Prediction understanding

Mateusz Staniak

Outline

1. Intro & `breakDown` package.
2. Local approximations: `live` package.
3. Summary & state of the art solutions (LIME and Shapley values)

Intro & prediction breakdown

Why explain a single prediction?

(Bird's-eye view)

- when important decision are made based on ML model, it needs to be **trustworthy**
- trust comes from **understanding**
- the demand for interpretable algorithms is growing (see: *Weapons of math destruction*, Facebook feed controversies etc.)

(Worm-eye view)

- this demand is transfered into legal regulations (see: RODO)

=> more and more institutions have to explain model predictions (debt collection, loans ...)

- understanding models helps improve them

Which predictions need explanation?

1. Every prediction the client (or the boss) wants to understand
2. Predictions that seem suspicious
    - How to spot them?
    - How to explain them?

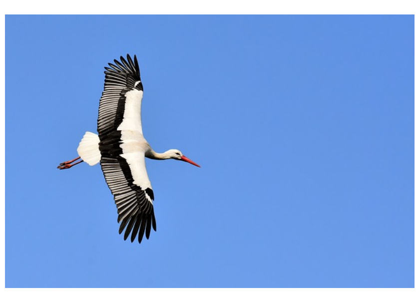


Figure 1:

Figure 2:

=> model performance

=> model diagnostics

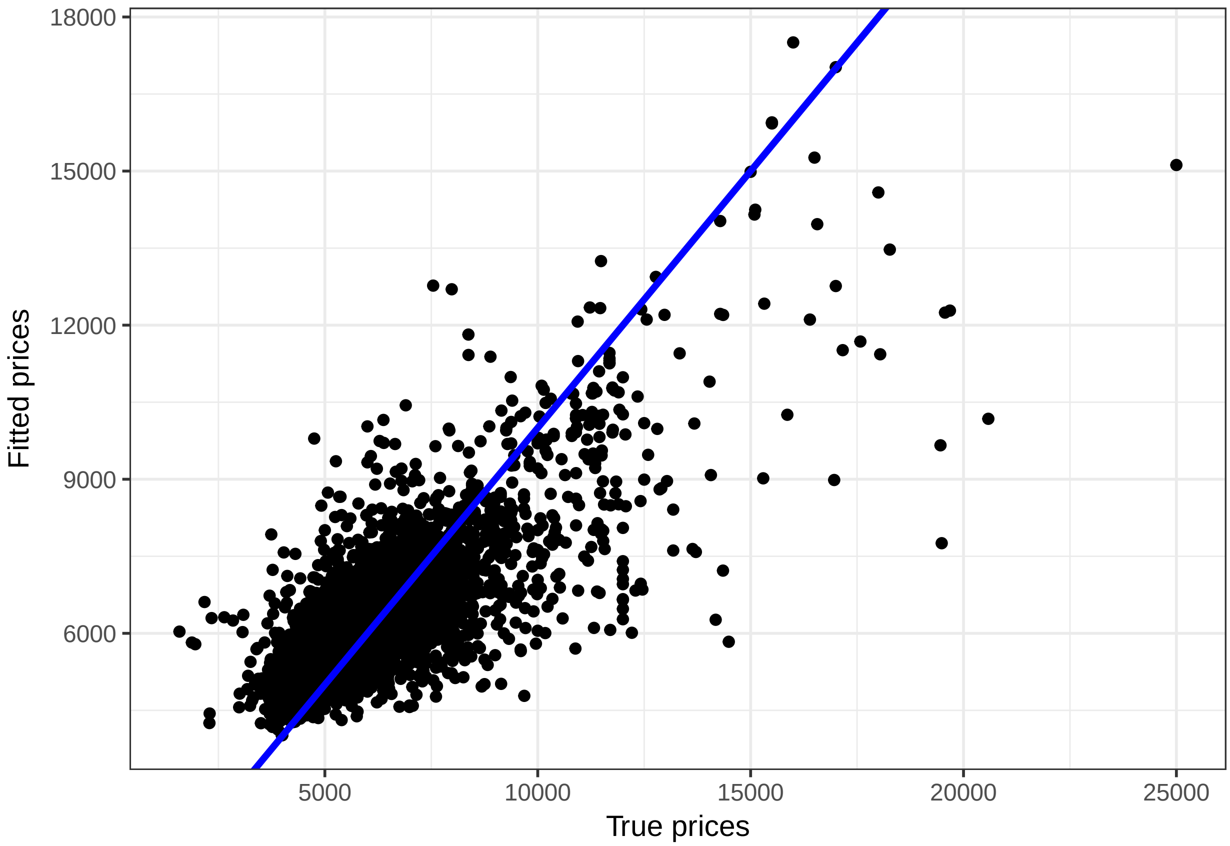
Model diagnostics: example data

```
library(tidyverse)
library(randomForest)
load("./rda_files/houses.rda")
houses
```

```
## # A tibble: 5,851 x 7
##    rooms_num  area sqm_price  year floor max_floor district
##        <int> <dbl>     <int> <int> <int>     <int> <fct>
##  1         3  89.0      5270  2007     2         2 Krzyki
##  2         4 163.       6687  2002     2         2 Psie Pole
##  3         3  52.0      6731  1930     1         2 Srodmiescie
##  4         4  95.0      5525  2016     1         2 Krzyki
##  5         4  88.0      5216  1930     3         4 Srodmiescie
##  6         2  50.0      5600  1915     3         4 Krzyki
##  7         2  48.0      9146  2010     2         6 Fabryczna
##  8         2  46.0      9761  2011     4         6 Stare Miasto
##  9         3  67.0      5209  1990     1        10 Fabryczna
## 10         2  45.8      6878  2000     3         4 Krzyki
## # ... with 5,841 more rows
```

```
hrf <- randomForest(sqm_price ~., data = houses)
```

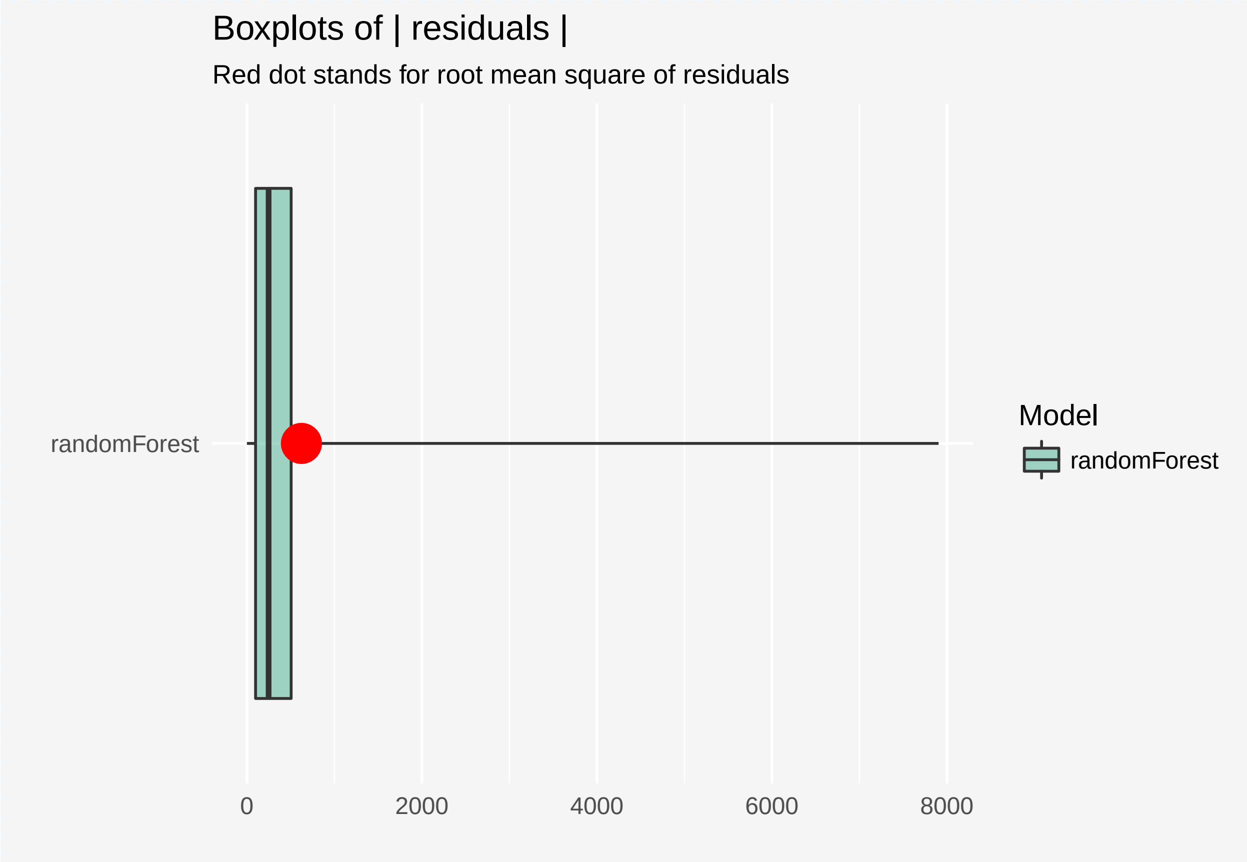
Model diagnostics: predicted vs observed values



- Points on the plot should be close to the `y = x` line,
- Questions:
  - is there a pattern? (For example: does the devation from true value grow as true value grows?)
  - are there any points especially far from the line (meaning: points with large residuals)?
- More diagnostic tools: `auditor` package

Model performance

```
library(DALEX)
rf_explainer <- explain(hrf, data = houses, y = houses$sqm_price)
rf_perf <- model_performance(rf_explainer)
plot(rf_perf, geom = "boxplot")
```



- shortly summarizes the distribution of the absolute value of residuals
- red dot is the root mean square error
- we can put boxplots for several models on the same plot (simply by passing the as arguments to `plot`) to compare models
- boxplots help discover outliers

Single prediction explanation

- Once we identified predictions we want to explain, we need tools that will help us!
- Methods:
  - LIME
  - Shapley values
  - Break Down
  - LIVE
- Two main ideas:
  - Attribute scores to explanatory variables according to their influence on the prediction (**contributions**)
  - Fit a model locally around an observation and investigate it
  - NOTE: both approaches lead to local feature importance

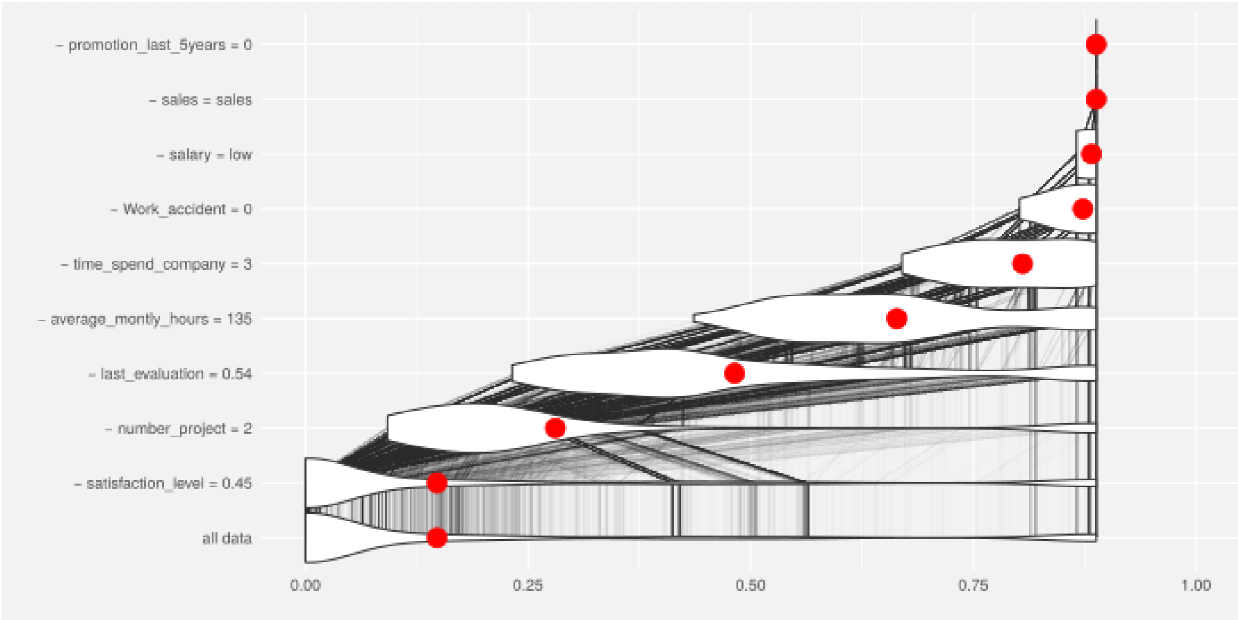


Figure 3:

- Contributions: Shapley values and Break Down
- Local models: LIME and LIVE

Break Down

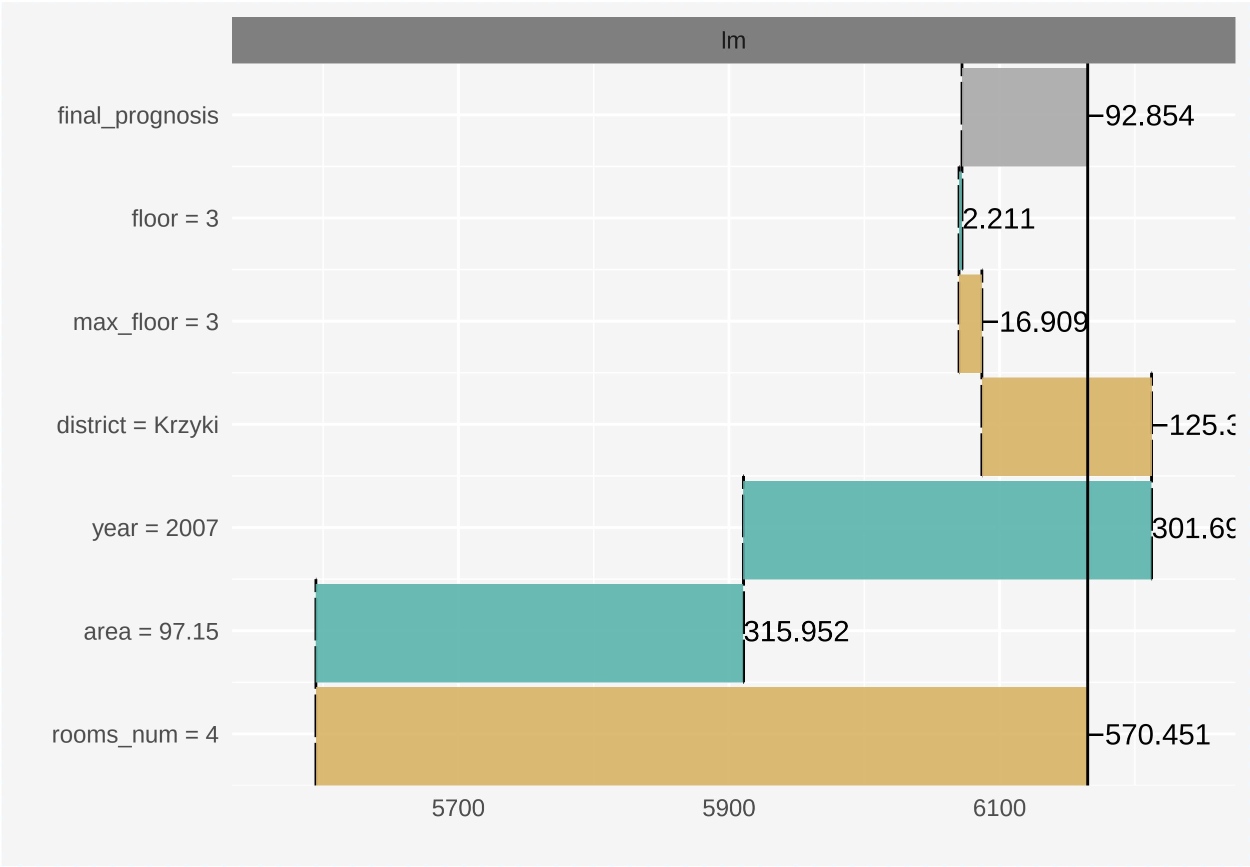
- General idea
- Another approach to finding *additive* feature contributions
- Contributions are assigned in a greedy manner
- Waterfall plots as a visual tool

=> more intuitive interpretation

Example

- Break Down for linear models

```
linear_model <- lm(sqm_price ~., data = houses)
lm_explainer <- DALEX::explain(linear_model, data = houses, y = houses$sqm_price)
breakdown_linear <- single_prediction(lm_explainer, houses[4036, -3])
plot(breakdown_linear)
```

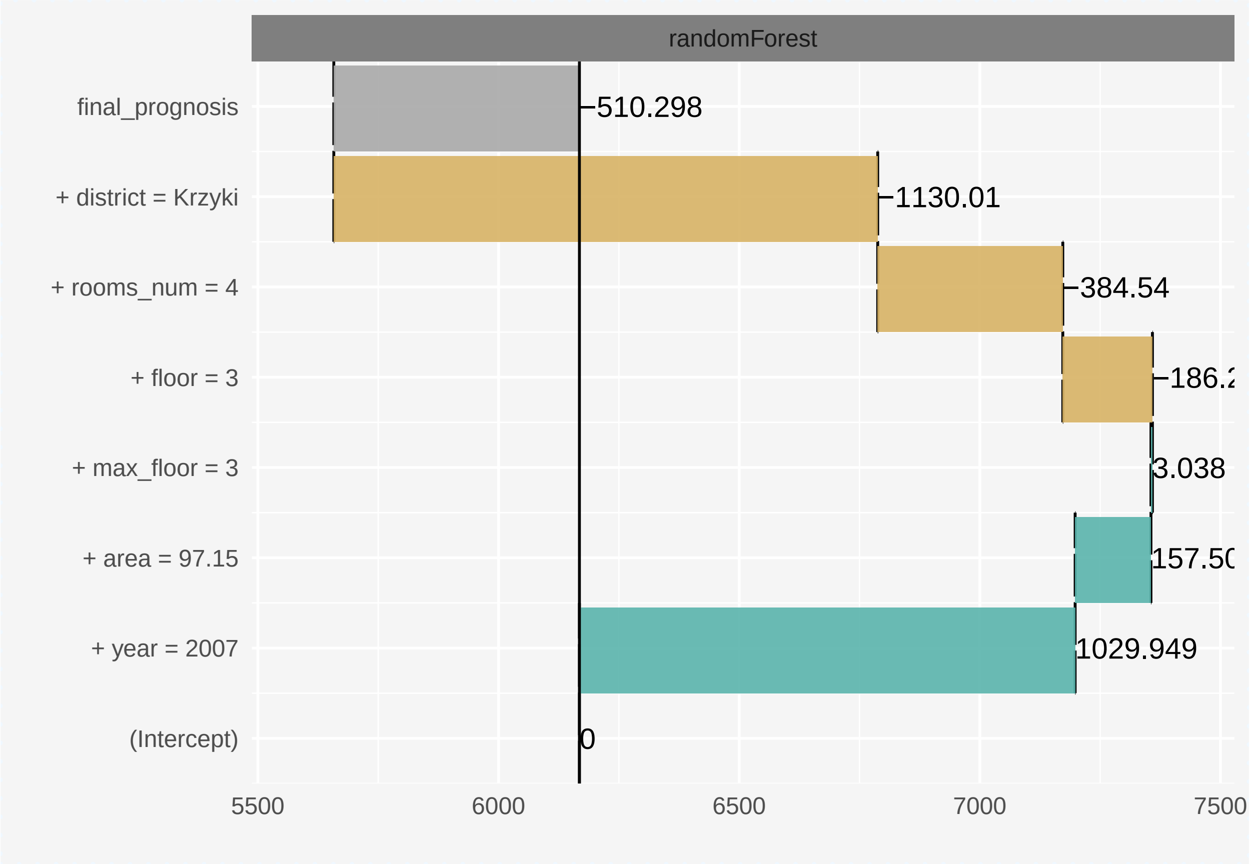


- Contributions are scaled, so they do not depend on the scale of the data (insensitive to location/scal

- We can see actual contributions, not just the weights (as in LIME plots)

- Model-agnostic Break Down

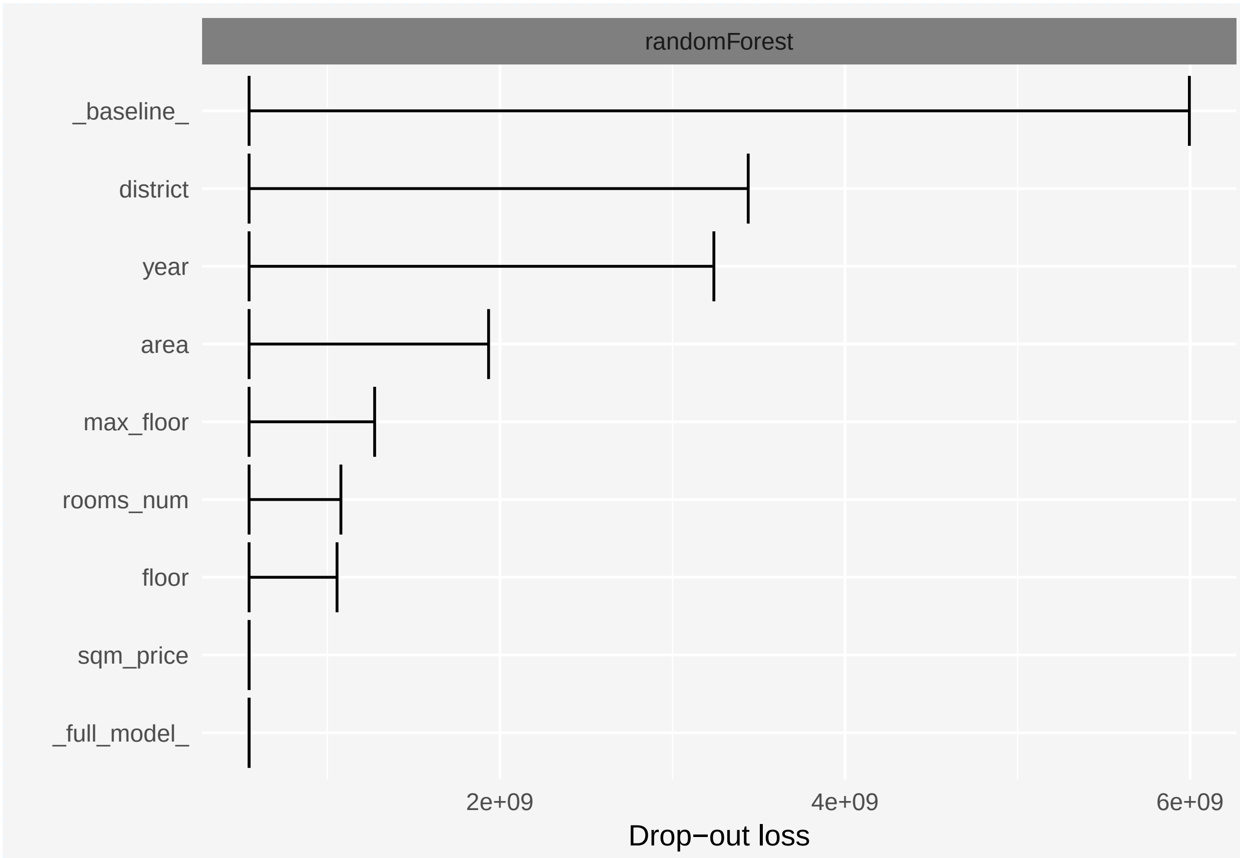
```
breakdown_explanation <- single_prediction(rf_explainer, houses[4036, -3])
plot(breakdown_explanation)
```



- We can see how important District and Year are in this random forest prediction

But isn't it enough to calculate feature importance?

```
global_feat_imp <- DALEX::variable_importance(rf_explainer)
plot(global_feat_imp)
```



- No. Particular instances can be influenced the most by different features, not necessarily the ones that are most important globally.

Time to practice!

First set of exercises goes here

Part II: local approximations

LIVE (Local Interpretable Visual Explanations)

- General idea
  - Modification of `LIME` for tabular data and regression problems with emphasis on model visualization.
  - Similar observations for *fake* dataset are sampled from empirical distributions.
  - Variable selection is possible (LASSO, then explanation model is fitted to selected features).
- More details
  - Two methods of creating the new dataset are available: by permuting each variable and by changing one feature per observations
  - We can control which variables are allowed to vary through `fixed` variable argument to `sample_locally` (keeping date/factor/correlated variables unchanged)

Example

```
library(live)
library(mlr)
new_dataset <- sample_locally2(data = houses,
                               explained_instance = houses[4036, ],
                               explained_var = "sqm_price",
                               size = 1000)
with_predictions <- add_predictions2(new_dataset, hrf)
live_explanation <- fit_explanation2(with_predictions, "regr.lm")
live_explanation
```

```
## Dataset:
##  Observations:  1000
##  Variables:  7
##  Response variable:  sqm_price
## Explanation model:
##  Name:  regr.lm
##  Variable selection wasn't performed
##  Weights present in the explanation model
##  R-squared: 0.9889
```

Aktualnie standaryzacja zmiennych jest niepotrzebnie w sample_locally, jak to zmienię, forestplot będzie lepiej wyglądał, bo będzie po standaryzacji

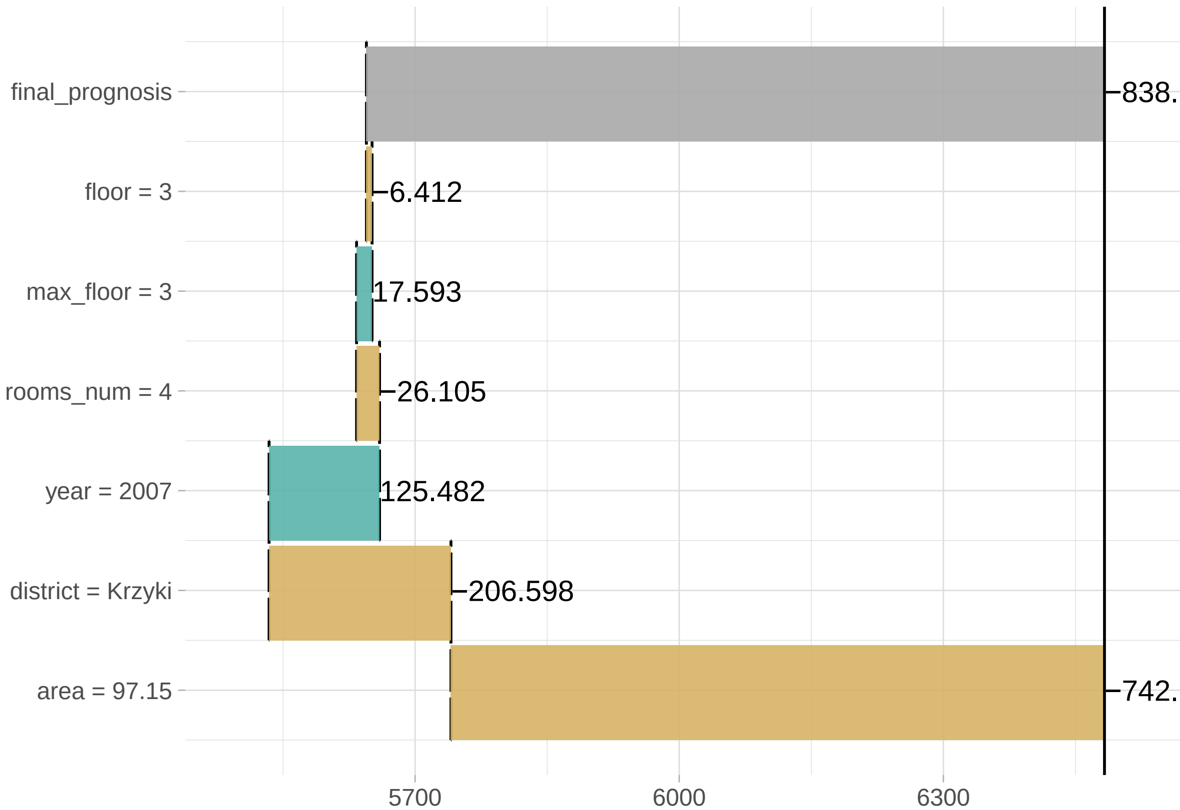
- Default method of sampling is *live*, default explanation model is linear regression and distance is measured (weights are assigned) by gaussian kernel.
- Plot local model structure: forest plot

```
plot_explanation2(live_explanation, "forest")
```

Variable		N	Estimate		p
rooms_num		1000		−118.12 (−135.56, −100.68)	<0.001
area		1000		−144.53 (−270.30, −18.75)	0.024
year		1000		39.25 (−8.76, 87.26)	0.109
floor		1000		−80.15 (−99.78, −60.52)	<0.001
max_floor		1000		−150.37 (−170.09, −130.65)	<0.001
district	Fabryczna	41		Reference	
	Krzyki	886		39.67 (14.55, 64.80)	0.002
	Psie Pole	19		−571.74 (−635.10, −508.38)	<0.001
	Srodmiescie	30		1749.10 (1696.60, 1801.61)	<0.001
	Stare Miasto	24		7062.97 (7005.53, 7120.41)	<0.001
(Intercept)				−57964.87 (−155073.63, 39143.90)	0.242

- Plot local variable contributions: waterfall plot (Break Down)

```
plot_explanation2(live_explanation, "waterfall")
```



Time to practice!

Second set of exercises goes here

Part III: state-of-the-art solutions & summary

LIME (Locally Interpretable Model-agnostic Explanations)

- General idea
- Some details:
  - Gaussian sampling for tabular, uniform sampling from interpretable inputs for image/text.
- Scores for new observation are weighted by the distance from original observation.
- Variable selection is usually based on ridge/lasso regression.
- Weights are assigned to interpretable inputs to decide if they *vote* for or against a given label.
- Note: method depends on many hyperparameters

Example

```
library(mlr)
house_task <- makeRegrTask(data = houses, target = "sqm_price")
house_rf_mlr <- train("regr.randomForest", house_task)
```

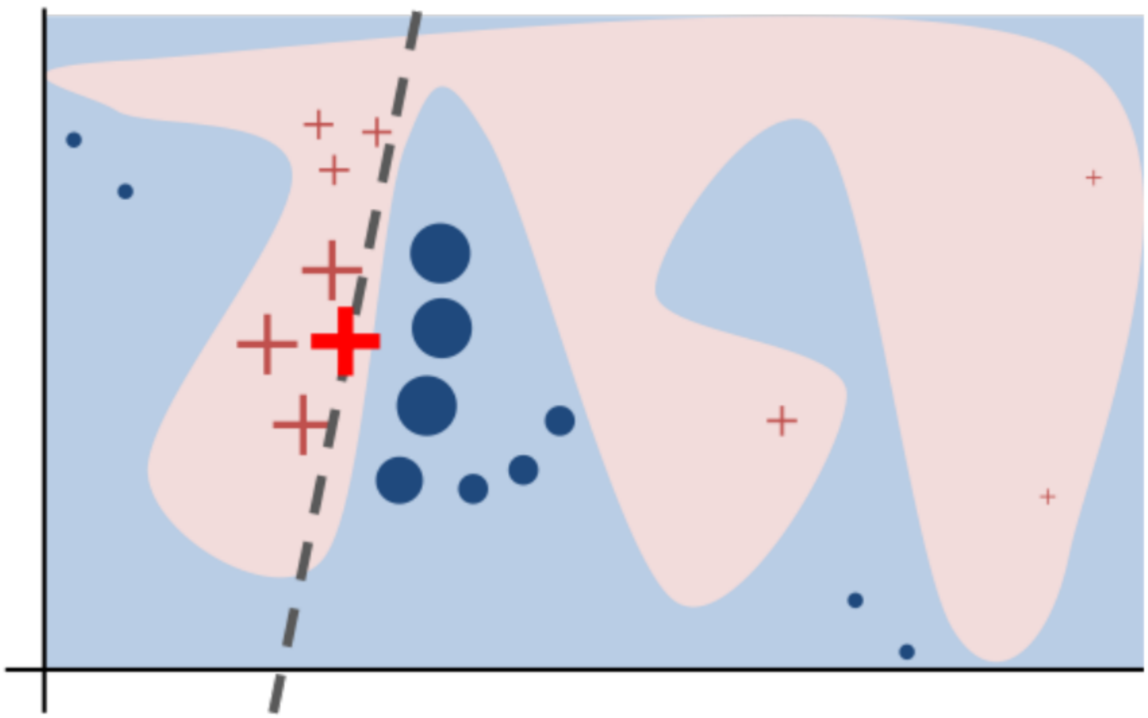


Figure 4:

```r
library(lime)
explained_prediction <- houses[4036, ]
lime_explainer <- lime(houses,
                       model = house_rf_mlr)
lime_explanation <- lime::explain(houses[4036, ],
                                  explainer = lime_explainer,
                                  n_features = 5)
```

- weights from ridge regression are on the plot (NOT weights multiplied by actual feature values)
- positive weights are *for*, negative weights are *against*

Shapley values

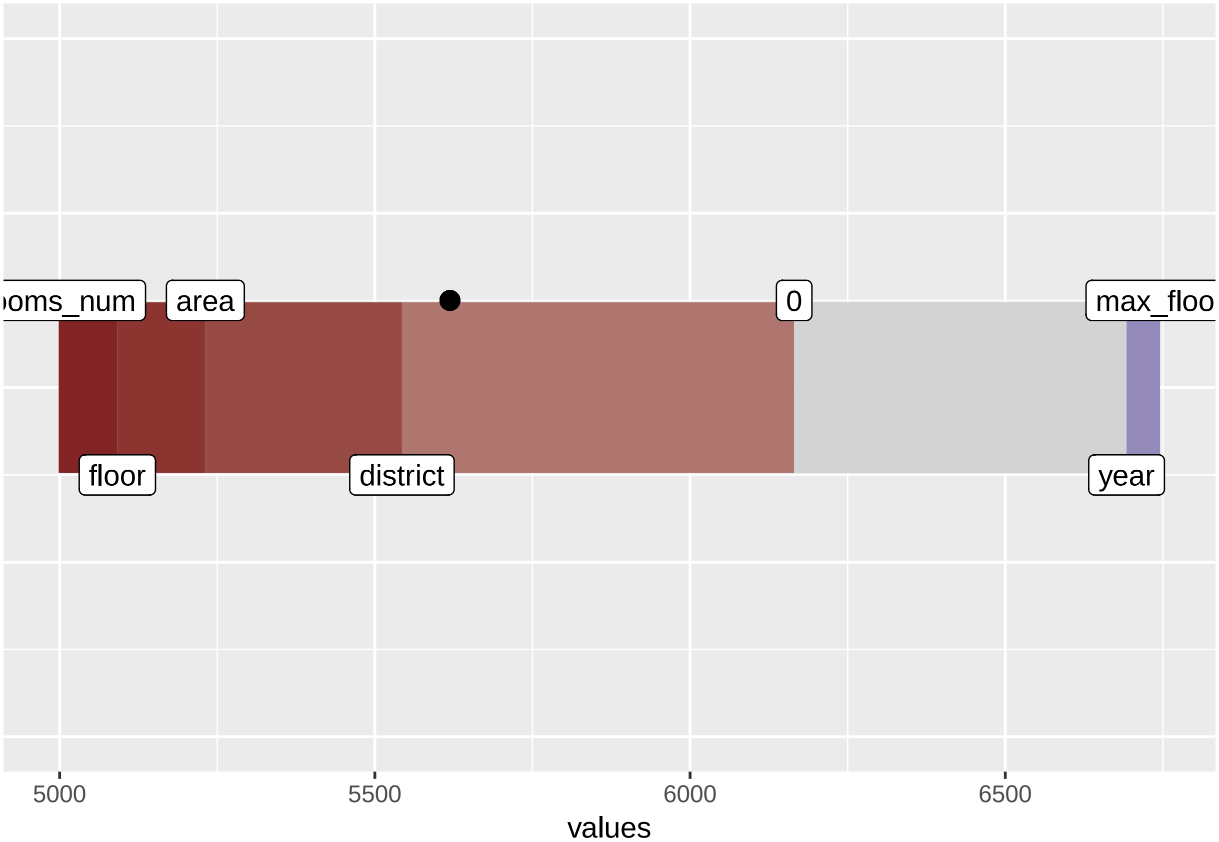
- General idea
  - The goal is a decomposition of prediction into a **sum** of scores related to (simplified) features.
  - The problem is solved using game theory: *Shapley values*. Variables are *players* who contribute to the outcome - the prediction - and we try to *pay* them accordingly to their contributions.
- This approach unifies several methods (including `LIME`).
- Some details
  - Exact methods exist for linear models and tree ensemble methods. In other cases, approximations are needed.
  - The classic way: sample permutations of variables, then average contributions.
  - The better way: approximation based on LIME and Shapley values for regression.
- This method has good theoretical properties, but will not produce sparse explanations

Example

```r
library(shapleyr)
shapley_explanation <- shapley(4036,
                               task = house_task,
                               model = house_rf_mlr)
class(shapley_explanation) <- c("shapley.singleValue", "list")
gather(shapley_explanation$values, "feature", "shapley.score")
```

```
##     feature shapley.score
## 1       _Id          4036
## 2    _Class          <NA>
## 3 rooms_num       -93.112
## 4      area      -311.833
## 5      year       527.284
## 6     floor      -139.601
## 7 max_floor        53.036
## 8  district      -622.128
```

```r
plot(shapley_explanation)
```



- **0** is the mean of all predictions
- the **black dot** is the prediction we are explaining
- values and the plot describe how we move from the global mean of predictions to this particular predictions
- most important features are the ones that help move the most

Acknowledgement

Podziekowanie dla UWr…

Time to practice!

Third set of exercises goes here