Capstone House Prices Project

HarvardX (edX) Data Science Professional Certificate: PH125.9x Capstone

Jonathan Ostler

2021-05-28

Contents

1 Abstract

Data science has become the super-power behind the success of many businesses of today. These companies need and use data science approaches like machine learning to extract patterns, trends and other actionable information from large data sets (Galwey, 2014). In turn, companies use these findings to improve performance, grow and, perhaps most importantly, to provide better products and services to consumers (Irizarry, 2020).

Not surprisingly, prediction and recommendation systems are increasingly used by businesses to provide personalized insights to consumers. Often companies make competitions out of the challenge to stimulate ideas and of course produce a better, more workable and more dependable model.

One of the most popular competitions on kaggle is the "House Prices: Advanced Regression Techniques" competition. This competition was the inspiration for this project and the subject of this report.

Ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. But the dataset used here proved that much more influences price negotiations than the number of bedrooms or a white-picket fence.

With 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, U.S.A, the aim was to predict the Sale Prices with the highest possible accuracy.

The Ames Housing dataset (covering sales during the early 2000s) was compiled by Dean De Cock for use in data science education. It's an alternative for data scientists looking for a modernized and expanded version of the often cited Boston Housing dataset.

This report details some of the possible techniques used to predict the Sale Price and how they were built up from first principles.

The kaggle challenge used the standard error loss function, Root Mean Squared Errors (**RMSE**), based on the logarithms of the predicted and observed sale prices:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i}^{N}(log(\hat{y}_i) - log(y_i))^2}$$

with the winner being chosen based on the lowest RMSE value. (Taking logs meant that errors in predicting expensive houses and cheap houses would affect the result equally.)

This report shows the evolution of the RMSE value from an initial simple solution to the ultimate solution as more complex models were created.

Since the ultimate sale price in the validation data was not provided, this report based its final RMSE value on a sample of the initial training data for validation/verification.

The GitHub repository for this report can be found at : https://github.com/ostlerjo/CapstoneHousePrices.git

2 Introduction

2.1 Data Used

The original data came from the publication Dean De Cock "Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project", Journal of Statistics Education, Volume 19, Number 3 (2011).

The data used in this report was obtained from kaggle.com.

2.2 Libraries Used

The following r libraries were used to generate the data, perform the analysis and create the algorithms detailed in this report:

```
library(caret)
library(data.table)
library(Boruta)
library(plyr)
library(dplyr)
library(pROC)
library(tidyverse)
library(FeatureHashing)
library(Matrix)
library(xgboost)
require(randomForest)
require(ggplot2)
library(stringr)
library(dummies)
library(Metrics)
library(kernlab)
library(corrplot)
library(car)
library(lars)
library(kableExtra)
```

3 Data Cleaning and Preparation

3.1 Initial Data Cleaning

Due to the way R treats variable names, three columns were mutated to put a dummy X character in front e.g. 1stFlrSF became X1stFlrSF etc.

Secondly several columns had false NAs in them that would cause the code to give errors e.g. the column "Alley" had a set of NAs which in fact meant "No Alley". As a consequence, all of these types of values were explored and then updated to their "correct" values.

Finally, a couple of numeric columns also had NAs when they should in fact have been 0. As a consequence, all of these types of values were updated to be 0 rather than NA.

3.2 Data Partitioning

Two sets of data were available i.e. a train set and a test set. Since the data was originally going to be used in a competition to predict the Sale Prices, the test set provided did not have a Sale Price column. As a consequence for this project where predictive results needed to be reported, the train data set was used for training, testing and the final validation through partitioning.

Initially the train set was partitioned into a Modeling set and a Final Validation set. The split was 90% / 10% in favor of the Modeling set. The small percentage assigned to the Final Validation set was because the overall data set did not have so many values (around 1'500) and most of the data should be used for modeling. The Final Validation set was set aside and not used again until the Final RMSE calculation.

Next the Modeling set was further partitioned into a Training set and a Testing set that would be used for analysis, model creation, training and initial validation. The split was 60 % / 40% in favor of the Training set. The split was chosen to be almost equal but at the same time due to the restricted number of data points, a slightly larger value was assigned to the Training set.

4 Data Exploration

It is a good practice to first understand the data itself before attempting to gather insights or draw conclusions (Viswanathan, 2015). Exploratory data analysis is a necessary and critical process for preforming these initial investigations on data. This process allows investigators to discover patterns, spot anomalies, test hyptheses, and to scrutinize assumptions with the assistance of summary statistics and graphical representations (visualizations).

The raw data contained 1'460 rows (each representing a sale of a property) and 81 columns (i.e. an Id, a Sale Price and 79 potential predictors of the Sale Price).

The data covered 25 neighborhoods in Ames, Iowa, U.S.A.

After partitioning, the Final Validation data had 148 rows, the Testing data had 527 rows and the Training data had 785 rows.

The training data covered properties built from 1872 to 2009 with Sale Prices ranging from as low as USD 34'900 to USD 755'000. The training data captured actual sales from 2006 to 2010.

Of the 79 potential predictor columns, 44 were characters and the remaining 35 were numeric. Later on in the analysis, some of the key character variables wereconverted to numerical values to assist with the algorithm building.

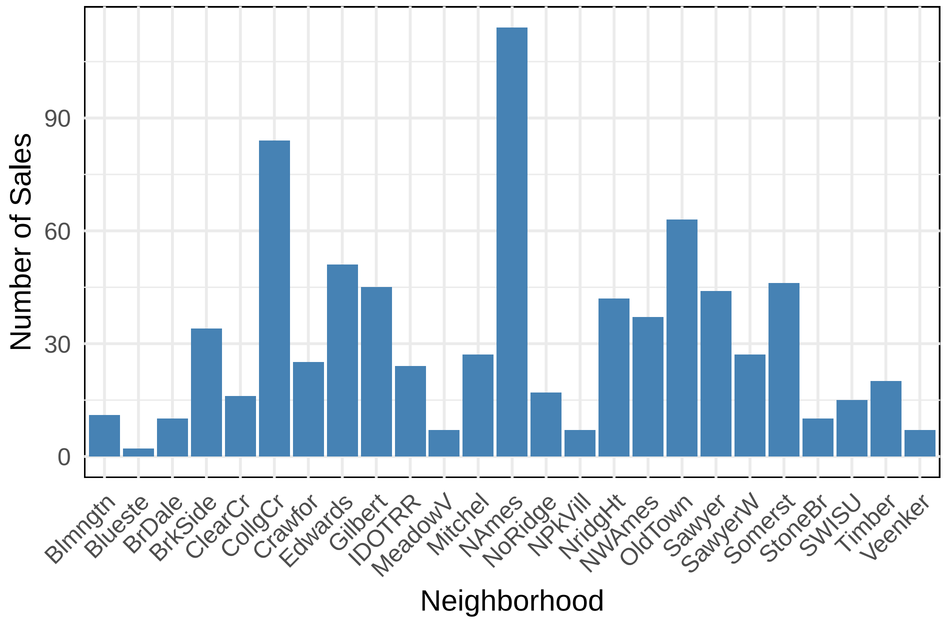

Figure 1: Location of Sales

Figure 1 shows that sales occurred in many different neighborhoods and so this should be an indicator that it will be an important factor in the algorithm.

4.1 Boruta Importance Analysis

Exploring a dataset could be difficult when the number of variables was as large as it was i.e. 79. As a result, it made sense to start with a so called Boruta Feature Importance analysis to show which items were deemed important. See the appendix for more background on how the algorithm worked.

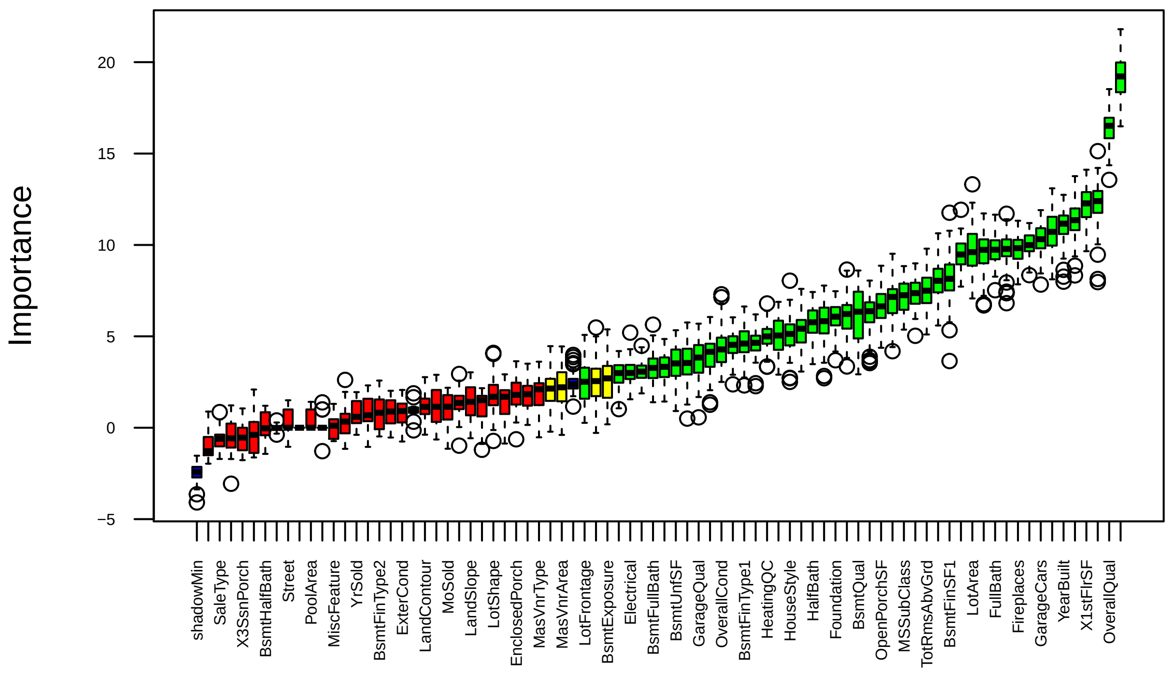

Figure 2: Boruta Importance Analysis

Figure 2 shows the relative importance of each candidate explanatory attribute. The x-axis represents each of candidate explanatory variables. Green boxes indicate the attributes that were relevant to prediction. Red boxes indicate attributes that were not relevant. Yellow boxes indicate attributes that may or may not be relevant to predicting the Sales Price.

Of the 79 variables, 46 were deemed important, 29 were deemed not important and the remaining 4 were considered “tentative”.

Table 1, Table 2 and Table 3 list all of the variables with their “importance”.

Table 1: Confirmed Variables

	medianImp	decision
GrLivArea	19.218927	Confirmed
OverallQual	16.518309	Confirmed
TotalBsmtSF	12.398605	Confirmed
X1stFlrSF	12.273444	Confirmed
GarageArea	11.353860	Confirmed
YearBuilt	11.163964	Confirmed
X2ndFlrSF	10.716789	Confirmed
GarageCars	10.313716	Confirmed
GarageType	9.992553	Confirmed
Fireplaces	9.822708	Confirmed
KitchenQual	9.781477	Confirmed
FullBath	9.739102	Confirmed
YearRemodAdd	9.732214	Confirmed
LotArea	9.606212	Confirmed
ExterQual	9.478835	Confirmed
BsmtFinSF1	8.149144	Confirmed
MSZoning	8.038653	Confirmed
TotRmsAbvGrd	7.499223	Confirmed
Neighborhood	7.365211	Confirmed
MSSubClass	7.248376	Confirmed
GarageYrBlt	7.156602	Confirmed
OpenPorchSF	6.639716	Confirmed
BedroomAbvGr	6.374635	Confirmed
BsmtQual	6.348388	Confirmed
CentralAir	6.220869	Confirmed
Foundation	6.077120	Confirmed
GarageFinish	5.827744	Confirmed
HalfBath	5.769889	Confirmed
BldgType	5.419458	Confirmed
HouseStyle	5.132012	Confirmed
FireplaceQu	5.046505	Confirmed
HeatingQC	4.996898	Confirmed
GarageCond	4.647962	Confirmed
BsmtFinType1	4.605484	Confirmed
KitchenAbvGr	4.539544	Confirmed
OverallCond	4.283791	Confirmed
Exterior1st	4.153420	Confirmed
GarageQual	3.839559	Confirmed
Exterior2nd	3.544360	Confirmed
BsmtUnfSF	3.518722	Confirmed
RoofStyle	3.332114	Confirmed
BsmtFullBath	3.273959	Confirmed
PavedDrive	3.069011	Confirmed
Electrical	2.998243	Confirmed
Alley	2.993329	Confirmed
LotFrontage	2.515646	Confirmed

Table 2: Tentative Variables

	medianImp	decision
BsmtExposure	2.701795	Tentative
WoodDeckSF	2.552385	Tentative
MasVnrArea	2.210707	Tentative
ScreenPorch	2.142979	Tentative

Table 3: Rejected Variables

	medianImp	decision
MasVnrType	2.1026243	Rejected
Functional	1.8357503	Rejected
EnclosedPorch	1.7991222	Rejected
SaleCondition	1.6963043	Rejected
LotShape	1.6959792	Rejected
BsmtFinSF2	1.5075910	Rejected
LandSlope	1.4179034	Rejected
BsmtCond	1.3591696	Rejected
MoSold	1.1509607	Rejected
Fence	1.1476007	Rejected
LandContour	1.1418781	Rejected
RoofMatl	0.9716203	Rejected
ExterCond	0.9124730	Rejected
Condition1	0.8872872	Rejected
BsmtFinType2	0.8207598	Rejected
Heating	0.6807009	Rejected
YrSold	0.5983857	Rejected
LotConfig	0.3363768	Rejected
MiscFeature	0.1098972	Rejected
Street	0.0000000	Rejected
Utilities	0.0000000	Rejected
PoolArea	0.0000000	Rejected
PoolQC	0.0000000	Rejected
BsmtHalfBath	-0.0311961	Rejected
LowQualFinSF	-0.3897656	Rejected
X3SsnPorch	-0.5428625	Rejected
MiscVal	-0.5808725	Rejected
SaleType	-0.6028046	Rejected
Condition2	-1.2836306	Rejected

4.2 Correlations

Due to the large number of predictors, it was decided to initially focus on the exploration of the numeric variables using correlation plots. (Later in this report, the important character variables were also considered.)

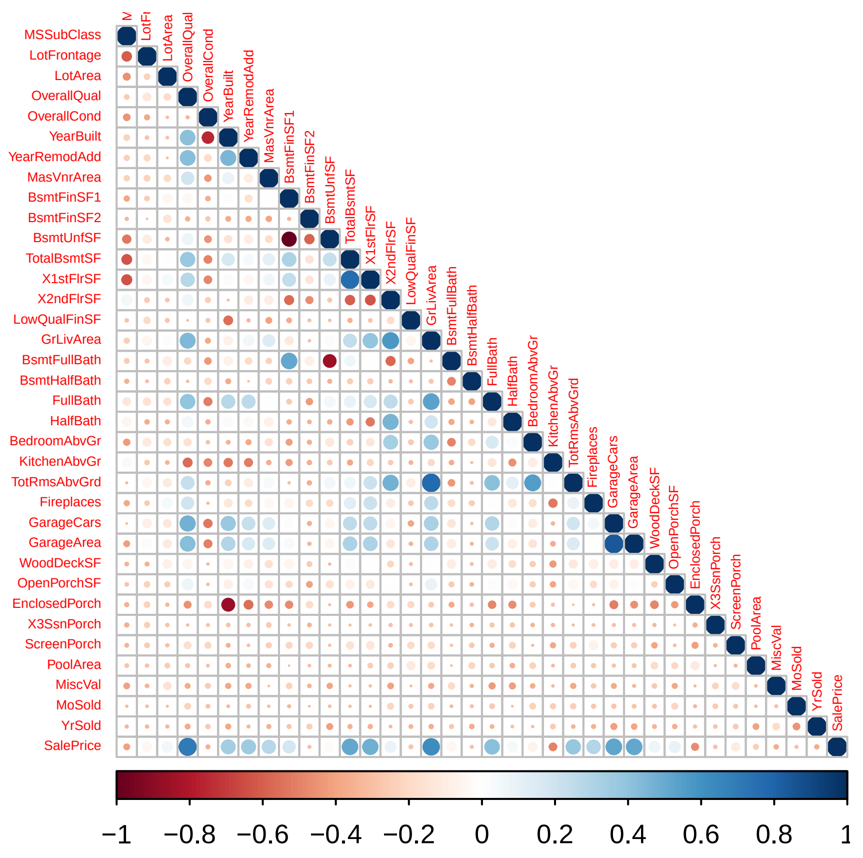

Figure 3: Correlation Matrix for Numerical Variables

Figure 3 shows that of the original 35 numeric variables, 13 of them were considered important (correlation > 0.35) and this tied in with the Boruta analysis. Table 4 lists these variables.

Table 4: Important Numeric Variables

OverallQual
YearBuilt
YearRemodAdd
MasVnrArea
BsmtFinSF1
TotalBsmtSF
X1stFlrSF
GrLivArea
FullBath
TotRmsAbvGrd
Fireplaces
GarageCars
GarageArea

4.3 Further Cleaning of Data and Correlations

From the Boruta analysis, of the top 20 variables only 5 were character variables. After converting to numerical values, the correlations with SalePrice are high. See Figure 4.

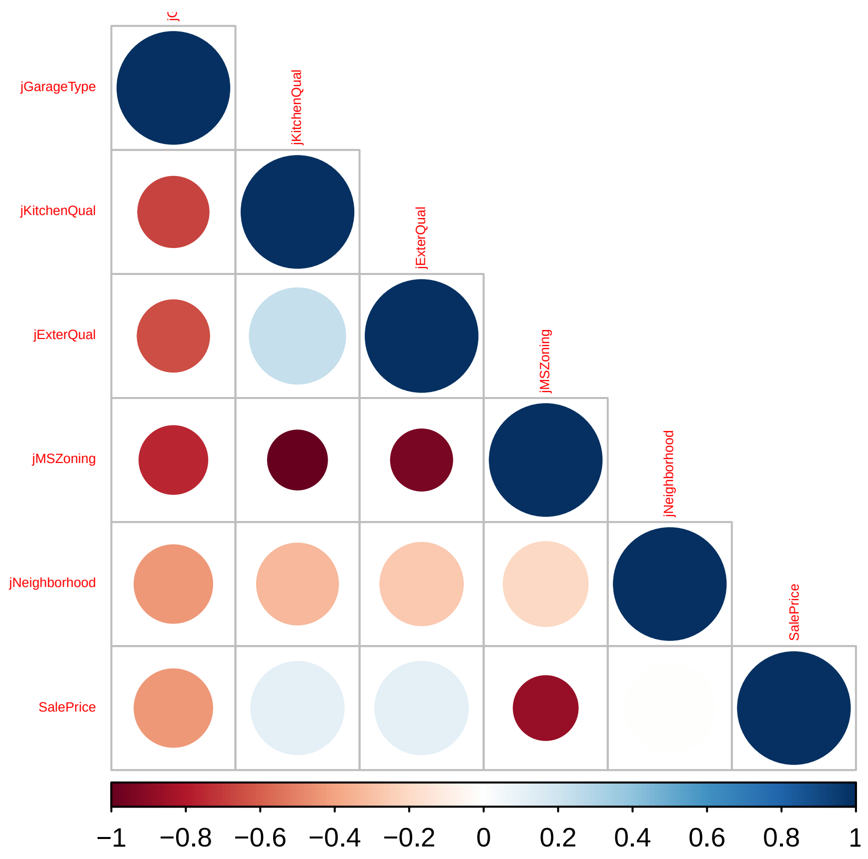

Figure 4: Correlation Matrix for Character Variables

Table 5 lists these variables.

Table 5: Important Character Variables

jGarageType
jKitchenQual
jExterQual
jMSZoning
jNeighborhood

4.4 Scatter Plots

The correlation chart (Figure 3) indicates there is a relationship between the sale price and some numeric variables, but what the relationship is is unknown. Using simple scatter plots the relationships (or not) can be more easily seen. See Figure 5, Figure 6 and Figure 7.

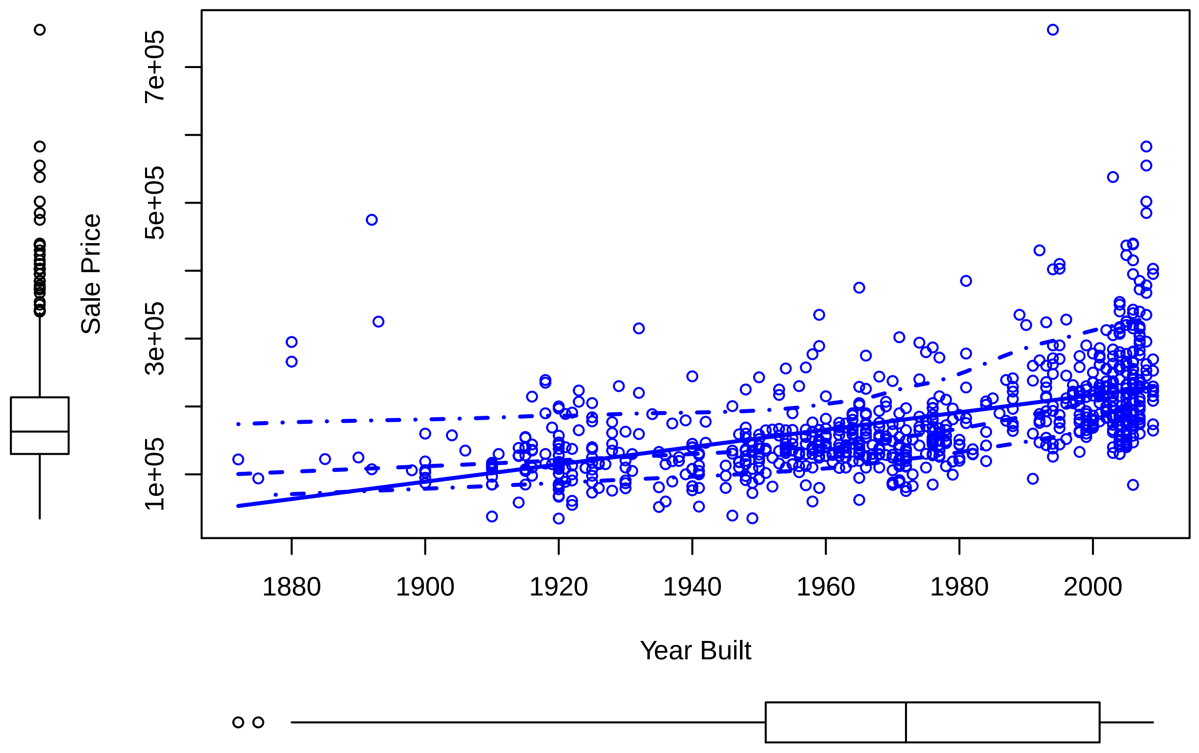

Figure 5: Sale Price vs. Year Built

Figure 5 shows that newer houses are worth more which makes sense. It is not difficult to see that the price of a property increases generally with the year built; the trend is obvious.

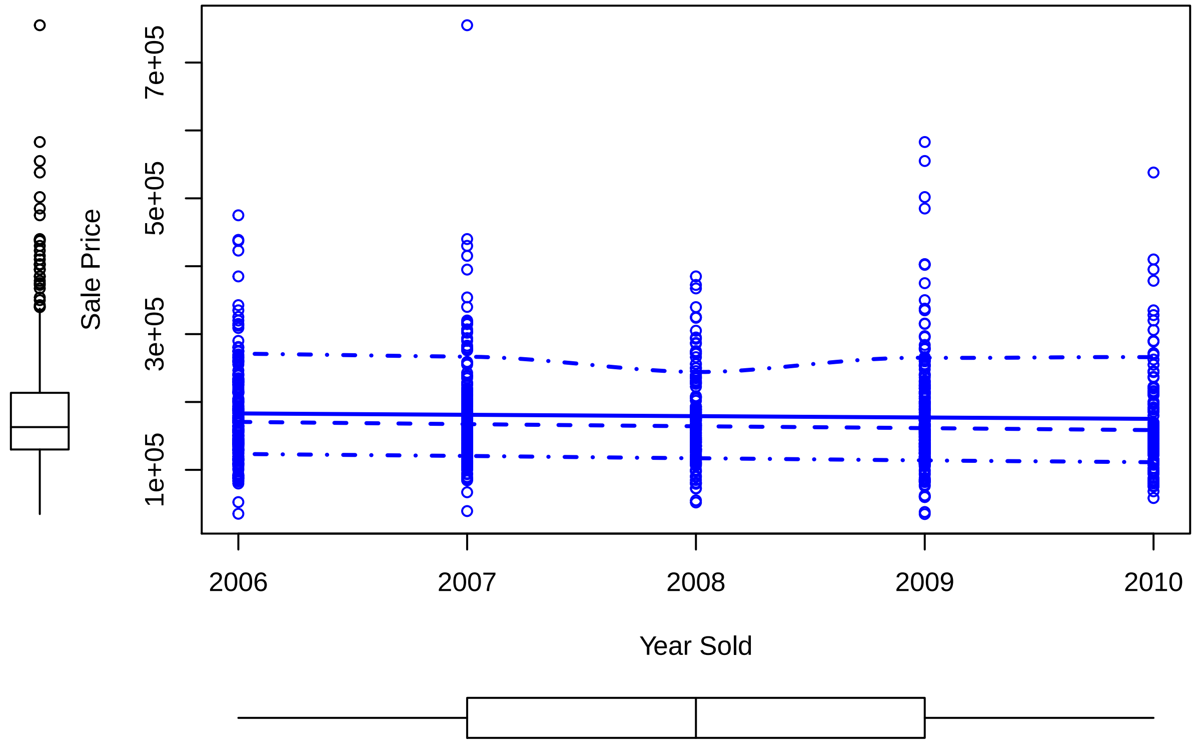

Figure 6: Sale Price vs. Year Sold

Figure 6 shows that sale prices dip in 2008 (during the financial crisis) but overall Year of Sale doesn't appear to be a major driver and can therefore be excluded.

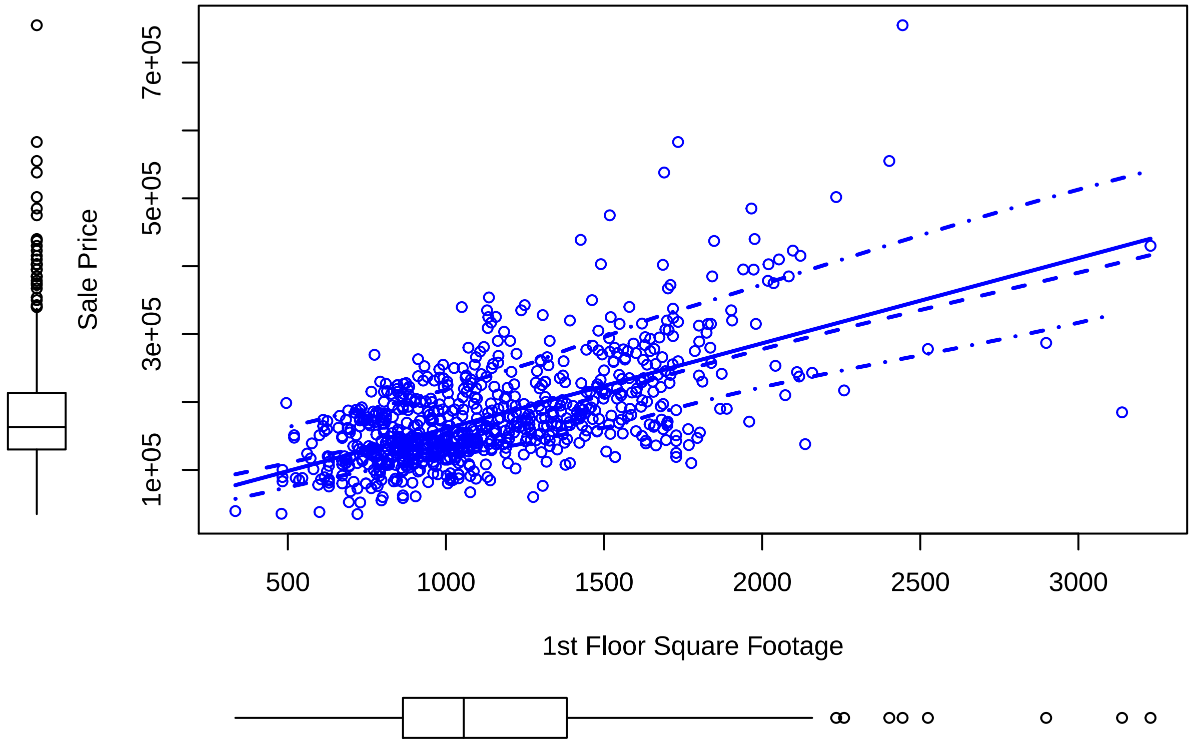

Figure 7: Sale Price vs. 1st Floor Square Footage

Figure 7 shows there were some strange outliers on the influence of first floor square footage on the sale price - it maybe bad data but they should not have a huge influence on the modelling.

4.5 Pairs Analysis

Using a scatterplot matrix to observe the bivariate relationship between different combinations of variables is useful. Each scatter plot in the matrix visualizes the relationship between a pair of variables, allowing many relationships to be explored in one chart.

The earlier correlation plots show that the SalesPrice variable was well correlated with a number of the independent variables. In Figure 8 it can be observed that some of the independent variables of interest also had high correlation with each other. An extreme case of correlated variables produces multicollinearity - a condition in which there is redundancy among the predictor variables (Perfect multicollinearity occurs when one predictor variable can be expressed as a linear combination of others.) The problem of multicollinearity is clear and should be addressed when present - variables should be removed until the multicollinearity is gone.

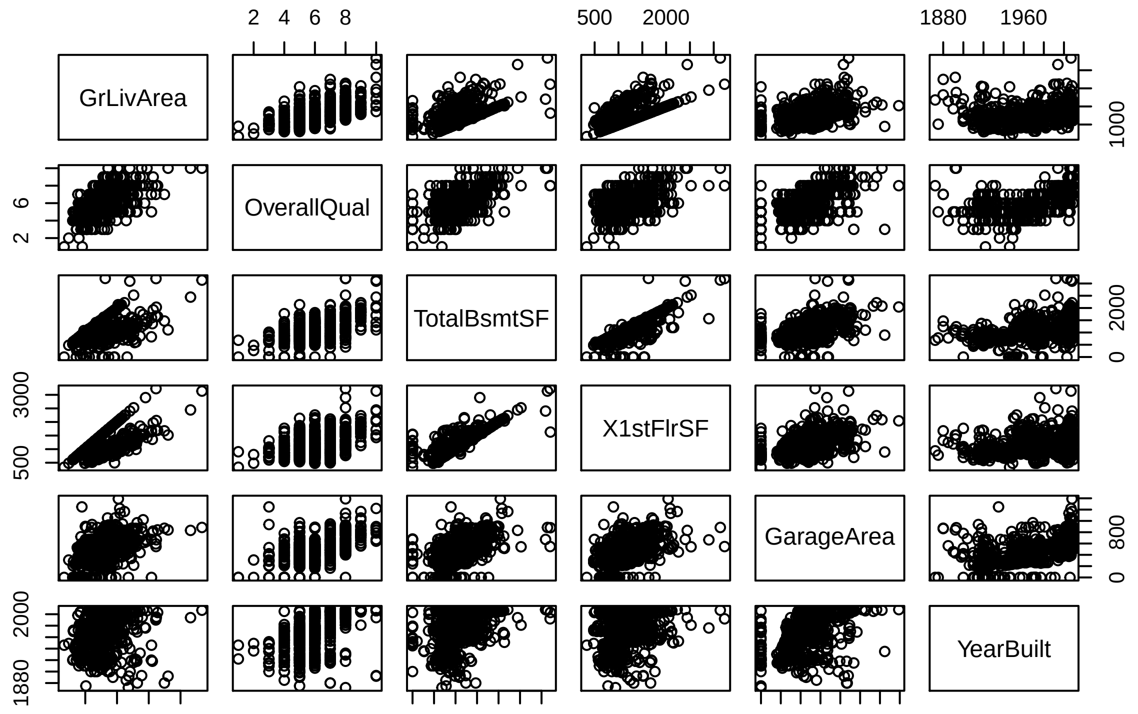

Figure 8: Scatterplot Matrix

4.6 Summary

Based on the above data exploration, it made sense to use a reduced list of variables going forward in the creation of the models. The following sets of data were used in building the models:

```
numberCols
```

```
##  [1] "MSSubClass"    "LotFrontage"   "LotArea"       "OverallQual"
##  [5] "OverallCond"   "YearBuilt"     "YearRemodAdd"  "MasVnrArea"
##  [9] "BsmtFinSF1"    "BsmtFinSF2"    "BsmtUnfSF"     "TotalBsmtSF"
## [13] "X1stFlrSF"     "X2ndFlrSF"     "LowQualFinSF"  "GrLivArea"
## [17] "BsmtFullBath"  "BsmtHalfBath"  "FullBath"      "HalfBath"
## [21] "BedroomAbvGr"  "KitchenAbvGr"  "TotRmsAbvGrd"  "Fireplaces"
## [25] "GarageCars"    "GarageArea"    "WoodDeckSF"    "OpenPorchSF"
## [29] "EnclosedPorch" "X3SsnPorch"    "ScreenPorch"   "PoolArea"
## [33] "MiscVal"       "MoSold"        "YrSold"        "SalePrice"
```

```
NamesInterest1
```

```
##  [1] "OverallQual"  "YearBuilt"    "YearRemodAdd" "MasVnrArea"   "BsmtFinSF1"
##  [6] "TotalBsmtSF"  "X1stFlrSF"    "GrLivArea"    "FullBath"     "TotRmsAbvGrd"
## [11] "Fireplaces"   "GarageCars"   "GarageArea"
```

```
charCols
```

```
## [1] "jGarageType"   "jKitchenQual"  "jExterQual"    "jMSZoning"
## [5] "jNeighborhood"
```

Along with the prediction variable itself i.e. SalePrice.

5 Analysis and Model Creation

5.1 Average Sale Price

Motivation

The analysis started with the simplest model where the predicted Sale Price was the average of all observed Sale Prices in the training set. This produced a maximum RMSE that would be used as an initial benchmark.

Model

$$Y = \mu + \varepsilon$$

where ε are the independent errors sampled from the same distribution.

Prediction

$$\hat{y} = \hat{\mu}$$

RMSE Results

Model	Log RMSE
Average Sale Price	0.4092573

5.2 Linear Model

Motivation

After the cleaning and inspection of the training data, the creation of the linear model represented the building of the first proper model. Since the desired outcome was a continuous numeric variable, a linear model was chosen and not a GLM.

Model

The results of the model can be seen in Figure 9.

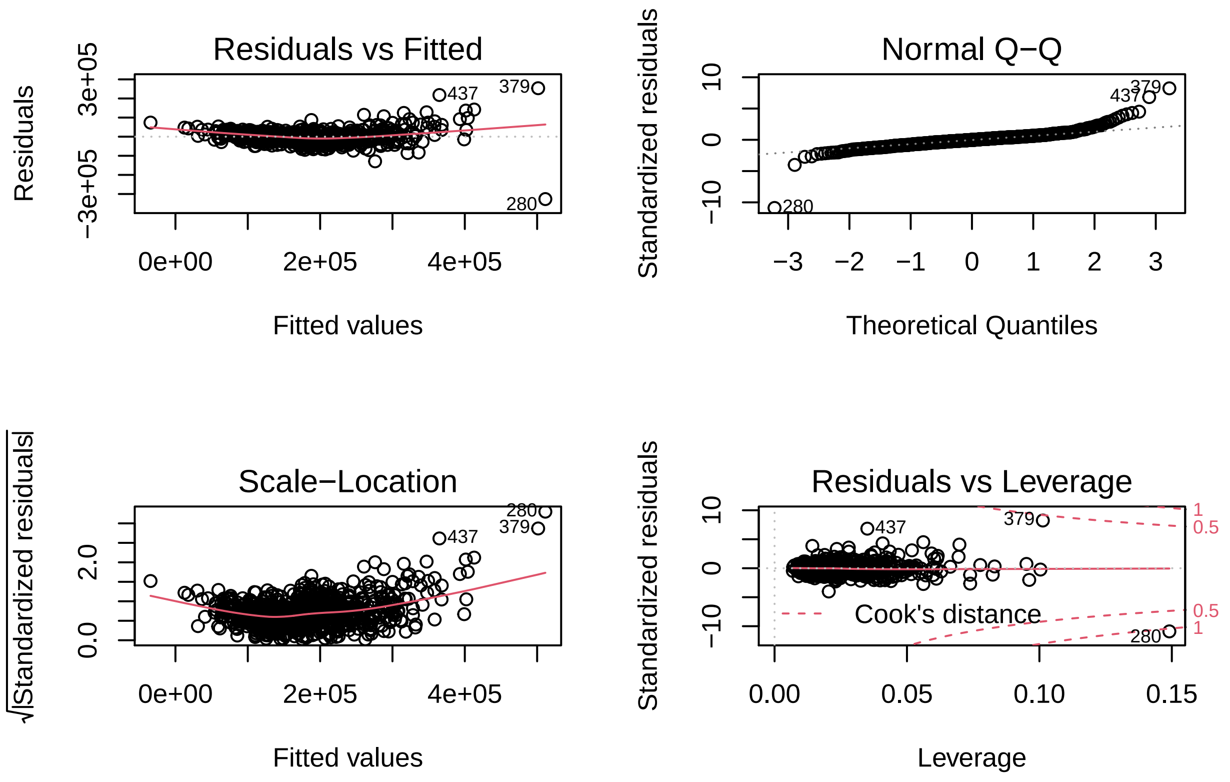

Figure 9: Linear Model

The Adjusted R-squared value was relatively good (0.8274) and all variables passed the Hypothesis Test.

RMSE Results

Model	Log RMSE
Average Sale Price	0.4092573
Linear Model	0.1771038

5.3 LASSO Regression

Motivation

For the avoidance of multicollinearity, a LASSO regression was implemented (see the appendix for background information on LASSO regression).

Having transformed the variables into the form of a matrix, it was possible to automate the selection of variables by implementing the “lars” method in the Lars package.

Two models were created - the first one used just numeric variables and the second one used the “important” variables as defined in the data exploration section.

5.3.1 Model 1 - Numeric Variables

Model

The results of the first model can be seen in Figure 10.

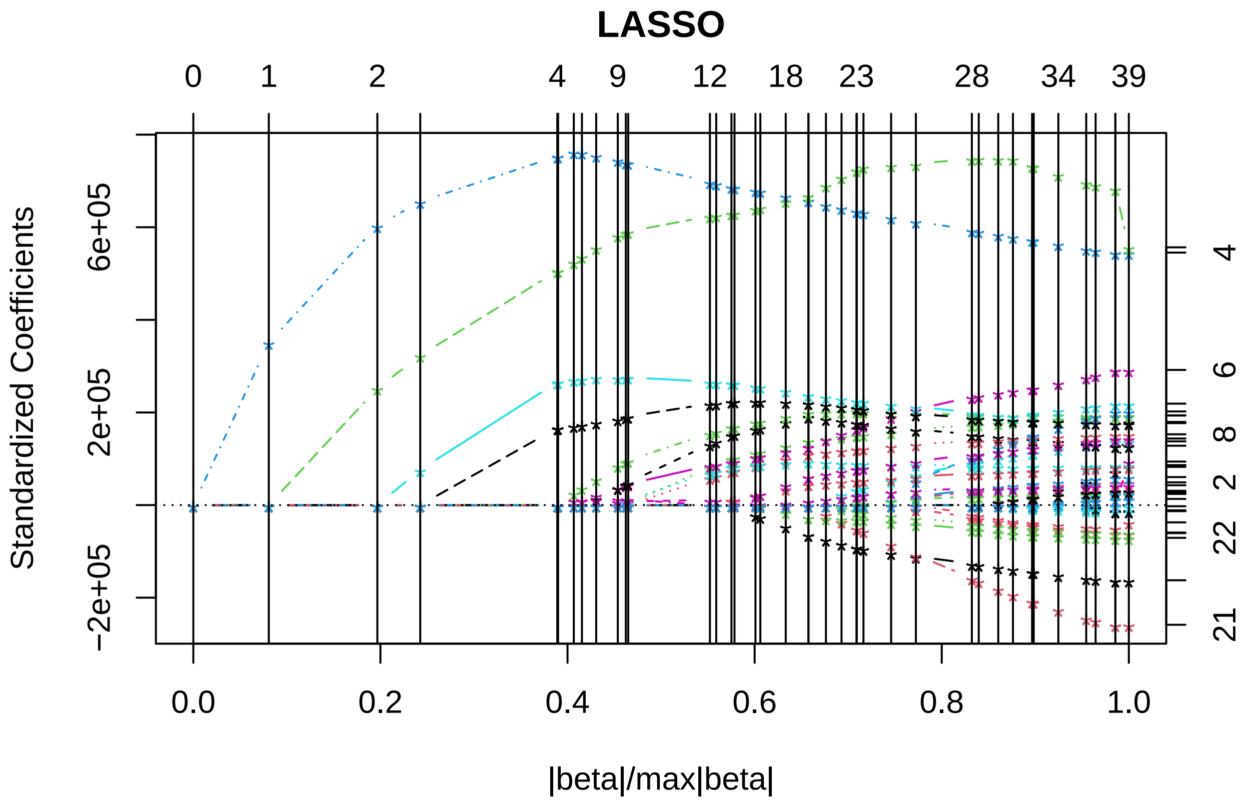

Figure 10: Lasso Regression - Numeric Variables

The plot is not very easy to interpret as the quantity of variables was so large. Despite that, R was still able to be used to find the model with the least multicollinearity. The selection procedure is based on the value of Mallow's Cp (named after a variant of Akaike's Information Criteria, AIC, developed by Colin Mallows), an important indicator of multicollinearity. The prediction was done through the script-chosen "best step" and RMSE was used to assess the model.

5.3.2 Model 2 - Important Variables

Model

The results of the second model can be seen in Figure 11.

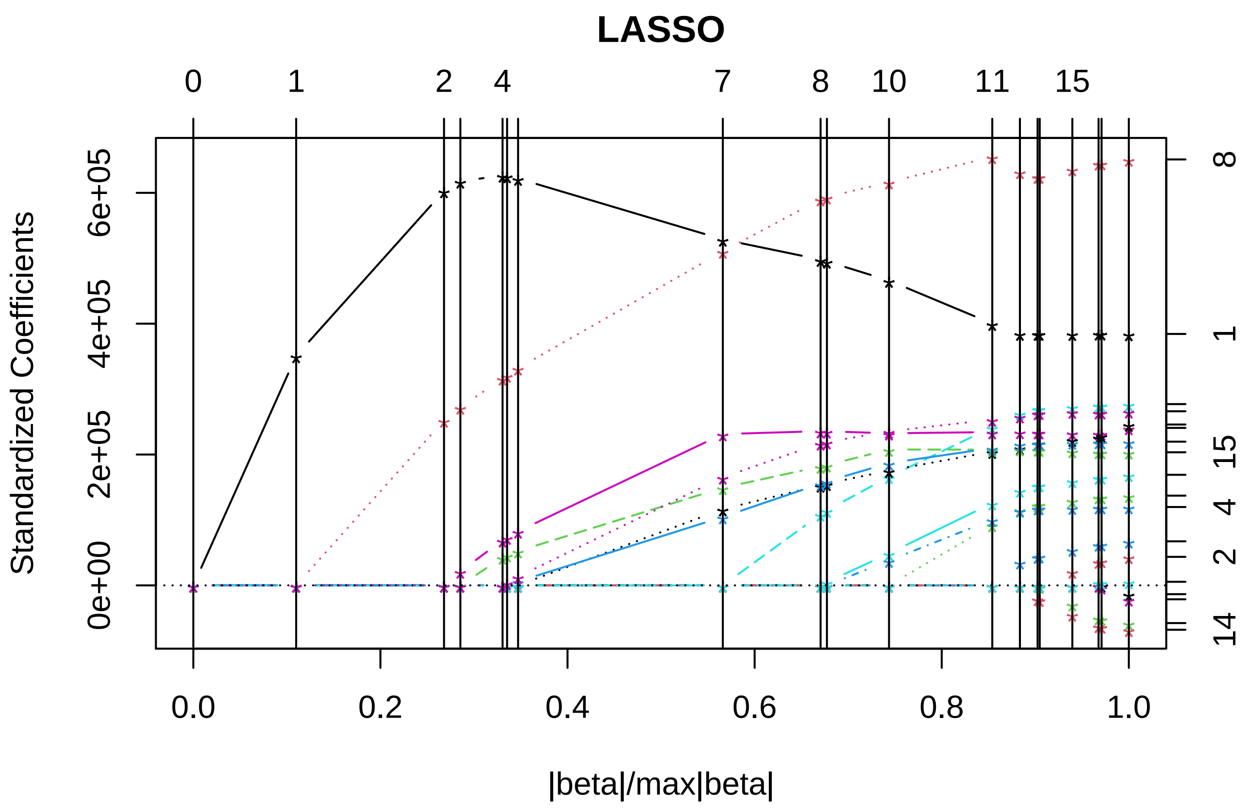

Figure 11: Lasso Regression - Important Variables

As with the first LASSO model, the plot is not very interpretable. However, again using the script to tune the model and find the "best step", the resulting RMSE showed an improvement over the linear model.

RMSE Results

Model	Log RMSE
Average Sale Price	0.4092573
Linear Model	0.1771038
Lasso Model - Numeric	0.2019934
Lasso Model - Important	0.1748782

5.4 Random Forest

Motivation

The next model chosen to train the data was a Random Forest (see the appendix for background information on Random Forests).

All the variables were used as inputs to the model, since the random forest algorithm does its own feature selection.

Model

Figure 12 shows how the model stabilised as the number of trees increased and from about 80 trees there wasn't a significant change in the observed error.

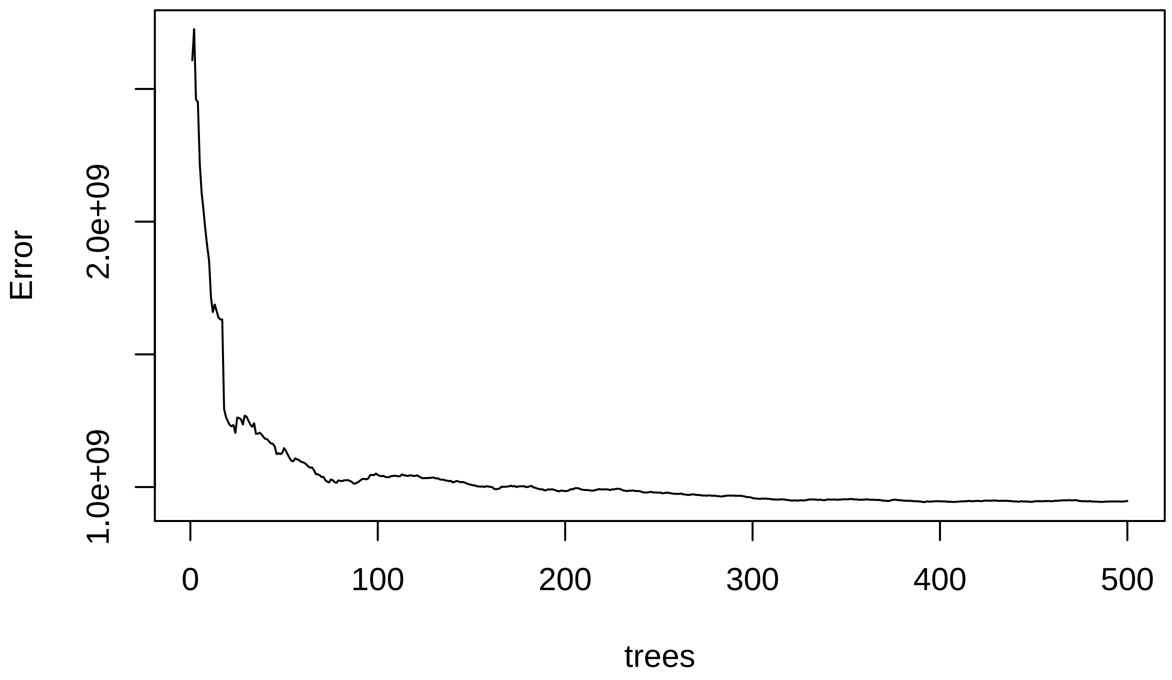

Figure 12: Random Forest model

RMSE Results

Model	Log RMSE
Average Sale Price	0.4092573
Linear Model	0.1771038
Lasso Model - Numeric	0.2019934
Lasso Model - Important	0.1748782
Random Forest	0.1355792

5.5 XGBoost (eXtreme Gradient Boosting)

Motivation

The final model chosen for this project was the XGBoost. It is the most widely used public domain software for boosting (see the appendix for background information on the XGBoost algorithm).

Use of this algorithm resulted in the largest improvement in the accuracy of the model.

RMSE Results

Model	Log RMSE
Average Sale Price	0.4092573
Linear Model	0.1771038
Lasso Model - Numeric	0.2019934
Lasso Model - Important	0.1748782
Random Forest	0.1355792
XGBoost	0.0432758

5.6 Final Validation

Motivation

As can be seen in the previous section, the XGBoost method gave the lowest RMSE therefore this model was chosen as the final solution.

The model was then tested with the final validation data set to see if the RMSE was in line with expectations. As expected, there was a small increase in the RMSE but was minimal.

RMSE Results

Model	Log RMSE
Average Sale Price	0.4092573
Linear Model	0.1771038
Lasso Model - Numeric	0.2019934
Lasso Model - Important	0.1748782
Random Forest	0.1355792
XGBoost	0.0432758
Final RMSE	0.0451926

6 Results

Table 6 shows the overall results of all models that were trained during the analysis. The table also includes the RMSE for the final chosen model applied to the final validation data set.

Table 6: Overall Results

Model	**Log RMSE**
Average Sale Price	0.4092573
Linear Model	0.1771038
Lasso Model - Numeric	0.2019934
Lasso Model - Important	0.1748782
Random Forest	0.1355792
XGBoost	0.0432758
Final RMSE	0.0451926

As can be seen the lowest RMSE from the training was **0.0432758**. This was achieved through the **XGBoost** model implementation. This compares with **0.0451926** for the final validation data set. The small increase in value is to be expected since the model was trained and optimised on the training set.

7 Conclusion

Through the gradual increase in complexity of the models, the XGBoost algorithm produced an improvement of around 89% over the original model.

Even the linear model, LASSO model and Random Forest model made improvements but the clear "winner" was XGBoost.

I believe with more work on the choice of different predictors the overall model could be further improved. In addition the XGBoost algorithm is so powerful and complex for a beginner in the field, further work on tuning the hyperparameters would yield more success.

Running the analysis on a relatively old PC also caused a few issues and for the further work (suggested below) it would warrant upgrading to a more powerful machine.

I would like to take the opportunity to thank the staff and other students at HarvardX for producing a very enjoyable, stimulating and often challenging course. In particular I would like to thank Prof. Irizarry for his knowledge and clarity with the online videos and assessment material.

Whilst I was aware of the topics of R, probability and regression from my former studies it has been many years since I have used them and the topics of wrangling and machine learning were completely new to me. They were topics that I thoroughly enjoyed learning albeit at a level that I know is just scratching the surface. I believe my data science journey has just begun…

(Un)fortunately due to my current job situation, I was able to complete the course in a shorter than recommended period. I believe this was ultimately very beneficial to me and allowed me to keep a high level of focus. I'm not sure I would have managed to study so hard and maintain a full time job at the same time had the situation occurred!!

7.1 Limitations

Along with the aforementioned slow PC, a relatively small data set (in terms of rows) might have had an impact on the results. Updating the data set to include sales from 2000 to 2005 and then from 2011 to 2020 would certainly benefit the process.

Aside from the limitations in data, there is also a limitation on the author's knowledge! Further techniques and methods for sure should be explored once the knowledge has improved!

7.2 Future Work

Whilst the above analysis has already produced a lot of insights and a meaningful model, further investigations could be carried out along the following lines:

- Input Data
  - Extend the data to other areas of the country - e.g. different counties, different states etc.
  - Increase period of when sales occurred - e.g. 2000 to 2005 and/or 2011 to 2020
  - Speak with real estate agents to see if there are other predictors that should be incorporated
  - Is there a way to capture the "Realtor" effect i.e. do the realtors have an inside line on what drives the sale price?
  - Extend the data to include commercial property
- Data Manipulation
  - Combine certain predictors
  - Give more/less weight to certain predictors
  - Remove extreme predictor values and sale prices
- The following points relate to the actual analysis that is performed and the creation of the models:
  - Find a better way to pick different variables - Boruta / correlations
  - Employ neural networks
  - Explore other (currently) unknown models/techniques

A Appendix

The following sections provide some background to the analysis that was carried out for this report.

A.1 Boruta Importance Analysis

Earlier in this report, the R package Boruta (Kursa and Rudnicki, 2020) was used to select all relevant variables from the House Price data set. This section gives a little background to the algorithm.

The referenced article describes the R package Boruta, implementing a novel feature selection algorithm for finding all relevant variables. The algorithm is designed as a wrapper around a Random Forest classification algorithm. It iteratively removes the features which are proved by a statistical test to be less relevant than random probes. The Boruta package provides a convenient interface to the algorithm.

Feature selection is often an important step in applications of machine learning methods and there are good reasons for this. Modern data sets are often described with far too many variables for practical model building. Usually most of these variables are irrelevant to the classification, and obviously their relevance is not known in advance. There are several disadvantages of dealing with overlarge feature sets. One is purely technical - dealing with large feature sets slows down algorithms, takes too many resources and is simply inconvenient. Another is even more important - many machine learning algorithms exhibit a decrease in accuracy when the number of variables is significantly higher than optimal. Therefore selection of the smallest feature set giving the best possible classification results is desirable for practical reasons.

Nevertheless, this very practical goal shadows another very interesting problem - the identification of all attributes which are in some circumstances relevant for classification, the so-called all-relevant problem. Finding all relevant attributes, instead of only the non-redundant ones, may be very useful in itself. In particular, this is necessary when one is interested in understanding the mechanisms related to the subject of interest, instead of merely building a black box predictive model.

The all-relevant problem of feature selection is more difficult than the usual minimal-optimal one. One reason is that we cannot rely on the classification accuracy as the criterion for selecting the feature as important (or rejecting it as unimportant). The degradation of the classification accuracy, upon removal of the feature from the feature set, is sufficient to declare the feature important, but lack of this effect is not sufficient to declare it unimportant. One therefore needs another criterion for declaring variables important or unimportant. Moreover, one cannot use filtering methods, because the lack of direct correlation between a given feature and the decision is not a proof that this feature is not important in conjunction with the other features. One is therefore restricted to wrapper algorithms, which are computationally more demanding than filters.

In a wrapper method the classifier is used as a black box returning a feature ranking, therefore one can use any classifier which can provide the ranking of features. For practical reasons, a classifier used in this problem should be both computationally efficient and simple, possibly without user defined parameters. The algorithm uses a wrapper approach built around a random forest (Breiman 2001) classifier.

The following is a high-level outline of the algorithm:

1. A copy is made of each explanatory variable. The values in these copies are permuted to remove any correlation with the target variable. The copies are called Shadow variables.
2. A random forest model is fitted on this expanded data set.
3. For each variable (the original and Shadow) the Z-score of the loss in accuracy is calculated. The Z-score is the average loss divided by the standard deviation.
4. Keep track of the original attributes z-score that score higher than the maximum of z-score of the shadow attributes.
5. Repeat the above steps numerous times. Original attributes that are significantly–statistically–higher then the maximum z-score of shadow attributes are deemed relevant to the prediction. Attributes that are significantly below the maximum z-score of shadow attributes are deemed not relevant.

Boruta is a god of the forest in Slavic mythology!!

A.2 LASSO Regression

The acronym “LASSO” stands for Least Absolute Shrinkage and Selection Operator.

In statistics and machine learning, LASSO is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the resulting statistical model. It was originally introduced in geophysics and later by Robert Tibshirani who coined the term (Wikipedia).

LASSO regression is a type of linear regression that uses shrinkage (From Statistics How To). Shrinkage is where data values are shrunk towards a central point, like the mean. The LASSO procedure encourages simple, sparse models (i.e. models with fewer parameters). This particular type of regression is well-suited for models showing high levels of muticollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination.

LASSO regression performs L1 regularization, which adds a penalty equal to the absolute value of the magnitude of coefficients. This type of regularization can result in sparse models with few coefficients; some coefficients can become zero and eliminated from the model. Larger penalties result in coefficient values closer to zero, which is the ideal for producing simpler models. On the other hand, L2 regularization (e.g. Ridge regression) doesn’t result in elimination of coefficients or sparse models. This makes LASSO far easier to interpret than Ridge.

LASSO solutions are quadratic programming problems. The goal of the algorithm is to minimize:

$$\sum_{i=i}^{N}(y_i - \sum_{j} x_{i,j}\beta_j)^2 + \lambda \sum_{j=1}^{p} \mid \beta_j \mid$$

Which is the same as minimizing the sum of squares with constraint $\sum \mid \beta_j \mid \leq s$, where s is a pre-specified parameter that determines the degree of regularization. Some of the βs are shrunk to exactly zero, resulting in a regression model that is easier to interpret.

A tuning parameter, λ, controls the strength of the L1 penalty. λ is basically the amount of shrinkage:

- When $\lambda = 0$, no parameters are eliminated. The estimate is equal to the one found with linear regression.
- As λ increases, more and more coefficients are set to zero and eliminated (theoretically, when $\lambda = \infty$, all coefficients are eliminated).
- As λ increases, bias increases.
- As λ decreases, variance increases.
- If an intercept is included in the model, it is usually left unchanged.

A.3 Random Forest

Random forests are a very popular machine learning approach that addresses the shortcomings of decision trees using a clever idea. Random forests are frequently used as “blackbox” models in businesses, as they generate reasonable predictions across a wide range of data while requiring little configuration.

Random decision forests correct for decision trees’ habit of overfitting their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees (see next section). However, data characteristics can affect their performance (Wikipedia).

The goal is to improve prediction performance and reduce instability by averaging multiple decision trees (a forest of trees constructed with randomness). It has two features that help accomplish this.

The first step is bootstrap aggregation or bagging. The general idea is to generate many predictors, each using regression or classification trees, and then secondly, forming a final prediction based on the average prediction of all these trees. To assure that the individual trees are not the same, bootstrapping is used to induce randomness. These two features combined explain the name: the bootstrap makes the individual trees **randomly** different, and the combination of trees is the **forest**. The specific steps are as follows:

1. Build B decision trees using the training set. The fitted models are referred to as $T_1, T_2, \ldots, T_B$.
2. For every observation in the test set, form a prediction $\hat{y}_j$ using tree T_j.
3. For continuous outcomes, form a final prediction with the average $\hat{y} = \frac{1}{B}\sum_{j=1}^{B}\hat{y}_j$. For categorical data classification, predict $\hat{y}$ with a majority vote (most frequent class among $\hat{y}_1, \ldots, \hat{y}_T$).

Getting different decision trees from a single training is achieved through the following steps and using randomness in two ways. Let N be the number of observations in the training set. To create T_j, $j = 1, \ldots, B$ from the training set the following is performed:

1. Create a bootstrap training set by sampling N observations from the training set with replacement. This is the first way to induce randomness.
2. A large number of features is typical in machine learning challenges. Often, many features can be informative but including them all in the model may result in overfitting. The second way random forests induce randomness is by randomly selecting features to be included in the building of each tree. A different random subset is selected for each tree. This reduces correlation between trees in the forest, thereby improving prediction accuracy.

As was observed in Figure 12 as more trees were added to the random forest model for house prices, the error rate changed, the accuracy improved and with about 80 trees the accuracy stabilized.

Random forests often perform well in most situations. However, a disadvantage of random forests is the loss of interpretability. An approach that helps with interpretability is to examine variable importance. To define variable importance a count is kept of how often a predictor is used in the individual trees. The caret package in R includes the function varImp that extracts variable importance from any model in which the calculation is implemented.

A.4 XGBoost (e*X*treme *G*radient *B*oosting)

A.4.1 Background

Ensemble models have become a standard tool for predictive modeling (Bruce, Bruce and Gedeck, 2020). Boosting is a general technique to create an ensemble of models. It was developed around the same time as bagging. Like bagging, boosting is most commonly used with decision trees. Despite their similarities, boosting takes a very different approach - one that comes with more bells and whistles. As a result, while bagging can be done with relatively little tuning, boosting requires much greater care in its application. If these two methods were cars, bagging could be considered a Honda Accord (reliable and steady), whereas boosting could be considered a Porsche (powerful but requires more care).

In linear regression models, the residuals are often examined to see if the fit can be improved. Boosting takes this concept much further and fits a series of models, in which each successive model seeks to minimize the error of the previous model. Several variants of the algorithm are commonly used: Adaboost, gradient boosting and stochastic gradient boosting.

The most widely used public domain software for boosting is XGBoost (and the one used in this report). It is an implementation of stochastic gradient boosting that is both computationally efficient and made up of many options/paramters. Two very important parameters are subsample, which controls the fraction of observations that should be sampled at each iteration, and eta, a shrinkage factor applied in the boosting algorithm.

Using subsample makes boosting act like the random forest except that the sampling is done without replacement. The shrinkage parameter eta is helpful to prevent overfitting by reducing the change in the weights (a smaller change in the weights means the algorithm is less likely to overfit to the training set).

A.4.2 Summary

- Boosting is a class of ensemble models based on fitting a sequence of models, with more weight given to records with large errors in successive rounds.
- Stochastic gradient boosting is the most general type of boosting and offers the best performance. The most common form of stochastic gradient boosting uses tree models.
- XGBoost is a popular and computationally efficient software package for stochastic gradient boosting; it is available in all common languages used in data science.
- Boosting is prone to overfitting the data, and the hyperparameters need to be tuned to avoid this.
- Regularization is one way to avoid overfitting by including a penalty term on the number of parameters (e.g. tree size) in a model.
- Cross-validation (see next section) is especially important for boosting due to the large number of hyperparameters that need to be set.

A.4.3 Tutorial

The XGBoost documentation site (XGBoost Documentation) provides further insights and tutorials on this very powerful tool.

A.5 Cross Validation

A.5.1 Background

A common goal of machine learning is to find an algorithm that produces predictors $\hat{Y}$ of an outcome Y that minimizes the MSE:

$$MSE = E\left\{\frac{1}{N}\sum_{i=i}^{N}(\hat{Y}_i - Y_i)^2\right\}$$

When there is only one dataset, it is possible to estimate the MSE with the observed MSE like this:

$$M\hat{S}E = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

These two are referred to as the true error and apparent error, respectively.

There are two important characteristics of the apparent error that should always be kept in mind:

- Because the data is random, the apparent error is a random variable. For example, the given dataset may be a random sample from a larger population. An algorithm may have a lower apparent error than another algorithm due to luck.

- If an algorithm is trained on the same dataset that is used to compute the apparent error, it might result in overtraining. In general, when this action is performed, the apparent error will be an underestimate of the true error.

Cross-validation is a technique that permits the alleviation of both these problems. To understand cross-validation, it helps to think of the true error, a theoretical quantity, as the average of many apparent errors obtained by applying the algorithm to B new random samples of the data, none of them used to train the algorithm. The true error can be thought of as:

$$\frac{1}{B}\sum_{b=1}^{B}\frac{1}{N}\sum_{i=i}^{N}(\hat{y}_i^b - y_i^b)^2$$

with B a large number that can be thought of as practically infinite. As already mentioned, this is a theoretical quantity because there is only one available set of outcomes: $y_1, \ldots, y_n$. Cross validation is based on the idea of imitating the theoretical setup above as best as possible with the available data. To do this, it is necessary to generate a series of different random samples. There are several approaches that can be employed, but the general idea for all of them is to randomly generate smaller datasets that are not used for training, and instead used to estimate the true error.

For most machine learning algorithms it is necessary to select parameters, let's say λ, that minimises the value of MSE. It is known that if optimsation and evaluation take place on the training set, overtraining will occur. It is also the golden rule of machine learning to not use the original validation data set and this is where cross-validation is most useful.

A.5.2 K-fold cross-validation

One of the possibilities is K-fold cross-validation. This technique divides the data into K subsets (folds) of almost equal size. Out of these K folds, one subset is used as a validation set and the others are involved in training the model.

The following describes the complete working procedure of this method:

1. Split the training dataset randomly into K subsets
2. Use K-1 subsets for the training of the model
3. Test the model against that one subset which was left out in the previous step:

$$M\hat{S}E_b(\lambda) = \frac{1}{M}\sum_{i=1}^{M}(\hat{y}_i^b(\lambda) - y_i^b)^2$$

4. Repeat the above steps K times i.e. until the model is trained and tested on all subsets. This generates:

$$M\hat{S}E_1(\lambda), M\hat{S}E_2(\lambda), \ldots, M\hat{S}E_K(\lambda)$$

5. Generate an overall prediction error by taking the average of prediction errors in every case:

$$M\hat{S}E(\lambda) = \frac{1}{B}\sum_{b=1}^{K} M\hat{S}E_b(\lambda)$$

6. The above method describes one iteration of the process with one value of the tuning parameter λ being fixed. In general, the above is repeated with several different values of the tuning parameter to find the optimal value of λ that minimises the MSE.
7. Fit the final model with the obtained optimal tuning parameter(s).
8. Finally test the results from the previous step on the original independent validation dataset using the final model.

Large values of K are preferable because the training data better imitates the original dataset. However, larger values of K will have much slower computation time: for example, 100-fold cross validation will be 10 times slower than 10-fold cross validation. For this reason, the choices of $K = 5$ and $K = 10$ are popular.

A.5.3 Other techniques and implementations

One way to improve the variance of the final estimate is to take more samples. To do this, it would no longer require the training set to be partitioned into non-overlapping sets. Instead, just pick K sets of some size at random.

One version of this technique, at each fold, picks observations at random with replacement (which means the same observation can appear twice). This approach has some advantages and is generally referred to as the bootstrap.

XGBoost, used in this report, uses cross validation as a method in its implementation.

A.6 Data Source

The Ames Housing dataset, used in this report, was compiled by Dean De Cock for use in data science education. It's an alternative for data scientists looking for a modernized and expanded version of the often cited Boston Housing dataset.

The original dataset can be found under : https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data.

For the purposes of this report, a copy has been stored under the "data" directory of the project's github repository : https://github.com/ostlerjo/CapstoneHousePrices.git

To be precise, located here : https://github.com/ostlerjo/CapstoneHousePrices/tree/master/data

A.7 References

Galwey, N. (2014). Introduction to mixed modelling: Beyond regression and analysis of variance.

Irizarry, R. A. (2020). Introduction to data science: Data analysis and prediction algorithms with R : https://doi.org/10.1201/9780429341830.

Kaggle: Your home for data science : https://www.kaggle.com

Viswanathan, V. (2015). R data analysis cookbook: Over 80 recipes to help you breeze through your data analysis projects using R.

Kursa, M. and Rudnicki, W. (2020). Journal of Statistical Software - Feature Selection with the Boruta Package : https://www.jstatsoft.org/index.php/jss/article/view/v036i11/v36i11.pdf

LASSO statistics from Wikipedia : https://en.wikipedia.org/wiki/Lasso

Statistics How To : https://www.statisticshowto.com/lasso-regression/

Random Forest from Wikipedia : https://en.wikipedia.org/wiki/Random-forest

Peter Bruce, Andrew Bruce and Peter Gedeck (2020). Practical Statistics for Data Scientists.

XGBoost Documentation : https://xgboost.readthedocs.io/en/latest/index.html

XGBoost R Tutorial : https://xgboost.readthedocs.io/en/latest/R-package/xgboostPresentation.html

A.8 Course Material

Links to edX and HarvardX :
https://www.edx.org
https://www.edx.org/school/harvardx

Link to this course (including some of the theory quoted above):
https://www.edx.org/professional-certificate/harvardx-data-science