WELCOME TO AMES, IOWA

House Prices Prediction Model

Abstract

This research aims at predicting prices of residential homes in Ames, Iowa based on a large set of features. It starts by acquiring and defining input data, data cleaning, feature engineering, and finally by model training, selection, and prediction. The research predicts house prices by comparing the results of several predictive models and selecting the most accurate modeling option to predict the outcome

Gayathri Movva, Giri Nanduru, Pedro Ortiz, Hassaan Saleh, Umar Salam
George Mason University

Table of Contents

Executive Summary

The goal of this project is to use exploratory data analysis (EDA), visualization, data cleaning, preprocessing and feature engineering, and apply linear, nonlinear, and tree-based models to predict home prices, given the features of the home. The dataset originated from the effort of a group of students which undertook the task of updating the assessment model used by Ames City, Iowa Assessor's Office in 2011 (DeCock, 2011).

To understand the intrinsic value of a home, a prospective buyer or investor must investigate those characteristics that make the housing prices fluctuate. According to a real estate website, Homelight.com, factors such as total square footage, number of bathrooms, and year built, play a crucial role when comparing home values (Barsch, 2019).

This report includes a subset of a subset of the original dataset introduced by Kaggle in 2016 (Kaggle, 2016). This dataset consists of 1,460 observations with 79 explanatory variables describing every aspect of residential homes in Ames, IA. After performing EDA, with 36 numerical variables and 43 categorical variables with many missing values, the need for preprocessing steps such as data transformation and feature engineering became apparent.

Once these were completed, our study group applied a total of fifteen models involving eighteen data frames created in R and evaluated the results. Multiple linear regression (Ordinary Least Squares) and Gradient Boost Machine (GBM) models performed the best overall. We applied the OLS and GBM models to the Kaggle test dataset where our predictions were very close to similarly configured properties from the training data. In the end, the GBM model was selected as the preferred model because of overall better predictive accuracy. We found features like overall quality, lot area, age, usable living area, garage capacity, number of full bathrooms, fireplaces, and foundation type are some of the most important predictors.

Data Preprocessing/Exploratory Data Analysis

The House Prices dataset used in this report was sourced from Kaggle. The Training set has 1460 samples with 79 predictors and a sale price target variable. And the Test set has 1459 samples with 79 predictors without the response variable. There is also a metadata file available with the description of the data fields used. Here is the link House Prices Dataset available on Kaggle. Here is the excerpt of the Training Dataset in Appendix A.

After importing and analyzing the dataset in R, our team used a comprehensive strategy to preprocess the data for visualizations and model building. The following figure 1 provides the steps performed for preparing the data.

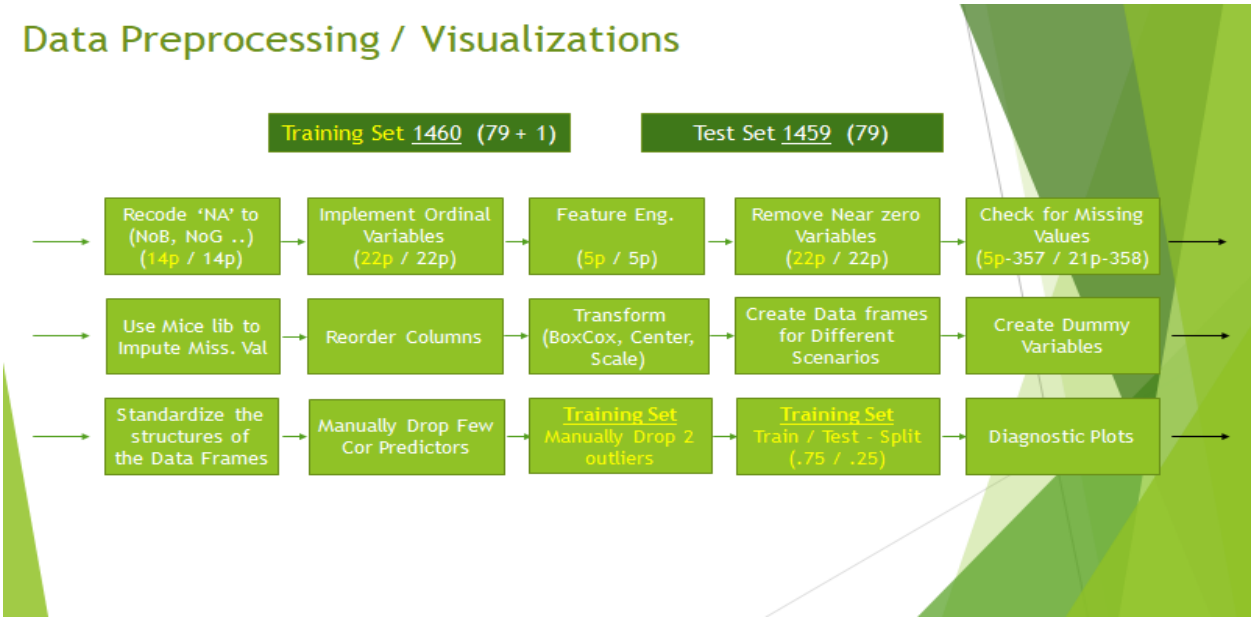

Figure 1. Data Preprocessing Steps

First, R treats "NA" values as missing. There were 14 predictors found with "NA" values which were not actually missing. These "NA" values were **recoded** to relevant values. Next, there were 22 predictors (eg- ExterQual, KitchenQual) which were implemented as **ordinal variables**.

In real life scenario when we go purchase a house we look for different attributes eg – House Age, Square feet, House Improvements, Bathrooms etc. Next logical step was to **create new features** to enrich the dataset for better predictive power. There were total of 4 bathroom variables which could be combined into a single predictor. A new feature Remod was created to indicate if there were any improvements done to the house. It also makes sense to add a feature to indicate the age of the house. We also create a feature to indicate if the house is new or old and a feature for the TotalsqFeet. In the figure 2 below, you can see the definition of 5 new features.

Feature Name	Formula
TotBathrooms	FullBath + (HalfBath*0.5) + $BsmtFullBath +(BsmtHalfBath*0.5)
Remod	ifelse(YearBuilt==YearRemodAdd, 0, 1)
HouseAge	YrSold – YearRemodAdd
IsNew	ifelse(YrSold==YearBuilt, 1, 0)
TotalSqFeet	GrLivArea + TotalBsmtSF

Figure 2. Showing 5 new features

We calculated the correlation of the new TotalsqFeet predictor with the response variable. In the figure A-4 in Appendix A you can see that new feature TotalSqFeet has a strong positive correlation of .78 with the SalePrice. You can also see that there are 2 outliers present. This correlation will improve further after the 2 outliers are removed from the dataset. We used **upfront check with Random Forest** to see if the model will pick this new variable as one of the top predictors. In figure A-5, you can see that TotalSqFeet, TotBathrooms, and HouseAge are among the top predictors to determine the house sale price. We also want to find **out what other predictors have strong correlation** with the target variable. In figure A-7, we can see that TotalSqFeet, GrLivArea, GarageArea have positive correlation with the SalePrice. And yearRemodAdd, BsmtUnfSF, and OpenPorchSF seem to have weak correlation with the response variable. The HouseAge has negative correlation with the SalePrice target variable. It is important to identify and

remove predictors with low predictive power and we used **NearZero function to remove 22** predictors with low variance.

In every predictive modeling exercise, **the issue with the missing values** in the data needs to be addressed. The HousePrice datasets have many missing values that will need to be imputed. The Training dataset has 357 missing values in 5 predictors and the Test dataset has 358 missing values in 21 predictors. We used **Mice Library** to impute the missing values in both the datasets. The Mice Library provides different functions to impute categorical and numeric values. You can see the snippet of the R code used in Figure A-3. In this case, we used polyreg (Poly Regression) function to impute categorical values and cart (Classification and Regression Tree) function to impute numerical values. One issue with the Mice library is that it is computationally intensive, and we dropped 21 missing values manually across 16 predictors from the Test set to save on code execution time. In Figure 3 you can see the breakdown of the missing values in the dataset.

- Training Set 5 Predictors -357 Missing
- LotFrontage GarageYrBlt MasVnrType MasVnrArea Electrical
- 259 81 8 8 1

- Test Set 21 Predictors -358 Missing
- LotFrontage GarageYrBlt MasVnrType MasVnrArea MSZoning Utilities BsmtFullBath
- 227 78 16 15 4 2 2
- BsmtHalfBath Functional TotBathrooms Exterior1st Exterior2nd BsmtFinSF1 BsmtFinSF2
- 2 2 2 1 1 1 1
- BsmtUnfSF TotalBsmtSF KitchenQual GarageCars GarageArea SaleType TotalSqFeet
- 1 1 1 1 1 1 1

Figure 3. Showing breakdown of Missing values

As we would expect that **there is skewness** across many numeric predictors as can be seen in the Figure A-6. The predictive models get negatively impacted with skewness and also if the numeric predictors are not at the same scale. We used BoxCox, Center, and Scale functions to transform the data.

One of the important Data Preprocessing tasks was to **create dataframes for the different scenarios** to be evaluated during modeling phase. These scenarios included with/without datasets for NearZero, Correlated Predictors, Outliers, and Correlated Predictors/Outliers. You can refer to the complete set of dataframes in figure A-14.

Our team evaluated both Regression and the Tree models. We also created **dummy variables** for categorical variables for the dataframes. For Regression based dataframes we dropped one value/column and for the Tree based dataframes all value/columns were kept. These training dataframes were also **standardized** to have a similar structure with the Kaggle Test set so that our models fitted on the training data can predict the house price on the Test set.

In the next step, we **dropped 5 correlated predictors** as can be seen in the figure A-15. For example, GarageAre was dropped as the GarageCars has higher correlation with the SalePrice. We observed that House Price data had few extreme values in the predictors as shown by the figure A-16. We **removed 2 values considered outliers** as identified by the Cook's distance in the Diagnostic plots figure A-17. The training data was split with .75/.25 ratio.

Model Fitting/Validation

Model Scenarios

In predictive modeling, the team identified four modeling scenarios and designed the dataframes accordingly. Please refer to Appendix D for more detail about all the scenarios listed below.

- Baseline Scenario 1 (Preprocessed Dataframe) => **All regression models** except OLS & RLM
- Baseline Scenario 2 (Preprocessed Dataframe) + multicollinear Predictors Removed => OLS & RLM
- Scenario 3 Baseline + Outliers Removed => Several models executed to assess the outliers handling
- Scenario 4 Baseline + (Both Outliers and Multicollinear Predictors Removed) => Optimal scenario executed for all models

The preprocessed training datasets were split in the 75%/25% ratio to generate training and validation steps respectively. The team standardized training control with kfold cross validation (with 3 repeats only to contain long running times). Prior to training the models, the baseline training and validation dataframes for the models OLS, RLM were subject to an additional step of removal of correlated variables. Hence, for these two models, Scenario 2 was used as the baseline scenario. For the RLM, Ridge and ElasticNet models, dataframes for all scenarios (above) were preprocessed with a NearZero function to eliminate the near zero predictors as these models would fail without this step. To clearly denote this, the models are separated into "With Near Zero" and "Without Near Zero" sections when tabulating the model results (see Appendix D).

From the initial OLS training results, it was readily apparent that the response variable required logarithmic transformation. This step was added to all model building and validation from this point forward. The Multiple Linear Regression model was trained with the defaults allowed by the "lm" method of train function.

Model Tuning

Accurately setting the hyperparameters is a key part of building dependable training models. These parameters seek to achieve balance between underfit and an overfit model. All the models were tuned using k-fold cross-validation model training. Sample model training output from R using MARS model is shown in Appendix C.

Multiple Linear regression (OLS): OLS is a simple algorithm and the model can be trained easily even on systems with low computational power compared to complex algorithms. It has lower time complexity relative to some of other machine algorithms and easy to understand. The performance of the linear regression model is reduced by the multicollinearity of the predictors and outliers. So, we handled the multicollinearity of the independent variables by removing the highly correlated predictors and outliers in the baseline and optimal model scenarios.

PCR & PLS Models: The PCR and PLS tuning plots in figure B-1 depict the tuning scenarios for the PLS and PCR models and show us the efficacy of the PLS model in reducing the predictors to a small set of components reduces the multicollinearity effect while striking a better RMSE in the process. We see two scenarios for PCR (baseline vs optimal model), both of which show continuous improvement in RMSE until the model reaches 80 components. For PLS, the best RMSE of 1.46 takes just 12 components. When

correlated predictors were dropped there's a marginal improvement at the same number of components. When the outliers are removed, the performance improves to 1.21 at the same component configuration. In spite of much complex model the PCR results are inferior to the optimal PLS model. We know that strength of the PLS model is ability to negotiate the collinearity, we see this in action in these comparison plots.

Robust Linear Regression: We preprocessed the RLM model with PCA and ran the baseline and eliminated the nearzero predictors from the model as part of model training control.

Ridge & ElasticNet models: For Ridge and ElasticNet models, our tuning consisted of λ_1 range of (0,1) and λ_2 range of (0,1) as well. The optimal Weight decay for both models was very close to 0. The plot below shows the tuning we performed for ElasticNet which marginally outperformed the Ridge regression. The best RMSE (on Y axis) occurs when weight decay (λ1) is 0 or 0.11 depending upon the training samples selected by the model at the time of execution. The RMSE values improve with increasing fraction values (except when λ1 is zero). The best RMSE of 0.14 was observed with a Lasso coefficient of 0.68 as shown in figure B-4.

Support Vector Machines: For SVM model, we preprocessed the data with PCA and used the svmRadial method for training the models. A tune length of 4 was used which added a cost values of 0.25, 0.5, 1.0 and 2.0 to the model while leaving the sigma value to the model function to determine. As the cost increases, the SVM increases the penalty for RMSE so the model tends to overfit. At the lower cost, the model is more generalized and may underfit. We found Optimal value at cost 1.0 as shown in Figure B-3.

Multivariate Adaptive Regression Splines: For the MARS model, we began with an nprune parameter range (2, 15) in conjunction with degree of 1:4 (linear, polynomial, polynomial, step function) in the first iteration and found that degree of 1 was the best fit for this model. The higher order degree range 2:4 was dropped in the subsequent iterations to control the model execution times. Since the RMSE was still trending, the nprune was gradually extended to 40 until there was no further improvement. Optimal RMSE found at 38 terms can be seen in figure B-5.

K-Nearest Network: For KNN model, we used the tune length parameter for model training. Up to 30 neighbors were tried to understand the RMSE trend. As graphically shown in Figure B-2, both baseline and optimal models indicate that number of neighbors (K) value of 5 to 7 was the most optimal range for this dataset. However, the KNN model was among the weakest models in our project.

Neural Network & Average Neural Network: Our team experimented with the hidden layer size of 1:3 and wide decay range with big steps to find a direction to optimize the model. Further iterations reduced the size to 1 based on the training results while the decay range was condensed to 0.4 to 0.5 where the optimal RMSE was found for Neural Networks. We used a similar approach for Average Neural Networks which uses the train method "avnet" instead of "nnet". For ANN model, the size of 2 was more optimal with a decay close to 0.65 for both baseline and best scenarios. The Average Neural Networks outperformed the Neural Network in both baseline and optimal scenarios.

Tree based Models: For the tree models, the team used different types of hyperparameters to train each type of tree models we used in the project. In training Simple Regression Tree, three different values 0, 0.1 and 0.01 were assigned to the complexity parameter. This helped prevent overfitting by providing a trade-off between the tree size and error rate. The Random Forest was tuned on a mtry hyperparameter which represents the number of candidate variables sampled at each split. The value given ranged between 5 to 8 in order to find an optimal value that could maintain the balance between correlation amongst trees and the strength of an individual tree.

For the Gbm model training, we used parameters n. trees, interaction. depth, shrinkage and bag.fraction. The value for interaction.depth was kept at an optimal value of 6 to capture unique interactions but at the same time avoid overfitting. The value of shrinkage was given a value of 0.1, thus allowing the model to generalize well, avoid overfitting and also be less computationally demanding.

Lastly, for the XgBoost model, a number of tuning hyperparameters were used to achieve the optimal performance. Some of the important ones were max.depth and eta. We experimented with different values for max.depth ranging from 10 to 25 with an increment of 5, so as to find the optimal value that prevents overfitting and avoid complexity. The eta controls the learning rate for the tree. We set an optimal eta value of 0.1, as to improve model learning without causing the model to overfit.

Model Results

Baseline Scenario Results

The Table D-1 (Appendix D) shows the results we obtained with the Baseline scenario for the various models. We note that Average Neural Networks (ANN) RMSE of 0.130 is the best among all baseline training configurations. The two Neural Network models also give us the least standard deviation in RMSE. When trained models are used to predict against the test split, we found that XGBoost gives us the best RMSE. The Baseline XGBoost Test RMSE is incidentally very close to its Train RMSE which is usually a good indicator that model was well fit. We found that Test R^2 value of Ridge model was off the cliff at 0.660. We will discuss more about it later.

Optimal Scenario Results

After the baseline numbers gave us the benchmark estimates of RMSE of 0.13 to 0.15 range and R^2 of 0.84 to 0.9, the team was ready to introduce additional refinements to improve the model performance. The team removed the outliers found by diagnostics of the residuals from multiple linear regression residual plots which we refer to as “Scenario 3”. Our final permutation was removing correlated predictors as well, as described in the preprocessing section, which we called Optimal model or Scenario 4. The full set of models from Baseline scenario were rerun for the Scenario 4 shown in the Table below.

The Ridge model which showed a poor Test R^2 in baseline model improved to such an extent in the optimal model that its RMSE of 0.884 is on par with the rest of the models. Since the model tuning configuration was similar, the removal of outliers in test split may have contributed to this performance gain. We noticed substantial improvements across the board when compared to baseline results. The OLS, PLS,

ANN, and Boosted models returned RMSE <= 0.127 and Test R^2 >= 0.9 both of which exceeded the baseline metrics gathered prior to these model enhancements. In relation to the train metrics, most of the models measured quite well except MARS model which dropped the R^2 by 0.03 indicating an overfit model. PLS test metrics bettered the training metrics by about 2%. Table D-2 in Appendix D shows an additional scenario we ran with just the outliers dropped from Baseline (Scenario 3).

Without Near Zero Function on Categorical Predictors							
Model	Train RMSE	Train R^2	Train RMSE SD	Train R^2 SD	Test RMSE	Test R^2	Hyperparameter Tuning
Multiple Linear Regression (OLS)	**0.124**	**0.903**	**0.016**	**0.021**	**0.127**	**0.903**	
Prin. Comp. Regression (PCR)	**0.140**	**0.880**	**0.007**	**0.001**	**0.139**	**0.883**	**ncomp=79**
Partial Least Squared (PLS)	**0.129**	**0.898**	**0.007**	**0.0109**	**0.121**	**0.912**	**ncomp=14**
Multi Variate. Reg. Splines (MARS)	**0.124**	**0.905**	**0.016**	**0.023**	**0.141**	**0.870**	**degree=1, nprune=36**
Support Vector Machine (SVM) with PCA	**0.137**	**0.885**	**0.022**	**0.033**	**0.136**	**0.880**	**c=1.0**
Neural Network	**0.134**	**0.890**	**0.009**	**0.012**	**0.131**	**0.890**	**size=1, decay=0.47**
Av. Neural Network	**0.127**	**0.900**	**0.008**	**0.01**	**0.125**	**0.900**	**size=2, decay=0.65**
K-Nearest Network	**0.160**	**0.853**	**0.211**	**0.026**	**0.167**	**0.830**	**k=7**
Simple Regression Tree	**0.191**	**0.772**			**0.197**	**0.772**	**cp = seq(0,0.1,0.01)**
Random Forest							**mtry = 8; 8**
Boosted Model					**26914**	**0.911**	**See Note 1 below**
XgBoost	**0.125**	**0.901**			**0.132**	**0.894**	**See Note 2 below**
With Near Zero Function on Categorical Predictors							
Robust Linear Regression with PCA							
Ridge	**0.130**	**0.900**			**0.133**	**0.884**	**Lambda=0**
ElasticNet	**0.130**	**0.900**			**0.133**	**0.884**	**Lambda=0, Fraction =0.47**

Figure 4: Home price sale prediction - Optimal Model (Baseline + Corr predictors Drop + Outliers Drop)

Note 1: Boosted Model Tuning: n.trees =2000, interaction.depth=6, shrinkage=0.1,bag.fraction = .75

Note 2: XgBoost Tuning: nrounds = c(100,200),max_depth = c(10, 15, 20, 25),colsample_bytree = seq(0.5, 0.9, length.out = 5),eta = 0.1,gamma=0,min_child_weight = 1,subsample = 1

Model Analysis

In the previous section, we discussed the methodology, scenarios and the results obtained for the Ames, Iowa home price prediction analysis. We now explore the fundamentals of the models and how they influenced the model results. Furthermore, we will discuss insights from the application of the best models to our unseen test dataset without response data. We will explore the most important predictors that emerged from the model validation exercise and reconcile with the real-world factors we are aware of as home buyers and sellers.

Model Behavior

Multiple Linear Regression: OLS model was one of the best models in our analysis with a p-value of 2.2e-16 < 0.05. We reject the null hypothesis and conclude that there is a linear relationship between the predictors and the outcome. The actuals vs predicted values of the Sale price of a house using the OLS regression model are shown in the figure D-2 in Appendix D. The OLS model yielded an R^2 value of 0.903. The higher the R^2 and lower the RMSE, the better is the goodness-of-fit of the model. From the plot, we can see that the data points are closer to the diagonal indicating a good linear fit with slight deviations.

The figure D-1 shows predictors like LotArea, YearBuilt, GarageCars, TotalSqFeet of the OLS model with p-values < 0.05 and hence significant. The equation shown in the image is used to interpret the OLS model to understand how the Sale price varies with the change in the predictors. For example, a unit increase in the TotalSqFeet area increases the Sale price by 0.14 times, and a unit increase in BsmtUnfSF decreases the Sale price by 0.03 times. The residual standard error is 0.09795. Lower residual standard error indicates the higher accuracy of the predictions. We did observe that some of the coefficients are negative. According to Tarafder (2020), the negative coefficient sign is a result of the range of values within the categorical variable which will require additional feature engineering steps including dropping some features.

Multivariate Adaptive Regression Splines: MARS model also uses surrogate features, breaking the predictor value sets into groups and modeling linear relationships between the predictors and outcomes for every group (Kuhn & Johnson, 2016 p.145). Albeit strong, MARS was not among the best models.

Robust Linear Regression: RLM model uses Huber function to contain influential samples which computes residual errors as squared when the magnitude is small and simple absolute residual difference when the error is large. This approach can be useful for outliers where the residual error can be large. In our model, the RLM was suboptimal compared to other options.

Ridge & ElasticNet models: Ridge and ElasticNet are penalty-based algorithms that seek to regularize the linear regression models to prevent large coefficients from overly influencing the model. Ridge regression L_2 Norm is quadratic and ElasticNet regression L_1 Norm is linear. They are effective in controlling the correlated predictors and overfitting because of the penalty terms. Other than the aberration with outliers in the baseline Test metrics for Ridge model, both the models showcase strong performance across the board which confirms the model strengths. The best test RMSE for these penalty models is a strong 0.133 and an R^2 of 0.884. These metrics are slightly lower than the correlation treated baseline OLS model.

Neural Network/Average Neural Network: Neural networks solutions are known to work well even with limited set of training samples. The model learning is time consuming due to special algorithm backward

propagation. Neural Networks tend to overfit due to model complexity, so we tuned the number of hidden layers in the network and the regularization parameter (decay) to avoid overfitting the model (Kuhn & Johnson, 2016 p.144-145). We obtained strong results with Average Neural networks with the lowest RMSE of 0.125 and R^2 of 0.9. Please refer to figure D-3 for a graphical view of prediction accuracy.

Support Vector Machine with PCA: According to Berwick (2011), Support Vector Machine model uses hyperplanes to separate buckets according to the feature set. SVM is known to be less sensitive to outliers. We observed this aspect manifested in the baseline and optimal modeling results where the Test R^2 was essentially unchanged from 0.867 to 0.870. However, this was not our strongest model.

Tree-based models: The tree models are generally more robust to outliers compared to other models and can handle multicollinearity quite well. However, as we can see in the results table, when we run the tree models based on scenario 3 with outliers removed, we can observe an increase in R^2 and reduction in RMSE for all the models. This result is further improved when we run the models on scenario 4 by removing correlated predictors as well. Furthermore, compared to some other models used in this project, the performance of tree models on baseline is much better.

Simple regression tree is known to be simple and easy to implement, but on the other hand their performance is low compared to other tree models such as Random Forest and Boosting trees. It is generally because simple regression tree generates only a single tree and is also prone to overfitting. We can observe in the table, the best performance we get for the Simple Regression Tree is the R^2 of 0.772 and RMSE of 0.197 on the test dataset. In comparison to Simple Regression Tree, Random Forests generates a large number of trees and averages out their performance to get more accurate results. The Random Forest was able to give us a better performance than simple regression tree with an RMSE of 0.153 and R2 of 0.875 on the test dataset.

The boosting tree models Gbm and XgBoost are similar to Random Forest as they also use a group of decision trees, however, unlike Random Forests, they tend to build one tree at a time. Also, boosting models combine results on the way over, unlike Random Forest that does it at the end. Performance wise, gradient boosting methods perform better compared to Simple Regression Tree and Random Forests. The results table shows that the best performing tree model Gbm has an actual RMSE of 26914 and R^2 of 0.911. The XgBoost model also shows good performance with an RMSE of 0.132 and R^2 of 0.894 on the test dataset. Comparing the running times of both models, XgBoost tends to take more time than the Gbm model. Also, the XgBoost model uses more computation as it applies parallelization and distributed computing.

The Gbm model, similar to other tree-based model, handles outliers better than other linear and non-linear models because it divides the node into two on each split, so even if the value is bigger in magnitude, it doesn't have a major influence on the model. The same node splitting method also manages multicollinearity. It chooses only one variable at each split in case there are two correlated variables. The Gbm tree removes the other corelated variable before performing model calculation.

Best Fit Models

The OLS, PLS, ANN and Boosted Model are the best performing models from validation exercise. The results from the various models indicate that problem is essentially linear. There were clues in the optimization in the non-linear models such as MARS where degree of 1 was chosen for relationships between the predictors and the response. This is the reason why some of the non-linear models closely matched the optimized multiple linear regression model and did not outperform by any significant metric.

We learnt in this course when selecting model between multiple similarly performing models, we look for more simpler option that is explainable and generalized. With this approach, the team selected OLS model as well as the GBM model, the two standout models and applied these to the unseen Kaggle test data for making comparisons. While models such as GBM are strong in handing outliers and correlated predictors, removing them manually improved the models' performance little further.

MODEL APPLICATION

The picture below shows the Sales price of a typical property from our training data set for which the sales price was known. We identified a similar property in the Kaggle test data set for which the sales price was not known to help us validate our model predictions. The sales price for the property was $556,581.

TotalSqFeet	ExterQual	HouseAge	GrLivArea	TotalBsmtSF	GarageArea	LotArea	Fireplaces	KitchenQual	OverallQual	Remod	IsNew	TotBathrooms	SalePrice
4355	Ex	0	2945	1410	641	13682	1	Gd	10	0	1	3.5	438780
4450	Ex	0	2234	2216	1166	17423	1	Ex	9	1	0	3	501837
3712	Gd	0	1856	1856	834	11394	1	Ex	9	0	1	2.5	394432
3640	Gd	0	1826	1814	758	11302	1	Gd	8	1	0	3	319000
2838	Gd	0	1419	1419	567	8089	1	Gd	8	0	1	3	392000
3810	Ex	0	2576	1234	666	12438	1	Ex	8	0	1	2.5	361919
4860	Gd	0	2868	1992	716	16056	1	Ex	9	1	0	3.5	556581

Figure 5: Observed Sales Price from training data

When we applied the GBM model to the Kaggle test data set, the model was able to predict the price of $553,356.00 for a 4,918 square foot house in the affluent Stonebridge subdivision, as shown in box plot E-1 in Appendix E. The predicted price is very close to the observed sales price for roughly an equivalent property. The model output from the figure 6 below shows the descriptive results from the GBM model predictions.

gbmPred_Dfp

TotalSqFeet	ExterQual	HouseAge	GrLivArea	TotalBsmtSF	GarageArea	LotArea	Fireplaces	KitchenQual	OverallQual	Remod	IsNew	TotBathrooms	PredSalePrice
4998	Ex	0	2338	2660	1110	51974	2	Gd	9	1	0	3.5	465079.4
3784	Ex	0	2042	1742	724	13870	1	Ex	10	1	0	3	429582.0
4196	Gd	0	2798	1398	670	16023	1	Ex	9	1	0	4.5	478457.5
4918	Ex	0	3390	1528	758	18062	1	Ex	10	0	1	3.5	553356.6
3567	Gd	0	2473	1094	675	12292	1	Gd	9	0	1	2.5	343918.3
4548	Ex	0	2698	1850	736	16052	1	Ex	10	0	1	3.5	508415.0
4185	Gd	0	2795	1390	660	15922	1	Ex	9	1	0	3.5	450517.4

Figure 6: Results from the GBM model run to the Kaggle test set

We also applied the multiple linear regression model to the Kaggle test data set. We were able to predict the house sale price of $558,200 for the same 4,918 square foot home in the Stonebridge subdivision reflected in the box plot E-2 in Appendix E and the OLS results shown in the figure 7 below.

lmPred_Dfp

TotalSqFeet	ExterQual	HouseAge	GrLivArea	TotalBsmtSF	GarageArea	LotArea	Fireplaces	KitchenQual	OverallQual	Remod	IsNew	TotBathrooms	PredSalePrice
4998	Ex	0	2338	2660	1110	51974	2	Gd	9	1	0	3.5	565755.2
3784	Ex	0	2042	1742	724	13870	1	Ex	10	1	0	3	440996.2
4196	Gd	0	2798	1398	670	16023	1	Ex	9	1	0	4.5	637355.9
4918	Ex	0	3390	1528	758	18062	1	Ex	10	0	1	3.5	558282.8
3567	Gd	0	2473	1094	675	12292	1	Gd	9	0	1	2.5	395701.2
4548	Ex	0	2698	1850	736	16052	1	Ex	10	0	1	3.5	531903.3
4185	Gd	0	2795	1390	660	15922	1	Ex	9	1	0	3.5	454678.8

Figure 7: Results from the OLS model run to the Kaggle test set

As we can see from the above discussion, both GBM and OLS model were effective in predicting house sale price when compared to a similar type of the house we found in the training data for which sale price was known.

VARIABLE IMPORTANCE

Out of the 79 predictors, which predictors did our models deem as the most important for sales price prediction? Real estate experts consider the following as the most important factors in home purchase: **location, lot size, age, house style, Square footage, cost, structure and remodel age**. Our models have agreed that lot size, age, square footage, structure and remodel age are indeed top predictors. In particular, the Square Footage (TotalSqFeet, GrLivArea), age (HouseAge, YearBuilt) have prominently featured among the top three predictors across the best performing models listed in the figure 8 below.

The datasets contained several fields of interest for features like structure, such as Foundation, OverallQual, ExterQual, BsmtQual OverallCond, GarageQual etc. so we suspect that with a reduced set of predictors, the model results could converge more. Another example is the Square footage where the raw data contains columns like above ground living area, basement area (finished, unfinished), first floor living area, second floor living area etc. for full disclosure to the buyer. It is conceivable that models diluted weights across the related predictors so our conventional methods of capturing variable importance are not ideal.

Multiple Linear Regression (OLS)	PLS	Average Neural Networks	Gradient Boosted Model (GBM)
LotArea	TotalSqFeet	TotalSqFeet	TotalSqFeet
YearBuilt	GrLivArea	GrLivArea	YearBuilt
BsmtUnfSF	GarageCars	GarageCars	GarageCars
GarageCars	GarageArea	X1stFlrSF	HouseAge
TotalSqFeet	X1stFlrSF	FullBath	BsmtQual
MSZoning	TotalBsmtSF	HouseAge	LotArea
FullBath	FullBath	ExterQual_TA	Fireplaces
HouseAge	YearBuilt	YearBuilt	ExterQual
Foundation	GarageYrBlt	TotRmsAbvGrd	GrLivArea
CentralAir	HouseAge	Foundation	KitchenQual
KitchenQual	YearRemodAdd	KitchenQual	BsmtUnfSF
GarageQual	TotRmsAbvGrd	ExterQual_Gd	OverallQual
Fireplaces	Fireplaces	Fireplaces	FireplaceQu
Neighborhood	MasVnrArea	LotArea	X1stFlrSF
OverallQual	LotArea	BsmtQual	MasVnrArea
OverallCond	LotFrontage	HeatingQC	OpenPorchSF
MasVnrType	BsmtFinSF1	X2ndFlrSF	X2ndFlrSF
GarageType	OpenPorchSF	MasVnrArea	LotFrontage
	WoodDeckSF	OverallQual	BsmtFinType
	X2ndFlrSF		TotRmsAbvGrd

Figure 8: Variable Importance Table - Top Performing Models

Note: Where number of dummy variables for the same categorical variable were found, they were pooled into the root predictor to eliminate redundancy

We determined that Garage characteristics such as area, type of garage, number of car spaces, Garage quality etc. were all important. The Number of fireplaces was picked by every single model in the table above which is very interesting. Number of Full Baths, Kitchen Quality found place in three out of four top models which we can relate to as home buyers and sellers. It was fascinating to watch Foundation (concrete) named among the top predictors by our models. It must be a feature regarded highly in the Ames region.

SUMMARY RESULTS AND CONCLUSION

This group undertook the test of determining the sale price of a house in Ames, IA. This project made into a competition in Kaggle in 2011. This study reviewed the original dataset and decided it met the challenges necessary to accomplish the learning goals. The scope of this work included exploratory data analysis (EDA), visualizations, and data transformation. For the modeling part, there were fifteen linear, nonlinear, and tree models to eight different scenarios. We created eighteen data frames in R for these models to run over the training and test datasets and examined the results. After a review of the modeling results, the OLS and GBM model stood out as the best performers.

From the results, Multiple linear regression (OLS) displayed the best overall performance among linear models applied. As shown in figure 9 below, OLS displayed the lowest test RMSE with 0.127 and a test R^2 value of 0.903, meaning that approximately 90% of the variation can be explained by this model. OLS model can be expressed into a linear combination of the parameters, so easier to interpret and explain.

Models / Scenarios (Without Nearzero)		W/O Near Zero, Outliers Removed Manually							W/O Near Zero, Drop Correlated Predictors and Outliers Removed Manually						
		Train RMSE	Train R-SQ	Train RMSE SD	Train R-SQ SD	Test R-SQ	Test RMSE	Hyperparameter Tuning	Train RMSE	Train R-SQ	Train RMSE SD	Train R-SQ SD	Test RMSE	Test R-SQ	Hyperparameter Tuning
Linear	Multiple Linear Regression (OLS)								0.124	0.903	0.016	0.023	0.127	0.903	
	Prin. Comp. Regression (PCR)	0.140	0.880	0.007	0.001	0.139	0.883								
	Partial Least Squared (PLS)	0.130	0.897	0.007	0.011	0.121	0.911		0.129	0.898	0.007	0.011	0.121	0.912	
Non-Linear	Multi Variate. Reg. Splines (MARS)								0.124	0.905	0.016	0.023	0.141	0.870	degree=1, nprune=36
	Support Vector Machine (SVM) with PCA	0.137	0.885			0.136	0.880		0.137	0.885	0.022	0.033	0.136	0.880	c=1.0
	Neural Network								0.134	0.890	0.009	0.012	0.131	0.890	size=1, decay=0.47
	Av. Neural Network	0.127	0.900			0.123	0.900		0.127	0.900	0.008	0.010	0.125	0.900	size=2, decay=0.65
	K-Nearest Network								0.160	0.853	0.211	0.026	0.167	0.830	k=7
Tree	Simple Regression Tree	0.190	0.774			0.197	0.772		0.191	0.772			0.197	0.772	
	Random Forest	0.155	0.872			0.153	0.875								
	Boosted Model					27424	0.907						26914	0.911	
	XgBoost	0.125	0.900			0.132	0.893		0.125	0.901			0.132	0.894	
Models / Scenarios (With Nearzero)		With Near Zero, Outliers Removed Manually							With Near Zero, Drop Correlated Predictors and Outliers Removed Manually						
		Train RMSE	Train R-SQ	Train RMSE SD	Train R-SQ SD	Test R-SQ	Test RMSE	Hyperparameter Tuning	Train RMSE	Train R-SQ	Train RMSE SD	Train R-SQ SD	Test RMSE	Test R-SQ	Hyperparameter Tuning
Linear	Robust Linear Regression with PCA	0.131	0.893	0.015	0.020	0.138	0.885								
	Ridge					0.135	0.883		0.130	0.900			0.133	0.884	0.884
	ElasticNet								0.130	0.900			0.133	0.884	0.884

Figure 9: Summary of Model Results

Figure 9 also highlights the Gradient boosting machine (GBM) as the other model selected for our prediction for their overall performance: RMSE of 26,914 and R-squared of 0.911, the highest overall. Even though OLS was a simpler model to interpret and provided comparable predictive results, the team selected GBM model as OLS model requires additional feature engineering to address the negative coefficient values.

Based on the model diagnostics, we can deduce that lot area, age, usable living area, garage capacity, number of full bathrooms, condition/quality, fireplaces, and foundation (concrete) are some of the most important predictors for Ames, Iowa home buyers. Location is not included in this list because "Neighborhood" categorical variable had to be split into 25 dummy variables, distributing the weights across all of them. In addition, one of the sought-after neighborhoods, Stone Bridge was found among the top 25 predictors in OLS model.

To further improve the predictive accuracy, the models should be ingested with more extensive sample training data and further tuning for hyperparameters should be considered. In addition, the neighborhood predictors should be consolidated into a new feature (high, medium, low end neighborhoods) to better measure its importance. The Ames, Iowa home price prediction team thoroughly enjoyed exploring several popular regression techniques to solve a real-world business problem during this semester long project journey.

Ames House Price Raw Data Excerpt

Id	LotFrontage	LotArea	Neighborhood	BldgType	HouseStyle	OverallQual	OverallCond	YearBuilt	ExterQual	BsmtUnfSF
1	65	8,450	CollgCr	1Fam	2Story	7	5	2003	Gd	150
2	80	9,600	Veenker	1Fam	1Story	6	8	1976	TA	284
3	68	11,250	CollgCr	1Fam	2Story	7	5	2001	Gd	434
4	60	9,550	Crawfor	1Fam	2Story	7	5	1915	TA	540
5	84	14,260	NoRidge	1Fam	2Story	8	5	2000	Gd	490
6	85	14,115	Mitchel	1Fam	1.5Fin	5	5	1993	TA	64
7	75	10,084	Somerst	1Fam	1Story	8	5	2004	Gd	317
8	NA	10,382	NWAmes	1Fam	2Story	7	6	1973	TA	216
9	51	6,120	OldTown	1Fam	1.5Fin	7	5	1931	TA	952
10	50	7,420	BrkSide	2fmCon	1.5Unf	5	6	1939	TA	140

TotalBsmtSF	GrLivArea	FullBath	HalfBath	BedroomAbvGr	TotRmsAbvGrd	Fireplaces	GarageCars	YrSold	SaleCondition	SalePrice
856	1,710	2	1	3	8	0	2	2008	Normal	$ 208,500
1,262	1,262	2	0	3	6	1	2	2007	Normal	$ 181,500
920	1,786	2	1	3	6	1	2	2008	Normal	$ 223,500
756	1,717	1	0	3	7	1	3	2006	Abnorml	$ 140,000
1,145	2,198	2	1	4	9	1	3	2008	Normal	$ 250,000
796	1,362	1	1	1	5	0	2	2009	Normal	$ 143,000
1,686	1,694	2	0	3	7	1	2	2007	Normal	$ 307,000
1,107	2,090	2	1	3	7	2	2	2009	Normal	$ 200,000
952	1,774	2	0	2	8	2	2	2008	Abnorml	$ 129,900
991	1,077	1	0	2	5	2	1	2008	Normal	$ 118,000

Figure A-1 House Price Dataset

Data Preprocessing - Near Zero Predictors removed (22)

```
> colnames(HouseFullDF)[nZeroCol]
 [1] "Street"       "Alley"        "LandContour"  "Utilities"    "LandSlope"
 [6] "Condition2"   "RoofMatl"     "BsmtCond"     "BsmtFinType2" "BsmtFinSF2"
[11] "Heating"      "LowQualFinSF" "KitchenAbvGr" "Functional"   "EnclosedPorch"
[16] "X3SsnPorch"   "ScreenPorch"  "PoolArea"     "PoolQC"       "MiscFeature"
[21] "MiscVal"      "HouseAge"
```

Figure A-2. Near Zero Predictors Removed

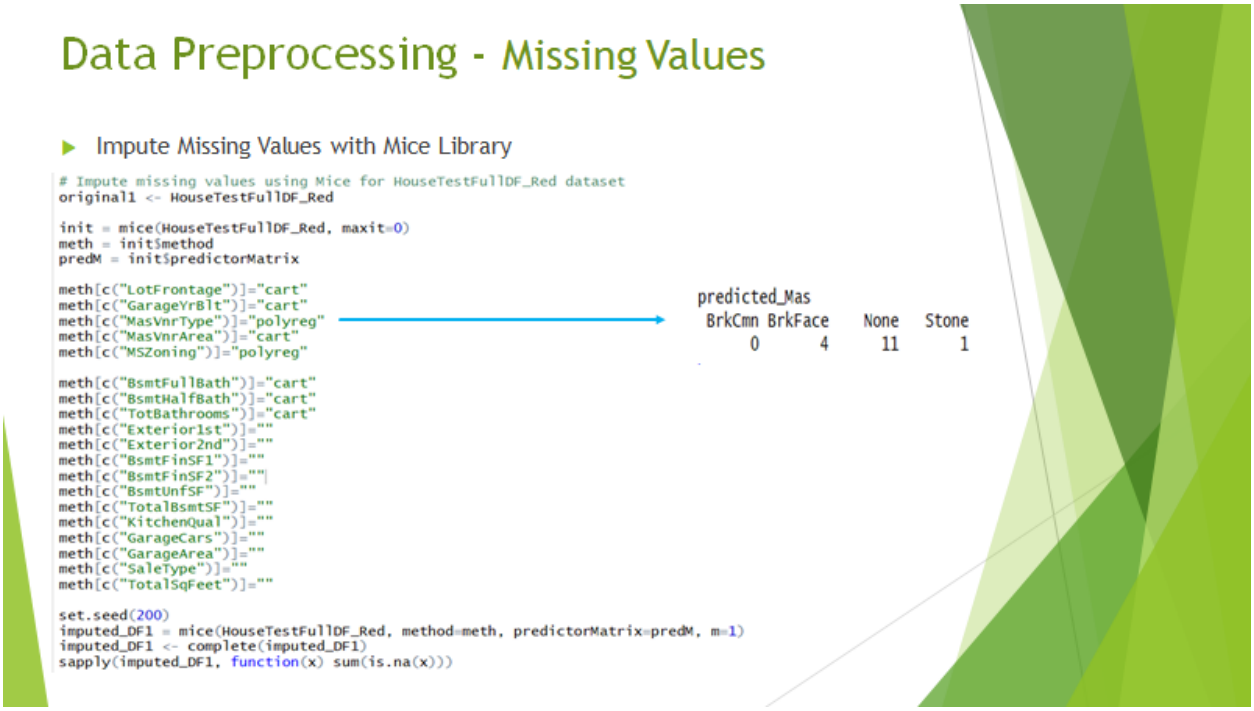

Figure A-3. Mice Library to impute missing

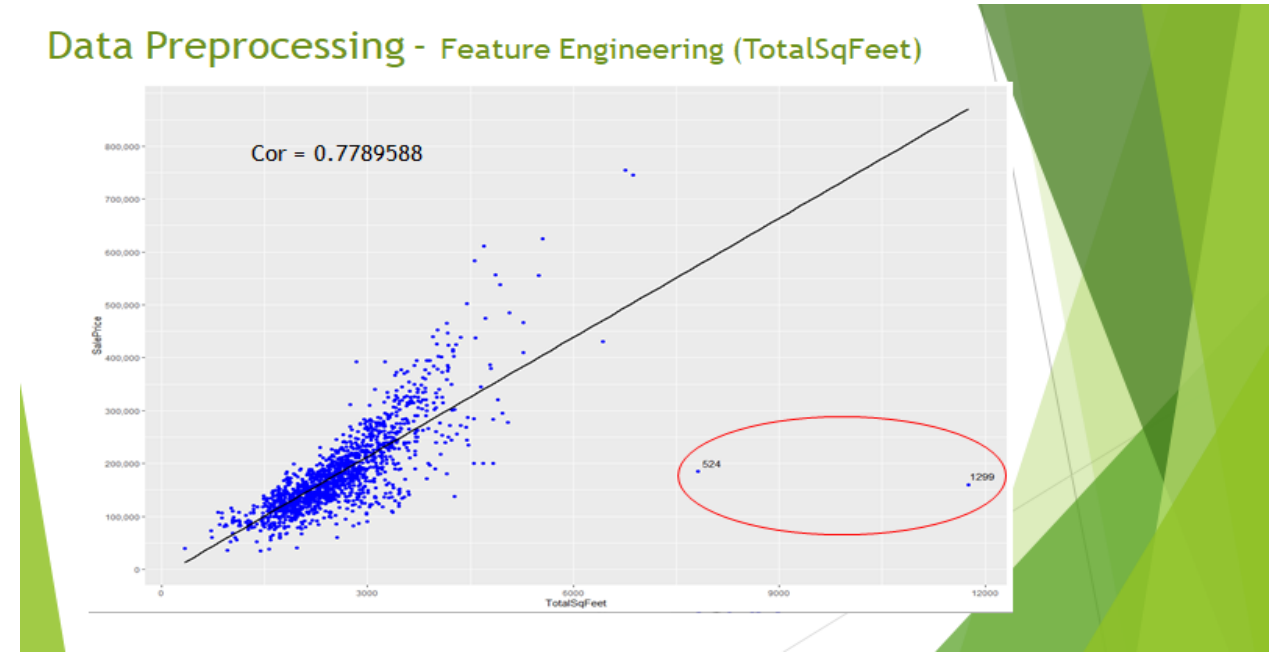

Figure A-4. Correlation Chart TotalSqFeet vs SalesPrice

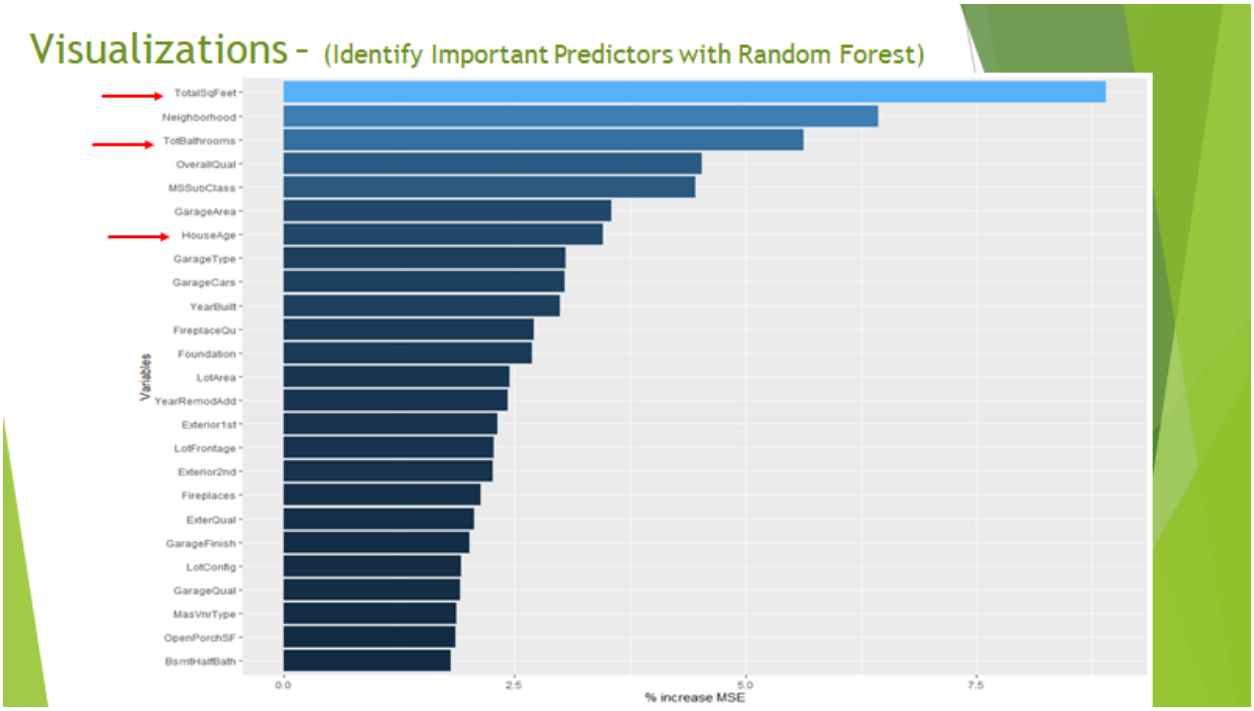

Figure A-5. Initial check of important variables with Random Forest

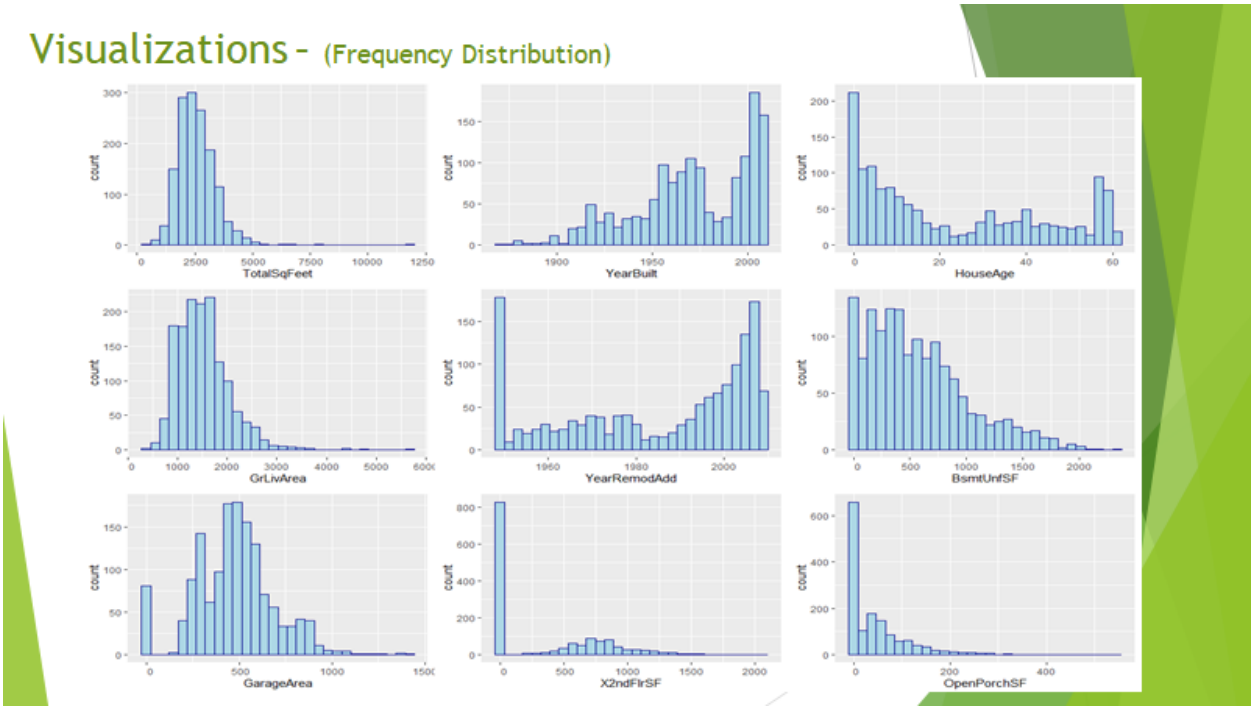

Figure A-6. Frequency Plot showing skewness

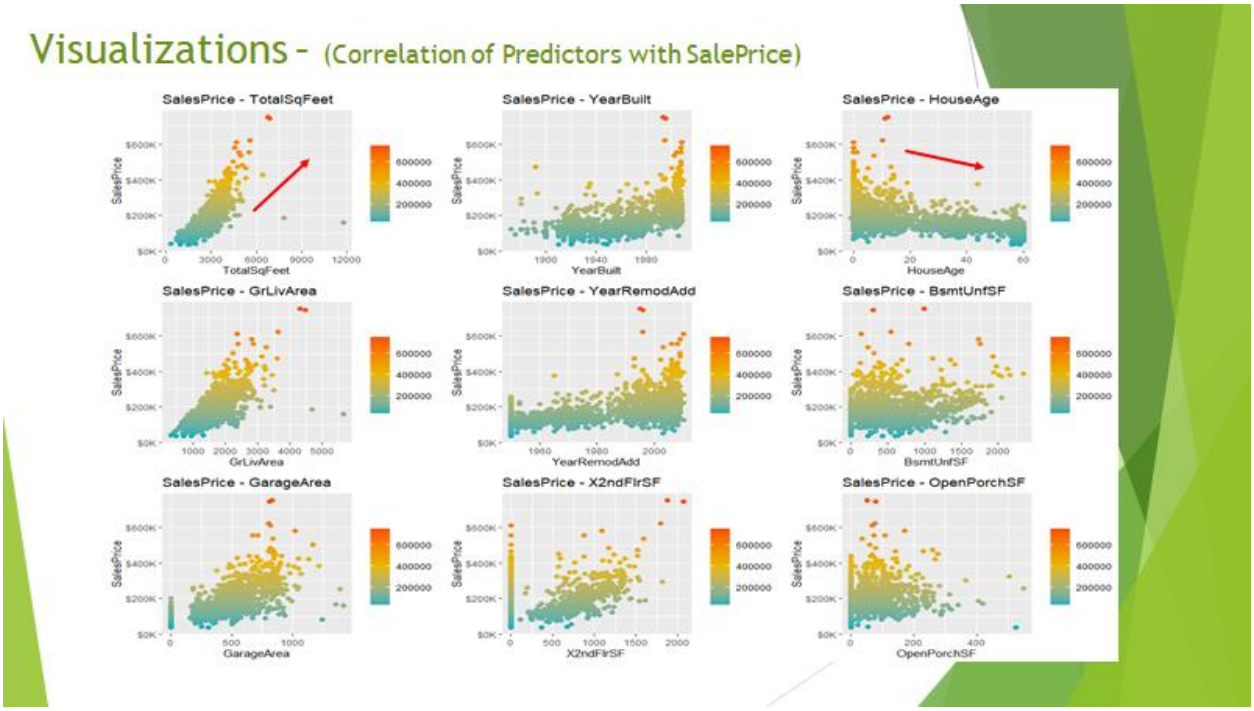

Figure A-7. Scatter Plot showing predictor correlation with SalePrice

Ames House Price Raw Data Excerpt

Id	LotFrontage	LotArea	Neighborhood	BldgType	HouseStyle	OverallQual	OverallCond	YearBuilt	ExterQual	BsmtUnfSF
1	65	8,450	CollgCr	1Fam	2Story	7	5	2003	Gd	150
2	80	9,600	Veenker	1Fam	1Story	6	8	1976	TA	284
3	68	11,250	CollgCr	1Fam	2Story	7	5	2001	Gd	434
4	60	9,550	Crawfor	1Fam	2Story	7	5	1915	TA	540
5	84	14,260	NoRidge	1Fam	2Story	8	5	2000	Gd	490
6	85	14,115	Mitchel	1Fam	1.5Fin	5	5	1993	TA	64
7	75	10,084	Somerst	1Fam	1Story	8	5	2004	Gd	317
8	NA	10,382	NWAmes	1Fam	2Story	7	6	1973	TA	216
9	51	6,120	OldTown	1Fam	1.5Fin	7	5	1931	TA	952
10	50	7,420	BrkSide	2fmCon	1.5Unf	5	6	1939	TA	140

TotalBsmtSF	GrLivArea	FullBath	HalfBath	BedroomAbvGr	TotRmsAbvGrd	Fireplaces	GarageCars	YrSold	SaleCondition	SalePrice
856	1,710	2	1	3	8	0	2	2008	Normal	$ 208,500
1,262	1,262	2	0	3	6	1	2	2007	Normal	$ 181,500
920	1,786	2	1	3	6	1	2	2008	Normal	$ 223,500
756	1,717	1	0	3	7	1	3	2006	Abnorml	$ 140,000
1,145	2,198	2	1	4	9	1	3	2008	Normal	$ 250,000
796	1,362	1	1	1	5	0	2	2009	Normal	$ 143,000
1,686	1,694	2	0	3	7	1	2	2007	Normal	$ 307,000
1,107	2,090	2	1	3	7	2	2	2009	Normal	$ 200,000
952	1,774	2	0	2	8	2	2	2008	Abnorml	$ 129,900
991	1,077	1	0	2	5	2	1	2008	Normal	$ 118,000

Figure A-8. House Prices Training Dataset

Data Preprocessing - Near Zero Predictors removed (22)

```
> colnames(HouseFullDF)[nZeroCol]
 [1] "Street"        "Alley"         "LandContour"   "Utilities"     "LandSlope"
 [6] "Condition2"    "RoofMatl"      "BsmtCond"      "BsmtFinType2"  "BsmtFinSF2"
[11] "Heating"       "LowQualFinSF"  "KitchenAbvGr"  "Functional"    "EnclosedPorch"
[16] "X3SsnPorch"    "ScreenPorch"   "PoolArea"      "PoolQC"        "MiscFeature"
[21] "MiscVal"       "HouseAge"
```

Figure A-9. Near Zero Predictors Removed

Data Preprocessing - Missing Values

- Impute Missing Values with Mice Library

```
# Impute missing values using Mice for HouseTestFullDF_Red dataset
original1 <- HouseTestFullDF_Red

init = mice(HouseTestFullDF_Red, maxit=0)
meth = init$method
predM = init$predictorMatrix

meth[c("LotFrontage")]="cart"
meth[c("GarageYrBlt")]="cart"
meth[c("MasVnrType")]="polyreg"
meth[c("MasVnrArea")]="cart"
meth[c("MSZoning")]="polyreg"

meth[c("BsmtFullBath")]="cart"
meth[c("BsmtHalfBath")]="cart"
meth[c("TotBathrooms")]="cart"
meth[c("Exterior1st")]=""
meth[c("Exterior2nd")]=""
meth[c("BsmtFinSF1")]=""
meth[c("BsmtFinSF2")]=""
meth[c("BsmtUnfSF")]=""
meth[c("TotalBsmtSF")]=""
meth[c("KitchenQual")]=""
meth[c("GarageCars")]=""
meth[c("GarageArea")]=""
meth[c("SaleType")]=""
meth[c("TotalSqFeet")]=""

set.seed(200)
imputed_DF1 = mice(HouseTestFullDF_Red, method=meth, predictorMatrix=predM, m=1)
imputed_DF1 <- complete(imputed_DF1)
sapply(imputed_DF1, function(x) sum(is.na(x)))
```

```
predicted_Mas
 BrkCmn BrkFace    None   Stone
      0       4      11       1
```

Figure A-10. Mice Library to impute missing values

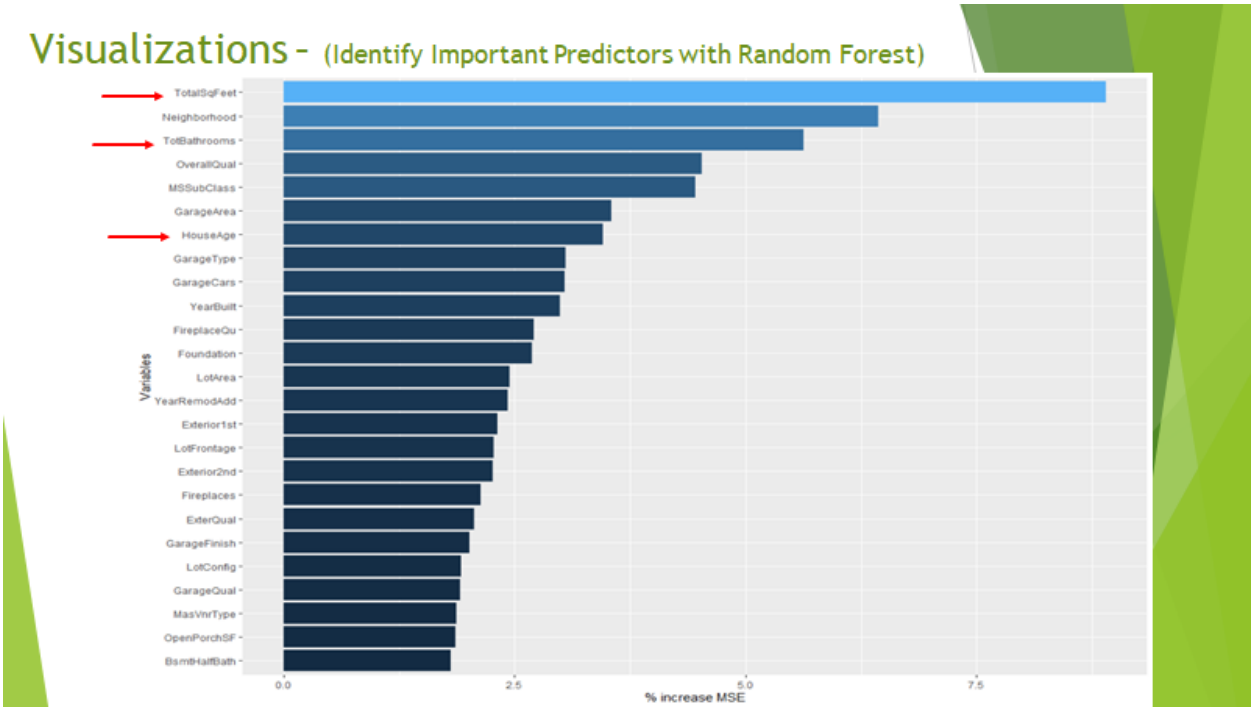

Figure A-11. Initial check of important variables with Random Forest

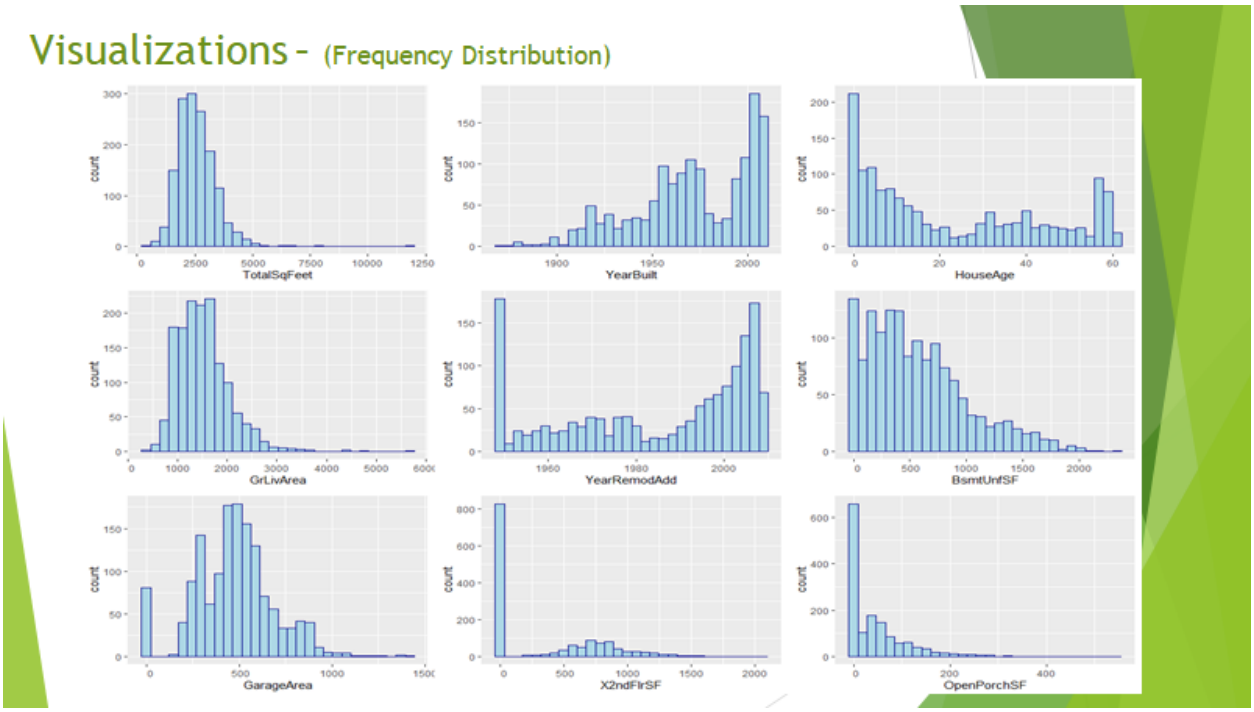

Figure A-12. Frequency Plot showing skewness

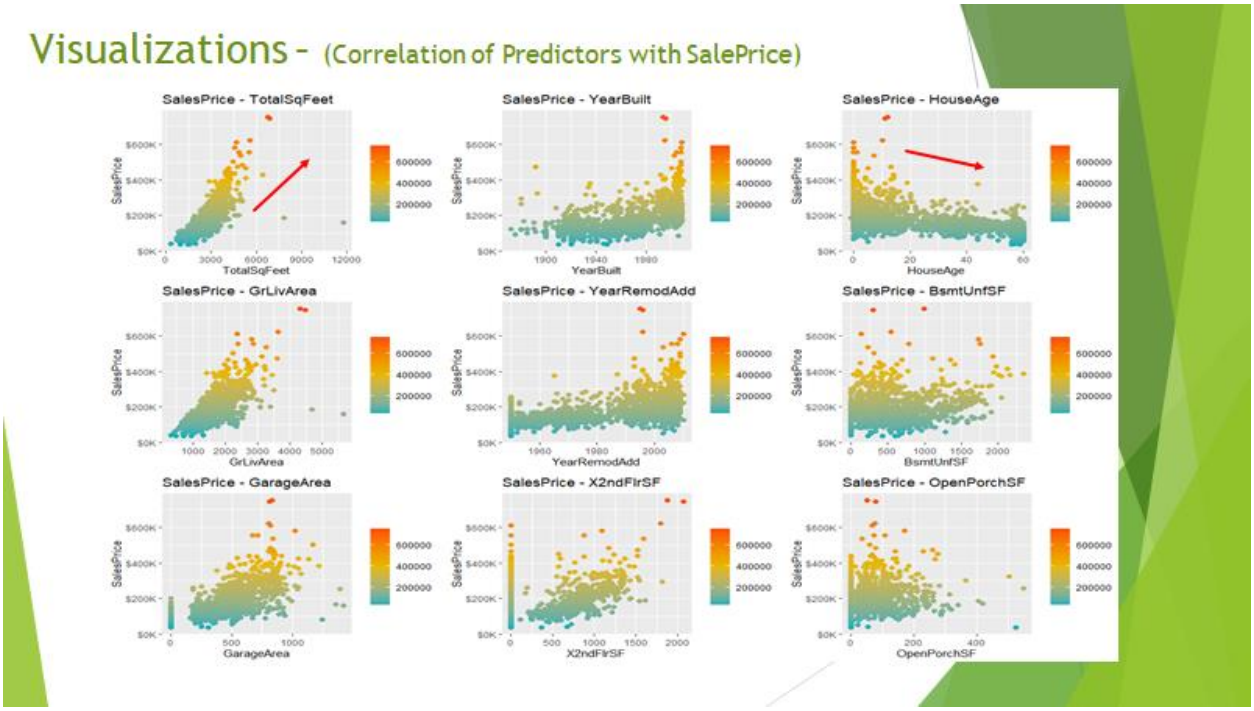

Figure A-13. Scatter Plot showing predictor correlation with SalePrice

Dataframes/Scenarios Blue = Regression D.Frames Grey = Tree D.Frames Yellow = Test D.Frames	W/O NZ on Categorical Predictors	W/O NZ. Drop Correlated Predictors Manually	W/O NZ. Outliers Removed Manually	W/O. NZ. Drop Correlated (P) + Outliers Removed Manually	W NZ on Categorical Predictors	W NZ. Drop Correlated Predictors Manually	W NZ. Outliers Removed Manually	W NZ. Drop Correlated (P) + Outliers Removed Manually
House_Trnf_Dum_Less	x							
House_Trnf_Dum_Less1					x			
House_Trnf_Cor_Less		x						
House_Trnf_Cor_Less1						x		
RmOtlr_House_Trnf_Dum_Less			x					
RmOtlr_House_Trnf_Dum_Less1							x	
RmOtlr_House_Trnf_Cor_Less				x				
RmOtlr_House_Trnf_Cor_Less1								x
HouseTest_Trnf_Dum_Less	x		x					
HouseTest_Trnf_Cor_Less		x		x				
HouseTest_Trnf_Dum_Less1					x		x	
HouseTest_Trnf_Cor_Less1						x		x

House_Trnf_DumAll1	x							
House_Trnf_DumAll					x			
RmOtlr_House_Trnf_DumAll1			x					
RmOtlr_House_Trnf_Cor_DumAll1				x				
HouseTest_Trnf_DumAll1	x		x					
HouseTest_Trnf_Cor_DumAll1				x				

Figure A-14. Data Preprocessing – Create Dataframes

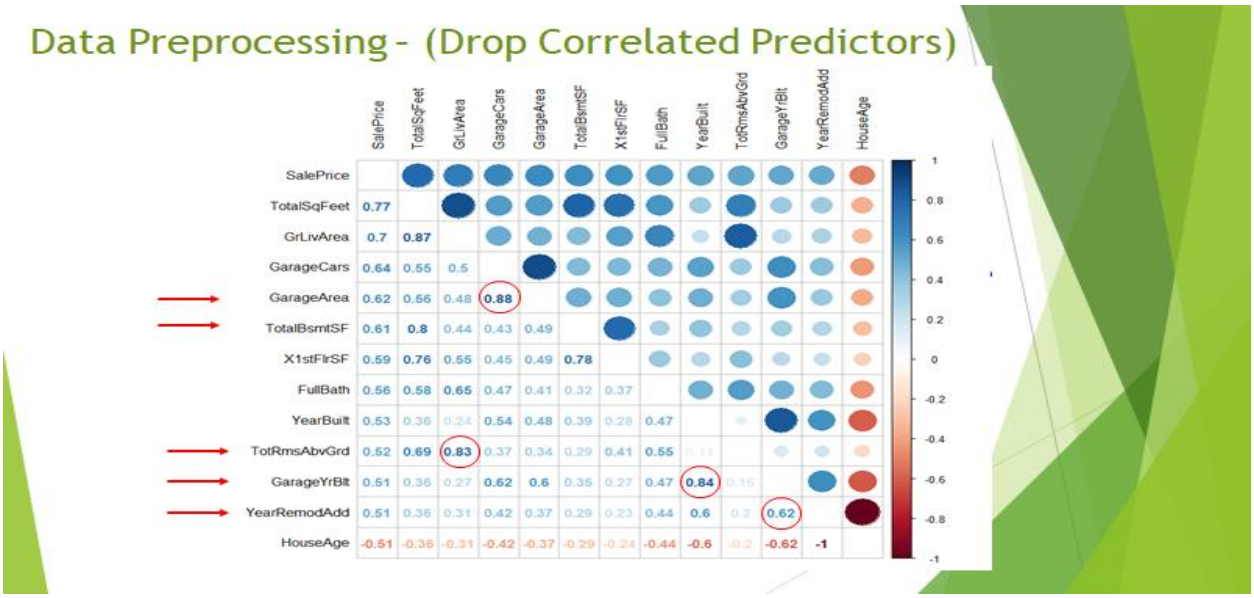

Figure A-15. Data Preprocessing – Correlated Predictor Identification

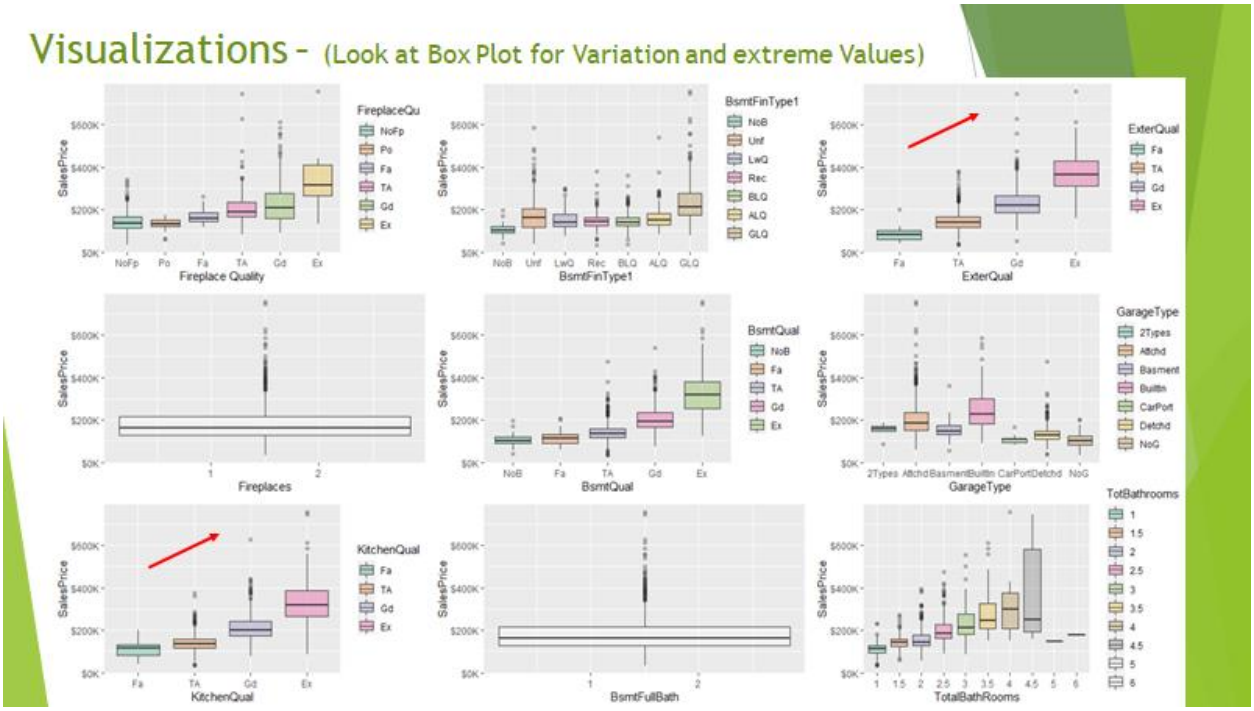

Figure A-16. Box Plot showing extreme values and predictor correlation with SalePrice

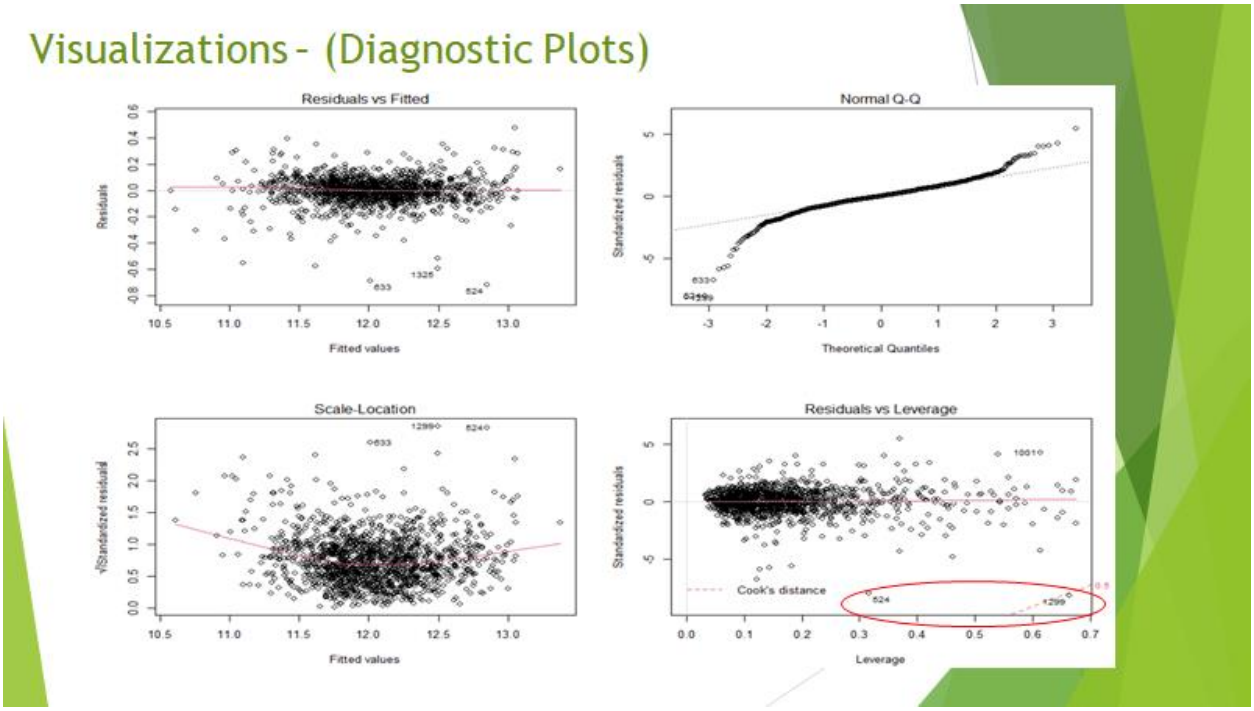

Figure A-17. Diagnostic Plot showing Linearity assumptions and outliers

APPENDIX B – TUNING VISUALIZATIONS

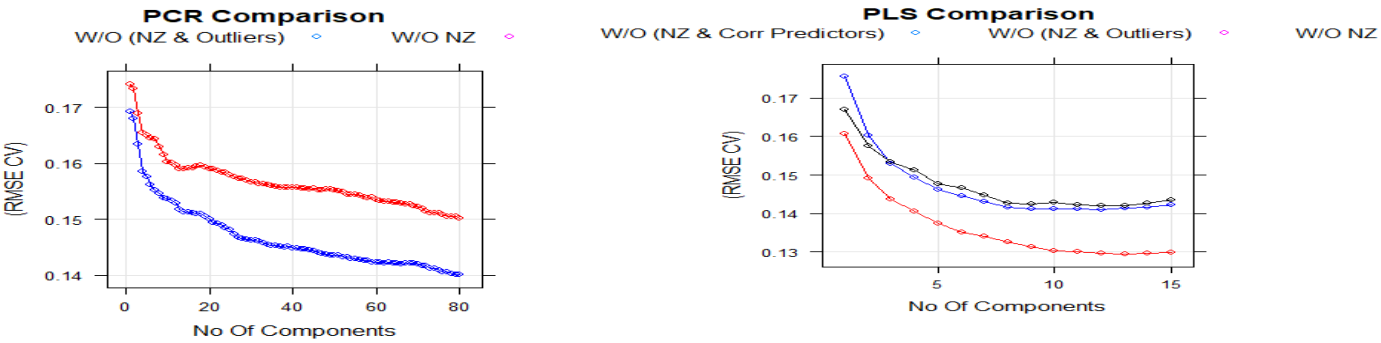

Figure B-1: Comparison of Tuning Components for PCR vs PLS

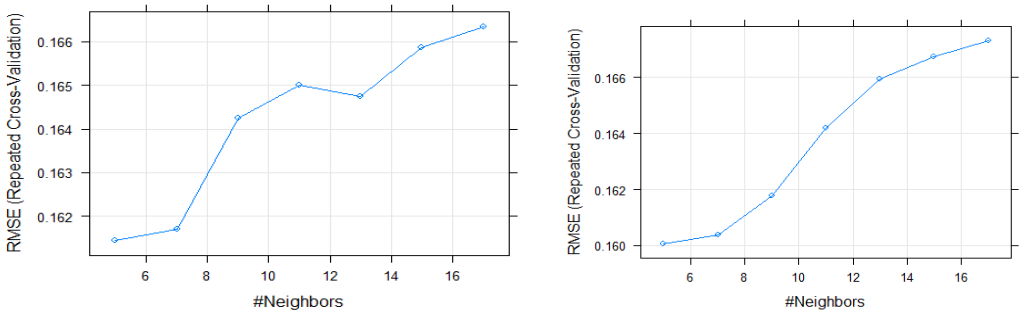

Figure B-2: K-Nearest Network Baseline vs Optimal Model (Baseline + Drop Corr predictors + Drop Outliers)

Baseline (left: best RMSE 0.161 @ k=5) Optimal Model (right: best RMSE 0.160 @ k=7)

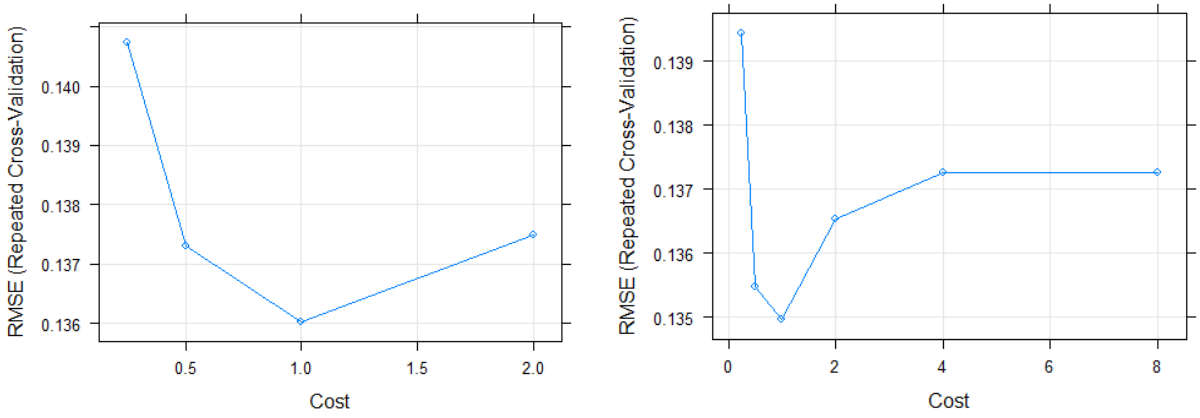

Figure B-3: Support Vector Machine (Baseline (left: best RMSE 0.136 @ Cost=1) Optimal Model (right: best RMSE 0.135 @ Cost=1)

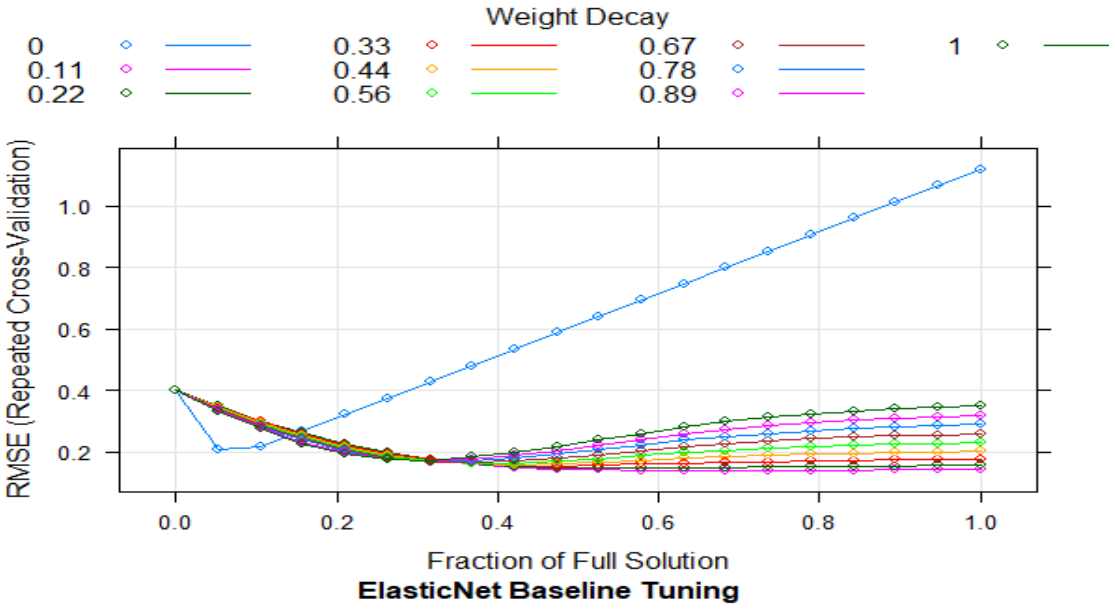

Figure B-4: ElasticNet Tuning – RMSE by Weight Decay and Fraction

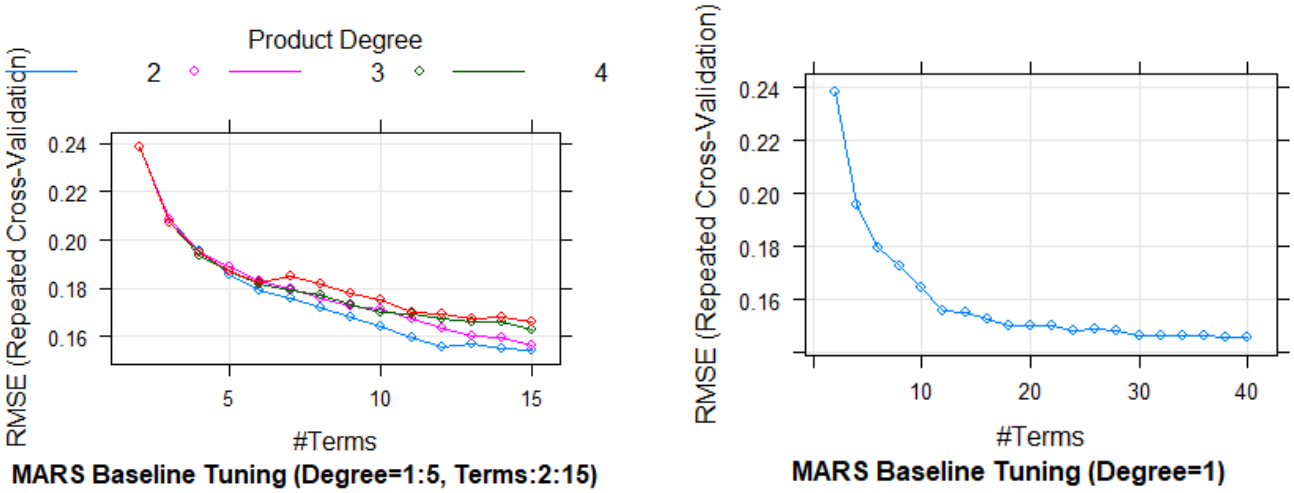

Figure B-5: MARS Model Incremental Tuning Exercise Example

APPENDIX C - R SAMPLE TRAINING MODEL OUTPUT

MARS Training Models performance Tuning

Training Model (Baseline) Output

```
> marsModel
```

Multivariate Adaptive Regression Spline

1097 samples
257 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 989, 988, 987, 986, 987, 987, ...
Resampling results across tuning parameters:

nprune	RMSE	Rsquared	MAE
2	0.2385729	0.6544718	0.17710175
4	0.1956756	0.7677123	0.14133851
6	0.1793666	0.8054290	0.12951704
8	0.1723166	0.8199990	0.12344527
10	0.1642804	0.8368429	0.11740215
12	0.1559862	0.8526639	0.11101976
14	0.1553618	0.8532262	0.10886878
16	0.1523193	0.8583435	0.10478663
18	0.1499043	0.8630440	0.10155361
20	0.1498212	0.8638834	0.10019790
22	0.1498251	0.8642974	0.09970472
24	0.1480319	0.8673436	0.09826148
26	0.1486835	0.8655984	0.09804293
28	0.1480157	0.8671185	0.09736330
30	0.1465993	0.8701388	0.09661699
32	0.1459879	0.8712596	0.09589256
34	0.1460736	0.8710893	0.09573459
36	0.1459932	0.8709613	0.09610720
38	0.1454202	0.8718192	0.09559634
40	0.1454675	0.8717118	0.09562050

Tuning parameter 'degree' was held constant at a value of 1
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were nprune = 38 and degree = 1.

Training Model – Optimal Model (Baseline + Drop Corr pred and outliers) Output

```
> marsotlrModel
```

Multivariate Adaptive Regression Spline

1095 samples
252 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 986, 986, 986, 986, 985, 986, ...
Resampling results across tuning parameters:

nprune	RMSE	Rsquared	MAE
2	0.2340990	0.6574014	0.17509204
4	0.1888814	0.7772803	0.13830988
6	0.1720113	0.8157459	0.12666320
8	0.1654772	0.8289680	0.12163468
10	0.1587079	0.8427495	0.11626230
12	0.1535083	0.8524336	0.11166950
14	0.1477414	0.8634990	0.10590764
16	0.1414615	0.8752346	0.10136606
18	0.1369342	0.8828872	0.09780739
20	0.1334077	0.8890564	0.09554196
22	0.1301111	0.8937594	0.09279061
24	0.1262610	0.8999811	0.09020183
26	0.1252131	0.9016962	0.08901850
28	0.1249638	0.9022104	0.08839135
30	0.1247538	0.9022992	0.08786887
32	0.1244901	0.9027652	0.08743854
34	0.1247347	0.9024377	0.08747180
36	0.1246588	0.9025848	0.08726841
38	0.1247081	0.9024714	0.08731201
40	0.1246517	0.9025273	0.08725858

Tuning parameter 'degree' was held constant at a value of 1
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were nprune = 32 and degree = 1.

APPENDIX D – MODEL RESULTS

Table D-1: Home price sale prediction - **Baseline Scenario 1 and 2 (With and W/O Nearzero)**
Refer to Note 3 for explanation

Without Near Zero Function on Categorical Predictors							
Model	Train RMSE	Train R^2	Train RMSE SD	Train R^2 SD	Test RMSE	Test R^2	Hyperparameter Tuning
Multiple Linear Regression (OLS)	0.140	0.882	0.029	0.044	0.150	0.859	See Note 1 below
Prin. Comp. Regression (PCR)	0.151	0.865	0.0108	0.0211	0.172	0.811	ncomp=80
Partial Least Squared (PLS)	0.142	0.880	0.009	0.0164	0.146	0.862	ncomp=12
Multi Variate. Reg. Splines (MARS)	0.135	0.882	0.021	0.037	0.162	0.847	degree=1, nprune=38
Support Vector Machine (SVM) with PCA	0.137	0.880	0.023	0.035	0.153	0.867	c=1.0
Neural Network	0.140	0.863	0.009	0.017	0.149	0.871	size=1, decay=0.45
Av. Neural Network	0.130	0.890	0.009	0.016	0.145	0.880	size=2, decay=0.65
K-Nearest Network	0.174	0.837	0.019	0.313	0.174	0.837	k=5 (base)
Simple Regression Tree	0.190	0.784			0.193	0.763	cp = seq(0,0.1,0.01)
Random Forest	0.159	0.868			0.149	0.863	mtry = 8; 8
Boosted Model					28993	0.866	
XgBoost	0.134	0.890			0.131	0.883	See Note 2 below
With Near Zero Function on Categorical Predictors							
Robust Linear Regression with PCA	0.138	0.884	0.028	0.044	0.174	0.806	See Note 1 below
Ridge	0.143	0.871	0.022	0.037	0.260	0.660	Lambda=0.03
ElasticNet	0.140	0.875	0.024	0.041	0.150	0.870	Lambda=0, Fraction=0.16

Note 1: For OLS, Robust Linear Regression models, correlated variables were dropped from the baseline dataframe

Note 2: XgBoost Tuning: nrounds = c(100,200),max_depth = c(10, 15, 20, 25),colsample_bytree = seq(0.5, 0.9, length.out = 5),eta = 0.1,gamma=0,min_child_weight = 1,subsample = 1

Note 3: For the Robust Linear Regression with PCA, Ridge and ElasticNet models, the baseline dataframe was additionally preprocessed to eliminate nearzero predictors as these models fail without this step

Table D-2: Home price sale prediction – **Scenario 3 (Baseline + Drop Outliers Only)**

With and W/O Nearzero Refer to Note 3 for explanation

Without Near Zero Function on Categorical Predictors							
Model	Train RMSE	Train R^2	Train RMSE SD	Train R^2 SD	Test RMSE	Test R^2	Hyperparameter Tuning
Multiple Linear Regression (OLS)							
Prin. Comp. Regression (PCR)	0.140	0.880	0.007	0.001	0.139	0.883	ncomp=79
Partial Least Squared (PLS)	0.130	0.897	0.007	0.011	0.121	0.911	ncomp=13
Multi Variate. Reg. Splines (MARS)							
Support Vector Machine (SVM) with PCA	0.137	0.885			0.136	0.880	
Neural Network							
Av. Neural Network	0.127	0.900			0.123	0.900	
K-Nearest Network							
Simple Regression Tree	0.190	0.774			0.197	0.772	cp = seq(0,0.1,0.01)
Random Forest	0.155	0.872			0.153	0.875	mtry = 8; 8
Boosted Model					27424	0.907	See Note 2
XgBoost	0.125	0.900			0.132	0.893	See Note 1
With Near Zero Function on Categorical Predictors							
Robust Linear Regression with PCA	0.131	0.893	0.015	0.020	0.138	0.885	
Ridge					0.135	0.883	
ElasticNet							

Note 1: XGBoost Tuning: nrounds = c(100,200),max_depth = c(10, 15, 20, 25),colsample_bytree = seq(0.5, 0.9, length.out = 5),eta = 0.1,gamma=0,min_child_weight = 1,subsample = 1

Note 2: Boosted Model Tuning: n.trees =2000, interaction.depth=6, shrinkage=0.1,bag.fraction = .75

Note 3: For the Robust Linear Regression with PCA, Ridge and ElasticNet models the baseline dataframe was additionally processed to eliminate nearzero predictors as these models fail without this step

Table D-3: Home price sale prediction – **Scenario 4 (Baseline + Drop Multicollinear Predictors + Drop Outliers)** Refer to Note 3 for explanation

Without Near Zero Function on Categorical Predictors							
Model	Train RMSE	Train R^2	Train RMSE SD	Train R^2 SD	Test RMSE	Test R^2	Hyperparameter Tuning
Multiple Linear Regression (OLS)	**0.124**	**0.903**	**0.016**	**0.021**	**0.127**	**0.903**	
Prin. Comp. Regression (PCR)	**0.140**	**0.880**	**0.007**	**0.001**	**0.139**	**0.883**	**ncomp=79**
Partial Least Squared (PLS)	**0.129**	**0.898**	**0.007**	**0.0109**	**0.121**	**0.912**	**ncomp=14**
Multi Variate. Reg. Splines (MARS)	**0.124**	**0.905**	**0.016**	**0.023**	**0.141**	**0.870**	**degree=1, nprune=36**
Support Vector Machine (SVM) with PCA	**0.137**	**0.885**	**0.022**	**0.033**	**0.136**	**0.880**	**c=1.0**
Neural Network	**0.134**	**0.890**	**0.009**	**0.012**	**0.131**	**0.890**	**size=1, decay=0.47**
Av. Neural Network	**0.127**	**0.900**	**0.008**	**0.01**	**0.125**	**0.900**	**size=2, decay=0.65**
K-Nearest Network	**0.160**	**0.853**	**0.211**	**0.026**	**0.167**	**0.830**	**k=7**
Simple Regression Tree	**0.191**	**0.772**			**0.197**	**0.772**	**cp = seq(0,0.1,0.01)**
Random Forest							**mtry = 8; 8**
Boosted Model					**26914**	**0.911**	**See Note 1 below**
XgBoost	**0.125**	**0.901**			**0.132**	**0.894**	**See Note 2 below**
With Near Zero Function on Categorical Predictors							
Robust Linear Regression with PCA							
Ridge	**0.130**	**0.900**			**0.133**	**0.884**	**Lambda=0**
ElasticNet	**0.130**	**0.900**			**0.133**	**0.884**	**Lambda=0, Fraction =0.47**

Note 1: XGBoost Tuning: nrounds = c(100,200),max_depth = c(10, 15, 20, 25),colsample_bytree = seq(0.5, 0.9, length.out = 5),eta = 0.1,gamma=0,min_child_weight = 1,subsample = 1

Note 2: Boosted Model Tuning: n.trees =2000, interaction.depth=6, shrinkage=0.1,bag.fraction = .75

Note 3: For the Ridge and ElasticNet models the baseline dataframe was additionally processed to eliminate nearzero predictors as these models fail without this step

		Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)		10.8297590	0.3102401	34.908	< 2e-16	***
LotArea	(X1)	0.0403421	0.0076601	5.267	1.76e-07	***
YearBuilt	(X2)	0.0519585	0.0127370	4.079	4.94e-05	***
BsmtUnfSF	(X3)	-0.0274143	0.0073327	-3.739	0.000197	***
GarageCars	(X4)	0.0263437	0.0062014	4.248	2.39e-05	***
HouseAge	(X5)	-0.0225256	0.0063339	-3.556	0.000397	***
TotalSqFeet	(X6)	0.1366860	0.0201016	6.800	1.97e-11	***
MSZoning_FV	(X7)	0.4268307	0.0582107	7.333	5.24e-13	***
MSZoning_RH	(X8)	0.4963646	0.0619685	8.010	3.74e-15	***
MSZoning_RL	(X9)	0.4160921	0.0488282	8.522	< 2e-16	***
MSZoning_RM	(X10)	0.3596313	0.0448607	8.017	3.55e-15	***
KitchenQual_TA	(X11)	-0.0725566	0.0204291	-3.552	0.000404	***
KitchenQual_Gd	(X12)	-0.0693725	0.0184262	-3.765	0.000178	***
Residual standard error: 0.09795						

y= 10.83 + 0.04X1+0.05X2-0.03X3+0.03X4-0.02X5+0.14X6+0.43X7+0.5X8+0.42X9+0.36X10-0.07X11-0.07X12+εi^2 where ε~N(0, 0.09795^2)

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i + \hat{\varepsilon}_i$$
$$\text{where } \varepsilon \sim \mathcal{N}(0,\, \hat{\sigma}^2)$$

Figure D-1: Multiple Linear Regression (OLS) Summary & Regression Equation

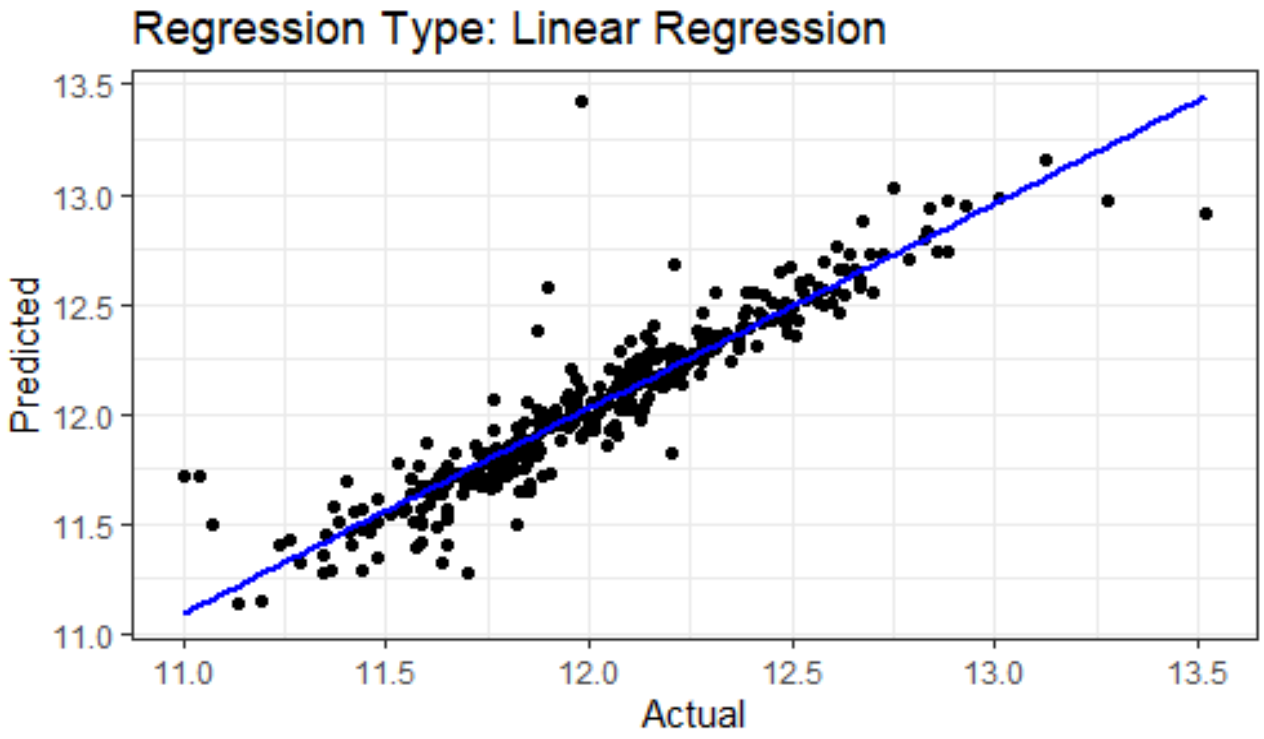

Figure D-2: Multiple Linear Regression (OLS) Actual vs Predicted Value Chart (Logarithmic Scale)

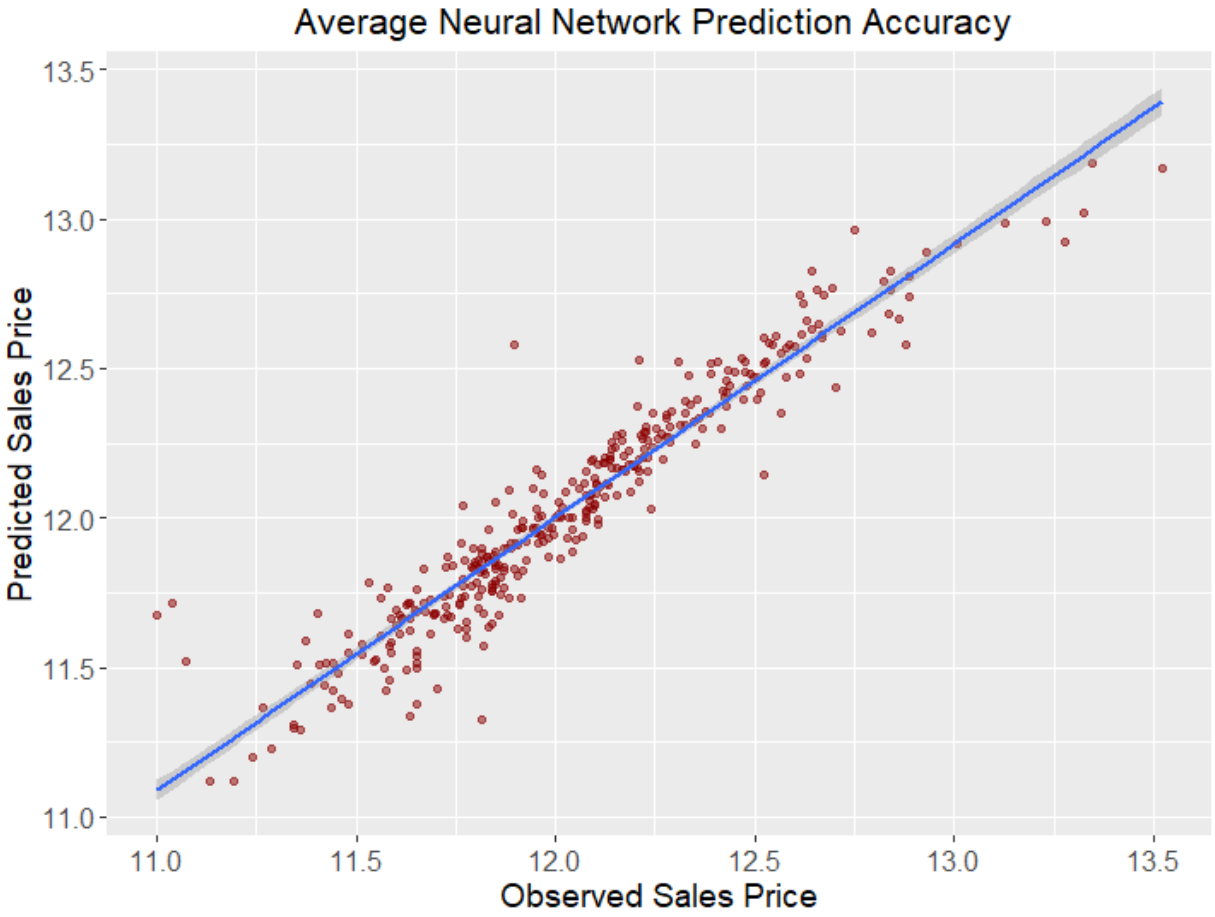

Figure D-3: Average Neural Networks - Actual vs Predicted Value Chart (Logarithmic Scale)

APPENDIX E – RESULTS OF MODEL APPLICATION TO TEST DATA

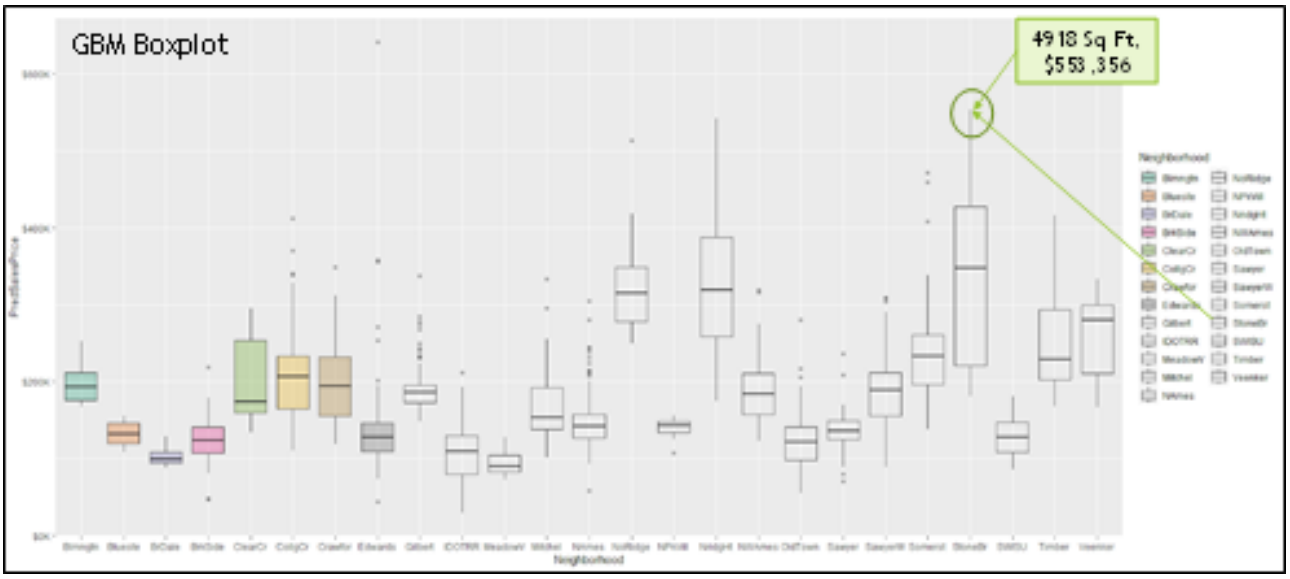

Figure E-1: Sample Property Sale Price Prediction from GBM Model

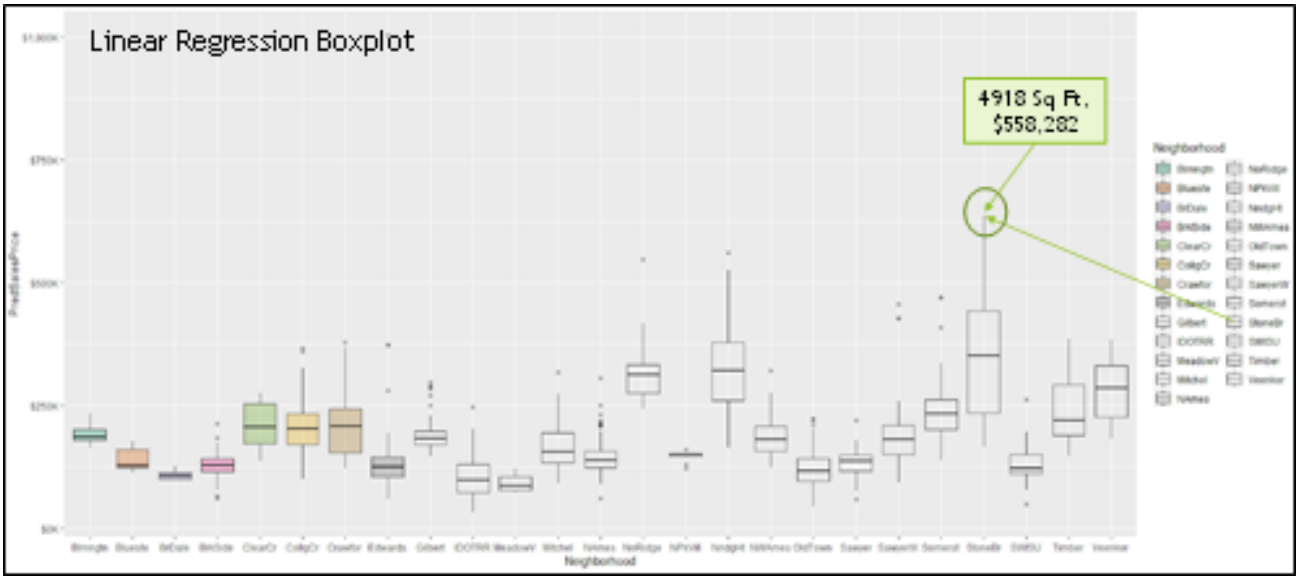

Figure E-2: Sample Property Sale Price Prediction from OLS Model

BIBLIOGRAPHY

Barsch, C. (2019, July 31). *From the Global Stage to Your Backyard: 10 Factors that Influence Real Estate Property Values*. Retrieved from Homelight.com: https://www.homelight.com/blog/real-estate-property-value/

Berwick, R. (2011). *Artificial Intelligence - Recitations*. Retrieved from An Idiot's guide to Support vector: http://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf

DeCock, D. (2011). Ames, IA: An Alternative to the Boston Housing Data project. *Journal of Statistics Education, Vol 19, Number 3*, 15.

Kaggle. (2016). *Kaggle*. Retrieved from House Prices: Advanced Regression Techniques: https://www.kaggle.com/c/house-prices-advanced-regression-techniques/overview

Kuhn M. and Johnson K. (2016*) Applied Predictive Modeling.* Springer Science + Business Media.

Realtor.com. (2020, November). *Ames, IA Real Estate Market*. Retrieved from Realtor.com: https://www.realtor.com/realestateandhomes-search/Ames_IA/overview

Tarafder S. (2020, December). HOW TO AVOID COMMON MISTAKES IN LINEAR REGRESSION. Retrieved from isixsigma.com: https://www.isixsigma.com/tools-templates/regression/how-to-avoid-common-mistakes-in-linear-regression/