Predicting home prices in Ames, Iowa using 7 machine learning algorithms

# Introduction

Home owners and home buyers often must deal with a lot of unknowns in trying to access the true value of a property. By demystifying the true cost of a home, we can bring transparency to house pricing negotiations and get a better estimation of how much a home should cost. Additionally, we can find the home features that lead to the highest prices. The most useful result of predicting home prices is that it allows for developers and even municipal planning committees to plan home building projects or help in zoning activities. The dataset I would like to examine for the capstone project will be the Ames Housing Dataset. The data set contains 79 explanatory variables describing numerous aspects of residential homes in Ames, Iowa in the United States in their relation to the final sale price of the home.

With this data set I aim to explore advanced regression and machine learning models (ols, random forest, XGboost, regularized net, ridge and lasso regressions) to best predict the true price of a home based on these variables. I aim to find the best mix of features for each of these models and then average the results of the best models to obtain the highest level of model accuracy, in predicting home prices based on a set of features.

# Literature Review

I looked at a few resources when approaching this problem. Firstly, I examined “Study of advanced regression models to determine prices for houses in Ames, Iowa based on their features” (Menkudale & Taluja 2016). Using the same dataset as me, they ran (after removing outliers and converting categorical variables into dummy variables) a simple OLS regression model on the dataset and the full list of 79 variables. They then compared that to the RMSE value after running a ridge and then lasso regression on the dataset. With ridge, they found a reduction in the RMSE value, with lasso they found a slight increase in RMSE value but with faster computation time.

Next, I looked at the paper “Feature Selection and Predictive Housing Data Using Random Forest” (Rai 2017). Rai uses the same dataset as myself, using a Boruta wrapper algorithm (adding randomness by creating shadow copies, running random forest declassifier on the expanded dataset, testing for highest mean values, and finally, iteratively discarding variables with means of shadow copy significantly smaller than the means of the feature copy) for feature selection. He reduces the list of features in the data set to 49 which are then used to develop predictive random forest models. He successfully increases his model accuracy (reduced r-square and RMSE) using this method of feature selection over using 57 variables that are extracted from the rough fix portion of the feature selection. This was useful because it introduced me to a method of feature selection that could possibly be better for my dataset over other methods.

Next I examined a few Kaggle Kernels that showcased useful techniques in visualizing and running modelling the data set. They include kernels by Nandy Singh, Stephanie Kirmer and Tanner Carbonati. I also looked at” Seven Techniques for Dimensionality Reduction” by Adae et all (2014) to see methods that could be used to reduce dimensionality.

Finally, I looked at “Advanced Regression Modeling on House Prices” (Jager & Yue 2016). Again, using the Ames, Iowa housing data set they ran GLM, Neural Network, Random Forest and Gradient Boosted Tree regressions. Seeing the lowest RMSE for Random Forest. However, to increase the models’ predictive capability the authors tried to stack their model neurally (Running k fold cross validations on each technique and combining the cross validated predicted values into one matrix that runs on a meta learning model). They found that their r square value didn’t improve substantially. They then tried to average their model by averaging out their predictions and placing the most weight on the models that preformed the best. This significantly improved their model accuracy.

# Dataset

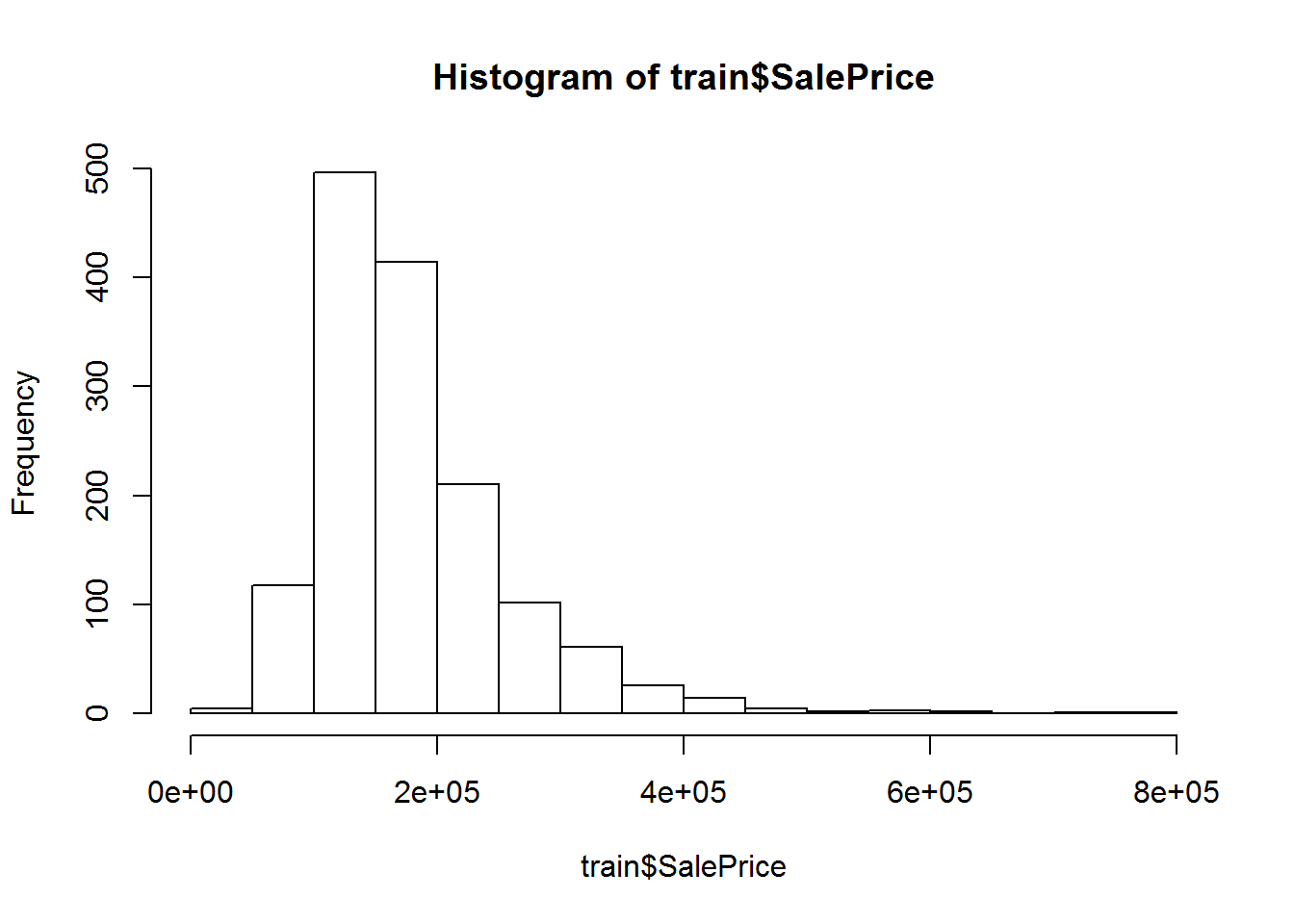
The Ames Housing dataset was found at the Kaggle website. It lists the residential properties that were sold in Ames, Iowa between 2006 to 2010. The dataset has three files, a train, a test set and a final submission test set.

The train file has 1460 observations, this set contains the response variable, which is the sale price of the home. This is what we will predict in the test set.

The test file has 1459 observations. They both have 79 explanatory variables 46 categorical and 33 continuous variables that describe house features such as neighborhood, square footage, number of full bathrooms, and other types of information the typical home buyer wants to know in a potential property purchase.

The final submission test set is a supplement to the test set and contains the true Sale Price of the homes in the test set by the home ID. This dataset will be merged with the test set to test how well our models did in predicting the sale price against the true Sale Price of the test set.

Figure 1: Histogram of Sale Price



When we explore our train data-set we find that the sale price of homes is heavily skewed towards the right (Figure 1). Meaning there are a quite a few homes (n=10) that have been sold for over $500,000. We will choose to normalize our sale price column with a log transformation. This will be necessary to obtain a good prediction especially when we construct linear regression models. We also see in the dataset that there are quite a few missing variables (nearly 10000 NAs). There also appears to be a lot of multicollinearity in the dataset (Pool Area strongly correlated with Pool quality for example). And there is a lot of mixed column structures (46 categorical and 33 continuous variables), we will have to deal with this because out machine learning algorithms will mostly only take numerical inputs.

# Approach & Results

As we identified above, the data set has a lot of variables. Before we can apply machine learning algorithms or regression techniques it’s important that we clean our dataset and pick the right set of features.

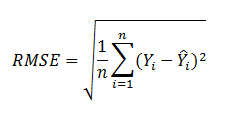
In this dataset, we are dealing with 6 issues that need to be addressed before we can begin applying regression/machine learning models to our data.

The six issues are:

1. Large number of NA values
2. A mix of character and numerical variables
3. Correlations between variables
4. Some outliers
5. Skewed data
6. Low variance variables.

I will outline how we tried to address these issues before applying our models below. Once the data was cleaned I applied 6 models (7th as an ensemble) on this dataset. I tested the model against the training test, which I partitioned using a 10-fold cross validation. I used the RMSE score as the benchmark of how well the model performed. I then tested the model against the actual test set that had the unobserved Sale Price. To create the test set I merged the sample submission dataset (that had the true Sale’s Price of the test set) with our test set to get the sale price of those homes which were unobserved earlier. The prediction again was compared against the actual Sale Price using RMSE as a benchmark (Figure 2).

Figure 2: RMSE Calculation where Y is actual and Yhat is predicted



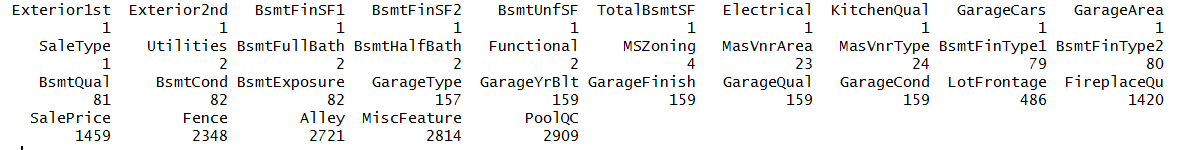
The six models I tested were:

1. Simple Linear OLS model
2. Random Forest
3. XGBoost Gradient Boosted Tree
4. Ridge Regression
5. Lasso Regression
6. Elastic Net Regression

The final model I ran (seventh) was an ensemble model that combined the last three models together averaging their results.

## Step 1: Cleaning up NAs

After joining the training and testing data sets we want to explore whether there are any NA values. When we run our code to see NA values we find the following below.



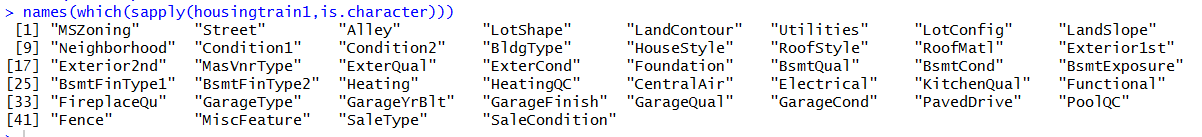
To build any predictive model, we’ll clean our data by filling in all NA’s with appropriate values. For each column, I tried to replace NA’s through two ways. I cross checked each variable to see if any NA’s represented the absence of that feature in the house. For example, certain homes didn’t have Pools. To find out which PoolQC NA values corresponded with homes that didn’t have pools I used homes that had a pool area of 0. I then filled in those NA’s with “No Pool”

The second way I removed NAs was to use values that most closely corresponded to that feature. For example, if most pools in a neighborhood of the home had a pool quality of excellent. I decided to allocate the NA pool quality value in this home as excellent as well. As its most likely to be the case. In our RMD file we can see the steps taken to clear the NAs in each of these 35 columns under steps labelled with #1.

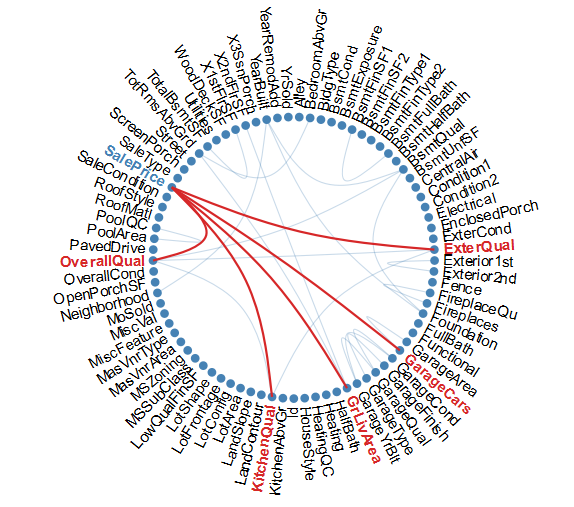
## Step 2: Converting Character values to Numerical

After filling in our NA values. I isolated which character values needed to be converted to numerical (figure below). I needed to convert them to numeric because most of the models we want to run on the dataset can only understand numerical inputs. This is particularly true for regressions.

Now to do this we must convert these into values that correspond to the possible effects each category had on the price. For example, if we converted a PoolQC (Pool quality) of good to the number 3 and a pool quality of excellent to the number 4. Our future model should be able to pick up that this increase in one does not correspond to a single numerical point increase but to another level of quality.

To do this I utilized the fact that these character levels would be strongly correlated with sales price. In the example above, a pool with excellent quality would have a stronger impact on price than would a low-quality pool. I summarized each of these 44-character columns by the effect each of its values had on the final Sales Price and allocated it a numerical value. Starting from 0 onwards based on the value of that feature in reference to the mean Sale Price associated with it. For example 0 was assigned for no feature existing there (for example a home with no pool would be assigned a 0) ascending in numeric value by the mean sales price associated with that value. In our RMD file we can see the steps taken to convert the character vectors in numeric ones under #2

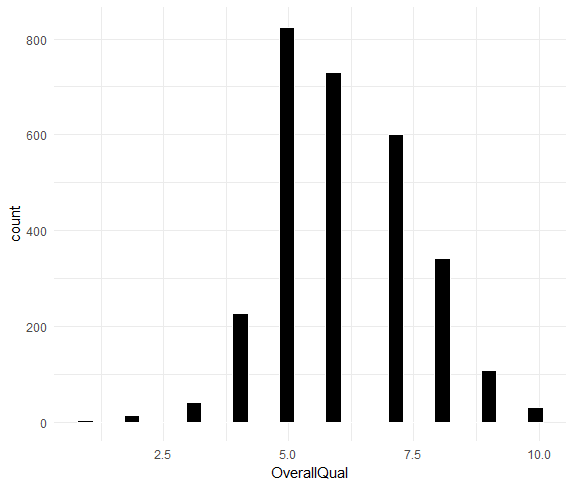
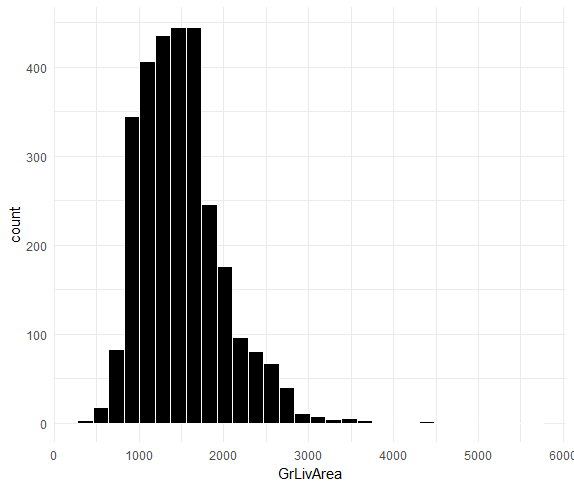
## Step 3: Examining Correlations in our Dataset with Sales Price

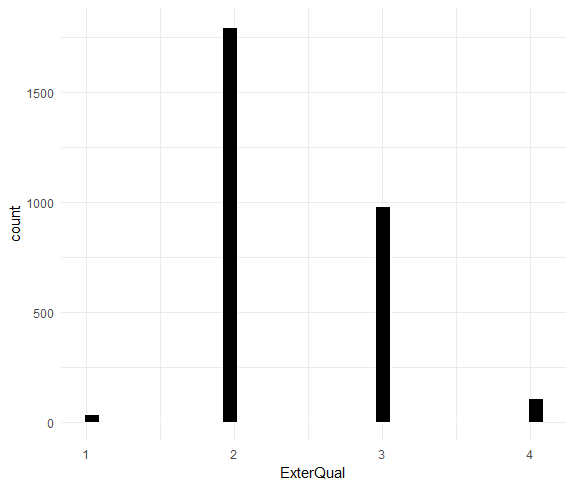
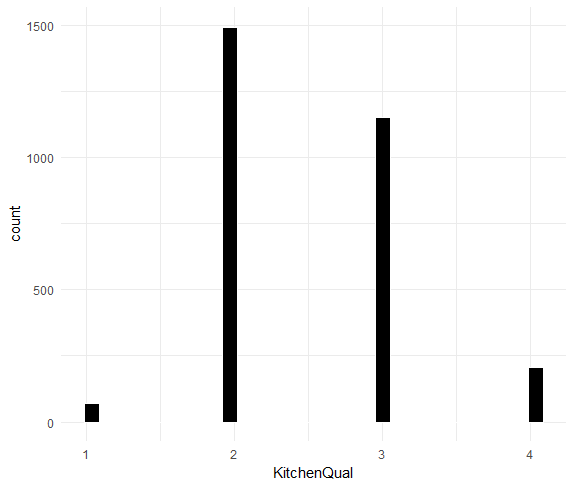


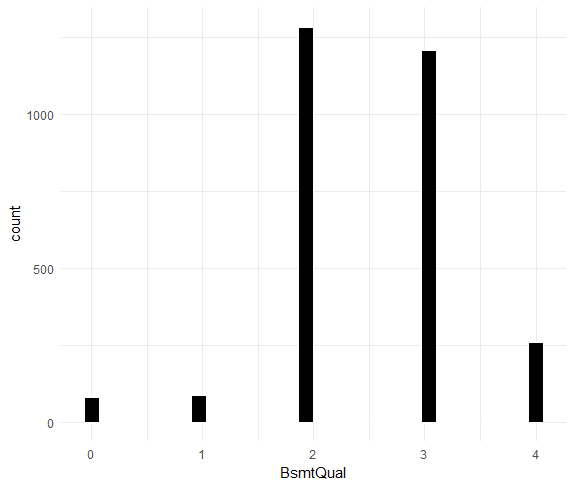
After transforming all our variables I checked to find the top 5 (0.65 threshold) variables associated with Sale Price of a home. In our RMD file we can find the code under #3. The top 5 variables were the Overall Quality, External Quality, Kitchen Quality, Ground Floor Square Feet and Number of Cars in the Garage. Any outliers in here would impact the price significantly so we will look for them in Step 4.

## Step 4: Dropping Outliers

Below we have plotted the histograms of these five variables.







## I found that Ground Floor Living Area has 2 large outliers (over 40000 sq feet). Upon further investigation we find that these rows also have a very low Sale Price. Now these will significantly affect our model’s ability to predict the Sale’s Price particularly in a regression model so we should drop the few data points that correspond to this oddity in our dataset. No other outliers were identified. The code for this is in our RMD file under #4.

## Step 5: Normalizing Data for Regression

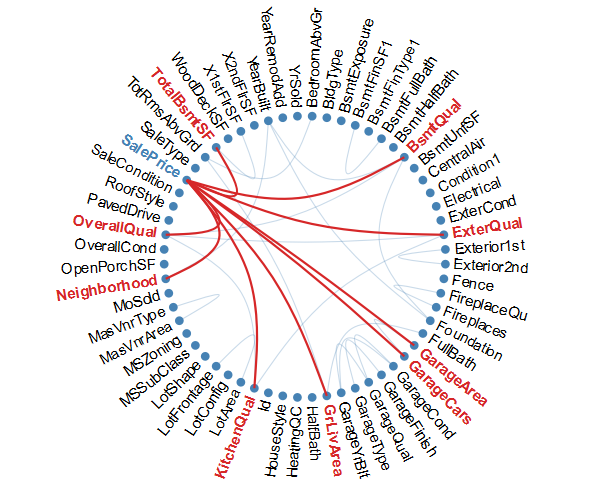
Again since regression algorithms assume that all variables are normally distributed. To deal with that I found variables with a skewness greater than 0.75. The skewed data was then normalized using log (x+1). The RMD file under #5 shows us the method we used to accomplish this.

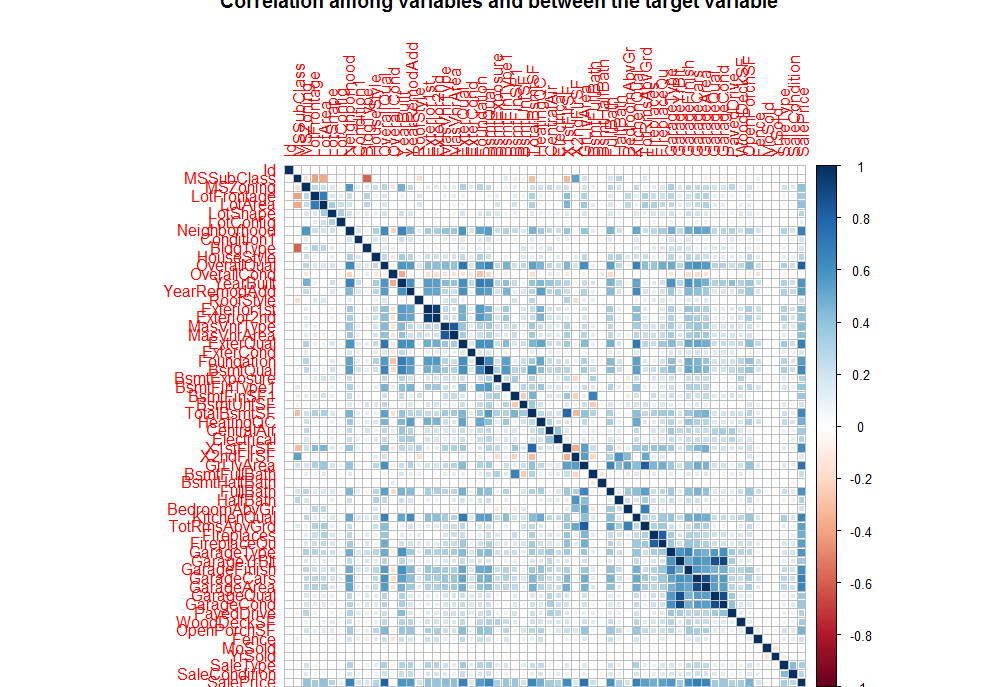
## Step 6: Removing Low Variance Variables

Next, I removed variables with near zero variance in our dataset (under #6 in our RMD file). These variables have little to no predictive influence on our dataset. It was a good idea to remove them as these variables may have caused overfitting or an incorrect model generalization.

I then examined correlations again in this data set. Interestingly we see a jump in the number of variables associated with Sale Price after we sufficiently cleaned up our dataset. We see now some interesting additions to our correlations. Four new additions are added at the same threshold we held last time (0.65 correlation). We see Total Basement Square footage, Neighborhood, Garage Area and Basement Quality now seem to influence our Sale Price too.

Because we had limited time we couldn’t take full advantage of this step. Other than showing us the relevant factors that influence Sale’s Price. I could have used a PCA analysis to try and create a new set of variables that would bind like variables together in relation to their correlation with each PCA components. This would have helped us create a restricted regression that would have allowed for reduced overfitting, multicollinearity and a potentially lower RMSE score.



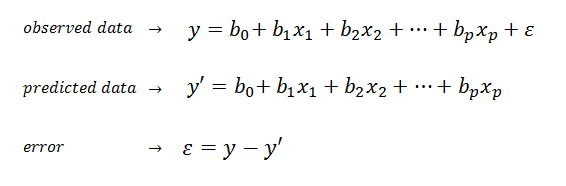


## Step 7: Running 6 Models

After we spent all this time cleaning and preparing our dataset. I could now build and run our models on this data and find the models with the best predicative power.

1. Linear Model:

The first model I ran was a simple OLS regression model (RMD #7). OLS creates a linear model that minimizes the squared error terms (or deviations from the predicted line) it uses the following formula to obtain the line



As stated earlier I partitioned our training data to build and internally test the model first. I got a very good RMSE of 0.11698. We then, to minimize against overfitting, can chose to switch to using K Cross Validated testing.

I used 10-fold partitions. Our data is split into 10 folds, then our model is trained on the 9 partitions. The test error is predicted on the 10th. This process escalates from 1 to 10 partitions and averages the results. This provides a robust way to guard against over fitting in our training data.

I used 10-fold partition throughout our model building attempts, for the sake of consistency, after this. The RMSE using this method was the same so we have some confidence overfitting at least within the training set was already minimized.

Next, we ran our model on the test set (which remember had the unseen Sale’s Price which was then merged into the test dataset) and our RMSE score shot up dramatically to 0.3889. This means that our model had limited predictive power in its ability extrapolate unseen prices and that we may have over fit our data in the training set.

2 Random Forest (RMD #8)

The second model I ran was a Random Forest again using 10-fold cross validation. Random Forest generates multiple decision trees based on the training set and takes the mean prediction value of each of the trees combined as an output. I used one mtry value to begin with. Mtry means the number of variables that are randomly sampled as a candidate at each decision tree split in the data. So, in this case we started with one as this is the default.

The result of our random forest on our split training data led to a larger but still relatively good RMSE score compared to our linear model of 0.12952. I then again ran the predictive values against the test data sale price and we interestingly even though our RMSE score was higher in the train group compared to the linear model our test RMSE score went down by almost 0.04 points to reach 0.35289. Even though our model still had a limited ability to extrapolate and predict prices I found it interesting that the random forest was better able to extrapolate than the linear model, this could have been down to the random forest’s ability to do its own feature selection and find the mix of variables that provide the best mean prediction from each of the trees.

To see if we could increase the model’s ability to predict prices I tried tuning some parameters of the random forest. I tried using 2 and then 3 mtry values in the model building stage. This increased the number of randomly sampled variables from each decision tree. In both these cases our Train and Test RMSE score went up. With 3 mtry values for example our Train RMSE was 0.131555 and the Test went up to 0.363475. We know then that the default 1 mtry, the model assumes, is perhaps the best in its ability to predict the sale price from our data set.

The next thing I tried was to select the top 20 explanatory variables from our dataset. Though we didn’t have time to perform a PCA. This method was somewhat like a PCA analysis in that it finds the top 20 features that can explain close to 90% of the variance in our dataset. Of course, it was different in that we couldn’t join variables that had correlations with the PCA components. But it did provide a good mix of variables that best explained the total variance of our data. However, what we find is like the one to three mtry parameter tuning when it came to the testing data. We saw a large decrease in our train RMSE to 0.05929. This means (as perhaps expected) we have explained the variance quite well in the training data and that it gives us a very strong prediction on the partioned data. However again we see the same issue as before, our model has a hard time extrapolating unseen Sale’s Price data in the test set. Our test RMSE score was very slightly higher, in the top 20 features, compared to the simple random forest we ran at the beginning. Our test RMSE score is 0.359501.

3. XGBoost. (RMD#9)

Seeing that our models haven’t performed any better we tried to move to using a form of stochastic gradient boosting machine that aggressively guards against overfitting (a refined version of random forest), called XGBoost.

I spent more time understanding this model than the rest of the other models so far because it promised the best result. It’s an ensemble learning method that generates large number of decision trees, and combines them into one final prediction. For gradient boosted trees, the new trees added to the model are the weak learnings which means that they follow this formula



Where F(x) is the model at t-1 time and h(x) is the new tree that will be added to the model. This is done over many rounds. The objective function F(x) will also be minimized at time t by the XGBoost.



L(Ft) is the loss function and the horseshoe (Ft) is the regularized function

It should in theory perform better on unseen data of the test set by guarding against overfitting through the concept of regularization. It guards against overfitting in the individual decision trees generated from the training data. XGBoost calculates the regularized function through the function below. W is the score of the jth tree leaf. T is the number of leaves in the tree and Y is the tree size penalty. **It** fits these shallow regression trees to our data and then additional trees to the residuals.



However, when I ran this tree I was quite surprised to find that our RMSE score increased for both our train and testing data. Our train RMSE increased to 0.14405 and our testing RMSE increased to 0.37841. Which was by far the least powerful prediction yet. Because XGBoosts can be tuned, perhaps this model, required further parameter tuning to achieve a better score that lives up to the hype around it. Because the gap between our training and test score are higher than before, this means our model is too complex and that we are indeed still overfitting the data. If we had more time we may have been able to tweak our model to better guard against overfitting and increased bias over random forest (higher RMSE score).

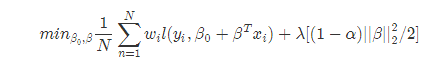
However, I suspected on thing, the unseen Sale’s Price we are seeing in the test set is outside of the range of prices given in our training set. When we test on the training set our RMSE score is relatively respectable hovering between 0.12 and 0.14. When we apply our model to the test set’s range of prices however, we see a much higher RMSE of 0.35 to 0.38. This happened to all our models. This is because the models we used so far generally perform poorly in their ability to extrapolate from unseen ranges. Regression models are better at this but in our linear model we didn’t see much of a difference. This was because lots of variables in our data had some multicollinearity as we saw in the correlation matrix.

4. Ridge-Lasso-Net Ensemble Regression Model. (RMD #10)

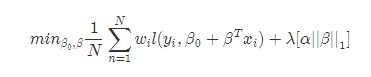
Because of this reason I decided to use a regression technique called Ridge Lasso Net Regression. The combination of lasso and ridge regression helps to prevent overfitting and has a good ability to extrapolate prices outside of our training set.

Lasso (L2) regression penalizes the number of non-zero coefficients by picking one set of correlated predictors with a coefficient and discards the others. It’s a parameter that penalizes fitting too many variables. It allows the shrinkage of variable coefficients to 0. This results in variables having little to no effect. It reduces the total dimensionality of the dataset. Reducing dimensionality may increase prediction accuracy, but if, it is reduced too much we may have a high bias appearing in our model as lots of coefficient begin to drop away.

Ridge (L1) penalizes the absolute magnitude of each coefficient by shrinking the coefficients of correlated predictors towards each other. By adding the penalties of both these regressions we get a constraint on the coefficients of our model and this helps to diminish the effect of collinearity in our regression model. The formula for ridge is below, when alpha is 0 we get our complete formula for ridge.



The formula for Lasso is below this time when our alpha is 1 we get the complete lasso formula.



Next, we use our elastic-net regularized regression. The formula for which is below. Elastic-net allows us to leverage the gap between ridge and lasso. In all these formula L(y,n) is the negative log likelihood at each observation (i).  is the shrinkage parameter. As it increases from 0 (no shrinkage) our coefficients are shrunk together at larger and larger rate. Our net regularized formula allows us to leverage the gap between ridge and lasso (L1 and L2) through alpha which is important because we don’t want to reduce the dimensionality too much with Lasso as it can lower the model’s robustness and introduce too much bias. We will pick a shrinkage parameter in our model that gives us the smallest error rate. This will give us the most predictive power in our model.



With this model, we got the best predictive power both in the test set and training set. Our train RMSE score for the ridge regression were 0.0618, for lasso it was 0.0125 and for net it was 0.0424. Which are all very significant decreases. When we applied these models to our test data we got for ridge 0.2038, Lasso 0.0036 (a very strong prediction) and for net-elastic 0.2033. We got a very strong prediction in the test set of dropping almost 0.13 points for ridge and elastic net. For lasso, we see the strongest drop in RMSE for which I initially got quite excited. Lasso alone seems to be the best model until we notice that the model performs better on the test set than the training set. This means that the Lasso regression has a large amount of bias in the model (perhaps because of the large multicollinearity between variables and lots of constants are dropped) so we should consider this in the bias variance tradeoff before saying Lasso is the best model. The elastic net balances between ridge and lasso and allows us to deal with the multicollinearity much better.

For the final step, I took, was to create an ensemble model. This also has produced a lot of hype amongst machine learning enthusiasts and it made sense for us to try it in this case. I created an ensemble model of our three strongest models (Lasso, Ridge, Net-Elastic) averaging the results to get the strongest and most robust prediction set. The RMSE score on the test set came out to 0.1358225. This was a very good RMSE and proved that the ensemble model is a robust predictive model to use for our housing dataset.

To summarize. Below is a summary of all the models used and the RMSE scores associated with the training and test set.

|  |  |  |
| --- | --- | --- |
| **Model Used** | **Cross Validated Train RMSE** | **Cross Validated Test RMSE** |
| 1. Linear OLS Model | 0.1170 | 0.3889 |
| 1. Random Forest | 0.1295 | 0.3528 |
| 1. Random Forest with 3 mtry | 0.1315 | 0.3634 |
| 1. Random Forest Top 20 Features | 0.0592 | 0.3595 |
| 1. XGBoost | 0.1441 | 0.3784 |
| 1. Ridge Regression | 0.0618 | 0.2038 |
| 1. Lasso Regression | 0.0124 | 0.0004 |
| 1. Net-Elastic Regression | 0.0617 | 0.2033 |
| 1. Ensemble Model | 0.0424 | 0.1358 |

# Conclusions

In this project, I was able to explore various machine learning and regression algorithms in an attempt to understand how to predict home prices. We started our project to better understand how home features affect the final price of a home. The value proposition behind this exercise is that it can help municipal planning committees, housing developers and real estate agencies better allocate resources and plan zoning activities in housing areas. Our data contained close to 3000 homes and 80 variables. We had to fix the many NAs in the data set. Transform variable structures. Drop low variance predictors and normalize/scale our data. We were then able to run 7 models on the dataset. We found that when we ran our models on our train data our RMSE scores were respectable, however, when we ran them on the test data our RMSE scores shot up dramatically. That meant we overfit our training set. Our final three models attempted to address this by running regressions that dealt with multicollinearity and the ability to extrapolate beyond the range of data seen in the training set. These final three models were made into an ensemble model and that allowed us to obtain a predictive model that was robust and able to predict across both training and testing datasets.

Some things I found interesting in this exercise was the fact that our machine learning algorithms like XGBoost couldn’t be robust across training and testing sets. If we had more time we could have spent more time experimenting with our parameters to obtain a better prediction in these algorithms. If we had more time we could have also done a PCA analysis and created grouped variables based on their correlation with the PCA components to cut down on multicollinearity and created a better set of features for our models to run on. I also learned that feature selection is a big part of running models. Cleaning and selecting our data is a very time-consuming process and choosing the correct set of parameters for any one model is a complex task. This truly allowed me to get a good sense of running an end to end data science prediction process and the time required to run one with a good measure of success.