Problem set 7

Matt He

2022-11-05

```
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
library(randomForest)
```

```
## randomForest 4.7-1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(AppliedPredictiveModeling)
library(ggthemes)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --

## v tibble  3.1.6      v dplyr   1.0.10
## v tidyr   1.2.0      v stringr 1.4.0
## v readr   2.1.2      v forcats 0.5.1
## v purrr   0.3.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::combine()       masks randomForest::combine()
## x dplyr::filter()        masks stats::filter()
## x dplyr::lag()           masks stats::lag()
## x purrr::lift()          masks caret::lift()
## x randomForest::margin() masks ggplot2::margin()
```

Reading data in

```
pre_housing <- read_csv('~/Downloads/PreCrisisCV.csv')
```

```
## Rows: 1978 Columns: 18
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (18): Property, LandValue, BuildingValue, Acres, AboveSpace, Basement, D...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
post_housing <- read_csv('~/Downloads/PostCrisisCV.csv')
```

```
## Rows: 1657 Columns: 18
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (18): Property, LandValue, BuildingValue, Acres, AboveSpace, Basement, D...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
OnMarketTest <- read_csv('~/Downloads/OnMarketTest-1.csv')
```

```
## Rows: 2000 Columns: 18
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (18): Property, LandValue, BuildingValue, Acres, AboveSpace, Basement, D...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
pre_housing = pre_housing %>%
  select(-Property)%>%
  mutate_at(vars(PoorCondition, GoodCondition, AC), .funs = factor) %>%
  mutate(mkt = "PreCrisis")

post_housing = post_housing %>%
  select(-Property)%>%
  mutate_at(vars(PoorCondition, GoodCondition, AC), .funs = factor) %>%
  mutate(mkt = "PostCrisis")

OnMarketTest = OnMarketTest %>%
  select(-Property)%>%
  mutate_at(vars(PoorCondition, GoodCondition, AC), .funs = factor) %>%
  mutate(mkt = "PostCrisis")

whole_housing <- rows_append(pre_housing,post_housing)
```

For this part, I read the original data in and made several changes. For example, I labeled them as pre-crisis and post-crisis. For all the data in 'OnMarketTest', since it is not sold yet, so I labeled them as post-crisis as well.

```
set.seed(123)

pre_samp = caret::createDataPartition(pre_housing$Price, p = 0.6, list = FALSE)
post_samp = caret::createDataPartition(post_housing$Price, p = 0.6, list = FALSE)
whole_samp = caret::createDataPartition(whole_housing$Price, p = 0.6, list = FALSE)

pre_train = pre_housing[pre_samp, ]
pre_test = pre_housing[-pre_samp, ]

post_train = post_housing[post_samp, ]
post_test = post_housing[-post_samp, ]

whole_train = whole_housing[whole_samp,]
whole_test = whole_housing[whole_samp,]
```

Here, I combined two tables together so we can analyze those two together, and I did data partitioning. To distinguish, every row has been labeled “PreCrisis” or “PostCrisis”. By train the model on the all data before and after the bubble, the models I create can capture the difference in before, during and after crisis.

Calculating the error for assessed value.

```
whole_housing %>%
  mutate(AssessedValue = BuildingValue+LandValue)%>%
  mutate(AssessedError = Price - AssessedValue) -> assessed_error

assessed_error %>%
  group_by(mkt) %>%
  summarize(ME = mean(AssessedError),
            RMSE = sqrt(sum(AssessedError^2)/(nrow(assessed_error)-1)),
            MAE = mean(abs(AssessedError)),
            MAPE = mean(abs(AssessedError/Price)))
```

```
## # A tibble: 2 x 5
##   mkt            ME   RMSE    MAE  MAPE
##   <chr>       <dbl>  <dbl>  <dbl> <dbl>
## 1 PostCrisis -4579. 12926. 14331. 0.185
## 2 PreCrisis   1216. 13129. 13613. 0.161
```

In our entire data set, we can see the RMSE for assessed value is 13128.67 pre-crisis and 12925.61 post-crisis. According to this data, actually, the assessed value model becomes slightly better after the bubble burst, but overall, it just slightly changed. So I believe it does not become a better pridictor. ## Creating model

```
ctrl = trainControl(method = "repeatedcv",
                    number = 5,
                    repeats = 5,
                    search = "random")
rf_model = train(
  Price ~ .,
  data = whole_train,
  method = "ranger",
  metric = "RMSE")

rf_model_pre = train(
  Price ~ .,
  data = select(pre_train, -mkt),
  method = "ranger",
  metric = "RMSE")

rf_model_post = train(
  Price ~ .,
  data = select(post_train, -mkt),
  method = "ranger",
  metric = "RMSE")
```

Here, I created three random forest models for whole dataset, before crisis and after crisis.

```
whole_test%>%
  mutate(pred = predict(rf_model, newdata =whole_test))%>%
  mutate(predError = Price - pred) -> whole_error

whole_error %>%
  group_by(mkt) %>%
  summarize(ME = mean(predError),
            RMSE = sqrt(sum(predError^2)/(nrow(whole_error)-1)),
            MAE = mean(abs(predError)),
            MAPE = mean(abs(predError/Price)))
```

```
## # A tibble: 2 x 5
##   mkt           ME  RMSE   MAE   MAPE
##   <chr>      <dbl> <dbl> <dbl>  <dbl>
## 1 PostCrisis -325. 5661. 6354. 0.0812
## 2 PreCrisis   371. 6332. 6073. 0.0750
```

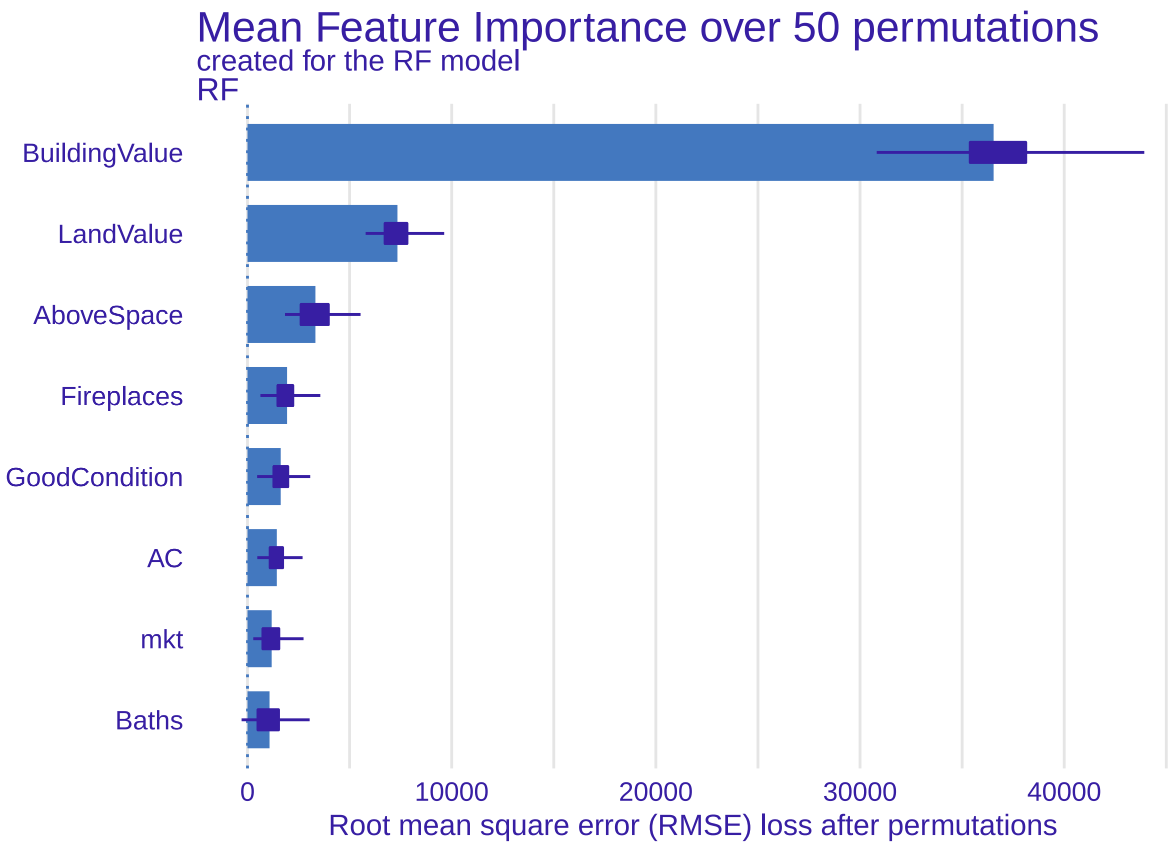
When we apply the random forest model into whole_test dataset, we can compare the real price and the prediction. Compare to the assessed value, our model is much more accurate.

```
set.seed(4568)
rf_explain = DALEX::explain(model = rf_model,
                                  data = whole_housing,
                                  y = whole_housing$Price,
                                  type = "regression",
                                  label = "RF")
```

```
## Preparation of a new explainer is initiated
##   -> model label       :  RF
##   -> data              :  3635  rows  18  cols
##   -> data              :  tibble converted into a data.frame
##   -> target variable   :  3635  values
##   -> predict function  :  yhat.train  will be used (  default  )
##   -> predicted values  :  No value for predict function target column. (  default  )
##   -> model_info        :  package caret , ver. 6.0.93 , task regression (  default  )
##   -> model_info        :  type set to  regression
##   -> predicted values  :  numerical, min =  19516.94 , mean =  115024.3 , max =  546332.3
##   -> residual function :  difference between y and yhat (  default  )
##   -> residuals         :  numerical, min =  -60843.14 , mean =  -10.85438 , max =  118541.7
##   A new explainer has been created!
```

```
rf_mp = DALEX::model_parts(explainer = rf_explain,
                   B = 50,
                   type = "difference")

plot(rf_mp, max_vars = 8) +
  ggtitle("Mean Feature Importance over 50 permutations")
```

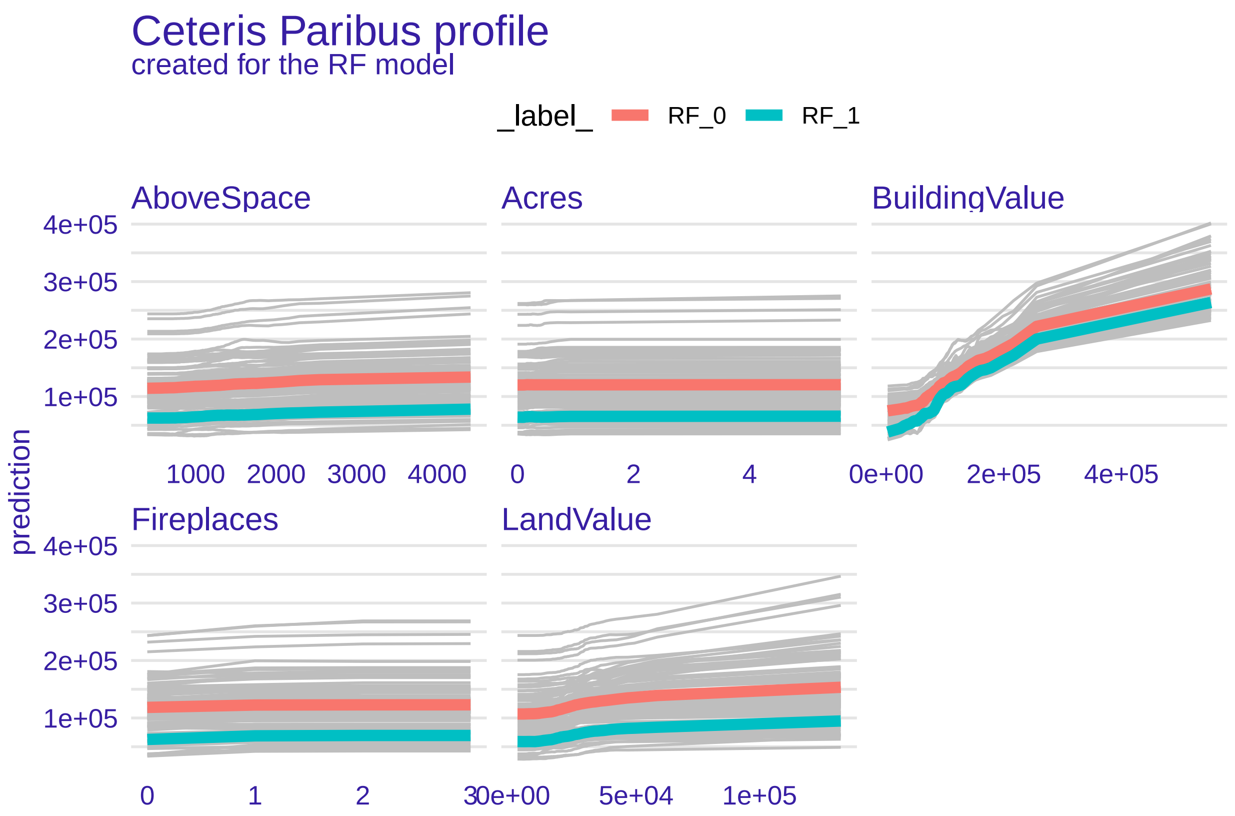


According to this plot, we can see the BuildingValue is the most important value that affects price. This is how this explainer or chart works. If we make everything else in the predictor variables constant, and we only change one of them, which will make RMSE change the most. In this case, we make everything else constant, and change BuildingValue. Our RMSE have a big difference than original price. We can tell, the BuildingValue matters the most for Price.

```
rf_pd = DALEX::model_profile(rf_explain,
                              variables = c("BuildingValue", "LandValue", "AboveSpace","Fireplaces", "Po
                              N = 100,
                              groups = "PoorCondition")
```

```
## Non-numerical variables (from the 'variables' argument) are rejected.
```

```
plot(rf_pd, geom = "profiles")
```



Same as this chart. If we only change the value from one of these variables, this is how prediction is going to be different.

```
whole_housing%>%
  mutate(BigError = ifelse(abs(Price-BuildingValue+LandValue)>15000, 1, 0)) %>%
  summarize("P(Error > $15,000)" = mean(BigError))
```

```
## # A tibble: 1 x 1
##   `P(Error > $15,000)`
##                  <dbl>
## 1                0.892
```

This is probability of predicted price being off by more than $15000 using assessed value model.

```
whole_housing %>%
  mutate(pred = predict(rf_model, newdata = whole_housing))%>%
  mutate(predError = Price - pred)%>%
  mutate(BigError = ifelse(abs(predError)>15000, 1, 0)) %>%
  summarize("P(Error > $15,000)" = mean(BigError))
```

```
## # A tibble: 1 x 1
##   `P(Error > $15,000)`
##                  <dbl>
## 1                0.177
```

In our entire housing price set, this is probability of predicted price being off by more than $15000 using random forrest model.

```r
compare_table <- OnMarketTest%>%
  mutate(pred_pre_model = predict(rf_model_pre, newdata = OnMarketTest))%>%
  mutate(pred_post_model = predict(rf_model_post, newdata = OnMarketTest))%>%
  mutate(percent_diff = (100*(pred_pre_model-pred_post_model)/pred_post_model))%>%
  mutate(percent_diff_pre = (100*(pred_pre_model-LandValue+BuildingValue)/(LandValue+BuildingValue)))%>%
  mutate(percent_diff_post = (100*(pred_post_model-LandValue+BuildingValue)/(LandValue+BuildingValue)))

compare_table%>%
  summarise(percent_difference = mean(percent_diff),
            percent_difference_between_precrises = mean(percent_diff_pre),
            percent_difference_between_postcrisis = mean(percent_diff_post))
```

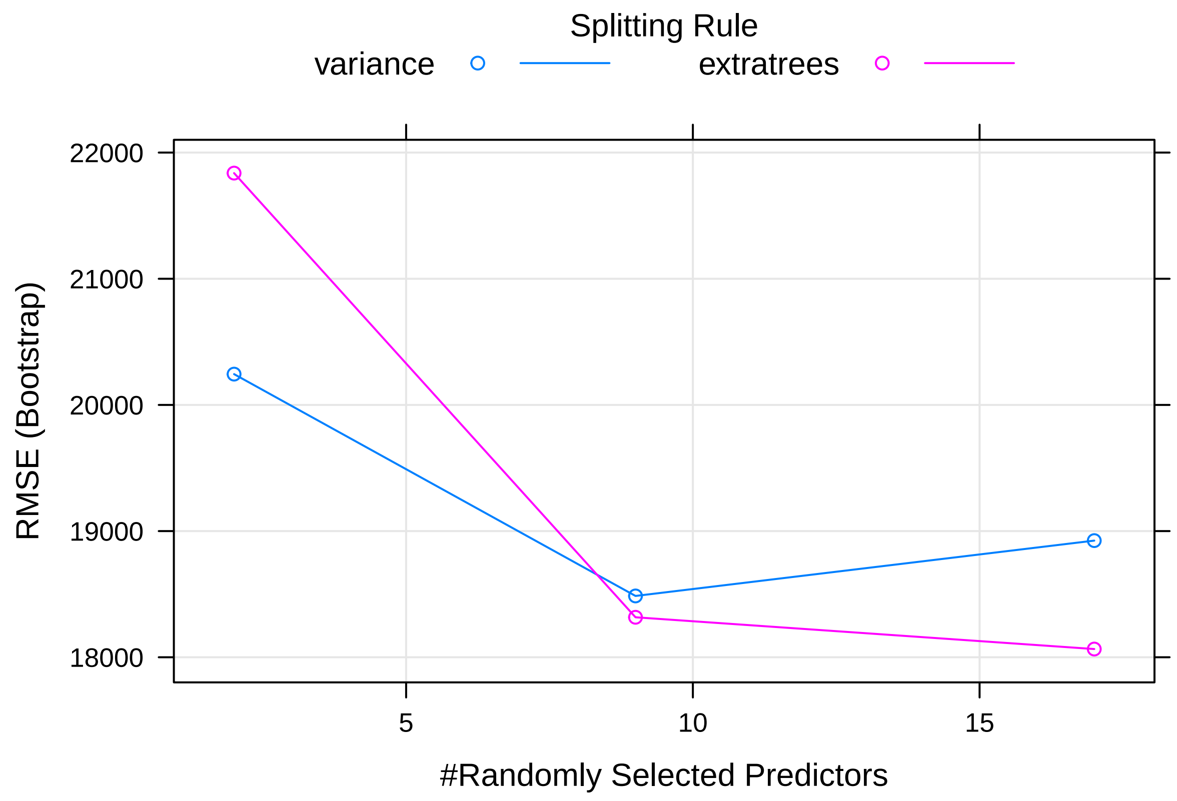
```
## # A tibble: 1 x 3
##   percent_difference percent_difference_between_precrises percent_difference_be~
##                <dbl>                                <dbl>                  <dbl>
## 1               5.08                                 163.                   159.
```

The percent difference is 4.46% if we use pre-crisis data and post-crisis data to train the model. The I do not think predicted price are very different from the assessed value. The difference is 163.4% and 159.8%.

```r
rf_model
```

```
## Random Forest
##
## 2183 samples
##   17 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 2183, 2183, 2183, 2183, 2183, 2183, ...
## Resampling results across tuning parameters:
##
##   mtry  splitrule   RMSE      Rsquared   MAE
##    2    variance    20244.12  0.8631041  14204.54
##    2    extratrees  21837.35  0.8504662  15443.16
##    9    variance    18485.93  0.8751300  13792.52
##    9    extratrees  18316.93  0.8799410  13525.67
##   17    variance    18924.54  0.8681433  14203.49
##   17    extratrees  18065.09  0.8809572  13504.24
##
## Tuning parameter 'min.node.size' was held constant at a value of 5
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were mtry = 17, splitrule = extratrees
##  and min.node.size = 5.
```

```r
plot(rf_model)
```



This is our model rf_model. Just like the name, random forest, the model comes from tree regression model. We need to create a bootstrap dataset, which we choose random rows from the original set and get a new one.Then, we have 17 predictors in total, but we only just use several of them to create a simple tree model. Next, we continue with these 2 steps till we have all tree models, then combine them together. Finally, we will have a random forest model.

```
OnMarketTest<- OnMarketTest%>%
  mutate(pred = predict(rf_model,newdata = OnMarketTest))
s = t.test(OnMarketTest$pred)
s$conf.int
```

```
## [1] 104261.2 108408.7
## attr(,"conf.level")
## [1] 0.95
```

So the 95% confident level is between 104147.8 and 108285.5