# 230 Project Report

Zack Brodtman

## 1. Introduction

The sale price of a residential home is influenced by a wide range of factors, including the physical characteristics of the home, the location of the home, and various economic and demographic factors. Understanding the factors that influence house prices is essential for homeowners, real estate agents, and policymakers who want to make informed decisions about buying, selling, and investing in residential real estate.

**What are the key factors that influence the sale price of residential homes in Ames, Iowa, and how accurately can we predict the sale price of homes using regression models?**

Developing an accurate and robust regression model for predicting house prices in Ames, Iowa, can have important practical implications. In this project, we will focus on predicting house prices in Ames, Iowa, using multiple linear regression analysis. Ames is a small city located in the heart of Iowa, with a population of approximately 66,000 people. Despite its small size, Ames is home to Iowa State University, which has a large student population and a significant impact on the local economy. The city has a diverse range of housing options, from historic homes in the downtown Area to modern developments on the outskirts of the city. Understanding the factors that influence house prices in Ames can provide insights into the broader factors that influence housing markets in small and mid-sized cities across the United States. Policymakers can use the results to develop policies and programs that can promote affordable housing and improve access to housing for low- and middle-income families.

## 2. The Dataset

The Ames House Prices dataset (De Cock (2011)) contains a total of 2930 observations, with 2930 residential homes sold between 2006 and 2010 in Ames, Iowa. The dataset includes a total of 80 explanatory variables (features) and a target variable (SalePrice), which represents the sale price of each house. The dataset includes a variety of features related to the physical characteristics of the homes, such as the size of the lot, the number of bedrooms and bathrooms, the Overall Quality and Condition of the home, and the year the home was built or remodeled. The dataset also includes information on the location of the home, such as the neighborhood and proximity to various amenities. The dataset contains several categorical variables, such as the type of dwelling, the style of the home, and the type of foundation, which will need to be converted into numerical variables with dummy encoding before they can be used in a regression analysis. Below is a small sample of the data.

```
##         PID Lot.Area Overall.Qual Full.Bath Year.Built Heating SalePrice
## 1 526301100    31770            6         1       1960    GasA    215000
## 2 526350040    11622            5         1       1961    GasA    105000
## 3 526351010    14267            6         1       1958    GasA    172000
## 4 526353030    11160            7         2       1968    GasA    244000
## 5 527105010    13830            5         2       1997    GasA    189900
## 6 527105030     9978            6         2       1998    GasA    195500
```

## 3. Pre-Processing and Feature Engineering

In the pre-processing stage of our analysis, I encountered several missing values in both numerical and categorical features. To handle these missing values, I chose to replace numerical missing values with 0. One option I considered was replacing with the mean of that feature, though in this context I decided that this was inappropriate. It is a reasonable assumption that a missing value in a numerical feature could indicate that the value is actually zero. For example, if a house does not have a garage, then the value for the Garage Area feature would be missing and replacing it with 0 would make sense. On the other hand, replacing it with the mean value had a high chance of introducing information into our dataset which was not likely to be true.

For categorical features, I replaced missing values with "missing". Replacing categorical missing values with the string "missing" allows us to treat these values as a separate category. This is important because the 'missingness' itself could be informative and we do not want to lose this information. By creating a separate category for missing values, we also allow our model to be usable with incomplete new data. This is important because it allows agents to use the model without having to ensure they have recorded every piece of data about the property. Overall, these pre-processing steps allowed us to handle missing values in a way that preserves the integrity of the data and ensures that we are not making assumptions that could impact the performance of our models.

I then one-hot encoded all the categorical variables, giving 328 total variables, and split the data into a train and test set by year. This was appropriate because the distribution is constant across all years, as shown by the box plots below. The training set was made up of the 2006-2008 sales and consisted of 1,941 rows. The test set had 2009 and 2010 sales, consisting of 989 rows. This represented a 66%:34% split, which gives us enough data to capture the variance of sale prices, while offering a large enough test set that we can be confident in our final model's predictive power.

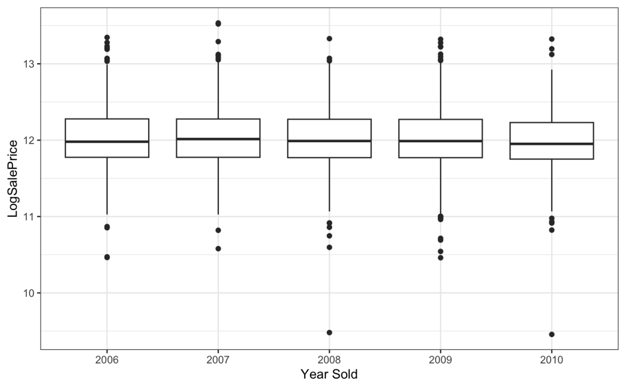


Figure 1: Box plots of log(Sale Price) by year

The histograms below show that the distribution of Sale Price is slightly skewed, with a long tail towards higher values, while the histogram of log(Sale Price) has a shape much more similar to that of a normal distribution. Skew of the target variable can cause issues with model performance, as linear models assume normality and homoscedasticity of the residuals. Therefore, in order to improve the model performance and account for the skewed distribution, the decision was made to model the logarithm of the sale price instead. This allowed for a more reliable and accurate modeling of the target variable.

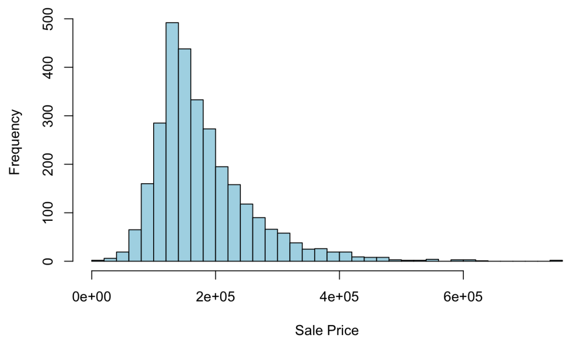


Figure 2: Sale Price histogram

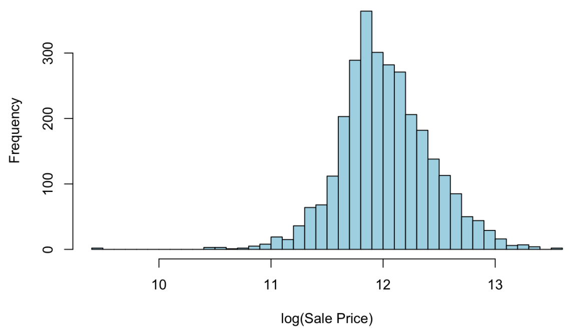


Figure 3: log(Sale Price) histogram

## 4. Exploratory Data Analysis

In order to better understand the relationships between the features and the target variable, I conducted a graphical exploration of the data. Specifically, I created plots of each feature against the log sale price, both for numerical and discrete features. Through this process, I was able to identify certain features (Overall Quality & Condition, Second Floor Qquare Feet, Fireplaces, Garage Area) that had clear linear relationships with log(Sale Price), while others had relationships which appeared to be less linear. By visually inspecting these relationships, I gained insight into which features might be the most important predictors of the target variable, and how their relationships might be modeled in the subsequent analysis, such as which features to make polynomial. This analysis also helped identify Overall Quality levels with distinct price ranges, highlighting its strong relationship with sale price. This exploration also helped me identify potential outliers and anomalies in the data that needed to be monitored when looking at the diagnostic plots to see if removing them would improve the model.

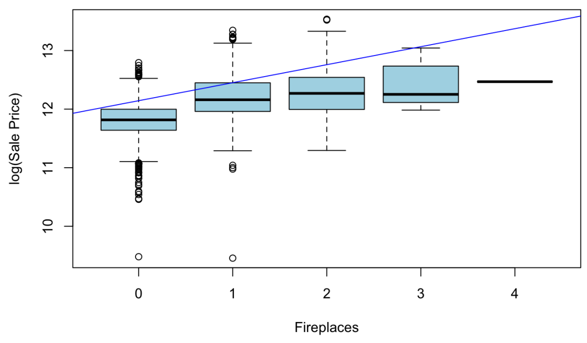


Figure 4: Relationship between Fireplaces and log(Sale Price)

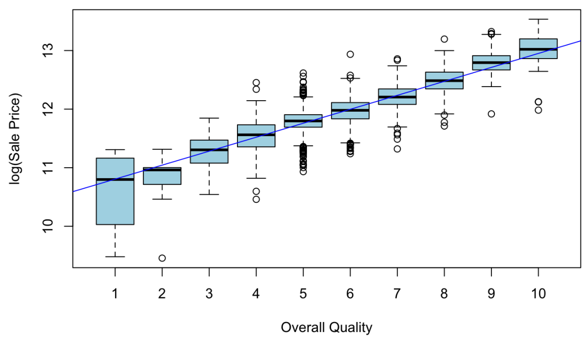


Figure 5: Relationship between Overall Quality and log(Sale Price)

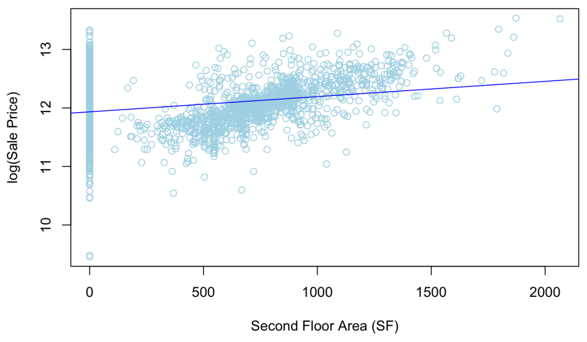


Figure 6: Relationship between Second Floor Area and log(Sale Price)

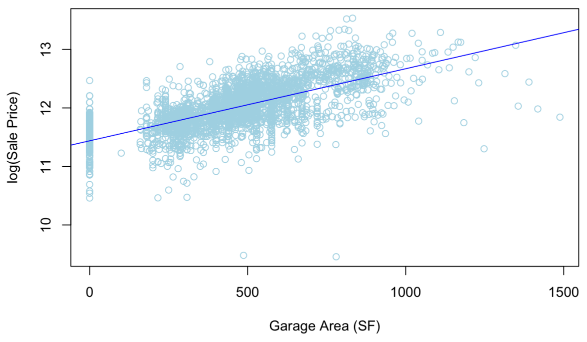


Figure 7: Relationship between Garage Area and log(Sale Price)

The correlation plot below provides a comprehensive visual representation of the relationships between different numerical variables in the dataset. It offers insights into the strength and direction of these relationships, aiding in the identification of potentially significant predictors for the target variable, Log(SalePrice). Upon examining the correlation plot, several noteworthy observations can be made. First, variables such as Overall Quality, Ground Floor Living Area, and Garage Area exhibit strong positive correlations with LogSalePrice. This suggests that as these variables increase, the sale price tends to increase as well. Conversely, variables such as Age at Sale and Years Since Remodeling show negative correlations, indicating that older properties or those that have not been remodeled recently tend to have lower sale prices.

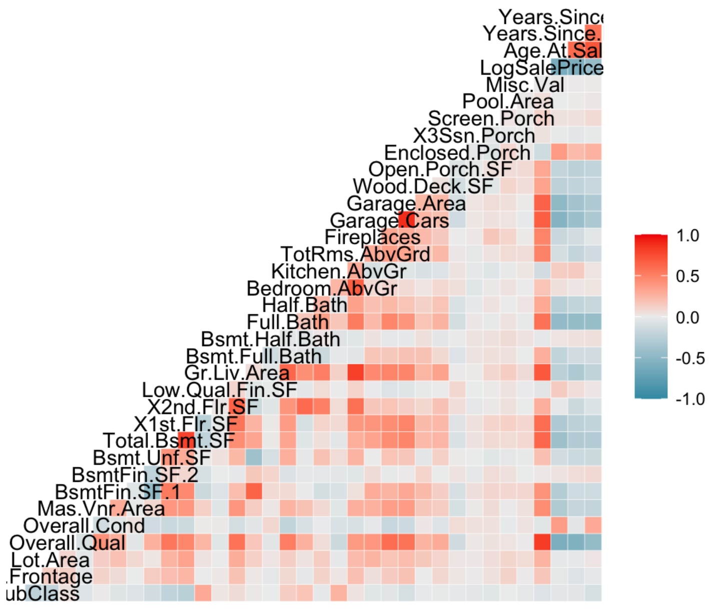


Figure 8: Pairwise correlation

Additionally, some variables display notable correlations with each other. For example, Garage Cars and Garage Area are highly correlated, indicating that the size of the Garage is closely related to the number of cars it can accommodate. Similarly, Total Basement SF and 1st Floor SF exhibit a strong positive correlation, suggesting that properties with larger first-Floor Areas tend to have larger Basement spaces. For this reason, I removed the redundant features Garage Cars and Total Basement SF.

## 5. Final Model

$$\log(\textbf{Sale Price}) = 0.4 \cdot \textbf{Lasso predictions} + 0.6 \cdot \textbf{XGBoost predictions} \tag{1}$$

My final model is a weighted average of predictions from Lasso regression and a gradient boosting model. It used just 181 features out of the 328 in the original dataset, chosen by the Lasso regression. I then performed a grid search to find the weights that minimized the mean absolute error for the combination of the models. Some of the features kept by the Lasso model were:

- Lot Area
- Overall Quality and Condition
- Number of Bathrooms
- Bedrooms Above Grade
- Number of Fireplaces
- Land Contour
- Pool Quality
- Neighborhood

My most successful regression model was found using Lasso regression. Lasso seeks to solve the optimization problem:

$$\hat{\beta}(\lambda) = \underset{b_0, b_1, \ldots, b_p}{\arg \in} \left\{ RSS(b_0, b_1, ..., b_p) + \lambda \sum_{j=1}^{p} |b_j| \right\} \tag{2}$$

It imposes a penalty on the sum of the absolute values of the coefficients in the regression model, which causes some of the coefficients to be shrunk to zero. In other words, Lasso minimizes the residual sum of squares subject to the sum of the absolute values of the regression coefficients being less than a constant, known as the tuning parameter, $\lambda$. This is different from Ridge regression, where in general the coefficients will be close to 0 but not identically 0. This property is particularly valuable for the Ames Housing dataset because we have so many features, all of which require cost and effort to collect. It reduced the total number of features by 147 with $\lambda_{min} = 0.0016$. This reduction in the number of features is important as it helps to reduce the risk of overfitting, and can improve the generalizability and interpretability of the model. The out-of-sample R-squared of the Lasso model was 87.0%, indicating that the model explains a large proportion of the variance in the data. Additionally, the mean absolute error of the model was \$14,345, and the root mean squared error was \$22,469. This was significantly better than the full model which had 328 features, an OSR$^2$ of 84.6%, MAE of \$19,062 and RMSE of \$29,655.

This model provided a significant improvement on the full model, but I felt that we could capture more of the variance of the data if more advanced machine learning techniques were used. After testing Random Forests and fairly shallow Neural Networks, I found the best performance came from XGBoost, a gradient boosting model. XGBoost (eXtreme Gradient Boosting, Chen and Guestrin (2016)) is an ensemble learning method that combines multiple weak prediction models to create a stronger one. It is a tree-based algorithm that builds decision trees in a sequential manner, where each subsequent tree is constructed to correct the errors of the previous one. In contrast to traditional decision trees, XGBoost builds these trees in a greedy fashion by iteratively adding new splits that minimize the mean square error. To prevent overfitting, XGBoost employs early stopping and has complexity parameters which prevent each tree from growing too deep. XGBoost has won several awards and it is frequently the winning algorithm in Kaggle competitions for high-dimensional datasets. It is also highly interpretable, making it attractive for this project. I trained an XGBoost model on the features selected by Lasso regression, achieving an OSR$^2$ of 88.5%, MAE of $15,170 and RMSE of $24,885.

When combined with the Lasso regression, the weighted predictions had an OSR$^2$ of 89.4%, MAE of $14,061 and RMSE of $22,763. This was an average improvement of $6892 on the full model, with 147 fewer features.

## 6. Research Findings - Feature Importance

I had two goals when I began this project: accurately predict the sale price of houses in Ames, Iowa, and derive the relative importance of the house features. All of my models, discussed in more detail in the next section, provided fairly good accuracy for predicting Sale Price. However, they provide varying degrees of usefulness in fulfilling my second goal - understand which factors contribute to a higher Sale Price. Both Lasso and XGBoost are highly interpretable models which allow us to answer that question. Before performing the Lasso regression, all the features are standardized. This means we can directly interpret the magnitude of their coefficients as their importance in the model because the coefficient is the amount that log(Sale Price) changes if all other features are kept constant and that feature increases by 1 unit. The graph below details the 25 most important features in the Lasso model. The values are relative to the highest coefficient.

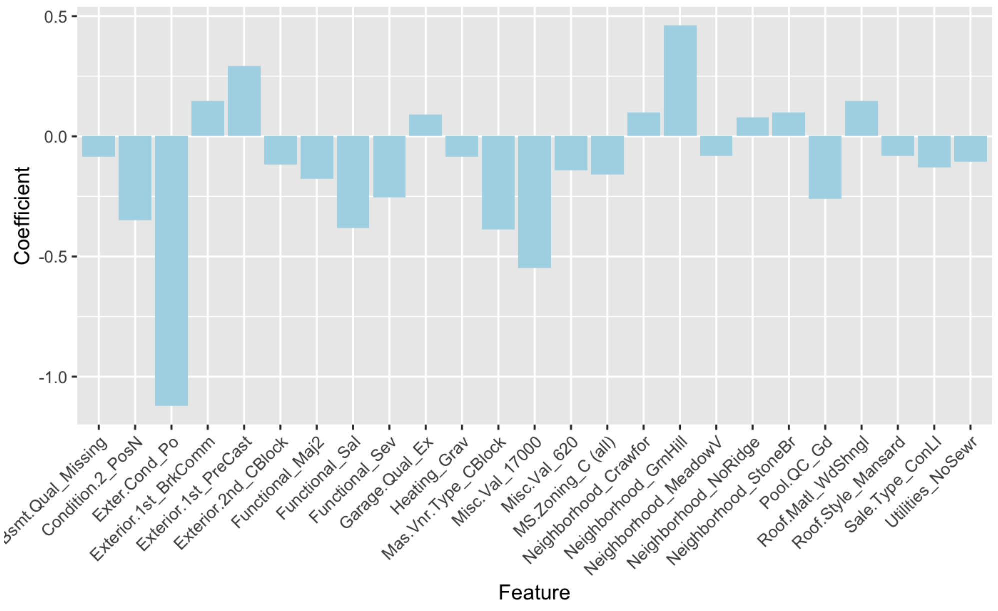


Figure 9: Relative feature importance according to the Lasso model

Analysing the largest coefficients, we see that an increase of 1 unit in Overall Quality translates to an increase of 5.68% in Sale Price, each additional fireplace on the property suggests an increase of 2.53% in Sale Price, while having central air conditioning adds 2.01%. We also see from the graph that neighborhood is among the most important of features: being located in Green Hills, a location known for its pensioner population, increases Sale Price, while being located closer to the Iowa State University campus in Meadow Village reduces Sale Price.

The XGBoost library in R measures feature importance in 3 ways:

- Gain: the contribution of the feature to the model calculated by the total reduction of the loss when it is used in the trees of the model
- Cover: the number of times a feature appears in the trees of the model

- Frequency: the frequency of the feature being used in the trees of the model

Relative importance combines these metrics into 1, and the graph below is useful for visualizing this. Again, Overall Quality and number of fireplaces are important features, though now we also see the importance of age-at-sale, living Area and Lot Area (both measured in square feet). Most of the important features are in line with our intuition, though there are some surprises, such as the presence of a paved drive having the 14th highest relative importance (out of 181 features).

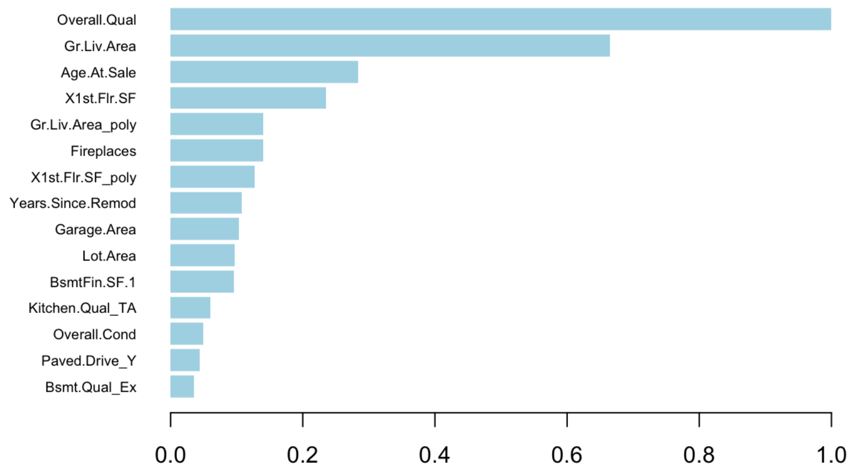


Figure 10: Relative Feature Importance

Understanding the important factors that affect sale price is crucial for making informed decisions when buying or selling a property. The feature importance graphs and information obtained from the models provide valuable insights into which variables have the most impact on the sale price. By interpreting the feature importance information, we can identify which features should be given the most attention and potentially be prioritized when making changes to the property or selecting a property to buy. Furthermore, understanding the importance of different features can also inform our understanding of the housing market, helping us make more informed decisions about where and when to buy or sell properties. Therefore, the interpretations of feature importance from the models are essential for making informed decisions in the real estate market.

## 7. Limitations

There are several limitations associated with this analysis. Firstly, the predictive models developed in this project are based on historical data and assumptions that may not hold in the future. Economic conditions, market trends, and external factors such as policy changes or major events like the COVID-19 pandemic and the increase in remote work can significantly impact the housing market. Therefore, if there are dramatic shifts in the economy or unforeseen events, the models developed in this project may not accurately predict housing prices.

Secondly, the models heavily rely on the features and variables available in the dataset. While efforts were made to include a comprehensive set of predictors, there could still be important factors that were not captured in the data. Variables such as changes in local infrastructure, demographic shifts, or specific market conditions may influence housing prices but were not included in the analysis. Therefore, the models may not fully capture all the relevant factors affecting housing prices in the Ames area.

Another limitation is related to the availability and quality of data. The Ames housing dataset, although extensive, is still a sample of the overall housing market. It is possible that certain segments or types of properties are underrepresented or not well-represented in the dataset. This can lead to biases and limitations in the generalizability of the models to the entire population of properties. Furthermore, the residential market in the United States is so heterogenous that a model developed for one town is likely to be completely unsuitable for almost any other town. This makes generalisability and minimizing features even more important.

## 8. Conclusion

In conclusion, this project aimed to develop predictive models for housing prices in the Ames area and understand the most important features for this task. Through exploratory data analysis, feature engineering, and model training, we have gained valuable insights into the factors influencing housing prices and built several models with varying complexity. The results indicate that the chosen models, including the full linear model, lasso regression, ridge regression, PCR, and XGBoost, demonstrate reasonable predictive performance. The weighted average model, which combined the strengths of Lasso regression and XGBoost, achieved the highest out-of-sample R-squared, lowest MAE, and lowest RMSE, suggesting its effectiveness in predicting housing prices. Additionally, the feature importance analysis provided valuable insights into the significant predictors of housing prices. Features such as Overall Quality, Ground Floor Living Area, Age At Sale, and Lot Area emerged as important factors across multiple models, highlighting their consistent influence on housing prices. Moreover, neighborhood-related variables were found to play a crucial role in determining housing prices, with specific neighborhoods showing a positive or negative impact on property values.

## 9. Model Diagnostics

Performing diagnostics on the full model allows us to check if the assumptions (independence of errors, constant variance, and normality of residuals) are met. We also use it to evaluate our model performance and improve our model. The plots below identified a few outliers in the data, and so these were removed before the data was used in models fit after the full model.

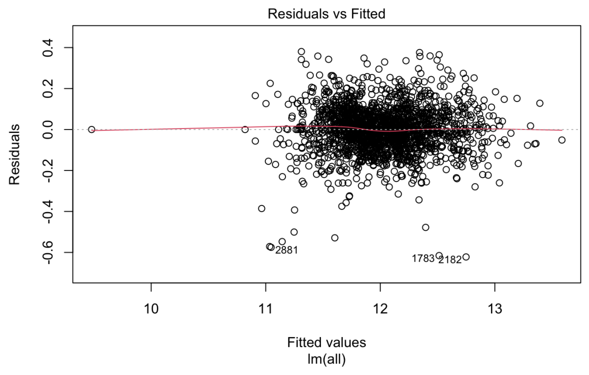


Figure 11: Residuals versus Fitted Values

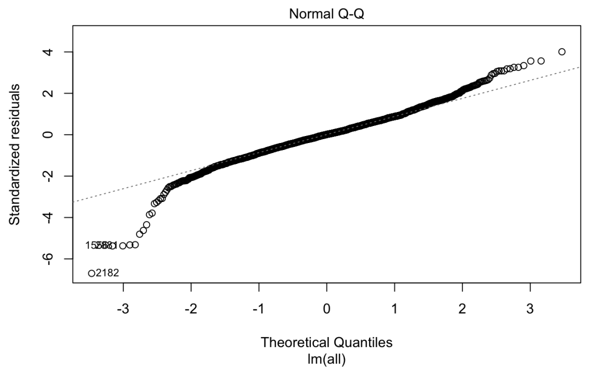


Figure 12: Normal Q-Q plot

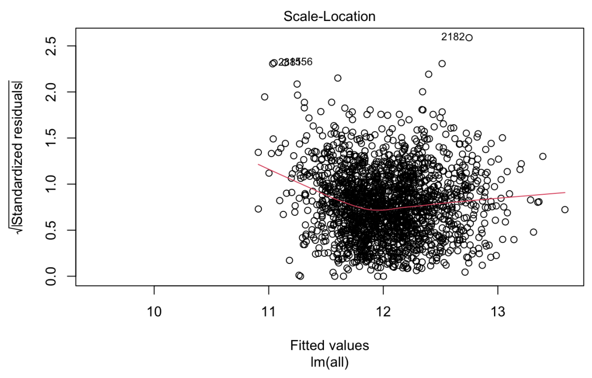


Figure 13: Scale Location

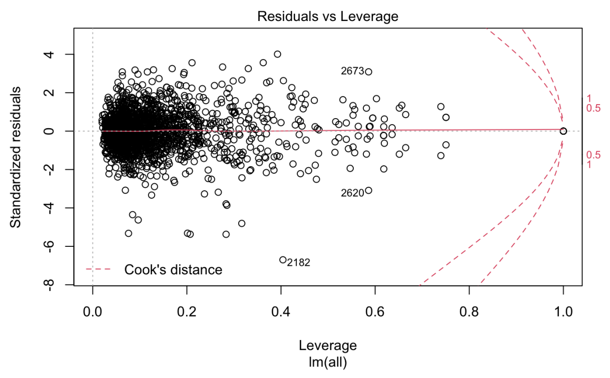
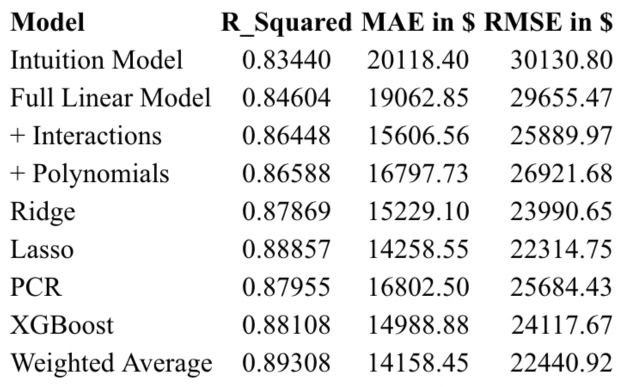


Figure 14: Residuals versus Leverage

- Normal QQ plot: The majority of the points lie close to the diagonal line, indicating that the residuals follow approximately a normal distribution. However, there are some outliers at very negative x-values (less than -2), suggesting deviations from normality in the lower tail of the distribution.
- Residuals vs Fitted Values plot: Most of the residuals are centered around zero, indicating that the model captures the Overall relationship between the predictors and the target variable. However, there are approximately 5 outliers with noticeably negative residuals, suggesting potential issues with those observations that are not well explained by the model.
- Scale-Location plot: The red line, which represents the square root of the standardized residuals, is relatively horizontal on the right-hand side, indicating homoscedasticity (constant variance). However, there is a slight upward tilt on the left side, suggesting a possible violation of the constant variance assumption for the lower range of fitted values.
- Residuals vs Leverage plot: The majority of the residuals are concentrated around the straight red line passing through zero, indicating that the model is not strongly influenced by influential observations. However, there are a few outliers that exhibit higher leverage, indicating that those observations may have a larger impact on the model's fit.

Analysis of the residual plots for the Lasso model gave similar conclusions, so I am confident that my regression does not significantly break the assumptions neccessary for linear modelling.

## 10. Less Successful Models



| Model | R_Squared | MAE in $ | RMSE in $ |
|---|---|---|---|
| Intuition Model | 0.83440 | 20118.40 | 30130.80 |
| Full Linear Model | 0.84604 | 19062.85 | 29655.47 |
| + Interactions | 0.86448 | 15606.56 | 25889.97 |
| + Polynomials | 0.86588 | 16797.73 | 26921.68 |
| Ridge | 0.87869 | 15229.10 | 23990.65 |
| Lasso | 0.88857 | 14258.55 | 22314.75 |
| PCR | 0.87955 | 16802.50 | 25684.43 |
| XGBoost | 0.88108 | 14988.88 | 24117.67 |
| Weighted Average | 0.89308 | 14158.45 | 22440.92 |

Figure 15: Model Comparison

The first model I fit was an ‘Intuition Model’. It is a linear model created based on common sense and domain knowledge and serves as a simple baseline for comparison. It is in a sense a proxy for the ‘expert’ opinion that a real estate agent might give as a sale price prediction and only Neighborhood, Ground Floor Living Area, Age At Sale, Overall Quality, Years Since Remodelling and Pool Area. The next model was the full model with 328 features, which predictably achieved a lower MSE but suffered from overfitting when using the test set.

I then added polynomial and interaction terms, and was surprised to see that these terms explained an additional amount of the variance in the test data. The polynomial terms added were chosen by looking at my plots of log(Sale Price) against each variable and looking for nonlinear relationships. They were Lot Area, First and Second Floor Area, Area and Pool Area, all raised to the second power. The interactions I chose were between neighborhood and first Floor Area, Overall Condition and Quality, and Lot Area. These were chosen based on my intuition about real estate and the theory that neighborhood and quality/size of the house will have a complex relationship that is not separable.

Before trying Lasso, I tried principal component regression and ridge regression. Principal component regression combines dimensionality reduction through PCA with linear regression. It addresses multicollinearity issues and can improve model performance by capturing the most informative principal components. My final PCR model used the first 190 principal components, significantly less than the 328 coefficients we have in the full model, though we still need to measure all those features for this model, so we still have the issue of high data collection costs. Looking at the scree plot for the PCA, we see that most of the variation in the data can be explained using the first 20 or so principal components.

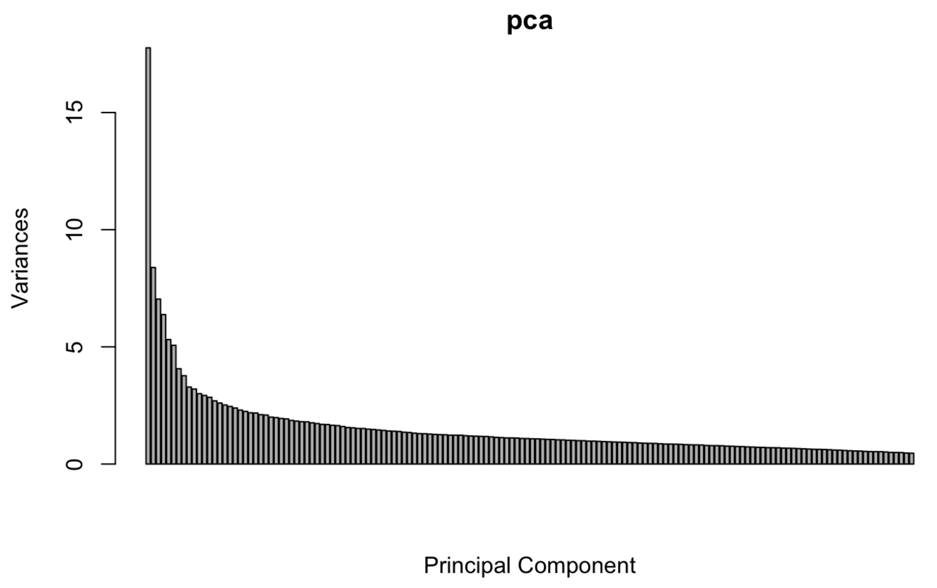


Figure 16: Principal component analysis scree plot

Before implementing my Lasso model, I also tried another regularization technique - Ridge regression. Ridge regression reduces the impact of highly correlated predictors, helps prevent overfitting and improve model generalization. The Ridge regression optimal lambda was found by cross validation to be 0.03326, and again this improved the out of sample $R^2$ of our model, and provided a significant improvement in RMSE.

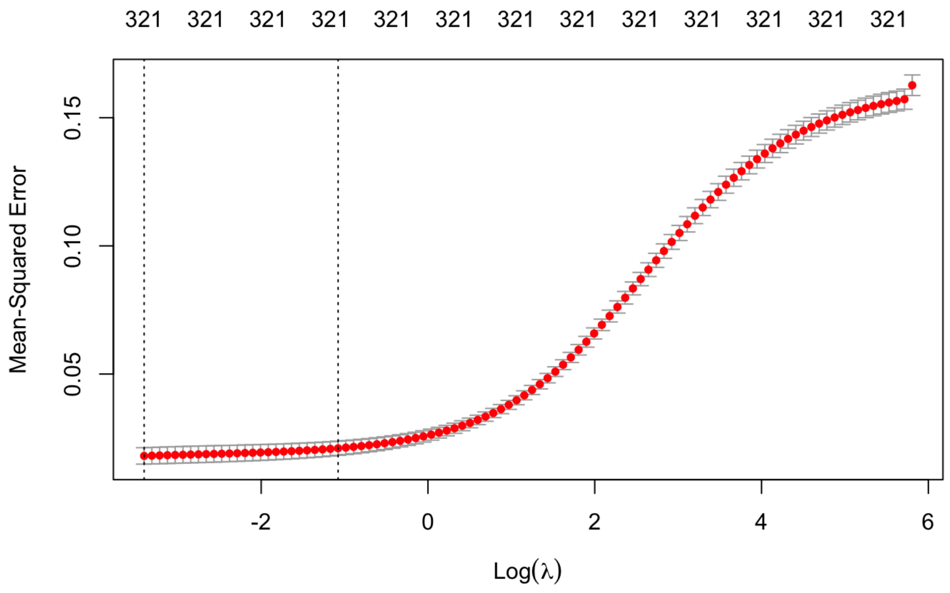
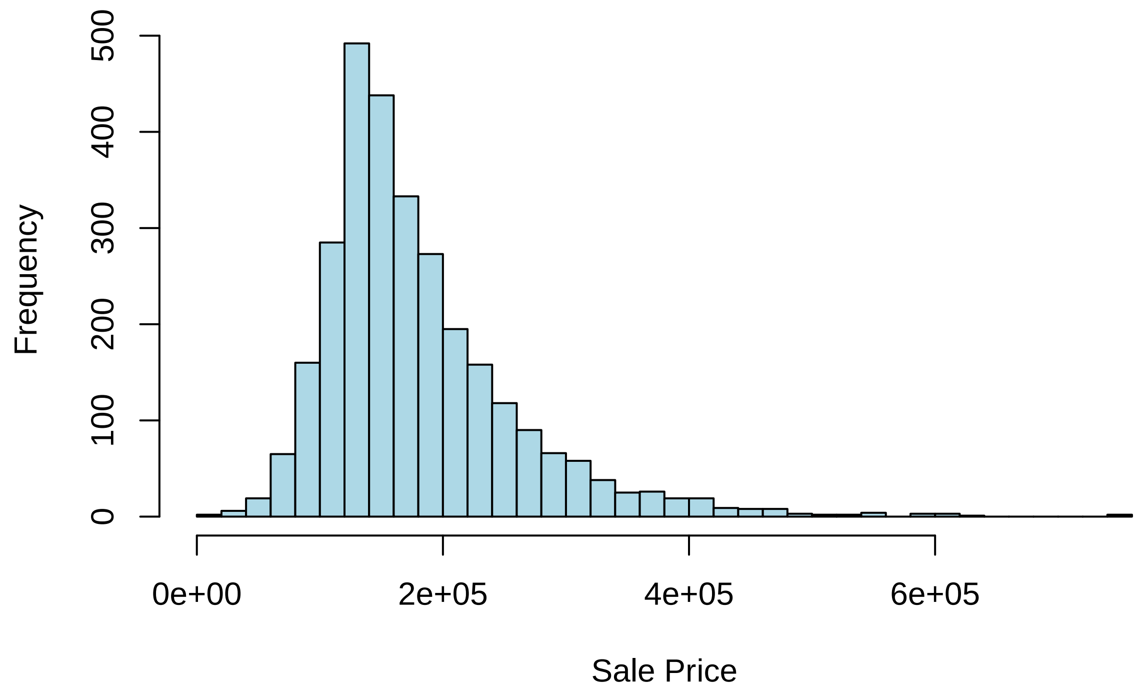


Figure 17: Ridge regression cross validation

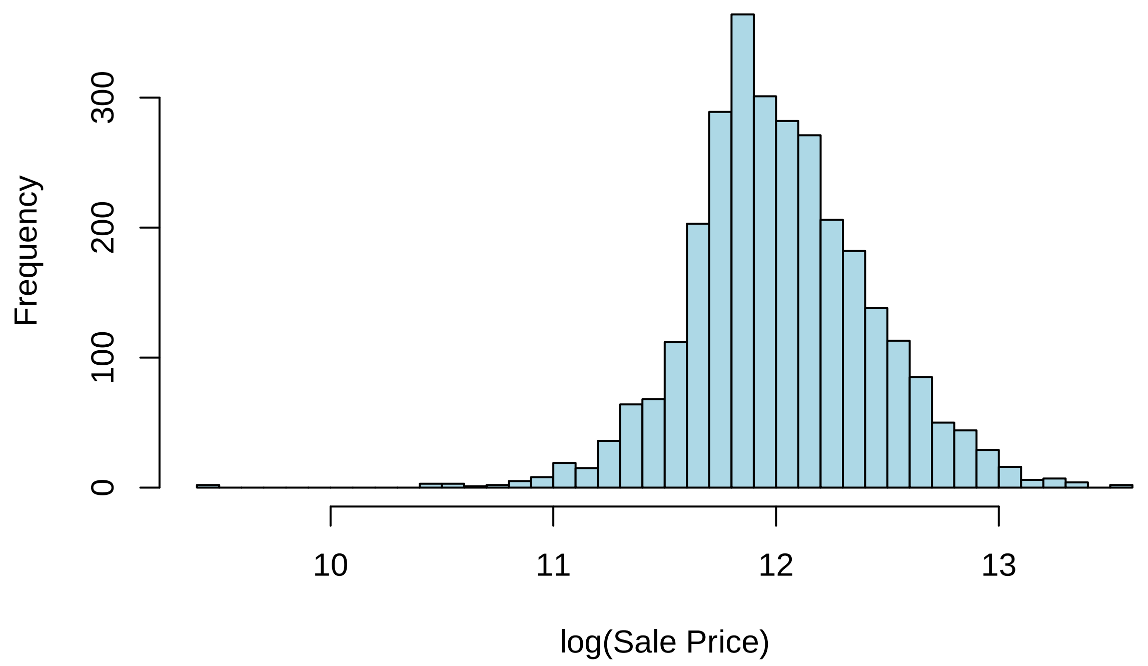
## 11. Appendix - R Code

**Feature Engineering and Data Cleaning**

```
rsq <- function(y, y_hat) {
    1 - sum((y - y_hat)^2) / sum((y - mean(y))^2)
}


# Read in the dataset
ames_data <- read.csv("/Users/zackbrodtman/Downloads/AmesHousing.csv.xls")

# Histogram of SalePrice
hist(x=ames_data$SalePrice, xlab = 'Sale Price', breaks = 50, col = '#ADD8E6', main =NULL)
```



```
# Histogram of log(SalePrice)
hist(x=log(ames_data$SalePrice), xlab = 'log(Sale Price)', breaks = 50, col = '#ADD8E6', main = NULL)
```

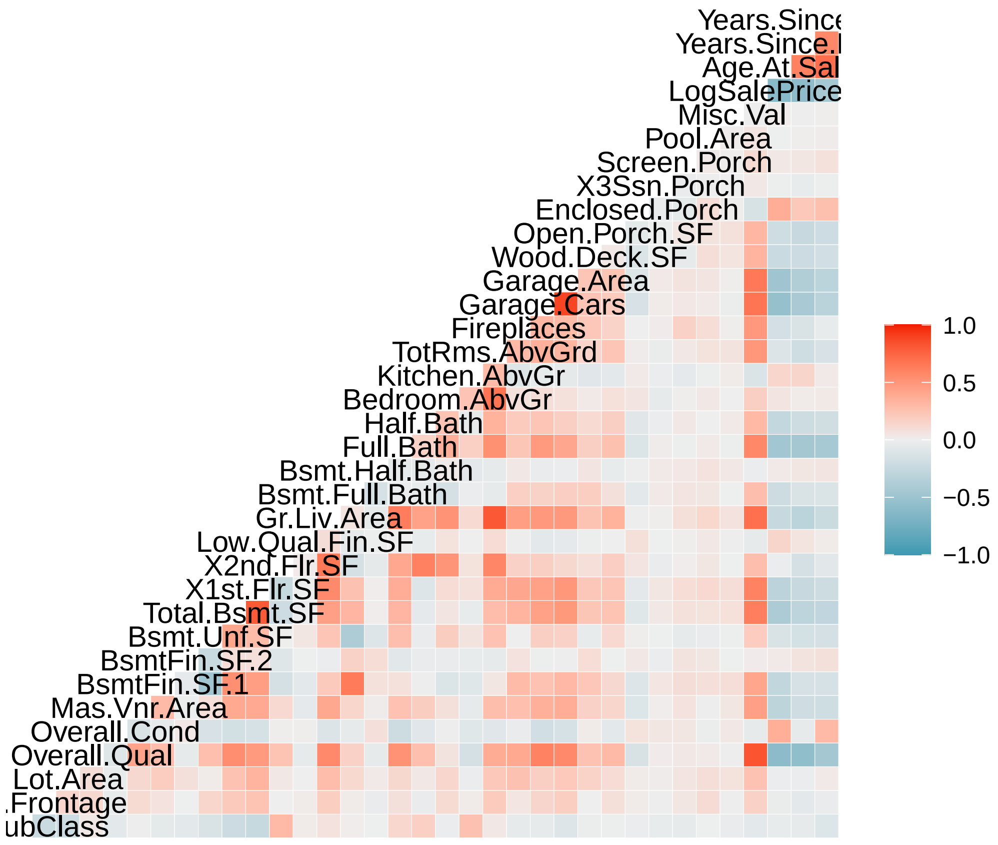


```
# Use log so that outliers have less impact on model
ames_data$LogSalePrice = log(ames_data$SalePrice)

# Add age and years since built/garage added
ames_data$Age.At.Sale <- ames_data$Yr.Sold - ames_data$Year.Built
ames_data$Years.Since.Remod <- ames_data$Yr.Sold - ames_data$Year.Remod.Add
ames_data$Years.Since.Garage <- ames_data$Yr.Sold - ames_data$Garage.Yr.Blt
ames_data <- subset(ames_data, select =-c(Year.Built, Year.Remod.Add, Garage.Yr.Blt, SalePrice, Order, 

# Convert month, year sold to categorical data and fix bath field
ames_data$Mo.Sold <- as.character(ames_data$Mo.Sold)
ames_data$Yr.Sold <- as.character(ames_data$Yr.Sold)
ames_data$Bsmt.Full.Bath <- as.numeric(ames_data$Bsmt.Full.Bath)
ames_data$Bsmt.Half.Bath <- as.numeric(ames_data$Bsmt.Half.Bath)



# Replace null values with "Missing" for categorical data and 0 for numerical data
ames_data$Alley[is.na(ames_data$Alley)] = "Missing"
ames_data$Bsmt.Qual[is.na(ames_data$Bsmt.Qual)] = "Missing"
ames_data$Bsmt.Cond[is.na(ames_data$Bsmt.Cond)] = "Missing"
ames_data$Bsmt.Exposure[is.na(ames_data$Bsmt.Exposure)] = "Missing"
ames_data$BsmtFin.Type.1[is.na(ames_data$BsmtFin.Type.1)] = "Missing"
ames_data$BsmtFin.Type.2[is.na(ames_data$BsmtFin.Type.2)] = "Missing"
ames_data$Fireplace.Qu[is.na(ames_data$Fireplace.Qu)] = "Missing"
ames_data$Garage.Qual[is.na(ames_data$Garage.Qual)] = "Missing"
```

```r
ames_data$Garage.Cond[is.na(ames_data$Garage.Cond)] = "Missing"
ames_data$Garage.Type[is.na(ames_data$Garage.Type)] = "Missing"
ames_data$Garage.Finish[is.na(ames_data$Garage.Finish)] = "Missing"
ames_data$Pool.QC[is.na(ames_data$Pool.QC)] = "Missing"
ames_data$Fence[is.na(ames_data$Fence)] = "Missing"
ames_data$Misc.Feature[is.na(ames_data$Misc.Feature)] = "Missing"
ames_data$Lot.Frontage[is.na(ames_data$Lot.Frontage)] = 0
ames_data$Mas.Vnr.Area[is.na(ames_data$Mas.Vnr.Area)] = 0
ames_data$BsmtFin.SF.1[is.na(ames_data$BsmtFin.SF.1)] = 0
ames_data$BsmtFin.SF.2[is.na(ames_data$BsmtFin.SF.2)] = 0
ames_data$Bsmt.Unf.SF[is.na(ames_data$Bsmt.Unf.SF)] = 0
ames_data$Total.Bsmt.SF[is.na(ames_data$Total.Bsmt.SF)] = 0
ames_data$Garage.Cars[is.na(ames_data$Garage.Cars)] = 0
ames_data$Garage.Area[is.na(ames_data$Garage.Area)] = 0
ames_data$Years.Since.Garage[is.na(ames_data$Years.Since.Garage)] = 0
ames_data$Bsmt.Full.Bath[is.na(ames_data$Bsmt.Full.Bath)] = 0
ames_data$Bsmt.Half.Bath[is.na(ames_data$Bsmt.Half.Bath)] = 0

# Pairwise correllation

# Select only the numeric columns
ames_data_numeric <- select_if(ames_data, is.numeric)

# Calculate the correlation matrix
cor_matrix <- cor(ames_data_numeric)

find_high_cor_pairs <- function(cor_matrix) {
  # Set the diagonal to 0
  diag(cor_matrix) <- 0

  # Calculate absolute correlation matrix
  abs_cor_matrix <- abs(cor_matrix)

  # Get the row and column indices of the high correlation pairs
  high_cor_pairs <- which(abs_cor_matrix > 0.80 & abs_cor_matrix < 1, arr.ind = TRUE)

  # Create a dataframe to store the pairs and their correlations
  cor_df <- data.frame(Variable_1 = rownames(cor_matrix)[high_cor_pairs[,1]],
                       Variable_2 = colnames(cor_matrix)[high_cor_pairs[,2]],
                       Correlation = cor_matrix[high_cor_pairs])

  # Sort the dataframe by correlation in descending order
  cor_df <- cor_df[order(-abs(cor_df$Correlation)),]

  # Return the dataframe
  cor_df
}

# Call the function with the correlation matrix
find_high_cor_pairs(cor_matrix)
```

```
##      Variable_1    Variable_2 Correlation
## 6   Garage.Area   Garage.Cars   0.8898660
## 7   Garage.Cars   Garage.Area   0.8898660
## 1  LogSalePrice  Overall.Qual   0.8256450
## 8  Overall.Qual  LogSalePrice   0.8256450
## 4 TotRms.AbvGrd   Gr.Liv.Area   0.8077721
## 5   Gr.Liv.Area TotRms.AbvGrd   0.8077721
## 2   X1st.Flr.SF Total.Bsmt.SF   0.8004287
## 3 Total.Bsmt.SF   X1st.Flr.SF   0.8004287
```

```
# Graphical representation
ggcorr(ames_data_numeric)
```

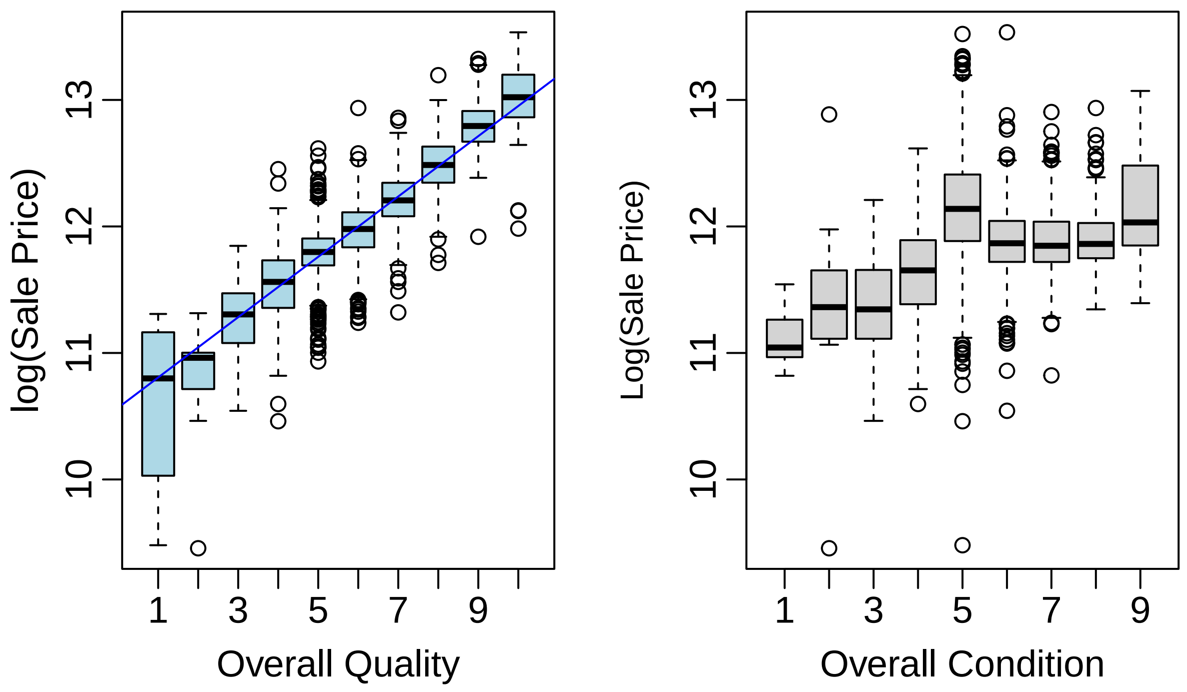


```
# Garage Area and Garage Cars highly correllated
# as are Total Rooms Above Grade and Above Grade Living Area
# and First Floor SF and Basement SF. Lets remove Total Rooms Above Grade, Total Basement SF and Garage
ames_data <- subset(ames_data, select =-c(TotRms.AbvGrd, Garage.Cars, Total.Bsmt.SF))

# We also notice which variables are positively correllated with LogSalePrice: Garage Area, Ground Livi
# and which are negatively correllated: Age at sale, Years Since remodelling
```

## EDA

```
# Quality and condition linear relationship with log(SalePrice)
par(mfrow=c(1,2), mar=c(4,4,2,1), oma=c(0,0,2,0), cex.lab=1.2, cex.axis=1.2, mgp=c(2,0.6,0))
```

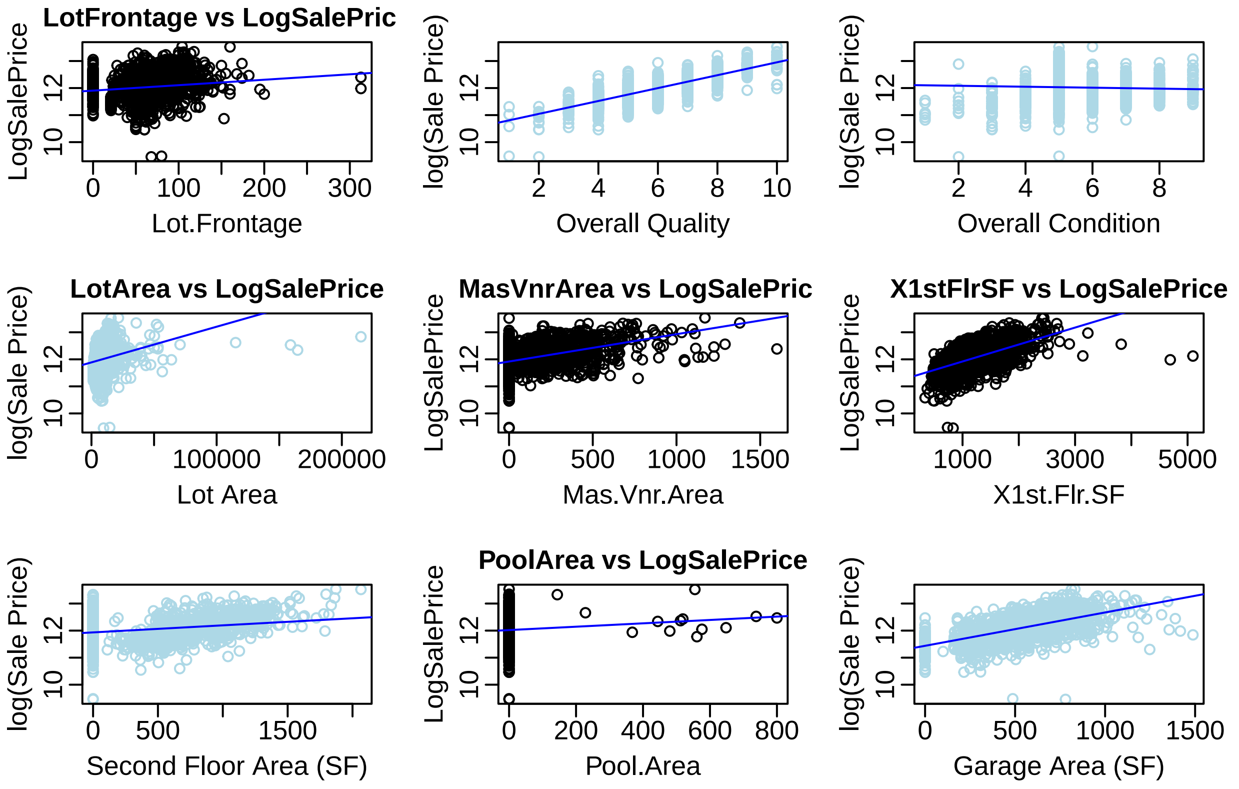
```
boxplot(LogSalePrice ~ Overall.Qual, data = ames_data, xlab = "Overall Quality", ylab = "log(Sale Price)
abline(lm(LogSalePrice ~ Overall.Qual, data = ames_data), col="blue")

boxplot(LogSalePrice ~ Overall.Cond, data = ames_data, xlab = "Overall Condition", ylab = "")
mtext("Log(Sale Price)", side=2, line=2.5)
```



```
# Nonlinear relationships
par(mfrow=c(3,3), figsize=c(15,15))
```

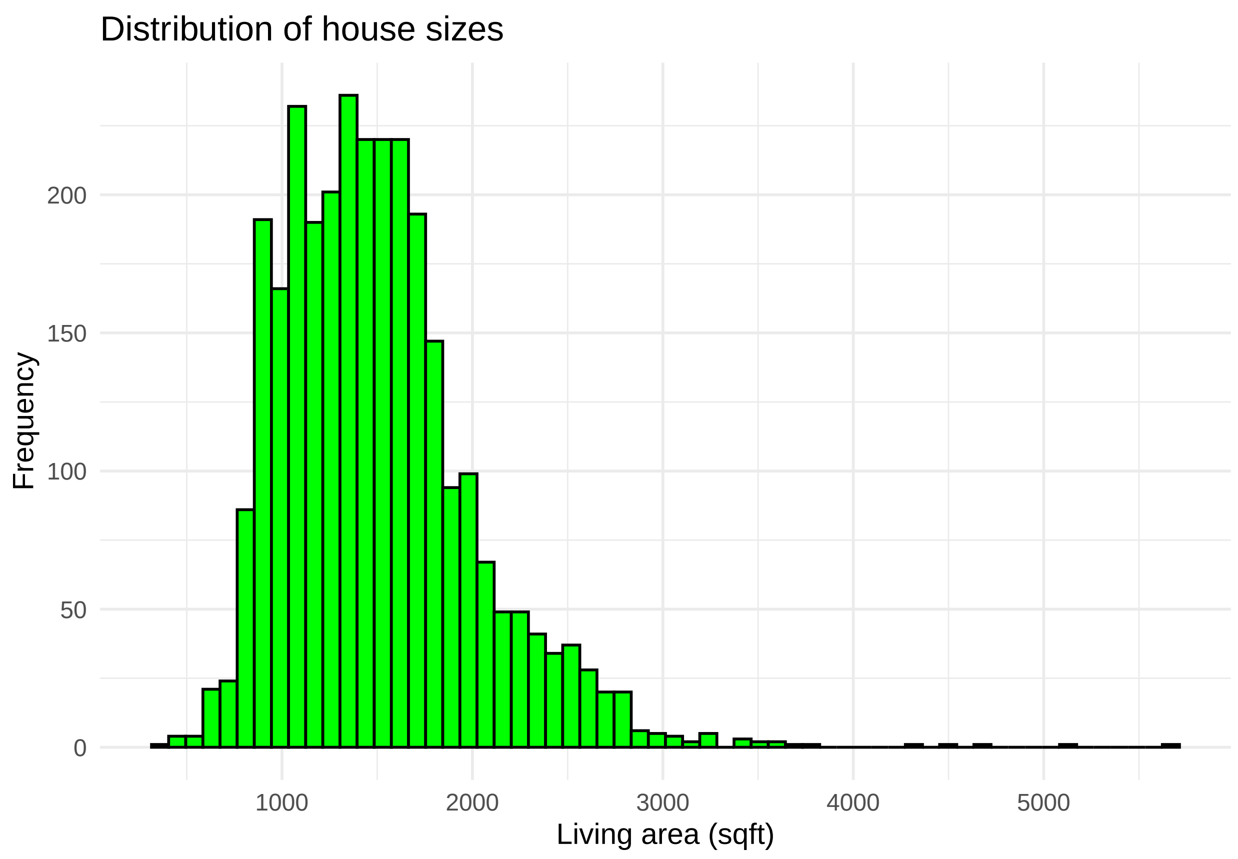
```
## Warning in par(mfrow = c(3, 3), figsize = c(15, 15)): "figsize" is not a
## graphical parameter
```

```
with(ames_data, {
  plot(Lot.Frontage, LogSalePrice, main="LotFrontage vs LogSalePrice")
  abline(lm(LogSalePrice ~ Lot.Frontage), col="blue")
  plot(Overall.Qual, LogSalePrice, main=NULL, xlab = 'Overall Quality', ylab = 'log(Sale Price)', col =
  abline(lm(LogSalePrice ~ Overall.Qual), col="blue")
  plot(Overall.Cond, LogSalePrice, main=NULL, xlab = 'Overall Condition', ylab = 'log(Sale Price)', col
  abline(lm(LogSalePrice ~ Overall.Cond), col="blue")
  plot(Lot.Area, LogSalePrice, main="LotArea vs LogSalePrice", xlab = 'Lot Area', ylab = 'log(Sale Pric
  abline(lm(LogSalePrice ~ Lot.Area), col="blue")
  plot(Mas.Vnr.Area, LogSalePrice, main="MasVnrArea vs LogSalePrice")
  abline(lm(LogSalePrice ~ Mas.Vnr.Area), col="blue")
  plot(X1st.Flr.SF, LogSalePrice, main="X1stFlrSF vs LogSalePrice")
  abline(lm(LogSalePrice ~ X1st.Flr.SF), col="blue")
  plot(X2nd.Flr.SF, LogSalePrice, main=NULL, xlab = 'Second Floor Area (SF)', ylab = 'log(Sale Price)',
  abline(lm(LogSalePrice ~ X2nd.Flr.SF), col="blue")
  plot(Pool.Area, LogSalePrice, main="PoolArea vs LogSalePrice")
  abline(lm(LogSalePrice ~ Pool.Area), col="blue")
  plot(Garage.Area, LogSalePrice,main=NULL, xlab = 'Garage Area (SF)', ylab = 'log(Sale Price)', col =
  abline(lm(LogSalePrice ~ Garage.Area), col="blue")
})
```



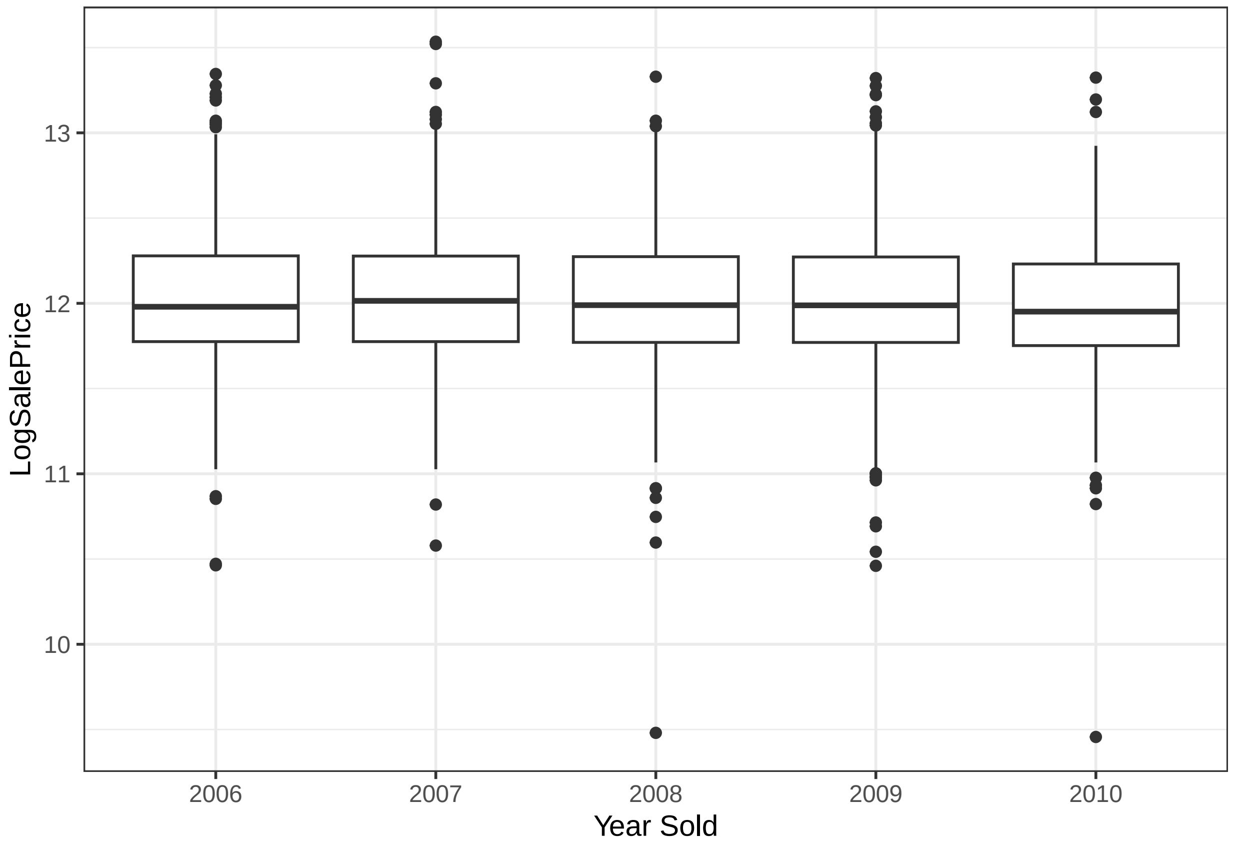
```
# Discrete
#par(mfrow=c(4,4), mar=c(5,5,3,1), oma=c(2,2,0,0),
#    fig=c(0,1,0,1), cex.lab=1.2, cex.main=1.2, cex.axis=1.2)
#boxplot(LogSalePrice ~ Bsmt.Full.Bath, data=ames_data,
#         xlab="Bsmt Full Bath", ylab="Log Sale Price")
#boxplot(LogSalePrice ~ Bsmt.Half.Bath, data=ames_data,
#         xlab="Bsmt Half Bath", ylab="Log Sale Price")
#boxplot(LogSalePrice ~ Full.Bath, data=ames_data,
#         xlab="Full Bath", ylab="Log Sale Price")
#boxplot(LogSalePrice ~ Half.Bath, data=ames_data,
#         xlab="Half Bath", ylab="Log Sale Price")
#boxplot(LogSalePrice ~ Bedroom.AbvGr, data=ames_data,
#         xlab="Bedroom Above Grade", ylab="Log Sale Price")
#boxplot(LogSalePrice ~ Kitchen.AbvGr, data=ames_data,
#         xlab="Kitchen Above Grade", ylab="Log Sale Price")
#boxplot(LogSalePrice ~ Fireplaces, data=ames_data,
```

```r
#         xlab="Fireplaces", ylab="log(Sale Price)", col = '#ADD8E6')
#abline(lm(LogSalePrice ~ Fireplaces, data = ames_data), col="blue")


barplot(table(ames_data$Age.At.Sale),
        main = "How old are houses at sale?",
        xlab = "Years",
        ylab = "Number of houses",
        col = brewer.pal(9, "Greens"))

ggplot(ames_data, aes(x = Gr.Liv.Area)) +
  geom_histogram(color = "black", fill = "green", bins = 60) +
  labs(title = "Distribution of house sizes", x = "Living area (sqft)", y = "Frequency") +
  theme_minimal()

# We see which variables have a nonlinear relationship with LogSalePrice: LotArea, TotalBsmtSF, X1st.Fl
```



**Regression analysis**

```r
# Encode categorical variables
vars_to_encode <- c("MS.Zoning", "Street", "Alley", "Lot.Shape", "Land.Contour",
                    "Utilities", "Lot.Config", "Land.Slope", "Neighborhood", "Condition.1",
                    "Condition.2", "Bldg.Type", "House.Style",
                     "Roof.Style", "Exterior.1st", "Exterior.2nd", "Mas.Vnr.Type",
                     "Exter.Qual", "Exter.Cond", "Foundation", "Bsmt.Qual", "Bsmt.Cond",
                     "Bsmt.Exposure", "BsmtFin.Type.1", "BsmtFin.Type.2", "Heating", "Heating.QC",
                     "Central.Air", "Electrical",  "Kitchen.Qual",
                     "Functional", "Fireplace.Qu", "Garage.Type", "Garage.Finish",
                      "Garage.Qual", "Garage.Cond", "Paved.Drive", "Pool.QC", "Fence", "Misc.Feature",
                      "Misc.Val", "Mo.Sold", "Sale.Type", "Sale.Condition", "Roof.Matl")

ames_data_encoded <- dummy_cols(ames_data, select_columns =  vars_to_encode, remove_first_dummy = TRUE,
all = 'LogSalePrice ~ .  - BsmtFin.SF.1 - Overall.Qual -Gr.Liv.Area- X1st.Flr.SF'
# After encoding variables, we have ~400 variables and ~2000 observations so p is almost equal to n. Th

#Box Plots show stable data across all years so we can split by year
ggplot(data=ames_data, aes(x=factor(Yr.Sold), y=LogSalePrice)) +
  geom_boxplot() +
  labs(title=NULL, x="Year Sold", y="LogSalePrice") +
  theme_bw()
```



```
# Split train/test by year
train <- ames_data_encoded[ames_data_encoded$Yr.Sold==2006 | ames_data_encoded$Yr.Sold==2007  | ames_da
test <- ames_data_encoded[  ames_data_encoded$Yr.Sold==2010|ames_data_encoded$Yr.Sold==2009,]

train <- subset(train, select =-Yr.Sold)
test <- subset(test, select =-Yr.Sold)
```

```
# Fit a linear model on all the features
model_full <- lm(formula = all, data = train)

# Evaluate the model on the test set
predictions_full <- predict(model_full, newdata = test)
```

```
## Warning in predict.lm(model_full, newdata = test): prediction from a
## rank-deficient fit may be misleading
```

```
# Calculate R^2, MAE, and RMSE for LogSalePrice
rsq_full <- rsq(test$LogSalePrice, predictions_full)
mae_full <- mae(exp(test$LogSalePrice), exp(predictions_full))
rmse_full <- rmse(exp(test$LogSalePrice), exp(predictions_full))

paste('OSR^2:', rsq_full, 'MAE:', mae_full, 'RMSE:', rmse_full)
```

```
## [1] "OSR^2: 0.846042885309308 MAE: 19062.8525227969 RMSE: 29655.4713233356"
```

```
#anova(model_full)
```

### Intuitive Model

```
# Fit a model that a non-technical person might fit based on their intuition about property prices

model_int <- lm(LogSalePrice ~  Neighborhood_Blueste + Neighborhood_BrDale +
    Neighborhood_BrkSide + Neighborhood_ClearCr + Neighborhood_CollgCr +
    Neighborhood_Crawfor + Neighborhood_Edwards + Neighborhood_Gilbert +
    Neighborhood_Greens + Neighborhood_GrnHill + Neighborhood_IDOTRR +
    Neighborhood_Landmrk + Neighborhood_MeadowV + Neighborhood_Mitchel +
    Neighborhood_NAmes + Neighborhood_NoRidge + Neighborhood_NPkVill +
    Neighborhood_NridgHt + Neighborhood_NWAmes + Neighborhood_OldTown +
    Neighborhood_Sawyer + Neighborhood_SawyerW + Neighborhood_Somerst +
    Neighborhood_StoneBr + Neighborhood_SWISU + Neighborhood_Timber +
    Neighborhood_Veenker +  Gr.Liv.Area + Age.At.Sale +
    Overall.Qual+ Years.Since.Remod + Pool.Area
    , data = train)

# Evaluate the model on the test set
predictions_int <- predict(model_int, newdata = test)


# Calculate R^2, MAE, and RMSE for LogSalePrice
rsq_int <- rsq(test$LogSalePrice, predictions_int)
mae_int <- mae(exp(test$LogSalePrice), exp(predictions_int))
rmse_int <- rmse(exp(test$LogSalePrice), exp(predictions_int))

paste('OSR^2:', rsq_int, 'MAE:', mae_int, 'RMSE:', rmse_int)
```

```
## [1] "OSR^2: 0.834398209116415 MAE: 20118.3961734162 RMSE: 30130.8037132026"
```

**Add polynomials**

```r
# columns for polynomial terms:
# "Lot.Area", "X1st.Flr.SF", "X2nd.Flr.SF",
# "Gr.Liv.Area", "Garage.Area", "Pool.Area"

ames_data_encoded$Lot.Area_poly <- (ames_data_encoded$Lot.Area)**2
ames_data_encoded$X1st.Flr.SF_poly <- (ames_data_encoded$X1st.Flr.SF)**2
ames_data_encoded$X2nd.Flr.SF_poly <- (ames_data_encoded$X2nd.Flr.SF)**2
ames_data_encoded$Gr.Liv.Area_poly <- (ames_data_encoded$Gr.Liv.Area)**2
ames_data_encoded$Garage.Area_poly <- (ames_data_encoded$Garage.Area)**2
ames_data_encoded$Pool.Area_poly <- (ames_data_encoded$Pool.Area)**2



# same train/test split
train_poly <- ames_data_encoded[ames_data_encoded$Yr.Sold==2006 | ames_data_encoded$Yr.Sold==2007 | ames
test_poly <- ames_data_encoded[ ames_data_encoded$Yr.Sold==2010 | ames_data_encoded$Yr.Sold==2009 ,]

train_poly <- subset(train_poly, select =-Yr.Sold)
test_poly <- subset(test_poly, select =-Yr.Sold)

# Fit a linear model on all the features
model_full_poly <- lm(formula = all, data = train_poly)
#summary(model_full_poly)

# Evaluate the model on the test set
predictions_full_poly <- predict(model_full_poly, newdata = test_poly)
```

```
## Warning in predict.lm(model_full_poly, newdata = test_poly): prediction from a
## rank-deficient fit may be misleading
```

```r
# Calculate R^2, MAE, and RMSE for LogSalePrice
rsq_full_poly <- rsq(test_poly$LogSalePrice, predictions_full_poly)
mae_full_poly <- mae(exp(test_poly$LogSalePrice), exp(predictions_full_poly))
rmse_full_poly <- rmse(exp(test_poly$LogSalePrice), exp(predictions_full_poly))

paste('OSR^2:', rsq_full_poly, 'MAE:', mae_full_poly, 'RMSE:', rmse_full_poly)
```

```
## [1] "OSR^2: 0.865882022011524 MAE: 16797.7345931931 RMSE: 26921.6777682068"
```

**interaction neighborhood & quality, age at sale & exterior material quality,**

```r
# List of neighborhood dummy variables
neighborhood_vars <- c("Neighborhood_Blueste", "Neighborhood_BrDale", "Neighborhood_BrkSide", "Neighbor
                       "Neighborhood_CollgCr", "Neighborhood_Crawfor", "Neighborhood_Edwards", "Neighbo
                       "Neighborhood_Greens", "Neighborhood_GrnHill", "Neighborhood_IDOTRR", "Neighborh
                       "Neighborhood_MeadowV", "Neighborhood_Mitchel", "Neighborhood_NAmes", "Neighborh
                       "Neighborhood_NPkVill", "Neighborhood_NridgHt", "Neighborhood_NWAmes", "Neighbor
                       "Neighborhood_Sawyer", "Neighborhood_SawyerW", "Neighborhood_Somerst", "Neighbor
```

```
                        "Neighborhood_SWISU", "Neighborhood_Timber", "Neighborhood_Veenker")

# List of OverallQual dummy variables
overall_qual_vars <- c("Overall.Qual", "Overall.Cond")

# Create a matrix with all combinations of neighborhood and overall quality variables
interaction_matrix <- outer(neighborhood_vars, overall_qual_vars, paste, sep="*")

# Flatten the matrix to a single vector
interaction_vector <- c(interaction_matrix)

# Join all interactions with "+" sign
interaction_formula <- paste(interaction_vector, collapse=" + ")


model_interact <- lm(LogSalePrice ~ . + Neighborhood_Blueste*Lot.Area + Neighborhood_BrDale*Lot.Area + 
Neighborhood_Blueste*Overall.Qual + Neighborhood_BrDale*Overall.Qual + Neighborhood_BrkSide*Overall.Qua
data = train)

# Evaluate the model on the test set

predictions_interact <- predict(model_interact, newdata = test)
```

```
## Warning in predict.lm(model_interact, newdata = test): prediction from a
## rank-deficient fit may be misleading
```

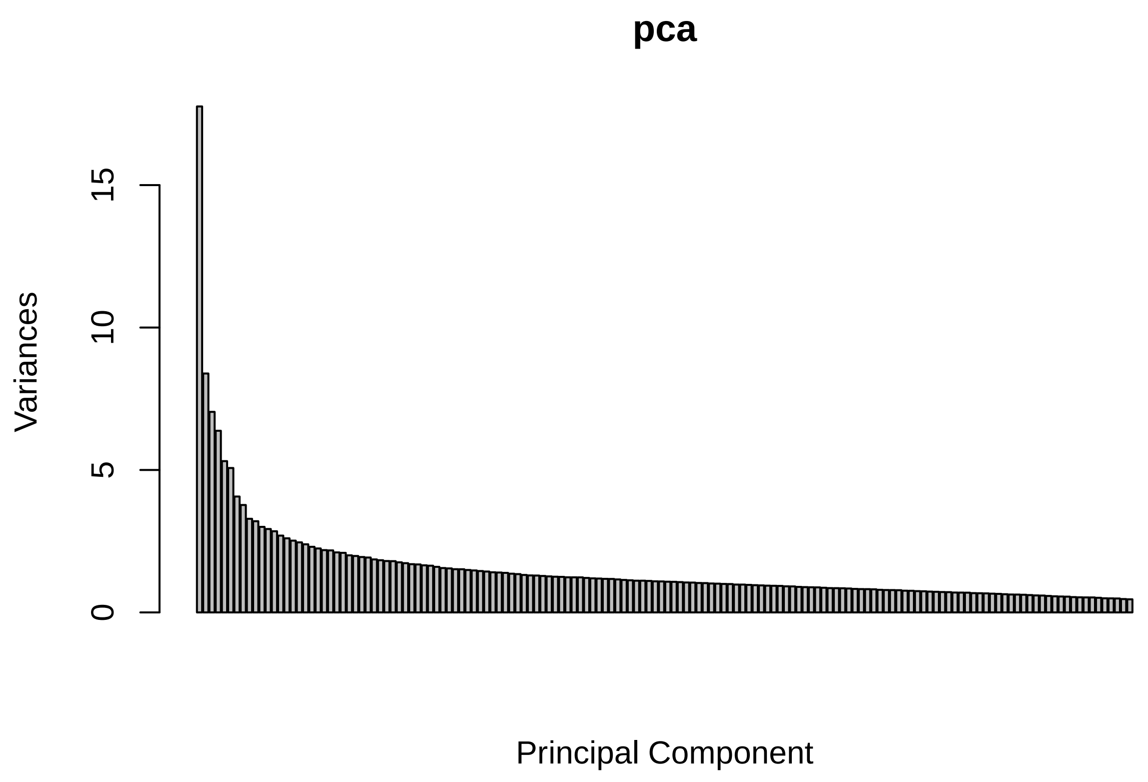
```
# Calculate R^2, MAE, and RMSE for LogSalePrice
rsq_interact <- rsq(test$LogSalePrice, predictions_interact)
mae_interact <- mae(exp(test$LogSalePrice), exp(predictions_interact))
rmse_interact <- rmse(exp(test$LogSalePrice), exp(predictions_interact))

paste('OSR^2:', rsq_interact, 'MAE:', mae_interact, 'RMSE:', rmse_interact)
```

```
## [1] "OSR^2: 0.864480535823022 MAE: 15606.5560254488 RMSE: 25889.9716279566"
```

**Principal component regression**

```
# Perform principal component regression using train_poly and test_poly to predict LogSalePrice
set.seed(123) # for reproducibility

library(pls)

# Remove columns with low standard deviation
train_pca <- train_poly[, apply(train_poly, 2, sd) > 0.05]
test_pca <- test_poly[, apply(train_poly, 2, sd) > 0.05]

# Perform PCA
pca <- prcomp(subset(train_pca, select = -LogSalePrice), scale. = TRUE)

# Transform train and test data using PCA
train_pca <- data.frame(predict(pca, subset(train_pca, select = -LogSalePrice)))
```

```r
test_pca <- data.frame(predict(pca, subset(test_pca, select = -LogSalePrice)))
train_pca$LogSalePrice = train_poly$LogSalePrice
test_pca$LogSalePrice = test_poly$LogSalePrice
train_pca <- train_pca %>% relocate(LogSalePrice)
test_pca <- test_pca %>% relocate(LogSalePrice)

# Create scree plot
screeplot(pca, npcs = 150, xlab = "Principal Component")
```

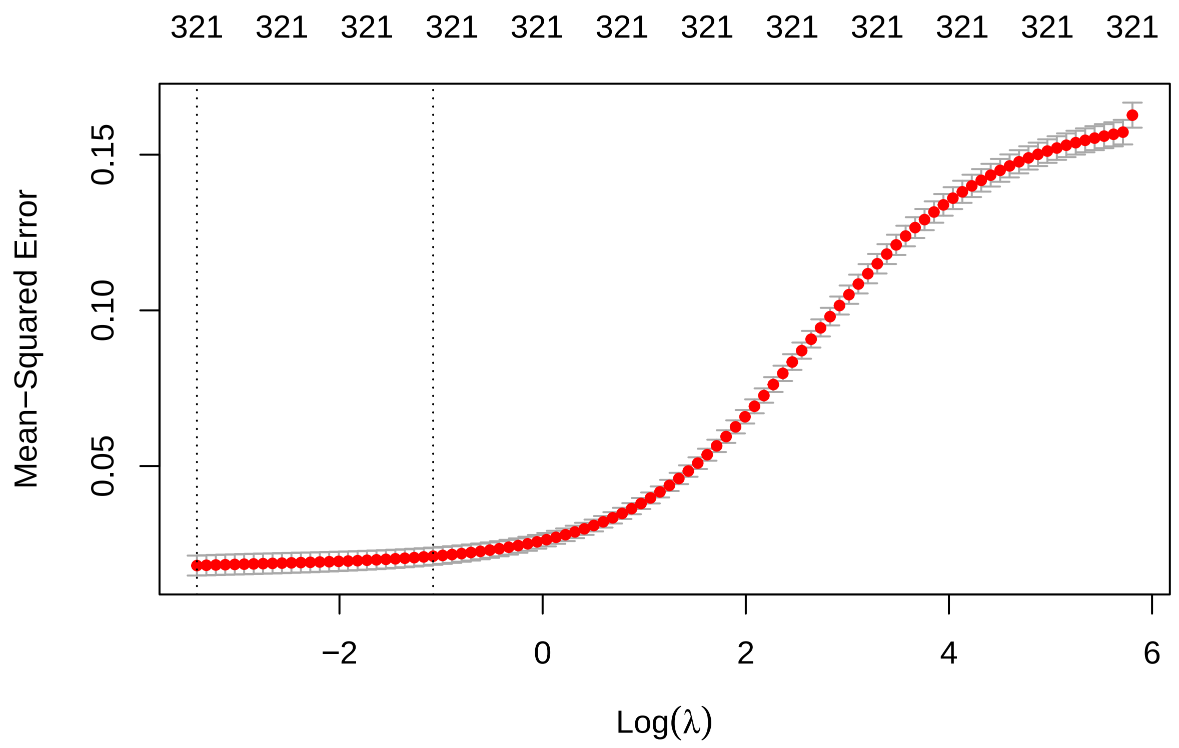


```r
# Fit linear regression model with 40 principal components
model <- lm(LogSalePrice ~ ., data = train_pca[,1:190])

# Make predictions on test data
predictions_pca <- predict(model, newdata = test_pca)

# Calculate R^2, MAE, and RMSE for LogSalePrice
rsq_pca <- rsq(test_pca$LogSalePrice, predictions_pca)
mae_pca <- mae(exp(test_pca$LogSalePrice), exp(predictions_pca))
rmse_pca <- rmse(exp(test_pca$LogSalePrice), exp(predictions_pca))

paste('OSR^2:', rsq_pca, 'MAE:', mae_pca, 'RMSE:', rmse_pca)
```

```
## [1] "OSR^2: 0.879550506393145 MAE: 16802.4969527413 RMSE: 25684.4322313652"
```

**Ridge Regression**

```r
lambdas <- 10^seq(7, -8, by = -.1)
cv_fit <- cv.glmnet(as.matrix(subset(train_poly, select = -LogSalePrice)), train_poly$LogSalePrice, alph
opt_lambda <- cv_fit$lambda.min

best_model <- glmnet(as.matrix(subset(train_poly, select = -LogSalePrice)), train_poly$LogSalePrice, alp

plot(cv_fit)
```



```r
ridgepred <- predict(best_model,  newx = as.matrix(subset(test_poly, select = -LogSalePrice)))

# Calculate R^2, MAE, and RMSE for LogSalePrice
rsq_ridge <- rsq(test_poly$LogSalePrice, ridgepred)
mae_ridge <- mae(exp(test_poly$LogSalePrice), exp(ridgepred))
rmse_ridge <- rmse(exp(test_poly$LogSalePrice), exp(ridgepred))

paste('OSR^2:', rsq_ridge, 'MAE:', mae_ridge, 'RMSE:', rmse_ridge)
```

```
## [1] "OSR^2: 0.878693656191292 MAE: 15229.1004511718 RMSE: 23990.6488997949"
```

### Lasso

```
# Setting alpha = 1 implements lasso regression
lasso_reg <- cv.glmnet(as.matrix(subset(train_poly, select = -LogSalePrice)), train_poly$LogSalePrice, a

# Best
lambda_best <- lasso_reg$lambda.min

lasso_model <- glmnet(as.matrix(subset(train_poly, select = -LogSalePrice)), train_poly$LogSalePrice, al

#coef(lasso_model)

lasso_pred <- predict(lasso_model,  newx = as.matrix(subset(test_poly, select = -LogSalePrice)))

# Calculate R^2, MAE, and RMSE for LogSalePrice
rsq_lasso <- rsq(test_poly$LogSalePrice, lasso_pred)
mae_lasso <- mae(exp(test_poly$LogSalePrice), exp(lasso_pred))
rmse_lasso <- rmse(exp(test_poly$LogSalePrice), exp(lasso_pred))

paste('OSR^2:', rsq_lasso, 'MAE:', mae_lasso, 'RMSE:', rmse_lasso)
```

```
## [1] "OSR^2: 0.888567353749822 MAE: 14258.5507720159 RMSE: 22314.7465245682"
```

```
length(which(lasso_model$beta!=0))
```

```
## [1] 181
```

### Diagnostics

```
## Residual analysis
#lasso_residuals <- test_poly$LogSalePrice - lasso_pred
#plot(lasso_pred, lasso_residuals, xlab = "Predicted Values", ylab = "Residuals", main = "Lasso Residua
#abline(h = 0, col = "red")
#
## Homoscedasticity
#plot(lasso_pred, lasso_residuals, xlab = "Predicted Values", ylab = "Residuals", main = "Lasso Residua
#abline(h = 0, col = "red")

# Normality of residuals
##hist(lasso_residuals, main = "Histogram of Residuals")
#qqnorm(lasso_residuals)
#qqline(lasso_residuals)

# Calculate the standardized residuals
#lasso_residuals <- lasso_residuals / sd(lasso_residuals)
#
## Create a scale-location plot
#plot(sqrt(abs(lasso_pred)), lasso_residuals, xlab = "Fitted Values (sqrt(|Predicted Values|))", ylab =
#abline(h = 0, col = "red")
```

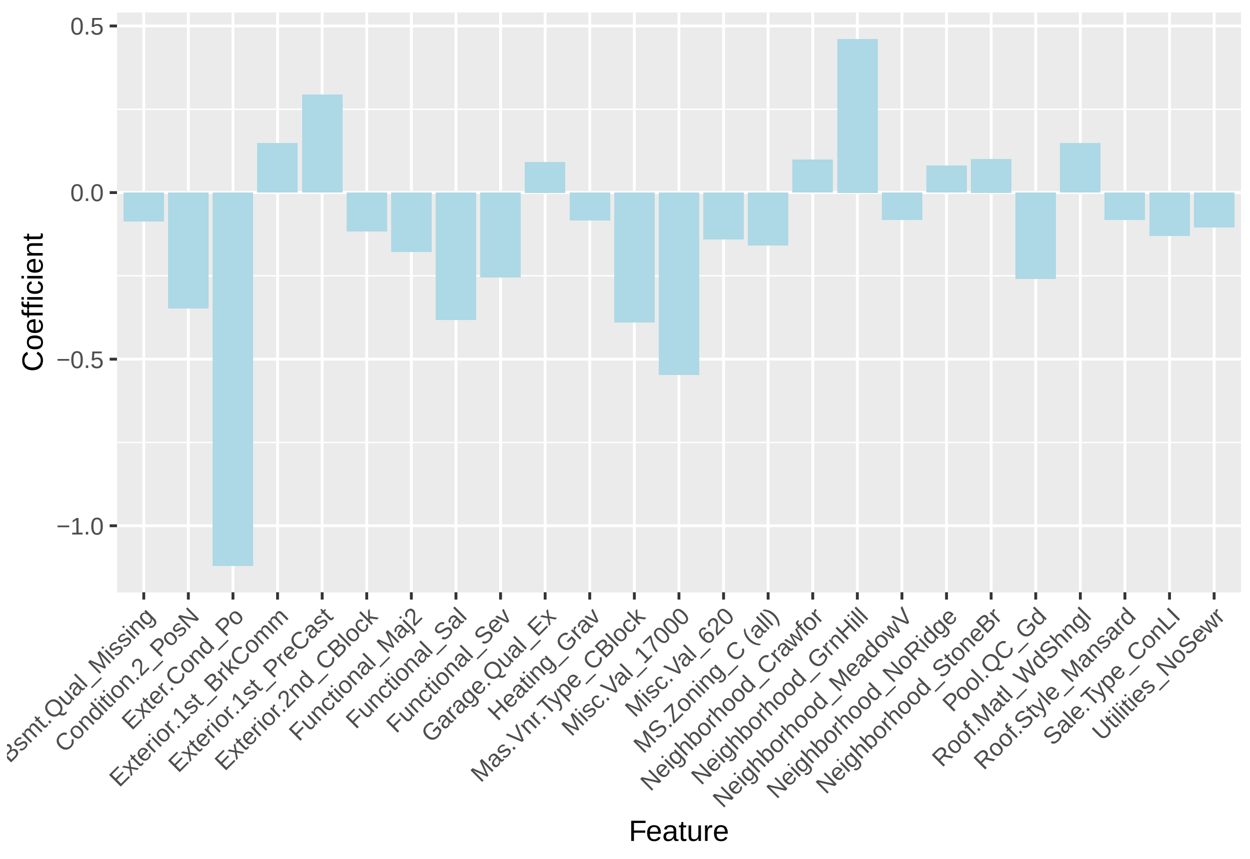
```r
#par(mfrow=c(4,4))
#plot(model_full)
```

### Feature Importance

```r
imp <- as.data.frame(varImp(model_full))
imp <- data.frame(overall = imp$Overall,
           names   = rownames(imp))
#imp[order(imp$overall,decreasing = T),]


#Most important features according to full model: Roof material, 1st floor SF, overall quality & condit

# How about for lasso?
length(which(lasso_model$beta!=0))
```

```
## [1] 181
```

```r
keep_variables = rownames(lasso_model$beta)[c(which(lasso_model$beta!=0))]






importance <- lasso_model$beta[c(which(abs(lasso_model$beta)>0.08))]
name <- rownames(lasso_model$beta)[c(which(abs(lasso_model$beta)>0.08))]

dflasso <- data.frame(name,importance)

# Plot a bar plot of feature importance
ggplot(dflasso, aes(x=name, y=importance)) +
  geom_bar(stat="identity", fill="#ADD8E6") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  xlab("Feature") +
  ylab("Coefficient")
```



```
#kept 181 features.
# Most important: overall qual and cond, basement full bath, full bath, half bath, bedrooms above grade
# neighborhood good: grn hill, crawford,
# neighborhood bad: NridgHt, noRidge, MeadowV

# Ridge? only got rid of 12 features
length(which(best_model$beta!=0))
```

```
## [1] 321
```

**XGBoost**

```
# Load required libraries
library(xgboost)
```

```
## Warning: package 'xgboost' was built under R version 4.1.2
```

```
##
## Attaching package: 'xgboost'
```

```
## The following object is masked from 'package:dplyr':
##
##     slice
```

```
library(caTools)
# Split the data into training, valid and testing sets

train_poly_xgb = train_poly[keep_variables]
test_poly_xgb = test_poly[keep_variables]
train_poly_xgb$LogSalePrice = train_poly$LogSalePrice
test_poly_xgb$LogSalePrice = test_poly$LogSalePrice


split <- sample.split(train_poly_xgb, SplitRatio = 0.7)
train_poly1 <- train_poly_xgb[split == TRUE,]
valid_poly <- train_poly_xgb[split == FALSE,]


# Prepare the data for XGBoost
train_matrix <- xgb.DMatrix(data = as.matrix(subset(train_poly1, select = -LogSalePrice)), label = trai
test_matrix <- xgb.DMatrix(data = as.matrix(subset(test_poly_xgb, select = -LogSalePrice)), label = tes
valid_matrix <- xgb.DMatrix(data = as.matrix(subset(valid_poly, select = -LogSalePrice)), label = valid



# Set the hyperparameters
params <- list(
  booster = "gbtree",
  objective = "reg:squarederror",
  eval_metric = "rmse",
  eta = 0.1,
  max_depth = 6,
  subsample = 0.7,
  colsample_bytree = 0.7,
  min_child_weight = 1,
  seed = 123
)

# Train the model
xgb_model <- xgb.train(
  params = params,
  data = train_matrix,
  nrounds = 1000,
  early_stopping_rounds = 10,
  verbose = FALSE,
  watchlist = list(train = train_matrix, test = valid_matrix)
)
```

```
## Warning in xgb.train(params = params, data = train_matrix, nrounds = 1000, :
## xgb.train: 'seed' is ignored in R package. Use 'set.seed()' instead.
```

```
# Make predictions on the test set
xgb_pred <- predict(xgb_model, newdata = test_matrix)


# Calculate R^2, MAE, and RMSE for LogSalePrice
rsq_xgb <- rsq(test_poly$LogSalePrice, xgb_pred)
```

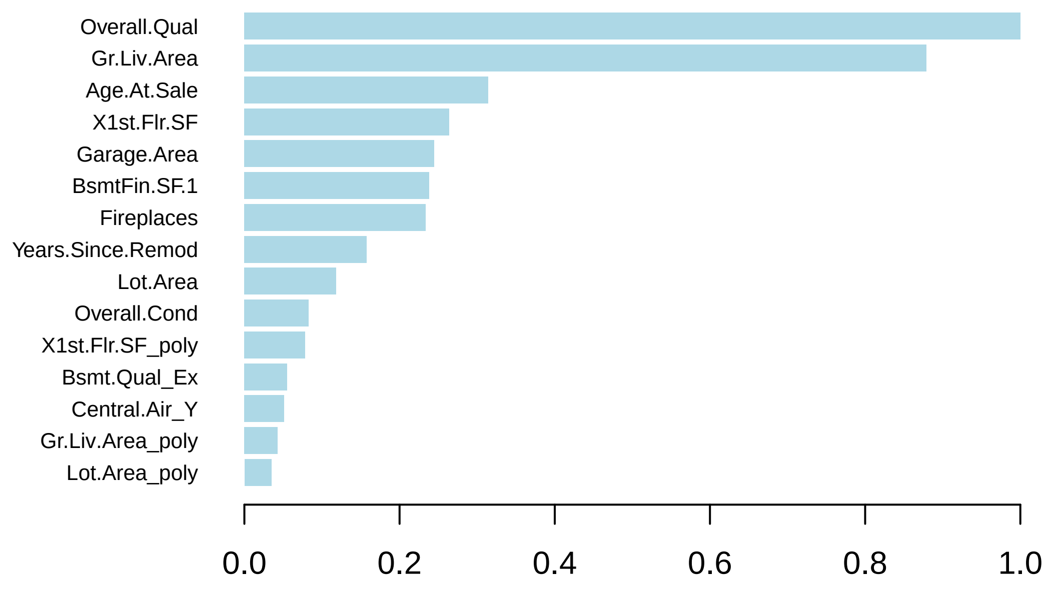
```r
mae_xgb <- mae(exp(test_poly$LogSalePrice), exp(xgb_pred))
rmse_xgb <- rmse(exp(test_poly$LogSalePrice), exp(xgb_pred))

paste('OSR^2:', rsq_xgb, 'MAE:', mae_xgb, 'RMSE:', rmse_xgb)
```

```
## [1] "OSR^2: 0.886955290431169 MAE: 15083.5122590726 RMSE: 23173.3249474187"
```

```r
xgb_varImp <- xgb.importance(model = xgb_model)

xgb_varImp
```

```
##                  Feature         Gain        Cover    Frequency
##   1:        Overall.Qual 2.269272e-01 4.807872e-02 0.0392335766
##   2:         Gr.Liv.Area 1.994573e-01 5.477467e-02 0.0526155718
##   3:         Age.At.Sale 7.131550e-02 3.452740e-02 0.0339111922
##   4:         X1st.Flr.SF 5.990831e-02 5.835677e-02 0.0629562044
##   5:         Garage.Area 5.551561e-02 5.058674e-02 0.0548965937
##  ---
## 148: Neighborhood_BrDale 1.808926e-05 1.197982e-03 0.0003041363
## 149:       Misc.Val_4500 1.353160e-05 2.608231e-03 0.0004562044
## 150:       Sale.Type_Con 1.072473e-05 2.824141e-05 0.0001520681
## 151:       Lot.Shape_IR3 7.309126e-06 8.836828e-05 0.0003041363
## 152:    Condition.1_RRNn 1.334631e-06 1.184317e-05 0.0001520681
```

```r
xgb.plot.importance(xgb_varImp, top_n = 15, rel_to_first = T, col = '#ADD8E6')
```



**combine best models**

```r
# Find best weights

# Define the possible weights to try
weights <- seq(0, 1, by = 0.05)

# Initialize the minimum MAE and best weights
min_mae <- Inf
best_weights <- c()

# Loop through all possible combinations of weights
for (w1 in weights) {
  for (w2 in weights) {
    # Check that the weights add up to 1
    if (w1 + w2 == 1) {
      # Calculate the weighted average predictions
      weighted_avg_pred <- w1 * lasso_pred + w2 * xgb_pred

      # Calculate the MAE
      model_mae <- mae(exp(test_poly$LogSalePrice), exp(weighted_avg_pred))

      # Check if this is the best MAE so far
      if (model_mae < min_mae) {
        min_mae <- model_mae
        best_weights <- c(w1, w2)
      }
    }
  }
}

# Print the best weights and the minimum MAE
cat("Best weights:", best_weights, "\n")
```

```
## Best weights: 0.7 0.3
```

```r
cat("Minimum MAE:", min_mae, "\n")
```

```
## Minimum MAE: 13913.96
```

```r
weighted_avg_pred <- (0.4* lasso_pred) + (0.6 * xgb_pred)

# Calculate R^2, MAE, and RMSE for LogSalePrice
rsq_meta <- rsq(test_poly$LogSalePrice, weighted_avg_pred)
mae_meta <- mae(exp(test_poly$LogSalePrice), exp(weighted_avg_pred))
rmse_meta <- rmse(exp(test_poly$LogSalePrice), exp(weighted_avg_pred))

paste('OSR^2:', rsq_meta, 'MAE:', mae_meta, 'RMSE:', rmse_meta)
```

```
## [1] "OSR^2: 0.896213951802872 MAE: 14134.5006152368 RMSE: 21764.9980401662"
```

**Results**

```
# create a list of model names and results
model_names <- c("Intuition Model", "Full Linear Model", "+ Interactions", "+ Polynomials", "Ridge", "La
rsquared <- c(rsq_int, rsq_full, rsq_interact, rsq_full_poly, rsq_ridge, rsq_lasso, rsq_pca, rsq_xgb, rs
mae <- c(mae_int, mae_full, mae_interact, mae_full_poly, mae_ridge, mae_lasso, mae_pca, mae_xgb, mae_me
rmse <- c(rmse_int, rmse_full, rmse_interact, rmse_full_poly, rmse_ridge, rmse_lasso, rmse_pca, rmse_xgb

# create a data frame to store the results
results <- data.frame(Model=model_names, R_Squared =rsquared, MAE=mae, RMSE =rmse)

results <- rename(results, 'MAE in $' = MAE, 'RMSE in $' = RMSE)
# print the table using the kable function from the knitr package
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 4.1.2
```

```
kable(results, digits = c(2,5,2), align = c("l", "c", "c", "r"), format = 'markdown')
```

| Model | R_Squared | MAE in $ | RMSE in $ |
|:---|:---:|:---:|---:|
| Intuition Model | 0.83440 | 20118.40 | 30130.80 |
| Full Linear Model | 0.84604 | 19062.85 | 29655.47 |
| + Interactions | 0.86448 | 15606.56 | 25889.97 |
| + Polynomials | 0.86588 | 16797.73 | 26921.68 |
| Ridge | 0.87869 | 15229.10 | 23990.65 |
| Lasso | 0.88857 | 14258.55 | 22314.75 |
| PCR | 0.87955 | 16802.50 | 25684.43 |
| XGBoost | 0.88696 | 15083.51 | 23173.32 |
| Weighted Average | 0.89621 | 14134.50 | 21765.00 |

## 12. References

Chen, Tianqi, and Carlos Guestrin. 2016. "XGBoost." In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. https://doi.org/10.1145/2939672.2939785.

De Cock, Dean. 2011. "Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project." *Journal of Statistics Education* 19 (3).