521 - Final Project

Team Potato

April 28, 2017

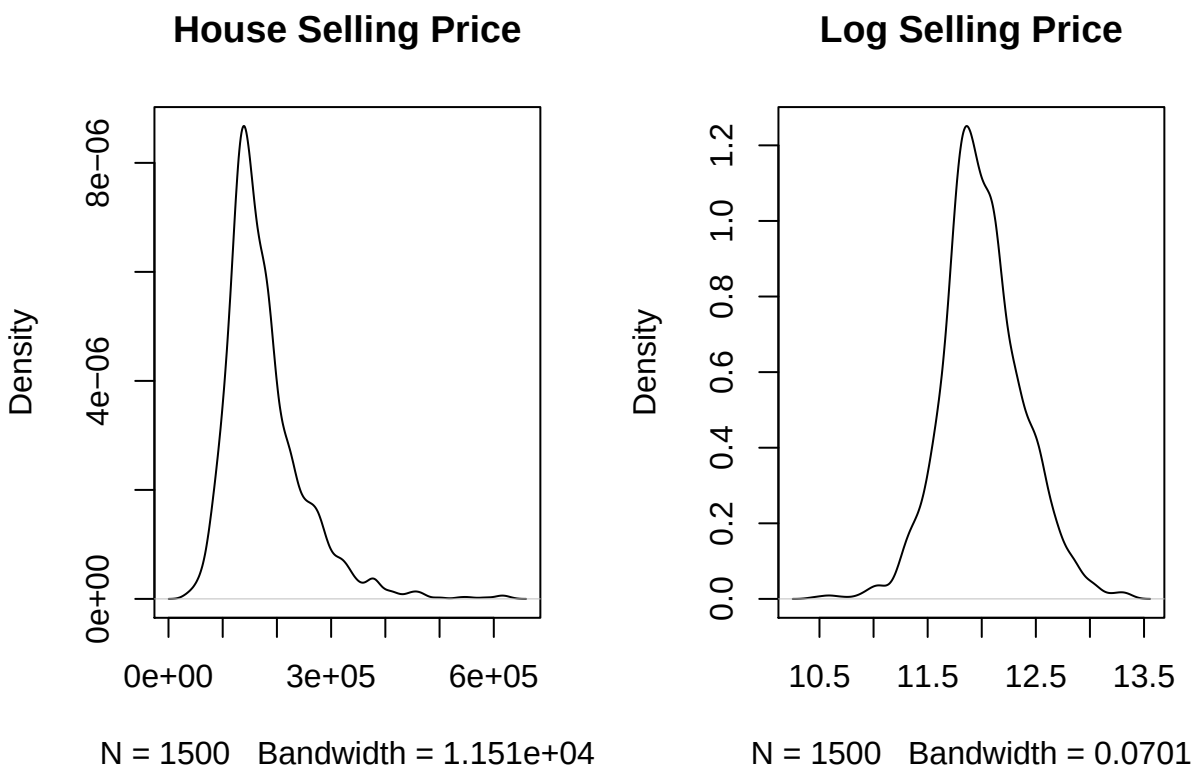
1. Introduction

We are given a dataset of house prices from Ames, Iowa. We are tasked with indentifying house prices that are over or undervalued. We approached this problem by trying various models in an attempt to predict house prices accuractly, quantify varience, and identify top features that might impact prices.

2. Exploratory Data Analysis

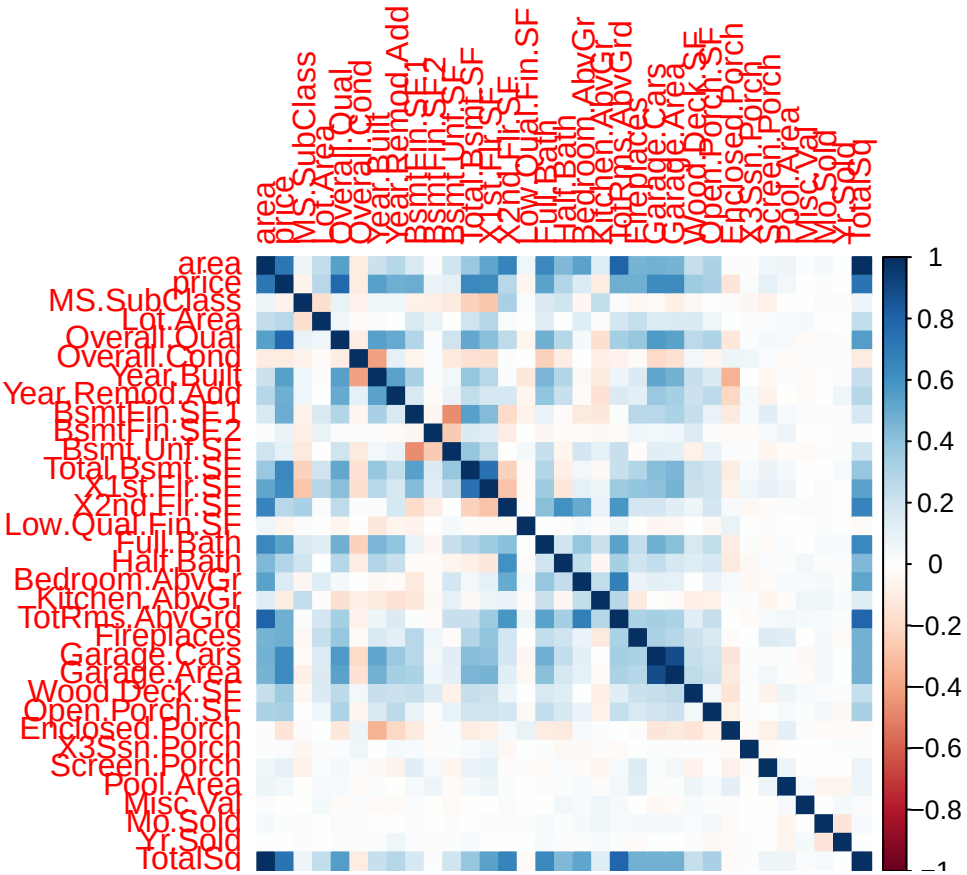
Skew of Response

We are interested in predicting house price given a slew of of other variables. First we investigated the response variable and found that it was signifcantly right-skewed. A log transform would center the data better.



Correlation

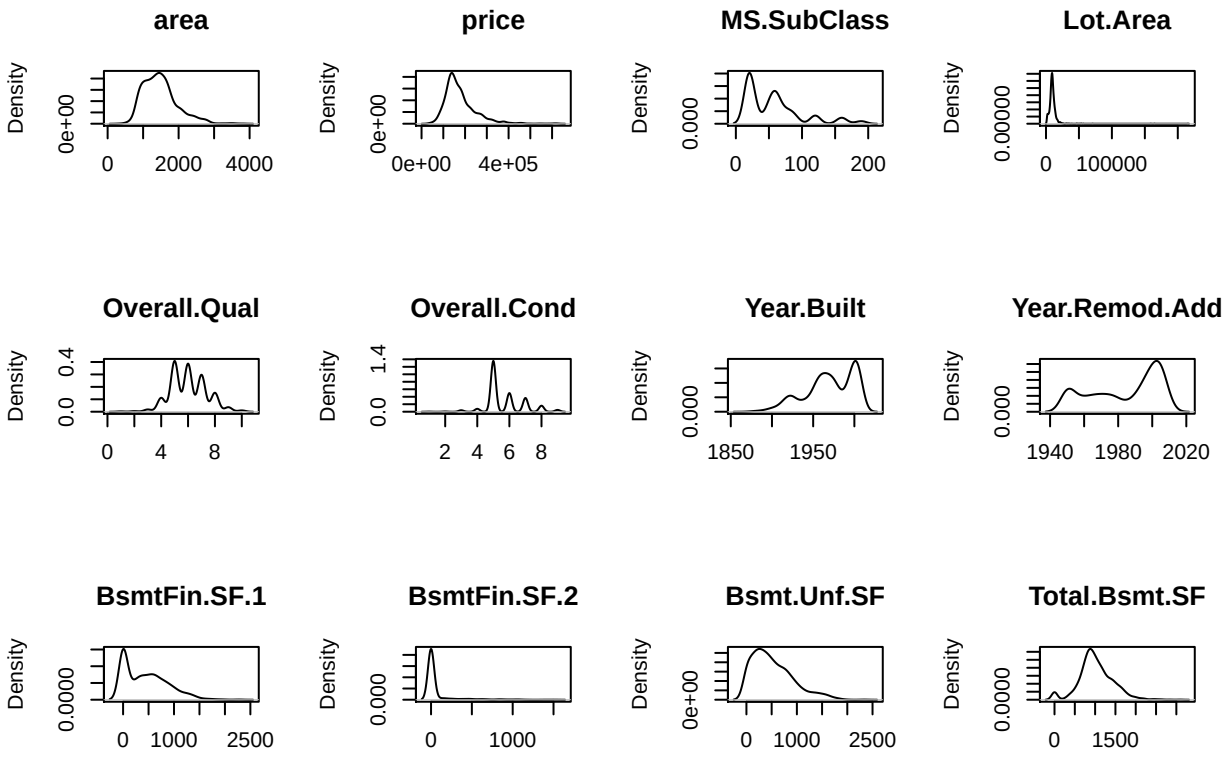
Next we looked at the pairwise correlation between all the numeric features (without `PID` and without missing values).

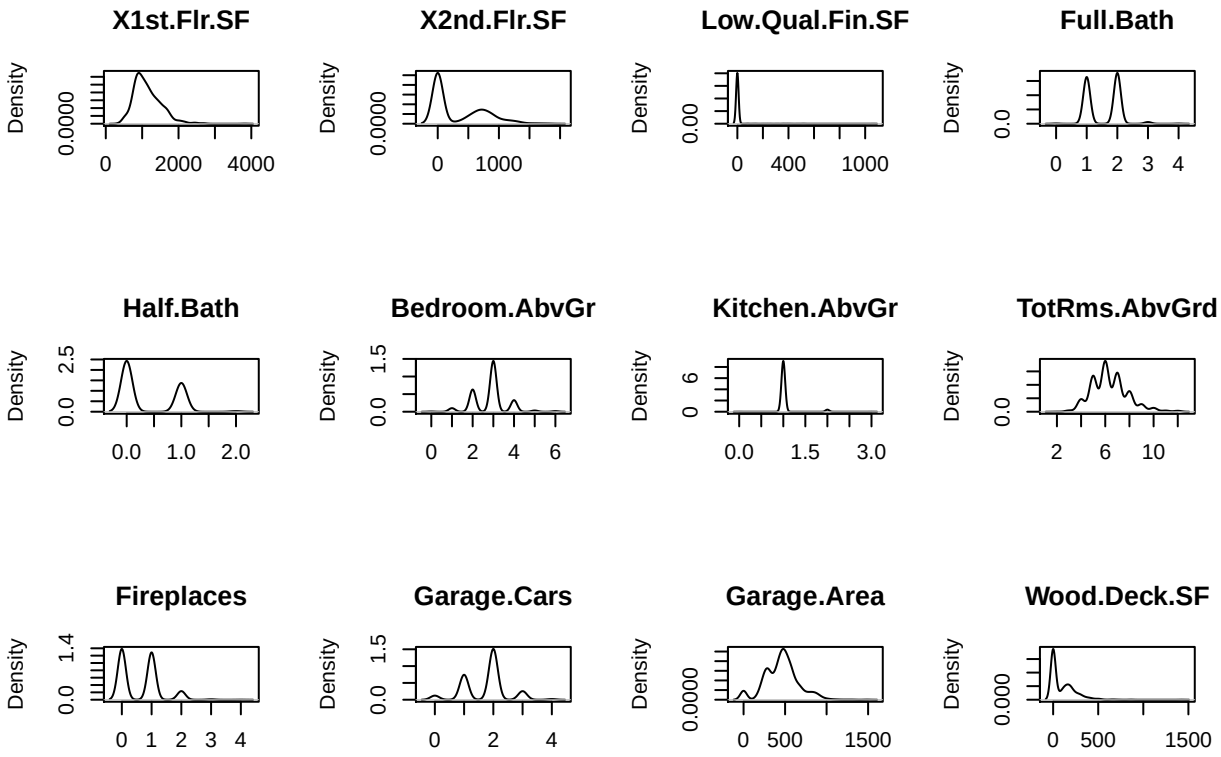


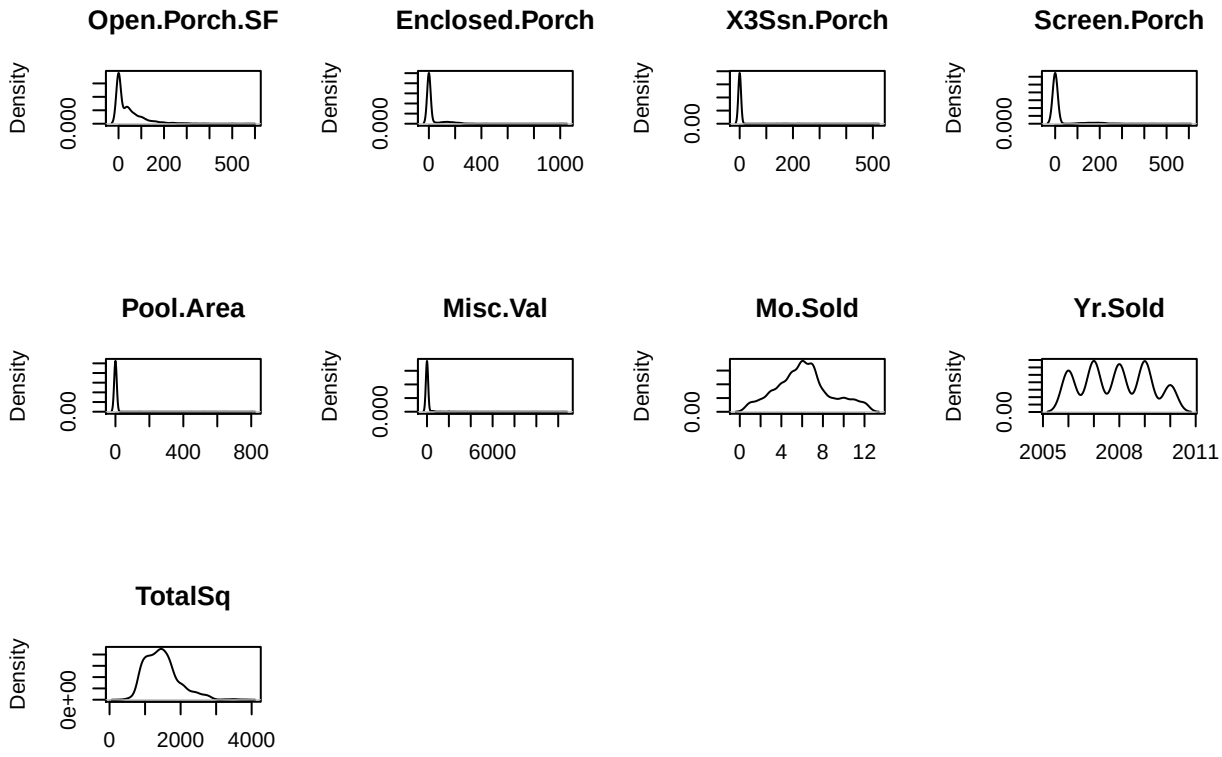
We see that there are 80 numeric variables and several are strongly correlated with price: `area`, `Overall.Qual`, and `Total.SQ`. Some variables are highly correlated with each other like `Garage.Cars` and `Garage.Area`, which makes sense. This gives us an idea of which variables might be important.

Numeric Variables

Now we investigate the numeric data.


area
Density
0e+00
0
2000
4000
price
Density
0e+00
0e+00
4e+05
MS.SubClass
Density
0.000
0
100
200
Lot.Area
Density
0.00000
0
100000
Overall.Qual
Density
0.0
0.4
0
4
8
Overall.Cond
Density
0.0
1.4
2
4
6
8
Year.Built
Density
0.000
1850
1950
Year.Remod.Add
Density
0.000
1940
1980
2020
BsmtFin.SF.1
Density
0.0000
0
1000
2500
BsmtFin.SF.2
Density
0.000
0
1000
Bsmt.Unf.SF
Density
0e+00
0
1000
2500
Total.Bsmt.SF
Density
0.0000
0
1500


X1st.Flr.SF
Density
0.0000
0
2000
4000
X2nd.Flr.SF
Density
0.0000
0
1000
Low.Qual.Fin.SF
Density
0.00
0
400
1000
Full.Bath
Density
0.0
0
1
2
3
4
Half.Bath
Density
0.0
2.5
0.0
1.0
2.0
Bedroom.AbvGr
Density
0.0
1.5
0
2
4
6
Kitchen.AbvGr
Density
0
6
0.0
1.5
3.0
TotRms.AbvGrd
Density
0.0
2
6
10
Fireplaces
Density
0.0
1.4
0
1
2
3
4
Garage.Cars
Density
0.0
1.5
0
2
4
Garage.Area
Density
0.0000
0
500
1500
Wood.Deck.SF
Density
0.000
0
500
1500



Some of the numeric factors are very skewed. Some have patterns that make it look like the it has bins (is a factor in disguise). Some the variables are like `Kitchen.Qual` are very bimodal.

The exploratory data analysis tells us that variable selection and transformation might be fruitful.

3. Data Cleaning

The data, as it was provided, is very messy. Below we have documentented our changes to the data in the data cleaning steps.

Missing Data

Because we saw that there are many variables with missing values, we tried handling these. After checking the data set codebook, we set all the missing factor data as a new `NA` level, and all the missing numeric data as 0. Take the training data as an example: since `Bsmt.Full.Bath` and `Bsmt.Half.Bath` both have missing rate less than 1%, and they have values of 0, 1 or 2. So we just replace the NA with 0; For `Mas.Vnr.Area`, we refer to the variable `Mas.Vnr.Type` and found that it contains values of `None` and `NA`, and for each NA value of `Mas.Vnr.Type`, `Mas.Vnr.Area` equals NA too. So we decide to bound them in regression function as interation, and so here we replace NAs with 0; For `Garage.Yr.Blt`, we decide to bound it with `Garage.Type` as interation in regression. So here we just treat NAs to be 0; For `Lot.Frontage`, since we have the variable `Lot.Area`, which contains the information of `Lot.Frontage`, so we deicide not to include this variable in the regression function.

Modifing Variables

The fact that some variables in test data contain new levels makes it difficult for us to predict. Therefore, our idea is to change their factor levels into numeric ones. Specifically, for `Heating.QC`,`Kitchen.Qual`,`Garage.Qual`,`Garage.Cond`, we change their factor levels as follows:

numeric	level	condition
5	Ex	Excellent
4	Gd	Good
3	TA	Typical/Average
2	Fa	Fair
1	Po	Poor
0	NA	No Garage

Dropping Variables

Because some variables had observations in the testing set that aren't in the training set (which creates some issues with certain models) or variables were found to be redundant, some variables were dropped. This includes `Condition.2`, `Electrical`, and `Lot.Frontage`.

4. Initial Model

Going by what we have seen in the exploratory data analysis, we tried a linear regression model with a subset of predictors that are correlated with price. We also take the log of the selling price.

```
ames_train_subset2 = select(ames_train,
                            price, area, Lot.Area, Overall.Qual, Overall.Cond,
                            Full.Bath, Enclosed.Porch, TotRms.AbvGrd,Garage.Area)
model.simple = lm(log(price) ~ ., ames_train_subset2)
```

This gives us a adjusted R2 of 0.82, which is reasonable given this very simple model.

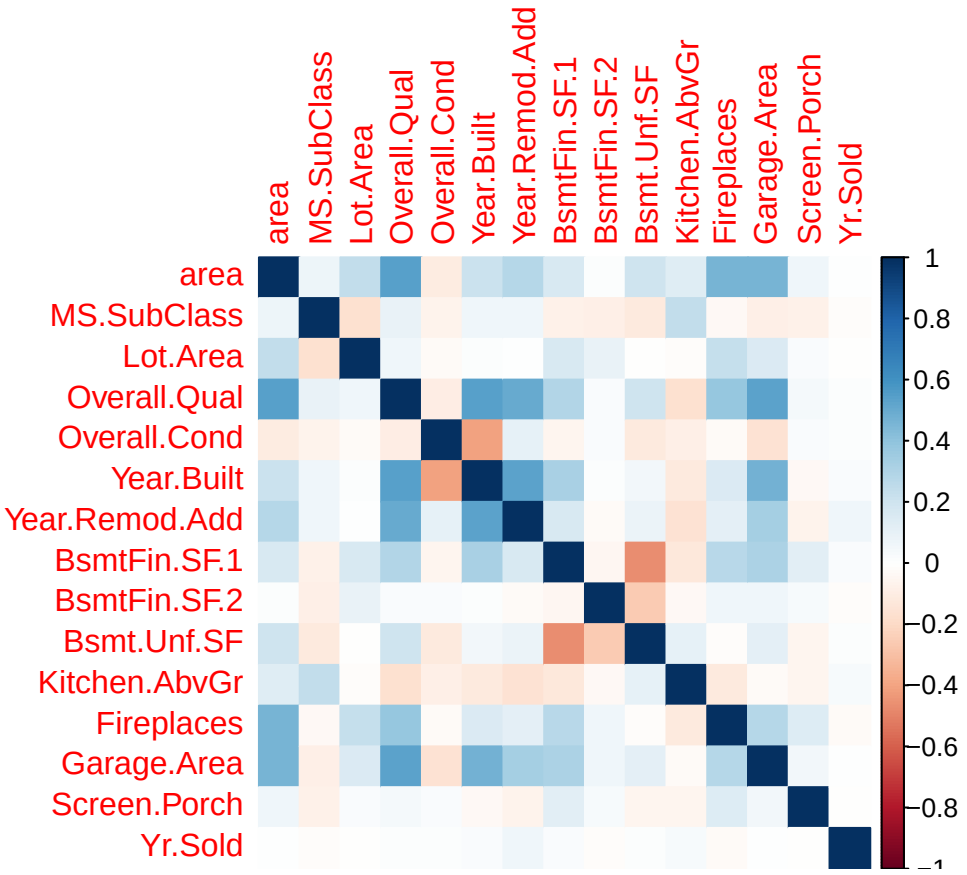
We aren't limited by just these variables though. Looking at all the numeric variables (without missing values), there are 32 of them. But we think that many of the variables will be neglible in impact. Therefore we used stepwise selection to obtain a best linear model. Since our model should include no more than 20 variables (for interpretability), here we choose backwards selection with BIC as model selection criteria.

```
model.simple2 <- lm(log(price)~., ames_train_subset)
n = nrow(ames_train_subset)
model1.bic <- step(model.simple2, k = log(n), trace = 0) # model selection (BIC)
```

	coefficients.model1.bic.
(Intercept)	14.5038352
area	0.0002807
MS.SubClass	-0.0004214
Lot.Area	0.0000027
Overall.Qual	0.0789916
Overall.Cond	0.0440441
Year.Built	0.0031164
Year.Remod.Add	0.0012082
BsmtFin.SF.1	0.0002099
BsmtFin.SF.2	0.0001466
Bsmt.Unf.SF	0.0001073
Kitchen.AbvGr	-0.0501433
Fireplaces	0.0369779
Garage.Area	0.0001683
Screen.Porch	0.0001842
Yr.Sold	-0.0061656

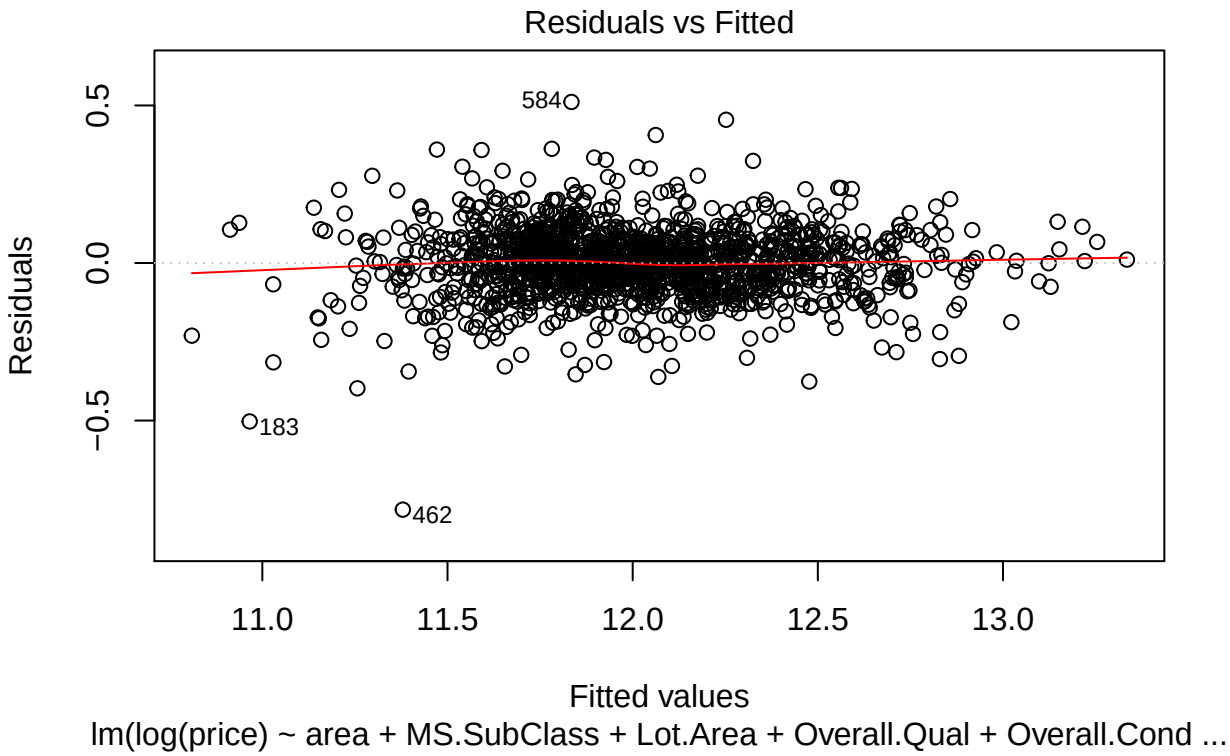
Our best BIC model has 15 predictors and has adjusted R2 of 0.92. Some important vairables not included in our first model is `Bsmt.SF.1`, `Fireplaces`, and `Yr.Sold`. When we look at the coefficients, most of them are small compared to the intercept. Selling price increases with quality and condition of the house. Price seems to decrease the older a house is, but also increases with a more recent remodeling.

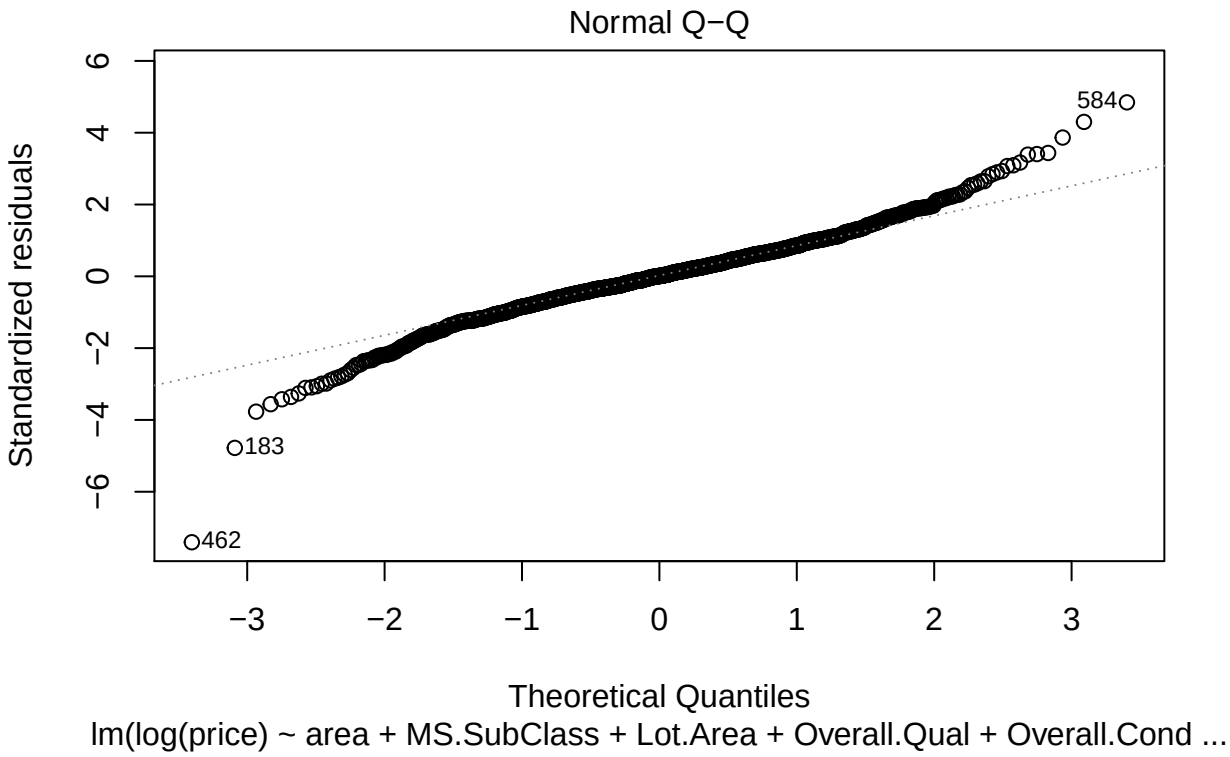
There is a concern of redundancy. For example, `Year.Built` and `Year.Remod.Add` seem to be correlated. Therefore, we use correlation plot to take a brief view.

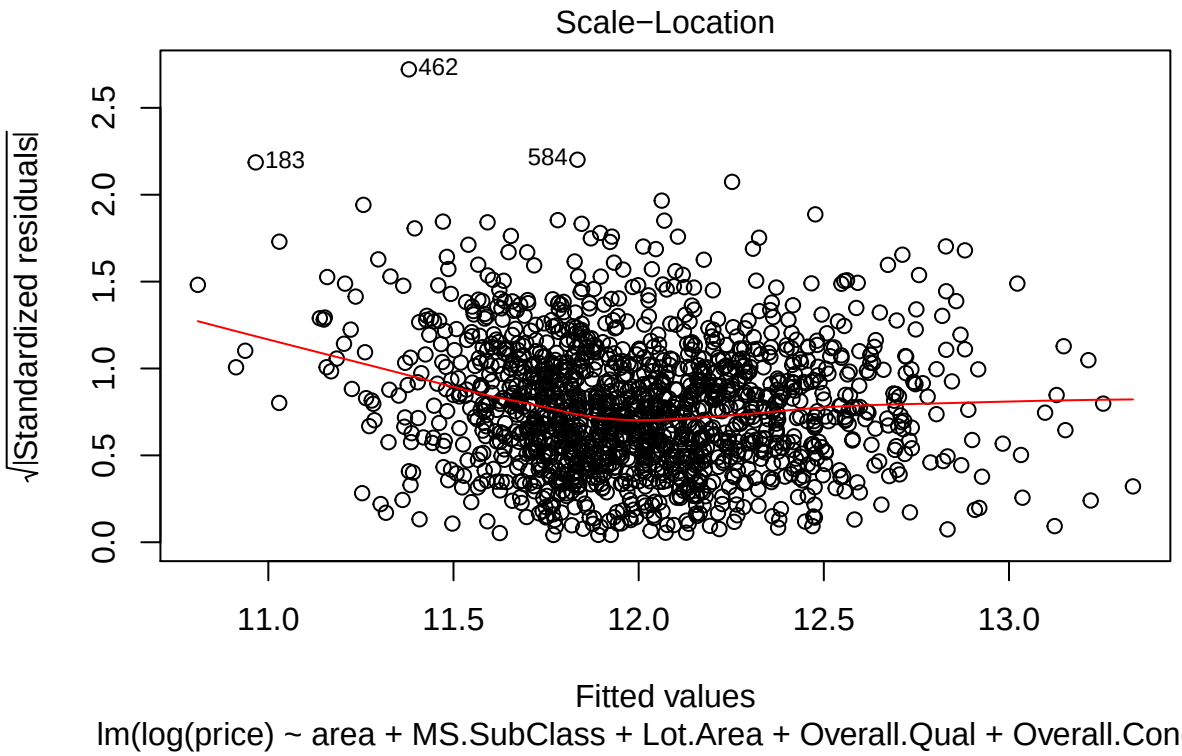


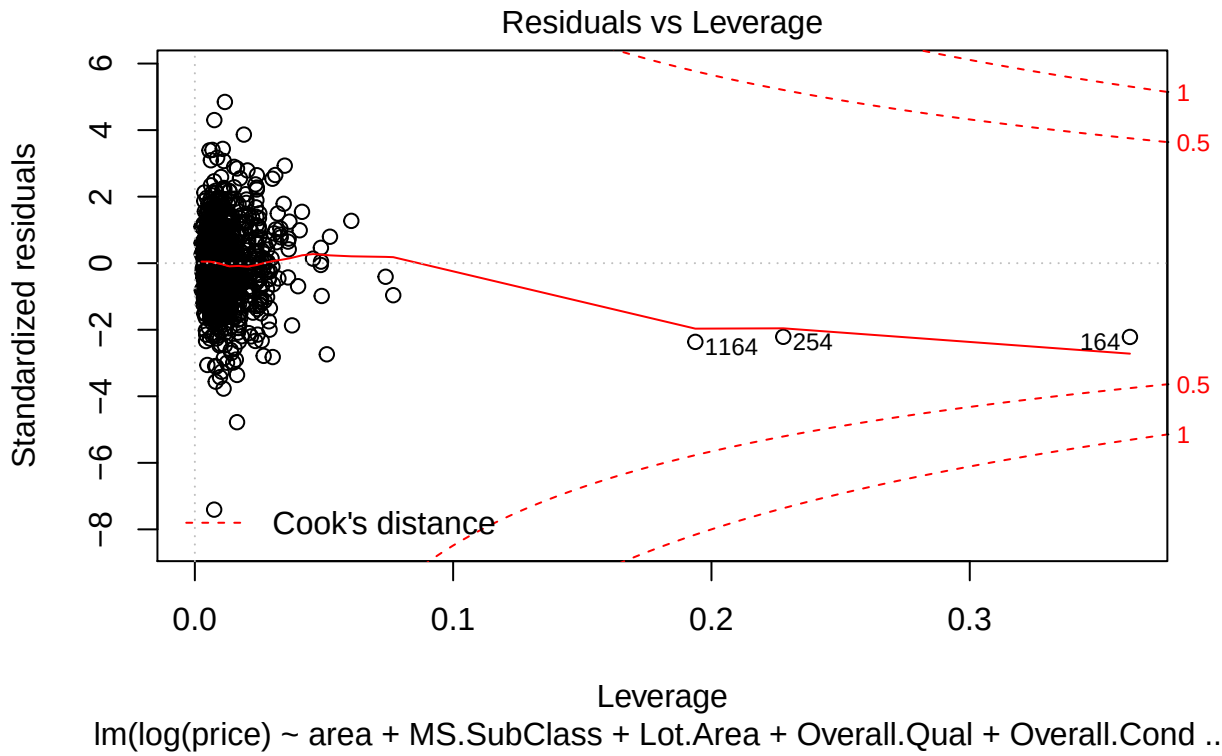
There does appear to be some correlated variables such as `BsmtFin.SF.1` and `BsmtFin.SF.2`. This gives us ideas for future feature engineering.

Now we will investigate the residuals plots to look at any issues with model fit / outliers.


Residuals vs Fitted
Residuals
0.5
0.0
−0.5
584
183
462
11.0
11.5
12.0
12.5
13.0
Fitted values
lm(log(price) ~ area + MS.SubClass + Lot.Area + Overall.Qual + Overall.Cond ...


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(log(price) ~ area + MS.SubClass + Lot.Area + Overall.Qual + Overall.Cond ...
584
183
462
6
4
2
0
−2
−4
−6
−3
−2
−1
0
1
2
3


Scale−Location
462
183
584
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
11.0
11.5
12.0
12.5
13.0
Fitted values
lm(log(price) ~ area + MS.SubClass + Lot.Area + Overall.Qual + Overall.Cond ...



The residuals vs fitted values plot tells us that for the most part the residuals are normally distributed and have constant varience. The normal QQ plot shows some values very far from normal (house 584, 462, and 183). From the leverge plot, it doesn't appear that these values are too influential.

Now we make predictions on the test data. The RMSE on the best linear model chosen with BIC is 1.914×10^4 and the coverage is 0.94.

5. Final Model

We tried several models to find out important variables and interactions.

5.1 Boosting

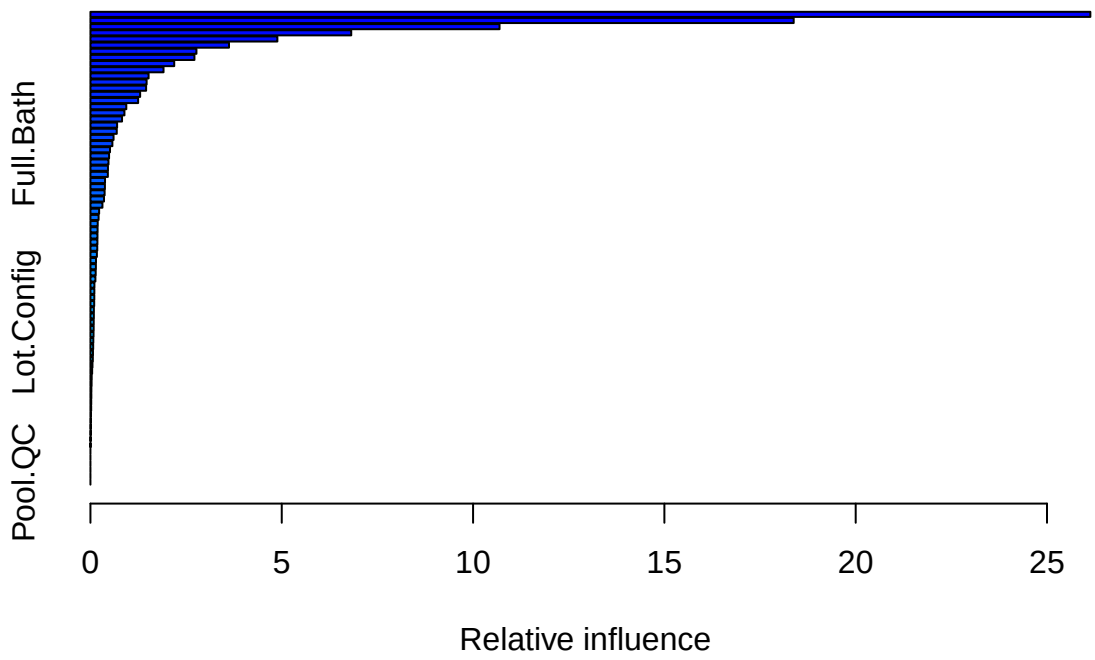
Because we think that the log price might not linearly depend on the factors, we first tried a non-linear tree-based model.

```
set.seed(1)
boost=gbm(log(price)~.,
              data=ames_train,
              distribution="gaussian",
              n.trees=10000,
              interaction.depth=4,
              shrinkage = 0.01,
              cv.folds = 5
          )
yhat.boost=exp(predict(boost,
```

```
                          newdata=ames_test,
                          n.trees=10000,
                          type="response"))
RMSE.boost =  sqrt( mean( (yhat.boost - ames_test$price)^2, na.rm = TRUE) )
```

The RMSE for the boosted model is 1.3849×10^4.

We can look at the variables importance from the boosted model.



	var	rel.inf
Overall.Qual	Overall.Qual	26.132959
Neighborhood	Neighborhood	18.379453
TotalSq	TotalSq	10.691493
area	area	6.815730
Total.Bsmt.SF	Total.Bsmt.SF	4.885260
Garage.Area	Garage.Area	3.620994
X1st.Flr.SF	X1st.Flr.SF	2.770324
BsmtFin.SF.1	BsmtFin.SF.1	2.717837
Kitchen.Qual	Kitchen.Qual	2.189707
Overall.Cond	Overall.Cond	1.909975
Garage.Type	Garage.Type	1.519814
Lot.Area	Lot.Area	1.468325
Year.Remod.Add	Year.Remod.Add	1.455502
Central.Air	Central.Air	1.298766
Year.Built	Year.Built	1.246544

The top most important variables seem to include `Overall.Qual`, `Neighborhood`, and `TotalSq`. In particular the Neighborhood variable seems to indicate how important location is to home selling price.

5.2 Variable Interactions

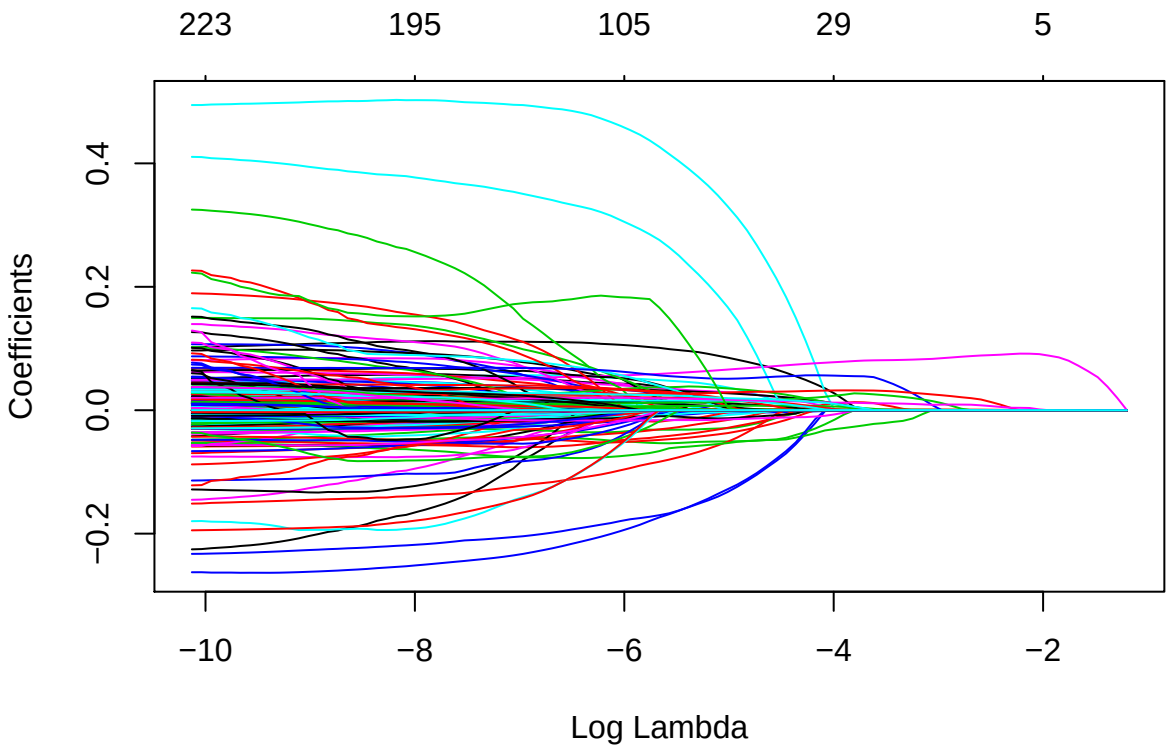
We attempted to add variable interactions to the model, by adding multiplicative effects on the top 15 most important variables via boosting. However including interactions did not improve our RMSE scores.

We speculate that the reason is tree-based models don't need explicit interaction terms as much as linear models. Another reason might be that by limiting the number of iterations, our trees are not fully utlizing the added features.

5.3 Variable Selection

There are many factors in our original dataframe, even after the data-cleaning step. In addition many of the factor variables are very unbalanced. Because some terms may not be relevant, we tried Lasso as a way to select important variables.

```
sub_train = ames_train %>% select(-c(Utilities,price))
y = ames_train %>% select(price) %>% as.matrix()
train_sparse <- model.matrix(~.,sub_train)
glmlas <- glmnet(train_sparse,log(y),alpha = 1)

plot(glmlas,xvar = "lambda")
```



This plot shows how variables drop out as λ increases. Next we implment 10-fold cross-validation to refine our model by choosing the best λ.

```
glmlas.cv <- cv.glmnet(train_sparse,log(y),nfolds=10,alpha = 1)

sub_test = ames_test %>% select(-c(price,Utilities))
test_sparse <- model.matrix(~., sub_test)

yhat.glmlas <- exp(predict(glmlas, test_sparse, type="response", s= glmlas.cv$lambda.min))
```

The RMSE for Lasso is 1.5433069×10^4 which is higher than the boosting model. Therefore we continue with refining the boosting model.

5.4 Final Boosting Model

We will continue to optimize the boosting model. Because of a fear of overfitting, we use five-fold cross-validation to find the best number of trees. We also use a regularization term to further prevent overfitting.

We include all the variables in the boosting model to predict the price. We set the interaction depth of our boosting model as 4, as each tree in the model has four splits with a four-way interaction.

We have the shrinkage parameter or learning rate of the boosting model as 0.01. It controls the rate of minimizing the loss function and shrink each tree in the model.

```
best.iter <- gbm.perf(boost, plot.it = FALSE, oobag.curve = FALSE, overlay = TRUE)
```

```
## Using cv method...
```

Using the gbm perf function, the estimated number of trees is 4565. Now we fit our final model and predict the new results.

```
#check the performance of the model optimized after 5-fold cross validation
set.seed(1)
boost.cv=gbm(log(price)~.,
             data=ames_train,
             distribution="gaussian",
             n.trees=best.iter,
             interaction.depth=4,
             shrinkage = 0.01)
```

The RMSE for the boosted model after cross validation is 1.3827×10^4 which is better than our other models.

6. Results

Finally we get results from our final boosting model.

Coverage with Quantile Regression

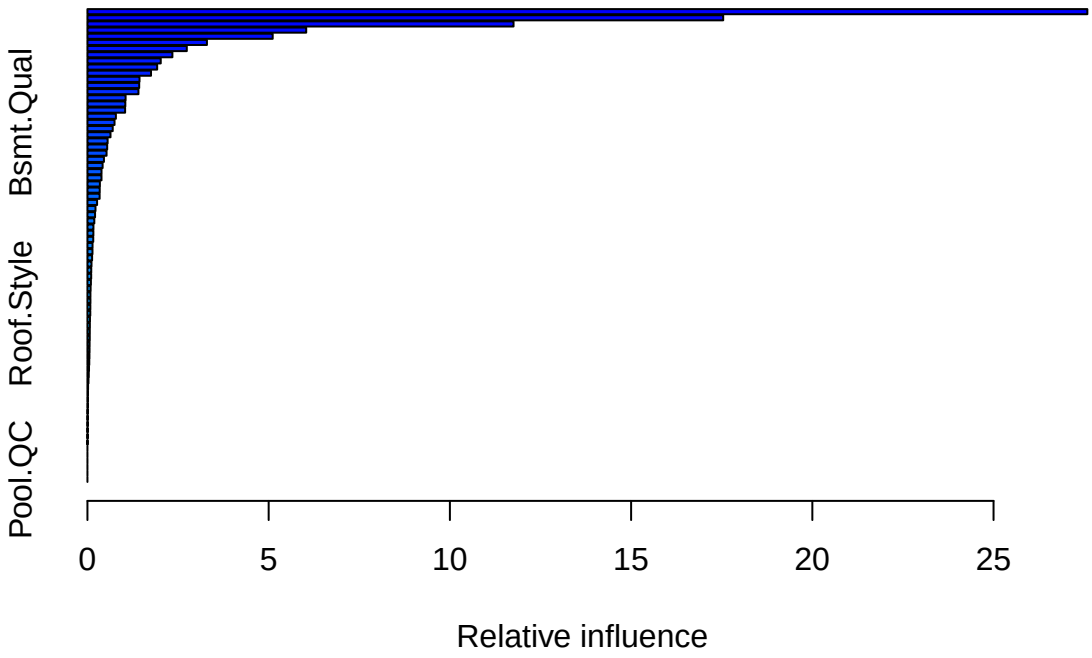
```
boost.pi1=gbm(boost.cv,
            distribution=list(name = "quantile", alpha = 0.025),
            data = ames_test,
            n.trees=best.iter)
boost.pi2=gbm(boost.cv,
            distribution=list(name = "quantile", alpha = 0.975),
            data = ames_test,
            n.trees=best.iter)


# coverage
cov.boost = mean((ames_test$price) >= exp(boost.pi1$fit) & ames_test$price <= exp(boost.pi2$fit))
```

We wanted to quantify uncertainty in our test predictions, so we used quantile regression. The coverage for the 95% confidence interval is 0.954. Finally,We made a dataframe of predictions and upper and lower CI on the testing and validation set.

6.1 Model Diganostics

We now look at model diganostics.

Here are the top variables chosen by our model.

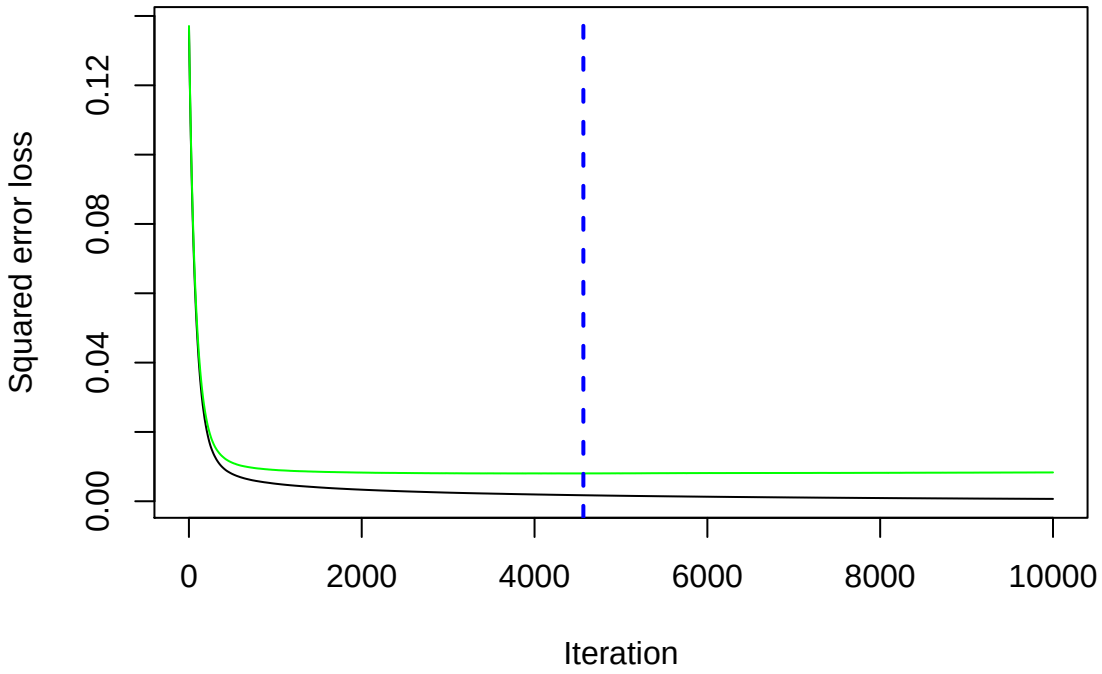


	var	rel.inf
Overall.Qual	Overall.Qual	27.5879852
Neighborhood	Neighborhood	17.5408811
TotalSq	TotalSq	11.7562728
area	area	6.0360007
Total.Bsmt.SF	Total.Bsmt.SF	5.1111584
Garage.Area	Garage.Area	3.2972008
BsmtFin.SF.1	BsmtFin.SF.1	2.7415530
X1st.Flr.SF	X1st.Flr.SF	2.3441500
Kitchen.Qual	Kitchen.Qual	2.0226767
Overall.Cond	Overall.Cond	1.9236879
Garage.Type	Garage.Type	1.7539018
Lot.Area	Lot.Area	1.4365734
Central.Air	Central.Air	1.4277887
Year.Remod.Add	Year.Remod.Add	1.4074012
Garage.Cars	Garage.Cars	1.0530177

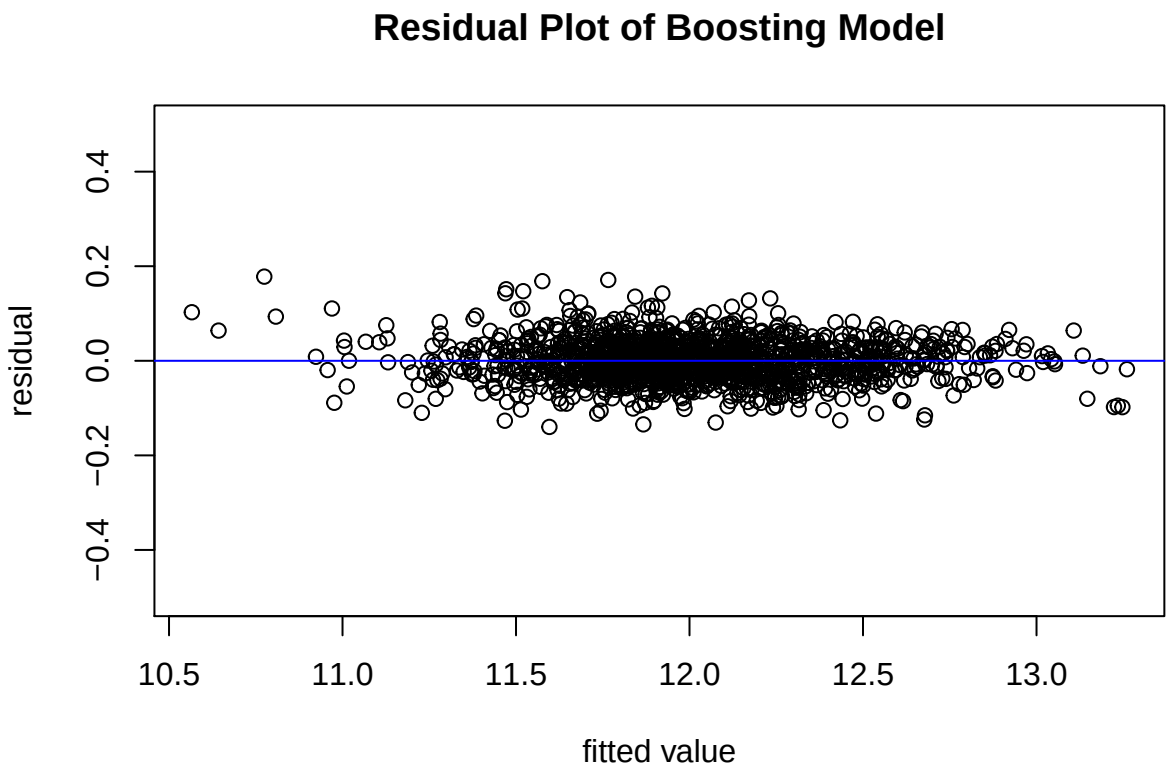
	var	rel.inf
Year.Built	Year.Built	1.0457442
Exter.Qual	Exter.Qual	1.0392332
Fireplace.Qu	Fireplace.Qu	0.7851516
Bsmt.Qual	Bsmt.Qual	0.7477071
BsmtFin.Type.1	BsmtFin.Type.1	0.6972331

We can see that the best number of trees chosen by cross-validation. As the number of trees increase, the model fits better until overfitting occurs. The green line is the out-of-sample error as predicted by cross-validation.

```
## Using cv method...
```



Finally we look at the residual plots.



The residuals seem to be centered around zero. Compared to our simple linear model, the residuals are much smaller. The model seems to fit more poorly for cheaper houses.

7. Analysis of House Valuation

We find the top 10 and bottom 10 best houses to buy according our model.

```
ames_test$price_diference = (yhat.boost.cv-ames_test$price)/ames_test$price
under_value <-ames_test %>%top_n(10, wt=price_diference)%>%arrange(desc(price_diference))
kable(select(under_value, Under_PID = PID, price))
```

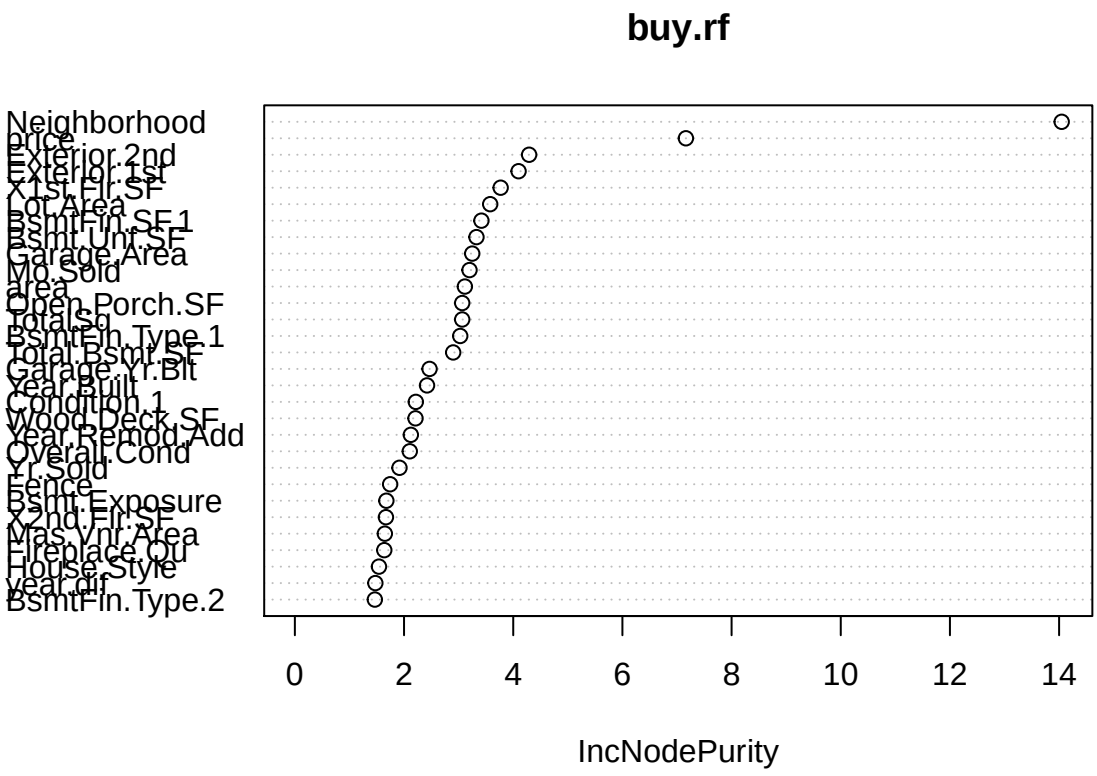
Under_PID	price
905200160	80000
527182020	130000
535478070	120000
909101010	110000
528102010	315000
917425190	186700
902105130	64000
532351140	112000
535150070	104900
533251110	255000

```
over_value <- ames_test %>%top_n(-10, price_diference)%>%arrange(price_diference)
kable(select(over_value, Over_PID = PID, price))
```

Over_PID	price
535454070	166000
904300150	107500
905376090	216000
903231060	60000
905427010	235000
535382020	160000
905452050	113000
903429110	179900
528186180	254000
909275110	238000

We seperate the over and undervalued houses and do further analysis. In the end we want to find variables that are related to house valuation.

```
ames_test$should_buy = ifelse(ames_test$price_diference>0,1,0)
ames_test$year.dif = ames_test$Year.Remod.Add - ames_test$Year.Built

buy.rf = randomForest(select(ames_test, -should_buy,-price_diference, -PID), ames_test$should_buy)
varImpPlot(buy.rf)
```



```
kable(head(buy.rf$importance, 20))
```

	IncNodePurity
area	3.1141044
price	7.1633790
MS.SubClass	0.9641022
MS.Zoning	0.3562148
Lot.Area	3.5781766
Street	0.0017267
Alley	0.2033911
Lot.Shape	0.7754176
Land.Contour	0.3318270
Utilities	0.0000000
Lot.Config	1.1312158
Land.Slope	0.1456845
Neighborhood	14.0470781
Condition.1	2.2155078
Bldg.Type	0.8169073
House.Style	1.5404716
Overall.Qual	0.9324498
Overall.Cond	2.1048091
Year.Built	2.4206555
Year.Remod.Add	2.1241868

This results suggest that comparing to predicting the price, predicting wether a house is undervalued will depend more on exterior covering of the house. Lot area and area of open porch will also be important. While the neighborhood is the most important predictor for predicting both price and undervalue. Location does matter the most.

```
buy1 = glm(should_buy ~ price + area+Year.Built+Year.Remod.Add+Overall.Qual + Neighborhood + TotalSq + T
```

We fit a logistic regression model to predict whether a house is overpriced or underpriced. Logistic regression using shows that houses with larger basement area, Lot area, higher overal quality, more fireplaces and lower price tend to be more under-valued.

```
buy.under <- ames_test %>% select(price, area,should_buy, Fireplaces, X1st.Flr.SF, Lot.Area) %>%
  group_by(should_buy)%>%
  summarise(mean(price),mean(area), mean(Fireplaces),mean(X1st.Flr.SF), mean(Lot.Area))
kable(buy.under)
```

should_buy	mean(price)	mean(area)	mean(Fireplaces)	mean(X1st.Flr.SF)	mean(Lot.Area)
0	177161.9	1493.426	0.4956522	1093.657	9544.713
1	165443.1	1482.489	0.6185185	1131.022	10389.570

Under-valued houses seem to have lower price, more fireplaces, larger first floor area and lot area. One thing might be a good indicator for a undervalued house is the number of fireplaces as 82% of the houses with 2 fireplaces are undervalued.

```
kable(table(ames_test$should_buy, ames_test$Fireplaces))
```

	0	1	2	3
0	123	101	5	1

	0	1	2	3
1	128	118	23	1

8. Conclusion

The results of the models below, including our simple initial model and our final model based on boosting. The simple model was restricted to only numerical covariates. Because we wanted a parsimonious model, we used BIC to do variable selection. We tried two methods for a more complex model including Lasso and tree-based boosting. We settled on the boosting model as this gave very good results on the testing set.

models	RMSE	cov
Linear (BIC)	19139.95	0.940
Boost1	13848.71	NA
Lasso	15433.07	NA
Boost-Final	13826.81	0.954

Our final model is a gradient boosted trees (gbm) model with log transformed price. By decreasing the learning rate(shrinkage term), we achieved good results with the original covariates. However the small shrinkage value does increase the time of computation significantly; it takes several minutes to fit our model compared to a few seconds for the linear model.

We set the maximum number of trees to be high so that our model could learn well. However, we were concerned about issues with overfitting. One of the models that we tried, but did not end up using was the Random Forest model. If the number of trees for Random Forest is too high, there is a big chance of overfitting. As part of our model testing, we used five-fold cross validation with gbm built in functions to find the best number of iterations for our boosting model. This significantly improved our results in terms of RMSE.

One thing that surprised us is that scaling the covariates and manually adding interaction terms did not improve our model fit. We speculate that this is due the true model being significantly non-linear. The tree-based model may have discovered these non-linear interactions better than, for example, Lasso because trees are strongly non-linear.

Finally, we produced predictions on the test and varlidation data. We used quantile regression to estimate confidence intervals on the test data, but could not do something similar for the validation. Our final model achieves an RMSE of lower than 14000. We used other modeling techniques to dervive insights on over or under valued houses. We found that variables such as number of fireplaces may have a surprising correlation with undervalued houses.

Some things that we could have done for the future is spend more time optimizing the hyperparameters of the boosting model. Another next step would be to look at other models that could help quantify varience like Bayesian methods. Finally, we acknowledge that while the model we chose gives accurate predictions, it is very hard to interpret.