HOUSING PRICE – DATA ANALYSIS AND PREDICTION

ABSTRACT
This paper outlines the data analysis and machine learning methodology to analyze and predict housing price from medium-sized data sample.

By: Zach Nguyen

Contents

Executive summary

Despite having been around for a long time, Real Estate Price is one of the most unpredictable commodities in the economy. Many buyers and sellers are often unsure of the true value of their properties or those they want to buy. Traditional market research methods such as Similar home or Precedent price comparison introduce significant personal bias and subjectivity. Thus, financial mistakes are myriad in the bidding and negotiation process. This report aims to describe the process of making use of the big data available recently and cutting-edge data science and machine learning methodologies to predict the price of future Real Estate. As another point of reference for property price that is easy to use and interpret, this model can provide Real Estate participants with an advantage in negotiation. To build the model, a sample dataset was taken from Kaggle's "House Prices: Advanced Regression Techniques" and the R Programming language was use as tool of analysis.

Section 1: Data preparation is a minor description of the importing of the sample data and various R packages a practitioner may need when programming and reproducing the methodology. *Section 2: Data cleaning, wrangling and Imputation* describes the data preprocessing phases. Data in real life rarely conforms to ideal expectations of the data scientist. Thus, various techniques to visualize, clean and fill in missing data are conducted. *Section 3: Exploratory Data Analysis* use visualizations to familiarize the analyst with the structure, state of, and correlation between the dependent variable (Sale Price of the property), in which we hope to predict, and the independent variables, from which we base our predictions. This section also introduces a random forest implementation to grasp important variables that the analyst may need to scrutinize.

Section 4: Feature engineering/Extraction helps convert the survey data into human-sensible data and extract new features (predictors) based on domain understanding. These features reflect real-life concerns of a typical homebuyer looking for a piece of real estate, as surveyed by the analyst. Binary, binning and feature interaction were the techniques used to achieve feature engineering. In *Section 5: Encoding the data to be used in Machine Learning*, the analyst converts the data into machine-sensible data (using dummy variables), reduce any data bias that may prevent the machine from learning efficiently (with normalization) and remove outliers and non-zero-variance data that would introduce unnecessary noise to the model. Finally, *Section 6: Model Selection and Model Training* explains the rationale in selecting a regression model and its modification, coupled with implementation and results obtained by the model.

Overall, you would expect the model to perform with 11.4% mean error in real life. This means that it is overwhelmingly confident that the real value of a $100,000 property will lie between $90,000 and $110,000. The model's regressive nature also introduce interpretability in the form of co-efficient and may help to explain the reason behind the price of certain properties. *Section 7: Conclusion and Risk assessment* will finalize the report and recommend future improvements to the model.

Section 1: Data preparation

This section describes the preliminary data preparation for the rest of the analysis. The data sample was imported from Kaggle.com to build the model. It represents data collected on residential homes in Ames, Iowa. Thus, it is highly advisable for an analyst to find and conduct similar techniques, but on dataset regarding to homes in proximate locations.

The following R packages were used to help with the analysis:

```
# Loading libraries

library(dplyr)             ## Used for Data manipulation, data wrangling
library(tidyr)             ## Used for Data manipulation, data wrangling
library(VIM)               ## Used to visualization of missing variables
library(mice)              ## Used for imputation of missing variables
library(psych)             ## Used to get Descriptive statistics
library(ggplot2)           ## Used to plot graphs
library(gridExtra)         ## Used to arrange visualization
library(corrplot)          ## Used for drawing correlation plots
library(caret)             ## Used for regression
library(randomForest)      ## Used to conduct random forest

```

Data was also imported, and non-predictor variables such as ID removed:

```
# Importing the data
train <- read.csv("train.csv", stringsAsFactors = FALSE)
test <- read.csv("test.csv", stringsAsFactors = FALSE)

test_id <- test$Id
test$Id <- NULL
train$Id <- NULL

test$SalePrice <- NA
full <- rbind(train,test)
```

Section 2: Data cleaning, wrangling and Imputation

This section depicts the steps to clean and fill in missing data from the dataset. Upon closer look at the dataset, we recognize that some of the data is missing, some are illogical/incorrect, and some are too few and far in between to matter.

We find and clean some illogical data points:

```
# Find any year value above 2010 and replace with suitable values
full %>%
  select(GarageYrBlt, YrSold, YearBuilt, YearRemodAdd) %>%
  filter_all(any_vars(.>2010))
full[which(full$GarageYrBlt  > 2010), "GarageYrBlt"] <- 2007
```

```
# Find illogical year value (Sold before Built, Remodeled before Built, Garage Built before Built)
full %>%
  select(GarageYrBlt, YrSold, YearBuilt, YearRemodAdd) %>%
  filter(YrSold < YearBuilt | YearRemodAdd < YearBuilt | GarageYrBlt < YearBuilt)
full[which(full$YearBuilt > full$YrSold), "YrSold"] <-  2008
full[which(full$YearBuilt > full$YearRemodAdd), "YearRemodAdd"] <- 2002
```

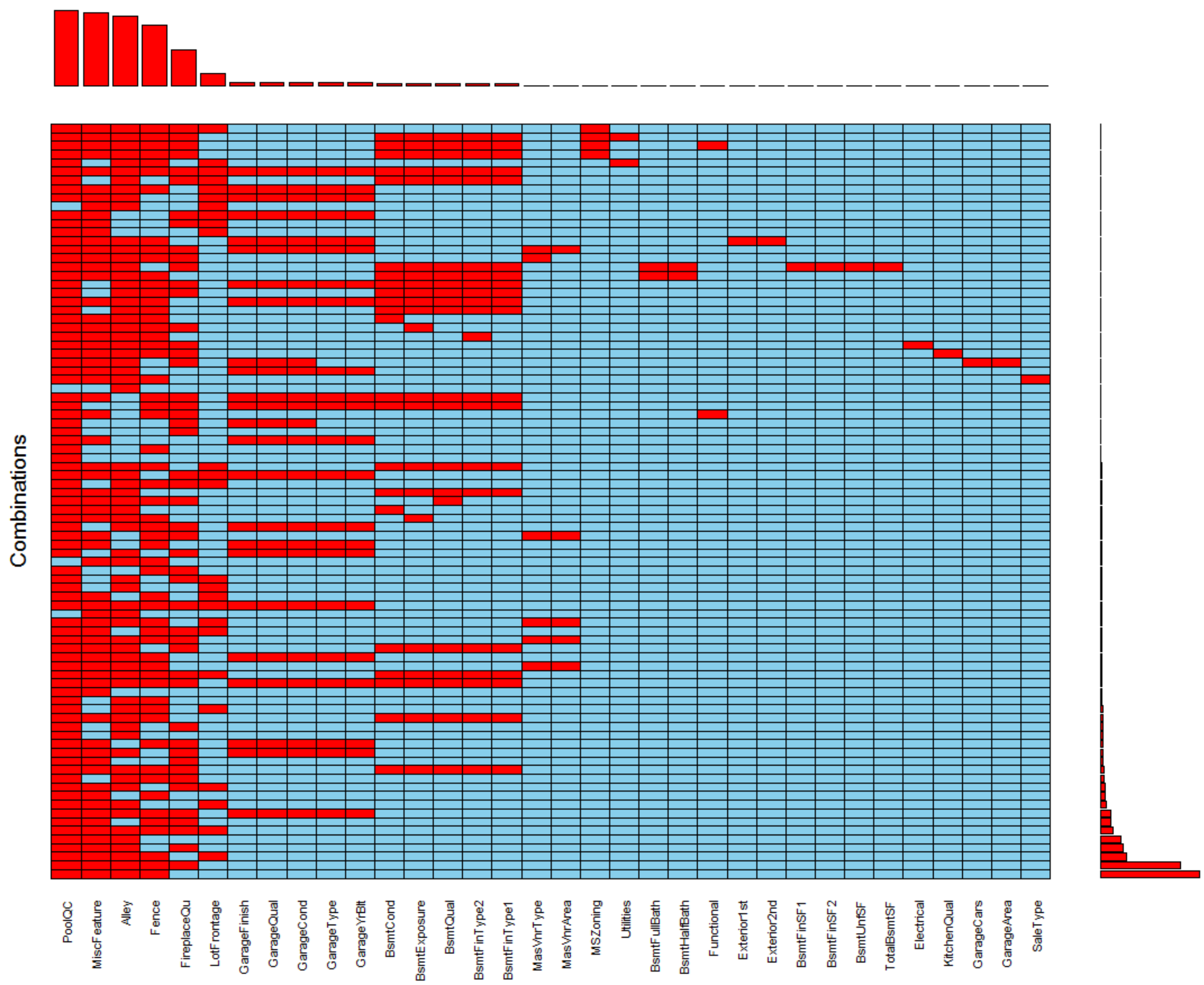
```
# Find properties with missing garage year built, set them to year built
which(!is.na(full$GarageType) & is.na(full$GarageYrBlt))
full[c(2127,2577), c("GarageType", "GarageYrBlt", "YearBuilt")]
full[c(2127,2577), "GarageYrBlt"] <- full[c(2127,2577), "YearBuilt"]
```

We find sparse categories and assign them to the nearest neighbor:

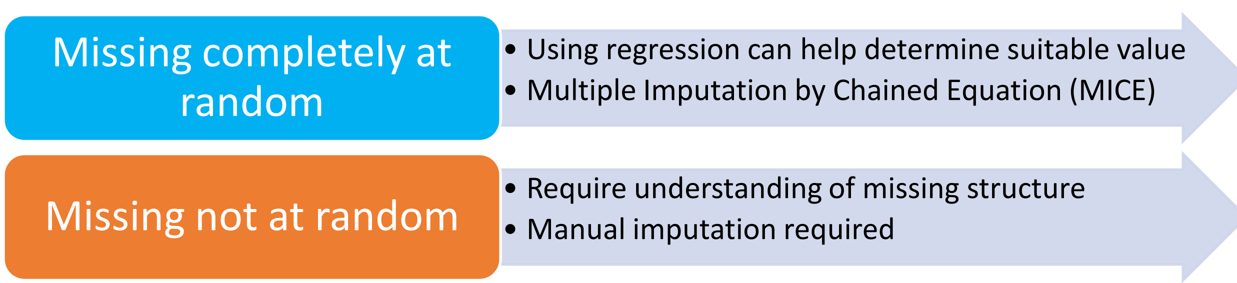
```
# Find Predictors catagories with too few data point and assign to nearest neighbors
which(full$MSSubClass == 150)
full[2819, "MSSubClass"] <- 160

which(full$TotRmsAbvGrd == 13 | full$TotRmsAbvGrd == 14 | full$TotRmsAbvGrd == 15)
full[c(636,1903,2550), "TotRmsAbvGrd"] <- 12

which(full$Fireplaces == 4)
full[2711, "Fireplaces"] <- 3

which(full$GarageCars == 5)
full[1829, "GarageCars"] <- 4
```

In order to tackle missing data, we first visualize the structure of the missing data:

```
# Visualize the missingness of the data

var_with_missing <- colnames(full[colSums(is.na(full)) >0])
var_with_missing <- var_with_missing[-35]
aggr(full[,var_with_missing],
     combined = TRUE, bars = TRUE, sortVars = TRUE, cex.axis = 0.69)
```



We also recognize that while some missing data are Missing Completely at Random (MCAR), some missing data such as PoolQC (the variable with the most number data point missing) does not appear random and has a meaning according to the data dictionary provided. This mean those are Missing Not at Random (MNAR). To reasonably impute the missing data, we will manually fill in variables with MNAR data and use the MICE package to impute the rest (MCAR).

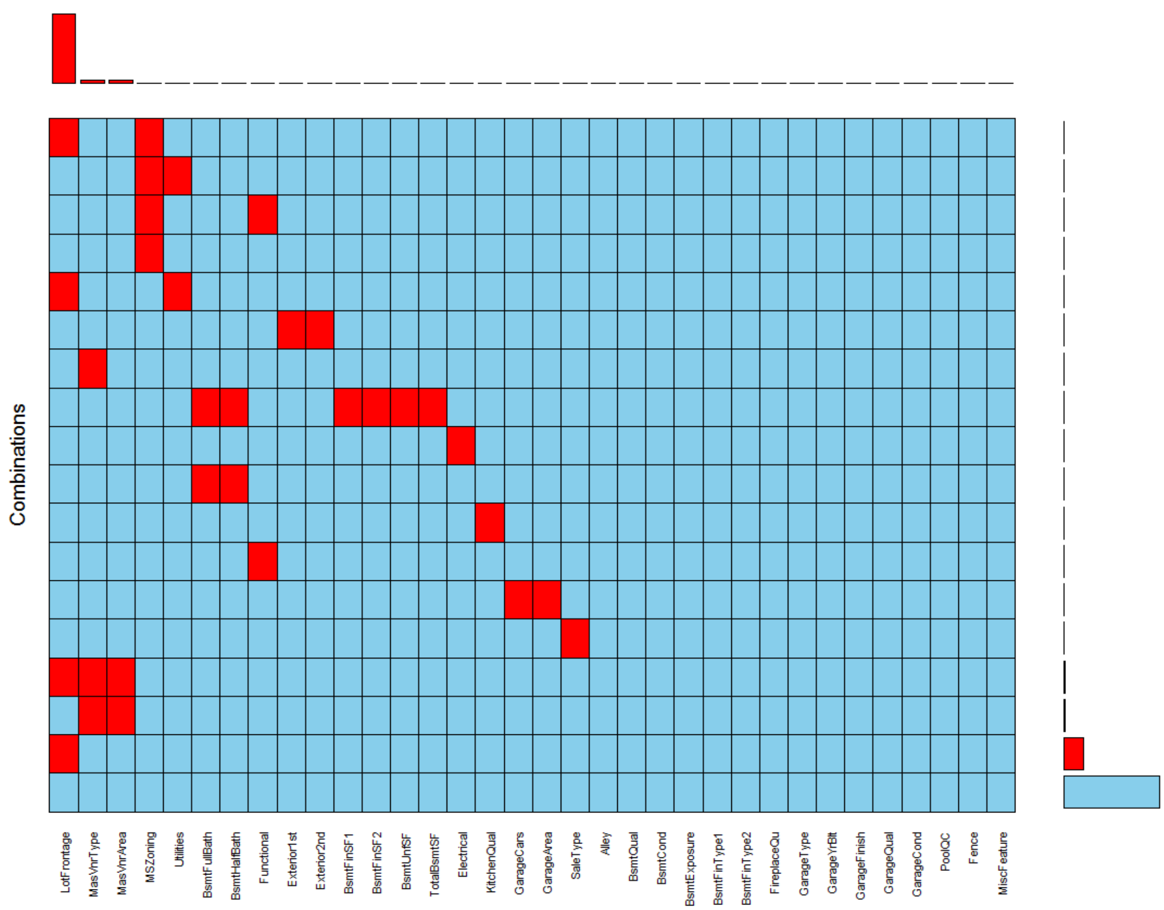


We fill in the data Missing not at random:

```r

# Group MNAR vars, where NA indicate the subject is not there
var_MNAR <- c("Alley", "BsmtQual", "BsmtCond","BsmtExposure","BsmtFinType1",
              "BsmtFinType2","FireplaceQu","GarageType", "GarageYrBlt",
              "GarageFinish","GarageQual","GarageCond","PoolQC","Fence",
              "MiscFeature")

# Create a function to fill in missing data
NA_to_NotAvailable <- function (data, predictor) {
                        levels(data[,predictor]) <- c(levels(data[,predictor]), "Not Available")
                        data[,predictor][is.na(data[,predictor])] <- "Not Available"
                        return(data[,predictor])
                        }

for (i in 1:length(var_MNAR)) {
                      full.MNAR[,var_MNAR[i]] <- NA_to_NotAvailable(full, var_MNAR[i])
                      }

```

And visualize the data after the imputation:

```r

# Visualize data after MNAR variables imputation
aggr(full.MNAR[,var_with_missing],
     combined = TRUE, bars = TRUE, sortVars = TRUE, cex.axis = 0.69)

```

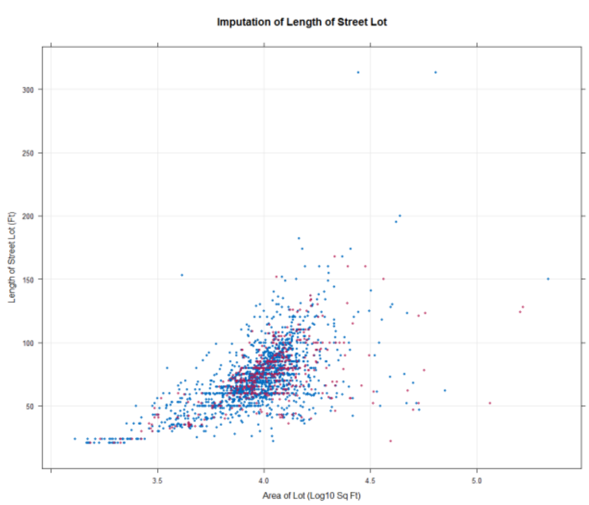


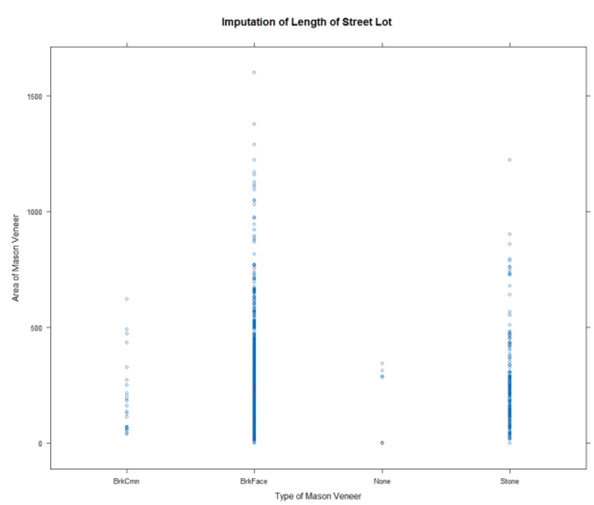
It seems that quite a lot of the missing data has been dealt with. Now we use M.I.C.E to impute the remaining missing data points.

```

# Group the remaining MAR variables
var_MAR <- c("MasVnrType", "MSZoning", "Utilities", "Functional",
             "Exterior1st", "Exterior2nd", "Electrical",
             "KitchenQual", "SaleType")

# Convert these into factors for mice imputation
full.MNAR <- full.MNAR %>%
  mutate_at(var_MAR, funs(factor(.)))

# We use cart method since imputation is fairly linear
imputed.data <- mice(full.MNAR[,!(names(full.MNAR) %in% "SalePrice")], m=1, method='cart')
full.complete <- mice::complete(imputed.data, action = 1, include = FALSE)

```

We visualize our imputation to make sure that the values are befitting:

```

# Visualize Lot Front and Mass Veneer which has the most imputation.
# We also use log transformation to linearize graph and aid visualization.
xyplot(imputed.data,
       LotFrontage ~ log10(LotArea), pch = 20, grid = TRUE,
       xlab = "Area of Lot (Log10 Sq Ft)",
       ylab = "Length of Street Lot (Ft)",
       main = "Imputation of Length of Street Lot")

xyplot(imputed.data,
       MasVnrArea ~ MasVnrType,
       xlab = "Type of Mason Veneer",
       ylab = "Area of Mason Veneer",
       main = "Imputation of Length of Street Lot")

```





The values are quite close, and that’s what we want.

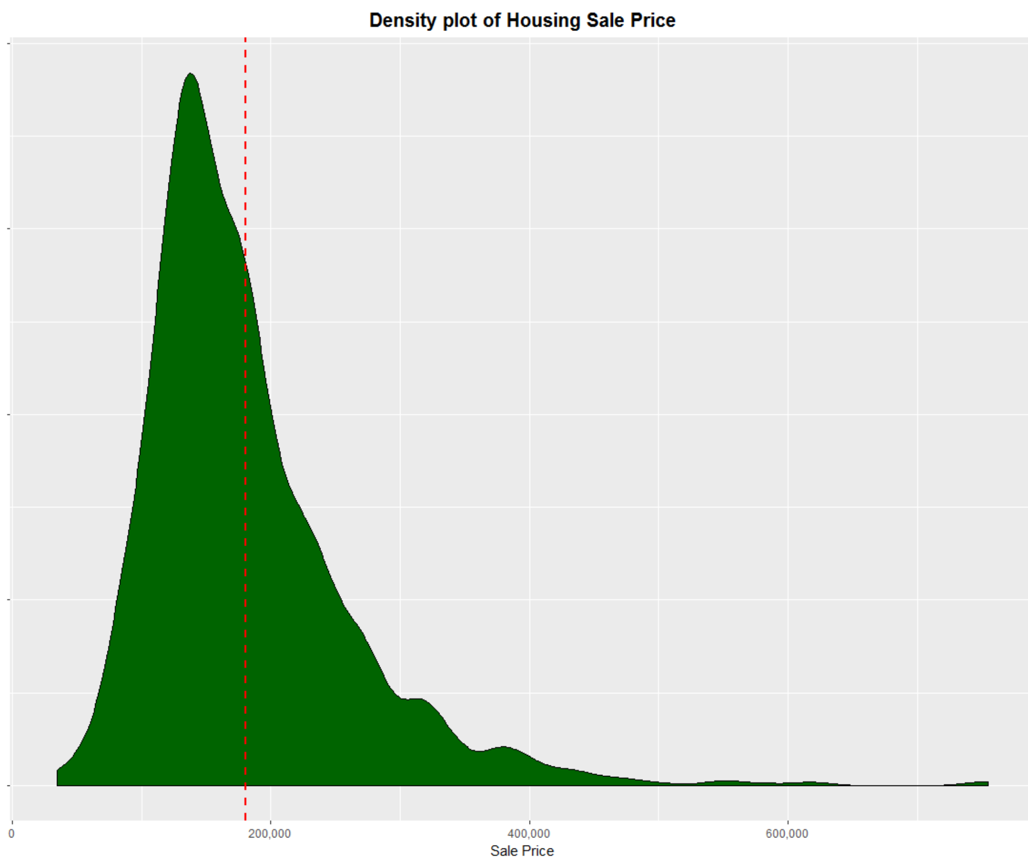
```

# Finalize our data set after imputation
train.complete <- full.complete[1:1460,]
train.complete$SalePrice <- train$SalePrice
test.complete <- full.complete[1460:2919,]

```

Section 3: Exploratory Data Analysis

This section helps the analyst gain intuition on the story the data is telling and aid in feature selection and sanity testing. Having a complete dataset, we can now explore the dataset to gain more understanding and intuition behind the numbers presented. It is, therefore, important to visualize the data structure of the dependent variable, some independent variable that stands out and their correlations.

First, we look at the dependent variable or response variable:

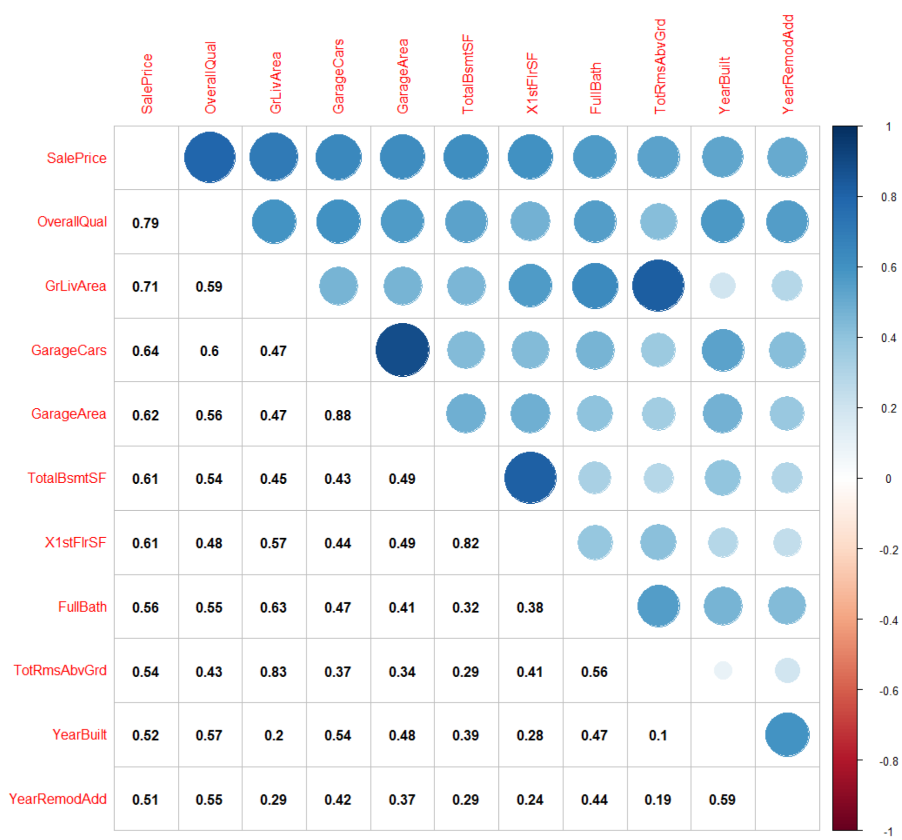
```r

 # Visualize the dependent variable SalePrice
density_plot_depvar <- ggplot(train.complete, aes(x= SalePrice)) +
  geom_density(color = "black", fill = "darkgreen") +
  geom_vline(aes(xintercept = mean(SalePrice)),
             color = "red", linetype = "dashed", size = 1) +
  scale_x_continuous(labels = scales::comma) +
  labs(title = "Density plot of Housing Sale Price", x = "Sale Price", y = "Density") +
  theme(plot.title = element_text(hjust = 0.5, size = 15, face = "bold"),
        axis.text.y = element_blank(),
        axis.title.y = element_blank())
density_plot_depvar

```



It seems that the average price of a property is around $175,000. However, many more properties have below average price while the expensive properties “tail off”. This makes sense, as a small percentage of houses is expected to have extraordinary features (mansion, huge land lots, etc. …) that command much higher price. Therefore, the Sale Price variable may be negatively skewed and will be normalized in later parts.

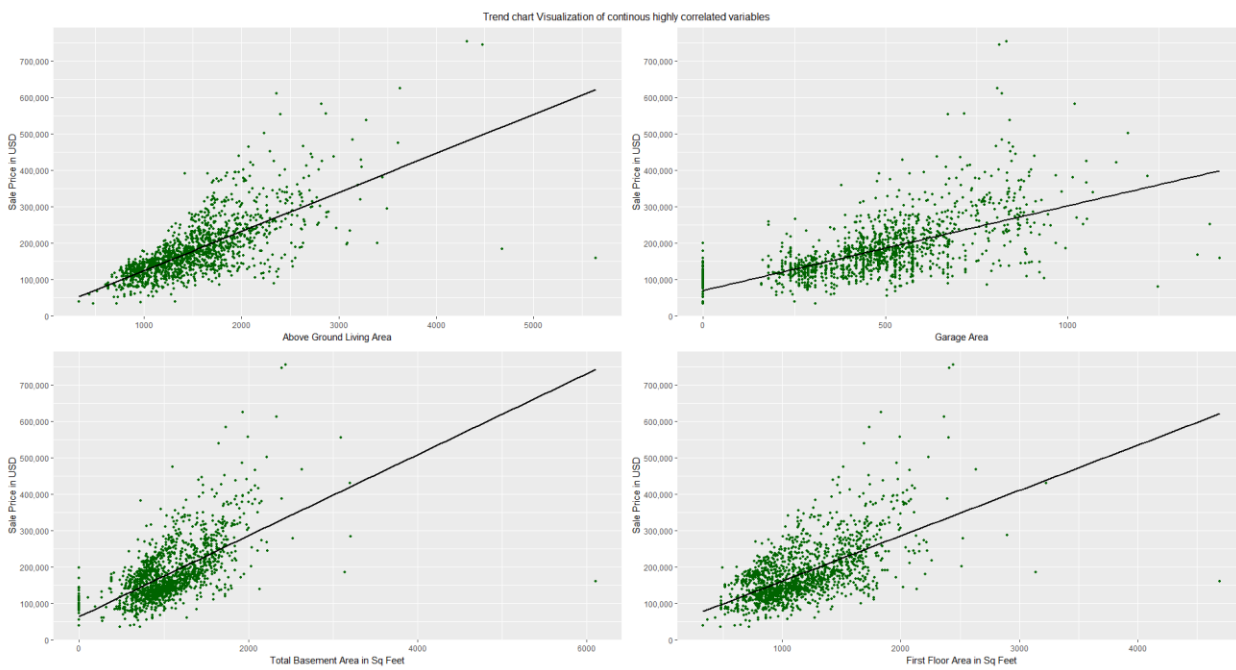
Next, we'll look at correlations of numeric variables to the Sale Price. This may hint us to some numeric variables that contributes significantly to predicting Sale Price and is, therefore, worth the effort.

```r
# Visualization of numeric variable correlations to SalePrice

numeric_var <- which(sapply(train.complete, is.numeric))
numeric_var_names <- names(numeric_var)

# Make correlation chart with all numeric variable
cor <- cor(train.complete[,numeric_var], use = "pairwise.complete.obs")

# Sort chart column in descending order
cor_sales_vector <- as.matrix(sort(cor[,"SalePrice"], decreasing = TRUE))

# Find name index of columns, row with high correlation
high_cor_var <- names(which(apply(cor_sales_vector,1, function(x) abs(x)> 0.5)))

# Plot correlation chart
cor.data <- as.data.frame(cor)[high_cor_var, high_cor_var] # Subset correlation chart
cor.data <- cor.data[order(-cor.data$SalePrice),] # Sort row in descending order
corrplot.mixed(as.matrix(cor.data), lower = "number", lower.col = "black", tl.pos = "lt")

```



It seems OverallQual score is understandably very highly correlated with SalePrice, along with GrLivArea, 1stFlrSF and TotalBsmtSF. People do prefer bigger houses after all. The YearBuilt and YearRemodAdd variables here also suggest people are looking for newer houses. It is also worth noting that there are highly correlated variables such as GarageCars and GarrageAreas which can cause multicollinearity and bias the model and must be eliminated in latter parts. For now, let's take a closer look at these variables (continuous and discreet) through visualization:

```r
# Visualization of relationships between predictors highly correlated to SalePrice

Visualize_numeric <- function(data, n_predictor, xname) {
  ggplot(data, aes(x = data[,n_predictor], y = SalePrice)) +
    geom_point(na.rm = TRUE, col = "darkgreen", size = 1) +
    geom_smooth(na.rm = TRUE, method = "lm", se=FALSE, color="black", aes(group=1)) +
    scale_y_continuous(breaks = seq( 0, max(data$SalePrice), by = 100000), labels = scales::comma) +
    ylab("Sale Price in USD") +
    xlab(xname)
}

p1 <- Visualize_numeric(train.complete, "GrLivArea", "Above Ground Living Area")
p2 <- Visualize_numeric(train.complete, "GarageArea", "Garage Area")
p3 <- Visualize_numeric(train.complete, "TotalBsmtSF", "Total Basement Area in Sq Feet")
p4 <- Visualize_numeric(train.complete, "X1stFlrSF", "First Floor Area in Sq Feet")
grid.arrange(p1, p2, p3, p4, ncol = 2, top = "Trend chart Visualization of continous highly correlated variables")
```



We can definitely see a trend line in the relationship between these variable and Sale Price, the dependent variable. These variables hold promise as important predictors for Sale Price. However, we also notice some outliers.

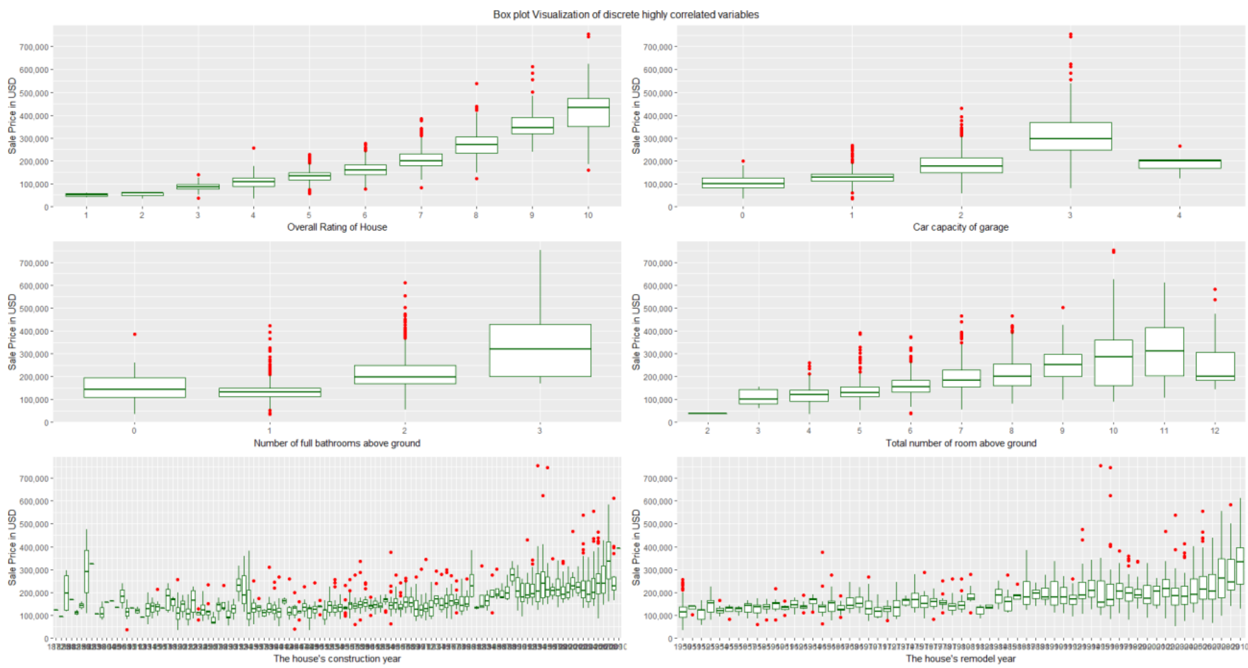
```r
# We can also spot the outliers

which(train$GrLivArea > 4500)
which(train$GarageArea > 1250)
which(train$TotalBsmtSF > 4000)
which(train$X1stFlrSF > 4000)
```

```
> which(train$GrLivArea > 4500)
[1]  524 1299
> which(train$GarageArea > 1250)
[1]  582 1191 1299
> which(train$TotalBsmtSF > 4000)
[1] 1299
> which(train$X1stFlrSF > 4000)
[1] 1299
```

It is necessary to consider removing observation 524 and 1299 at the end. Their low Sale Price despite huge square footage may signify consequence of an imperfect market (e.g.: family, heir transactions, under-the-floor deals …) or failed negotiation. Though we would generally expect some of these noises to get through our data collection, we can create a better fit model by removing them.

For now, we'll visualize discrete category variables from the correlation plot with Boxplots:

```r

# Visualization of discrete numeric variables
Visualize_category <- function(data, c_predictor, xname) {
  ggplot(data, aes(x= factor(data[,c_predictor]), y = SalePrice)) +
    geom_boxplot(na.rm = TRUE, col='darkgreen', outlier.color = "red") +
    scale_y_continuous(breaks= seq( 0, max(data$SalePrice), by = 100000), labels = scales::comma) +
    ylab("Sale Price in USD") +
    xlab(xname)
}

p5 <- Visualize_category(train.complete, "OverallQual", "Overall Rating of House")
p6 <- Visualize_category(train.complete, "GarageCars", "Car capacity of garage")
p7 <- Visualize_category(train.complete, "FullBath", "Number of full bathrooms above ground")
p8 <- Visualize_category(train.complete, "TotRmsAbvGrd", "Total number of room above ground")
p9 <- Visualize_category(train.complete, "YearBuilt", "The house's construction year")
p10 <- Visualize_category(train.complete, "YearRemodAdd", "The house's remodel year")
grid.arrange(p5, p6, p7, p8, p9, p10, ncol = 2, top = "Box plot Visualization of discrete highly correlated variables")

```



The correlation with SalePrice for these variables can be spotted, especially OverallQual (overall rating). Categories from GarageCars, Full Bath and TotRmsAbvGrd has very discrete boxes (in terms of box height and length) and suggests many areas of differences that could be explored. The Year variables, however, are too spread out (longer boxes) to see a definite trend.

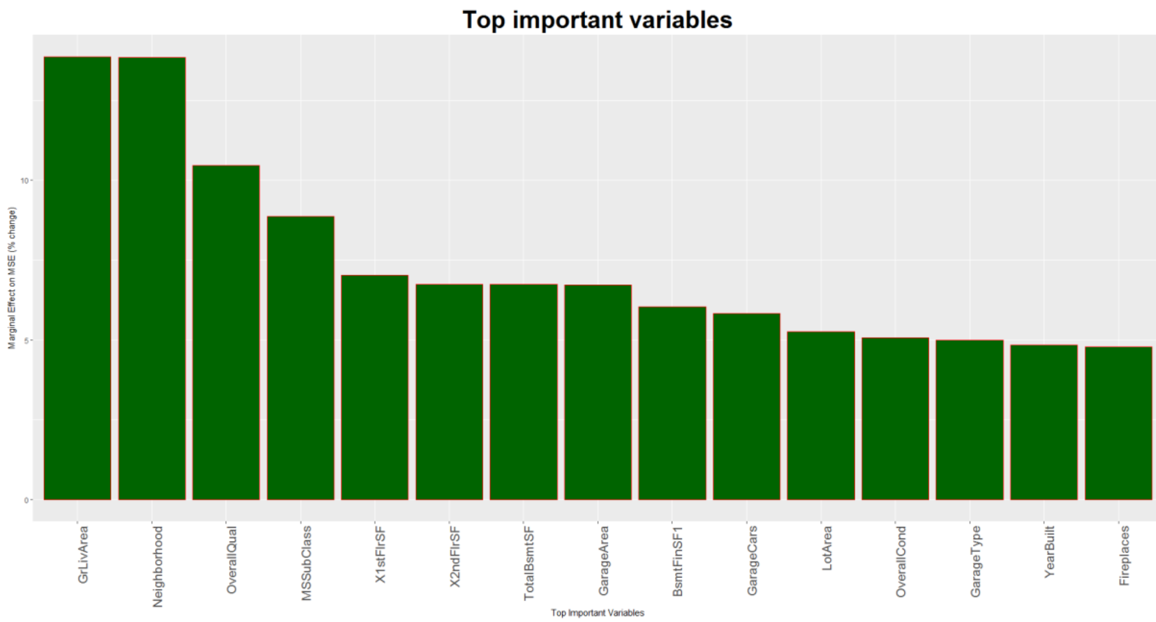
In short, the correlation plot gives us an idea of some classes of important numerical predictors, namely those related to overall quality, garage size and square footage of living spaces. However, we would want to explore important categorial variables as well. Therefore, the plan is to pick up important independent variables through a preliminary random forest implementation.

First, we need to categorize all our independent variables so that R can understand how to feed our data into the machine learning algorithm:

```r
# Specify Factor, ordered factor and numeric values
factor_var <- c("MSSubClass", "MSZoning", "Street", "Alley", "LandContour", "LotConfig",
                "Neighborhood", "Condition1", "Condition2", "BldgType", "RoofStyle", "RoofMatl",
                "Exterior1st", "Exterior2nd", "MasVnrType", "Foundation", "Heating", "GarageType",
                "PavedDrive", "MiscFeature", "SaleType", "SaleCondition", "CentralAir", "MSZoning",
                "HouseStyle", "Fence", "MoSold")
ordered_var <- c("LotShape", "Utilities", "LandSlope", "OverallQual",
                 "OverallCond", "ExterQual", "ExterCond", "BsmtQual",
                 "BsmtCond", "BsmtExposure", "BsmtFinType1", "BsmtFinType2", "HeatingQC", "Electrical",
                 "KitchenQual", "Functional", "FireplaceQu", "GarageFinish", "GarageCars",
                 "GarageQual", "GarageCond", "PoolQC", "GarageYrBlt",
                 "BedroomAbvGr", "KitchenAbvGr", "TotRmsAbvGrd", "Fireplaces")
numeric_var <- c("LotFrontage", "YearBuilt", "YearRemodAdd", "YrSold", "LotArea", "MasVnrArea",
                 "BsmtFinSF1", "BsmtFinSF2", "BsmtUnfSF", "TotalBsmtSF","X1stFlrSF", "X2ndFlrSF",
                 "LowQualFinSF", "GrLivArea", "GarageArea", "WoodDeckSF", "OpenPorchSF",
                 "EnclosedPorch", "X3SsnPorch", "ScreenPorch", "PoolArea", "MiscVal",  "FullBath",
                 "HalfBath", "BsmtFullBath", "BsmtHalfBath")
# Check: length(factor_var) + length(ordered_var) + length(numeric_var) # We have 80 variables
```

Then we train a random forest and use its feature importance methodology to pick out 20% of the most important predictors it considered. The predictors are chosen based on the sensitivity of the model's MSE to changes in said predictors. The 20% choice is a heuristic used only for data exploration, since 20% of predictors usually net 80% of the predictive power of the Machine Learning model.

```r
# Duplicate the dataset to be used in random forest
full.rf <- full.complete %>%
  mutate_at(factor_var, funs(factor(.))) %>%
  mutate_at(ordered_var, funs(ordered(.)))

 # str(full.rf)

# Train random forest and pick out 20% of important variables
train.rf <- full.rf[1:1460,]
train.rf$SalePrice <- train$SalePrice

set.seed(123)
Rf <- randomForest(x = train.rf[,-80], y=train.rf$SalePrice, ntree=100, importance=TRUE)
var.importance <- importance(Rf)
var.importance <- data.frame(Variables = row.names(var.importance), MSE = var.importance[,1])
var.importance <- var.importance[order(var.importance$MSE, decreasing = TRUE),]
var.importance$Variables <- as.factor(var.importance$Variables)

# Plot important predictor variables
ggplot(var.importance[1:(nrow(var.importance)/5),], aes(x = reorder(Variables, desc(abs(MSE))), y = MSE)) +
  geom_bar(stat = "identity", color = "red", fill = "darkgreen") +
  labs(title = "Top important variables", x = "Top Important Variables", y = "Marginal Effect on MSE (% change)") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 1, size = 16),
        plot.title = element_text(hjust = 0.5, size = 15, face = "bold"))
```



Random forest mostly supports our findings from the correlation plot (overall quality, square footage variables, garage and year built). From here, it is clear that most of our efforts should be on these variables. We also discovered important non-numeric categorical variables such as Neighborhood and MSSubClass.

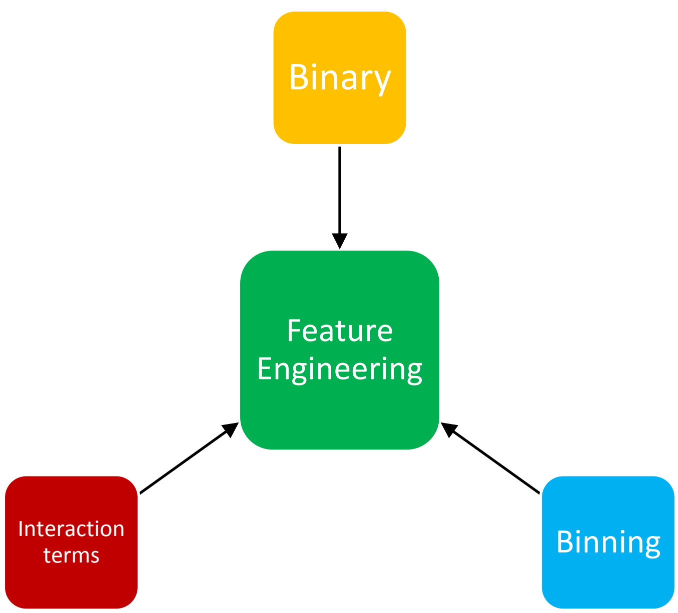
Section 4: Feature Engineering & Extraction

This section describes the feature engineering process. Feature engineering is extremely important in making accurate predictions. Machine Learning models running on insightful data will obtain much better results than those on raw collected data. Due to importance of domain knowledge to feature engineering, I have done market surveys with a sample of 17 of my friends and acquaintances in Toronto to figure out what are the most common questions one would ask when buying a house. Even though bias (demographic, geographic, personal) maybe introduced into the process, it is reasonable to assume that real-estate seekers in North America have more concerns in common than not.

To begin, let us build a function to visualize extractable predictors to see if the common concerns really match the data:

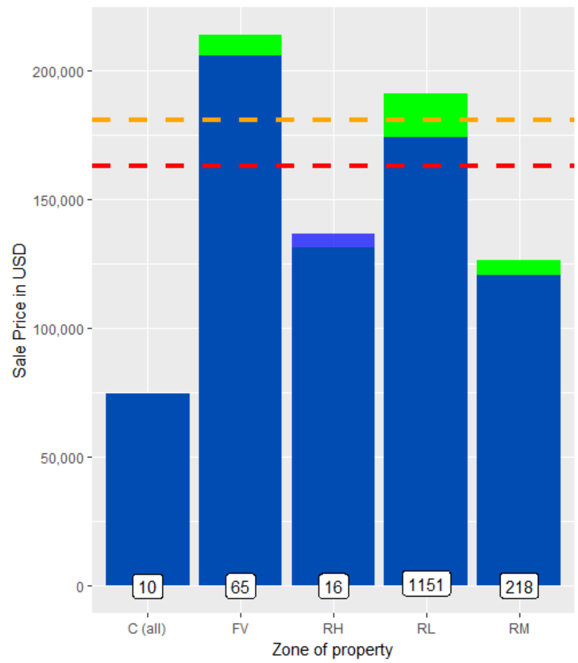
```r

  # Build a function to help visualize available variables to extract new sensible ones
full.final <- full.complete
Visualize_suspect_var <- function(data, s_predictor, xname) {
                          ggplot(data, aes(x = as.factor(data[,s_predictor]), y = data[,"SalePrice"])) +
                              geom_bar(stat = "summary", fun.y = "mean", fill = "green") +
                              geom_bar(stat = "summary", fun.y = "median", fill = "blue", alpha = 0.7) +
                              scale_y_continuous(labels = scales::comma) +
                              geom_hline(yintercept= median(data$SalePrice), linetype="dashed", color = "red", size = 1.5) +
                              geom_hline(yintercept= mean(data$SalePrice), linetype="dashed", color = "orange", size = 1.5) +
                              geom_label(stat = "count", aes(label = ..count.., y = ..count..)) +
                              ylab("Sale Price in USD") +
                              xlab(xname)
                          }
```

Let us also categorize the methods in which we can extract new features from outstanding ones:

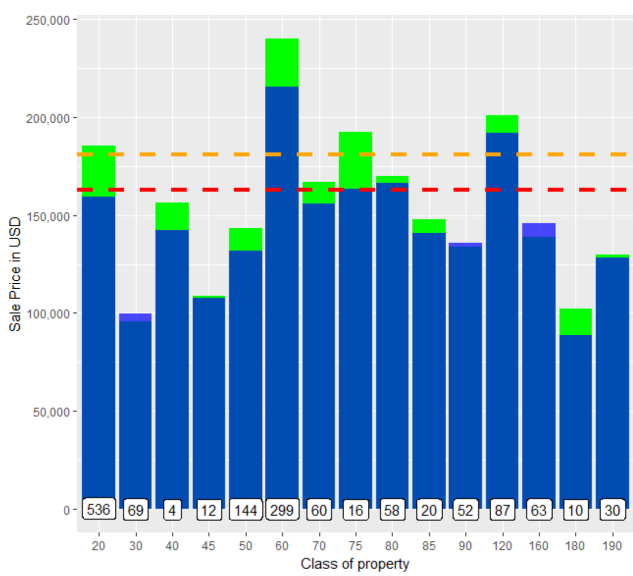


Some new features can be extracted using the survey's recurring questions and the visualization of the data to recognize distinct binary categories. These features describe whether the property is densely populated, new, bought in peak season, has deformed shapes, has detached garage, has paved drive, has been remodelled and bought under a mortgage. These all seem legitimate concerns that would be affect the price of a property. Thus, we devise the categories of the respective variables into reasonable binaries.

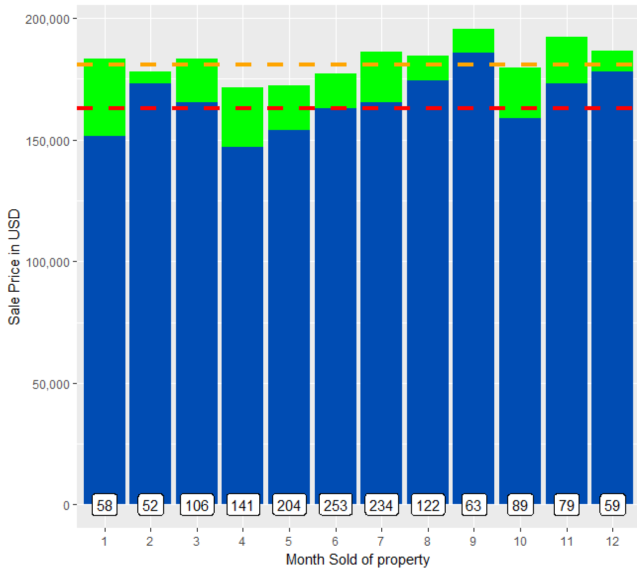
```
# Is the location densely populated?
Visualize_suspect_var(train.complete, "MSZoning", "Zone of property")
full.final$IsDensePop <- ifelse(full.final$MSZoning %in% c("FV", "RL"),"Yes", "No")
full.final$IsDensePop <- as.factor(full.final$IsDensePop)
```



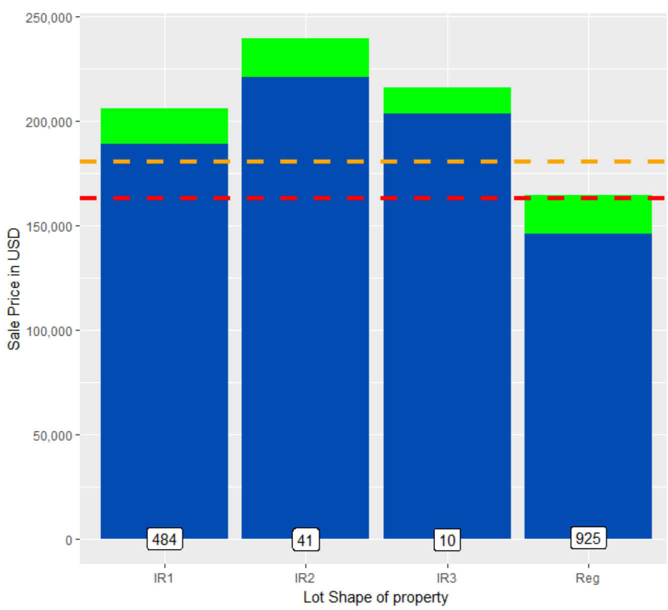

```
# Is the property newer than most?
Visualize_suspect_var(train.complete, "MSSubClass", "Class of property")
full.final$IsNewer <- ifelse(full.final$MSSubClass %in% c("20","60", "120","160"), "Yes", "No")
full.final$IsNewer <- as.factor(full.final$IsNewer)
```



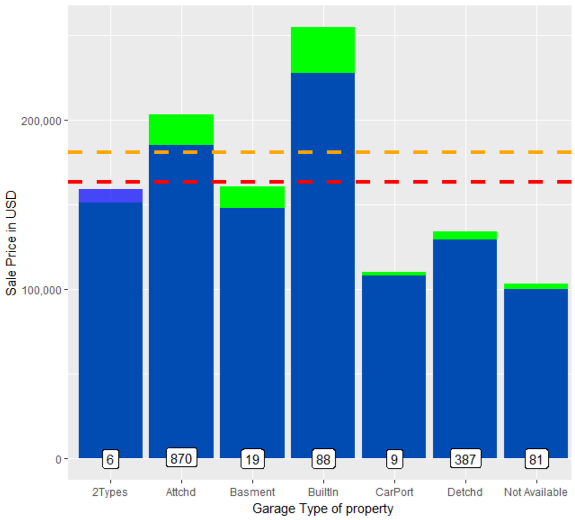
```r
# Was the property bought during peak season?
Visualize_suspect_var(train.complete, "MoSold", "Month Sold of property")
full.final$IsPeak <- ifelse(full.final$MoSold %in% c("3","4", "5","6", "7", "8"), "Yes", "No")
full.final$IsPeak <- as.factor(full.final$IsPeak)
```



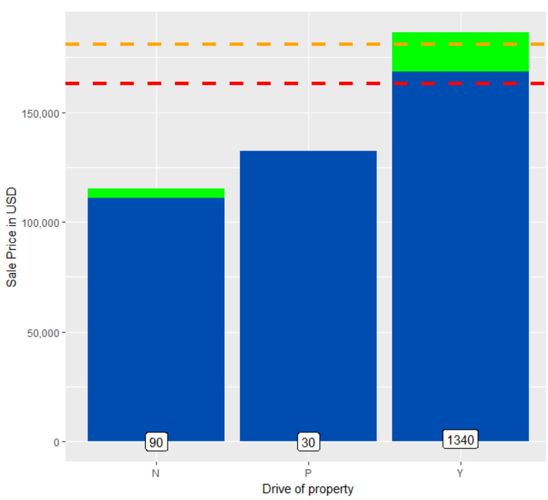

```r
# Is lot shape regular, any deformities?
Visualize_suspect_var(train.complete, "LotShape", "Lot Shape of property")
full.final$IsRegular <- ifelse(full.final$LotShape %in% c("Reg"), "Yes", "No")
full.final$IsRegular <- as.factor(full.final$IsRegular)
```



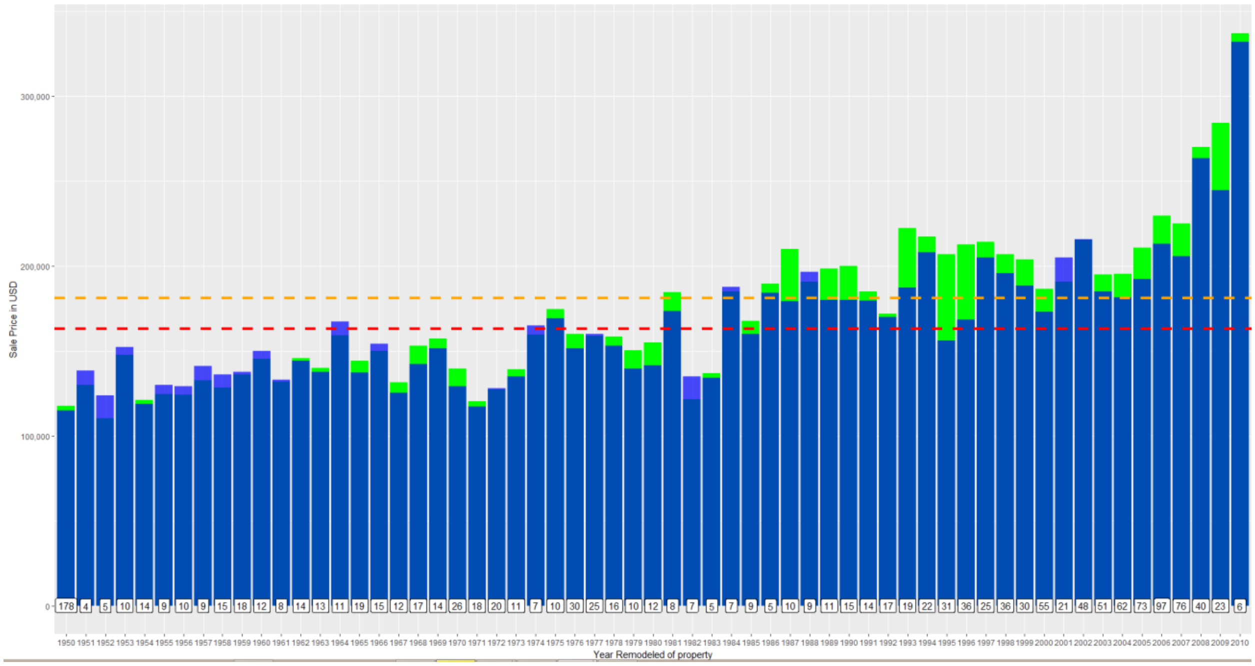
```
# Is Garage Detached?
Visualize_suspect_var(train.complete, "GarageType", "Garage Type of property")
full.final$IsDetached <- ifelse(full.final$GarageType %in% c("Detchd", "Not Available", "CarPort"),
                                "Yes", "No")
full.final$IsDetached <- as.factor(full.final$IsDetached)
```



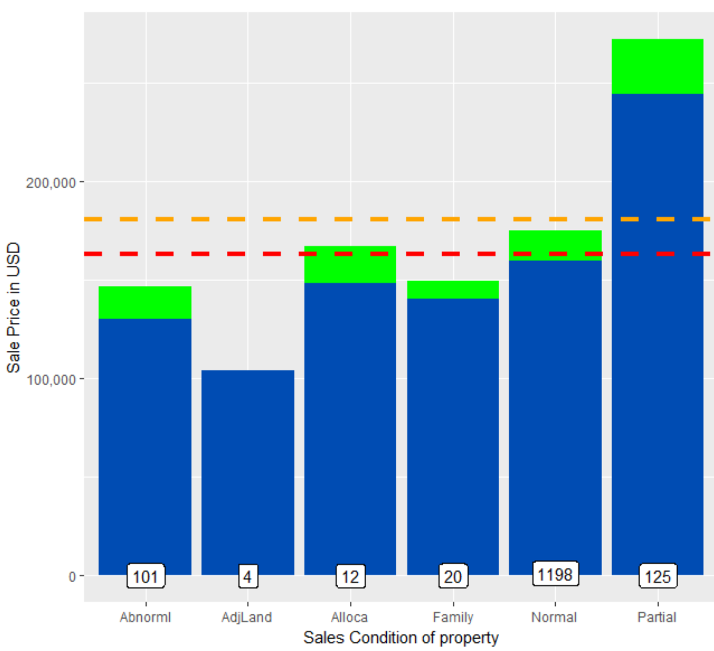

```
# Is the Drive Paved?
Visualize_suspect_var(train.complete, "PavedDrive", "Drive of property")
full.final$IsPaved <- ifelse(full.final$PavedDrive %in% c("Y"), "Yes", "No")
full.final$IsPaved <- as.factor(full.final$IsPaved)
```




```
# Has the house been changed/remodeled?
Visualize_suspect_var(train.complete, "YearRemodAdd", "Year Remodeled of property")
full.final$IsRemod <- ifelse(full.final$YearBuilt != full.final$YearRemodAdd, "Yes", "No")
full.final$IsRemod <- as.factor(full.final$IsRemod)
```



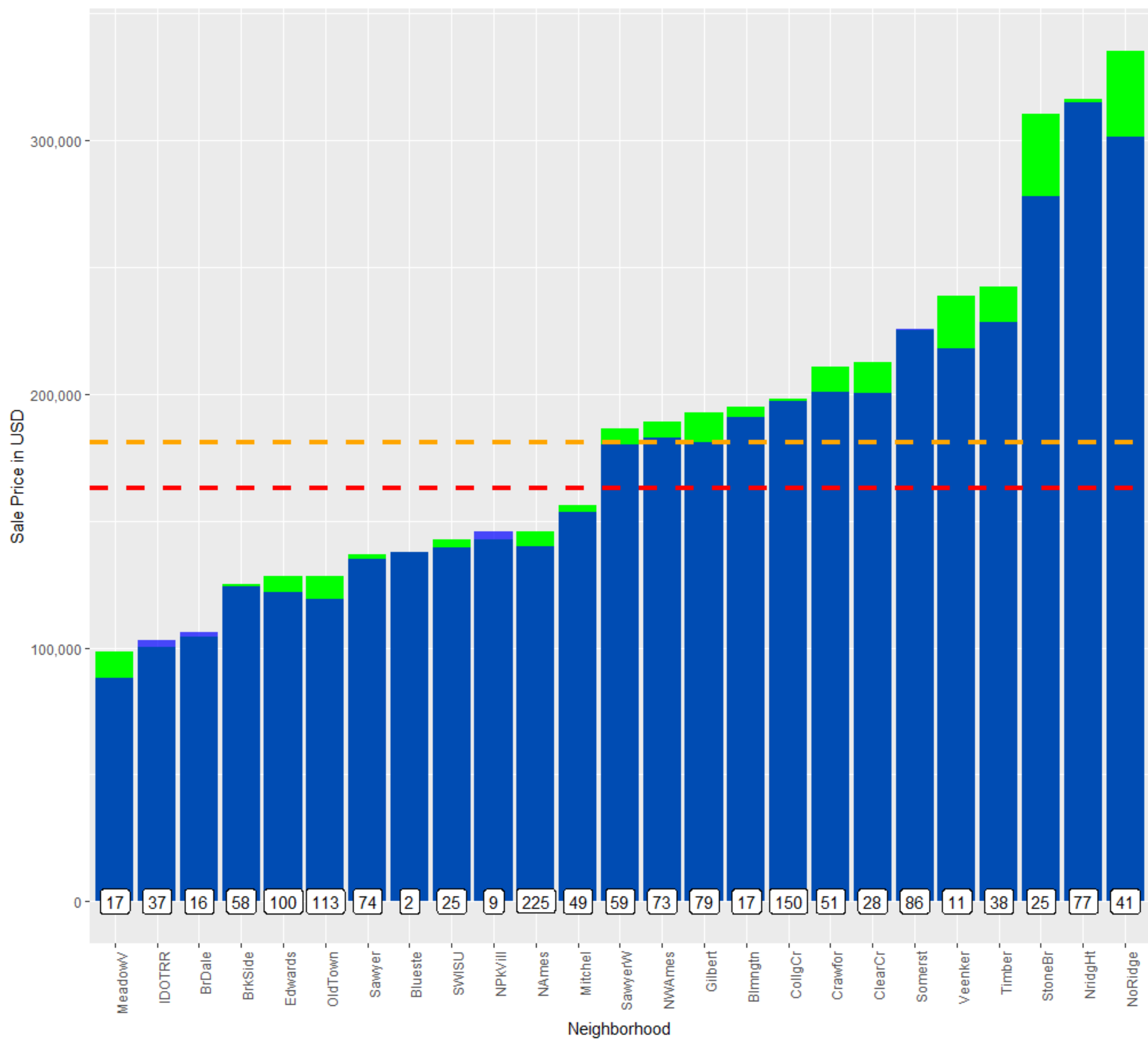

```
# Was the property bought under loan/mortgage?
Visualize_suspect_var(train.complete, "SaleCondition", "Sales Condition of property")
full.final$Invested <- ifelse(full.final$SaleCondition %in% c("Partial"), "Yes", "No")
full.final$Invested <- as.factor(full.final$Invested)
```



```r
# Does the property have a fireplace?
full.final$HasFireplace <- ifelse(full.final$FireplaceQu %in% c("Not Available"), "No", "Yes")
full.final$HasFireplace <- as.factor(full.final$HasFireplace)

# Does the property have a basement?
full.final$HasBsmt <- ifelse(full.final$BsmtQual %in% c("Not Available"), "No", "Yes")
full.final$HasBsmt <- as.factor(full.final$HasBsmt)

# Is the property new?
full.final$IsNew <- ifelse(full.final$YearBuilt == full.final$YrSold, "Yes", "No")
full.final$IsNew <- as.factor(full.final$IsNew)

# Does the property have a second floor?
full.final$Has2ndFlr <- ifelse(full.final$X2ndFlrSF != 0, "Yes", "No")
full.final$Has2ndFlr <- as.factor(full.final$Has2ndFlr)

# Does the property have a Garage?
full.final$HasGarage <- ifelse(full.final$GarageArea != 0, "Yes", "No")
full.final$HasGarage <- as.factor(full.final$HasGarage)

# Does the property have a WoodDeck?
full.final$HasWD <- ifelse(full.final$WoodDeckSF != 0, "Yes", "No")  # Has Wood Deck?
full.final$HasWD <- as.factor(full.final$HasWD)

# Does the property have a Open Porch?
full.final$HasOporch <- ifelse(full.final$OpenPorchSF != 0, "Yes", "No")  # Has Open Porch?
full.final$HasOporch <- as.factor(full.final$HasOporch)

# Does the property have a Close Porch?
full.final$HasCporch <- ifelse(full.final$EnclosedPorch != 0, "Yes", "No")  # Has Closed Porch?
full.final$HasCporch <- as.factor(full.final$HasCporch)

# Does the property have a Screen Porch?
full.final$HasSporch <- ifelse(full.final$ScreenPorch != 0, "Yes", "No")  # Has Screen Porch?
full.final$HasSporch <- as.factor(full.final$HasSporch)

# Does the property feels like its protecting the owners?
Visualize_suspect_var(train.complete, "Fence", "Fence of property") # Feels Protected?
full.final$Protected <- ifelse(full.final$Fence %in% c("GdPrv", "Not Available"), "Yes", "No")
full.final$Protected <- as.factor(full.final$Protected)

```

The Neighborhood feature can also be extracted and divided into four distinct categories: Bad, Fair, Good, Excellent. Neighborhood is binned to engineer the NeighborhoodQual feature:

```r
ggplot(train.complete, aes(x = reorder(Neighborhood, SalePrice, FUN = mean), y = SalePrice)) +
  geom_bar(stat = "summary", fun.y = "mean", fill = "green") +
  geom_bar(stat = "summary", fun.y = "median", fill = "blue", alpha = 0.7) +
  scale_y_continuous(labels = scales::comma) +
  geom_hline(yintercept= median(train.complete$SalePrice), linetype="dashed", color = "red", size = 1.5) +
  geom_hline(yintercept= mean(train.complete$SalePrice), linetype="dashed", color = "orange", size = 1.5) +
  geom_label(stat = "count", aes(label = ..count.., y = ..count..)) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 1)) +
  ylab("Sale Price in USD") +
  xlab("Neighborhood")
```



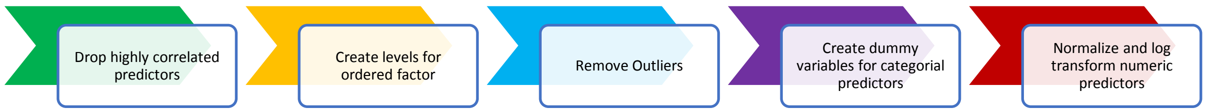

```r
full.final$NeighborhoodQual <- ifelse(full.final$Neighborhood %in% c("MeadowV"), "Bad",
                                      ifelse(full.final$Neighborhood %in% c("OldTown", "Edwards", "BrkSide", "Sawyer", "Blueste", "SWISU", "NAmes", "NPkVill"),"Fair",
                                             ifelse(full.final$Neighborhood %in% c("Mitchel", "SawyerW", "Gilbert", "NWAmes",
                                                                                   "Blmngtn", "CollgCr"), "Good", "Excellent")))
full.final$NeighborhoodQual <- as.ordered(full.final$NeighborhoodQual) # Categorize neighborhood by cluster visually
```

The following features are also extracted using some of the survey's recurring questions. Contrary to its real value, the property price is often determined by subjective human generalizations. Therefore, we may be able to more accurately predict price if we incorporate some aggregate features such as total bathrooms, total living space and house age.

```r
# Aggregating some features to better reflect human decision making mentality
full.final$TotBath <- full.final$FullBath + (full.final$HalfBath*0.5) +
                      full.final$BsmtFullBath + (full.final$BsmtHalfBath*0.5) # Total number of bathrooms
full.final$TotSF <- full.final$GrLivArea + full.final$TotalBsmtSF # Total living space
full.final$HouseAge <- full.final$YrSold - full.final$YearBuilt # Total house Age
```

Section 5: Data encoding for Machine Learning

This section displays the steps to encode the data for machine learning. Now that we have accomplished a reasonable tailoring of the data and derived the final dataset, we will have to convert the features into their machine-readable forms. These include:



- Dropping highly correlated variables: Correlated independent variables can cause multi-collinearity and overfitting in the model, thereby reducing the interpretability and generalizability power of the model (reducing the precision of a predictor's impact on the independent variable).

```r

# Drop variables with lack of information/ duplicate information

Visualize_suspect_var(train.complete, "Utilities", "Utilities of property") # Very few observation to matter

cor.high <- cor
cor.high[lower.tri(cor.high ,diag = TRUE)] <- NA
cor.high <- subset(na.omit(data.frame(expand.grid(dimnames(cor.high)), value = c(cor.high))), value > .7)
                                     # Subset to find pairs with high correlation
                                     # cor.high

var.drop <- c("YearRemodAdd", "Utilities", "GarageYrBlt", "GarageArea", "GarageCond", "TotalBsmtSF",
                    "TotalRmsAbvGrd", "BsmtFinSF1")
full.final <- full.final[,!(names(full.final) %in% var.drop)]  # Drop these variables

```

- Create levels for ordered factors: We create levels for the algorithm to use as dummy variables (with 0 or 1 values for each observation). This is to get around the fact that the regression can only understand numbers and not string categories. Ordered factors will also give the Machine Learning algorithm more information to work with than ordinary factors.

```r

# Convert the remaining into suitable variable type and add correct levels to ordered

ordered_var <- ordered_var[!ordered_var %in% c("Utilities", "GarageCond", "GarageYrBlt")]
full.final <- full.final %>%
  mutate_at(factor_var, funs(factor(.))) %>%
  mutate_at(ordered_var, funs(ordered(.)))

 str(full.final[,ordered_var])

full.final$LotShape <- ordered(full.final$LotShape, levels = c("IR3", "IR2", "IR1", "Reg"))
full.final$LandSlope <- ordered(full.final$LandSlope, levels = c("Sev", "Mod", "Gtl"))
full.final$ExterQual <- ordered(full.final$ExterQual, levels = c("Po", "Fa", "TA", "Gd", "Ex"))
full.final$ExterCond <- ordered(full.final$ExterCond, levels = c("Po", "Fa", "TA", "Gd", "Ex"))
full.final$BsmtQual <- ordered(full.final$BsmtQual, levels = c("Not Available", "Fa", "TA", "Gd", "Ex"))
full.final$BsmtCond <- ordered(full.final$BsmtCond, levels = c("Not Available","Po", "Fa", "TA", "Gd"))
full.final$BsmtExposure <- ordered(full.final$BsmtExposure, levels = c("Not Available","No", "Mn", "Av", "Gd"))
full.final$BsmtFinType1 <- ordered(full.final$BsmtFinType1, levels = c("Not Available","Unf", "LwQ", "Rec", "BLQ", "ALQ", "GLQ"))
full.final$BsmtFinType2 <- ordered(full.final$BsmtFinType2, levels = c("Not Available","Unf", "LwQ", "Rec", "BLQ", "ALQ", "GLQ"))
full.final$HeatingQC <- ordered(full.final$HeatingQC, levels = c("Po", "Fa", "TA", "Gd", "Ex"))
full.final$Electrical <- ordered(full.final$Electrical, levels = c("Mix", "FuseP", "FuseF", "FuseA", "SBrkr"))
full.final$KitchenQual <- ordered(full.final$KitchenQual, levels = c("Fa", "TA", "Gd", "Ex"))
full.final$Functional <- ordered(full.final$Functional, levels = c("Sal", "Sev", "Maj1", "Maj2", "Mod", "Min2", "Min1", "Typ"))
full.final$FireplaceQu <- ordered(full.final$FireplaceQu, levels = c("Not Available", "Po","Fa", "TA", "Gd", "Ex"))
full.final$GarageFinish <- ordered(full.final$GarageFinish, levels = c("Not Available", "Unf","RFn", "Fin"))
full.final$GarageQual <- ordered(full.final$GarageQual, levels = c("Not Available", "Po","Fa", "TA", "Gd", "Ex"))
full.final$PoolQC <- ordered(full.final$PoolQC, levels = c("Not Available","Fa", "Gd", "Ex"))
full.final$TotRmsAbvGrd <- ordered(full.final$TotRmsAbvGrd, levels = c("2","3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15"))
full.final$Fireplaces <- ordered(full.final$Fireplaces, levels = c("0","1", "2", "3", "4"))

```

- Remove outliers: The outliers identified previously may cause lower the model's accuracy and its ability to generalize outside the test set.

```r

# Removing outliers
full.final$SalePrice <- full$SalePrice
full.final <- full.final[-c(524,1299),]

```

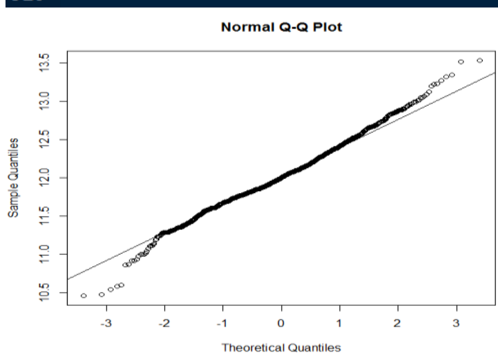
- Create dummy variables: The dummy variables will greatly increase our number of predictors. Therefore, we will also remove some dummy variable with less than 10 positive observations.

```
## Prepocess numeric variables

nums <- unlist(lapply(full.final, is.numeric))
full.numeric <- full.final[,nums]
full.numeric <- full.numeric[,!(names(full.numeric) %in% "SalePrice")]
full.factor <- full.final[,!nums]
cat('There are', length(full.numeric), 'numeric variables, and', length(full.factor), 'factor variables')

# Create dummy variables

full.dummy <- as.data.frame(model.matrix(~.-1, full.factor))
dim(full.dummy)

# Taking out a few categorical/dummy variables with very little information
var.lowinfo <- which(colSums(full.dummy[1:nrow(full.final[!is.na(full.final$SalePrice),]),])<10)
colnames(full.dummy[var.lowinfo])

full.dummy <- full.dummy[,-var.lowinfo] #removing variables
dim(full.dummy)
```

- Normalization & Log transformation: We observe skewness in some independent variable and the response variable. Therefore, we will log transform these variables. We will also normalize the numeric predictors to center and scale them for the Machine Learning algorithm.

```
# Log transformation of numeric variables
describe(full.numeric)

for(i in 1:ncol(full.numeric)){
  if (abs(skew(full.numeric[,i]))>0.8){
    full.numeric[,i] <- log(full.numeric[,i] +1)
  }
}

# Normalizing the numeric variables
num.process <- preProcess(full.numeric, method=c("center", "scale"))
print(num.process)
full.numeric.scaled <- predict(num.process, full.numeric)
```

```
# Dealing with skewness in response variable
skew(full.final$SalePrice)
full.encode$SalePrice <-  log(full.final$SalePrice)
qqnorm(full.encode$SalePrice)
qqline(full.encode$SalePrice)     # Log transform and visualize SalePrice
```



Finally, we combine the two subsets to an encoded dataset. We will use this to derive the encoded training set and the encoded test set.

```
# Combining the predictor variables
full.encode <- cbind(full.numeric.scaled, full.dummy)
```

```
# Divide encoded data into train and test set
train.encode <-  full.encode[1:1458,]
test.encode <- full.encode[1459:2917,]
```

Section 6: Machine Learning Model Selection and Training

This section attempts to train a suitable machine learning algorithm to predict the sale price of future properties. The problem here is a prediction of continuous value for the response variable. Thus, we can begin with conducting a simple regression. However, with only a few predictors showing strong predictive power out of over 200 features, we will want to introduce a regularization parameter to penalize the number and/or effect of features used by our regression. Thus, we will use an elastic net via cross-validation to best tune our regularization parameters (alpha as degree of mix between LASSO and Ridge regression and lambda as the shrinkage parameter of the penalty).

```
set.seed(123)
mod.control <-trainControl(method="cv", number=5)
grid <- expand.grid(alpha = 1, lambda = seq(0.001,0.1,by = 0.001))

regressor <- train(x=train.encode[,!(names(train.encode) %in% "SalePrice")], y=train.encode$SalePrice,
                   method='glmnet', trControl= mod.control, tuneGrid= grid)
regressor$bestTune
```

```
  alpha lambda
2     1  0.002
```

Our hyperparameter tuning shows best results for LASSO regression with a shrinkage parameter of 0.002. This will also result in a RMSE score of ~ **11.4%:**

```
> min(regressor$results$RMSE)
[1] 0.1138914
```

Let's also check if Lasso Regression did its job by cutting down on many of the variables.

```
# Check the number of variables being selected for the best model.
contrib <- varImp(regressor,scale=F)$importance
var.selected <- length(which(contrib$Overall!=0))
var.nselected <- length(which(contrib$Overall==0))

cat('L-Regression uses', var.selected, 'variables in its model, and did not select', var.nselected, 'variables.')
```

```
L-Regression uses 110 variables in its model, and did not select 128 variables.
```

It has removed 128 variables and used 110. The regularized regression seems to be using only independent variables with statistical significant. It therefore looks to be a viable model.

Section 7: Conclusion and Risk assessment

Throughout this report, we have prepared, cleaned and explored the dataset collected. Then, we engineered new features with domain understanding to help our predictive model perform better and chose a suitable machine learning model for the prediction task. Our model is likely to score 11.4% precision with any test set and has been rigorously devised to generalize well in reality.

However, improvements can certainly be investigated. The following are risks and improvements that users of this report can make to improve the model's performance:

- First, the user should care to use a training dataset that accurately reflects the geographic, social and demographic trends relevant to real estate transactions in the setting of the consulting property.
- Second, the user should explore more algorithm options. Although simple linear regression models are usually best when predicting a generally linear problem such as this, results could be enhanced with incorporation of non-linear terms. If accuracy over speed is the user's primary concern, he/she should explore using neural networks, gradient boosting trees or other combination and ensemble of models to derive predictions.
- Lastly, the user should only consider the results of the algorithm as a referred generalized price point. Some rare properties and real-estates can have characteristics that make them extra-ordinary and will most likely not be accounted for in the model.