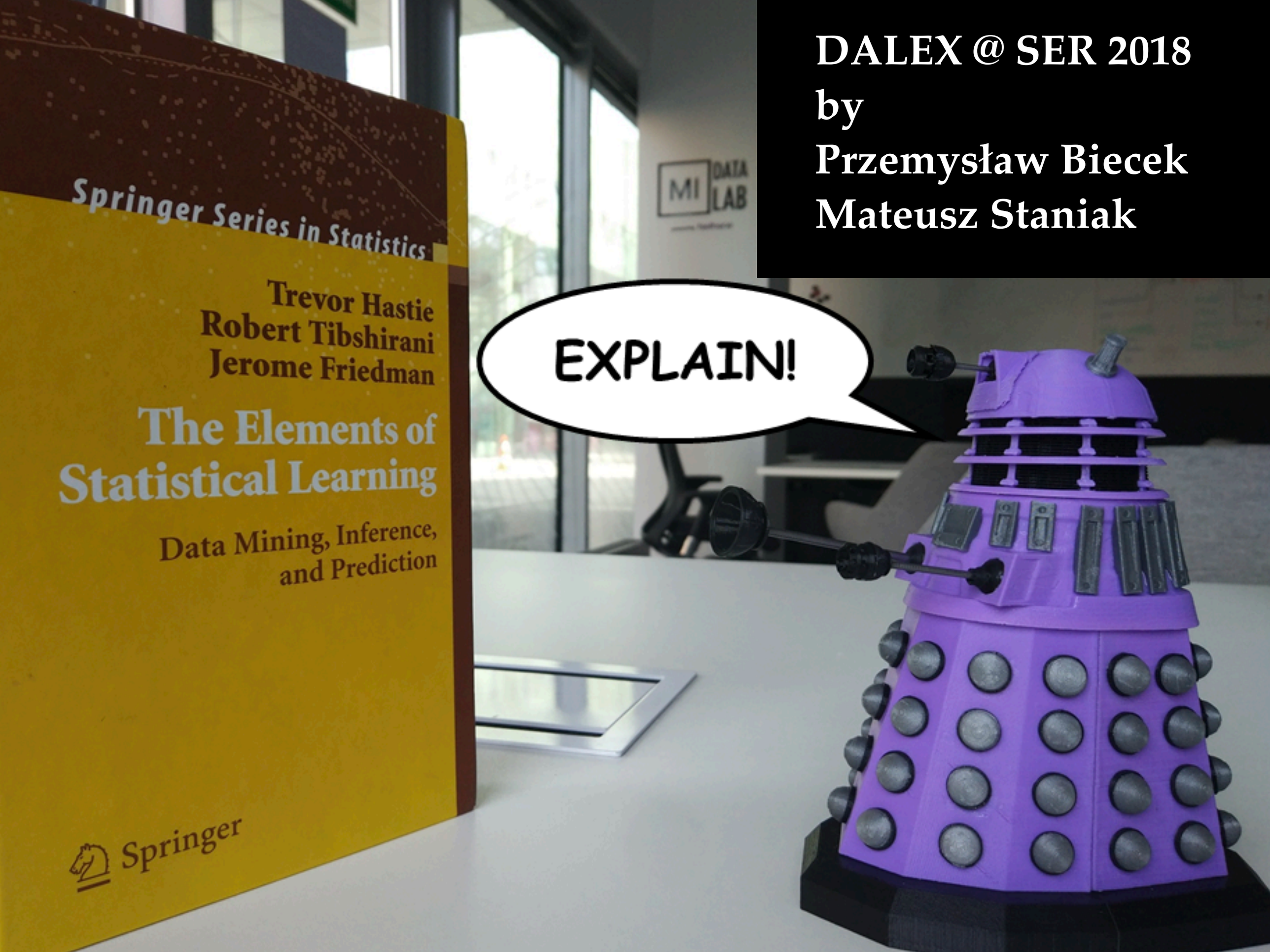
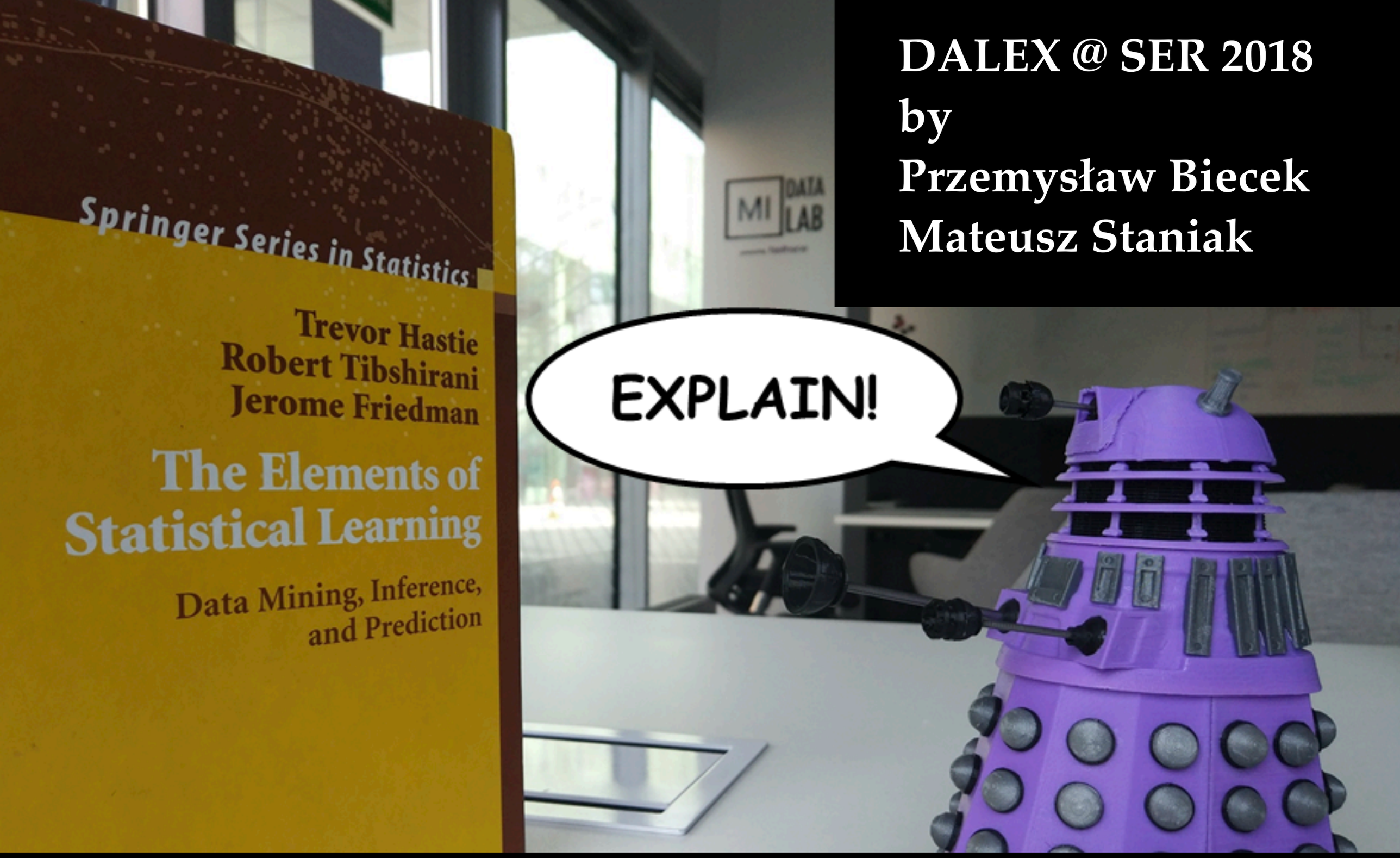
DALEX @ SER 2018

by

Przemysław Biecek
Mateusz Staniak



DALEX @ SER 2018
by
Przemysław Biecek
Mateusz Staniak



DALEX explains Machine Learning Models

The Good, the Bad and the Ugly

SCIENTIFIC REPORTS

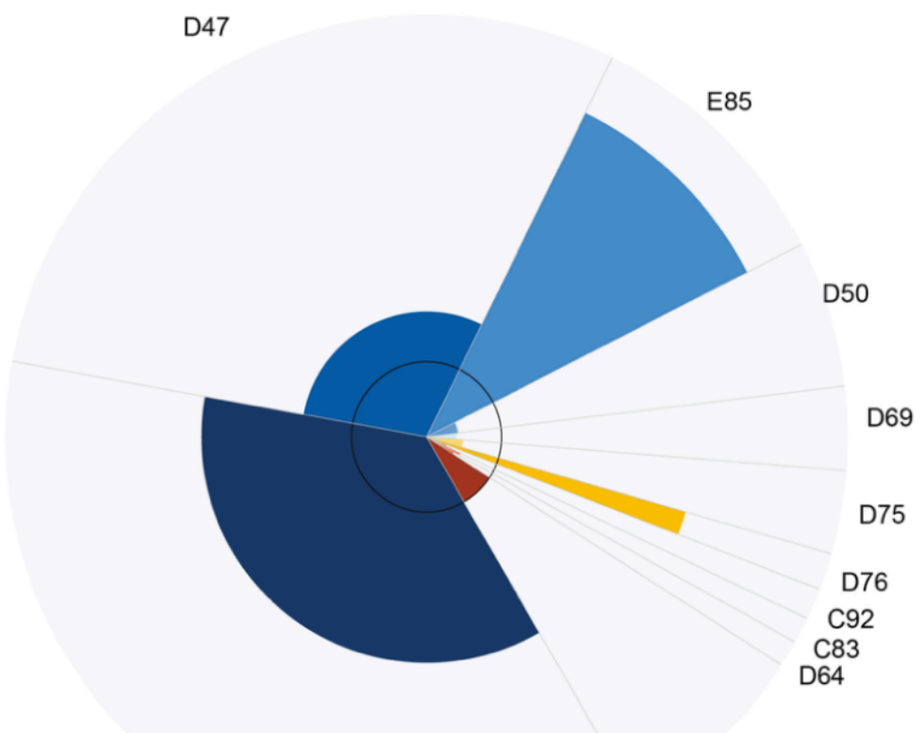
OPEN

An application of machine learning to haematological diagnosis

Gregor Gunčar[1], Matjaž Kukar[1], Mateja Notar[1], Miran Brvar[2], Peter Černelč[3], Manca Notar[1] & Marko Notar[1]

d: 2 August 2017
d: 14 December 2017
ed online: 11 January 2018

Quick and accurate medical diagnoses are crucial for the successful treatment of diseases. Using machine learning algorithms and based on laboratory blood test results, we have built two models to predict a haematologic disease. One predictive model used all the available blood test parameters and the other used only a reduced set that is usually measured upon patient admittance. Both models produced good results, obtaining prediction accuracies of 0.88 and 0.86 when considering the list of five most likely diseases and 0.59 and 0.57 when considering only the most likely disease. The models did not d "fingerpr and indic clinical t specialis tests alo unprece



ICD code	Prediction	Information score	Disease category
C90	36.20%	2.01	Multiple myeloma and malignant plasma cell neoplasms
D47	29.40%	0.67	Other neoplasms of uncertain or unknown behaviour of lymphoid. haematopoietic and related tissue
E85	10.20%	3.81	Amyloidosis
D50	5.60%	-1.37	Iron deficiency anaemia
D69	3.20%	-1.50	Purpura and other haemorrhagic conditions
D75	3.20%	-1.05	Other diseases of blood and blood-forming organs
D76	1.40%	2.60	Certain diseases involving lymphoreticular tissue and reticulohistiocytic system
C92	1.20%	-2.55	Myeloid leukaemia
C83	1.00%	-1.01	Diffuse non-Hodgkin lymphoma
D64	1.00%	-1.41	Other anaemias

nature biomedical engineering

ARTICLES

https://doi.org/10.1038/s41551-018-0195-0

Prediction of cardiovascular risk factors from retinal fundus photographs via deep learning

Ryan Poplin[1,4], Avinash V. Varadarajan[1,4], Katy Blumer[1], Yun Liu[1], Michael V. McConnell[2,3], Greg S. Corrado[1], Lily Peng[1,4]* and Dale R. Webster[1,4]

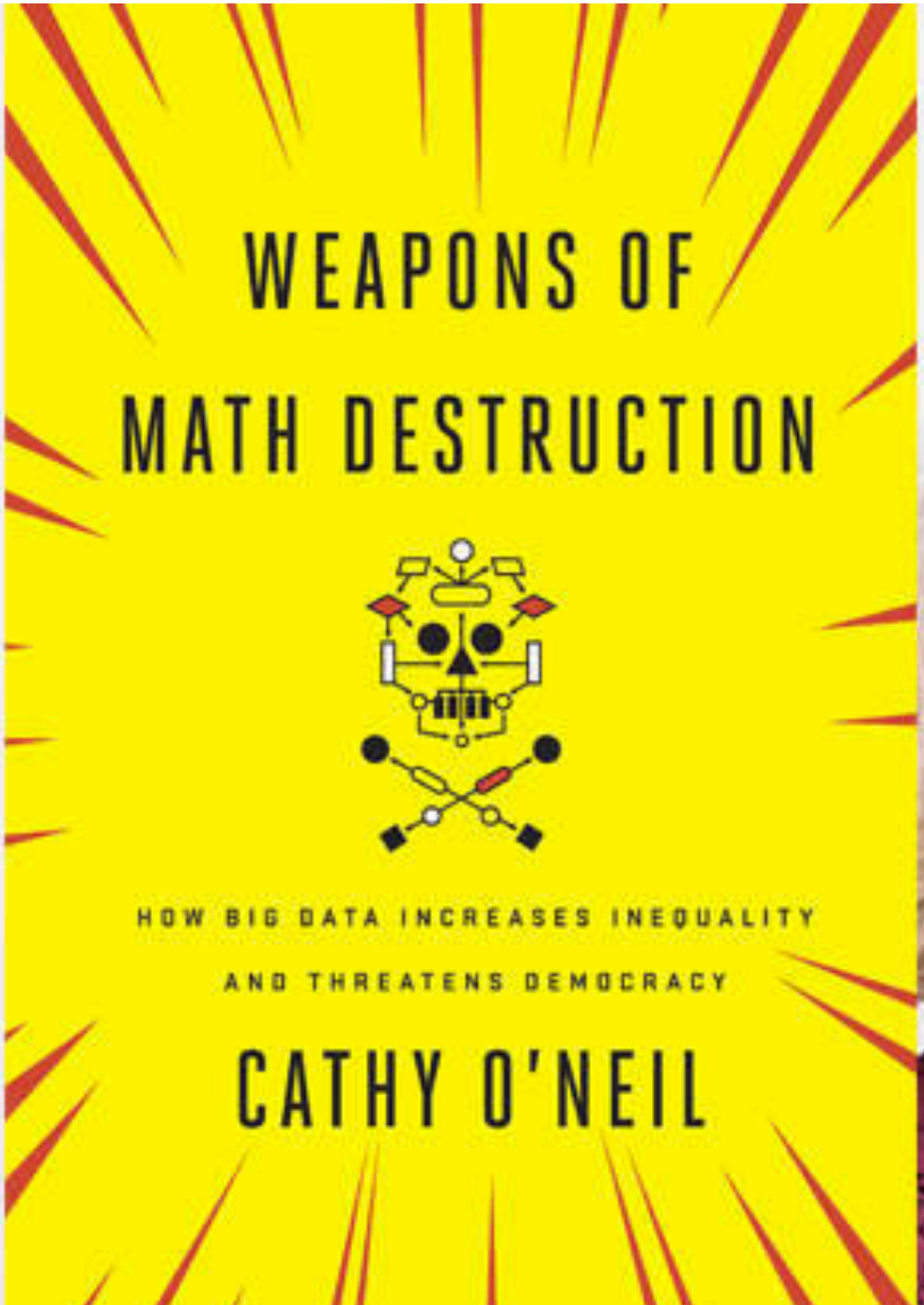
Traditionally, medical discoveries are made by observing associations, making hypotheses from them and then designing and running experiments to test the hypotheses. However, with medical images, observing and quantifying associations can often be difficult because of the wide variety of features, patterns, colours, values and shapes that are present in real data. Here, we show that deep learning can extract new knowledge from retinal fundus images. Using deep-learning models trained on data from 284,335 patients and validated on two independent datasets of 12,026 and 999 patients, we predicted cardiovascular risk factors not previously thought to be present or quantifiable in retinal images, such as age (mean absolute error within 3.26 years), gender (area under the receiver operating characteristic curve (AUC) = 0.97), smoking status (AUC = 0.71), systolic blood pressure (mean absolute error within 11.23 mmHg) and major adverse cardiac events (AUC = 0.70). We also show that the trained deep-learning models used anatomical features, such as the optic disc or blood vessels, to generate each prediction.

Risk stratification is central to identifying and managing groups at risk for cardiovascular disease, which remains the leading cause of death globally[1]. Although the availability of cardiovascular disease risk calculators, such as the Pooled Cohort equations[2], Framingham[3–5] and Systematic COronary Risk Evaluation (SCORE)[6,7], is widespread, there are many efforts to improve risk predictions. Phenotypic information, particularly of vascu-

changes[22,23] ar
temic health c
clinical utility
work, we dem
cardiovascula

Machine le
ety of classifi

Cathy O'Neil: The era of blind faith in big data must end

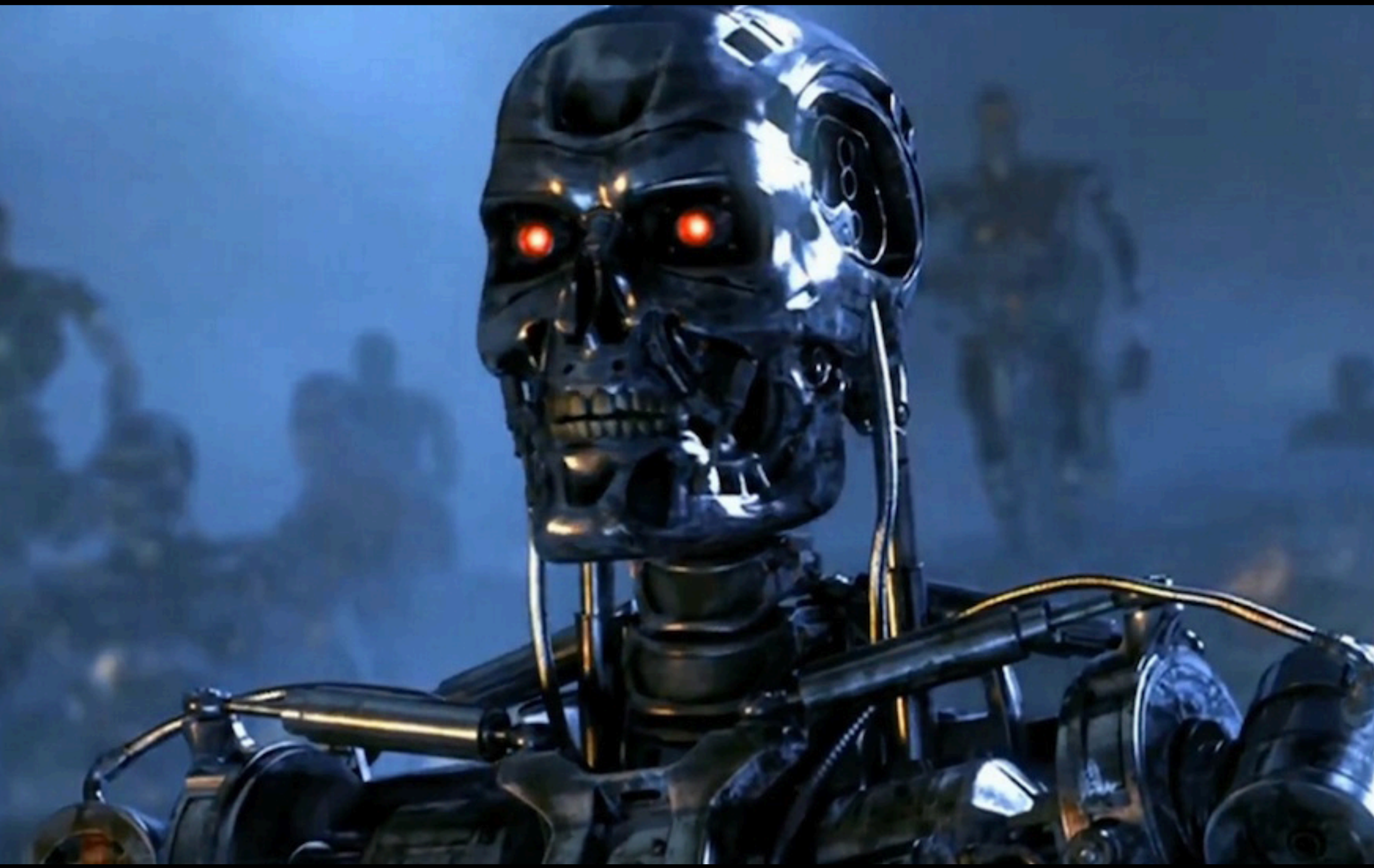


- “You don’t see a lot of skepticism,” she says. “The algorithms are like shiny new toys that we can’t resist using. We trust them so much that we project meaning on to them.”

- Ultimately algorithms, according to O’Neil, reinforce discrimination and widen inequality, “using people’s fear and trust of mathematics to prevent them from asking questions”.

https://www.theguardian.com/books/2016/oct/27/cathy-oneil-weapons-of-math-destruction-algorithms-big-data

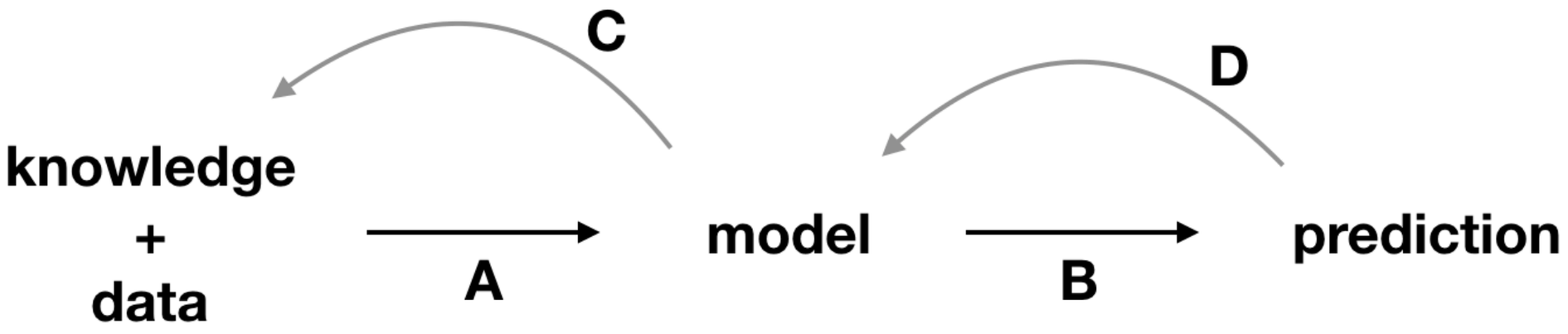
Future A: Machine Learning replaces humans



Future B: Machine Learning empowers humans

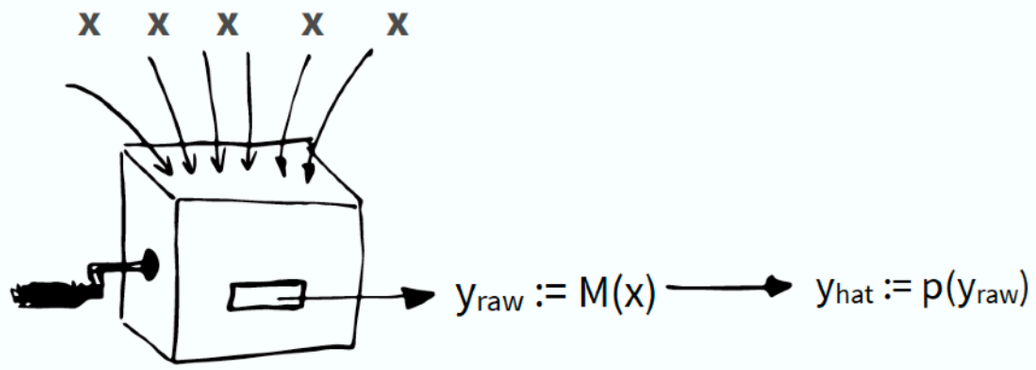
Architecture of DALEX

Typical workflow



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.*

A)



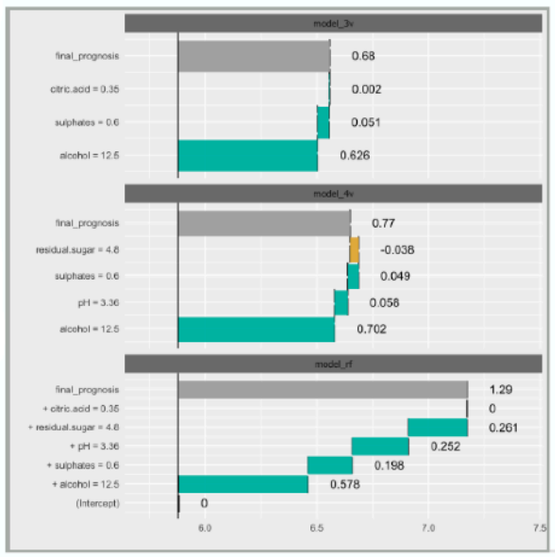
model M

B)

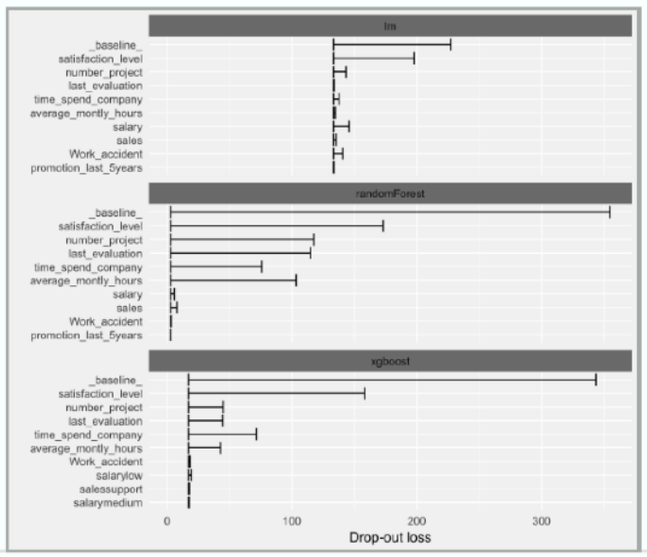
explain(model; data; y; predict_function; trans)

C)

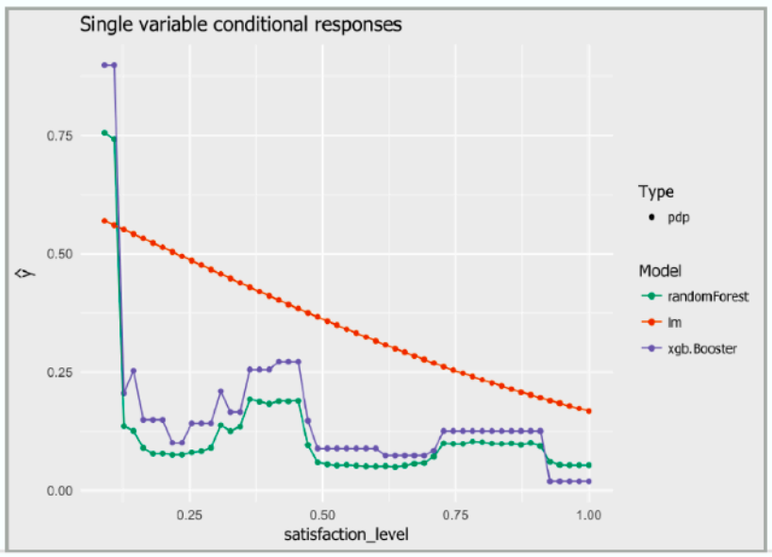
single_prediction(explainer, x)

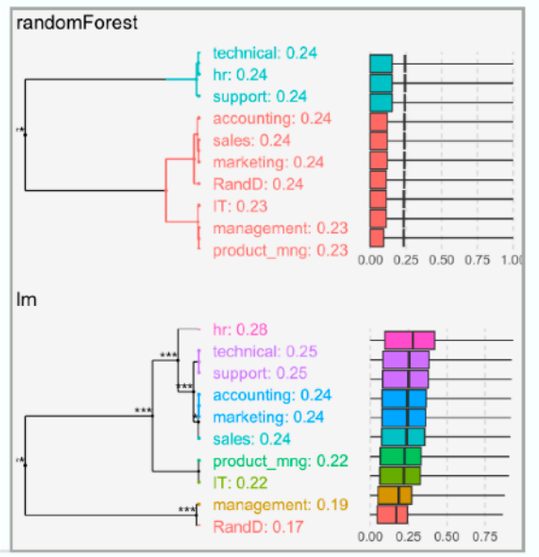


variable_dropout(explainer)

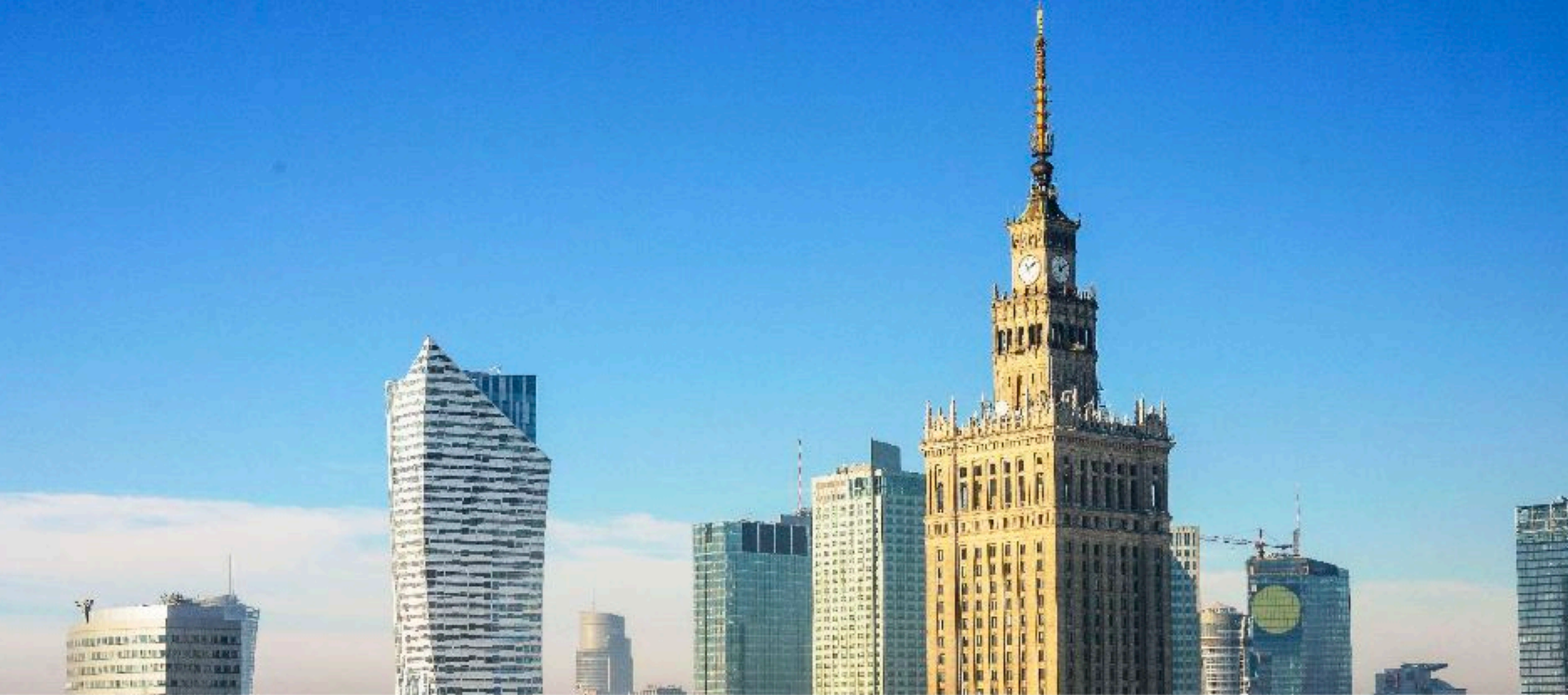


single_variable(explainer, variable)





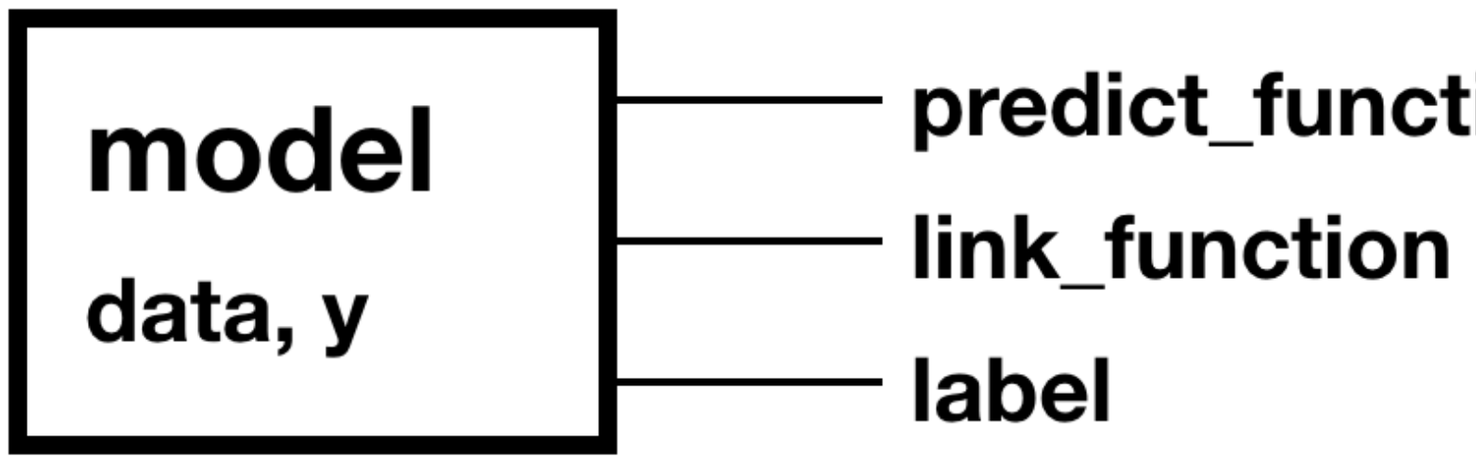
http://www.warszawa.pl/biznes/panorama-warszawy/galeria/dzis_palac_kultury_i_nauki/



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

m2.price	construction.year	surface	floor	no.rooms	district
5897	1953	25	3	1	Srodmiescie
1818	1992	143	9	5	Bielany
3643	1937	56	1	2	Praga
3517	1995	93	7	3	Ochota
3013	1992	144	6	5	Mokotow

First: Create explainer

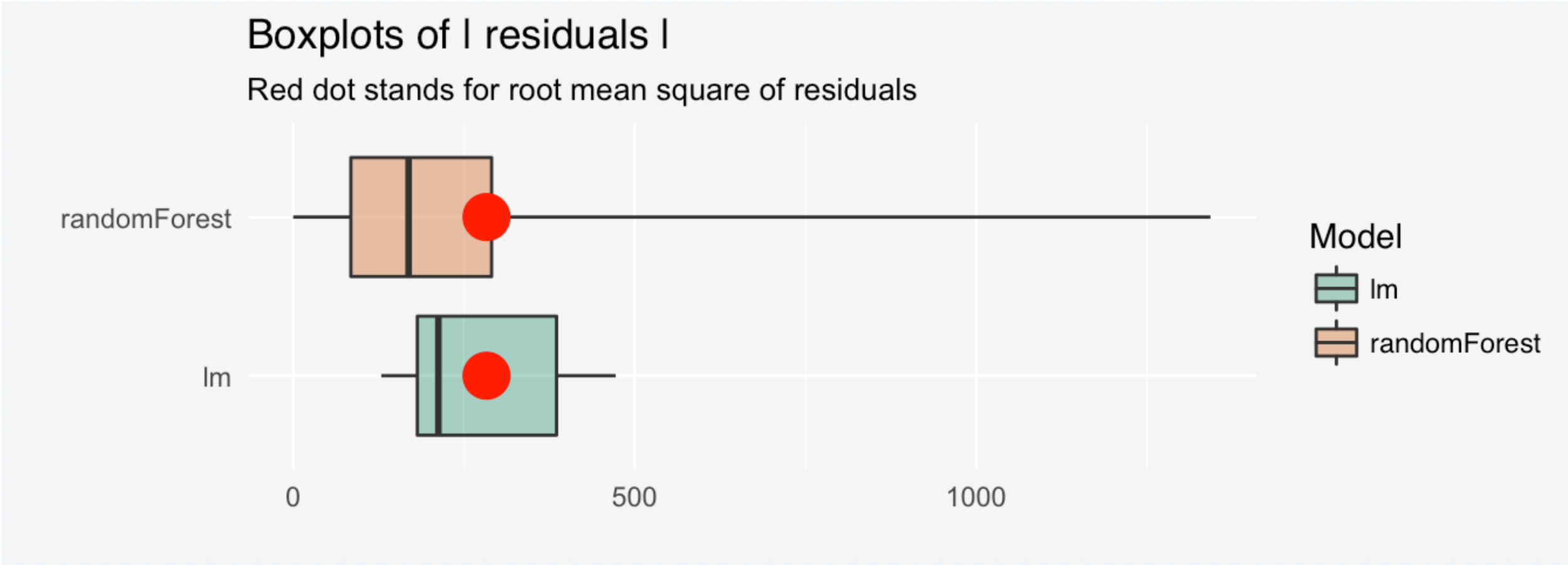



```
explain(model, data, y, predict_function,
                link, ..., label)
```

Model understanding

1. Model performance

```r
plot(mp_lm, mp_rf, geom = "boxplot")
```



Model understanding

2. Variable importance

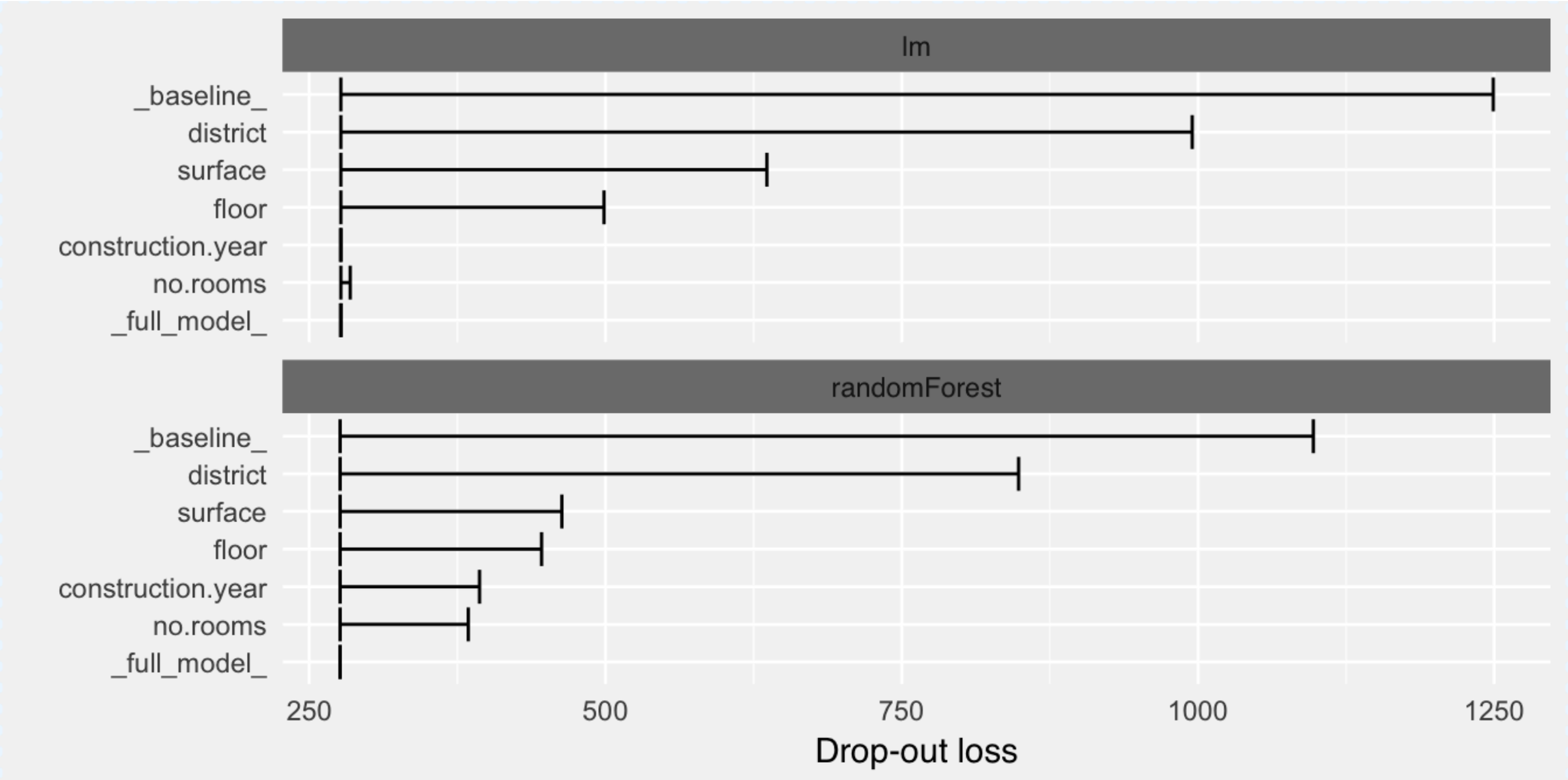
Which features are the most influential?

$$\hat{e}_{\text{orig}}(f) := \frac{1}{n}\sum_{i=1}^{n} L(f, \mathbf{Z}_{[i,\cdot]}) = \frac{1}{n}\sum_{i=1}^{n} L\{f, (\mathbf{y}_{[i]}, \mathbf{X}_{1[i,\cdot]}, \mathbf{X}_{2[i,\cdot]})\} = \hat{\mathbb{E}} L(f, Z),$$

$$\begin{aligned}\hat{e}_{\text{switch}}(f) &:= \frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j\neq i} h_f(\mathbf{Z}_{[i,\cdot]}, \mathbf{Z}_{[j,\cdot]}) \\ &= \frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j\neq i} L\{f, (\mathbf{y}_{[j]}, \mathbf{X}_{1[i,\cdot]}, \mathbf{X}_{2[j,\cdot]})\}.\end{aligned}$$

$$\text{Model Class Reliance} = \frac{\hat{e}_{\text{switch}}(f)}{\hat{e}_{\text{orig}}(f)},$$

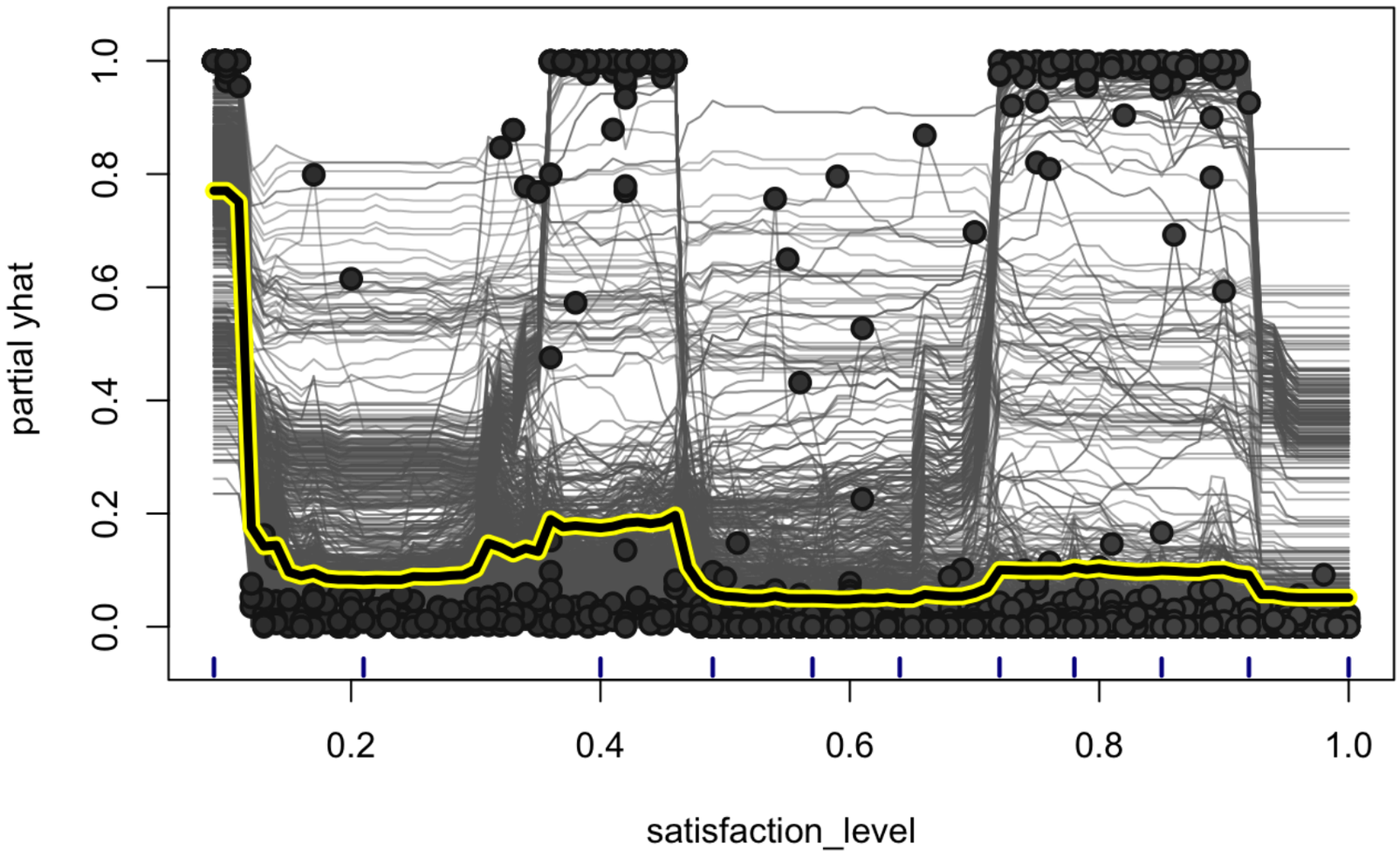
Model Class Reliance: Variable Importance Measures for any Machine Learning Model Class, from the "Rashomon" Perspective
Aaron Fisher, Cynthia Rudin, Francesca Dominici


lm
baseline
district
surface
floor
construction.year
no.rooms
_full_model_
randomForest
baseline
district
surface
floor
construction.year
no.rooms
_full_model_
250
500
750
1000
1250
Drop-out loss

Model understanding

3. Variable response

What is the conditional model response as a function of a single feature?



Partial Dependence Plots, Greenwell 2017
Accumulated Local Effects Plots, Apley 2017
Individual Conditional Expectations, Goldstein , Kapelner, Bleich, Pitkin 2015
FactorMerger: Hierarchical Algorithm for Post-Hoc Testing, Sitko Biecek 2017

What about categorical dependent variable?

Algorithm 1 The outline of the Merging Path Plot algorithm implemented in **factorMerger**

function MERGEFACTORS($responseVariable, groupingVariable, adjacent$)
  $currentModel := createModel(responseVariable, groupingVariable)$
  $mergingPath := list(currentModel)$
  while $|levels(groupingVariable)| \geq 1$ **do**
    $pairsSet := generatePairs(groupingVariable, responseVariable, adjacent)$
    $selectedPair := \text{argmax}_{\text{pair} \in \text{pairsSet}} objectiveFunction(pair, responseVariable, groupingVariable)$
    $groupingVariable := mergeLevels(groupingVariable, selectedPair)$
    $currentModel := createModel(responseVariable, groupingVariable)$
    $mergingPath := add(mergingPath, currentModel)$
  end while
  return(mergingPath)
end function

The Merging Path Plot: adaptive fusing of k-groups with likelihood-based model selection
Agnieszka Sitko, Przemysław Biecek (2017)
https://arxiv.org/abs/1709.04412
https://github.com/geneticsMiNIng/FactorMerger

factorMerger
Cheat Sheet

Agnieszka Sitko [aut, cre]
Przemysław Biecek [aut, ths]
University of Warsaw

Introduction

How to check if averages are different among k groups? Use **ANOVA**!

How to visualise how these groups are different? Use **factorMerger**!

The aim of **factorMerger** is to provide informative and easy to understand visualisations of *post-hoc* comparisons. It gives consistent and non-overlapping adaptive fusing of groups based on likelihood ratio test (LRT). The package **factorMerger** works for wide spectrum of families like Gaussian, binomial or survival.

Results from the adaptive fusing are presented with the *Merging Paths Plots* - a hierarchical representation of LRT-based distances among groups.

In addition, the *Generalized Information Criterion (GIC)* is presented for fused models. This criterion may be used to choose the optimal segmentation of groups.

Graphical summary of the variable of interest in each group is presented in the right panel.

Find more in *https://arxiv.org/abs/1709.04412*

Example

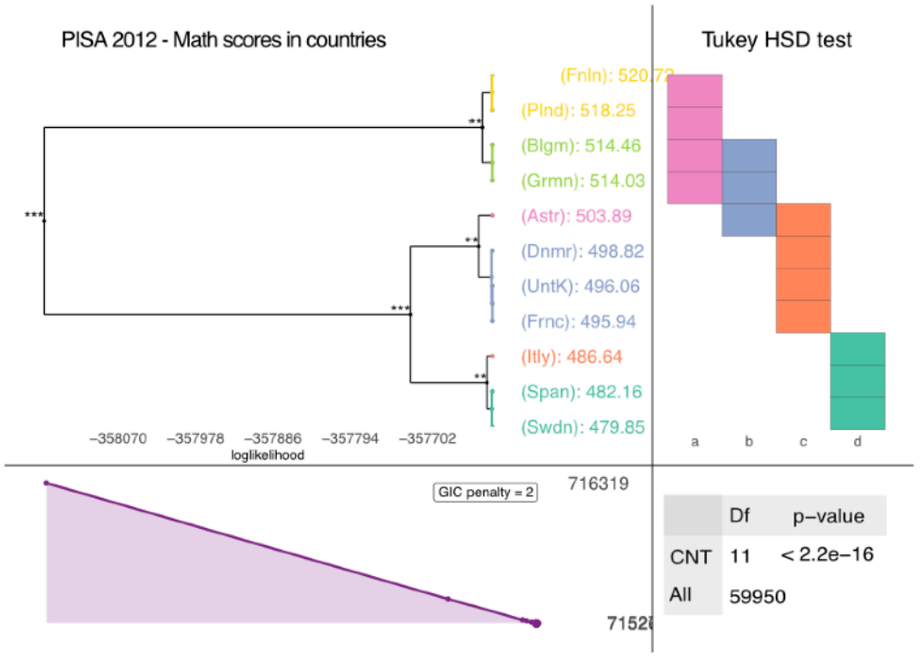
```
library(factorMerger)

fmAll <- mergeFactors(
    response  = pisaEuro$math,
    factor = pisaEuro$country,
    method = "fast-adaptive",
    family = "gaussian")

print(fmAll)

plot(fmAll,
    panel = "all",
    responsePanel = "tukey")
```

Set of tools to support post-hoc segmentation of groups

Four panels to see it all

A. The Merging Paths Plot panel
Shows the hierarchical structure of similarity between groups.
Stars presents how significant are differences between two clusters.
Height of the join corresponds to the likelihood of the combined model.

B. The response panel
Shows graphical summaries for group separately.
Use the responsePanel argument to select the desired summary.



C. The GIC panel
Shows the *Generalized Information Criterion* for all models in MPP panel.
Default *penalty=2* corresponds to the AIC criterion.
Colours in the MPP panel corresponds to the optimal model.

D. The summary panel
Shows the no. groups, no. observations,
and the p-value for global hypothesis
that parameter of interest is equall in all groups.

Key arguments of factorMerger

The **factorMerger** package is highly customisable.

The **panel** argument selects panels to be plotted.

The **nodesSpacing** argument selects how MPP nodes shall be spaced.

The **responsePanel** argument selects which summary shall be presented in panel B.

The **responsePanelPalette** argument selects the colouring palette.

The **colorClusters** argument selects if segments of groups shall be colour coded.

The **penalty** argument selects penalty for GIC criterion.

Argument	Summary
`panel = "all"`	All panels
`panel = "response"`	Panels A and B
`panel = "GIC"`	Panels A and C
`panel = "tree"`	Only panel A

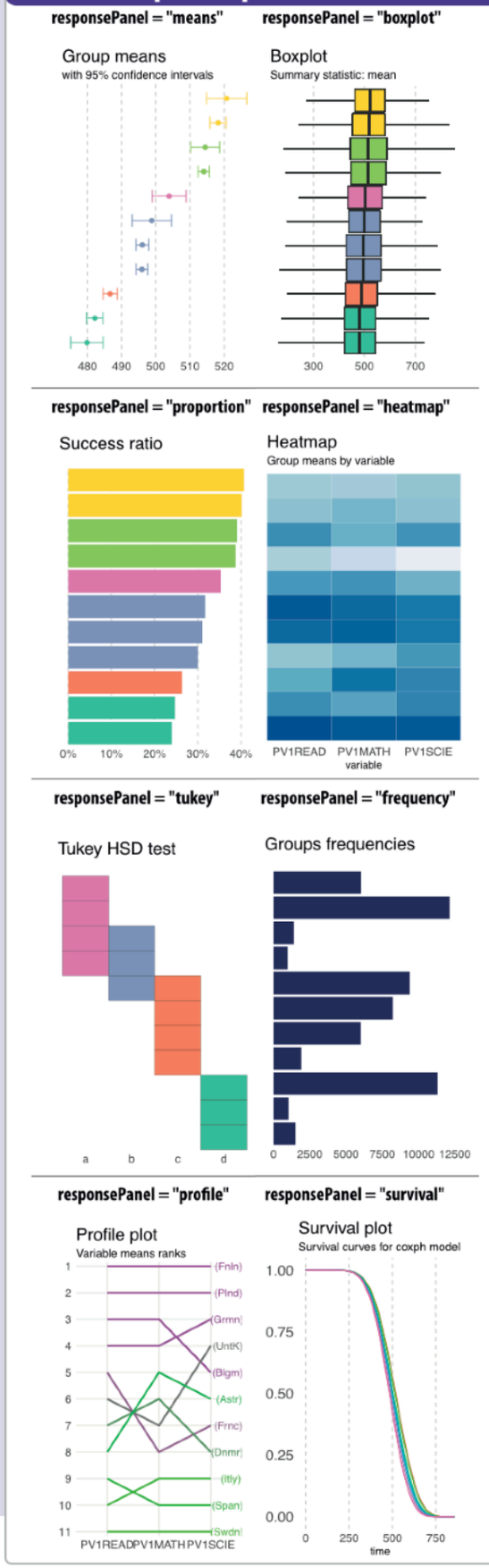
The **method** argument determines how pairs of groups are considered for grouping. The most computational demanding is *adaptive*. For larger number of groups one may consider only pairs of groups with closest averages or use the *fast-fixed* heuristic (fastest solution).

Argument	Summary
`method = "adaptive"`	Compare all pairs of groups
`method = "fast-adapive"`	Compare nearby groups
`method = "fixed"`	Delete or merge heuristic
`method = "fast-fixed"`	Reduced DMR method

The **family** argument determines what models are to be compared with likelihood ratio test. Either linear models, generalised linear models or Cox regression models.

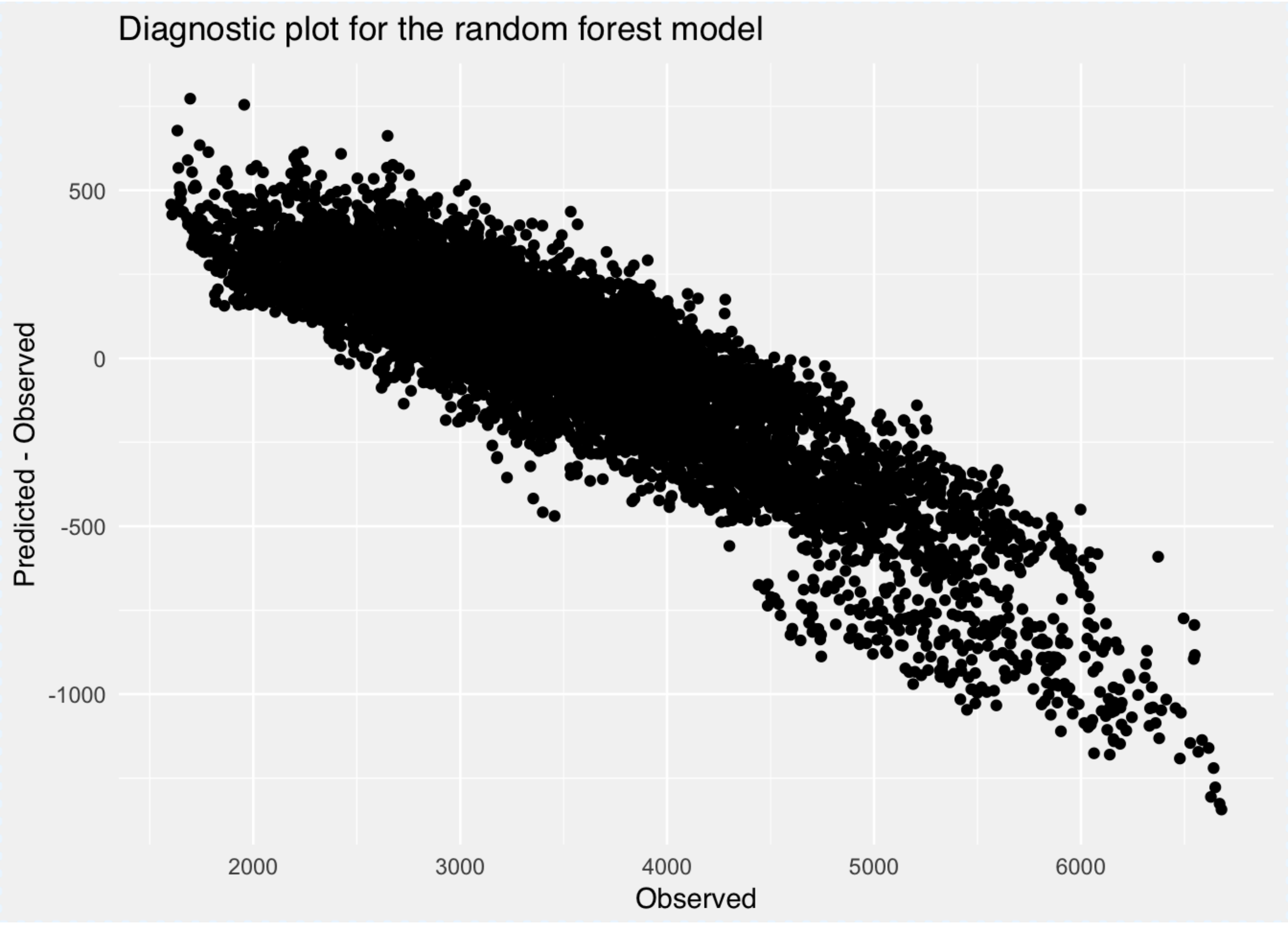
Argument	Summary
`family = "gaussian"`	For 1/n-dimensional Gaussian
`family = "binomial"`	For binomial regression
`family = "survival"`	For Cox regression

The response panel summaries



Prediction understanding

4. Outlier detection


Diagnostic plot for the random forest model
500
0
-500
-1000
Predicted - Observed
2000
3000
4000
5000
6000
Observed

Error analysis with auditor : : CHEAT SHEET

Basics

Package **auditor** provides several methods for model verification and validation by error analysis. This includes both, graphical methods and scores, which that can be useful for comparison of models performance.

Residual analysis is widely used for linear and generalized linear models, so there are many statistical tools to evaluate the goodness of fit. However, these methods are not suitable for each machine learning model. In particular, it is difficult to apply them to black box models such us random forest, due to the lack of information about the error distribution.

The tools included in auditor can be used to assess the fit of many types of models, including black boxes.

MODEL PREPARATION

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

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

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

In order to analyse the model residuals, we need to convert 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

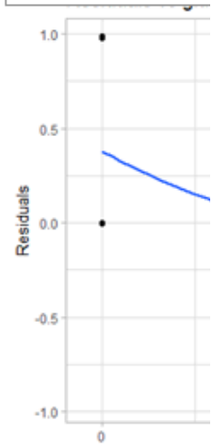
REFERNCES

- *Moral, R., Hinde, J., & Demétrio, C. (2017). Half-Normal Plots and Overdispersed Models in R: The hnp Package. Journal of Statistical Software, 81(10), 1 - 23.*

Basic plo

The **plot()** function
By default it is drav

```
plot(audit_gl
```

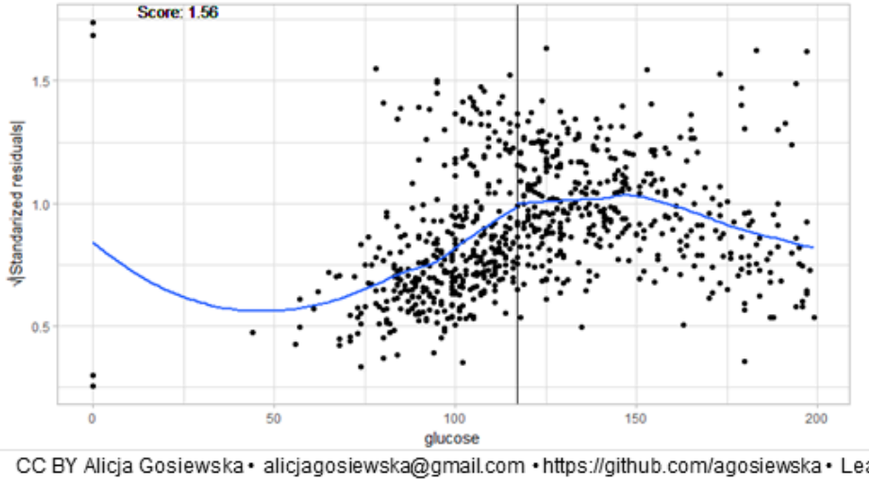


In the example al
to provide any pr
To use non-defau
prediction, you can use arguments **residual.function** and **predict.function** while using an audit function.

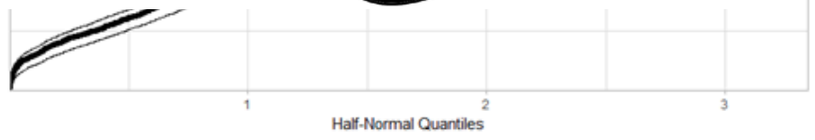
```
p.fun <- function(model, data){
           predict(model, data, type="link")
         }
r.fun <- function(model,y){residuals(model)}

audit_glm_dev <- audit(mod_glm,
                       predict.function = p.fun,
                       residual.function = r.fun)
```

Use **type=plot name** to draw different types of diagnostic plots.
The available parameter values are **'ACF', 'Autocorrelation', 'Cook', 'HalfNormal', 'Residuals' and 'ScaleLocation'**.

```
plot(audit_glm_dev, type="ScaleLocation",
  variable = "glucose")
```

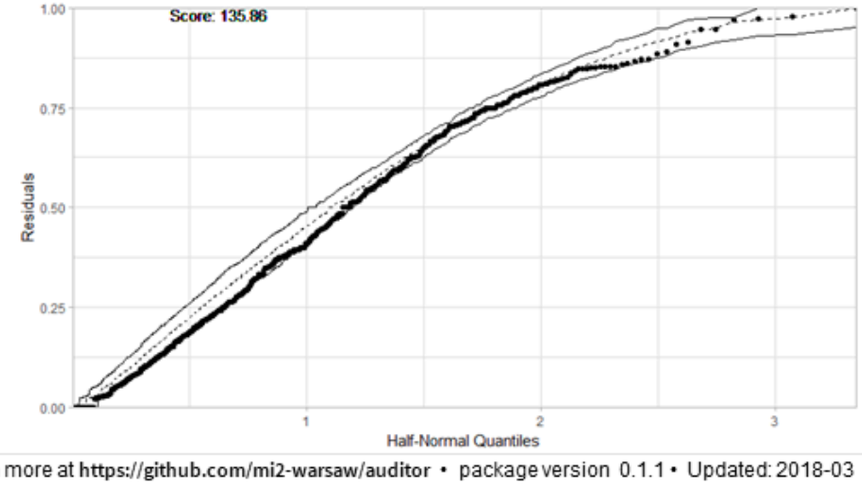


Half-Normal plots



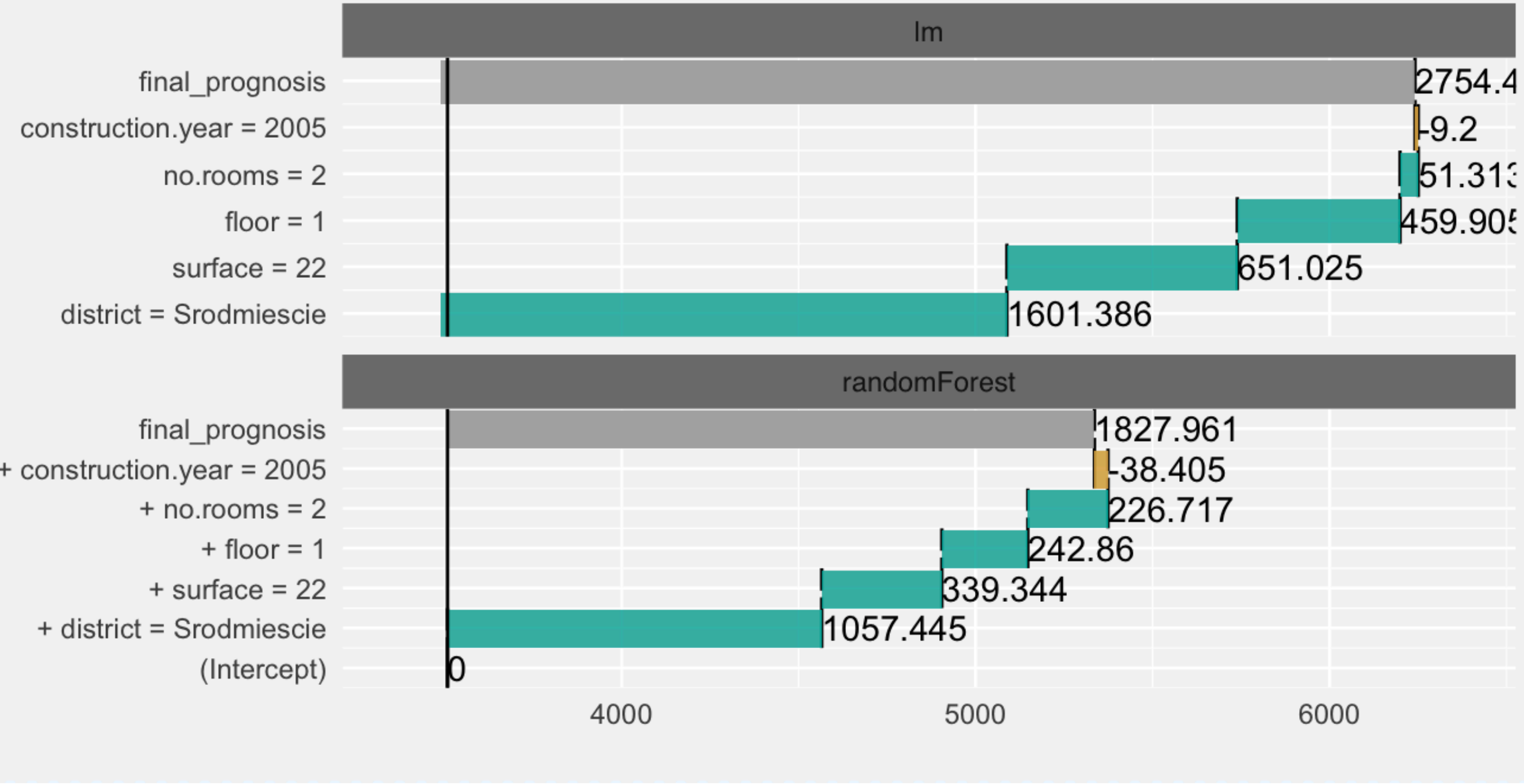
Half-Normal plots allow to check the model fit but comparing two models may be difficult.
A useful tool to compare goodness-of-fit of two models is score displayed on the plot. Score is a sum of logarithms of estimated PDF at each point. It is calculated on the basis of simulated data, so it may differ between function calls.

```
library(randomForest)
mod_rf <- randomForest(diabetes~.,
                       data=PimaIndiansDiabetes, ntree=100)
audit_rf <- audit(mod_rf)
plotHalfNormal(audit_rf)
```

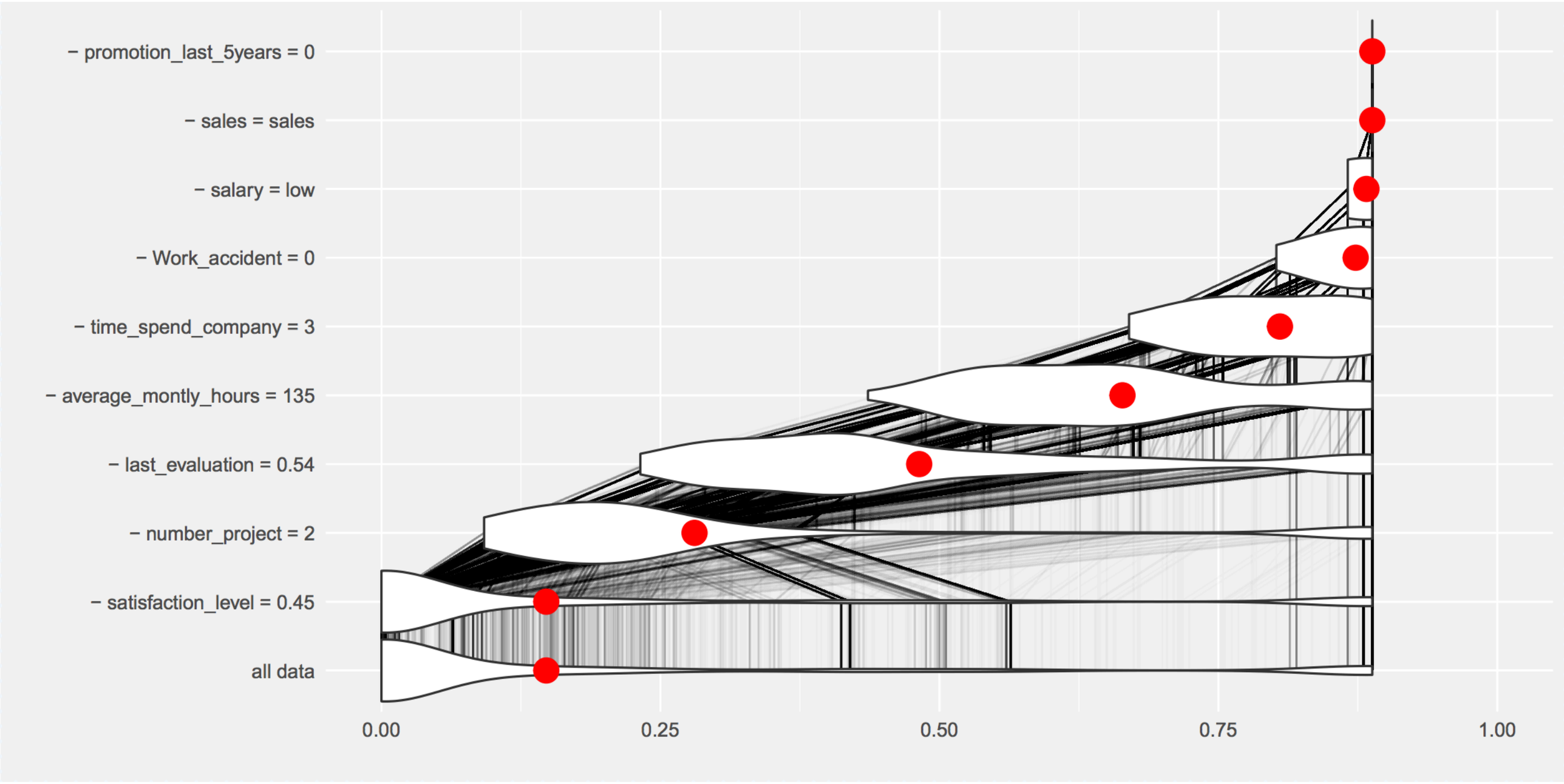


Prediction understanding

5. Prediction breakDown


lm
final_prognosis
construction.year = 2005
no.rooms = 2
floor = 1
surface = 22
district = Srodmiescie
2754.4
-9.2
51.313
459.905
651.025
1601.386
randomForest
final_prognosis
+ construction.year = 2005
+ no.rooms = 2
+ floor = 1
+ surface = 22
+ district = Srodmiescie
(Intercept)
1827.961
-38.405
226.717
242.86
339.344
1057.445
0
4000
5000
6000

Relaxed decomposition of a single prediction



Definition **(Distance to relaxed model prediction)** *For a set of indexes indSet let us define the distance between model prediction and relaxed model prediction.*

$$d(x^{new}, indSet) := |f^{IndSet}(x^{new}) - f^{(}x^{new})|.$$

Definition **(Relaxed model prediction)** *Let $f^{IndSet}(x^{new})$ denote an expected model prediction for x^{new} relaxed on the set of indexes $IndSet \subset \{1, \ldots, p\}$.*

$$f^{IndSet}(x^{new}) = E[f(x)|x_{indSet} = x^{new}_{indSet}].$$

It is an expected value for model response conditioned on variables from set indSet in a way, that $\forall_{i \in indSet} x_i = x_i^{new}$.

Algorithm 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
    $j_{min} \leftarrow argmin_j dist(j)$
    $Contributions(i) \leftarrow f^{IndSet}(x^{new}) - f^{IndSet \setminus \{j_{min}\}}(x^{new})$
    $Variables(i) \leftarrow j_{min}$
    $IndSet \leftarrow IndSet \setminus \{j_{min}\}$
end for

breakDown plots : : **visual explanations for lm/glm models**

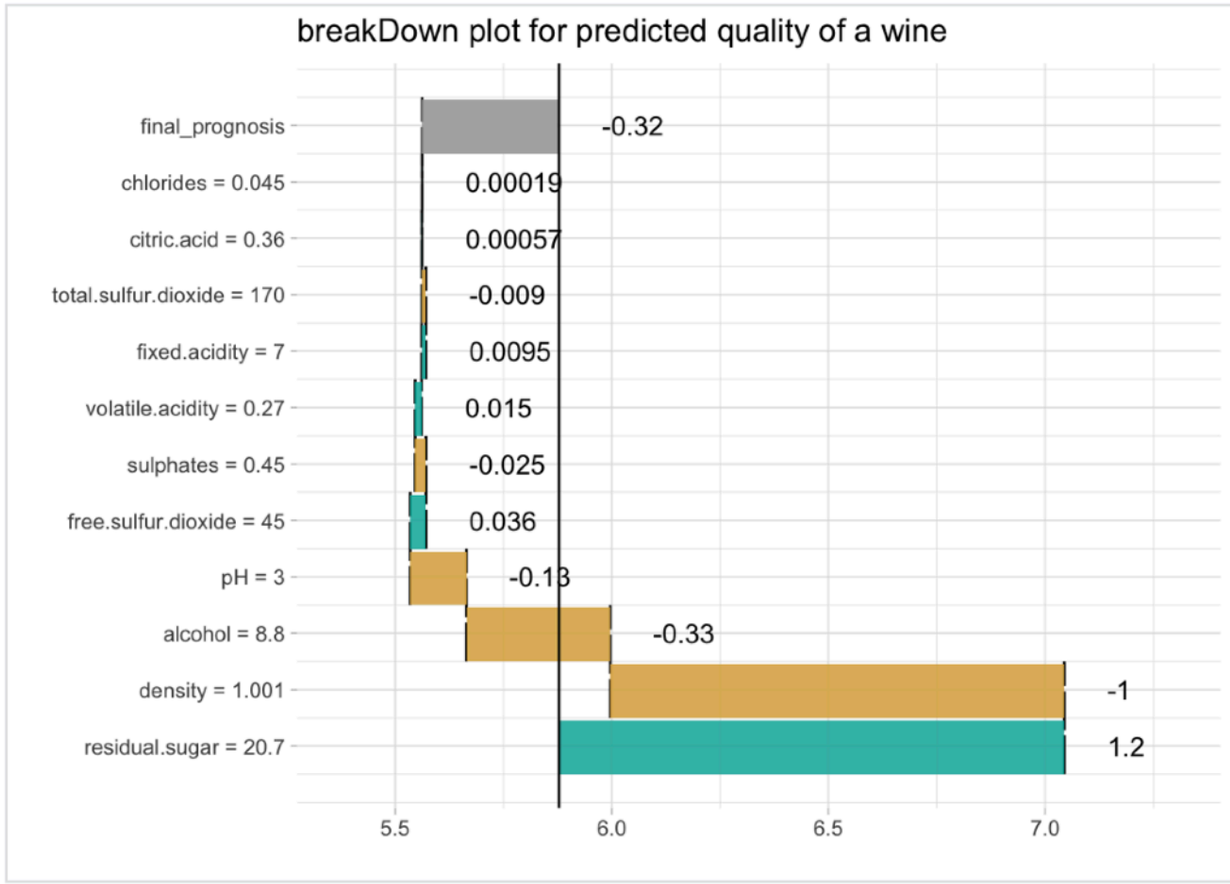
Linear model

Linear models are widely used in predictive modeling.
They have simple structure, which makes them easy to deploy or implement.
But models with many variables are hard to understand.

The **breakDown** plot explains the relation between variables and model prediction for a new observation.

Explanations are generated in three steps:
1) create model with **lm**() function
2) break down model predictions with the **broken()** function
3) plot the graphical summary with the generic **plot**() function.

```
library(breakDown)
library(ggplot2)
model <- lm(quality ~ ., data = wineQuality)
br <- broken(model, wineQuality[1,],
          baseline = "Intercept")
br
#>                                contribution
#> residual.sugar = 20.7               1.20000
#> density = 1.001                    -1.00000
#> alcohol = 8.8                      -0.33000
#> pH = 3                             -0.13000
#> free.sulfur.dioxide = 45            0.03600
#> sulphates = 0.45                   -0.02500
#> volatile.acidity = 0.27             0.01500
#> fixed.acidity = 7                   0.00950
#> total.sulfur.dioxide = 170         -0.00900
#> citric.acid = 0.36                  0.00057
#> chlorides = 0.045                   0.00019
#> final_prognosis                    -0.32000
#> baseline:  5.877909
plot(br)
```



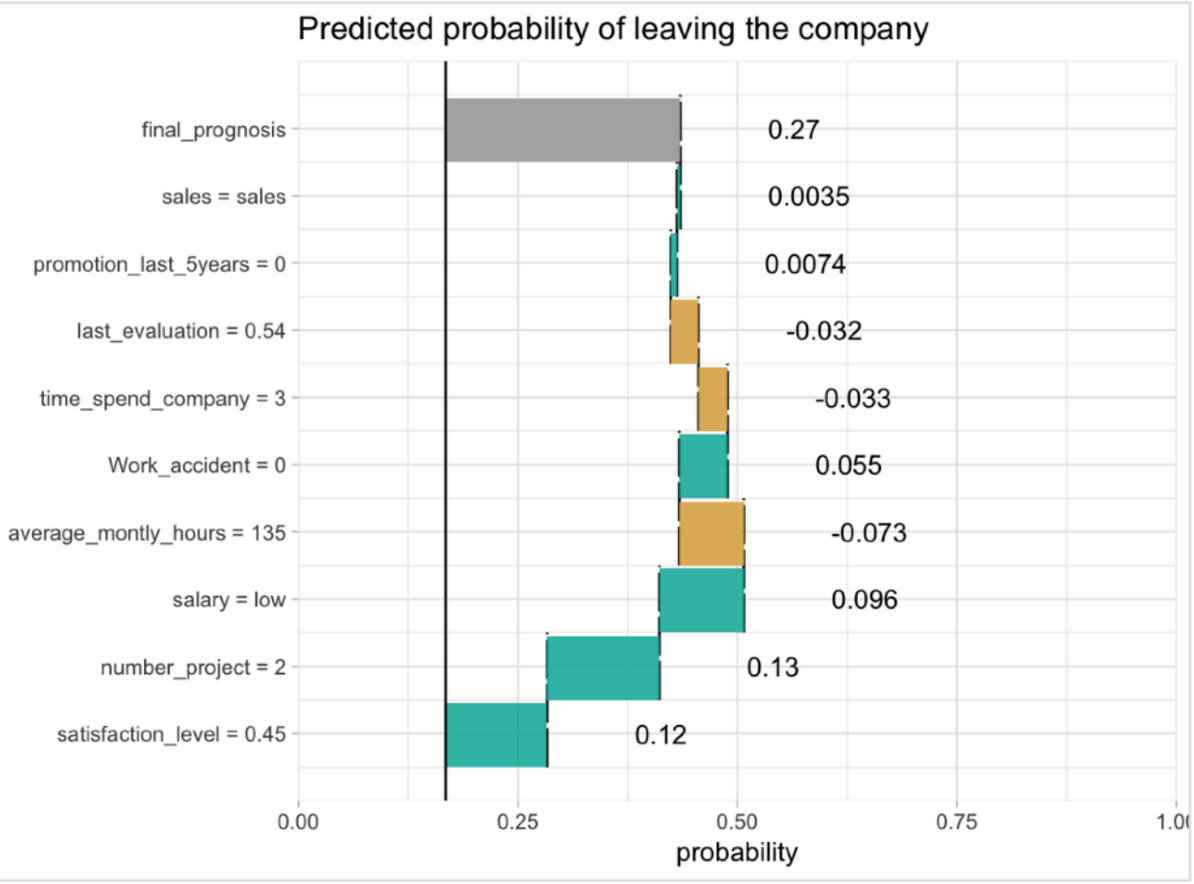
Logistic regression

breakDown plots may be also used to explain predictions from the logistic regression model.

On the OX axis one may present linear predictions (default) or use probit/logit transformation to present contributions of variables from the model. Use the trans= argument to define the transformation.

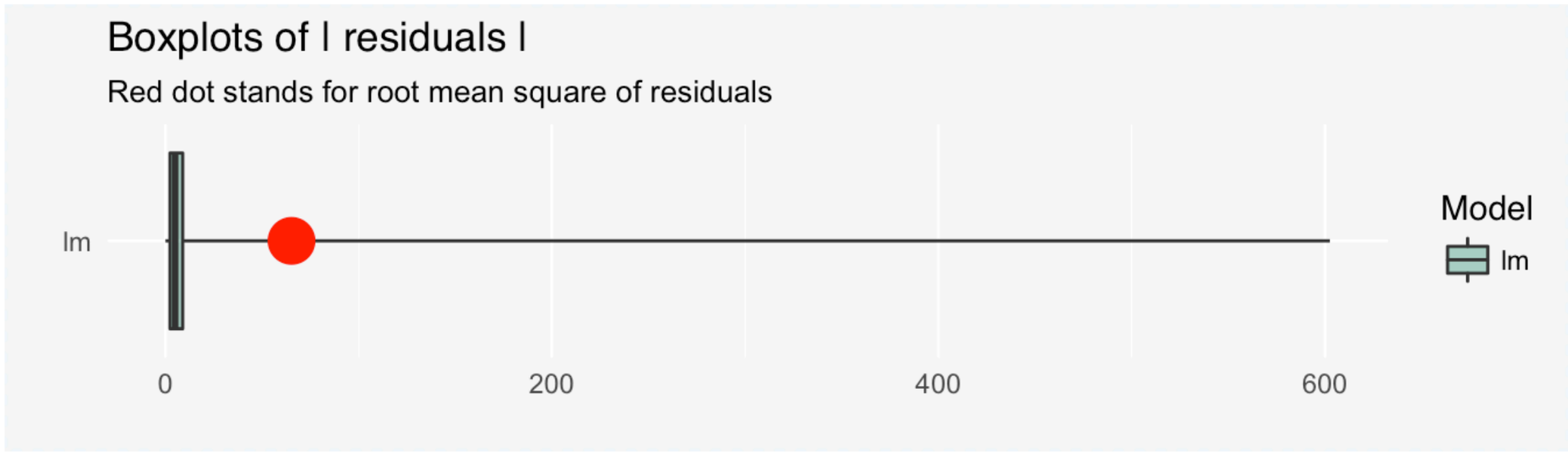
The baseline is presented by the vertical black line in the plot. One may set the baseline to 0 or to population average *(use the baseline = "intercept" argument).*

```
library(breakDown)
library(ggplot2)
model <- glm(left~., data = HR_data,
                family = "binomial")
explain_1 <- broken(model, HR_data[11,],
                baseline = "intercept")
explain_1
#>                                contribution
#> satisfaction_level = 0.45             0.670
#> number_project = 2                    0.570
#> salary = low                          0.390
#> average_montly_hours = 135           -0.290
#> Work_accident = 0                     0.220
#> time_spend_company = 3               -0.130
#> last_evaluation = 0.54               -0.130
#> promotion_last_5years = 0             0.030
#> sales = sales                         0.014
#> final_prognosis                       1.300
#> baseline:  -1.601457
plot(explain_1, trans = function(x)
              exp(x)/(1+exp(x)))
```

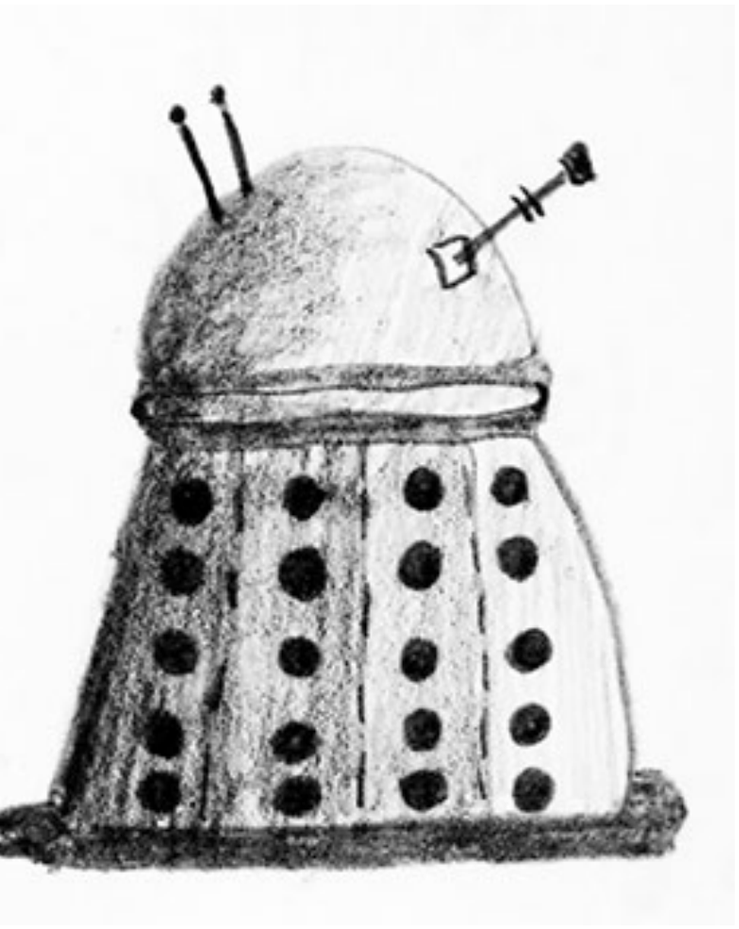


- Two models with equal performance
- Random forest model has smaller residuals than the linear model but there is a small fraction of very large residuals
- Random forest model under-predicts expensive apartments. It is not a model that we would like to employ
- Construction_year is important for the random forest model
- Relation between construction_year and price of square meter is non linear

```r
library("DALEX")

apartments_lm_model_improved <- lm(m2.price ~ I(construction.year < 1935 | construction.year > 1995
                        no.rooms + district, data = apartments)

explainer_lm_improved <- explain(apartments_lm_model_improved,
                         data = apartmentsTest[,2:6], y = apartmentsTest$m2.price)

mp_lm_improved <- model_performance(explainer_lm_improved)
plot(mp_lm_improved, geom = "boxplot")
```



Methodology & Tools



auditor

randomForestExplainer

live

archivist

factorMerger

RTCGA

GOpro

cr17

MLGenSig

cancerML

genetic signatures

Applications in medicine