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 at https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data.

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 the entire dataset.

```r
library(data.table)
library(dplyr)
library(scales)
library(gridExtra)
library(ggplot2)
library(caret)
library(corrplot)
library(moments)
library(Matrix)
library(mice)
library(VIM)
library(randomForest)
library(xgboost)
library(glmnet)
library(microbenchmark)

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

set.seed(1234)

source("Dataset.R")

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

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)

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

In this section, 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. We then display the unique values of these categorical features. Then, we will compare the values mentioned in the code book with the values we have in the dataset.

```
## $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
##
## $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,
##
## $Exterior2nd
## [1] "AsbShng, AsphShn, Brk Cmn, BrkFace, CBlock, CmentBd, HdBoard, ImStucc, MetalSd, Other, Plywood,
##
## $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 need to harmonize the feature names to be coherent with the code book. Comparing with the code book's possible codes manually, the followings have difference:

Feature	Dataset	CodeBook
MSZoning	C (all)	C
MSZoning	NA	No corresponding value
Alley	Empty string	No corresponding value
Utilities	NA	No corresponding value
Neighborhood	NAmes	Names (should be NAmes)
BldgType	2fmCon	2FmCon
BldgType	Duplex	Duplx

Feature	Dataset	CodeBook
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'	Named 'Bedroom', but to be coherent, it should be named 'BedroomAbvGr'

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. Note that we will assume that the string 'Twnhs' corresponds to the string 'TwnhsI' in the code book.

```
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 programming language when refering to them, 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 mathematics logic to detect anomalies in this dataset. We will need to identify these anomalies and check how many of them we will find to calculate the percentage of abnormal houses 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.

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

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

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

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

print(id)
```

```
## Source: local data frame [75 x 4]
##
##       Id SecondFloorArea HouseStyle MSSubClass
##    (int)           (int)      (chr)      (int)
## 1     10               0     1.5Unf        190
## 2     16               0     1.5Unf         45
## 3     22               0     1.5Unf         45
## 4     52               0     1.5Fin         50
## 5     89               0     1.5Fin         50
## 6    126               0     1.5Fin        190
## 7    128               0     1.5Unf         45
## 8    164               0     1.5Unf         45
## 9    171               0     1.5Fin         50
## 10   264               0     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, MSSubClass)

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

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

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

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

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

print(id)
```

```
## Source: local data frame [44 x 3]
##
##       Id HouseStyle MSSubClass
```

```
##    (int)      (chr)      (int)
## 1    608     2Story         20
## 2    730     1.5Fin         30
## 3   1444     1.5Unf         30
## 4   2197     1.5Fin         30
## 5   2555     1.5Fin         40
## 6     75     2Story         50
## 7     80     2Story         50
## 8   1449     2Story         50
## 9   2792     1.5Unf         50
## 10  2881     2Story         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 4]
##
##       Id YearBuilt MSSubClass HouseStyle
##    (int)     (int)      (int)      (chr)
## 1  1333      1938         20     1Story
## 2  1783      1939         60     2Story
## 3  2127      1910         60     2.5Unf
## 4  2487      1920         60     2Story
## 5  2491      1945         20     1Story
## 6   837      1948         30     1Story
## 7  2130      1952         70     2Story
## 8  2499      1958         30     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.

```r
garage.none <- dataset %>%
    filter(is.na(GarageType))

id <- garage.none %>% filter(!is.na(GarageYrBlt)) %>% select(Id)
id <- bind_rows(id, garage.none %>% filter(!is.na(GarageFinish)) %>% select(Id))
id <- bind_rows(id, garage.none %>% filter(GarageCars != 0) %>% select(Id))
id <- bind_rows(id, garage.none %>% filter(GarageArea != 0) %>% select(Id))
id <- bind_rows(id, garage.none %>% filter(!is.na(GarageQual)) %>% select(Id))
id <- bind_rows(id, garage.none %>% filter(!is.na(GarageCond)) %>% select(Id))

print(id)
```

```
## 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.

```
basement.none <- dataset %>%
    filter(TotalBsmtSF == 0)

id <- basement.none %>% filter(BsmtUnfSF != 0) %>% select(Id)
id <- bind_rows(id, basement.none %>% filter(BsmtFinSF1 != 0) %>% select(Id))
id <- bind_rows(id, basement.none %>% filter(BsmtFinSF2 != 0) %>% select(Id))
id <- bind_rows(id, basement.none %>% filter(BsmtHalfBath != 0, !is.na(BsmtHalfBath)) %>% select(Id))
id <- bind_rows(id, basement.none %>% filter(BsmtFullBath != 0, !is.na(BsmtFullBath)) %>% select(Id))
id <- bind_rows(id, basement.none %>% filter(!is.na(BsmtQual)) %>% select(Id))
id <- bind_rows(id, basement.none %>% filter(!is.na(BsmtCond)) %>% select(Id))
id <- bind_rows(id, basement.none %>% filter(!is.na(BsmtExposure)) %>% select(Id))
id <- bind_rows(id, basement.none %>% filter(!is.na(BsmtFinType1)) %>% select(Id))
id <- bind_rows(id, basement.none %>% filter(!is.na(BsmtFinType2)) %>% select(Id))

print(id)
```

```
## 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.

```
dataset %>%
    filter(Fireplaces != 0 & is.na(FireplaceQual)) %>%
    select(Id, Fireplaces, FireplaceQual)
```

```
## 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.

```
dataset %>%
    filter(PoolArea != 0, is.na(PoolQualCond)) %>%
    select(Id, PoolArea, PoolQualCond)
```

```
##      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)), PoolQualCond
```

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.

```
dataset %>%
    filter(YearRemodAdd < YearBuilt) %>%
    select(Id, YearBuilt, YearRemodAdd, GarageYrBlt)
```

```
##      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.

```
dataset %>%
    select(Id, GarageArea, GarageCars) %>%
    filter(GarageArea == 0 & GarageCars > 0)
```

```
## 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.

```
dataset %>%
    filter(is.na(BsmtCond), !is.na(BsmtQual)) %>%
    select(Id, BsmtCond, BsmtQual)
```

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

```
dataset %>%
    filter(!is.na(BsmtCond), is.na(BsmtQual)) %>%
    select(Id, BsmtCond, BsmtQual)
```

```
##      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

```
dataset %>%
    filter(MasVnrType == "None", MasVnrArea != 0) %>%
    select(Id, MasVnrType, MasVnrArea)
```

```
##      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
```

```
dataset %>%
    filter(MasVnrType != "None", MasVnrArea == 0) %>%
    select(Id, MasVnrType, MasVnrArea)
```

```
##      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, 0

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

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 ft^2, the sale price should be part of the highest sale prices. If there are houses respecting these conditions with a sale price over 300000$ than what the regression model gives, then this may be possible, but if it is lower, than this is exceptionnel.

```
mod <- lm(formula = train$SalePrice ~ train$GrLivArea)

anomalies <- train %>%
    filter(OverallQual == 10, GrLivArea > 4000) %>%
    select(Id, GrLivArea, SalePrice)
print(anomalies)
```

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

```
price.eq <- coef(mod)["(Intercept)"] + coef(mod)["train$GrLivArea"] * anomalies$GrLivArea
prices <- data.frame(Id = anomalies$Id,
                     ApproxPrice = price.eq,
```

```
                      SalePrice = anomalies$SalePrice,
                      PriceDifference = abs(anomalies$SalePrice - price.eq))
print(prices)
```

```
##     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
```

```
ids <- prices$Id[prices$PriceDifference > 300000]

dataset <- dataset %>%
    filter(!(Id %in% ids))
```

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.

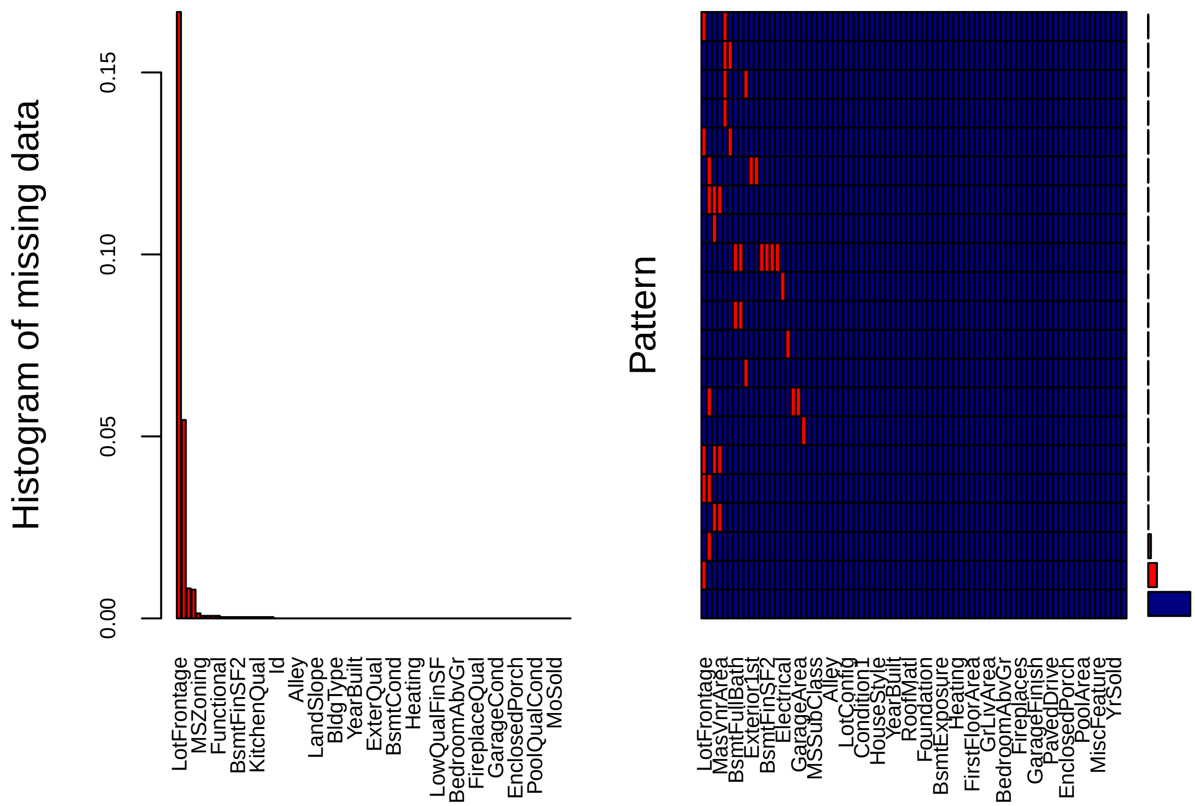
For "year" features (e.g. GarageYrBlt), if the values are NA, then we can replace them by 0 without loss of generality. A year 0 is theorically possible, but in our context, it is impossible. But, using 0 will decrease the mean and will add noise to the data since the difference between the minimum year 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 0.1666095303
##           GarageYrBlt 0.0545080562
##            MasVnrType 0.0082276311
##            MasVnrArea 0.0078848132
##              MSZoning 0.0013712719
##             Utilities 0.0006856359
##          BsmtFullBath 0.0006856359
##          BsmtHalfBath 0.0006856359
```

```
##              Functional 0.0006856359
##             Exterior1st 0.0003428180
##             Exterior2nd 0.0003428180
##              BsmtFinSF1 0.0003428180
##              BsmtFinSF2 0.0003428180
##               BsmtUnfSF 0.0003428180
##             TotalBsmtSF 0.0003428180
##              Electrical 0.0003428180
##             KitchenQual 0.0003428180
##              GarageCars 0.0003428180
##              GarageArea 0.0003428180
##                SaleType 0.0003428180
##                      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

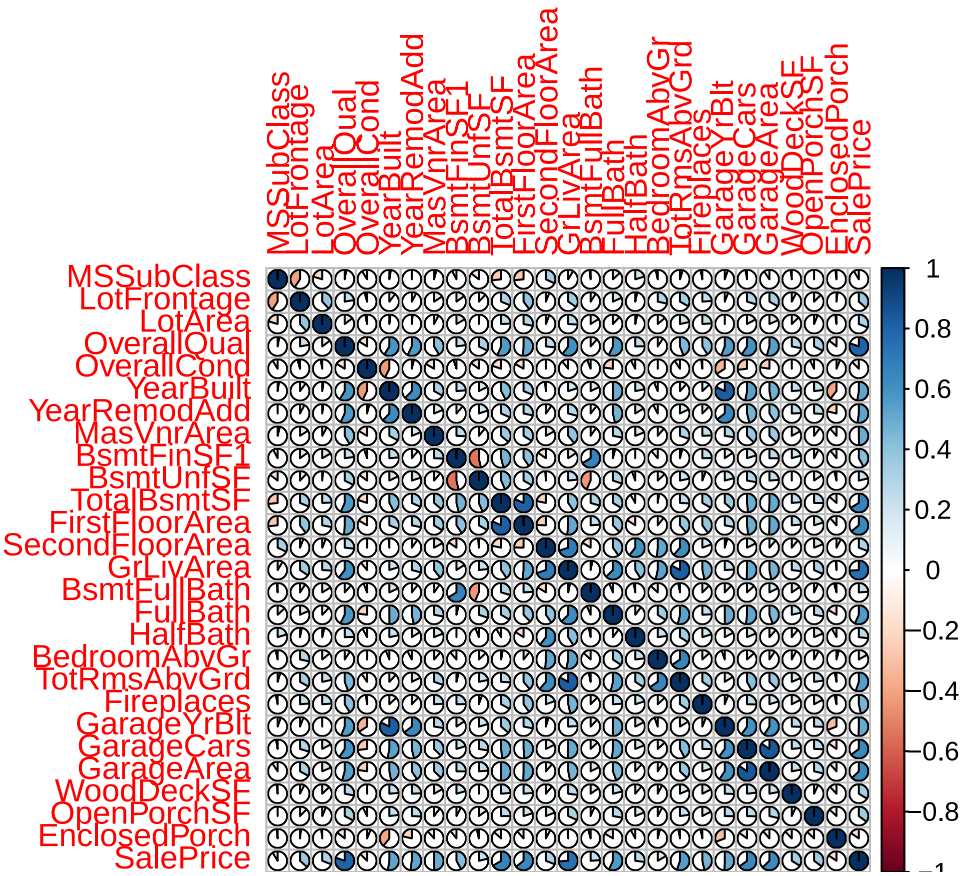
Here is the list of features with their type.

```
## Classes 'data.table' and 'data.frame':   2917 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              0.80412102
## GrLivArea                0.74046603
## TotalBsmtSF              0.66461127
## GarageCars               0.64798727
## FirstFloorArea           0.64263073
## GarageArea               0.62827868
## FullBath                 0.56872528
```

```
## TotRmsAbvGrd               0.55285444
## YearBuilt                  0.52635209
## YearRemodAdd               0.52210205
## GarageYrBlt                0.50567335
## MasVnrArea                 0.49528817
## Fireplaces                 0.46601824
## BsmtFinSF1                 0.42031701
## LotFrontage                0.36470782
## OpenPorchSF                0.35174623
## WoodDeckSF                 0.33743599
## LotArea                    0.30987091
## SecondFloorArea            0.30855321
## HalfBath                   0.26929280
## BsmtFullBath               0.23877313
## BsmtUnfSF                  0.21311035
## BedroomAbvGr               0.16687526
## MSSubClass                -0.08800998
## OverallCond               -0.12457590
## EnclosedPorch             -0.15496825
```

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.

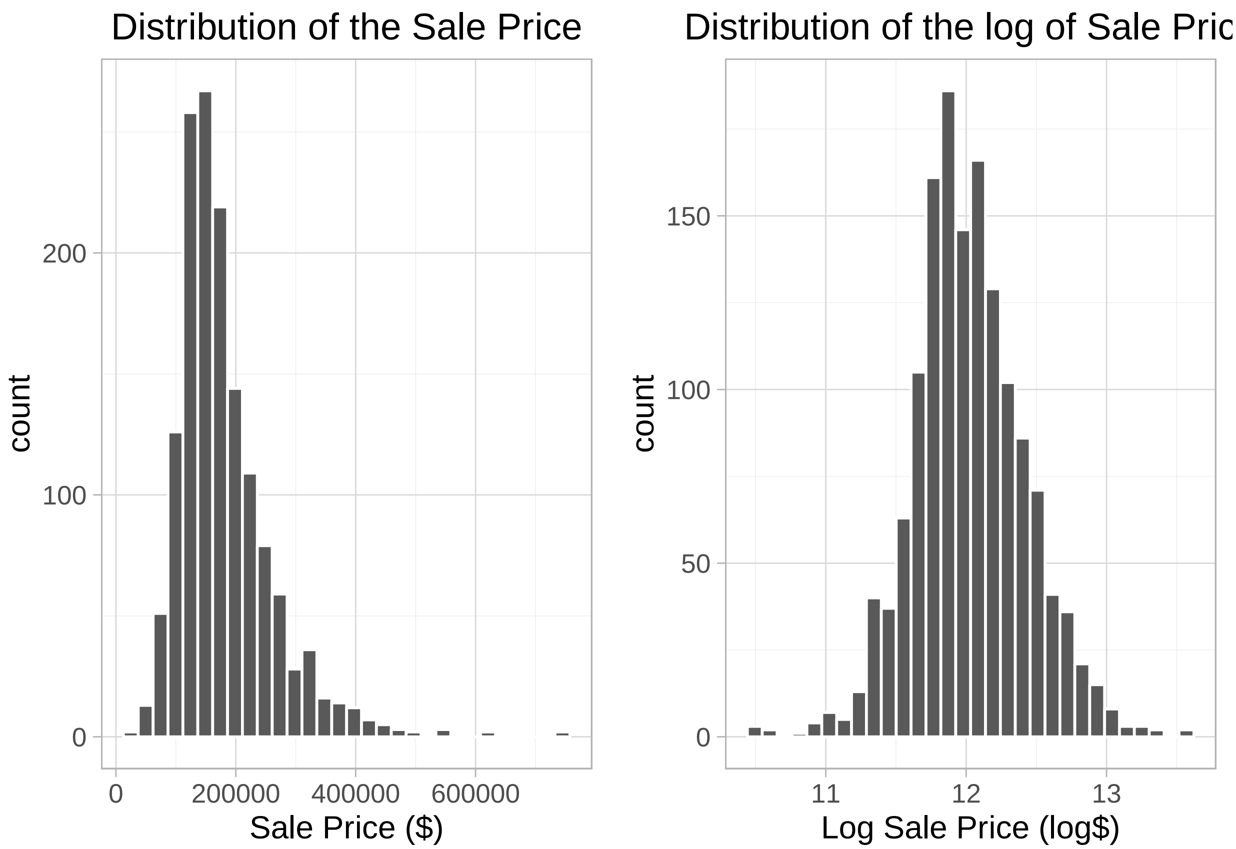
Dependant vs Independent Features

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  180900  214000  755000
```

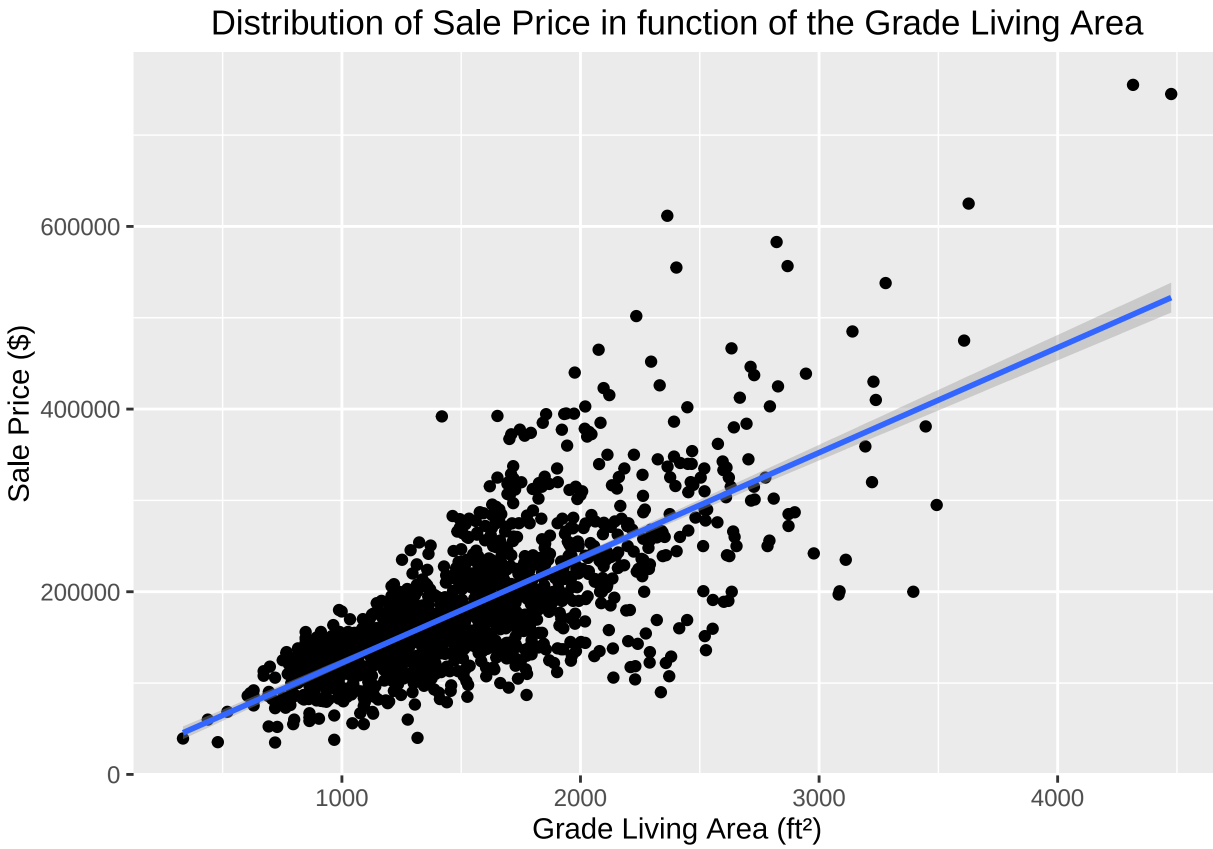
Overall Quality Rate

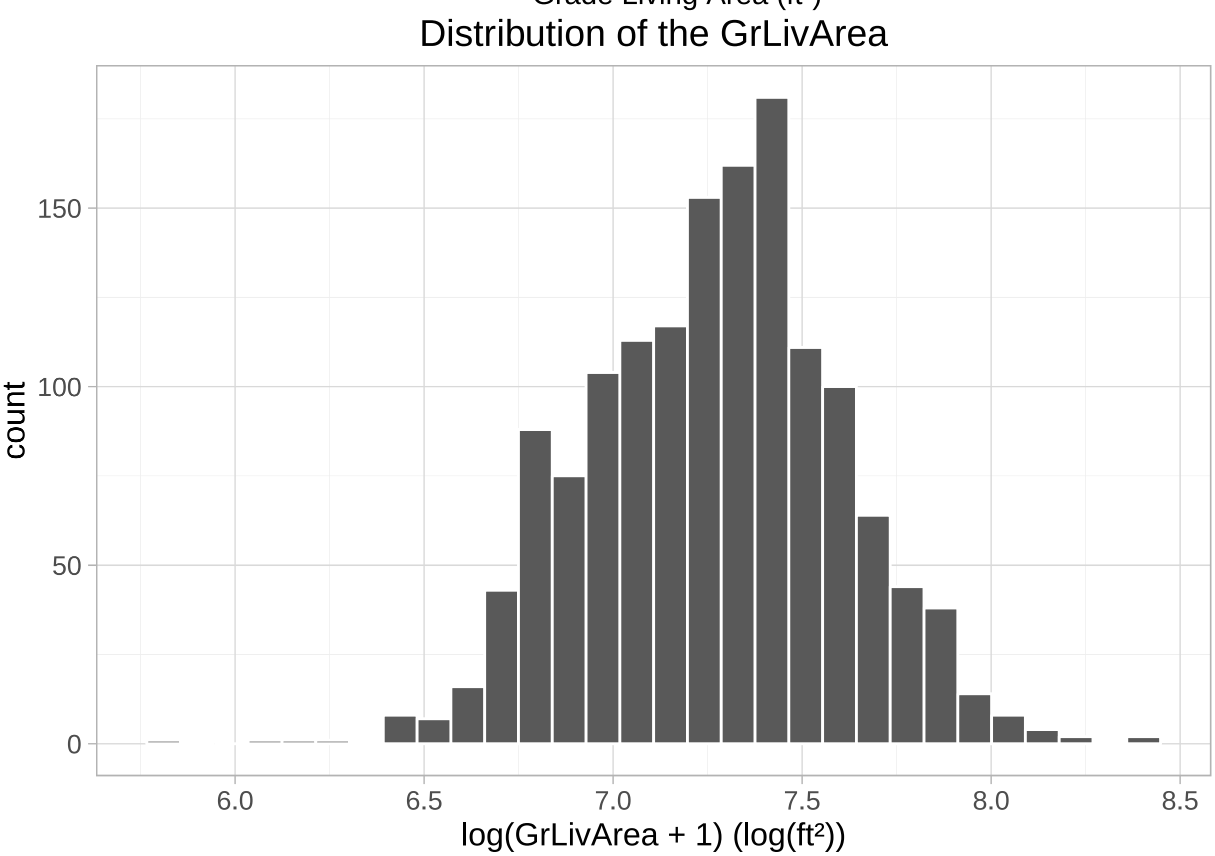
The overall quality rate is the most correlated feature to the sale price as seen previously. We look at the average sale price for each overall quality rate and try to figure out an equation that will best approximate our data.

```
## 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     471865.06
```

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.

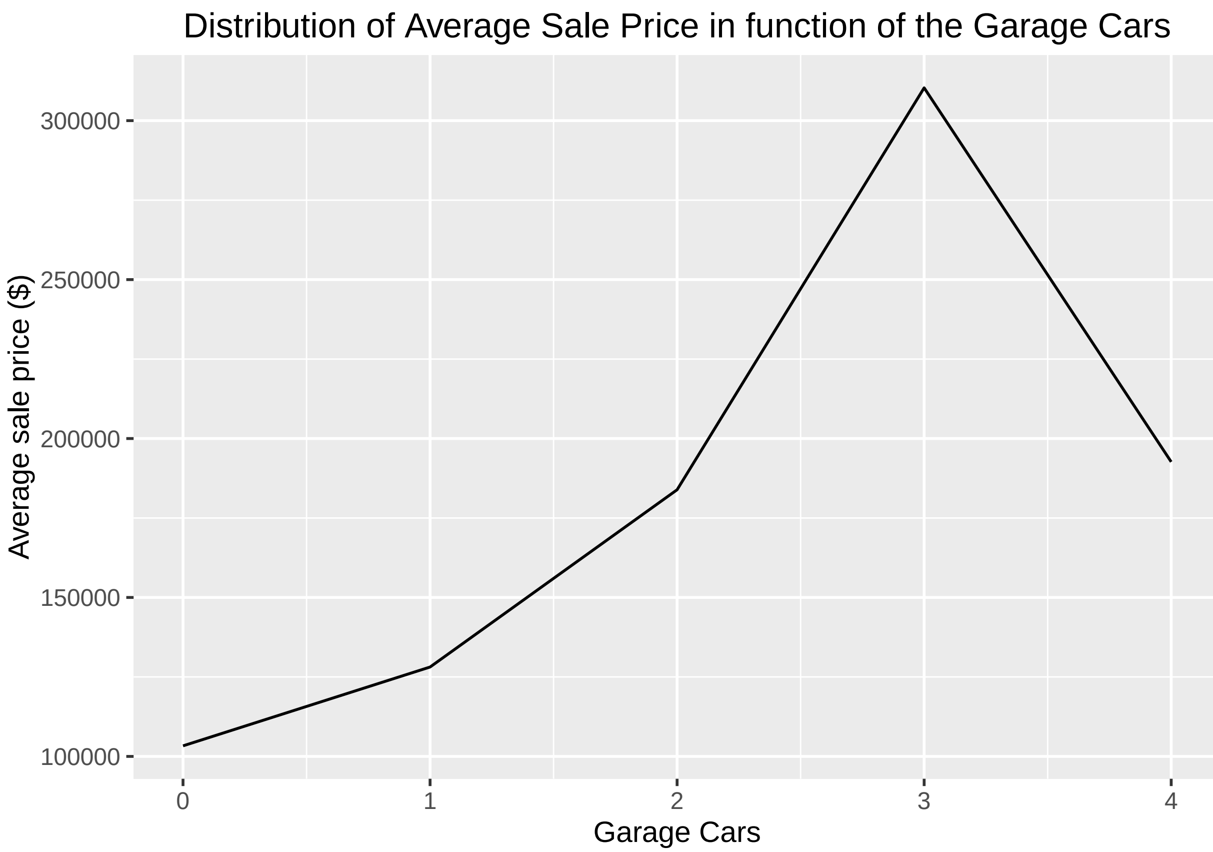
Above grade (ground) living area



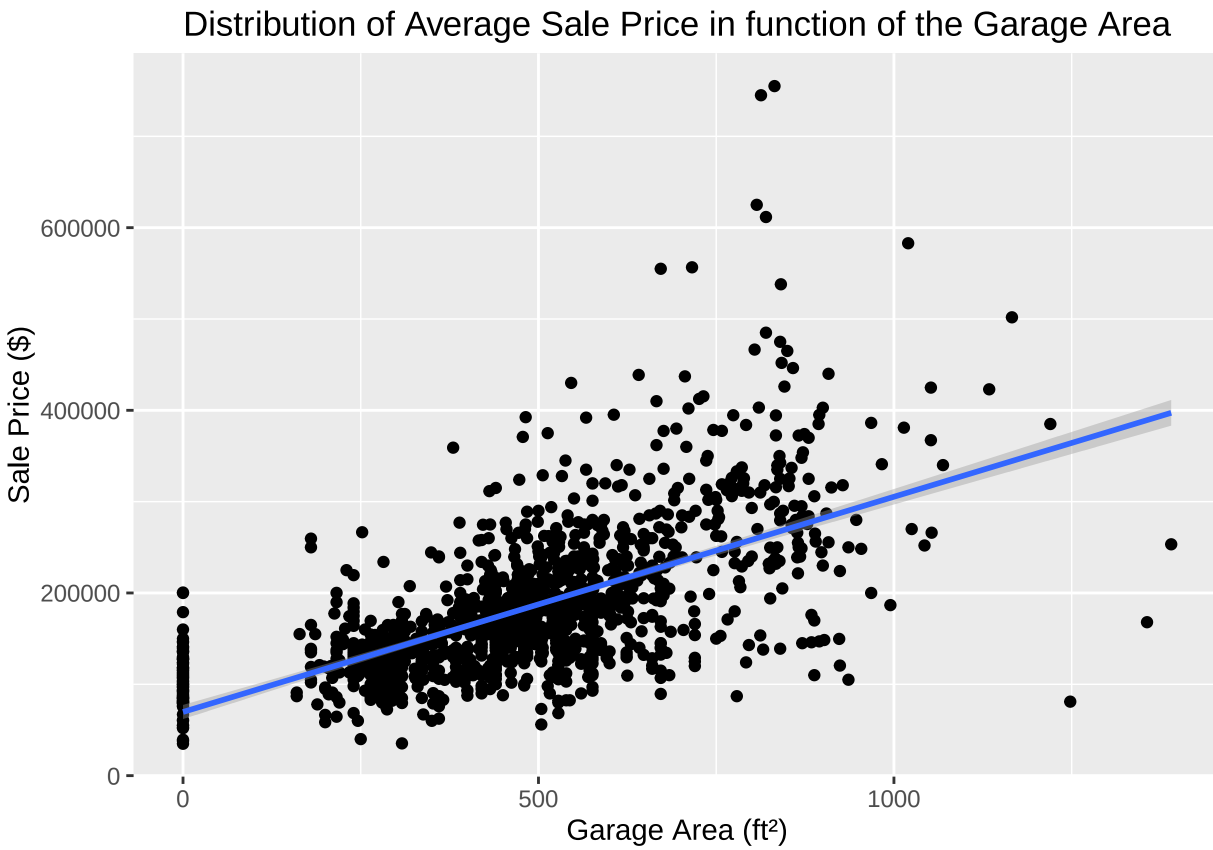


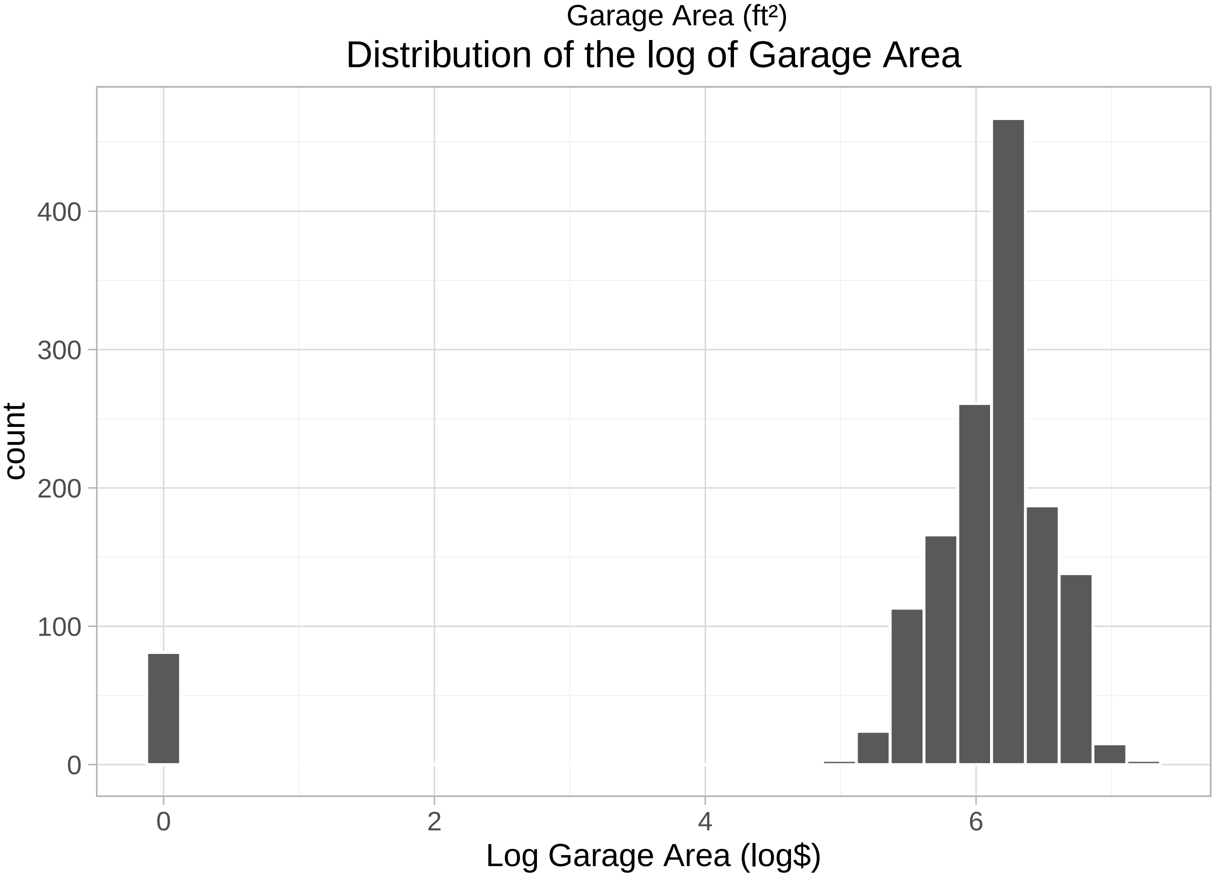
Garage Cars

```
## 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      310329.9
## 5          4      192655.8
```

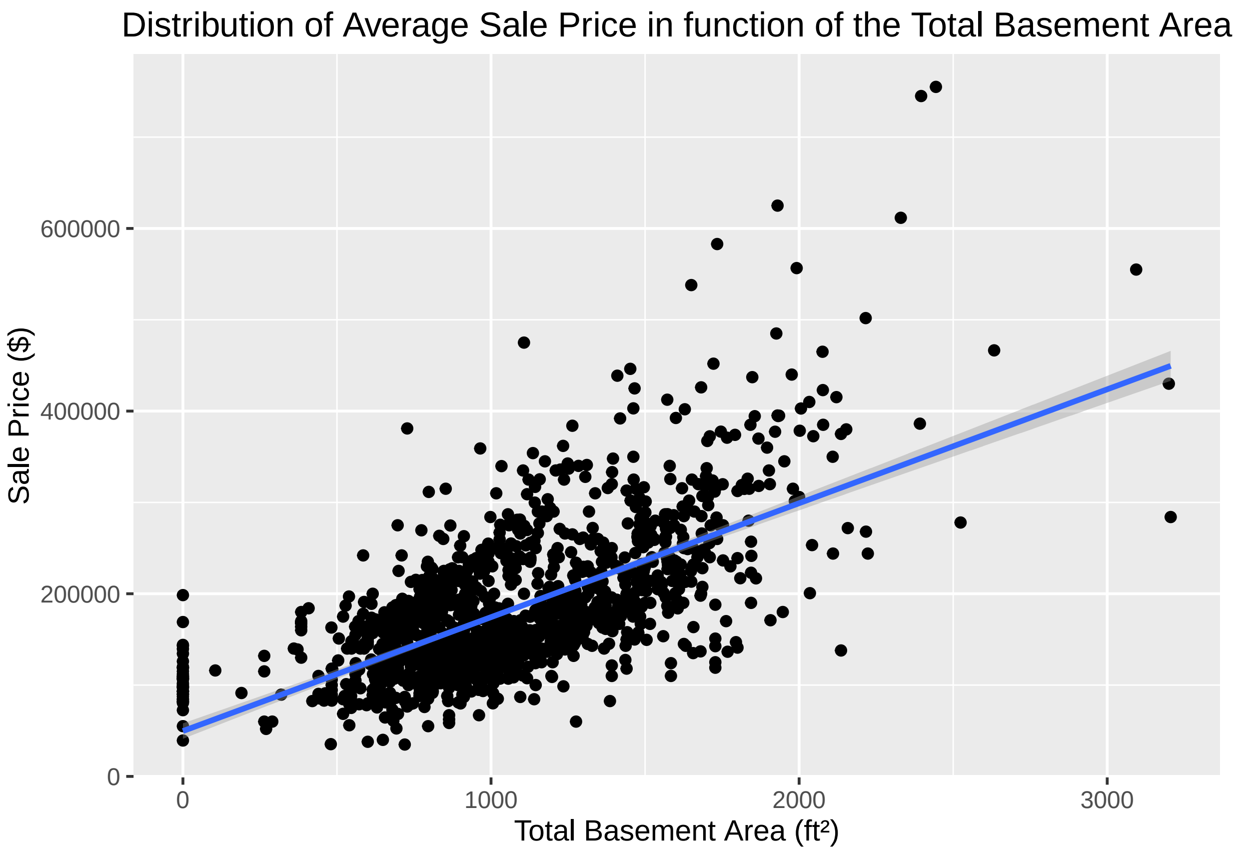


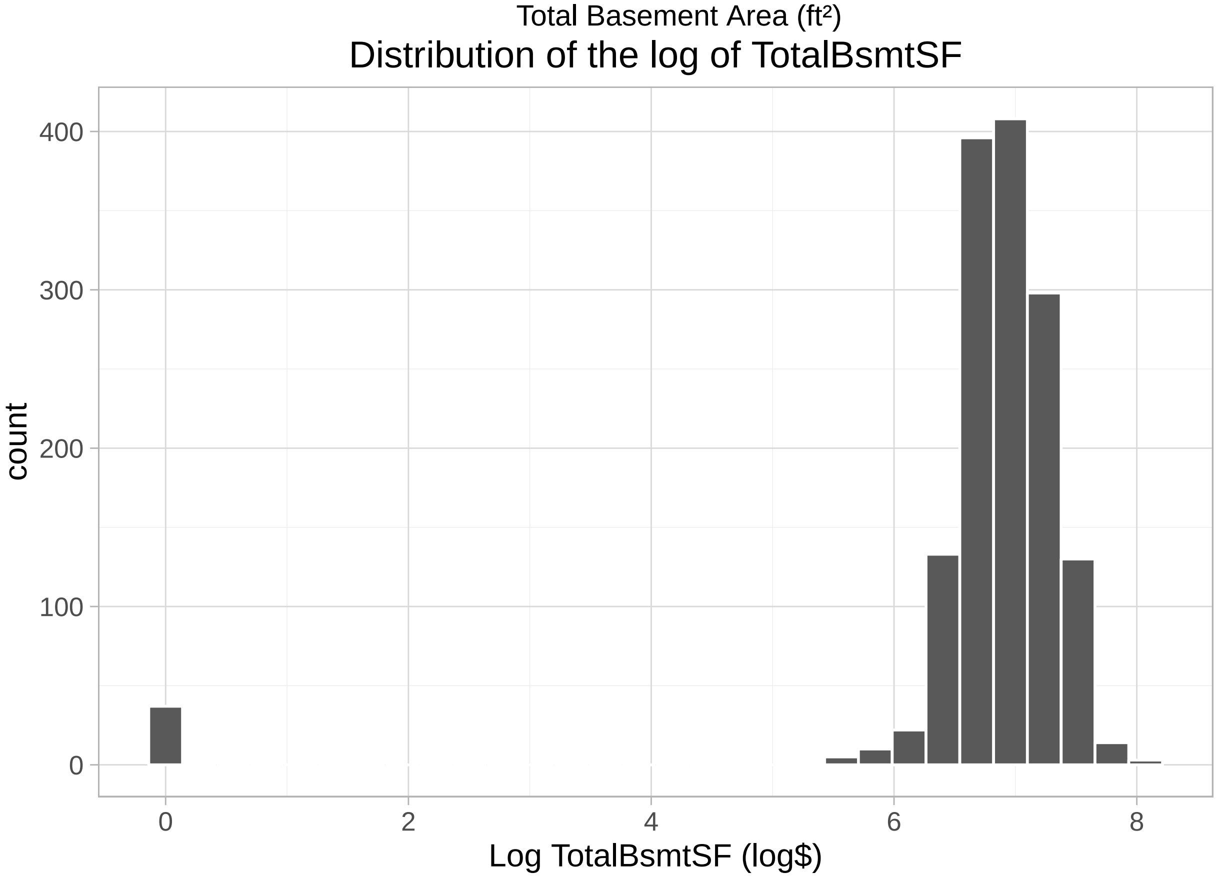
Garage Area


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

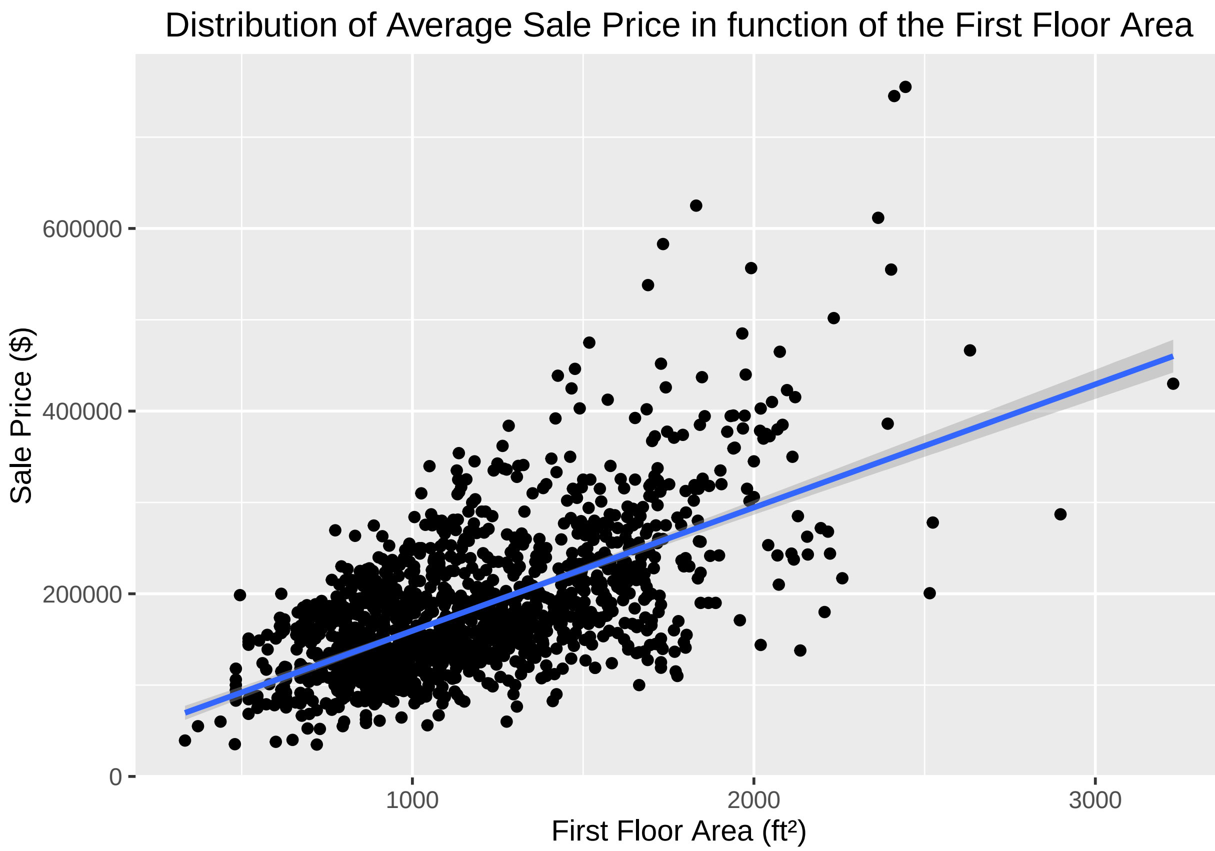

Distribution of the log of Garage Area
count
400
300
200
100
0
0
2
4
6
Log Garage Area (log$)

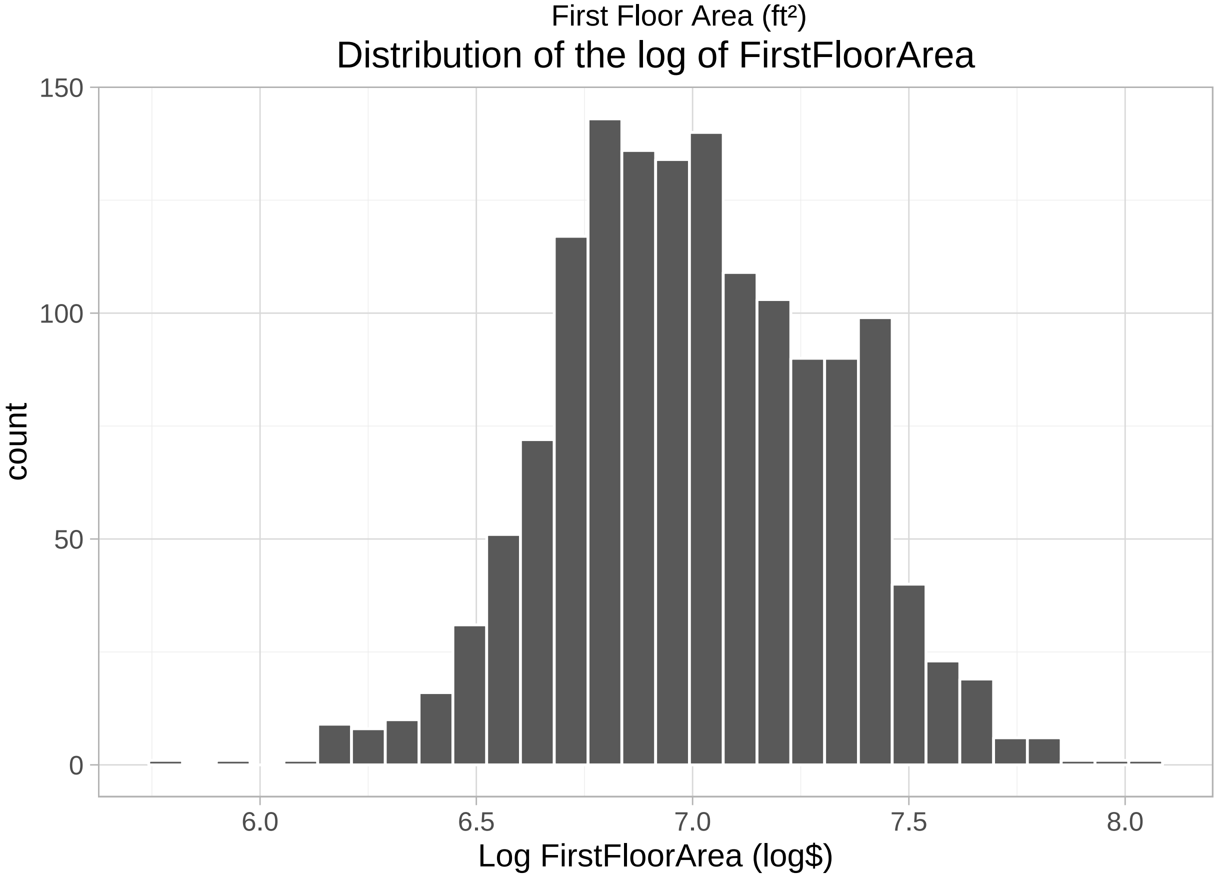
Total Basement Area





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.

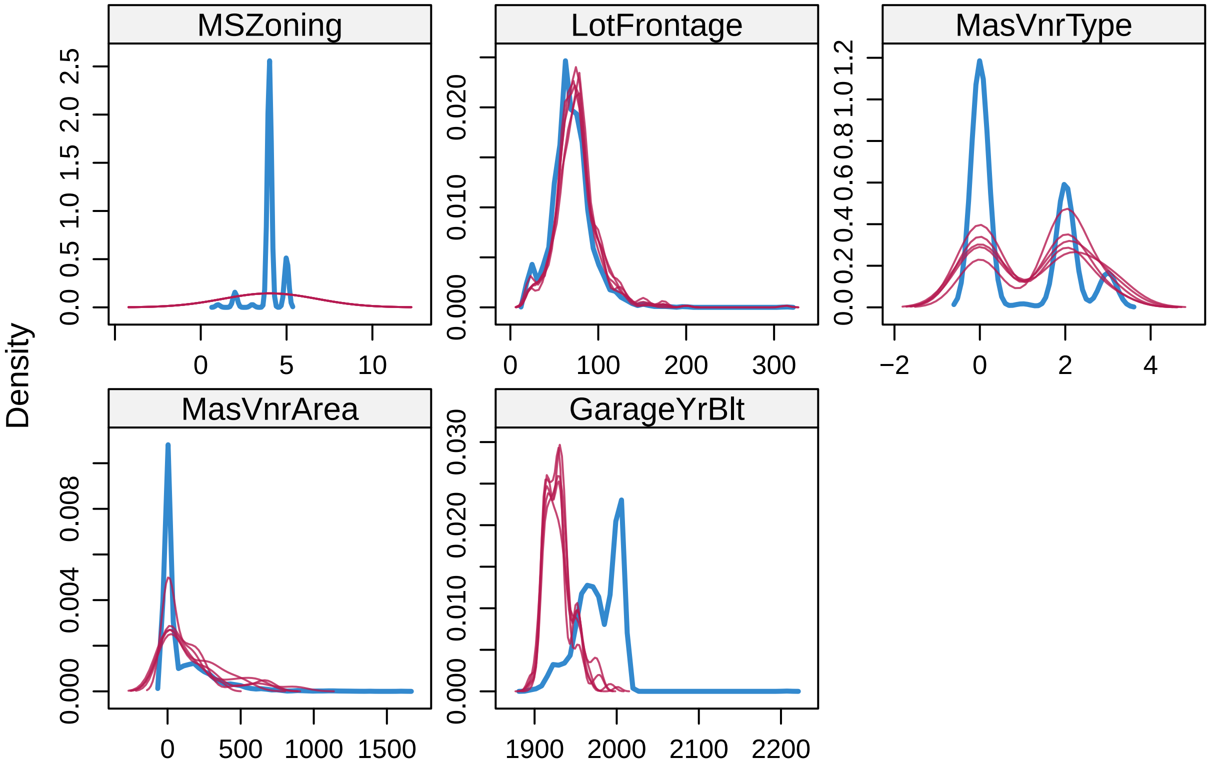
```
dataset <- dataset %>%
    mutate(MasVnrType = replace(MasVnrType, MasVnrType == "None", 0))

## 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

method[c("MSZoning", "Utilities", "Exterior1st", "Exterior2nd",
         "KitchenQual", "Functional", "Electrical", "SaleType")] <- "mean"

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

dataset <- complete(imputed, 1)
```

```r
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.

```r
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 $f(Q) = 2.5Q$.

```r
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
```

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.

```
skewed <- apply(train.numeric, 2, function(feature) skewness(feature, na.rm = TRUE))
skewed <- setdiff(names(skewed[skewed > 0.75]), c("SalePrice"))
skewed
```

```
##  [1] "MSSubClass"           "LotFrontage"          "LotArea"
##  [4] "MasVnrArea"           "BsmtFinSF1"           "BsmtFinSF2"
##  [7] "BsmtUnfSF"            "FirstFloorArea"       "SecondFloorArea"
## [10] "LowQualFinSF"         "GrLivArea"            "BsmtHalfBath"
## [13] "KitchenAbvGr"         "WoodDeckSF"           "OpenPorchSF"
## [16] "EnclosedPorch"        "ThreeSeasonPorchArea" "ScreenPorch"
## [19] "PoolArea"             "MiscVal"
```

```
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(MeanQuality = (ExterQual + BsmtQual + HeatingQualCond + KitchenQual +
    #                       FireplaceQual + GarageQual + OverallQual + PoolQualCond) / 8) %>%
    # mutate(MeanCondition = (HeatingQualCond + BsmtCond + GarageCond + PoolQualCond +
    #                         OverallCond + ExterCond) / 6) %>%

    # mutate(AgeAtSold = YrSold - YearBuilt) %>%
    # mutate(AgeRemodeled = YrSold - YearRemodAdd) %>%
    # mutate(YearsSinceRemodel = YearRemodAdd - YearBuilt) %>%

    # mutate(AboveGroundBaths = FullBath + HalfBath) %>%
    # mutate(BasementBaths = BsmtFullBath + BsmtHalfBath) %>%
    mutate(TotalBaths = FullBath + HalfBath + BsmtFullBath + BsmtHalfBath) %>%
    mutate(TotalArea = TotalBsmtSF + GrLivArea)
    #
    # mutate(HasGarage = as.integer(GarageType > 0)) %>%
    # mutate(HasBasement = as.integer(BsmtQual > 0)) %>%
    # mutate(HasFireplace = as.integer(FireplaceQual > 0)) %>%
    # mutate(HasPool = as.integer(PoolQualCond > 0))
```

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.

```
features.exclude <- c("ThreeSeasonPorchArea")
features <- setdiff(names(dataset), features.exclude)
dataset <- dataset[, colnames(dataset) %in% features]
```

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 33764 zeros which is 14.46863 % of the dataset.
```

Extreme Gradient Boosted Regression Trees

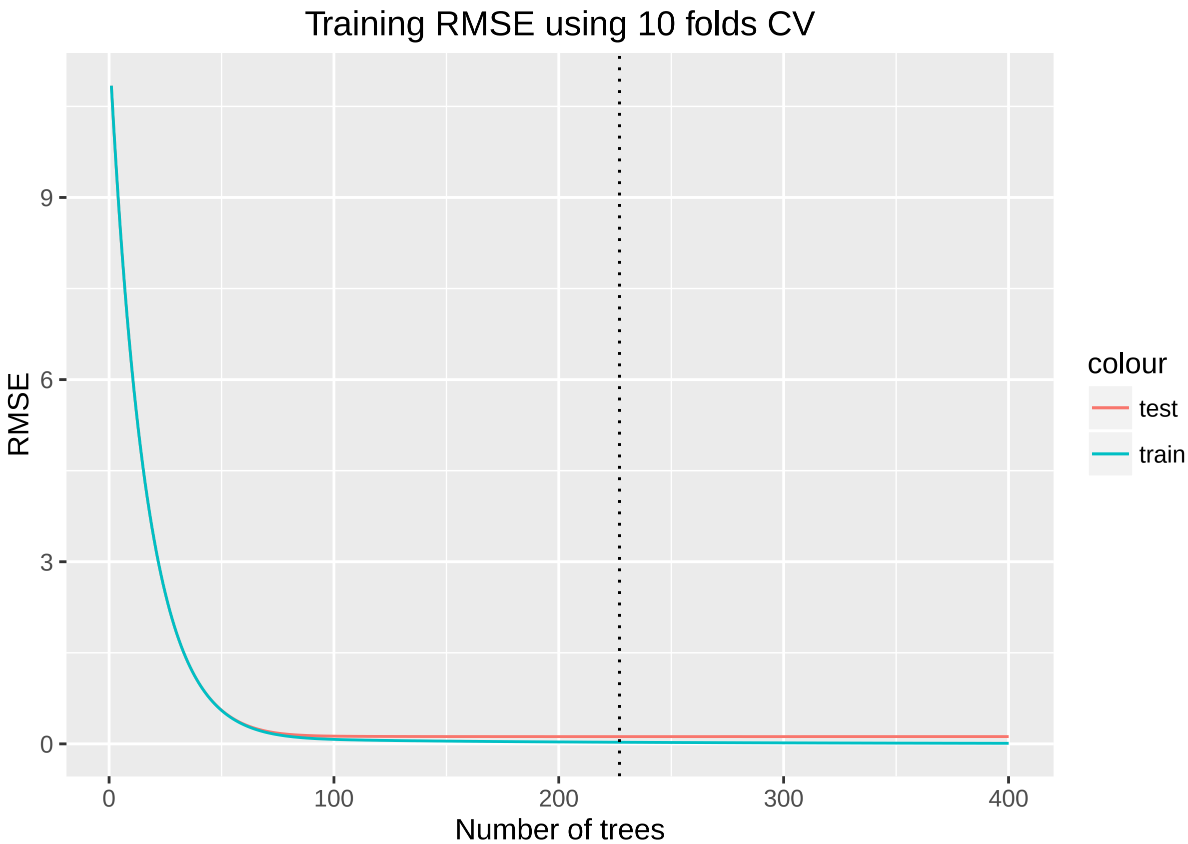
We proceed to a 10-fold cross-validation to get the optimal number of trees and the RMSE score which is the metric used for the accuracy of our model. We use randomly subsamples representing 80% of the training set. The training set will be split in 10 samples where each sample has 145 observations (activities).

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              = 0.06,
              subsample        = 0.8,
              colsample_bytree = 0.7,
              min_child_weight = 4,
              max_depth        = 7)

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), ]
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") +
```

```r
        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:       10.840557       0.004082      10.840683      0.037384     1
##   2:       10.191570       0.003949      10.191693      0.037486     2
##   3:        9.581890       0.003750       9.582009      0.037681     3
##   4:        9.008551       0.003506       9.008665      0.037933     4
##   5:        8.469430       0.003377       8.469540      0.038012     5
##  ---
## 396:        0.009195       0.000340       0.119682      0.017981   396
## 397:        0.009139       0.000341       0.119675      0.017981   397
## 398:        0.009091       0.000344       0.119680      0.017976   398
## 399:        0.009050       0.000344       0.119682      0.017978   399
## 400:        0.009007       0.000349       0.119677      0.017976   400
```

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

```
##
## Optimal testing set RMSE score: 0.119473
```

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

```
##
## Associated training set RMSE score: 0.026503
```

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

```
##
## Interval testing set RMSE score: [ 0.101555 , 0.137391 ].
```

```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: 0.09297
```

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

```
##
## Optimal number of trees: 227
```

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

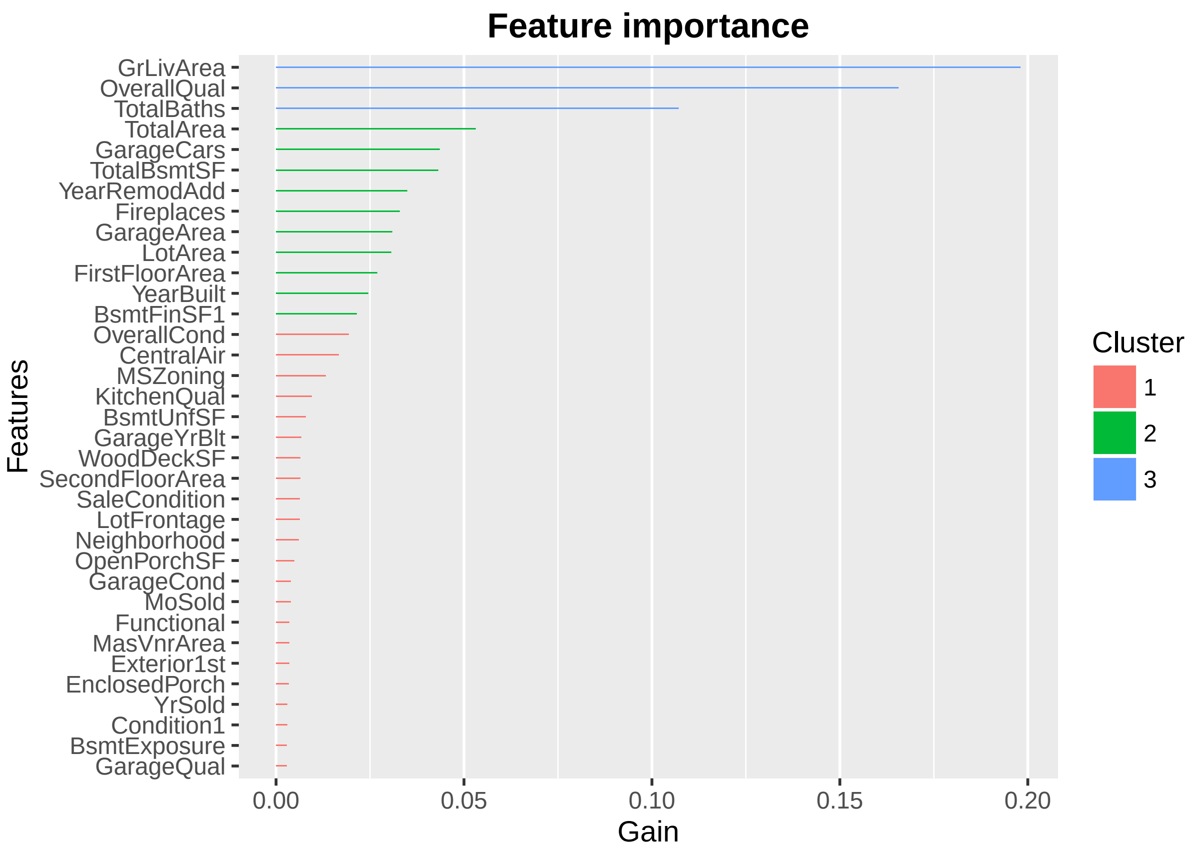
```r
nrounds <- as.integer(best$names)

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:       GrLivArea 0.198089137105 0.06661536365 0.0500129567
##  2:     OverallQual 0.165562720151 0.04749325084 0.0259134491
##  3:      TotalBaths 0.107143119992 0.02586106450 0.0117906193
##  4:       TotalArea 0.053057992528 0.01477474180 0.0139932625
##  5:      GarageCars 0.043565000669 0.00625115613 0.0057009588
##  6:     TotalBsmtSF 0.043164959102 0.03216225094 0.0400362788
##  7:    YearRemodAdd 0.034948205990 0.02225756527 0.0259134491
##  8:      Fireplaces 0.032953006446 0.01051031579 0.0094584089
##  9:      GarageArea 0.030923648942 0.04640838297 0.0506607930
## 10:         LotArea 0.030619073641 0.05319473181 0.0494946877
## 11:  FirstFloorArea 0.026926681394 0.02245571165 0.0320031096
## 12:       YearBuilt 0.024548965694 0.02932500548 0.0304483027
## 13:      BsmtFinSF1 0.021519431426 0.04088266641 0.0472920446
## 14:     OverallCond 0.019302600599 0.03963190854 0.0182689816
## 15:      CentralAir 0.016677488924 0.00651447357 0.0031096139
## 16:        MSZoning 0.013275882000 0.01487085267 0.0079036020
## 17:     KitchenQual 0.009574196113 0.00664547400 0.0054418243
## 18:       BsmtUnfSF 0.007978798107 0.05077550279 0.0505312257
## 19:     GarageYrBlt 0.006678887426 0.02818023289 0.0256543146
## 20:      WoodDeckSF 0.006407807849 0.02693079162 0.0349831563
```

```
## 21: SecondFloorArea 0.006387422448 0.03086870400 0.0305778699
## 22:   SaleCondition 0.006384817695 0.01698792492 0.0104949469
## 23:     LotFrontage 0.006297278456 0.02451748724 0.0305778699
## 24:    Neighborhood 0.006078653290 0.02567015935 0.0264317181
## 25:     OpenPorchSF 0.004862103629 0.02202123787 0.0380927701
## 26:      GarageCond 0.003961220240 0.00130144647 0.0010365380
## 27:          MoSold 0.003952403310 0.01602418307 0.0316144079
## 28:      Functional 0.003522740243 0.01499132040 0.0060896605
## 29:      MasVnrArea 0.003508689127 0.01725519213 0.0264317181
## 30:     Exterior1st 0.003485108878 0.01920308292 0.0164550402
## 31:   EnclosedPorch 0.003331596233 0.02712103847 0.0156776367
## 32:          YrSold 0.002986813383 0.00696211323 0.0180098471
## 33:      Condition1 0.002933169893 0.01771073130 0.0099766779
## 34:    BsmtExposure 0.002813180645 0.00823130331 0.0099766779
## 35:      GarageQual 0.002796404059 0.00217566038 0.0018139414
## 36:    GarageFinish 0.002771690061 0.00255944556 0.0060896605
## 37:       ExterQual 0.002764709238 0.00409919431 0.0025913449
## 38:      GarageType 0.002753494810 0.00249756596 0.0053122571
## 39:     TotRmsAbvGrd 0.002478409557 0.00580088330 0.0124384556
## 40: HeatingQualCond 0.002398941652 0.00449548707 0.0080331692
## 41:      MSSubClass 0.002278330749 0.00435987858 0.0116610521
## 42:    KitchenAbvGr 0.002158901116 0.00276285828 0.0015548069
## 43:        BsmtQual 0.002002265528 0.00296429613 0.0058305260
## 44:     ScreenPorch 0.001761334901 0.01937028950 0.0079036020
## 45:   FireplaceQual 0.001718932774 0.00310253779 0.0082923037
## 46:       ExterCond 0.001654679231 0.00408866162 0.0042757191
## 47:    BedroomAbvGr 0.001531922409 0.00499381533 0.0080331692
## 48:      BsmtFinSF2 0.001387735389 0.00662770007 0.0062192278
## 49:     Exterior2nd 0.001369861332 0.00447639655 0.0108836486
## 50:      HouseStyle 0.001099107674 0.00613990451 0.0067374968
## 51:    BsmtFinType1 0.001077202075 0.00551979193 0.0093288417
## 52:      MasVnrType 0.000977722676 0.00256076215 0.0064783623
## 53:      PavedDrive 0.000964914001 0.00392408821 0.0029800466
## 54:        LotShape 0.000898465719 0.00354491109 0.0068670640
## 55:        SaleType 0.000855792695 0.00534271095 0.0050531226
## 56:       LotConfig 0.000853687475 0.00639598072 0.0072557657
## 57:        BsmtCond 0.000780544347 0.00297614541 0.0023322104
## 58:       RoofStyle 0.000768145295 0.00283658717 0.0044052863
## 59:      Foundation 0.000674678888 0.00253113893 0.0029800466
## 60:        PoolArea 0.000606792635 0.01058733614 0.0018139414
## 61:           Fence 0.000583071339 0.00245609346 0.0040165846
## 62:           Alley 0.000547858094 0.00293203974 0.0020730759
## 63:    PoolQualCond 0.000455834331 0.00753680355 0.0034983156
## 64:         Heating 0.000430381050 0.00022645300 0.0002591345
## 65:     LandContour 0.000387020194 0.00193867468 0.0024617777
## 66:    BsmtFullBath 0.000369214425 0.00188469461 0.0038870174
## 67:        FullBath 0.000368779662 0.00085709828 0.0016843742
## 68:        BldgType 0.000360789277 0.00173592025 0.0018139414
## 69:    BsmtFinType2 0.000356164553 0.00213813765 0.0024617777
## 70:        HalfBath 0.000325265563 0.00188930266 0.0027209122
## 71:      Electrical 0.000265088371 0.00094135986 0.0032391811
## 72:       LandSlope 0.000262610709 0.00238894751 0.0019435087
## 73:    LowQualFinSF 0.000201403610 0.00317231691 0.0009069707
## 74:     MiscFeature 0.000111796429 0.00048977045 0.0002591345
```

```
## 75:         MiscVal 0.000082578942 0.00037720224 0.0005182690
## 76:        RoofMatl 0.000060529853 0.00083471630 0.0006478362
## 77:      Condition2 0.000017882626 0.00076427888 0.0002591345
## 78:    BsmtHalfBath 0.000006263116 0.00009479428 0.0001295672
##           Feature           Gain         Cover    Frequence
```

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




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

```
## RMSE = 0.02762876
```

We can see that the model overfits. Indeed, the RMSE by the cross-validation for the test set is 0.119473 since the RMSE for the train set is 0.0276288.

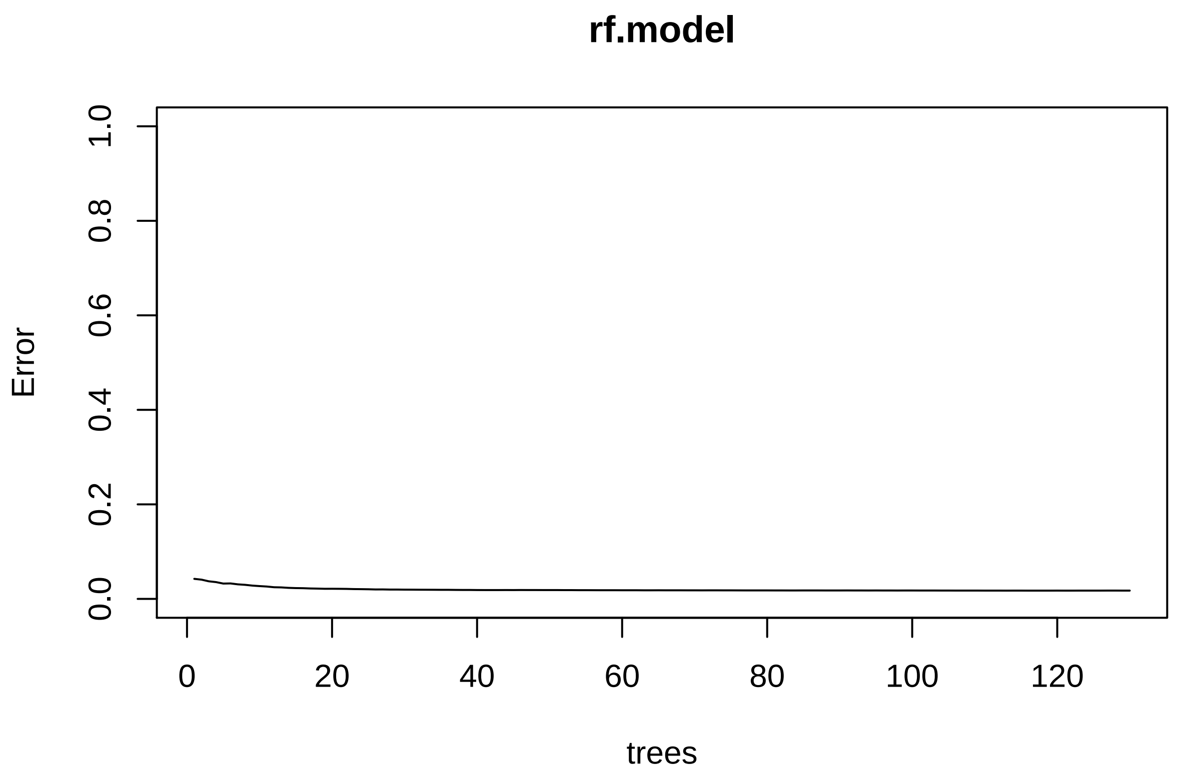
Random Forest

```
rf.model <- randomForest(log(SalePrice + 1) ~ .,
                         data = train.original,
                         importance = TRUE,
                         proximity = TRUE,
                         ntree = 130,
                         do.trace = 5)
```

```
##      |      Out-of-bag   |
## Tree |      MSE  %Var(y) |
##    5 |  0.03241    20.30 |
```

```
##    10 |   0.02702    16.93 |
##    15 |   0.02279    14.28 |
##    20 |   0.02139    13.40 |
##    25 |   0.02035    12.75 |
##    30 |   0.01955    12.24 |
##    35 |   0.01916    12.00 |
##    40 |   0.01868    11.70 |
##    45 |   0.01867    11.69 |
##    50 |   0.01865    11.68 |
##    55 |    0.0184    11.52 |
##    60 |   0.01825    11.43 |
##    65 |   0.01819    11.39 |
##    70 |   0.01806    11.31 |
##    75 |     0.018    11.28 |
##    80 |   0.01791    11.22 |
##    85 |   0.01786    11.19 |
##    90 |    0.0178    11.15 |
##    95 |   0.01779    11.14 |
##   100 |   0.01773    11.11 |
##   105 |   0.01764    11.05 |
##   110 |   0.01753    10.98 |
##   115 |    0.0175    10.96 |
##   120 |    0.0175    10.96 |
##   125 |   0.01753    10.98 |
##   130 |   0.01752    10.98 |
```

```r
plot(rf.model, ylim = c(0, 1))
```

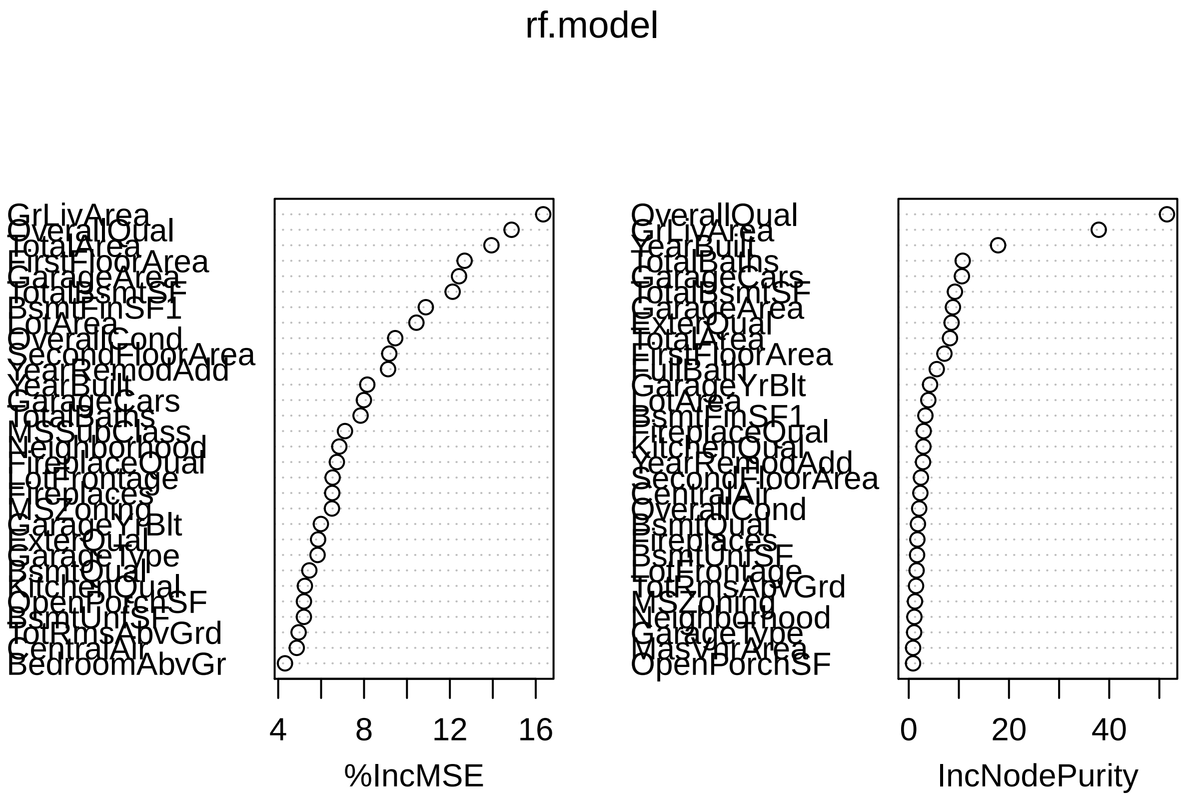



```r
print(rf.model)
```

```
##
```

```
## Call:
##  randomForest(formula = log(SalePrice + 1) ~ ., data = train.original,      importance = TRUE, proxi
##                Type of random forest: regression
##                      Number of trees: 130
## No. of variables tried at each split: 26
##
##           Mean of squared residuals: 0.01752455
##                     % Var explained: 89.02
```

```r
varImpPlot(rf.model)
```



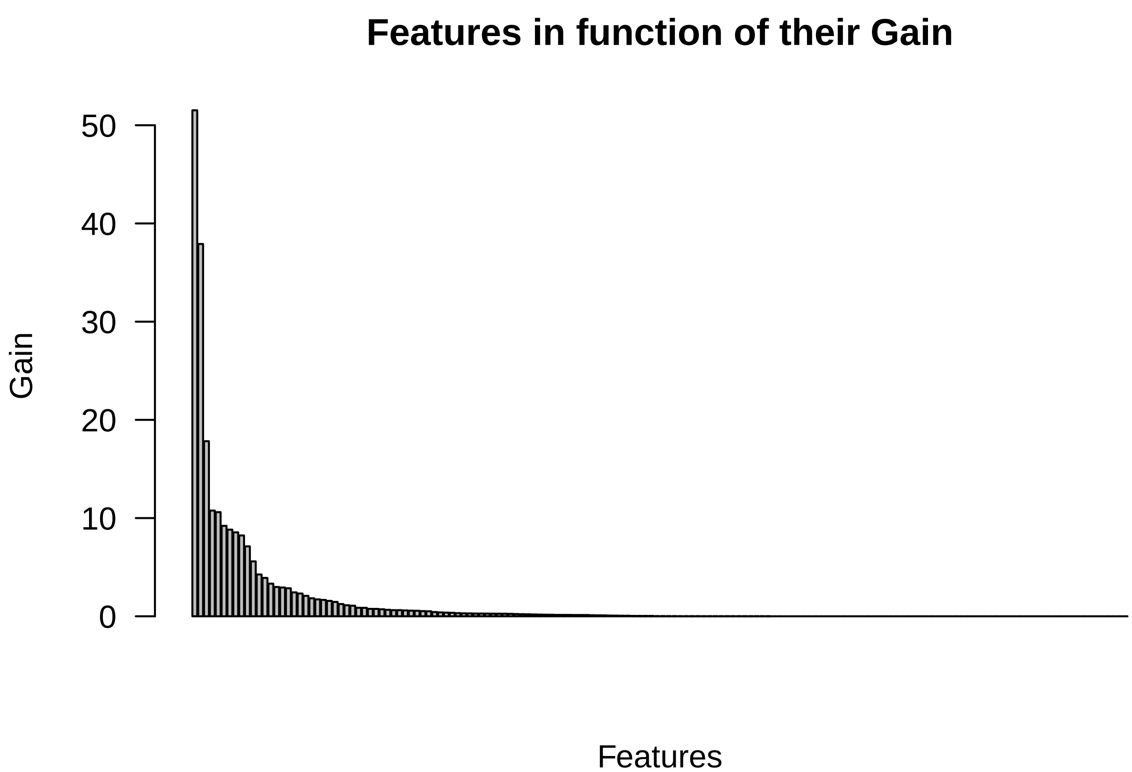

```r
importance(rf.model)
```

```
##                   %IncMSE IncNodePurity
## MSSubClass     7.11127442   0.753670332
## MSZoning       6.50474131   1.245569522
## LotFrontage    6.53663353   1.571686314
## LotArea       10.44026715   3.906232558
## Street         0.00000000   0.011863225
## Alley          1.69043698   0.098292922
## LotShape       3.79325440   0.279208157
## LandContour    1.96852571   0.261672842
## Utilities      0.00000000   0.000000000
## LotConfig      0.52740758   0.185805359
## LandSlope      1.24062478   0.171682444
## Neighborhood   6.84806541   1.123773997
## Condition1    -0.01648701   0.203048658
## Condition2     0.79302514   0.030047726
## BldgType       4.16858366   0.227096530
## HouseStyle     4.10095358   0.319559416
```

```
## OverallQual     14.87392461  51.521077218
## OverallCond      9.45219784   2.084061966
## YearBuilt        8.15118323  17.826971522
## YearRemodAdd     9.11953818   2.851969157
## RoofStyle        3.33993416   0.211208711
## RoofMatl        -2.06901576   0.068549790
## Exterior1st      1.75312589   0.573018146
## Exterior2nd      2.71167204   0.527589600
## MasVnrType       2.48214024   0.163296372
## MasVnrArea       3.86358624   0.866405641
## ExterQual        5.85978257   8.546830529
## ExterCond        1.81167600   0.432392101
## Foundation       1.93278483   0.265068837
## BsmtQual         5.45010636   1.835119145
## BsmtCond         1.99892918   0.248895044
## BsmtExposure     3.88719783   0.273266354
## BsmtFinType1     4.06731577   0.512575311
## BsmtFinSF1      10.87866672   3.323020339
## BsmtFinType2    -0.28465971   0.138571006
## BsmtFinSF2      -0.01059971   0.128587408
## BsmtUnfSF        5.19717028   1.665711105
## TotalBsmtSF     12.12743347   9.210749063
## Heating          1.09013517   0.074054896
## HeatingQualCond  2.58995812   0.277505843
## CentralAir       4.86384651   2.318137643
## Electrical      -0.68106399   0.137653164
## FirstFloorArea  12.68559691   7.117965897
## SecondFloorArea  9.17495087   2.432790648
## LowQualFinSF    -1.55573410   0.034914749
## GrLivArea       16.34882998  37.908987803
## BsmtFullBath     2.61694861   0.299693174
## BsmtHalfBath     1.04940093   0.055889892
## FullBath         4.30012160   5.588304469
## HalfBath         3.44698300   0.154935641
## BedroomAbvGr     4.31625010   0.563385711
## KitchenAbvGr     3.55016392   0.263252152
## KitchenQual      5.24352805   2.929915147
## TotRmsAbvGrd     4.95067719   1.461313214
## Functional       3.26906740   0.290425362
## Fireplaces       6.51860692   1.722801038
## FireplaceQual    6.73389294   2.983185779
## GarageType       5.83455165   1.074961153
## GarageYrBlt      5.98841366   4.258172153
## GarageFinish     4.05340357   0.625365054
## GarageCars       7.99263419  10.605017966
## GarageArea      12.42639363   8.821398497
## GarageQual       3.04444003   0.659488530
## GarageCond       1.18352942   0.715913954
## PavedDrive       2.05901983   0.393713175
## WoodDeckSF       4.10746605   0.623909841
## OpenPorchSF      5.19915765   0.861012991
## EnclosedPorch   -0.71379856   0.354741061
## ScreenPorch      1.27343725   0.130192396
## PoolArea         0.00000000   0.006593939
```

```
## PoolQualCond     -1.40134462   0.094658496
## Fence             0.48569460   0.141901191
## MiscFeature       0.68492563   0.131498459
## MiscVal          -1.19179439   0.053682722
## MoSold           -0.17364021   0.755080450
## YrSold            0.45918708   0.369290411
## SaleType          0.43191688   0.104889736
## SaleCondition     0.05752866   0.595442190
## TotalBaths        7.83861808  10.764246573
## TotalArea        13.93674403   8.229151660
```

```
# 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)
```

```
## RMSE = 0.05482786
```

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) or the Ridge (L2-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.

```
## Note 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] 0.002159425
```

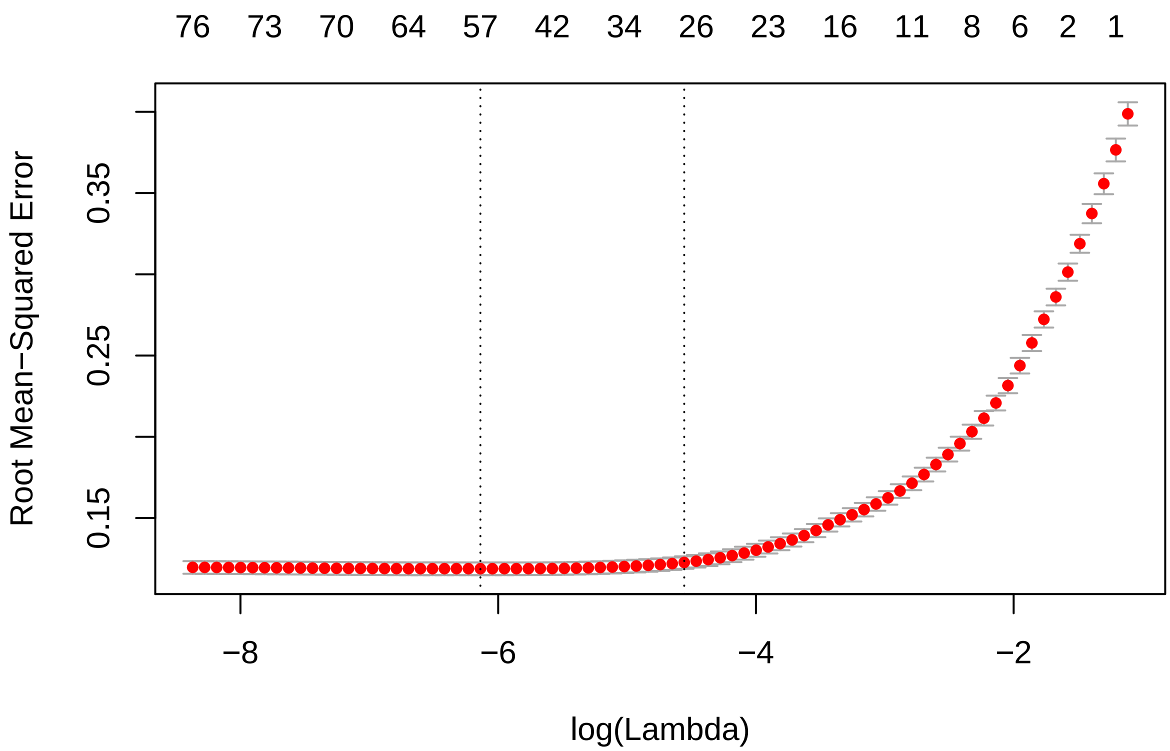
```
cv.model$cvm <- sqrt(cv.model$cvm)
cv.model$cvlo <- sqrt(cv.model$cvlo)
cv.model$cvup <- sqrt(cv.model$cvup)

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

```
##          coef.name     coef.value
## 1      (Intercept) 11.55064520033
## 2      (Intercept)  0.00000000000
## 3       MSSubClass  0.00000000000
## 4         MSZoning -0.00374071725
## 5      LotFrontage  0.00000000000
## 6          LotArea  0.08869469961
## 7           Street  0.14430616551
## 8            Alley  0.00251803168
## 9         LotShape -0.00081114796
## 10     LandContour -0.00558707898
## 11       Utilities -0.02322323404
## 12       LotConfig -0.00127913256
## 13       LandSlope  0.00000000000
## 14    Neighborhood  0.00000000000
## 15      Condition1  0.00000000000
## 16      Condition2  0.00000000000
## 17        BldgType  0.00000000000
## 18      HouseStyle  0.00156357078
## 19     OverallQual  0.06103656693
## 20     OverallCond  0.04205052123
## 21       YearBuilt  0.00154020676
## 22    YearRemodAdd  0.00072910975
## 23       RoofStyle  0.00010765590
```

```
## 24         RoofMatl  0.00000000000
## 25      Exterior1st -0.00068905494
## 26      Exterior2nd  0.00000000000
## 27       MasVnrType  0.00000000000
## 28       MasVnrArea  0.00000000000
## 29        ExterQual -0.00937575393
## 30        ExterCond  0.00463622252
## 31       Foundation  0.01080841226
## 32         BsmtQual -0.00972439790
## 33         BsmtCond  0.00288136160
## 34     BsmtExposure -0.00332453983
## 35     BsmtFinType1  0.00000000000
## 36       BsmtFinSF1  0.00890097713
## 37     BsmtFinType2  0.00006029939
## 38       BsmtFinSF2 -0.00052786335
## 39        BsmtUnfSF -0.00101024027
## 40      TotalBsmtSF  0.00010406751
## 41          Heating  0.00000000000
## 42 HeatingQualCond -0.00149811572
## 43       CentralAir  0.06270271053
## 44       Electrical  0.00000000000
## 45   FirstFloorArea  0.03030070024
## 46 SecondFloorArea  0.00000000000
## 47     LowQualFinSF -0.00328986920
## 48        GrLivArea  0.36568420716
## 49     BsmtFullBath  0.00985835038
## 50     BsmtHalfBath -0.00474400774
## 51         FullBath  0.00000000000
## 52         HalfBath  0.00000000000
## 53     BedroomAbvGr -0.00526803311
## 54     KitchenAbvGr -0.19245422045
## 55      KitchenQual -0.01013412239
## 56     TotRmsAbvGrd  0.00218282092
## 57       Functional  0.01861426278
## 58       Fireplaces  0.02534054634
## 59    FireplaceQual  0.00000000000
## 60       GarageType  0.00143982035
## 61      GarageYrBlt  0.00000000000
## 62     GarageFinish -0.00098311618
## 63       GarageCars  0.02628039791
## 64       GarageArea  0.00006444738
## 65       GarageQual  0.00000000000
## 66       GarageCond  0.00096183108
## 67       PavedDrive  0.01861330974
## 68       WoodDeckSF  0.00269010296
## 69      OpenPorchSF  0.00000000000
## 70    EnclosedPorch  0.00012576076
## 71      ScreenPorch  0.00668816230
## 72         PoolArea  0.01005732591
## 73     PoolQualCond -0.00019424373
## 74            Fence  0.00000000000
## 75      MiscFeature  0.00001911476
## 76          MiscVal -0.00368657328
## 77           MoSold  0.00000000000
```

```
## 78          YrSold -0.00443503936
## 79        SaleType -0.00044963848
## 80   SaleCondition  0.02123122578
## 81      TotalBaths  0.02454968158
## 82       TotalArea  0.00002275463
```

```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"        "LotArea"         "Street"
##  [4] "Alley"           "LotShape"        "LandContour"
##  [7] "Utilities"       "LotConfig"       "HouseStyle"
## [10] "OverallQual"     "OverallCond"     "YearBuilt"
## [13] "YearRemodAdd"    "RoofStyle"       "Exterior1st"
## [16] "ExterQual"       "ExterCond"       "Foundation"
## [19] "BsmtQual"        "BsmtCond"        "BsmtExposure"
## [22] "BsmtFinSF1"      "BsmtFinType2"    "BsmtFinSF2"
## [25] "BsmtUnfSF"       "TotalBsmtSF"     "HeatingQualCond"
## [28] "CentralAir"      "FirstFloorArea"  "LowQualFinSF"
## [31] "GrLivArea"       "BsmtFullBath"    "BsmtHalfBath"
## [34] "BedroomAbvGr"    "KitchenAbvGr"    "KitchenQual"
## [37] "TotRmsAbvGrd"    "Functional"      "Fireplaces"
## [40] "GarageType"      "GarageFinish"    "GarageCars"
## [43] "GarageArea"      "GarageCond"      "PavedDrive"
## [46] "WoodDeckSF"      "EnclosedPorch"   "ScreenPorch"
## [49] "PoolArea"        "PoolQualCond"    "MiscFeature"
## [52] "MiscVal"         "YrSold"          "SaleType"
## [55] "SaleCondition"   "TotalBaths"      "TotalArea"
```

```r
#train <- train[, colnames(train) %in% features]
#test <- test[, colnames(test) %in% features]

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

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

```
##            Overall
## 1  13.10784930605
## 2   0.00000000000
## 3   0.00110814421
## 4   0.00550558595
## 5   0.00000000000
## 6   0.09123903106
## 7   0.16616867928
## 8   0.00733646363
## 9   0.00071388354
## 10  0.00715656689
## 11  0.06689416379
## 12  0.00179683051
## 13  0.00039554154
## 14  0.00014559028
## 15  0.00046905481
## 16  0.00258539346
## 17  0.00000000000
## 18  0.00230263212
## 19  0.05877331343
## 20  0.04423120725
## 21  0.00160792674
## 22  0.00067752294
## 23  0.00126580960
## 24  0.00000000000
## 25  0.00260970041
## 26  0.00159716163
## 27  0.00000000000
## 28  0.00039779865
## 29  0.00959907787
## 30  0.00518241359
## 31  0.01199984338
## 32  0.00970613987
## 33  0.00368363595
## 34  0.00344735101
## 35  0.00000000000
## 36  0.00895100199
## 37  0.00039086631
## 38  0.00159222519
## 39  0.00278961961
## 40  0.00011214639
## 41  0.00000000000
## 42  0.00153752258
```

```
## 43  0.06454081340
## 44  0.00000000000
## 45  0.03421194881
## 46  0.00000000000
## 47  0.00490539137
## 48  0.36121706967
## 49  0.00863304180
## 50  0.00990915448
## 51  0.00000000000
## 52  0.00000000000
## 53  0.00922837416
## 54  0.21354226066
## 55  0.01000569070
## 56  0.00526951327
## 57  0.01951542911
## 58  0.02492668151
## 59  0.00000000000
## 60  0.00313419767
## 61  0.00000000000
## 62  0.00396328608
## 63  0.02464949608
## 64  0.00006189533
## 65  0.00000000000
## 66  0.00136784024
## 67  0.02010499001
## 68  0.00314798637
## 69  0.00000000000
## 70  0.00151663879
## 71  0.00775937967
## 72  0.01538591400
## 73  0.00244154165
## 74  0.00038714484
## 75  0.00000000000
## 76  0.00465262224
## 77  0.00000000000
## 78  0.00523001468
## 79  0.00147911062
## 80  0.02229881452
## 81  0.02451867589
## 82  0.00002022940
```

```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 = 0.1116869
```

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.

Submission

We write the 'Id' associated to the predicted SalePrice in the submission file.

```r
prediction.test <- 0.5 * net.prediction.test + 0.5 * xgb.prediction.test

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

head(submission, 10)
```

```
##      Id SalePrice
## 1  1461  124543.4
## 2  1462  158906.7
## 3  1463  183698.9
## 4  1464  194196.1
## 5  1465  184178.7
## 6  1466  175180.8
## 7  1467  172505.4
## 8  1468  164211.4
## 9  1469  189484.5
## 10 1470  119379.5
```

Benchmark

Conclusion