House Prices Seattle

–

Individual Assignment

GitHub Repository

The whole project is under this repository at my GitHub account.
https://github.com/stsentemeidis/House_Prices_Seattle

Loading packages & Data

Before starting our EDA and model development pipeline, we first need to install and load:

- The necessary **packages**
- The data file named **house prices** we are about to use
- The custom **functions** that have been created to make the main script more clear, interpreatable and efficient.

Explanation & summary of the dataset

To begin with let's have a look at the meaning of the initial variables provided for the dataset.

- **id**: a notation for a house
- **date**: date house was sold
- **price**: price is prediction target
- **bedrooms**: number of bedrooms per house
- **bathrooms**: number of bathrooms per house
- **sqft_living**: square footage of the home
- **sqft_lot**: square footage of the lot
- **floors**: total floors (levels) in house
- **waterfront**: house which has a view to a waterfront
- **view**: has been viewed
- **condition**: how good the condition is (overall)
- **grade**: overall grade given to the housing unit, based on King County grading system
- **sqft_above**: square footage of house apart from basement
- **sqft_basement**: square footage of the basement
- **yr_built**: built year
- **yr_renovated**: year when house was renovated
- **zipcode**: zip of a house
- **lat**: latitude coordinate
- **long**: longtitude coordinate

- **sqft_living15**: living room are in 2015 (implies some renovations). This might or might not have affected the lotsize area
- **sqft_lot15**: lotsize area in 2015 (implies some renovations)

However, as we can see in the below correlation plots, the variable **view** has a high correlation with **waterfront**. Adding to that the variable **view** has many zeros. These 2 things lead me to the conclusion that variable view is not the amount of times viewed before sold, but **whether the view is good or not.**

From the below summary and description, we can see that:

1. There are no **missing values**, so our dataset is complete.
2. The variables *condition* and *grade* are ordinal variables.
3. The variables above need to be converted from int to *factors*.
4. *Bedrooms*, *bathrooms* and *floors* are variables with discrete levels that can be converted to factors probably later on.

```
## 'data.frame':    17277 obs. of  21 variables:
##  $ id           : num  9.18e+09 4.64e+08 2.22e+09 6.16e+09 6.39e+09 ...
##  $ date         : Factor w/ 372 levels "1/10/2015","1/12/2015",..: 211 331
290 20 226 223 94 34 76 294 ...
##  $ price        : num  225000 641250 810000 330000 530000 ...
##  $ bedrooms     : int  3 3 4 4 4 4 4 3 4 3 ...
##  $ bathrooms    : num  1.5 2.5 3.5 1.5 1.75 3.5 3.25 2.25 2.5 1.5 ...
##  $ sqft_living  : int  1250 2220 3980 1890 1814 3120 4160 1440 2250 2540
...
##  $ sqft_lot     : int  7500 2550 209523 7540 5000 5086 47480 10500 6840
9520 ...
##  $ floors       : num  1 3 2 1 1 2 2 1 2 1 ...
##  $ waterfront   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ view         : int  0 2 2 0 0 0 0 0 0 0 ...
##  $ condition    : int  3 3 3 4 4 3 3 3 3 3 ...
##  $ grade        : int  7 10 9 7 7 9 10 8 9 8 ...
##  $ sqft_above   : int  1250 2220 3980 1890 944 2480 4160 1130 2250 1500
...
##  $ sqft_basement: int  0 0 0 0 870 640 0 310 0 1040 ...
##  $ yr_built     : int  1967 1990 2006 1967 1951 2008 1995 1983 1987 1959
...
##  $ yr_renovated : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ zipcode      : int  98030 98117 98024 98155 98115 98115 98072 98023
98058 98115 ...
##  $ lat          : num  47.4 47.7 47.6 47.8 47.7 ...
##  $ long         : num  -122 -122 -122 -122 -122 ...
```

```
##  $ sqft_living15: int  1260 2200 2220 1890 1290 1880 3400 1510 2480 1870
...
##  $ sqft_lot15   : int  7563 5610 65775 8515 5000 5092 40428 8125 7386 6800
...
```

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_l
	Min. :1.000e+06	6/23/2014: 118	Min. : 78000	Min. : 1.000	Min. :0.500	Min. : 370	Min. : 520
	1st Qu.:2.114e+09	6/25/2014: 113	1st Qu.: 320000	1st Qu.: 3.000	1st Qu.:1.750	1st Qu.: 1430	1st Qu 5050
	Median :3.902e+09	6/26/2014: 110	Median : 450000	Median : 3.000	Median :2.250	Median : 1910	Media 7620
	Mean :4.566e+09	4/22/2015: 108	Mean : 539865	Mean : 3.369	Mean :2.114	Mean : 2080	Mean 15186
	3rd Qu.:7.303e+09	7/8/2014 : 104	3rd Qu.: 645500	3rd Qu.: 4.000	3rd Qu.:2.500	3rd Qu.: 2550	3rd Qu 10695
	Max. :9.900e+09	4/14/2015: 102	Max. :7700000	Max. :33.000	Max. :8.000	Max. :13540	Max. :11647

Summary of Dataset (Scrollable in Rmd) 1

At this point it is decided:

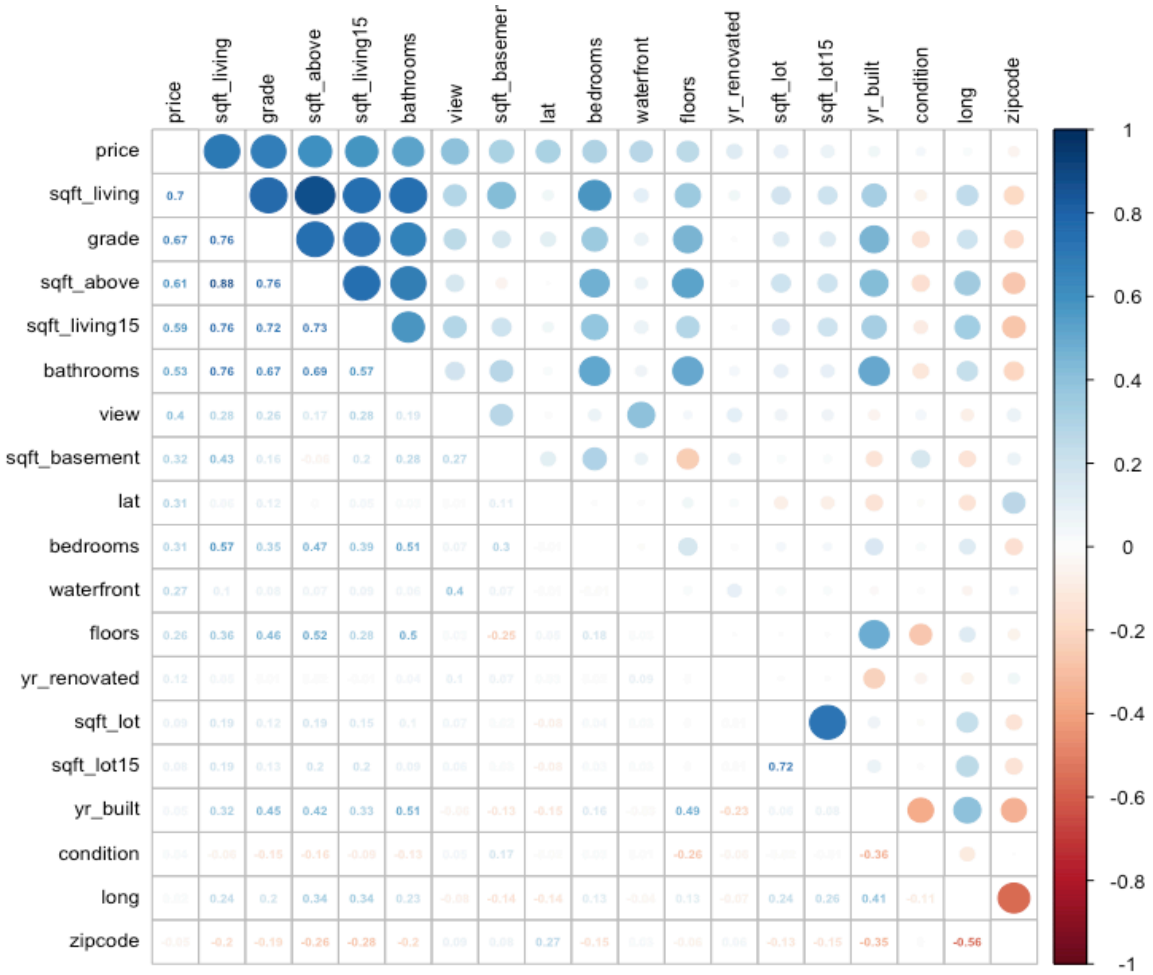
- to keep *numeric* as the common data type
- change the *dates* to a universal date type format of "%m/%d/%Y"
- create the column *price* for the test set as it is missing
- subset the numerical data columns in order to plot some initial EDA.

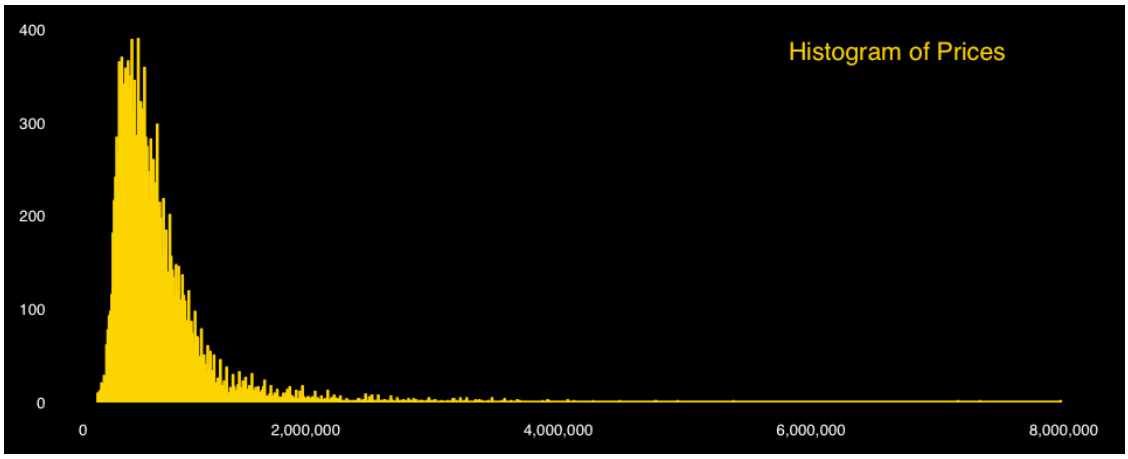
As a first stage approach, the initial **correlation matrix** is plotted. Based on that we can notice that:

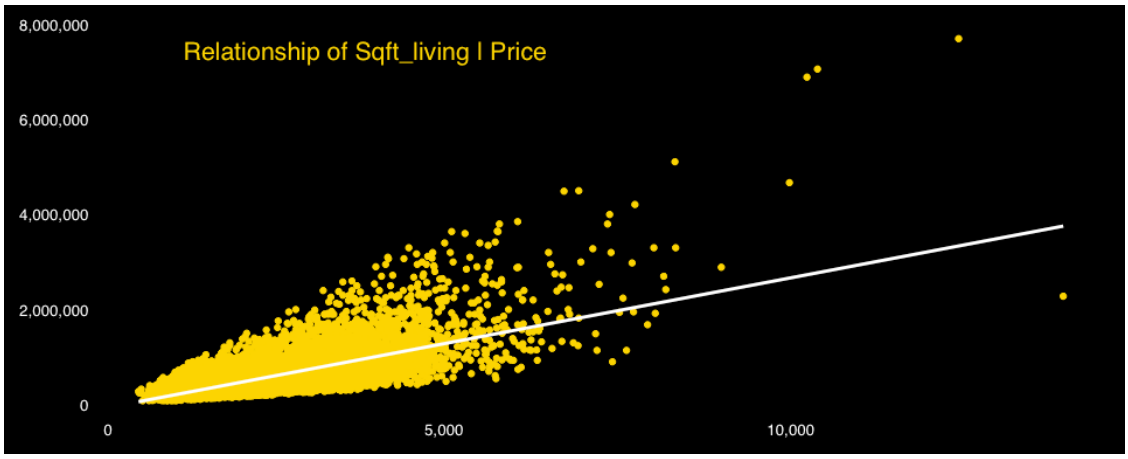
- **Price** is highly correlated with the **sqft_living** variables, something that we would expect from intuition.
- **Sqft_living** is highly correlated with the **sqft_above**

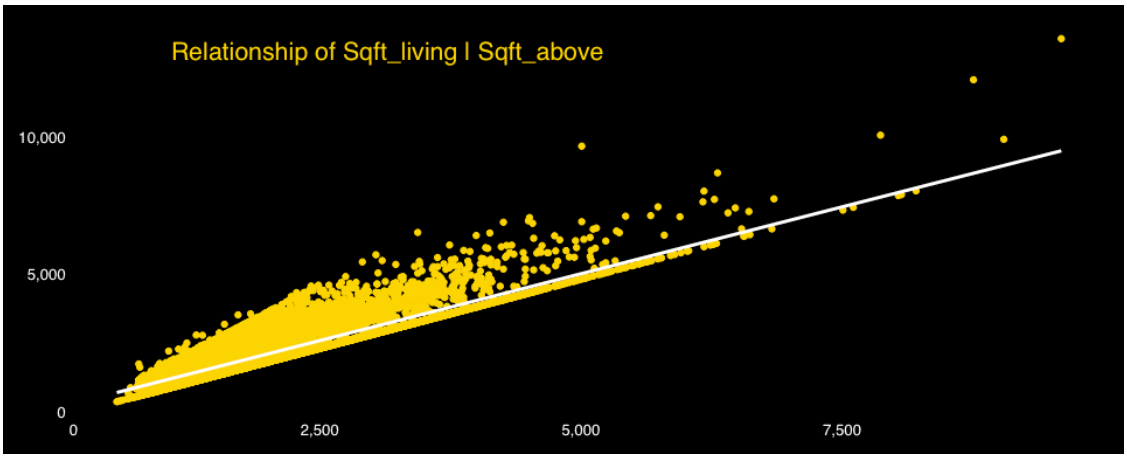
Furthermore, in order to understand our target variable better, the histogram of the prices is plotted. Referring to that we notice that the distribution of the prices is heavily right skewed, which makes sense as there are few really expensive ones.

In order to help our models predictability and performance, at this stage it is decided to apply the **log** transformation on the target column, but also make sure we undo that after the predictions by applying the **exp** function.



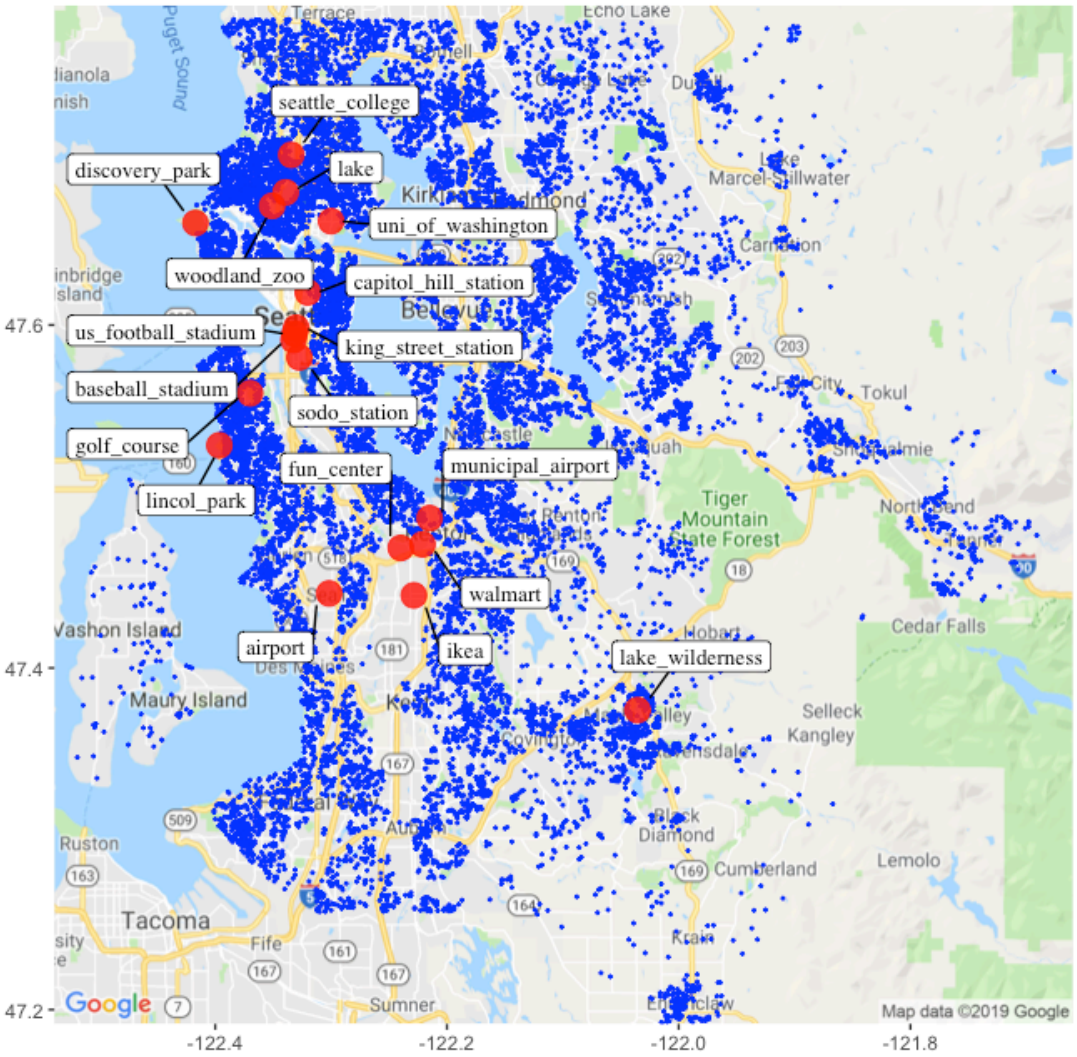

Histogram of Prices
400
300
200
100
0
0
2,000,000
4,000,000
6,000,000
8,000,000


Relationship of Sqft_living I Price
8,000,000
6,000,000
4,000,000
2,000,000
0
0
5,000
10,000


Relationship of Sqft_living I Sqft_above
10,000
5,000
0
0
2,500
5,000
7,500

After having some first point of views on our dataset, it is time to take advantage of the coordinates given for the houses sold. By using the **Google Maps API** through the **ggmap** of ggplot2 in R, we depict different maps of the location of the places. Below 3 different maps can be noticed:

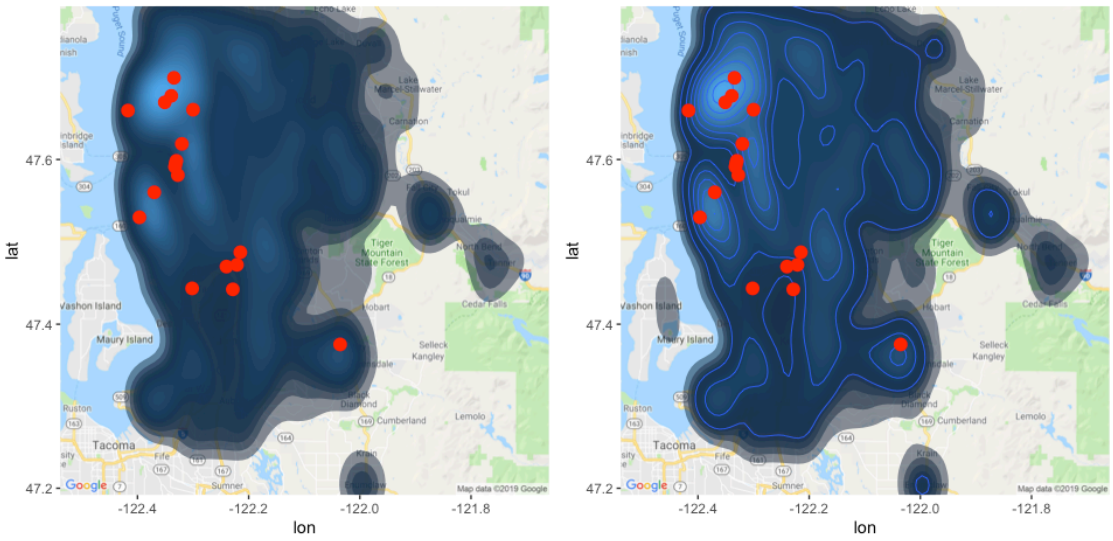
- First we have a simple roadmap with the location as points.
- Second, we have a density map of the points.
- Third we have again a density map, but with the different density polygons shaped.



After creating the initial map, it is observed that some of the points are concentrated in specific areas. By actually observing manually the google maps, points of interest are defined which are also depicted and are going to be used as part of the feature engineering process later on.

These spots are:

- **Airport**
- **IKEA**
- **Fun Center**
- **Municipal Airport**
- **WalMart**
- **University of Washington**
- **Seattle College**
- **Woodland Zoo**
- **Lake**
- **US football stadium**
- **Discovery Park**
- **Lincoln Park**
- **Baseball Stadium**
- **King Street Station**
- **Sodo Station**
- **Capitol Hill Station**
- **Golf Course**
- **Lake Wilderness**



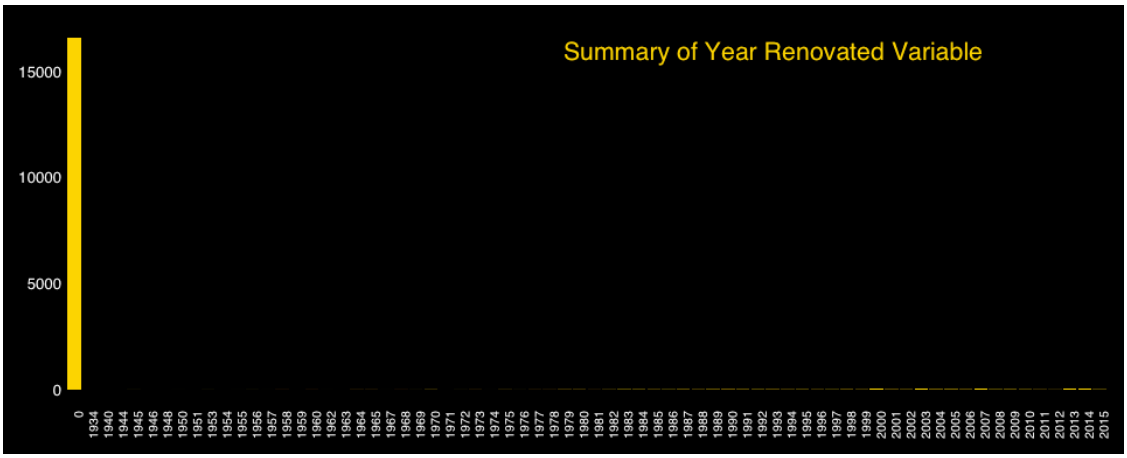
Feature Engineering Process

Having a great overview of our dataset both from an intuitive, but also from a machine learning perspective, it is time to move on to taking care of our dataset.

Some of the obvious features that are going to be created are:

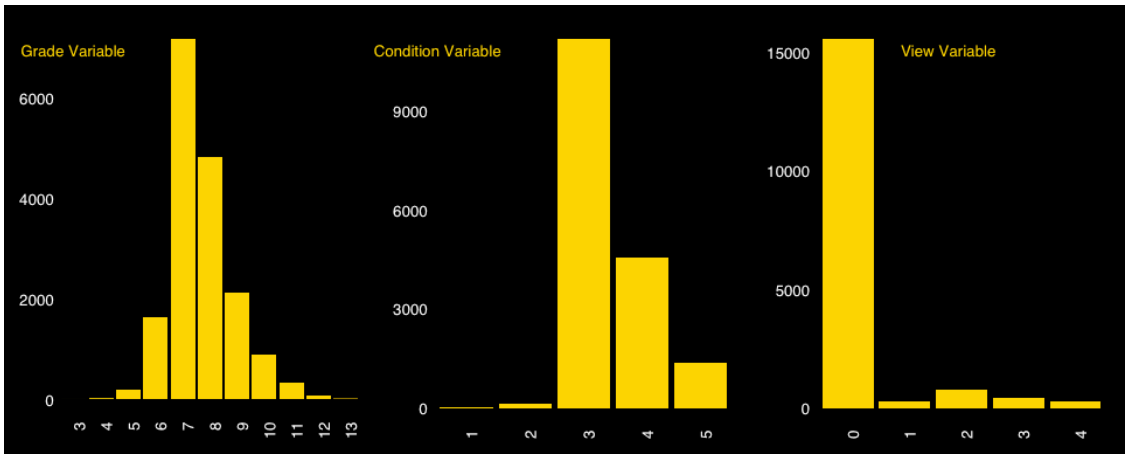
- **Month** out of the date.
- **Year** out of the date.
- **Age when sold** which comes out of year minus year built
- **Age till today** which comes out of the year 2019 minus year built.

Furthermore, while looking at the description of the variables, it is noticed that even though some houses might now have a year of renovation, they might actually do. More specifically, if the values in **sqft_living15** and **sqft_lot15** do not much the ones in **sqft_living** and **sqft_lot**, it is implied that the house has been renovated at some point. This new insight affects also the variable **yr_renovation**, as it is decided to be dropped cause of many zero values (*as seen in the plot below*) and be replaced by a new one called again **yr_renovated**, but actually consisting of a binary variable on *whether the house has been under renovation or not*. The binary value is actually been calculated by checking the difference between the variables mentioned imidiatelly before.

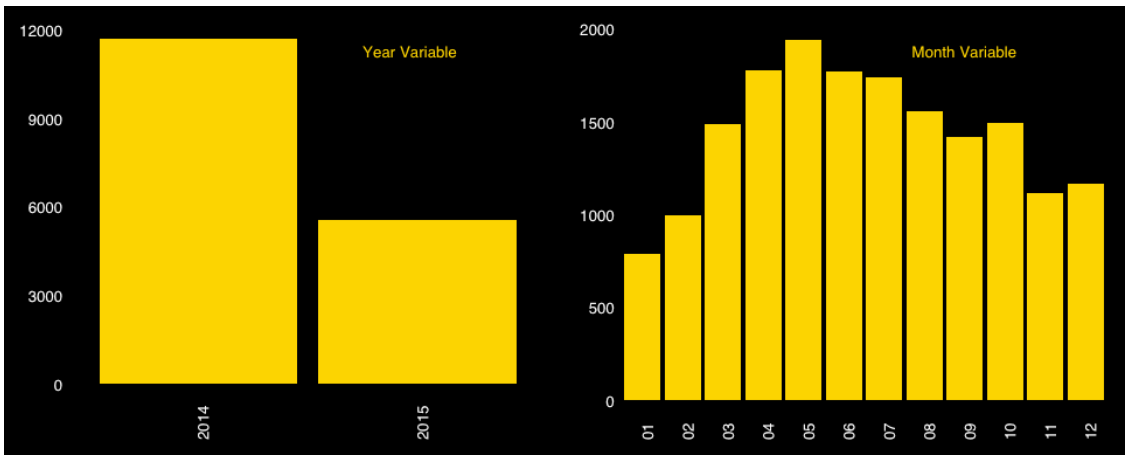


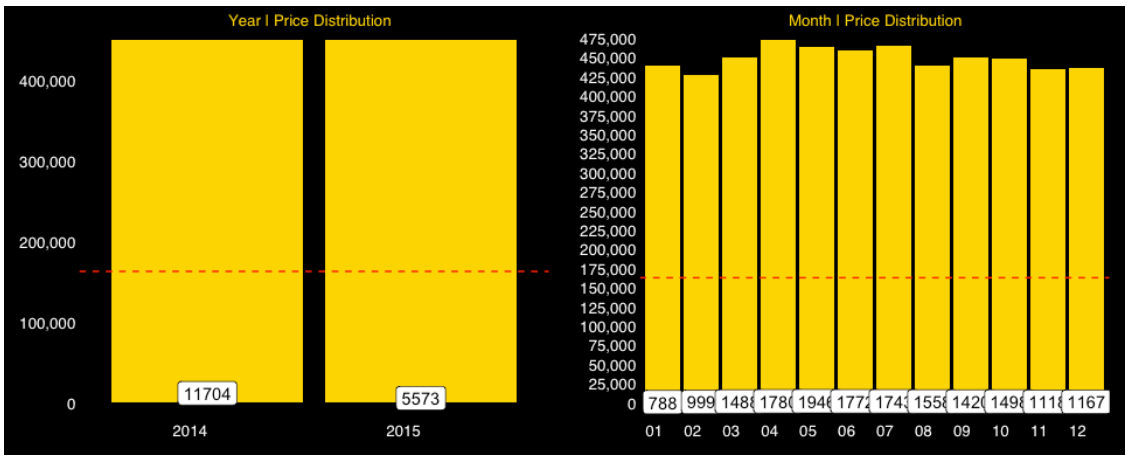
As already discussed before, some variables are not just numerical, but have a deeper meaning that classifies them as **ordinal** and **factors**. These variables are the below and are transformed and plotted in order to get a better overview of their behavior and distribution across our dataset.

- **Grade**
- **Condition**
- **View**
- **Year** and **Month**, as they are not useful as numeric anymore after the columns needed are created.



Another thing that would be useful to dive into is the distribution of **Year** along with the **Price**, but also the one between **Month** with the **Price**.





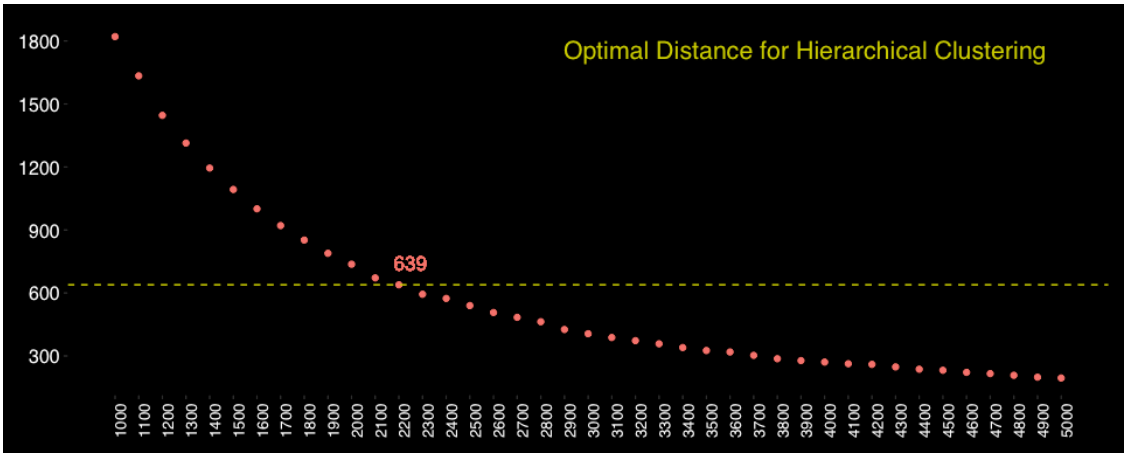
As previously mentioned, while exploring the geospatial dimension of our dataset, several possible hotspots were detected. At this time, we calculate the distances of all the houses from all the hotspots. The type of distance measurement method used, is the **haversine** that is more accurate when calculating distances on a sphere. The coordinates along with the point are shown in the table below.

name	long	lat
airport	-122.3017	47.44355
ikea	-122.2286	47.44239
fun_center	-122.2399	47.47006
municipal_airport	-122.2150	47.48749
walmart	-122.2211	47.47218
uni_of_washington	-122.3000	47.66035
seattle_college	-122.3344	47.69894
woodland_zoo	-122.3509	47.66926

Hotspot Coordinates (Scrollable Rmd) 1

Now that we have the distance from all the hotspots, another next step is to apply clustering to our houses. In order to cluster them, we are going to use the **leaderCluster** package in which we have the option to cluster points based on a **radius** that we define. This approach makes more sense as we do not choose the amount of clusters beforehand and we also use radius that is more reasonable for clustering static points referring to the location of houses. More on that, the algorithm gives us the **number of clusters** in which we do not have any predefined influence (like in K-Means).

In order to determine the optimal number for the radius (in metres), different values are tried and the below *scatterplot* is shown. Based on trial and error, along with some empirical rules, the **knee** of the plot is chosen as the optimal to be used for the clustering. In our case we choose **2200** as our radius.



Some further variables that come to play are:

- **new**: whether the house is new or not. This is defined by checking if the **yr_built** is the same as **year**.

- **total_area**: is pretty much described by the name itself and is calculated by the sum of **sqft_living** and the **sqft_basement**.

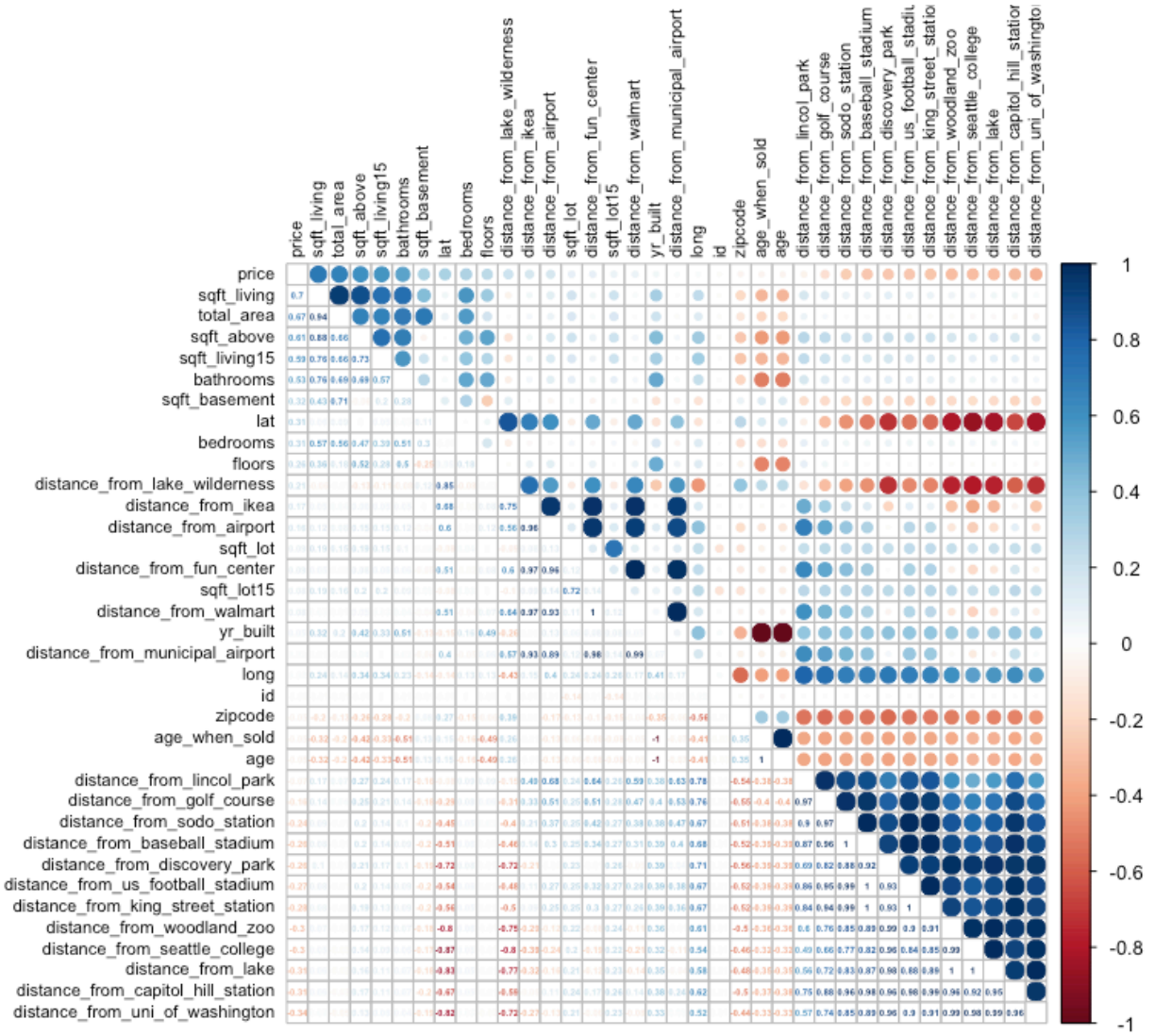
Now that we have all our variables that came up from the Feature Engineering step. It is time to have an overview of our dataset once more. Not only the **summary** and **structure** but also the new **correlation matrix**.

```
## 'data.frame':    17277 obs. of  46 variables:
##  $ id                                : num  9.18e+09 4.64e+08 2.22e+09
6.16e+09 6.39e+09 ...
##  $ date                              : Date, format: "2014-05-13" "2014-
08-27" ...
##  $ price                             : num  225000 641250 810000 330000
530000 ...
##  $ bedrooms                          : num  3 3 4 4 4 4 4 3 4 3 ...
##  $ bathrooms                         : num  1.5 2.5 3.5 1.5 1.75 3.5 3.25
2.25 2.5 1.5 ...
##  $ sqft_living                       : num  1250 2220 3980 1890 1814 ...
##  $ sqft_lot                          : num  7500 2550 209523 7540 5000 ...
##  $ floors                            : num  1 3 2 1 1 2 2 1 2 1 ...
##  $ waterfront                        : chr  "0" "0" "0" "0" ...
##  $ view                              : chr  "0" "2" "2" "0" ...
##  $ condition                         : chr  "3" "3" "3" "4" ...
##  $ grade                             : chr  "7" "10" "9" "7" ...
##  $ sqft_above                        : num  1250 2220 3980 1890 944 2480
4160 1130 2250 1500 ...
##  $ sqft_basement                     : num  0 0 0 0 870 640 0 310 0 1040
...
##  $ yr_built                          : num  1967 1990 2006 1967 1951 ...
##  $ yr_renovated                      : chr  "YES" "YES" "YES" "YES" ...
##  $ zipcode                           : num  98030 98117 98024 98155 98115
...
##  $ lat                               : num  47.4 47.7 47.6 47.8 47.7 ...
##  $ long                              : num  -122 -122 -122 -122 -122 ...
##  $ sqft_living15                     : num  1260 2200 2220 1890 1290 1880
3400 1510 2480 1870 ...
##  $ sqft_lot15                        : num  7563 5610 65775 8515 5000 ...
##  $ month                             : chr  "05" "08" "07" "01" ...
##  $ year                              : chr  "2014" "2014" "2014" "2015"
...
##  $ age_when_sold                     : num  47 24 8 48 63 6 19 31 27 55
...
##  $ age                               : num  52 29 13 52 68 11 24 36 32 60
...
##  $ distance_from_airport             : num  10.3 29 33.5 34.5 26.8 ...
##  $ distance_from_ikea                : num  7.91 30.85 28.5 35.27 27.18
...
##  $ distance_from_fun_center          : num  11.1 27.7 28 32.1 24 ...
##  $ distance_from_municipal_airport   : num  12.9 26.8 25.6 30.6 22.4 ...
##  $ distance_from_walmart             : num  11.2 28.1 26.6 32.1 24 ...
##  $ distance_from_uni_of_washington   : num  32.74 8.04 32.84 10.45 2.99
...
##  $ distance_from_seattle_college     : num  37.5 4.4 36.87 6.19 4.33 ...
```

```
##  $ distance_from_woodland_zoo        : num  34.64 4.36 36.76 9.69 5.49 ...
##  $ distance_from_lake                : num  35.24 4.56 36.24 8.64 4.43 ...
##  $ distance_from_us_football_stadium : num  26.3 12.2 33.4 17.7 10.6 ...
##  $ distance_from_discovery_park      : num  35.42 4.49 41.15 12.83 10.56
...
##  $ distance_from_lincol_park         : num  22.3 18.5 38.2 25.6 19.2 ...
##  $ distance_from_baseball_stadium    : num  26 12.5 33.4 18.1 11 ...
##  $ distance_from_king_street_station : num  26.7 11.9 33.4 17.3 10.2 ...
##  $ distance_from_sodo_station        : num  24.8 13.7 33 19.2 12 ...
##  $ distance_from_capitol_hill_station: num  28.62 10.2 33.02 14.97 7.82
...
##  $ distance_from_golf_course         : num  24 15.2 36 21.9 15.3 ...
##  $ distance_from_lake_wilderness     : num  13.5 44.7 23 47.1 39 ...
##  $ cluster                           : int  1 2 3 4 5 5 6 7 8 9 ...
##  $ new                               : chr  "NO" "NO" "NO" "NO" ...
##  $ total_area                        : num  1250 2220 3980 1890 2684 ...
```

	id	date	price	bedrooms	bathrooms	sqft_living	sc
	Min. :1.000e+06	Min. :2014-05-02	Min. : 78000	Min. : 1.000	Min. :0.500	Min. : 370	M 5:
	1st Qu.:2.114e+09	1st Qu.:2014-07-21	1st Qu.: 320000	1st Qu.: 3.000	1st Qu.:1.750	1st Qu.: 1430	1: 5(
	Median :3.902e+09	Median :2014-10-16	Median : 450000	Median : 3.000	Median :2.250	Median : 1910	M 7(
	Mean :4.566e+09	Mean :2014-10-28	Mean : 539865	Mean : 3.369	Mean :2.114	Mean : 2080	M 1:
	3rd Qu.:7.303e+09	3rd Qu.:2015-02-17	3rd Qu.: 645500	3rd Qu.: 4.000	3rd Qu.:2.500	3rd Qu.: 2550	3(1(
	Max. :9.900e+09	Max. :2015-05-27	Max. :7700000	Max. :33.000	Max. :8.000	Max. :13540	M :1

Summary of Dataset (Scrollable in Rmd) 2



Now that we have all the necessary content, it is time to convert it to the appropriate format. This means:

- **One hot encoding** of the factor variables
- **Scaling numerical variables** from 0 to 1
- Applying the **logarithmic** transformation on the price to remove the effect that the right skewed distribution might have to our predictive models.

Machine Learning Pipeline

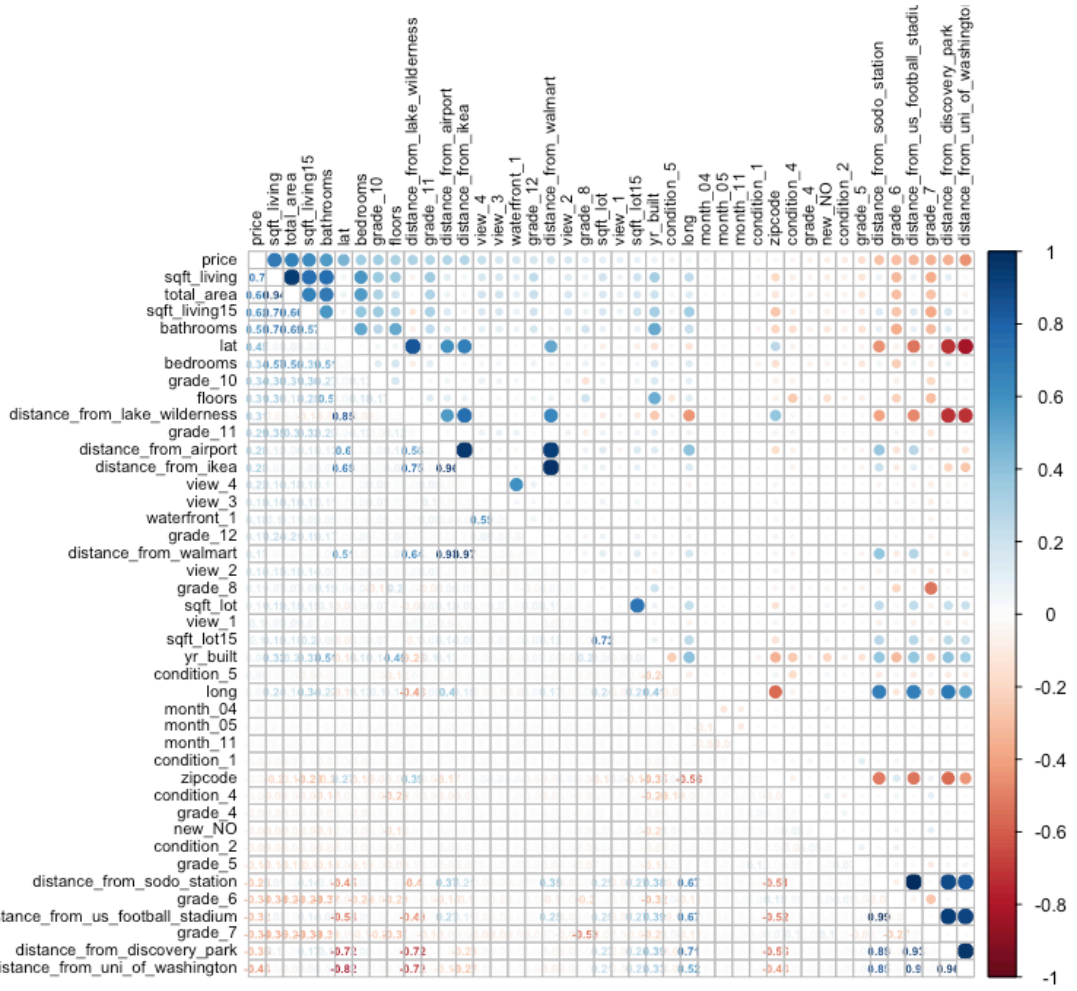
Dealing with Multicollinearity | Applying Lasso Regression Model

One of the first things we need to do in order to make sure the models do not **overfit**, is to deal with the issue of *multicollinearity* that is obviously appearing in our dataset, just by looking at the correlation matrix immediately above. To find the perfect set of variables, a **lasso regression** model is created that is not only trained to predict, but is able to eliminate unnecessary variables. To train the model a **grid search** is applied to find the optimal set of parameters. The variables and the model are shown below, along with the new correlation amtrix with the variables kept.

*(the predictions are stored in the **predictions_lasso** variable)*

```
## Lasso uses 48 variables in its model, and did not select 28 variables.
```

```
## glmnet
##
## 17277 samples
##    48 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 13822, 13822, 13821, 13822, 13821
## Resampling results:
##
##   RMSE    Rsquared  MAE
##   0.1983  0.8591    0.1486
##
## Tuning parameter 'alpha' was held constant at a value of 1
##
## Tuning parameter 'lambda' was held constant at a value of 0.001
```



Applying Recursive Feature Elimination | Random Forest modelling & prediction

Apart from just the set of variables that lasso regression comes up with, a **Recursive Feature Elimination (RFE)** is done to our full dataset, in order to both determine the optimal variables, but also the set of parameters for a random forest algorithm (which is the algorithm used from RFE variable selection).

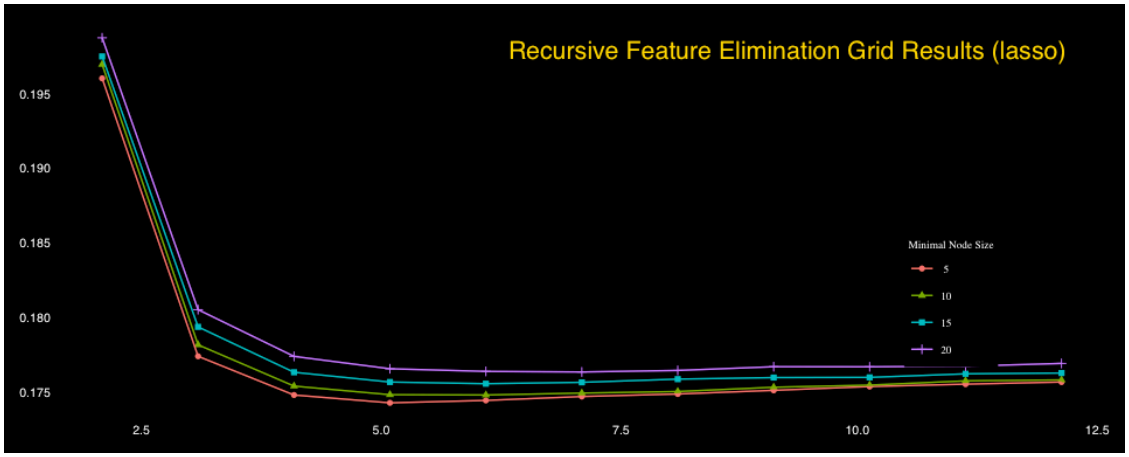
In order to cover all different possibilities, another **RFE approach** is followed apart from the one on the **whole dataset**. The other one is performing an **RFE analysis** on only the variables that came out as important from the **Lasso regression elimination.** The results are shown below and the plots are of 2 kinds. :

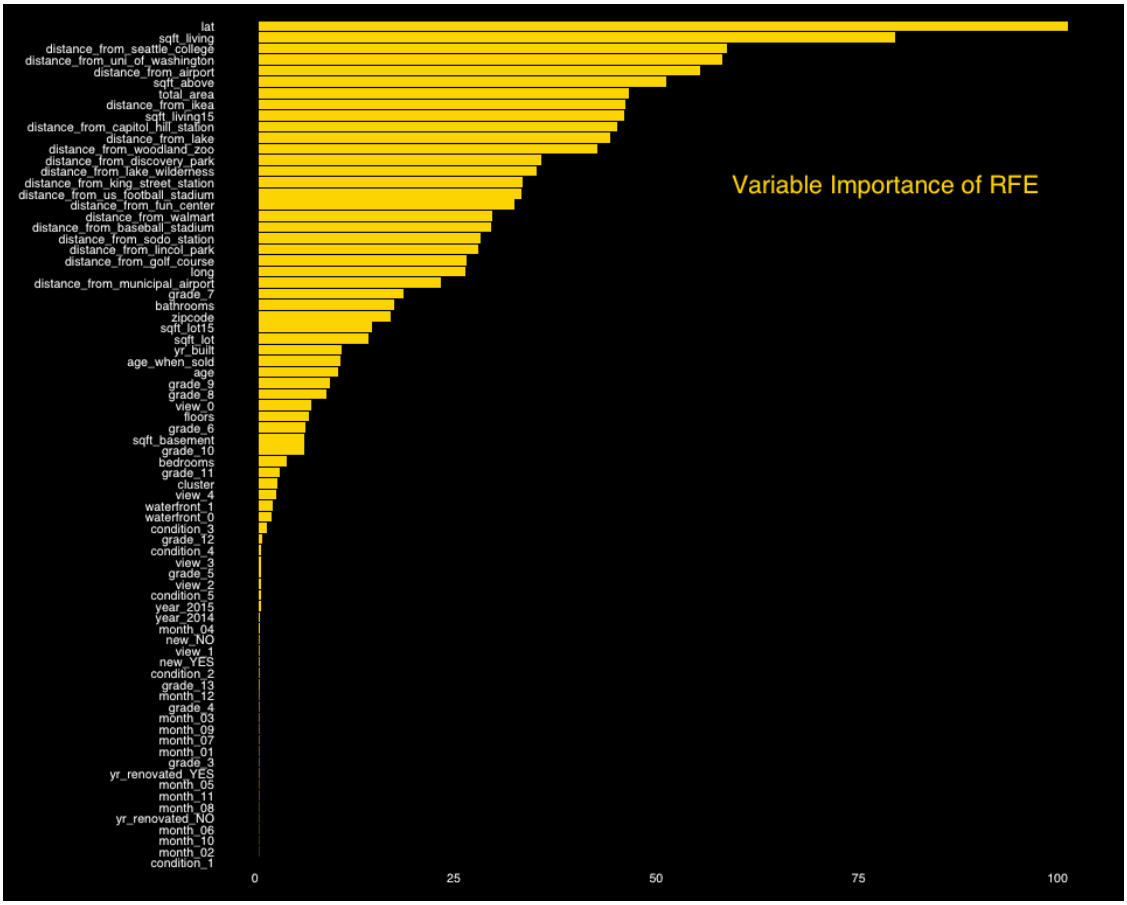
- The first plot shows the performance of the model based on **RMSE** metric, along with the **min.node.size** parameter.

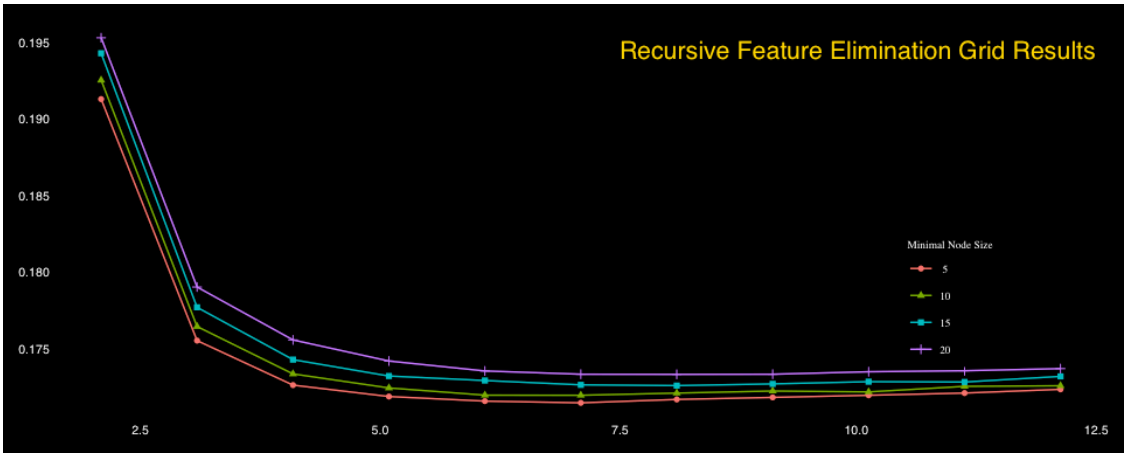
- The second plot shows the **variable importance** that comes out of the RFE method.

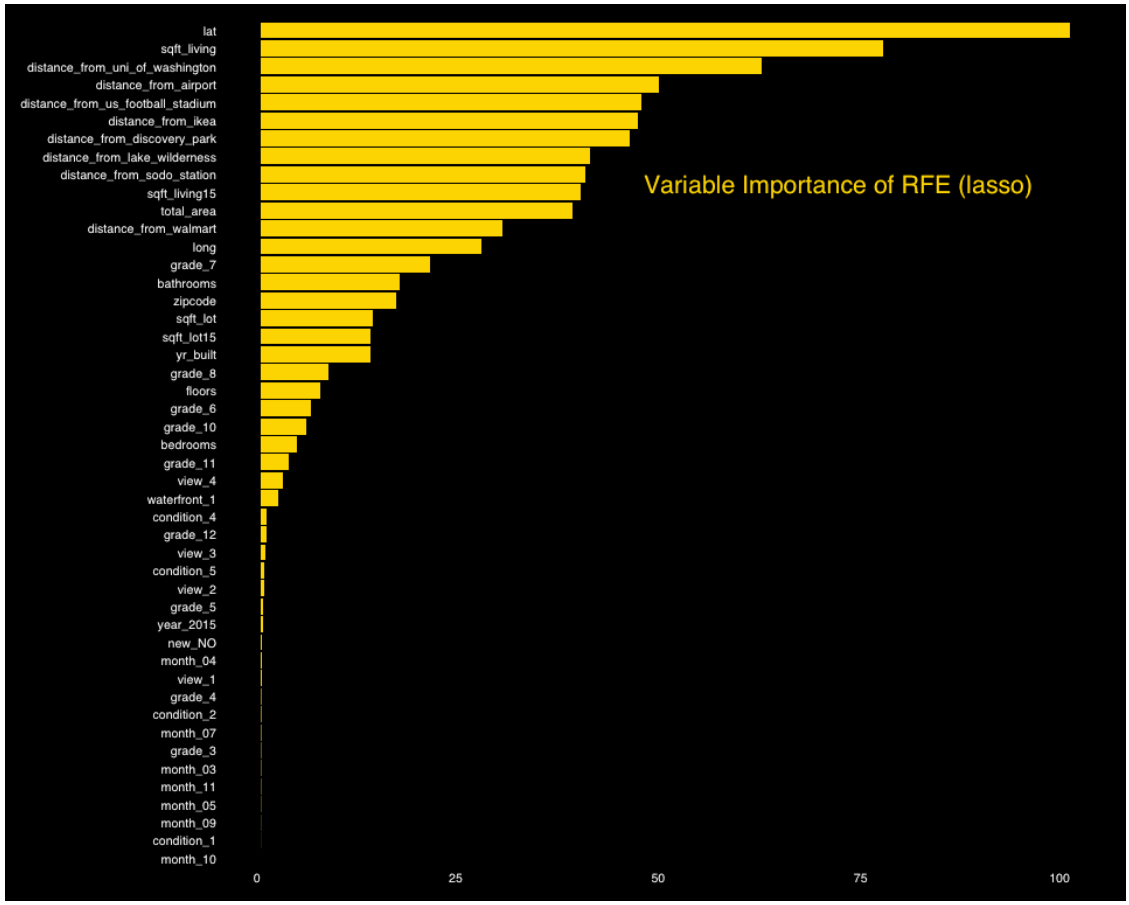
```
## Random Forest
##
## 17277 samples
##    76 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 13822, 13821, 13823, 13821, 13821
## Resampling results across tuning parameters:
##
##   mtry  min.node.size  RMSE    Rsquared  MAE
##    2     5             0.1913  0.8902    0.1345
##    2    10             0.1925  0.8894    0.1355
##    2    15             0.1943  0.8882    0.1367
##    2    20             0.1953  0.8871    0.1374
##    3     5             0.1755  0.8962    0.1221
##    3    10             0.1764  0.8958    0.1228
##    3    15             0.1777  0.8947    0.1238
##    3    20             0.1790  0.8938    0.1249
##    4     5             0.1726  0.8972    0.1199
##    4    10             0.1733  0.8967    0.1205
##    4    15             0.1743  0.8961    0.1214
##    4    20             0.1755  0.8951    0.1224
##    5     5             0.1718  0.8971    0.1195
##    5    10             0.1724  0.8967    0.1200
##    5    15             0.1732  0.8962    0.1207
##    5    20             0.1742  0.8954    0.1216
##    6     5             0.1715  0.8969    0.1194
##    6    10             0.1719  0.8967    0.1199
##    6    15             0.1729  0.8958    0.1206
##    6    20             0.1735  0.8954    0.1213
##    7     5             0.1714  0.8966    0.1195
##    7    10             0.1719  0.8963    0.1200
##    7    15             0.1726  0.8957    0.1206
##    7    20             0.1733  0.8951    0.1212
##    8     5             0.1717  0.8961    0.1197
##    8    10             0.1721  0.8958    0.1202
##    8    15             0.1726  0.8954    0.1207
##    8    20             0.1733  0.8948    0.1214
##    9     5             0.1718  0.8957    0.1199
```

```
##     9    10             0.1722  0.8954    0.1204
##     9    15             0.1727  0.8950    0.1209
##     9    20             0.1733  0.8945    0.1215
##    10     5             0.1719  0.8954    0.1201
##    10    10             0.1721  0.8953    0.1204
##    10    15             0.1728  0.8946    0.1211
##    10    20             0.1735  0.8940    0.1217
##    11     5             0.1721  0.8951    0.1203
##    11    10             0.1725  0.8947    0.1208
##    11    15             0.1728  0.8945    0.1211
##    11    20             0.1735  0.8938    0.1218
##    12     5             0.1723  0.8947    0.1205
##    12    10             0.1726  0.8945    0.1209
##    12    15             0.1732  0.8939    0.1214
##    12    20             0.1737  0.8935    0.1219
##
## Tuning parameter 'splitrule' was held constant at a value of variance
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were mtry = 7, splitrule =
##  variance and min.node.size = 5.
```




Variable Importance of RFE
lat
sqft_living
distance_from_seattle_college
distance_from_uni_of_washington
distance_from_airport
sqft_above
total_area
distance_from_ikea
sqft_living15
distance_from_capitol_hill_station
distance_from_lake
distance_from_woodland_zoo
distance_from_discovery_park
distance_from_lake_wilderness
distance_from_king_street_station
distance_from_us_football_stadium
distance_from_fun_center
distance_from_walmart
distance_from_baseball_stadium
distance_from_sodo_station
distance_from_lincol_park
distance_from_golf_course
long
distance_from_municipal_airport
grade_7
bathrooms
zipcode
sqft_lot15
sqft_lot
yr_built
age_when_sold
age
grade_9
grade_8
view_0
floors
grade_6
sqft_basement
grade_10
bedrooms
grade_11
cluster
view_4
waterfront_1
waterfront_0
condition_3
grade_12
condition_4
view_3
grade_5
view_2
condition_5
year_2015
year_2014
month_04
new_NO
view_1
new_YES
condition_2
grade_13
month_12
grade_4
month_03
month_09
month_07
month_01
grade_3
yr_renovated_YES
month_05
month_11
month_08
yr_renovated_NO
month_06
month_10
month_02
condition_1
0
25
50
75
100


Recursive Feature Elimination Grid Results
0.195
0.190
0.185
0.180
0.175
Minimal Node Size
5
10
15
20
2.5
5.0
7.5
10.0
12.5



Based on the results shown above the optimal set of parameters is:

- **mtry** = 7
- **splitrule** = variance
- **min.node.size** = 5.

Furthermore, the number of variables that seem to be important are **46** on the one case and **26** on the lasso variables set and are depicted on the following table:

	Importance RFE
lat	100.0000000
sqft_living	78.6635231
distance_from_seattle_college	57.8078595
distance_from_uni_of_washington	57.2729206
distance_from_airport	54.5906466
sqft_above	50.4365000
total_area	45.7571791
distance_from_ikea	45.2591976

	Importance RFE Lasso
lat	100.0000000
sqft_living	76.9222295
distance_from_uni_of_washington	61.8814926
distance_from_airport	49.1568816
distance_from_us_football_stadium	47.0835634
distance_from_ikea	46.6801766
distance_from_discovery_park	45.5734396
distance_from_lake_wilderness	40.7768531

RFE importances (Scrollable in Rmd) 1

At this stage, it is decided to train a **RandomForest**, based on the parameters and variables found below. The results of the model are shown below.

(the predictions are stored in the **predictions_rf** and **predictions_rf_lasso** variable)

Random Forest with RFE

```
## Random Forest
##
## 17277 samples
##    42 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 13822, 13822, 13821, 13821, 13822
## Resampling results:
##
##   RMSE   Rsquared  MAE
##   0.174  0.8924    0.1217
##
## Tuning parameter 'mtry' was held constant at a value of 7
## Tuning
##  parameter 'splitrule' was held constant at a value of variance
##
## Tuning parameter 'min.node.size' was held constant at a value of 5
```

Random Forest with RFE(lasso)

```
## Random Forest
##
## 17277 samples
##    26 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 13822, 13822, 13821, 13821, 13822
## Resampling results:
##
##   RMSE    Rsquared  MAE
##   0.1794  0.8851    0.1256
##
## Tuning parameter 'mtry' was held constant at a value of 7
## Tuning
##  parameter 'splitrule' was held constant at a value of variance
##
## Tuning parameter 'min.node.size' was held constant at a value of 5
```

XGBoost Modelling & Prediction

Another algorithm that is proved really strong perfomer, is the **XGBoost** one (Extreme Gradient Boosting). The algorithm is trained on the **RFE features** both of lasso and of the full train set and an extensive **grid search** is applied, in order to determine the optimal set of parameters. The results are show below.

(the predictions are stored in the **predictions_xgb** and **predictions_xgb_lasso** variable)

Extreme Gradient Boosting with RFE

```
## eXtreme Gradient Boosting
##
## 17277 samples
##    43 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 13822, 13823, 13822, 13820, 13821
## Resampling results across tuning parameters:
##
##   eta   max_depth  min_child_weight  RMSE    Rsquared  MAE
##   0.01  2          1                 0.1984  0.8604    0.1458
##   0.01  2          2                 0.1984  0.8604    0.1457
##   0.01  2          3                 0.1984  0.8604    0.1457
##   0.01  2          4                 0.1984  0.8604    0.1457
##   0.01  2          5                 0.1984  0.8604    0.1457
##   0.01  3          1                 0.1835  0.8796    0.1336
##   0.01  3          2                 0.1836  0.8796    0.1336
##   0.01  3          3                 0.1835  0.8797    0.1336
##   0.01  3          4                 0.1835  0.8797    0.1336
##   0.01  3          5                 0.1835  0.8797    0.1336
##   0.01  4          1                 0.1764  0.8885    0.1277
##   0.01  4          2                 0.1764  0.8885    0.1277
##   0.01  4          3                 0.1763  0.8886    0.1276
##   0.01  4          4                 0.1762  0.8888    0.1275
##   0.01  4          5                 0.1762  0.8887    0.1276
##   0.01  5          1                 0.1724  0.8934    0.1239
##   0.01  5          2                 0.1724  0.8934    0.1239
##   0.01  5          3                 0.1722  0.8936    0.1239
##   0.01  5          4                 0.1722  0.8936    0.1239
##   0.01  5          5                 0.1721  0.8938    0.1239
##   0.01  6          1                 0.1699  0.8965    0.1215
##   0.01  6          2                 0.1702  0.8961    0.1216
##   0.01  6          3                 0.1699  0.8965    0.1215
##   0.01  6          4                 0.1698  0.8966    0.1215
##   0.01  6          5                 0.1696  0.8968    0.1214
```

```
##   0.05  2           1           0.1733  0.8923    0.1248
##   0.05  2           2           0.1733  0.8923    0.1248
##   0.05  2           3           0.1732  0.8924    0.1248
##   0.05  2           4           0.1731  0.8925    0.1247
##   0.05  2           5           0.1731  0.8925    0.1247
##   0.05  3           1           0.1688  0.8977    0.1207
##   0.05  3           2           0.1686  0.8980    0.1206
##   0.05  3           3           0.1686  0.8979    0.1206
##   0.05  3           4           0.1685  0.8980    0.1205
##   0.05  3           5           0.1684  0.8982    0.1204
##   0.05  4           1           0.1673  0.8995    0.1191
##   0.05  4           2           0.1673  0.8995    0.1189
##   0.05  4           3           0.1673  0.8995    0.1190
##   0.05  4           4           0.1669  0.9000    0.1187
##   0.05  4           5           0.1670  0.8999    0.1188
##   0.05  5           1           0.1671  0.8999    0.1183
##   0.05  5           2           0.1671  0.8998    0.1184
##   0.05  5           3           0.1668  0.9001    0.1185
##   0.05  5           4           0.1670  0.8999    0.1184
##   0.05  5           5           0.1671  0.8998    0.1184
##   0.05  6           1           0.1679  0.8988    0.1185
##   0.05  6           2           0.1674  0.8994    0.1182
##   0.05  6           3           0.1676  0.8992    0.1183
##   0.05  6           4           0.1674  0.8995    0.1183
##   0.05  6           5           0.1675  0.8993    0.1182
##   0.10  2           1           0.1698  0.8965    0.1217
##   0.10  2           2           0.1697  0.8966    0.1216
##   0.10  2           3           0.1696  0.8968    0.1216
##   0.10  2           4           0.1694  0.8970    0.1215
##   0.10  2           5           0.1695  0.8968    0.1215
##   0.10  3           1           0.1676  0.8992    0.1194
##   0.10  3           2           0.1677  0.8990    0.1194
##   0.10  3           3           0.1678  0.8990    0.1195
##   0.10  3           4           0.1677  0.8991    0.1194
##   0.10  3           5           0.1678  0.8989    0.1195
##   0.10  4           1           0.1682  0.8986    0.1194
##   0.10  4           2           0.1685  0.8982    0.1194
##   0.10  4           3           0.1689  0.8977    0.1198
##   0.10  4           4           0.1684  0.8983    0.1194
##   0.10  4           5           0.1684  0.8982    0.1193
##   0.10  5           1           0.1698  0.8966    0.1197
##   0.10  5           2           0.1698  0.8966    0.1199
##   0.10  5           3           0.1693  0.8972    0.1198
##   0.10  5           4           0.1698  0.8966    0.1199
##   0.10  5           5           0.1701  0.8963    0.1200
##   0.10  6           1           0.1710  0.8952    0.1207
```

```
##   0.10  6          2                0.1706  0.8956    0.1201
##   0.10  6          3                0.1711  0.8951    0.1207
##   0.10  6          4                0.1709  0.8953    0.1203
##   0.10  6          5                0.1709  0.8953    0.1205
##
## Tuning parameter 'nrounds' was held constant at a value of 1000
##
## Tuning parameter 'colsample_bytree' was held constant at a value of
##  1
## Tuning parameter 'subsample' was held constant at a value of 1
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were nrounds = 1000, max_depth =
##  5, eta = 0.05, gamma = 0, colsample_bytree = 1, min_child_weight = 3
##  and subsample = 1.
```

Extreme Gradient Boosting with RFE(lasso)

```
## eXtreme Gradient Boosting
##
## 17277 samples
##    26 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 13821, 13821, 13822, 13821, 13823
## Resampling results:
##
##   RMSE   Rsquared  MAE
##   0.169  0.8975    0.1199
##
## Tuning parameter 'nrounds' was held constant at a value of 1000
##  1
## Tuning parameter 'min_child_weight' was held constant at a value of
##  3
## Tuning parameter 'subsample' was held constant at a value of 1
```

Final Metrics of all the algorithms to choose

In order to have a better view of our true algorithms performance, we test models on the **hold out set**. At this step i would like to re assure you that the test set **has not been used** at any step of the training procedure apart from this stage just for verification purposes. The script that reads the file and extracts the hold out set is the **final_metrics.R** one and is read only at this point in time.

Moreover, apart from the individual model predictions, it is decided to stack the predictions of the best 3 models using a **simple AVG** and a **weighted AVG**. The reason behind this choice is the fact that, by having a look at the predictions of the different models, it can be noticed that *lasso* along with *ranger* and *xgboost* are pretty much uncorrelated. This lead us to the conclusion that they capture different aspects of the test set. As a result an average could possibly get the best out of the three approaches.

Minimum RMSE through 10-fold Cross Validation

LASSO	XGB	XGB_LASSO	RF	RF_LASSO
0.1964	0.1668	0.169	0.174	0.1794

Overall Metrics for the models

	RMSE	MAE	MAPE
LASSO	220898	156152	0.2739
RF	165372	110364	0.2130
RF_LASSO	175149	112263	0.2152
XGB	142374	80158	0.1448
XGB_LASSO	139424	82463	0.1530
AVG_RF_XGB_LASSO	**117791**	**70740**	**0.1289**
W_AVG_RF_XGB_LASSO	120675	71875	0.1303

Predictions on the hold-out set & Submission

Based on the results of the different models on Cross Validation, we would choose the XGBoost which seems to perform better. By having a look at the predictions and performance on the actual hold out set we verify this assumption that XGboost would be the model to go, However, we choose the **Averaged Predictions of our 3 best models XGB, RF and LASSO** as it manages to do what we thought it would, which is capturing more and more of the variance of the hold out set by far better than a simple algorithm. After that we create the submission file that has 2 fields: **id** of the house, plus the predicted **price**.

Submission File

id	price
5200087	574489
11500240	835831
11520200	776025
13001215	255904
13001795	382882
13001991	276700

Conclusions

As next steps, some of which were tried but not to the full extent to the pipeline would be:

- Stacking models in order to improve predictions.(caretStack was creating errors for regression, you can find the script named as *stacking.R*)

- Applying neural networks in order to try improving the performance.(an initial approach was followed but needed a lot more optimization)

- Think of further external variables as Feature Engineering that would give more insights on the data. For example, maybe driving distance from hotspots would be more interpretable than just the haversine one (could try but google needed paying fee)