**INTRODUCTION**

Purpose of this project is to implement machine learning algorithms on a real data set Visualization and exploration of the dataset and data preparation are also the important parts of this project. Linear Regression, SVR and Random Forest have been implemented. After, tuning process of the algorithms, performance changings have been analyzed and optimal parameters have been found for the required functions of the algorithms. Performance comparison among these algorithms has been done by using different evaluation metrics. To complete all of those statistical computing and graphics operations, R language has been used.

**UNDERSTANDING THE DATA**

In this part, we take an initial look at the data understand a few basics. So, Name of the dataset that has been used in this project is “House Sales in King County, USA”. This dataset contains house sale prices for King County, which includes Seattle. It includes homes sold between May 2014 and May 2015. Thus, aim of this project is to implement machine learning models to predict the price of the houses. Also, there are 21613 rows and 21 columns in this dataset. Description for each attribute explained below:

**ID:** Unique ID for each home sold

**Date:** Date of the home sale

**Price:** Price of each home sold

**Bedrooms:** Number of bedrooms/house

**Bathrooms:** Number of bathrooms/house

**Sqft\_living:** Square footage of the apartments interior living space

**Sqft\_lot:** Square footage of the land space

**Floors:** Number of floors

**Waterfront:** A variable for whether the apartment was overlooking the waterfront or not

**View:** An index from 0 to 4 of how good the view of the property was

**Condition:** An index from 1 to 5 on the condition of the apartment

**Grade:** An index from 1 to 13, where 1-3 falls short of building construction and design, 7 has an average level of construction and design, and 11-13 have a high-quality level of construction and design.

**Aqft\_above:** The square footage of the interior housing space that is above ground level

**Aqft\_basement:** The square footage of the interior housing space that is below ground

**Yr\_built:** The year the house was initially built

**Yr\_renovated:** The year of the house’s last renovation

**Zipcode:** What zip code area the house is in

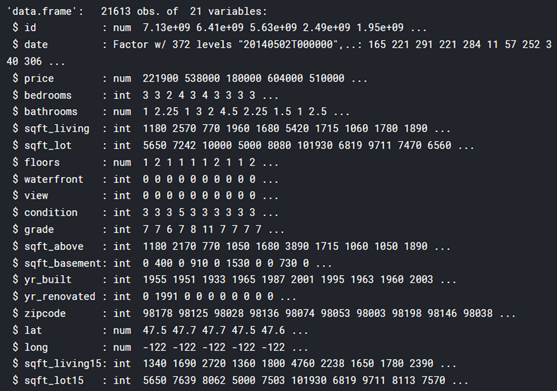
**Lat:** Latitude coordinate

**Long:** Longitude coordinate

**Sqft\_living15:** Living room area in 2015

**Sqft\_lot15:** Lot size area in 2015

Firstly, data types of the features have been checked to understand the dataset easily and to decide which visualization technique would be applied. As you can see below, except “date” feature, all of the features are numeric values. “zipcode” feature is also numeric value, but it needed to be converted to categorical feature. “date” feature also has to be parsed and converted to categorical feature.

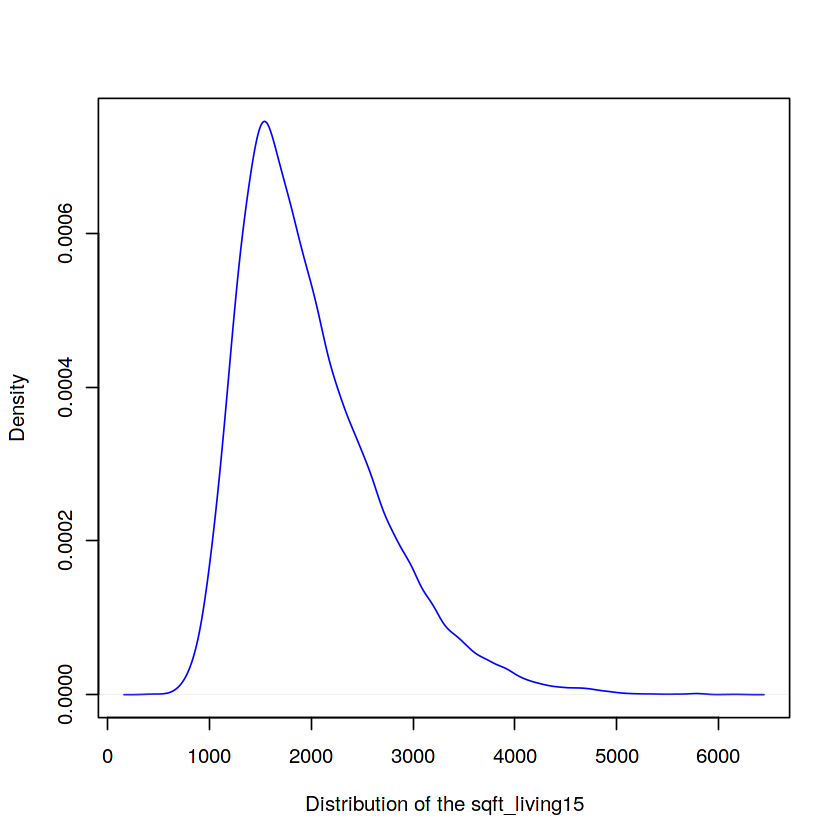
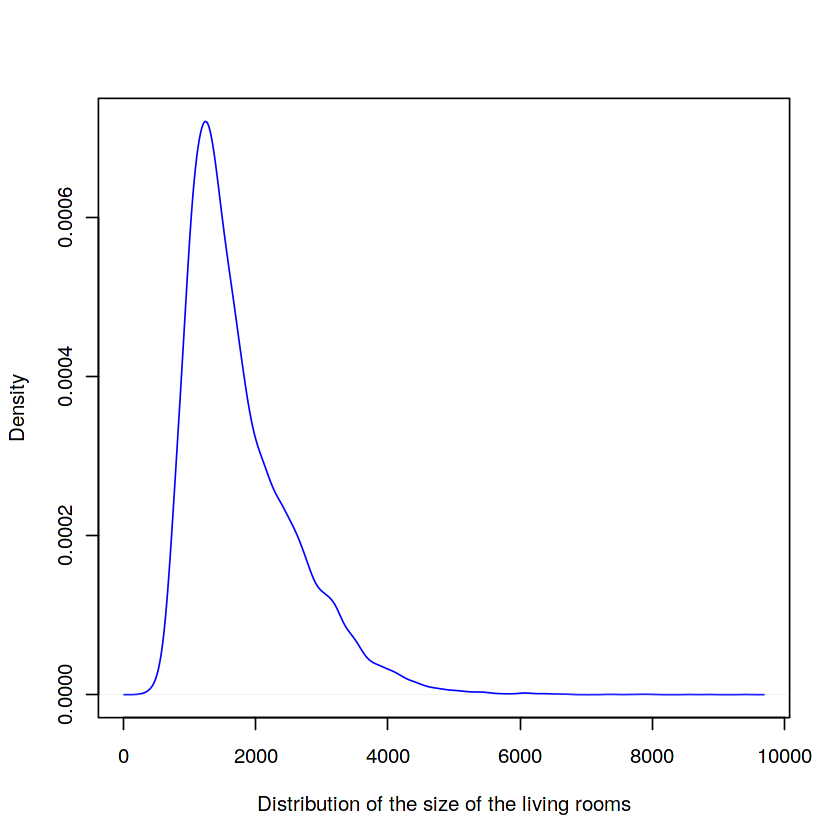
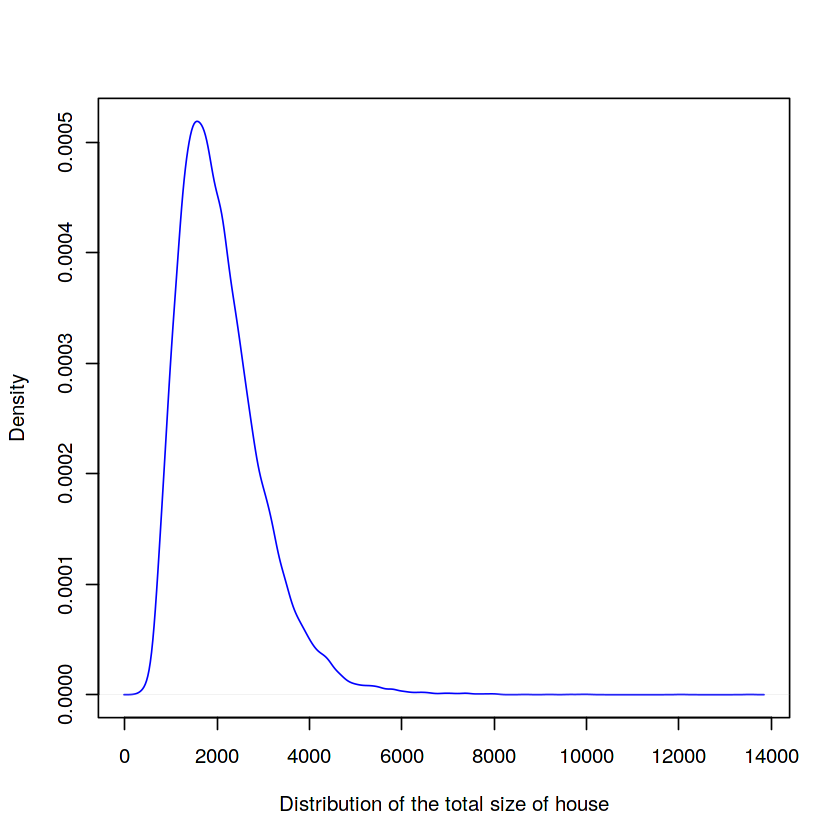
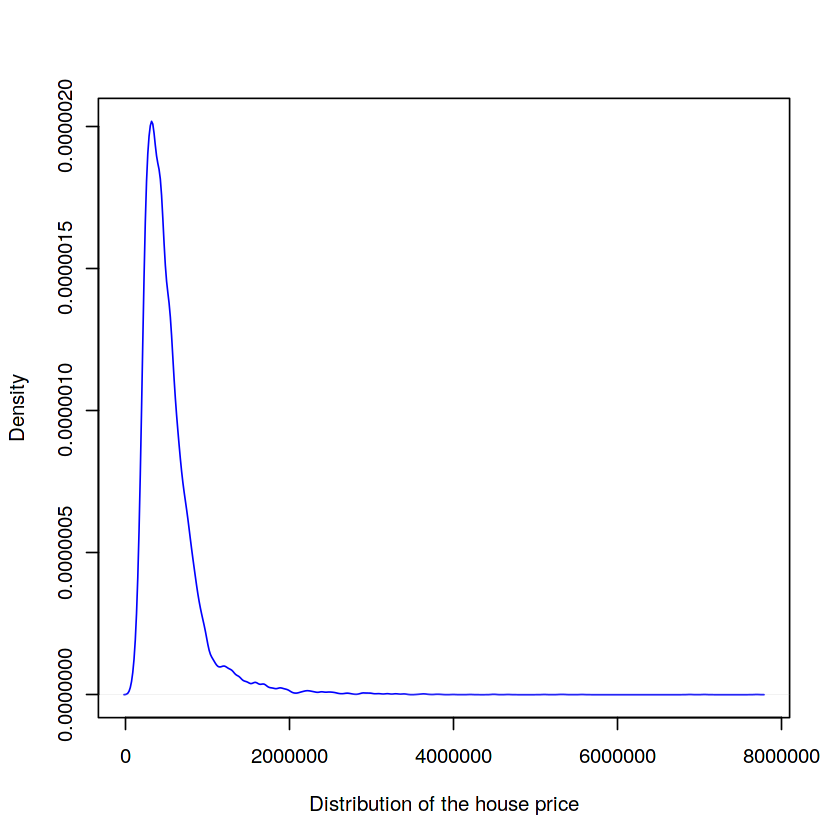


**EXPLORATORY DATA ANALYSIS AND VISUALIZATION**

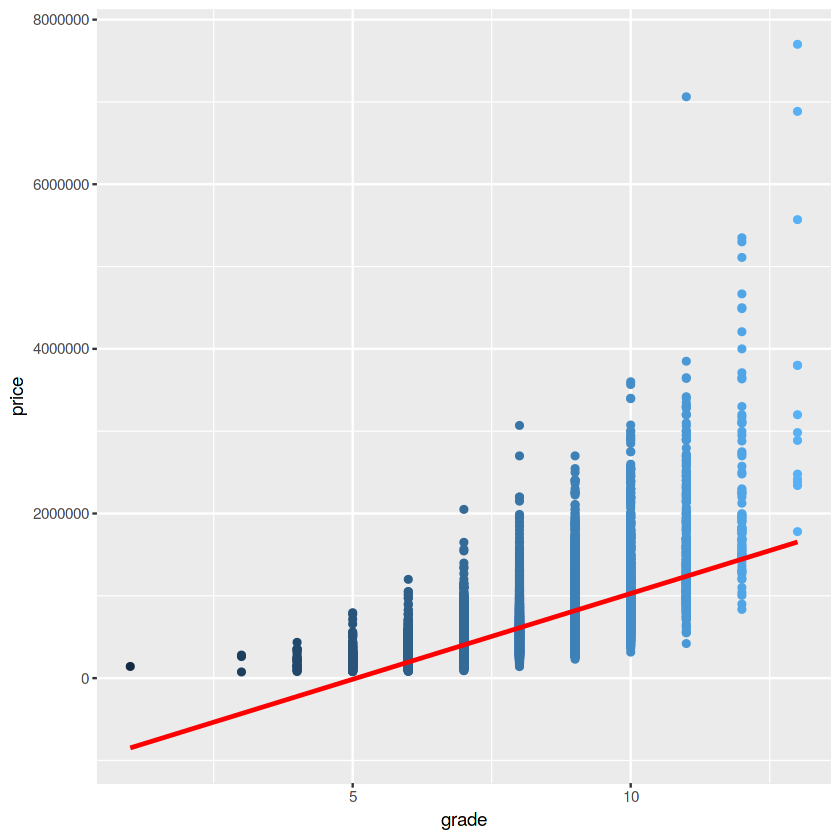
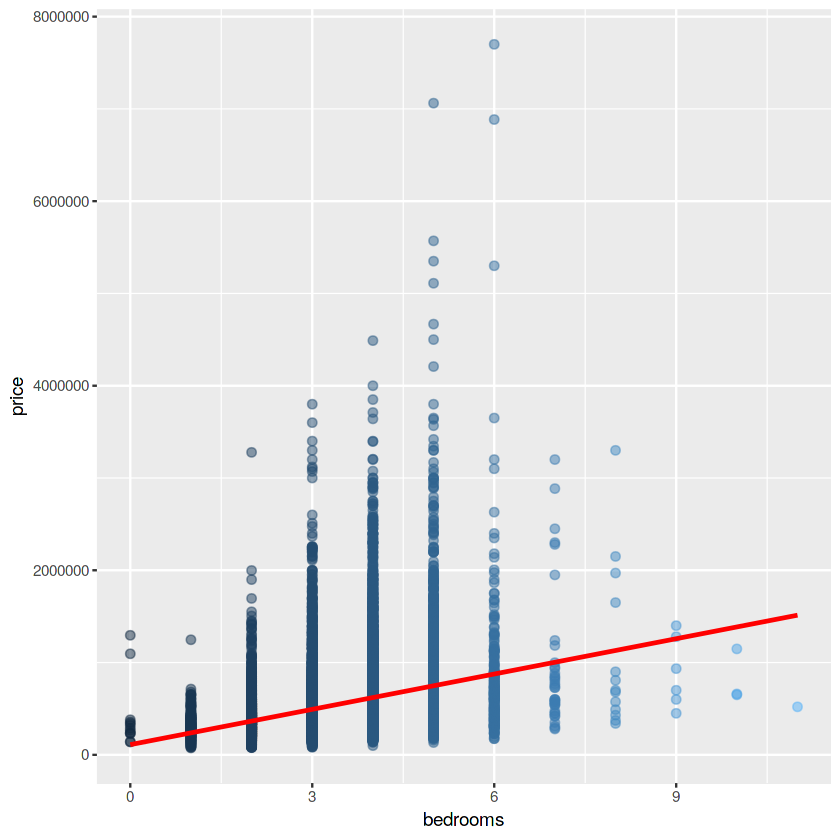
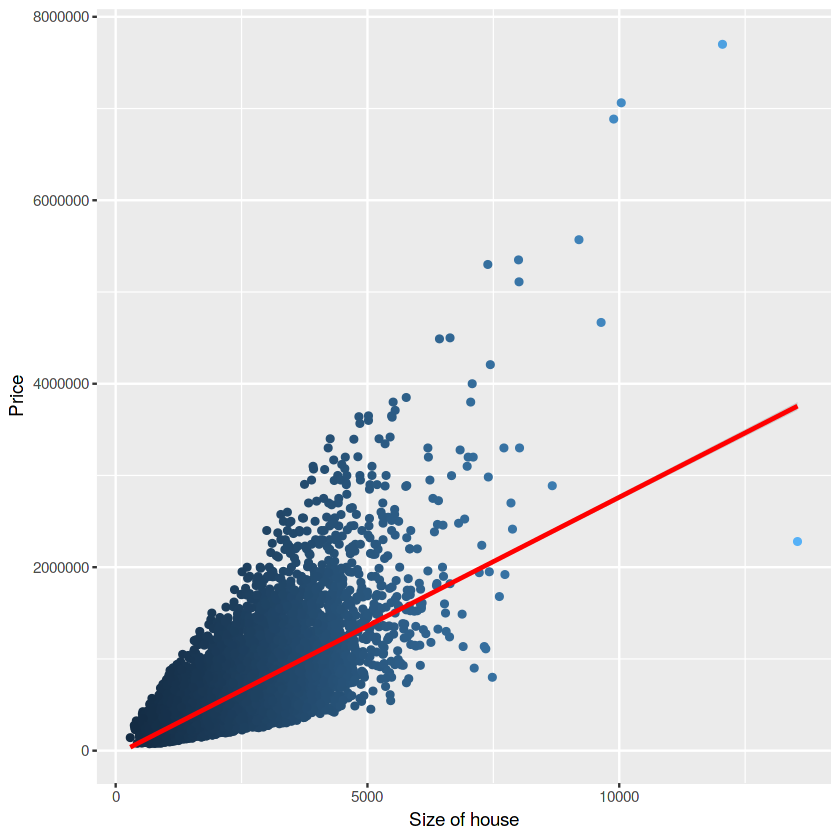
In this part, to get insights from the, box plots, histograms, scatter plots and some graphics helped us a lot. Distributions, outliers, correlations of the features have been examined. To create those plots, “ggplot2” library has frequently used. All the significant points about this dataset reported in this chapter.

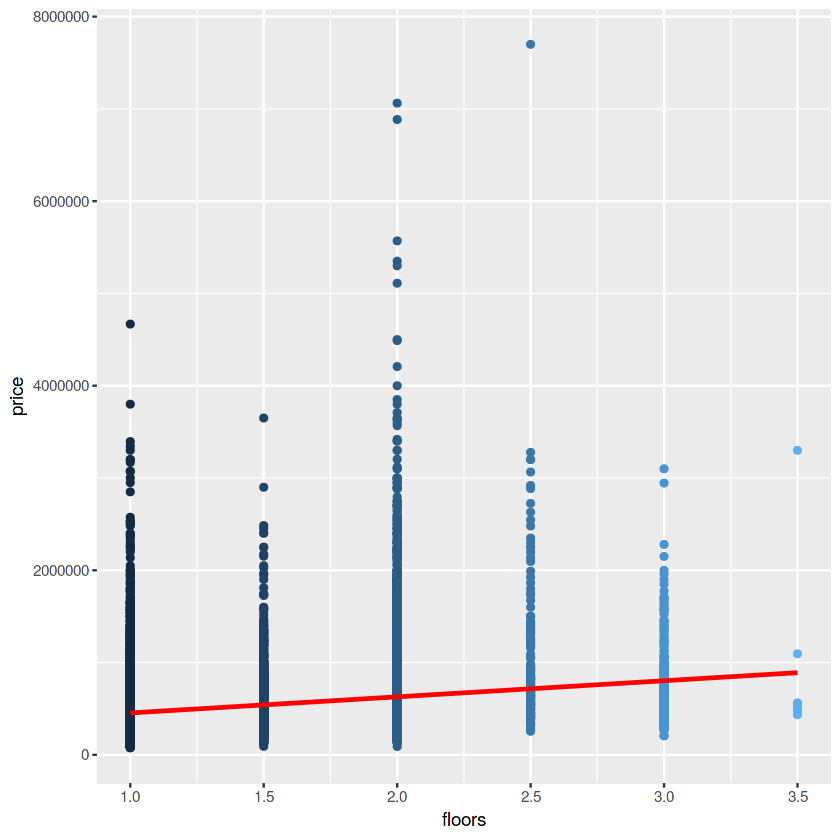
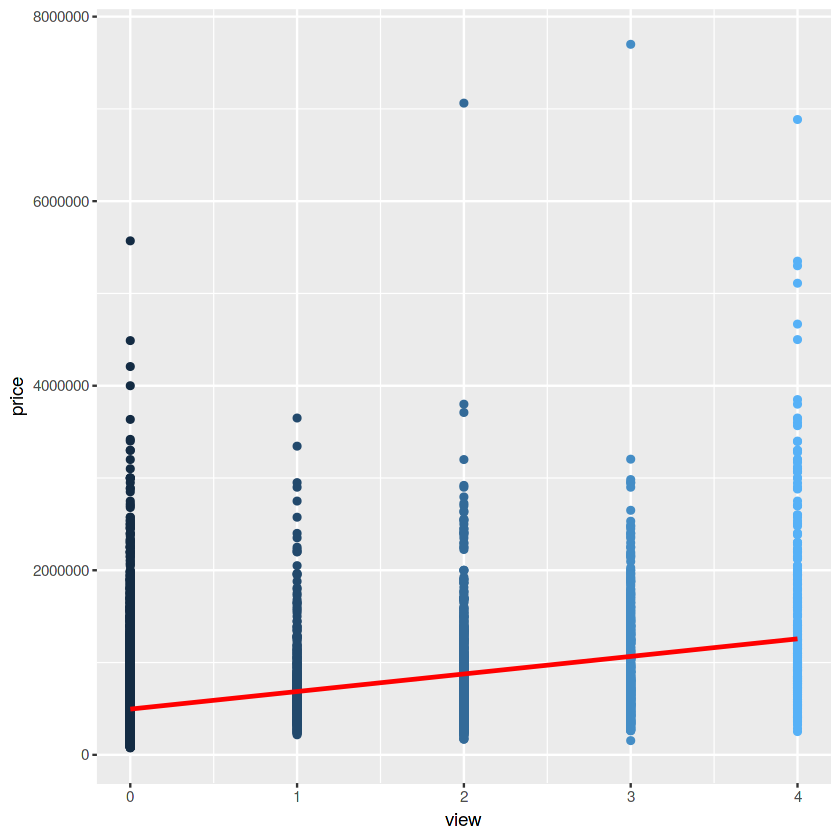
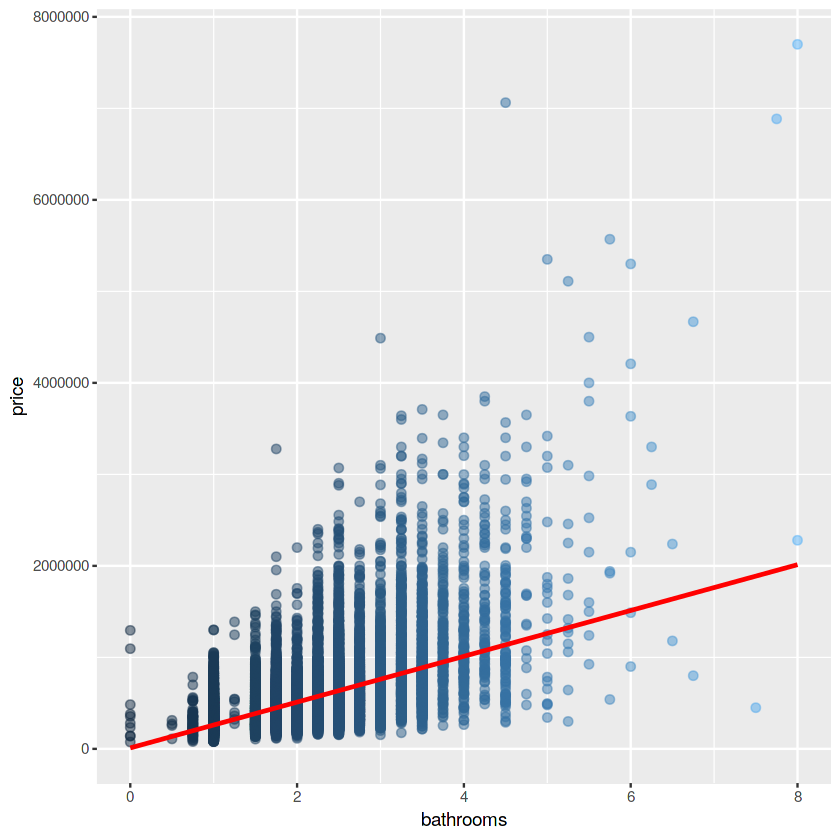
First of all, missing values have been checked and there is not any missing value on any feature. So, this makes easier to move on the data analysis.

Distribution of the target feature is shown below. It is like right-skewed normal distribution. Besides, “sqft\_above”, “sqft\_living “and “sqft\_living15” features are right-skewed normal distribution.

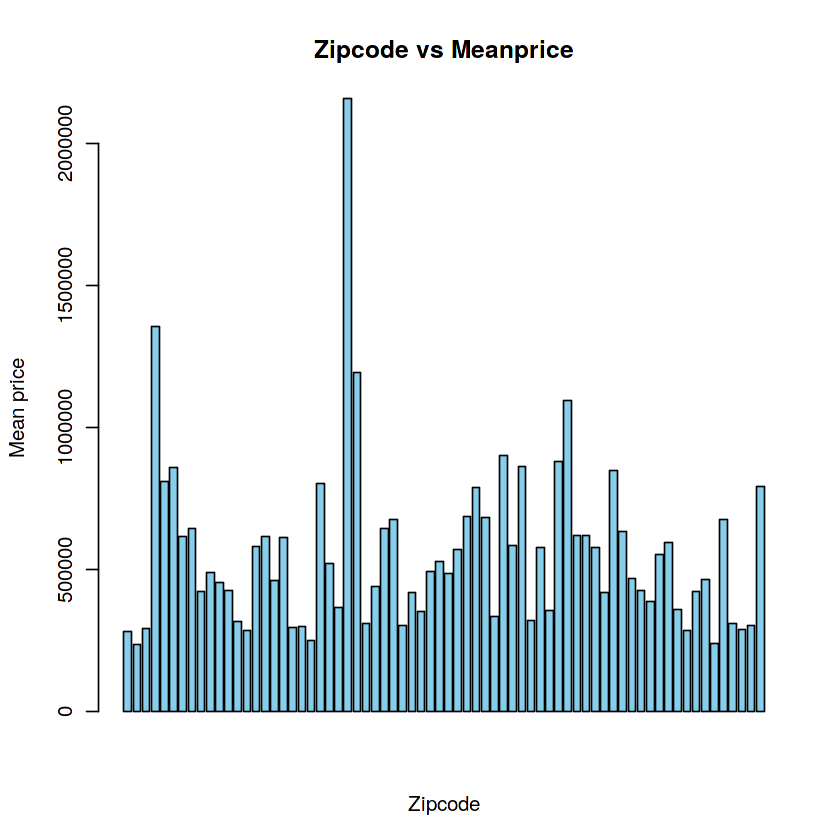
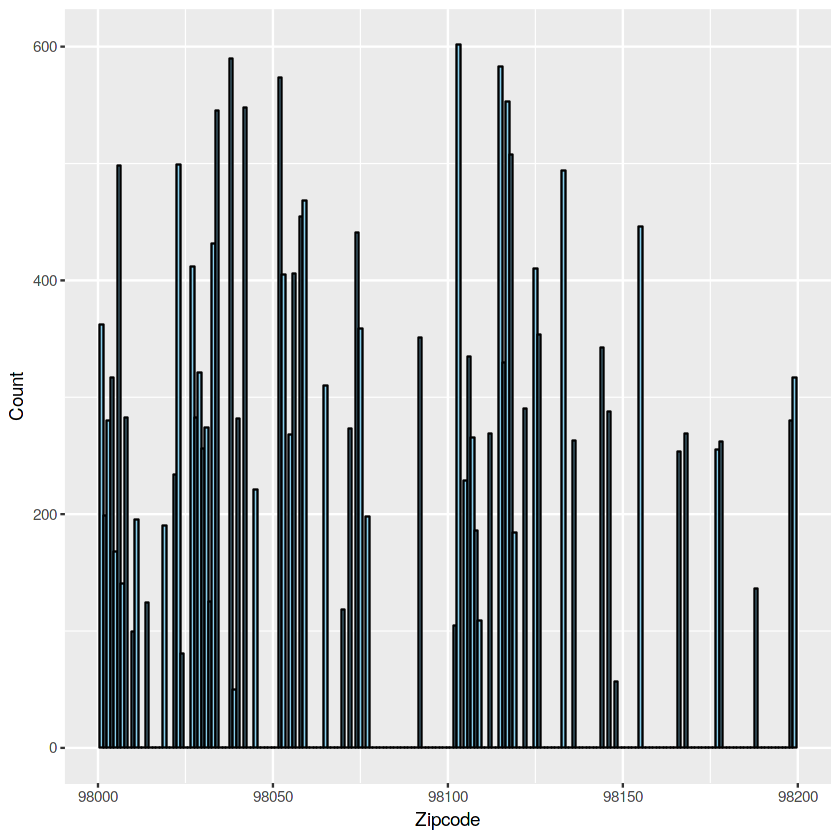


Then, we can observe that the relationship between price and the bathrooms, bedrooms, grade, view, the size of the house. Bigger number of this features means that more price the house would cost.

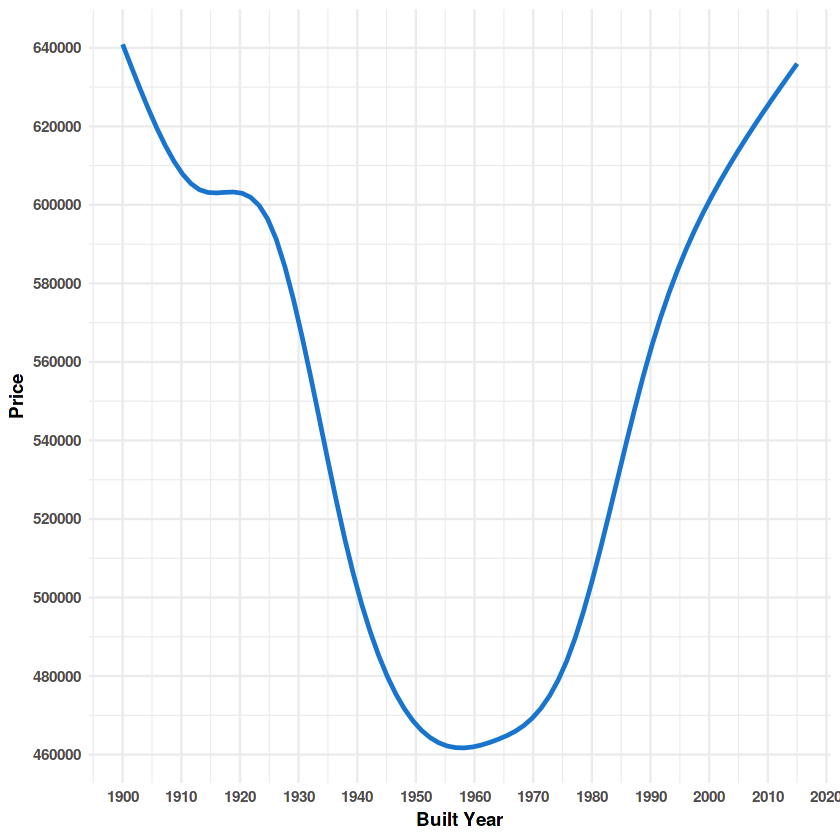
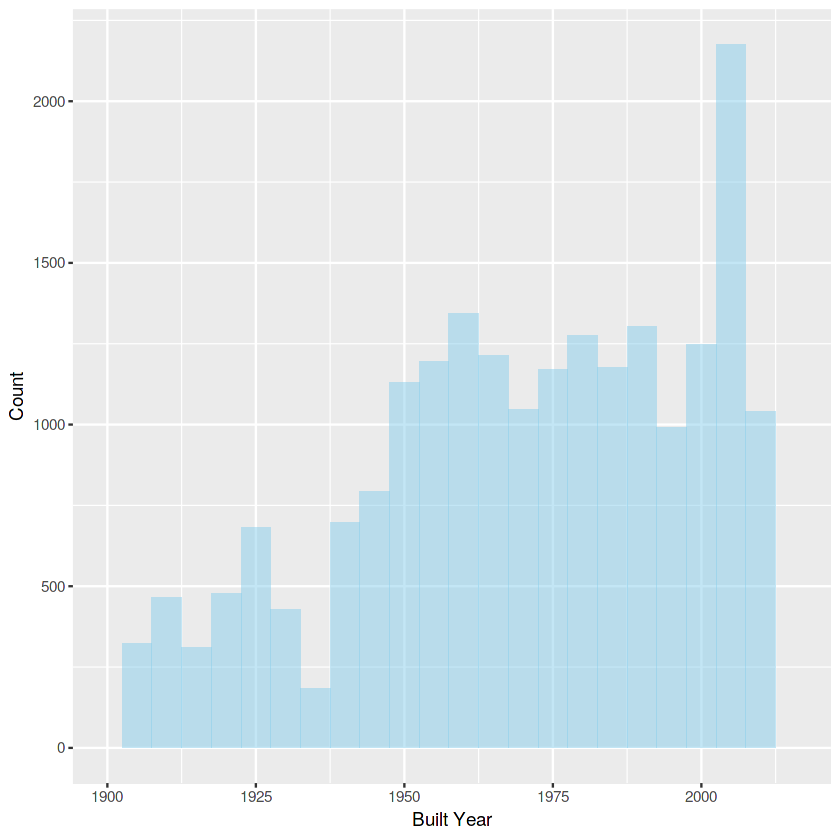




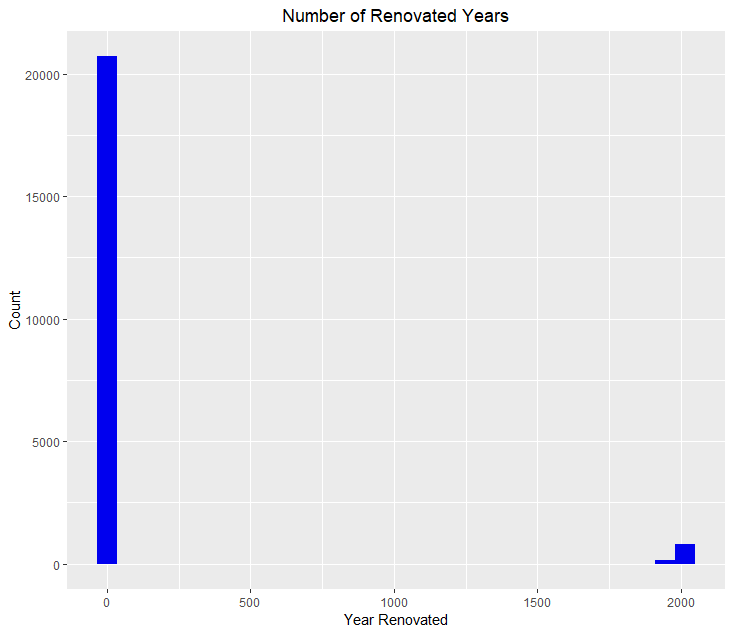
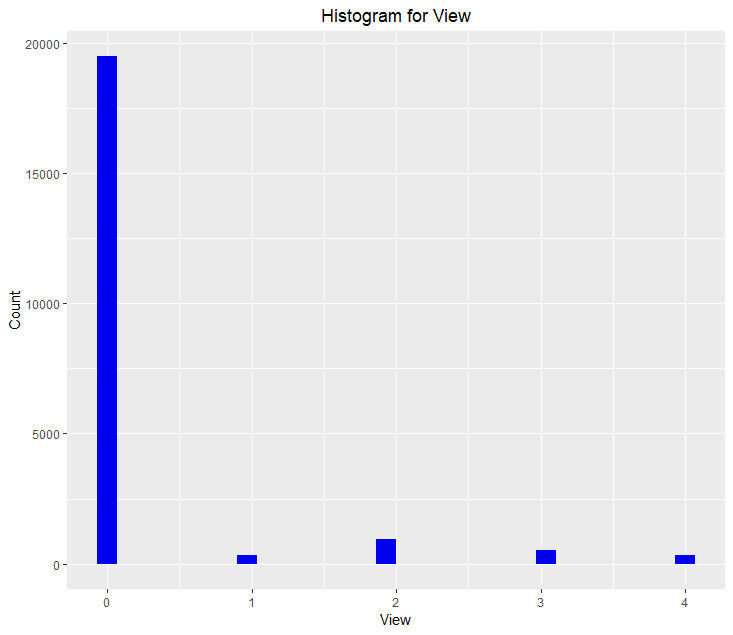
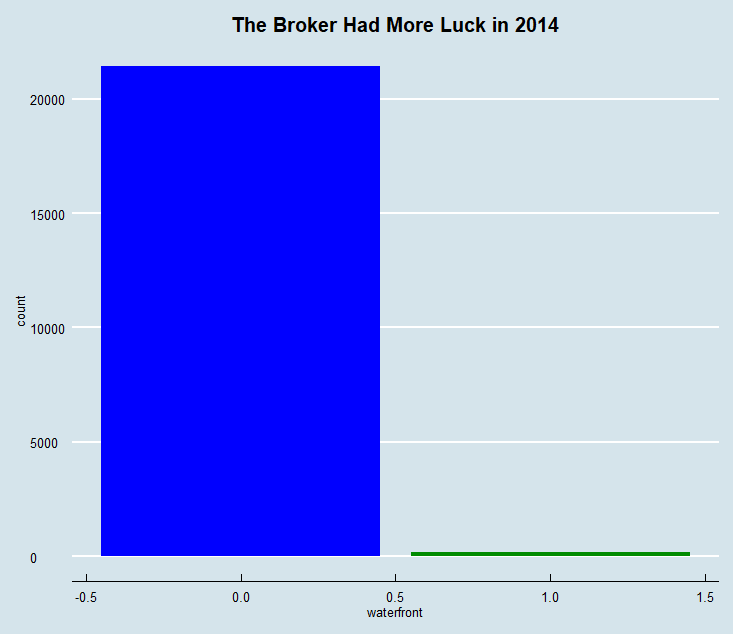
Since “zipcode” attribute would be use as categorical feature, count of each value of zipcode and average prices of each value needs to be observed. There are 70 different categories for “zipcode”.

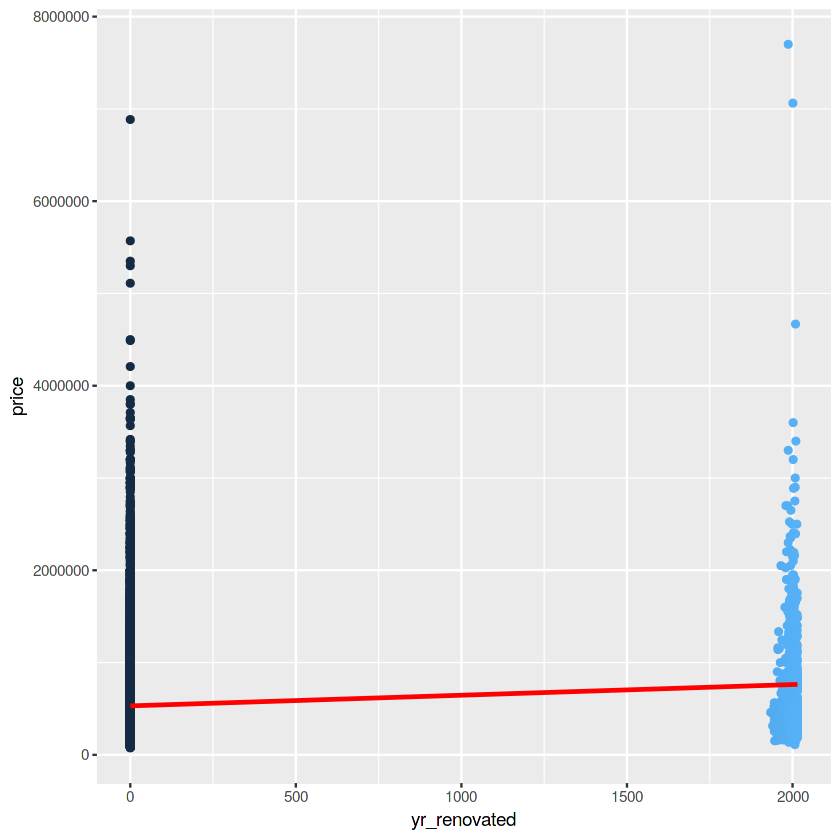
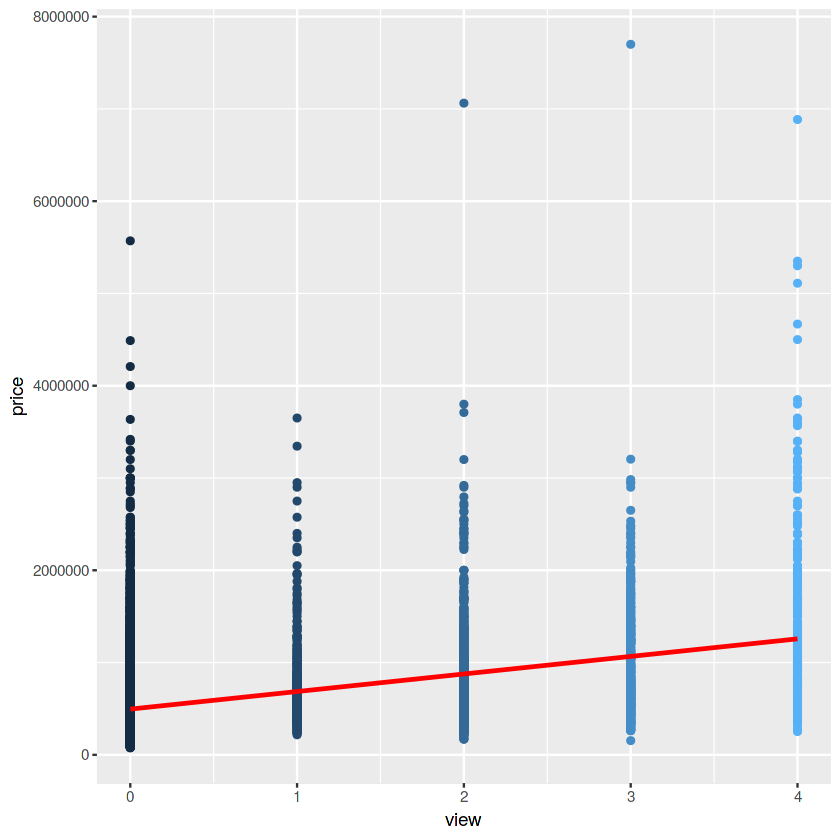
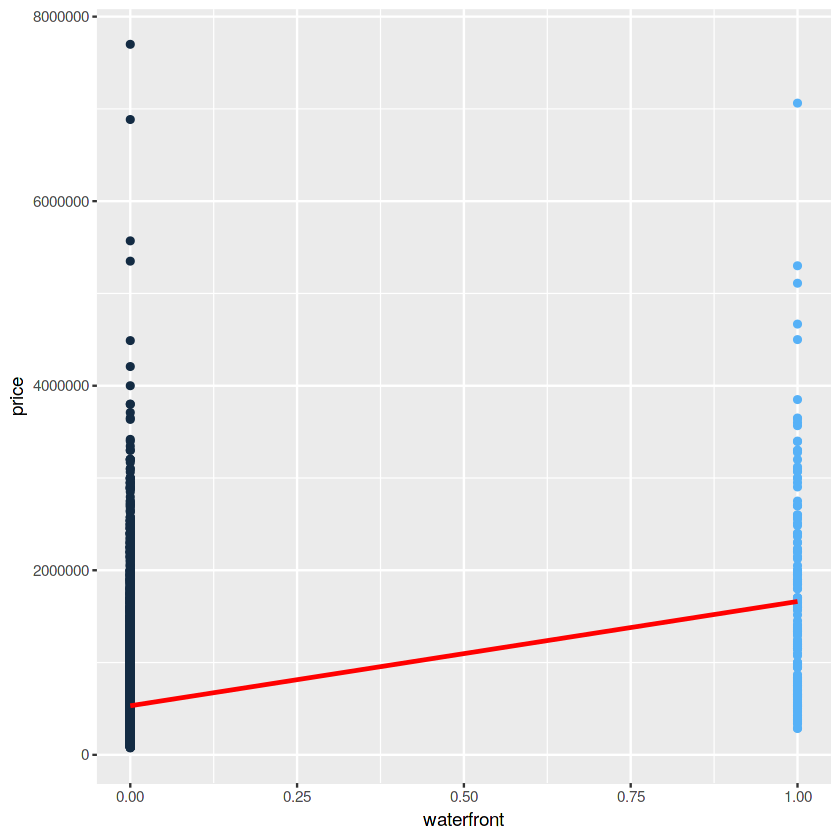


Distribution of the years that the houses were built has been analyzed So, the changes in the price according to built year of the houses is like “U” shape in the plot. Oldest and newest houses have most expensive houses.

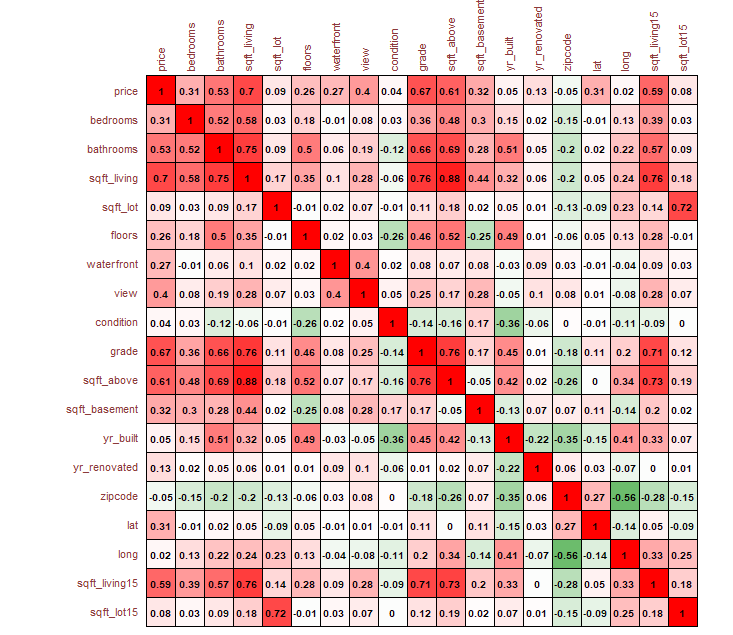


Any missing value has not found at the beginning, but some of the features don’t have discrimination power so much, because of the accumulation on some values at these detected features. %99 of the “waterfront”, %95 of the “yr\_renovated” and %90 of the “view” attribute is 0. However, they are still affecting the price. Thus, these features cannot be omitted during the model creation. All other features have been analyzed also with plotting histograms, but any significant point could be found.

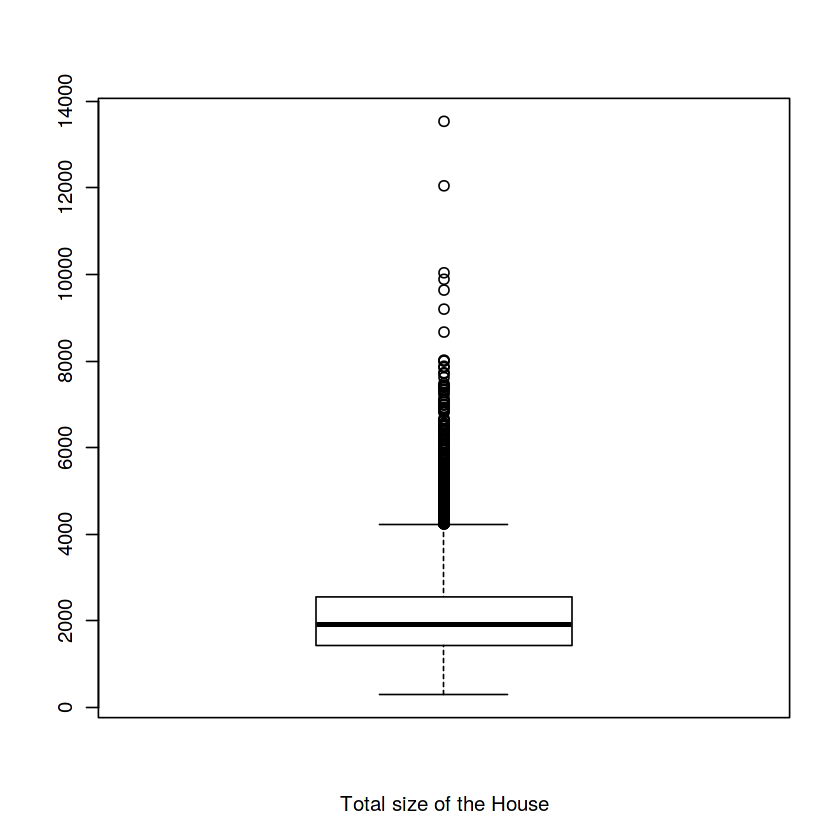
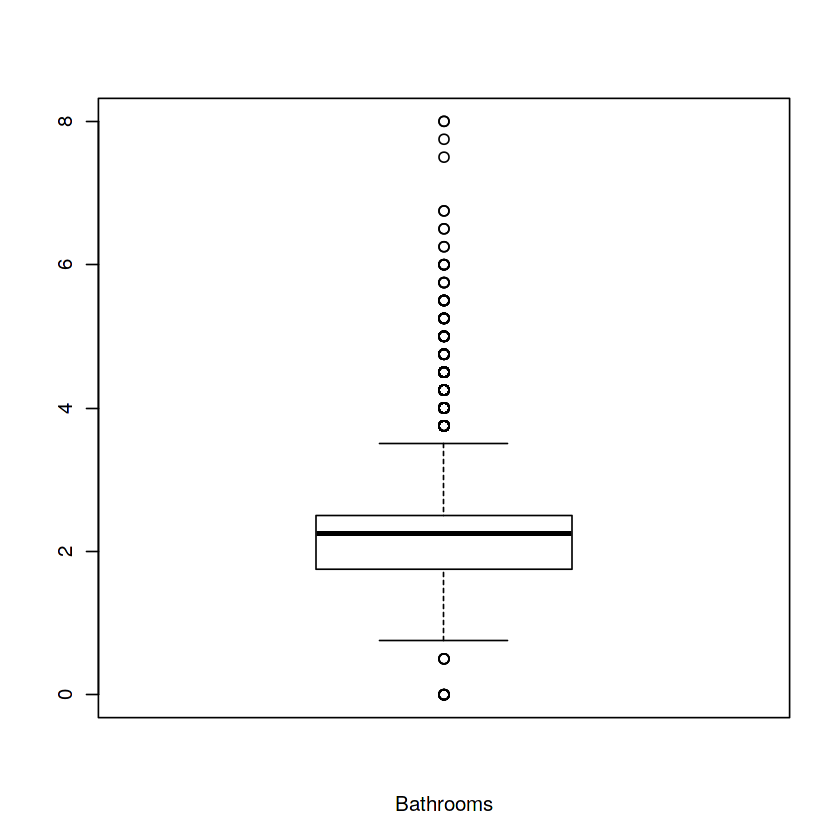
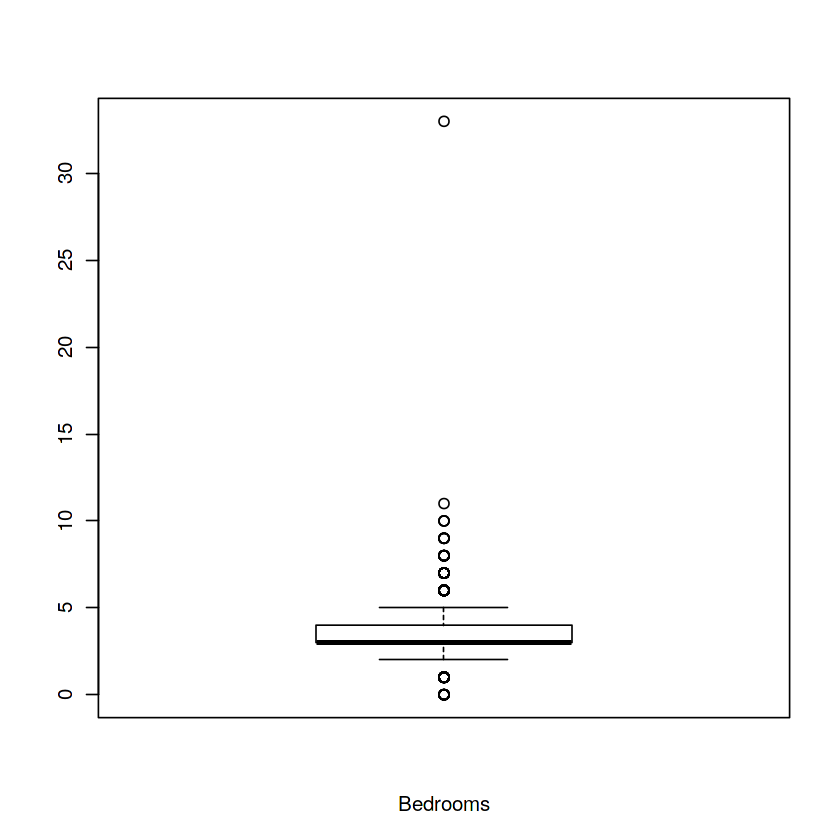


****

Another important thing in this part is calculating Pearson correlation among the features. Pearson’s correlation coefficient is the test statistics that measures the statistical relationship, or association, between two continuous variables. It is known as the best method of measuring the association between variables. So, the correlation among all of the features could be seen the table below. Our target feature is correlated with “sqft\_living” (0.7), “grade” (0.67) “sqft\_above” (0.61) “sqft\_living15” (0.59) and “bathrooms” (0.53). Furthermore, “sqft\_living” is correlated with “bedrooms” (0.58), “bathrooms” (0.75), “grade” (0.76), “sqft\_above” (0.88) and “sqft\_living15” (0.76). “sqft\_lot” and “sqft\_lot15” (0.72) has also correlation. “zipcode” and “lot” is negatively correlated (-0.56). In data preparation chapter we will handle with these correlated attributes. Some of them will be omitted with reasonable reasons.



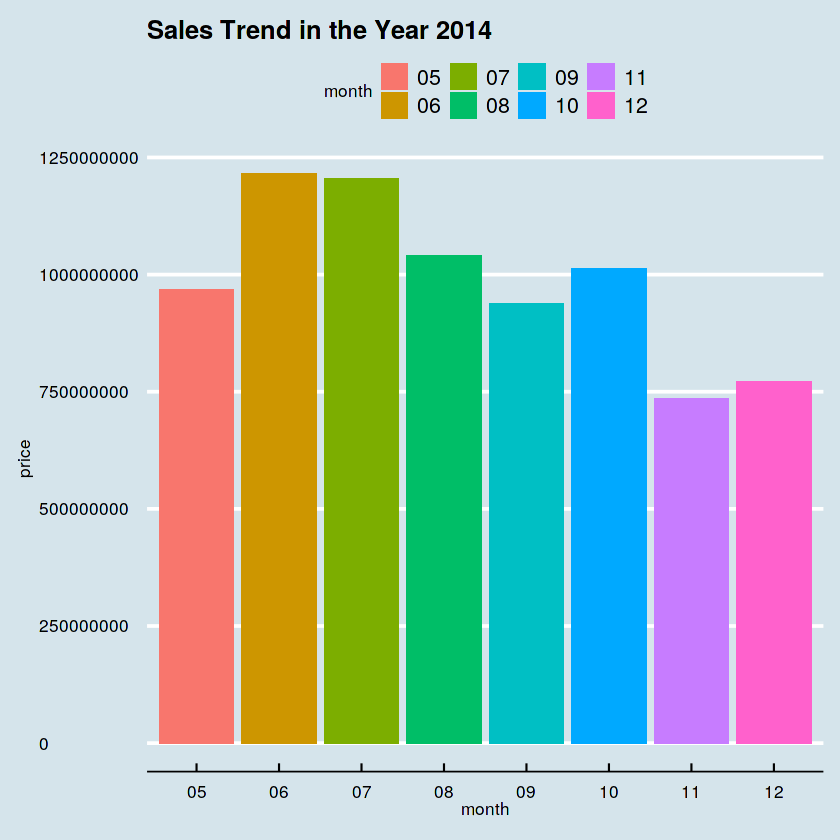
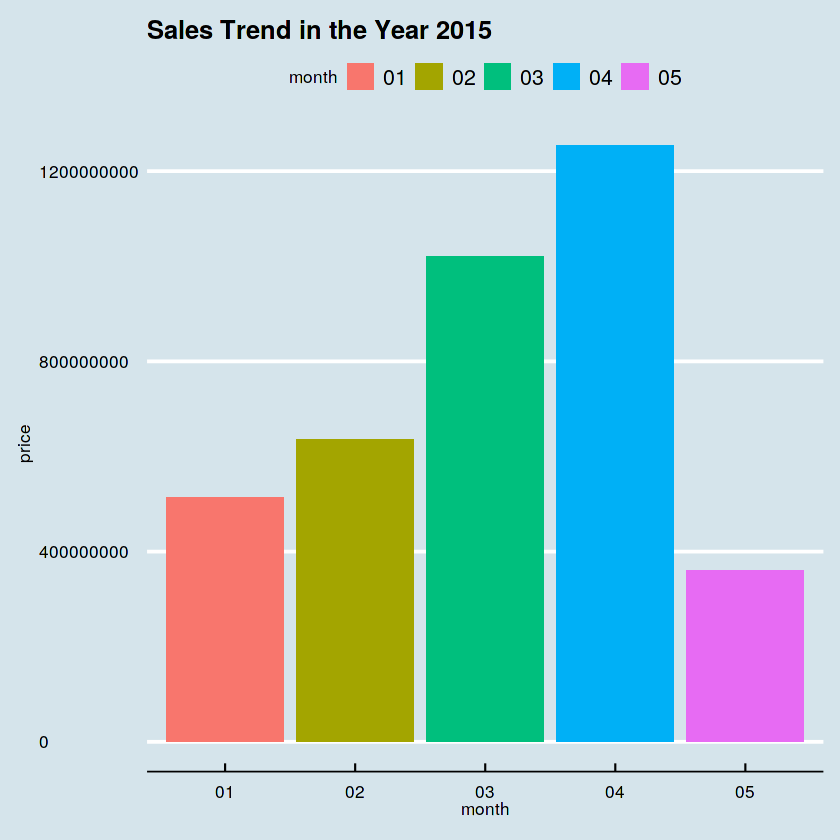
Lastly, outliers have been observed for each features. As a result, for bedrooms and bathrooms, outliers needs to be handled. There are lots of outliers for “sqft\_living” feature, but “bathrooms” and “sqft\_living” is highly affecting the “price”. Therefore, just the rows that has non-sense data would be searched.



**DATA PREPARATION**

Our dataset needed to be audited for anomalies to be corrected and cleaned the data. Because, some some attributes couldn’t be used as it is. Therefore, some features have to be rearranged or parsed to multiple features or never used for the predictions. These operations would make our prediction more accurately.

Beginning of the project, we had already realized that the “ID” column is unnecessary for our model that will be implemented. Firstly, that attribute was removed from the dataset. After that, “date” feature has noticed that this column has to be gone through a parsing process, since the data was meaningless date format like “20141013T000000, 20141209T000000, 20150225T000000, …”. All column has been parsed to year, month and day. So, we 3 more feature extracted from this day format. Then, all of 3 features converted to categorical feature which is much more meaningful for the models. In this way, prices change through each month or through the 2014 to 2015 could be observed easily.

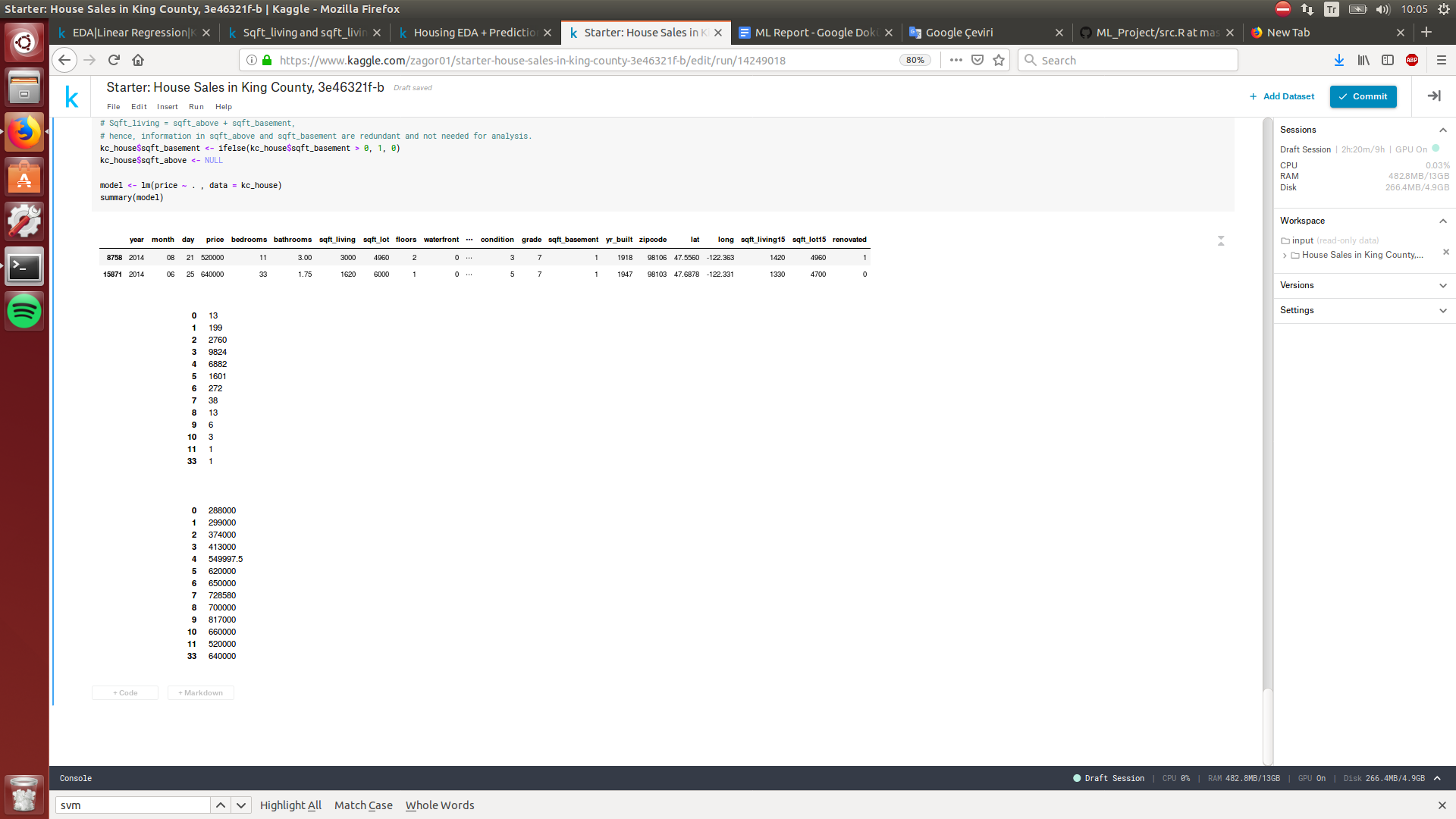
 

Then, as mentioned in data analysis chapter, there are highly correlated relationship between “sqft\_living” and “sqft\_above”. “sqft\_living” is the total size of the house and “sqft\_above” is the size of the house that above the underground or total size of the house except basement size. Thus, this equation came in front of us: “sqft\_living” = “sqft\_above” + “sqft\_basement”. %70 of the houses have no basement and their “sqft\_basement” values are 0. If a house has not a basement, its “sqft\_living” and “sqft\_above” values are going to be equal. This means that, the “sqft\_above” feature has mostly redundant values. Because of that, that column was dropped from the dataset. Afterward, “sqft\_basement” feature convert to 0 or 1. If the house has basement, it would be 1, otherwise it would be 0. In that way, we got rid of the redundant features without any information loss.

“yr\_renovated” column also has consisted of mostly 0’s (%90). If value for this feature is 0, it means that the house has never renovated before. Since this feature was generally consist of 0’s, we applied same thing as we did to “sqft\_basement” feature. If the house renovated before, it would be 1, otherwise it remains 0.

Another correlated feature with “sqft\_living” and that decided to be removed is “sqft\_living15”. Reason of the removal of this column is inconsistent values of that and detection a problem about that. In the description of the dataset, “sqft\_living15” is explained that is living room area in 2015. Even though the lots of rows didn’t renovate before, their “sqft\_living15” values dramatically change from the “sqft\_living”. Same problem exists for the relationship between “sqft\_lot” and “sqft\_lot15”. These are too unstable values and couldn’t interpretable. Therefore, these two columns “sqft\_lot15” and “sqft\_living15” were dropped from the dataset.

Lastly, as already mentioned, there some outlier on “bedrooms” and “bathrooms” columns. We thought that apply clamp transformation for outliers, then our decision changed and we decided to just search for non-sense values. Two rows that has really don’t have any sense was found on the “bedrooms” features. These houses have respectively 33 and 11 bedrooms, but their prices really really low compared to other houses. Their prices is close to mean house prices that has 4-5 bedrooms and their conditions and grades are not so bad. Mean price for houses that has more than 6 bedrooms is really high. They may lead to result high errors. These rows could be wrongly recorded, so we just eliminated them from our dataset. Except of these two rows, any abnormal record has found.



After, the “zipcode” feature converted to categorical feature from numeric feature, data preparation part was completed and ready to used in the machine learning models. Finally, the new dimensions for the dataset are 21611 rows and 19 columns.

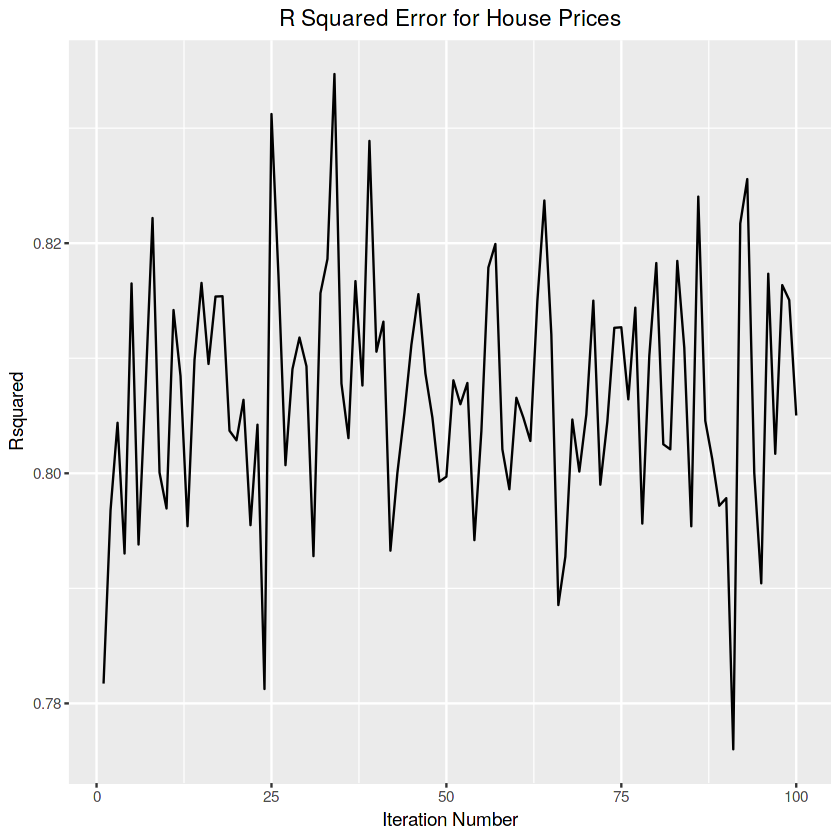
**IMPLEMENTATION OF THE MACHINE LEARNING ALGORITHMS**

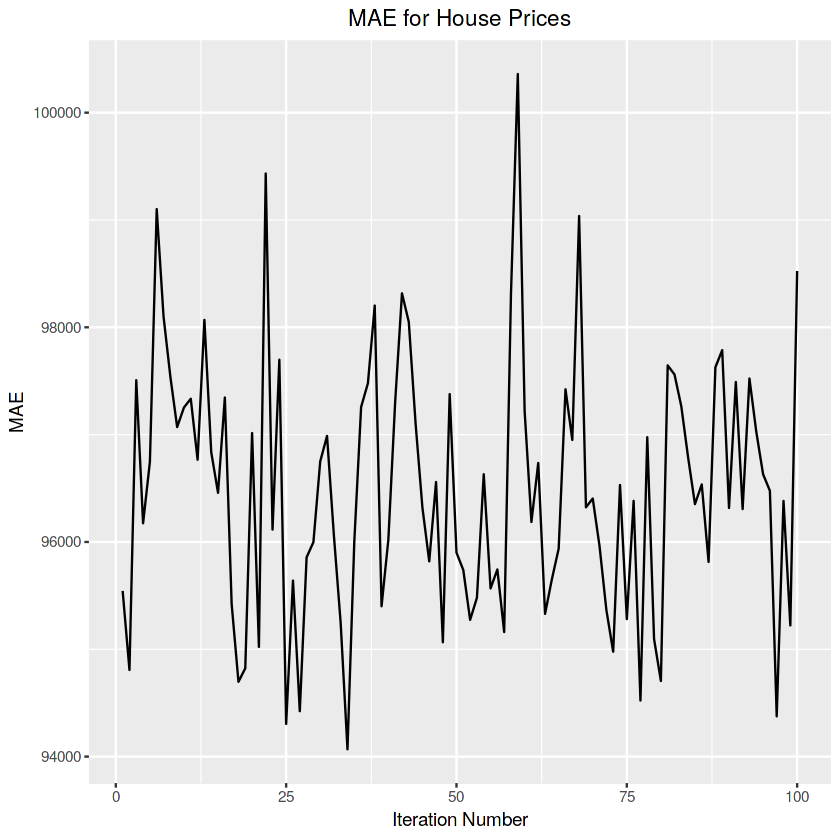
**1- Linear Regression**

When the dataset is ready for the creating model, linear regression algorithm has been implemented at first. All 18 descriptive features have been used. So, 18 weight values and 1 constant intercept value needs to be calculated to obtain the linear equation that makes the significant predictions. Some feature combinations also have been tried, but none of them have not given better errors than the model that have been used all of the descriptive features.



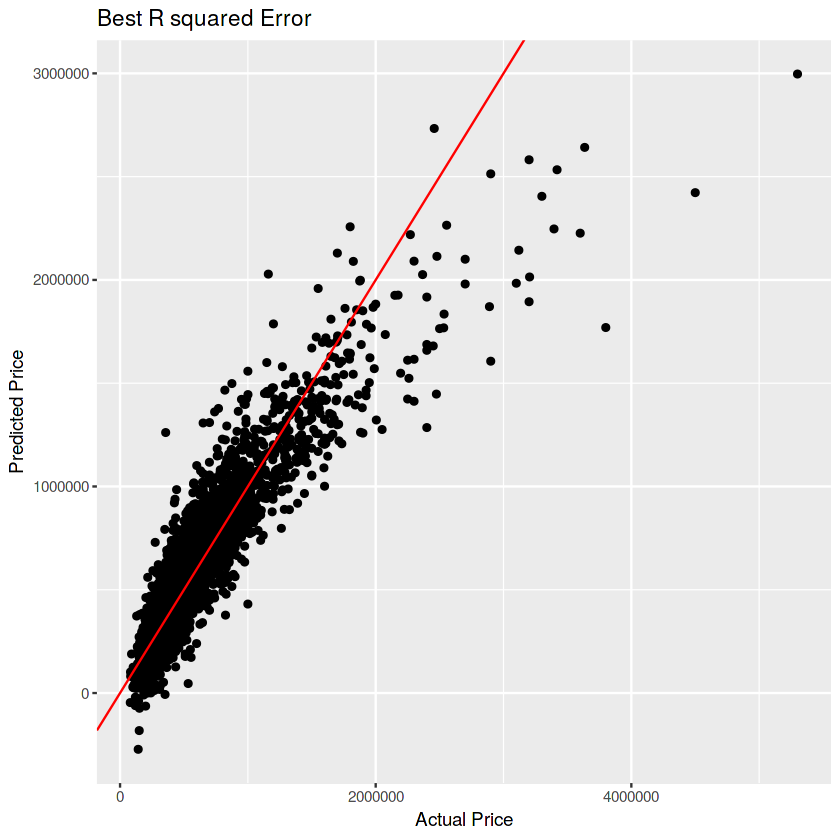
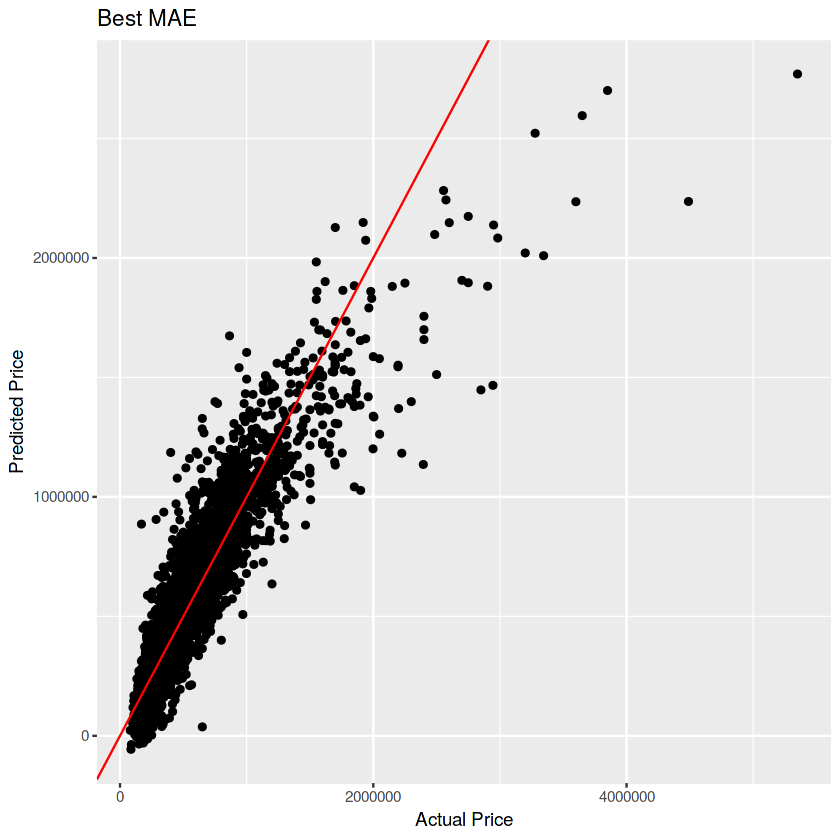
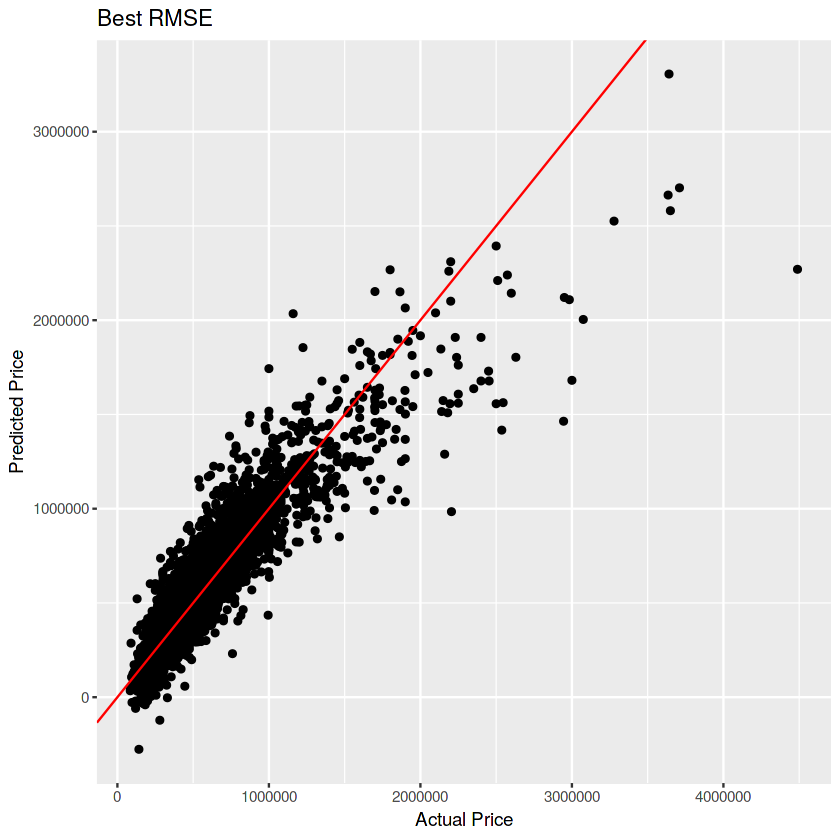
“lm” function has been used provided by R without any extra library. %75 of the whole date has been used for training and remaining part has been used for test and evaluation. To get more reasonable and reliable error values from the model, model trained and 100 times with randomly selected data on each iteration. When the 100 iterations have finished, all required error values have been calculated. Additionally, mean and best error values for each error type have been calculated and visualized. R-squared error changes between 0.83 amd 0.77, RMSE changes between 188.000 and 152.000, MAE changes between 94.000 and 100.000. You can see the plot below, all error values on each iteration may be observed.







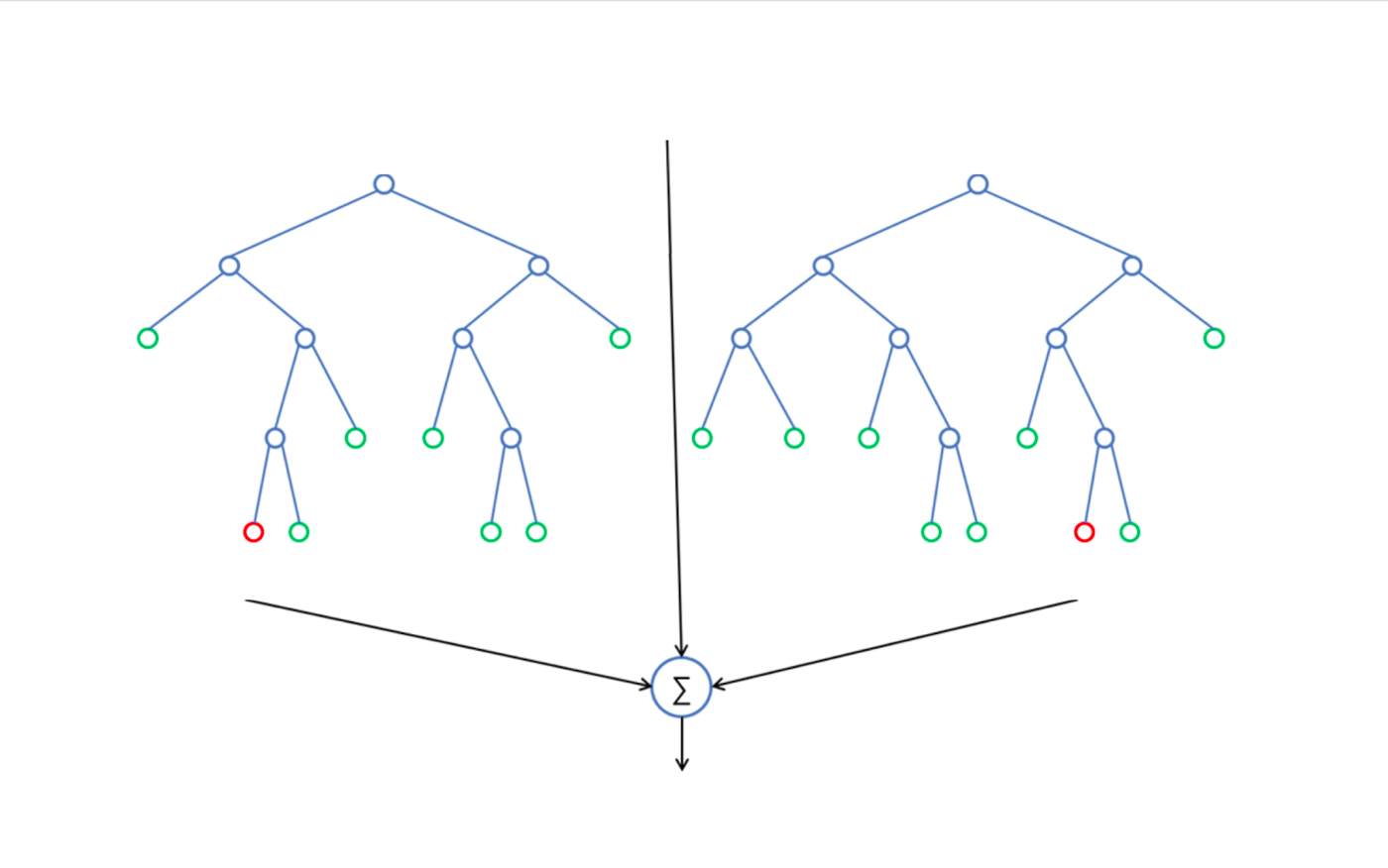
Average errors have been calculated as 0.806 for r-squared, 96493.15 for mean absolute error and 162255.6 for root mean squared error. Best error values have been calculated as 0.834 for R-squared, 94067.82 for MAE and 144999.4 for RMSE, after 100 iterations. For example, predictions of the models that gave the best values for each error type might be seen below.



**2- Random Forest**

Second machine learning model that we have been implemented for our dataset was random forest with bagging. Random forest builds lots of decision trees and merges them together to get a more accurate and stable prediction. When the training set for the current tree is drawn by sampling with replacement, about %33 of the cases are left out of the sample (out-of-bag data). Each tree is constructed using a different bootstrap sample from the original data. Then, out-of-bag data is used for testing on current tree. Error obtained from that test is called out-of-bag error. Each tree is grown as follows:

1. If the number of cases in the training set is N, sample N cases at random with replacement, from the original data. This sample will be the training set for growing the tree.
2. If there are M input variables, a number m<<M is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
3. Each tree is grown to the largest extent possible. There is no pruning.



In this project, “ranger” library has been used to implement random forest models. This library uses the random forest algorithm that created by Leo Breiman. Reason of the using two of “ranger” library is that “ranger” function is at least 6 times faster than “randomForest” function. Too much time would have been needed, if “randomForest” function would have been used to tuning. Although they use same algorithm, “ranger” takes advantage of recursive partitioning and works faster on high dimensional data.

To train best model, tuning needed to be done to select best hyperparameters for our dataset. Hyperparameters are the variables that is set before the learning process begins, by contrast the values of other parameters are derived on training. These hyperparameters may not be determined intuitively. So, grid search technique has been used to find hyperparameters. Grid search builds a model for every combination of hyperparameters that specified before and evaluates each model. There are also couple of search techniques, but we have chosen that one.

There lots of hyperparameters for random forest in our case, but just the most important ones have been tried. Hyperparameters that set-in grid search is:

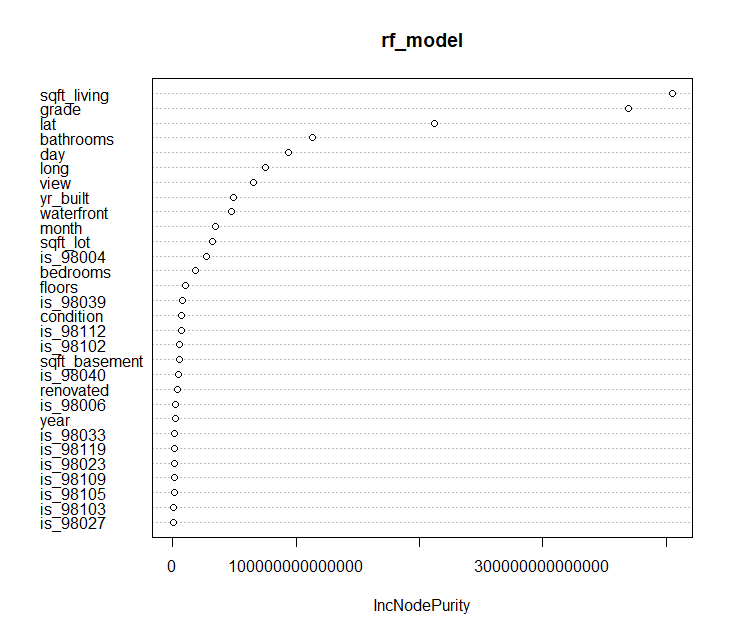
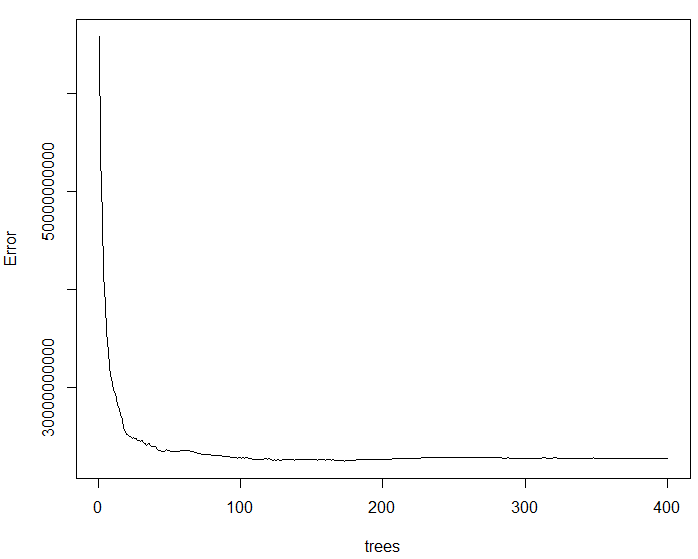
**num.trees:** Number of trees.

**mtry:** Number of variables randomly sampled as candidates at each split.

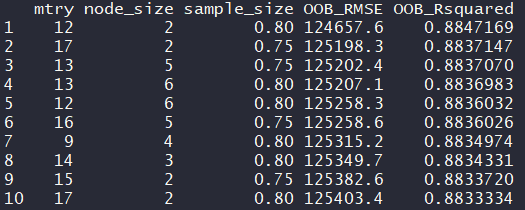
**min.node.size:** Minimum size of terminal nodes. Setting this number larger causes smaller trees to be grown (and thus take less time).

**sample.fraction:** Size fraction of the sample to be drawn.

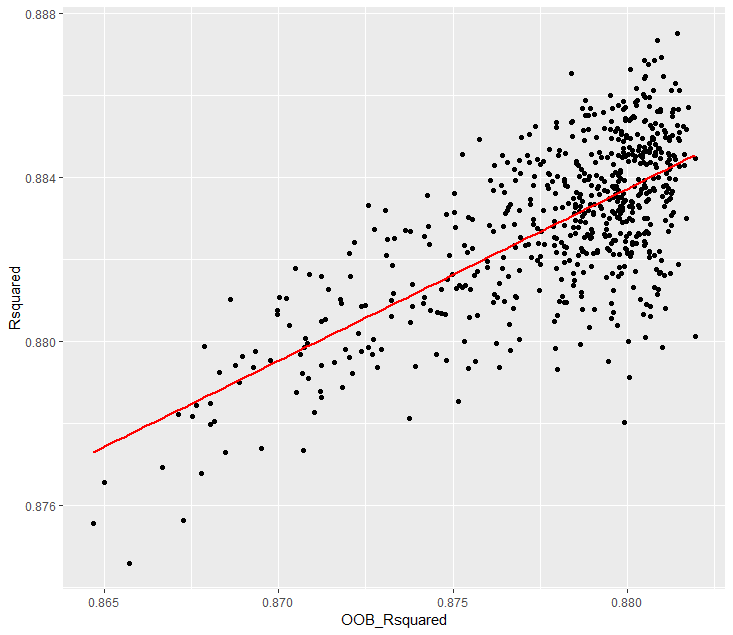
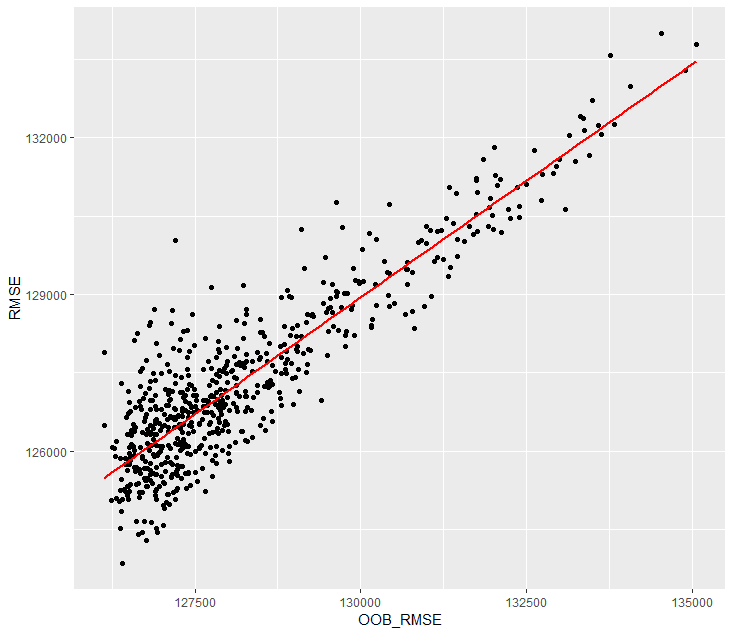
Initially, error changes for number of tree parameter has been observed with default hyperparameter values using “randomForest” function to just get visualization benefits of that. So, model don’t have to be trained about more than 100 trees. After that, importance of the descriptive features found and total size of the found that the most impactful feature on the house price. These initial operations were just to get an intuition before the grid search.



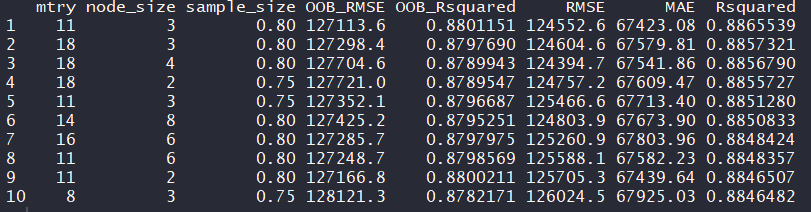
After getting little intuition, we were ready for the grid search. Intervals for each hyperparameter have been assigned as from 1 to 18 for “mtry”, from 2 to 9 for “min.node.size”, (.632, .75, .80) for “sample.fraction”. Then, all combinations have been used in “ranger” function and OOB (out-of-bag) errors have been calculated for each combination. In random forest, there is no need to split data to train and test samples, internally random forest do that while each tree creation as mentioned above. Firstly, grid search applied on whole dataset and the best results is the for OOB RMSE and OOB R-squared. Best hyperparameters are 12 for “mtry”, 2 for “min.node.size” and 0.80 “sample.fraction”. Best 10 results are shown below (Sorted by descending order for R-squared).



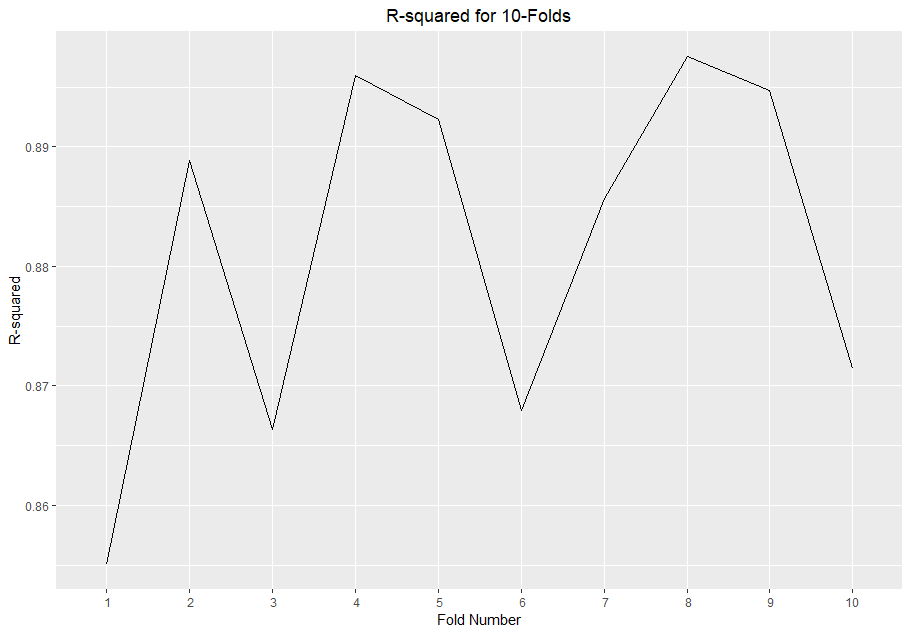
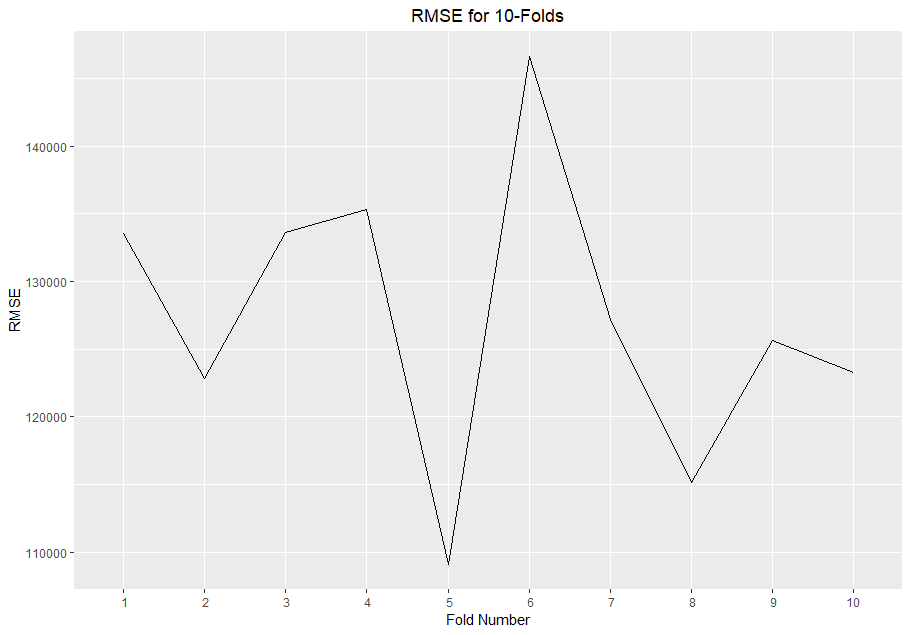
Most of the sources say that “In random forests, there is no need for cross-validation or a separate test set to get an unbiased estimate of the test set error. It is estimated internally with out-of-bag data.”. However, we have applied 10-fold cross validation to same grid search values together with OOB Error to see that sentiment is true or false. Averages of the OOB errors have been compared with average errors come from values of 10-fold cross validation. If RMSE values (average result of 10-fold cross validation and RMSE from OOB) are compared, we can say that the sentiment is mostly correct. But, if R-squared values (average result of 10-fold cross validation and R-squared from OOB) are compared, small differences might be seen.

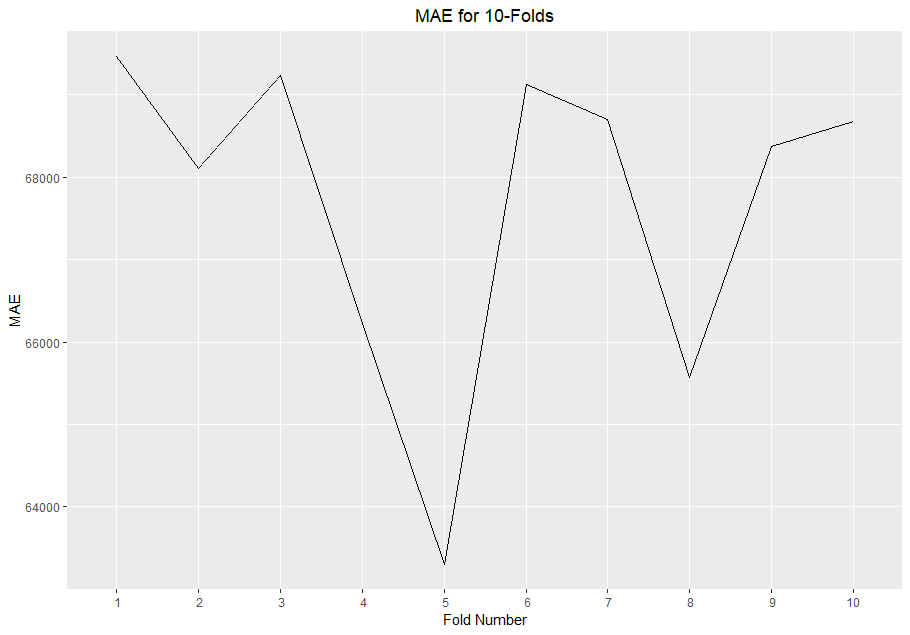


The best 10 R-squared values that resulted from 10-fold cross validated grid search shown below. Hyperparameters of the best result would be used for evaluation.



11 for “mtry”, 3 for “min.node.size” and the 0.8 for “sample.fraction” have been chosen as best hyperparameter combination for evaluation and the average errors after 10-fold cross validation were 127224.7 for RMSE, 0.881 for R-squared and 67677.97 for MAE. These values are consistent with results of the tuning operation and may be used as the best model for random forest algorithm. Here are the error changes of the 10-folds.

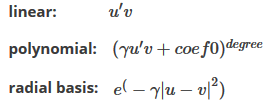




**3- Support Vector Regression**

Third machine learning model that have been implemented for the dataset was Support Vector Regression. The classification algorithm that Support Vector Machine can also be used as regression method that maintaining all the main features that characterize the algorithm (maximum margin). The Support Vector Regression (SVR) uses the same principles as the SVM for classification, with a few minor differences. Firstly, because output is a real number it becomes very difficult to predict the target feature that has infinite possibilities. In the case of regression, a margin of tolerance (epsilon) is set in approximation to the SVM which would have already requested from the problem. The main idea is the same with other techniques that to minimize error, individualizing the hyperplane which maximizes the margin, keeping in mind that part of the error is tolerated. The kernel functions transform the data into a higher dimensional feature space to make it possible to perform the linear separation.

The kernels used in training and predicting. The changing of the error rates with some of the following parameters that depending on the kernel types might be considered.



And some of the parameters that used for these kernel types to optimize the model in this project are:

**degree**: Degree is a parameter used when kernel is set to 'poly'. It's basically the degree of the polynomial used to find the hyperplane to split the data.

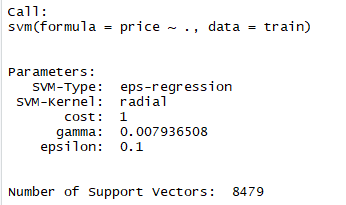
**gamma**: Parameter needed for all kernels except linear. A small gamma gives you a pointed bump in the higher dimensions, a large gamma gives you a softer, broader bump.

**cost**: It is the ‘C’-constant of the regularization term in the Lagrange formulation. Small C makes the cost of misclassification low ("soft margin"). Large C makes the cost of misclassification high ('hard margin"), thus forcing the algorithm to explain the input data stricter and potentially overfit.

**epsilon**: Epsilon in the insensitive-loss function. The value of ϵ defines a margin of tolerance where no penalty is given to errors.

The main idea of this part of the project is that split the data into two parts that train and test sets, create an svm model (that might be used for the regression models in R) for data train set, predict the results of data test set and then, improve the model with tuning which used to select the best parameters for the model. The tuning operation covers k-fold cross validation which means we look for 5 different models on every iteration. The process of these choosing the best parameters also called “hyperparameter optimization” or “model selection” and the standard way of doing this operation is “grid search” which used in this project. The selection of cost parameter and 5-fold cross validation also give an advantage to avoid overfitting. The grid search technique used for polynomial, radial and linear kernels. The 5-fold cross validation also applied in each combination (hyperparameter combinations).

In first; when SVM model created with default parameters and kernel after data splitting into test and train set, summary of this model and RMSE of the prediction seen below.



The RMSE value of this prediction of the model is: **149114.9**

To improve this model, tuning process would be used on every kernel type to select best hyperparameters to create the best model.

**a) Polynomial Kernel**

To apply tuning for training a lot of polynomial kernel SVR models for different parameter

couples of ϵ, cost, gamma and degree ranges, determined as follows:

**epsilon** = seq(0, 1, by = 0.1)

**cost** = 2^(2:9)

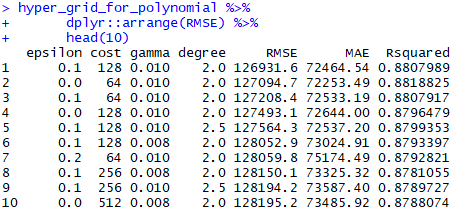
**gamma** = c(0.008,0.001 ,0.01)

**degree** = c(2,2.5)

It means 11x8x3x2 = 528 different models will be scanned with 5-fold cross validation. After scanned and calculated MAE, RMSE and R-Square Error values of every created model, the grid object that we can created firstly, become the following. Totally, grid has 528 different rows.

When arrange the grid according to different error rates, we can see the best error values for each type and can see which parameters are created this error rates. As seen in below, the model that has best RMSE rate, has epsilon=0.1, cost=128, gamma=0.010, degree=2 values.

**Note**: These error rates are the average of error rates from each values of the 5-fold cross validation.

****

As seen, best RMSE value is **126931.6**. Using same way, best MAE and R-squared values and their parameter values might be seen.

The best MAE value is **72253.49** with epsilon=0.0, cost=64, gamma=0.010, degree=2 parameters and the best R-squared value is **0.8818825** with epsilon 0.0, cost=64, gamma=0.010, degree=2 parameters.

**b) Radial Kernel**

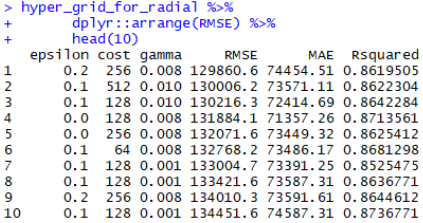
To apply tuning to the radial kernel SVR models for different parameter ranges of ϵ, cost and gamma, initially determined in grid as follows.

**epsilon** = seq(0, 1, by = 0.1)

**cost** = 2^(2:9)

**gamma** = c(0.008,0.001 ,0.01)

Totally, grid has created with 264 rows (11x8x3=264). When same process with polynomial kernel tuning is applied, best RMSE, MAE and R-Squared error rates which mean of 5-fold cross validation for each model has been calculated after error rates calculated for each 264 combinations.



As seen, best RMSE value for radial kernel after grid search process and cross validation (tuning process) **129860.6** and with same way; the best MAE value is **73110.21** with epsilon=0.1, cost=128, gamma=0.010 parameters and the best R-squared value is **0.8762618** with epsilon 0.1, cost=128, gamma=0.010 parameters. At this point, we can say that polynomial kernel seems better than radial kernel for our dataset.

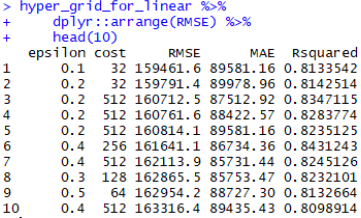
**c) Linear Kernel**

Epsilon (ϵ) and cost parameter ranges initially determined on grid for the linear kernel as well as for the radial and polynomial kernels. The difference is there is no gamma parameter in linear kernel as described before.

**epsilon** = seq(0, 1, by = 0.1)

**cost** = 2^(2:9)

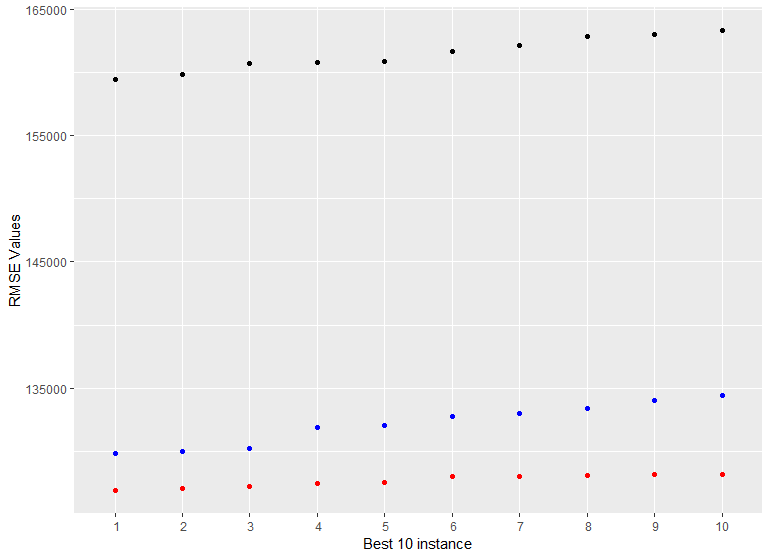
Totally, grid has created with 88 rows (11x8=88). It includes all the combinations of parameters. When same process with radial and polynomial kernel tuning is applied, best RMSE, MAE and R-Squared error values are obtained. Best 10 RMSE value can be seen below.



Best RMSE value in linear kernel is **159461.6** with epsilon=0.1 and cost=32 , with same way; best MAE value is **85562.24** with epsilon = 0.4 and cost = 512, best R-Squared value is **0.8431243** with epsilon=0.4 and cost=256.

As a result of this part, after the tuning process with determined parameters ranges and 5-fold cross validation, the linear kernel gives worst error results. On the other hand, polynomial and radial kernels give better error values. According the first findings, polynomial kernel seems to best kernel of these three kernel types. Better error results can be obtained by performing one more tuning operation with lower intervals according to parameter ranges that gives good results. But the first findings were interpreted here. The tuning process of these 3 kernel types takes 240 hours. Therefore, another parameter selection will not be done again. The reason for using 5-fold cross validation instead of 10-fold is the same.

After all the processes that grid search hyperparameter selection and 5-fold cross validation, the comparison of the RMSE plot according to all kernel’s RMSE values is seen in the following figure.



The MAE and R-squared graphs are looks same.

**CONCLUSION**

As a conclusion, the best model for our dataset is SVR using polynomial kernel and the worst model is the Linear Regression according to the error values as shown before.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **RMSE** | **R-Squared** | **MAE** |
| **Linear Regression** | 162255.6 | 0.8062512 | 94493.15 |
| **Random Forest** | 127224.7 | 0.8818544 | 67677.97 |
| **SVR using linear kernel** | 159461.6 | 0.8431243 | 85562.24 |
| **SVR using polynomial kernel** | 126931.6 | 0.8818825 | 72253.49 |
| **SVR using radial kernel** | 129860.6 | 0.8762618 | 73110.21 |