Housing Prices Analysis

Mandey Brown, Megan Dunnahoo, Emma Hamilton

```
library(knitr)
library(dplyr)
library(tidyr)
library(tidyverse)
library(ggplot2)
library(naniar)
library(GGally)
library(tree)
library(glmnet)
library(rpart)
library(gbm)
library(randomForest)
library(pls)
library(e1071)
library(formatR)
```

```
#reproducibility
set.seed(445)

knitr::opts_chunk$set(tidy.opts=list(width.cutoff=80), tidy=TRUE, fig.width = 6, fig.height=4)
```

Motivation

Housing prices play a central role in the U.S. economy. According to a *Congressional Research Service* article, *Introduction to U.S. Economy: Housing Market*, "at the individual level, roughly 65% of occupied housing units are owner occupied, homes are a substantial source of household wealth in the United States… housing accounts for a significant portion of all economic activity, and changes in the housing market can have broader effects on the economy." Buying a house is considered the most utilized and profitable investment for most of the population. The housing market is also incorporated into gross domestic product (GDP), which is considered the primary measure of economic activity for a country. Also, according to the article, *Introduction to U.S. Economy: Housing Market*, "as of 2020, spending on housing services was about $2.8 trillion, accounting for 13.3% of GDP. Taken together, spending within the housing market accounted for 17.5% of GDP in 2020."

In addition to the majority of the population benefiting from predictions of housing prices, many professions and industries would benefit as well. Home appraisers, mortgage lenders, insurers, and tax assessors would be able to more accurately asses the value of a home. Housing price predictions would also prove invaluable for home builders. For this project, the co-owner of Deluxe Homes LLC was interviewed in order to gain more industry insight. The co-owner, Stu Sprecher emphasized the need for flexible pricing predictions that would enable home builders to maintain a profit margin while ordering materials and hiring contractors for each home built. The ability to customize house price predictions to a specific home could prove invaluable for him as an industry professional.

Methodology

Built off the Kaggle competition, *House Prices - Advanced Regression Techniques,* this project utilizes housing prices compiled by Dean De Cock in 2011, which describes the sale of individual residential property in Ames, Iowa from 2006 to 2010. The dataset includes 79 explanatory variables (23 nominal, 23 ordinal, 14 discrete, and 20 continuous) all involved in evaluating home values. In order to build predictive models for housing price, an exploratory analysis was conducted followed by preprocessing. The project focuses on advanced regression models including a Decision Tree, Random Forest, Bagging, LASSO, and out of curiosity, Boosting.

A Decision Tree model is first explored as the output is easily interpreted and its graphical representation can be straightforwardly related to the predicting housing price. Though the downfalls of Decision Trees are known, such as overfitting. Instead of taking the approach of pre-pruning the Decision Tree using χ^2 test, which is "an algorithm used to find out the statistical significance between parent and child nodes" (Analytics Vidhya), or post-pruning using error estimation, in order to avoid overfitting and get a better understanding of the significance of the data's variables, other models were explored, including Random Forest.

A Random Forest model was also fit to the dataset as it is able to handle large datasets containing higher dimensionality, which was thought to be an appropriate choice as the Ames housing dataset contains 79 variables. A Random Forest model was also chosen as it is able to "reduce correlation between trees by injecting more randomness into the tree-growing process" (Greenwell et al). It was also chosen as a means of identifying which of the 79 variables were significant while predicting house price.

In addition to a Random Forest model, a Bagging model was also fit to the dataset as it was thought that a Bagging model's methods of using collections of training data subsets to train multiple decision trees, of which the average would be used, would not only help avoid overfitting the data, but provide a more robust prediction of housing prices than a single Decision Tree model.

As feature selection is a large component of this project, it was thought that a LASSO model would also prove useful for reducing dimensionality. A LASSO model was thought to offer high prediction accuracy for this dataset since the model's method includes shrinking the regression coefficients (some of them to zero), while also reducing variance and minimizing bias.

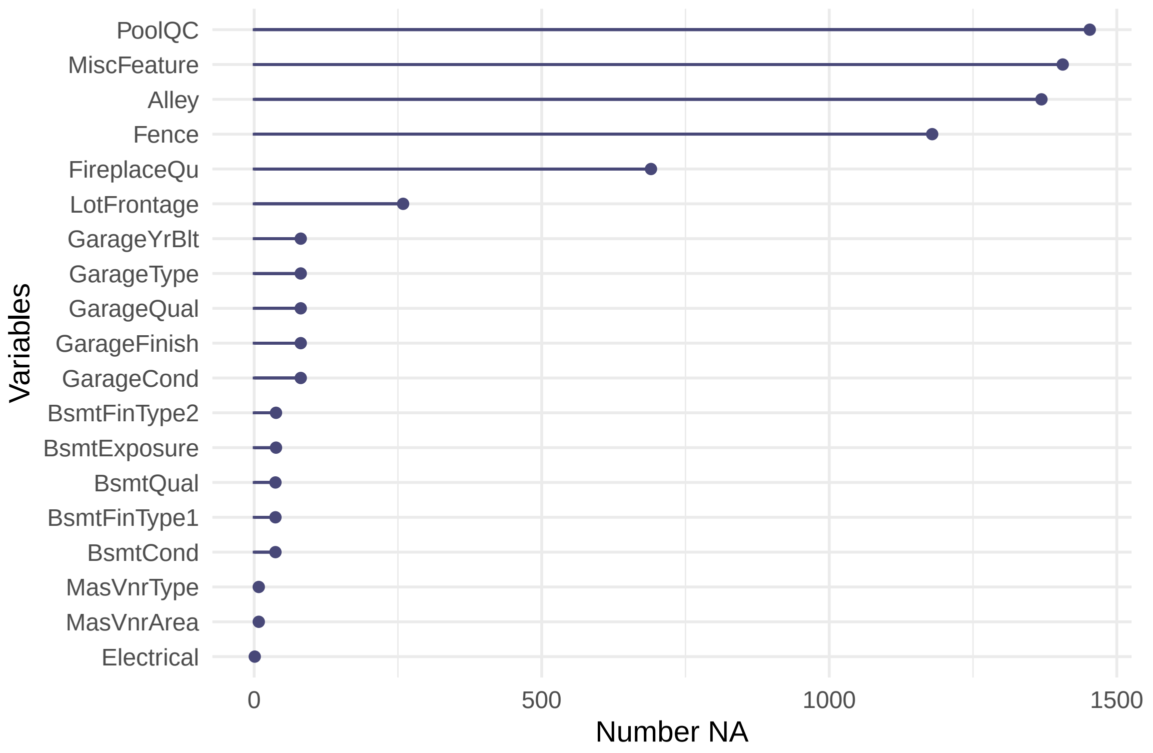
An AdaBoost Boosting model was also fit as it was thought that a Boosting model may increase predictive accuracy of housing prices via its ability add strength or weight to specific classifiers after taking into account the previous classifier's success. It was thought that a Boosting model would help reduce dimensionality by resulting in significant classifiers being assigned higher weights than less significant classifiers. However, it is believed that a Bagging model will perform better with this dataset than the chosen AdaBoost Boosting model, as Boosting does not help avoid over-fitting as a Bagging model does.

```r
test <- read.csv("/cloud/project/test.csv")
train <- read.csv("/cloud/project/train.csv")
```

Missing Values

After importing the testing and training data from the Kaggle repository, an initial analysis of missing values was conducted:

```r
# handle NA/missing values Approach: convert NA to another level name for
# categorical or 0 for nominal dont remove any columns

# visualize number of NA per variable
train_na <- train[, which(colSums(is.na(train)) > 0)]
gg_miss_var(train_na) + labs(y = "Number NA")
```



```r
# convert these levels to characters (later be converted to levels) not
# continuous variables
train$MSSubClass <- as.character(train$MSSubClass)
train$OverallQual <- as.character(train$OverallQual)
train$OverallCond <- as.character(train$OverallCond)

# Only 1 NA in Electrial --> replace with Mode Also impractical to not have
# electricity

# function to calculate the mode
getmode <- function(n) {
    uniq <- unique(n)
    uniq[which.max(tabulate(match(n, uniq)))]
}

mode_Electrical <- getmode(train$Electrical)
train$Electrical[is.na(train$Electrical)] <- mode_Electrical

# change the NA level in the categorical variables to NONE
train1 <- train %>%
    mutate_if(is.character, ~fct_explicit_na(., na_level = "None"))

# change the NA level in the continuous variables to 0
train1[is.na(train1)] = 0

train1 <- as.data.frame(unclass(train1), stringsAsFactors = TRUE)

# convert numeric classes to integer classes
for (col in colnames(train1)) {
    if (class(train1[, col]) == "numeric") {
        train1[, col] <- as.integer(train1[, col])
    }
}


orig_SalePrice <- train1$SalePrice
# sapply(train, class) cbind(lapply(lapply(train1, is.na), sum))
```

It was found that the data for the following variables contained the most Missing Values:

PoolQC > 1250 (>85%) Missing Values
MiscFeature > 1250 (>85%) Missing Values
Alley > 1250 (>85%) Missing Values
Fence > 1000 (>68%) Missing Values
FireplaceQu > 500 (>34%) Missing Values
LotFrontface > 250 (>17%) Missing Values

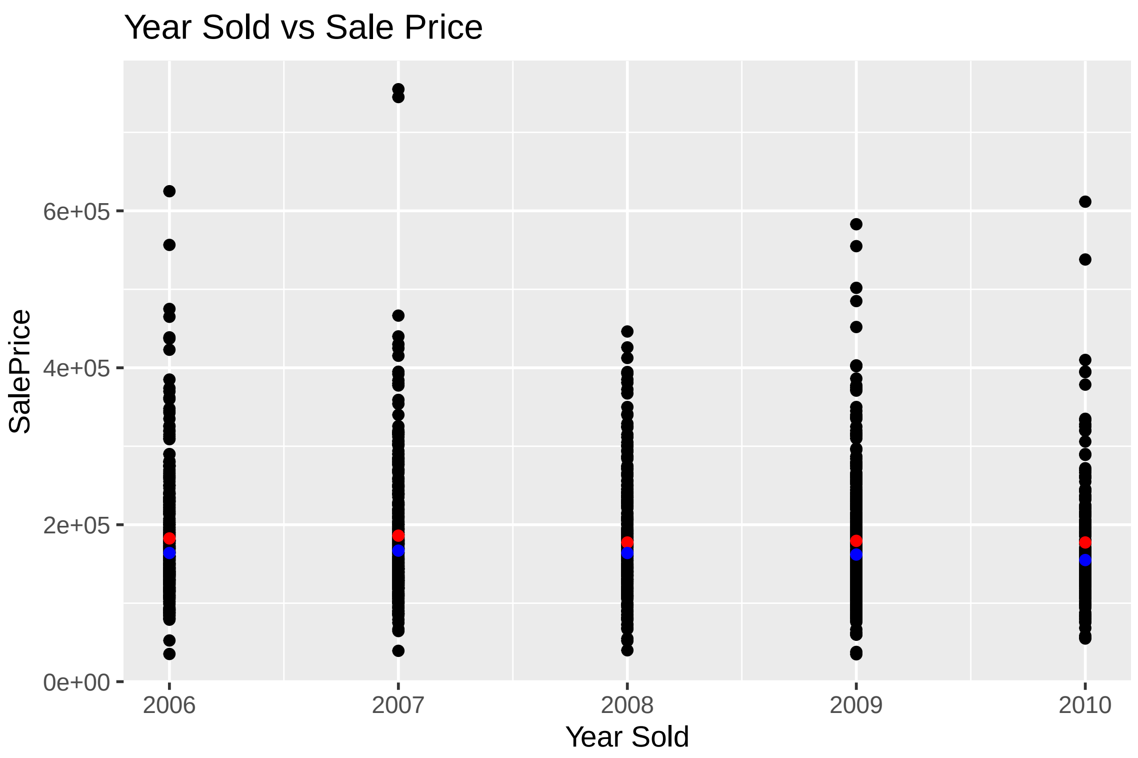
In order to handle the NA/Missing Values, the NA level in the categorical variables were changed to 'None' as these NA values could not be imputed by using the mean, mode, or interpolation of the feature. This was because it was impossible and impractical to impute the value for the quality of a home's pool (PoolQC), when the home did not come with a pool.

In order to handle the NA/Missing Values of continuous variables, all NA values were converted to zero. This made intuitive sense as as it did not make sense to impute any values other than zero for continuous variables such as MasVnrArea or masonry veneer area in square feet if a home did not contain any masonry veneer area.

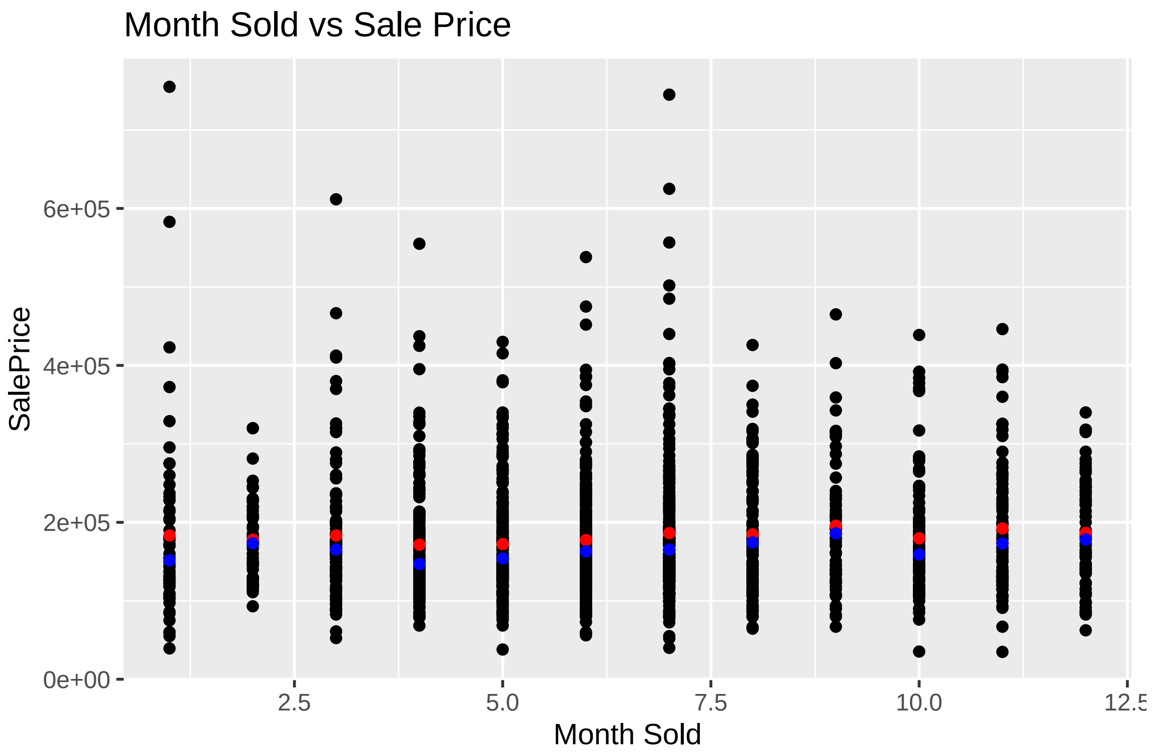
Exploratory Data Analysis

In order to better understand the data and gain insight into the relationship between variables, an explanatory data analysis was performed.

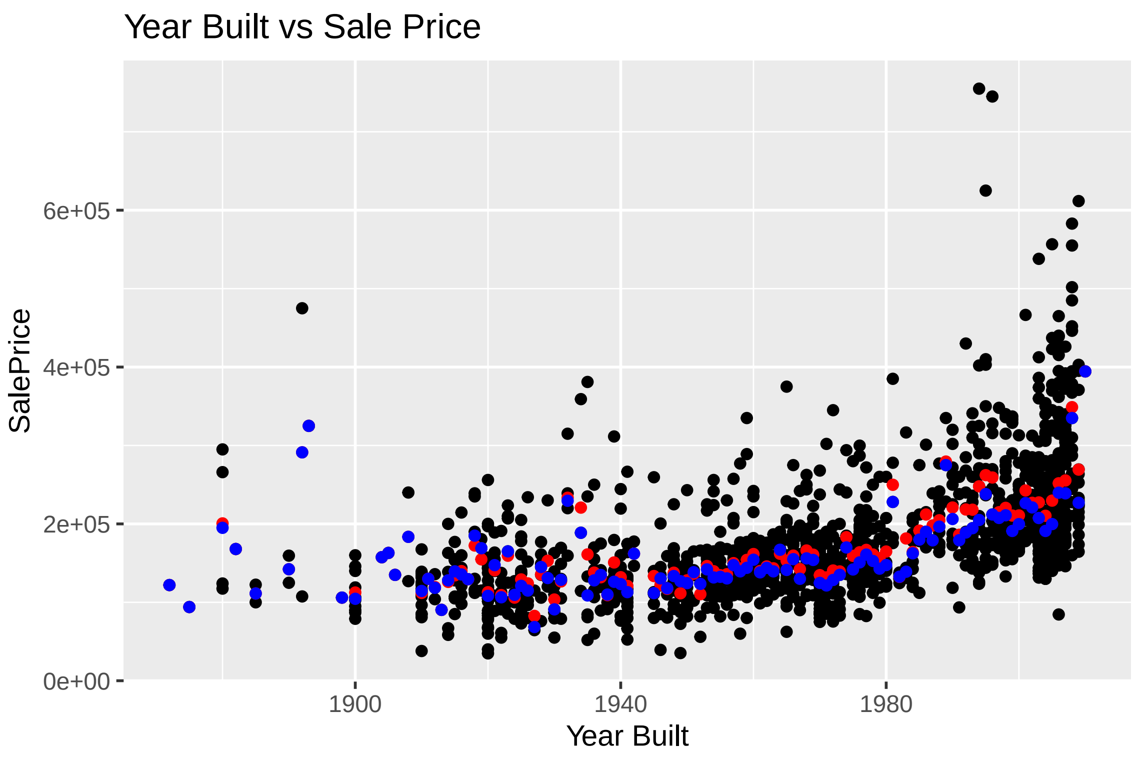
```
# See how time variables effect sale price
ggplot(train1, aes(x = YrSold, y = SalePrice)) + geom_point() + stat_summary(fun = "mean",
    geom = "point", color = "red") + stat_summary(fun = "median", geom = "point",
    color = "blue") + ggtitle("Year Sold vs Sale Price") + xlab("Year Sold")
```



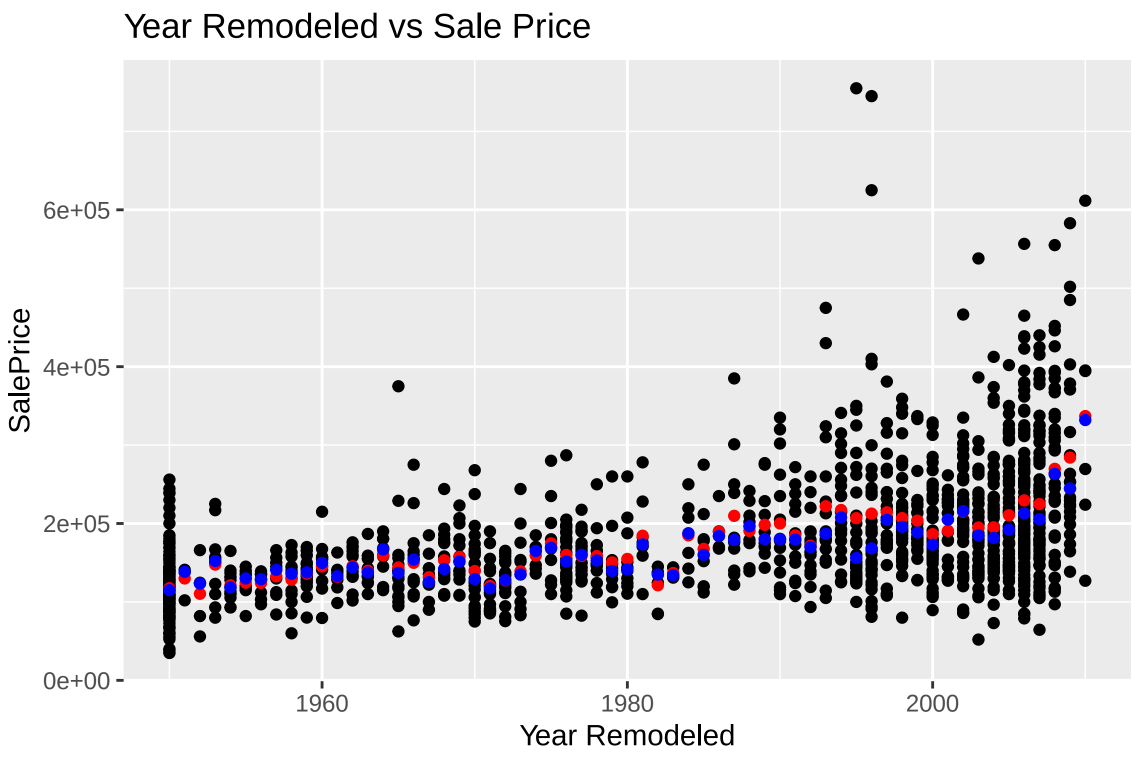
```
ggplot(train1, aes(x = MoSold, y = SalePrice)) + geom_point() + stat_summary(fun = "mean",
    geom = "point", color = "red") + stat_summary(fun = "median", geom = "point",
    color = "blue") + ggtitle("Month Sold vs Sale Price") + xlab("Month Sold")
```



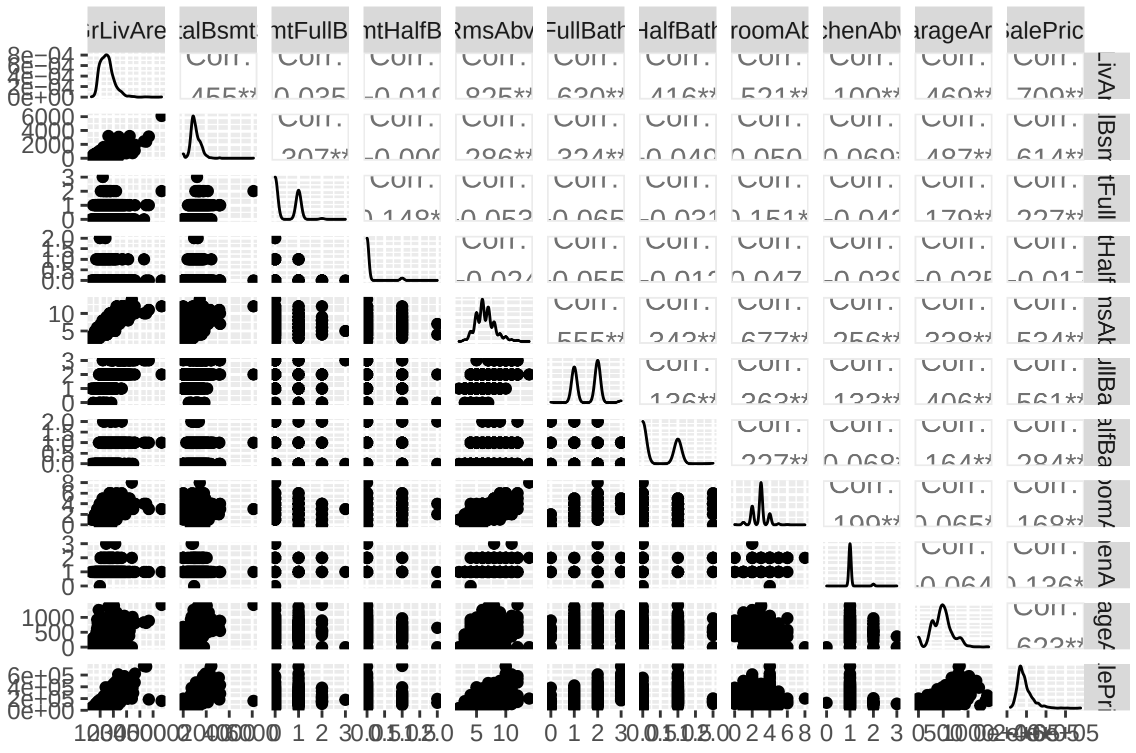
```
ggplot(train1, aes(x = YearBuilt, y = SalePrice)) + geom_point() + stat_summary(fun = "mean",
    geom = "point", color = "red") + stat_summary(fun = "median", geom = "point",
    color = "blue") + ggtitle("Year Built vs Sale Price") + xlab("Year Built")
```



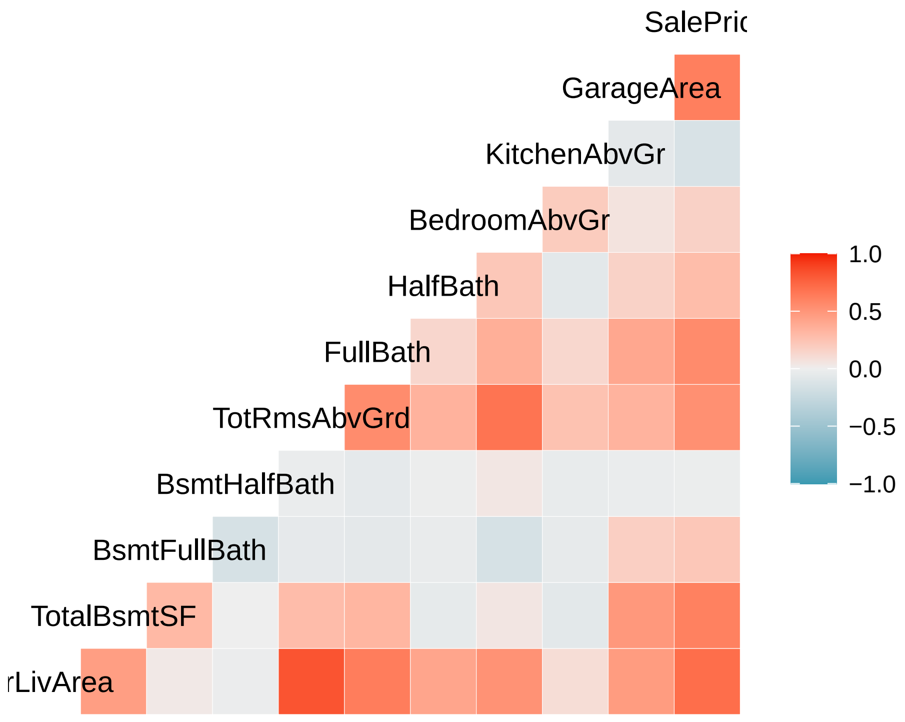
```
ggplot(train1, aes(x = YearRemodAdd, y = SalePrice)) + geom_point() + stat_summary(fun = "mean",
    geom = "point", color = "red") + stat_summary(fun = "median", geom = "point",
    color = "blue") + ggtitle("Year Remodeled vs Sale Price") + xlab("Year Remodeled")
```



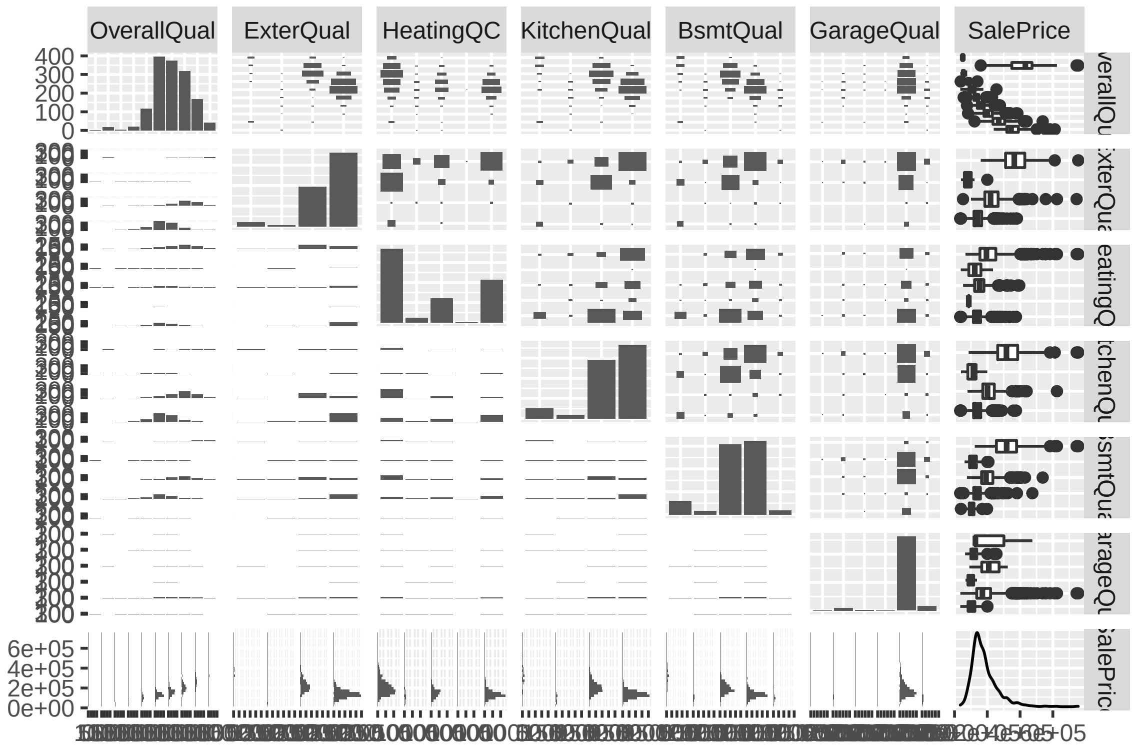
```
# Size indicator variables
df_size <- train1 %>%
    select(c(GrLivArea, TotalBsmtSF, BsmtFullBath, BsmtHalfBath, TotRmsAbvGrd, FullBath,
        HalfBath, BedroomAbvGr, KitchenAbvGr, GarageArea, SalePrice))
ggpairs(df_size)
```



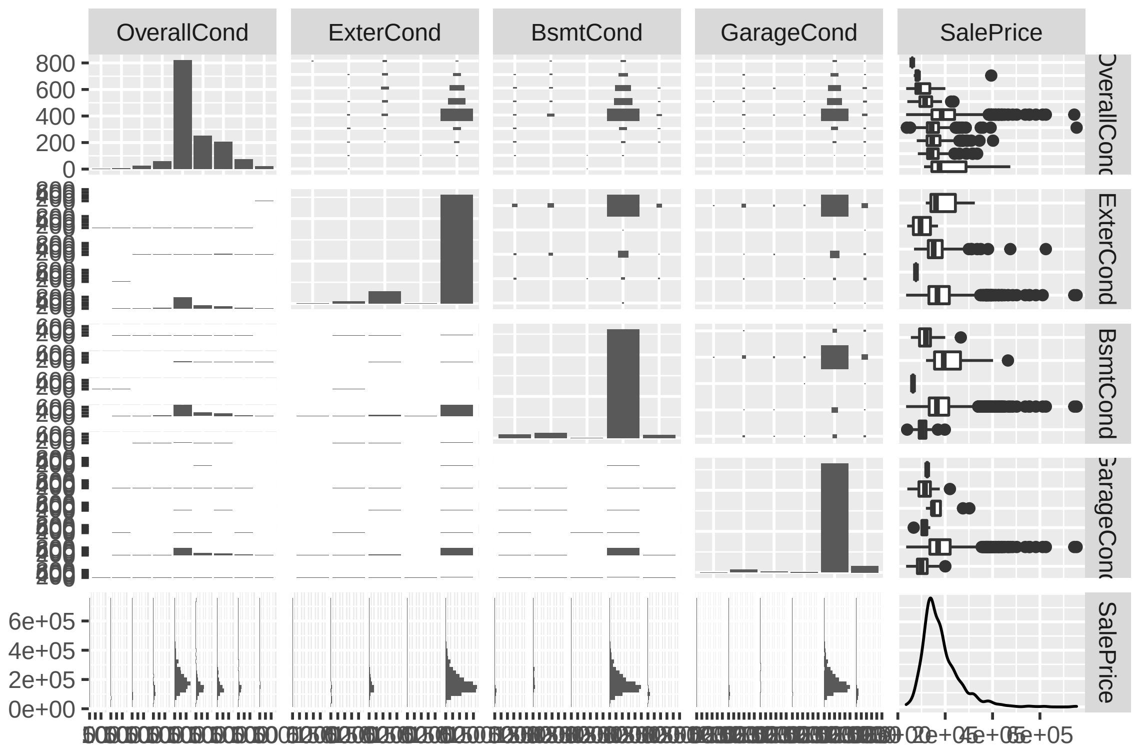
```
ggcorr(df_size, method = c("everything", "pearson"))
```



```
# Quality variables
df_qual <- train1 %>%
    select(c(OverallQual, ExterQual, HeatingQC, KitchenQual, BsmtQual, GarageQual,
        SalePrice))
ggpairs(df_qual)
```



```
# Condition variables
df_cond <- train1 %>%
    select(c(OverallCond, ExterCond, BsmtCond, GarageCond, SalePrice))
ggpairs(df_cond)
```



The first relationship explored in the Exploratory Data Analysis was the relationship between variables which involve time and SalePrice. These relationships were plotted as:

Sale Price vs. Year Sold
Sale Price vs. Month Sold
Sale Price vs. Year Built
Sale Price vs. Year Remodeled

The rather flat relationship between Sale Price and Year Sold proved interesting as it was thought that this relationship would show to be predominantly positive. It's been hypothesized that the housing market crash of 2007 might have affected this relationship.

The relationship between Sale Price and Month Sold was expected, with the exception of sale prices in January. It was expected that homes would sell for more money in the summer months, but it was not expected that January would prove to have the highest sale price of all months.

The relationship between Sale Price and Year Built was expected with a slightly positive relationship shown.

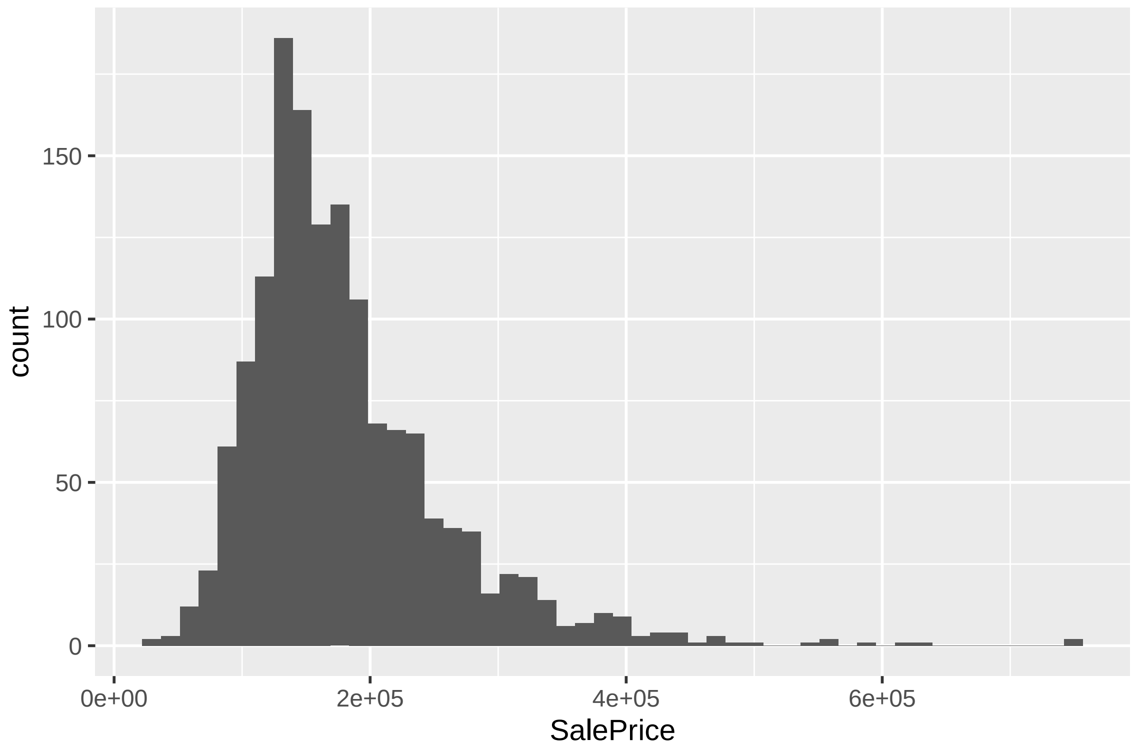
The relationship between Sale Price and Year Remodeled also was expected with a slightly positive relationship shown.

The second part of the Exploratory Data Analysis was to create multiple scatter plot matrices for indicator variables, quality variables, and condition variables. Here a predominantly positive relationship between Sale Price and the indicator variables was found. However, what stood out most was the skewed distribution of the Sale Price variable across all three scatterplot matrices. Upon further analysis it was discovered that the SalePrice variable should be log transformed.

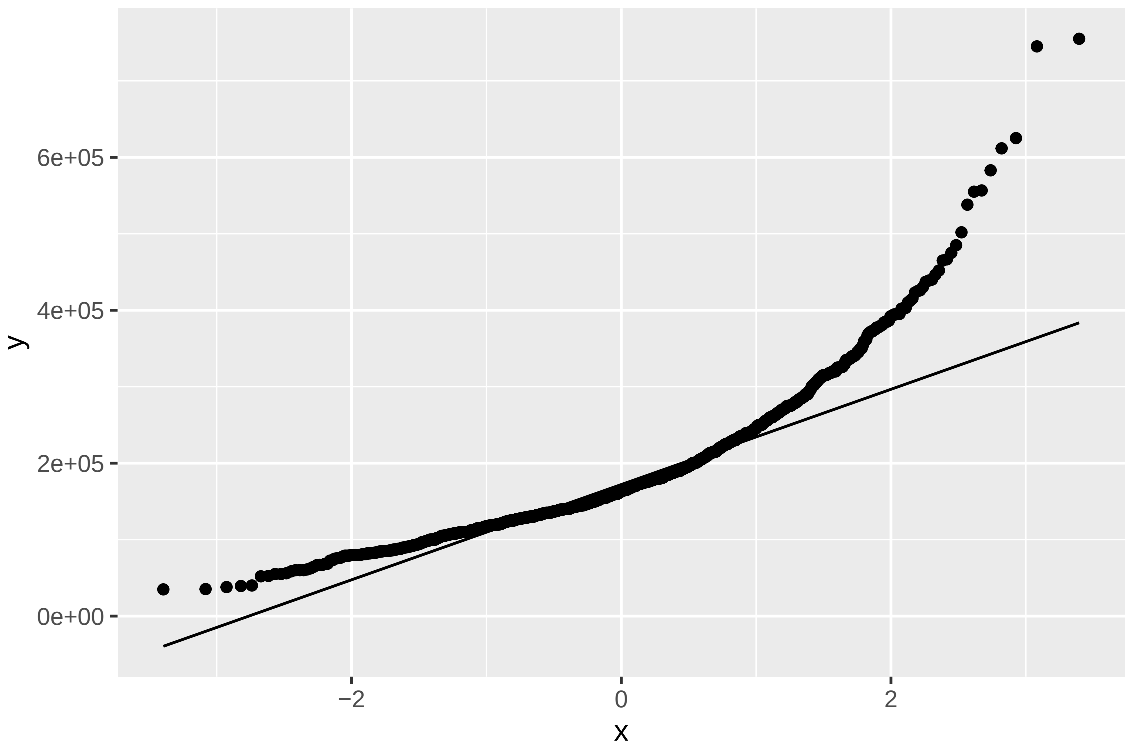
Log Transform the Data

The SalePrice variable was log transformed due to having a skewed distribution first discovered in the Exploratory Data Analysis.

```r
ggplot(data = train1, aes(SalePrice)) + geom_histogram(bins = 50)
```

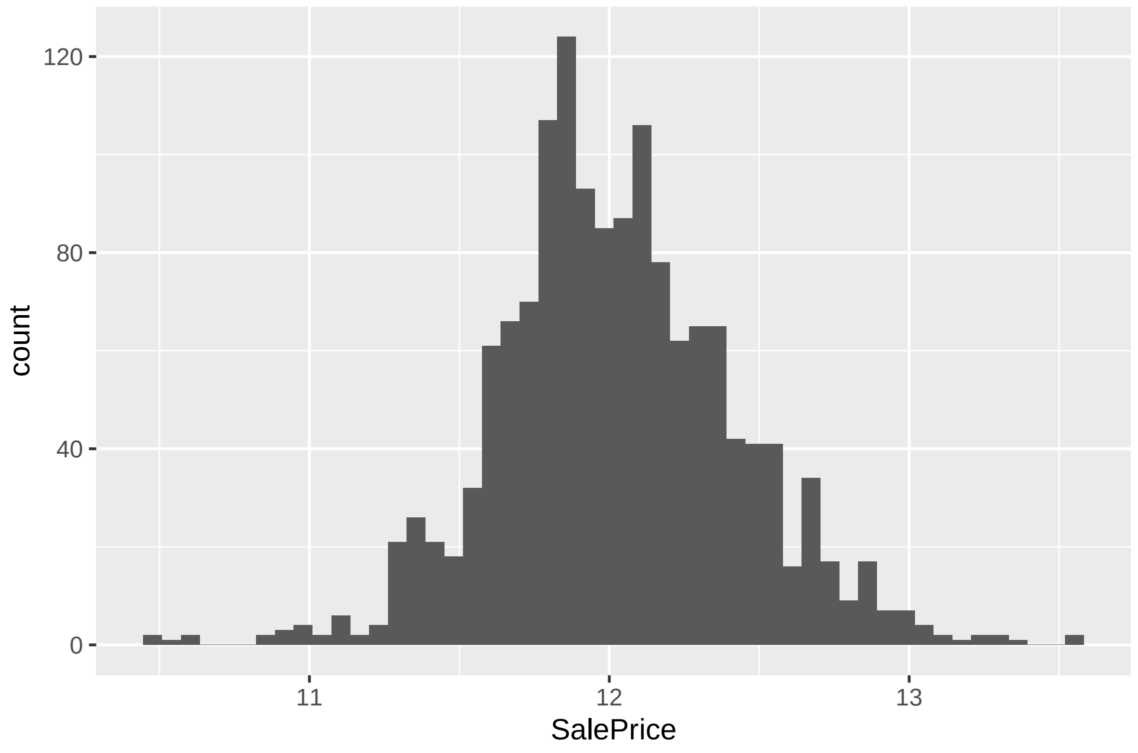


```r
ggplot(data = train1, aes(sample = SalePrice)) + stat_qq() + stat_qq_line()
```

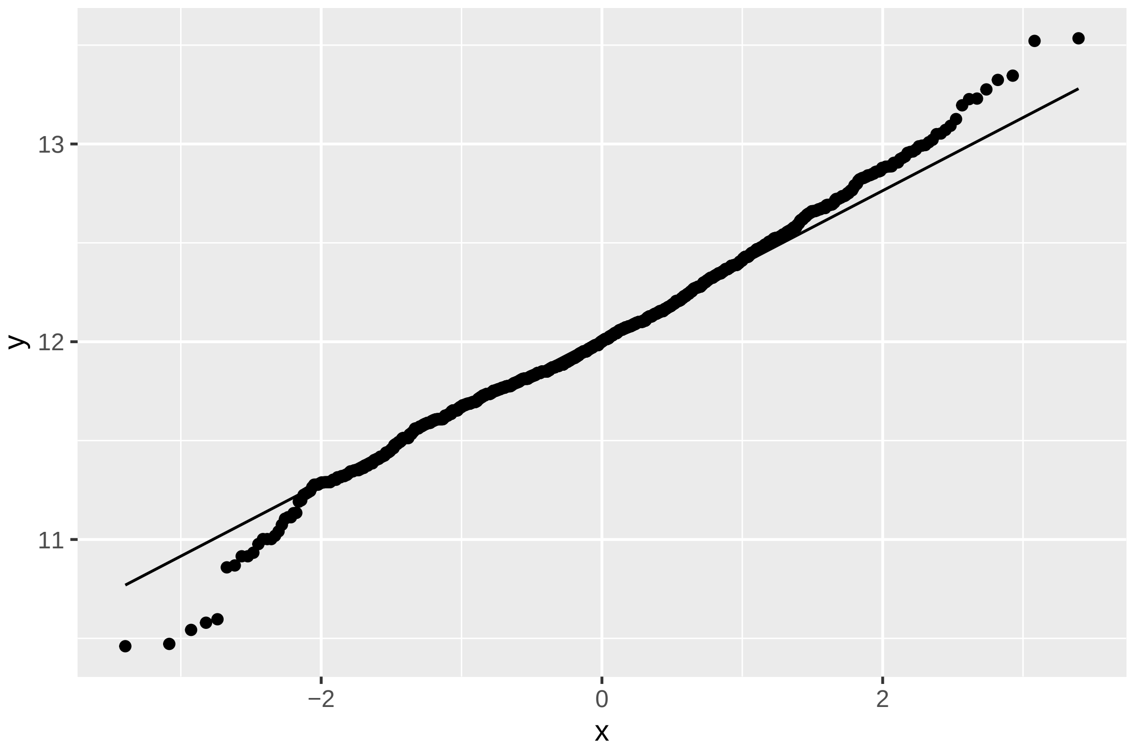


Further analysis of the SalePrice variable showed a non-normal right-skewed distribution in the above histogram. This non-normal distribution is also evident in the above (Q-Q) plot where the observations curve off of the line indicating the distribution is skewed.

```
train1$SalePrice <- log(train1$SalePrice)

ggplot(data = train1, aes(SalePrice)) + geom_histogram(bins = 50)
```



```
ggplot(data = train1, aes(sample = SalePrice)) + stat_qq() + stat_qq_line()
```



After log-transforming the SalePrice variable we now see a normal bell-curve shape in the distribution in the above histogram. The above (Q-Q) plot now shows the observations sticking closely to the line without any curvature away from the line.

Provide test set that contains Sale Price

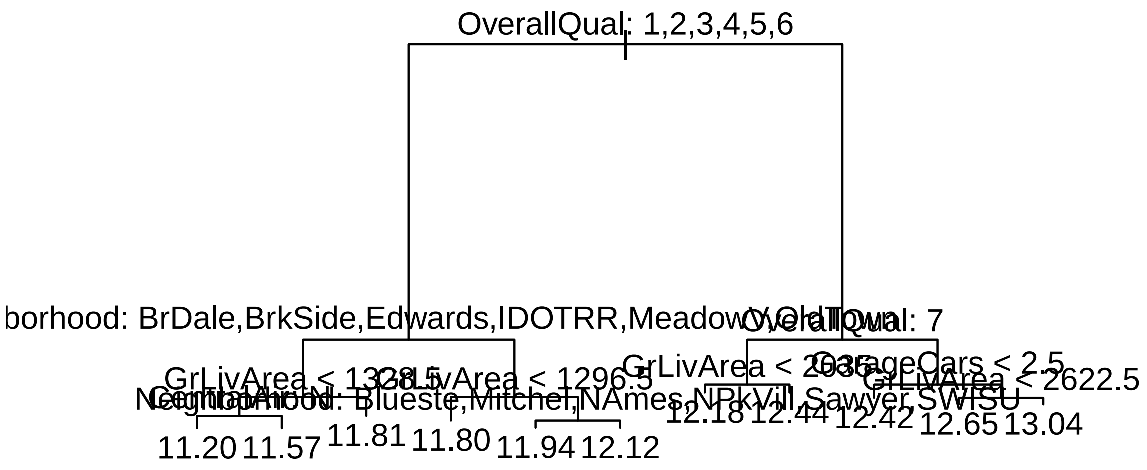
Upon further analysis, it was discovered that an aspect of the Kaggle Competition was that the test dataset did not contain a SalePrice column as this is the condition for ranking the effectiveness of the predictive models submitted. Therefore, the training dataset was split into new test and training sets in order to analyze the data and fit the models.

```
# Split training set into new test and train sets so test set has SalePrice
n <- nrow(train1)
train_split <- seq_len(n) %in% sample(seq_len(n), round(0.7 * n))
train2 <- data.frame(train1[train_split, ])
test2 <- data.frame(train1[-train_split, ])
```

Decision Tree

A Decision Tree model was first explored as was thought that the output would be easily interpreted and its graphical representations would be straightforwardly related to predicting Sale Price.
In order to fit the Decision Tree, a SalePrice tree was first created. From that tree, the variables used were listed along with omitted variables. The tree was subsequently plotted and the test MSE of the tree was calculated. The test MSE of the Decision Tree was later compared with the test MSE of the other chosen models in order to choose the most accurate model for predicting Sale Price.

```
# Create SalePrice tree
tree.SP <- tree(SalePrice ~ ., data = train2)

# Variables that get used:
print("**** Below are variables used in the tree ****")
## [1] "**** Below are variables used in the tree ****"
summary(tree.SP)$used
## [1] OverallQual  Neighborhood GrLivArea    CentralAir   GarageCars
## 81 Levels: <leaf> Id MSSubClass MSZoning LotFrontage LotArea Street ... SaleCondition

# Variables that are omitted in the tree
cat("\n")
print("**** Below are omitted variables ****")
## [1] "**** Below are omitted variables ****"
names(train2)[which(!(names(train2) %in% summary(tree.SP)$used))]
##  [1] "Id"            "MSSubClass"    "MSZoning"      "LotFrontage"
##  [5] "LotArea"       "Street"        "Alley"         "LotShape"
##  [9] "LandContour"   "Utilities"     "LotConfig"     "LandSlope"
## [13] "Condition1"    "Condition2"    "BldgType"      "HouseStyle"
## [17] "OverallCond"   "YearBuilt"     "YearRemodAdd"  "RoofStyle"
## [21] "RoofMatl"      "Exterior1st"   "Exterior2nd"   "MasVnrType"
## [25] "MasVnrArea"    "ExterQual"     "ExterCond"     "Foundation"
## [29] "BsmtQual"      "BsmtCond"      "BsmtExposure"  "BsmtFinType1"
## [33] "BsmtFinSF1"    "BsmtFinType2"  "BsmtFinSF2"    "BsmtUnfSF"
## [37] "TotalBsmtSF"   "Heating"       "HeatingQC"     "Electrical"
## [41] "X1stFlrSF"     "X2ndFlrSF"     "LowQualFinSF"  "BsmtFullBath"
## [45] "BsmtHalfBath"  "FullBath"      "HalfBath"      "BedroomAbvGr"
## [49] "KitchenAbvGr"  "KitchenQual"   "TotRmsAbvGrd"  "Functional"
## [53] "Fireplaces"    "FireplaceQu"   "GarageType"    "GarageYrBlt"
## [57] "GarageFinish"  "GarageArea"    "GarageQual"    "GarageCond"
## [61] "PavedDrive"    "WoodDeckSF"    "OpenPorchSF"   "EnclosedPorch"
## [65] "X3SsnPorch"    "ScreenPorch"   "PoolArea"      "PoolQC"
## [69] "Fence"         "MiscFeature"   "MiscVal"       "MoSold"
## [73] "YrSold"        "SaleType"      "SaleCondition" "SalePrice"

plot(tree.SP)
text(tree.SP, pretty = 0)
```

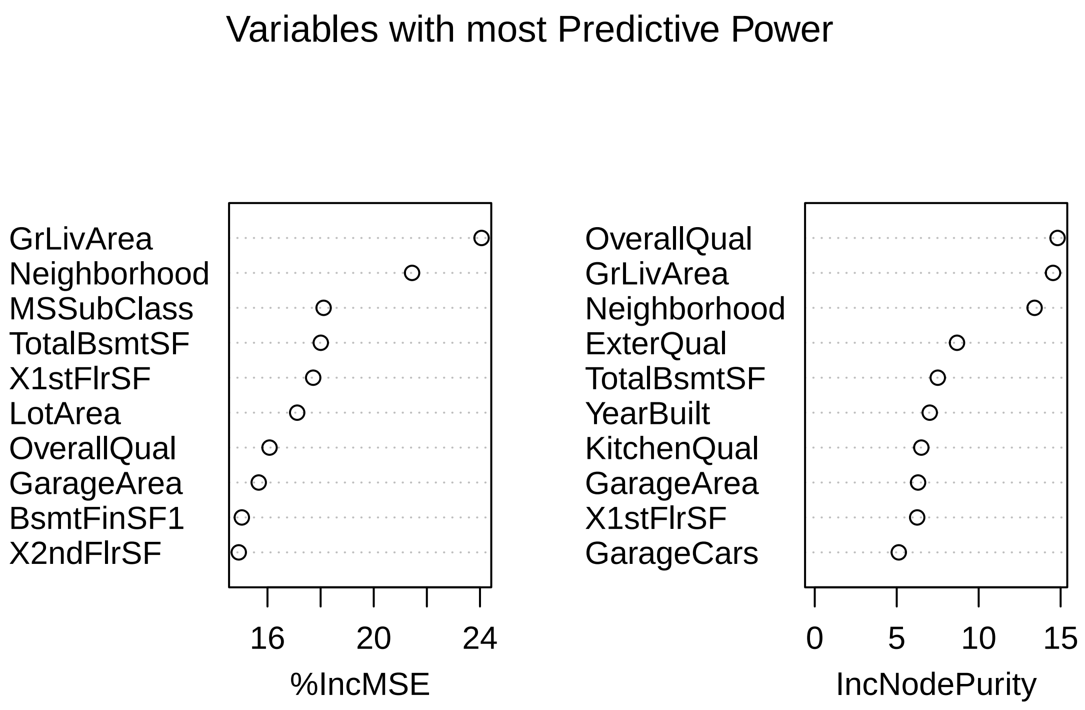


```
dt_MSE <- mean((predict(tree.SP, test2) - test2$SalePrice)^2)
print(paste("Decision Tree Test MSE = ", dt_MSE))
## [1] "Decision Tree Test MSE =  0.0393537499977425"
```

Random Forest

A Random Forest model was also fit as it was believed that it would be able to handle large datasets containing higher dimensionality, which was thought to be an appropriate choice as the Ames housing dataset contains 79 variables. A Random Forest model was also chosen as it is able to "reduce correlation between trees by injecting more randomness into the tree-growing process" (Greenwell et al). It was also chosen as a means of identifying which of the 79 variables were significant while predicting house price. \
After fitting the Random Forest model, the variables with the most predictive power were found. The test MSE of the Random Forest model was then calculated. The test MSE of the Random Forest model was later compared with the test MSE of the other chosen models in order to choose the most accurate model for predicting Sale Price.

```
rf <- randomForest(SalePrice ~ ., train2, mtry = floor(sqrt(ncol(train2) - 1)), importance = TRUE)

varImpPlot(rf, sort = TRUE, n.var = 10, main = "Variables with most Predictive Power")
```



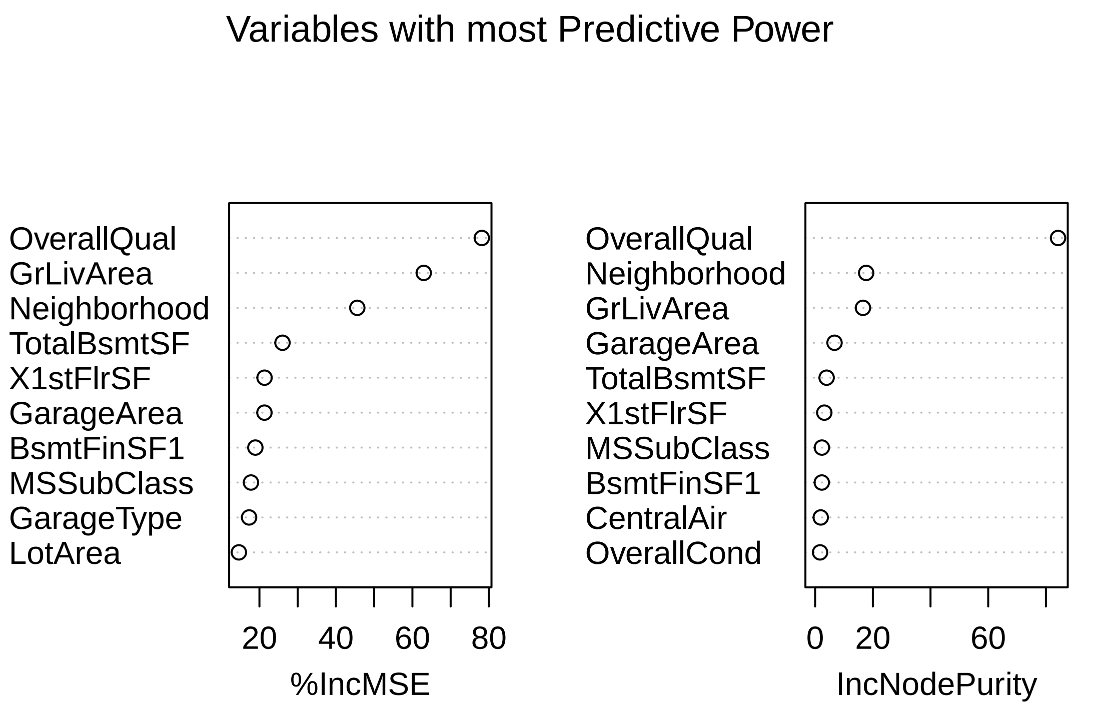
```
rf_MSE <- mean((predict(rf, test2) - test2$SalePrice)^2)
print(paste("Random Forest Test MSE = ", rf_MSE))
```

```
## [1] "Random Forest Test MSE =  0.00970699191926917"
```

Bagging

Next, a Bagging model was fit to the dataset as it was thought that a Bagging model's methods of using collections of training data subsets to train multiple decision trees, of which the average would be used, would not only help avoid overfitting the data, but provide a more robust prediction of housing prices than a single Decision Tree model. The test MSE of the Bagging model was later compared with the test MSE of the other chosen models in order to choose the most accurate model for predicting Sale Price. \

```
bag_fit <- randomForest(SalePrice ~ ., data = train2, mtry = ncol(train2) - 1, importance = TRUE)

varImpPlot(bag_fit, sort = TRUE, n.var = 10, main = "Variables with most Predictive Power")
```



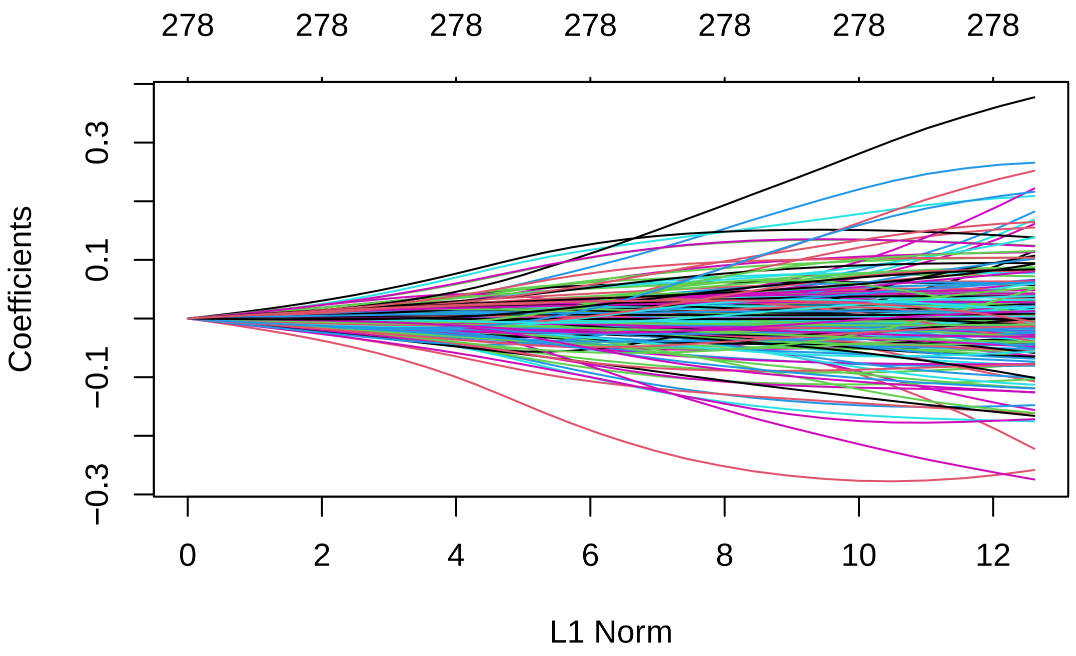
```r
bag_MSE <- mean((predict(bag_fit, test2) - test2$SalePrice)^2)
bag_MSE
```

```
## [1] 0.009636063
```

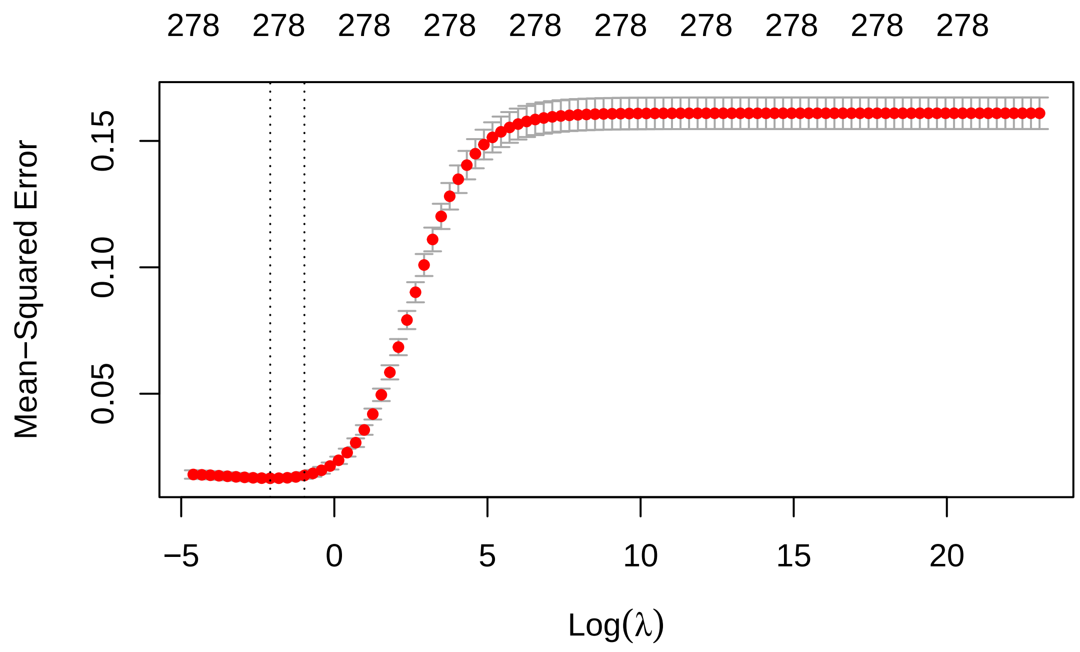
LASSO

As feature selection was thought to be a large component of the project, a LASSO model was chosen as it offers high prediction accuracy as aids with high dimensionality by shrinking the regression coefficients (some of them to zero).\
In order to fit a LASSO model, first the training and testing split data were transformed into matrices and lambda values were added. The coefficients were plotted along with test MSE at different lambta values. The best test MSE of the LASSO model was then calculated. The test MSE of the LASSO model was later compared with the test MSE of the other chosen models in order to choose the most accurate model for predicting Sale Price.

```r
trnmat <- model.matrix(SalePrice ~ ., data = train2)
tstmat <- model.matrix(SalePrice ~ ., data = test2)

lambda = 10^seq(-2, 10, length.out = 100)

lasso.mod <- glmnet(trnmat, train2$SalePrice, alpha = 0, lambda = lambda)

plot(lasso.mod)
```



```
cv.lasso <- cv.glmnet(trnmat, train2$SalePrice, alpha = 0, lambda = lambda, folds = 10)

plot(cv.lasso)
```



```
bestlam.lasso <- cv.lasso$lambda.min
print(paste("LASSO Test MSE = ", bestlam.lasso))
```

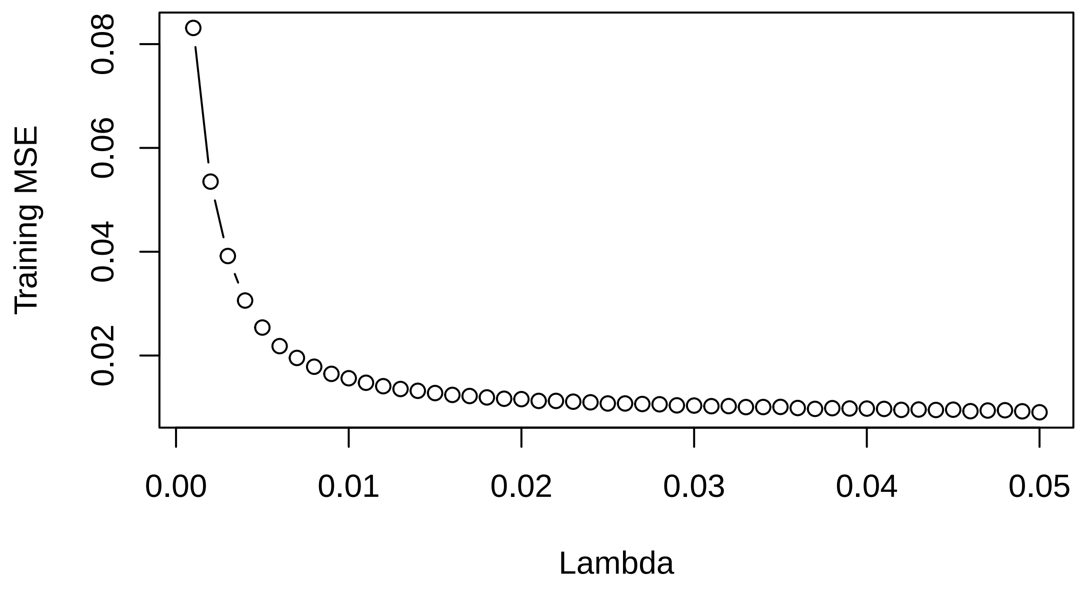
```
## [1] "LASSO Test MSE =  0.123284673944207"
```

```
best.lasso <- glmnet(trnmat, train2$SalePrice, alpha = 0, lambda = bestlam.lasso)

pred.lasso <- predict(lasso.mod, s = bestlam.lasso, newx = tstmat)

lasso_MSE <- mean((test2$SalePrice - pred.lasso)^2)
```

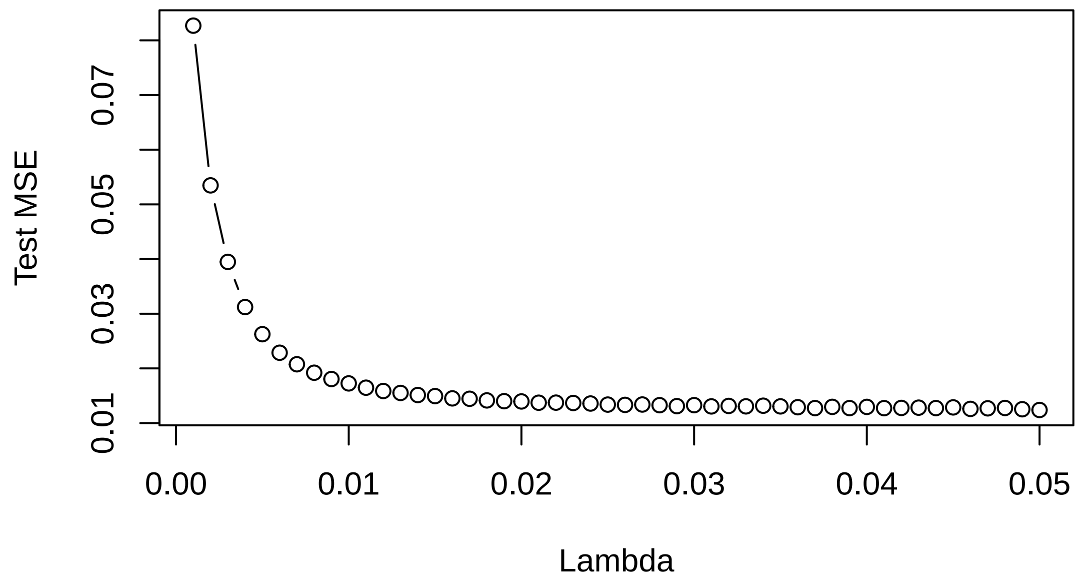
Boosting

An AdaBoost Boosting model was also fit as it was thought that a Boosting model may increase predictive accuracy of housing prices via its ability add strength or weight to specific classifiers after taking into account the previous classifier's success. It was thought that a Boosting model would help reduce dimensionality by resulting in significant classifiers being assigned higher weights than less significant classifiers. The test MSE of the AdaBoost Boosting model was later compared with the test MSE of the other chosen models in order to choose the most accurate model for predicting Sale Price.

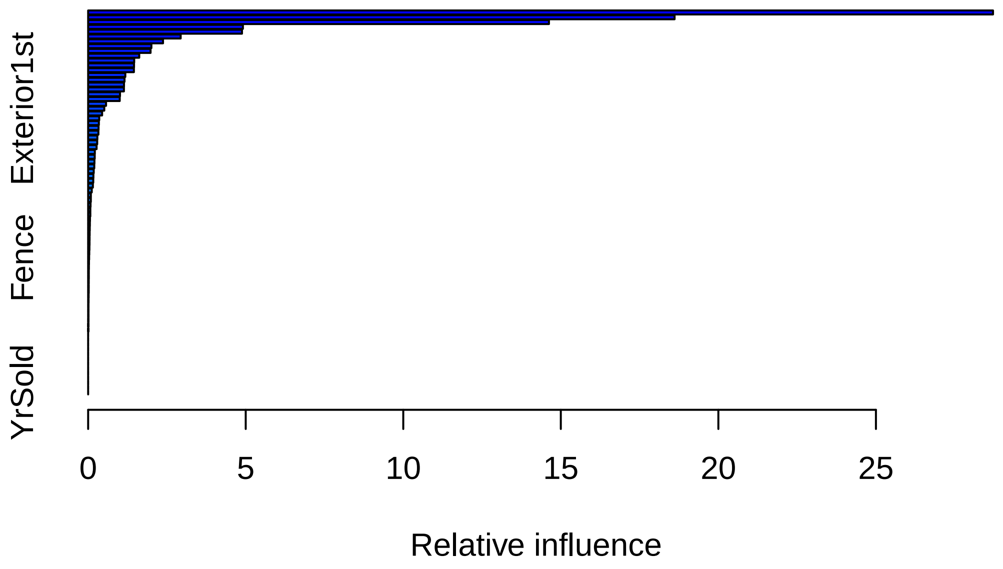
```
lambs <- seq(0.001, 0.05, length.out = 50)
length_lamb <- length(lambs)
tr_err <- rep(NA, length_lamb)
test_err <- rep(NA, length_lamb)

for (i in 1:length_lamb) {
    boost_hit <- gbm(SalePrice ~ ., data = train2, distribution = "gaussian", n.trees = 1000,
        shrinkage = lambs[i], verbose = F)
    tr_pred <- predict(boost_hit, train2, n.trees = 1000)
    test_pred <- predict(boost_hit, test2, n.trees = 1000)
    tr_err[i] <- mean((tr_pred - train2$SalePrice)^2)
    test_err[i] <- mean((test_pred - test2$SalePrice)^2)
}
plot(lambs, tr_err, type = "b", xlab = "Lambda", ylab = "Training MSE")
```



```
plot(lambs, test_err, type = "b", xlab = "Lambda", ylab = "Test MSE")
```



```r
boost_fit <- gbm(SalePrice ~ ., data = train2, distribution = "gaussian", n.trees = 1000,
    shrinkage = lambs[which.min(test_err)])

summary(boost_fit)
```



```
##                           var      rel.inf
## OverallQual      OverallQual 28.716246891
## GrLivArea          GrLivArea 18.611557652
## Neighborhood    Neighborhood 14.623299752
## ExterQual          ExterQual  4.911408710
## TotalBsmtSF      TotalBsmtSF  4.882320561
## GarageArea        GarageArea  2.936627495
## OverallCond      OverallCond  2.376467286
## X1stFlrSF          X1stFlrSF  2.007089543
## CentralAir        CentralAir  1.980939637
## MSSubClass        MSSubClass  1.621464759
## YearRemodAdd    YearRemodAdd  1.461286197
## GarageType        GarageType  1.456031510
## BsmtFinSF1       BsmtFinSF1  1.452267045
## LotArea             LotArea  1.179430782
## KitchenQual     KitchenQual  1.154701824
## GarageCars       GarageCars  1.137940832
## FireplaceQu     FireplaceQu  1.135751444
## SaleCondition SaleCondition  1.011012295
## MSZoning           MSZoning  1.003169041
## BsmtQual           BsmtQual  0.568865994
## Exterior1st     Exterior1st  0.512101236
## Functional       Functional  0.443052628
## ExterCond         ExterCond  0.351713696
## SaleType           SaleType  0.340458055
## Exterior2nd     Exterior2nd  0.330585212
## GarageYrBlt     GarageYrBlt  0.325577967
## GarageQual       GarageQual  0.292292664
## GarageCond       GarageCond  0.287472438
## Condition1       Condition1  0.268567681
## YearBuilt         YearBuilt  0.212149486
## BsmtUnfSF         BsmtUnfSF  0.209035694
## WoodDeckSF       WoodDeckSF  0.199432529
## BsmtCond           BsmtCond  0.194336819
## BsmtFinType1   BsmtFinType1  0.177307305
## Id                       Id  0.166807131
## ScreenPorch     ScreenPorch  0.162221936
## BsmtExposure   BsmtExposure  0.150257243
## OpenPorchSF     OpenPorchSF  0.115028463
## FullBath           FullBath  0.086225436
## HeatingQC         HeatingQC  0.085835574
## BsmtFullBath   BsmtFullBath  0.076250445
## Fireplaces       Fireplaces  0.069755731
## Alley                 Alley  0.069570557
## BedroomAbvGr   BedroomAbvGr  0.058094005
## LotFrontage     LotFrontage  0.054694329
## LotShape           LotShape  0.051908421
## X2ndFlrSF         X2ndFlrSF  0.048400463
## LotConfig         LotConfig  0.045552841
## EnclosedPorch EnclosedPorch  0.043836923
## LowQualFinSF   LowQualFinSF  0.040564019
## HalfBath           HalfBath  0.035403540
## Fence                 Fence  0.030943782
## Electrical       Electrical  0.026268402
## RoofStyle         RoofStyle  0.023973876
## PavedDrive       PavedDrive  0.021481764
## LandContour     LandContour  0.021358610
## MiscVal             MiscVal  0.019718920
## HouseStyle       HouseStyle  0.018668038
## TotRmsAbvGrd   TotRmsAbvGrd  0.018349795
## BldgType           BldgType  0.015217350
## Foundation       Foundation  0.013902940
## LandSlope         LandSlope  0.011101216
## MasVnrArea       MasVnrArea  0.010938763
## MoSold               MoSold  0.010919914
## X3SsnPorch       X3SsnPorch  0.010371747
## Heating             Heating  0.007502745
## BsmtFinSF2       BsmtFinSF2  0.006912420
## Street               Street  0.000000000
## Utilities         Utilities  0.000000000
## Condition2       Condition2  0.000000000
## RoofMatl           RoofMatl  0.000000000
## MasVnrType       MasVnrType  0.000000000
## BsmtFinType2   BsmtFinType2  0.000000000
## BsmtHalfBath   BsmtHalfBath  0.000000000
## KitchenAbvGr   KitchenAbvGr  0.000000000
## GarageFinish   GarageFinish  0.000000000
## PoolArea           PoolArea  0.000000000
## PoolQC               PoolQC  0.000000000
## MiscFeature     MiscFeature  0.000000000
## YrSold               YrSold  0.000000000
```

```r
boost_MSE <- min(test_err)
boost_MSE
```

```
## [1] 0.01239545
```

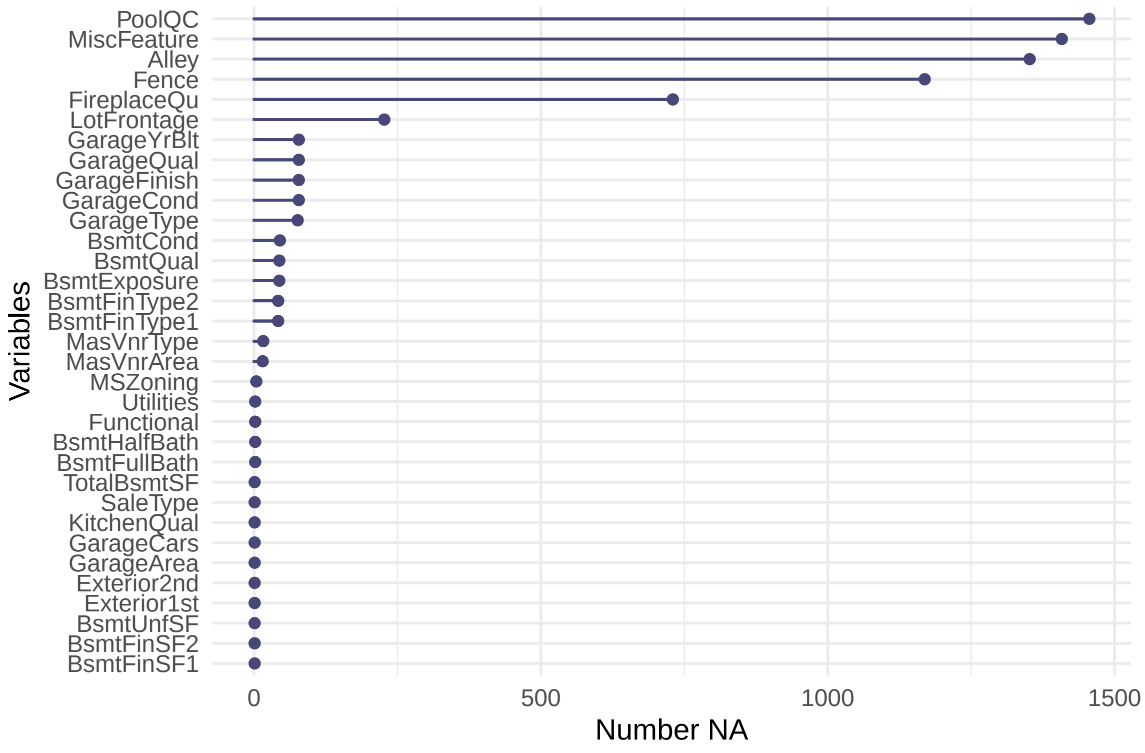
Table of Test MSE Values

In order to compare the test MSE values of the chosen models, a table was created. The table shows that the MSE for the Bagging and Random Forest models to be the smallest with Bagging having a slightly smaller MSE. However, the Random Forest model was still selected as it has better interpretability when compared to Bagging. And the difference in MSE is not drastic.

```r
MSE_table <- data.frame(Model = c("Decision Tree", "Random Forest", "Bagged Forest",
    "Boosted Forest", "LASSO"), MSE = c(dt_MSE, rf_MSE, bag_MSE, boost_MSE, lasso_MSE))

kable((MSE_table), caption = "MSE Values for Different Models", digits = 4)
```

Table 1: MSE Values for Different Models

Model	MSE
Decision Tree	0.0394
Random Forest	0.0097
Bagged Forest	0.0096
Boosted Forest	0.0124
LASSO	0.0166



```
# Remove rows that contain levels in test that are not in train cant perform
# any prediction if this is the case

# MSSubClass --> 150 only one instance --> remove that row
test1 %>%
    group_by(MSSubClass) %>%
    summarise(no_rows = length(MSSubClass))
```

```
## # A tibble: 16 x 2
##    MSSubClass no_rows
##    <fct>        <int>
##  1 120             95
##  2 150              1
##  3 160             65
##  4 180              7
##  5 190             31
##  6 20             543
##  7 30              70
##  8 40               2
##  9 45               6
## 10 50             143
## 11 60             276
## 12 70              68
## 13 75               7
## 14 80              60
## 15 85              28
## 16 90              57
```

```r
# save Id and row number from where this was taken will need to add a value for
# this later
row_Sub_150 <- which(test1$MSSubClass == "150")
id_Sub_150 <- test1[row_Sub_150, "Id"]

test3 <- droplevels(test1[!test1$MSSubClass == "150", ])

# str(train1) str(test3)

# make levels in train present in test
for (col in colnames(test3)) {
    if (class(test3[, col]) == "factor") {
        test3[, col] <- factor(test3[, col], levels = levels(train1[, col]))
    }
}

# test3$Exterior1st <- factor(test3$Exterior1st, levels =
# levels(train1$Exterior1st)) identical(levels(test3$Exterior1st),
# levels(train1$Exterior1st))
```

```r
# fit rf (chosen model of the ones used)
rf2 <- randomForest(SalePrice ~ ., train1, mtry = floor(sqrt(ncol(train1) - 1)),
    importance = TRUE)

pred <- data.frame(predict(rf2, test3))

# take exp of sale price --> did log transformation previously
pred.sale <- exp(pred)

# make table with SalePrice and Id
names(pred.sale)[1] <- "SalePrice"
pred.sale$Id <- test3$Id

pred2 <- pred.sale[c("Id", "SalePrice")]

# get mean SalePrice --> will add this to row where the MSSubClass of 150 was
# removed
mean_price <- mean(pred2$SalePrice)
```

```r
# add row for the MSSubClass row that was removed
r <- row_Sub_150
newrow <- cbind(id_Sub_150, mean_price)

insertRow <- function(existingDF, newrow, r) {
    existingDF[seq(r + 1, nrow(existingDF) + 1), ] <- existingDF[seq(r, nrow(existingDF)),
        ]
    existingDF[r, ] <- newrow
    existingDF
}

pred3 <- insertRow(pred2, newrow, r)
```

Results

Outside Exploration

References

Ames, Iowa: Alternative to the Boston Housing Data as an . . . http://jse.amstat.org/v19n3/decock.pdf.

"Convert Character to Factor in R: Vector, Data Frame Columns & Variable." Statistics Globe, 14 June 2021, https://statisticsglobe.com/convert-character-to-factor-in-r.

Greenwell, Bradley Boehmke & Brandon. "Hands-on Machine Learning with R." Chapter 11 Random Forests, 1 Feb. 2020, https://bradleyboehmke.github.io/HOML/random-forest.html#fn29.

Holtz, Yan. "Correlation Matrix with GGALLY." – The R Graph Gallery, https://www.r-graph-gallery.com/199-correlation-matrix-with-ggally.html#:~:text=The%20ggpairs()%20function%20of,is%20displayed%20on%20the%20right.

"House Prices - Advanced Regression Techniques." Kaggle, https://www.kaggle.com/c/house-prices-advanced-regression-techniques.

Sprecher, Stu. "Deluxe Homes LLC Housing Prices." 1 Dec. 2021.

"Tree Based Algorithms: Implementation in Python & R." Analytics Vidhya, 26 Aug. 2021, https://www.analyticsvidhya.com/blog/2016/04/tree-based-algorithms-complete-tutorial-scratch-in-python/.