edX Data Science Capstone: Choose Your Own (CYO): Housing Prices

Komalkumar Tagdiwala

October 2020

Contents

Introduction

Machine Learning has several applications in different industries. Some of the most common applications include determining credit worthiness for issuing a loan, detecting fraud, making recommendations for movies or purchases (think Netflix, Amazon, etc.). Making predictions is another big area where machine learning plays a huge part by employing algorithms trained against an existing set of data that are used for making predictions against future data streams of the same kind. This paper is focused on making predictions of housing prices using machine learning.

Executive Summary

Dataset

This project explores the use of machine learning to predicting housing prices for a dataset available on www.kaggle.com at https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data. The underlying source of this data is the Ames Housing dataset (Ames, Iowa) compiled by Dean De Cook for use in data science education. It's an incredible alternative for data scientists looking for a modernized and expanded version of the often cited Boston Housing dataset.

Goal

The goal of this Capstone is to train a machine learning algorithm using the inputs in one subset to predict housing prices in the validation set. The data set already includes separate train.csv and test.csv files provided by Kaggle for training and validation sets. RMSE, known as the Root Mean Square Error, will be used to evaluate how close the predictions made by our model are to the actual/true values contained in the validation set. We will pick the model that yields the lowest RMSE.

Overview

The dataset is provided as 2 separate csv files - one for training set (train.csv) and one for validation (test.csv). As we shall see in depth below, the **training** set has 1460 rows with 81 features whereas the **validation** set has 1459 rows of the same features except for our outcome/dependent variable, **SalePrice**. *Consequently, when running our chosen model against the Validation set, we will only be able to make predictions for the independent variables and not be able to compute accuracy or RMSE values*. All the features represent different characteristics of a house that will have some influence on the **SalePrice**.

Note: Given the comprehensive list of features (81) in this data set, **please be patient with the length of this report** as many of the outputs used in data exploration may run into several pages. I have attempted to comment some of them and limit outputs to one or two fields to represent what the actual output contains but just setting expectations that the report is lengthy because of the above reasons.

Before we can review what is in the dataset, we need to load the dataset using the train.csv and test.csv files provided by Kaggle.

```
# Load the required libraries

#For Data Load/Processing/Generation of Training and Validation Data
library(tidyverse)
library(caret)
library(data.table)
```

```r
library(DataExplorer) # Library for comprehensive data exploration
library(mosaic) #favstats and utility functions for data exploration
                 # (missing values, etc.)

library(Hmisc) # For describe() for data exploration. Hmisc Contains many
               # utility operations, etc.

library(ggplot2) #For Data Visualization (ggplot, qplot, etc.)

library(rafalib) # For mypar() to optimize graphical parameters for the RStudio plot window

library(dplyr) # For utility functions (%.%, %>%, etc.)
library(plyr) # Tools for Splitting, Applying and Combining Data (Example: ddply())

library(caTools) # Utility functions for splitting dataset

library(Boruta) # For Data Exploration and determining important features for analysis

library(randomForest) # For modeling using Random Forest

library(forecast) # For computing accuracy of predictions

library(rpart) # For Classification and Regression Tree-based modeling (using cart)
library(rpart.plot) # To visualize results from cart-based model

library(broom) # To view tidy results
```

Create training and validation set (final hold-out test set). The file train.csv was downloaded and made available in the local folder. Read it from the local file system.

```r
# setwd(dirname(rstudioapi::getActiveDocumentContext()$path))
training <- read_csv('train.csv') # Tibble of 1460 observations of 81 variables
validation <- read_csv('test.csv') # Tibble of 1459 observations of 80 variables
```

We begin by first examining what training and validation are.

```r
class(training)
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

```r
class(validation)
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

Let us now review the structure of the dataset to understand its composition.

```r
str(training)
```

```
## tibble [1,460 x 81] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Id           : num [1:1460] 1 2 3 4 5 6 7 8 9 10 ...
```

```
##  $ MSSubClass   : num [1:1460] 60 20 60 70 60 50 20 60 50 190 ...
##  $ MSZoning     : chr [1:1460] "RL" "RL" "RL" "RL" ...
##  $ LotFrontage  : num [1:1460] 65 80 68 60 84 85 75 NA 51 50 ...
##  $ LotArea      : num [1:1460] 8450 9600 11250 9550 14260 ...
##  $ Street       : chr [1:1460] "Pave" "Pave" "Pave" "Pave" ...
##  $ Alley        : chr [1:1460] NA NA NA NA ...
##  $ LotShape     : chr [1:1460] "Reg" "Reg" "IR1" "IR1" ...
##  $ LandContour  : chr [1:1460] "Lvl" "Lvl" "Lvl" "Lvl" ...
##  $ Utilities    : chr [1:1460] "AllPub" "AllPub" "AllPub" "AllPub" ...
##  $ LotConfig    : chr [1:1460] "Inside" "FR2" "Inside" "Corner" ...
##  $ LandSlope    : chr [1:1460] "Gtl" "Gtl" "Gtl" "Gtl" ...
##  $ Neighborhood : chr [1:1460] "CollgCr" "Veenker" "CollgCr" "Crawfor" ...
##  $ Condition1   : chr [1:1460] "Norm" "Feedr" "Norm" "Norm" ...
##  $ Condition2   : chr [1:1460] "Norm" "Norm" "Norm" "Norm" ...
##  $ BldgType     : chr [1:1460] "1Fam" "1Fam" "1Fam" "1Fam" ...
##  $ HouseStyle   : chr [1:1460] "2Story" "1Story" "2Story" "2Story" ...
##  $ OverallQual  : num [1:1460] 7 6 7 7 8 5 8 7 7 5 ...
##  $ OverallCond  : num [1:1460] 5 8 5 5 5 5 5 6 5 6 ...
##  $ YearBuilt    : num [1:1460] 2003 1976 2001 1915 2000 ...
##  $ YearRemodAdd : num [1:1460] 2003 1976 2002 1970 2000 ...
##  $ RoofStyle    : chr [1:1460] "Gable" "Gable" "Gable" "Gable" ...
##  $ RoofMatl     : chr [1:1460] "CompShg" "CompShg" "CompShg" "CompShg" ...
##  $ Exterior1st  : chr [1:1460] "VinylSd" "MetalSd" "VinylSd" "Wd Sdng" ...
##  $ Exterior2nd  : chr [1:1460] "VinylSd" "MetalSd" "VinylSd" "Wd Shng" ...
##  $ MasVnrType   : chr [1:1460] "BrkFace" "None" "BrkFace" "None" ...
##  $ MasVnrArea   : num [1:1460] 196 0 162 0 350 0 186 240 0 0 ...
##  $ ExterQual    : chr [1:1460] "Gd" "TA" "Gd" "TA" ...
##  $ ExterCond    : chr [1:1460] "TA" "TA" "TA" "TA" ...
##  $ Foundation   : chr [1:1460] "PConc" "CBlock" "PConc" "BrkTil" ...
##  $ BsmtQual     : chr [1:1460] "Gd" "Gd" "Gd" "TA" ...
##  $ BsmtCond     : chr [1:1460] "TA" "TA" "TA" "Gd" ...
##  $ BsmtExposure : chr [1:1460] "No" "Gd" "Mn" "No" ...
##  $ BsmtFinType1 : chr [1:1460] "GLQ" "ALQ" "GLQ" "ALQ" ...
##  $ BsmtFinSF1   : num [1:1460] 706 978 486 216 655 ...
##  $ BsmtFinType2 : chr [1:1460] "Unf" "Unf" "Unf" "Unf" ...
##  $ BsmtFinSF2   : num [1:1460] 0 0 0 0 0 0 0 32 0 0 ...
##  $ BsmtUnfSF    : num [1:1460] 150 284 434 540 490 64 317 216 952 140 ...
##  $ TotalBsmtSF  : num [1:1460] 856 1262 920 756 1145 ...
##  $ Heating      : chr [1:1460] "GasA" "GasA" "GasA" "GasA" ...
##  $ HeatingQC    : chr [1:1460] "Ex" "Ex" "Ex" "Gd" ...
##  $ CentralAir   : chr [1:1460] "Y" "Y" "Y" "Y" ...
##  $ Electrical   : chr [1:1460] "SBrkr" "SBrkr" "SBrkr" "SBrkr" ...
##  $ 1stFlrSF     : num [1:1460] 856 1262 920 961 1145 ...
##  $ 2ndFlrSF     : num [1:1460] 854 0 866 756 1053 ...
##  $ LowQualFinSF : num [1:1460] 0 0 0 0 0 0 0 0 0 0 ...
##  $ GrLivArea    : num [1:1460] 1710 1262 1786 1717 2198 ...
##  $ BsmtFullBath : num [1:1460] 1 0 1 1 1 1 1 1 0 1 ...
##  $ BsmtHalfBath : num [1:1460] 0 1 0 0 0 0 0 0 0 0 ...
##  $ FullBath     : num [1:1460] 2 2 2 1 2 1 2 2 2 1 ...
##  $ HalfBath     : num [1:1460] 1 0 1 0 1 1 0 1 0 0 ...
##  $ BedroomAbvGr : num [1:1460] 3 3 3 3 4 1 3 3 2 2 ...
##  $ KitchenAbvGr : num [1:1460] 1 1 1 1 1 1 1 1 2 2 ...
##  $ KitchenQual  : chr [1:1460] "Gd" "TA" "Gd" "Gd" ...
##  $ TotRmsAbvGrd : num [1:1460] 8 6 6 7 9 5 7 7 8 5 ...
```

```
##  $ Functional   : chr [1:1460] "Typ" "Typ" "Typ" "Typ" ...
##  $ Fireplaces   : num [1:1460] 0 1 1 1 1 0 1 2 2 2 ...
##  $ FireplaceQu  : chr [1:1460] NA "TA" "TA" "Gd" ...
##  $ GarageType   : chr [1:1460] "Attchd" "Attchd" "Attchd" "Detchd" ...
##  $ GarageYrBlt  : num [1:1460] 2003 1976 2001 1998 2000 ...
##  $ GarageFinish : chr [1:1460] "RFn" "RFn" "RFn" "Unf" ...
##  $ GarageCars   : num [1:1460] 2 2 2 3 3 2 2 2 2 1 ...
##  $ GarageArea   : num [1:1460] 548 460 608 642 836 480 636 484 468 205 ...
##  $ GarageQual   : chr [1:1460] "TA" "TA" "TA" "TA" ...
##  $ GarageCond   : chr [1:1460] "TA" "TA" "TA" "TA" ...
##  $ PavedDrive   : chr [1:1460] "Y" "Y" "Y" "Y" ...
##  $ WoodDeckSF   : num [1:1460] 0 298 0 0 192 40 255 235 90 0 ...
##  $ OpenPorchSF  : num [1:1460] 61 0 42 35 84 30 57 204 0 4 ...
##  $ EnclosedPorch: num [1:1460] 0 0 0 272 0 0 0 228 205 0 ...
##  $ 3SsnPorch    : num [1:1460] 0 0 0 0 0 320 0 0 0 0 ...
##  $ ScreenPorch  : num [1:1460] 0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolArea     : num [1:1460] 0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolQC       : chr [1:1460] NA NA NA NA ...
##  $ Fence        : chr [1:1460] NA NA NA NA ...
##  $ MiscFeature  : chr [1:1460] NA NA NA NA ...
##  $ MiscVal      : num [1:1460] 0 0 0 0 0 700 0 350 0 0 ...
##  $ MoSold       : num [1:1460] 2 5 9 2 12 10 8 11 4 1 ...
##  $ YrSold       : num [1:1460] 2008 2007 2008 2006 2008 ...
##  $ SaleType     : chr [1:1460] "WD" "WD" "WD" "WD" ...
##  $ SaleCondition: chr [1:1460] "Normal" "Normal" "Normal" "Abnorml" ...
##  $ SalePrice    : num [1:1460] 208500 181500 223500 140000 250000 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Id = col_double(),
##   ..   MSSubClass = col_double(),
##   ..   MSZoning = col_character(),
##   ..   LotFrontage = col_double(),
##   ..   LotArea = col_double(),
##   ..   Street = col_character(),
##   ..   Alley = col_character(),
##   ..   LotShape = col_character(),
##   ..   LandContour = col_character(),
##   ..   Utilities = col_character(),
##   ..   LotConfig = col_character(),
##   ..   LandSlope = col_character(),
##   ..   Neighborhood = col_character(),
##   ..   Condition1 = col_character(),
##   ..   Condition2 = col_character(),
##   ..   BldgType = col_character(),
##   ..   HouseStyle = col_character(),
##   ..   OverallQual = col_double(),
##   ..   OverallCond = col_double(),
##   ..   YearBuilt = col_double(),
##   ..   YearRemodAdd = col_double(),
##   ..   RoofStyle = col_character(),
##   ..   RoofMatl = col_character(),
##   ..   Exterior1st = col_character(),
##   ..   Exterior2nd = col_character(),
##   ..   MasVnrType = col_character(),
```

```
##   ..   MasVnrArea = col_double(),
##   ..   ExterQual = col_character(),
##   ..   ExterCond = col_character(),
##   ..   Foundation = col_character(),
##   ..   BsmtQual = col_character(),
##   ..   BsmtCond = col_character(),
##   ..   BsmtExposure = col_character(),
##   ..   BsmtFinType1 = col_character(),
##   ..   BsmtFinSF1 = col_double(),
##   ..   BsmtFinType2 = col_character(),
##   ..   BsmtFinSF2 = col_double(),
##   ..   BsmtUnfSF = col_double(),
##   ..   TotalBsmtSF = col_double(),
##   ..   Heating = col_character(),
##   ..   HeatingQC = col_character(),
##   ..   CentralAir = col_character(),
##   ..   Electrical = col_character(),
##   ..   `1stFlrSF` = col_double(),
##   ..   `2ndFlrSF` = col_double(),
##   ..   LowQualFinSF = col_double(),
##   ..   GrLivArea = col_double(),
##   ..   BsmtFullBath = col_double(),
##   ..   BsmtHalfBath = col_double(),
##   ..   FullBath = col_double(),
##   ..   HalfBath = col_double(),
##   ..   BedroomAbvGr = col_double(),
##   ..   KitchenAbvGr = col_double(),
##   ..   KitchenQual = col_character(),
##   ..   TotRmsAbvGrd = col_double(),
##   ..   Functional = col_character(),
##   ..   Fireplaces = col_double(),
##   ..   FireplaceQu = col_character(),
##   ..   GarageType = col_character(),
##   ..   GarageYrBlt = col_double(),
##   ..   GarageFinish = col_character(),
##   ..   GarageCars = col_double(),
##   ..   GarageArea = col_double(),
##   ..   GarageQual = col_character(),
##   ..   GarageCond = col_character(),
##   ..   PavedDrive = col_character(),
##   ..   WoodDeckSF = col_double(),
##   ..   OpenPorchSF = col_double(),
##   ..   EnclosedPorch = col_double(),
##   ..   `3SsnPorch` = col_double(),
##   ..   ScreenPorch = col_double(),
##   ..   PoolArea = col_double(),
##   ..   PoolQC = col_character(),
##   ..   Fence = col_character(),
##   ..   MiscFeature = col_character(),
##   ..   MiscVal = col_double(),
##   ..   MoSold = col_double(),
##   ..   YrSold = col_double(),
##   ..   SaleType = col_character(),
##   ..   SaleCondition = col_character(),
```

```
##   ..   SalePrice = col_double()
##   .. )
```

We see that *our training set* has **1,460** observations of 81 variables/features.

```r
# Validation Set = validation
colnames(validation)
```

```
##  [1] "Id"            "MSSubClass"    "MSZoning"      "LotFrontage"
##  [5] "LotArea"       "Street"        "Alley"         "LotShape"
##  [9] "LandContour"   "Utilities"     "LotConfig"     "LandSlope"
## [13] "Neighborhood"  "Condition1"    "Condition2"    "BldgType"
## [17] "HouseStyle"    "OverallQual"   "OverallCond"   "YearBuilt"
## [21] "YearRemodAdd"  "RoofStyle"     "RoofMatl"      "Exterior1st"
## [25] "Exterior2nd"   "MasVnrType"    "MasVnrArea"    "ExterQual"
## [29] "ExterCond"     "Foundation"    "BsmtQual"      "BsmtCond"
## [33] "BsmtExposure"  "BsmtFinType1"  "BsmtFinSF1"    "BsmtFinType2"
## [37] "BsmtFinSF2"    "BsmtUnfSF"     "TotalBsmtSF"   "Heating"
## [41] "HeatingQC"     "CentralAir"    "Electrical"    "1stFlrSF"
## [45] "2ndFlrSF"      "LowQualFinSF"  "GrLivArea"     "BsmtFullBath"
## [49] "BsmtHalfBath"  "FullBath"      "HalfBath"      "BedroomAbvGr"
## [53] "KitchenAbvGr"  "KitchenQual"   "TotRmsAbvGrd"  "Functional"
## [57] "Fireplaces"    "FireplaceQu"   "GarageType"    "GarageYrBlt"
## [61] "GarageFinish"  "GarageCars"    "GarageArea"    "GarageQual"
## [65] "GarageCond"    "PavedDrive"    "WoodDeckSF"    "OpenPorchSF"
## [69] "EnclosedPorch" "3SsnPorch"     "ScreenPorch"   "PoolArea"
## [73] "PoolQC"        "Fence"         "MiscFeature"   "MiscVal"
## [77] "MoSold"        "YrSold"        "SaleType"      "SaleCondition"
```

Our *validation set* has **1,459** observations of the same 80 features with the exception of the SalePrice column, which is our outcome/dependent variable that must be predicted.

We are interested in the prediction of SalePrice (y). Consequently, we can make the following determination:

- SalePrice = Outcome/Dependent Variable "y"
- Remaining 80 variables are the predictors/independent variables that will be used for our analysis.

Let us see a few of the records to examine type different values in some of the rows

```r
head(training)
```

```
## # A tibble: 6 x 81
##      Id MSSubClass MSZoning LotFrontage LotArea Street Alley LotShape
##   <dbl>      <dbl> <chr>          <dbl>   <dbl> <chr>  <chr> <chr>
## 1     1         60 RL                65    8450 Pave   <NA>  Reg
## 2     2         20 RL                80    9600 Pave   <NA>  Reg
## 3     3         60 RL                68   11250 Pave   <NA>  IR1
## 4     4         70 RL                60    9550 Pave   <NA>  IR1
## 5     5         60 RL                84   14260 Pave   <NA>  IR1
## 6     6         50 RL                85   14115 Pave   <NA>  IR1
## # ... with 73 more variables: LandContour <chr>, Utilities <chr>,
## #   LotConfig <chr>, LandSlope <chr>, Neighborhood <chr>, Condition1 <chr>,
```

```
## #   Condition2 <chr>, BldgType <chr>, HouseStyle <chr>, OverallQual <dbl>,
## #   OverallCond <dbl>, YearBuilt <dbl>, YearRemodAdd <dbl>, RoofStyle <chr>,
## #   RoofMatl <chr>, Exterior1st <chr>, Exterior2nd <chr>, MasVnrType <chr>,
## #   MasVnrArea <dbl>, ExterQual <chr>, ExterCond <chr>, Foundation <chr>,
## #   BsmtQual <chr>, BsmtCond <chr>, BsmtExposure <chr>, BsmtFinType1 <chr>,
## #   BsmtFinSF1 <dbl>, BsmtFinType2 <chr>, BsmtFinSF2 <dbl>, BsmtUnfSF <dbl>,
## #   TotalBsmtSF <dbl>, Heating <chr>, HeatingQC <chr>, CentralAir <chr>,
## #   Electrical <chr>, `1stFlrSF` <dbl>, `2ndFlrSF` <dbl>, LowQualFinSF <dbl>,
## #   GrLivArea <dbl>, BsmtFullBath <dbl>, BsmtHalfBath <dbl>, FullBath <dbl>,
## #   HalfBath <dbl>, BedroomAbvGr <dbl>, KitchenAbvGr <dbl>, KitchenQual <chr>,
## #   TotRmsAbvGrd <dbl>, Functional <chr>, Fireplaces <dbl>, FireplaceQu <chr>,
## #   GarageType <chr>, GarageYrBlt <dbl>, GarageFinish <chr>, GarageCars <dbl>,
## #   GarageArea <dbl>, GarageQual <chr>, GarageCond <chr>, PavedDrive <chr>,
## #   WoodDeckSF <dbl>, OpenPorchSF <dbl>, EnclosedPorch <dbl>,
## #   `3SsnPorch` <dbl>, ScreenPorch <dbl>, PoolArea <dbl>, PoolQC <chr>,
## #   Fence <chr>, MiscFeature <chr>, MiscVal <dbl>, MoSold <dbl>, YrSold <dbl>,
## #   SaleType <chr>, SaleCondition <chr>, SalePrice <dbl>
```

A quick look at the **top 6 rows** in the dataset reveals that each row represents the comprehensive characteristics of a single house along with the SalePrice of the house.

The different features present in the dataset can be easily summarized using the *introduce()* function in the **dataexplorer** library as follows.

```r
introduce(training)
```

```
## # A tibble: 1 x 9
##    rows columns discrete_columns continuous_colu~ all_missing_col~
##   <int>   <int>            <int>            <int>            <int>
## 1  1460      81               43               38                0
## # ... with 4 more variables: total_missing_values <int>, complete_rows <int>,
## #   total_observations <int>, memory_usage <dbl>
```

We see that of the 81 features, **43** are **discrete** while **38** are **continuous** in nature.

Given the volume of features available in this dataset, the provider of this dataset has included a separate **text** file, **description.txt**, that includes comprehensive descriptions for each field along with a detailed explanation of the encoded values contained in each field. For ease of reference, I am pasting the description for each field below.

Outcome (y)

SalePrice: Contains the SalePrice a buyer paid for a given house. This is what we want to predict using machine learning.

FIELD DESCRIPTIONS from data_description.txt

1. Id: House ID
2. 1stFlrSF: First Floor square feet
3. 2ndFlrSF: Second floor square feet
4. 3SsnPorch: Three season porch area in square feet
5. Alley: Type of alley access to property

6. Bedroom: Bedrooms above grade (does NOT include basement bedrooms)
7. BldgType: Type of dwelling
8. BsmtCond: Evaluates the general condition of the basement
9. BsmtExposure: Refers to walkout or garden level walls
10. BsmtFinSF1: Type 1 finished square feet
11. BsmtFinSF2: Type 2 finished square feet
12. BsmtFinType1: Rating of basement finished area
13. BsmtFinType2: Rating of basement finished area (if multiple types)
14. BsmtFullBath: Basement full bathrooms
15. BsmtHalfBath: Basement half bathrooms
16. BsmtQual: Evaluates the height of the basement
17. BsmtUnfSF: Unfinished square feet of basement area
18. CentralAir: Central air conditioning
19. Condition1: Proximity to various conditions
20. Condition2: Proximity to various conditions (if more than one is present)
21. Electrical: Electrical system
22. EnclosedPorch: Enclosed porch area in square feet
23. ExterCond: Evaluates the present condition of the material on the exterior
24. Exterior1st: Exterior covering on house
25. Exterior2nd: Exterior covering on house (if more than one material)
26. ExterQual: Evaluates the quality of the material on the exterior
27. Fence: Fence quality
28. FireplaceQu: Fireplace quality
29. Fireplaces: Number of fireplaces
30. Foundation: Type of foundation
31. FullBath: Full bathrooms above grade
32. Functional: Home functionality (Assume typical unless deductions are warranted)
33. GarageArea: Size of garage in square feet
34. GarageCars: Size of garage in car capacity
35. GarageCond: Garage condition
36. GarageFinish: Interior finish of the garage
37. GarageQual: Garage quality
38. GarageType: Garage location

39. GarageYrBlt: Year garage was built
40. GrLivArea: Above grade (ground) living area square feet
41. HalfBath: Half baths above grade
42. Heating: Type of heating
43. HeatingQC: Heating quality and condition
44. HouseStyle: Style of dwelling
45. Kitchen: Kitchens above grade
46. KitchenQual: Kitchen quality
47. LandContour: Flatness of the property
48. LandSlope: Slope of property
49. LotArea: Lot size in square feet
50. LotConfig: Lot configuration
51. LotFrontage: Linear feet of street connected to property
52. LotShape: General shape of property
53. LowQualFinSF: Low quality finished square feet (all floors)
54. MasVnrArea: Masonry veneer area in square feet
55. MasVnrType: Masonry veneer type
56. MiscFeature: Miscellaneous feature not covered in other categories
57. MiscVal: $Value of miscellaneous feature
58. MoSold: Month Sold (MM)
59. MSSubClass: Identifies the type of dwelling involved in the sale.
60. MSZoning: Identifies the general zoning classification of the sale.
61. Neighborhood: Physical locations within Ames city limits
62. OverallCond: Rates the overall condition of the house
63. OverallQual: Rates the overall material and finish of the house
64. PavedDrive: Paved driveway
65. penPorchSF: Open porch area in square feet
66. PoolArea: Pool area in square feet
67. PoolQC: Pool quality
68. RoofMatl: Roof material
69. RoofStyle: Type of roof
70. SaleCondition: Condition of sale
71. SalePrice: Selling price of the house

72. SaleType: Type of sale
73. ScreenPorch: Screen porch area in square feet
74. Street: Type of road access to property
75. TotalBsmtSF: Total square feet of basement area
76. TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)
77. Utilities: Type of utilities available
78. WoodDeckSF: Wood deck area in square feet
79. YearBuilt: Original construction date
80. YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)
81. YrSold: Year Sold (YYYY)

Key Steps

We will be undertaking the following steps to achieve our goal of predicting the **SalePrice**:

- Data Wrangling/Cleaning/Pre-processing
- Data Exploration
- Data Visualization
- Insights Gained from the prior steps
- Build one or more Machine Learning Models
- Make Predictions using our model(s)
- Choose the model with lowest RMSE

We begin data exploration using the **DataExplorer** package, which allows us to quickly explore key characteristics of the data set including several visualization plots with minimum lines of code. Similarly, **Hmisc** provides a comprehensive summary of the various features of the data set including missing values, unique values, etc. Using the findings from both these packages, we will undertake **Feature Engineering** by dropping columns that contain several missing values while keeping others. Next, we will employ the **Boruta** package to further improve our feature engineering exercise to narrow down on those features that have the potential for maximum impact on our outcome variable, SalePrice.

For the model building, we will begin with **Random Forest**. Random forest is a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of overcoming over-fitting problem of individual decision tree. It is a good technique to employ particularly when we have a large set of features which can otherwise lead to the curse of dimensionality problem. Too many dimensions causes every observation in your dataset to appear equidistant from all the others. It can also lead to overfitting.

Next, we will build a **Classification and Regression Tree (CART)**-based model to analyze the splits and understand how decisions are made at the various nodes. We will then undertake **linear regression** to identify statistically significant features and compute RMSE. Finally, we will be **comparing the RMSEs** obtained from each of the models and **choose the one with the lowest RMSE**.

Methods/Analysis

Process and Techniques

Data Cleaning/Pre-processing/Wrangling

Data is generated and maintained differently in different systems. Depending on how efficient business constraints exist in a system for capturing data, performing input validations, etc., the data captured by the system may contain a lot of junk/irrelevant kind of data for one or more fields in addition to being plagued by a very common issue, missing data.

We begin our data analysis by first examining the current data structure, looking for obvious signs of missing values, identify the need to either discard the missing values or substitute them with best practices, such as average of the given field across all observations. As such, Data exploration and Data pre-processing go hand in hand and may require multiple iterations before proceeding with actual analysis.

Building upon our knowledge from executive summary, we now dive deep into the constituents of our dataset to observe signs of any missing data as well as get a summary of how the different features of our data can potentially impact our analysis.

We start with the summary() function to obtain the big picture on our dataset for things, such as min, max, median, quantiles, class, etc.

```r
summary(training)
```

```
##        Id           MSSubClass      MSZoning          LotFrontage    
##  Min.   :   1.0   Min.   : 20.0   Length:1460        Min.   : 21.00  
##  1st Qu.: 365.8   1st Qu.: 20.0   Class :character   1st Qu.: 59.00  
##  Median : 730.5   Median : 50.0   Mode  :character   Median : 69.00  
##  Mean   : 730.5   Mean   : 56.9                      Mean   : 70.05  
##  3rd Qu.:1095.2   3rd Qu.: 70.0                      3rd Qu.: 80.00  
##  Max.   :1460.0   Max.   :190.0                      Max.   :313.00  
##                                                      NA's   :259     
##     LotArea          Street             Alley             LotShape        
##  Min.   :  1300   Length:1460        Length:1460        Length:1460       
##  1st Qu.:  7554   Class :character   Class :character   Class :character  
##  Median :  9478   Mode  :character   Mode  :character   Mode  :character  
##  Mean   : 10517                                                           
##  3rd Qu.: 11602                                                           
##  Max.   :215245                                                           
##                                                                           
##  LandContour         Utilities          LotConfig          LandSlope        
##  Length:1460        Length:1460        Length:1460        Length:1460       
##  Class :character   Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character  
##                                                                             
##                                                                             
##                                                                             
##                                                                             
##  Neighborhood        Condition1         Condition2          BldgType        
##  Length:1460        Length:1460        Length:1460        Length:1460       
##  Class :character   Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character  
##
```

```
##
##
##
##   HouseStyle         OverallQual      OverallCond      YearBuilt
##  Length:1460        Min.   : 1.000   Min.   :1.000   Min.   :1872
##  Class :character   1st Qu.: 5.000   1st Qu.:5.000   1st Qu.:1954
##  Mode  :character   Median : 6.000   Median :5.000   Median :1973
##                     Mean   : 6.099   Mean   :5.575   Mean   :1971
##                     3rd Qu.: 7.000   3rd Qu.:6.000   3rd Qu.:2000
##                     Max.   :10.000   Max.   :9.000   Max.   :2010
##
##   YearRemodAdd   RoofStyle           RoofMatl          Exterior1st
##  Min.   :1950   Length:1460        Length:1460        Length:1460
##  1st Qu.:1967   Class :character   Class :character   Class :character
##  Median :1994   Mode  :character   Mode  :character   Mode  :character
##  Mean   :1985
##  3rd Qu.:2004
##  Max.   :2010
##
##  Exterior2nd         MasVnrType          MasVnrArea      ExterQual
##  Length:1460        Length:1460        Min.   :   0.0   Length:1460
##  Class :character   Class :character   1st Qu.:   0.0   Class :character
##  Mode  :character   Mode  :character   Median :   0.0   Mode  :character
##                                        Mean   : 103.7
##                                        3rd Qu.: 166.0
##                                        Max.   :1600.0
##                                        NA's   :8
##   ExterCond          Foundation          BsmtQual           BsmtCond
##  Length:1460        Length:1460        Length:1460        Length:1460
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  BsmtExposure       BsmtFinType1         BsmtFinSF1     BsmtFinType2
##  Length:1460        Length:1460        Min.   :   0.0   Length:1460
##  Class :character   Class :character   1st Qu.:   0.0   Class :character
##  Mode  :character   Mode  :character   Median : 383.5   Mode  :character
##                                        Mean   : 443.6
##                                        3rd Qu.: 712.2
##                                        Max.   :5644.0
##
##    BsmtFinSF2        BsmtUnfSF       TotalBsmtSF       Heating
##  Min.   :   0.00   Min.   :   0.0   Min.   :   0.0   Length:1460
##  1st Qu.:   0.00   1st Qu.: 223.0   1st Qu.: 795.8   Class :character
##  Median :   0.00   Median : 477.5   Median : 991.5   Mode  :character
##  Mean   :  46.55   Mean   : 567.2   Mean   :1057.4
##  3rd Qu.:   0.00   3rd Qu.: 808.0   3rd Qu.:1298.2
##  Max.   :1474.00   Max.   :2336.0   Max.   :6110.0
##
##   HeatingQC          CentralAir         Electrical           1stFlrSF
##  Length:1460        Length:1460        Length:1460        Min.   : 334
##  Class :character   Class :character   Class :character   1st Qu.: 882
```

```
##  Mode  :character   Mode  :character   Mode  :character   Median :1087
##                                                           Mean   :1163
##                                                           3rd Qu.:1391
##                                                           Max.   :4692
##
##     2ndFlrSF     LowQualFinSF       GrLivArea     BsmtFullBath
##  Min.   :   0   Min.   :  0.000   Min.   : 334   Min.   :0.0000
##  1st Qu.:   0   1st Qu.:  0.000   1st Qu.:1130   1st Qu.:0.0000
##  Median :   0   Median :  0.000   Median :1464   Median :0.0000
##  Mean   : 347   Mean   :  5.845   Mean   :1515   Mean   :0.4253
##  3rd Qu.: 728   3rd Qu.:  0.000   3rd Qu.:1777   3rd Qu.:1.0000
##  Max.   :2065   Max.   :572.000   Max.   :5642   Max.   :3.0000
##
##   BsmtHalfBath        FullBath        HalfBath       BedroomAbvGr
##  Min.   :0.00000   Min.   :0.000   Min.   :0.0000   Min.   :0.000
##  1st Qu.:0.00000   1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:2.000
##  Median :0.00000   Median :2.000   Median :0.0000   Median :3.000
##  Mean   :0.05753   Mean   :1.565   Mean   :0.3829   Mean   :2.866
##  3rd Qu.:0.00000   3rd Qu.:2.000   3rd Qu.:1.0000   3rd Qu.:3.000
##  Max.   :2.00000   Max.   :3.000   Max.   :2.0000   Max.   :8.000
##
##   KitchenAbvGr   KitchenQual         TotRmsAbvGrd     Functional
##  Min.   :0.000   Length:1460        Min.   : 2.000   Length:1460
##  1st Qu.:1.000   Class :character   1st Qu.: 5.000   Class :character
##  Median :1.000   Mode  :character   Median : 6.000   Mode  :character
##  Mean   :1.047                      Mean   : 6.518
##  3rd Qu.:1.000                      3rd Qu.: 7.000
##  Max.   :3.000                      Max.   :14.000
##
##    Fireplaces    FireplaceQu         GarageType         GarageYrBlt
##  Min.   :0.000   Length:1460        Length:1460        Min.   :1900
##  1st Qu.:0.000   Class :character   Class :character   1st Qu.:1961
##  Median :1.000   Mode  :character   Mode  :character   Median :1980
##  Mean   :0.613                                         Mean   :1979
##  3rd Qu.:1.000                                         3rd Qu.:2002
##  Max.   :3.000                                         Max.   :2010
##                                                        NA's   :81
##  GarageFinish         GarageCars      GarageArea      GarageQual
##  Length:1460        Min.   :0.000   Min.   :   0.0   Length:1460
##  Class :character   1st Qu.:1.000   1st Qu.: 334.5   Class :character
##  Mode  :character   Median :2.000   Median : 480.0   Mode  :character
##                     Mean   :1.767   Mean   : 473.0
##                     3rd Qu.:2.000   3rd Qu.: 576.0
##                     Max.   :4.000   Max.   :1418.0
##
##   GarageCond         PavedDrive          WoodDeckSF      OpenPorchSF
##  Length:1460        Length:1460        Min.   :  0.00   Min.   :  0.00
##  Class :character   Class :character   1st Qu.:  0.00   1st Qu.:  0.00
##  Mode  :character   Mode  :character   Median :  0.00   Median : 25.00
##                                        Mean   : 94.24   Mean   : 46.66
##                                        3rd Qu.:168.00   3rd Qu.: 68.00
##                                        Max.   :857.00   Max.   :547.00
##
##  EnclosedPorch      3SsnPorch       ScreenPorch        PoolArea
```

```
##  Min.   :  0.00   Min.   :  0.00   Min.   :  0.00   Min.   :  0.000
##  1st Qu.:  0.00   1st Qu.:  0.00   1st Qu.:  0.00   1st Qu.:  0.000
##  Median :  0.00   Median :  0.00   Median :  0.00   Median :  0.000
##  Mean   : 21.95   Mean   :  3.41   Mean   : 15.06   Mean   :  2.759
##  3rd Qu.:  0.00   3rd Qu.:  0.00   3rd Qu.:  0.00   3rd Qu.:  0.000
##  Max.   :552.00   Max.   :508.00   Max.   :480.00   Max.   :738.000
##
##     PoolQC             Fence            MiscFeature           MiscVal
##  Length:1460        Length:1460        Length:1460        Min.   :    0.00
##  Class :character   Class :character   Class :character   1st Qu.:    0.00
##  Mode  :character   Mode  :character   Mode  :character   Median :    0.00
##                                                            Mean   :   43.49
##                                                            3rd Qu.:    0.00
##                                                            Max.   :15500.00
##
##      MoSold           YrSold       SaleType          SaleCondition
##  Min.   : 1.000   Min.   :2006   Length:1460        Length:1460
##  1st Qu.: 5.000   1st Qu.:2007   Class :character   Class :character
##  Median : 6.000   Median :2008   Mode  :character   Mode  :character
##  Mean   : 6.322   Mean   :2008
##  3rd Qu.: 8.000   3rd Qu.:2009
##  Max.   :12.000   Max.   :2010
##
##    SalePrice
##  Min.   : 34900
##  1st Qu.:129975
##  Median :163000
##  Mean   :180921
##  3rd Qu.:214000
##  Max.   :755000
##
```

Let us obtain a more comprehensive review of what each column possesses. We can employ the Hmisc::describe() to obtain this comprehensive view.

```
hd<-Hmisc::describe(training)
```

Because the output of above function runs over 14 pages, I have commented out the printing of the return value but we will be using this variable, **(hd)**, later in the analysis of missing values. The 14 pages contain comprehensive output for each of the 81 fields contained in the data set including information, such as count of missing values, unique values, mean, lowest values, highest values, etc.

For example, the Id field has the following characteristics

```
hd$Id
```

```
## Id
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1460        0     1460        1    730.5      487    73.95   146.90
##      .25      .50      .75      .90      .95
##   365.75   730.50  1095.25  1314.10  1387.05
##
## lowest :    1    2    3    4    5, highest: 1456 1457 1458 1459 1460
```

Similarly, the MSZoning field has the following characteristics

```
hd$MSZoning
```

```
## MSZoning
##        n  missing distinct
##     1460        0        5
##
## lowest : C (all) FV      RH      RL      RM
## highest: C (all) FV      RH      RL      RM
##
## Value      C (all)      FV      RH      RL      RM
## Frequency       10      65      16    1151     218
## Proportion   0.007   0.045   0.011   0.788   0.149
```

We see that most houses are built in the area of *(RL)* Residential Low Density(1151 houses), followed by *(RM)* Residential Medium Density(218 houses). Few houses are built in Commercial *(C)*, Floating Village *(FV)* and Residential High Density *(RH)*.

Since a large amount of houses comes to the categories of Residential Low Density and Residential Medium Density, these two areas should be paid more attention for housing price analysis.

Unique Values

```
# Extract unique values for each column in the dataset
df_unique_values <- lapply(training, unique)
```

Get a count of unique values in each column. Then print the count for unique values in each column sorted in ascending order of count

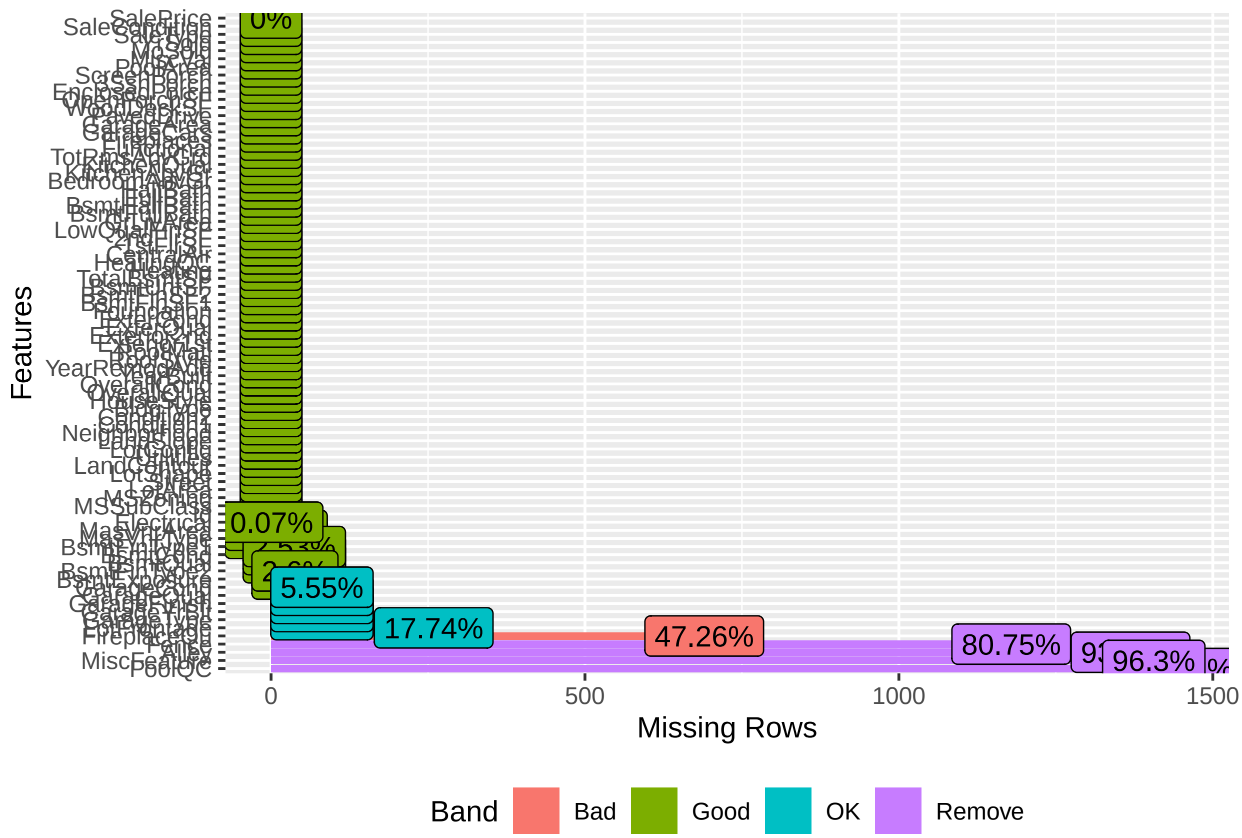
```r
k <- lengths(df_unique_values)
sort.int(k)
```

```
##        Street    Utilities    CentralAir         Alley     LandSlope 
##             2            2             2             3             3 
##  BsmtHalfBath     HalfBath    PavedDrive      LotShape   LandContour 
##             3            3             3             4             4 
##     ExterQual  BsmtFullBath      FullBath  KitchenAbvGr   KitchenQual 
##             4            4             4             4             4 
##    Fireplaces  GarageFinish        PoolQC      MSZoning     LotConfig 
##             4            4             4             5             5 
##      BldgType    MasVnrType     ExterCond      BsmtQual      BsmtCond 
##             5            5             5             5             5 
##  BsmtExposure     HeatingQC    GarageCars         Fence   MiscFeature 
##             5            5             5             5             5 
##        YrSold     RoofStyle    Foundation       Heating    Electrical 
##             5            6             6             6             6 
##   FireplaceQu    GarageQual    GarageCond SaleCondition  BsmtFinType1 
##             6            6             6             6             7 
##  BsmtFinType2    Functional    GarageType    Condition2    HouseStyle 
##             7            7             7             8             8 
##      RoofMatl  BedroomAbvGr      PoolArea    Condition1   OverallCond 
##             8            8             8             9             9 
##      SaleType   OverallQual  TotRmsAbvGrd        MoSold    MSSubClass 
##             9           10            12            12            15 
##   Exterior1st   Exterior2nd     3SsnPorch       MiscVal  LowQualFinSF 
##            15           16            20            21            24 
##  Neighborhood  YearRemodAdd   ScreenPorch   GarageYrBlt   LotFrontage 
##            25           61            76            98           111 
##     YearBuilt EnclosedPorch    BsmtFinSF2   OpenPorchSF    WoodDeckSF 
##           112          120           144           202           274 
##    MasVnrArea      2ndFlrSF    GarageArea    BsmtFinSF1     SalePrice 
##           328          417           441           637           663 
##   TotalBsmtSF      1stFlrSF     BsmtUnfSF     GrLivArea       LotArea 
##           721          753           780           861          1073 
##            Id 
##          1460
```

Missing Data

Our key objective is to predict SalePrice. So let us first check to see if there are any rows in the dataset that are missing values. Let us visualize missing profile for each feature in our training set

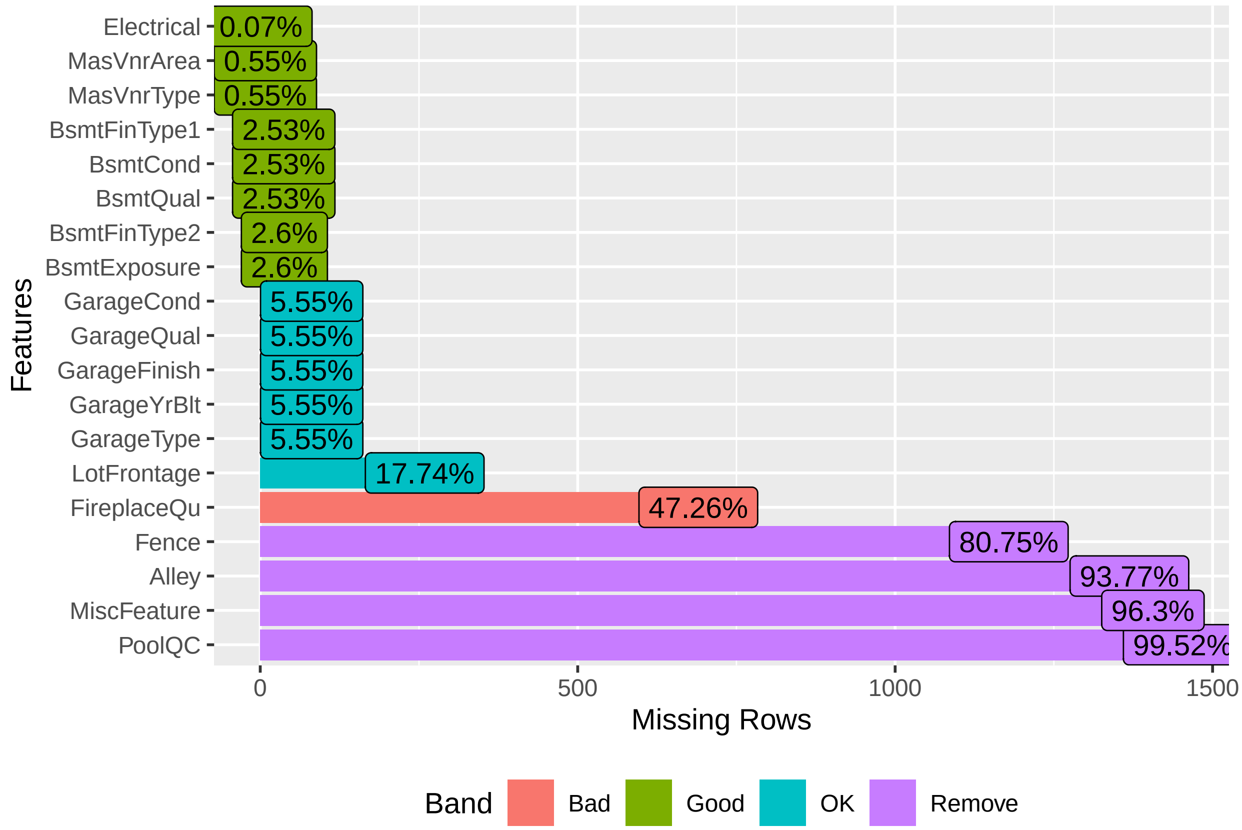
```r
plot_missing(training)
```



We see that most of the features have no missing values but there are some that do have some missing values while others have a huge amount of missing values. Because it hard to visualize the above graph with the one having 0 missing values, we will now focus only on the ones that do have missing values.

Plot only those features with missing values

```r
plot_missing(training,missing_only = TRUE)
```



We find that the following features have a ton of missing values:

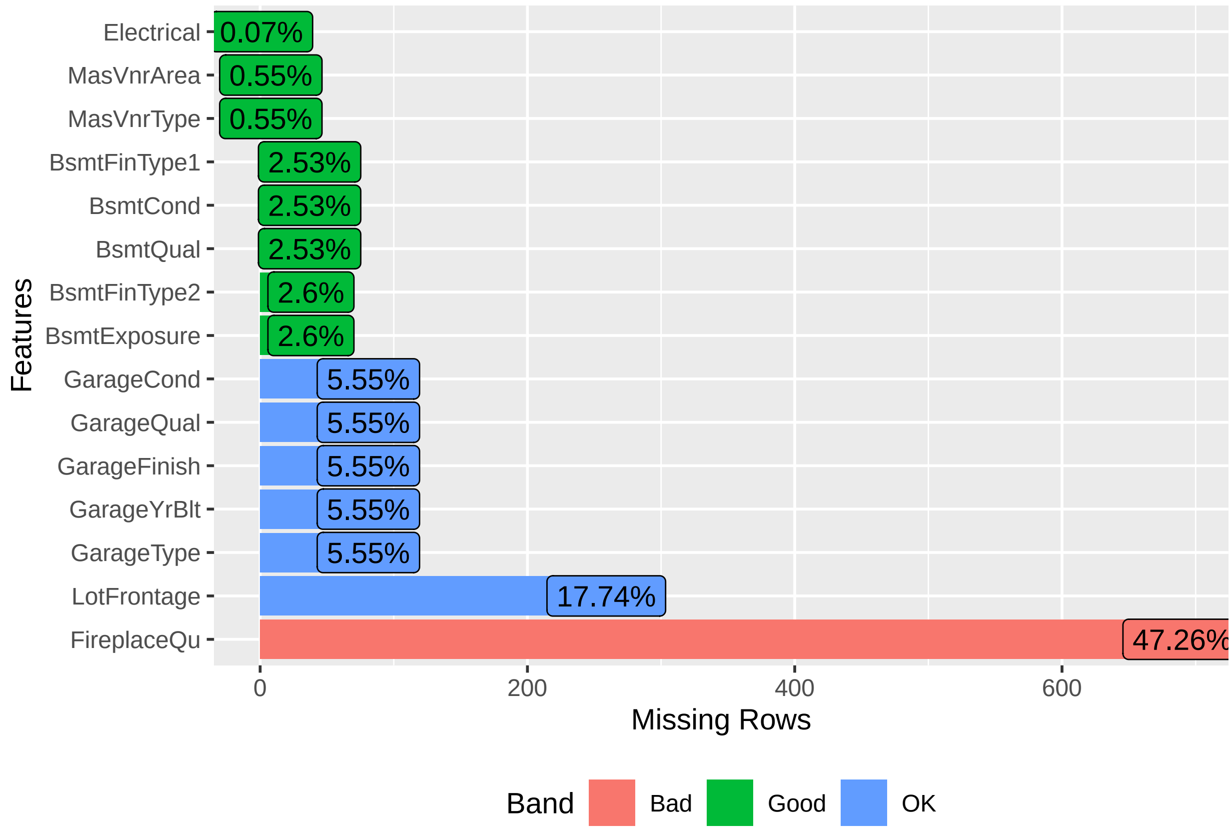
1. PoolQC (99.52% missing)
2. MiscFeature (96.3%)
3. Alley (93.77%)
4. Fence (80.75%)

We are better off dropping these columns from our analysis as these won't help much with our analysis.

```
final_training <- drop_columns(training,
                               c("PoolQC","MiscFeature","Alley","Fence"))
```

Let us plot the missing values again.

```
plot_missing(final_training,missing_only = TRUE)
```



The **FireplaceQu** field has 47.26% missing values and a recommendation that it is "Bad" for analysis. Let us get more details. We will check the Hmisc::describe done earlier for more details about **FireplaceQu**

```
hd$FireplaceQu
```

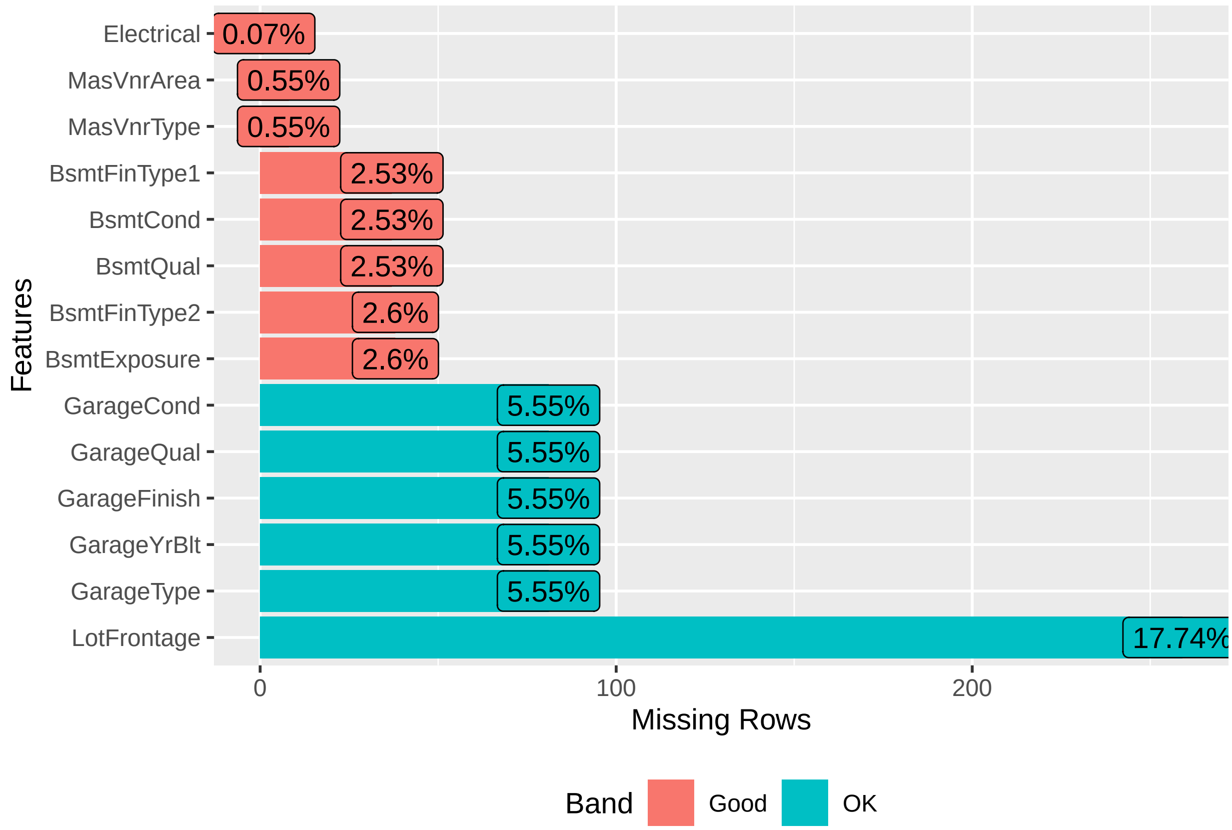
```
## FireplaceQu
##        n  missing distinct
##      770      690        5
##
## lowest : Ex Fa Gd Po TA, highest: Ex Fa Gd Po TA
##
## Value         Ex    Fa    Gd    Po    TA
## Frequency     24    33   380    20   313
## Proportion 0.031 0.043 0.494 0.026 0.406
```

We find that of the 1460 records in the training set, **690 records have missing values for FireplaceQu** and it has categorical data. We are better off dropping this column as well from our analysis.

```
final_training <- drop_columns(final_training,"FireplaceQu")
```

We will make one final plot for missing values.

```
plot_missing(final_training,missing_only = TRUE)
```



The plot looks good to proceed now. Let us review the number of remaining columns in our final_training set which will be used for further analysis including the splitting for **training_set** and **test_set** for use against one or more models.

```
dim(final_training)
```

```
## [1] 1460   76
```

The final_training contains 1460 records of 76 features.

Rename 2 of the features that begin with a number since certain functions cannot deal with variables that begin with a number.

```
names(final_training)[names(final_training) == "1stFlrSF"] <- "First_Floor_SF"
names(final_training)[names(final_training) == "2ndFlrSF"] <- "Second_Floor_SF"

# Perform the same change for validation set
names(validation)[names(validation) == "1stFlrSF"] <- "First_Floor_SF"
names(validation)[names(validation) == "2ndFlrSF"] <- "Second_Floor_SF"
```

Data Pre-processing

To avoid the issue of overfitting by building different models and subjecting them to the same validation/final hold-out set (as confirmed with the TA/edx Staff in edX discussion forum), we will split our current University provided edx training set into a train and test set.

```
library(caTools)
library(dplyr)
library(tidyr)
set.seed(123)

############################################
# Generate the Training and Test set from pre-processed training set,
# final_training, to avoid overfitting
############################################

#We will go for 80-20 split of Train and Test data
split = sample.split(final_training$SalePrice, SplitRatio = 0.8)

# New Training Set to be used by all models
training_set = subset(final_training, split == TRUE) #  1258 obs. of 76 variables (80% of training set)

# New Test Set to be used by all models
test_set = subset(final_training, split == FALSE)#  202 obs. of 76 variables (20% of training set)
```

Effectively, our new test set has 20% of original training data and new training set has 80% of training data

For **ALL** our models, we will be using these new training_set and test_set to make predictions and compare RMSE values.

For the **CHOSEN** model, we will additionally subject that to the original validation set to make predictions on validation data set that **does not contain our outcome variable, SalePrice**. This way, we will NOT be overfitting and be in compliance with the requirements confirmed by the TA in the edx discussion forum.

Data Exploration and Visualization

We begin exploring the different features of the dataset including the outcome **SalePrice** that we want to predict. Understanding what the different dataset features contain helps us gain meaningful insights about how each attribute/feature contributes to our data analysis in addition to determining the appropriate modeling technique.

Categorical Features

Upon close inspection of the feature description provided in data_description.txt and examining the dataset values, we identify the following 38 categorical features.

```
categorical_features<-c("BldgType","BsmtCond","BsmtExposure","BsmtFinType1",
                        "BsmtFinType2","BsmtQual","CentralAir","Condition1",
                        "Condition2","Electrical","ExterCond","Exterior1st",
                        "Exterior2nd","ExterQual","Foundation","Functional",
                        "GarageCond","GarageFinish","GarageQual","GarageType",
                        "Heating","HeatingQC","HouseStyle","KitchenQual",
                        "LandContour","LandSlope","LotConfig","LotShape",
```

```r
                         "MasVnrType","MSZoning","Neighborhood","PavedDrive",
                         "RoofMatl","RoofStyle","SaleCondition","SaleType",
                         "Street","Utilities")

categorical_features
```

```
##  [1] "BldgType"      "BsmtCond"      "BsmtExposure"  "BsmtFinType1"
##  [5] "BsmtFinType2"  "BsmtQual"      "CentralAir"    "Condition1"
##  [9] "Condition2"    "Electrical"    "ExterCond"     "Exterior1st"
## [13] "Exterior2nd"   "ExterQual"     "Foundation"    "Functional"
## [17] "GarageCond"    "GarageFinish"  "GarageQual"    "GarageType"
## [21] "Heating"       "HeatingQC"     "HouseStyle"    "KitchenQual"
## [25] "LandContour"   "LandSlope"     "LotConfig"     "LotShape"
## [29] "MasVnrType"    "MSZoning"      "Neighborhood"  "PavedDrive"
## [33] "RoofMatl"      "RoofStyle"     "SaleCondition" "SaleType"
## [37] "Street"        "Utilities"
```

Three of the above 38 features include **numeric values** that will need to be **one-hot encoded** later in the modeling and analysis section below. The 3 features involved are:

1. MSSubClass
2. OverallCond
3. OverallQual

```r
class(final_training$MSSubClass)
```

```
## [1] "numeric"
```

```r
class(final_training$OverallCond)
```

```
## [1] "numeric"
```

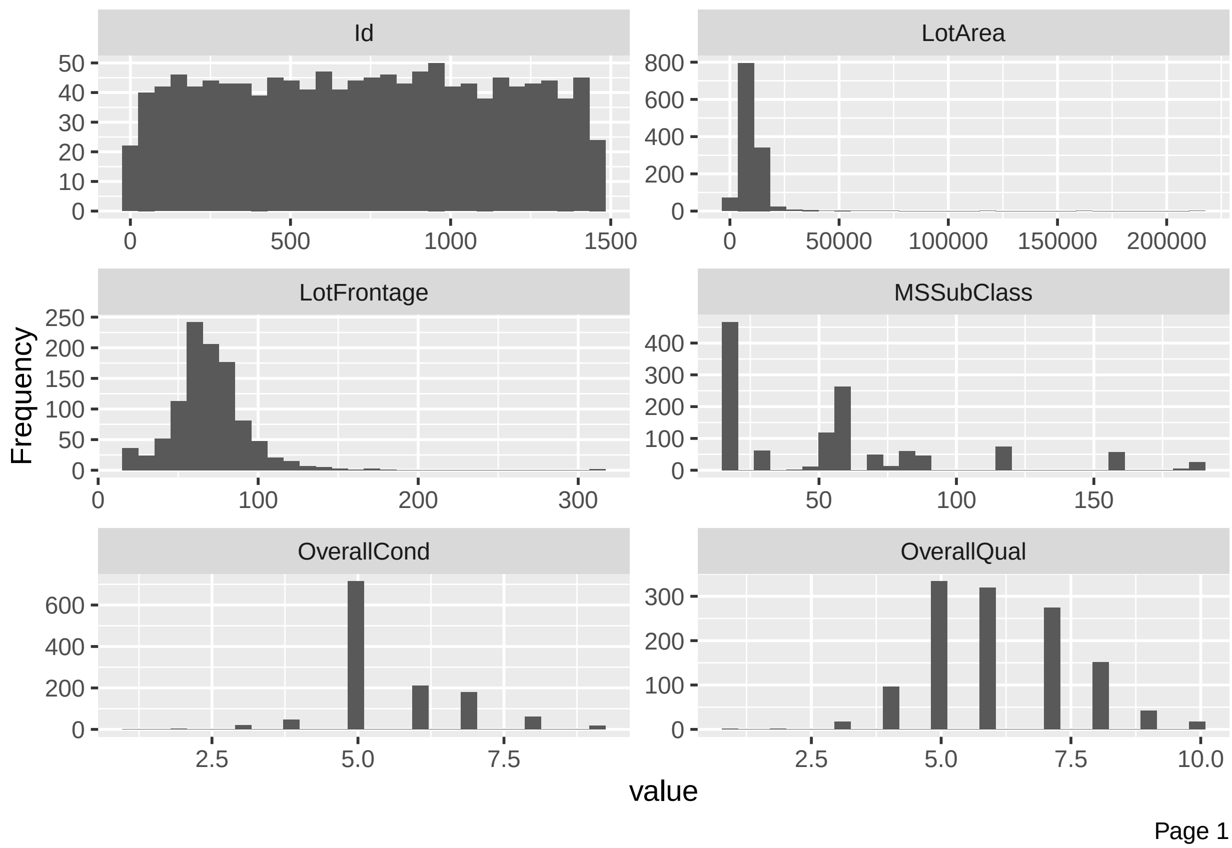
```r
class(final_training$OverallQual)
```

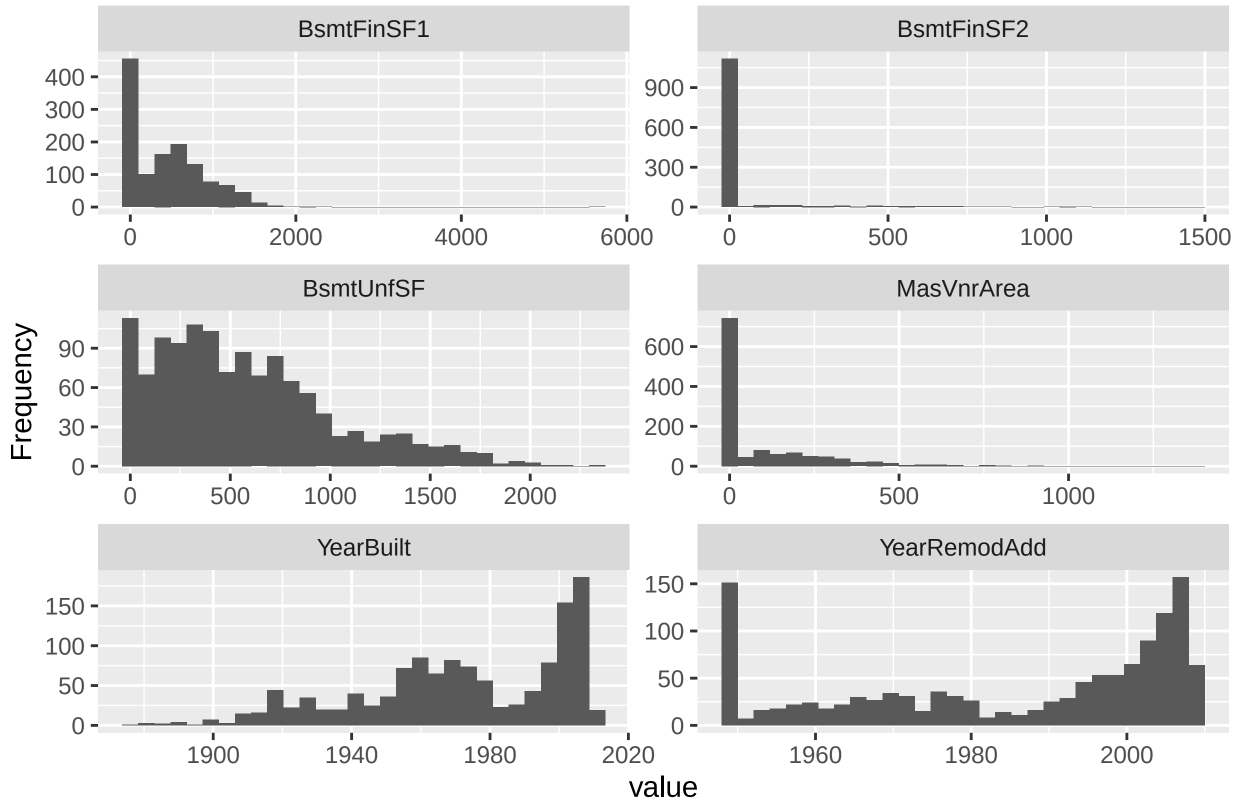
```
## [1] "numeric"
```

Visualize Histogram of Continuous Variables

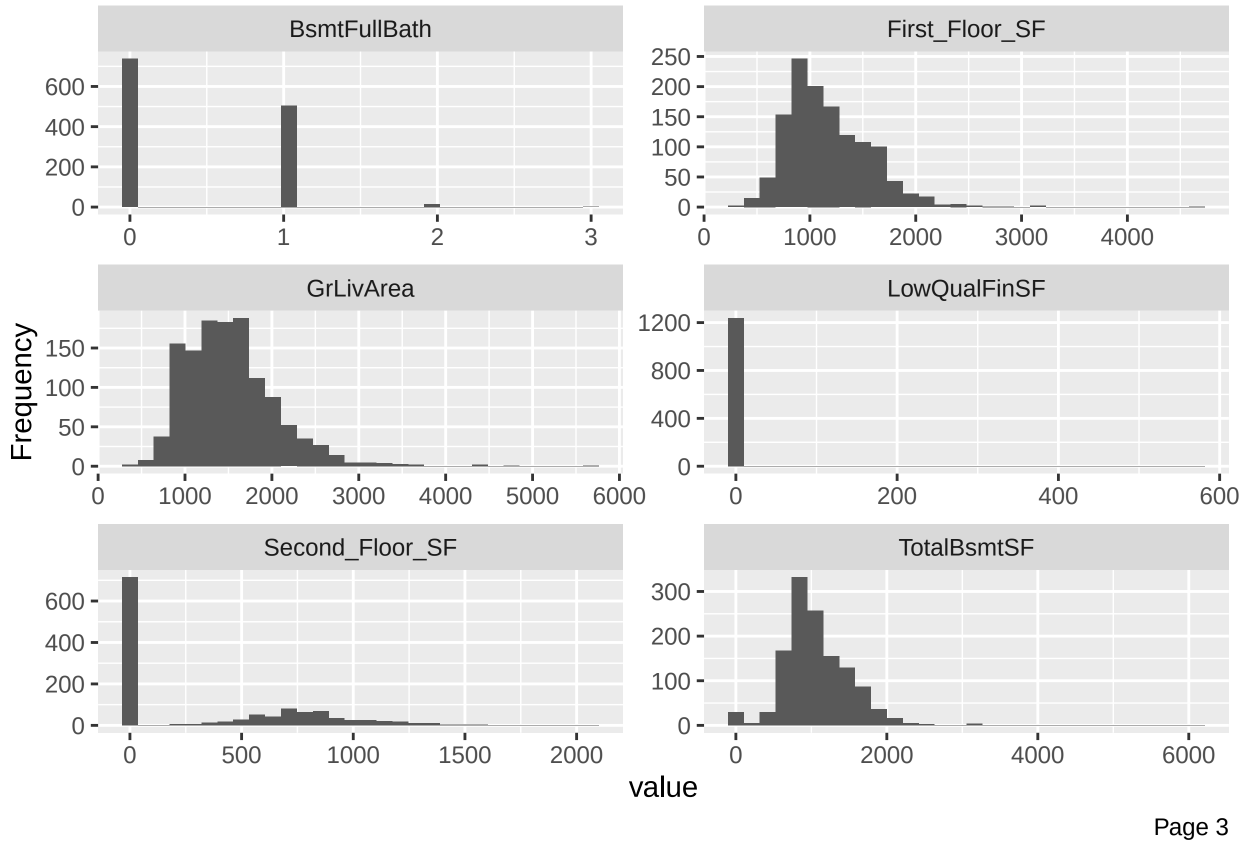
Plotting the histogram for continuous variables, including our outcome **SalePrice** helps us visualize the distribution of these variables. We will use the plot_histogram() function in the DataExplorer package. Because we have nearly 38 continuous features in the data set, the output of this command will be 38 histograms that will run into multiple pages. Performing this comprehensive visualization helps us better understand our data.

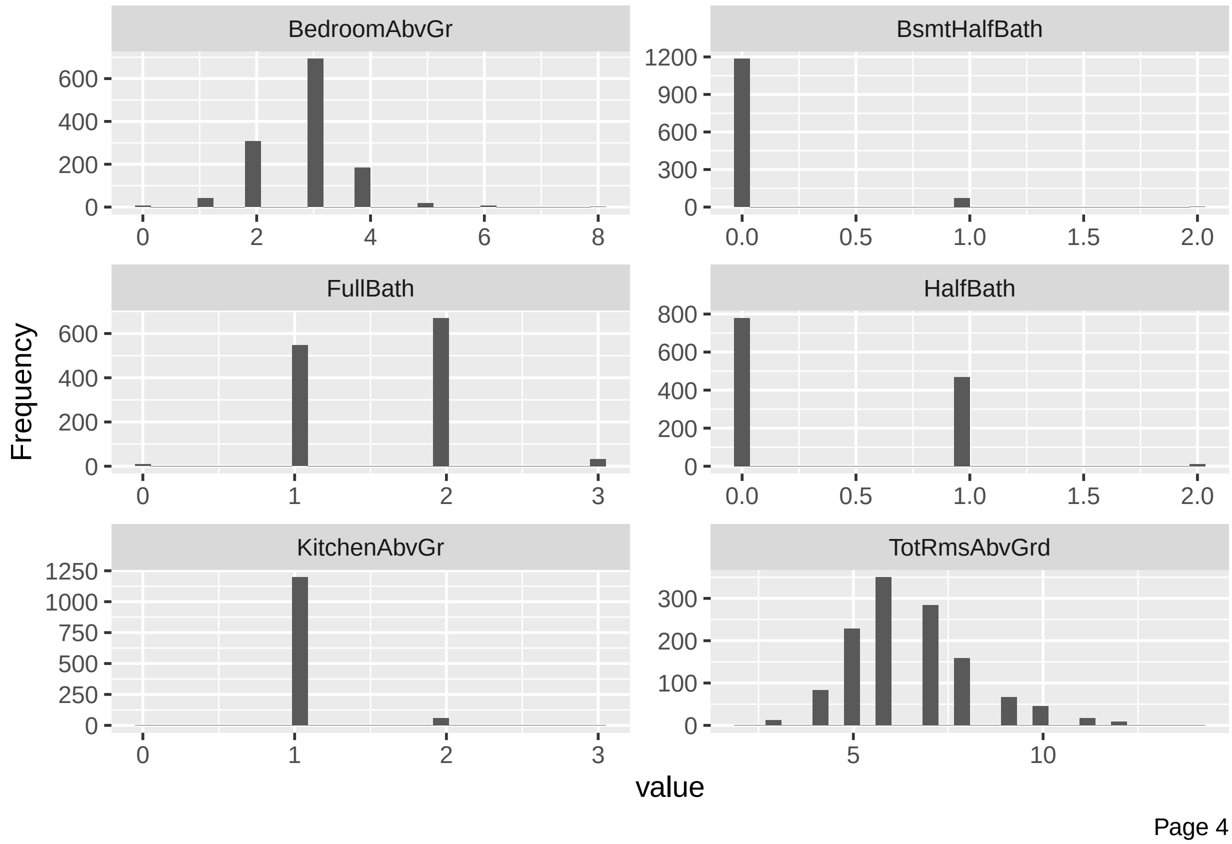
```
options(repr.plot.width = 4, repr.plot.height = 3,scipen=999)
plot_histogram(training_set,nrow=3L,ncol=2L)
```

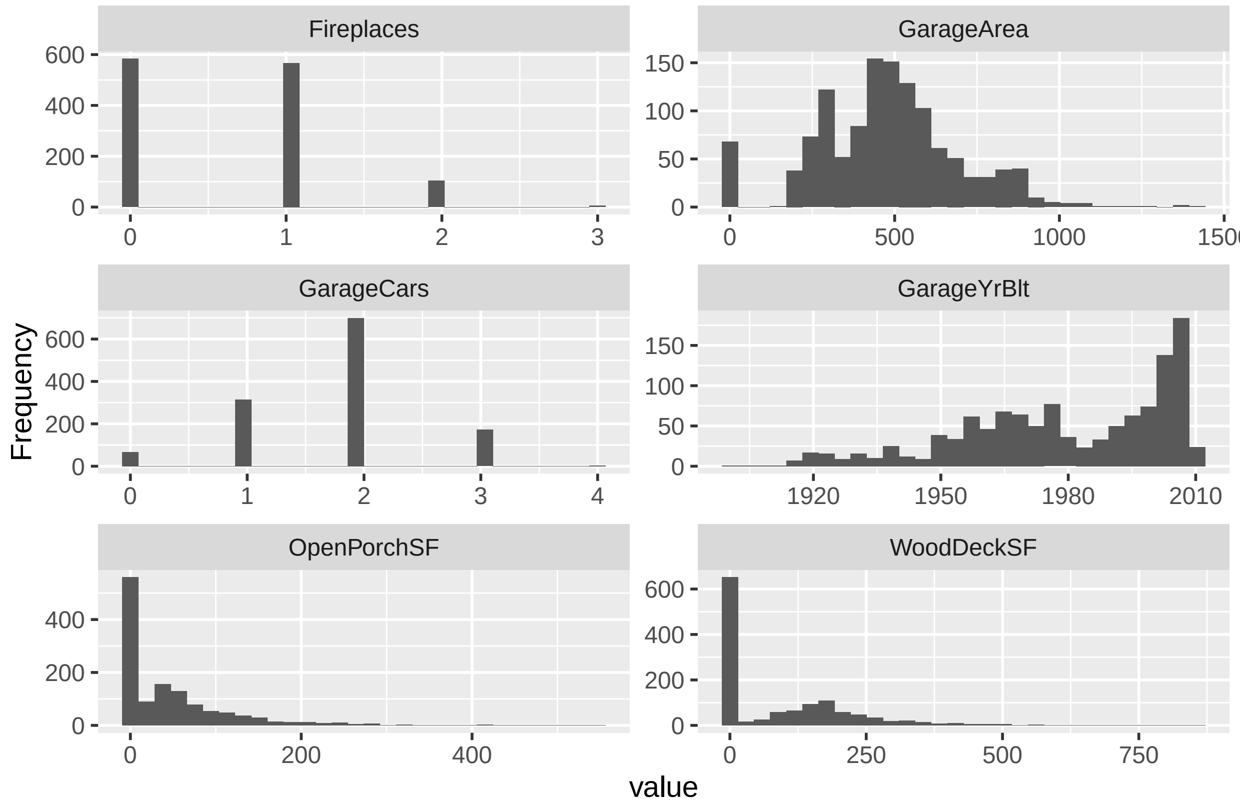



BsmtFinSF1
BsmtFinSF2
BsmtUnfSF
MasVnrArea
YearBuilt
YearRemodAdd
Frequency
value

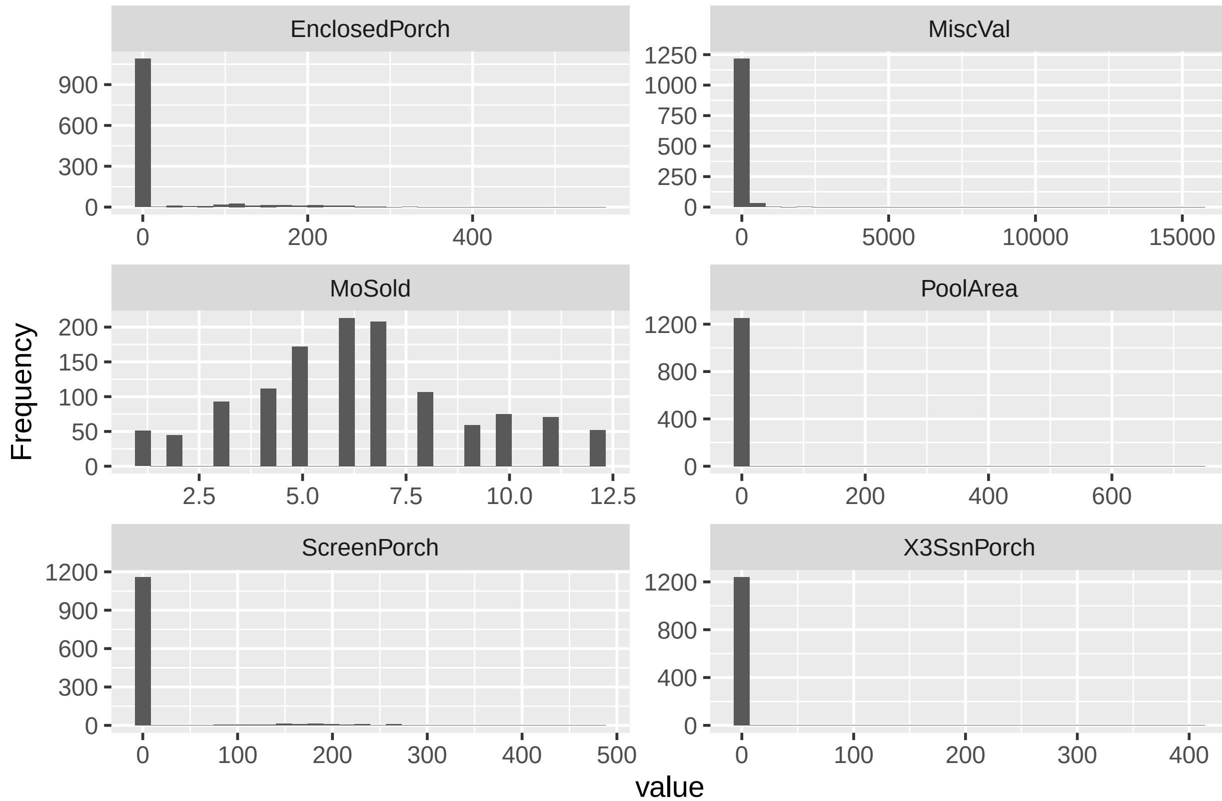
Page 2


BsmtFullBath
First_Floor_SF
GrLivArea
LowQualFinSF
Second_Floor_SF
TotalBsmtSF
Frequency
value
Page 3


BedroomAbvGr
BsmtHalfBath
FullBath
HalfBath
KitchenAbvGr
TotRmsAbvGrd
Frequency
value
Page 4

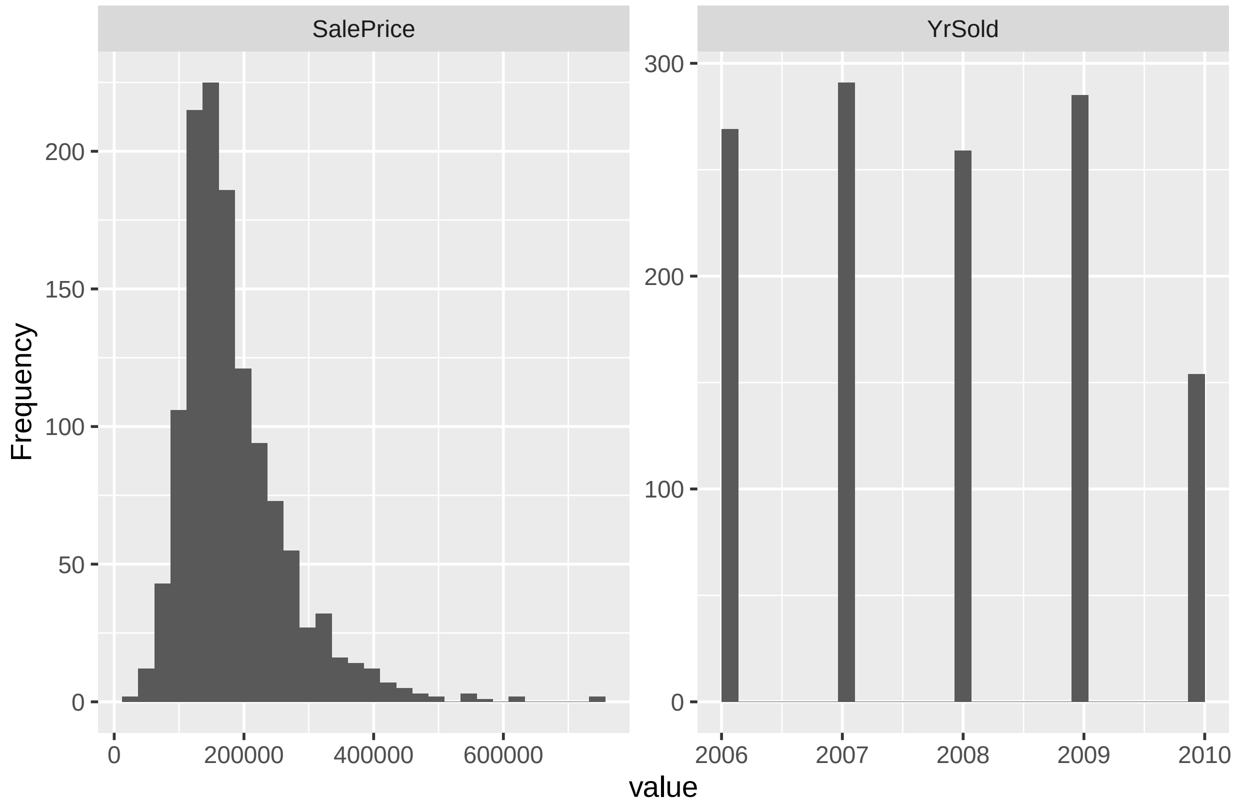

Fireplaces
GarageArea
GarageCars
GarageYrBlt
OpenPorchSF
WoodDeckSF
Frequency
value

Page 5



EnclosedPorch
MiscVal
MoSold
PoolArea
ScreenPorch
X3SsnPorch
Frequency
value

Page 6

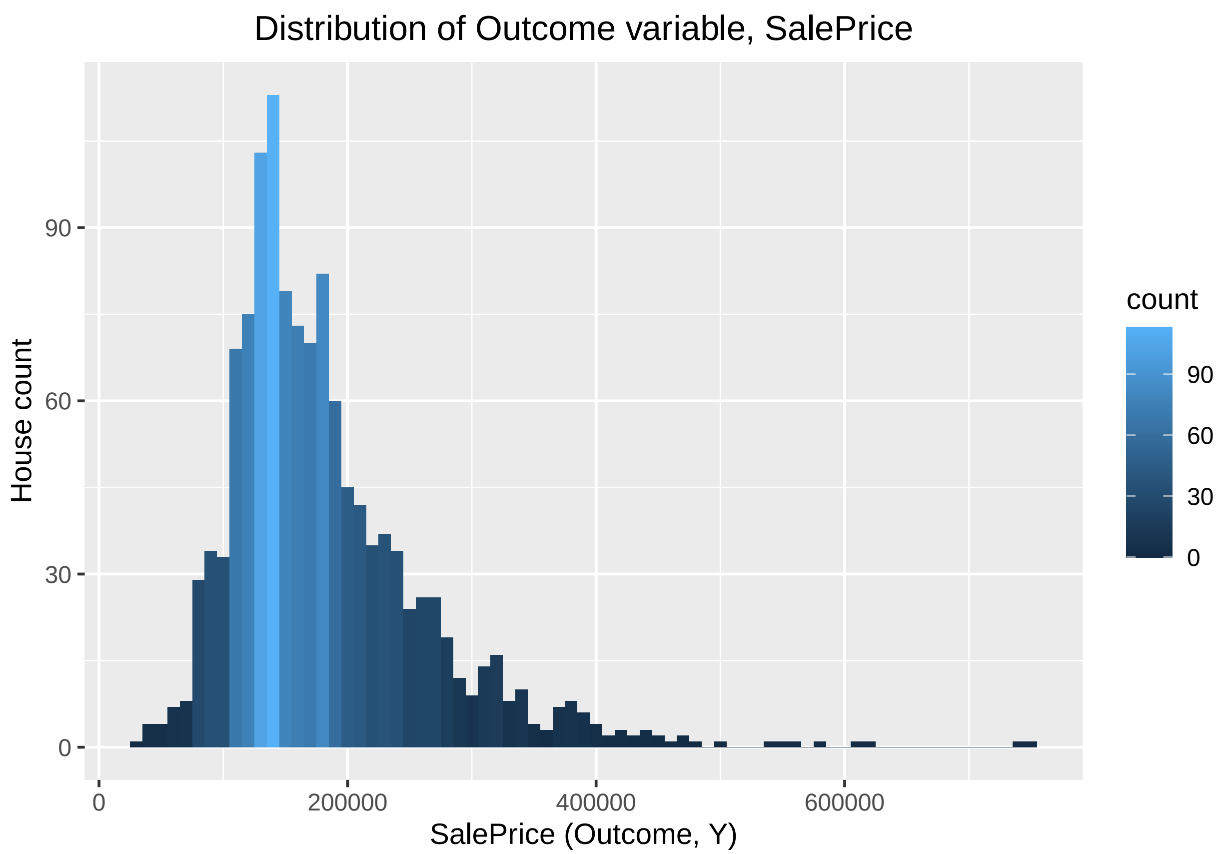


Page 7

Insights on Outcome: SalePrice Distribution

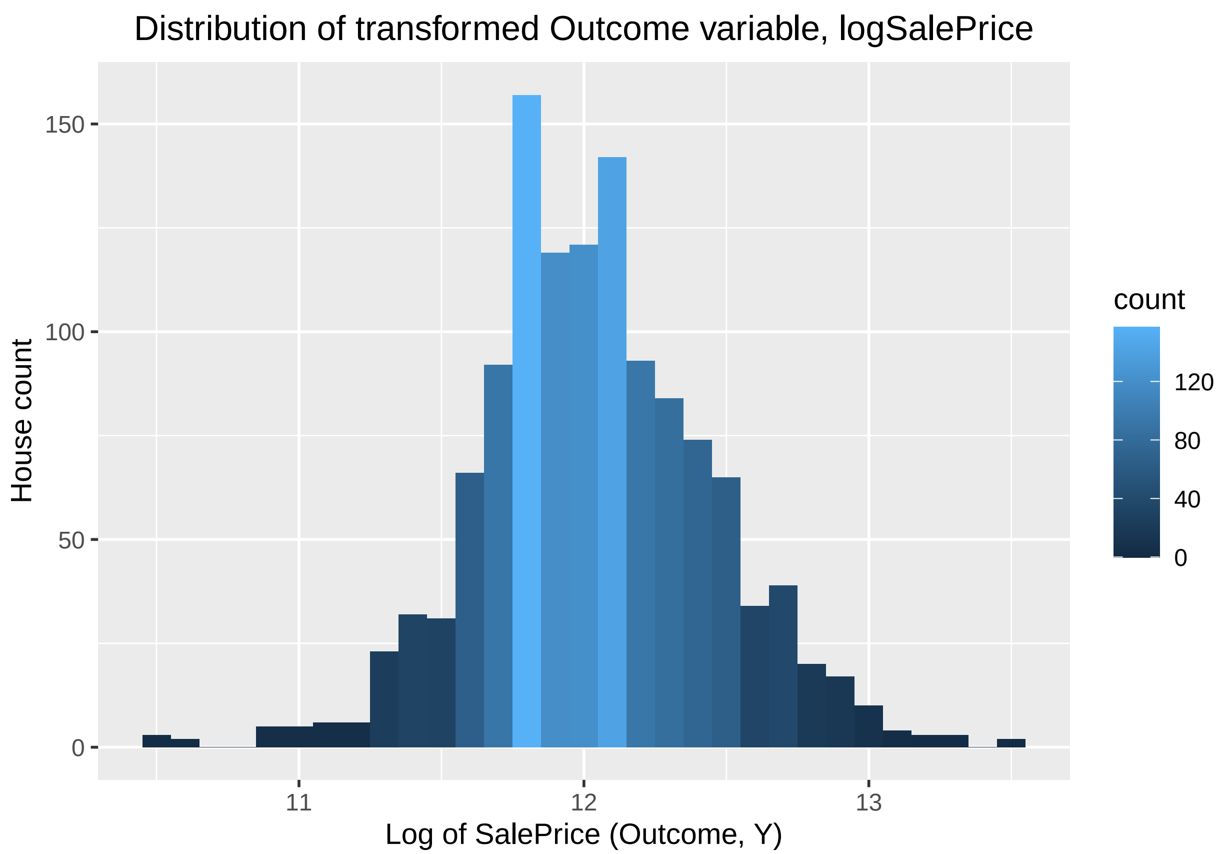
One of the assumptions of Linear Regression is that for any fixed value of X, outcome, Y, is normally distributed. Let us confirm if that is indeed the case with our outcome variable, SalePrice.

```
# options(scipen=999)
ggplot(training_set, aes(x = SalePrice, fill = ..count..)) +
  geom_histogram(binwidth = 10000) +
  ggtitle("Distribution of Outcome variable, SalePrice") +
  ylab("House count") +
  xlab("SalePrice (Outcome, Y)") +
  theme(plot.title = element_text(hjust = 0.5))
```



We see that distribution for SalePrice is skewed. To account for this discrepancy, let us take a log of SalePrice to adjust the distribution for our further analysis and plot the distribution again.

```
training_set$logSalePrice <- log(training_set$SalePrice)
options(scipen=10000)

ggplot(training_set, aes(x = logSalePrice, fill = ..count..)) +
  geom_histogram(binwidth = 0.1) +
  ggtitle("Distribution of transformed Outcome variable, logSalePrice") +
  ylab("House count") +
  xlab("Log of SalePrice (Outcome, Y)") +
  theme(plot.title = element_text(hjust = 0.5))
```



Insights on Categorical Feature: MSZoning

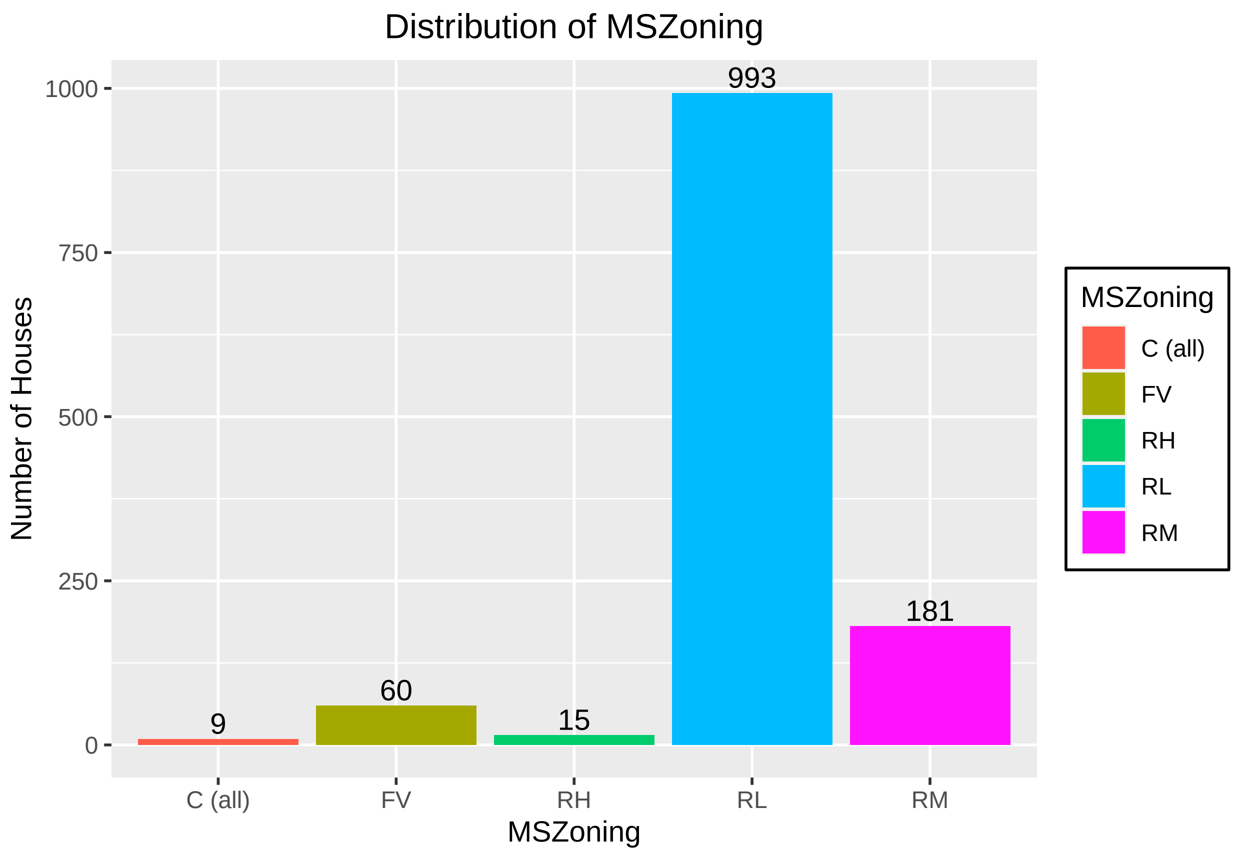
MSZoning identifies the general zoning classification of the sale using the following scheme.

- A: Agriculture
- C: Commercial
- FV: Floating Village Residential
- I: Industrial
- RH: Residential High Density
- RL: Residential Low Density
- RP: Residential Low Density Park
- RM: Residential Medium Density

Let us see how the houses are distributed across these different zones.

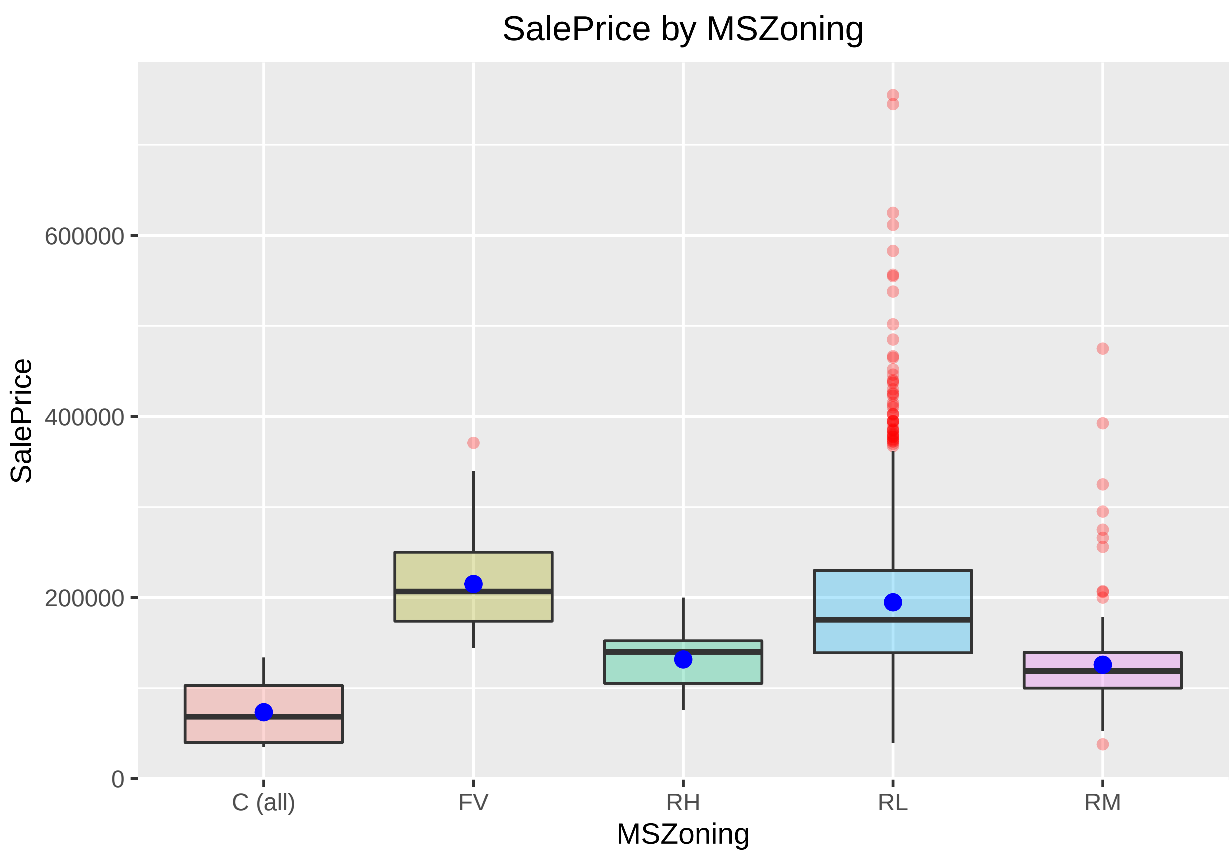
```
ggplot(training_set, aes(x = MSZoning, fill = MSZoning )) +
      geom_bar()+
      scale_fill_hue(c = 150)+
      ggtitle("Distribution of MSZoning")+
      xlab("MSZoning")+
      ylab("Number of Houses")+
      theme(plot.title = element_text(hjust = 0.5),
            legend.position="right",
            legend.background = element_rect(fill="grey100", size=0.5,
```

```
                                          linetype="solid",colour ="black"))+
    geom_text(stat='count',aes(label=..count..),vjust=-0.25)
```



We see that most homes fall under the Residential Low Density Zone (RL) followed by Residential Medium Density (RM). Let us see how the SalePrice is distributed across these different zones.

```
options(repr.plot.width=9, repr.plot.height=6)
# boxplot of SalePrice by MSZoning
# Display average value of SalePrice as a BLUE dot
ggplot(training_set, aes(MSZoning,SalePrice, fill=MSZoning)) +
  geom_boxplot(alpha=0.3,outlier.colour = "red") +
  stat_summary(fun=mean, geom="point", shape=20, size=4, color="blue", fill="red")+
  theme(legend.position="none")+
  ggtitle("SalePrice by MSZoning")+
  theme(plot.title = element_text(hjust = 0.5))
```

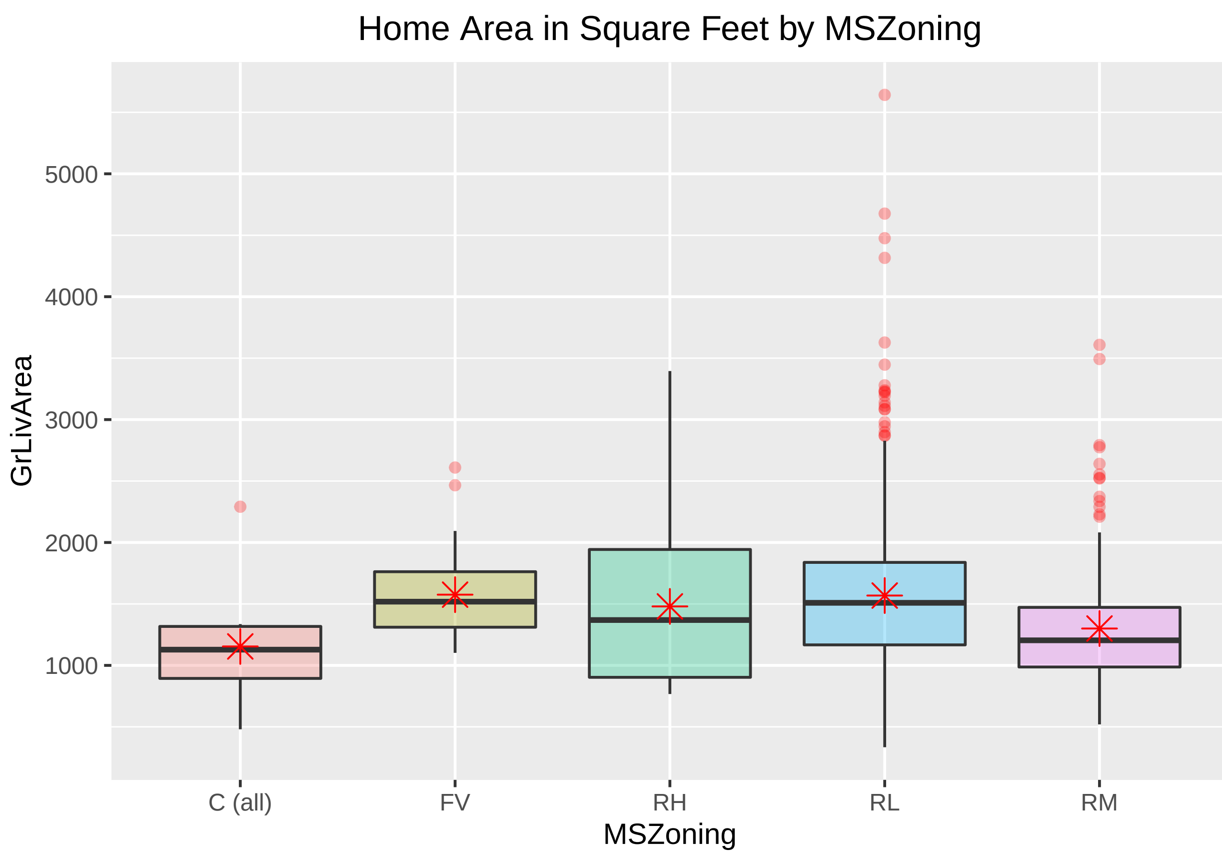


Looking at the above plot, we find that Floating Village (FV) has the highest average SalePrice followed by Residential Low Density (RL). This makes sense because Floating zones provide flexibility for developers, who can use the zone to obtain density bonuses, height extensions, etc., in exchange for meeting other requirements or goals in the floating zone, such as affordable housing, public transit, etc. The flexibility in development can be attributed to the higher cost.

Another possibility is that the square footage available in FV might be more than its Residential counterparts making it more desirable and hence, more expensive. We also find that the lowest average SalePrice is in the Commercial Zone. **Is it possible that the square foot area available is indeed a contributing factor to price?**

Let's find out by checking the average square foot of houses in each Zone. The area information for each home is available in the **GrLivArea** feature, which is described as "Above grade (ground) living area square feet".

```
options(repr.plot.width=9, repr.plot.height=6)
# boxplot of Home Area by MSZoning
# Display average square feet area as a RED asterisk

ggplot(training_set, aes(x=MSZoning,y=GrLivArea, fill=MSZoning)) +
  geom_boxplot(alpha=0.3, outlier.colour = "red") +
  stat_summary(fun=mean, geom="point", shape=8, size=4, color="red", fill="red")+
  theme(legend.position="none")+
  ggtitle("Home Area in Square Feet by MSZoning")+
  theme(plot.title = element_text(hjust = 0.5))
```



Let us also see the actual values for average square foot per zone.

```
library(plyr)

x<-ddply(training_set, .(MSZoning), summarize,  average_area=mean(GrLivArea))
x[order(-x$average_area),] # Display in descending order of average area
```

```
##   MSZoning average_area
## 2       FV     1575.533
## 4       RL     1568.366
## 3       RH     1479.867
## 5       RM     1300.011
## 1  C (all)     1154.222
```

The table above validates our finding that the average area of homes in Floating Village is higher (1,575 sq.ft.) than its Residential counterparts and the average area for homes in the Commercial zone is the lowest (1154 sq.ft.).

Insights on Categorical Feature: Building Type (BldgType)

BldgType represents the type of dwelling and it can take any of the following five values:

1. 1Fam: Single-family Detached
2. 2FmCon: Two-family Conversion; originally built as one-family dwelling
3. Duplx: Duplex
4. TwnhsE: Townhouse End Unit
5. TwnhsI: Townhouse Inside Unit

Just as we determined the average SalePrice by MSZoning in the prior section, we will now determine the average, maximum, and minimum SalePrice, and number of houses for each of the dwelling types.

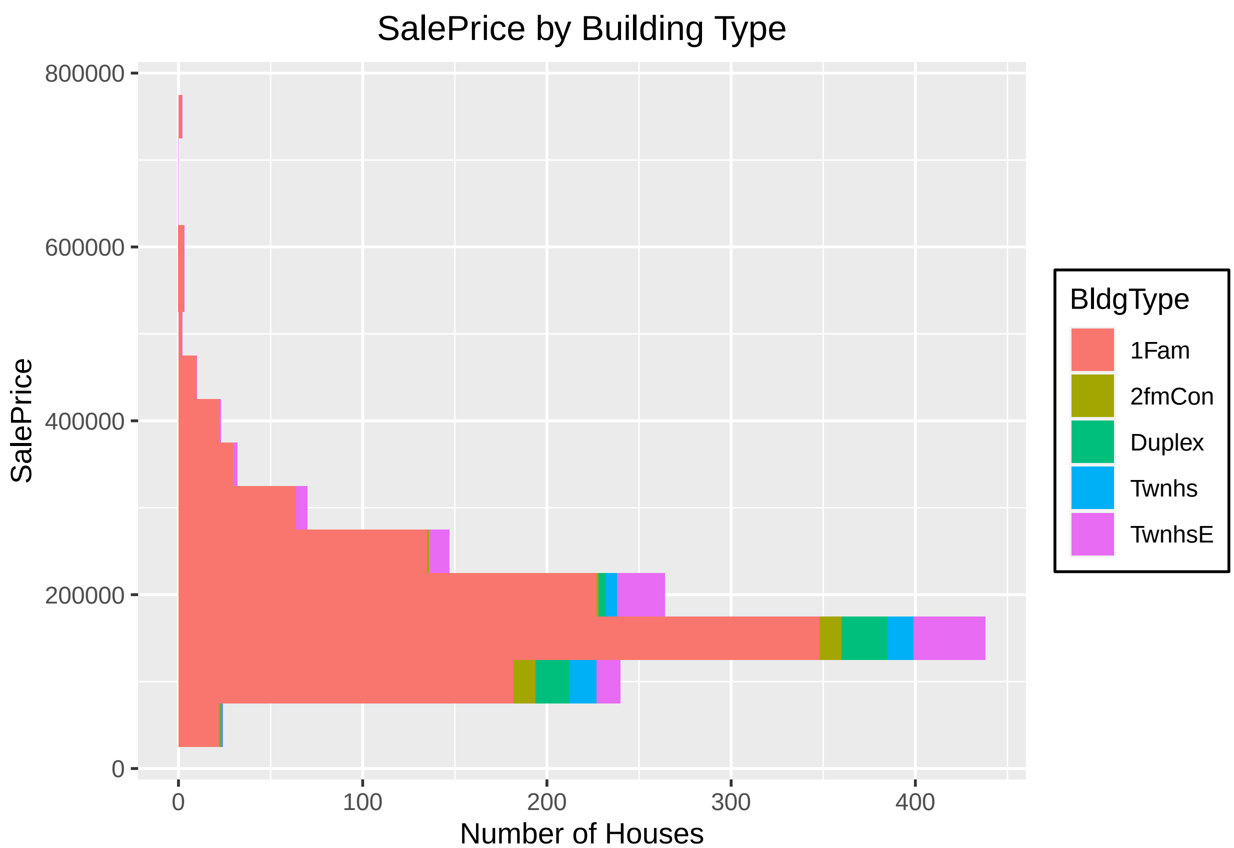
```
x<-ddply(training_set, .(BldgType), summarize,
                                   average_price=mean(SalePrice),
                                   number_of_houses=length(BldgType),
                                   minimum_price=min(SalePrice),
                                   maximum_price=max(SalePrice))
x[order(-x$average_price),] # Display in descending order of average price
```

```
##   BldgType average_price number_of_houses minimum_price maximum_price
## 1     1Fam      189686.4             1050         34900        755000
## 5   TwnhsE      183463.9               98         75500        392500
## 4    Twnhs      133922.2               36         75000        209500
## 3   Duplex      133290.1               47         82000        206300
## 2   2fmCon      129033.3               27         55000        228950
```

We note that **Single-Family Detached** houses outnumber the other dwelling types not only in count (1050) but also in terms of average price ($189,686) and maximum price ($755,000) paid for a house in that category of building. There are only 27 **Two-family Conversion** type houses and that category possesses the lowest average price of $129,033.

We can find additional details about the distribution of houses in each category by plotting a histogram.

```
ggplot(training_set, aes(SalePrice)) +
 geom_histogram(aes(fill = BldgType),
                position = position_stack(reverse = TRUE),
                binwidth = 50000) +
 coord_flip() +
 ggtitle("SalePrice by Building Type") +
 ylab("Number of Houses") +
 xlab("SalePrice") +
 theme(plot.title = element_text(hjust = 0.5),
       legend.position="right",
       legend.background = element_rect(size=0.5,
                                        linetype="solid",
                                        colour ="black"))
```



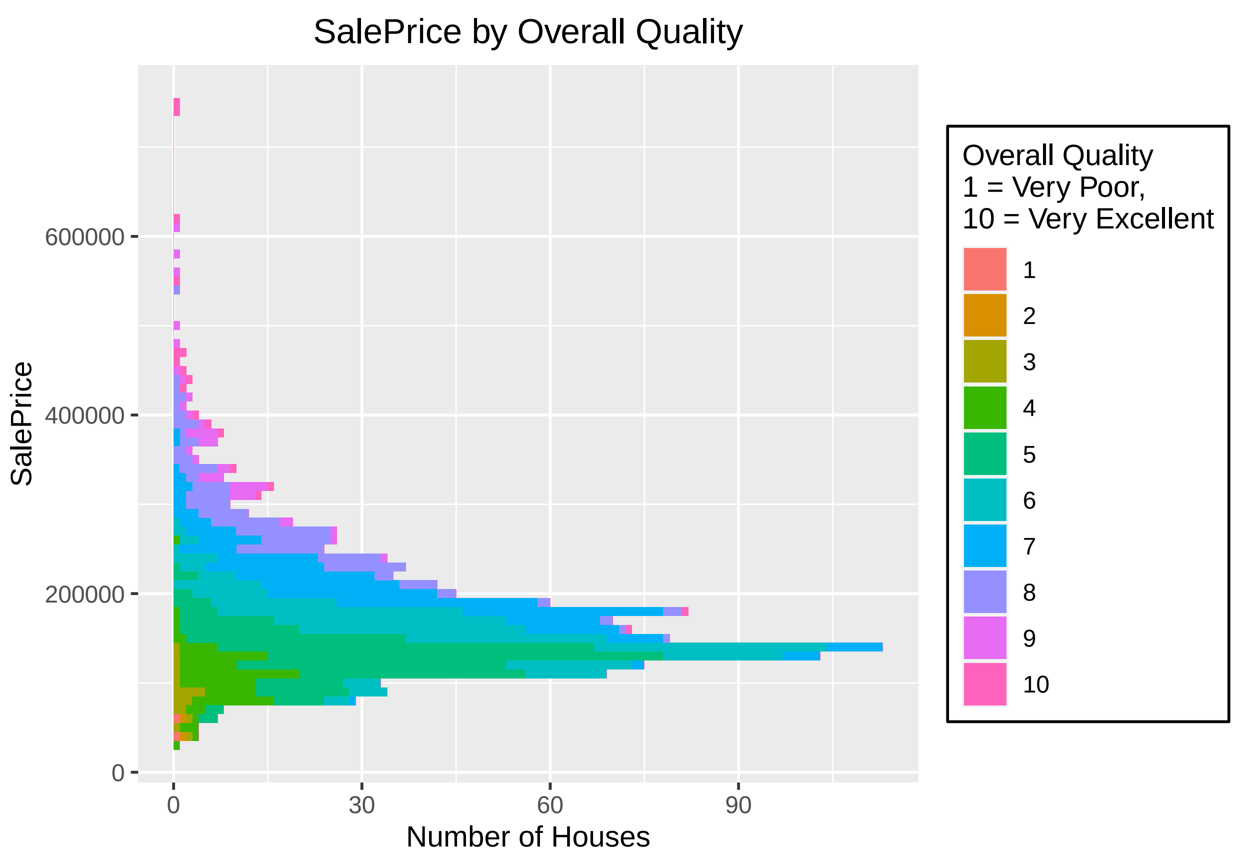
Combining the results from the table showing minimum and maximum price by dwelling type with the Histogram plotted above, we make the following inferences:

- Houses that sold in the higher price range of 400K to 755K were Single-Family Detached.
- Houses of other dwelling types were confined to the range of 55K to 393K
- While it makes sense that the most expensive house ($755,000) was a Single-Family Detached type, it is interesting to see that the least expensive ($34,900) also happens to be a Single-Family detached.
- In summary, Single-Family Detached spans the overall range of SalePrice, indicating the higher spread (and hence choice for homeowners) from a financial standpoint.

Insights on Categorical Feature: Overall Quality (OverallQual)

While we reviewed the distribution of SalePrice a few sections earlier, we will now review that in the context of Overall Quality (OverallQual) of the house which represents the overall material and finish of the house with values ranging from 1 through 10 with 1 being Very poor and 10 being Very Excellent.

```
ggplot(training_set, aes(SalePrice)) +
 geom_histogram(aes(fill = as.factor(OverallQual)),
                position = position_stack(reverse = TRUE),
                binwidth = 10000) +
 coord_flip() +
 ggtitle("SalePrice by Overall Quality") +
 ylab("Number of Houses") +
 xlab("SalePrice") +
 scale_fill_discrete(name="Overall Quality\n1 = Very Poor, \n10 = Very Excellent")+
 theme(plot.title = element_text(hjust = 0.5),
       legend.position="right",
       legend.background = element_rect(size=0.5,
                                        linetype="solid",
                                        colour ="black"))
```



We make the following findings:

- Houses with higher Overall Quality (9-10) sold at much higher prices ($275,000 to $755,000) than the rest (which makes logical sense).

- Houses with Above Average quality (6) but below Excellent (9) sold in the mid-range of \$100,000 to \$350,000.
- Both of the above findings jive well with the common sense that higher the overall quality, higher the price a homeowner should expect to pay for the house.
- The distribution of houses in each quality category is fairly even/symmetric.

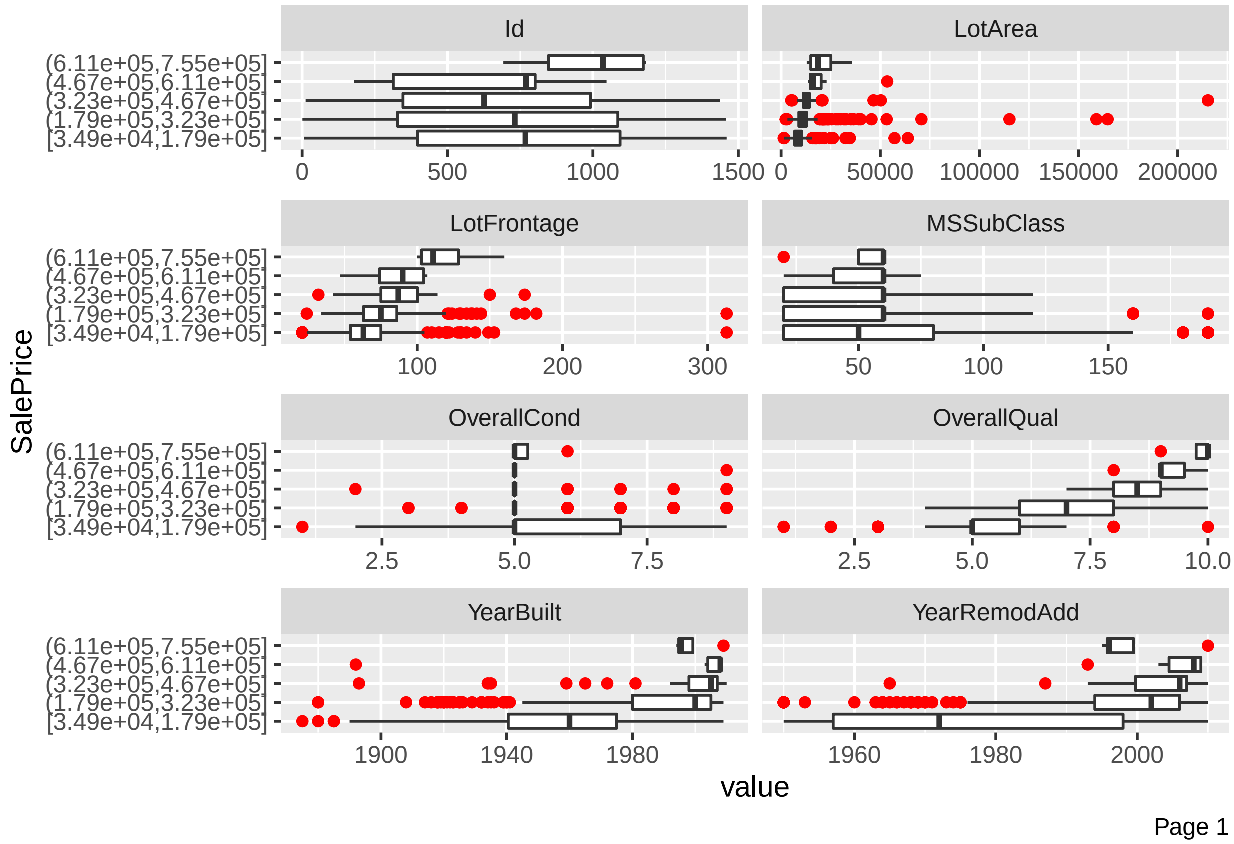
Visualize ALL Continuous Features by SalePrice

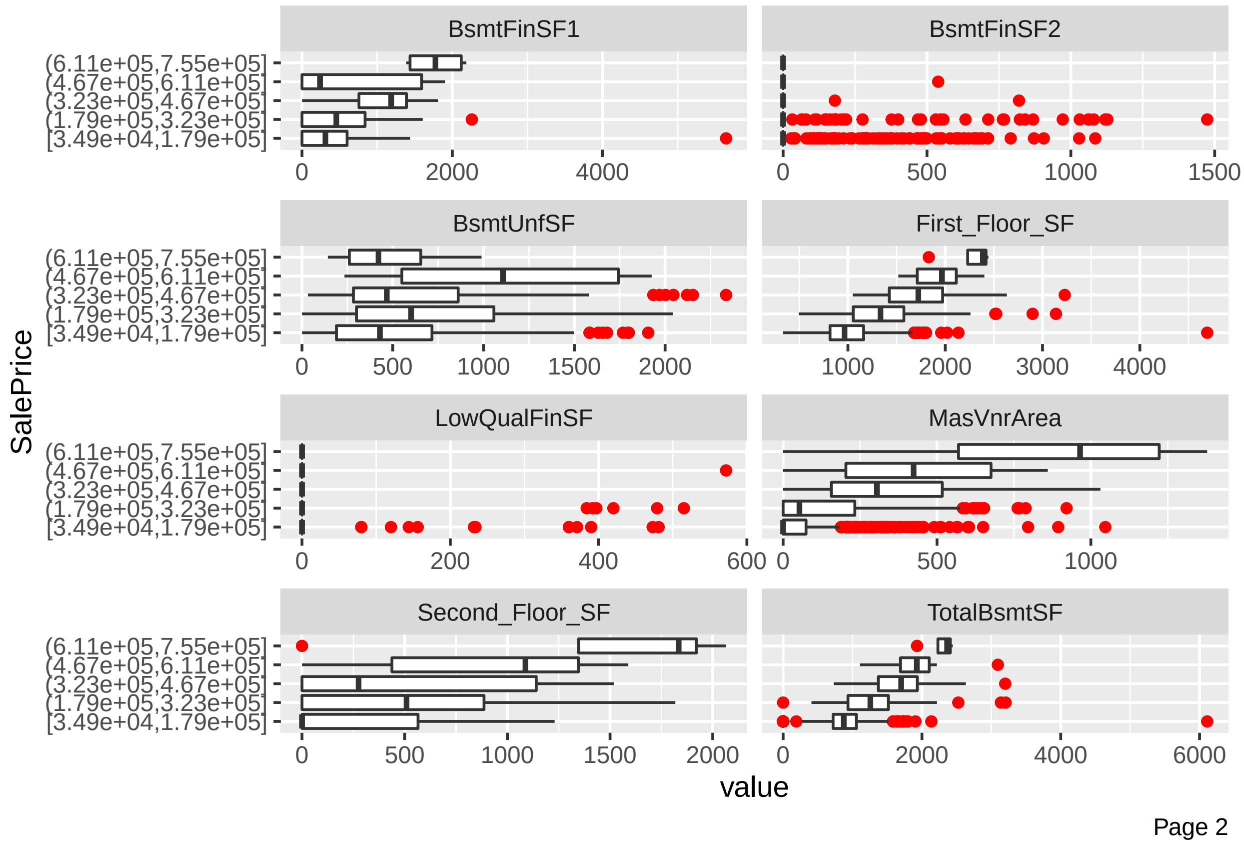
We will now make box plots for all the continuous features by SalePrice to see the data distribution, central values, and variability. For better viewing to eliminate outliers, we will plot them in RED. Again, we will have 38 plots spanning multiple pages.

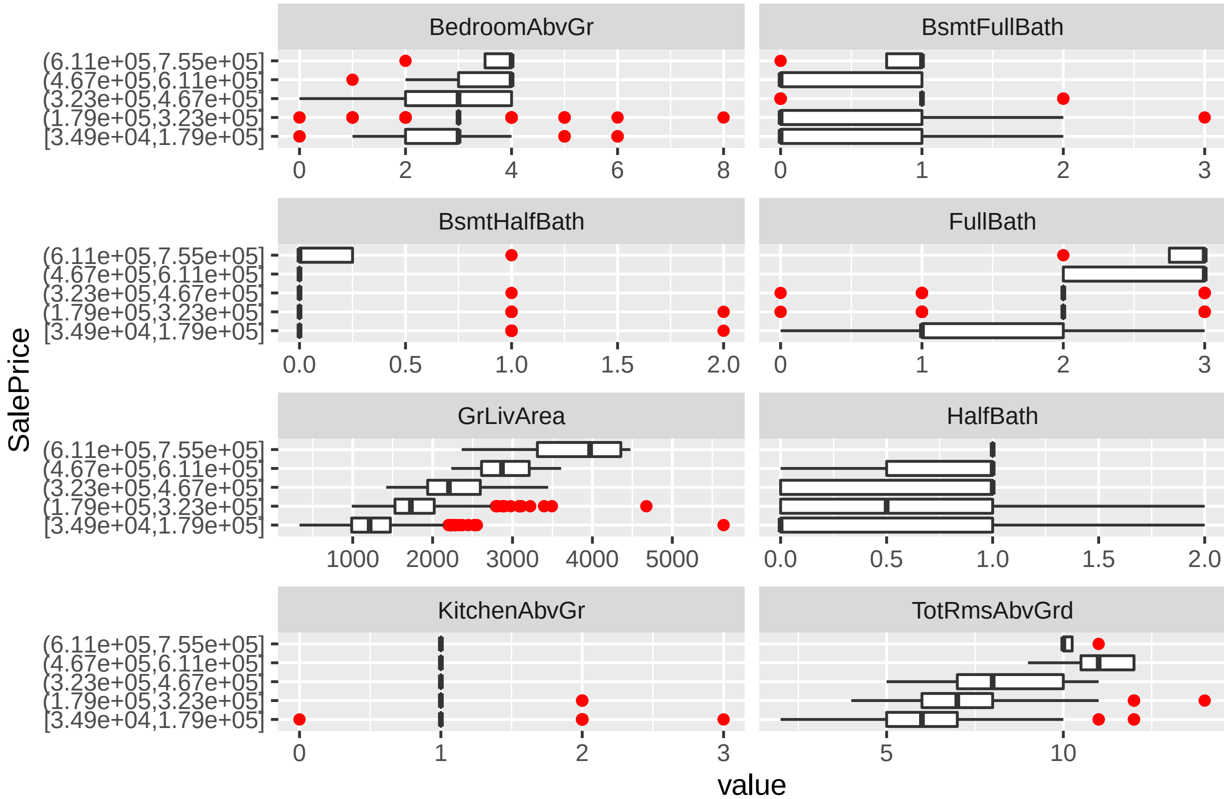
The Y-axis shows 5 ranges for SalePrice because by default, plot_boxplot's **by** argument for a continuous feature will be grouped by 5 equal ranges. The first range is a SalePrice from $34,900 to $179,000 while the top range is from $611,000 to $755,000.

- For example, on **Page# 1 in the graphs below** for the **OverallQual** feature (Rating of the overall material and finish of the house), we find that the overall quality of the house was higher for higher ranges of **SalePrice**. This sounds very logical because houses with better quality will likely cost more than ones with lower overall quality score.
- For the **YearBuilt** feature on **Page# 1**, we see that houses created between the year 1940 and 1975 were in the range of **$34,900 to $179,000**. In other words, the records show that a house in those times cost less than $200K! Of course, considering the timevalue for money, $200K in those times might be much more in today's dollars.
- For the **BedroomAbvGr** (Bedroom above ground, not including basement) feature on **Page# 3**, we see that homes with SalePrice above $467,000 had at least 3 bedrooms above ground and at least 2 Full Baths (see the plot for **FullBath** in the adjacent column).
- On **Page# 5**, in the plot for **MoSold** feature (Month Sold) we see that May through August was the period with most activity for all the ranges of SalePrice for houses. This makes sense because summer time is when people are likely to move because of kids having school vacation and it is a perfect time to move so that the family is settled before the new school year begins; not to mention the longer days giving more time for a move and settling down.

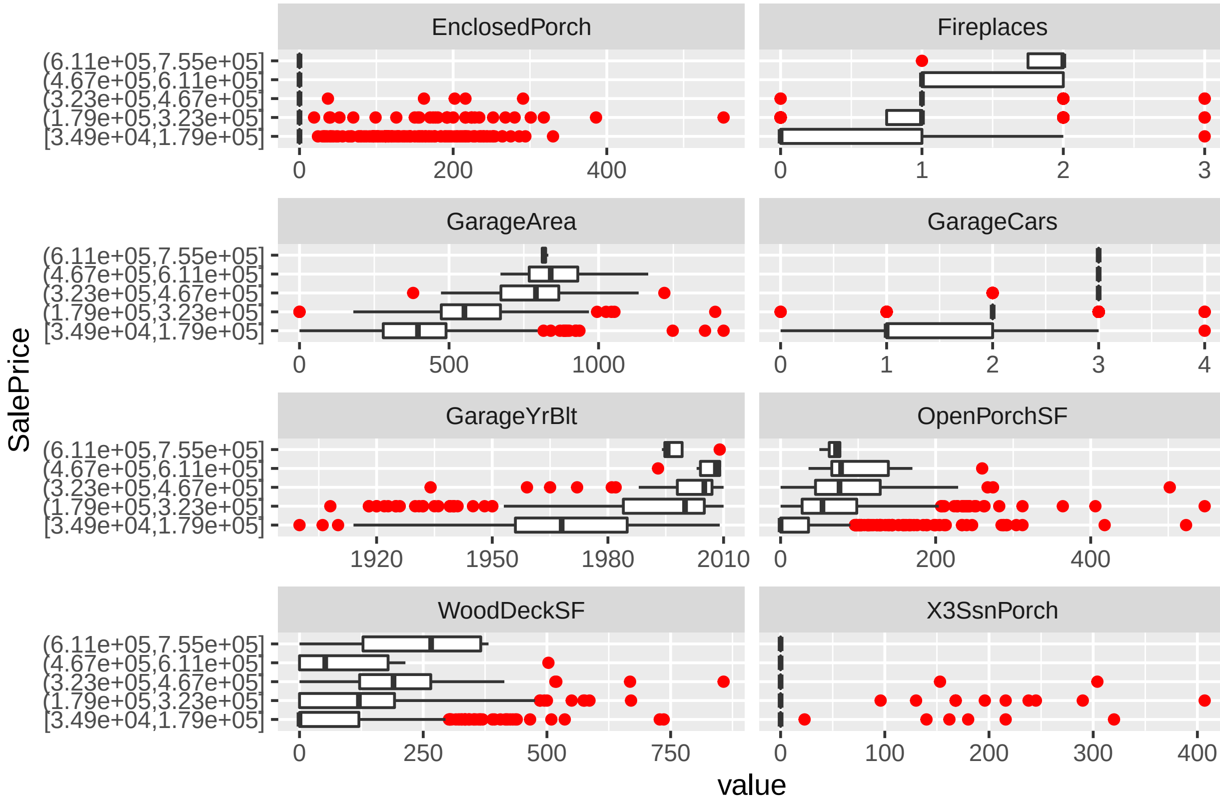
```
plot_boxplot(training_set, by="SalePrice",
             geom_boxplot_args = list("outlier.color" = "red"),
             nrow = 4L, ncol=2L)
```


Id
LotArea
LotFrontage
MSSubClass
OverallCond
OverallQual
YearBuilt
YearRemodAdd
(6.11e+05,7.55e+05]
(4.67e+05,6.11e+05]
(3.23e+05,4.67e+05]
(1.79e+05,3.23e+05]
[3.49e+04,1.79e+05]
SalePrice
value
Page 1

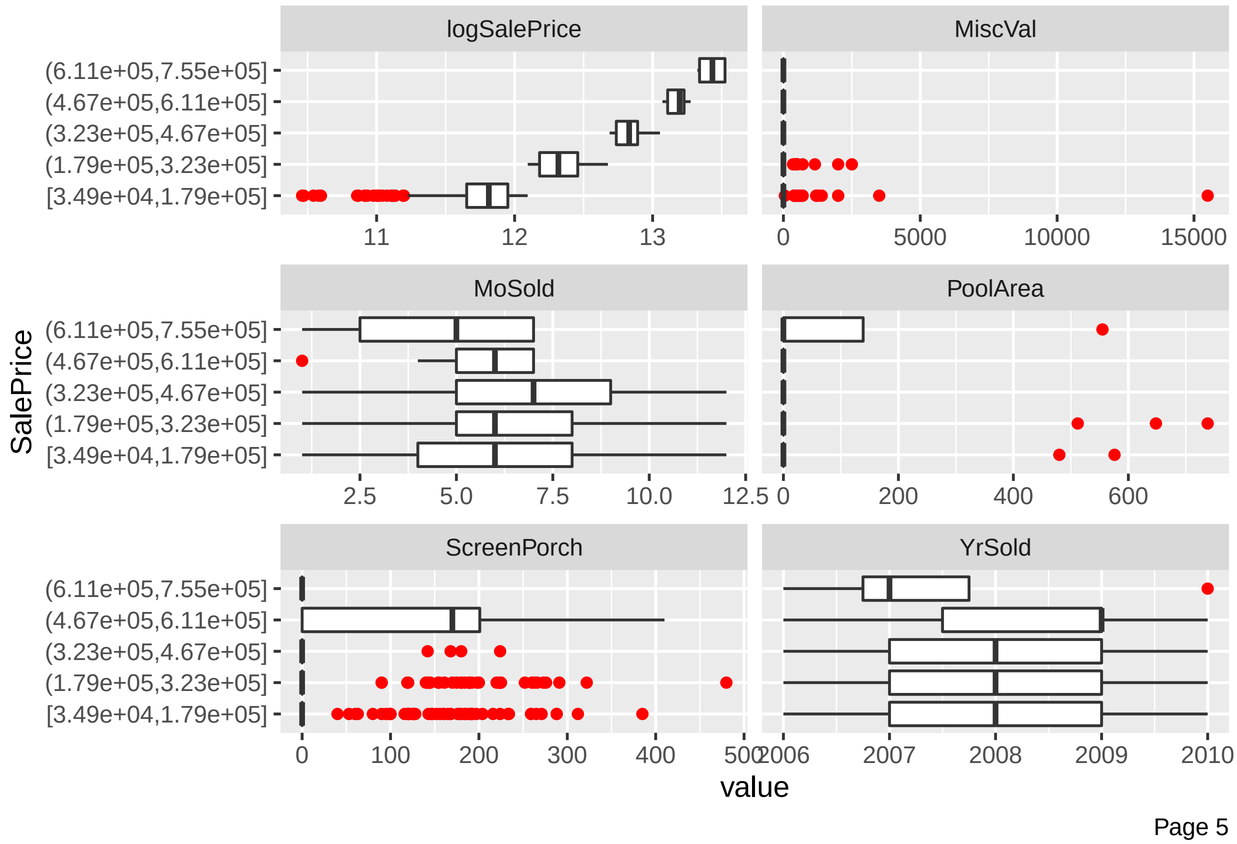

BsmtFinSF1
BsmtFinSF2
BsmtUnfSF
First_Floor_SF
LowQualFinSF
MasVnrArea
Second_Floor_SF
TotalBsmtSF
SalePrice
(6.11e+05,7.55e+05]
(4.67e+05,6.11e+05]
(3.23e+05,4.67e+05]
(1.79e+05,3.23e+05]
[3.49e+04,1.79e+05]
value
Page 2


BedroomAbvGr
BsmtFullBath
BsmtHalfBath
FullBath
GrLivArea
HalfBath
KitchenAbvGr
TotRmsAbvGrd
SalePrice
(6.11e+05,7.55e+05]
(4.67e+05,6.11e+05]
(3.23e+05,4.67e+05]
(1.79e+05,3.23e+05]
[3.49e+04,1.79e+05]
value

Page 3


EnclosedPorch
Fireplaces
GarageArea
GarageCars
GarageYrBlt
OpenPorchSF
WoodDeckSF
X3SsnPorch
SalePrice
value
(6.11e+05,7.55e+05]
(4.67e+05,6.11e+05]
(3.23e+05,4.67e+05]
(1.79e+05,3.23e+05]
[3.49e+04,1.79e+05]

Page 4


logSalePrice
MiscVal
MoSold
PoolArea
ScreenPorch
YrSold
SalePrice
(6.11e+05,7.55e+05]
(4.67e+05,6.11e+05]
(3.23e+05,4.67e+05]
(1.79e+05,3.23e+05]
[3.49e+04,1.79e+05]
11
12
13
0
5000
10000
15000
2.5
5.0
7.5
10.0
12.5
0
200
400
600
0
100
200
300
400
500
2006
2007
2008
2009
2010
value
Page 5

Feature Engineering: Determine Important Features for Analysis

To get a sense on some of the features that should be important for our analysis, we will employ **Boruta**, an all relevant feature selection wrapper algorithm, capable of working with any classification method that output variable importance measure (VIM); by default, Boruta uses Random Forest.

The method performs a top-down search for relevant features by comparing original attributes' importance with importance achievable at random, estimated using their permuted copies, and progressively eliminating irrelevant features to stabilize that test. **For more details on Boruta**, check out: **https://www.rdocumentation.org/packages/Boruta/versions/7.0.0/topics/Boruta**.

Perform Boruta search and check the output.

```
boruta_output <- Boruta(SalePrice ~ ., data=na.omit(training_set), doTrace=0)
names(boruta_output)
```

```
##  [1] "finalDecision" "ImpHistory"    "pValue"        "maxRuns"
##  [5] "light"         "mcAdj"         "timeTaken"     "roughfixed"
##  [9] "call"          "impSource"
```

Get significant variables including tentative.

```
boruta_signif <- getSelectedAttributes(boruta_output, withTentative = TRUE)
print(boruta_signif) # Display Boruta significant results.
```

```
##  [1] "MSSubClass"      "MSZoning"        "LotFrontage"     "LotArea"
##  [5] "LandContour"     "Neighborhood"    "BldgType"        "HouseStyle"
##  [9] "OverallQual"     "OverallCond"     "YearBuilt"       "YearRemodAdd"
## [13] "RoofStyle"       "Exterior1st"     "Exterior2nd"     "MasVnrType"
## [17] "MasVnrArea"      "ExterQual"       "Foundation"      "BsmtQual"
## [21] "BsmtExposure"    "BsmtFinType1"    "BsmtFinSF1"      "BsmtUnfSF"
## [25] "TotalBsmtSF"     "HeatingQC"       "CentralAir"      "First_Floor_SF"
## [29] "Second_Floor_SF" "GrLivArea"       "BsmtFullBath"    "FullBath"
## [33] "HalfBath"        "BedroomAbvGr"    "KitchenAbvGr"    "KitchenQual"
## [37] "TotRmsAbvGrd"    "Fireplaces"      "GarageType"      "GarageYrBlt"
## [41] "GarageFinish"    "GarageCars"      "GarageArea"      "WoodDeckSF"
## [45] "OpenPorchSF"     "SaleCondition"   "logSalePrice"
```

If you are not sure about the tentative variables being selected for granted, you can choose a TentativeRoughFix on boruta_output.

Do a tentative rough fix.

```
roughFixMod <- TentativeRoughFix(boruta_output)
boruta_signif <- getSelectedAttributes(roughFixMod)
print(boruta_signif)
```

```
##  [1] "MSSubClass"      "MSZoning"        "LotFrontage"     "LotArea"
##  [5] "Neighborhood"    "BldgType"        "HouseStyle"      "OverallQual"
##  [9] "OverallCond"     "YearBuilt"       "YearRemodAdd"    "Exterior1st"
## [13] "Exterior2nd"     "MasVnrArea"      "ExterQual"       "Foundation"
## [17] "BsmtQual"        "BsmtExposure"    "BsmtFinType1"    "BsmtFinSF1"
## [21] "BsmtUnfSF"       "TotalBsmtSF"     "HeatingQC"       "CentralAir"
```

```
## [25] "First_Floor_SF"  "Second_Floor_SF" "GrLivArea"       "BsmtFullBath"
## [29] "FullBath"        "HalfBath"        "BedroomAbvGr"    "KitchenAbvGr"
## [33] "KitchenQual"     "TotRmsAbvGrd"    "Fireplaces"      "GarageType"
## [37] "GarageYrBlt"     "GarageFinish"    "GarageCars"      "GarageArea"
## [41] "WoodDeckSF"      "OpenPorchSF"     "logSalePrice"
```

Boruta has decided on the 'Tentative' variables on our behalf. Let's find out the importance scores of these variables.

Variable Importance Scores

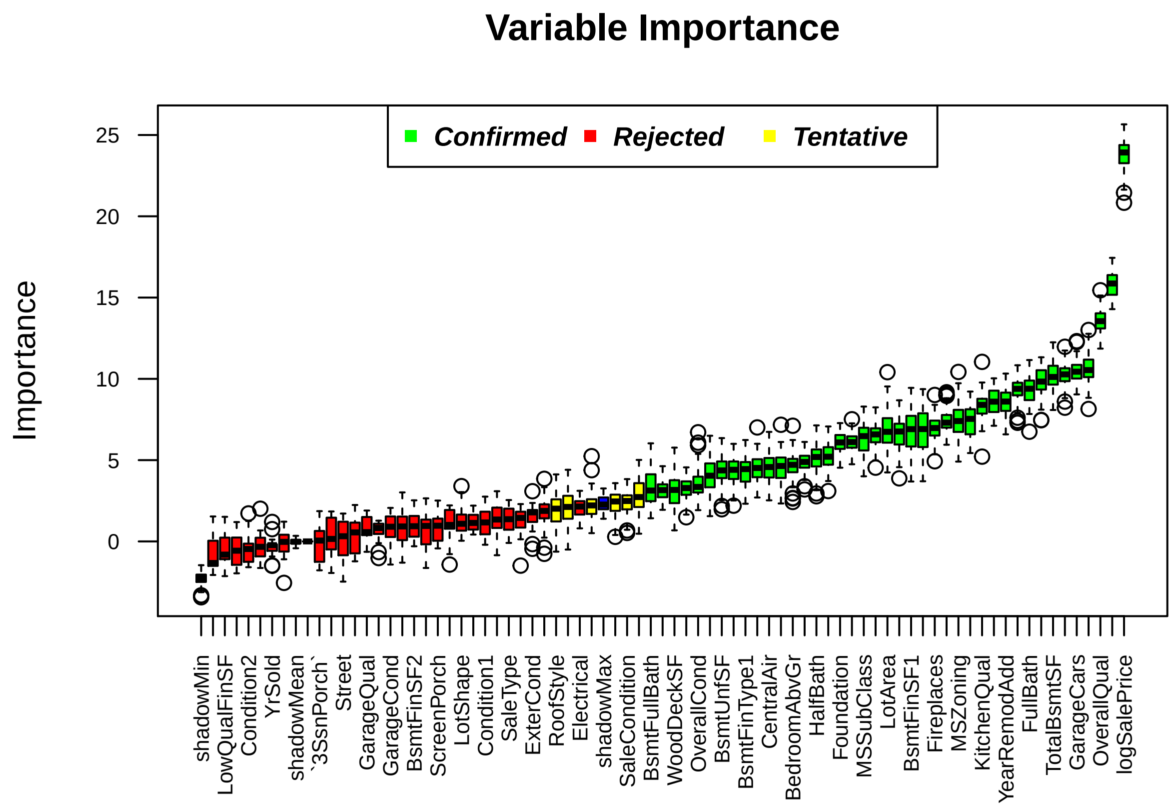
```
imps <- attStats(roughFixMod)
confirmed_features = imps[imps$decision != 'Rejected', c('meanImp', 'decision')]
head(confirmed_features[order(-confirmed_features$meanImp), ])  # descending sort
```

```
##                  meanImp  decision
## logSalePrice    23.80924 Confirmed
## GrLivArea       15.82062 Confirmed
## OverallQual     13.56826 Confirmed
## Second_Floor_SF 10.59711 Confirmed
## GarageCars      10.46490 Confirmed
## First_Floor_SF  10.28283 Confirmed
```

Plot variable importance

Let us make a Variable Importance plot to get a visual on each feature's relative importance.

```
plot(boruta_output, cex.axis=.7, las=2,
     xlab="", main="Variable Importance")
legend('top', legend=c("Confirmed","Rejected","Tentative"),
       col=c("Green","Red","Yellow"),horiz = TRUE,
       text.font=4, cex=0.8,pch=15)
```



This plot reveals the importance of each of the features. The columns in green are ‘confirmed’ and the ones in red are not. There are couple of blue bars representing ShadowMax and ShadowMin. They are not actual features, but are used by the boruta algorithm to decide if a variable is important or not.

Here’s a list of all the features that were confirmed using Boruta in descending order of importance. We will be using a subset of these features in our **Modeling Approach** with some of the categorical features that were analyzed earlier.

```r
rownames(confirmed_features[order(-confirmed_features$meanImp), ])
```

```
##  [1] "logSalePrice"    "GrLivArea"       "OverallQual"     "Second_Floor_SF"
##  [5] "GarageCars"      "First_Floor_SF"  "TotalBsmtSF"     "YearBuilt"
##  [9] "FullBath"        "ExterQual"       "GarageArea"      "YearRemodAdd"
## [13] "KitchenQual"     "MSZoning"        "GarageType"      "GarageYrBlt"
## [17] "Fireplaces"      "LotArea"         "BsmtFinSF1"      "BsmtQual"
## [21] "TotRmsAbvGrd"    "GarageFinish"    "MSSubClass"      "Neighborhood"
## [25] "Foundation"      "OpenPorchSF"     "HalfBath"        "HeatingQC"
## [29] "BedroomAbvGr"    "CentralAir"      "MasVnrArea"      "BldgType"
## [33] "BsmtUnfSF"       "HouseStyle"      "BsmtFinType1"    "LotFrontage"
## [37] "OverallCond"     "BsmtFullBath"    "KitchenAbvGr"    "Exterior1st"
## [41] "WoodDeckSF"      "Exterior2nd"     "BsmtExposure"
```

Insights using Correlation

Given the large set of features, we have to be diligent in picking the features that seem most relevant and contributing to the prediction of our outcome, **SalePrice**. Aside from insights gained in prior sections, we need to have **domain knowledge** about the Real Estate industry to select the relevant features.

Common sense along with some experience having put multiple offers on different homes and research conducted when purchasing my own home leads me to at least expect some of these features to be relevant: Total Lot size (Square foot area), Living Area inside the home, number of bedrooms, number of bathrooms, age of the house, overall condition of the house, quality of construction, does the house have central air conditioning and heating, does it have a basement, etc.

Let us first undertake correlation analysis against the top 20 features recommended by Boruta in the prior section.

```r
#' List the top 20 important features confirmed by Boruta.
#' Here, we take 21 because the 1st one, logSalePrice is something we added
#' as part of prior data processing and actually represents the outcome. So we will
#' take the first 21 and skip the logSalePrice for this correlation analysis.

top_20_boruta<-rownames(confirmed_features[order(-confirmed_features$meanImp), ])[2:21]
top_20_boruta<- c(top_20_boruta,"SalePrice")
top_20_boruta<- training_set[top_20_boruta]
```

Before we make correlation plot, we need to make sure the features are numeric. We see that five of the top_20_boruta features are categorical in nature. Let us first convert them to numeric.

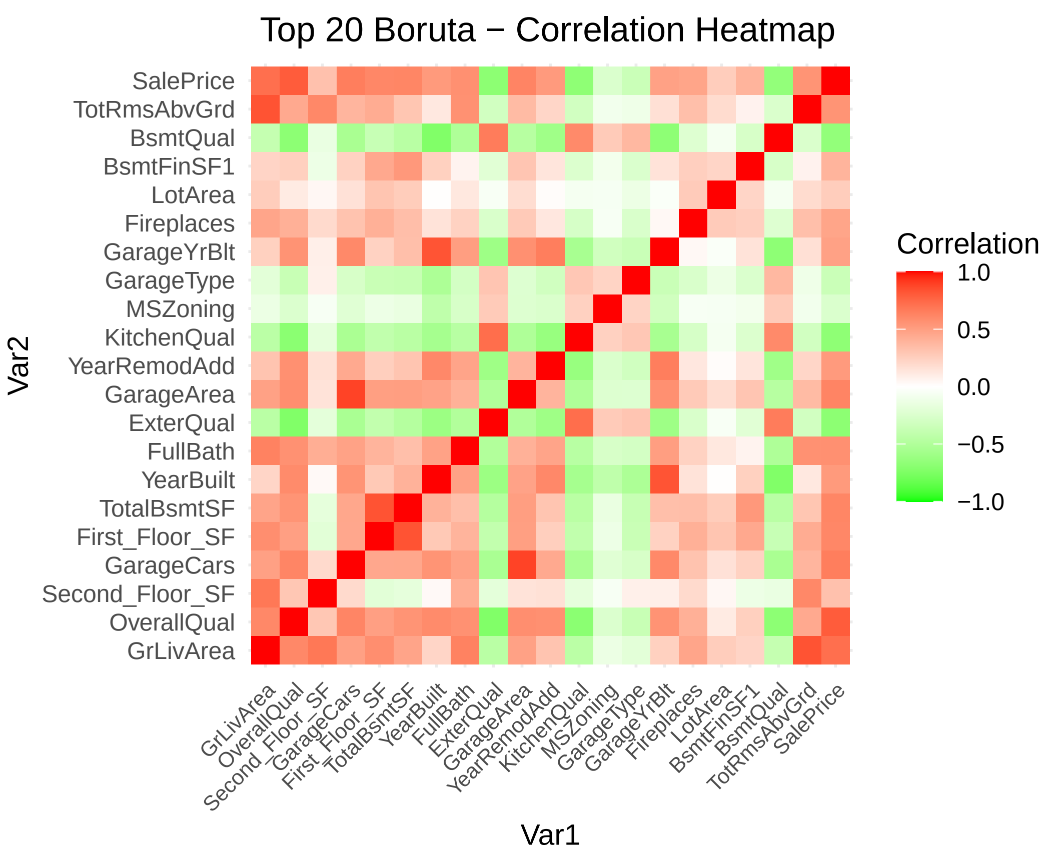
The three quality features - BsmtQual, ExterQual, and KitchenQual are encoded such that Excellent=5 and Poor=1 with the rest of the values in diminishing value of quality.

The remaining 2 categorical features are encoded as shown in the code below.

```r
# convert factor to numeric
top_20_boruta$BsmtQual <- as.numeric(factor(top_20_boruta$BsmtQual,
                                  levels = c("Ex", "Gd","TA", "Fa","Po"),
                                  labels = c(5,4,3,2,1) ,ordered = TRUE))

top_20_boruta$ExterQual <- as.numeric(factor(top_20_boruta$ExterQual,
                                  levels = c("Ex", "Gd","TA", "Fa","Po"),
                                  labels = c(5,4,3,2,1) ,ordered = TRUE))

top_20_boruta$KitchenQual <- as.numeric(factor(top_20_boruta$KitchenQual,
                                  levels = c("Ex", "Gd","TA", "Fa","Po"),
                                  labels = c(5,4,3,2,1) ,ordered = TRUE))

top_20_boruta$MSZoning <- as.numeric(factor(top_20_boruta$MSZoning,
                                  levels = c("A", "C","FV", "I","RH","RL","RP","RM"),
                                  labels = c(1,2,3,4,5,6,7,8) ,ordered = TRUE))

top_20_boruta$GarageType <- as.numeric(factor(top_20_boruta$GarageType,
                                  levels = c("2Types", "Attchd","Basment",
                                             "BuiltIn","CarPort","Detchd","NA"),
                                  labels = c(1,2,3,4,5,6,7) ,ordered = TRUE))
```

We can now proceed with our Correlation plot.

```r
#plot correlation heatmap for SalePrice for the top_20_boruta confirmed features
options(repr.plot.width=8, repr.plot.height=6)
library(ggplot2)
library(reshape2)
qplot(x=Var1, y=Var2, data=melt(cor(top_20_boruta, use="p")), fill=value, geom="tile") +
   scale_fill_gradient2(low = "green", high = "red", mid = "white",
   midpoint = 0, limit = c(-1,1), space = "Lab",
   name="Correlation") +
   theme_minimal()+
   theme(axis.text.x = element_text(angle = 45, vjust = 1, size = 8, hjust = 1))+
   coord_fixed()+
   ggtitle("Top 20 Boruta - Correlation Heatmap") +
   theme(plot.title = element_text(hjust = 0.4))
```



We make the following observations:

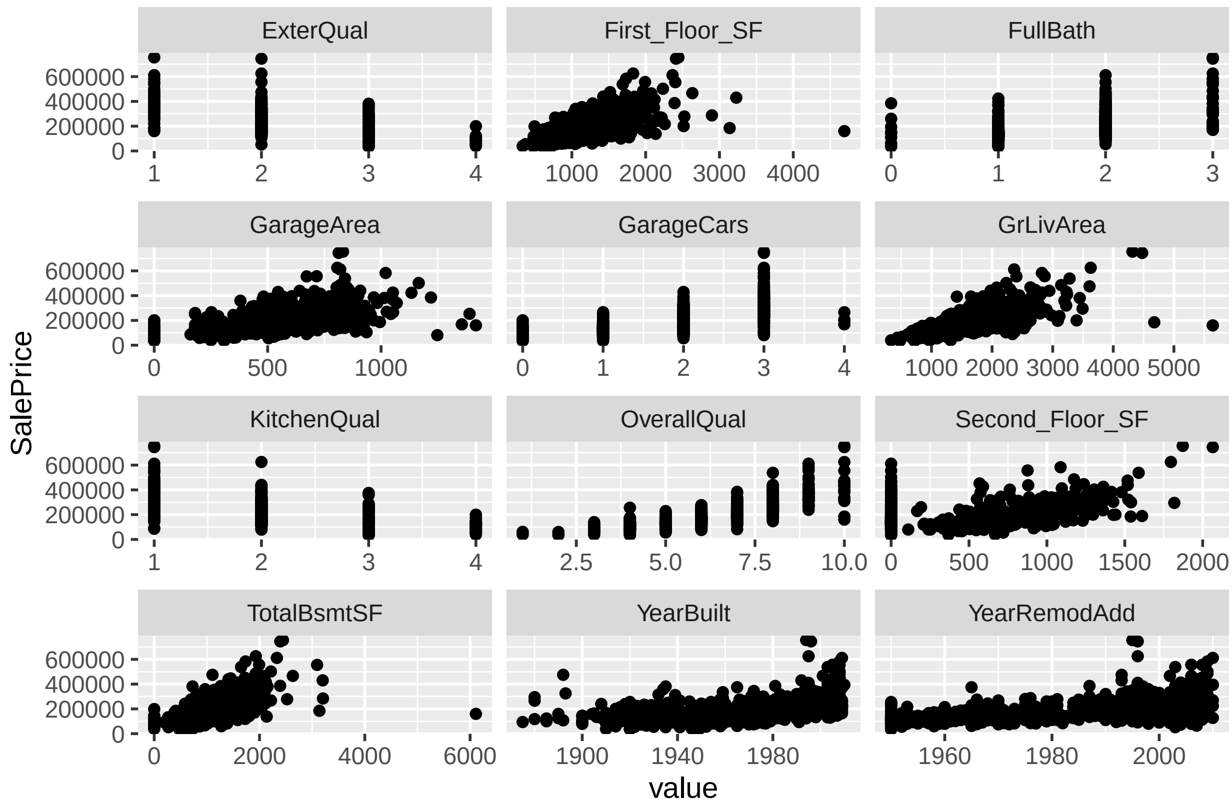
1. Red shows positive correlation, whereas Green shows negative correlation.
2. Pretty much all the 20 features confirmed by Boruta as being important show strong correlation with SalePrice. This is confirmed by looking at the plot and focusing on the **top row** for **SalePrice** which shows higher shades of Red or Green for almost all of the 20 features.

We can safely proceed with further analysis having validated the strong correlation for the top 20 Boruta confirmed features.

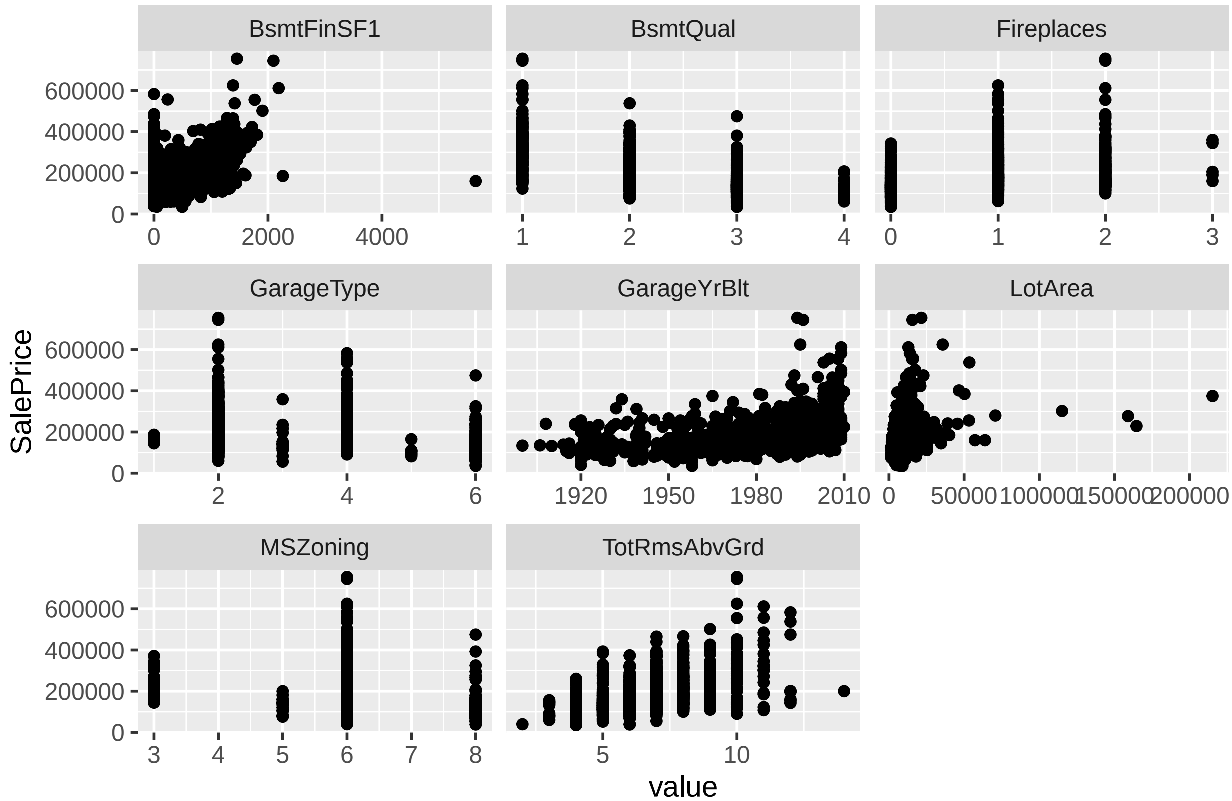
Insights using ScatterPlot by SalePrice

We make one final plot to visualize the relationship of these Boruta-recommended top 20 features with SalePrice. This is a Scatter Plot of the features by SalePrice.

```
plot_scatterplot(top_20_boruta, by="SalePrice",nrow = 4L, ncol=3L)
```



Page 1



Page 2

We can confirm that:

- First_Floor_SF, Second_Floor_SF, GarageArea, TotalBstmtSF, BsmtFinSF1, and LotArea have a positive relationship with SalePrice. This means that effectively, higher the square footage of 1st Floor, 2nd Floor, etc., higher the price a buyer should expect to pay for the house.
- Similarly, the newer the house (YearBuilt, GarageYrBlt) or the more recent the house was remodeled (YearRemodAdd), the higher the expected price for the house.
- The plots for FullBath and TotRmsAbvGrd plot show that the more the number of full baths and number of rooms available above the ground, the higher the SalePrice.

Modeling Approach

For each of the models we build below, we will be measuring their performance using a metric, Root Mean Squared Error (RMSE). RMSE is square root of the average of the residuals squared.

Data Pre-processing

As noted earlier in the Data Exploration and Visualization section under Insights on Outcome: SalePrice Distribution sub-section, we see that the SalePrice, our outcome variable, is skewed and we needed to do a log transformation to get a normal distribution for the same.

Let us now update our training_set and test_set with the following changes:

1. Only include the top 20 Boruta confirmed features.
2. Include the log transformed SalePrice for modeling.
3. Apply the one-hot encoding for the categorical features as was done for correlation

```
# Add logSalePrice to test_set
test_set$logSalePrice <- log(test_set$SalePrice)

# Add the outcome variable to our Validation set because it does not contain that column.
validation["SalePrice"]=0
validation["logSalePrice"]=0

# Reduce the number of features used for analysis by limiting our feature set
# to the ones confirmed by Boruta as being important.
training_set<-training_set[c(names(top_20_boruta),"logSalePrice")]
test_set<-test_set[c(names(top_20_boruta),"logSalePrice")]
validation<-validation[names(top_20_boruta)]


# Convert any character vectors to factors in both the training and test set
training_set$ExterQual<- as.factor(training_set$ExterQual)
training_set$KitchenQual<- as.factor(training_set$KitchenQual)
training_set$MSZoning<- as.factor(training_set$MSZoning)
training_set$GarageType<- as.factor(training_set$GarageType)
training_set$BsmtQual<- as.factor(training_set$BsmtQual)


test_set$ExterQual<- as.factor(test_set$ExterQual)
test_set$KitchenQual<- as.factor(test_set$KitchenQual)
test_set$MSZoning<- as.factor(test_set$MSZoning)
test_set$GarageType<- as.factor(test_set$GarageType)
test_set$BsmtQual<- as.factor(test_set$BsmtQual)

validation$ExterQual<- as.factor(validation$ExterQual)
validation$KitchenQual<- as.factor(validation$KitchenQual)
validation$MSZoning<- as.factor(validation$MSZoning)
validation$GarageType<- as.factor(validation$GarageType)
validation$BsmtQual<- as.factor(validation$BsmtQual)
```

Random Forest

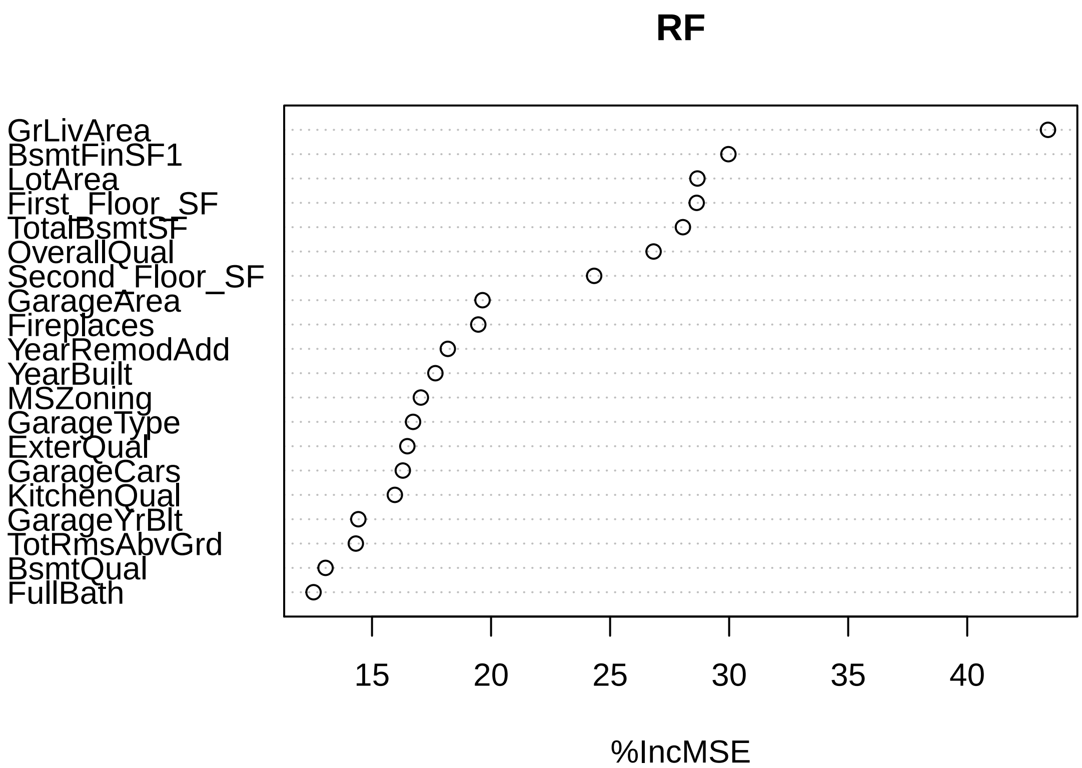
We will run the Random Forest algorithm against the training_set for different values of several of its key parameters - ntree, nodesize, and mtry. We are undertaking supervised learning and for regression using Random Forest, we set **nodesize = 5**.

The parameter **mtry** represents the number of variables randomly sampled as candidates at each split. Note that the default values are different for classification (sqrt(p) where p is number of variables in x) and regression (p/3). In our case, we are going to limit our analysis to the top 20 Boruta confirmed features so a value of 20/3 = 6.33 is a good start. After playing with different values, I settled for **mtry=5** to obtain the lowest RMSE in combination with **ntree=600** and **nodesize=5**.

```
# Model 1: Random Forest
set.seed(500)
RF <- randomForest(logSalePrice ~.-SalePrice,
                   data = training_set,
                   na.action=na.roughfix,
                   importance =TRUE,
                   ntree=600,
                   nodesize=5,
                   mtry=5)
```

Now let us plot the Dotchart of variable importance as measured by a Random Forest using the **varImpPlot()** function.

```
# variable importance
options(repr.plot.width=9, repr.plot.height=6)
varImpPlot(RF, type=1)
```



Next, we will make predictions against the test_set and compute accuracy using the *accuracy()* function from the **forecast** library. The accuracy function will compute the following measures:

- ME: Mean Error
- RMSE: Root Mean Squared Error
- MAE: Mean Absolute Error
- MPE: Mean Percentage Error
- MAPE: Mean Absolute Percentage Error
- MASE: Mean Absolute Scaled Error
- ACF1: Autocorrelation of errors at lag 1

```
#prediction
pred_rf <- predict(RF, newdata=test_set )
acc_rf<-accuracy(pred_rf, test_set$logSalePrice)
acc_rf
```

```
##                     ME     RMSE        MAE        MPE     MAPE
## Test set -0.01328804 0.107672 0.08046845 -0.1172022 0.672915
```

```
acc_rf<- as.data.frame(acc_rf)
```

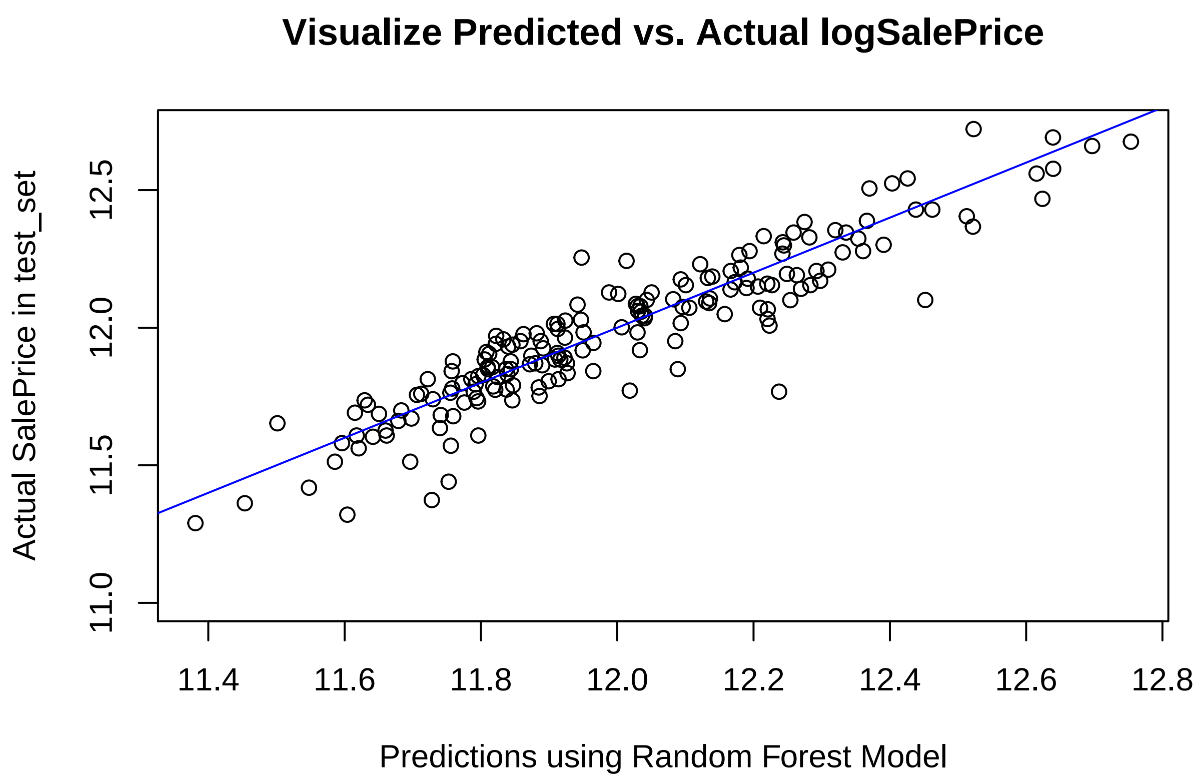
We will now build a table to keep track of the RMSE results from each of our models.

```
rmse_results <- data.frame(Model_Method="Random Forest",RMSE_Values=acc_rf$RMSE)
rmse_results %>% knitr::kable()
```

Model_Method	RMSE_Values
Random Forest	0.107672

Let us visualize how our predictions compare against actual values present in the test_set.

```
# Visualize Predicted versus Actual
plot(pred_rf, test_set$logSalePrice,
     main = "Visualize Predicted vs. Actual logSalePrice",
     xlab="Predictions using Random Forest Model",
     ylab="Actual SalePrice in test_set")
abline(a=0,b=1,col="blue")
```



Classification and Regression Tree-based Model using CART

We will now build a regression tree model using the **cart** package. We set the method to "anova" and run *rpart* against our training_set's Boruta recommended top 20 features.

```
set.seed(500)
# Generate regression tree using rpart
fit <- rpart(logSalePrice ~.-SalePrice,
             data = training_set,
             method="anova")
```

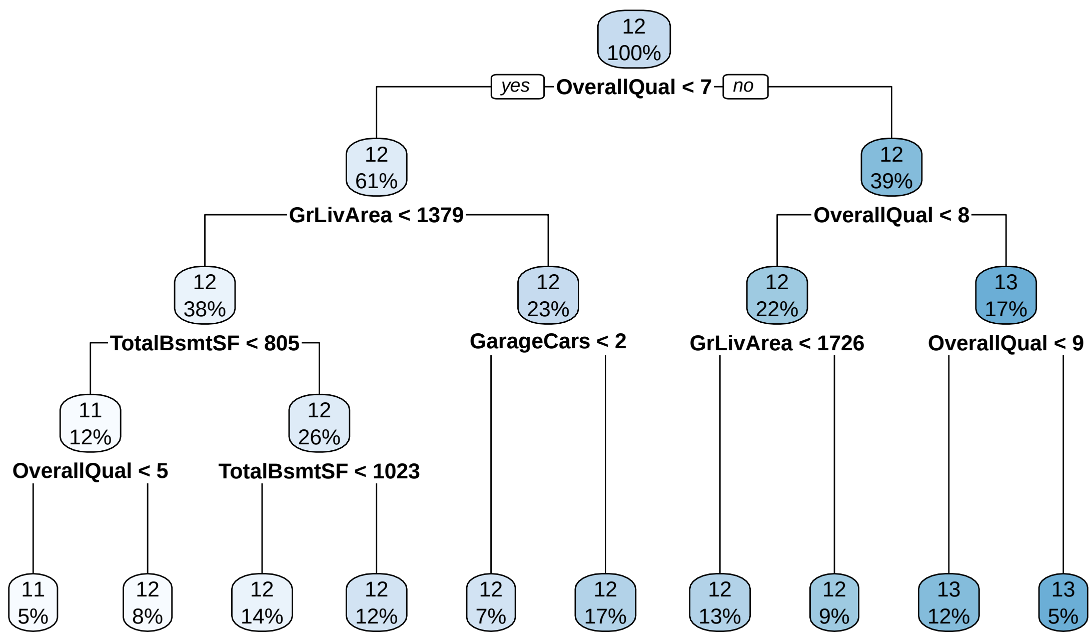
Print the results and make a plot to visualize the results.

```
cp_table<-as.data.frame(printcp(fit)) # display the results
```

```
##
## Regression tree:
## rpart(formula = logSalePrice ~ . - SalePrice, data = training_set,
##     method = "anova")
##
## Variables actually used in tree construction:
## [1] GarageCars  GrLivArea   OverallQual TotalBsmtSF
##
## Root node error: 212.15/1258 = 0.16864
##
## n= 1258
##
##          CP nsplit rel error  xerror     xstd
## 1  0.466218      0   1.00000 1.00102 0.046832
## 2  0.083398      1   0.53378 0.53474 0.028271
## 3  0.078935      2   0.45038 0.46066 0.025807
## 4  0.045609      3   0.37145 0.38185 0.022047
## 5  0.020161      4   0.32584 0.33548 0.019165
## 6  0.017215      5   0.30568 0.32237 0.018147
## 7  0.014444      6   0.28846 0.31373 0.017812
## 8  0.013189      7   0.27402 0.30478 0.017219
## 9  0.011996      8   0.26083 0.29778 0.016654
## 10 0.010000      9   0.24884 0.29723 0.016648
```

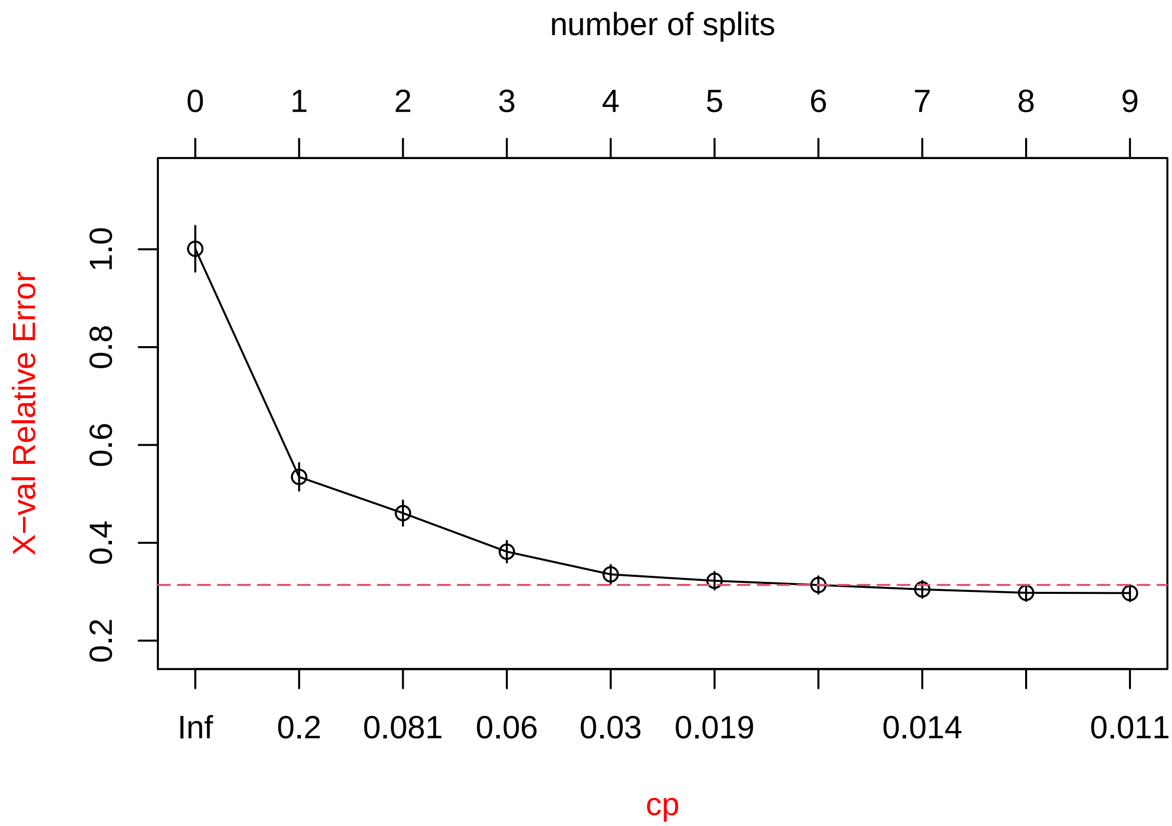
```
rpart.plot(fit,
           fallen.leaves=TRUE,
           main="Regression Tree using rpart")
```

Regression Tree using rpart



Plot a Complexity Parameter Table for our fitted model.

```
plotcp(fit,
       minline = TRUE,
       lty = 5,
       col.lab = "red",
       col=2,
       upper="splits") # visualize cross-validation results
```

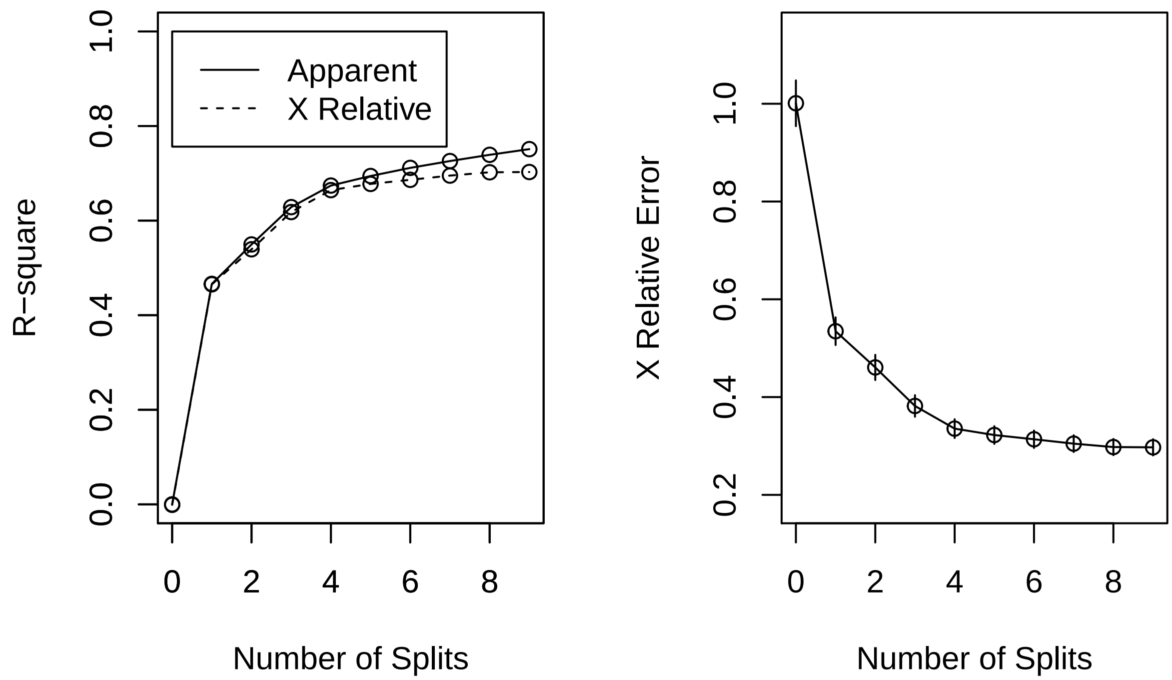


Plot the Approximate R-Square for different Splits.

```
par(mfrow=c(1,2)) # two plots on one page
rsq.rpart(fit) # Plots the Approximate R-Square for different Splits
```

```
##
## Regression tree:
## rpart(formula = logSalePrice ~ . - SalePrice, data = training_set,
##     method = "anova")
##
## Variables actually used in tree construction:
## [1] GarageCars  GrLivArea   OverallQual TotalBsmtSF
##
## Root node error: 212.15/1258 = 0.16864
##
## n= 1258
##
##          CP nsplit rel error  xerror     xstd
## 1  0.466218      0   1.00000 1.00102 0.046832
## 2  0.083398      1   0.53378 0.53474 0.028271
## 3  0.078935      2   0.45038 0.46066 0.025807
## 4  0.045609      3   0.37145 0.38185 0.022047
## 5  0.020161      4   0.32584 0.33548 0.019165
## 6  0.017215      5   0.30568 0.32237 0.018147
## 7  0.014444      6   0.28846 0.31373 0.017812
## 8  0.013189      7   0.27402 0.30478 0.017219
```

```
## 9  0.011996      8   0.26083 0.29778 0.016654
## 10 0.010000      9   0.24884 0.29723 0.016648
```



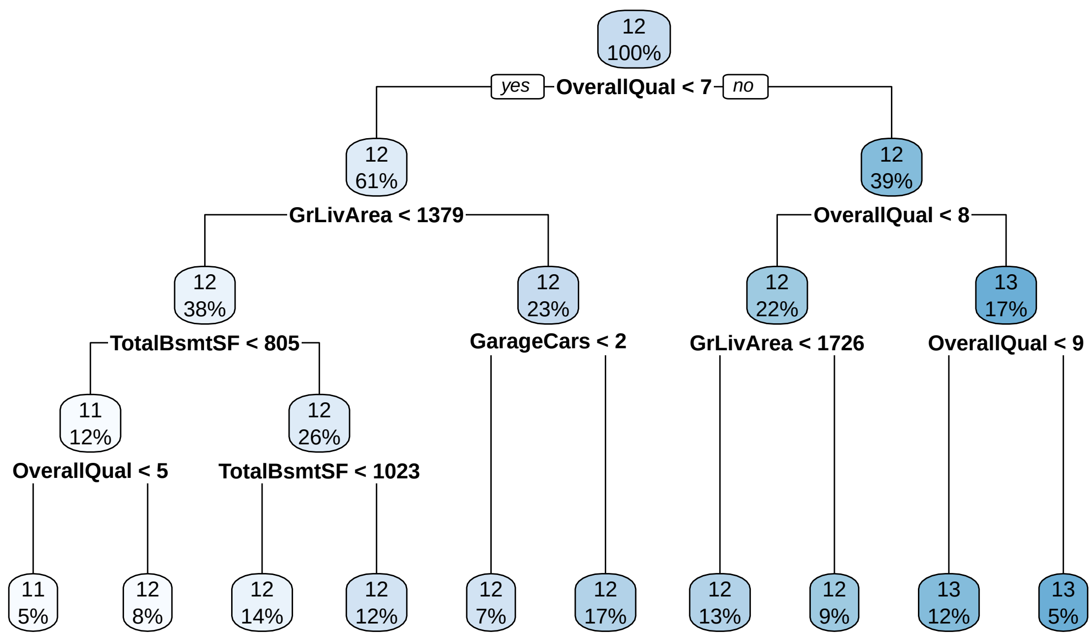
Prune the tree and plot the pruned tree. We begin by first identifying the minimum value for **xerror** and looking for the corresponding **cp** value to prune the tree from.

```
min_xerror<-min(cp_table$xerror)
min_xerror # 0.2972265
```

```
## [1] 0.2972265
```

```
cp_4_min_xerror<- cp_table$CP[which.min(cp_table$xerror)]

pfit<- prune(fit, cp=cp_4_min_xerror) # # prune the tree using the cp value from cptable

# plot the pruned tree
rpart.plot(pfit,
           fallen.leaves=TRUE,
           main=paste("Pruned Tree for cp=",
                      toString(cp_4_min_xerror),
                      " and min_xerror=",
                      min_xerror))
```

Pruned Tree for cp= 0.01 and min_xerror= 0.297226501215655



We essentially get the same tree even after pruning.

Let us now make predictions using our rpart fitted model.

```
cart_pred<- predict(fit, test_set,na.action = na.roughfix) # Make Predictions
```

Compute accuracy of this CART-based model.

```
acc_cart<- accuracy(cart_pred, test_set$logSalePrice) # Compute Accuracy
acc_cart<-as.data.frame(acc_cart)
```

Let us add the resulting RMSE value to our RMSE_Values table.

```
rmse_results <- bind_rows(rmse_results,
                          data.frame(
                            Model_Method="CART-based Regression Tree with Pruning",
                            RMSE_Values = acc_cart$RMSE)) # Add RMSE to our table
rmse_results %>% knitr::kable()
```

Model_Method	RMSE_Values
Random Forest	0.1076720
CART-based Regression Tree with Pruning	0.2063366

Linear Regression-based Models

We will now build a linear regression model using the lm() function using the training_set. **logSalePrice** is going to be a linear combination of multiple independent variables, the top 20 Boruta recommended features.

```
regressor = lm(formula = logSalePrice ~.-SalePrice, data = training_set)

# Let us review the contents of regressor using summary()
# summary(regressor)
# summary(regressor)$coefficients[,4]

# create a dataframe from model's output
tm = tidy(regressor)

# visualize dataframe of the model using non scientific notation of numbers
options(scipen = 999)
tm
```

```
## # A tibble: 34 x 5
##    term                estimate std.error statistic  p.value
##    <chr>                  <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)      7.53        0.885        8.52    5.21e-17
##  2 GrLivArea        0.00000103  0.000114     0.00904 9.93e- 1
##  3 OverallQual      0.0758      0.00637     11.9     6.57e-31
##  4 Second_Floor_SF  0.000139    0.000114     1.21    2.26e- 1
##  5 GarageCars       0.0791      0.0145       5.47    5.57e- 8
##  6 First_Floor_SF   0.000197    0.000118     1.68    9.34e- 2
##  7 TotalBsmtSF     -0.0000356   0.0000274   -1.30    1.93e- 1
##  8 YearBuilt       -0.000185    0.000381    -0.487   6.27e- 1
##  9 FullBath         0.00213     0.0129       0.166   8.68e- 1
## 10 ExterQualFa     -0.0803      0.0706      -1.14    2.55e- 1
## # ... with 24 more rows
```

Let us now identify statistically significant variables returned by our linear regression model. To do this, we will filter out the coefficients that possess a p-value <=0.05.

```
# get variables with p-value less than 0.05 (Statistically Significant)
signif_coeff<- tm %>% filter(tm$p.value <= 0.05)
```

Let us display these coefficients in ascending order of their p-values. The ones at the top are the most statistically significant.

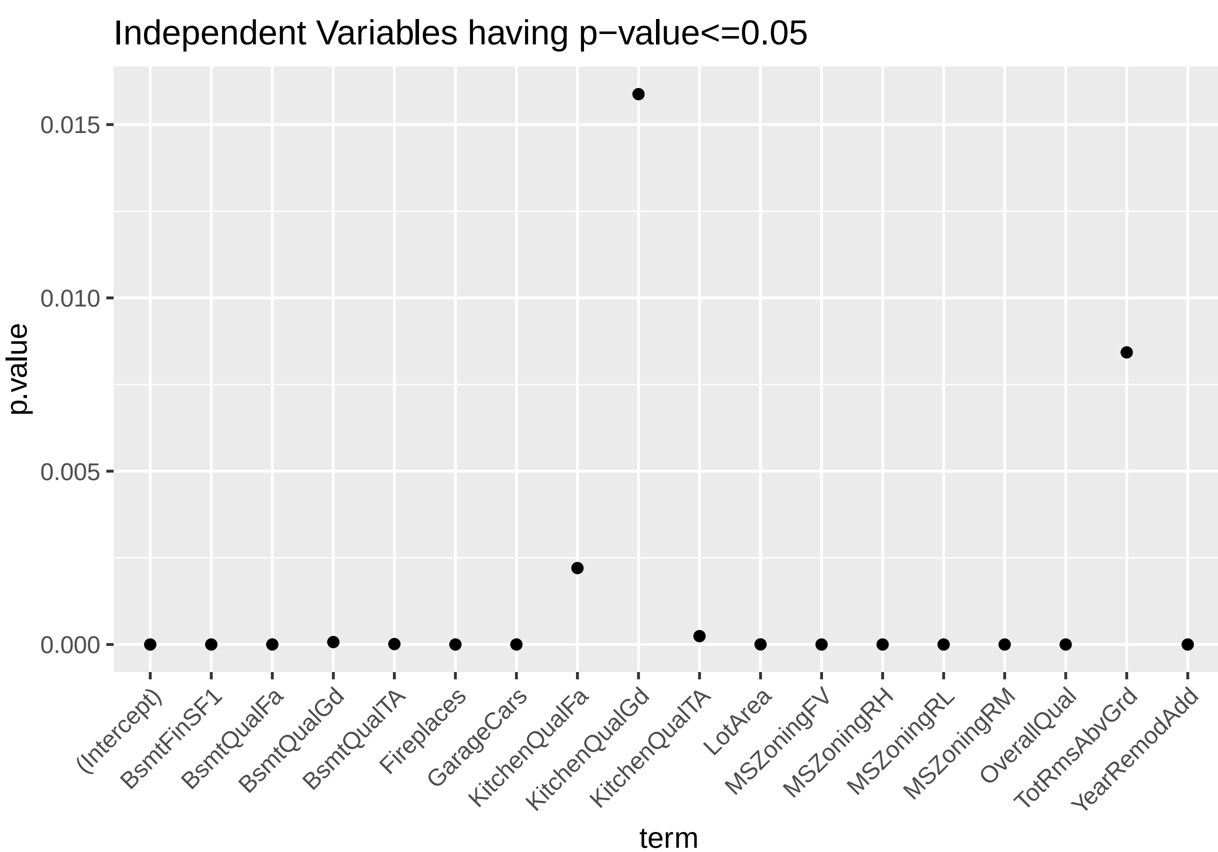
```
signif_coeff[order(signif_coeff$p.value),] # Display in ascending order of p.value
```

```
## # A tibble: 18 x 5
##    term              estimate   std.error statistic  p.value
##    <chr>                <dbl>       <dbl>     <dbl>    <dbl>
##  1 OverallQual    0.0758     0.00637          11.9   6.57e-31
##  2 (Intercept)    7.53       0.885             8.52 5.21e-17
##  3 MSZoningRL     0.470      0.0573            8.21 5.94e-16
##  4 MSZoningFV     0.487      0.0614            7.94 4.68e-15
##  5 YearRemodAdd   0.00212    0.000344          6.17 9.44e-10
```

```
##   6 MSZoningRM     0.347      0.0575           6.03 2.21e- 9
##   7 MSZoningRH     0.408      0.0721           5.66 1.93e- 8
##   8 GarageCars     0.0791     0.0145           5.47 5.57e- 8
##   9 BsmtFinSF1     0.0000636  0.0000118        5.38 9.19e- 8
##  10 Fireplaces     0.0437     0.00845          5.17 2.75e- 7
##  11 BsmtQualFa    -0.197      0.0402          -4.91 1.05e- 6
##  12 LotArea        0.00000211 0.000000445      4.73 2.48e- 6
##  13 BsmtQualTA    -0.108      0.0246          -4.39 1.25e- 5
##  14 BsmtQualGd    -0.0789     0.0198          -3.99 6.99e- 5
##  15 KitchenQualTA -0.0939     0.0255          -3.68 2.41e- 4
##  16 KitchenQualFa -0.130      0.0425          -3.07 2.21e- 3
##  17 TotRmsAbvGrd   0.0141     0.00534          2.64 8.43e- 3
##  18 KitchenQualGd -0.0534     0.0221          -2.42 1.59e- 2
```

Here's a visual on these coefficients using ggplot.

```
ggplot(signif_coeff, aes(x=term, y=p.value)) +
  geom_point(stat="identity") +
  theme(axis.text.x = element_text(angle=45, hjust=1, vjust = 1))+
  labs(title = "Independent Variables having p-value<=0.05")
```

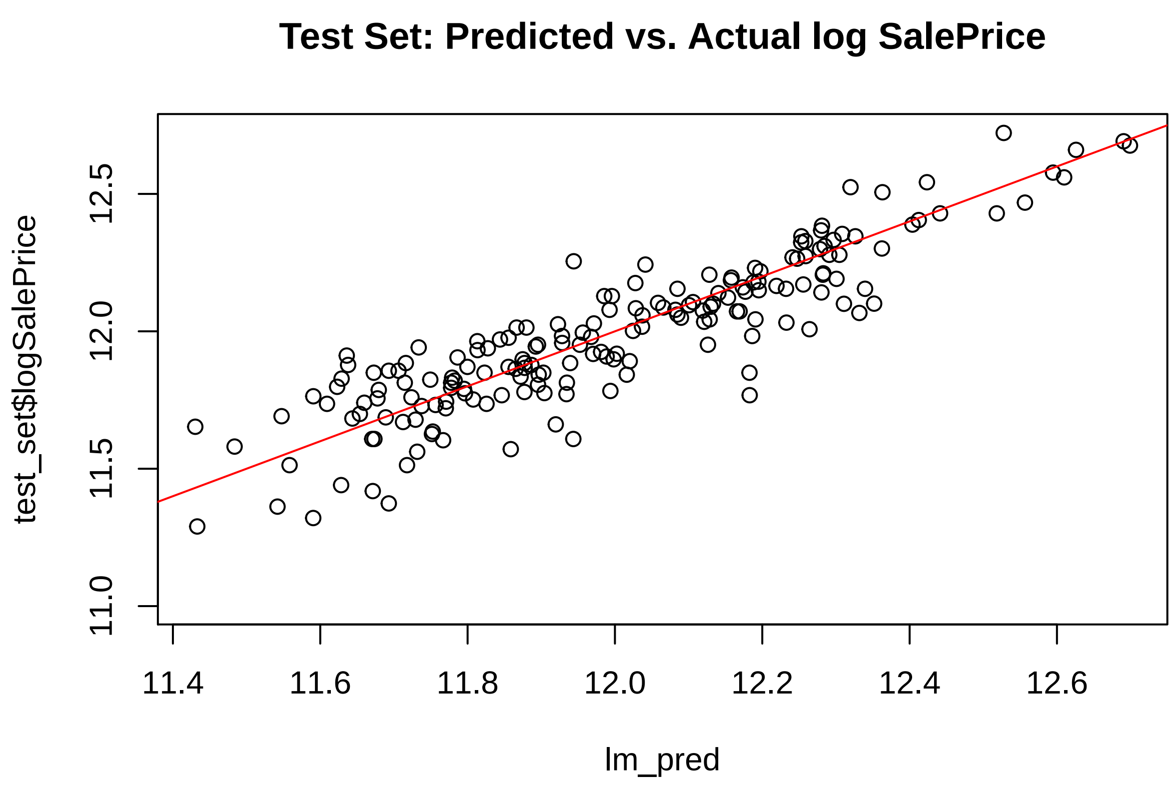


Let us make our initial prediction against the test_set using the predict() function.

```
lm_pred <- predict(regressor,test_set,type = "response") # Make predictions
```

Compute residuals.

```r
residuals <- test_set$logSalePrice - lm_pred # Compute residuals

linreg_pred <- data.frame("Predicted" = lm_pred,
                          "Actual" = test_set$logSalePrice,
                          "Residual" = residuals)

plot(lm_pred,
     test_set$logSalePrice,
     main = "Test Set: Predicted vs. Actual log SalePrice")
abline(0,1,col="red")
```



Compute accuracy for our linear regression model.

```r
acc_lm<-as.data.frame(accuracy(lm_pred, test_set$logSalePrice))
acc_lm
```

```
##                   ME      RMSE        MAE        MPE      MAPE
## Test set -0.01206446 0.1253918 0.09514597 -0.1080535 0.7991163
```

Compute the RMSE on this initial model to get a sense on our model's ability to make good predictions.

```r
rmse_results <- bind_rows(rmse_results,
                          data.frame(
                            Model_Method="Linear Regression",
                            RMSE_Values = acc_lm$RMSE)) # Add RMSE to our table
rmse_results %>% knitr::kable()
```

Model_Method	RMSE_Values
Random Forest	0.1076720
CART-based Regression Tree with Pruning	0.2063366
Linear Regression	0.1253918

Insights Gained so Far

We make the following findings:

1. Of the 3 models we experimented with, Random Forest yielded the lowest RMSE of 0.1076720.
2. The CART-based model yielded the highest RMSE and despite pruning for the lowest value of xerror, the Regression Tree remained the same. The CART model did allow us to visualize how decisions are made at the various splits for the important features included in our training_set.
3. The features that were recommended by Boruta turned out to be very valuable and were validated to have significant importance as confirmed by the varImpPlot and the statistically significant variables identified in Linear Regression ($p\text{-value} < 0.05$).

Using RMSE as our metric for model selection, we will now employ Random Forest against our Validation set.

Results

Predictions using Random Forest on Validation Set

As a final test before choosing this model, we will run this model against our our final holdout validation set. As noted in the **Overview** section, our validation set provided as *test.csv* file **does not** contain the outcome variable, SalePrice. Consequently, **we won't have anything to compare our predictions against** or compute RMSE against the validation set. **We will merely make predictions** against the validation set.

```
#' ###########################################################################
#' FINAL TEST against our hold-out validation set
#' ###########################################################################

final_pred_rf <- predict(RF, newdata=validation)

validation$logSalePrice<- final_pred_rf
validation$SalePrice <- exp(validation$logSalePrice)

# Reset NA with 0. We want this to help compute the min and max.
validation["SalePrice"][is.na(validation["SalePrice"])] <- 0
validation["logSalePrice"][is.na(validation["logSalePrice"])] <- 0

# Display the minimum SalePrice predicted for data in the validation set
validation_with_non_zero_SalePrice<- validation %>% filter(SalePrice>0)
min(validation_with_non_zero_SalePrice$SalePrice)
```

```
## [1] 56947.36
```

```
# Display the maximum SalePrice predicted for data in the validation set
max(validation_with_non_zero_SalePrice$SalePrice)
```

```
## [1] 478835.8
```

```
# Display the average SalePrice predicted for data in the validation set
mean(validation_with_non_zero_SalePrice$SalePrice)
```

```
## [1] 181872.5
```

Model Performance

The results documented in the above table demonstrate how we considered several modeling techniques and compare their accuracy to yield our chosen metric, **RMSE**. The **Random Forest** model yielded the best performance and we employed it against the final hold-out validation set to make our predictions as it yielded the **lowest RMSE score of 0.1076720**.

Conclusion

We started our analysis by data exploration and visualization, something that should always be undertaken to get a sense of what it is that needs to be analyzed. The findings of this exploration helped us evaluate if the data required some pre-processing or data wrangling work, as we call it.

After building a decent understanding of our data set, we employed **Feature Engineering** using knowledge obtained from **Boruta** to build 3 different models using the top 20 Boruta-confirmed features - Random Forest, CART-based Regression Tree, and Linear Regression. We made predictions using these three models and tracked the resulting RMSE values that was helpful in comparing the chosen metric across each model.

The Random Forest-based model helped achieve a better performance as measured using the RMSE score of 0.1076720 and it was used to make predictions for our final hold-out validation set. Because our validation set did not include the outcome variable, we stopped our analysis at making predictions.

Chosen Model: We choose the **Random Forest-based model** for this project as it yielded the lowest RMSE score of 0.1076720.

Limitations

One of the biggest challenges with this dataset was the sheer number of independent variables that can require us to spend countless hours of data exploration. While Boruta helped us identify nearly 43 features, we limited our analysis to the top 20 of those recommended features for building our models. We chose to do so to avoid overfitting and the curse of dimensionality problem and try different modeling techniques.

Also, because we did not have the outcome available in our validation set, we had to stop our analysis with making predictions. As such, we could not compute the accuracy of our model against the validation set but that is exactly how real world works. We use some level of supervised learning to train our model and put it to use against new data to make predictions.

Future Work

We can reexamine the remaining 23 features confirmed by Boruta for our future analysis and see if there are additional features that significantly impact our outcome variable, SalePrice. We can also explore other modeling techniques instead of limiting ourselves to the three that we tried in this project.