DALEX: Descriptive mAchine Learning Explanations

Tools for exploration, validation and explanation of complex machine learning models

Mateusz Staniak, Wrocław 2.07.2018

DALEX invasion!

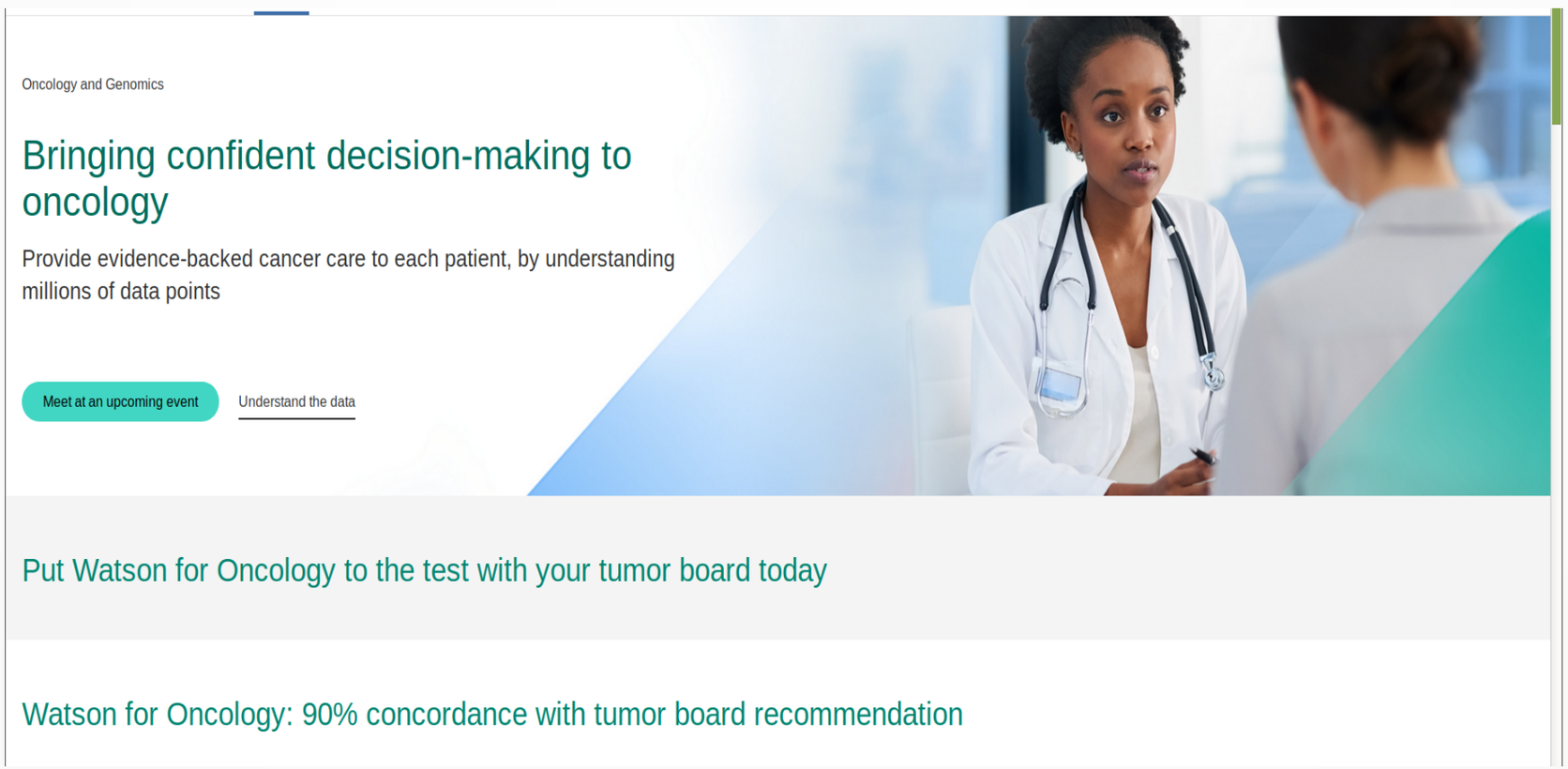
- SER meeting in Warsaw, April 2018
- eRum conference workshop, May 2018
- STWUR meeting in Wrocław, June 2018
- Why R? 2018 conference in Wrocław, July 2018
- useR! Conference in Brisbane, July 2018
- T.B.C...

Materials for the workshop

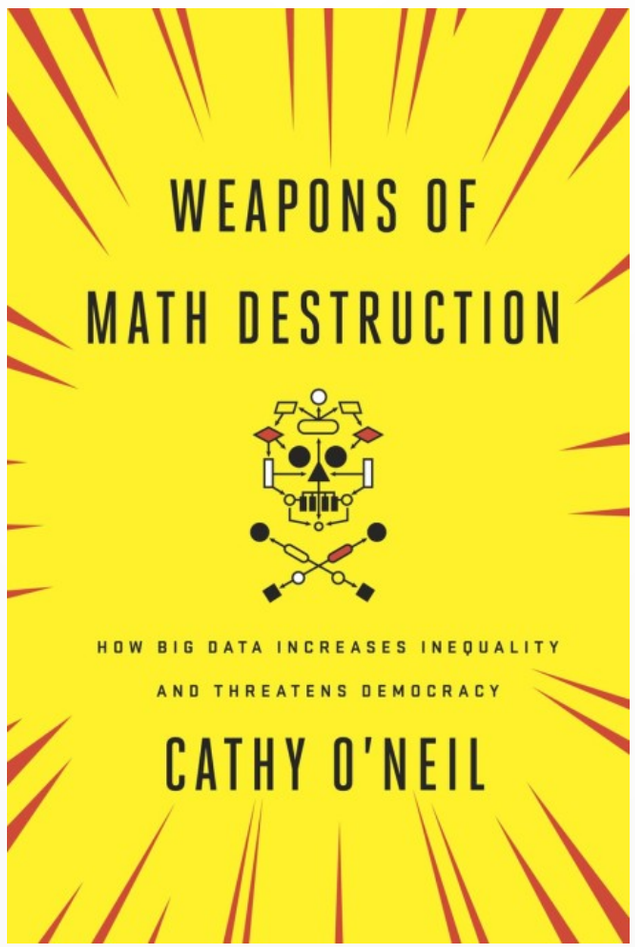
https://bit.ly/2KngMch

https://github.com/pbiecek/DALEX_docs/tree/master/workshops/WhyR2018

Why explain predictive models?



Why explain predictive models? cont.



Why explain predictive models? cont.

Overview Data Kernels Discussion Leaderboard Rules

Public Leaderboard Private Leaderboard

This leaderboard is calculated with approximately 29% of the test data.
The final results will be based on the other 71%, so the final standings may be different.

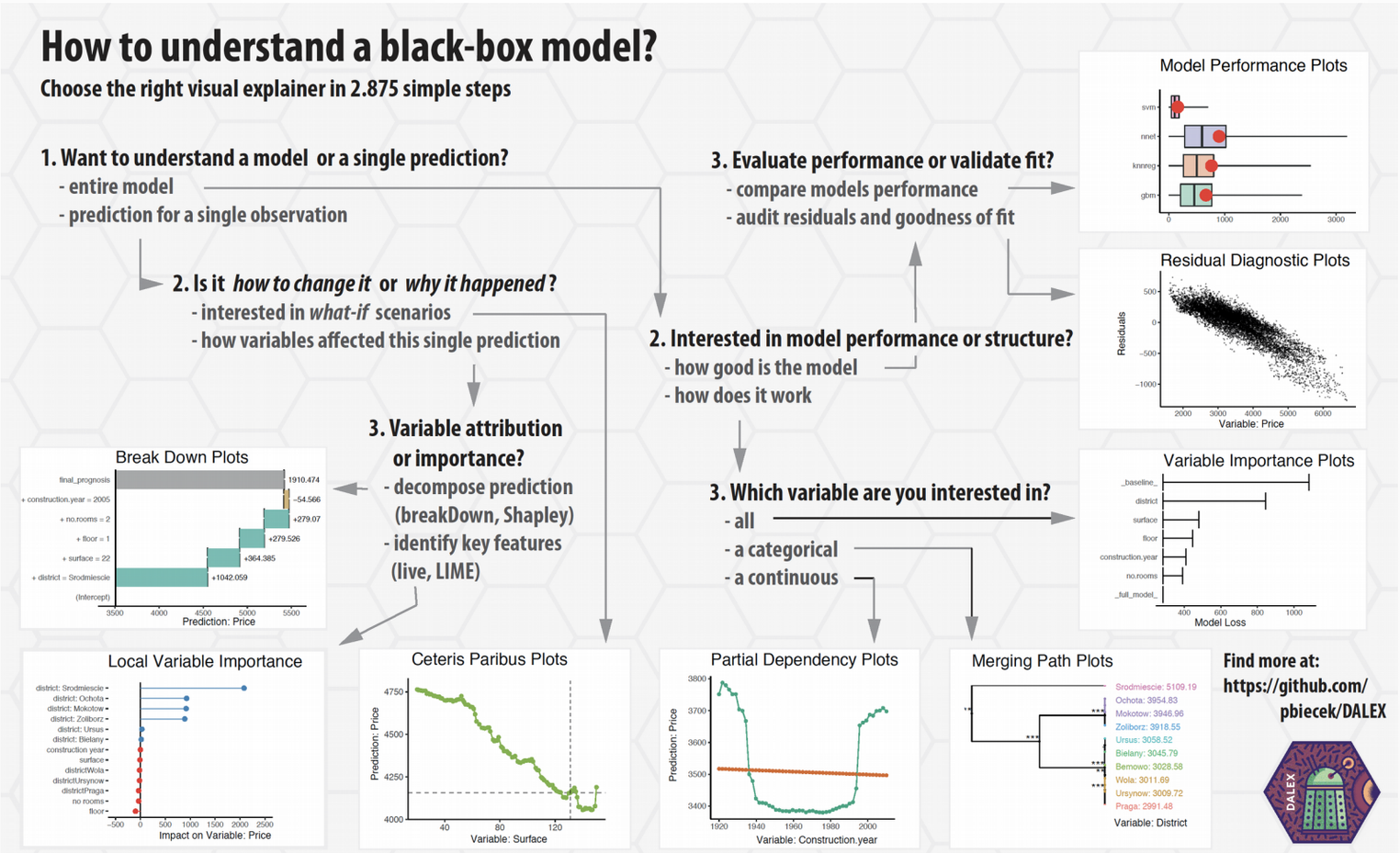
Raw Data Refresh

In the money Gold Silver Bronze

#	△1w	Team Name	Kernel	Team Members	Score	Entries	Last
1	–	Mickey			0.6405	7	9d
2	▲257	Zidmie			0.5829	5	3d
3	▼1	Vicens Gaitan			0.4969	2	6d
4	▲1	Lin12345			0.4775	16	20h
5	▲5	HomerJSimpson			0.4306	11	11h
6	▼3	Grzegorz Sionkowski			0.4225	12	1d
7	▲28	Andrea			0.4219	4	2d
8	new	Vivek			0.4207	4	1d
9	new	dohlee			0.4107	1	1d
10	▲129	all_random			0.4063	13	18h

Why explain predictive models? cont.

Machine learning explanations: road map



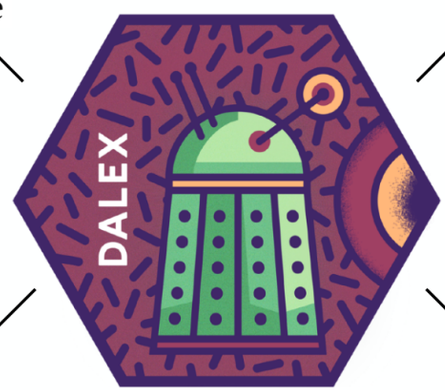
http://smarterpoland.pl/index.php/2018/06/not-only-lime/

DALEX: overview of the ecosystem

DALEX is a set of tools that
helps to understand the way complex predictive models work

How good is the predictive model?

Which variables are the most important in general?

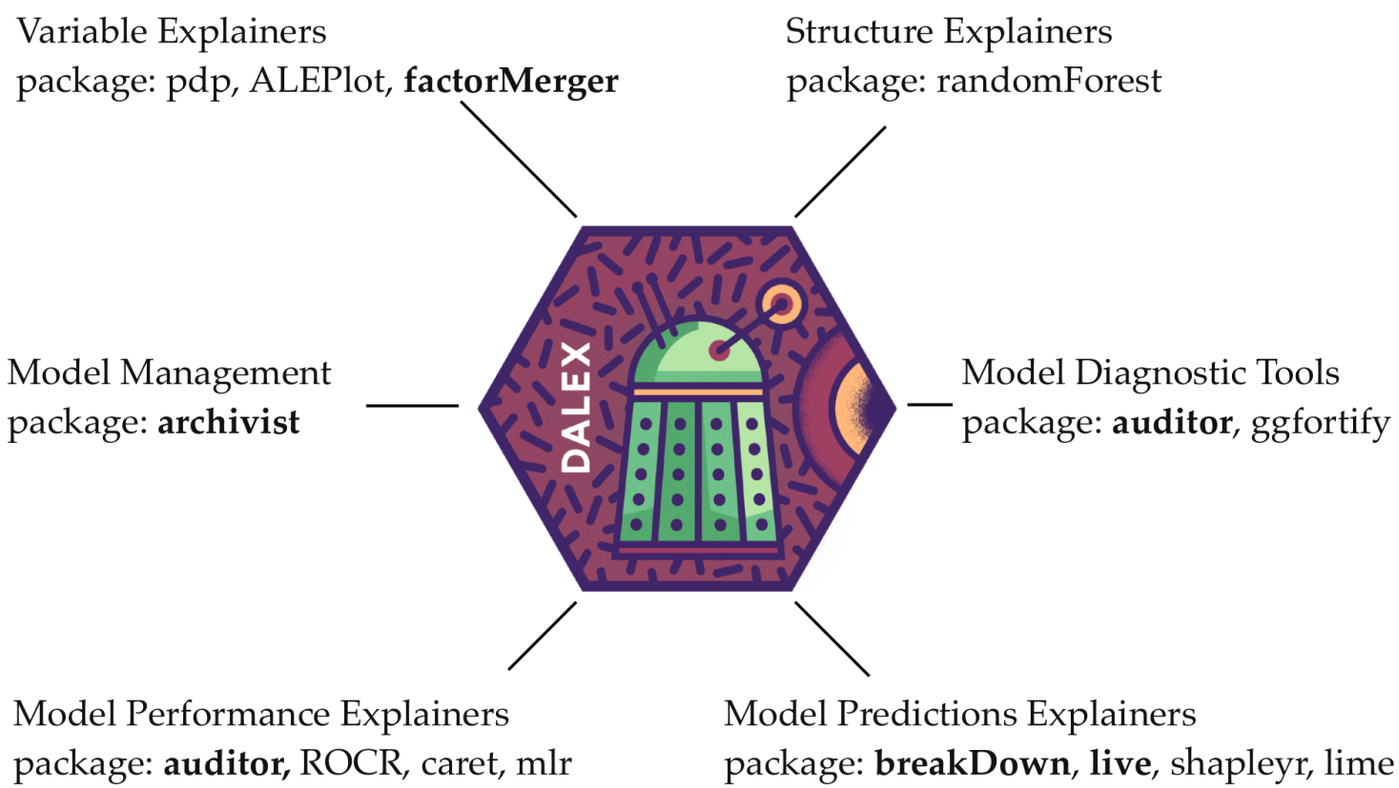


How good is the model fit?

Which variables influence the single prediction?

https://github.com/pbiecek/DALEX_docs/blob/master/workshops/eRum2018/Workshop_eRum_2018_part1.pdf

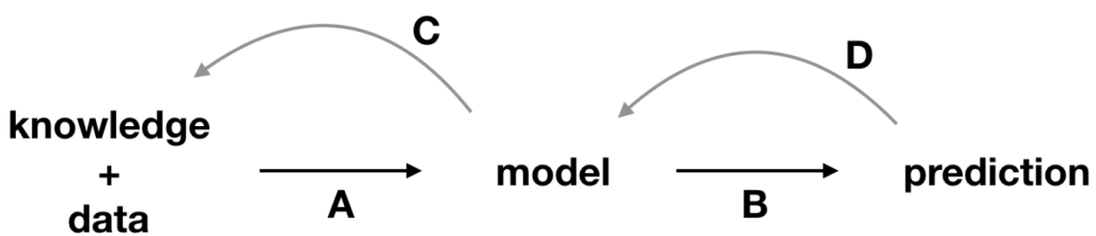
DALEX: overview of the ecosystem



https://github.com/pbiecek/DALEX_docs/blob/master/workshops/eRum2018/Workshop_eRum_2018_part1.pdf

DALEX: overview of the ecosystem

Typical workflow in ML



A. Modelling is a process in which domain knowledge and data are turned into models.
B. Models are used to generate predictions.
C. Understanding of model structure may increase our knowledge and in consequence leads to a better model. *DALEX helps here.*
D. Understanding of drivers behind particular model predictions may help to correct wrong decisions and in consequence leads to a better model. *DALEX helps here.*

https://github.com/pbiecek/DALEX_docs/blob/master/workshops/eRum2018/Workshop_eRum_2018_part1.pdf

DALEX: overview of the ecosystem

DALEX

CRAN 0.2.2 | downloads 2014 | build passing | coverage 92%

Descriptive mAchine Learning EXplanations

DALEX Stories

- A gentle Introduction to DALEX with examples
- How to use DALEX with caret
- How to use DALEX with mlr
- How to use DALEX with xgboost package
- Talk about DALEX at Complexity Institute / NTU February 2018
- Talk about DALEX at SER / WTU April 2018
- How to use DALEX for teaching. Part 1

Install

From GitHub

```
# dependencies
devtools::install_github("MI2DataLab/factorMerger")
devtools::install_github("pbiecek/breakDown")

# DALEX package
devtools::install_github("pbiecek/DALEX")
```

or from CRAN

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

Explaining single prediction

Why explain a single prediction?

European Union regulations on algorithmic decision-making and a "right to explanation"

Bryce Goodman,[1*] Seth Flaxman,[2]

[1]Oxford Internet Institute, Oxford
1 St Giles', Oxford OX1 3LB, United Kingdom
[2]Department of Statistics, University of Oxford,
24-29 St Giles', Oxford OX1 3LB, United Kingdom

*To whom correspondence should be addressed; E-mail: flaxman@stats.ox.ac.uk.

Abstract

We summarize the potential impact that the European Union's new General Data Protection Regulation will have on the routine use of machine learning algorithms. Slated to take effect as law across the EU in 2018, it will restrict automated individual decision-making (that is, algorithms that make decisions based on user-level predictors) which "significantly affect" users. The law will also effectively create a "right to explanation," whereby a user can ask for an explanation of an algorithmic decision that was made about them. We argue that while this law will pose large challenges for industry, it highlights opportunities for computer scientists to take the lead in designing algorithms and evaluation frameworks which avoid discrimination and enable explanation.

1 Introduction

In April 2016, for the first time in over two decades, the European Parliament adopted a set of comprehensive regulations for the collection, storage and use of personal information, the General Data Protection Regulation (GDPR)[1] [26]. The new regulation has been described as a "Copernican Revolution" in data protection law, "seeking to shift its focus away from paper-based, bureaucratic requirements and towards compliance in practice, harmonization of the law, and individual empowerment" [22]. Much of the regulations are clearly aimed at perceived gaps and inconsistencies in the EU's current approach to data protection. This includes, for example, the codification of the "right to be forgotten" (Article 17), and regulations for foreign companies collecting data from European citizens (Article 44).

However, while the bulk of language deals with how data is collected and stored, the regulation contains *Article 22: Automated individual decision-making, including profiling* (see figure 1) potentially prohibiting a wide swath of algorithms currently in use in, e.g. recommendation systems, credit and insurance risk assessments, computational advertising, and social networks. This raises important issues that are of particular concern to the machine learning community. In its current form, the GDPR's requirements could require a complete overhaul of standard and widely used algorithmic techniques. The GDPR's policy on the right of citizens to receive an explanation for algorithmic decisions highlights the pressing importance of human interpretability in algorithm

https://arxiv.org/abs/1606.08813

Why explain a single prediction?

Understanding a single prediction:

- might help discover problems in the model (validation)
- could help compare models and choose better one
- could help improve the model
- is the most important part from consumer's perspective

How it's (usually) done

Two main ideas:

- (additive) decomposition of the prediction (Shapley values, **Break Down Plots**)
- Local approximation by a simple model (LIME, **live**)

breakDown: overview

CRAN 0.1.5 | downloads 636/month | downloads 3050 | build passing Pending Pull-Requests Github Issues

Break Down Plots: Model Agnostic Explainers for Individual Predictions

`breakDown` package decomposes individual predictions into parts attributed to particular variables. See vignettes for more examples.

Bookdown website for `breakDown` package: https://pbiecek.github.io/breakDown/

Methodology behind prediction explainers implemented in `breakDown` and `live` : https://arxiv.org/abs/1804.01955

How to install

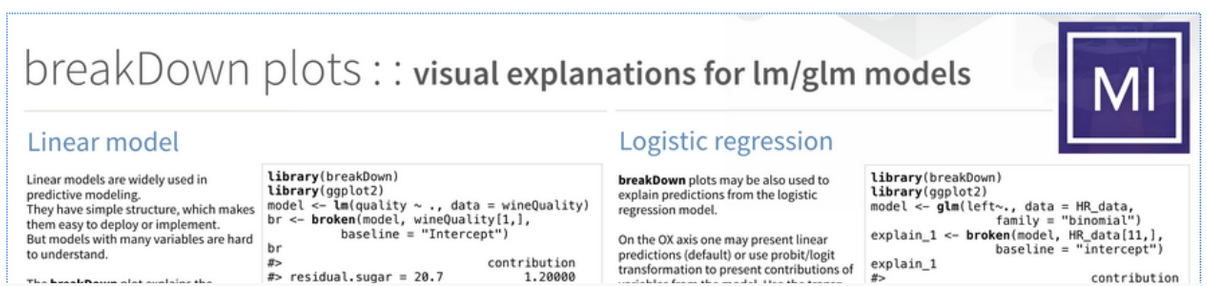
Install from GitHub

```
devtools::install_github("pbiecek/breakDown")
```

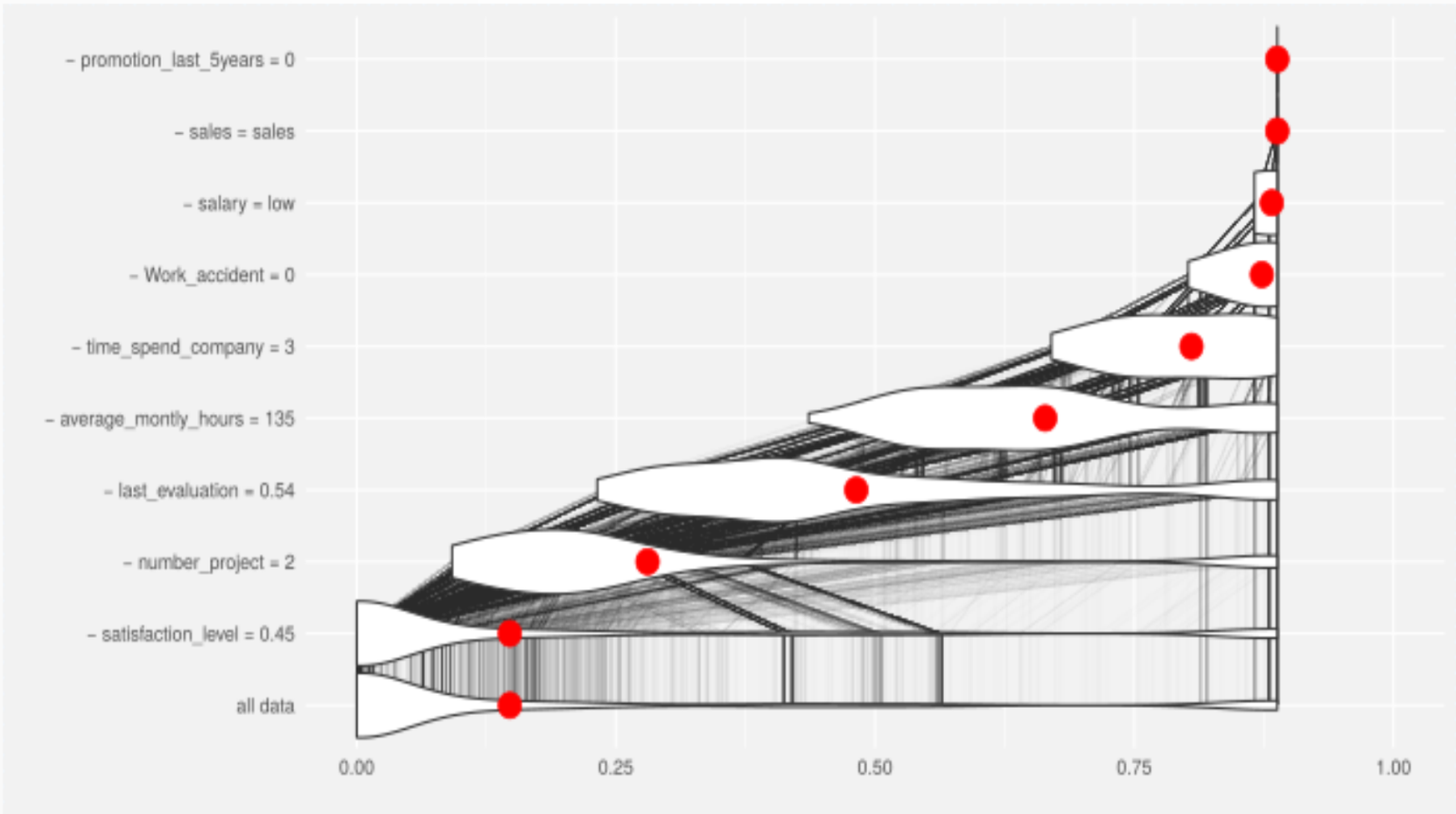
Install from CRAN

```
install.packages("breakDown")
```

Cheatsheets



Prediction Break Down: idea



Prediction Break Down: algorithm

Algorithm 2 Model agnostic break down of model predictions. The *step-down* approach.

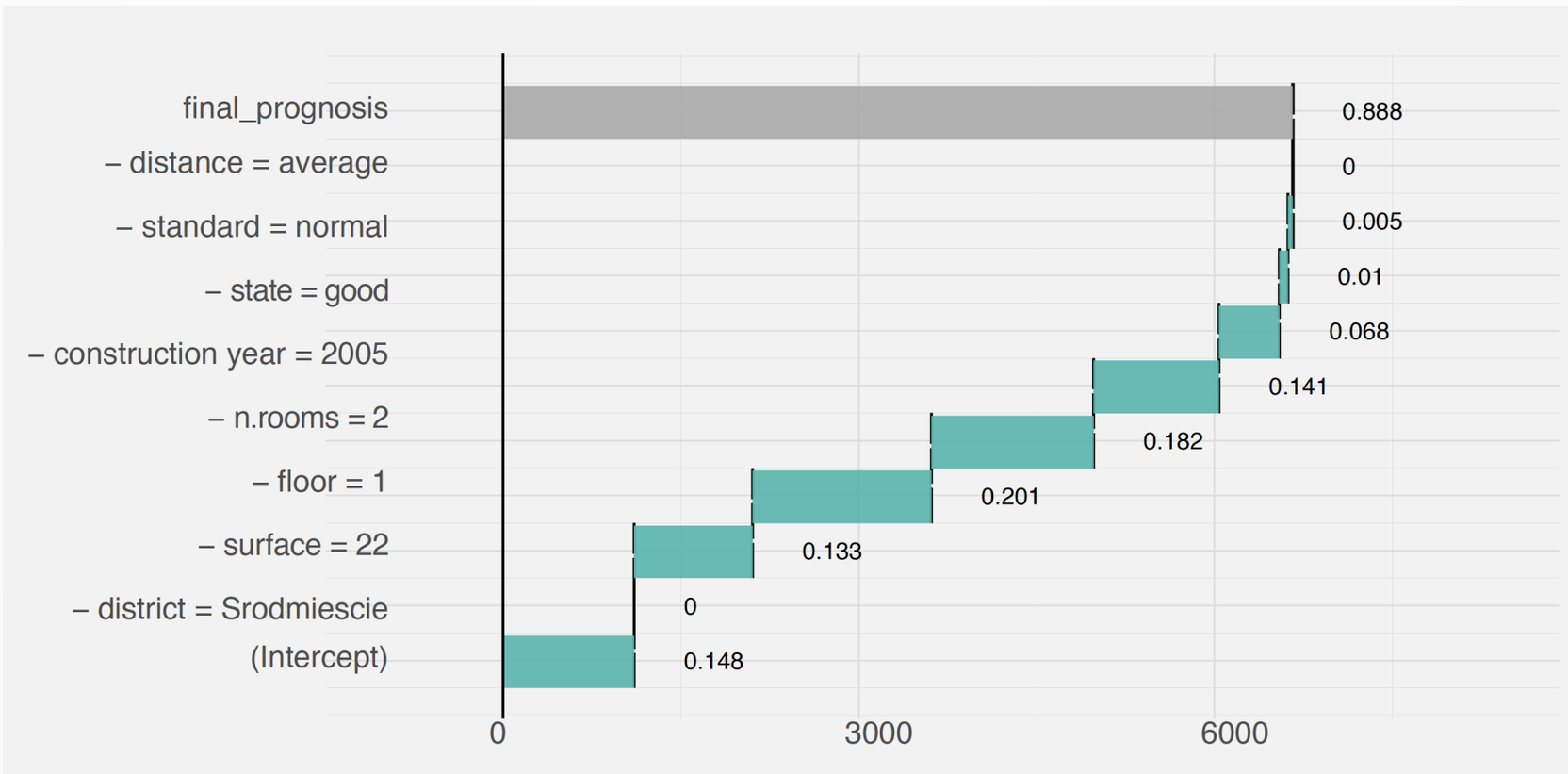
$p \leftarrow$ number of variables
$IndSet \leftarrow \{1, \ldots, p\}$ set of indexes of all variables
for i in $\{1, \ldots, p\}$ **do**
    Find new variable that can be relaxed with small loss in relaxed distance to $f(x^{new})$
    for j in $IndSet$ **do**
        Calculate relaxed distance with j removed
        $dist(j) \leftarrow d(x^{new}, IndSet \setminus \{j\})$
    end for
    Find and remove j that minimizes loss
    $j_{min} \leftarrow \arg\min_j dist(j)$
    $Contribution^{IndSet}(i) \leftarrow f^{IndSet}(x^{new}) - f^{IndSet \setminus \{j_{min}\}}(x^{new})$
    $Variables(i) \leftarrow j_{min}$
    $IndSet \leftarrow IndSet \setminus \{j_{min}\}$
end for

Algorithm 3 Model agnostic break down of model predictions. The *step-up* approach.

$p \leftarrow$ number of variables
$IndSet \leftarrow \emptyset$ empty set
for i in $\{1, \ldots, p\}$ **do**
    Find new variable that can be relaxed with large distance to $f^{\emptyset}(x^{new})$
    for j in $\{1, \ldots, p\} \setminus IndSet$ **do**
        Calculate relaxed distance with j added
        $dist(j) \leftarrow d(x^{new}, IndSet \cup \{j\})$
    end for
    Find and add j that maximize distance
    $j_{max} \leftarrow \arg\max_j dist(j)$
    $Contribution^{IndSet}(i) \leftarrow f^{IndSet \cup \{j_{max}\}}(x^{new}) - f^{IndSet}(x^{new})$
    $Variables(i) \leftarrow j_{max}$
    $IndSet \leftarrow IndSet \cup \{j_{max}\}$
end for

https://arxiv.org/abs/1804.01955

Prediction Break Down: usage



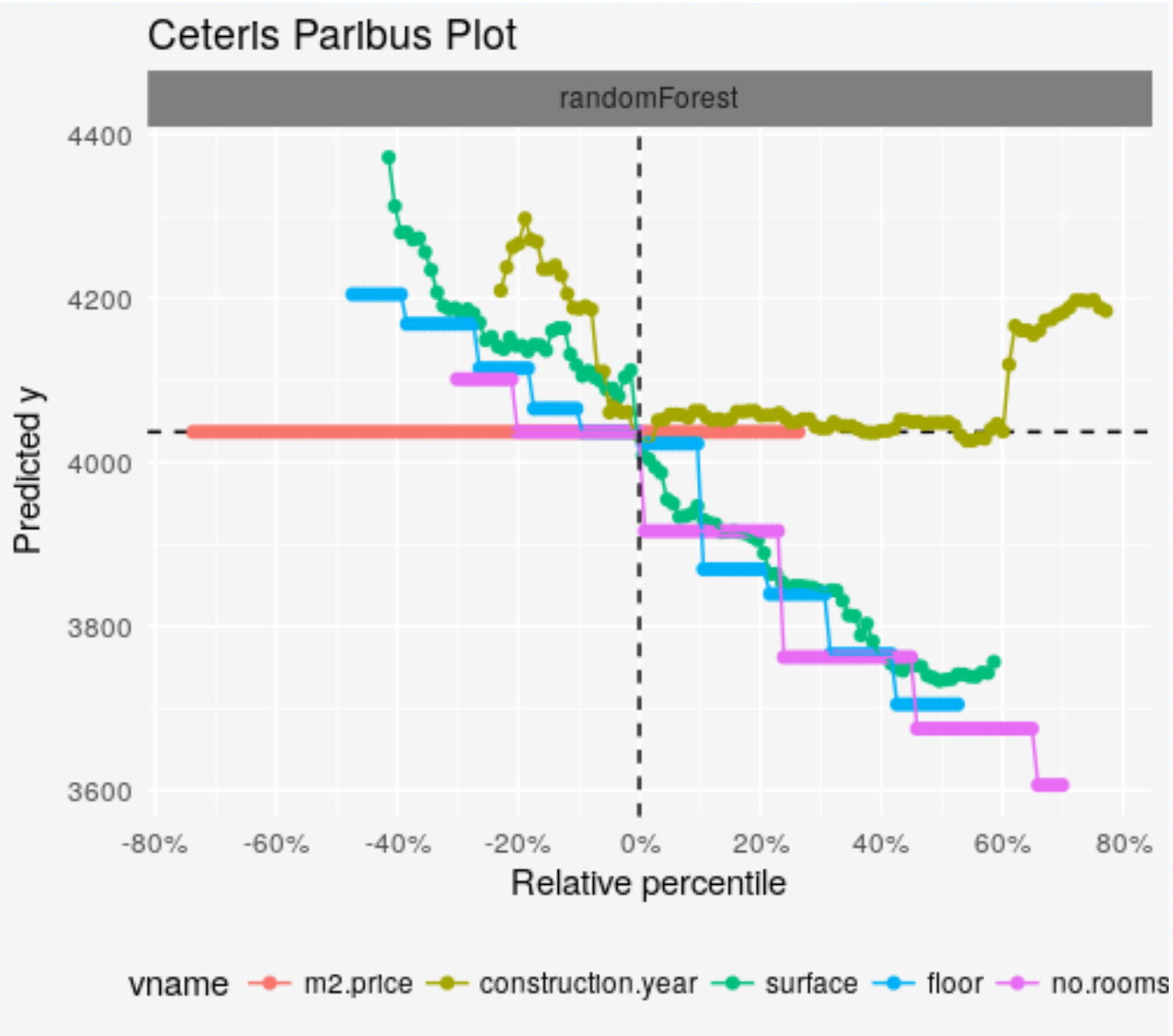
Prediction Break Down: exercises

- Check that you have all the libraries installed
- Fit random forest model and linear regression to apartments data.
- Create DALEX explainers for both models.
- Use prediction_breakdown function for linear model to explain 10th prediction
- Use prediction_breakdown function for random forest to explain 10th prediction
- Compare the results
- Do the same for another observation and compare results

Ceteris Paribus plots: idea

- „What can I do to change the verdict?”
- Ceteris Paribus (What-If) plots show predicted response in different possible scenarios
- We fix values of all but one predictor and plot changes in model response while this one predictor varies

Ceteris Paribus plots: usage



Ceteris Paribus plots: exercises

- Fit another non-linear models to the apartments data and create DALEX explainer.
- Create Ceteris Paribus plot for your model and compare it to the plot for random forest.

live: overview

live: Local Interpretable (Model-agnostic) Visual Explanations

Installation

Install stable CRAN version:

```
install.packages("live")
```

or the development version:

```
devtools::install_github("MI2DataLab/live")
```

See the latest changes.

Features coming up next:

- more methods of sampling,
- better support for comparing explanations for different models / different instances,
- improved Shiny application (see `live_shiny` function in development version).

If you have any bug reports, feature requests or ideas to improve the methodology, feel free to leave an issue.

Materials

Find the paper about `live` and breakDown on arXiv.

Website: https://mi2datalab.github.io/live/

Conference talk on `live`: https://github.com/mstaniak/Berlin_2017

Cheatsheet:

Local explanations: how it's done

- Original idea: LIME
- We simulate a *fake* dataset of observations similar to observation which we explain
- We add black box predictions for observation in the *fake* dataset
- We fit a simple (interpretable) model to these predictions
- We analyze and visualize the simple model

live vs LIME

- Different methods of sampling available
- Focus on model visualization
- More flexibility
- Designed for tabular data

live: advanced options

- Variable selection
- Setting some variables constant
- Changing the method of sampling
- Using different explanation models

live: exercises

- Create a simulated dataset for your model.
- Add predictions to this dataset.
- Fit linear regression model to the predictions.
- Create forest plot and prediction break down plot for the explanation model.

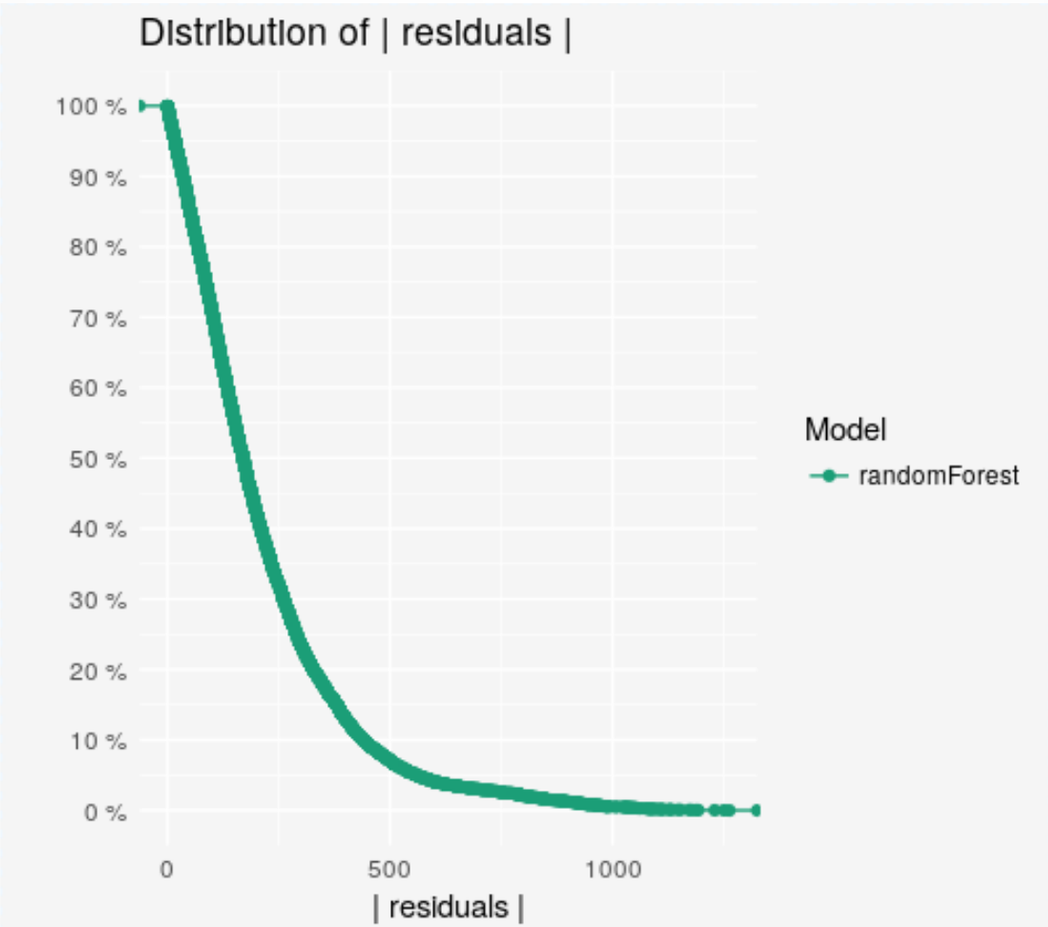
Let's take a short break!

Global explanations

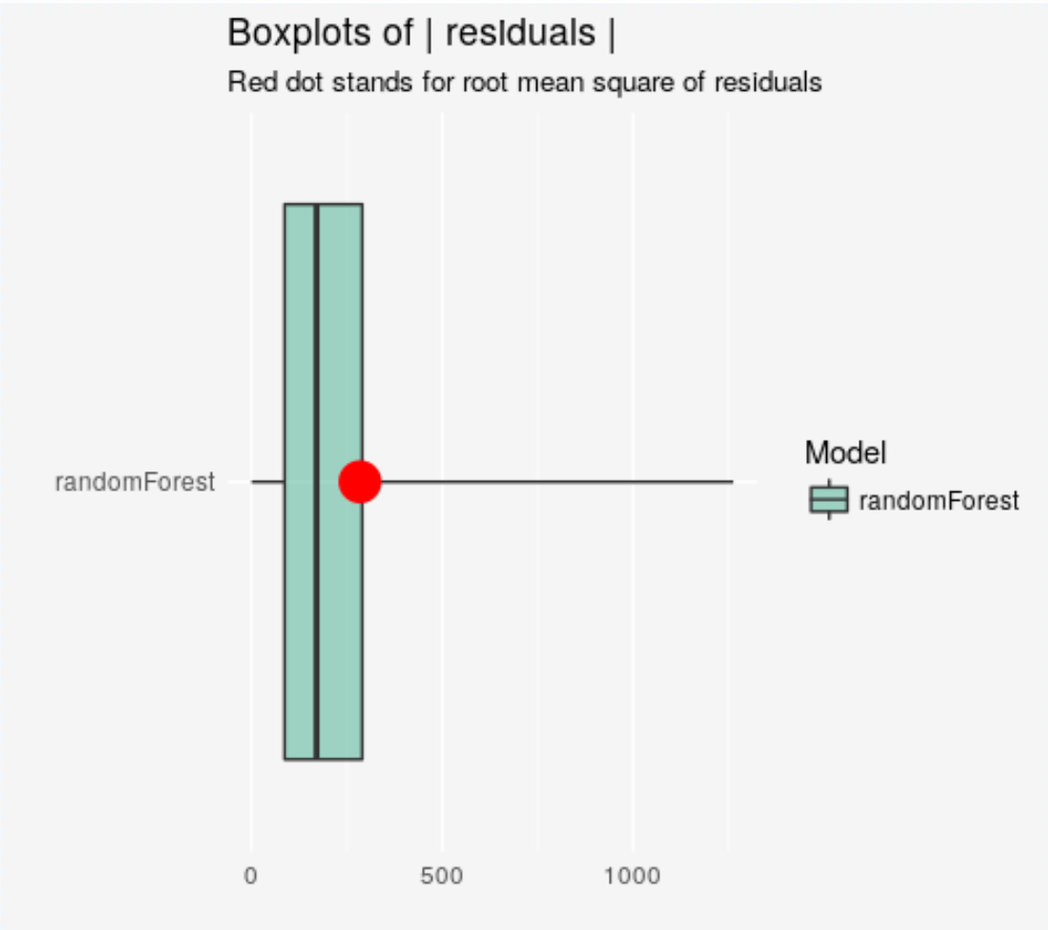
model_performance: why?

- Usually, we judge a model based on a single metric (for example RMSE)
- Single number is not enough!
- Proof → exercises

model_performance: usage 1



model_performance: usage 2



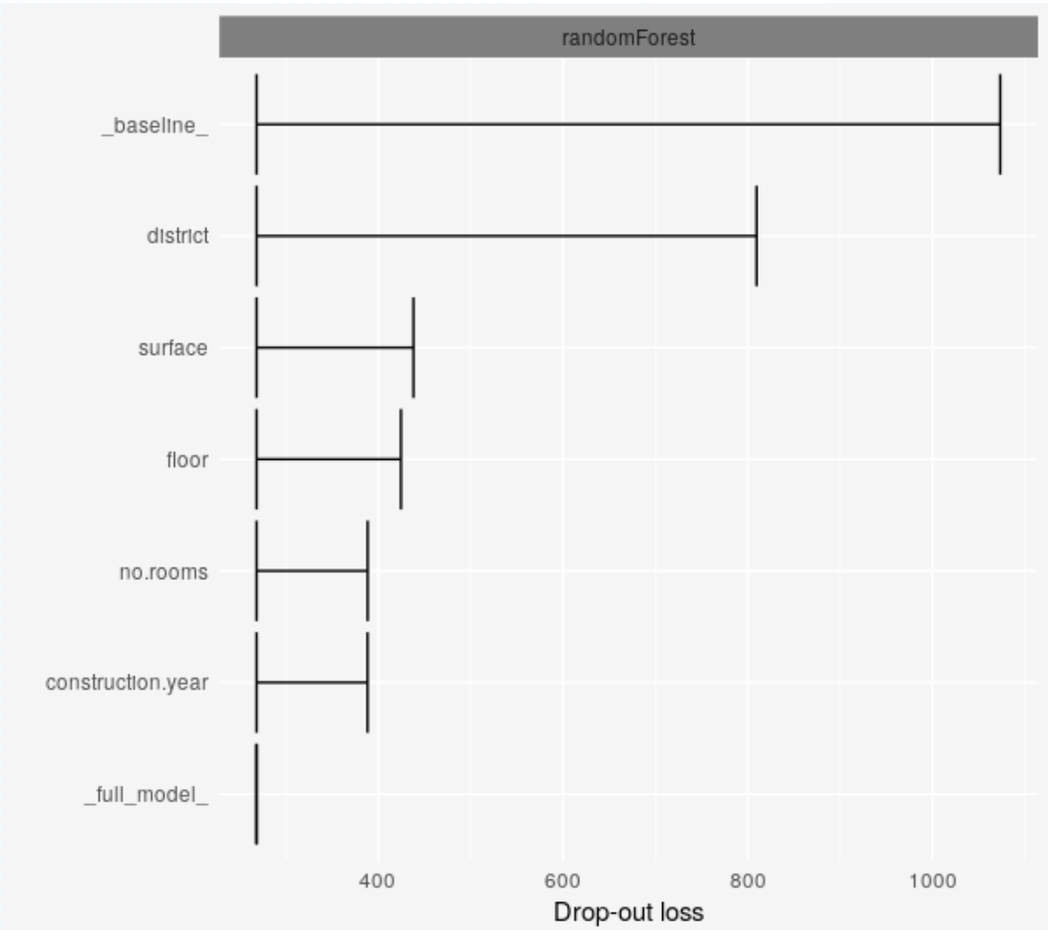
Model performance: exercises

- Fit linear regression model and a model of your choice (for example: SVM, K-NN, GBM) to apartments data.
- Create performance explainers for your models
- Plot CFD and boxplots for these explainers
- Compare them to plots for random forest

Variable importance: why & how?

- We want to know which variable have the strongest influence on the predictions
- We choose a **loss function**
- We assume that if a variable is important, removing it from the model will result in a large increase in the loss
- Variables are „removed” via random permutations

variable_importance: usage



variable_importance: exercises

- Calculate variable importance for your models.
- Compare these result to variable importance for random Forest.

Bonus:

- Compare the result to local variable importance calculated earlier.

single_variable: why?

- In simpler models relationship between predictors and response in either assumed (linear regression) or easy to visualize (GAM)
- In non-parametric setting, it is hard to understand this relationship
- Despite this, we want to see how a single variable influences the predictions

single_variable: methods

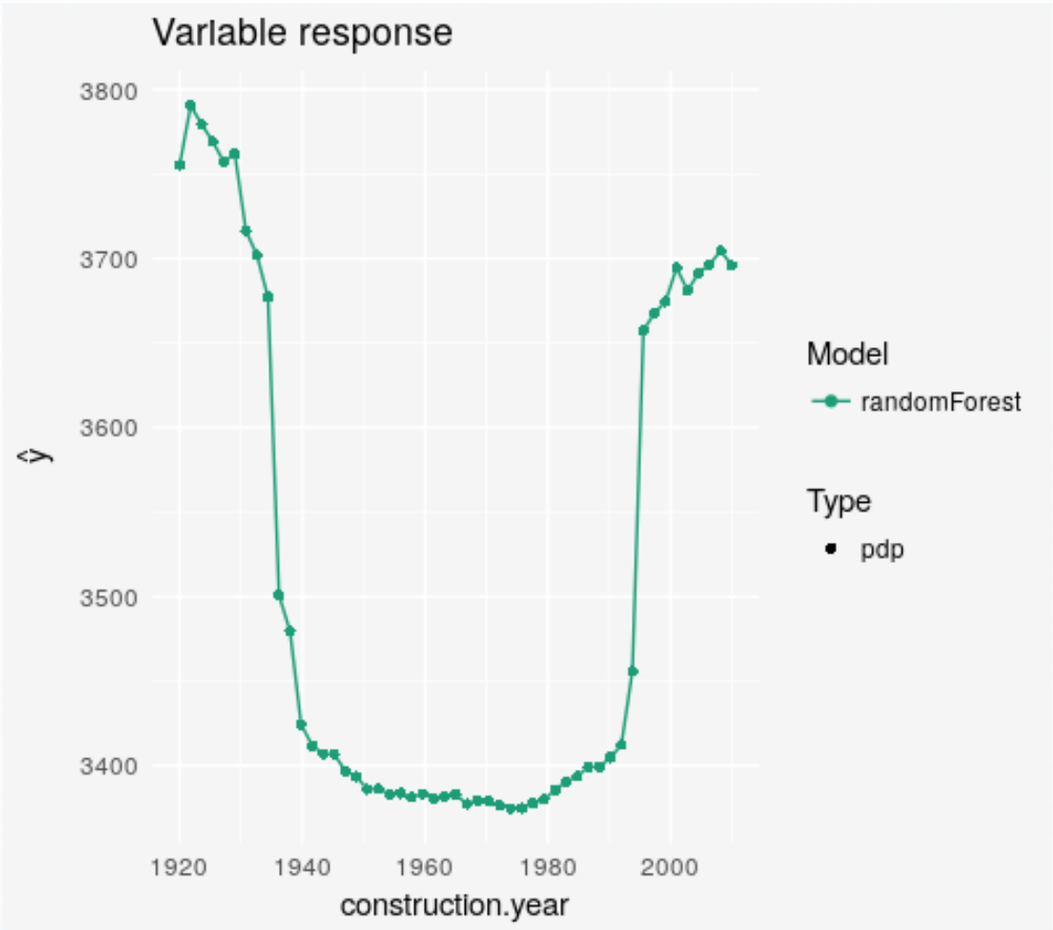
- Partial Dependence Plots (PDP)
- Individual Conditional Expectation (ICE)
- Accumulated Local Effects Plots (ALE Plots)
- Merging Path Plots

single_variable: methods cont.

- Basic idea: Partial Dependence Plots (Friedman)
- We want to see, how predictions depend on a single variable (or a group of variables)
- We ignore the effects of other variables
- We calculate predictions for a range of values of chosen variable (or a group of variables)

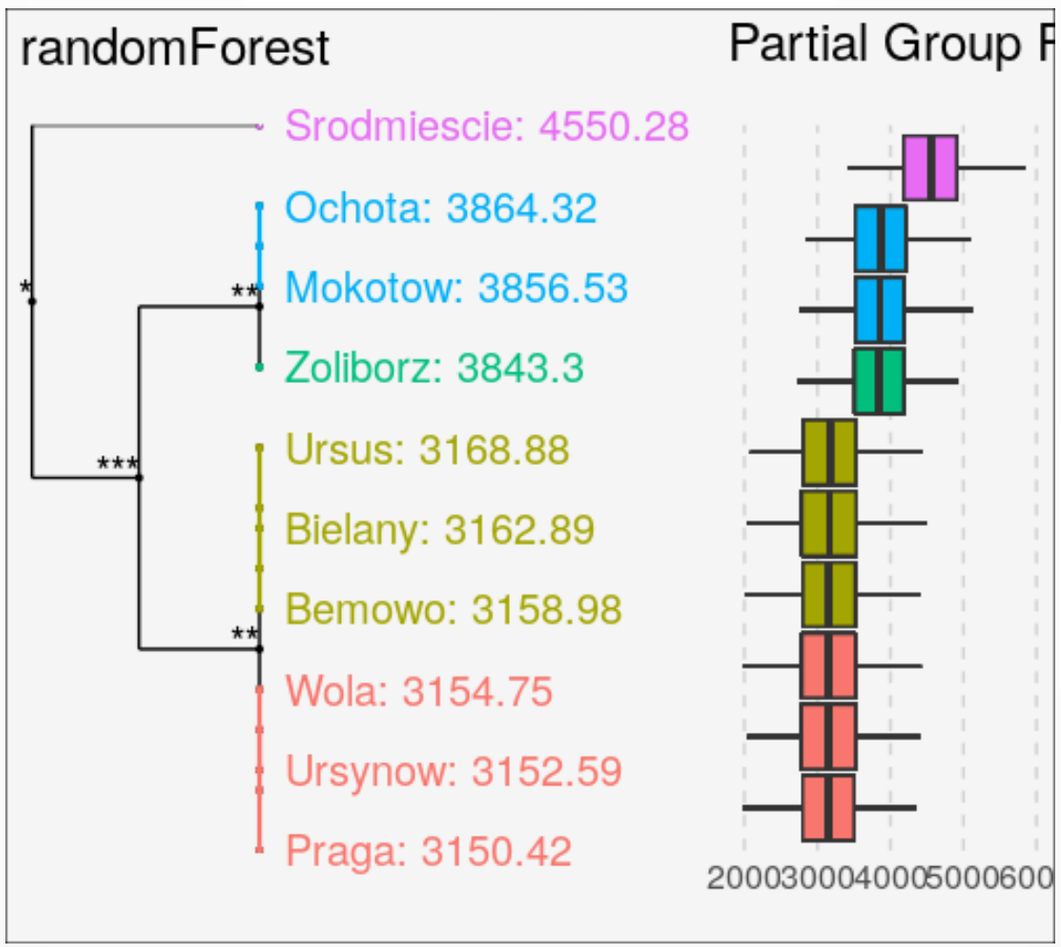
$$\phi_j(x) = \frac{1}{N} \sum_{k=1}^{N} F(x_{1,k}, \ldots, x_{j-1,k}, x, x_{j+1,k}, \ldots, x_{p,k}).$$

single_variable: usage 1



Partial Dependence Plot

single_variable: usage 2



Merging Paths Plot

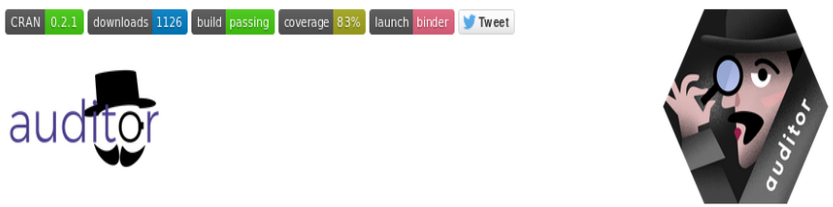
single_variable: exercises

- Create single variable explainers for construction.year and district in your models.
- Compare the resulting plots to the plots for random forest

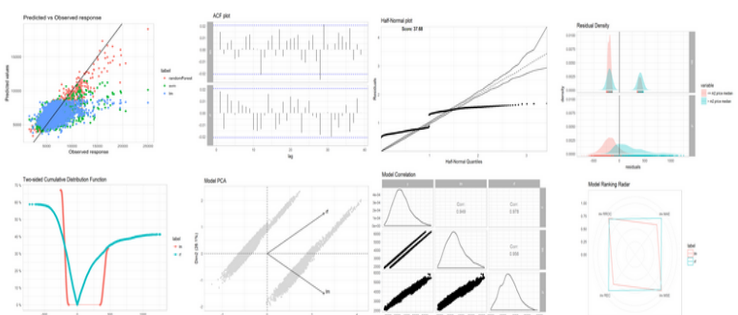
Tools for validation and management

auditor: overview

The auditor package - model verification, validation, and error analysis



auditor's pipeline: *model %>% audit() %>% plot(type=...)*



Installation

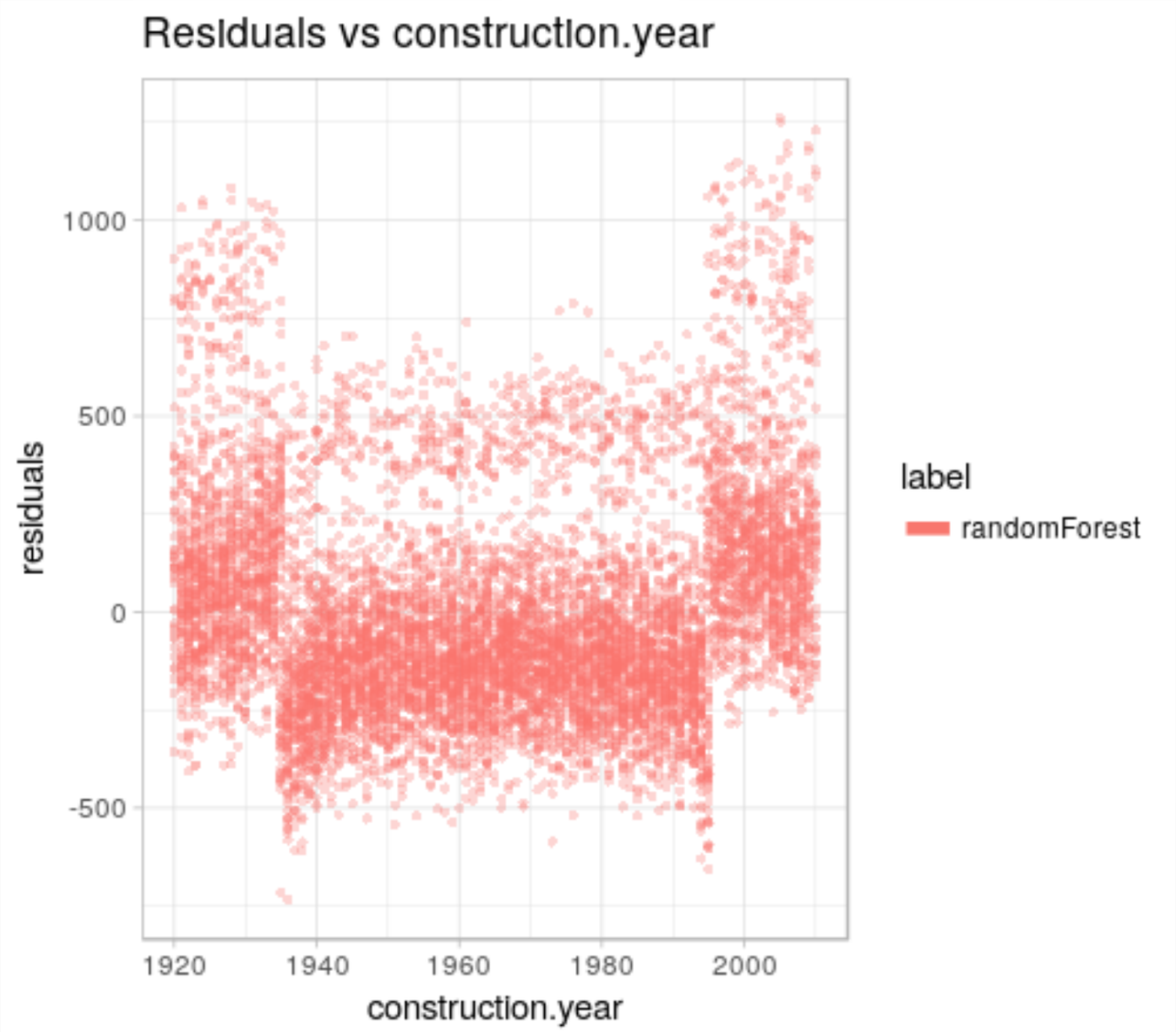
from GitHub

```
devtools::install_github("mi2-warsaw/auditor")
```

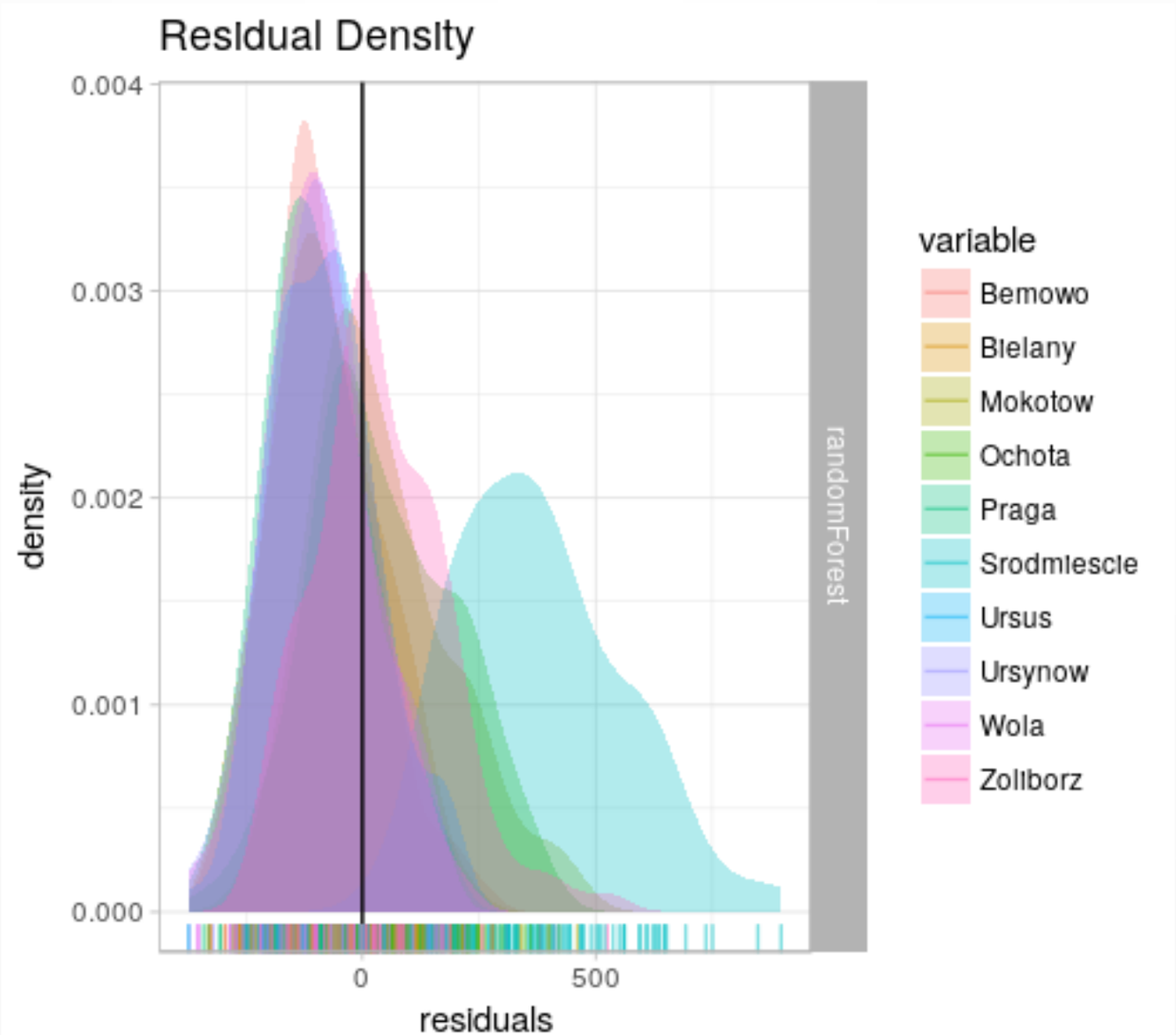
and from CRAN

```
install.packages("auditor")
```

auditor: use case 1



auditor: use case 2



auditor: more

ROC plots with auditor : : CHEAT SHEET

Basics

Package **auditor** provides several methods for model verification and validation.
This includes both, graphical methods and scores.

In this cheatsheet, we present ROC curves and their extensions for regression problem.

ROC analysis is a very popular tool for the assessment of classifier performance. So far there have been several approaches to adapt ROC curves to regression.
In the auditor package, there is a possibility to use two approaches: Regression Receiver Operating Characteristic (RROC) and Regression Error Characteristic (REC).

MODEL PREPARATION

We will show the use of a package for a logistic regression model.
The example uses *the Pima Indian Diabetes* dataset.

```
library(mlbench)
data("PimaIndiansDiabetes")

mod_glm <- glm(diabetes~.,
               family=binomial,
               data=PimaIndiansDiabetes)
```

In order to analyze the model performance, we need to convert the model into a uniform structure readable by the auditor package.

```
library(auditor)
audit_glm <- audit(mod_glm)
```

An object created with the **audit()** function can be used to draw different diagnostic plots. We will show some of them in this cheatsheet.

More detailed description and additional functionalities are presented in the package vignettes which can be found on the auditor website:
https://mi2-warsaw.github.io/auditor

REFERENCES

- *Bi J., Bennett K.P. (2003), Regression error characteristic curves, in: Twentieth International Conference on Machine Learning (ICML-2003), Washington, DC.*
- *Hernández-Orallo, J. (2013). ROC curves for regression. Pattern Recognition, 46, 3395-3411.*

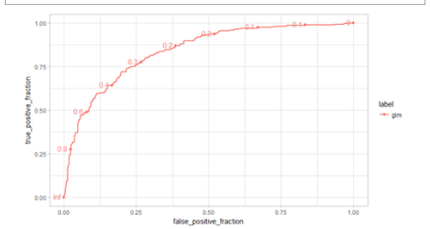
ROC curves

The **plot()** function can be used to draw several different graphs.
Use **type=plot name** to draw different types of diagnostic plots.

RECEIVER OPERATING CHARACTERISTICS (ROC)

Receiver Operating Characteristic Curve is a plot of the true positive rate (TPR) against the false positive rate (FPR) for the different thresholds.
The area under the curve (AUC) is a measure of model accuracy.

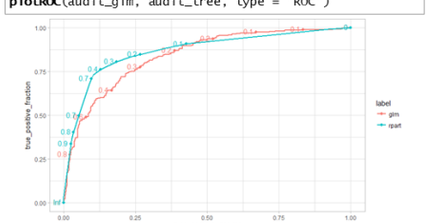
```
plot(audit_glm, type="ROC")
```



OVERLYING RESPONSES FROM DIFFERENT MODELS

A very useful feature of the auditor is the possibility to overlay performance of different models on a single plot. Below we present results for logistic regression model and classification tree.

```
library(rpart)
mod_tree <- rpart(diabetes~.,
        data=PimaIndiansDiabetes)
p.fun <- function(model, data){predict(mod_tree)[,2]}
audit_tree <- audit(mod_tree, data=PimaIndiansDiabetes,
                y = PimaIndiansDiabetes$diabetes,
                predict.function = p.fun)
plotROC(audit_glm, audit_tree, type = "ROC")
```



Regression ROC curves

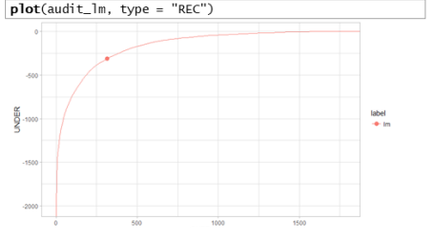
auditor package provides functions which are adaptations of ROC curves for regression. Below we present them for a linear model.
As for ROC curves, for regression curves, there is a possibility to overlay performance of different models.

```
library(car)
model_lm <- lm(prestige~education + women + income,
               data = Prestige)
audit_lm <- audit(model_lm)
```

REGRESSION RECEIVER OPERATING CHARACTERISTIC (RROC)

The basic idea of the RROC is to show model asymmetry. The RROC is a plot where on the x-axis we depict total over-estimation and on the y-axis total under-estimation.
For RROC curves we use a shift, which is an equivalent to the threshold for ROC curves. For each observation we calculate new prediction: $\hat{y}^s = \hat{y} + s$ where s is the shift. The shift equals 0 is represented by a dot.

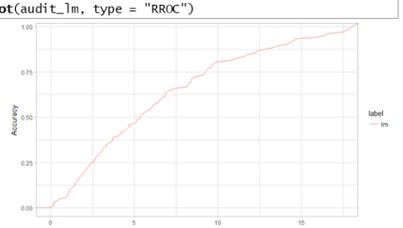
```
plot(audit_lm, type = "REC")
```



REGRESSION ERROR CHARACTERISTIC (REC)

REC curves illustrate the accuracy allowing for a certain level of error tolerance.
On the x-axis of the plot there is an error tolerance and on the y-axis, there is a percentage of observations predicted within the given tolerance. The REC curve estimates the cumulative distribution function (CDF) of the error.

```
plot(audit_lm, type = "RROC")
```



Exercises: auditor

- Create audit object for your models
- Draw plots of predicted vs observed and residuals by district for your models
- Compare them to the plots for random forest

archivist: overview

A set of tools for datasets and plots archiving

Everything that exists in *R* is an object. `archivist` is an R package that stores copies of all objects along with their metadata. It helps to manage and recreate objects with final or partial results from data analysis.

Use the `archivist` to record every result, to share these results with future you or with others, to search through repository of objects created in the past but needed now.

Installation

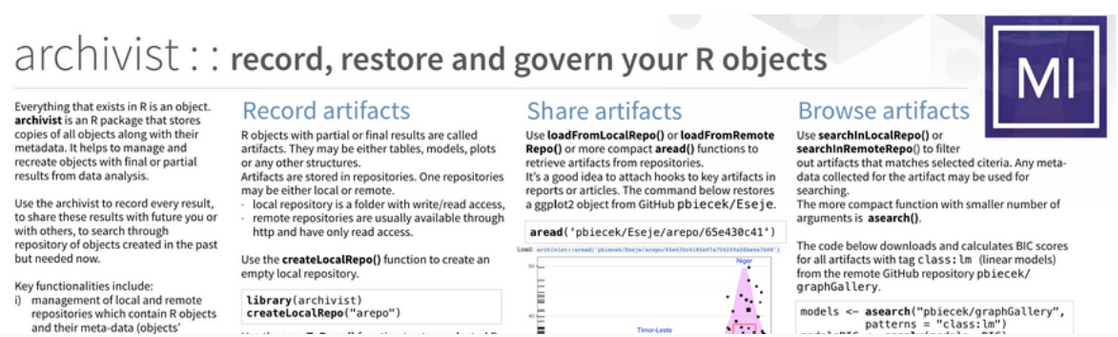
To get started, Install the latest version of **archivist** from CRAN:

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

or from GitHub:

```
devtools::install_github("pbiecek/archivist")
```

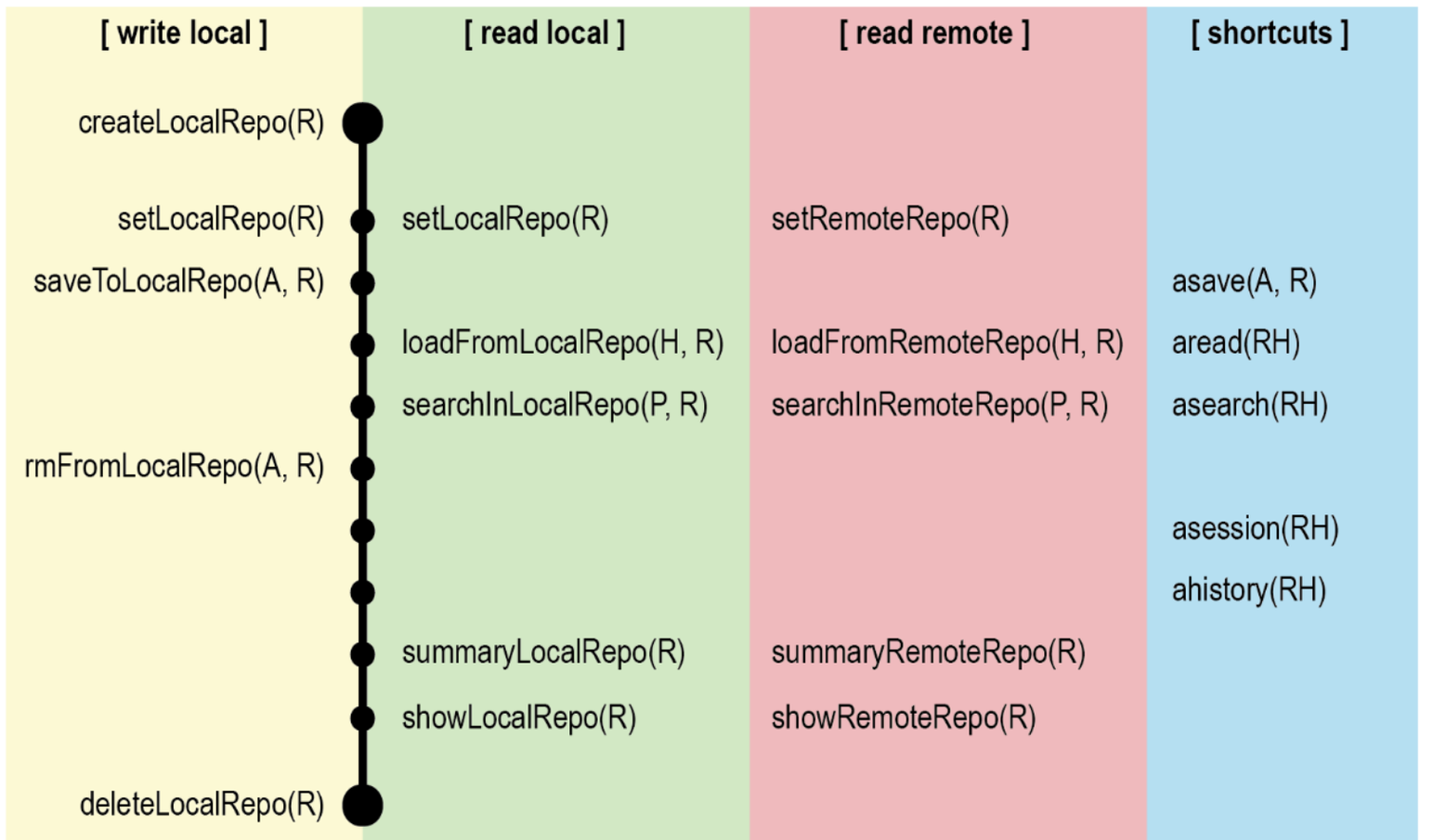
Cheatsheet



Why use archivist?

- Reproducibility
- Model / object sharing
- Model management
- Models change in time
- Environment changes in time

How does archivist work?



https://www.jstatsoft.org/article/view/v082i11

Exercises: archivist

- Create a local repository
- Save your model to the local repository with a tag
- Refresh your session (optional)
- Restore the model from local repository

ModelDown

modelDown

build passing

`modelDown` generates a website with HTML summaries for predictive models. Is uses DALEX explainers to compute and plot summaries of how given models behave. We can see how exactly scores for predictions were calculated (Prediction BreakDown), how much each variable contributes to predictions (Variable Response), which variables are the most important for a given model (Variable Importance) and how well out models behave (Model Performance).

`pkgdown` documentation: https://ml2datalab.github.io/modelDown/

An example website for regression models: https://ml2datalab.github.io/modelDown_example/

Getting started

Do you want to start right now ? Check out our getting started guide.

Index page

Basic information

- 14999 observations
- 10 columns

Explainers

- ranger
- lm

Numerical variables

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
satisfaction_level	1	14999	0.6128335	0.2486307	0.64	0.6301642	0.281694	0.09	1	0.91	-0.4762651	-0.6713455	0.0020301
last_evaluation	2	14999	0.7161017	0.1711691	0.72	0.7164670	0.222390	0.36	1	0.64	-0.0266164	-1.2392621	0.0013976
number_project	3	14999	3.8030535	1.2325924	4.00	3.7382718	1.482600	2.00	7	5.00	0.3376381	-0.4960467	0.0100644
average_montly_hours	4	14999	201.0503367	49.9430994	200.00	200.6359470	65.234400	96.00	310	214.00	0.0528314	-1.1352519	0.4077973
time_spend_company	5	14999	3.4982332	1.4601362	3.00	3.2769769	1.482600	2.00	10	8.00	1.8529484	4.7701835	0.0119224
Work_accident	6	14999	0.1446096	0.3517186	0.00	0.0558287	0.000000	0.00	1	1.00	2.0207445	2.0835473	0.0028719
left	7	14999	0.2380825	0.4259241	0.00	0.1726523	0.000000	0.00	1	1.00	1.2297966	-0.4876329	0.0034778
promotion_last_5years	8	14999	0.0212681	0.1442815	0.00	0.0000000	0.000000	0.00	1	1.00	6.6356410	42.0345334	0.0011781

Index page presents basic information about data provided in explainers. You can also see types of all explainers given as parameters. Additionally, summary statistics are available for numerical variables. For categorical variables, tables with frequencies of factor levels are presented.

Model Performance

Alternative software

- iml
- pdp
- ICEbox
- ALEPlot
- lime
- ShapleyR

Model-specific explanations

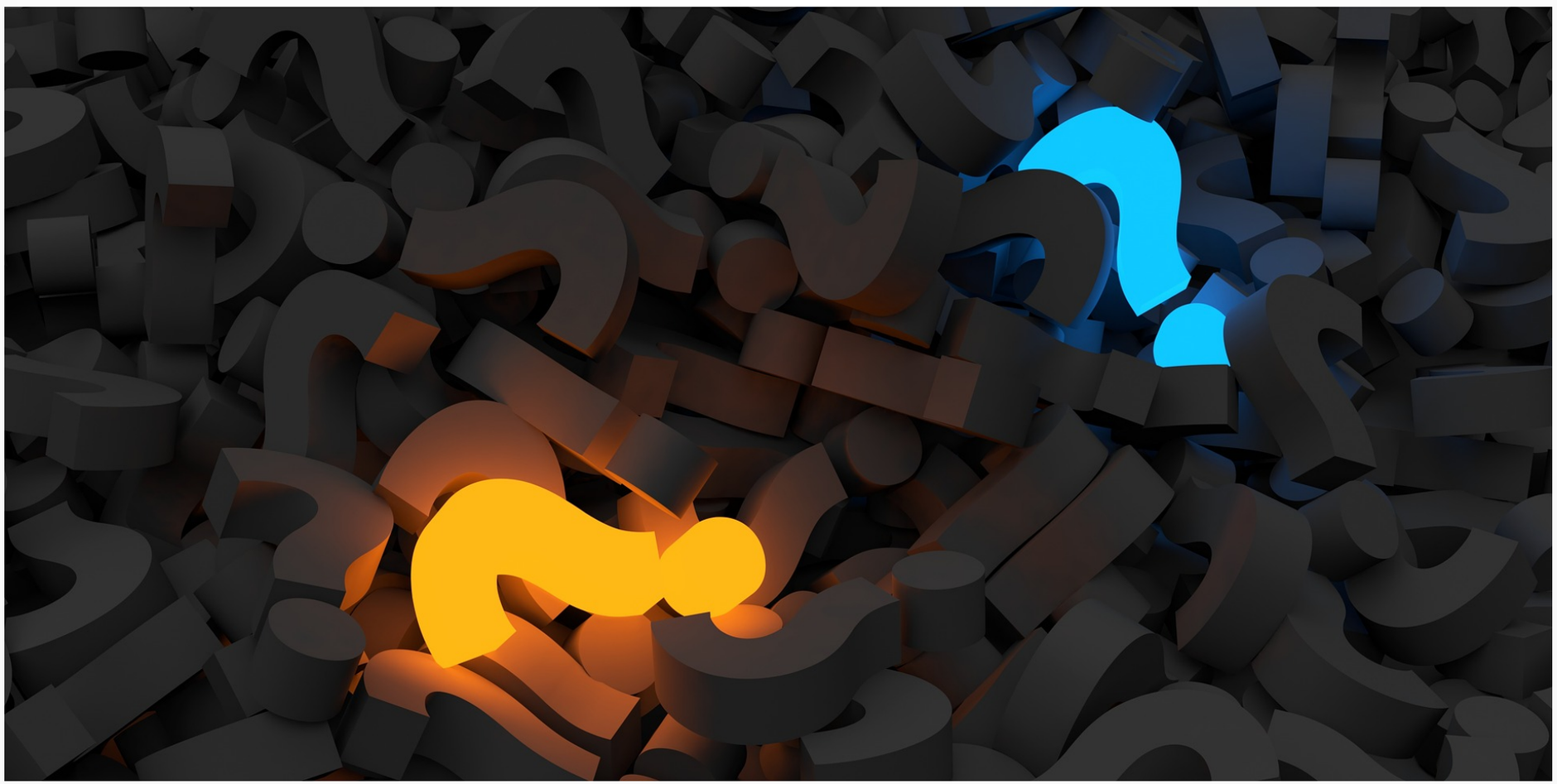
- randomForestExplainer
- xgboostExplainer
- condvis
- forestmodel / forestplot / sjPlot
- randomForest and other specific packages

More information

https://pbiecek.github.io/DALEX_docs/

https://christophm.github.io/interpretable-ml-book/

Discussion



Contact information

- https://github.com/mi2datalab
- http://mi2.mini.pw.edu.pl/
- https://github.com/pbiecek/DALEX/issues
- http://pbiecek.github.io
- http://mstaniak.pl/

Thank you for your attention!