Predicting Housing Prices with R

Samson Haile

I. Executive Summary

For my project I will be analyzing a cleaned version of Ames Housing dataset was compiled by Dean De Cock for use in data science education.

The dataset records medv (median house value) for 1,460 houses in Ames, Iowa. We will seek to predict SalePrice using 24 predictors such as; OverallQual (Rates the overall material and finish of the house). YearBuilt (Original construction date), a X1stFlrSF (First Floor square feet).

The statistical methods I will use include:

Linear Regression, Regression Trees, Subset Selection, Generalized Additive Models, Random Forest, Gradient Boosting, and Bagging.

We will test our models performance by calculating the Mean Squared Error (MSE).

Results Overview

Each model used resulted in fairly similar predictors that strongly correlate to house prices. These predictors include:

LotArea - Lot size in square feet

OverallQual - Rates the overall material and finish of the house

```
10   Very Excellent
9    Excellent
8    Very Good
7    Good
6    Above Average
5    Average
4    Below Average
3    Fair
2    Poor
1    Very Poor
```

X1stFlrSF - First Floor square feet

GarageCars/GarageArea - Size of garage in car capacity/Garage square feet

II. Description of Data

The dataset is split into a training set and a testing set. There are 1460 rows and 24 predictor columns.

```r
# importing libraries
library(ISLR)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.1.3
```

```
## randomForest 4.7-1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```r
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```r
library(xgboost)
```

```
## Warning: package 'xgboost' was built under R version 4.1.3
```

```r
library(FNN)
```

```
## Warning: package 'FNN' was built under R version 4.1.3
```

```r
library(MASS)
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 4.1.3
```

```r
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.1.3
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-4
```

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.1.3
```

```
## corrplot 0.92 loaded
```

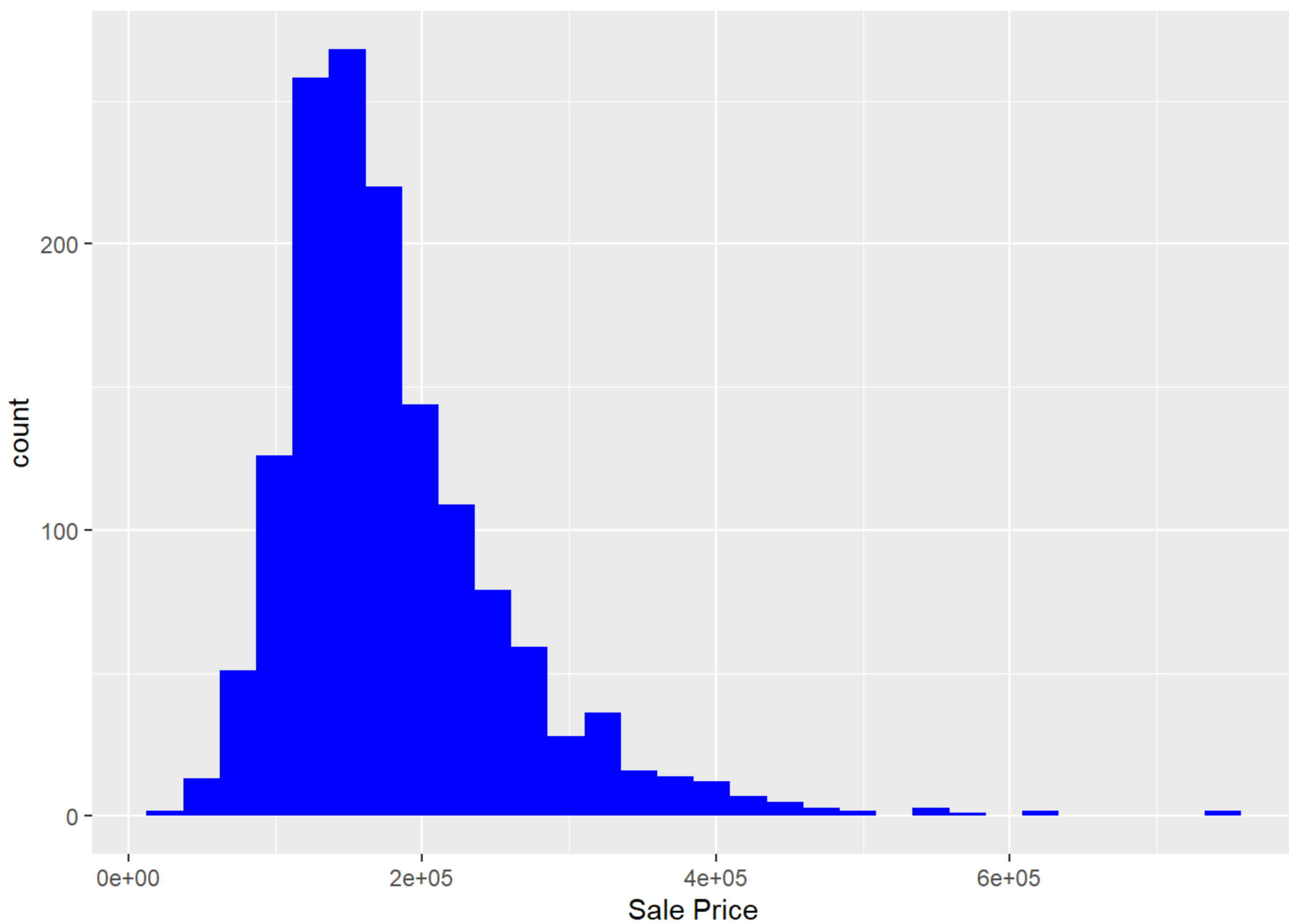
```
#importing data
train <- read.csv('train_new.csv')
test <- read.csv('test_new.csv')
```

The response variable; SalePrice

As you can see, the sale prices are right skewed. This was expected as few people can afford very expensive houses. I will keep this in mind, and take measures before modeling.

```
ggplot(data=train, aes(x=SalePrice)) +
        geom_histogram(fill="blue") + labs(x='Sale Price')
```

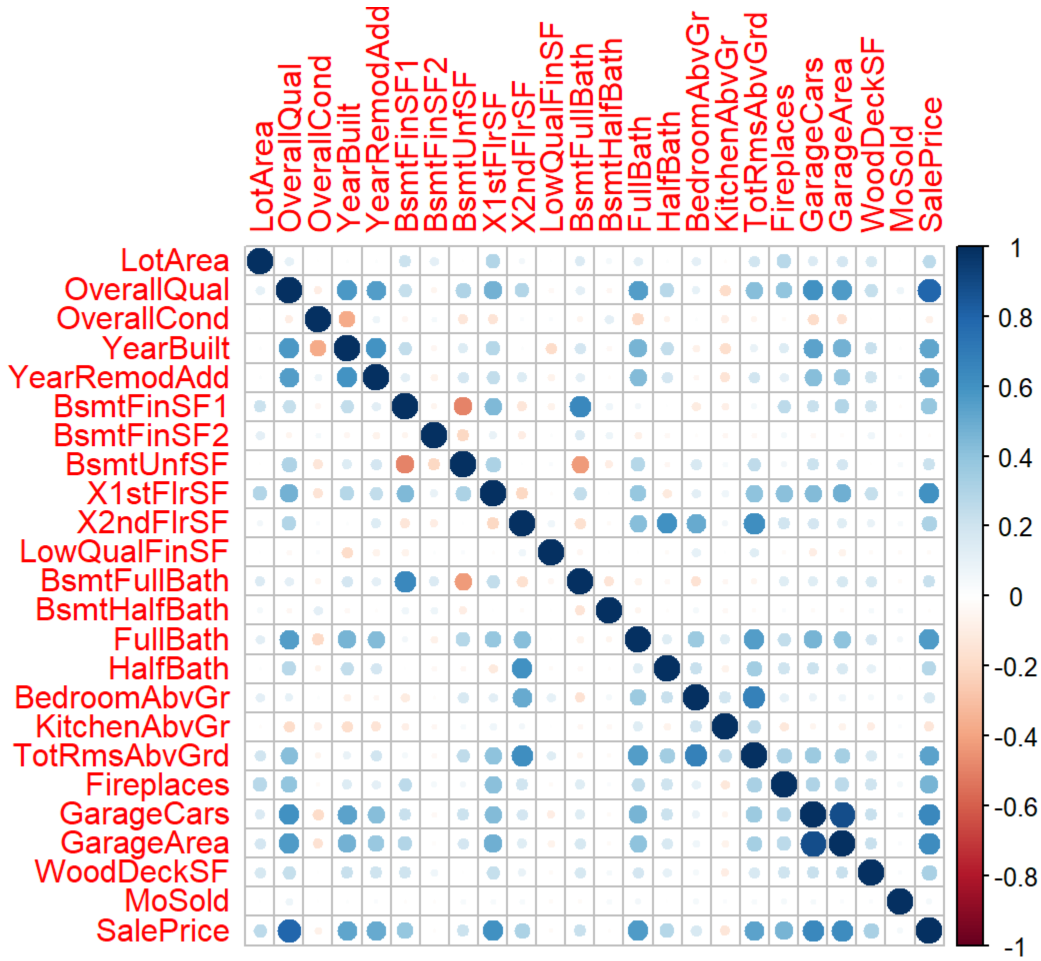
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



It’s no surprise that SalePrice is skewed, less people can afford expensive homes. We will keep this into consideration when applying our models.

Correlation plot

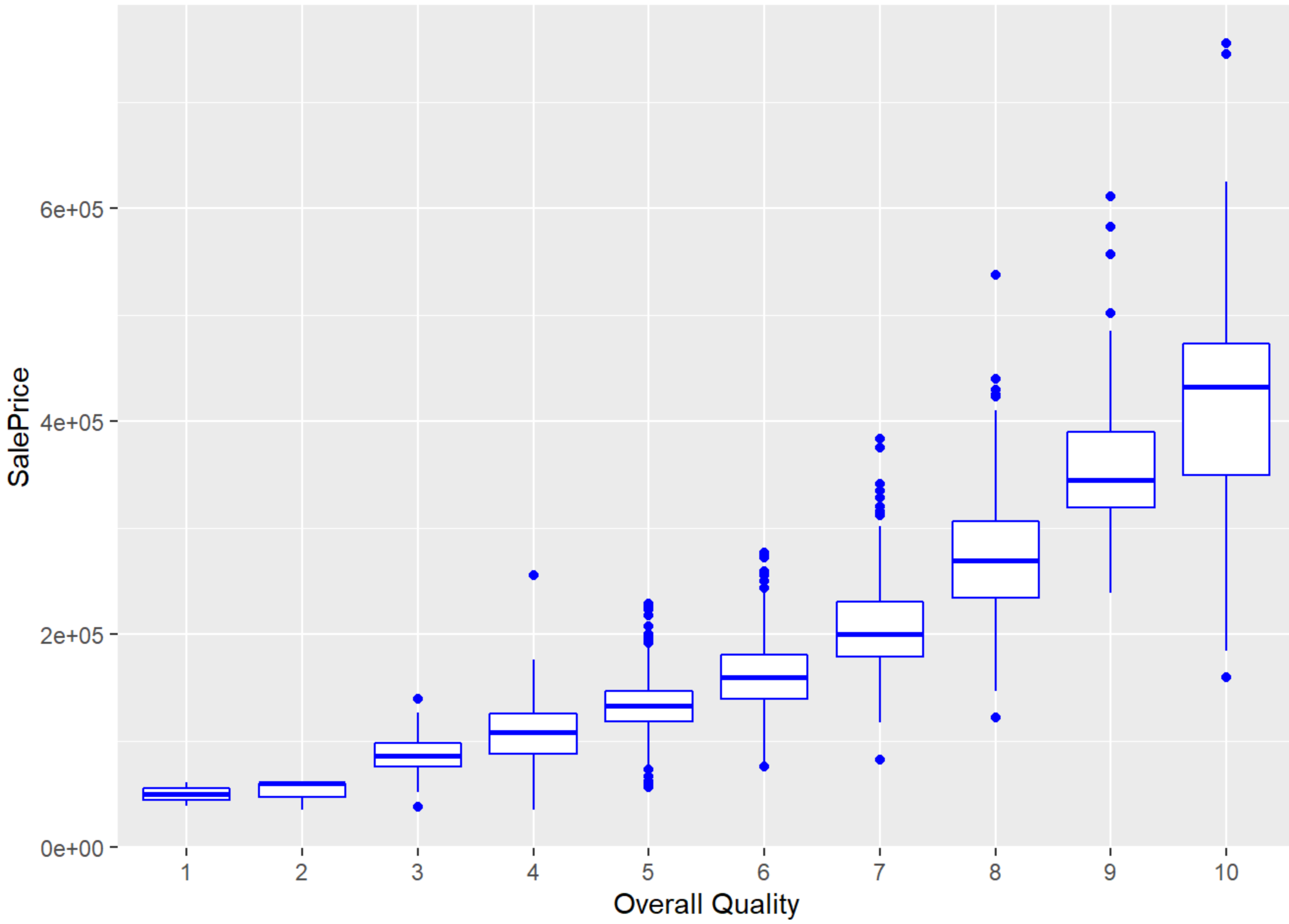
```
corrplot(cor(train))
```



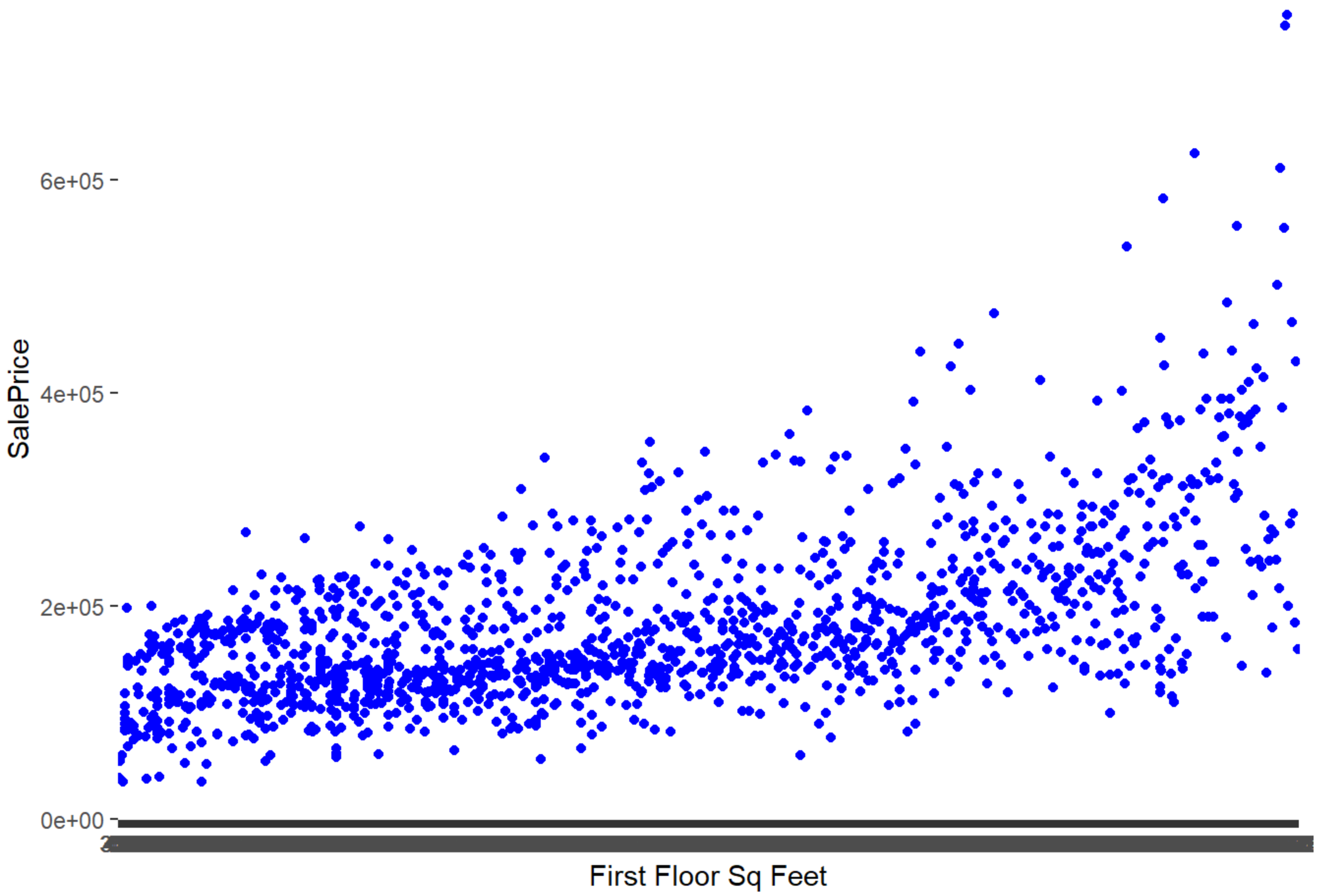
Using all variables from our dataset, we can see how our predictors correalte to one another. We will focus on their interation will SalePrice, that correlate the stronget to SalePrice are; OverallQual, X1stFlrSF(First Florr Square Feet), and GarageArea/GarageCars. We will further investigate what other predictors correlate to SalesPrice

Overall Quality has the highest correlation with SalePrice among the predictor variables. It rates the overall material and finish of the house on a scale from 1 (very poor) to 10 (very excellent).

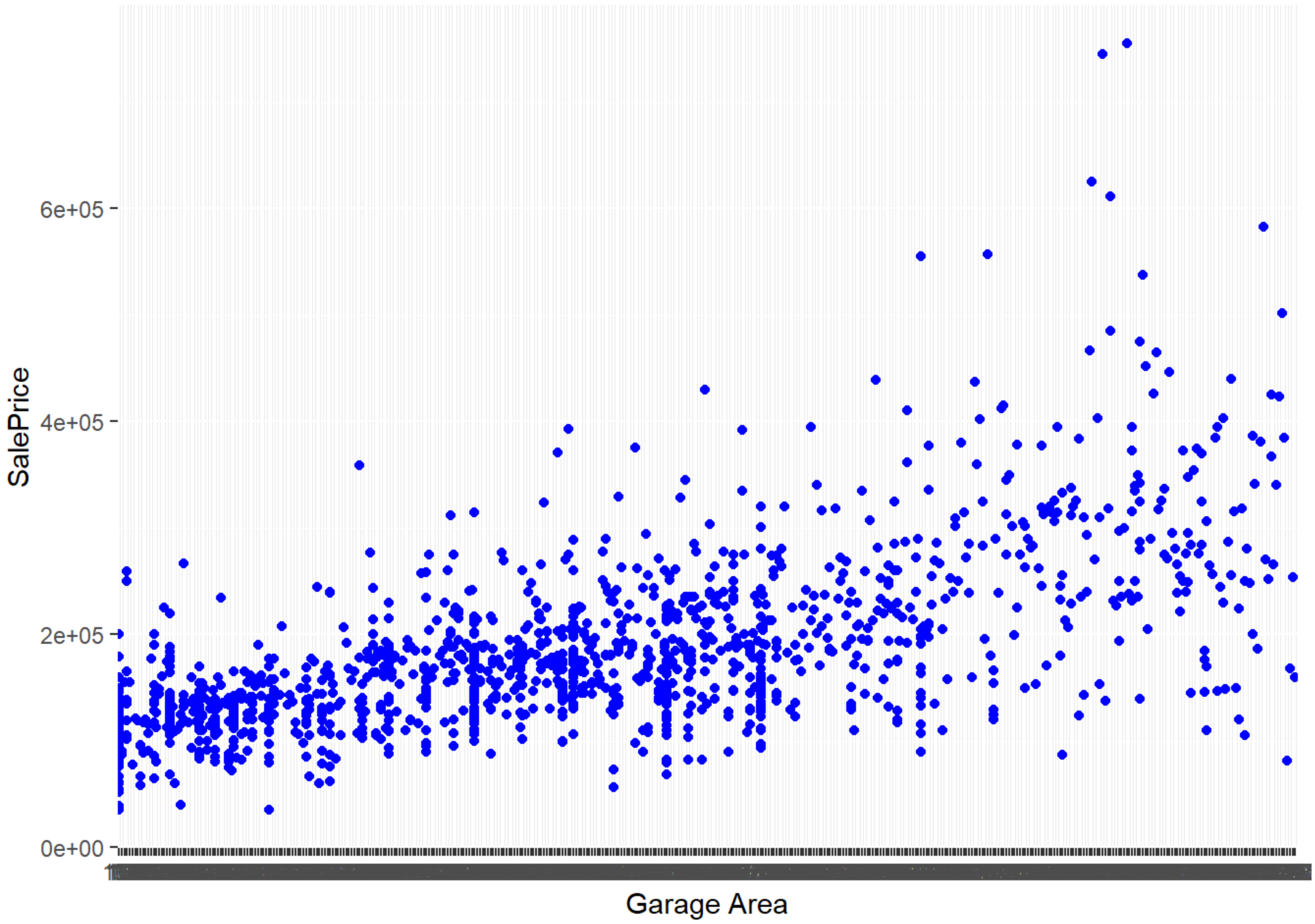
```
ggplot(data= train, aes(x=factor(OverallQual), y=SalePrice))+
        geom_boxplot(col='blue') + labs(x='Overall Quality')
```




```
ggplot(data= train, aes(x=factor(X1stFlrSF), y=SalePrice))+
        geom_point(col='blue') + labs(x='First Floor Sq Feet')
```




```
ggplot(data= train, aes(x=factor(GarageArea), y=SalePrice))+
        geom_point(col='blue') + labs(x='Garage Area')
```



III Code

Linear Regression

The first method we will use to predict SalePrice is linear regression. We will first fit a model with all predictors. Next after checking for outliers, we will fit an updated model to see if our predictions improved.

Fit a linear regression model using all the predictors

```
lm.fit <- lm(SalePrice ~., data = train)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = SalePrice ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -515526  -17128   -2129   13537  290610
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -9.410e+05  1.329e+05  -7.079 2.26e-12 ***
## LotArea       4.416e-01  1.026e-01   4.305 1.79e-05 ***
## OverallQual   1.730e+04  1.199e+03  14.424  < 2e-16 ***
## OverallCond   4.737e+03  1.037e+03   4.569 5.32e-06 ***
## YearBuilt     3.121e+02  5.842e+01   5.342 1.07e-07 ***
## YearRemodAdd  1.292e+02  6.707e+01   1.926 0.054286 .
## BsmtFinSF1    2.319e+01  4.723e+00   4.912 1.01e-06 ***
## BsmtFinSF2    9.958e+00  7.178e+00   1.387 0.165565
## BsmtUnfSF     1.276e+01  4.262e+00   2.994 0.002805 **
## X1stFlrSF     5.171e+01  5.837e+00   8.860  < 2e-16 ***
## X2ndFlrSF     4.313e+01  4.867e+00   8.861  < 2e-16 ***
## LowQualFinSF  1.480e+01  2.005e+01   0.738 0.460532
## BsmtFullBath  7.064e+03  2.648e+03   2.667 0.007732 **
## BsmtHalfBath  1.253e+03  4.174e+03   0.300 0.764104
## FullBath      2.500e+03  2.865e+03   0.873 0.383034
## HalfBath     -5.415e+02  2.709e+03  -0.200 0.841604
## BedroomAbvGr -9.028e+03  1.716e+03  -5.260 1.66e-07 ***
## KitchenAbvGr -2.524e+04  4.955e+03  -5.095 3.96e-07 ***
## TotRmsAbvGrd  6.007e+03  1.252e+03   4.798 1.77e-06 ***
## Fireplaces    4.130e+03  1.794e+03   2.302 0.021482 *
## GarageCars    1.102e+04  2.909e+03   3.787 0.000159 ***
## GarageArea    5.430e+00  9.861e+00   0.551 0.581942
## WoodDeckSF    2.125e+01  8.036e+00   2.644 0.008278 **
## MoSold        5.284e+01  3.480e+02   0.152 0.879328
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 35640 on 1436 degrees of freedom
## Multiple R-squared:  0.8019, Adjusted R-squared:  0.7987
## F-statistic: 252.7 on 23 and 1436 DF,  p-value: < 2.2e-16
```

```
lm_pred <- predict(lm.fit, test)
mse.lm <- mean((lm_pred - test$SalePrice)^2)

mse.lm
```

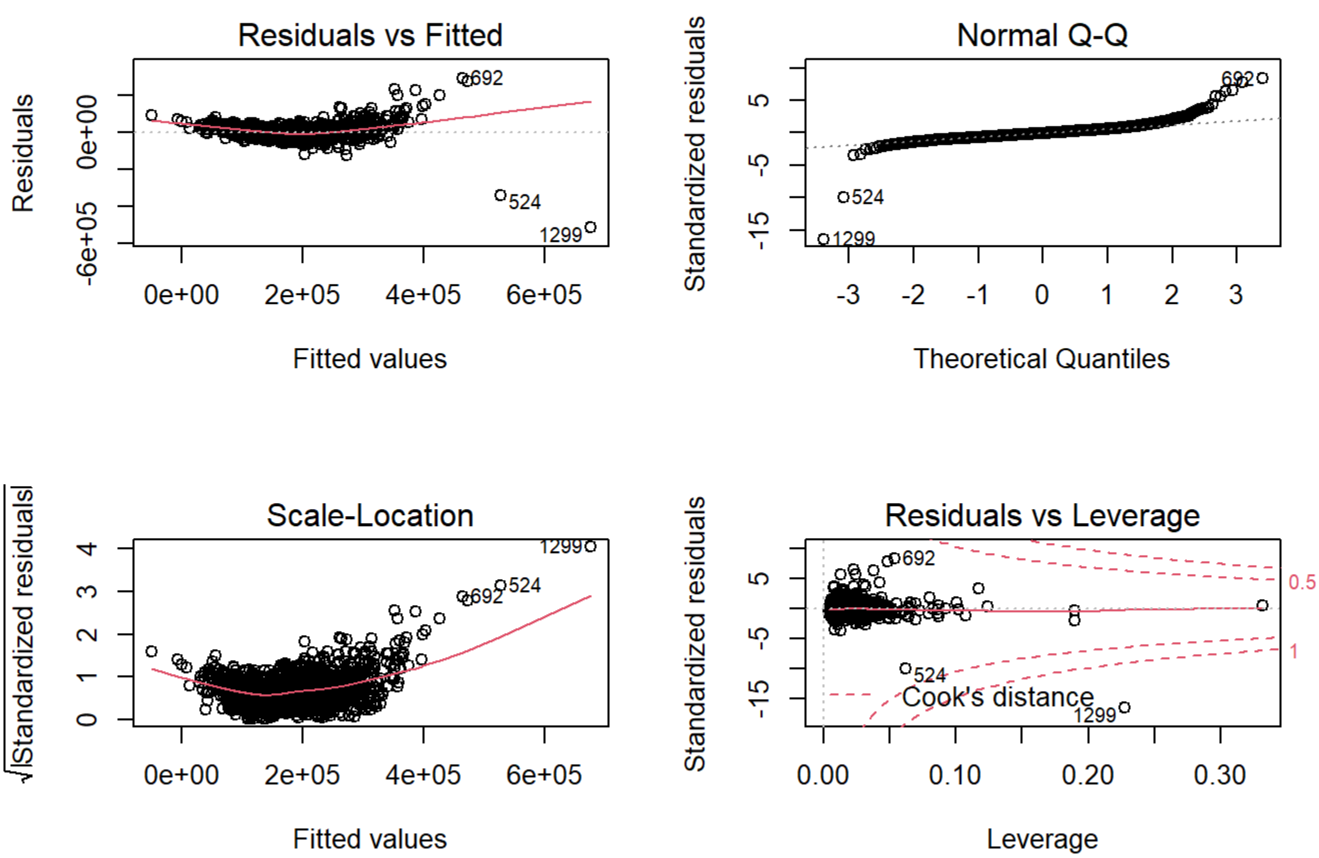
```
## [1] 36626890801
```

From our first Linear Regression model we our R-squared is 0.8019. This tells us our model can explain

80.19% of the variation in SalePrice.

Residual diagnostics

```
par(mfrow=c(2,2))
plot(lm.fit)
```



From the Residuals vs Fitted plot, the constant variance assumption is violated at a few points.524, 1299, 692

From the Normal Q-Q plot, we see our data is heavy tailed. Our revised model will address those data points.

The Scale-Location plot, shows homoskedasticity. We see a clear upward trend.

From the Residual vs Leverage plot, observation 1299 has large Cook's distance (larger than 1), so I will identify this observation as large leverage point.

What are the remedies if the model assumptions are violated? After these remedies, check the residual diagnostics again. Do they look better?

We will drop the outliers and take the log of SalePrice

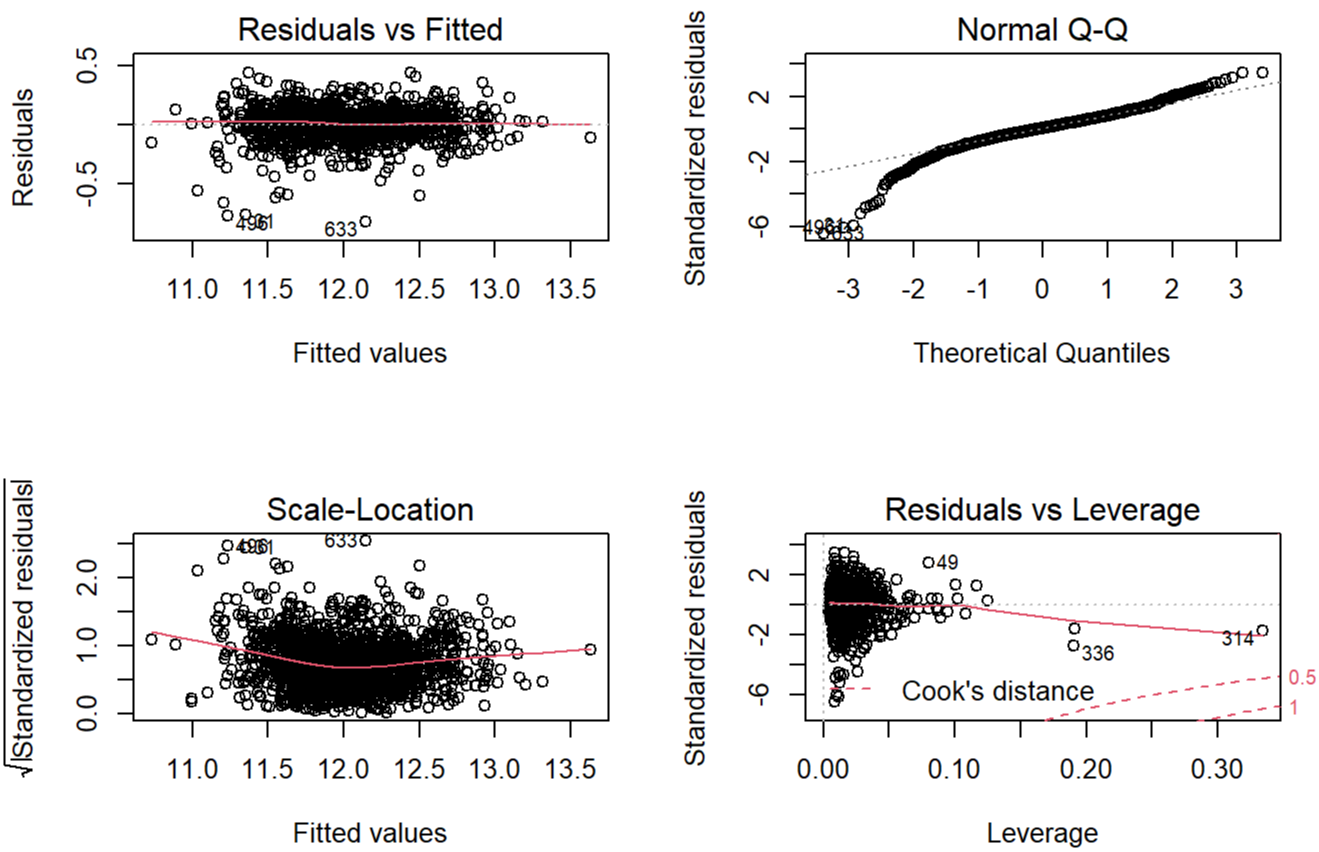
```
lm_fit2 <- lm(log(SalePrice) ~ ., data = train[-c(1299,692 ,524),])
summary(lm_fit2)
```

```
##
## Call:
## lm(formula = log(SalePrice) ~ ., data = train[-c(1299, 692, 524),
##     ])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.82245 -0.06024  0.00650  0.07265  0.43886
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.800e+00  4.779e-01   5.859 5.78e-09 ***
## LotArea       2.543e-06  3.689e-07   6.894 8.10e-12 ***
## OverallQual   7.210e-02  4.330e-03  16.650  < 2e-16 ***
## OverallCond   5.128e-02  3.722e-03  13.778  < 2e-16 ***
## YearBuilt     2.836e-03  2.100e-04  13.503  < 2e-16 ***
## YearRemodAdd  1.103e-03  2.410e-04   4.575 5.16e-06 ***
## BsmtFinSF1    1.965e-04  1.775e-05  11.073  < 2e-16 ***
## BsmtFinSF2    1.364e-04  2.590e-05   5.266 1.61e-07 ***
## BsmtUnfSF     1.214e-04  1.554e-05   7.812 1.08e-14 ***
## X1stFlrSF     2.717e-04  2.137e-05  12.710  < 2e-16 ***
## X2ndFlrSF     2.422e-04  1.828e-05  13.252  < 2e-16 ***
## LowQualFinSF  1.197e-04  7.195e-05   1.664 0.096406 .
## BsmtFullBath  2.409e-02  9.636e-03   2.500 0.012522 *
## BsmtHalfBath -3.096e-03  1.507e-02  -0.205 0.837225
## FullBath      1.324e-02  1.033e-02   1.281 0.200232
## HalfBath      1.802e-02  9.737e-03   1.851 0.064411 .
## BedroomAbvGr -8.842e-03  6.184e-03  -1.430 0.152999
## KitchenAbvGr -9.766e-02  1.779e-02  -5.490 4.74e-08 ***
## TotRmsAbvGrd  1.178e-02  4.522e-03   2.605 0.009276 **
## Fireplaces    3.736e-02  6.476e-03   5.769 9.74e-09 ***
## GarageCars    3.583e-02  1.059e-02   3.382 0.000739 ***
## GarageArea    1.066e-04  3.568e-05   2.987 0.002861 **
## WoodDeckSF    4.390e-05  2.890e-05   1.519 0.129017
## MoSold        8.347e-04  1.253e-03   0.666 0.505508
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1279 on 1433 degrees of freedom
## Multiple R-squared:  0.8983, Adjusted R-squared:  0.8967
## F-statistic: 550.5 on 23 and 1433 DF,  p-value: < 2.2e-16
```

From our updated Linear Regression model, which takes the log(SalePrice) and excludes the outliers, we recieve an R-squared of 0.8983.

This is better than our base model which had an R-squard of 0.8019.

```
par(mfrow=c(2,2))
plot(lm_fit2)
```



Our updated residuals show that removing the outlines mentioned above, improves out diagnostic results.

###Explain some of the most significant coefficients.

From our Linear Regression model, we have 23 significant coefficients. They are what you expect for every unit increase in OverallQua, SalePrice increases by 7.210e-02. For every unit increase in X1stFlrSF, Sale Price increases by 2.717e-04
Perform prediction.

```
lm_pred <- predict(lm_fit2, test)
mse.lm <- mean((lm_pred - test$SalePrice)^2)

mse.lm
```

```
## [1] 144.4138
```

Subset Selection

We used forward stepwise for our subset selection method.

Cp = 18 , and Adj R2= 21 the minimum size for the subset. BIC has a size = 12. We pick 12 as the best subset size based on BIC criterion.

Some of the coefficients in this model include:

GarageCars BsmtFullBath

Perform prediction.

We will take the log(SalePrice)

```
library(leaps)


step_fwd <- regsubsets(log(SalePrice) ~., data = train,
nvmax = 25, method = "forward")


gam_summary = summary(step_fwd)
min.cp = which.min(gam_summary$cp)

print(min.cp)
```

```
## [1] 18
```

```
min.bic = which.min(gam_summary$bic)
print(min.bic)
```

```
## [1] 12
```

```
max.adjr2 = which.max(gam_summary$adjr2)
print(max.adjr2)
```

```
## [1] 20
```

```
coefi = coef(step_fwd, id = min.bic)
coefi
```

```
##   (Intercept)       LotArea   OverallQual   OverallCond     YearBuilt
##  2.538272e+00  2.254784e-06  8.679639e-02  5.050417e-02  3.037263e-03
##  YearRemodAdd     X1stFlrSF     X2ndFlrSF  BsmtFullBath  KitchenAbvGr
##  1.038945e-03  2.791243e-04  1.779746e-04  6.961001e-02 -1.100127e-01
##  TotRmsAbvGrd    Fireplaces    GarageCars
##  2.002863e-02  4.845773e-02  7.960498e-02
```

Our coefficients shown above from our BIC forward selection have OverallQual as one of the most important features

Generalized Additive Models

```r
library(gam)
```

```
## Warning: package 'gam' was built under R version 4.1.3
```

```
## Loading required package: splines
```

```
## Loading required package: foreach
```

```
## Warning: package 'foreach' was built under R version 4.1.3
```

```
## Loaded gam 1.20.1
```

```r
gam.fit = gam(SalePrice ~ .,data = train)

gam.fit
```

```
## Call:
## gam(formula = SalePrice ~ ., data = train)
##
## Degrees of Freedom: 1459 total; 1436 Residual
## Residual Deviance: 1.824253e+12
```

Degrees of Freedom: 1459 Residual Deviance: 1.822808e+12

```r
gam.pred <-predict(gam.fit, test)
mse.gam <- mean((gam.pred - test$SalePrice)^2)
mse.gam
```

```
## [1] 36626890801
```

We obtain a test mean squared error of 36635169466 using GAM with 11 predictors.

```r
summary(gam.fit)
```

```
##
## Call: gam(formula = SalePrice ~ ., data = train)
## Deviance Residuals:
##     Min      1Q  Median      3Q     Max
## -515526  -17128   -2129   13537  290610
##
## (Dispersion Parameter for gaussian family taken to be 1270371112)
##
##     Null Deviance: 9.207911e+12 on 1459 degrees of freedom
## Residual Deviance: 1.824253e+12 on 1436 degrees of freedom
## AIC: 34774.46
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                Df     Sum Sq    Mean Sq   F value    Pr(>F)
## LotArea         1 6.4099e+11 6.4099e+11  504.5716 < 2.2e-16 ***
## OverallQual     1 5.4222e+12 5.4222e+12 4268.1865 < 2.2e-16 ***
## OverallCond     1 3.2156e+08 3.2156e+08    0.2531  0.614963
## YearBuilt       1 9.5942e+10 9.5942e+10   75.5231 < 2.2e-16 ***
## YearRemodAdd    1 2.5679e+10 2.5679e+10   20.2135 7.483e-06 ***
## BsmtFinSF1      1 2.3956e+11 2.3956e+11  188.5775 < 2.2e-16 ***
## BsmtFinSF2      1 7.0028e+09 7.0028e+09    5.5124  0.019017 *
## BsmtUnfSF       1 1.2293e+11 1.2293e+11   96.7635 < 2.2e-16 ***
## X1stFlrSF       1 1.4690e+11 1.4690e+11  115.6318 < 2.2e-16 ***
## X2ndFlrSF       1 4.9833e+11 4.9833e+11  392.2717 < 2.2e-16 ***
## LowQualFinSF    1 1.2005e+09 1.2005e+09    0.9450  0.331168
## BsmtFullBath    1 1.0759e+10 1.0759e+10    8.4691  0.003668 **
## BsmtHalfBath    1 5.2072e+07 5.2072e+07    0.0410  0.839587
## FullBath        1 3.1790e+08 3.1790e+08    0.2502  0.616981
## HalfBath        1 1.6185e+08 1.6185e+08    0.1274  0.721192
## BedroomAbvGr    1 2.5737e+10 2.5737e+10   20.2592 7.309e-06 ***
## KitchenAbvGr    1 2.4890e+10 2.4890e+10   19.5929 1.031e-05 ***
## TotRmsAbvGrd    1 3.8104e+10 3.8104e+10   29.9941 5.110e-08 ***
## Fireplaces      1 7.6938e+09 7.6938e+09    6.0564  0.013973 *
## GarageCars      1 6.5608e+10 6.5608e+10   51.6445 1.065e-12 ***
## GarageArea      1 3.7280e+08 3.7280e+08    0.2935  0.588095
## WoodDeckSF      1 8.8987e+09 8.8987e+09    7.0048  0.008218 **
## MoSold          1 2.9292e+07 2.9292e+07    0.0231  0.879328
## Residuals    1436 1.8243e+12 1.2704e+09
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Our gam models returned 12 significant variables that reinforce previous models.

The significant features include; LotArea, OverallQual, YearBuilt, X1stFlrSF, KitchenAbvGr, and GarageCars

6. Regression Trees

We fit a tree to the training data, with SalePrice as the response and the other variables as predictors. By summary() function, it can be seen that the training MSE is .481e+09 and the tree has 12 terminal nodes.

```r
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.1.3
```

```r
house_tree <- tree(SalePrice ~ ., data = train)

summary(house_tree)
```

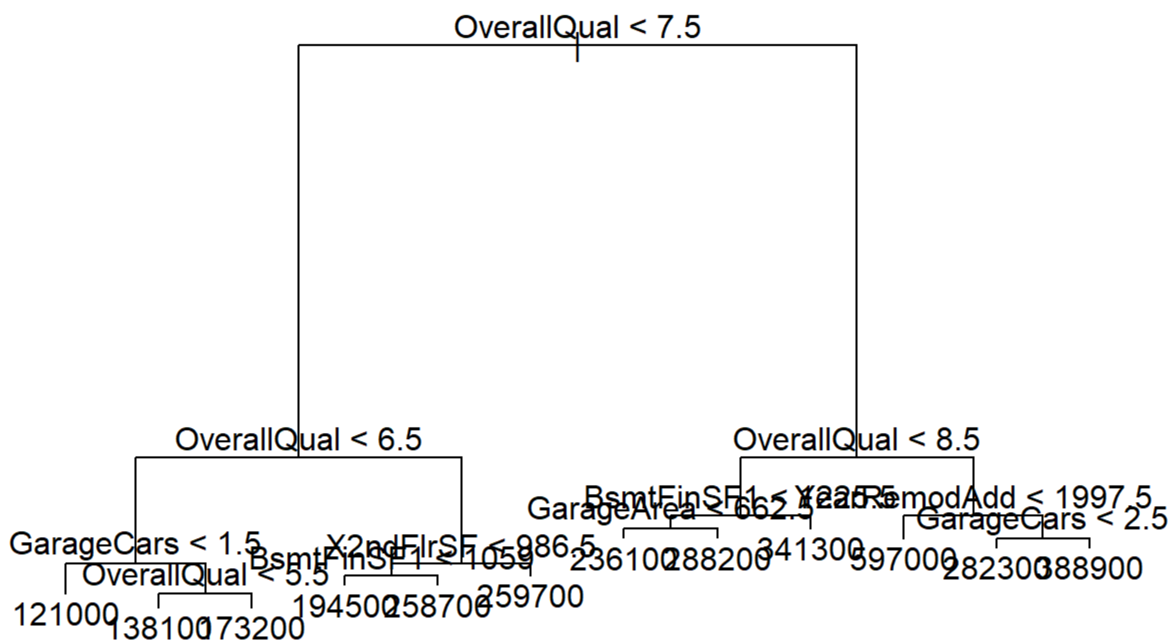
```
##
## Regression tree:
## tree(formula = SalePrice ~ ., data = train)
## Variables actually used in tree construction:
## [1] "OverallQual"  "GarageCars"   "X2ndFlrSF"    "BsmtFinSF1"   "GarageArea"
## [6] "YearRemodAdd"
## Number of terminal nodes:  12
## Residual mean deviance:  1.481e+09 = 2.145e+12 / 1448
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -212000  -22210   -1040       0   18940  222800
```

```r
house_tree
```

```
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 1460 9.208e+12 180900
##    2) OverallQual < 7.5 1231 2.988e+12 157800
##      4) OverallQual < 6.5 912 1.287e+12 140400
##        8) GarageCars < 1.5 427 3.672e+11 121000 *
##        9) GarageCars > 1.5 485 6.196e+11 157400
##         18) OverallQual < 5.5 218 1.975e+11 138100 *
##         19) OverallQual > 5.5 267 2.739e+11 173200 *
##      5) OverallQual > 6.5 319 6.288e+11 207700
##       10) X2ndFlrSF < 986.5 279 4.335e+11 200300
##         20) BsmtFinSF1 < 1059 254 2.899e+11 194500 *
##         21) BsmtFinSF1 > 1059 25 4.972e+10 258700 *
##       11) X2ndFlrSF > 986.5 40 7.169e+10 259700 *
##    3) OverallQual > 7.5 229 2.037e+12 305000
##      6) OverallQual < 8.5 168 6.819e+11 274700
##       12) BsmtFinSF1 < 1225.5 142 4.290e+11 262500
##         24) GarageArea < 662.5 70 1.562e+11 236100 *
##         25) GarageArea > 662.5 72 1.765e+11 288200 *
##       13) BsmtFinSF1 > 1225.5 26 1.167e+11 341300 *
##      7) OverallQual > 8.5 61 7.756e+11 388500
##       14) YearRemodAdd < 1997.5 5 1.075e+11 597000 *
##       15) YearRemodAdd > 1997.5 56 4.313e+11 369900
##         30) GarageCars < 2.5 10 2.596e+10 282300 *
##         31) GarageCars > 2.5 46 3.121e+11 388900 *
```

Tree interpretation: A house with OverallQual greater than 7.5 with a GarageArea greater than 662.5 has a price of $288200.

```
par(mfrow=c(1,1))
plot(house_tree)
text(house_tree, pretty = 0)
```



Finding MSE

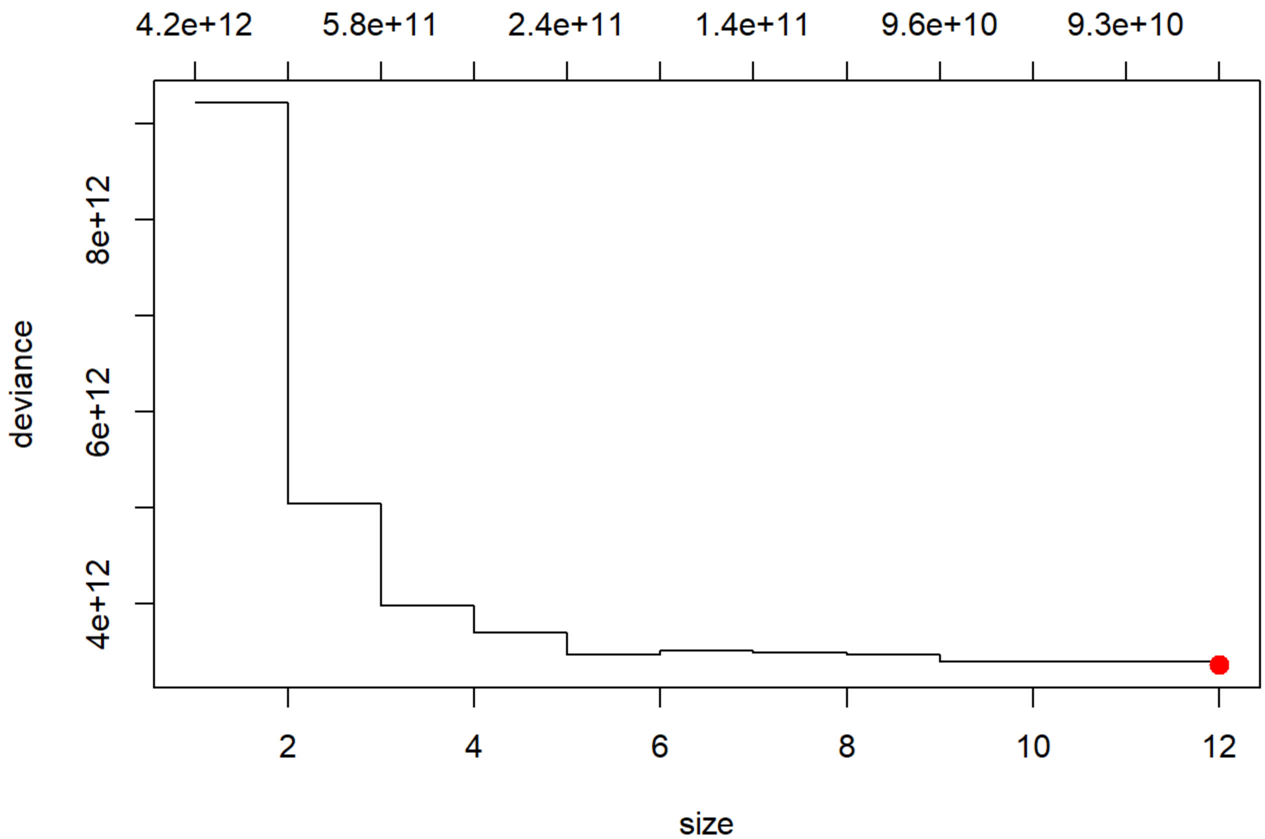
```
pred.tree <- predict(house_tree, newdata = test)
mse.tree <- mean((pred.tree - test$SalePrice)^2)
mse.tree
```

```
## [1] 37959706613
```

Our MSE is 37959706613

```
tree.cv.house <- cv.tree(house_tree)

par(mfrow = c(1,1))
plot(tree.cv.house)
points(tree.cv.house$size[which.min(tree.cv.house$dev)],
tree.cv.house$dev[which.min(tree.cv.house$dev)],
col = "red", cex = 2, pch = 20)
```



A tree that's size 12 corresponds to the lowest cv error

Bagging

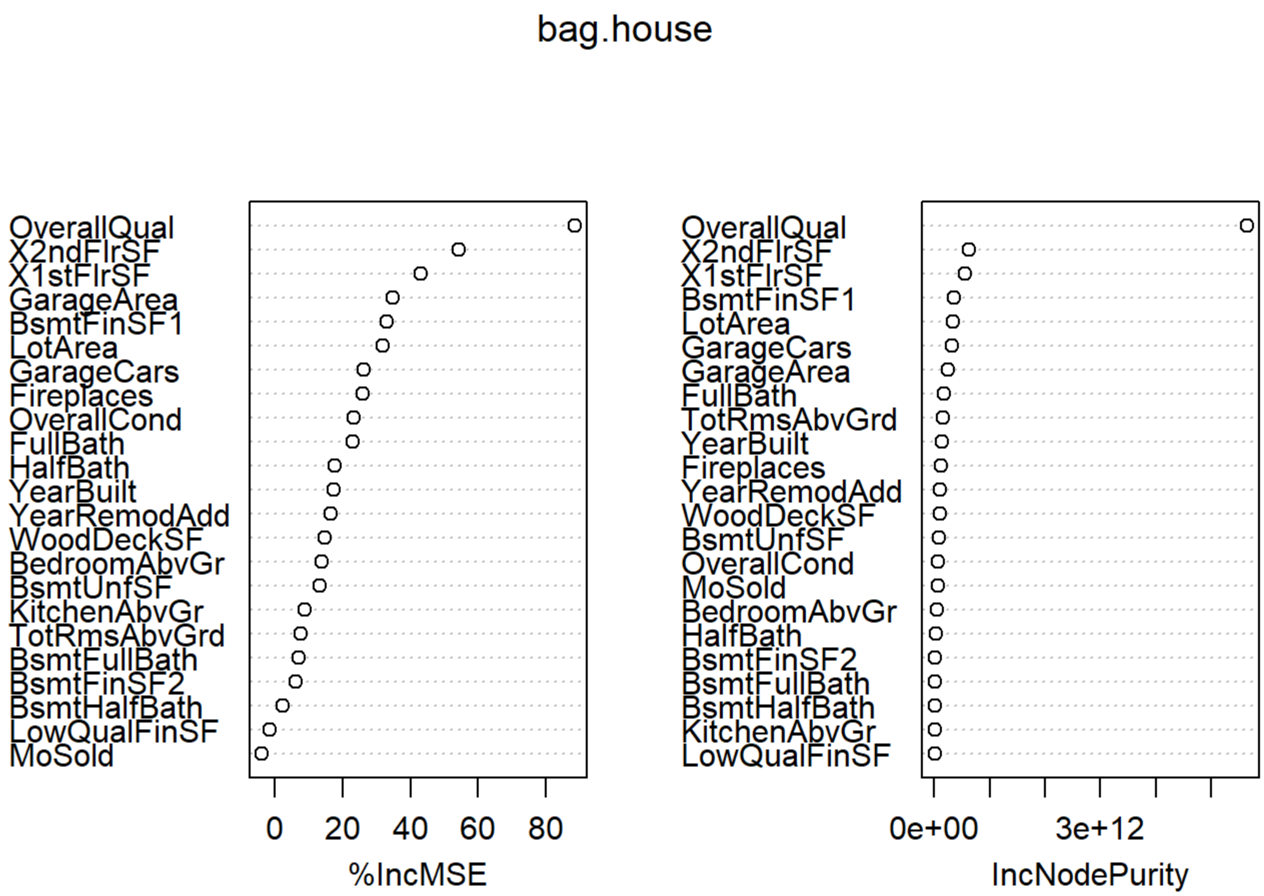
We fit a bagging model on the training data. mtry = ncol(Credit.train) -1 because p =ncol(train) - 1. Recall that bagging is a special case of random forest when m = p. From the importance() function, Rating and Limit appear to be the most important two variables.

```
library(randomForest)

bag.house <- randomForest(SalePrice ~ ., data = train,
mtry = ncol(train) - 1, importance = TRUE,
ntree = 1000)
importance(bag.house)
```

```
##                %IncMSE IncNodePurity
## LotArea      31.771063  3.279241e+11
## OverallQual  88.447511  5.639738e+12
## OverallCond  23.218954  6.352016e+10
## YearBuilt    17.426156  1.293172e+11
## YearRemodAdd 16.362414  1.061650e+11
## BsmtFinSF1   32.970652  3.516388e+11
## BsmtFinSF2    6.018203  1.374996e+10
## BsmtUnfSF    13.092418  8.618252e+10
## X1stFlrSF    43.206688  5.487051e+11
## X2ndFlrSF    54.156330  6.211956e+11
## LowQualFinSF -1.695565  2.376045e+09
## BsmtFullBath  7.123440  1.300849e+10
## BsmtHalfBath  2.292952  5.702879e+09
## FullBath     23.029012  1.770276e+11
## HalfBath     17.773517  2.801669e+10
## BedroomAbvGr 13.823298  3.855767e+10
## KitchenAbvGr  8.755938  5.136953e+09
## TotRmsAbvGrd  7.710680  1.475592e+11
## Fireplaces   26.002486  1.136175e+11
## GarageCars   26.326390  3.097098e+11
## GarageArea   34.680673  2.470347e+11
## WoodDeckSF   14.705395  9.629050e+10
## MoSold       -3.805263  5.957640e+10
```

```
varImpPlot(bag.house)
```




```
pred.bag <- predict(bag.house, newdata = test)
mse.bag <- mean((pred.bag - test$SalePrice)^2)
mse.bag
```

```
## [1] 37081221701
```

We recieve a MSE of 37070454538 using bagging

Random Forest

```
rf.house <- randomForest((SalePrice) ~ ., data =train,
mtry = round(sqrt(ncol(train) - 1)),
importance = TRUE, ntree = 1000)
importance(rf.house)
```

```
##                %IncMSE IncNodePurity
## LotArea      24.9051785  3.824537e+11
## OverallQual  41.4636921  2.025393e+12
## OverallCond  24.2696534  7.811013e+10
## YearBuilt    29.4297877  8.303140e+11
## YearRemodAdd 23.9860436  3.722256e+11
## BsmtFinSF1   18.8599562  4.859618e+11
## BsmtFinSF2    4.5334835  2.198045e+10
## BsmtUnfSF    14.2599512  1.456342e+11
## X1stFlrSF    33.7416085  8.605774e+11
## X2ndFlrSF    34.4846200  4.606601e+11
## LowQualFinSF -1.3754828  6.379194e+09
## BsmtFullBath  8.7831663  5.577174e+10
## BsmtHalfBath  4.6369337  1.516918e+10
## FullBath     21.6736548  4.811725e+11
## HalfBath     18.2771493  7.161422e+10
## BedroomAbvGr 19.0018624  9.446186e+10
## KitchenAbvGr 10.4252917  2.376751e+10
## TotRmsAbvGrd 19.3521896  3.315452e+11
## Fireplaces   26.0267076  3.060430e+11
## GarageCars   25.4899495  1.008626e+12
## GarageArea   28.0442096  8.172260e+11
## WoodDeckSF   10.4785285  1.214602e+11
## MoSold       -0.2827311  7.732449e+10
```

In our random forest model we used 1000 trees From our random forest model we see the important features

We will now find the MSE for Random Forest

```
pred.rf <- predict(rf.house, newdata = test)
mse.rf <- mean((pred.rf - test$SalePrice)^2)
mse.rf
```

```
## [1] 37058051513
```

Our MSE is 37064259988

Boosting

We fit a boosting model on the training data. The most two important variables appear to be Limitand Rating.

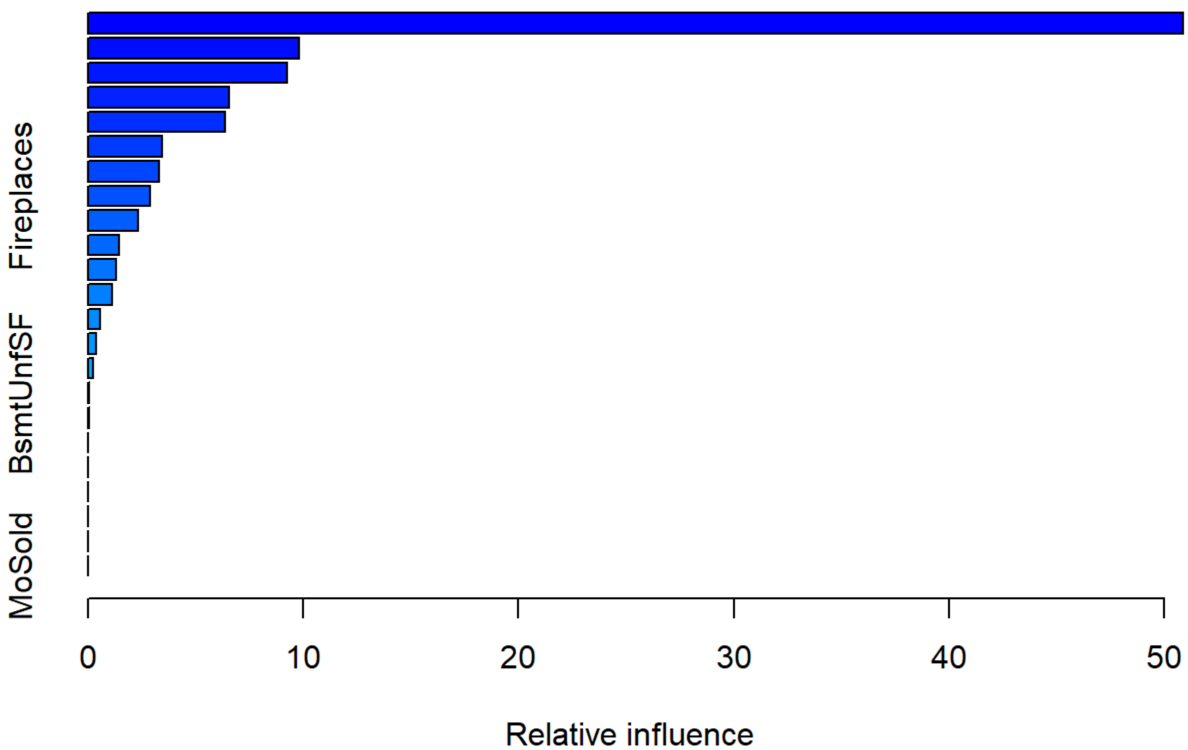
We will use 1000 trees again, with a shrinkage rate of 0.01 A shrinkage parameter applied to each tree in the expansion. Also known as the learning rate or step-size reduction

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 4.1.3
```

```
## Loaded gbm 2.1.8
```

```
boost.house <- gbm(SalePrice ~ . , data = train,
distribution = "gaussian", n.trees = 1000, shrinkage = 0.01)
summary(boost.house)
```



```
##                       var     rel.inf
## OverallQual   OverallQual 50.86001002
## X1stFlrSF       X1stFlrSF  9.79770858
## GarageCars     GarageCars  9.25625832
## BsmtFinSF1     BsmtFinSF1  6.56707639
## X2ndFlrSF       X2ndFlrSF  6.38879943
## LotArea           LotArea  3.42164275
## FullBath         FullBath  3.29733183
## Fireplaces     Fireplaces  2.86347184
## YearBuilt       YearBuilt  2.33478200
## YearRemodAdd YearRemodAdd  1.46770336
## GarageArea     GarageArea  1.30439049
## TotRmsAbvGrd TotRmsAbvGrd  1.12552461
## HalfBath         HalfBath  0.58257783
## OverallCond   OverallCond  0.37945814
## WoodDeckSF     WoodDeckSF  0.21738575
## BsmtUnfSF       BsmtUnfSF  0.06777809
## KitchenAbvGr KitchenAbvGr  0.04796047
## BsmtFullBath BsmtFullBath  0.02014009
## BsmtFinSF2     BsmtFinSF2  0.00000000
## LowQualFinSF LowQualFinSF  0.00000000
## BsmtHalfBath BsmtHalfBath  0.00000000
## BedroomAbvGr BedroomAbvGr  0.00000000
## MoSold             MoSold  0.00000000
```

Finally we focus on Boosting. OverallQual with a score of 51.36,had the highest relative influence on SalePrice. Followed by X1stFlrSF with a score of 10.17628098

```
yhat.boost <- predict(boost.house, newdata = test,
n.trees = 1000)

mse.boost <- mean((yhat.boost - test$SalePrice)^2)
mse.boost
```

```
## [1] 37314135235
```

We received a test MSE of 37415848858

We can also perform 10-fold cross-validation to select the best tree number. The best tree number is 1000 which is the same number as above. The test MSE is 24834.6. The reason the two MSEs are different is because these 1000 trees for both boosting models are different, which are chosen randomly.

```
boost.cv.house <- gbm(SalePrice ~ . , data = train,
distribution = "gaussian", n.trees = 1000, shrinkage = 0.01, cv.folds = 10)

best.ntrees <- which.min(boost.cv.house$cv.error)
best.ntrees
```

```
## [1] 1000
```

We got 995 trees as our most optimal tree number

We will test the MSE again using the best tree number.

```
yhat.boost <- predict(boost.cv.house, newdata = test,
n.trees = best.ntrees)
mse.boost.cv <- mean((yhat.boost - test$SalePrice)^2)
mse.boost.cv
```

```
## [1] 37274950131
```

Our MSE has slightly improved after using 10-fold cv. Our new MSE is 37298381468.

Comparing all methods

Out of all methods applied to this project, Linear regression performed the best, and tree performed the worst

```
misclass.all <- c(mse.tree,mse.lm ,mse.bag, mse.rf,
mse.boost, mse.boost.cv, mse.gam)
names(misclass.all) <- c("tree","lm" , "bagging",
"random forest", "boosting", "boosting.cv", "logistic")
misclass.all
```

```
##          tree            lm       bagging random forest      boosting
##  3.795971e+10  1.444138e+02  3.708122e+10  3.705805e+10  3.731414e+10
##   boosting.cv      logistic
##  3.727495e+10  3.662689e+10
```

Conclusion and summary

When predicting house prices, the most obvious features like square footage, overall quality, GarageArea and year built, tend to impact prices greatly. From the results of the project, we verified that’s true.

I am interested in exploring this data set, or one similar, to predict housing prices with mortgage data and interest rates. For example, does mortgage rates change the importance of certain house features. What is the most optimal interest rate for buyers/sellers. I also am interested in seeing what variables determine rent.