Final Project Report Stat 444

'Statistical Analysis on a Housing Dataset'

by Mark Christopher Uy (20870243)

I. Abstract

The goal of this report is to analyze a specific housing data set provided in the University of Waterloo's Stat 444 Winter 2020 course. By applying different statistical techniques and using different learning models, we wish to find the key indicators that provide valuable insights and information to a house's selling price. This report is subdivided into multiple sections that includes an introduction to the data set, a detailed description of the steps taken to pre-process and clean the data set, and lastly a detailed statistical analysis, which will provide valuable, and useful information towards the key indicators that drive housing prices. It is also the hope of this paper to create and find an accurate and successful regression model, which can be useful to investors and the general people alike in pricing future homes. Some of the models, which will be explored, include powerful tools such as Linear Regression, Random Forests, Gradient Boosted Trees, Lasso Regression, and lastly we will look have a quick look at how recent advances in Deep Learning can maybe provide us with a more powerful, and accurate model. The performance of the different models will also also be compared with one another in terms of performance, computational complexity, and interpretability. All of the R code, and Graphs that was used in the process of this statistical analysis will be included in separate appendix found after the pages of the report. The report will also be making occasional references to the appendix for the reader's convenience.

II. Introduction To The Data set

Throughout the report, we will be referring to the data set used and studied as Stat 440's Housing Prices Dataset, abbreviated as the S440 data set for convenience. The data set has 12,474 observations and each observation has 31 different variables. Of the 12474 data points, 10,002 of them are meant to be use for model training, and 2472 are part of the testing data set. Overall, we have a mix of both integer/ numeric data, and also factor variables.

The full list of numeric variables are: *BATHRM, HF_BATHRM, ROOMS, BEDRM, AYB, EYB, YR_RMDL, STORIES, GBA, KITCHENS, FIREPLACES, LANDAREA, ZIPCODE, LATITUDE, LONGITUDE.*

Similarly, the full list of factor variables are*: HEAT, AC, SALEDATE, STYLE, GRADE, CNDTN, EXTWALL, ROOF, INTWALL, ASSESSMENT_NBHD, ASSESSMENT_SUBNBHD, WARD, QUADRANT.*

(* For a full list of the descriptions of these variables. The reader may refer to the appendix section entitled Variable Description.)

Now, we will be looking at each variable more in depth to have a better understanding of the data we are working with. Below is some descriptive information of the numeric data.

Descriptive statistics

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
BATHRM	1	12474	2.58	1.17	2	2.49	1.48	0	12	12	1	2.26	0.01
HF_BATHRM	2	12474	0.81	0.63	1	0.75	0	0	5	5	0.36	0.33	0.01
ROOMS	3	12474	7.97	2.15	8	7.74	1.48	0	28	28	1.26	3.35	0.02
BEDRM	4	12474	3.8	1.11	4	3.7	1.48	0	15	15	0.97	2.22	0.01
AYB	5	12474	1942.02	25.81	1937	1938.72	17.79	1754	2018	264	0.99	2.22	0.23
EYB	6	12474	1972.64	16.1	1969	1970.7	10.38	1928	2018	90	1.07	0.82	0.14
STORIES	7	12474	1.98	0.57	2	2.01	0	1	25	24	11.51	451.45	0.01
GBA	8	12474	2134.52	1063.76	1857.5	1971.16	725.73	480	15902	15422	2.27	9.79	9.52
KITCHENS	9	12474	1.04	0.2	1	1	0	0	4	4	5	27.98	0
FIREPLACES	10	12474	1.16	0.9	1	1.08	0	0	13	13	1.58	7.54	0.01
LANDAREA	11	12474	6280.26	4018.5	5460	5733.11	2104.55	255	155905	155650	8.74	206.52	35.98
ZIPCODE	12	12474	20014.88	4.76	20016	20014.9	4.45	20001	20037	36	0.55	2.58	0.04
LATITUDE	13	12474	38.94	0.03	38.94	38.94	0.03	38.83	39	0.17	-0.92	0.39	0
LONGITUDE	14	12474	-77.03	0.05	-77.04	-77.04	0.06	-77.11	-76.91	0.2	0.48	-1.03	0

As we can see from the image above, our different indicators are of different scales. We see that for example general scale of the values in *LANDAREA* are in the thousands, while for *FIREPLACES* it is only the single digits. Obviously they represent different quantities, but the heterogeneity of the data is something we have to watch out for while performing our analysis, and our model building. Another thing to note is the skew, and kurtosis of the data. For those variables with large skewness, we may need to perform some kind of transformation, (i.e. log or box-cox).

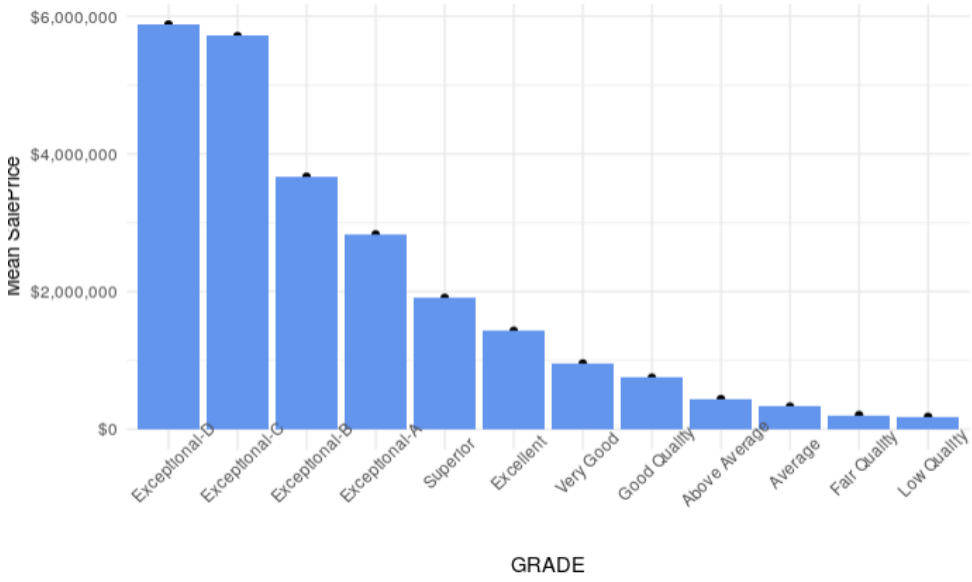
Next thing we want to look for is variables with not much variance in the values of their data points. This is important because such a column or variable does not provide much information to us with regards to our analysis. Hence, we may want to ignore it.

Variables with (near) zero variance

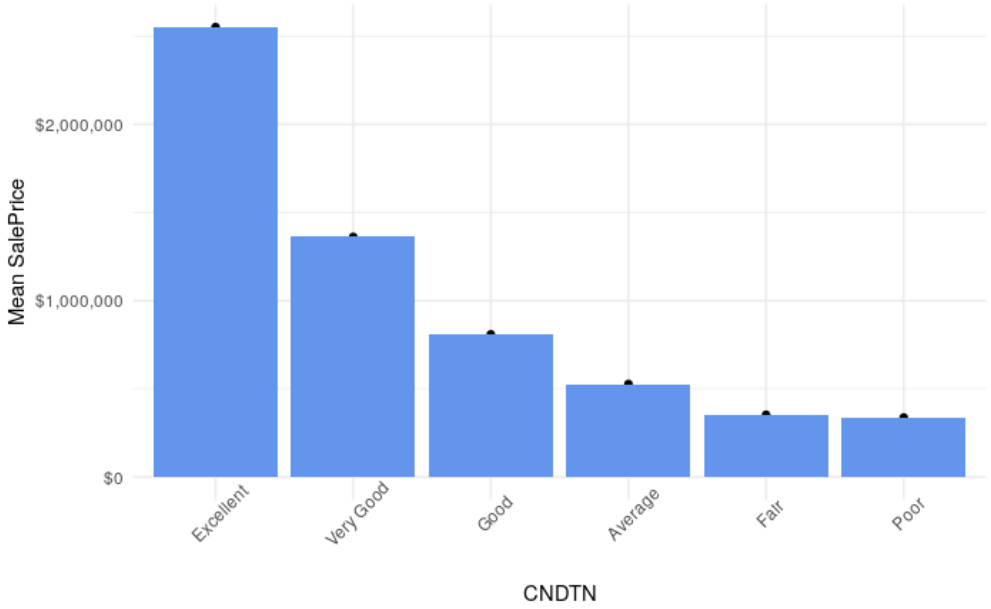
	freqRatio	percentUnique
KITCHENS	25.07	0.05

Ordering Factor Variables

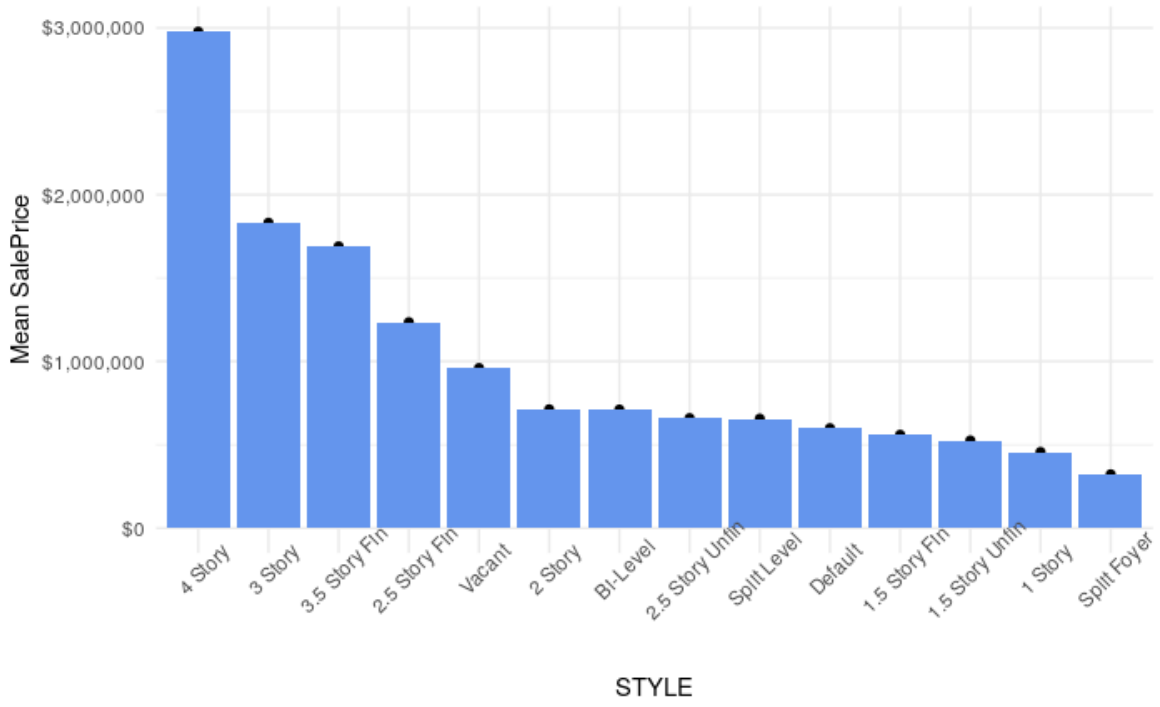
Some of the factor variables have a clear ordering to them, such as in terms of quality. We will order the following factor variables: *GRADE, and CNDTN, and STYLE*.



From the graph above, we order the levels of *GRADE* as follows: *"Low Quality", "Fair Quality", "Average", "Above Average", "Good Quality", "Very Good", "Excellent", "Superior", "Exceptional-A", "Exceptional-B", "Exceptional-C", "Exceptional-D"*.



From the graph above, we order the levels of *CNDTN* as follows: *"Poor", "Fair", "Average", "Good", "Very Good", "Excellent"*.



From the graph above, we order the levels of *CNDTN* as follows: *"Split Foyer", "1 Story", "1.5 Story Unfin", "1.5 Story Fin", "Default", "Split Level", "2.5 Story Unfin", "Bi-Level", "2 Story", "Vacant", "2.5 Story Fin", "3 Story", "3.5 Story", "4 Story"*. Notice the switch between 3 and 3.5, as this makes more sense logically.

III. Data Preprocessing and Feature Extraction

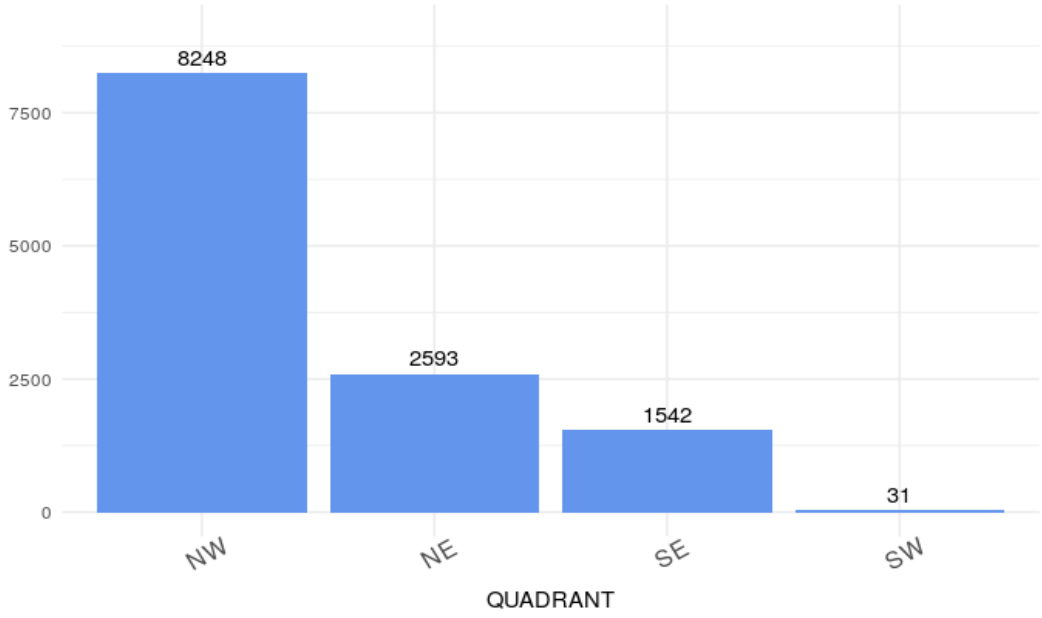
Handling Missing Data:

The first thing we are going to do is to address the problem of the missing data. The following columns have missing data:

YR_RMDL	ASSESSMENT_SUBNBHD	QUADRANT	AYB	STORIES	KITCHENS
5049	2994	60	28	7	1

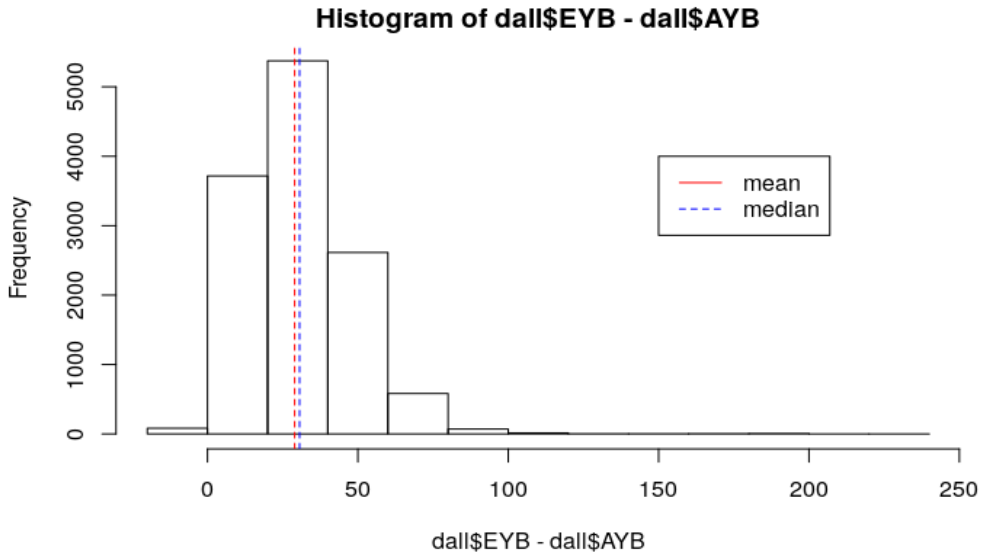
In order to fill in the missing the data, are are going to look at each variable. Firstly, due to the quantity of missing data for *YR_RMDL*, and *ASSESSMENT_SUBNBHD*, (> 25% of the total number of data points), we are going to be removing them from our analysis.

Secondly, we look at how the values in *QUADRANT* column as a factor variable are broken down. The chart in the next page shows that majority of the values are *NW* (North West). As a result of this, we have decided to fill in the missing *QUADRANT* values as *NW* as well.



Next, we look *AYB* (Actual Year Built), which is a variable which indicates the year a house was created or finished construction. Now, in order to fill in the missing values here, we will use another variable which has no missing data, *EYB*, which indicates the year a renovation was made to the house.

As seen in the chart below, the mean and median difference between the *AYB* and EYB of houses are and 29 years, and 30.63 years respectively. For our purposes, we will subtract the *EYB* value of data points with missing *AYB* data by the median difference of all data point's *EYB* and *AYB* , 30.63 years, in order to get our estimated *AYB* values.

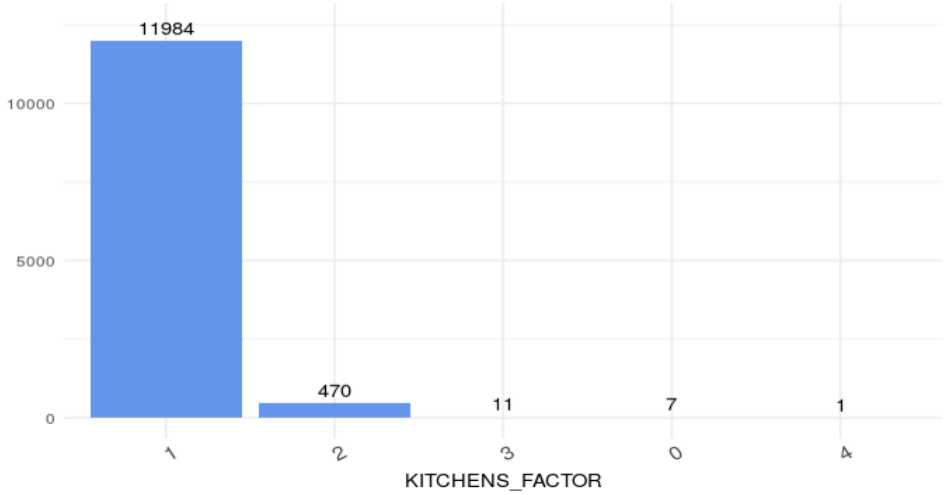


Next, we are going to be looking at STORIES. There's actually another variable that is quite similar to STORIES, and that is STYLE.

The following are the STYLE values of the 7 data points with missing stories value.

2 Story, 2.5 Story Fin, 2 Story, 2 Story, 2 Story, 2 Story, 2 Story

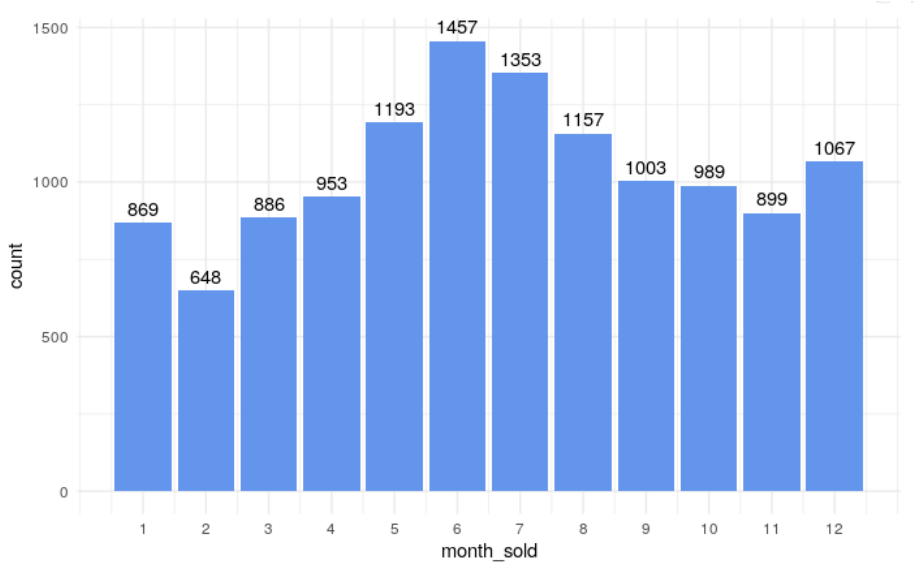
By looking at the co-occurrence table in the Appendix, we can see that the most common STORIES value for *2 story* is 2, and for *2.5 Story Fin*, it is 2.5. Hence, these are values we are going to be using.
Lastly, we are going to be looking at the *KITCHEN* variable. From the graph below, since almost an absolute majority of values are 1, that is the value we will choose to replace our single missing kitchen data value.



This concludes our handling of missing data, next we explore each variable more, and try to add new variables for our models to predict with.

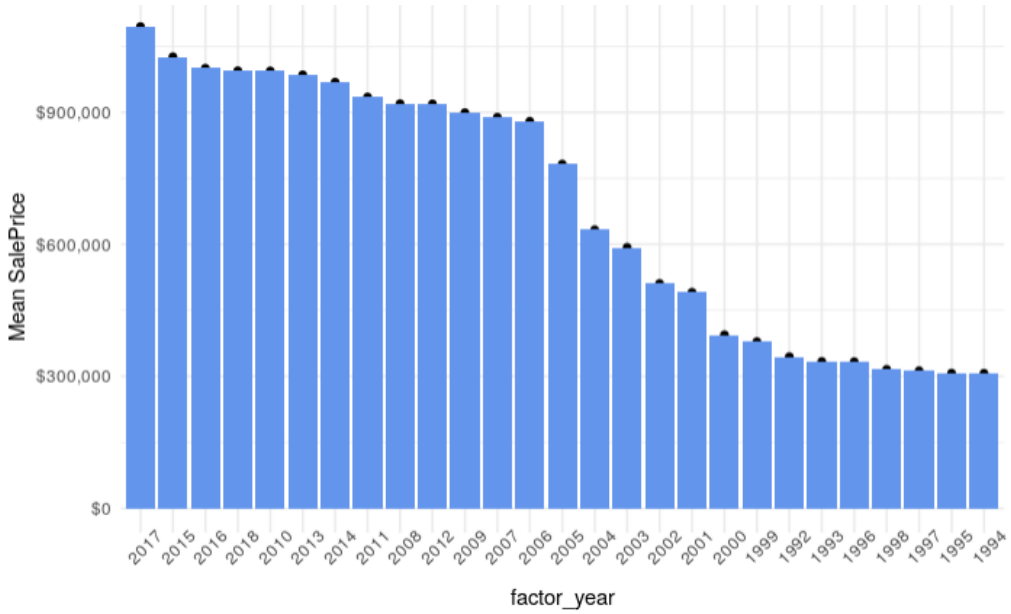
Feature Extraction

This portion of the report will focus on the steps taken to extract additional features to be used in our model building later on. The first thing we're going to be do is we're going to extract the sale date data. It is currently, in string format, but we will want to get the year and month a house was sold.



The graph in the previous page shows that there is a peak period around the summer months of May to August that sees a noticeable increase in sales. Hence, we add an additional variable, called *HighSeason* to help capture this information.

The next graph below shows the trend of housing prices through the years, and we can see an incredibly strong growth from 2000- 2007, and then slight grow from then onward. This makes sense as the years from 2000-2007 was during the period of the housing market pricing bubble.



Similar to *HighSeason*, we add the following new features to our data set:

a. *Remod* (a binary variable to indicate if a remodel was ever done to a house)

b. *RecentRemod* (a binary variable to indicate if a remod. was done on the year house was sold)

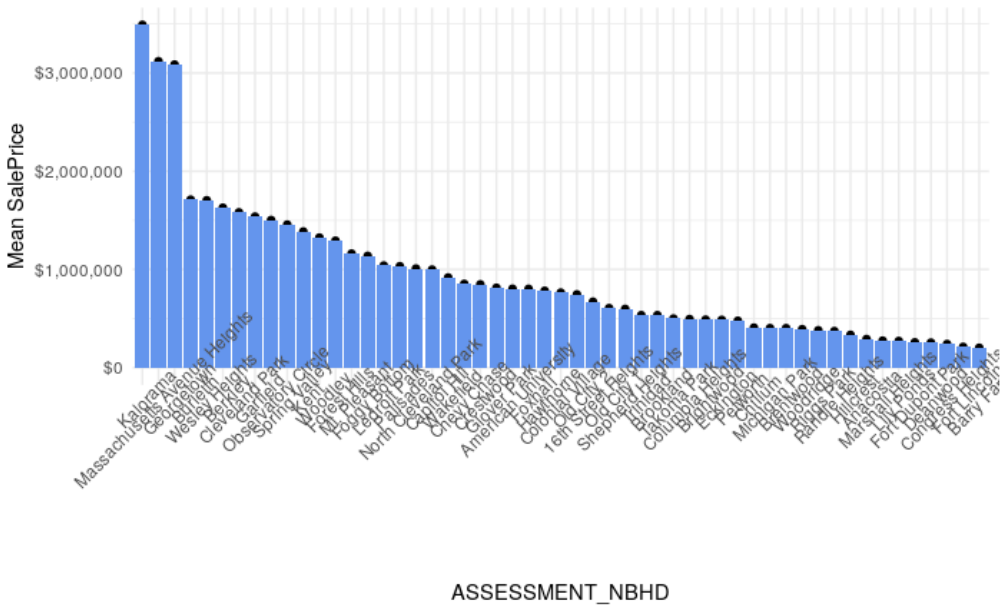
c. *yearSinceRemod* (a numeric variable to indicate time diff. between year sold and remod. year)

d. *IsNew*(a binary variable to indicate if house was sold on year it was made)

e. *Age* (how old house is)

f. *TimeSinceSold* (how much time has past since house has been sold)

The next set of feature we are going to be focusing on will be ASSESSMENT_NBHD. This variable is of great interest because of the number of factors it has, 53. This number is a bit too large, and makes model building difficult. Hence, we wish to reduce the number of levels to something more manageable like 5.



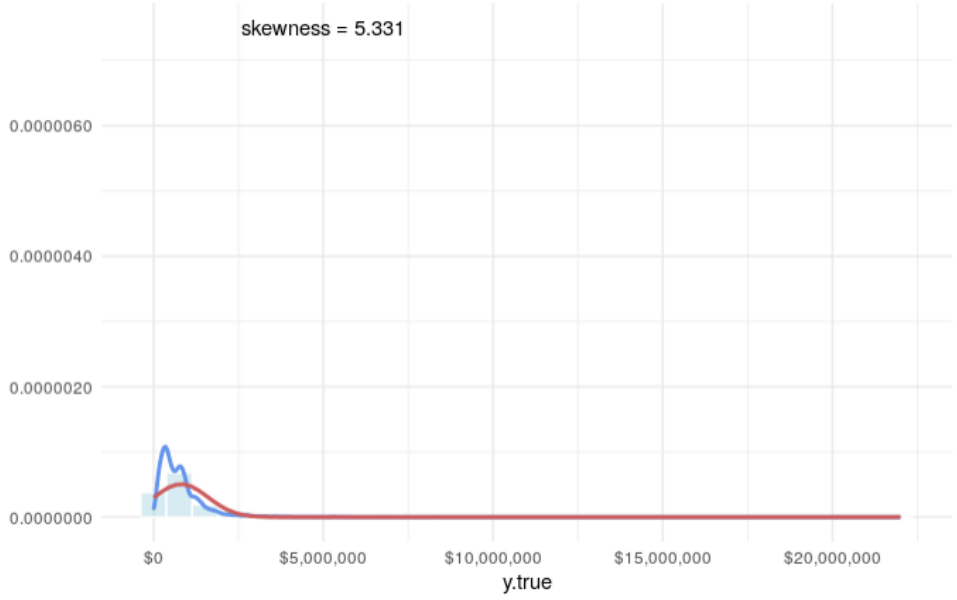
Using the graph above, we subdivide the 53 different neighborhoods into 5 levels: *Super Rich*, *Rich*, *Average*, *Below Average*, and *Poor*. The exact division can be found in the appendix.

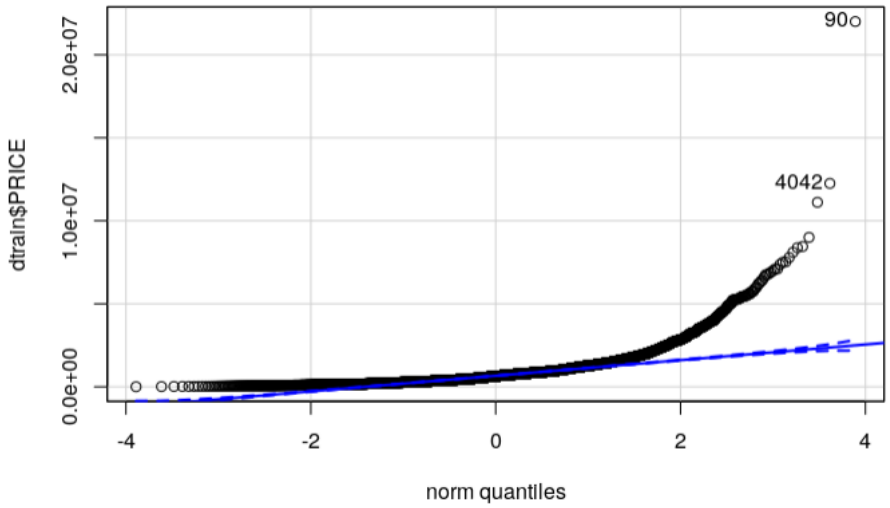
The last feature of interest in this section is the number of bathrooms in a house. There are 2 variables that pertain to this in our data set, namely, *BATHRM* and *HF_BATHRM*, and so for our purposes. We combine the two variables into a new variable called *TotBathrooms* using the following formula:

$$TotBathrooms = BATHRM + HF\_BATHRM*0.5$$

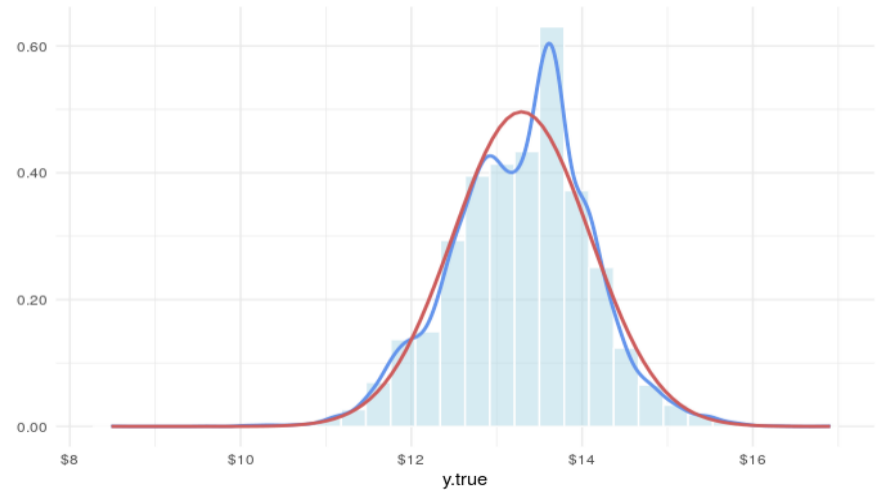
Data Transformation:

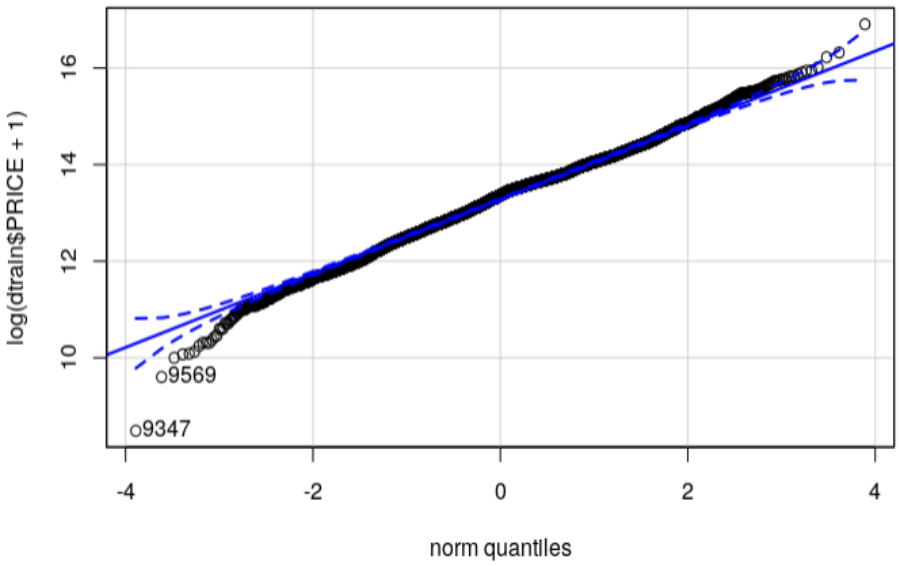
First thing we might want to transform is the response variable, *PRICE.* In order to assess whether we should transform, we will look at the quantile plots of the data, as well as how the distribution of the data compares with a normal distribution. The graph below show that the *PRICE* variable is highly skewed, and greatly deviates from the assumed normal distribution which the data is generated from.





Looking at the qqplot of the *PRICE* variables confirms the observation made in the previous page. As a result, the skewness of the response makes it much more difficult to build an accurate model. Hence, we decide to apply a log1p, (log (1+x)) , transformation to the response variable. By applying this transformation, we get the following qqplot, and histogram plots below.





	skew
BATHRM	1
HF_BATHRM	0.36
ROOMS	1.26
BEDRM	0.97
AYB	0.99
EYB	1.07
STORIES	11.51
PRICE	5.41
GBA	2.27
KITCHENS	5
FIREPLACES	1.58
LANDAREA	8.74
ZIPCODE	0.55
LATITUDE	-0.92
LONGITUDE	0.48
year_sold	-0.56
month_sold	-0.05
HighSeason	0.35
yearSinceRemod	-0.84
Age	-0.99
TimeSinceSold	0.56
TotBathrooms	0.98

Similarly, we look at the skewness of the other variates in the data set. Specifically, we are interested in the variables that have a skew greater than 0.75 and less than -0.75. They are: *BATHRM, ROOMS, BEDRM AYB, EYB, STORIES, PRICE, GBA, KITCHENS, FIREPLACES, LANDAREA, TotBathrooms, LATITUDE, yearSinceRemod, Age.* Similar, to the *PRICE* variable, we will apply a log1p transformation to these variables as well.

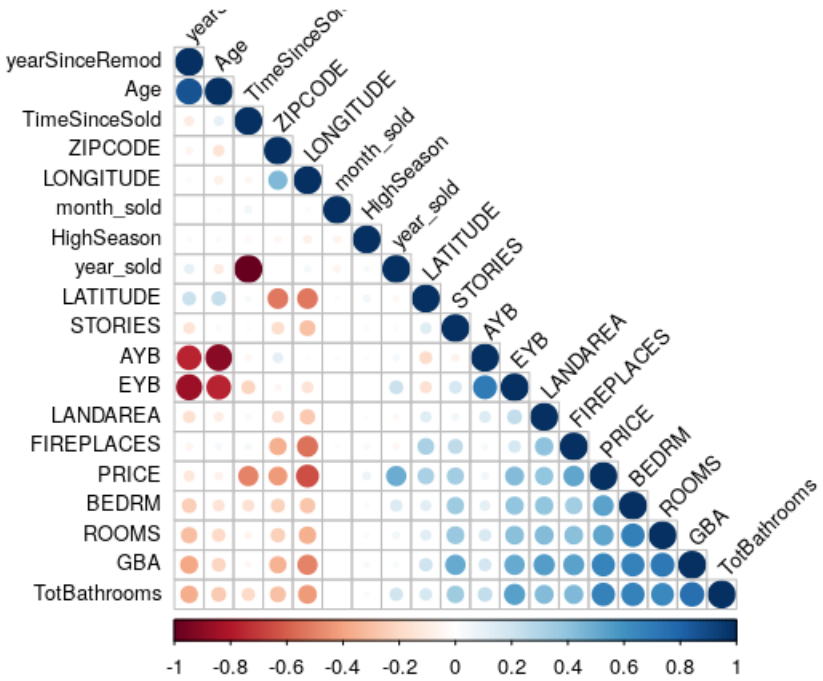
After transforming the variables to fix their skewness, we then normalize our variables by subtracting the mean, and dividing by the standard deviation of the columns. This in theory will help our models learn better, and quicker.

Another thing we do in this section is one hot encode our factor variables. This one hot encoding is a necessary format for the models to be used in the statistical analysis portion of this report. This one hot encoding is made possible using the *dummyVars* function in the *caret* package. This concludes our sections into data preprocessing and feature extract

IV. Statistical Analysis

We start this section by looking at the correlation matrix of our numeric variables. It is hard for our models to learn if our variables are highly correlated. This initial look will give us a better understanding of the data we are working with.

Correlation Matrix Analysis



Looking at this chart, we see that most pair of variates are not correlated with one another, which is good news. In fact, the only pair of variables that have a correlation greater than 0.8 or less than -0.8 are those along the main diagonal of the correlation matrix, which is the correlation of a variable with itself, which is obviously 1. Hence, we can see that the transformations performed in the previous section have made each of our variables more independent of one another.

Model Building

An important note with regards to model building needs to be made here. All models are trained and built using the caret package. The caret package is being used because of its wide selection of implemented models, and its ability to perform grid search on the values of different hyperparameters of different models. This is an important and crucial step in building top performing predictive models. Furthermore, a 5 fold cross validation, using only the training data set, is used in the building of all models.

For all models discussed in this section, we give the RMSLE score of the model with respect to the testing dataset. Please note as well that for some models, we will present the entire model building process, including the grid search performed, while for others, they will only be briefly described. For the reader's reference, the full model building process of all models discussed in this report can be found in the appendices.

Linear Regression

The first model we are going to be looking at is linear regression. The formulation of the model is:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

$$Y = X\beta + \varepsilon$$

(where $\epsilon \sim N(0, \sigma^2)$ *for an unknown constant* σ)

Using the full model built from all all predictors/ variables, we obtain a *RMSLE* score of **0.2510709.** Now, this serves as our baseline score for linear regression based models.

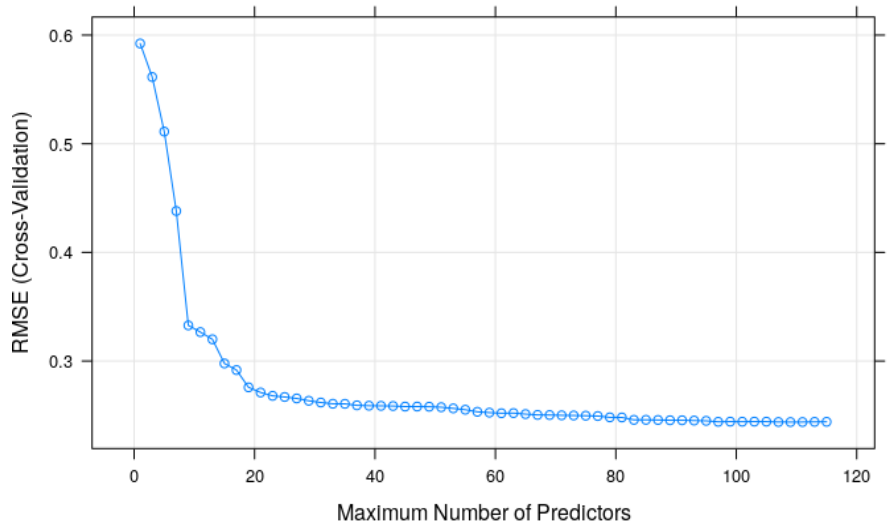
Outliers

Using the linear model, we can can search for outliers within the data using the outlierTest function in the car package. The following is the result of the outlier test.

	rstudent	unadjusted p-value	Bonferroni p
9347	-13.328910	3.4909e-40	3.4891e-36
4434	-10.454754	1.8916e-25	1.8906e-21
8746	-8.718185	3.2819e-18	3.2803e-14
8244	-8.217786	2.3331e-16	2.3320e-12
9866	-7.741285	1.0806e-14	1.0801e-10
7424	-7.660796	2.0213e-14	2.0203e-10
9698	-7.494364	7.2337e-14	7.2300e-10
9151	-7.481093	7.9986e-14	7.9946e-10
9569	-7.473524	8.4698e-14	8.4656e-10
8134	-7.113858	1.2068e-12	1.2062e-08

Here, we see that there 10 outliers. Hence, we remove these data points from our training data set as they may affect our prediction results.

We should next check to see if there exists some subset of all all our predictors or variables that would give us a higher final *RMSLE* score than our current baseline score. Unfortunately for us, we have a lot of variables. We have 120 different variables/ predictors. Hence, doing a best subset selector is too time costly. As a result, we will be performing a greedy variable selection method known as forward selection. This method takes in to the consideration the variables that have already been chosen and picks a new variable that would best improve the performance of the model.



The graph above shows the results of the forward selection training process of our linear model. The number of predictor of the model with the total lowest *RMSLE* is 109. Using the model the corresponds to 109 different predictors rather than 120 yields us a new *RMSLE* score of **0.245445**

.

While the linear regression gives us decent predictive performance, it has no real mechanism of allowing us to know which variables are the most important in determining the price of a home. Hence, we introduce a subclass of models that will allow us to do this. These models are called regularized linear regression models.

Regularized Linear Regression

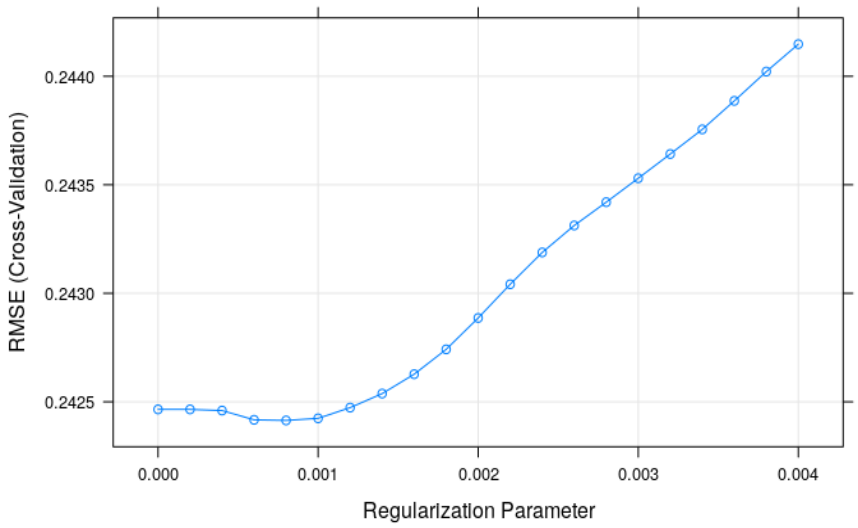
Regularized Linear Regression models as their name suggest provides some kind of regularization or penalty mechanism in addition to the standard linear regression solution. Usually the penalty is with regards to the coefficients of the model to prevent them from getting too large. In this report, we will present three different regularized regression models, Lasso(L1),, Ridge (L2) , and Elastic regression.

Lasso (L1) Regression

$$L(x,y) \equiv \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2 + \lambda \sum_{i=1}^{n} |\theta_i|$$

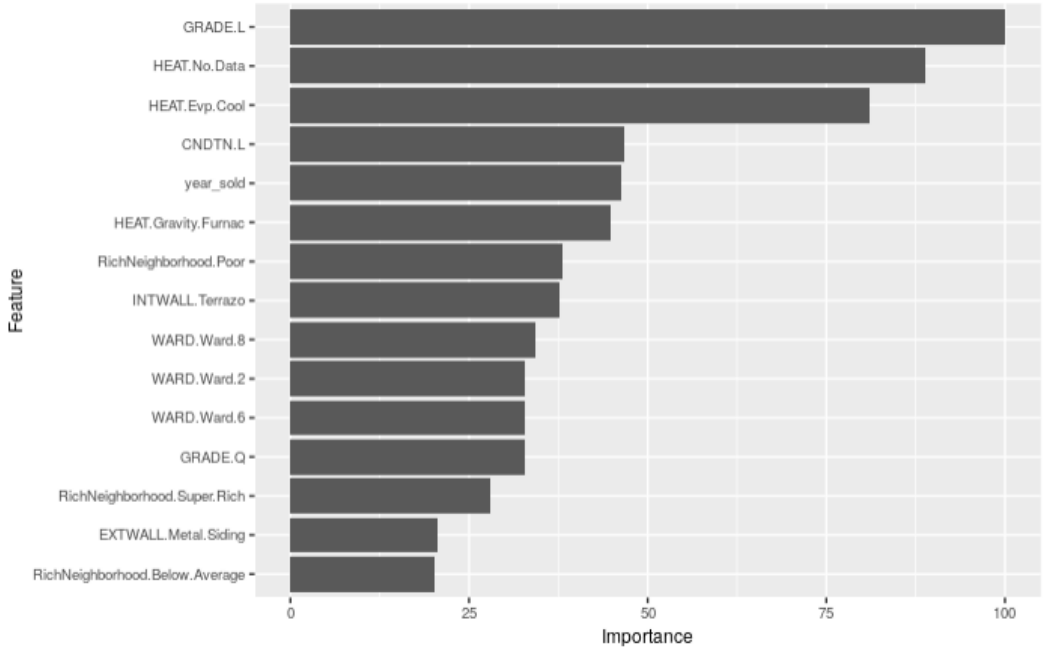
The formulation of Lasso regression model can be seen in the picture above, where lambda is the regularization coefficient. It is easy to see that as lambda goes to infinity, all coefficients, theta, go to 0.

Similarly, as lambda goes to 0, we have out original linear regression model. Hence, L1 regression is often thought of as an automatic variable selection method. Performing grid search on the parameter of lambda that minimizes *RMSLE*, we get an optimal value of *lambda* = 0.0008, which gives a final *RMSLE* of **0.2424749.**

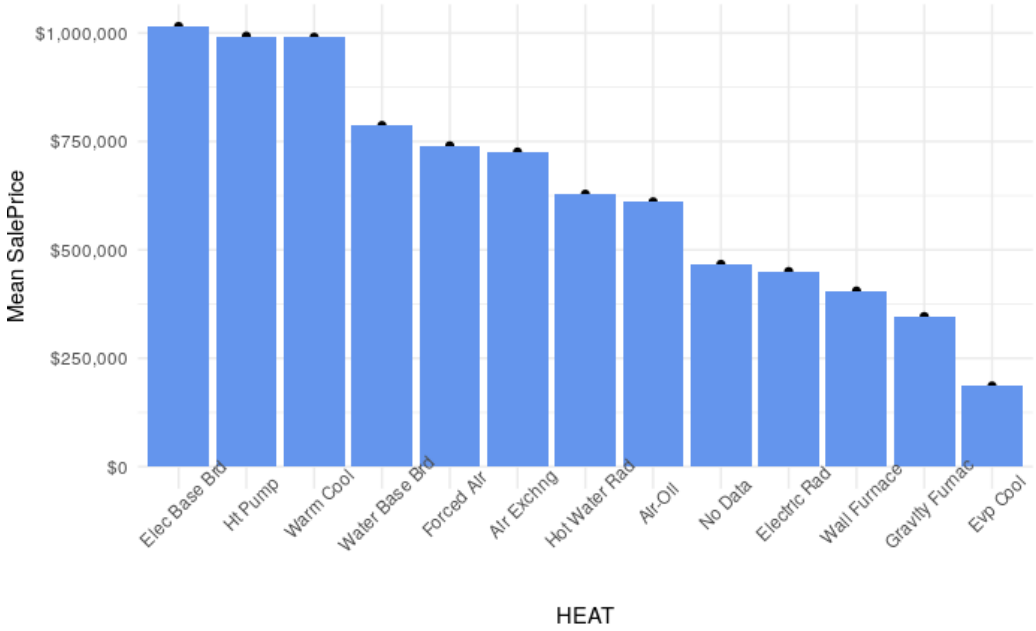


Variable Importance of Lasso

Using the lambda = 0.001 for our model gives 93 non zero coefficients, which means 28 coefficients were zeroed out. Furthermore, the chart below let's us see the most important feature according to the model.



The chart tell us that GRADE, HEAT, RichNeighboorhood, CNDTN, and year sold are the 5 most important variables. It is interesting that Heat:NoData is considered an important variable, but looking at the average housing prices per heating type, we can start to see why.

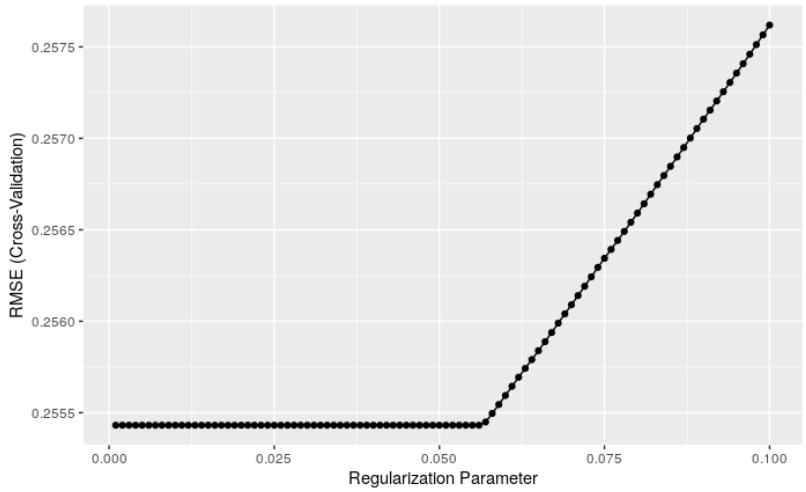


Having No Data, and Evp Cooling as your heating system is a strong indicator of cheaply priced homes. Hence, the important variables in the variable importance graph above are there not only to indicate expensive homes, but also to indicate homes on the other side of spectrum, (i.e. cheap homes).

Ridge (L2) Regression

$$L(x,y) \equiv \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2 + \lambda \sum_{i=1}^{n} \theta_i^2$$

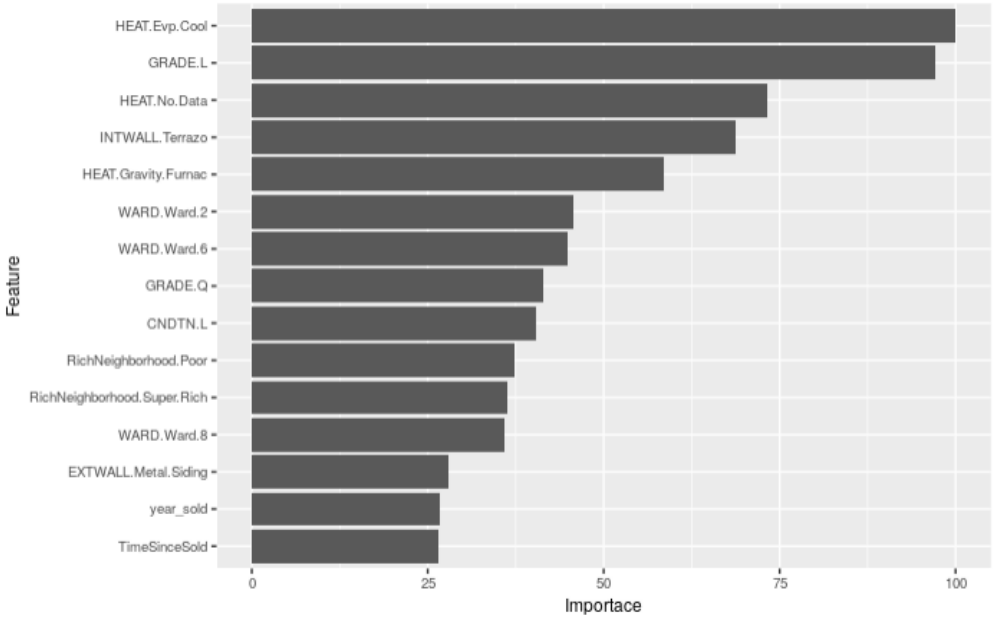
The formulation of Ridge regression model can be seen in the picture above, where lambda is the regularization coefficient. Similar to the L1 model, we also perform grid search for the value of lambda.



Our search leads us to a value of *lambda* = **0.056,** which gives a final *RMSLE* of **0.244699,** which is noticeably higher than the RMSLE of our Lasso Model. To understand why this is the case, we want to compare the regularization of effect of Lasso vs Ridge.

Looking at the coefficients of our final Ridge model, we discovered that none of the coefficients has a value of 0. Instead, there are plenty of coefficients with values close to 0. Hence, the resulting model formed by Ridge regression is much more complicated than the model formed from Lasso Regression. This may explain the poorer performance of Ridge on the testing data set.

Variable Importance of Ridge



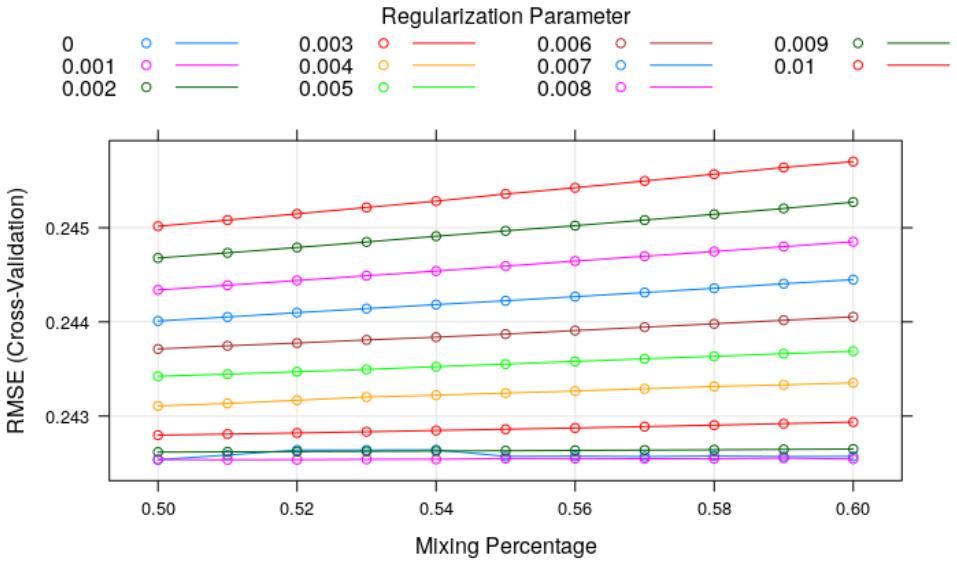
The variable importance chart in the previous page tell us that GRADE, HEAT, ROOF Composition, INTWAL, and WARD are the 5 most important variables. This confirms the important variables from our Lasso Model as the 2 charts are very similar.

Elastic Net (Hybrid L1 and L2 Regularization)

Elastic Net is model that lies somewhere between Lasso and Ridge. It combines the regularization terms of both Lasso and Ridge to try and find some kind of compromise between the two models. The exact formulation of Elastic Net is:

$$J(\boldsymbol{\beta}; \lambda, \alpha) = ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda \left(\alpha ||\boldsymbol{\beta}||_1 + \frac{1}{2}(1-\alpha)||\boldsymbol{\beta}||_2^2 \right)$$

Unlike Ridge, and Lasso, Elastic Net has 2 different hyper-parameters, alpha and lambda. This makes it much more difficult to tune. Here, we limit our search by only looking at values of alpha between 0.5 and 0.6, and values of lambda between 0 and 0.01.



Our best model has an *alpha* =**0.59**, and *lambda* = **0.001**. The resulting model has a *RMSLE* of **0.24253**, which is worse than our Lasso model. Similar to the Lasso model, Elastic net has 98 non-zero coefficients, and 23 zero coefficients. Now, we compare the variables whose coefficients have been zeroed-out by Elastic Net and Lasso Model.

On the table below we can see that both Elastic Net and Lasso have a similar list. Hence, the variables in common between the 2 models are comparatively less important, and next we try performing regression on all variables not in both the 2 tables below to see if we can get a better overall regression performance.

x
HEAT.Hot.Water.Rad
HEAT.Ht.Pump
HEAT.Wall.Furnace
HEAT.Water.Base.Brd
STYLE.L
STYLE.Q
STYLE.6
STYLE.7
STYLE.11
STYLE.13
GRADE.C
GRADE.4
GRADE.5
GRADE.6
GRADE.8
GRADE.11
CNDTN.4
EXTWALL.Brick.Siding
EXTWALL.Shingle
EXTWALL.Stone.Siding
LATITUDE
WARD.Ward.5

Variables with 0 as a coefficient in Elastic Net

x
HEAT.Electric.Rad
HEAT.Hot.Water.Rad
HEAT.Ht.Pump
HEAT.Wall.Furnace
HEAT.Water.Base.Brd
AYB
STORIES
STYLE.L
STYLE.Q
STYLE.4
STYLE.6
STYLE.7
STYLE.11
STYLE.13
GRADE.C
GRADE.4
GRADE.5
GRADE.6
GRADE.8
GRADE.11
EXTWALL.Brick.Siding
EXTWALL.Shingle
EXTWALL.Stone.Siding
ROOF.Shingle
ROOF.Wood..FS
INTWALL.Vinyl.Sheet
LATITUDE
WARD.Ward.5

Variables with 0 as a coefficient in Lasso

Subset Linear Regression

Performing a linear regression without the variables in both the 2 lists above, we get a final *RMSLE* of **0.2436351,** which is worse than our Lasso and Elastic Net scores, but better than our Ridge model.

Tree Based Models

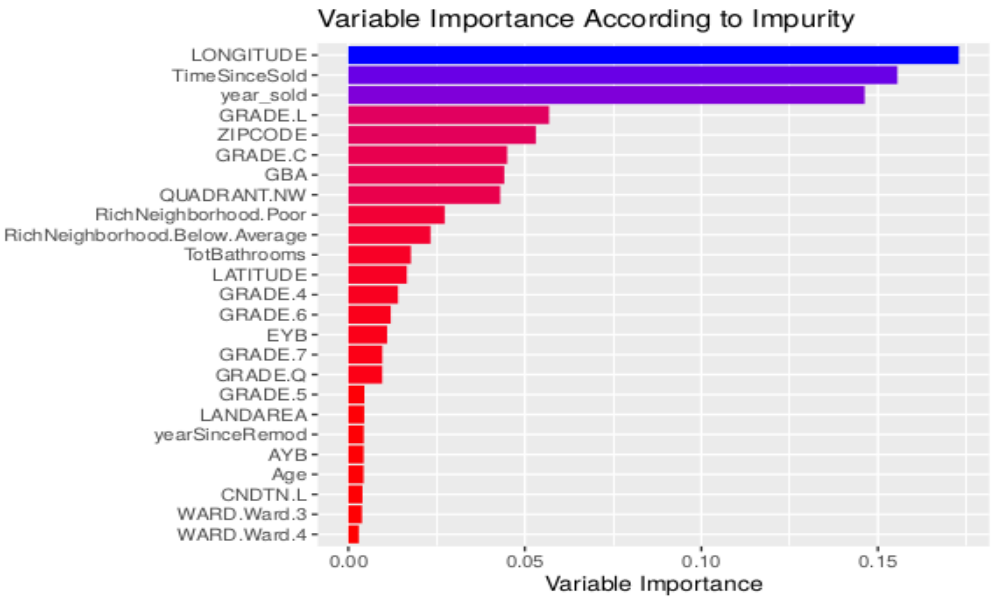
After exploring a variety of different linear regression based models, we will now move to tree based models. Specifically, we take a look at gradient boosted trees, and random forests. These powerful models, though more computationally expensive than linear regression, may hopefully allow us to gain better predictive accuracy.

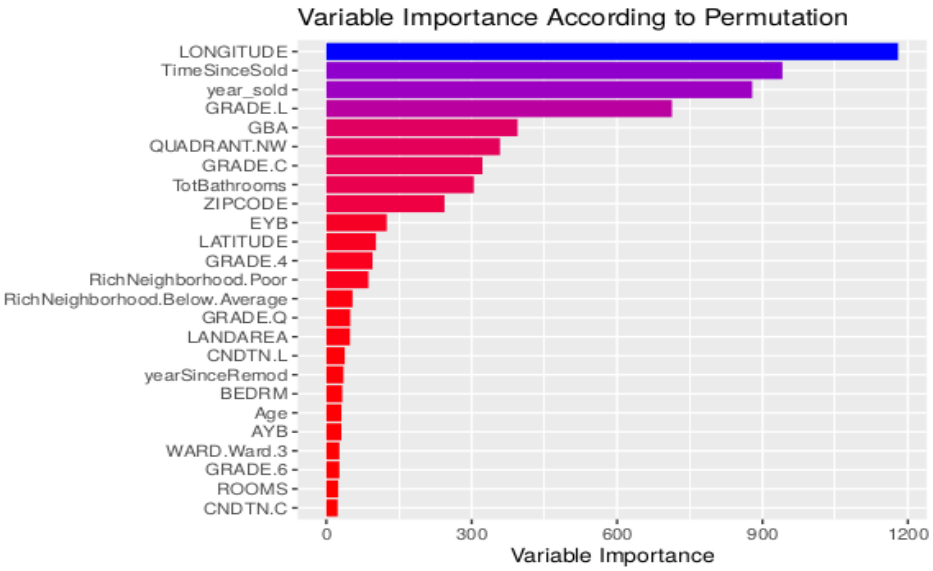
Random Forests

A random forest consists of multiple random decision trees. There are two types of randomness inherent in a random forest. First, each decision tree of the forest is constructed from a random sample of the original data, using a process known as bootstrapping. And secondly, each tree node uses a random subset of features their best split. For running the random fores based models on our data set, we will be using a fast C++ based package, called Ranger.

There are three different hyperparameters available for tuning in random forests. They are *mtry, splitrule,* and *min.node.size.* The full grid search can be found in the appendix, but here we will simply be presenting the best hyper-parameters. The best set of hyper-parameters values are *mtry* = 44, *splitrule* = "variance ", and *min.nodes.size* =3. The final *RMSLE* of the model is **0.1950247.** This is significantly better than the *RMSLE* of all our linear regression-based models.

Now, we look at the important variables in the random forest according to 2 measures impurity, and permutation.





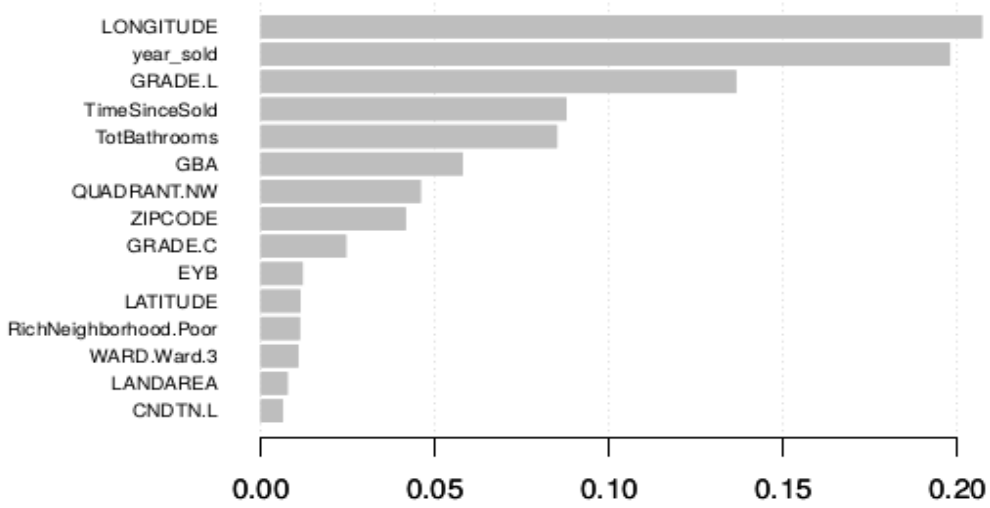
For regression tasks, impurity refers to variance. Hence, it seems *LONGITUDE, TimeSinceSold*, and *year_sold* contribute to the greatest reduction in variance among all variables in the data set. On the other hand permutation refers the increase in the model's prediction error after permuting the feature. The idea behind this measure is that a feature is "important" if shuffling its values increases the model error, because in this case the model relied on the feature for the prediction. Similar to the impurity measure, *LONGITUDE, TimeSinceSold*, and *year_sold* are our most important variables.

Gradient Boosting Trees

Gradient Boosting trains many models in a gradual, additive and sequential manner. For every iteration, each tree or model is updated in a way that helps them improve according to some specified loss function. For running gradient boosted tree models on our data set, we will be using a fast C++ based package, called xgboost.

Compared to random forests, there There are three different hyperparameters available for tuning in random forests, there are 6 different hyperparameters we can optimize. They are *eta, gamma, max_depth, min_child_weight, subsample, and colsample_bytree.* The full grid search can be found in the appendix, but here we will simply be presenting the best hyper-parameters. The best set of hyper-parameters values are e*ta=0.025, gamma=0.05, max_depth=7, min_child_weight=5, subsample=1, and colsample_bytree=0.4.* The final *RMSLE* of the model is **0.174768.** This is significantly better than the *RMSLE* of all our other models.

Now, we look at the importance of variables according to the gradient boosted tree model.



Neural Network Model

The last model we consider is a neural network-based model. The parameter of the best model after a very brief grid search is decay = 0.5, and size =8. The final *RMSLE* of the model is **0.181845.**

V. Summary of Finding

Model Performances

	Score (RMSLE)	Caveats and Important Notes
Full Linear Regression	0.2510709	Simple to Train, Very Quick Training Time
Forward Linear Regression	0.245445	Simple to Train, Very Quick Training Time
Lasso	0.2424749	Middle Complexity to Train (1 hyperparam), Very Quick Training Time
Ridge	0.244699	Middle Complexity to Train (1 hyperparam), Very Quick Training Time
Elastic	0.2425312	Complex to Train (2 hyperparam), Very Quick Training Time
Subset Linear Regression	0.2436351	Simple to Train, Very Quick Training Time
Random Forest (ranger)	0.1950247	Complex to Train (3 hyperpameters), Medium training time (Multiple cpu cores needed)
GBM (Xgboost)	0.1747683	Really Complex to Train (5 hyperpameters), Long training time (Multiple cpu cores needed)
Neural Network	0.181845	Complex to Train (2 hyperpameters) , Longest Training Time by far (No fast implementation)

Variable Importance

By looking at all the variable importance charts in the graphs, we can list down in this section the most relevant variables to look at in determining price. In this list, we do not separate out our factor variables, but rather simply mention the whole variable group. Furthermore, this list is not necessarily in order, but rather tries to combine all the important variables as reported by the different models.

1. *LONGTITUDE*
2. *TimeSinceSold*
3. *GRADE*
4. *TotBathrooms*
5. *year_sold*
6. *HEAT*
7. I*NT_WALL*
8. *GBA*
9. *CNDTL*
10. *RichNeighborhood*

VI. Conclusions

In this report, we performed an in-depth analysis and exploration of the housing dataset S440. In this report, we performed various preprocessing steps, feature extractions, and transformations on the data set to make it easier to conduct our statistical analysis, and perform model building. We specifically focused on three kinds of models in this report, Linear Regression models, and its variants, such as Lasso, Ridge, etc., Tree-Based Models, such as random forests, and lastly neural networks. Through our analysis, we see that the best model for this data set is a gradient boosting trees based model, implemented via the package '*xgboost*'. We also create a list of the most important variables in determining a house's price. Future work can focus on potentially ensembling all the different models to see whether the models can complement one another in their predictions