Magazine*, 35(4):105–120, Dec. 2014. URL http://doi.org/10.1609/aimag.v35i4.2513. [p2]

D. Apley. *ALEPlot: Accumulated Local Effects (ALE) Plots and Partial Dependence (PD) Plots*, 2018. URL https://CRAN.R-project.org/package=ALEPlot. R package version 1.1. [p15, 22]

D. W. Apley and J. Zhu. Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models. *arXiv*, 2016. URL https://arxiv.org/abs/1612.08468. [p4]

V. Arya, R. K. E. Bellamy, P.-Y. Chen, A. Dhurandhar, M. Hind, S. C. Hoffman, S. Houde, Q. V. Liao, R. Luss, A. Mojsilović, S. Mourad, P. Pedemonte, R. Raghavendra, J. T. Richards, P. Sattigeri, K. Shanmugam, M. Singh, K. R. Varshney, D. Wei, and Y. Zhang. Ai explainability 360: An extensible toolkit for understanding data and machine learning models. *Journal of Machine Learning Research*, 21(130):1–6, 2020. URL http://jmlr.org/papers/v21/19-1035.html. [p22]

H. Baniecki and P. Biecek. modelStudio: Interactive Studio with Explanations for ML Predictive Models. *Journal of Open Source Software*, 4(43):1798, Nov 2019. URL https://doi.org/10.21105/joss.01798. [p7, 25]

P. Biecek. DALEX: Explainers for Complex Predictive Models in R. *Journal of Machine Learning Research*, 19(84):1–5, 2018. URL http://jmlr.org/papers/v19/18-416.html. [p22]

P. Biecek and T. Burzykowski. *Explanatory Model Analysis. Explore, Explain, and Examine Predictive Models*. Chapman and Hall/CRC, New York, 2021. URL https://pbiecek.github.io/ema/. [p2, 4, 22]

P. Biecek, H. Baniecki, A. Izdebski, and K. Pekala. *ingredients: Effects and Importances of Model Ingredients*, 2019. URL http://CRAN.R-project.org/package=ingredients. [p7, 23]

O. Biran and C. Cotton. Explanation and justification in machine learning: A survey. In *IJCAI-17 workshop on explainable AI (XAI)*, volume 8, pages 8–13, 2017. [p2]

B. Bischl, M. Lang, L. Kotthoff, J. Schiffner, J. Richter, E. Studerus, G. Casalicchio, and Z. M. Jones. mlr: Machine Learning in R. *Journal of Machine Learning Research*, 17(170):1–5, 2016. URL http://jmlr.org/papers/v17/15-066.html. [p1]

L. Breiman. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statist. Sci.*, 16(3):199–231, 08 2001. doi: 10.1214/ss/1009213726. URL https://doi.org/10.1214/ss/1009213726. [p6]

R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad. Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-Day Readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 1721–1730, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336642. URL https://doi.org/10.1145/2783258.2788613. [p24]

A. Chatzimparmpas, R. M. Martins, I. Jusufi, and A. Kerren. A survey of surveys on the use of visualization for interpreting machine learning models. *Information Visualization*, 19(3):207–233, 2020. URL https://doi.org/10.1177/1473871620904671. [p2]

F. Chollet et al. Keras. https://keras.io, 2015. [p1]

J. Choo and S. Liu. Visual analytics for explainable deep learning. *IEEE computer graphics and applications*, 38(4):84–92, 2018. URL http://doi.org/10.1109/MCG.2018.042731661. [p2]

G. de Queiroz, S. Ronaghan, and S. Swaminathan. *aif360: Help Detect and Mitigate Bias in Machine Learning Models*, 2020. URL https://CRAN.R-project.org/package=aif360. R package version 0.1.0. [p14]

J. J. Dudley and P. O. Kristensson. A review of user interface design for interactive machine learning. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 8(2):1–37, 2018. URL https://doi.org/10.1145/3185517. [p2]

A. Fisher, C. Rudin, and F. Dominici. All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously. *arXiv*, 2018. URL https://arxiv.org/abs/1801.01489. [p4, 10]

J. Friedman, T. Hastie, and R. Tibshirani. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1):1–22, 2010. URL https://doi.org/10.18637/jss.v033.i01. [p23]

J. H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics*, 29:1189–1232, 2000. URL https://doi.org/10.1214/aos/1013203451. [p4]

M. K. C. from Jed Wing, S. Weston, A. Williams, C. Keefer, A. Engelhardt, T. Cooper, Z. Mayer, B. Kenkel, the R Core Team, M. Benesty, R. Lescarbeau, A. Ziem, L. Scrucca, Y. Tang, C. Candan, and T. Hunt. *caret: Classification and Regression Training*, 2019. URL https://CRAN.R-project.org/package=caret. R package version 6.0-84. [p1]

R. Garcia, A. C. Telea, B. C. da Silva, J. Tørresen, and J. L. D. Comba. A task-and-technique centered survey on visual analytics for deep learning model engineering. *Computers & Graphics*, 77:30–49, Dec. 2018. URL https://doi.org/10.1016/j.cag.2018.09.018. [p2]

N. Gill, P. Hall, K. Montgomery, and N. Schmidt. A Responsible Machine Learning Workflow with Focus on Interpretable Models, Post-hoc Explanation, and Discrimination Testing. *Information*, 11 (3):137, 2020. URL http://doi.org/10.3390/info11030137. [p1, 2]

L. Gilpin, D. Bau, B. Yuan, A. Bajwa, M. Specter, and L. Kagal. Explaining Explanations: An Overview of Interpretability of Machine Learning. pages 80–89, 10 2018. URL https://doi.org/10.1109/DSAA.2018.00018. [p2]

A. Goldstein, A. Kapelner, J. Bleich, and E. Pitkin. Peeking Inside the Black Box: Visualizing Statistical Learning With Plots of Individual Conditional Expectation. *Journal of Computational and Graphical Statistics*, 24(1):44–65, 2015. URL https://doi.org/10.1080/10618600.2014.907095. [p4, 12, 23]

B. Goodman and S. Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine*, 38(3):50–57, Oct. 2017. URL http://doi.org/10.1609/aimag.v38i3.2741. [p1]

A. Gosiewska and P. Biecek. auditor: an R Package for Model-Agnostic Visual Validation and Diagnostics. *The R Journal*, 11(2):85–98, 2019a. URL https://doi.org/10.32614/RJ-2019-036. [p7, 22]

A. Gosiewska and P. Biecek. Do Not Trust Additive Explanations. *arXiv*, 2019b. URL https://arxiv.org/abs/1903.11420v3. [p4, 23]

B. Greenwell. *fastshap: Fast Approximate Shapley Values*, 2020. URL https://CRAN.R-project.org/package=fastshap. R package version 0.0.5. [p22]

B. Greenwell, B. Boehmke, and B. Gray. *vip: Variable Importance Plots*, 2020. URL https://CRAN.R-project.org/package=vip. R package version 0.2.2. [p26]

B. M. Greenwell. pdp: An R Package for Constructing Partial Dependence Plots. *The R Journal*, 9(1): 421–436, 2017. URL http://doi.org/10.32614/RJ-2017-016. [p25]

A. Grudziaz, A. Gosiewska, and P. Biecek. survxai: an R package for structure-agnostic explanations of survival models. *Journal of Open Source Software*, 3(31):961, 2018. doi: 10.21105/joss.00961. URL https://doi.org/10.21105/joss.00961. [p26]

D. Gunning. *Explainable Artificial Intelligence (XAI)*, 2017. URL https://www.darpa.mil/attachments/XAIProgramUpdate.pdf. [p2]

H2O.ai. *H2O: Scalable Machine Learning*, 2015. URL http://www.h2o.ai. version 3.1.0.99999. [p1]

M. Hahsler. arulesViz: Interactive Visualization of Association Rules with R. *The R Journal*, 9(2): 163–175, 2017. doi: 10.32614/RJ-2017-047. URL https://doi.org/10.32614/RJ-2017-047. [p22]

M. Hahsler, S. Chelluboina, K. Hornik, and C. Buchta. The arules R-Package Ecosystem: Analyzing Interesting Patterns from Large Transaction Datasets. *Journal of Machine Learning Research*, 12: 1977–1981, 2011. URL http://jmlr.csail.mit.edu/papers/v12/hahsler11a.html. [p2, 22]

M. Hahsler, I. Johnson, T. Kliegr, and J. Kuchař. Associative Classification in R: arc, arulesCBA, and rCBA. *The R Journal*, 11(2):254–267, 2019. doi: 10.32614/RJ-2019-048. URL https://doi.org/10.32614/RJ-2019-048. [p22]

P. Hall. On the Art and Science of Machine Learning Explanations. *arXiv*, 2018. URL https://arxiv.org/abs/1810.02909. [p5]

P. Hall. awesome-machine-learning-interpretability. https://github.com/jphall663/awesome-machine-learning-interpretability, 2020. Access: 2020-07-01. [p4]

P. Hall and N. Gill. *An Introduction to Machine Learning Interpretability: An Applied Perspective on Fairness, Accountability, Transparency, and Explainable AI*. O'Reilly Media, 2018. ISBN 9781492033141. URL https://books.google.pl/books?id=vilpuwEACAAJ. [p5]

F. Hohman, M. Kahng, R. Pienta, and D. H. Chau. Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers. *IEEE Transactions on Visualization and Computer Graphics*, 25:2674–2693, 2019. URL http://doi.org/10.1109/TVCG.2018.2843369. [p2]

T. Hothorn, K. Hornik, and A. Zeileis. Unbiased Recursive Partitioning: A Conditional Inference Framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674, 2006. URL http://doi.org/10.1198/106186006X133933. [p2, 25]

H. Jopia. *smbinning: Scoring Modeling and Optimal Binning*, 2019. URL https://CRAN.R-project.org/package=smbinning. R package version 0.9. [p5, 26]

E. Karbowiak and P. Biecek. *EIX: Explain Interactions in 'XGBoost'*, 2020. URL https://CRAN.R-project.org/package=EIX. R package version 1.1. [p7, 22]

N. Kennedy. *forestmodel: Forest Plots from Regression Models*, 2020. URL https://CRAN.R-project.org/package=forestmodel. R package version 0.6.2. [p7, 23]

K. Komisarczyk, P. Kozminski, and P. Biecek. *treeshap: Fast shap values computations for ensamble models*, 2020. URL https://github.com/ModelOriented/treeshap. R package version 0.0.0.9000. [p11]

M. Korobov and K. Lopuhin. ELI5. https://eli5.readthedocs.io/en/latest/index.html, 2020. Accessed: 2020-07-21. [p22]

A. Kozak and P. Biecek. *vivo: Variable Importance via Oscillations*, 2020. URL https://CRAN.R-project.org/package=vivo. R package version 0.2.0. [p7, 26]

N. Kozodoi and T. V. Varga. *fairness: Algorithmic Fairness Metrics*, 2020. URL https://CRAN.R-project.org/package=fairness. R package version 1.1.1. [p22]

M. Kuhn and H. Wickham. *Tidymodels: a collection of packages for modeling and machine learning using tidyverse principles.*, 2020. URL https://www.tidymodels.org. [p1]

M. Lang, M. Binder, J. Richter, P. Schratz, F. Pfisterer, S. Coors, Q. Au, G. Casalicchio, L. Kotthoff, and B. Bischl. mlr3: A modern object-oriented machine learning framework in r. *Journal of Open Source Software*, 4(44):1903, 2019. URL https://doi.org/10.21105/joss.01903. [p1]

J. Lei, M. G'Sell, A. Rinaldo, R. J. Tibshirani, and L. Wasserman. Distribution-Free Predictive Inference for Regression. *Journal of the American Statistical Association*, 113(523):1094–1111, 2018. URL https://doi.org/10.1080/01621459.2017.1307116. [p4]

S. Liu, X. Wang, M. Liu, and J. Zhu. Towards better analysis of machine learning models: A visual analytics perspective. *Visual Informatics*, 1(1):48 – 56, 2017. ISSN 2468-502X. URL https://doi.org/10.1016/j.visinf.2017.01.006. [p2]

Y. Liu and A. Just. *SHAPforxgboost: SHAP Plots for 'XGBoost'*, 2020. URL https://CRAN.R-project.org/package=SHAPforxgboost. R package version 0.0.4. [p14]

J. Lu, W. Chen, Y. Ma, J. Ke, Z. Li, F. Zhang, and R. Maciejewski. Recent progress and trends in predictive visual analytics. *Frontiers of Computer Science*, 11(2):192–207, 2017a. URL http://doi.org/10.1007/s11704-016-6028-y. [p2]

Y. Lu, R. Garcia, B. Hansen, M. Gleicher, and R. Maciejewski. The State-of-the-Art in Predictive Visual Analytics. *Computer Graphics Forum*, 36(3):539–562, 2017b. URL https://doi.org/10.1111/cgf.13210. [p2]

S. M. Lundberg and S.-I. Lee. A Unified Approach to Interpreting Model Predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf. [p4, 11, 26]

M. Majka. *naivebayes: High Performance Implementation of the Naive Bayes Algorithm in R*, 2019. URL https://CRAN.R-project.org/package=naivebayes. R package version 0.9.7. [p23]

S. Maksymiuk and P. Biecek. *DALEXtra: Extension for 'DALEX' Package*, 2020. URL https://CRAN.R-project.org/package=DALEXtra. R package version 2.0.0. [p7, 22]

S. Maksymiuk, A. Gosiewska, and P. Biecek. *shapper: Wrapper of Python Library 'shap'*, 2019. URL https://CRAN.R-project.org/package=shapper. R package version 0.1.2. [p26]

E. Manuilova, A. Schuetzenmeister, and F. Model. *mcr: Method Comparison Regression*, 2014. URL https://CRAN.R-project.org/package=mcr. R package version 1.2.1. [p5, 25]

M. Mayer. *MetricsWeighted: Weighted Metrics, Scoring Functions and Performance Measures for Machine Learning*, 2020a. URL https://CRAN.R-project.org/package=MetricsWeighted. R package version 0.5.1. [p10]

M. Mayer. *flashlight: Shed Light on Black Box Machine Learning Models*, 2020b. URL https://CRAN.R-project.org/package=flashlight. R package version 0.7.0. [p7, 22]

C. Molnar. *Interpretable Machine Learning*. 2019. https://christophm.github.io/interpretable-ml-book/. [p2, 23]

C. Molnar, B. Bischl, and G. Casalicchio. iml: An R package for Interpretable Machine Learning. *JOSS*, 3(26):786, 2018. URL http://doi.org/10.21105/joss.00786. [p9]

H. Nori, S. Jenkins, P. Koch, and R. Caruana. InterpretML: A Unified Framework for Machine Learning Interpretability. *arXiv*, 2019. URL https://arxiv.org/abs/1909.09223. [p24]

Oracle and contributors. skater, 2020. URL https://github.com/datascienceinc/skater/. [p26]

A. Paluszynska, P. Biecek, and Y. Jiang. *randomForestExplainer: Explaining and Visualizing Random Forests in Terms of Variable Importance*, 2020. URL https://CRAN.R-project.org/package=randomForestExplainer. R package version 0.10.1. [p7, 26]

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf. [p1]

T. L. Pedersen and M. Benesty. *lime: Local Interpretable Model-Agnostic Explanations*, 2019. URL https://CRAN.R-project.org/package=lime. R package version 0.5.1. [p15, 25]

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. [p1]

K. Pekala and P. Biecek. *triplot: Explaining Correlated Features in Machine Learning Models*, 2020. URL https://CRAN.R-project.org/package=triplot. R package version 1.3.0. [p14]

P. Piątyszek and P. Biecek. *arenar: Arena for the Exploration and Comparison of any ML Models*, 2020. URL https://CRAN.R-project.org/package=arenar. R package version 0.1.8. [p14]

D. Rafacz, H. Baniecki, S. Maksymiuk, and M. Bakala. *deepdep: Visualise and Explore the Deep Dependencies of R Packages*, 2020. URL https://CRAN.R-project.org/package=deepdep. R package version 0.2.1. [p5]

M. T. Ribeiro, S. Singh, and C. Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016. URL https://doi.org/10.18653/v1/n16-3020. [p4, 25]

M. Robnik-Sikonja. *ExplainPrediction: Explanation of Predictions for Classification and Regression Models*, 2018. URL https://CRAN.R-project.org/package=ExplainPrediction. R package version 1.3.0. [p22]

K. Romaszko, M. Tatarynowicz, M. Urbański, and P. Biecek. modelDown: automated website generator with interpretable documentation for predictive machine learning models. *Journal of Open Source Software*, 4(38):1444, 2019. doi: 10.21105/joss.01444. URL https://doi.org/10.21105/joss.01444. [p7, 25]

A. Rydelek. *xai2cloud: Deploys An Explainer To The Cloud*, 2020. https://github.com/ModelOriented/xai2cloud. [p15]

D. Sacha, L. Zhang, M. Sedlmair, J. A. Lee, J. Peltonen, D. Weiskopf, S. C. North, and D. A. Keim. Visual interaction with dimensionality reduction: A structured literature analysis. *IEEE transactions on visualization and computer graphics*, 23(1):241–250, 2016. [p2]

K. Schliep and K. Hechenbichler. *kknn: Weighted k-Nearest Neighbors*, 2016. URL https://CRAN.R-project.org/package=kknn. R package version 1.3.1. [p2, 24]

M. Scutari. *fairml: Fair Models in Machine Learning*, 2020. URL https://CRAN.R-project.org/package=fairml. R package version 0.2. [p14]

N. Sellereite and M. Jullum. shapr: An R-package for explaining machine learning models with dependence-aware Shapley values. *Journal of Open Source Software*, 5(46):2027, 2019. URL https://doi.org/10.21105/joss.02027. [p14]

M. Staniak and P. Biecek. Explanations of Model Predictions with live and breakDown Packages. *The R Journal*, 10(2):395–409, 2018. URL https://doi.org/10.32614/RJ-2018-072. [p25]

J. Szlek, P. Adam, L. Raymond, J. Renata, and M. Aleksander. Heuristic modeling of macromolecule release from PLGA microspheres. *International Journal of Nanomedicine*, 8(1):4601–4611, 2013. URL http://doi.org/10.2147/IJN.S53364. R package version 0.8.5.3. [p7, 23]

K. Ushey, J. Allaire, and Y. Tang. *reticulate: Interface to 'Python'*, 2019. URL https://CRAN.R-project.org/package=reticulate. R package version 1.14. [p7]

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. URL http://www.stats.ox.ac.uk/pub/MASS4. ISBN 0-387-95457-0. [p2]

Wikipedia. Rashomon effect, 2020. URL https://en.wikipedia.org/wiki/Rashomon_effect. Online; accessed 22-09-2020. [p6]

J. Wiśniewski and P. Biecek. *fairmodels: Flexible Tool for Bias Detection, Visualization, and Mitigation*, 2020. URL https://CRAN.R-project.org/package=fairmodels. R package version 0.1.1. [p14]

M. N. Wright and A. Ziegler. ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. *Journal of Statistical Software*, 77(1):1–17, 2017. doi: 10.18637/jss.v077.i01. [p9]

B. Yu and K. Kumbier. Veridical data science. *Proceedings of the National Academy of Sciences*, 117(8): 3920–3929, 2020. ISSN 0027-8424. URL https://doi.org/10.1214/aos/1013203451. [p2]

Szymon Maksymiuk
Faculty of Mathematics and Information Science
Warsaw University of Technology
Poland
ORCiD: 0000-0002-3120-1601
sz.maksymiuk@gmail.com

Alicja Gosiewska
Faculty of Mathematics and Information Science
Warsaw University of Technology
Poland

ORCiD: 0000-0001-6563-5742
gosiewska@gmail.com

Przemysław Biecek
Faculty of Mathematics and Information Science
Warsaw University of Technology
Faculty of Mathematics, Informatics, and Mechanics
University of Warsaw
Poland
ORCiD: 0000-0001-8423-1823
przemyslaw.biecek@gmail.com

Appendix: Example gallery for XAI packages

The following paragraphs briefly discuss consecutive XAI packages. For each of the packages listed in Table 2, we prepared an example use-case in the form of a **markdown** document. The documents have similar chapters and sections to facilitate a comparison of the capabilities of each package. Also, in each use-case, we use the titanic dataset from **DALEX** package. It consists of both numerical and categorical dependent variables, and, therefore, it was possible to inspect how toolkits handle different types of features.

The Python library **aix360** (Arya et al., 2020) is a framework that aims at interpretability and explainability of datasets and machine learning models with the help of a method designed for that purpose. It implements, among others, local post-hoc methods like SHAP and LIME, global post-hoc *profWeight*. **aix360** also provides direct local and global-local explanations (for instance for Generalized Linear Models) as well as methods for explaining the data.

The R package **ALEPlot** (Apley, 2018) creates ALE profiles that demonstrate the behavior of the predictive model response based on one or two dependent variables. The method is described as an improvement to Partial Dependence Profiles and the library can work with any type of model. Find an example at `http://xai-tools.drwhy.ai/ALEplot.html`.

The R package **arules** (Hahsler et al., 2011) is dedicated for transaction data analysis. They provide a wide range of methods for rule modeling. **arulesCBA** (Hahsler et al., 2019) for **arules** is an extension package and provides support for classification glass-box predictive models. Another extension, **arulesViz** (Hahsler, 2017), equips users with plenty of methods to visualize and inspect a fitted model. Find an example at `http://xai-tools.drwhy.ai/arules.html`.

The R package **auditor** (Gosiewska and Biecek, 2019a) is dedicated for complex model diagnostics. It provides plots and metrics that allow the user to evaluate the performance of a model. On top of that, packages make it possible to conduct complex residual-based model diagnostics. Find an example at `http://xai-tools.drwhy.ai/auditor.html`.

The R package **DALEX** (Biecek, 2018) provides tools for Explanatory Model Analysis (Biecek and Burzykowski, 2021). It represents the global explanation approach by serving variable importance interface, ability to create Partial, Accumulated, and Conditional Dependence Profiles. It facilitates the performance of global and local residual-based model diagnostics. On the local explanations side, it implements SHAP, BreakDown, oscillation contributions, and Ceteris Paribus (ICE). Examples of PDP and SHAP explanations can be seen in the Figure 6 and Figure 7. Find an example at `http://xai-tools.drwhy.ai/DALEX.html`.

The R package **DALEXtra** (Maksymiuk and Biecek, 2020) serves as an extension for **DALEX**. Its main purpose is to provide a set of pre-defined predict functions for different ML frameworks. This facilitates the integration of XAI packages with modest popular model classes. Find an example at `http://xai-tools.drwhy.ai/DALEXtra.html`.

The R package **EIX** (Karbowiak and Biecek, 2020) is a model-specific tool designed to explain **xgboost** models. It provides the ability to diagnose the model by plotting its structure, find variables with the biggest interaction, and perform variable importance using them. It is also possible to explain a single observation using this package. Find an example at `http://xai-tools.drwhy.ai/EIX.html`.

The Python library **eli5** (Korobov and Lopuhin, 2020) provides two ways to inspect black-boxes: permutation importance and text explanations with LIME. The library **eli5** supports the most common Python frameworks and packages: scikit-learn, Keras, xgboost, LightGBM, CatBoost, lightning, and sklearn.

The R package **ExplainPrediction** (Robnik-Sikonja, 2018) allows users to explain predictive model instances through instances showing how values of variables contributed to the prediction of each observation in test data. Explanations can be then aggregated into one global variable importance. Find an example at `http://xai-tools.drwhy.ai/ExplainPrediction.html`.

The R package **fairness** (Kozodoi and V. Varga, 2020) is dedicated for fairness analysis. It provides 9 different metrics that allow the user to recognize if predicted values contain bias. Plots of metrics and density of probabilities in subgroups are also included. Find an example at `http://xai-tools.drwhy.ai/fairness.html`.

The R package **fastshap** (Greenwell, 2020) approximates Shapley values for any type of predictive model. Through this tool variable importance, profiles of variables, and contributions for a single observation can be acquired. Provides an interface for Python **shap** plots. An example of SHAP can be seen in the Figure 7. Find an example at `http://xai-tools.drwhy.ai/fastshap.html`.

The R package **flashlight** (Mayer, 2020b) provides methods that can be used for wide-model analysis, including variable importance, and methods of profiling variables like PDP, ALE, residual, target, and predicted value profiles. A single prediction can be explained with this tool as well with

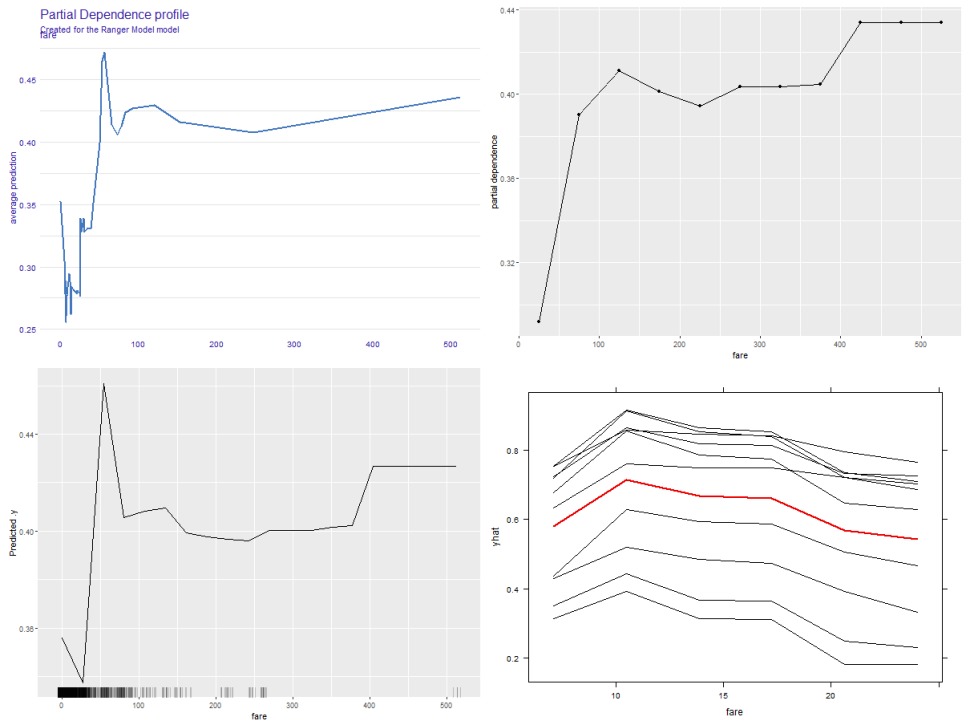


Figure 6: Partial Dependence Porfiles for the fare variable from the *titanic* dataset generated with **DALEX** (top-left), **flashlight** (right-top), **iml** (left-bottom), **pdp** (right-bottom). We can see that the profiles differs, which is due to the fact that profiles are calculated by default on different grids of points. The computation times of the plots are 0.42s for **DALEX**, 0.34s for **flashlight**, 1.88s for **iml**, 0.03s for **pdp**.

the help of SHAP, BreakDown, and ICE. An example of PDP explanation can be seen in Figure 6. Find an example at `http://xai-tools.drwhy.ai/flashlight.html`.

The R package **forestmodel** (Kennedy, 2020) generates forest plots of the estimated coefficients of models. The library supports objects produced by `lm()`, `glm()`, and `survival::coxph()` functions. Find an example at `http://xai-tools.drwhy.ai/forestmodel.html`.

The R package **fscaret** (Szlek et al., 2013) is associated with **caret**. It can train various types of predictive models while acquiring variable importance during that process. Any type of model supported by **caret** can be explained. Find an example at `http://xai-tools.drwhy.ai/fscaret.html`.

The R package **glmnet** (Friedman et al., 2010) is an interface for fitting Generalized Linear Models. They are examples of glass-box interpretable-by-design models that can be explored via analysis of coefficients provided by the package. Find an example at `http://xai-tools.drwhy.ai/glmnet.html`.

The R package **naivebayes** (Majka, 2019) is a tool dedicated to fitting Naive Bayes predictive models. It provides various types of support for different distributions. It also implements plots that allow users to inspect the model itself. Find an example at `http://xai-tools.drwhy.ai/naivebayes.html`.

The R package **iBreakDown** (Gosiewska and Biecek, 2019b) provides methods for local model explanations. It allows us to compute and visualize the additive and interaction BreakDown of a single observation. It also provides an interface for SHAP. Find an example at `http://xai-tools.drwhy.ai/iBreakDown.html`.

The R package **ICEbox** (Goldstein et al., 2015) allows user to create ICE curves across the dataset. It also aggregates them creating Partial Dependence Curves or Partial Derivative Curves, and provides an interface for clustering ICE curves. Find an example at `http://xai-tools.drwhy.ai/ICEbox.html`.

The R package **iml** (Molnar, 2019) implements a wide range of global and local model-agnostic explanation methods, such as feature importance, PDP, ALE, ICE, surrogate models, LIME, and SHAP. The **iml** library is characterized by the use of R6 classes. Examples of PDP and SHAP explanations can be seen in Figure 6 and Figure 7. Find an example at `http://xai-tools.drwhy.ai/iml.html`.

The R package **ingredients** (Biecek et al., 2019) implements techniques for both local and global explanations. It facilitates the computation of permutational variable importance, Accumulated, Partial, and Conditional dependence profiles. The package provides a tool for computing Ceteris

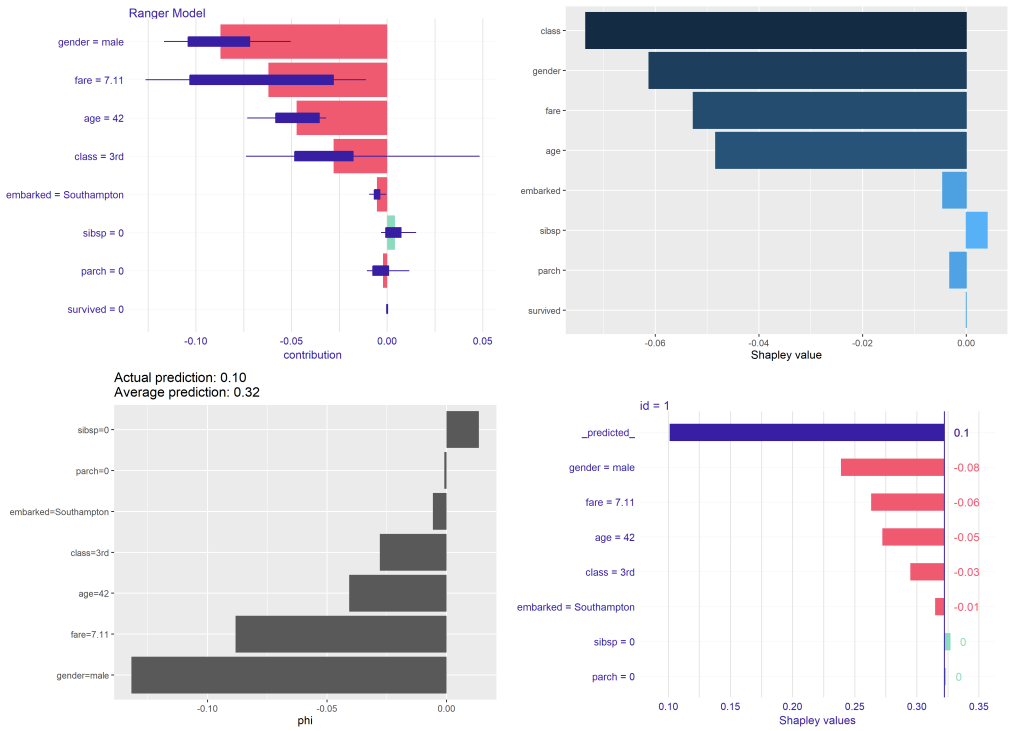


Figure 7: Contribution plots based on Shapley values for the same observation from *titanic* dataset generated with **DALEX** (top-left), **fastshap** (top-right), **iml** (bottom-left), **shapper** (bottom-right). The computation times of the plots are 41.08s for **DALEX**, 107.77s for **fastshap**, 0.39s for **iml**, 0.02s for **shapper**.

Paribus (ICE) curves, clustering them, and calculating oscillations in order to explain a single variable. Find an example at `http://xai-tools.drwhy.ai/ingredients.html`.

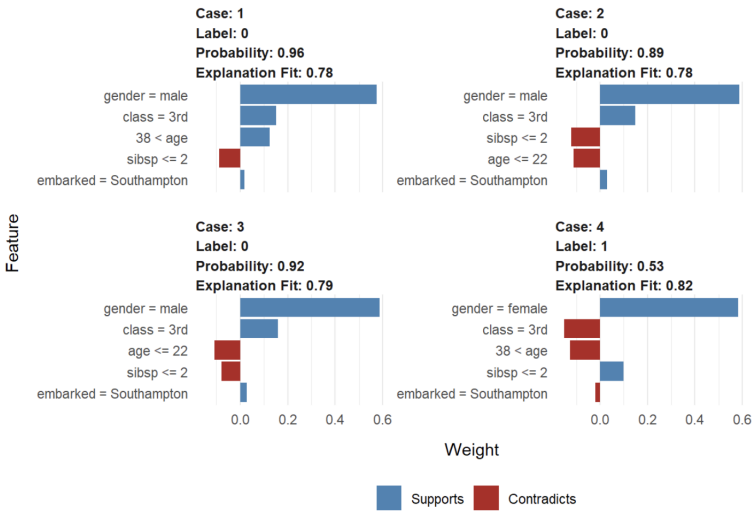


Figure 8: Snapshot of the explanations created with the **lime** package.

An R package **interpret** (Nori et al., 2019) is a tool for training Explainable Boosting Machine models (Caruana et al., 2015) that are high-performance generalized additive models with pairwise interactions. The Python version (library **interpret**) implements more interpretable models, which are decision trees, linear regression, and logistic regression. It also provides XAI methods, such as SHAP, Tree SHAP, LIME, Morris Sensitivity Analysis, and PDP.

The R package **kknn** (Schliep and Hechenbichler, 2016) provides an extended interface for training classifiers based on the k-Nearest Neighbors method. The package provides ways to inspect the nearest neighbors (the most similar observations). Find an example at `https://mi2datalab.github.io/XAIL-`

tools/kknn.html.

The Python library **lime** is an implementation of the LIME (Ribeiro et al., 2016) technique by the authors of this method. The library supports text, image, and tabular data explanations.

The R package **lime** (Pedersen and Benesty, 2019) is an implementation that is independent of the authors of the original Python library. An R tool supports a wide range of frameworks, e.g. caret, parsnip, and mlr. See an example in Figure 8. Find an example at http://xai-tools.drwhy.ai/lime.html.

The R package **live** (Staniak and Biecek, 2018) provides local, interpretable, and model-agnostic visual explanations. The idea behind LIVE is similar to LIME, the difference is the definition of the surroundings of an observation. A neighborhood of the observation of interest is simulated by perturbing one variable at a time. Therefore, numerical variables are used in the interpretable local model and are not discretized as in the basic LIME. Find an example at http://xai-tools.drwhy.ai/live.html.

The R package **mcr** (Manuilova et al., 2014) is a tool to compare two measurement methods using regression analysis. The package contains functions for summarizing and plotting results. The *mcr* provide comparisons of models, yet is limited only to the regression models. Find an example at http://xai-tools.drwhy.ai/mcr.html.

The R package **modelDown** (Romaszko et al., 2019) generates a website with HTML summaries for predictive models. The generated website provides information about model performance, variable response (PDP, ALE), and the importance of variables, and concept drift. The **modelDown** also provides a comparison of models. Additionally, data available on the website can be downloaded and recreated in the R session. Find an example at http://xai-tools.drwhy.ai/modelDown.html.

The R package **modelStudio** (Baniecki and Biecek, 2019) generates interactive and animated model explanations in the form of a serverless HTML site. modelStudio provides various explanations, such as SHAP, Ceteris Paribus, permutational variable importance, PDP, ALE, and plots for exploratory data analysis. The package can be easily integrated with the *scikit-learn* and *lightgbm* Python libraries. An example screenshot of the HTML site with explanations is presented in Figure 9. Find an example at http://xai-tools.drwhy.ai/modelStudio.html.

Figure 9: Snapshot of the HTML site created by **modelStudio**.

The R package **party** (Hothorn et al., 2006) is a tool dedicated for recursive partitioning. The core of the package is the implementation of the decision tree glass-box model. Decision paths can be plotted using it, along with the split rules. The visualization also covers the distribution of values in terminal nodes. Find an example at http://xai-tools.drwhy.ai/party.html.

The R package **pdp** (Greenwell, 2017) is a tool for computing PDP and ICE. This library works

with any predictive model. An example of the PDP explanation can be seen in Figure 6. Find an example at http://xai-tools.drwhy.ai/pdp.html.

The R package **randomForestExplainer** (Paluszynska et al., 2020) is a set of tools for explaining random forests. The existing explanations show variable importance, distribution of minimal depth for each variable, variable interactions, and prediction plot for two variables. What is more, a package provides an option to generate all explanations as an HTML report. Find an example at http://xai-tools.drwhy.ai/randomForestExplainer.html.

The Python library **shap** is an implementation of the Shapley-based explanations technique (SHAP) (Lundberg and Lee, 2017) provided by its authors. SHAP explanations are supported by many visualizations. The library provides also a high-speed algorithm for tree-based models and SHAP-based variable importance and detection of variable interactions.

The R package **shapper** (Maksymiuk et al., 2019) is an interface for Python library **shap**. A package implements new plots, different than in the Python version. The **shapper** provides plotting explanations for multiple models together. An example of the SHAP explanation can be seen in Figure 7. Find an example at http://xai-tools.drwhy.ai/shapper.html.

The Python library **Skater** (Oracle and contributors, 2020) is a tool for global and explanations. The implemented features are: variable importance, partial dependence profiles, LIME, Scalable Bayesian Rule Lists, Tree Surrogates, and two methods for deep neural networks, i.e. Layer-wise Relevance Propagation (e-LRP), and Integrated Gradient.

The R package **smbinning** (Jopia, 2019) provides tools to organize the end-to-end development process of building a scoring model. The library covers data exploration, variable selection, feature engineering, binning, and model selection. Find an example at http://xai-tools.drwhy.ai/smbinning.html.

The R package **survxai** (Grudziaz et al., 2018) is, to the best of our knowledge, the only tool for model-agnostic explanations of survival analysis models. The package implements local and global explanations, and it also provides comparisons of models. Find an example at http://xai-tools.drwhy.ai/survxai.html.

The R package **vip** (Greenwell et al., 2020) provides many ways to calculate variable importance and interaction strength measures. There is model-based variable importance for models such as random forest, gradient boosted decision trees and multivariate adaptive regression splines. The **vip** package also provides three model-agnostic variable importance measures: permutation-based, Shapley-based, and variance-based. Find an example at http://xai-tools.drwhy.ai/vip.html.

The R package **vivo** (Kozak and Biecek, 2020) is an implementation of variable importance measures. Global importance is based on oscillations of partial dependence profiles while local importance is based on oscillations of Ceteris Paribus profiles. Find an example at http://xai-tools.drwhy.ai/vivo.html.