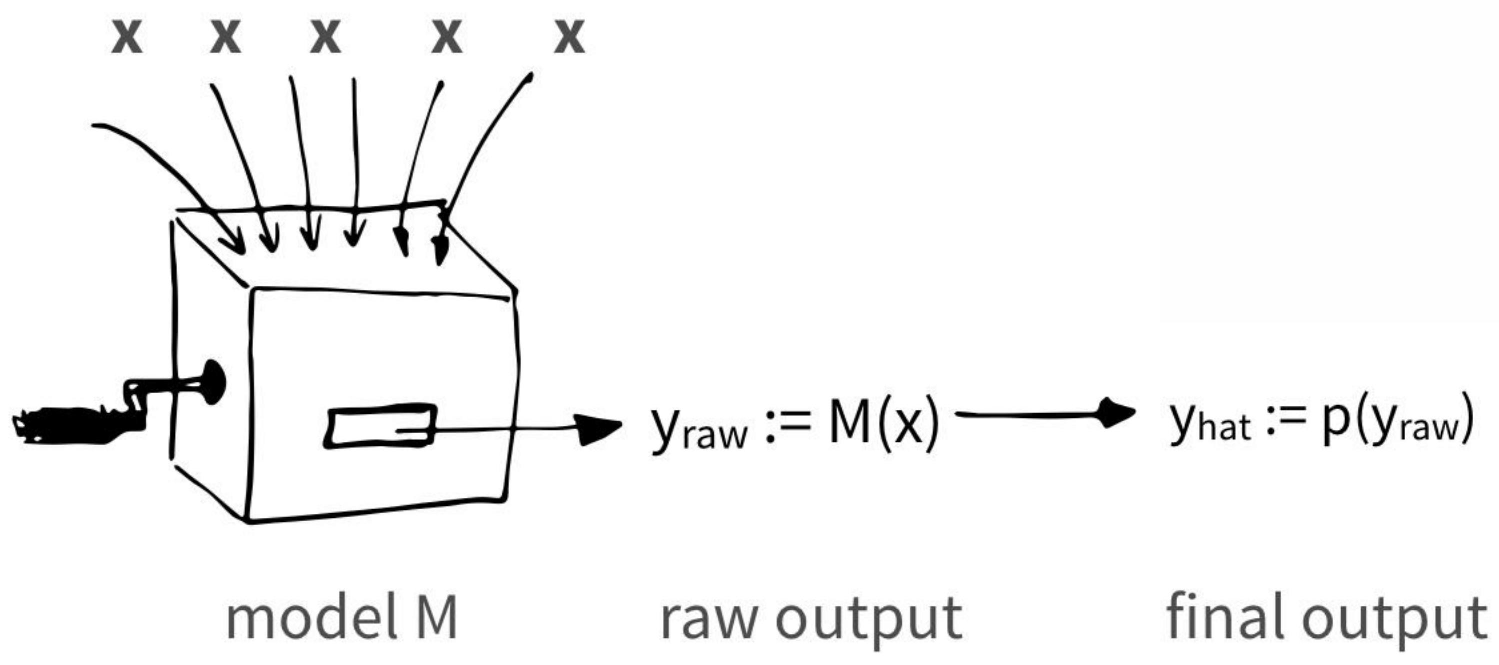
DALEX - Descriptive mAchine Learning EXplanations

DALEX explains black-box models. It's a methodology for better diagnostic of any black-box model.

This approach increases understanding of a model, increases trust in model predictions and allows to further improve the model. It also allows to compare two or more models in the scale space

Notation:

- **(x, y)** - pair of input and output data points. x may be anything (data.frame, factors, numbers, text, image), while here we assume that y is numerical or can be transformed to the numerical variable ($x \in X; y \in R$).
- **M** - a black box model, $M: X \to R$. Its output will be denoted as $y_{raw} = M(x)$
- **p** - a link function, transforms raw model output to the same space as y. Useful for classification, while for regression its usually the identity. $p: R \to R$. Its output will be denoted as $y_{hat} = p(y_{raw})$



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

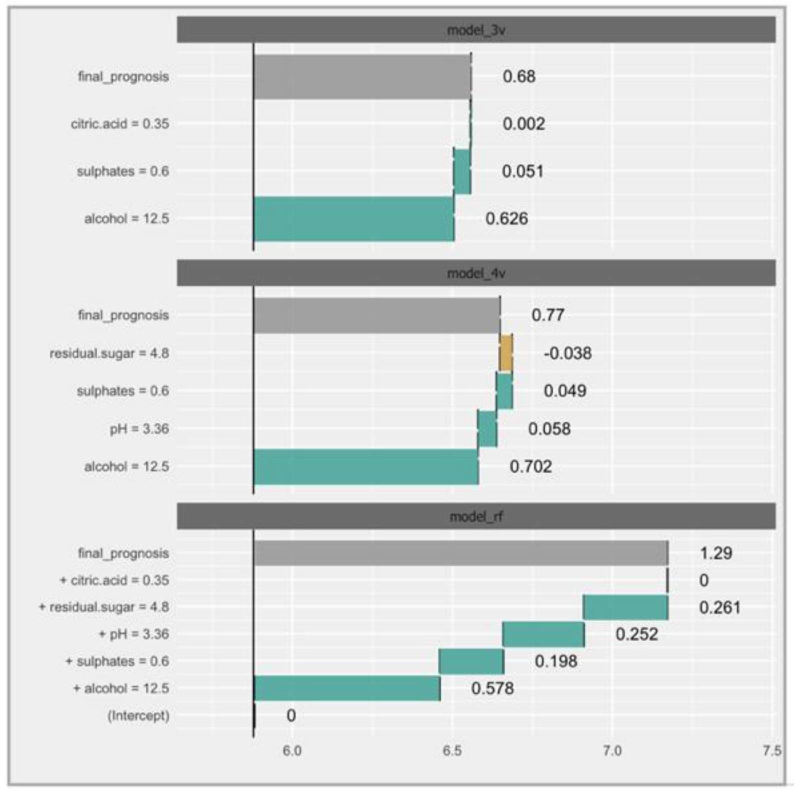
The *explain*() function creates a wrapper over a black-box model. This wrapper contains all necessary components for further processing.

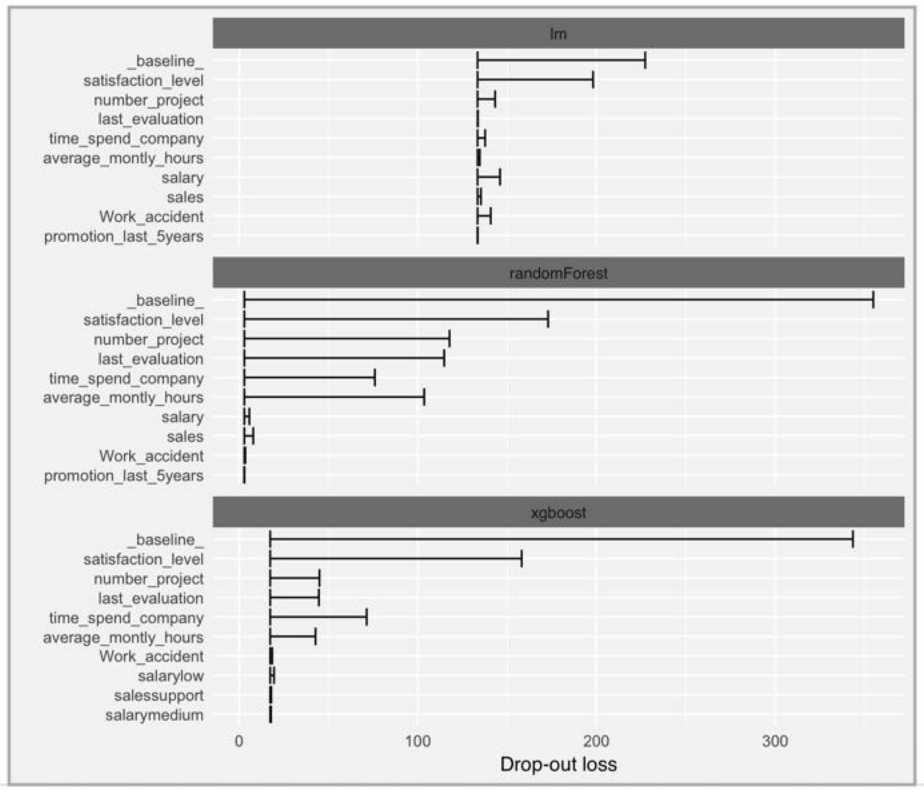
prediction_breakdown(explainer, x)   variable_importance(explainer)   variable_response(explainer, variable)

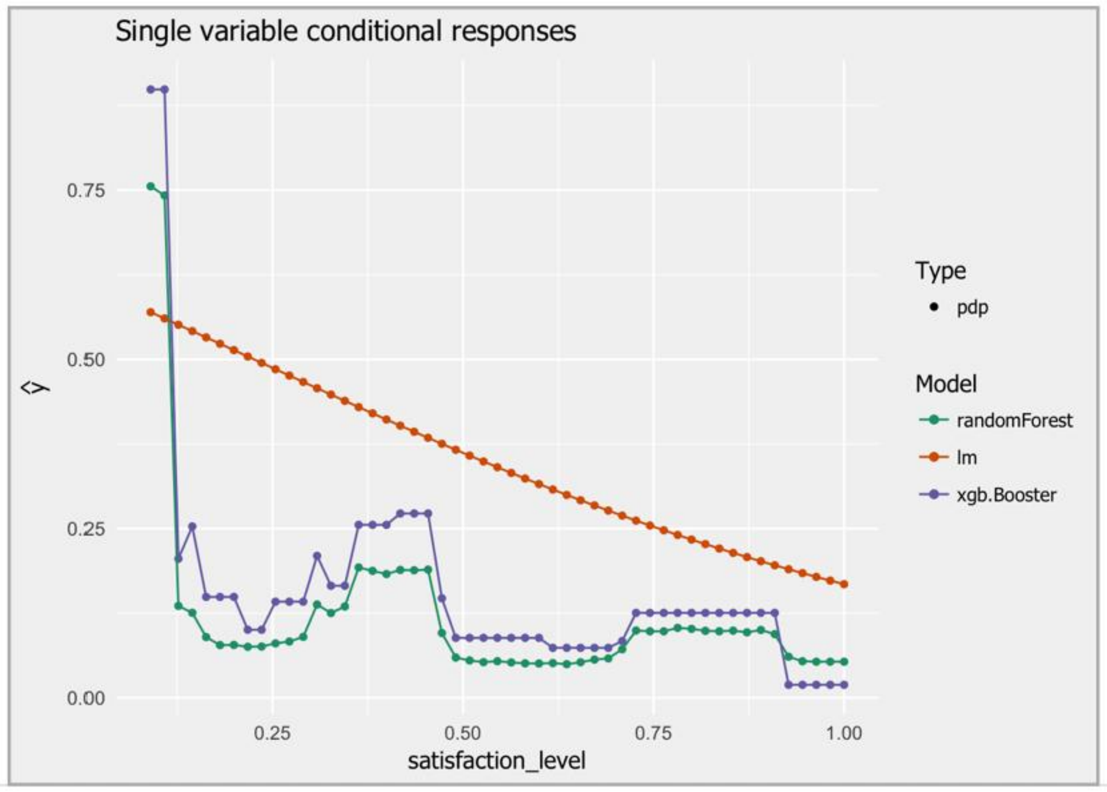
Prediction explainers shows features that drive model response for a selected observation

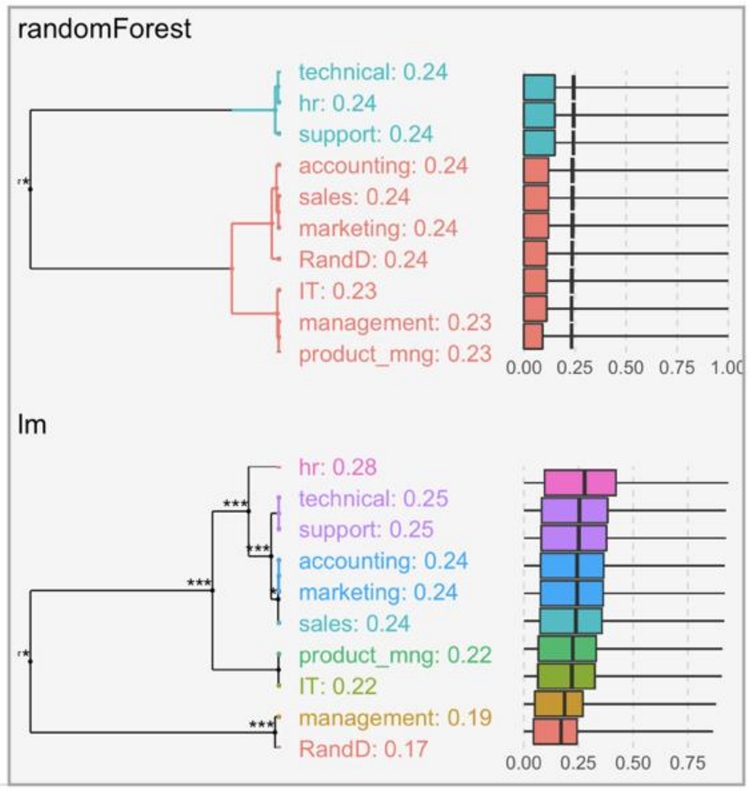
Variable importance explainers shows the drop in the model loss after permutations of a selected variable.

Single variable explainers show conditional relation between model output and a single variable.









Local Exploration, Explanation and Visualisation of Predictive Models

Introduction

When decisions from Machine Learning models are confronted with humans, following questions sparkle: Why? Why this decision was made? Which features support this decision? Can we trust this decision? How would it change if a single feature is slightly different?

Methods for local exploration/explanation are designed to improve out understanding of model predictions that refer to a particular observation.

Use-Case

As an use-case we are using **HR** dataset from the **DALEX** package. Five variables are used for a classification problem, would a given employee shall be fired, promoted or left as it is. It's an artificial dataset designed in a way that variable age and gender are in interaction.

```
library("DALEX")
head(HR, 2)
##   gender      age    hours evaluation salary   status
## 1   male 32.58267 41.88626          3      1    fired
## 2 female 41.21104 36.34339          2      5    fired
```

Presented tools are part of DALEXverse, set of tools for model agnostic exploration, explanation and visualisation of predictive models.

Find mode information about DALEX at the website https://pbiecek.github.io/DALEX_docs/ or online book https://pbiecek.github.io/PM_VEE/

Model preparation

Model exploration starts with a model to explore, and the observation of interest.

1. First we need to train a model

```
library("randomForest")
rf_model <- randomForest(
   status ~ gender + age + hours +
   evaluation + salary, data = HR)
```

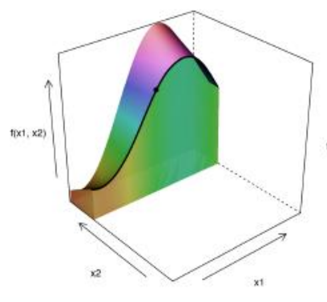
2. Then we need to build an explainer - model enriched with additional metadata like validation data, predict function, true labels. Use the **explain()** function from the **DALEX** package.

```
library("DALEX")
explainer_rf <- explain(rf_model,
   data = HR,
   y = HR$status == "fired")
```

3. Local explainers work for a selected observation / point in a feature space.

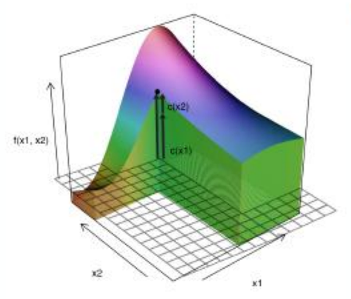
```
John1960 <- data.frame(
   gender = factor("male",
       levels = c("male","female")),
   age = 57.7,
   hours = 42.3,
   evaluation = 2,
   salary = 2)
```

Calculation of local explainers



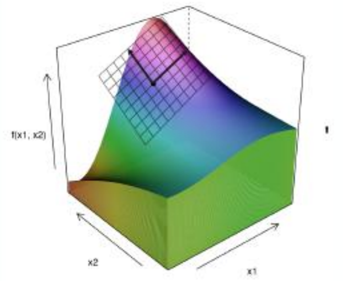
What-If scenarios with Ceteris Paribus Profiles are supported with the **ceterisParibus** package. unction. Use the function **ceteris_paribus()** with an explainer and the observation of interest as first arguments. You may also select variables of interest.

```
library("ceterisParibus")
cp_rf <- ceteris_paribus(explainer_rf, John1960,
    variables = c("age", "hours"))
cp_rf
## Top profiles    :
##      gender      age hours evaluation salary
## 1      male 20.00389  42.3          2      2
## 1.1    male 20.35994  42.3          2      2
##      _yhat_ _vname_ _ids_      _label_
## 1   0.4234617     age     1 randomForest
## 1.1 0.3761229     age     1 randomForest
```



Variable attributions are supported with the **breakDown** package.

```
library("breakDown")
bd_rf <- break_down(explainer_rf, John1960)
bd_rf
##                          contribution
## (Intercept)                     0.364
## * hours = 42                    0.161
## * age:gender = 58:male          0.120
## * salary = 2                   -0.045
##  final_prognosis                0.648
```



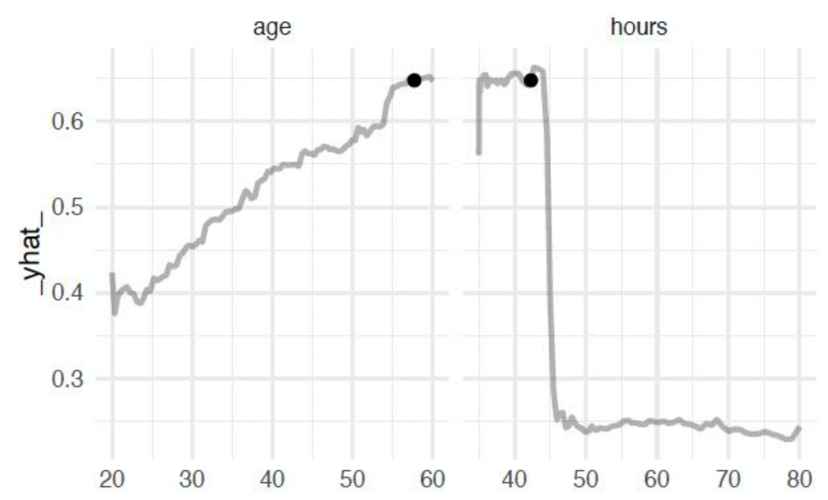
Local model approximation is supported with the **live** package.

```
library("live")
lm_rf <- local_approximation(explainer_rf,
   John1960, target_variable_name = "status")
lm_rf
## Explanation model:
## Name:   regr.lm
## R-squared: 0.9889
```

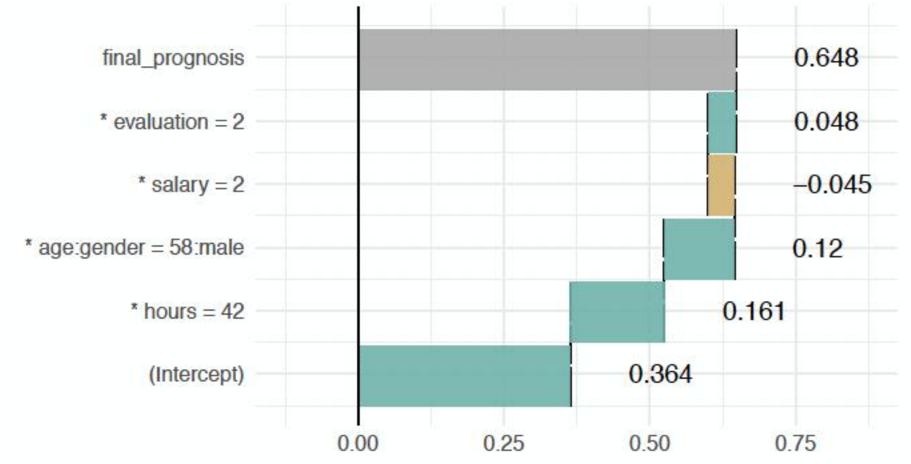
Visualisation of explainers

The generic function **plot()** works for every local explainer.

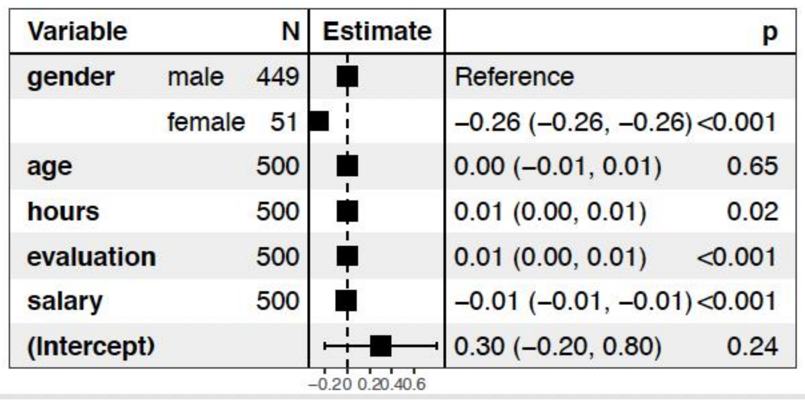
```
plot(cp_rf)
```



```
plot(bd_rf)
```



```
plot(lm_rf)
```



Variable		N	Estimate		p
gender	male	449	■	Reference	
	female	51	■	–0.26 (–0.26, –0.26)	<0.001
age		500	■	0.00 (–0.01, 0.01)	0.65
hours		500	■	0.01 (0.00, 0.01)	0.02
evaluation		500	■	0.01 (0.00, 0.01)	<0.001
salary		500	■	–0.01 (–0.01, –0.01)	<0.001
(Intercept)			■	0.30 (–0.20, 0.80)	0.24

DALEX: : prediction_breakdown() - Explanations for a Single Prediction

Basics

Black-box models, like random forest or extreme gradient boosting machines, are commonly used due to their high performance. They are very flexible, what often results in high accuracy.

The problem is, that due to their complicated structure it is hard to understand which variables were the most influential for a particular model prediction

Single prediction explainers are designed to decompose model prediction into parts that may be attributed for separate variables.

How does it work?
For linear models the idea is simple. Having the estimated of model coefficients the variable contributions are calculated as

$$\begin{aligned} \hat{y}^{new} &= baseline + \\ & (x_1^{new} - \bar{x}_1)\hat{\beta}_1 + \\ & ... + \\ & (x_p^{new} - \bar{x}_p)\hat{\beta}_p \end{aligned}$$

where

$$baseline = \hat{\mu} + \bar{x}_1\hat{\beta}_1 + ... + \bar{x}_p\hat{\beta}_p$$

Note that variables are centred in order to have contributions calculated against the total average.

For generalised linear models we explain the linear component of the model.

For non linear models we used a method implemented in the breakDown package that assess the variable contribution based on relaxed distances between model predictions. See more details in the breakDown package vignettes.

References
- Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. 2016. *"'Why Should I Trust You?': Explaining the Predictions of Any Classifier."* In, 1135–44. ACM Press.
- Staniak, Mateusz, and Przemysław Biecek. 2017. *Live: Local Interpretable (Model-Agnostic) Visual Explanations.* https://github.com/MI2DataLab/live
- Biecek, Przemyslaw. 2017. *BreakDown: BreakDown Plots.* https://CRAN.R-project.org/package=breakDown.

Use-Case

Why do you think that this will be a good wine? Let's use explanations for models prediction to explain factors that influences quality of wine. Data from UCI repository *https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/* .

Here we are using linear model to link physicochemical properties of a wine with it's quality. Then the model is used to predict quality for a wine with selected properties.

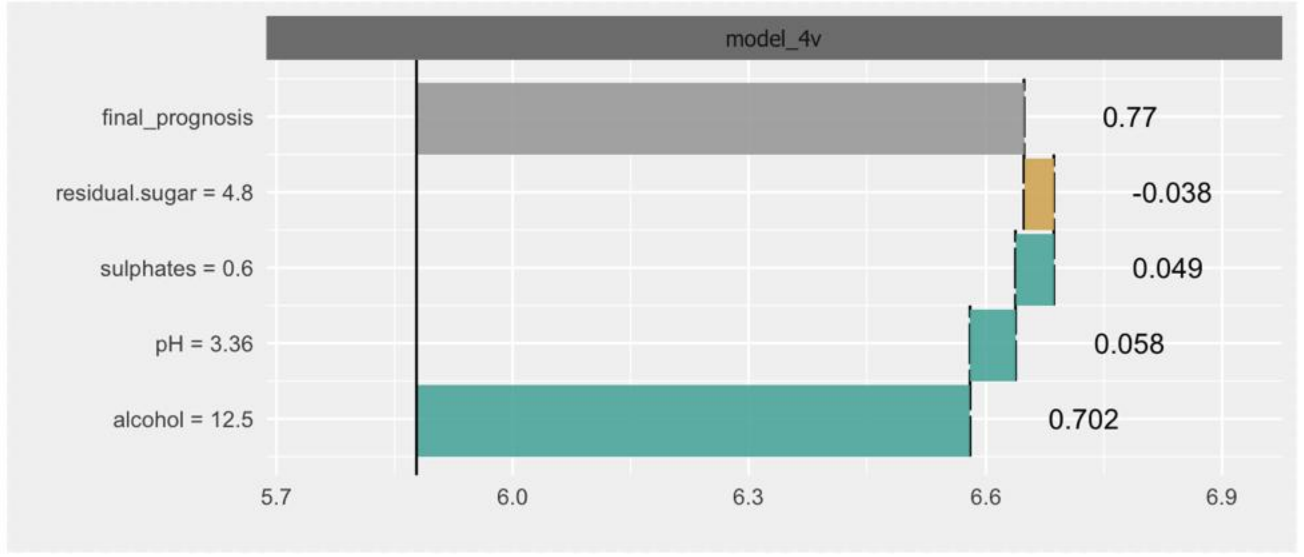
```
library("breakDown")
library("DALEX")
new.wine <- data.frame(citric.acid = 0.35,
                       sulphates = 0.6,
                       alcohol = 12.5,
                       pH = 3.36,
                       residual.sugar = 4.8)
wine_lm_model4 <- lm(quality ~ pH + residual.sugar +
                       sulphates + alcohol, data = wine)
predict(wine_lm_model4, newdata = new.wine)
## 6.648226
```

The predicted quality is high, but which properties of the wine was key for this prediction? Let's use the DALEX package to check this.
First we need a general model explainer for *wine_lm_model4* model.

```
wine_lm_explainer4 <- explain(wine_lm_model4,
                       data = wine, label = "model_4v")
```

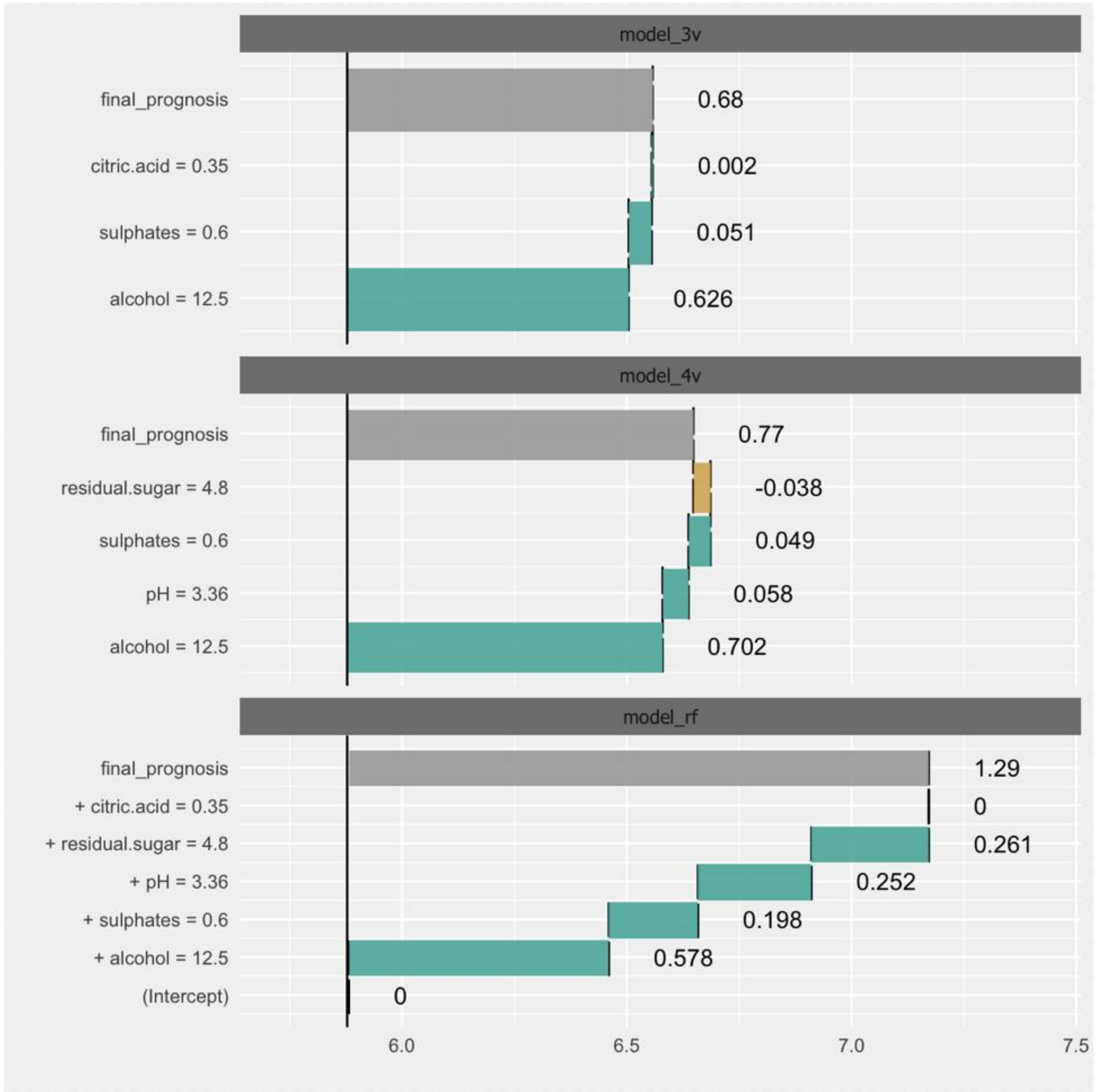
Now we can use the **prediction_breakdown()** function to decompose model prediction into a parts that can be attributed to consecutive variables.
For linear models it's easy. In this case it's just a wrapper over the *predict* function with *type = "terms"* parameter. For nonlinear model we are using the breakDown package.
Once the decomposition is made one can use the *plot* function to present key components with the Break Down Plot.
Here we see that high alcohol concentration was crucial for model prediction.

```
wine_lm_pred4 <- prediction_breakdown(wine_lm_explainer4,
                      observation = new.wine)
plot(wine_lm_pred4)
```



A very useful feature of the DALEX package is the capability to overlay responses from different models in a single plot.
Below we present results for a Random Forest model and two linear models.

```
lm_model3 <- lm(quality ~ citric.acid + sulphates +
                alcohol, data = wine)
lm_explainer3 <- explain(lm_model3, data = wine)
lm_predict3 <- prediction_breakdown(lm_explainer3,
                 observation = new.wine)
rf_model4 <- randomForest(quality ~ pH + residual.sugar +
                 sulphates + alcohol, data = wine)
rf_explainer4 <- explain(rf_model4, data = wine)
rf_predict4 <- prediction_breakdown(rf_explainer4,
                observation = new.wine)
plot(rf_predict4, wine_lm_pred4, lm_predict3)
```



DALEX: : variable_response() - Single Variable Conditional Responses

Basics

Black-box models, like random forest or extreme gradient boosting machines, are commonly used due to their high performance. They are very flexible, what often results in high accuracy.

The problem is, that due to their complicated structure it is hard to understand if and how a single variable affect model response.

Single variable explainers are designed to explain or at least interpolate the conditional effect of a single variable.

How does it work?

The idea is simple, for a selected variable x^i we would like to know what is the average output of the model as a function of this variable

$$E_{x_{-i}}[f(x^i, x^{-i}; \theta)]$$

But since we cannot calculate the expected value directly we need to estimate it.

1. Partial Dependency Plots [PDP] estimate this response as an average model response on all data points with overridden x^i.
2. Accumulated Local Effects Plots [ALEP] estimate the response as a cumulative sum of model responses calculated on data points similar to x^i under consideration.
3. Individual Conditional Expectations are model responses calculated for data points created from single observation with replaced x^i values.

Right now there are two types of single variable explainers implemented in the DALEX package: Partial Dependency Plots and Accumulated Local Effects Plots.

References

- Brandon Greenwell, *pdp: An R Package for Constructing Partial Dependence Plots*, The R Journal 9(2017), no. 1
- Apley, Dan. *ALEPlot: Accumulated Local Effects (Ale) Plots and Partial Dependence (Pd) Plots* (2017)
- Goldstein, Alex, Adam Kapelner, Justin Bleich, and Emil Pitkin. *Peeking Inside the Black Box: Visualizing Statistical Learning with Plots of Individual Conditional Expectation* Journal of Computational and Graphical Statistics (2015) 24 (1): 44–65.
- Sitko, Agnieszka, and Przemyslaw Biecek. *FactorMerger: Hierarchical Algorithm for Post-Hoc Testing (2017)* https://github.com/MI2DataLab/factorMerger.

Use-Case

Why are our best and most experienced employees leaving prematurely? Let's see with a dataset from Kaggle Human Resources competition https://www.kaggle.com/ludobenistant/hr-analytics/data.

Below we will explain how a single variable *satisfaction_level* affect the model outcome. First we will train a Random Forest classifier

```
library("randomForest")
library("breakDown")
HR_rf_model <- randomForest(left~., data = HR_data,
                    ntree = 500)
# getit: archivist::aread("pbiecek/DALEX/arepo/a79f3c7e")
```

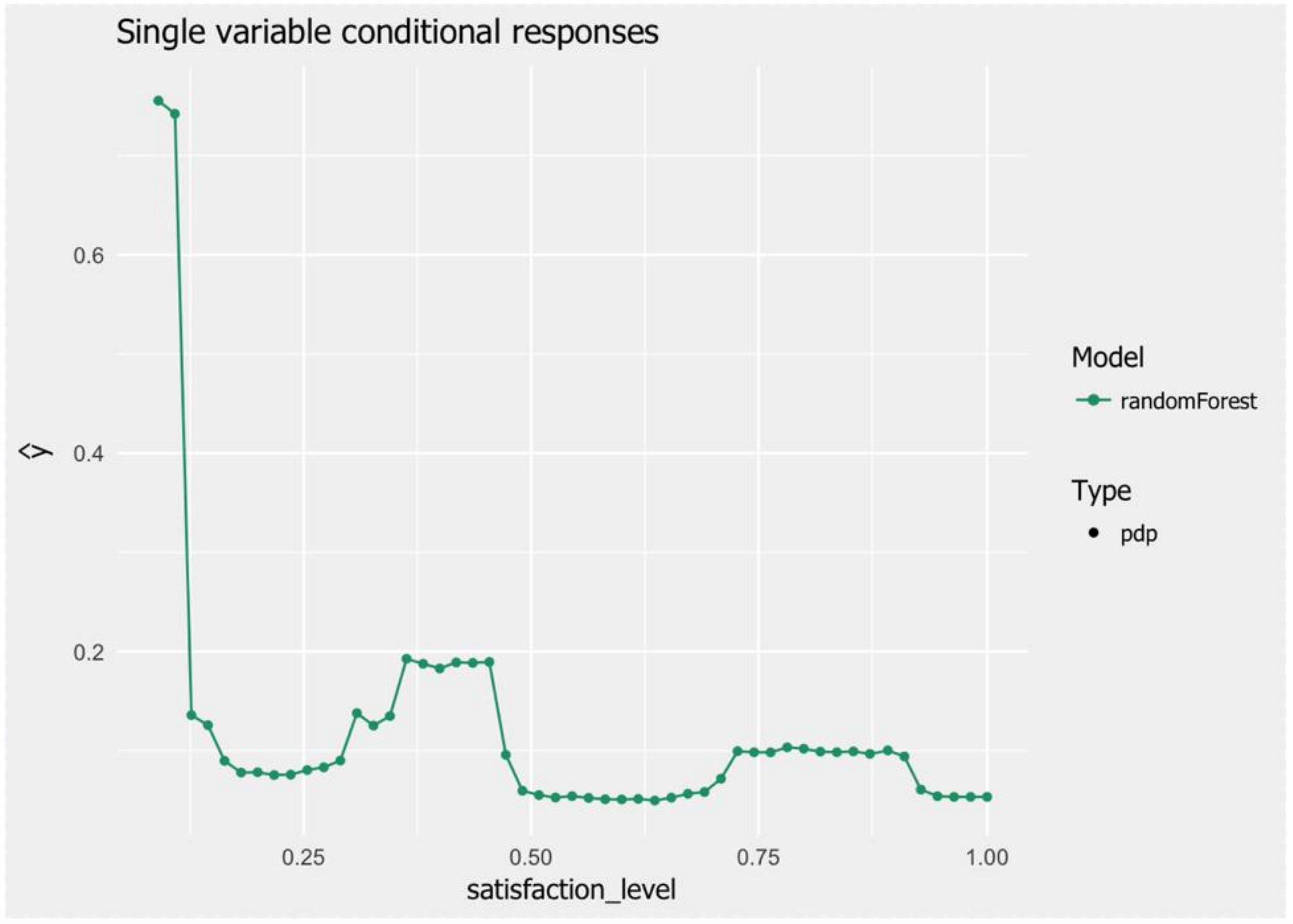
In order to generate explanations we need to convert the model into a uniform structure readable by the DALEX package. To do so we need to use the **explain()** function

```
library("DALEX")
explainer_rf  <- explain(HR_rf_model, data = HR_data)
```

Single variable explainer can be created with the **variable_response()** function. It's result can be printed or plotted. Use the type= parameter to choose PDP/ALEP effects.

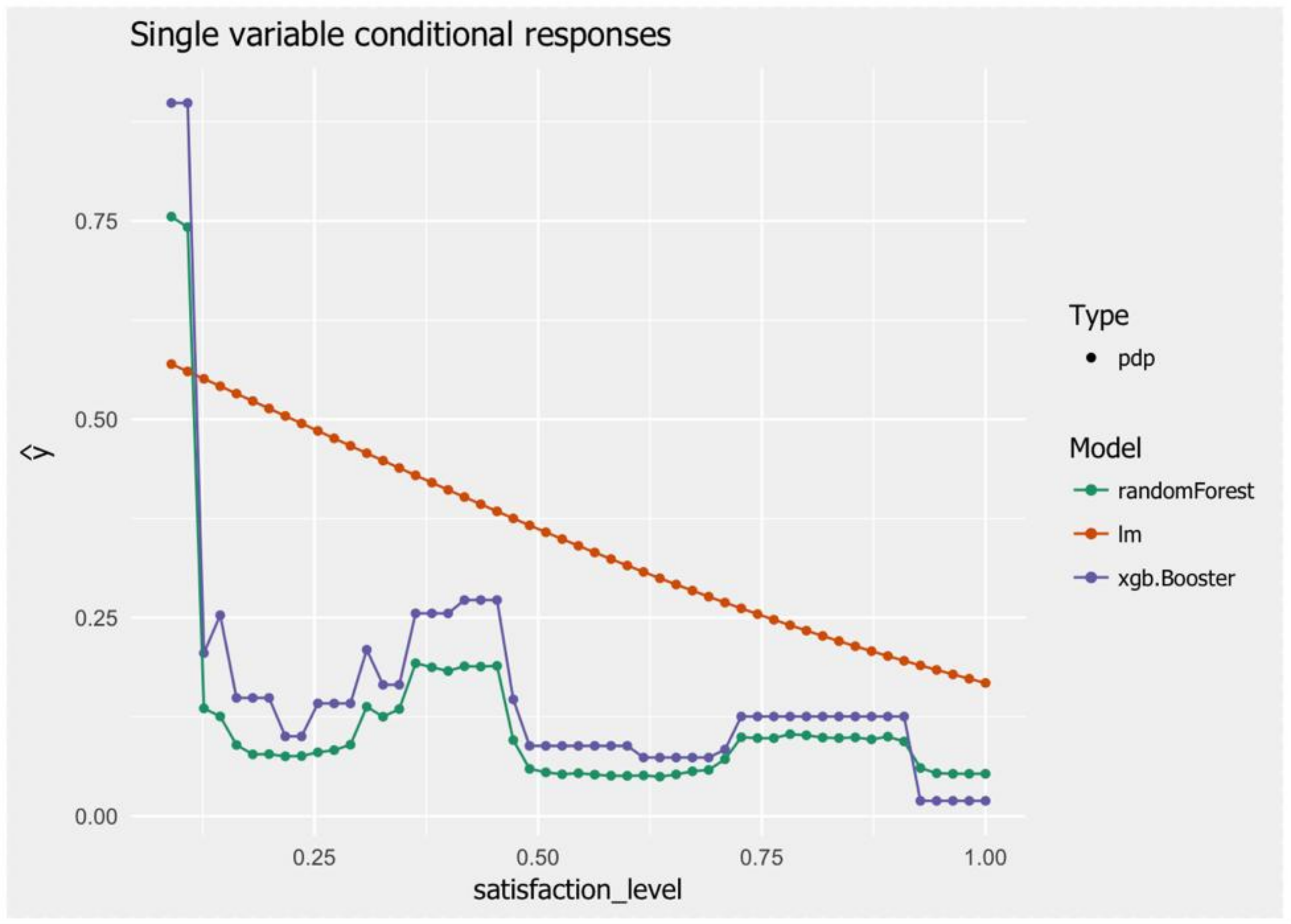
```
expl_rf  <- variable_response(explainer_rf,
            variable = "satisfaction_level",type = "pdp")
plot(expl_rf)
# getit: archivist::aread("pbiecek/DALEX/arepo/ad0f1699")
```

Result is a ggplot2 plot which present conditional profile of model responses as a function of satisfaction_level variable.



A very useful feature of the DALEX package is the possibility to overlay responses from different models in a single plot.
Below we present results for Random Forest, XGBoost model and Logistic regression model in a single plot. As we see XGB and RF are close to each other while GLM tries to find linear link.

```
library("xgboost")
model_martix_train <- model.matrix(left~.-1, HR_data)
data_train <- xgb.DMatrix(model_martix_train,
                 label = HR_data$left)
param <- list(max_depth=2, objective="binary:logistic")
HR_xgb_model <- xgb.train(param, data_train, nrounds=50)
# GLM model
explainer_glm <- explain(HR_glm_model, data=HR_data)
expl_glm <- variable_response(explainer_glm,
             "satisfaction_level", "pdp")
#XGB model
explainer_xgb <- explain(HR_xgb_model,
                  data = model_martix_train)
expl_xgb <- variable_response(explainer_xgb,
            "satisfaction_level", "pdp")
# plot results
plot(expl_rf, expl_glm, expl_xgb)
```



DALEX: : variable_importance() - Explanations for Variable Importance

Basics

Black-box models, like random forest or extreme gradient boosting machines, are commonly used due to their high performance. They are very flexible, what often results in high accuracy.

The problem is, that due to their complicated structure it is hard to understand which variables were the most influential for a particular model prediction

Variable Importance Explainers are designed to assess the influence of a single variable on the final model accuracy.

How does it work?
The concept is straightforward, calculate how much we will loose on accuracy if a selected variable is perturbed.

For a selected loss function the model loss is calculated for the trained model applied to the original dataset against the original target. Then the drop in model loss is calculated for each variable in the dataset. It is calculated as the loos for a model applied to a dataset with selected column being permuted.

The variable importance plots shows the initial loss of the trained model, size of the additional losses that come from permutations of selected variables and the 'baseline' which is the loss for permuted model responses.

It is useful when different model are compared, since on a single plot we see the initial model performance and also drops that come from permutations of a single variable in the selected model.

References

- Molnar, Christoph. 2018. *Interpretable Machine Learning*. https://christophm.github.io/
- Breiman, Leo. 2001. *Random Forests.* Machine Learning 45 (1). Springer: 5–32.
- Fisher, Aaron, Cynthia Rudin, and Francesca Dominici. 2018. *Model Class Reliance: Variable Importance Measures for any Machine Learning Model Class, from the 'Rashomon' Perspective.* http://arxiv.org/abs/1801.01489.

Use-Case

Why are our best and most experienced employees leaving prematurely? Let's see with a dataset from Kaggle Human Resources competition https://www.kaggle.com/ludobenistant/hr-analytics/data.

Here we are building a random forest model. The nice thing about this model is that it embeds some variable importance measure.

```
library("randomForest")
HR_rf_model <- randomForest(left~., data =
breakDown::HR_data, ntree = 100)
importance(HR_rf_model)
                      IncNodePurity
satisfaction_level       896.584411
last_evaluation          294.195269
number_project           496.397115
average_montly_hours     398.495343
```

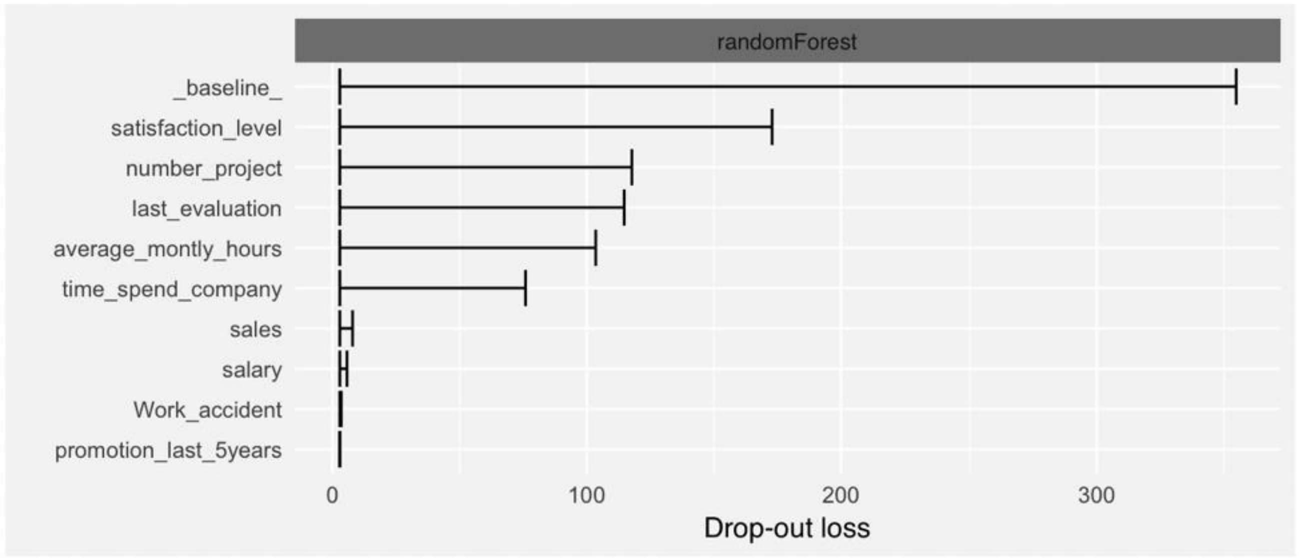
So, now let's build an explainer and use it to calculate our model agnostic variable importance measure with **variable_importance()** function.

```
explainer_rf  <- explain(HR_rf_model, data = HR_data,
                     y = HR_data$left)
vd_rf <- variable_importance(explainer_rf, type = "raw")
vd_rf
                variable dropout_loss        label
              _baseline_   335.556063 randomForest
      satisfaction_level   168.001733 randomForest
         last_evaluation   100.714534 randomForest
          number_project   100.568214 randomForest
       time_spend_company   90.349207 randomForest
```

Values are different because different measures of importance are being calculated, but the order is more or less the same. Now we are ready to plot the variable importance with the generic **plot**() function.

Note that in addition to *type="raw"* one can also use *"difference"* or *"ratio"* as suggested in the Fisher et all 2018 article.

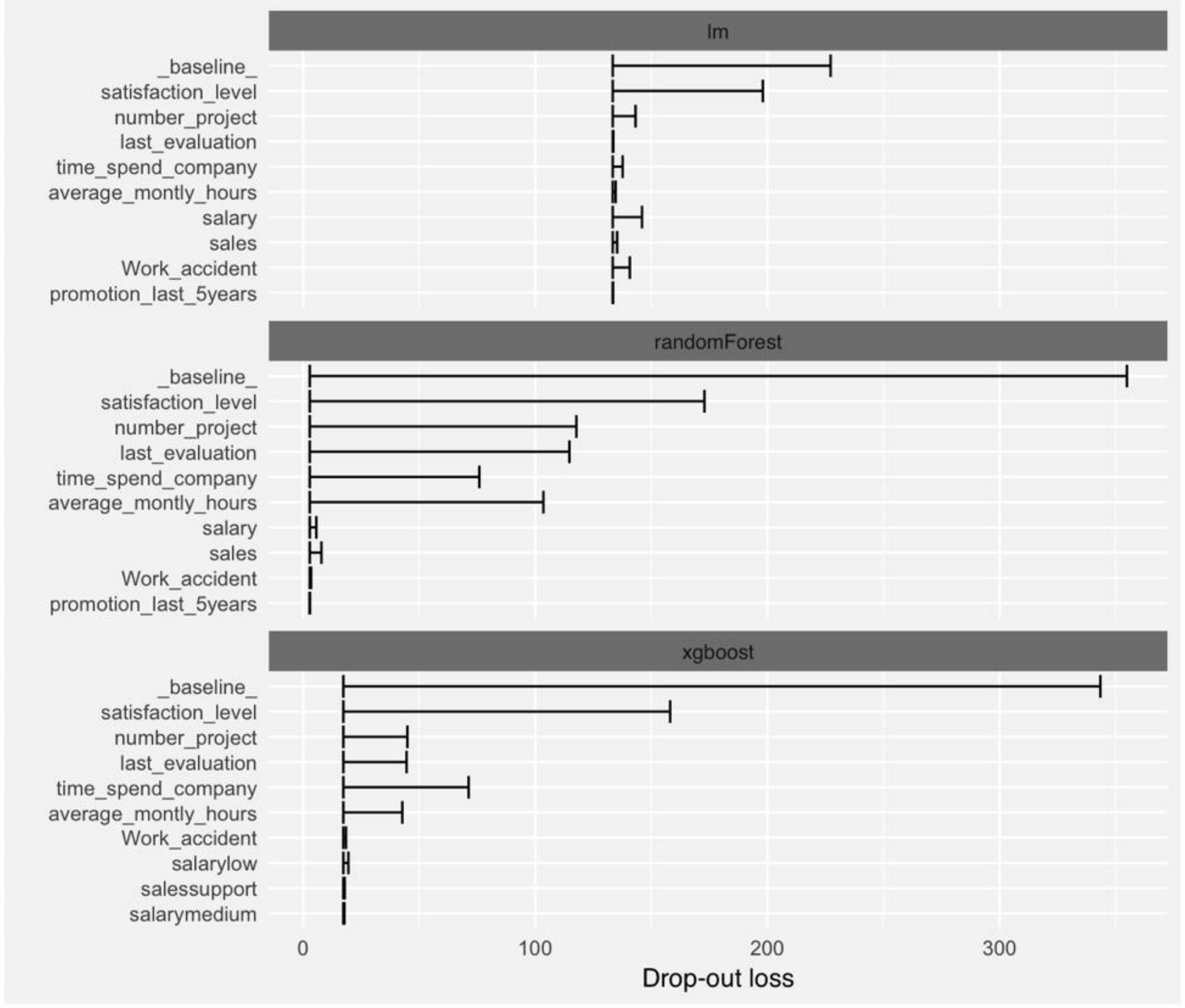
```
plot(vd_rf)
```



A very useful feature of the DALEX package is the capability to overlay responses from different models in a single plot.
Below we present results for a Random Forest, GLM and XGBoost.

```
library("xgboost")
mm_train <- model.matrix(left~.-1, HR_data)
data_train <- xgb.DMatrix(model_martix_train,
               label = HR_data$left)
param <- list(max_depth = 2,
      objective = "binary:logistic", eval_metric = "auc")
HR_xgb <- xgb.train(param, data_train, nrounds = 50)
ex_xgb <- explain(HR_xgb,  data = mm_train,
                y = HR_data$left, label = "xgboost")
vd_xgb <- variable_importance(ex_xgb, type = "raw")

HR_glm <- glm(left~., data=HR_data, family = "binomial")
ex_glm <- explain(HR_glm, data=HR_data, y = HR_data$left)
logit <- function(x) exp(x)/(1+exp(x))
vd_glm <- variable_importance(ex_glm, type = "raw")
plot(vd_rf, vd_glm, vd_xgb)
```


lm
baseline
satisfaction_level
number_project
last_evaluation
time_spend_company
average_montly_hours
salary
sales
Work_accident
promotion_last_5years
randomForest
baseline
satisfaction_level
number_project
last_evaluation
time_spend_company
average_montly_hours
salary
sales
Work_accident
promotion_last_5years
xgboost
baseline
satisfaction_level
number_project
last_evaluation
time_spend_company
average_montly_hours
Work_accident
salarylow
salessupport
salarymedium
Drop-out loss