**Predicting Housing Prices in Ames, Iowa**

*Blake Cizek*

*Foundations of Data Science Capstone*

*May 2018*

**Project Overview**

The goal of this Capstone Project is to predict housing prices in Ames, Iowa - based on Kaggle’s Advanced Regression Techniques Competition. While this competition is no longer active, its scope and data present itself to be a choice starting ground for data cleaning, exploratory data analysis, statistics and machine learning exercises.

The recipient of this project would ideally be Zillow. As a technology leader in the real estate industry, Zillow continually looks for data, analyses and algorithms to improve its platform, which serves both the buyer and seller of residential properties.

**1: Executive Summary**

**1.1: Project Results**

Following the outlined method on the Kaggle’s data sets, Lasso Regression and XGBoost models were used to predict the housing prices in the provided test set:

* The Lasso regressions performed best of the two models with a cross validation RMSE of 0.1142. Lasso did not select a substantial number of the available variables in its model. Due to the multicollinearity among most variables, this was expected.
* The XGBoost model performed a close 2nd with a cross validation RMSE of 0.1189.
* Although Lasso and XGBoost are fundamentally different models, averaging of the predictions performed to improve the test set results.

**1.2: Project Recommendations:**

1. Improve general quality of data entry validation and process. For the most part, values that were missing in this data set were related to the fact that certain homes did not have features such as pools, garages, alleys, basements, etc. The existence of binary markers for whether homes have these features would improve the process by avoiding imputation.
2. As outlined in the Feature Engineering section of this project, add variables such as ‘TotalBath’, ‘Age’, ‘Remod’, ‘New’, ‘TotalSqFt’, ‘TotalPorchSqFt’, and ‘NeighborhoodWealth’. These variables added significant value to the model both in feature importance as well as general data cleanliness to cut down on the sometimes 8+ variables that made up one feature of a house.
3. Use Lasso Regression models as the starting block for predicting housing prices as it was able to perform better than XGBoost, using significantly les variables. Averaging the two allows for a good balance, but fine tuning Lasso Regression or models related to it is the preferred method due to better accuracy and model runtime efficiency.

**2: Data & Method**

**2.1: Data**

The data provided by Kaggle’s Advanced Regression Techniques Competition will be the primary source used in this project. This data consists of a training set and a test set. Present in each set are 79 predictor variables (house attributes) and one target variable (price). Each predictor variable could be categorized as:

* lot/land variables
* location variables
* age variables
* appearance variables
* external features (pools, porches, etc.) variables
* room/bathroom variables
* kitchen variables
* basement variables
* roof variables
* garage variables
* utilities variables

**2.2: Method**

The workflow for this project is as follows from top to bottom:

* Brief EDA of Kaggle’s Data Sets
* Data Wrangling
  + Missing Data
  + Imputing Data
  + Label encoding/factorizing variables
* Detailed EDA of “Tidy Data”
  + Visualization & Stats work
* Feature Engineering
* Model Prep
  + Dropping highly correlated variables
  + Removing outliers
  + Data normalization
* Data Modeling
  + Lasso Regression
  + XGBoost Model
  + Model Average & Results

**3: Loading Packages & Data**

**3.1: Packages**

library(knitr)

library(ggplot2)

library(plyr)

library(dplyr)

library(corrplot)

library(caret)

library(gridExtra)

library(scales)

library(Rmisc)

library(ggrepel)

library(randomForest)

library(psych)

library(xgboost)

library(ckmeans.1d.dp)

**3.2: Data Sets**

* Original Sets
  + Train – 81 Variables, 1460 records
  + Test – 80 Variables, 1459 records
* Primary Working Set (Combined Test & Train)
  + df.combined – 80 Variables, 1459

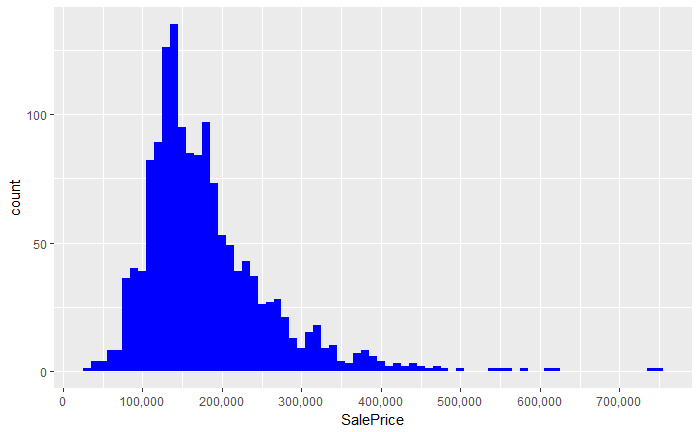
**4: EDA of “Untidy Data”**

**4.1: SalePrice - Response Variable**

Sale Prices are right skewed, as only a small population of people can afford expensive houses.

summary(df.combined$SalePrice)

## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's   
## 34900 129975 163000 180921 214000 755000 1459

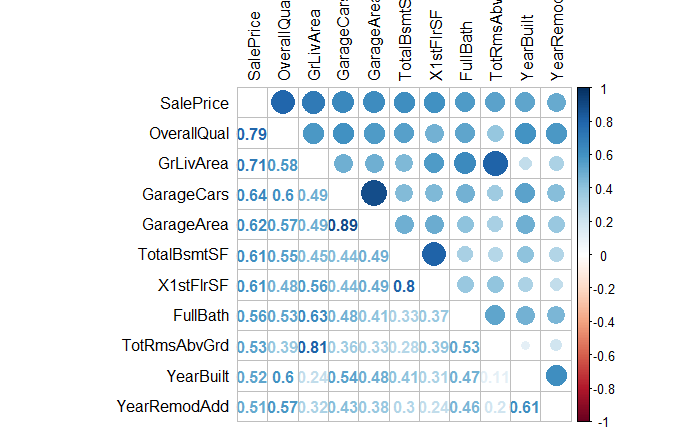


**4.2: Importance of Numeric Predictors (37 Variables)**

Quick EDA of numeric variables, as character variables need a lot of work.

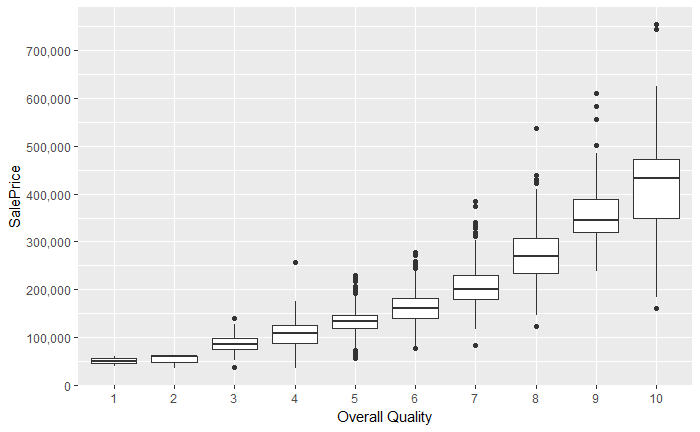
**4.2.1: Correlation with Sale Price**

There are 10 numeric variables with at least a 0.5 correlation with Sale Price. All correlations are positive. Through this plot, it becomes clear that multicollinearity is an issue. Brief EDA of top 3 correlations with Sales Price follows.

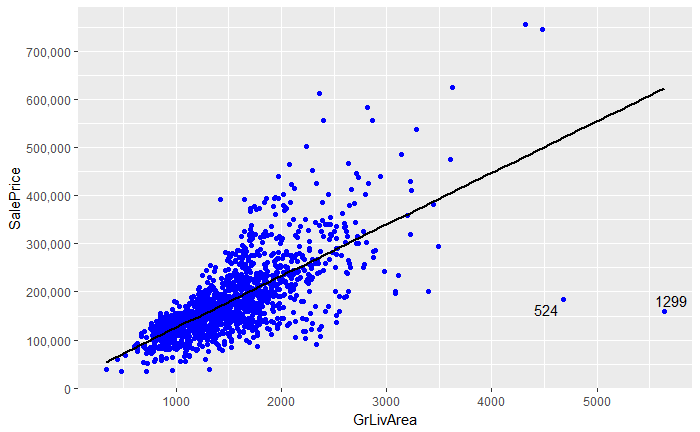


**4.2.2: OverallQual correlation with SalePrice**

Overall Quality had the highest correlation, 0.79, with SalePrice among numeric variables. SalePrice rises as OverallQual improved.



**4.2.3: GrLivArea correlation with SalePrice**

Overall Quality had the 2nd highest correlation, 0.71, with SalePrice among numeric variables. Generally, SalePrice is higher as GrLivArea increases. Outlier exceptions for house #524 and #1299

## SalePrice GrLivArea OverallQual  
## 524 184750 4676 10  
## 1299 160000 5642 10

**5: Data Wrangling (Missing Data, Label Encoding & Factorizing Variables)**

**5.1: Identifying Missing Data**

Of the original set, df.combined contains 34 variables are missing values. The 35th in this variable is SalePrice since this is a combined data frame with both train & test sets.

NAcol <- which(colSums(is.na(df.combined)) > 0)  
sort(colSums(sapply(df.combined[NAcol], is.na)), decreasing = TRUE)

## PoolQC MiscFeature Alley Fence SalePrice   
## 2909 2814 2721 2348 1459   
## FireplaceQu LotFrontage GarageYrBlt GarageFinish GarageQual   
## 1420 486 159 159 159   
## GarageCond GarageType BsmtCond BsmtExposure BsmtQual   
## 159 157 82 82 81   
## BsmtFinType2 BsmtFinType1 MasVnrType MasVnrArea MSZoning   
## 80 79 24 23 4   
## Utilities BsmtFullBath BsmtHalfBath Functional Exterior1st   
## 2 2 2 2 1   
## Exterior2nd BsmtFinSF1 BsmtFinSF2 BsmtUnfSF TotalBsmtSF   
## 1 1 1 1 1   
## Electrical KitchenQual GarageCars GarageArea SaleType   
## 1 1 1 1 1

**5.2: Imputing Missing Data**

34 variables contain missing values. This extensive data cleaning & imputation is outlined in the [code](https://github.com/bcizek91/capstone_project_springboard/blob/master/Data%20Wrangling/Data%20Wrangle.Rmd) and [results](https://github.com/bcizek91/capstone_project_springboard/blob/master/Data%20Wrangling/Data_Wrangle.docx) contained in this project’s [Data Wrangling repository](https://github.com/bcizek91/capstone_project_springboard/tree/master/Data%20Wrangling). Below is the general section by which the variables were handled

* 5.2.1 Pool variables
* 5.2.2 Miscellaneous Feature
* 5.2.3 Alley
* 5.2.4 Fence
* 5.2.5 Fireplace variables
* 5.2.6 Lot variables
* 5.2.7 Garage variables
* 5.2.8 Basement Variables
* 5.2.9 Masonry variables
* 5.2.10 MS Zoning
* 5.2.11 Kitchen variables
* 5.2.12 Utilities
* 5.2.13 Home functionality
* 5.2.14 Exterior variables
* 5.2.15 Electrical system
* 5.2.16 Sale Type and Condition

**5.3: Label encoding/factoring character variables**

After data imputing, there are 15 variables that remain with character values. This extensive label encoding & factorization is outlined in the [code](https://github.com/bcizek91/capstone_project_springboard/blob/master/Data%20Wrangling/Data%20Wrangle.Rmd) and [results](https://github.com/bcizek91/capstone_project_springboard/blob/master/Data%20Wrangling/Data_Wrangle.docx) contained in this project’s [Data Wrangling repository](https://github.com/bcizek91/capstone_project_springboard/tree/master/Data%20Wrangling). Below is the general section by which the variables were handled

character.VarNames <- names(df.combined[, sapply(df.combined, is.character)])  
character.VarNames

## [1] "LandContour" "LandSlope" "Neighborhood" "Condition1"   
## [5] "Condition2" "BldgType" "HouseStyle" "RoofStyle"   
## [9] "RoofMatl" "Foundation" "Heating" "HeatingQC"   
## [13] "CentralAir"

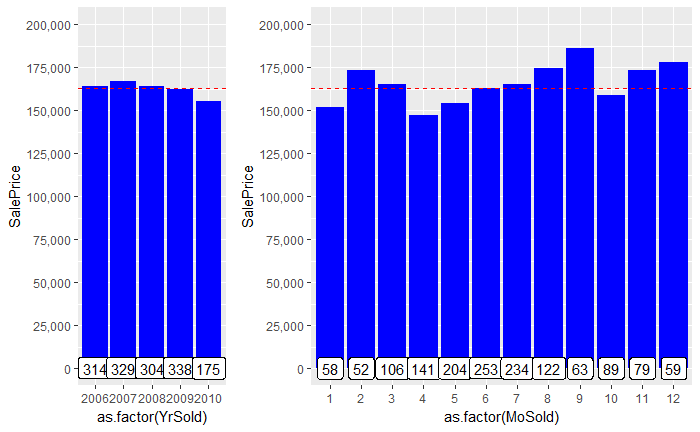
* 5.3.1 Foundation
* 5.3.2 Heating and Air
* 5.3.3 Roof
* 5.3.4 Land
* 5.3.5 Dwelling
* 5.3.6 Neighborhood and Conditions
* 5.3.7 Pavement of Street & Driveway

**5.4:** **Factorizing certain Numeric Variables**

All variables are completed without NAs, but there are 3 numeric variables that should be categorical.

**5.4.1:** **Year & Month Sold**

YrSold should be ordinal in order understand the age of a home from YrBuilt. MoSold is an integer and needs to be factorized because no one month holds value over another. Quick EDA of YrSold and MoSold against SalePrice is below.



**5.4.2: MSSubClass**

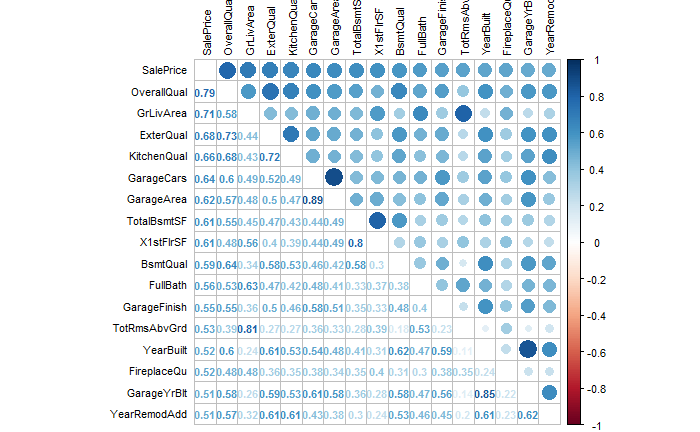
This variable is coded as an integer but needs to be categorical as these are building codes not values.

**6: Detailed EDA of Important Variables**

After a detailed data wrangling process, all predictor variables are now “tidy”. Of this set, 56 variables are numeric, 23 are categorical, and 1 (Utilities) was deleted due to irrelevance.

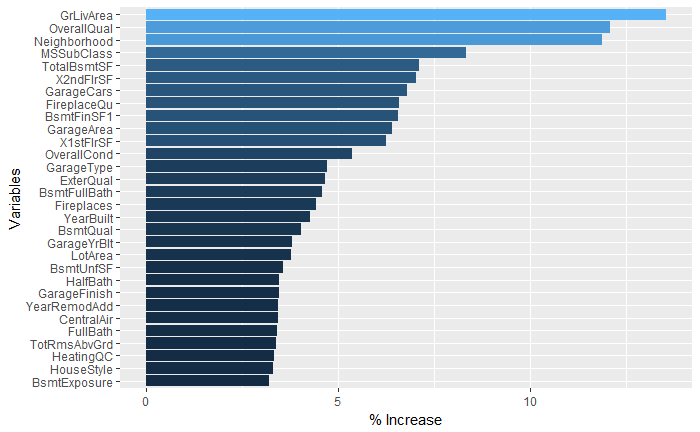
**6.1: Level-set Correlations with SalePrice**

After the data wrangling, the number of variables with a correlation of at least 0.5 with SalePrice increased from 10 (pre-cleaning) to 16 (post cleaning).



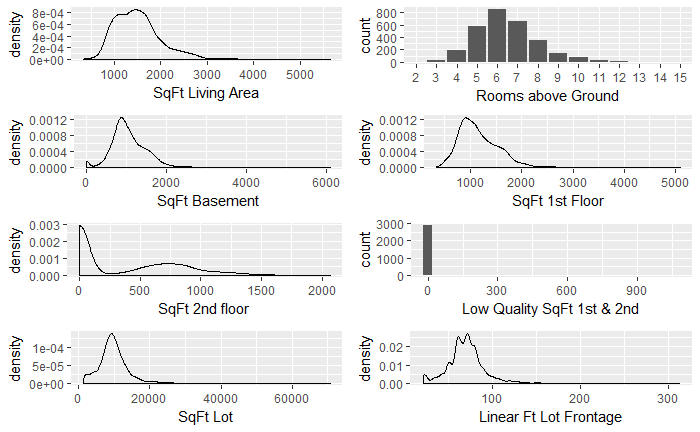
**6.2: Importance of Variables using RandomForest**

In addition to the overview which correlation give the importance of numeric variables and their multicollinearity. This quick RandomForest includes the importance of both numeric and categorical.



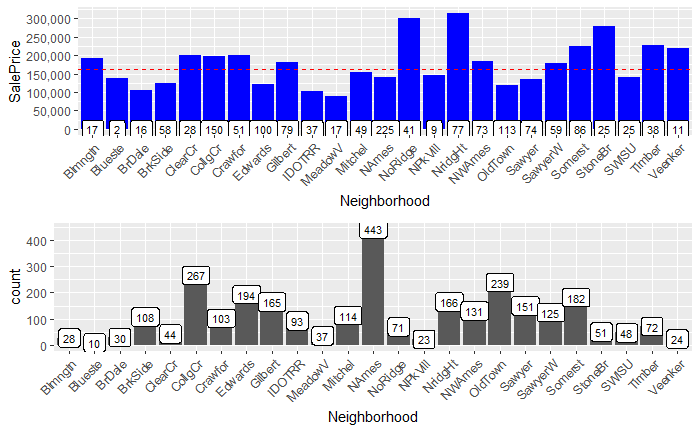
**6.2.1: EDA of GrLivArea and other “SqFt” Variables**

Visualizing the distribution of GrLivArea and other importance “SqFt” variables.



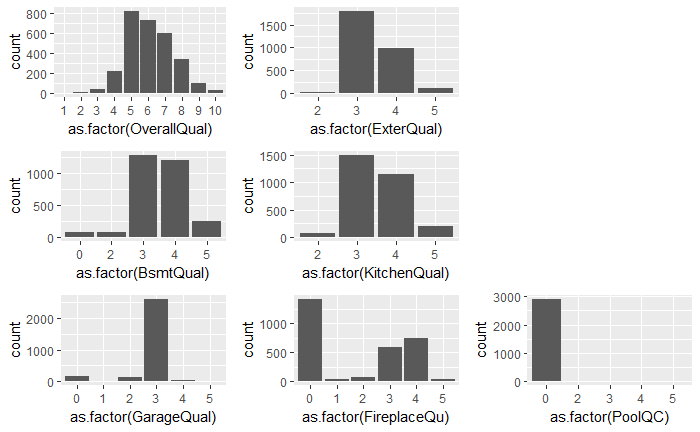
**6.2.2: EDA of Neighborhood with SalePrice and House Count**

According up the above quick RandomForest, Neighborhood is the most important categorical variable.



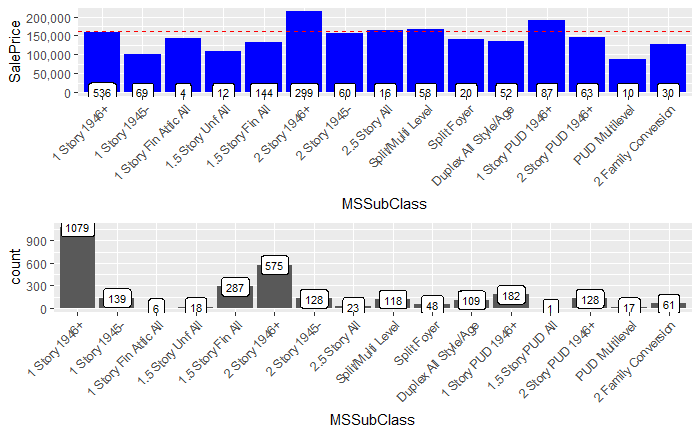
**6.2.3: EDA of OverallQual and other Quality Variables**

Frequency distribution of quality variables.



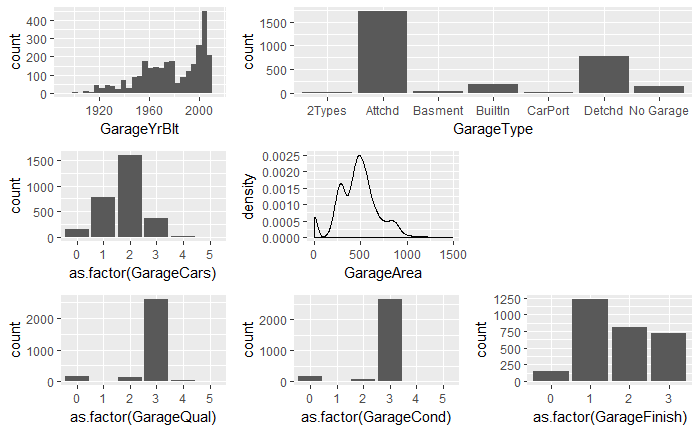
**6.2.4: EDA of MSSubClass with SalePrice and House Count**

Frequency distribution of 2nd most important categorical variable.



**6.2.5:** **EDA of Garage Variables**

7 variables have a high correlation with SalePrice, but multicollinearity exists among them.



**6.2.6: EDA of Basement Variables**

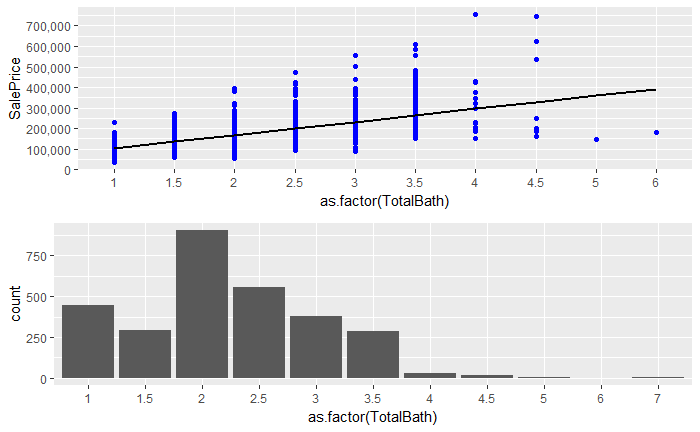
11 variables have a high correlation with SalePrice, but multicollinearity exists among them. TotalBsmtSF is further broken down into finished areas (2 if more than one type of finish), and unfinished area. Correlation of total of those 3 variables, and TotalBsmtSF is 1.



**7: Feature Engineering**

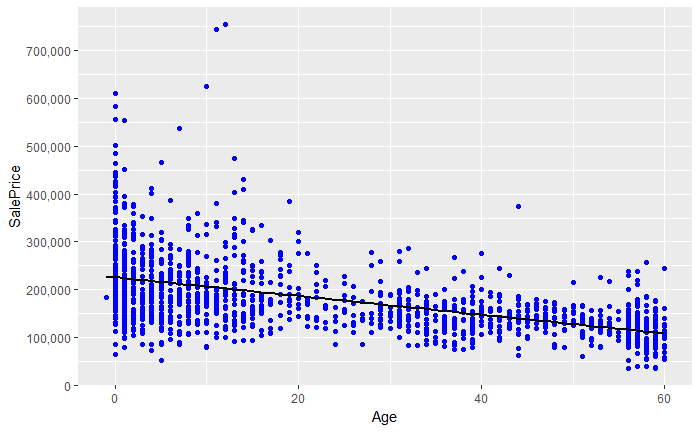
**7.1: Adding ‘TotalBath’ with EDA**

Combining 4 bathroom variables into 1. Quick EDA of TotalBath with SalePrice and Count



**7.2: Adding ‘Age’, ‘Remod’, and ‘New’ with EDA**

There are 3 variables that are relevant in regards to a house, the YearBlt, YearRemod and YearSold. If a house was remodeling, YearRemod is that year or defaults to YearBlt is no remodeling took place. It is useful to differentiate between the Age of the house, if it was modeled, and if it is a new build.

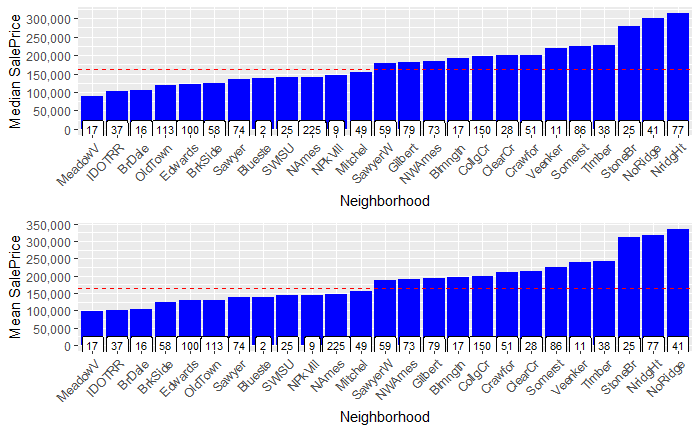


|  |  |
| --- | --- |
|  |  |

As houses age, generally, they are worth less. Surprisingly, houses that are remodeled are worth less, but houses that are newer are worth much more.

**7.3: Binning Neighborhoods**

Both median and mean SalePrice agree on the 3 “rich neighborhoods,” but are harder to differentiate in “poor neighborhoods”. 3 categories determining neighborhood “wealth” will be created.



#Categorizing Neighborhood Wealth (Poor - 0, Middle - 1, Wealthy - 2)  
df.combined$NeighborhoodWealth[df.combined$Neighborhood %in% c('StoneBr', 'NridgHt', 'NoRidge')] <- 2  
df.combined$NeighborhoodWealth[!df.combined$Neighborhood %in% c('MeadowV', 'IDOTRR', 'BrDale', 'StoneBr', 'NridgHt', 'NoRidge')] <- 1  
df.combined$NeighborhoodWealth[df.combined$Neighborhood %in% c('MeadowV', 'IDOTRR', 'BrDale')] <- 0  
table(df.combined$NeighborhoodWealth)

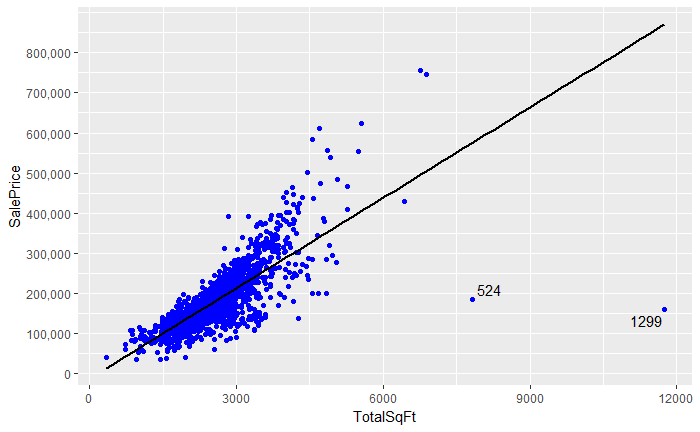
##   
## 0 1 2   
## 160 2471 288

sum(table(df.combined$NeighborhoodWealth))

## [1] 2919

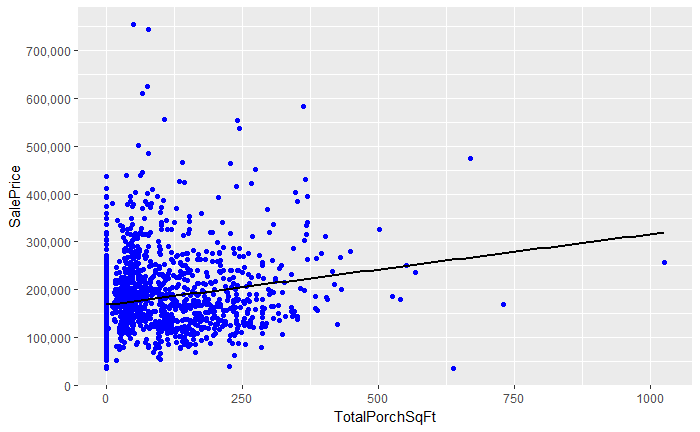
**7.4: Adding ‘TotalSqFt’ with EDA**

There are 2 variables detailing total SqFt, adding both together allows for a more complete picture of the house. Removing the two outliers, #524 and #1299, allows for SalePrice and TotalSqFt correlation to up by 5%.



**7.5: Adding ‘TotalPorchSqFt’ with EDA**

There are 5 different variables that detail an aspect of a porch. Despite having a low correlation with SalePrice, it doesn’t hurt to combine all 5 into one variable.



**8: Model Prep**

Removing highly correlated variables, outliers and processing variables for modeling.

**8.1: Removing Highly Correlated Variables**

Of variables pairs that are highly correlated, dropping variable with least correlation with SalePrice.

rmVars <- c('YearRemodAdd', 'GarageYrBlt', 'GarageArea', 'GarageCond', 'TotalBsmtSF', 'TotRmsAbvGrd')

**8.2: Removing Outliers**

As seen above in various EDAs, homes #524 an #1299 are outliners. Once removed, they increase correlation.

**8.3: Prepping Predictor Variables**

Splitting categoric and numeric predictor variables. Of this, there are 30 numeric and 49 categoric variables.

**8.4:** **Absolute Skewness and Normalization of Numeric Variables**

Log of all numeric predictor variables with an absolute skew grater than 0.8 to avoid division by 0.

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

Normalizing data

Predictor.Vars <- preProcess(df.numeric, method = c("center", "scale"))  
print(Predictor.Vars)

## Created from 2917 samples and 30 variables  
##   
## Pre-processing:  
## - centered (30)  
## - ignored (0)  
## - scaled (30)

df.normal <- predict(Predictor.Vars, df.numeric)  
dim(df.normal)

## [1] 2917 30

**8.3.2: Hot Encoding Categorical Variables**

Converting all factor variables to binary (Yes/No, 1s and 0s) to prep for ML models.

df.dummy <- as.data.frame(model.matrix(~.-1, df.categoric))  
dim(df.dummy)

## [1] 2917 201

**8.3.3: Removing records with few/none values in train or test sets**

In total 49 hot-encoded predictor variables were removed due to little or no variance.

##Absent Values in Train Set  
Values.Absent.Train <- which(colSums(df.dummy[1:1458, ]) == 0)  
colnames(df.dummy[Values.Absent.Train])

## [1] "MSSubClass1.5 Story PUD All"

##Removing Predictor Values  
df.dummy <- df.dummy[, -Values.Absent.Train]  
  
##Few Values (<10) in Train Set  
Values.Few.Train <- which(colSums(df.dummy[1:1458, ]) <10)  
colnames(df.dummy[Values.Few.Train])

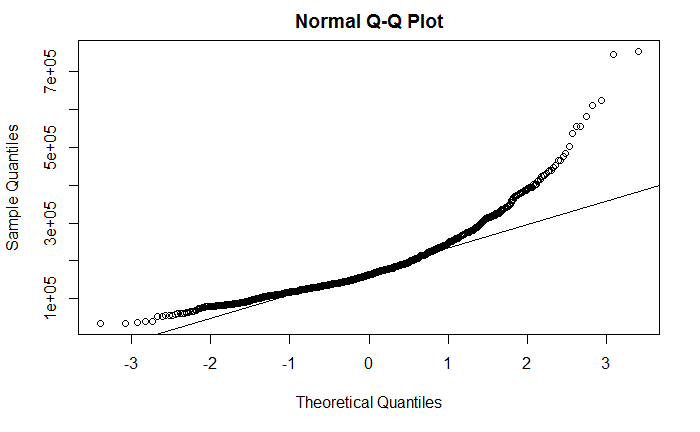
## [1] "MSSubClass1 Story Fin Attic All" "LotConfigFR3"   
## [3] "NeighborhoodBlueste" "NeighborhoodNPkVill"   
## [5] "Condition1PosA" "Condition1RRNe"   
## [7] "Condition1RRNn" "Condition2Feedr"   
## [9] "Condition2PosA" "Condition2PosN"   
## [11] "RoofStyleMansard" "RoofStyleShed"   
## [13] "RoofMatlWdShake" "RoofMatlWdShngl"   
## [15] "Exterior1stAsphShn" "Exterior1stBrkComm"   
## [17] "Exterior1stCBlock" "Exterior2ndAsphShn"   
## [19] "Exterior2ndBrk Cmn" "Exterior2ndCBlock"   
## [21] "Exterior2ndStone" "FoundationStone"   
## [23] "FoundationWood" "HeatingGrav"   
## [25] "HeatingWall" "ElectricalFuseP"   
## [27] "GarageTypeCarPort" "MiscFeatureOthr"   
## [29] "SaleTypeCon" "SaleTypeConLD"   
## [31] "SaleTypeConLI" "SaleTypeConLw"   
## [33] "SaleTypeCWD" "SaleTypeOth"   
## [35] "SaleConditionAdjLand"

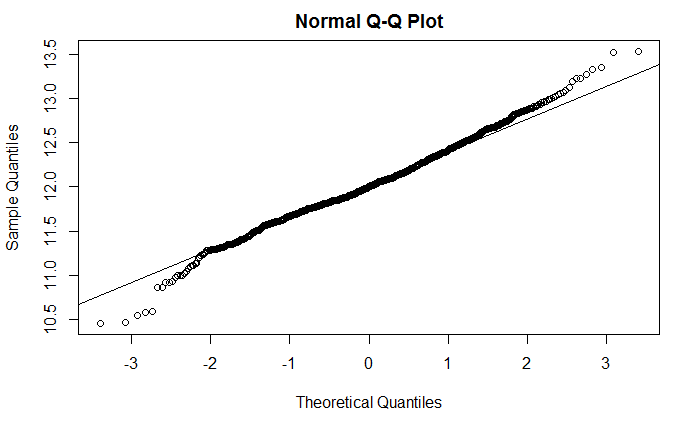
##Removing Predictor Values  
df.dummy <- df.dummy[, -Values.Few.Train]  
dim(df.dummy)

## [1] 2917 152

**8.4: Skewness of Response Variable (SalePrice)**

Normalization of SalePrice necessary as 1st EDA details that its log (2nd EDA) is needed to normally distribute.





**9: Modeling**

Lass Regression and XGBoost are the two models used in predicting SalePrice. The average of the two is complied to smooth out the results.

**9.1: Lasso Regression Model**

Caret Cross Validation is used to find the best value of lamda, the only hyperparameter tuning in Lasso. This model dealt with multicollinearity of variables very well by not using roughly 46% of the predictor variables. Lasso Regression performed with a cross validation RMSE of 0.1142

set.seed(12345678)  
Control.Train <-trainControl(method = "cv", number = 5)  
Grid.Lasso <- expand.grid(alpha = 1, lambda = seq(0.001, 0.1, by = 0.0005))  
  
Model.Lasso <- train(x = train.set, y = df.combined$SalePrice[!is.na(df.combined$SalePrice)], method = 'glmnet', trControl= Control.Train, tuneGrid = Grid.Lasso)  
Model.Lasso$bestTune

## alpha lambda  
## 5 1 0.003

min(Model.Lasso$results$RMSE)

## [1] 0.1142001

Vars.Important.Lasso <- varImp(Model.Lasso, scale = F)  
Importance.Lasso <- Vars.Important.Lasso$importance  
  
Vars.Selected.Lasso <- length(which(Importance.Lasso$Overall!= 0))  
Vars.NotSelected.Lasso <- length(which(Importance.Lasso$Overall == 0))  
  
cat('Lasso Model used', Vars.Selected.Lasso, 'variables & did not use', Vars.NotSelected.Lasso)

## Lasso Model used 85 variables & did not use 97

Prediction.Lasso <- predict(Model.Lasso, test.set)  
Prediction.Values.Lasso <- exp(Prediction.Lasso)  
  
View(Prediction.Values.Lasso)  
summary(Prediction.Values.Lasso)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 47844 126908 158759 178107 212839 685017

**9.2: XGBoost Model**

This model provides a cross validation function to determine the optimal number of rounds and supplemented with a grid search for hyperparameters. XGB performed with a cross validation RMSE of 0.1189.

##XGB Parameters  
Parameters.Model.XGB <- list(  
 objective = "reg:linear",  
 booster = "gbtree",  
 eta = 0.05,  
 gamma = 0,  
 max\_depth = 3,  
 min\_child\_weight = 3,  
 subsample = 1,  
 colsample\_bytree = 1  
)  
  
##XGB Cross Validation  
Cross.Validation.XGB <- xgb.cv(params = Parameters.Model.XGB, data = Train.Matrix.XGB, nrounds = 500, nfold = 5, showsd = T, stratified = T, print\_every\_n = 40, early\_stopping\_rounds = 10, maximize = F)

## [1] train-rmse:10.955587+0.004653 test-rmse:10.955522+0.019857   
## Multiple eval metrics are present. Will use test\_rmse for early stopping.  
## Will train until test\_rmse hasn't improved in 10 rounds.  
##   
## [41] train-rmse:1.428223+0.000292 test-rmse:1.428729+0.010585   
## [81] train-rmse:0.219372+0.000623 test-rmse:0.231261+0.004968   
## [121] train-rmse:0.101754+0.001338 test-rmse:0.130095+0.009549   
## [161] train-rmse:0.089498+0.001712 test-rmse:0.123474+0.010629   
## [201] train-rmse:0.083547+0.001728 test-rmse:0.121148+0.010802   
## [241] train-rmse:0.079052+0.001580 test-rmse:0.119955+0.011079   
## [281] train-rmse:0.075318+0.001398 test-rmse:0.119487+0.010939   
## [321] train-rmse:0.072064+0.001312 test-rmse:0.119232+0.011073   
## [361] train-rmse:0.069035+0.001125 test-rmse:0.118974+0.011057   
## Stopping. Best iteration:  
## [367] train-rmse:0.068597+0.001119 test-rmse:0.118900+0.011069

##Train Model using Best Round from Cross Validation  
Model.XGB <- xgb.train(data = Train.Matrix.XGB, params = Parameters.Model.XGB, nrounds = 367)  
  
Prediction.XGB <- predict(Model.XGB, Test.Matrix.XGB)  
Prediction.Values.XGB <- exp(Prediction.XGB)  
  
head(Prediction.Values.XGB)

## [1] 118315.0 161907.8 186297.7 187056.5 193928.6 166106.4

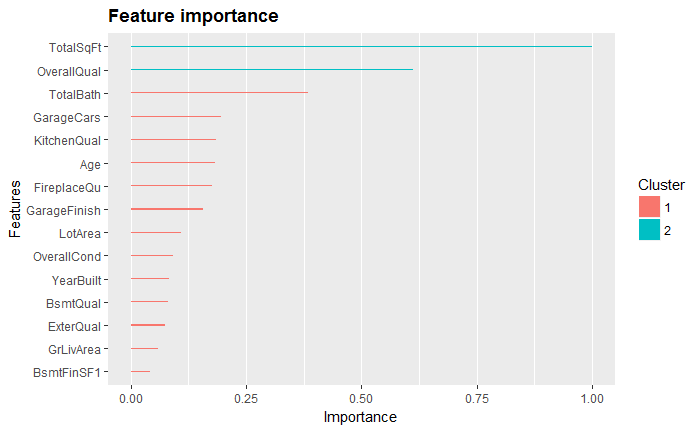
View(Prediction.Values.XGB)  
summary(Prediction.Values.XGB)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 42644 127825 156311 179084 209042 675123

**9.4: Feature Importance**

Using a kmeans package for ggplot clustering, Ckmeans.1d.dp, Feature Importance was determined according to XGBoost. This EDA is similar to the original quick RandomForest and Collinearity Plots EDAs. Clustering is essentially identifying most important variables and those that are supplemental for predicting SalePrice.

library(Ckmeans.1d.dp)  
  
Importance.XGB <- xgb.importance(feature\_names = colnames(train.set), Model.XGB)  
xgb.ggplot.importance(importance\_matrix = Importance.XGB[1:15], rel\_to\_first = TRUE)



**9.5: Averaging Lasso & XGBoost Model Predictions**

Although Lasso and XGBoost are fundamentally different models, averaging of the predictions performed to improve the test set results.

##Averaging Models  
Models.Average <- data.frame(Id = test.IDs, SalePrice = (Prediction.Values.XGB + Prediction.Values.Lasso)/2)  
  
head(Models.Average)

## Id SalePrice  
## 1461 1461 116905.4  
## 1462 1462 161620.0  
## 1463 1463 182991.7  
## 1464 1464 192431.8  
## 1465 1465 200390.8  
## 1466 1466 167897.1

View(Models.Average)  
summary(Models.Average$SalePrice)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 46871 127287 156887 178595 209627 680070