eRum exercises

Mateusz Staniak

Exercises

Exercise 1

Run the following code to fit random forest, linear regression and SVM to the housing prices data.

```
library(tidyverse)
library(live)
library(DALEX)
library(randomForest)
library(e1071)
library(auditor)
load("./rda_files/houses.rda")

house_rf <- randomForest(sqm_price ~., data = houses)
house_svm <- svm(sqm_price ~., data = houses)
house_lm <- lm(sqm_price ~., data = houses)
```

Create DALEX explainer object for each of the models. Create and compare boxplots of residuals for all the models (`model_performance`).

- Which model is the best?
- Are there any outlying predictions?
- Find the observation with the largest absolute value of residual among houses cheaper than 7000 PLN.

TIP: object returned by `model_performance` function is a data frame with colnames *predicted*, *observed*, *diff* and *label*.

Exercise 2

Create single prediction explainers for the instance chosen in **Exercise 1**. Create Break Down plots for each of the observations. What are the keys factors that drive the prediction? Are they the same for every model?

Bonus for part 1:

Draw plots of fitted vs observed values for each of these models. Can you spot any problems with the predictions? Are the prices usually underestimated of overestimated?

Exercise 3

```
n_obs <- 1189
```

Simulate new data around the observation from **Exercise 1** (its index is in the `n_obs` object.) and the add random forest predictions. Then fit a linear model locally.

TIP: remember to load `mlr` package. TIP2: don't use too small `size` for the simulated dataset. I recommend at least 1000.

Exercise 4

Visualize approximation created in **Exercise 3**. Use `plot_explanation2` function to create a forest plot of the linear model and then the Break Down plot.

Bonus for part 2

Use `add_predictions` function to add SVM and LM predictions to the simulated dataset. Compare plots for all three models.

Exercise 5

Run the following code to train model random forest model using `mlr` interface (this is necessary for `shapleyR` package).

```
library(mlr)
load("./rda_files/houses.rda")
n_obs <- 1189

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

Use shapleyR package to calculate Shapley values for prediction chosen in **Exercise 1** (its index is in `n_obs` object). Are the results consistent with Break Down results from **Exercise 2**? Draw a plot of Shapley values.

TIP: remember to set class of the object returned by `shapley` function to `shapley.singleValue` before using `plot`.

Exercise 6

Use `lime` package to approximate random forest model around prediction chosen in **Exercise 1** (its index is in `n_obs` object). Follow the `lime` work flow: 1. Create an explainer. 2. Approximate the model around the explained instane. 3. Use `plot_features` function to see, how features influence this prediction.

TIP: use `house_rf_mlr` object from **Exercise 5**, because `lime` works well with `mlr` objects.

Bonus for part 3:

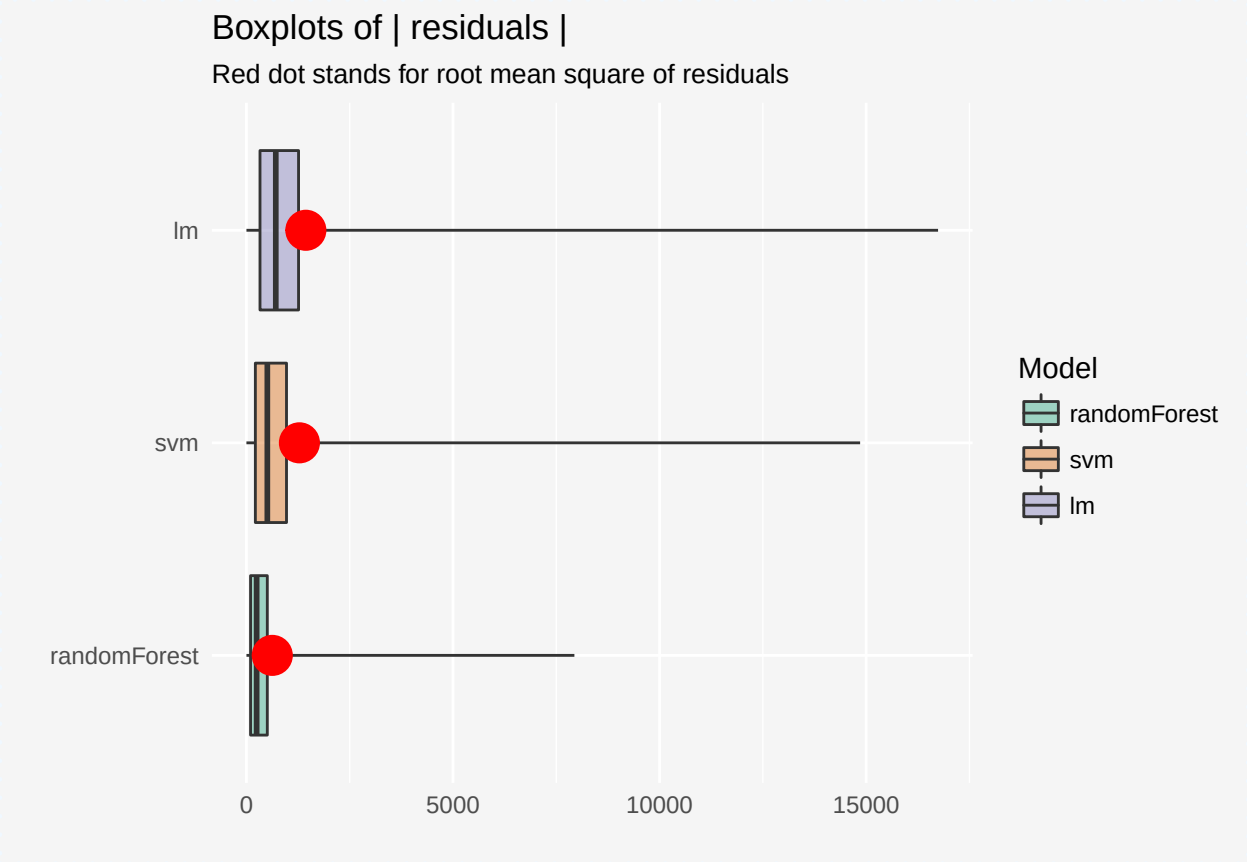
Create variable importance explainer. Compare global variable importance to scores obtained in **Exercise 5** and **Exercise 6**.

Solutions

Exercise 1

```
rf_expl <- DALEX::explain(house_rf, data = houses, y = houses$sqm_price)
svm_expl <- DALEX::explain(house_svm, data = houses, y = houses$sqm_price)
lm_expl <- DALEX::explain(house_lm, data = houses, y = houses$sqm_price)

rf_perf <- model_performance(rf_expl)
svm_perf <- model_performance(svm_expl)
```

```
lm_perf <- model_performance(lm_expl)

plot(rf_perf,
     svm_perf,
     lm_perf,
     geom = "boxplot")
```




```
arrange(rf_perf, desc(abs(diff))) %>%
  filter(observed < 7000) %>%
  head(5)
```

```
##   predicted observed     diff        label
## 1  4508.472     1585 2923.472 randomForest
## 2  7476.846     4750 2726.846 randomForest
## 3  4793.898     2174 2619.898 randomForest
## 4  6279.129     3742 2537.129 randomForest
## 5  5242.653     2843 2399.653 randomForest
```

```
arrange(svm_perf, desc(abs(diff))) %>%
  filter(observed < 7000) %>%
  head(5)
```

```
##   predicted observed     diff label
## 1  5771.421     1585 4186.421   svm
## 2  7983.699     4063 3920.699   svm
## 3  5722.197     1957 3765.197   svm
## 4  5907.319     2174 3733.319   svm
```

```
## 5  5544.827     1877 3667.827   svm
```

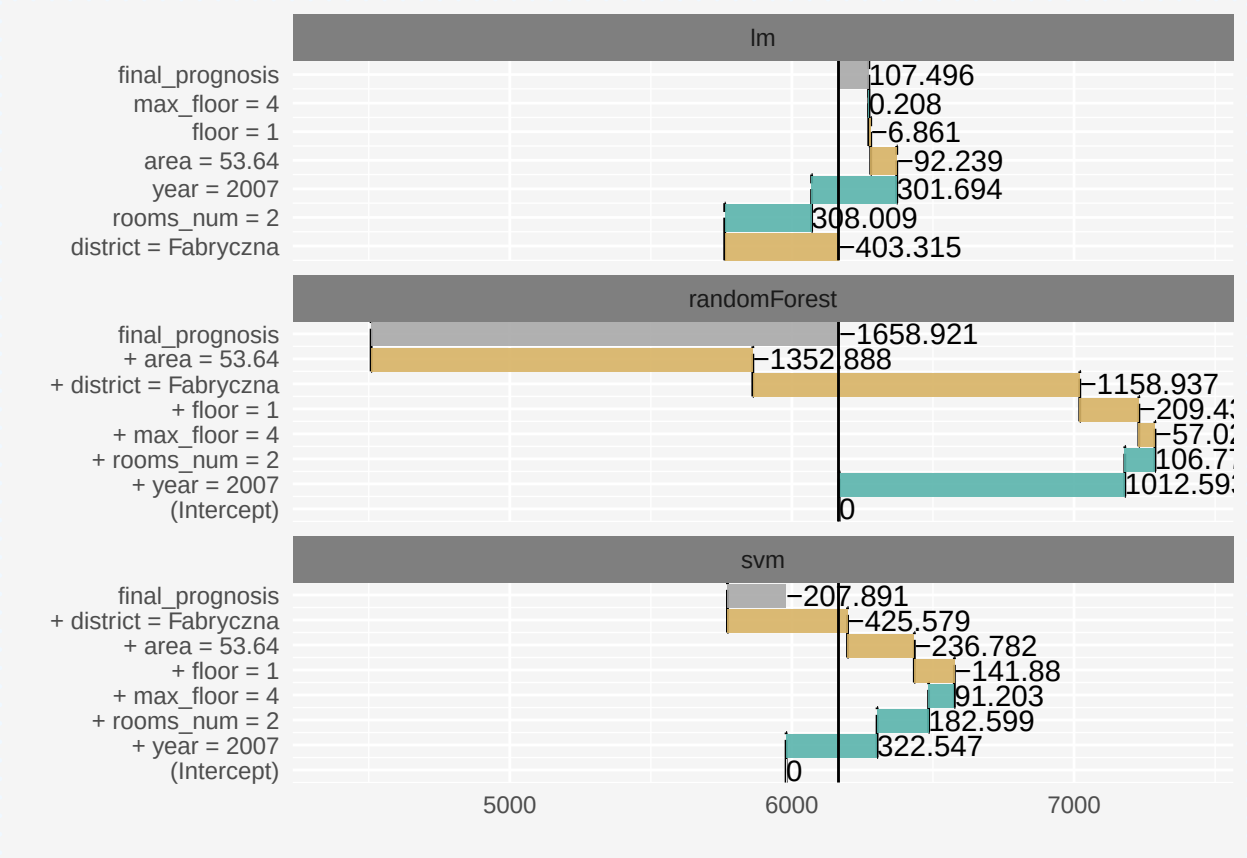
```
arrange(lm_perf, desc(abs(diff))) %>%
  filter(observed < 7000) %>%
  head(5)
```

```
##   predicted observed     diff label
## 1  6272.608     1585 4687.608    lm
## 2  8325.985     3702 4623.985    lm
## 3  8250.050     4267 3983.050    lm
## 4  6466.103     2638 3828.103    lm
## 5  5744.796     1957 3787.796    lm
```

```
n_obs <- which(houses$sqm_price == 1585)
```

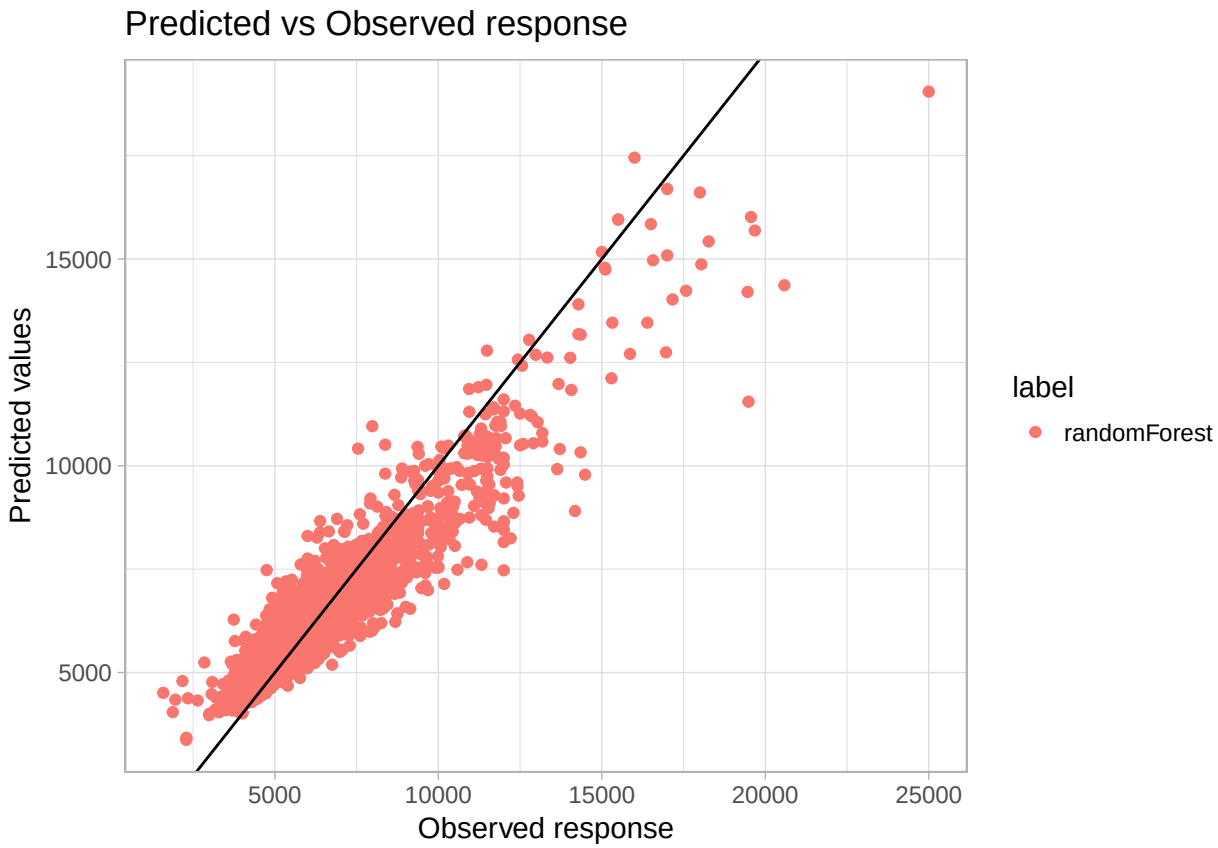
Exercise 2

```
rf_expl_sp <- single_prediction(rf_expl, houses[n_obs, -3])
svm_expl_sp <- single_prediction(svm_expl, houses[n_obs, -3])
lm_expl_sp  <-single_prediction(lm_expl, houses[n_obs, -3])

plot(rf_expl_sp,
     svm_expl_sp,
     lm_expl_sp)
```

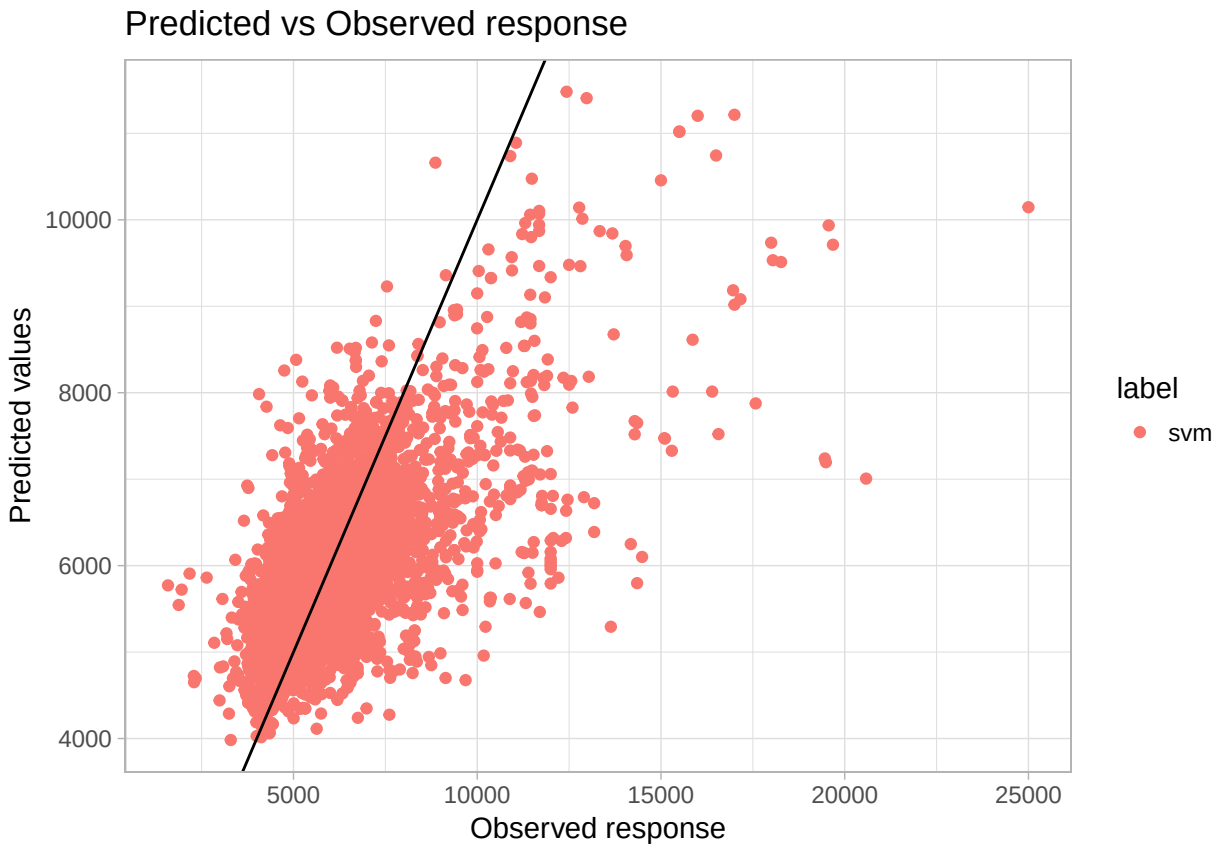


Bonus

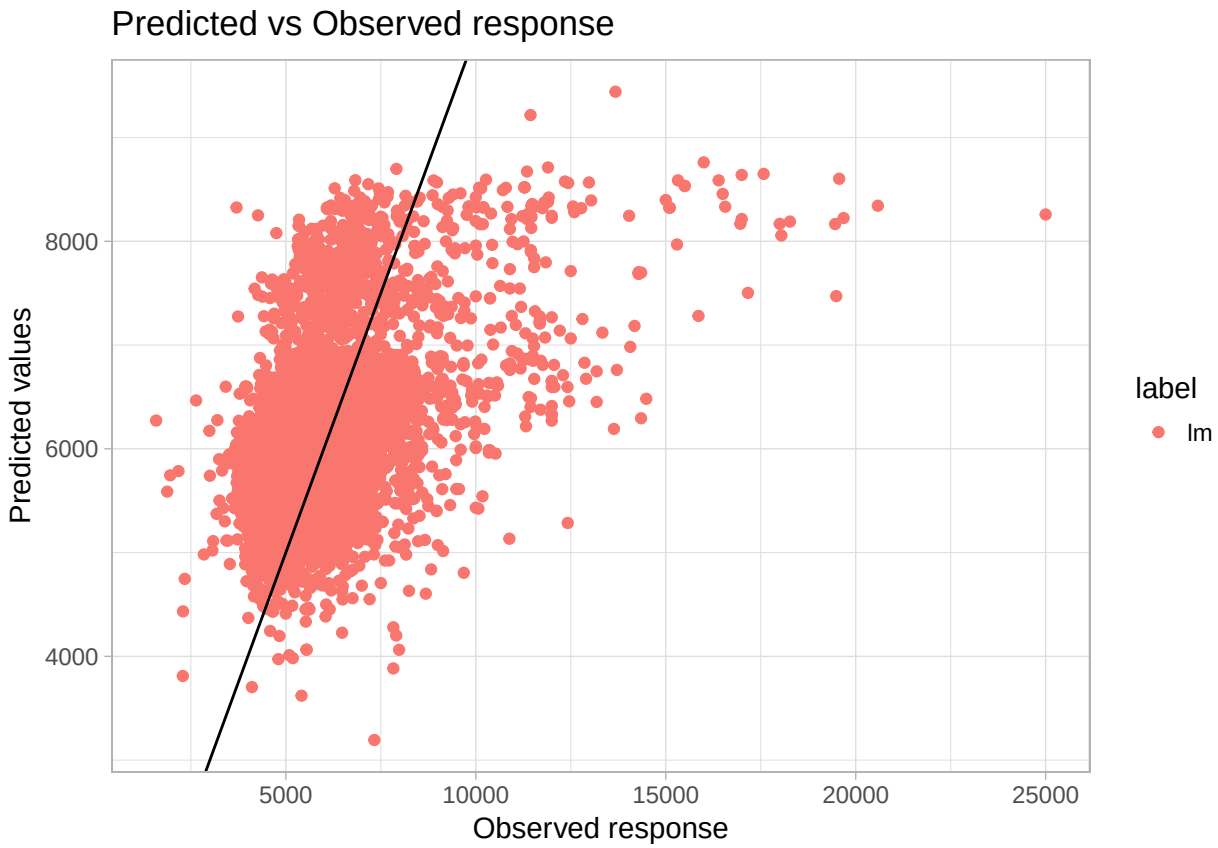
```
rf_audit <- audit(rf_expl)
svm_audit <- audit(svm_expl)
lm_audit <- audit(lm_expl)

plotPrediction(rf_audit)
```




```
plotPrediction(svm_audit)
```




```
plotPrediction(lm_audit)
```



Exercise 3

```
library(live)
library(mlr)

houses_similar <- sample_locally2(houses, houses[n_obs, ], "sqm_price", 1000)
houses_similar2 <- add_predictions2(houses_similar, house_rf)
lm_approx <- fit_explanation2(houses_similar2, "regr.lm")

lm_approx
```

```
## Warning in summary.lm(model_tmp): essentially perfect fit: summary may be
## unreliable
```

```
## 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: 0.9203
```

Exercise 4

```
plot_explanation2(lm_approx, regr_plot_type = "forest")
```

```
## Warning in summary.lm(x): essentially perfect fit: summary may be
## unreliable
```

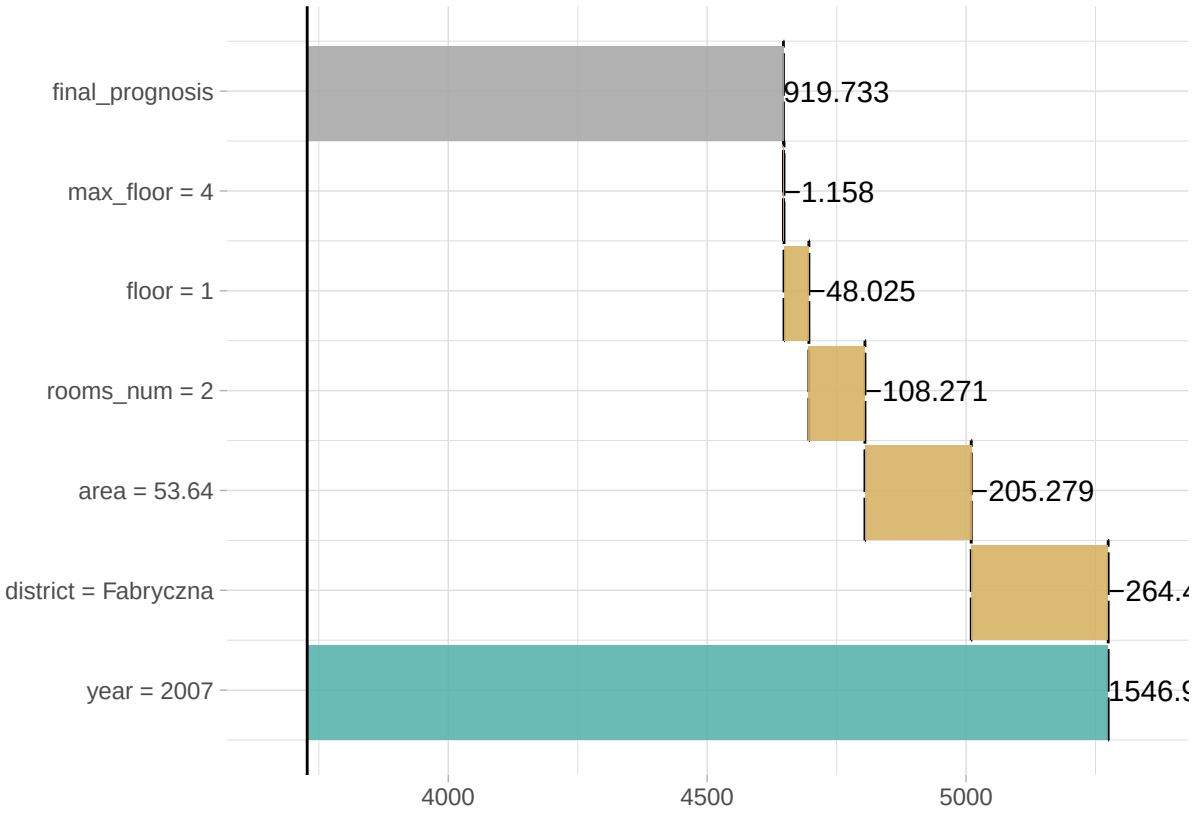
```
## Warning in summary.lm(object): essentially perfect fit: summary may be
## unreliable
```

```
## Warning in recalculate_width_panels(panel_positions, mapped_text =
## mapped_text, : Unable to resize forest panel to be smaller than its
## heading; consider a smaller text size
```

Variable		N	Estimate	p
rooms_num		1000	1002.51 (934.23, 1070.78)	<0.001
area		1000	115.56 (−40.84, 271.96)	0.15
year		1000	354.88 (190.86, 518.91)	<0.001
floor		1000	181.91 (118.46, 245.36)	<0.001
max_floor		1000	−82.71 (−163.29, −2.14)	0.04
district	Fabryczna	870	Reference	
	Krzyki	63	1319.08 (1251.49, 1386.67)	<0.001
	Psie Pole	13	1134.13 (990.26, 1278.00)	<0.001
	Srodmiescie	37	2066.41 (1979.72, 2153.10)	<0.001
	Stare Miasto	17	5304.19 (5178.03, 5430.36)	<0.001
(Intercept)			−715659.83 (−1044976.00, −386343.6	

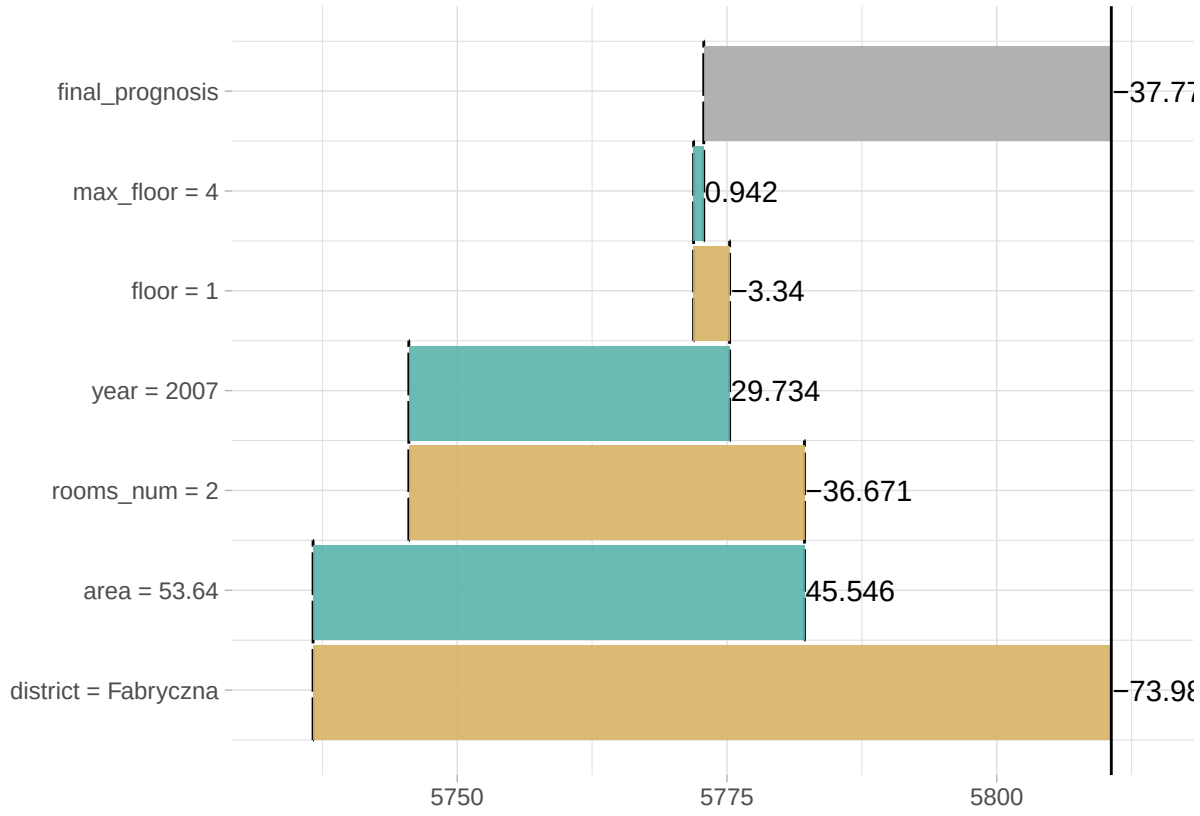
−1e+06 −8e+05 −6e+05 −4e+05 −2e+05 0e+00

```
plot_explanation2(lm_approx, regr_plot_type = "waterfall")
```

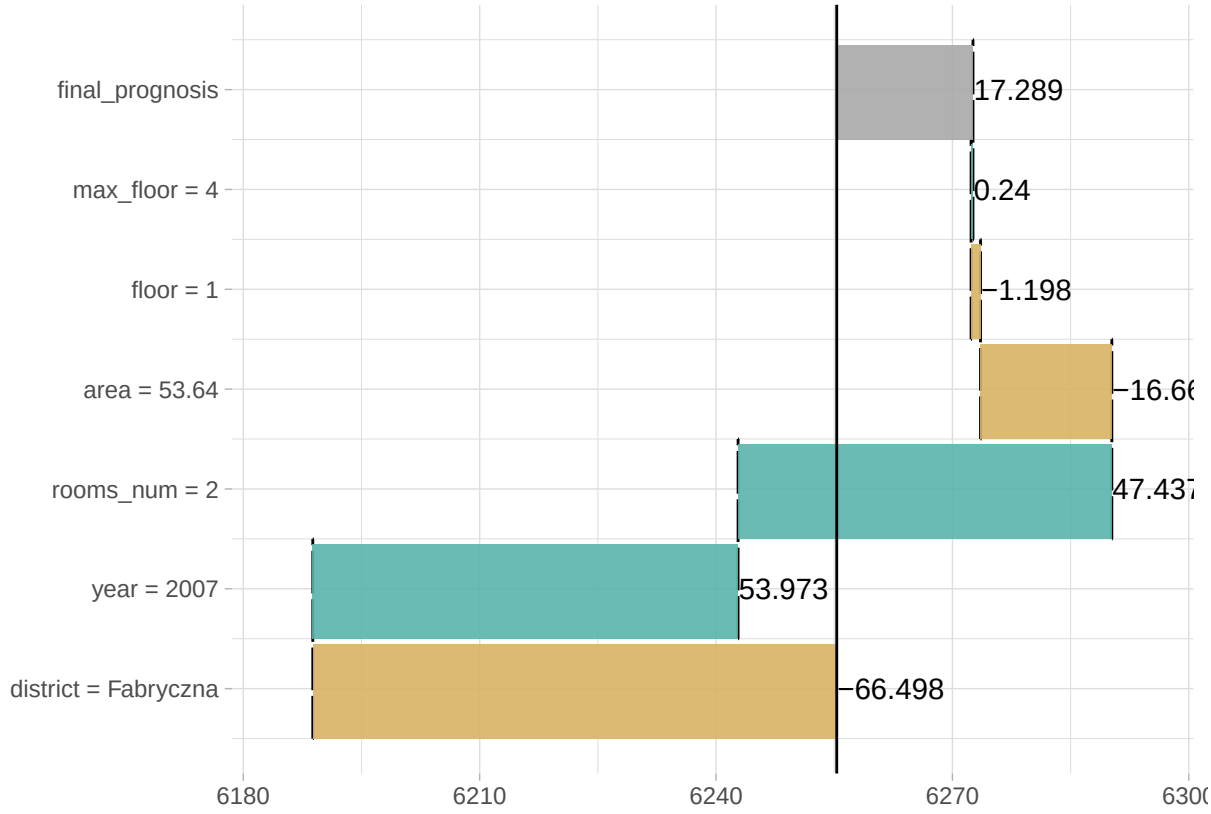


Bonus

```
houses_similar3 <- add_predictions2(houses_similar, house_svm)
houses_similar4 <- add_predictions2(houses_similar, house_lm)
lm_approx2 <- fit_explanation2(houses_similar3)
lm_approx3 <- fit_explanation2(houses_similar4)

plot_explanation2(lm_approx2, "waterfall")
```




```
plot_explanation2(lm_approx3, "waterfall")
```



```
plot_explanation2(lm_approx2, "forest")
```

```
## Warning in summary.lm(x): essentially perfect fit: summary may be
## unreliable
```

```
## Warning in summary.lm(object): essentially perfect fit: summary may be
## unreliable
```

```
## Warning in recalculate_width_panels(panel_positions, mapped_text =
## mapped_text, : Unable to resize forest panel to be smaller than its
## heading; consider a smaller text size
```

Variable		N	Estimate	p
rooms_num		1000	339.54 (337.29, 341.80)	<0.001
area		1000	−25.64 (−30.80, −20.48)	<0.001
year		1000	6.82 (1.41, 12.24)	0.01
floor		1000	12.65 (10.56, 14.75)	<0.001
max_floor		1000	67.27 (64.61, 69.93)	<0.001
district	Fabryczna	870	Reference	
	Krzyki	63	434.74 (432.51, 436.98)	<0.001
	Psie Pole	13	24.60 (19.85, 29.34)	<0.001
	Srodmiescie	37	754.98 (752.12, 757.84)	<0.001
	Stare Miasto	17	1078.80 (1074.64, 1082.97)	<0.001
(Intercept)			−7502.86 (−18373.50, 3367.79)	0.18

```
plot_explanation2(lm_approx3, "forest")
```

```
## Warning in summary.lm(x): essentially perfect fit: summary may be
## unreliable
```

```
## Warning in summary.lm(object): essentially perfect fit: summary may be
## unreliable
```

```
## Warning in recalculate_width_panels(panel_positions, mapped_text =
## mapped_text, : Unable to resize forest panel to be smaller than its
## heading; consider a smaller text size
```

Variable		N	Estimate	p
rooms_num		1000	−439.23 (−439.23, −439.23)	<0.001
area		1000	9.38 (9.38, 9.38)	<0.001
year		1000	12.38 (12.38, 12.38)	<0.001
floor		1000	4.54 (4.54, 4.54)	<0.001
max_floor		1000	17.12 (17.12, 17.12)	<0.001
district	Fabryczna	870	Reference	
	Krzyki	63	277.96 (277.96, 277.96)	<0.001
	Psie Pole	13	−412.57 (−412.57, −412.57)	<0.001
	Srodmiescie	37	569.83 (569.83, 569.83)	<0.001
	Stare Miasto	17	1956.82 (1956.82, 1956.82)	<0.001
(Intercept)			−18275.81 (−18275.81, −18275.81)	<0.001

Exercise 5

```
library(shapleyr)
```

```
## Loading required package: checkmate
## Loading required package: combinat
##
## Attaching package: 'combinat'
## The following object is masked from 'package:utils':
##
##     combn
## Loading required package: shiny
## Loading required package: shinydashboard
##
## Attaching package: 'shinydashboard'
## The following object is masked from 'package:graphics':
##
##     box
## Loading required package: reshape2
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
## Welcome to the ShapleyR package!
```

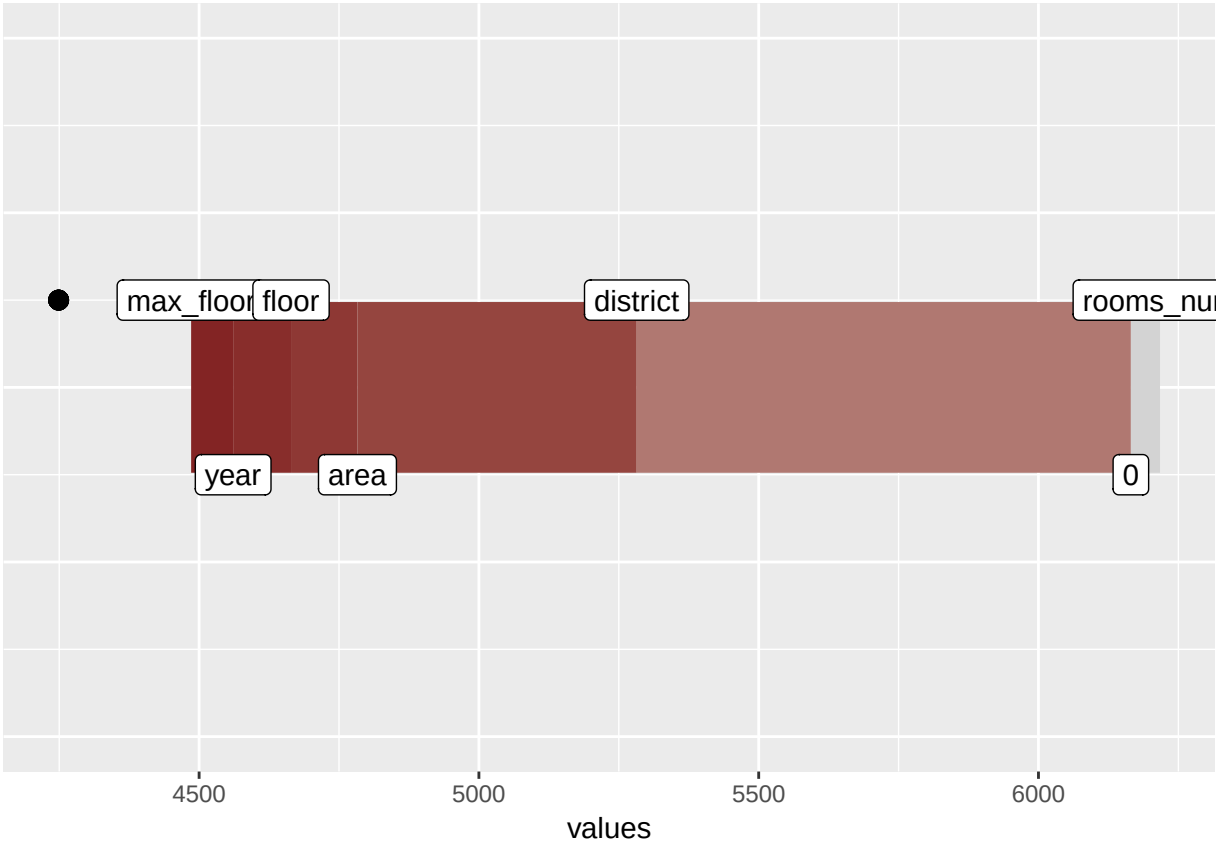
```
shapley_vals <- shapley(n_obs, house_task, house_rf_mlr)

gather(shapley_vals$values, "colname", "contribution") %>%
  filter(colname %in% colnames(houses)) %>%
  mutate(contribution = as.numeric(contribution)) %>%
  arrange(desc(abs(contribution)))
```

```
##     colname contribution
## 1  district     -883.119
## 2      area     -499.018
## 3     floor     -118.705
## 4      year     -103.425
## 5 max_floor      -75.097
## 6 rooms_num       52.103
```

```
# alternatively use just
shapley_vals
```

```
## $task.type
## [1] "regr"
##
## $feature.names
## [1] "rooms_num" "area"      "year"      "floor"     "max_floor" "district"
##
## $predict.type
## [1] "response"
##
## $prediction.response
## [1] 4248.977
##
## $data.mean
## [1] 6165.112
##
## $values
##    _Id _Class rooms_num     area     year    floor max_floor district
## 1 1189     NA    52.103 -499.018 -103.425 -118.705   -75.097 -883.119
```

```
class(shapley_vals) <- c("shapley.singleValue", "list")
plot(shapley_vals)
```



Exercise 6

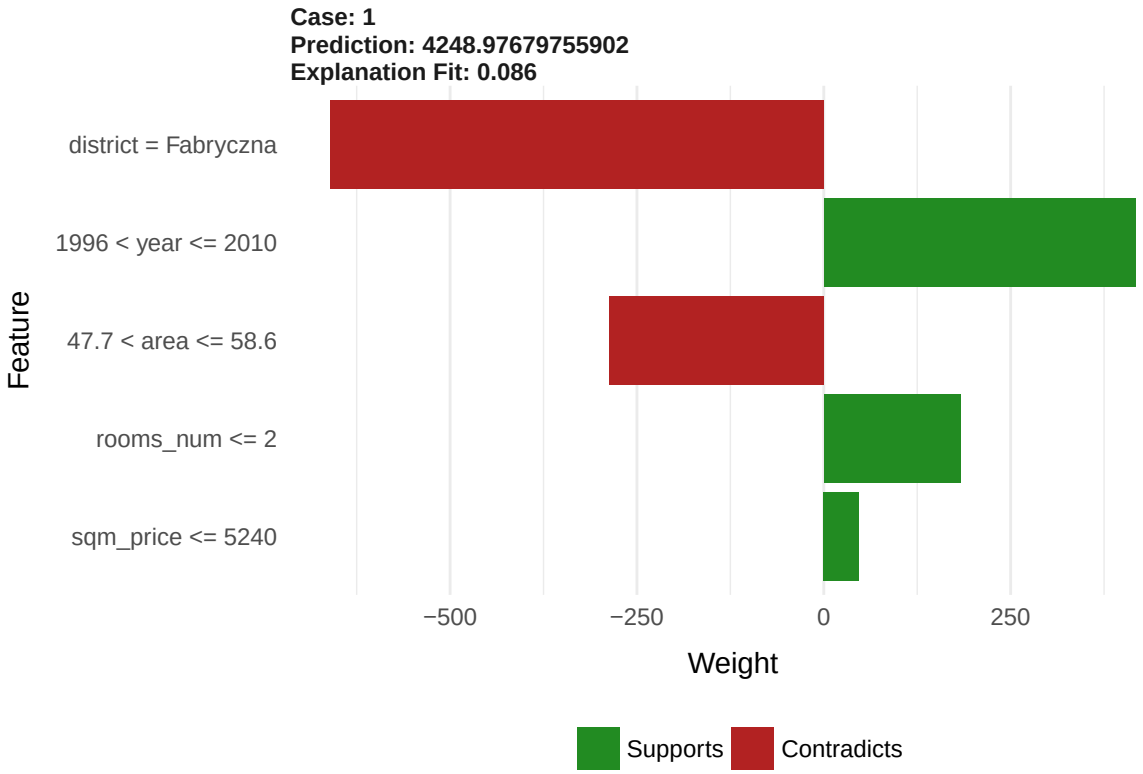
```
library(lime)
```

```
##
## Attaching package: 'lime'
```

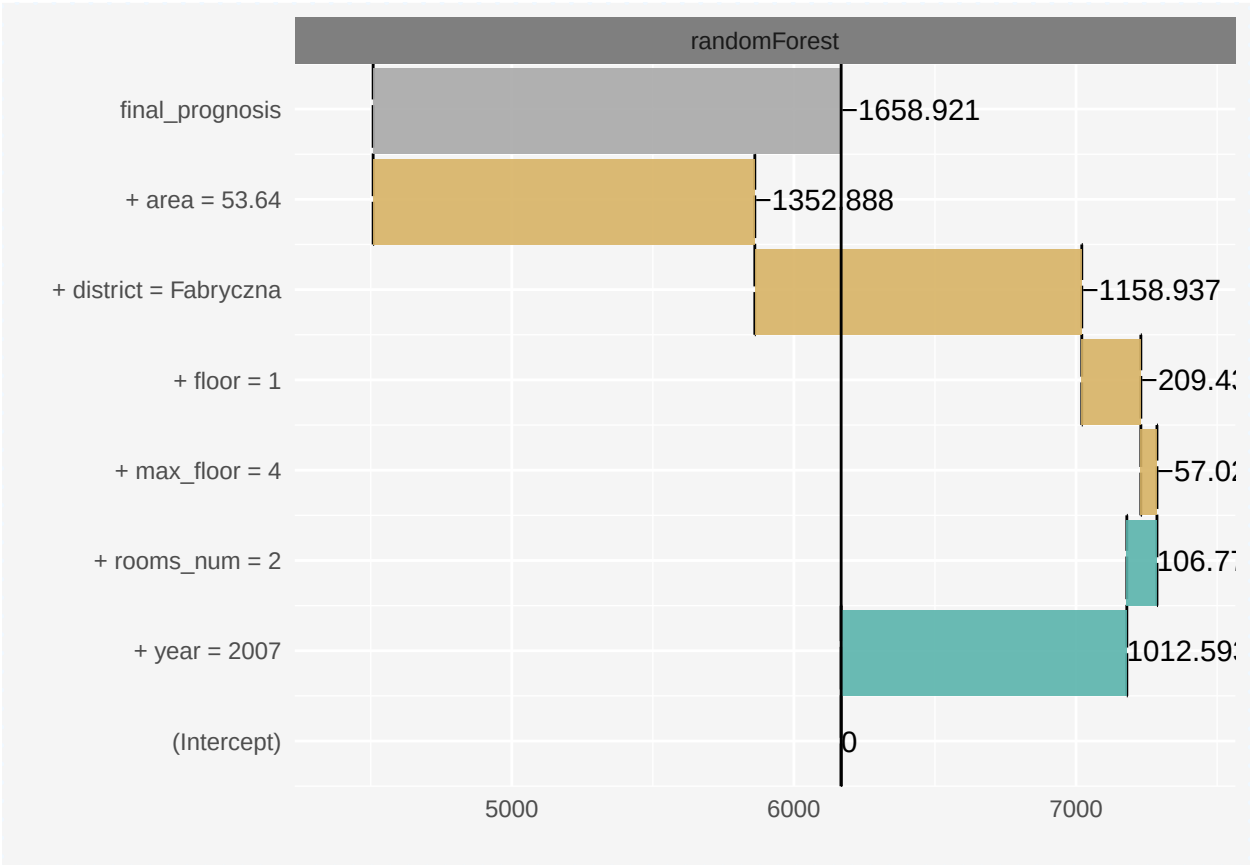
```
## The following object is masked from 'package:DALEX':
##
##     explain
```

```
## The following object is masked from 'package:dplyr':
##
##     explain
```

```
lime_rf <- lime(houses, house_rf_mlr)
lime_explanation <- lime::explain(houses[n_obs, ], lime_rf, n_features = 5)
plot_features(lime_explanation)
```




```
plot(rf_expl_sp)
```



Bonus

```
rf_global <- variable_importance(rf_expl)
plot(rf_global)
```