House Prices

Gabriel Lapointe

September 18, 2016

Contents

Data Acquisition

In this section, we specify the business problem to solve for this project. From the data source, we will ask questions on the dataset and establish a methodology to solve the problem.

Objective

With 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, we have to predict the final price of each home.

Data Source

The data is provided by Kaggle and can be found here.

Dataset Questions

Before we start the exploration of the dataset, we need to write a list of questions about this dataset considering the problem we have to solve.

- How big is the dataset?
- Does the dataset contains 'NA' or missing values? Can we replace them by a value? Why?
- Does the data is coherent (date with same format, no out of bound values, no misspelled words, etc.)?
- What does the data look like and what are the relationships between features if they exist?
- What are the measures used?
- Does the dataset contains abnormal data?
- Can we solve the problem with this dataset?

Evaluation Metrics

Submissions are evaluated on Root-Mean-Squared-Error (RMSE) between the logarithm of the predicted value and the logarithm of the observed sales price. (Taking logs means that errors in predicting expensive houses and cheap houses will affect the result equally.)

Methodology

In this document, we start by exploring the dataset and build the data story behind it. This will give us important insights which will answer our questions on this dataset. The next step is to proceed to feature engineering which consists to create, remove or replace features regarding insights we got when exploring the dataset. Then, we will peoceed to a features selection to know which features are strongly correlated to the outcome. We will ensure our new dataset is a valid input for each of our prediction models. We will fine-tune the model's parameters by cross-validating the model with the train set to get the optimal parameters. After applying our model to the test set, we will visualize the predictions calculated and explain the results. Finally, we will give our recommandations to fulfill the objective of this project.

Loading Dataset

We load 'train.csv' and 'test.csv'. Then, we merge them to proceed to the cleaning and exploration of this entire dataset.

```r
library(data.table)      # setDT, set
library(dplyr)           # select, filter, %>%
library(scales)          # Scaling functions used for ggplot
library(gridExtra)       # Grid of ggplot to save space
library(ggplot2)         # ggplot functions for visualization and exploration
library(caret)
library(corrplot)
library(moments)         # For skewness
library(Matrix)
library(mice)            # To replace NA values by a predicted one
library(VIM)
library(randomForest)
library(xgboost)
library(glmnet)
library(microbenchmark)  # benchmarking functions
library(knitr)           # opts_chunk

setwd("/home/gabriel/Documents/Projects/HousePrices")

set.seed(1234)

source("Dataset.R")

## Remove scientific notation (e.g. E-005).
options(scipen = 999)

## Remove hash symbols when printing results and do not show message or warning everywhere in this docu
opts_chunk$set(message = FALSE,
               warning = FALSE,
               comment = NA)

## Read csv files and ensure NA strings are converted to real NA.
na.strings <- c("NA")
train <- fread(input = "train.csv",
               showProgress = FALSE,
               stringsAsFactors = FALSE,
               na.strings = na.strings,
               header = TRUE)

test <- fread(input = "test.csv",
              showProgress = FALSE,
              stringsAsFactors = FALSE,
              na.strings = na.strings,
              header = TRUE)

## Merge the train and test sets.
test$SalePrice <- -1
dataset <- rbind(train, test)
```

Dataset	File Size (Kb)	# Houses	# Features
train.csv	460.7	1460	81
test.csv	451.4	1459	80
Total(dataset)	**912.1**	**2919**	**81**

These datasets are very small. Each observation (row) is a house where we want to predict their sale price in the test set.

Dataset Cleaning

The objective of this section is to detect all inconsistancies in the dataset and try to fix them all to gain as much accuracy as possible. We have to check if the dataset is valid with the possible values given in the code book. Thus, we need to ensure that there are no mispelled words or no values that are not in the code book. Also, all numerical values should be coherent with their description meaning that their bounds have to be logically correct. Regarding the code book, none of the categorical features have over 25 unique values. Then, we will compare the values mentioned in the code book with the values we have in the dataset. Finally, we have to detect anomalies and replace missing values with the most accurate ones.

```
$Id
NULL

$MSSubClass
[1] "20, 30, 40, 45, 50, 60, 70, 75, 80, 85, 90, 120, 150, 160, 180, 190"

$MSZoning
[1] "C (all), FV, RH, RL, RM, NA"

$LotFrontage
NULL

$LotArea
NULL

$Street
[1] "Grvl, Pave"

$Alley
[1] ", Grvl, Pave, NA"

$LotShape
[1] "IR1, IR2, IR3, Reg"

$LandContour
[1] "Bnk, HLS, Low, Lvl"

$Utilities
[1] "AllPub, NoSeWa, NA"

$LotConfig
[1] "Corner, CulDSac, FR2, FR3, Inside"

$LandSlope
[1] "Gtl, Mod, Sev"

$Neighborhood
[1] "Blmngtn, Blueste, BrDale, BrkSide, ClearCr, CollgCr, Crawfor, Edwards, Gilbert, IDOTRR, MeadowV, M

$Condition1
```

```
[1] "Artery, Feedr, Norm, PosA, PosN, RRAe, RRAn, RRNe, RRNn"

$Condition2
[1] "Artery, Feedr, Norm, PosA, PosN, RRAe, RRAn, RRNn"

$BldgType
[1] "1Fam, 2fmCon, Duplex, Twnhs, TwnhsE"

$HouseStyle
[1] "1.5Fin, 1.5Unf, 1Story, 2.5Fin, 2.5Unf, 2Story, SFoyer, SLvl"

$OverallQual
[1] "1, 2, 3, 4, 5, 6, 7, 8, 9, 10"

$OverallCond
[1] "1, 2, 3, 4, 5, 6, 7, 8, 9"

$YearBuilt
NULL

$YearRemodAdd
NULL

$RoofStyle
[1] "Flat, Gable, Gambrel, Hip, Mansard, Shed"

$RoofMatl
[1] "ClyTile, CompShg, Membran, Metal, Roll, Tar&Grv, WdShake, WdShngl"

$Exterior1st
[1] "AsbShng, AsphShn, BrkComm, BrkFace, CBlock, CemntBd, HdBoard, ImStucc, MetalSd, Plywood, Stone, St

$Exterior2nd
[1] "AsbShng, AsphShn, Brk Cmn, BrkFace, CBlock, CmentBd, HdBoard, ImStucc, MetalSd, Other, Plywood, St

$MasVnrType
[1] "BrkCmn, BrkFace, None, Stone, NA"

$MasVnrArea
NULL

$ExterQual
[1] "Ex, Fa, Gd, TA"

$ExterCond
[1] "Ex, Fa, Gd, Po, TA"

$Foundation
[1] "BrkTil, CBlock, PConc, Slab, Stone, Wood"

$BsmtQual
[1] "Ex, Fa, Gd, TA, NA"

$BsmtCond
```

```
[1] "Fa, Gd, Po, TA, NA"

$BsmtExposure
[1] "Av, Gd, Mn, No, NA"

$BsmtFinType1
[1] "ALQ, BLQ, GLQ, LwQ, Rec, Unf, NA"

$BsmtFinSF1
NULL

$BsmtFinType2
[1] "ALQ, BLQ, GLQ, LwQ, Rec, Unf, NA"

$BsmtFinSF2
NULL

$BsmtUnfSF
NULL

$TotalBsmtSF
NULL

$Heating
[1] "Floor, GasA, GasW, Grav, OthW, Wall"

$HeatingQC
[1] "Ex, Fa, Gd, Po, TA"

$CentralAir
[1] "N, Y"

$Electrical
[1] "FuseA, FuseF, FuseP, Mix, SBrkr, NA"

$`1stFlrSF`
NULL

$`2ndFlrSF`
NULL

$LowQualFinSF
NULL

$GrLivArea
NULL

$BsmtFullBath
[1] "0, 1, 2, 3, NA"

$BsmtHalfBath
[1] "0, 1, 2, NA"

$FullBath
```

```
[1] "0, 1, 2, 3, 4"

$HalfBath
[1] "0, 1, 2"

$BedroomAbvGr
[1] "0, 1, 2, 3, 4, 5, 6, 8"

$KitchenAbvGr
[1] "0, 1, 2, 3"

$KitchenQual
[1] "Ex, Fa, Gd, TA, NA"

$TotRmsAbvGrd
[1] "2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15"

$Functional
[1] "Maj1, Maj2, Min1, Min2, Mod, Sev, Typ, NA"

$Fireplaces
[1] "0, 1, 2, 3, 4"

$FireplaceQu
[1] "Ex, Fa, Gd, Po, TA, NA"

$GarageType
[1] "2Types, Attchd, Basment, BuiltIn, CarPort, Detchd, NA"

$GarageYrBlt
NULL

$GarageFinish
[1] "Fin, RFn, Unf, NA"

$GarageCars
[1] "0, 1, 2, 3, 4, 5, NA"

$GarageArea
NULL

$GarageQual
[1] "Ex, Fa, Gd, Po, TA, NA"

$GarageCond
[1] "Ex, Fa, Gd, Po, TA, NA"

$PavedDrive
[1] "N, P, Y"

$WoodDeckSF
NULL

$OpenPorchSF
```

```
NULL

$EnclosedPorch
NULL

$`3SsnPorch`
NULL

$ScreenPorch
NULL

$PoolArea
[1] "0, 144, 228, 368, 444, 480, 512, 519, 555, 561, 576, 648, 738, 800"

$PoolQC
[1] ", Ex, Fa, Gd, NA"

$Fence
[1] "GdPrv, GdWo, MnPrv, MnWw, NA"

$MiscFeature
[1] ", Gar2, Othr, Shed, TenC, NA"

$MiscVal
NULL

$MoSold
[1] "1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12"

$YrSold
[1] "2006, 2007, 2008, 2009, 2010"

$SaleType
[1] "COD, Con, ConLD, ConLI, ConLw, CWD, New, Oth, WD, NA"

$SaleCondition
[1] "Abnorml, AdjLand, Alloca, Family, Normal, Partial"

$SalePrice
NULL
```

Feature Names Harmonization

We start by harmonizing the feature names to be coherent with the code book. Comparing manually with the code book's possible codes, the following features have differences:

Feature	Dataset	CodeBook
MSZoning	C (all)	C
MSZoning	NA	No corresponding value
Alley	Empty string	No corresponding value

Feature	Dataset	CodeBook
Utilities	NA	No corresponding value
Neighborhood	NAmes	Names (should be NAmes)
BldgType	2fmCon	2FmCon
BldgType	Duplex	Duplx
BldgType	Twnhs	TwnhsI
Exterior1st	NA	No corresponding value
Exterior2nd	NA	No corresponding value
Exterior2nd	Wd Shng	WdShing
MasVnrType	NA	No corresponding value
Electrical	NA	No corresponding value
KitchenQual	NA	No corresponding value
Functional	NA	No corresponding value
MiscFeature	Empty string	No corresponding value
SaleType	NA	No corresponding value
Bedroom	Named 'BedroomAbvGr'	Should be named 'BedroomAbvGr' to follow the naming convention
Kitchen	Named 'KitchenAbvGr'	Should be named 'KitchenAbvGr' to follow the naming convention

The code book seems to have a naming convention but is not always respected. Since we do not know the reason behind each code and each feature name given in this code book, we will not change any of them in code book. The changes will be done on the dataset only.

To be coherent with the code book (assuming the code book is the truth), we will replace mispelled categories in the dataset by their corresponding one from the code book. Also, the empty strings and spaces will be replaced by NA since NA does not exist for these features. Note that we deduct that the string 'Twnhs' corresponds to the string 'TwnhsI' in the code book since the other codes can be easily associated.

```
## Replace all spaces or empty strings by NA.
feature.emptystring <- c("Alley", "MiscFeature")
dataset[, feature.emptystring] <- dataset %>%
    select(Alley, MiscFeature) %>%
    sapply(function(feature) gsub("^$|^ $", NA, feature))

dataset$MSZoning[dataset$MSZoning == "C (all)"] <- "C"

dataset$BldgType[dataset$BldgType == "2fmCon"] <- "2FmCon"
dataset$BldgType[dataset$BldgType == "Duplex"] <- "Duplx"
dataset$BldgType[dataset$BldgType == "Twnhs"] <- "TwnhsI"
```

```
dataset$Exterior2nd[dataset$Exterior2nd == "Wd Shng"] <- "WdShing"
```

Since we have feature names starting by a digit which is not allowed in many programming languages, we will rename them with their full name. We will also rename quality features having 'QC' or 'Qu' to keep coherence between names.

```
colnames(dataset)[colnames(dataset) == '1stFlrSF'] <- 'FirstFloorArea'
colnames(dataset)[colnames(dataset) == '2ndFlrSF'] <- 'SecondFloorArea'
colnames(dataset)[colnames(dataset) == '3SsnPorch'] <- 'ThreeSeasonPorchArea'
colnames(dataset)[colnames(dataset) == 'HeatingQC'] <- 'HeatingQualCond'
colnames(dataset)[colnames(dataset) == 'FireplaceQu'] <- 'FireplaceQual'
colnames(dataset)[colnames(dataset) == 'PoolQC'] <- 'PoolQualCond'
```

Data Coherence

We also need to check the logic in the dataset to make sure the data make sense. We will enumerate facts coming from the code book and from logic to detect anomalies in this dataset.

1. The feature 'FirstFloorArea' must not have an area of 0 ft². Otherwise, there would not have a first floor, thus no stories at all and then, no house.

The minimum area of the first floor is 334 ft². Looking at features 'HouseStyle' and 'MSSubClass' in the code book, there is neither NA value nor another value indicating that there is no story in the house. Indeed, we have 0 NA values for 'HouseStyle' and 0 NA values for 'MSSubClass'.

2. It is possible to have a second floor area of 0 ft². This is equivalent to say that there is no second floor. Therefore, the number of stories must be 1. Note that a 1.5 story house has 2 levels thus 2 floors and then the second floor area is greater than 0 ft².

The minimum area of the second floor is 0 ft². Looking at the feature 'MSSubClass' in the code book, the codes 45, 50, 60, 70, 75, 150, 160 must not be used. For the feature 'HouseStyle', the codes '1Story', 'SFoyer' and 'SLvl' are the possible choices.

```
id <- dataset %>%
    filter(SecondFloorArea == 0, !(HouseStyle %in% c("1Story", "SFoyer", "SLvl"))) %>%
    select(Id, SecondFloorArea, LowQualFinSF, HouseStyle, MSSubClass)

id <- bind_rows(id, dataset %>%
    filter(SecondFloorArea > 0, HouseStyle == "1Story") %>%
    select(Id, SecondFloorArea, LowQualFinSF, HouseStyle, MSSubClass))

id <- bind_rows(id, dataset %>%
    filter(SecondFloorArea == 0, MSSubClass %in% c(45, 50, 60, 70, 75, 150, 160)) %>%
    select(Id, SecondFloorArea, LowQualFinSF, HouseStyle, MSSubClass))

id <- bind_rows(id, dataset %>%
    filter(SecondFloorArea > 0, MSSubClass %in% c(20, 30, 40, 120)) %>%
    select(Id, SecondFloorArea, LowQualFinSF, HouseStyle, MSSubClass))

print(id)
```

```
Source: local data frame [75 x 5]

      Id SecondFloorArea LowQualFinSF HouseStyle MSSubClass
   (int)           (int)        (int)      (chr)      (int)
```

```
1     10              0          0     1.5Unf       190
2     16              0          0     1.5Unf        45
3     22              0          0     1.5Unf        45
4     52              0        360     1.5Fin        50
5     89              0        513     1.5Fin        50
6    126              0        234     1.5Fin       190
7    128              0          0     1.5Unf        45
8    164              0          0     1.5Unf        45
9    171              0        528     1.5Fin        50
10   264              0        390     1.5Fin        50
..   ...            ...        ...        ...       ...
```

3. The HouseStyle feature values must match with the values of the feature MSSubClass.

To check this fact, we have to do a mapping between values of 'HouseStyle' and 'MSSubClass'. We have to be careful with 'SLvl' and 'SFoyer' because they can be used for all types. Since we are not sure about them, we will validate with values we know they mismatch.

HouseStyle	MSSubClass
1Story	20
1Story	30
1Story	40
1Story	120
1.5Fin	50
1.5Unf	45
2Story	60
2Story	70
2Story	160
2.5Fin	75
2.5Unf	75
SFoyer	85
SFoyer	180
SLvl	80
SLvl	180

```r
houses <- dataset %>%
    filter(!(HouseStyle %in% c("SFoyer", "SLvl")))

id <- houses %>%
    filter(HouseStyle != "1Story", MSSubClass %in% c(20, 30, 40, 120)) %>%
    select(Id, HouseStyle, BldgType, MSSubClass)

id <- bind_rows(id, houses %>%
    filter(HouseStyle != "1.5Fin", MSSubClass == 50) %>%
    select(Id, HouseStyle, BldgType, MSSubClass))

id <- bind_rows(id, houses %>%
    filter(HouseStyle != "1.5Unf", MSSubClass == 45) %>%
    select(Id, HouseStyle, BldgType, MSSubClass))

id <- bind_rows(id, houses %>%
    filter(HouseStyle != "2Story", MSSubClass %in% c(60, 70, 160)) %>%
    select(Id, HouseStyle, BldgType, MSSubClass))
```

```
id <- bind_rows(id, houses %>%
    filter(HouseStyle != "2.5Fin", MSSubClass == 75) %>%
    select(Id, HouseStyle, BldgType, MSSubClass))

id <- bind_rows(id, houses %>%
    filter(HouseStyle != "2.5Unf", MSSubClass == 75) %>%
    select(Id, HouseStyle, BldgType, MSSubClass))

print(id)
```

```
Source: local data frame [44 x 4]

      Id HouseStyle BldgType MSSubClass
   (int)      (chr)    (chr)      (int)
1    608     2Story     1Fam         20
2    730     1.5Fin     1Fam         30
3   1444     1.5Unf     1Fam         30
4   2197     1.5Fin     1Fam         30
5   2555     1.5Fin     1Fam         40
6     75     2Story     1Fam         50
7     80     2Story     1Fam         50
8   1449     2Story     1Fam         50
9   2792     1.5Unf     1Fam         50
10  2881     2Story     1Fam         50
..   ...        ...      ...        ...
```

4. Per the code book, values of MSSubClass for 1 and 2 stories must match with the YearBuilt.

To verify this fact, we need to compare values of 'MSSubClass' with the 'YearBuilt' values. The fact is not respected if the year built is less than 1946 and values of 'MSSubClass' are 20, 60, 120 and 160. The case when the year built is 1946 and newer and values of 'MSSubClass' are 30 and 70 also show that the fact is not respected.

```
Source: local data frame [8 x 5]

     Id YearBuilt MSSubClass BldgType HouseStyle
  (int)     (int)      (int)    (chr)      (chr)
1  1333      1938         20     1Fam     1Story
2  1783      1939         60     1Fam     2Story
3  2127      1910         60   2FmCon     2.5Unf
4  2487      1920         60     1Fam     2Story
5  2491      1945         20     1Fam     1Story
6   837      1948         30     1Fam     1Story
7  2130      1952         70     1Fam     2Story
8  2499      1958         30     1Fam     1Story
```

We will make assumptions regarding the MSSubClass considering the house style and the year built. We know that a 2.5 story house cannot have a MSSubClass of 60. We also know that a MSSubClass set to 60 cannot have the year built older than 1946. Thus, we will assume that the code is 75 which corresponds to a 2.5 story house for all year built.

5. If there is no garage with the house, then GarageType = NA, GarageYrBlt = NA, GarageFinish = NA, GarageCars = 0, GarageArea = 0, GarageQual = NA and GarageCond = NA.

We need to get all houses where the GarageType is NA and check if the this fact's conditions are respected.

```
Source: local data frame [0 x 1]

Variables not shown: Id (int)
```

6. If there is no basement in the house, then TotalBsmtSF = 0, BsmtUnfSF = 0, BsmtFinSF2 = 0, BsmtHalfBath = 0, BsmtFullBath = 0, BsmtQual = NA and BsmtCond = NA, BsmtExposure = NA, BsmtFinType1 = NA, BsmtFinSF1 = 0, BsmtFinType2 = NA.

We need to get all houses where the TotalBsmtSF is 0 ft^2 and check if this fact's conditions are respected.

```
Source: local data frame [0 x 1]

Variables not shown: Id (int)
```

7. Per the code book, if there are no fireplaces, then FireplaceQual = NA.

We need to get all houses where the Fireplaces $\neq 0$ and check if the Fireplace Quality is NA.

```
Empty data.table (0 rows) of 3 cols: Id,Fireplaces,FireplaceQual
```

8. Per the code book, if there are no Pool, then PoolQualCond = NA.

We need to get all houses where the PoolArea $\neq 0$ ft^2 and check if the Pool Quality is NA. If there are houses, then we will replace NA values by the mean of the pool quality of all houses.

```
     Id PoolArea PoolQualCond
1: 2421      368           NA
2: 2504      444           NA
3: 2600      561           NA
```

```
PoolQualCond.mean <- getCategoryMean(dataset$PoolQualCond)

dataset <- dataset %>%
    mutate(PoolQualCond = replace(PoolQualCond, which(PoolArea != 0 & is.na(PoolQualCond)), PoolQualCon
```

9. Per the code book, the Remodel year is the same as the year built if no remodeling or additions. Then, it is true to say that YearRemodAdd $\geq$ YearBuilt.

The abnormal houses that are not respecting this fact are detected by filtering houses having the remodel year less than the year built. If it is the case, then we can verify the year when the garage was built if exists and compare with the house year built and remodeled.

```
     Id YearBuilt YearRemodAdd GarageYrBlt
1: 1877      2002         2001        2002
```

```
dataset <- dataset %>%
    mutate(YearRemodAdd = replace(YearRemodAdd, which(YearRemodAdd < YearBuilt), YearBuilt))
```

10. We verify that if the Garage Cars is 0, then the Garage Area is also 0. The converse is true since a Garage area of 0 means that there is no garage, thus no cars.

```
Empty data.table (0 rows) of 3 cols: Id,GarageArea,GarageCars
```

11. We have BsmtCond = NA (no basement per code book) if and only if BsmtQual = NA which means no basement per the code book.

```
     Id BsmtCond BsmtQual
1: 2041       NA       Gd
2: 2186       NA       TA
3: 2525       NA       TA
```

```
     Id BsmtCond BsmtQual
1: 2218       Fa       NA
```

```
2: 2219       TA       NA
```

```
dataset <- dataset %>%
    mutate(BsmtQual = replace(BsmtQual, !is.na(BsmtCond) & is.na(BsmtQual), BsmtCond)) %>%
    mutate(BsmtCond = replace(BsmtCond, is.na(BsmtCond) & !is.na(BsmtQual), BsmtQual))
```

12. We have MasVnrType = None if and only if MasVnrArea = 0 ft².

We have two cases where it is hard to check which one is right.

- Case when MasVnrType = 'None' and MasVnrArea $\neq$ 0 ft²
- Case when MasVnrType $\neq$ 'None' and MasVnrArea = 0 ft²

```
     Id MasVnrType MasVnrArea
1:  625       None        288
2:  774       None          1
3: 1231       None          1
4: 1301       None        344
5: 1335       None        312
6: 1670       None        285
7: 2453       None          1
```

```
     Id MasVnrType MasVnrArea
1:  689    BrkFace          0
2: 1242      Stone          0
3: 2320    BrkFace          0
```

```
MasVnrArea.threshold <- 10

dataset <- dataset %>%
    mutate(MasVnrType = replace(MasVnrType, MasVnrType != "None" & MasVnrArea == 0, "None")) %>%
    mutate(MasVnrArea = replace(MasVnrArea, MasVnrType == "None" & MasVnrArea <= MasVnrArea.threshold, 

MasVnrType.mean <- getCategoryMean(dataset$MasVnrType)
dataset <- dataset %>%
    mutate(MasVnrType = replace(MasVnrType, MasVnrType == "None" & MasVnrArea > MasVnrArea.threshold, M
```

Anomalies Detection

We define a house as being an anomaly if $\|Y - P\| > \epsilon$ where $Y = (x, y)$ is the point belonging to the regression linear model and $P = (x, z)$ a point not on the regression linear model. Also, x is the ground living area, y and z the sale price, and $\epsilon > 0$ the threshold.

Regarding the overall quality, the sale price and the ground living area, we expect that the sale price will increase when the overall quality increases and the ground living area increases. This is verified in the data exploratory section.

Taking houses having their overall quality = 10 and their ground living area greater than 4000 ft², the sale price should be part of the highest sale prices. If there are houses respecting these conditions with a sale price over 240000$ than what the regression model gives, then this may be possible, but if it is lower, than this is exceptionnel.

```
     Id GrLivArea SalePrice
1:  524      4676    184750
2:  692      4316    755000
3: 1183      4476    745000
4: 1299      5642    160000
```

```
    Id ApproxPrice SalePrice PriceDifference
1  524    519510.6    184750        334760.6
2  692    480943.7    755000        274056.3
3 1183    498084.5    745000        246915.5
4 1299    622998.5    160000        462998.5
```

Missing Values

Per the code book of this dataset, we know that generally, the NA values mean 'No' or 'None' and they are used only for some categorical features. The other NA values that are not in the code book will be explained case by case. This goes also for the empty strings that will be replaced by NA.

- Case when NA means 'None' or 'No'
- Case when an integer feature has 0 and NA as possible values
- Case when a numeric value has 0 and NA as possible values
- Case when a category is NA where NA means 'No', and the numeric feature is not zero
- Case when a category is not NA where NA means 'No', and the numeric feature is NA where 0 has a clear meaning

Features having NA values where NA means 'None' or 'No' will be replaced by 0.

```
dataset <- dataset %>%
    mutate(Alley = replace(Alley, is.na(Alley), 0)) %>%
    mutate(BsmtQual = replace(BsmtQual, is.na(BsmtQual), 0)) %>%
    mutate(BsmtCond = replace(BsmtCond, is.na(BsmtCond), 0)) %>%
    mutate(BsmtExposure = replace(BsmtExposure, is.na(BsmtExposure), 0)) %>%
    mutate(BsmtFinType1 = replace(BsmtFinType1, is.na(BsmtFinType1), 0)) %>%
    mutate(BsmtFinType2 = replace(BsmtFinType2, is.na(BsmtFinType2), 0)) %>%
    mutate(FireplaceQual = replace(FireplaceQual, is.na(FireplaceQual), 0)) %>%
    mutate(GarageType = replace(GarageType, is.na(GarageType), 0)) %>%
    mutate(GarageFinish = replace(GarageFinish, is.na(GarageFinish), 0)) %>%
    mutate(GarageQual = replace(GarageQual, is.na(GarageQual), 0)) %>%
    mutate(GarageCond = replace(GarageCond, is.na(GarageCond), 0)) %>%
    mutate(PoolQualCond = replace(PoolQualCond, is.na(PoolQualCond), 0)) %>%
    mutate(Fence = replace(Fence, is.na(Fence), 0)) %>%
    mutate(MiscFeature = replace(MiscFeature, is.na(MiscFeature), 0))
```

However, it is possible to solve some NA values by analysing the value used for other features strongly related. For example, some integer features like GarageCars and GarageArea have NA values. At the first glance, we cannot state that NA means 0 since 0 already has a meaning. It could be a "No Information", but looking at the GarageQual and GarageCond features, we notice that their value is NA as well. This means that this house has no garage per the code book. Therefore, we will replace NA values by 0 for GarageArea and GarageCars.

For features like "BsmtFullBath", the value 0 means that we do not have full bathroom in the basement. Thus, we cannot replace NA by 0 if there is a basement. Otherwise, the house has no basement, thus no full bathroom in the basement. In this case only, we can replace NA by 0.

We expect that numeric features where the value 0 means the same thing as a NA value. For example, a garage area of 0 means that there is no garage with this house. However, if the value 0 is used for an amount of money or for a geometric measure (e.g. area), then it is a real 0.

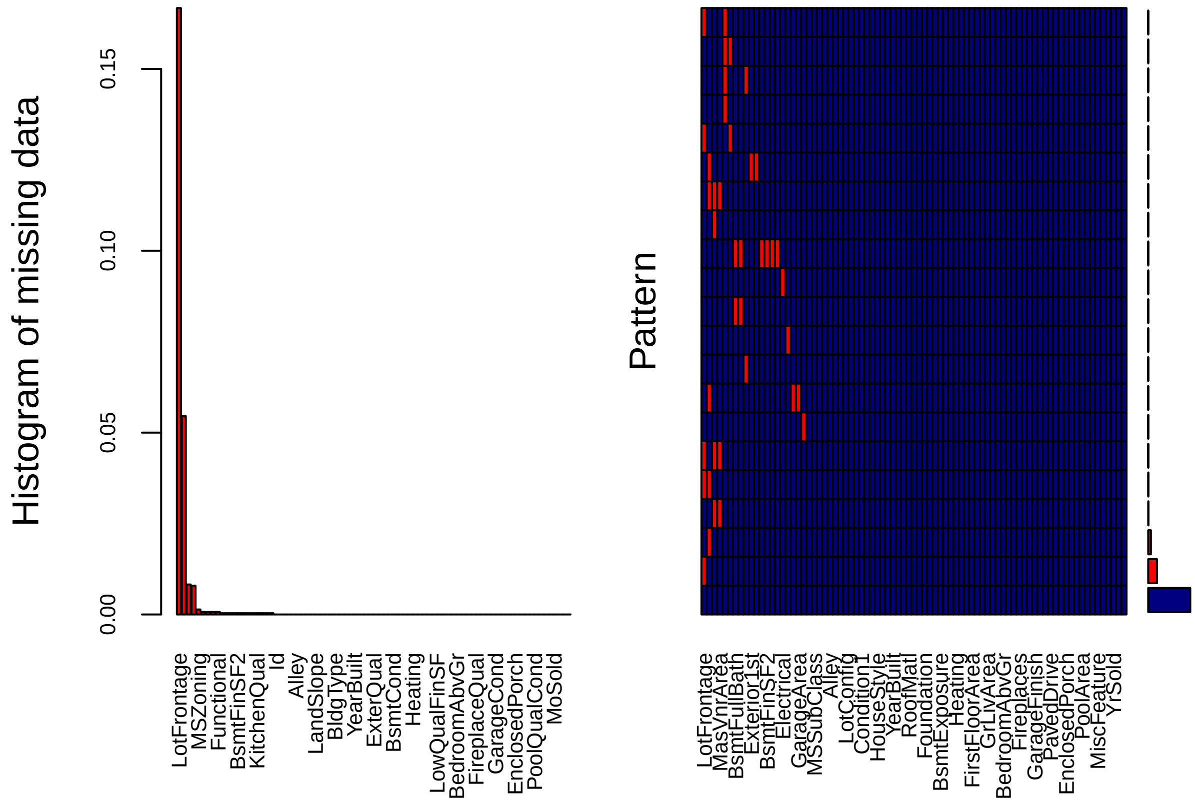
For "year" features (e.g. GarageYrBlt), if the values are NA, then we can replace them by 0 without loss of generality. A year 0 is theorically possible, but in our context, it is impossible. But, using 0 will decrease the mean and will add noise to the data since the difference between the minimum year and zero is large: NA.

Another case is when a feature uses the value NA to indicate that the information is missing. For example, the feature "KitchenQual" is not supposed to have the value NA per the code book. If the value NA is used, then it really means "No Information" and we cannot replace it by 0. Normally, we would exclude this house of the dataset, but this house is taken from the test set, thus we must not remove it.

For those cases, we need to use imputation on missing data (NA value). We could calculate the mean for a given feature and use this value to replace NA values. But it is more accurate to predict what value to use by using the other features since we have many of them.

```
           Id       MSSubClass        MSZoning
            0                0               4
  LotFrontage          LotArea          Street
          486                0               0
        Alley         LotShape     LandContour
            0                0               0
    Utilities        LotConfig       LandSlope
            2                0               0
 Neighborhood       Condition1      Condition2
            0                0               0
     BldgType       HouseStyle     OverallQual
            0                0               0
  OverallCond        YearBuilt    YearRemodAdd
            0                0               0
    RoofStyle         RoofMatl     Exterior1st
            0                0               1
  Exterior2nd       MasVnrType      MasVnrArea
            1               24              23
    ExterQual        ExterCond      Foundation
            0                0               0
     BsmtQual         BsmtCond    BsmtExposure
            0                0               0
 BsmtFinType1       BsmtFinSF1    BsmtFinType2
            0                1               0
   BsmtFinSF2        BsmtUnfSF     TotalBsmtSF
            1                1               1
      Heating  HeatingQualCond      CentralAir
            0                0               0
   Electrical   FirstFloorArea SecondFloorArea
            1                0               0
 LowQualFinSF        GrLivArea    BsmtFullBath
            0                0               2
 BsmtHalfBath         FullBath        HalfBath
            2                0               0
 BedroomAbvGr     KitchenAbvGr     KitchenQual
            0                0               1
 TotRmsAbvGrd       Functional      Fireplaces
            0                2               0
FireplaceQual       GarageType     GarageYrBlt
            0                0             159
 GarageFinish       GarageCars      GarageArea
            0                1               1
   GarageQual       GarageCond      PavedDrive
            0                0               0
   WoodDeckSF      OpenPorchSF   EnclosedPorch
            0                0               0
```

```
ThreeSeasonPorchArea          ScreenPorch             PoolArea
                   0                    0                    0
       PoolQualCond                Fence          MiscFeature
                   0                    0                    0
             MiscVal               MoSold               YrSold
                   0                    0                    0
            SaleType        SaleCondition            SalePrice
                   1                    0                    0
```




```
 Variables sorted by number of missings:
```

Variable	Count
LotFrontage	0.1667238422
GarageYrBlt	0.0545454545
MasVnrType	0.0082332762
MasVnrArea	0.0078902230
MSZoning	0.0013722127
Utilities	0.0006861063
BsmtFullBath	0.0006861063
BsmtHalfBath	0.0006861063
Functional	0.0006861063
Exterior1st	0.0003430532
Exterior2nd	0.0003430532
BsmtFinSF1	0.0003430532
BsmtFinSF2	0.0003430532
BsmtUnfSF	0.0003430532
TotalBsmtSF	0.0003430532
Electrical	0.0003430532
KitchenQual	0.0003430532

GarageCars	0.0003430532
GarageArea	0.0003430532
SaleType	0.0003430532
Id	0.0000000000
MSSubClass	0.0000000000
LotArea	0.0000000000
Street	0.0000000000
Alley	0.0000000000
LotShape	0.0000000000
LandContour	0.0000000000
LotConfig	0.0000000000
LandSlope	0.0000000000
Neighborhood	0.0000000000
Condition1	0.0000000000
Condition2	0.0000000000
BldgType	0.0000000000
HouseStyle	0.0000000000
OverallQual	0.0000000000
OverallCond	0.0000000000
YearBuilt	0.0000000000
YearRemodAdd	0.0000000000
RoofStyle	0.0000000000
RoofMatl	0.0000000000
ExterQual	0.0000000000
ExterCond	0.0000000000
Foundation	0.0000000000
BsmtQual	0.0000000000
BsmtCond	0.0000000000
BsmtExposure	0.0000000000
BsmtFinType1	0.0000000000
BsmtFinType2	0.0000000000
Heating	0.0000000000
HeatingQualCond	0.0000000000
CentralAir	0.0000000000
FirstFloorArea	0.0000000000
SecondFloorArea	0.0000000000
LowQualFinSF	0.0000000000
GrLivArea	0.0000000000
FullBath	0.0000000000
HalfBath	0.0000000000
BedroomAbvGr	0.0000000000
KitchenAbvGr	0.0000000000
TotRmsAbvGrd	0.0000000000
Fireplaces	0.0000000000
FireplaceQual	0.0000000000
GarageType	0.0000000000
GarageFinish	0.0000000000
GarageQual	0.0000000000
GarageCond	0.0000000000
PavedDrive	0.0000000000
WoodDeckSF	0.0000000000
OpenPorchSF	0.0000000000
EnclosedPorch	0.0000000000
ThreeSeasonPorchArea	0.0000000000

```
 ScreenPorch 0.0000000000
    PoolArea 0.0000000000
PoolQualCond 0.0000000000
       Fence 0.0000000000
 MiscFeature 0.0000000000
     MiscVal 0.0000000000
      MoSold 0.0000000000
      YrSold 0.0000000000
SaleCondition 0.0000000000
   SalePrice 0.0000000000
```

For the Masonry veneer type (MasVnrType) feature, the value "None" means that the house does not have a masonry veneer per the code book. If some houses have the value NA, then it will mean that the information is missing.

Note that it is possible to have information on the masonry veneer area but not on the type (vice-versa could be possible as well). In that case, we cannot deduct what will be the value to replace NA. We cannot replace NA by 0 for the area because 0 means *None* which is a valid choice. The best choice we can take is to replace NA value by the mean value of the feature.

Data Exploratory

The objective is to visualize and understand the relationships between features in the dataset we have to solve the problem. We will also compare changes we will make to this dataset to validate if they have significant influance on the sale price or not.

Features

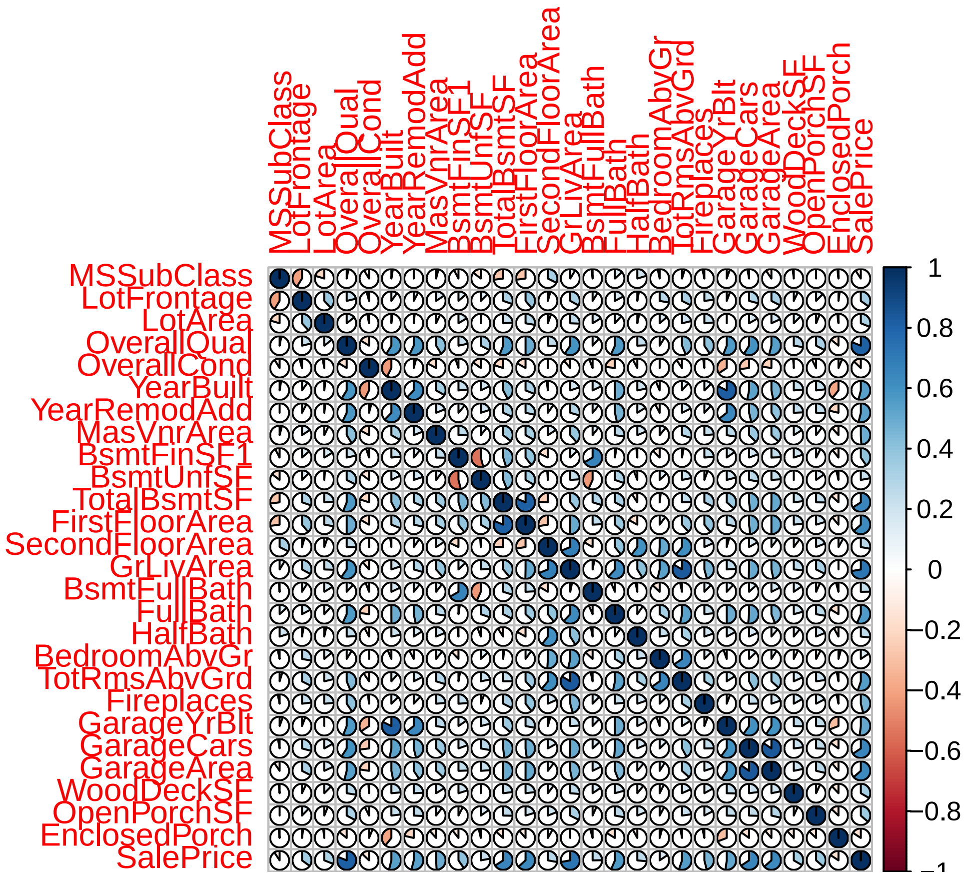
Here is the list of features with their type.

```
Classes 'data.table' and 'data.frame':  2915 obs. of  81 variables:
 $ Id                  : int  1 2 3 4 5 6 7 8 9 10 ...
 $ MSSubClass          : int  60 20 60 70 60 50 20 60 50 190 ...
 $ MSZoning            : chr  "RL" "RL" "RL" "RL" ...
 $ LotFrontage         : int  65 80 68 60 84 85 75 NA 51 50 ...
 $ LotArea             : int  8450 9600 11250 9550 14260 14115 10084 10382 6120 7420 ...
 $ Street              : chr  "Pave" "Pave" "Pave" "Pave" ...
 $ Alley               : chr  "0" "0" "0" "0" ...
 $ LotShape            : chr  "Reg" "Reg" "IR1" "IR1" ...
 $ LandContour         : chr  "Lvl" "Lvl" "Lvl" "Lvl" ...
 $ Utilities           : chr  "AllPub" "AllPub" "AllPub" "AllPub" ...
 $ LotConfig           : chr  "Inside" "FR2" "Inside" "Corner" ...
 $ LandSlope           : chr  "Gtl" "Gtl" "Gtl" "Gtl" ...
 $ Neighborhood        : chr  "CollgCr" "Veenker" "CollgCr" "Crawfor" ...
 $ Condition1          : chr  "Norm" "Feedr" "Norm" "Norm" ...
 $ Condition2          : chr  "Norm" "Norm" "Norm" "Norm" ...
 $ BldgType            : chr  "1Fam" "1Fam" "1Fam" "1Fam" ...
 $ HouseStyle          : chr  "2Story" "1Story" "2Story" "2Story" ...
 $ OverallQual         : int  7 6 7 7 8 5 8 7 7 5 ...
 $ OverallCond         : int  5 8 5 5 5 5 5 6 5 6 ...
 $ YearBuilt           : int  2003 1976 2001 1915 2000 1993 2004 1973 1931 1939 ...
 $ YearRemodAdd        : int  2003 1976 2002 1970 2000 1995 2005 1973 1950 1950 ...
 $ RoofStyle           : chr  "Gable" "Gable" "Gable" "Gable" ...
```

```
$ RoofMatl            : chr  "CompShg" "CompShg" "CompShg" "CompShg" ...
$ Exterior1st         : chr  "VinylSd" "MetalSd" "VinylSd" "Wd Sdng" ...
$ Exterior2nd         : chr  "VinylSd" "MetalSd" "VinylSd" "WdShing" ...
$ MasVnrType          : chr  "BrkFace" "None" "BrkFace" "None" ...
$ MasVnrArea          : num  196 0 162 0 350 0 186 240 0 0 ...
$ ExterQual           : chr  "Gd" "TA" "Gd" "TA" ...
$ ExterCond           : chr  "TA" "TA" "TA" "TA" ...
$ Foundation          : chr  "PConc" "CBlock" "PConc" "BrkTil" ...
$ BsmtQual            : chr  "Gd" "Gd" "Gd" "TA" ...
$ BsmtCond            : chr  "TA" "TA" "TA" "Gd" ...
$ BsmtExposure        : chr  "No" "Gd" "Mn" "No" ...
$ BsmtFinType1        : chr  "GLQ" "ALQ" "GLQ" "ALQ" ...
$ BsmtFinSF1          : int  706 978 486 216 655 732 1369 859 0 851 ...
$ BsmtFinType2        : chr  "Unf" "Unf" "Unf" "Unf" ...
$ BsmtFinSF2          : int  0 0 0 0 0 0 0 32 0 0 ...
$ BsmtUnfSF           : int  150 284 434 540 490 64 317 216 952 140 ...
$ TotalBsmtSF         : int  856 1262 920 756 1145 796 1686 1107 952 991 ...
$ Heating             : chr  "GasA" "GasA" "GasA" "GasA" ...
$ HeatingQualCond     : chr  "Ex" "Ex" "Ex" "Gd" ...
$ CentralAir          : chr  "Y" "Y" "Y" "Y" ...
$ Electrical          : chr  "SBrkr" "SBrkr" "SBrkr" "SBrkr" ...
$ FirstFloorArea      : int  856 1262 920 961 1145 796 1694 1107 1022 1077 ...
$ SecondFloorArea     : int  854 0 866 756 1053 566 0 983 752 0 ...
$ LowQualFinSF        : int  0 0 0 0 0 0 0 0 0 0 ...
$ GrLivArea           : int  1710 1262 1786 1717 2198 1362 1694 2090 1774 1077 ...
$ BsmtFullBath        : int  1 0 1 1 1 1 1 1 0 1 ...
$ BsmtHalfBath        : int  0 1 0 0 0 0 0 0 0 0 ...
$ FullBath            : int  2 2 2 1 2 1 2 2 2 1 ...
$ HalfBath            : int  1 0 1 0 1 1 0 1 0 0 ...
$ BedroomAbvGr        : int  3 3 3 3 4 1 3 3 2 2 ...
$ KitchenAbvGr        : int  1 1 1 1 1 1 1 1 2 2 ...
$ KitchenQual         : chr  "Gd" "TA" "Gd" "Gd" ...
$ TotRmsAbvGrd        : int  8 6 6 7 9 5 7 7 8 5 ...
$ Functional          : chr  "Typ" "Typ" "Typ" "Typ" ...
$ Fireplaces          : int  0 1 1 1 1 0 1 2 2 2 ...
$ FireplaceQual       : chr  "0" "TA" "TA" "Gd" ...
$ GarageType          : chr  "Attchd" "Attchd" "Attchd" "Detchd" ...
$ GarageYrBlt         : int  2003 1976 2001 1998 2000 1993 2004 1973 1931 1939 ...
$ GarageFinish        : chr  "RFn" "RFn" "RFn" "Unf" ...
$ GarageCars          : int  2 2 2 3 3 2 2 2 2 1 ...
$ GarageArea          : int  548 460 608 642 836 480 636 484 468 205 ...
$ GarageQual          : chr  "TA" "TA" "TA" "TA" ...
$ GarageCond          : chr  "TA" "TA" "TA" "TA" ...
$ PavedDrive          : chr  "Y" "Y" "Y" "Y" ...
$ WoodDeckSF          : int  0 298 0 0 192 40 255 235 90 0 ...
$ OpenPorchSF         : int  61 0 42 35 84 30 57 204 0 4 ...
$ EnclosedPorch       : int  0 0 0 272 0 0 0 228 205 0 ...
$ ThreeSeasonPorchArea: int  0 0 0 0 0 320 0 0 0 0 ...
$ ScreenPorch         : int  0 0 0 0 0 0 0 0 0 0 ...
$ PoolArea            : int  0 0 0 0 0 0 0 0 0 0 ...
$ PoolQualCond        : chr  "" "" "" "" ...
$ Fence               : chr  "0" "0" "0" "0" ...
$ MiscFeature         : chr  "0" "0" "0" "0" ...
$ MiscVal             : int  0 0 0 0 0 700 0 350 0 0 ...
```

```
$ MoSold              : int  2 5 9 2 12 10 8 11 4 1 ...
$ YrSold              : int  2008 2007 2008 2006 2008 2009 2007 2009 2008 2008 ...
$ SaleType            : chr  "WD" "WD" "WD" "WD" ...
$ SaleCondition       : chr  "Normal" "Normal" "Normal" "Abnorml" ...
$ SalePrice           : num  208500 181500 223500 140000 250000 ...
- attr(*, ".internal.selfref")=<externalptr>
```

We see now a plot of the correlation between numeric features of the train set.




```
               SalePriceCorrelation
SalePrice                1.00000000
OverallQual              0.81003032
GrLivArea                0.72186802
TotalBsmtSF              0.65889257
GarageCars               0.65650069
GarageArea               0.63608204
FirstFloorArea           0.63411577
FullBath                 0.56430696
TotRmsAbvGrd             0.55195364
YearBuilt                0.54048713
YearRemodAdd             0.53929377
GarageYrBlt              0.51998943
MasVnrArea               0.49159788
Fireplaces               0.46035638
BsmtFinSF1               0.40362064
OpenPorchSF              0.36363062
LotFrontage              0.34974593
```

```
WoodDeckSF              0.33393258
LotArea                 0.31017969
SecondFloorArea         0.27861778
HalfBath                0.26574979
BsmtFullBath            0.24744764
BsmtUnfSF               0.22112301
BedroomAbvGr            0.15584971
MSSubClass             -0.09306923
OverallCond            -0.12941377
EnclosedPorch          -0.15728895
```

Regarding the sale price, we note that some features are more than 60% correlated with the sale price. We will produce plots for each of them to get insights.

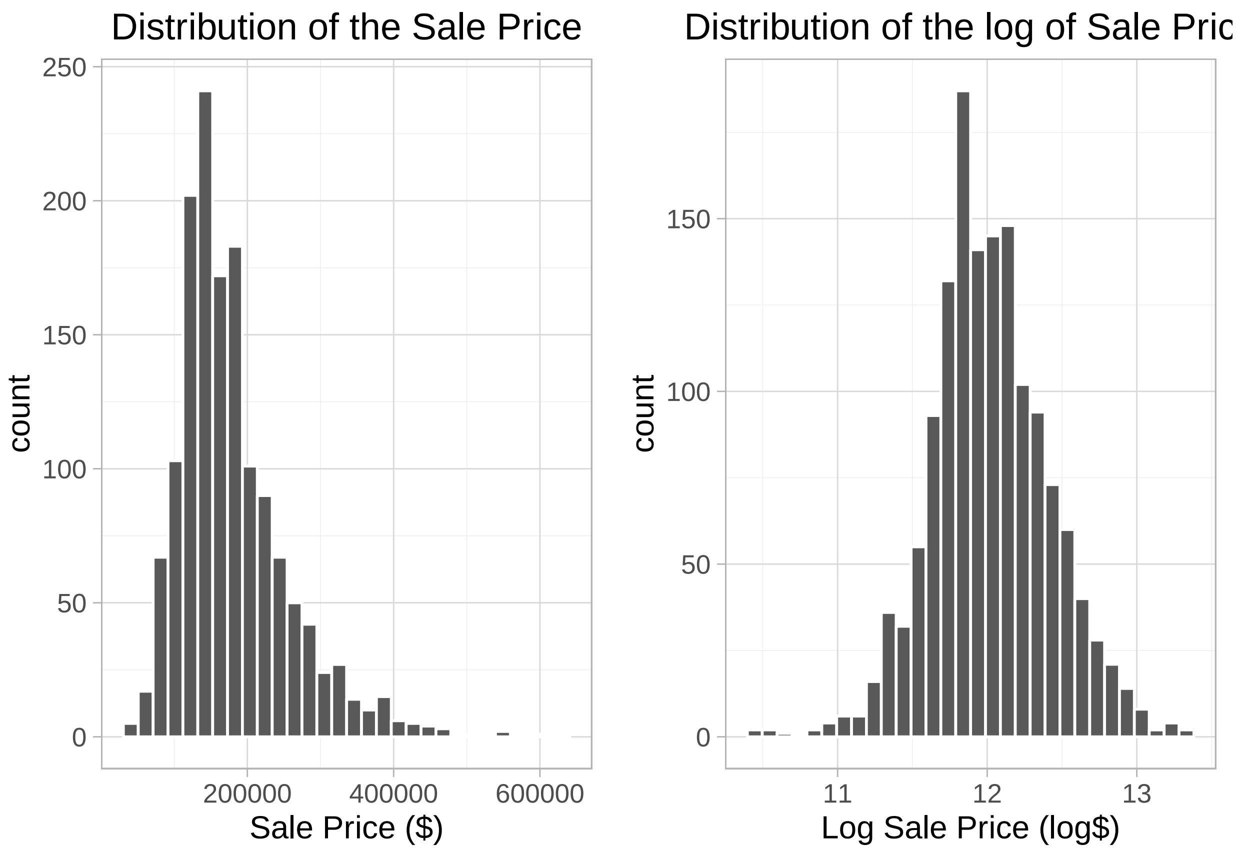
Dependant vs Independent Features

With the current features we have in the dataset, we have to check which features are dependent of other features versus which ones are independent. At first glance in the dataset, features representing totals and overalls seems dependent.

- $GrLivArea = FirstFloorArea + SecondFloorArea + LowQualFinSF$
- $TotalBsmtSF = BsmtUnfSF + BsmtFinSF1 + BsmtFinSF2$

Sale Price

The sale price should follow the normal distribution. However, the sale price does not totally follow the normal law, thus we need to normalize the sale price by taking its logarithm.

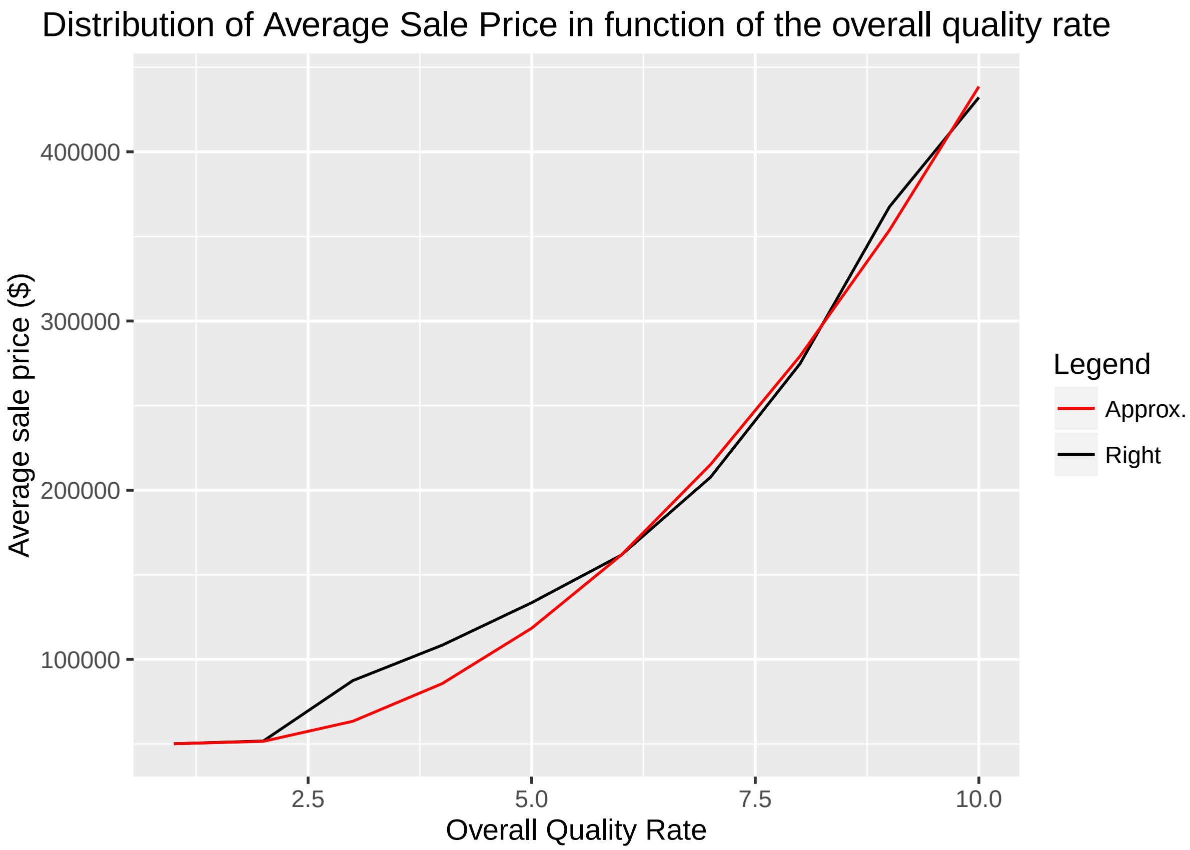



```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  34900  129900  163000  180200  214000  625000
```

Overall Quality Rate

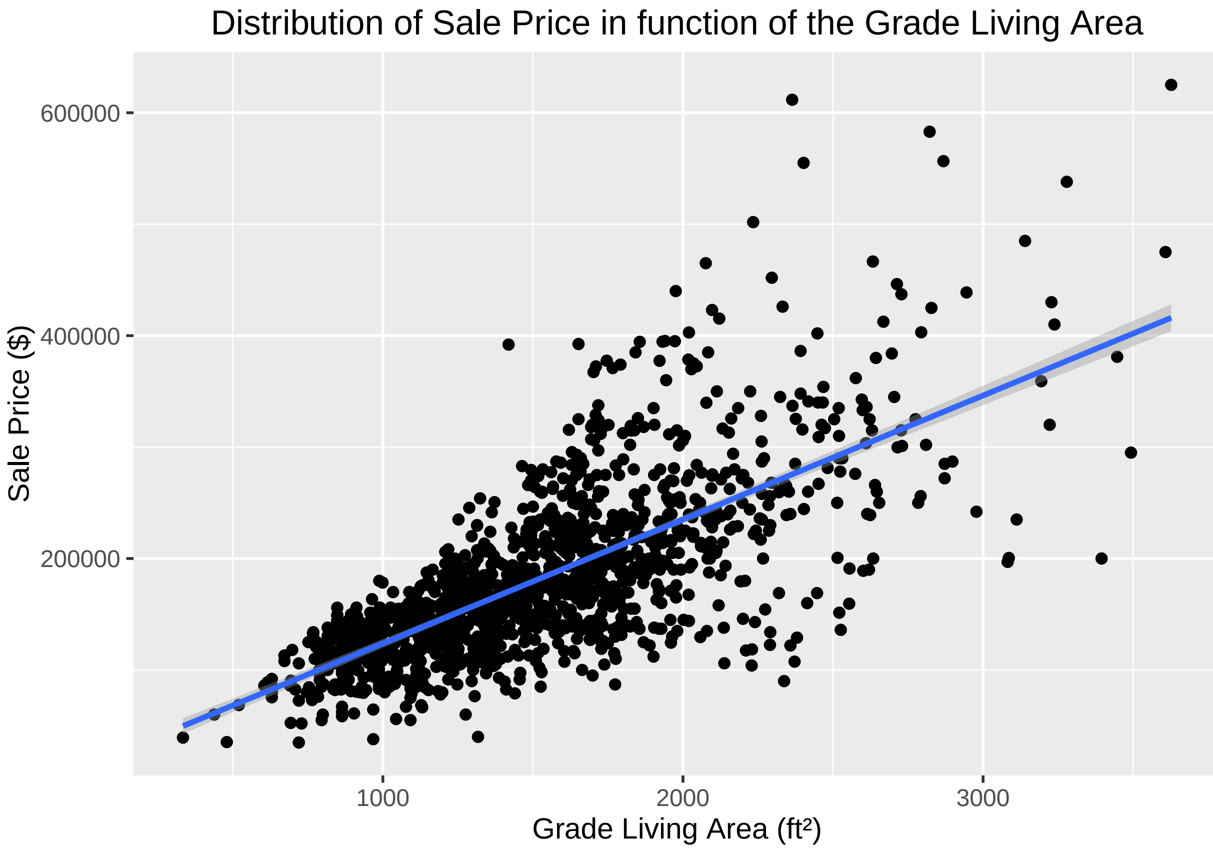
The overall quality rate is the most correlated feature to the sale price as seen previously. We look at the average sale price for each overall quality rate and try to figure out an equation that will best approximate our data.

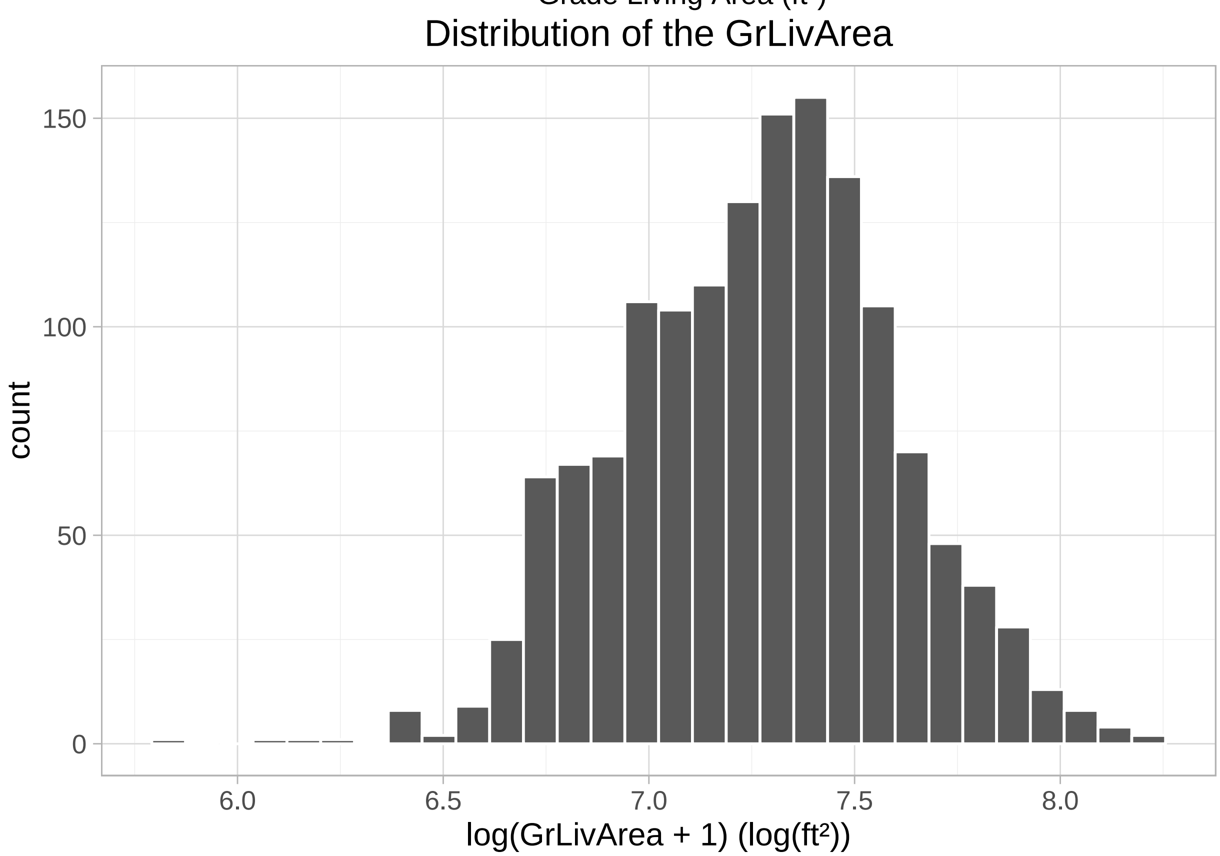
```
Source: local data table [10 x 2]

   OverallQual MeanSalePrice
         (int)         (dbl)
1            1      50150.00
2            2      51770.33
3            3      87473.75
4            4     108420.66
5            5     133523.35
6            6     161603.03
7            7     207716.42
8            8     274735.54
9            9     367513.02
10          10     432131.50
```



Note that the equation used to approximate is a parabola where the equation has been built from 3 points (OverallQual, MeanSalePrice) where the overall quality rates chosen are 1, 6 and 10 with their corresponding average sale price. The equation used to approximate is $M(Q) = \frac{939113}{180}Q^2 - \frac{2561483}{180}Q + \frac{354979}{6}$ where Q is the overall quality rate and $M(Q)$ is the mean sale price in function of Q.

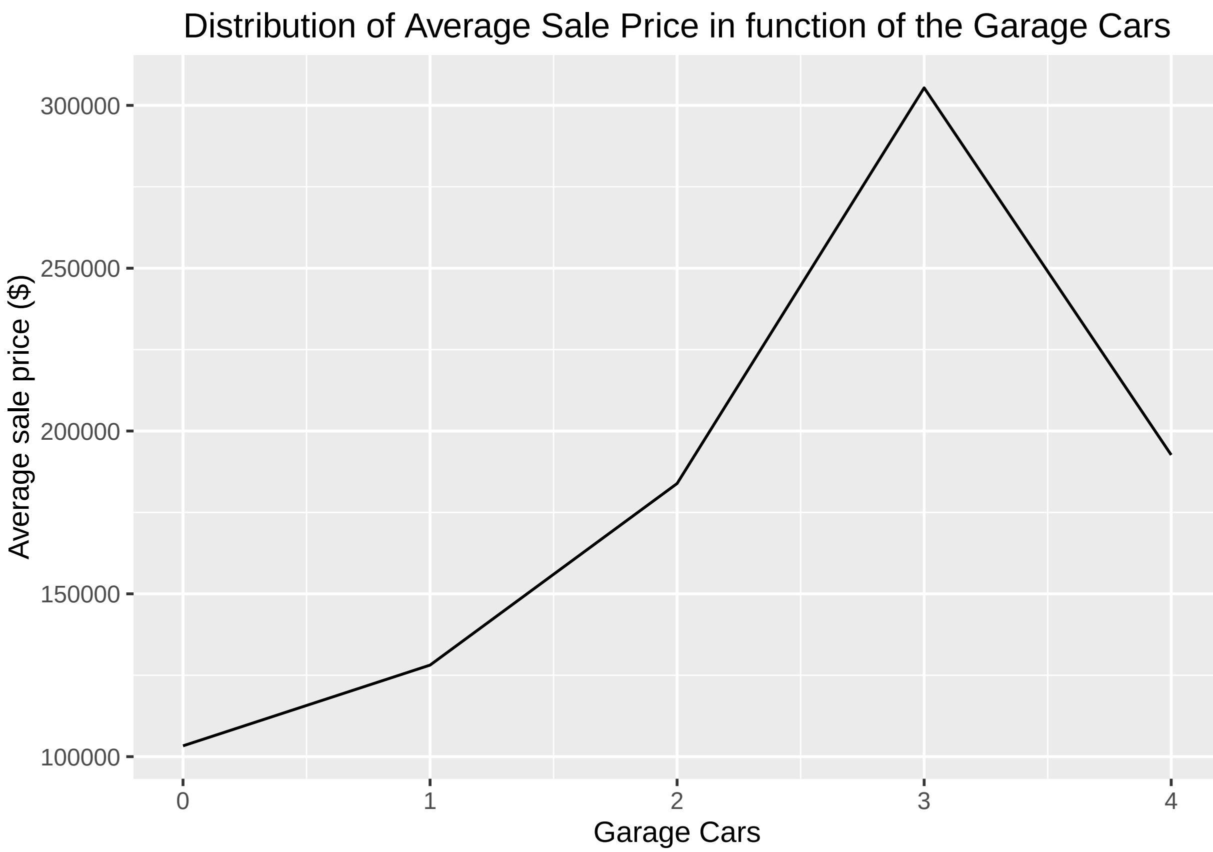
Above grade (ground) living area





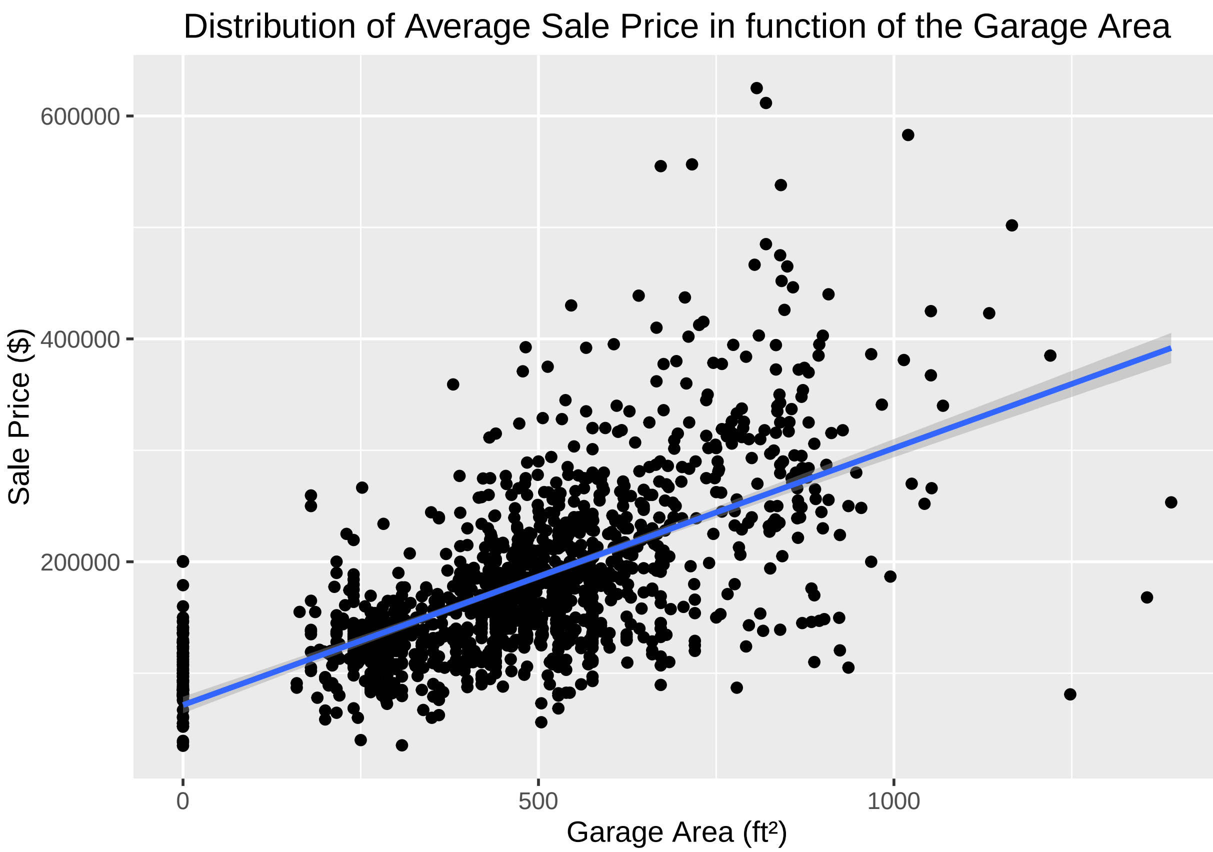
Garage Cars

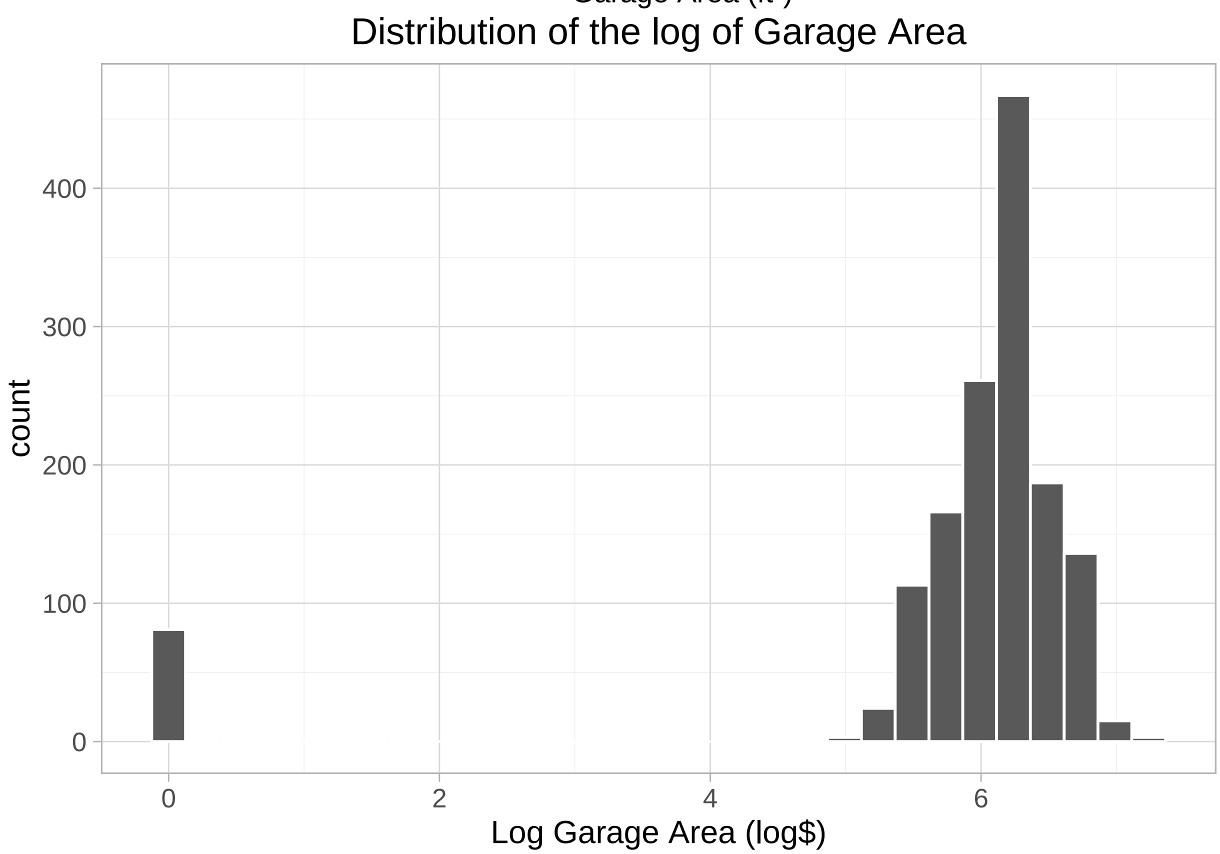
```
Source: local data table [5 x 2]

  GarageCars MeanSalePrice
       (int)         (dbl)
1          0      103317.3
2          1      128116.7
3          2      183880.6
4          3      305389.8
5          4      192655.8
```



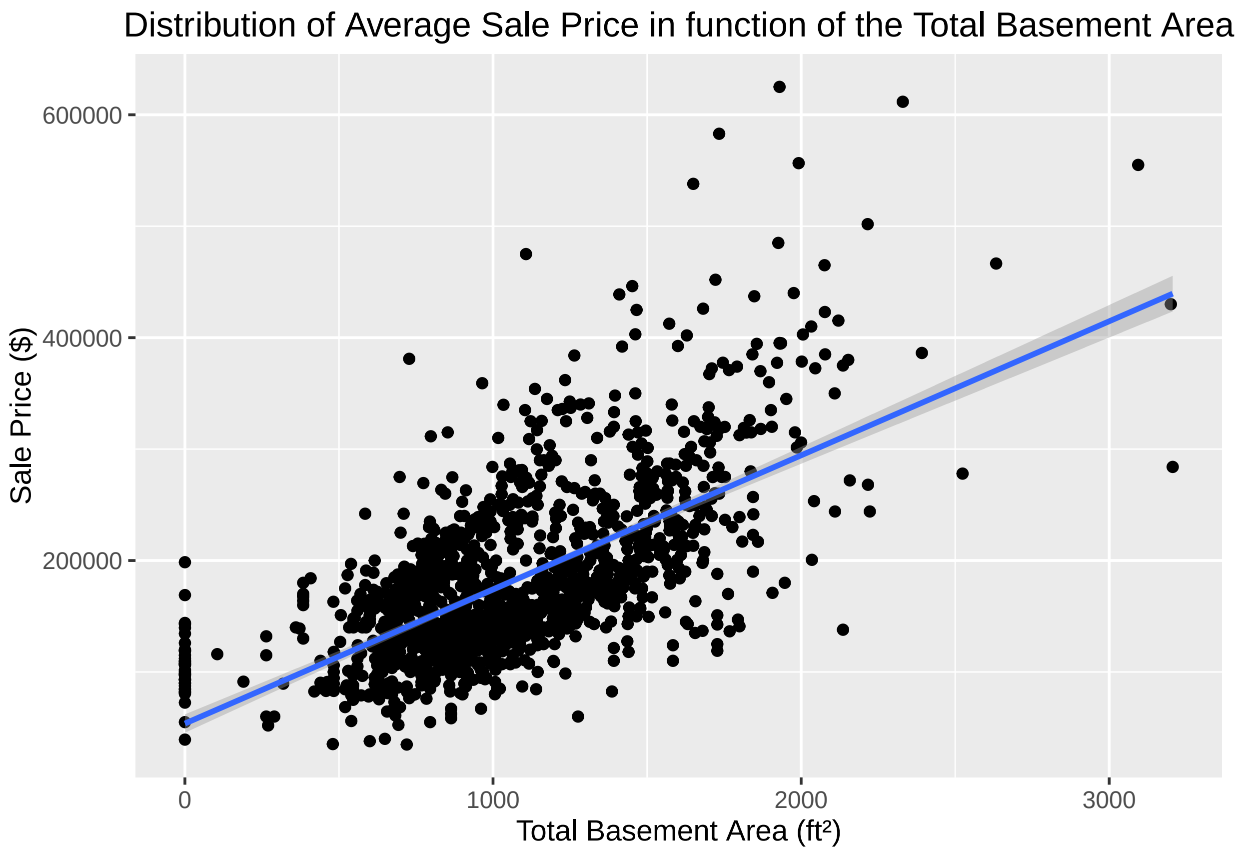

```
     Id OverallQual GrLivArea SalePrice
1:  421           7      1344    206300
2:  748           7      2640    265979
3: 1191           4      1622    168000
4: 1341           4       872    123000
5: 1351           5      2634    200000
```

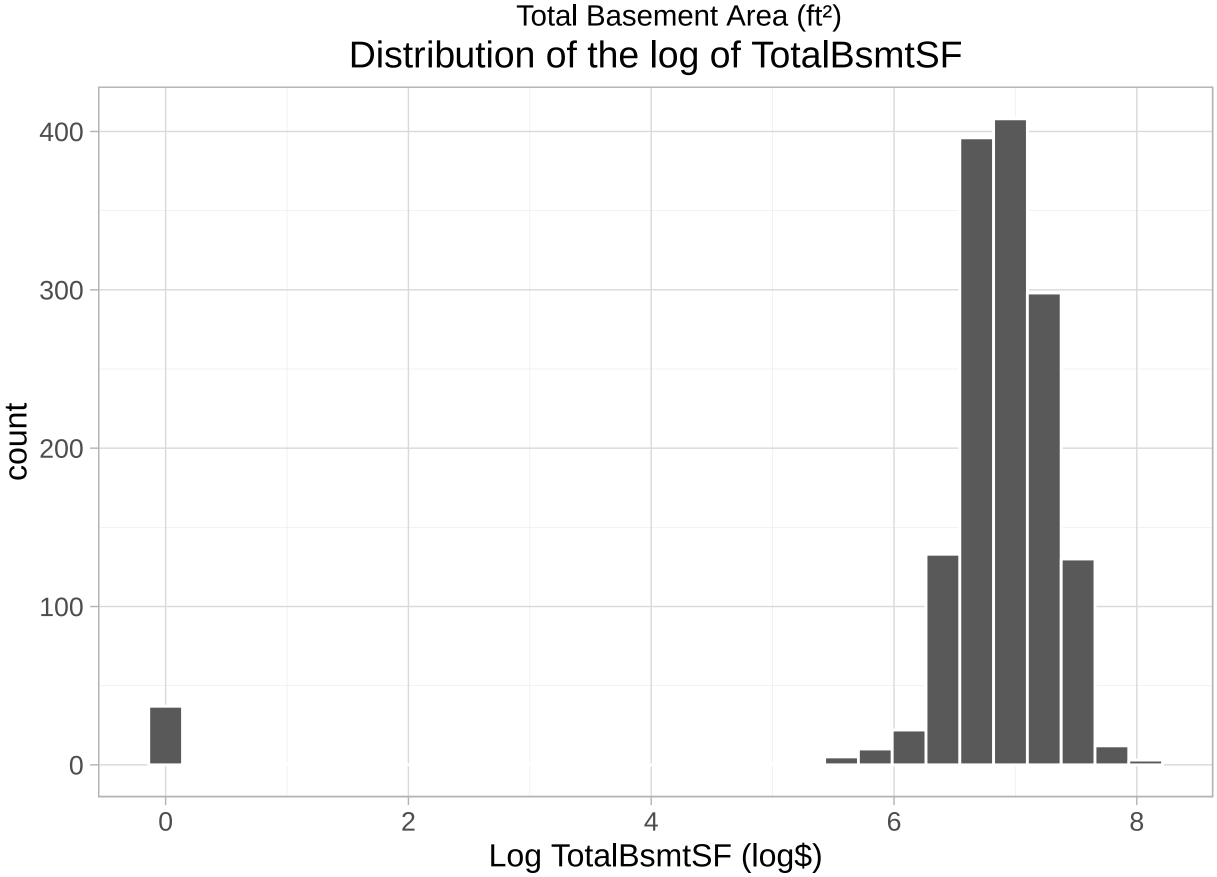
Garage Area


Distribution of Average Sale Price in function of the Garage Area
600000
400000
200000
Sale Price ($)
0
500
1000
Garage Area (ft²)

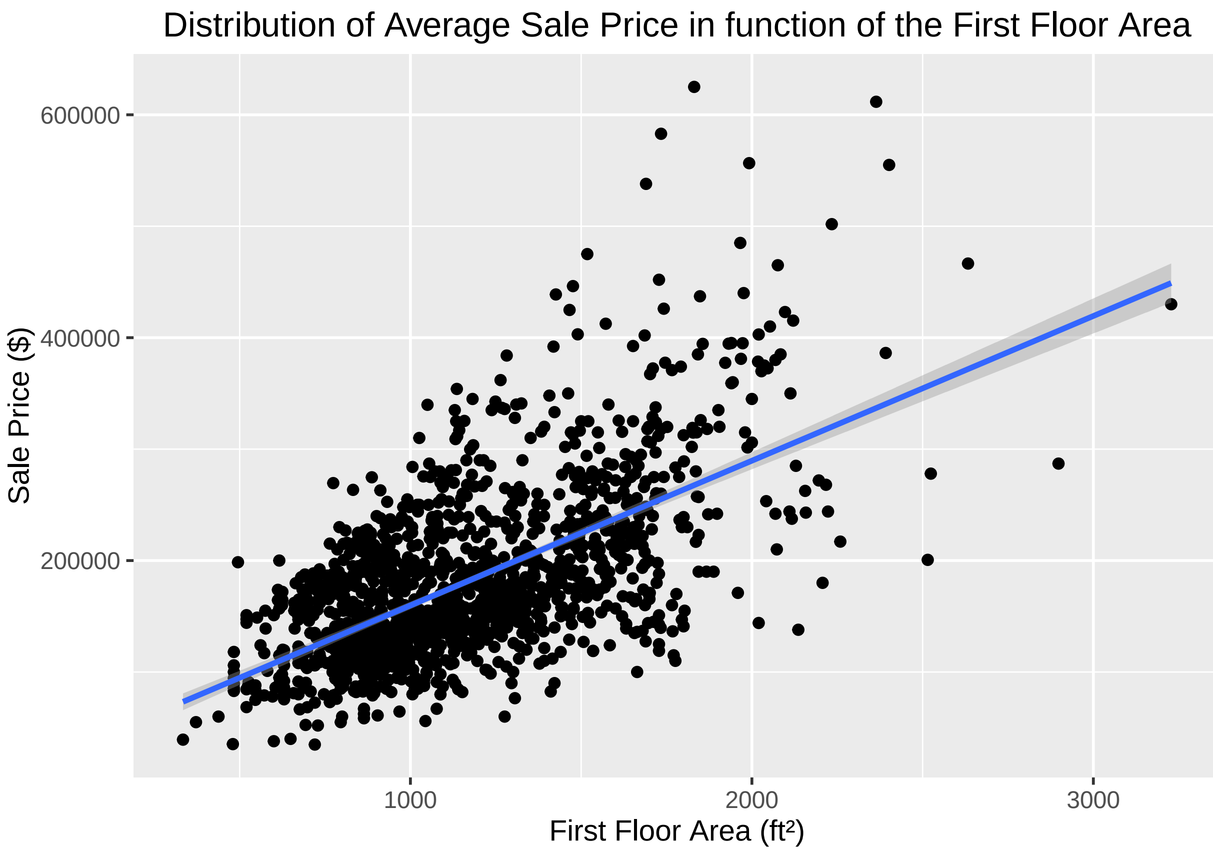

Distribution of the log of Garage Area
400
300
200
100
0
count
0
2
4
6
Log Garage Area (log$)

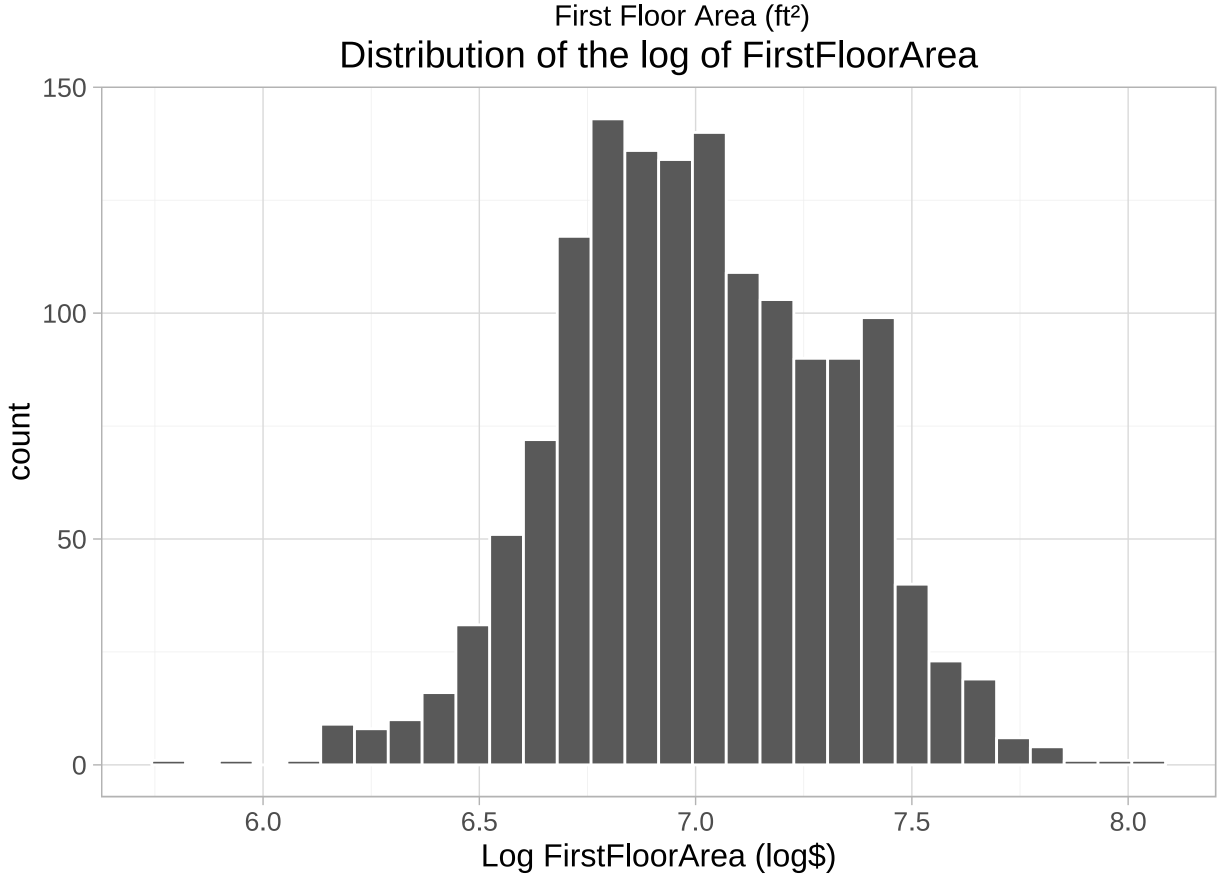
Total Basement Area





First Floor Area





Feature Engineering

In this section, we create, modify and delete features to help the prediction. We will impute missing values not yet resolved using the MICE package. We also scale some features like the quality ones. Then, we check for skewed features for which we normalize.

Feature Replacement

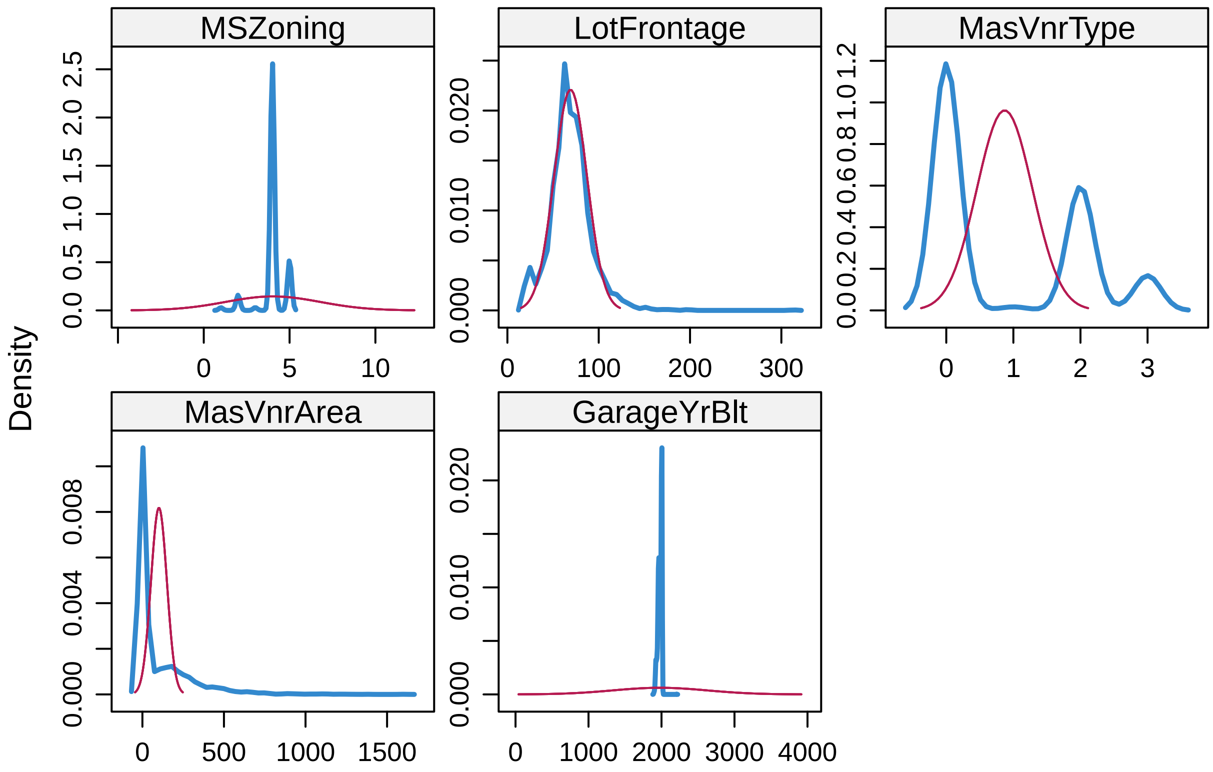
The categorical features will be 1-base except features having 'N', 'No' or 'None' as value.

```
dataset <- dataset %>%
    mutate(MasVnrType = replace(MasVnrType, MasVnrType == "None", 0))

## Transform all categorical features from string to numeric.
features.string <- which(sapply(dataset, is.character))
setDT(dataset)

for(feature in features.string)
{
    set(dataset, i = NULL, j = feature, value = as.numeric(factor(dataset[[feature]])))
}

test.id <- test$Id
dataset$Id <- NULL

## Since 'None' and 'N' is now 1, we subtract the vector by 1 to get back 0.
dataset$MasVnrType <- dataset$MasVnrType - 1
dataset$CentralAir <- dataset$CentralAir - 1
```

Missing Values Imputation

All other NA values that need a more complex method than just replacing them by a constant will get a predicted value. The objective is to use the other features to predict a value that will replace the NA value. Features enumerated in the code below will use the mean.

```
imputation.start <- mice(dataset, maxit = 0, print = FALSE)
method <- imputation.start$method
predictors <- imputation.start$predictorMatrix

## Exclude from prediction since these features will not help.
predictors[, c("SalePrice")] <- 0

imputed <- mice(dataset,
                method = "mean",
                predictorMatrix = predictors,
                m = 5,
                print = FALSE)

dataset <- complete(imputed, 1)

densityplot(imputed)
```



Feature Scaling

Some features do not have the right scale. For example, the overall quality is rate from 1 to 10, but the other quality features have been transformed from 0 to 5. If Q represents all quality features except the overall quality, then the scaling function will be $f(Q) = 2Q$ where $Q \in \{0, 1, 2, 3, 4, 5\}$. Thus, we obtain a scale from 0 to 10.

```
dataset$ExterQual <- dataset$ExterQual * 2
dataset$FireplaceQual <- dataset$FireplaceQual * 2
dataset$BsmtQual <- dataset$BsmtQual * 2
dataset$KitchenQual <- dataset$KitchenQual * 2
dataset$GarageQual <- dataset$GarageQual * 2
dataset$HeatingQualCond <- dataset$HeatingQualCond * 2
```

We apply the same scaling for the conditions except for PoolQC and HeatingQC which will use the function $f(Q) = 2.5Q$.

```
dataset$BsmtCond <- dataset$BsmtCond * 2
dataset$GarageCond <- dataset$GarageCond * 2
dataset$ExterCond <- dataset$ExterCond * 2

dataset$PoolQualCond <- dataset$PoolQualCond * 2.5
dataset$HeatingQualCond <- dataset$HeatingQualCond * 2.5
dataset$Fence <- dataset$Fence * 2.5
```

All area features are given in square feet, thus no need to convert any of them.

Skewed Features

We need to transform skewed features to ensure they follow the lognormal distribution. Thus, we will use the function $f(A) = \log(A+1)$, where $A \in \mathbb{R}^n$ is a vector representing a feature of the dataset and n the number of rows. We add 1 to avoid $\log 0$ which is not defined for real numbers.

```
                 Id           MSSubClass          LotFrontage 
        0.001340392          1.404916465          1.534509318 
            LotArea          OverallQual          OverallCond 
       12.574589806          0.183681245          0.689919090 
          YearBuilt         YearRemodAdd           MasVnrArea 
       -0.609457810         -0.499315879          2.646222191 
         BsmtFinSF1           BsmtFinSF2            BsmtUnfSF 
        0.744087901          4.244208669          0.920808873 
        TotalBsmtSF       FirstFloorArea      SecondFloorArea 
        0.485894099          0.866187326          0.777064755 
       LowQualFinSF            GrLivArea         BsmtFullBath 
        8.989291070          0.834331711          0.590543111 
       BsmtHalfBath             FullBath             HalfBath 
        4.124711657          0.017675354          0.683517982 
       BedroomAbvGr         KitchenAbvGr         TotRmsAbvGrd 
        0.214845115          4.476748271          0.660734867 
         Fireplaces          GarageYrBlt           GarageCars 
        0.632025598         -0.645115828         -0.343121438 
         GarageArea           WoodDeckSF          OpenPorchSF 
        0.132854112          1.549672002          2.337434927 
      EnclosedPorch ThreeSeasonPorchArea          ScreenPorch 
        3.081275084         10.279261797          4.111399891 
           PoolArea              MiscVal               MoSold 
       17.504555975         24.418175108          0.217658959 
             YrSold            SalePrice 
        0.093117776          1.564345548 
```

```
 [1] "LotFrontage"          "LotArea"              "MasVnrArea"          
 [4] "BsmtUnfSF"            "LowQualFinSF"         "GrLivArea"           
 [7] "BsmtHalfBath"         "KitchenAbvGr"         "WoodDeckSF"          
[10] "OpenPorchSF"          "EnclosedPorch"        "ThreeSeasonPorchArea"
[13] "ScreenPorch"          "PoolArea"             "MiscVal"
```

Let's apply the formula to the remaining features.

```
indices <- which(colnames(dataset) %in% skewed)
for(index in indices)
{
    dataset[[index]] <- log(dataset[[index]] + 1)
}
```

Features Construction

The objective is to add features that will be good predictors for models created in the section Models Building.

```
dataset <- dataset %>%
    mutate(YearsSinceBuilt = YrSold - YearBuilt) %>%
    mutate(YearsSinceRemodeled = YrSold - YearRemodAdd) %>%
    #mutate(TotalFloorsArea = FirstFloorArea + SecondFloorArea) %>%
```

```
    mutate(TotalBaths = FullBath + HalfBath + BsmtFullBath + BsmtHalfBath) %>%
    mutate(TotalArea = TotalBsmtSF + GrLivArea)
```

Noisy Features

In this section, we remove features that add noise to the predictions. We use 3 models in the section Models Building which gives the importance of features. The method used to eliminate noisy features is to look at the intersection of the less important features after applying the 3 models.

Models Building

In this section, we train different models and give predictions on the sale price of each house. We will use the extreme gradient boosting trees, the random forest and LASSO algorithms to build models.

Those algorithms need 2 inputs : the dataset as a matrix and the real sale prices from the train set. Since we had many NA and None values which have been replaced by 0, then it should be more efficient to use a sparse matrix to represent the dataset.

```
Dataset contains 34075 zeros which is 14.25553 % of the dataset.
```

Extreme Gradient Boosted Regression Trees

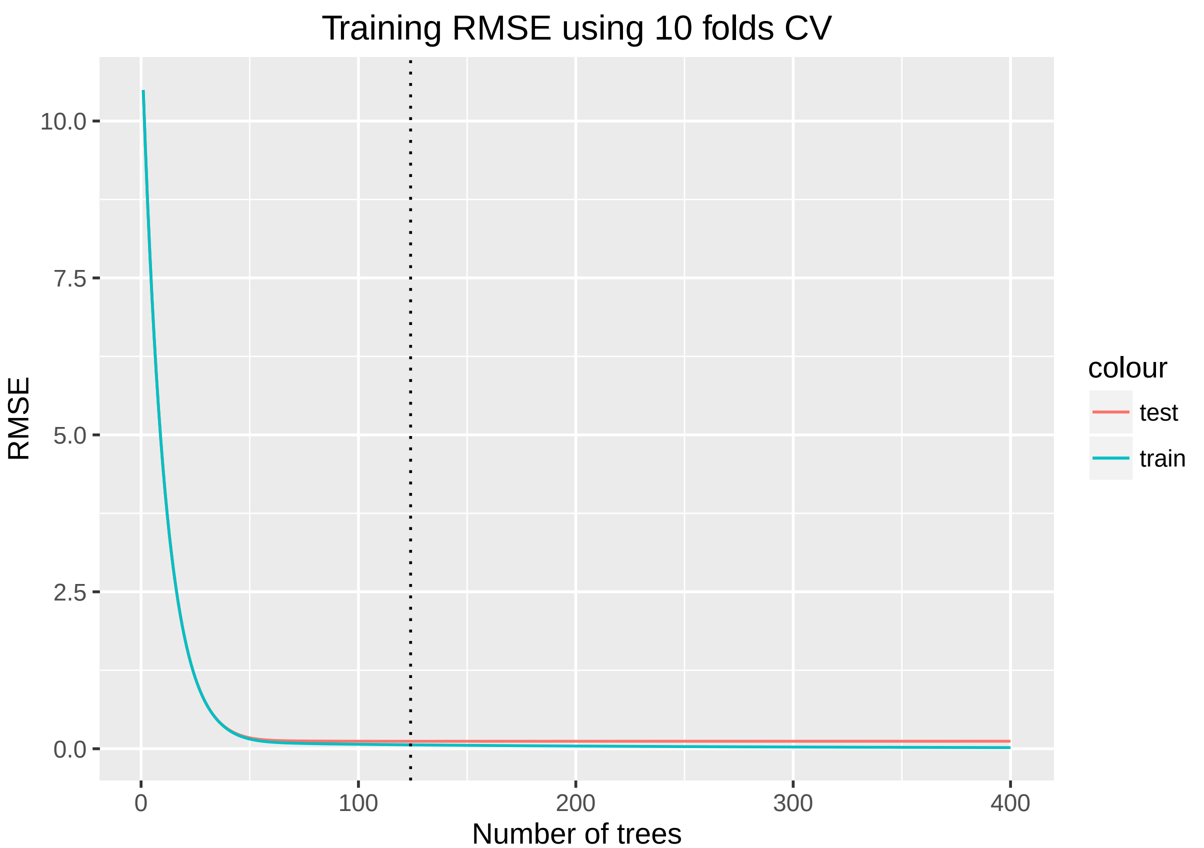
We proceed to a 10-fold cross-validation to get the optimal number of trees and the RMSE score which is the metric used for the accuracy of our model. We use randomly subsamples of the training set. The training set will be split in 10 samples where each sample has 145 observations (activities).

For each tree, we will have the average of 10 error estimates to obtain a more robust estimate of the true prediction error. This is done for all trees and we get the optimal number of trees to use for the test set.

We also display 2 curves indicating the test and train RMSE mean progression. The vertical dotted line is the optimal number of trees. This plot shows if the model overfits or underfits.

```
param <- list(objective        = "reg:linear",
              eta              = 0.09,
              subsample        = 0.5,
              colsample_bytree = 0.5,
              min_child_weight = 3,
              max_depth        = 5)

cv.nfolds <- 10
cv.nrounds <- 400

sale.price.log <- log(sale.price + 1)
train.matrix <- xgb.DMatrix(train, label = sale.price.log)
model.cv <- xgb.cv(data     = train.matrix,
                   nfold    = cv.nfolds,
                   param    = param,
                   nrounds  = cv.nrounds,
                   verbose  = 0)

model.cv$names <- as.integer(rownames(model.cv))
best <- model.cv[model.cv$test.rmse.mean == min(model.cv$test.rmse.mean), ]
```

```r
cv.plot.title <- paste("Training RMSE using", cv.nfolds, "folds CV")

print(ggplot(model.cv, aes(x = names)) +
          geom_line(aes(y = test.rmse.mean, colour = "test")) +
          geom_line(aes(y = train.rmse.mean, colour = "train")) +
          geom_vline(xintercept = best$names, linetype="dotted") +
          ggtitle(cv.plot.title) +
          xlab("Number of trees") +
          ylab("RMSE"))
```




```r
print(model.cv)
```

```
     train.rmse.mean train.rmse.std test.rmse.mean test.rmse.std names
  1:       10.493925       0.002615      10.493818      0.023304     1
  2:        9.552131       0.003115       9.552021      0.022853     2
  3:        8.695354       0.002600       8.695241      0.023449     3
  4:        7.915945       0.002699       7.915830      0.023382     4
  5:        7.206408       0.002257       7.206289      0.023713     5
 ---
396:        0.018273       0.000340       0.119495      0.010692   396
397:        0.018190       0.000332       0.119475      0.010673   397
398:        0.018108       0.000322       0.119462      0.010660   398
399:        0.018012       0.000312       0.119445      0.010654   399
400:        0.017937       0.000321       0.119460      0.010664   400
```

```r
cat("\nOptimal testing set RMSE score:", best$test.rmse.mean)
```

```
Optimal testing set RMSE score: 0.118219
```

```r
cat("\nAssociated training set RMSE score:", best$train.rmse.mean)
```

```
Associated training set RMSE score: 0.062051
```

```r
cat("\nInterval testing set RMSE score: [", best$test.rmse.mean - best$test.rmse.std, ",", best$test.rm
```

```
Interval testing set RMSE score: [ 0.10711 , 0.129328 ].
```

```r
cat("\nDifference between optimal training and testing sets RMSE:", abs(best$train.rmse.mean - best$tes
```

```
Difference between optimal training and testing sets RMSE: 0.056168
```

```r
cat("\nOptimal number of trees:", best$names)
```

```
Optimal number of trees: 124
```

Using the optimal number of trees given by the cross-validation, we can build the model using the test set as input.

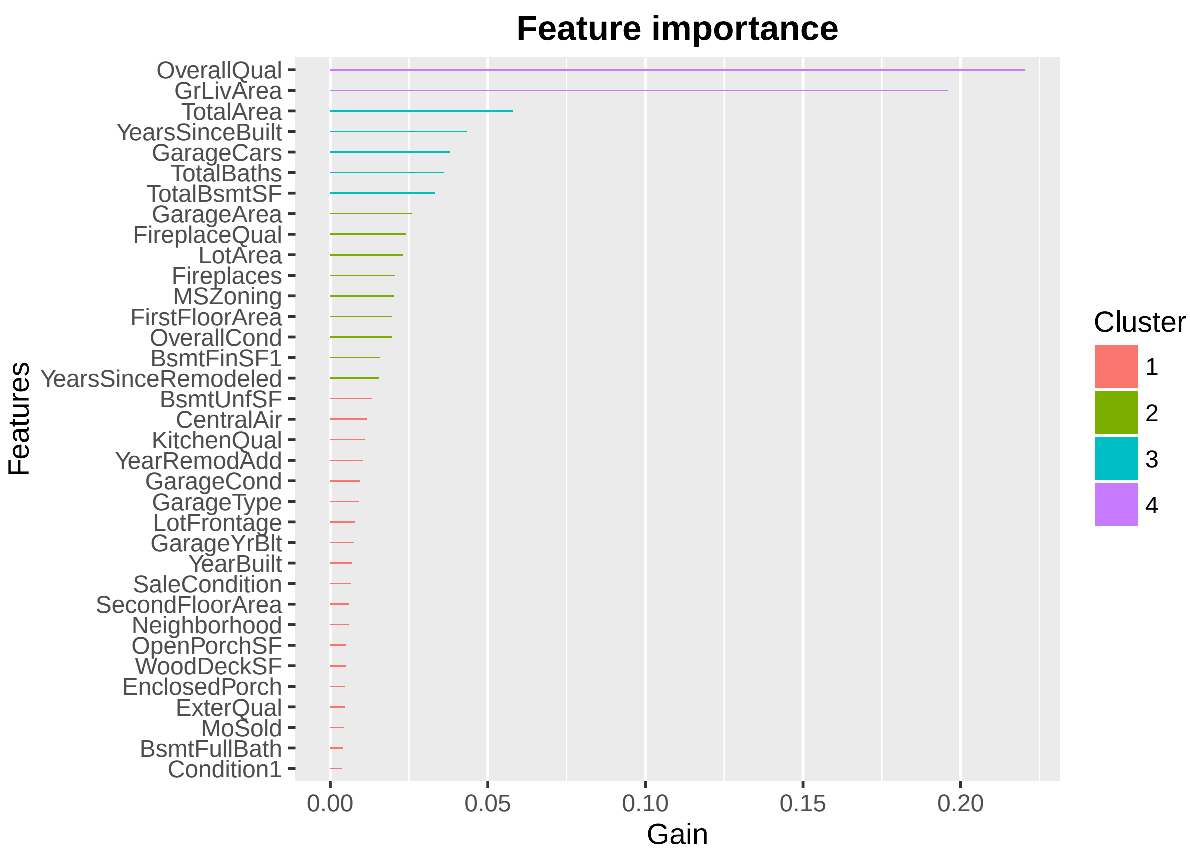
```r
nrounds <- as.integer(best$names)

model <- xgboost(param = param,
                 train.matrix,
                 nrounds = nrounds,
                 verbose = 0)

test.matrix <- xgb.DMatrix(test)

xgb.prediction.test <- exp(predict(model, test.matrix)) - 1
prediction.train <- predict(model, train.matrix)

# Check which features are the most important.
names <- dimnames(train)[[2]]
importance.matrix <- xgb.importance(names, model = model)
print(importance.matrix)
```

```
              Feature         Gain        Cover    Frequence
 1:       OverallQual 0.2203613421 0.0514370041 0.0308441558
 2:         GrLivArea 0.1959500056 0.0888614309 0.0616883117
 3:         TotalArea 0.0578250362 0.0127742691 0.0129870130
 4:    YearsSinceBuilt 0.0431669418 0.0279819892 0.0265151515
 5:        GarageCars 0.0377959717 0.0055224770 0.0043290043
 6:        TotalBaths 0.0360195130 0.0183426094 0.0129870130
 7:       TotalBsmtSF 0.0330772136 0.0378398940 0.0384199134
 8:        GarageArea 0.0257384185 0.0386357566 0.0459956710
 9:      FireplaceQual 0.0239732357 0.0061861455 0.0048701299
10:           LotArea 0.0231635803 0.0520494136 0.0481601732
11:        Fireplaces 0.0203211964 0.0040656438 0.0054112554
12:          MSZoning 0.0201701617 0.0164406327 0.0119047619
13:     FirstFloorArea 0.0195719757 0.0293848657 0.0308441558
14:       OverallCond 0.0195392179 0.0515557090 0.0297619048
15:        BsmtFinSF1 0.0156498468 0.0328327043 0.0319264069
```

16:	YearsSinceRemodeled	0.0153942626	0.0239918849	0.0324675325
17:	BsmtUnfSF	0.0130767635	0.0499316098	0.0514069264
18:	CentralAir	0.0115864855	0.0029649254	0.0043290043
19:	KitchenQual	0.0108028599	0.0091753515	0.0048701299
20:	YearRemodAdd	0.0101870630	0.0175764230	0.0205627706
21:	GarageCond	0.0094707099	0.0037338096	0.0021645022
22:	GarageType	0.0089148613	0.0051447795	0.0043290043
23:	LotFrontage	0.0078197628	0.0167427907	0.0297619048
24:	GarageYrBlt	0.0075273580	0.0234091516	0.0248917749
25:	YearBuilt	0.0066900001	0.0125935139	0.0189393939
26:	SaleCondition	0.0066769350	0.0238974605	0.0146103896
27:	SecondFloorArea	0.0060253681	0.0214990814	0.0243506494
28:	Neighborhood	0.0059936384	0.0280062698	0.0270562771
29:	OpenPorchSF	0.0048997002	0.0201798380	0.0216450216
30:	WoodDeckSF	0.0048272152	0.0172958478	0.0194805195
31:	EnclosedPorch	0.0045502689	0.0176519625	0.0162337662
32:	ExterQual	0.0045447331	0.0083606040	0.0059523810
33:	MoSold	0.0041534351	0.0106079041	0.0194805195
34:	BsmtFullBath	0.0041066392	0.0079640216	0.0075757576
35:	Condition1	0.0037018899	0.0134190527	0.0097402597
36:	Functional	0.0035173404	0.0146600588	0.0081168831
37:	Exterior1st	0.0033746611	0.0155665328	0.0167748918
38:	ScreenPorch	0.0028655049	0.0192248029	0.0113636364
39:	MasVnrArea	0.0028627026	0.0085548484	0.0194805195
40:	BsmtExposure	0.0027836011	0.0043678018	0.0097402597
41:	KitchenAbvGr	0.0022778461	0.0042949602	0.0037878788
42:	BldgType	0.0020682464	0.0026762566	0.0037878788
43:	HeatingQualCond	0.0018535249	0.0040980179	0.0102813853
44:	ExterCond	0.0018407618	0.0039118670	0.0043290043
45:	BsmtFinType1	0.0017792103	0.0068120442	0.0086580087
46:	BsmtCond	0.0016851082	0.0052553910	0.0054112554
47:	Exterior2nd	0.0016505597	0.0091025098	0.0119047619
48:	YrSold	0.0016121041	0.0050260746	0.0119047619
49:	BedroomAbvGr	0.0015706509	0.0017158258	0.0059523810
50:	LotShape	0.0015149024	0.0025035949	0.0064935065
51:	MasVnrType	0.0014884715	0.0059487357	0.0075757576
52:	BsmtQual	0.0014056385	0.0025143862	0.0059523810
53:	GarageQual	0.0013256680	0.0020611492	0.0032467532
54:	LandContour	0.0012782759	0.0065179798	0.0070346320
55:	LandSlope	0.0012665297	0.0051231968	0.0037878788
56:	GarageFinish	0.0012403762	0.0041438812	0.0054112554
57:	MiscVal	0.0012012213	0.0045296722	0.0037878788
58:	Fence	0.0011657319	0.0013947829	0.0037878788
59:	TotRmsAbvGrd	0.0011145814	0.0038660037	0.0043290043
60:	HalfBath	0.0011121198	0.0034019754	0.0048701299
61:	MSSubClass	0.0010923967	0.0019802140	0.0070346320
62:	Electrical	0.0009964086	0.0003588126	0.0021645022
63:	LotConfig	0.0008394550	0.0038039534	0.0070346320
64:	Foundation	0.0007709757	0.0015836317	0.0048701299
65:	RoofStyle	0.0007421485	0.0003830932	0.0027056277
66:	SaleType	0.0007062327	0.0029190621	0.0016233766
67:	Heating	0.0006735842	0.0002940645	0.0010822511
68:	HouseStyle	0.0006075938	0.0061672606	0.0064935065
69:	Alley	0.0005841497	0.0022553937	0.0027056277

```
70:            FullBath 0.0005700661 0.0027086307 0.0032467532
71:         MiscFeature 0.0005400680 0.0002401077 0.0010822511
72:          PavedDrive 0.0005146828 0.0004478413 0.0027056277
73:        LowQualFinSF 0.0004916061 0.0060296708 0.0032467532
74:          BsmtFinSF2 0.0004199825 0.0028381269 0.0032467532
75:            PoolArea 0.0004065149 0.0042410034 0.0016233766
76:            RoofMatl 0.0003619876 0.0037526945 0.0016233766
77:        PoolQualCond 0.0002907120 0.0027464004 0.0016233766
78:        BsmtFinType2 0.0001342855 0.0000620503 0.0005411255
79:        BsmtHalfBath 0.0001282038 0.0018588113 0.0010822511
             Feature         Gain        Cover    Frequence
```

```
# Display the features importance.
print(xgb.plot.importance(importance.matrix[1:35]))
```




```
rmse <- printRMSEInformation(prediction.train, sale.price)
```

```
RMSE = 0.06310109
```

We can see that the model overfits. Indeed, the RMSE by the cross-validation for the test set is 0.118219 since the RMSE for the train set is 0.0631011.

Random Forest

```
# rf.model <- randomForest(log(SalePrice + 1) ~ .,
#                          data = train.original,
#                          importance = TRUE,
```

```
#                           proximity = TRUE,
#                           ntree = 130,
#                           do.trace = 5)
#
# plot(rf.model, ylim = c(0, 1))
# print(rf.model)
# varImpPlot(rf.model)
# importance(rf.model)
#
# # Reduce the x-axis labels font by 0.5. Rotate 90° the x-axis labels.
# barplot(sort(rf.model$importance, dec = TRUE),
#         type = "h",
#         main = "Features in function of their Gain",
#         xlab = "Features",
#         ylab = "Gain",
#         las  = 2,
#         cex.names = 0.7)
#
# #rf.prediction.test <- exp(predict(rf.model, test.original)) - 1
# prediction.train <- predict(rf.model, train.original)
#
# rmse <- printRMSEInformation(prediction.train, sale.price)
```

LASSO Regressions

In this section, we will proceed to a features selection of the dataset. The objective is to keep only the features that have strong predictive accuracy on the sale price. Since this is a regression problem, we will use the LASSO (L1-norm) algorithm.

The Gaussian family is the most suitable for a linear regression problem. We proceed by cross-validation using 10 folds to know which features have a coefficient of zero or different of zero.

```
## alpha = 1 for lasso only
## alpha = 0 for ridge only
## alpha = 0.5 for elastic net

## Cross-validation
sale.price.log <- log(sale.price + 1)
cv.model <- cv.glmnet(x = train,
                      y = sale.price.log,
                      alpha = 1)
lambda.coef <- coef(cv.model, s = "lambda.min")
lambda.best <- cv.model$lambda.min
print(lambda.best)
```

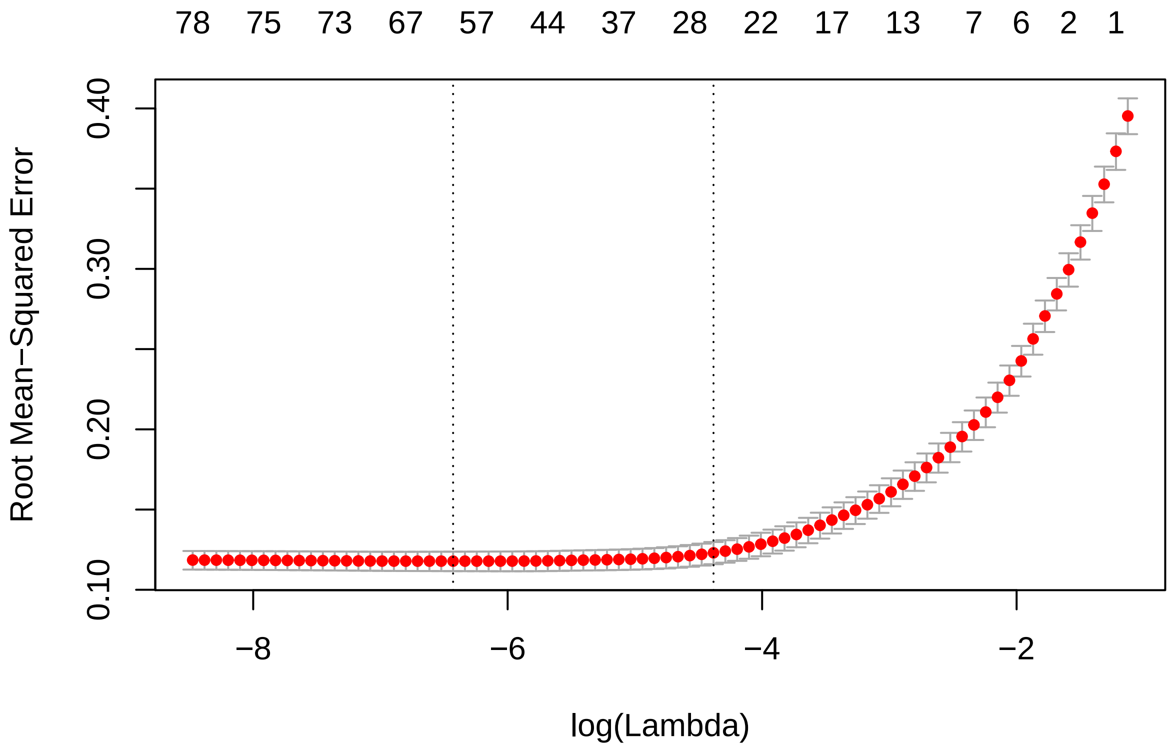
```
[1] 0.001614402
```

```
cv.model$cvm <- sqrt(cv.model$cvm)
cv.model$cvlo <- sqrt(cv.model$cvlo)
cv.model$cvup <- sqrt(cv.model$cvup)

selection <- data.frame(coef.name = dimnames(lambda.coef)[[1]],
                        coef.value = matrix(lambda.coef))
print(selection)
```

	coef.name	coef.value
1	(Intercept)	12.345069628663
2	(Intercept)	0.000000000000
3	MSSubClass	0.000000000000
4	MSZoning	-0.006098774739
5	LotFrontage	0.009371022374
6	LotArea	0.085506578113
7	Street	0.158486664313
8	Alley	0.004888809213
9	LotShape	-0.000869071661
10	LandContour	-0.007547472289
11	Utilities	-0.018223192515
12	LotConfig	-0.000818199517
13	LandSlope	0.000000000000
14	Neighborhood	-0.000005463342
15	Condition1	0.000175645816
16	Condition2	-0.002431140545
17	BldgType	0.000000000000
18	HouseStyle	0.000809251330
19	OverallQual	0.059875276274
20	OverallCond	0.042577823562
21	YearBuilt	0.000000000000
22	YearRemodAdd	0.000000000000
23	RoofStyle	0.000000000000
24	RoofMatl	0.000000000000
25	Exterior1st	-0.001335911164
26	Exterior2nd	0.000640480402
27	MasVnrType	0.000000000000
28	MasVnrArea	0.000000000000
29	ExterQual	-0.008693445969
30	ExterCond	0.004726860142
31	Foundation	0.009068334516
32	BsmtQual	-0.007654113827
33	BsmtCond	0.002952503800
34	BsmtExposure	-0.003394797060
35	BsmtFinType1	-0.002643722323
36	BsmtFinSF1	0.000067522217
37	BsmtFinType2	0.000000000000
38	BsmtFinSF2	0.000022100691
39	BsmtUnfSF	0.000000000000
40	TotalBsmtSF	0.000090614827
41	Heating	0.000000000000
42	HeatingQualCond	-0.001488884250
43	CentralAir	0.067363596724
44	Electrical	0.000000000000
45	FirstFloorArea	0.000073763051
46	SecondFloorArea	0.000076634543
47	LowQualFinSF	-0.000543135044
48	GrLivArea	0.278075642577
49	BsmtFullBath	0.010747837458
50	BsmtHalfBath	-0.004292046591
51	FullBath	0.000000000000
52	HalfBath	0.000000000000
53	BedroomAbvGr	-0.005858413902

```
54        KitchenAbvGr -0.187890437206
55         KitchenQual -0.009439924597
56        TotRmsAbvGrd  0.002095969447
57          Functional  0.017848604922
58          Fireplaces  0.023792823495
59        FireplaceQual  0.000000000000
60          GarageType  0.000976383932
61         GarageYrBlt  0.000000000000
62        GarageFinish -0.000796824177
63          GarageCars  0.027235064174
64          GarageArea  0.000051488392
65          GarageQual  0.000000000000
66          GarageCond  0.001756024360
67          PavedDrive  0.021881364413
68          WoodDeckSF  0.002597438027
69         OpenPorchSF  0.000000000000
70       EnclosedPorch  0.001006604512
71         ScreenPorch  0.007504550454
72            PoolArea  0.003578267201
73        PoolQualCond -0.000502667447
74               Fence -0.000112489981
75         MiscFeature  0.000703974135
76             MiscVal -0.004038285062
77              MoSold  0.000000000000
78              YrSold -0.002200810679
79            SaleType -0.001091833926
80       SaleCondition  0.022932351132
81     YearsSinceBuilt -0.001740065576
82 YearsSinceRemodeled -0.000766956273
83          TotalBaths  0.016689641119
84           TotalArea  0.000034130439
```

```r
plot(cv.model, ylab = "Root Mean-Squared Error")
```




```r
features <- as.vector(selection$coef.name[selection$coef.value != 0])
features <- setdiff(features, c("(Intercept)"))
print(features)
```

```
 [1] "MSZoning"           "LotFrontage"        "LotArea"
 [4] "Street"             "Alley"              "LotShape"
 [7] "LandContour"        "Utilities"          "LotConfig"
[10] "Neighborhood"       "Condition1"         "Condition2"
[13] "HouseStyle"         "OverallQual"        "OverallCond"
[16] "Exterior1st"        "Exterior2nd"        "ExterQual"
[19] "ExterCond"          "Foundation"         "BsmtQual"
[22] "BsmtCond"           "BsmtExposure"       "BsmtFinType1"
[25] "BsmtFinSF1"         "BsmtFinSF2"         "TotalBsmtSF"
[28] "HeatingQualCond"    "CentralAir"         "FirstFloorArea"
[31] "SecondFloorArea"    "LowQualFinSF"       "GrLivArea"
[34] "BsmtFullBath"       "BsmtHalfBath"       "BedroomAbvGr"
[37] "KitchenAbvGr"       "KitchenQual"        "TotRmsAbvGrd"
[40] "Functional"         "Fireplaces"         "GarageType"
[43] "GarageFinish"       "GarageCars"         "GarageArea"
[46] "GarageCond"         "PavedDrive"         "WoodDeckSF"
[49] "EnclosedPorch"      "ScreenPorch"        "PoolArea"
[52] "PoolQualCond"       "Fence"              "MiscFeature"
[55] "MiscVal"            "YrSold"             "SaleType"
[58] "SaleCondition"      "YearsSinceBuilt"    "YearsSinceRemodeled"
[61] "TotalBaths"         "TotalArea"
```

```r
## Create the model and get predictions on test and train sets.
model <- glmnet(train,
                sale.price.log,
                alpha = 1,
                lambda = 0.001)#lambda.best)
```

```r
varImp(model, lambda = lambda.best)
```

```
          Overall
1  13.253841379165
2   0.000000000000
3   0.000025515155
4   0.007326640501
5   0.011742816505
6   0.086889181646
7   0.170013715008
8   0.008461681528
9   0.000826413548
10  0.008281563063
11  0.046202835870
12  0.001008732197
13  0.000000000000
14  0.000150977326
15  0.000638163555
16  0.005820264156
17  0.000000000000
18  0.000884223856
19  0.058713336539
20  0.043739709766
21  0.001326747809
22  0.000724956535
23  0.000372471117
24  0.000000000000
25  0.002664276048
26  0.001805587718
27  0.000074086127
28  0.000000000000
29  0.008539639906
30  0.005123614839
31  0.009870459998
32  0.007964137938
33  0.003147005284
34  0.003423207548
35  0.003094215311
36  0.000066811934
37  0.001658112177
38  0.000032600290
39  0.000000000000
40  0.000118093518
41  0.000000000000
42  0.001485862786
43  0.069608154426
44  0.000000000000
45  0.000097511219
46  0.000098412191
47  0.000386659425
48  0.247952249800
49  0.013307926750
50  0.003468260833
```

```
51  0.001928060893
52  0.000000000000
53  0.008048329633
54  0.196000382814
55  0.009353102254
56  0.003322032885
57  0.018353424724
58  0.023372073541
59  0.000000000000
60  0.001779387269
61  0.000000000000
62  0.002404456388
63  0.026527486331
64  0.000049268338
65  0.000000000000
66  0.002086417226
67  0.022356884150
68  0.002809075173
69  0.000000000000
70  0.001798099710
71  0.008070485539
72  0.005986347166
73  0.001606063618
74  0.000387161055
75  0.000000000000
76  0.004506503243
77  0.000000000000
78  0.004617850718
79  0.001632278968
80  0.023621802743
81  0.000475838744
82  0.000002570796
83  0.014223352076
84  0.000004364393
```

```r
# make predictions
prediction.train <- as.vector(predict(model, s = lambda.best, train))
net.prediction.test <- as.vector(exp(predict(model, s = lambda.best, newx = test)) - 1)

rmse <- printRMSEInformation(prediction.train, sale.price)
```

```
RMSE = 0.1109649
```

This means that, in a linear regression represented by

$$y_j = \beta_0 + \sum_{i=1}^{n} \beta_i x_i$$

where β_i are the coefficient values, β_0 is the intercept value, x_i are the features (predictors) and y_j represents the j^{th} house, every feature having their coefficient equals to 0 is removed.

Results

We write the 'Id' associated to the predicted SalePrice in the submission file and we show first predicted sale prices.

```
prediction.test <- 0.6 * net.prediction.test + 0.4 * xgb.prediction.test

submission <- data.frame(Id = test.id, SalePrice = prediction.test)
write.csv(submission, "Submission_Combined.csv", row.names = FALSE)

head(submission, 15)
```

```
     Id SalePrice
1  1461 123633.13
2  1462 157155.67
3  1463 179946.00
4  1464 193791.76
5  1465 188333.09
6  1466 174386.54
7  1467 177775.99
8  1468 161510.22
9  1469 189493.36
10 1470 116878.36
11 1471 199721.36
12 1472  97635.21
13 1473  96113.83
14 1474 146486.63
15 1475 116861.94
```

Benchmark

Conclusion

From the previous sections and in virtue of results we got, this dataset is enough to solve the problem.