University of Essex

Department of Mathematical Sciences

MA321-SP : Applied Statistics

# MA321 group project assignment

Registration number:

1908015, 1901094, 1900716, 1907611, 1900906

Group : C

Professors: Dr Stella Hadjiantoni, Dr Fanlin Meng

Date of submission (27 March 2020)

**Contents**



Word count: 2490

**Abstract**

An aim of this report is to predict house sale prices by a given dataset. Random Forest and Gradient Boosting Machines model were employed for predicting. Two different validation which are validation set approach and leave-one-out cross-validation were used. Next, These model were compared which one outperform. Moreover, Multinomial Logistic Regression model was fitted to see the significant relationships and strength of impact between overall condition of house and other independent variables. Linear Discriminant Analysis (LDA) was also fitted to reduce dimensionality which dependent variable is overall condition of house. Boruta library was used to find an important features for overall condition of house and house sale prices. Finally, Root Mean Square Error (RMSE) was used to calculate test error of predicting house sale prices of two different model.

## 1 Introduction

When it comes to investments, purchasing a house will be the most important decision in many peoples lives. It is regarded as a safe investment by many and therefore is widely encouraged. There is a vast amount of factors to consider when purchasing a house including: neighbourhood, size, condition of the house, the year it was built in, materials used and etc., however; the most important factor for the vast majority will be the price of the house. In this report, sale price will be examined and predicted using machine learning algorithms, the most important variables will be determined when predicting house pricing.

## 2 Preliminary analysis

### *2.1 Variable analysis*

There are 50 variables in the data set of which 22 are numerical and 28 are factors. Firstly, the numerical variables will be examined.

In *Table 1* it can be observed that the standard deviation between the variables, and the minimum and maximum values vary greatly, therefore normalisation of the data will be required for further analysis. Some variables have a significant amount of missing values, which will have to be dealt with by either imputing these values, or getting rid of either the observations or the variables entirely.

**Table 1:** Summary of the numerical variables in the data set

| Variable | N | Mean | Std. dev. | Min. | 25 % | Median | 75 % | Max. |
|---|---|---|---|---|---|---|---|---|
| LotFrontage | 2,433 | 69.306 | 23.345 | 21.000 | 59.000 | 68.000 | 80.000 | 313.000 |
| LotArea | 2,919 | 10,168.114 | 7,886.996 | 1,300.000 | 7,478.000 | 9,453.000 | 11,570.000 | 215,245.000 |
| OverallQual | 2,919 | 6.089 | 1.410 | 1.000 | 5.000 | 6.000 | 7.000 | 10.000 |
| OverallCond | 2,919 | 5.565 | 1.113 | 1.000 | 5.000 | 5.000 | 6.000 | 9.000 |
| YearBuilt | 2,919 | 1,971.313 | 30.291 | 1,872.000 | 1,953.500 | 1,973.000 | 2,001.000 | 2,010.000 |
| MasVnrArea | 2,896 | 102.201 | 179.334 | 0.000 | 0.000 | 0.000 | 164.000 | 1,600.000 |
| TotalBsmtSF | 2,918 | 1,051.778 | 440.766 | 0.000 | 793.000 | 989.500 | 1,302.000 | 6,110.000 |
| X1stFlrSF | 2,919 | 1,159.582 | 392.362 | 334.000 | 876.000 | 1,082.000 | 1,387.500 | 5,095.000 |
| X2ndFlrSF | 2,919 | 336.484 | 428.701 | 0.000 | 0.000 | 0.000 | 704.000 | 2,065.000 |
| LowQualFinSF | 2,919 | 4.694 | 46.397 | 0.000 | 0.000 | 0.000 | 0.000 | 1,064.000 |
| GrLivArea | 2,919 | 1,500.760 | 506.051 | 334.000 | 1,126.000 | 1,444.000 | 1,743.500 | 5,642.000 |
| FullBath | 2,919 | 1.568 | 0.553 | 0.000 | 1.000 | 2.000 | 2.000 | 4.000 |
| BedroomAbvGr | 2,919 | 2.860 | 0.823 | 0.000 | 2.000 | 3.000 | 3.000 | 8.000 |
| KitchenAbvGr | 2,919 | 1.045 | 0.214 | 0.000 | 1.000 | 1.000 | 1.000 | 3.000 |
| TotRmsAbvGrd | 2,919 | 6.452 | 1.569 | 2.000 | 5.000 | 6.000 | 7.000 | 15.000 |
| Fireplaces | 2,919 | 0.597 | 0.646 | 0.000 | 0.000 | 1.000 | 1.000 | 4.000 |
| GarageArea | 2,918 | 472.875 | 215.395 | 0.000 | 320.000 | 480.000 | 576.000 | 1,488.000 |
| PoolArea | 2,919 | 2.252 | 35.664 | 0.000 | 0.000 | 0.000 | 0.000 | 800.000 |
| MiscVal | 2,919 | 50.826 | 567.402 | 0.000 | 0.000 | 0.000 | 0.000 | 17,000.000 |
| MoSold | 2,919 | 6.213 | 2.715 | 1.000 | 4.000 | 6.000 | 8.000 | 12.000 |
| YrSold | 2,919 | 2,007.793 | 1.315 | 2,006.000 | 2,007.000 | 2,008.000 | 2,009.000 | 2,010.000 |
| SalePrice | 1,460 | 180,921.196 | 79,442.503 | 34,900.000 | 129,975.000 | 163,000.000 | 214,000.000 | 755,000.000 |

For the factor variables, there were variables ranging from 2 to 25 levels with many missing values, a closer inspection will be required to see whether all of these variables are factors or some might be ordinal in nature.

### *2.2 Missing value analysis*

From *Table 1* it can be observed that some variables have many missing values, however upon reading the descriptions of the variables, it is clear that some of these values are not missing although they do need to be pre-processed before analysis.

- *PoolQC* missing values indicate that there is no pool at the household, therefore *'NAs'* will be changed to *'None'*

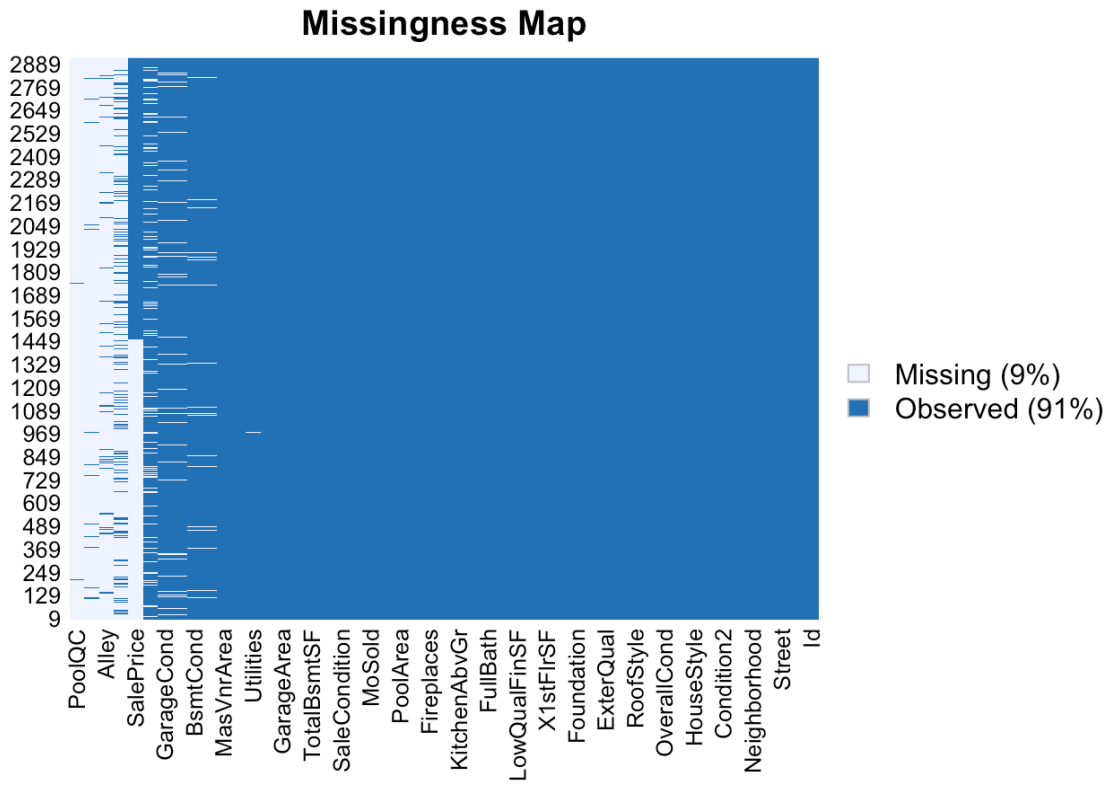


**Figure 1:** Variables plotted against counts of missing values within them

- *Alley* missing values indicate that there is no alley at the household, therefore *'NAs'* will be changed to *'None'*
- Similarly, *MiscFeature*, *Fence*, *GarageCond*, *BsmtCond*, *BsmtQual* missing values were changed to *'None'*
- *GarageType* missing values changed to *'No Garage'*
- *Utilities* missing values changed to the most frequent value *'AllPub'* as there were only a few missing
- *Functional* missing values changed to the most frequent value *'Typ'*
- *Exterior1st* missing values changed to the most frequent value *'Other'*
- *SaleType* missing values changed to the most frequent value *'WD'*
- *KitchenQual* missing values changed to the most frequent value *'TA'*

- *LotFrontage*, *GarageArea*, *TotalBsmtSF*, *MasVnrArea* missing values changed to *0* as these are numerical variables with corresponding factor variables as *'None'*

- *SalePrice* missing values were not changed as these will be predicted later

### *2.3 Data pre-processing*

Firstly, variables *MoSold* and *YrSold* were converted into factor variables as the former ranges from 1 to 12 and the latter only has 5 different years of being sold in, there is no order as a house being sold in the first month or a specific year is not necessarily better or worse than the next month or year.

The following variables: *OverallQual, YearBuilt, ExterQual, ExterCond, BsmtQual, BsmtCond, FullBath, BedroomAbvGr, KitchenAbvGr, KitchenQual, GarageCond, TotRmsAbvGrd, Fireplaces, PoolQC*, have a clear order to them, therefore they are changed to numerical ordinal variables for modelling.

The numerical variables are: *LotFrontage, LotArea, MasVnrArea, TotalBsmtSF, X1stFlrSF, X2ndFlrSF, LowQualFinSF, GrLivArea, GarageArea, PoolArea, MiscVal*, these will be kept as they are for the models.

The ordinal and numerical variables are then scaled and centered with a mean of 0, so all values lie between -1 and 1, this is done because some variables can vary greatly in their values, therefore they need to be normalised so that the increase or decrease in a variable is scaled correctly in the model and give a better indication on how important all the variables are.

The remaining variables are categorical (factors): *Street, Alley, Utilities, LotConfig, Neighborhood, Condition1, Condition2, BldgType, HouseStyle, OverallCond, RoofStyle, RoofMatl, Exterior1st, Foundation, Heating, Functional, GarageType, PavedDrive, Fence, MiscFeature, MoSold, YrSold, SaleType, SaleCondition*, since they are categorical and do not have any order to them, they cannot simply be converted to numerical variables. Dummy variables are made from factors, which means that a categorical variable is split into however many levels there are in a single factor and each observation can have a value of 1 if the observation belongs to this category, and a value of 0 if it does not. This way, the 24 factor variables that were in the dataset are split into 148 in total.

Now these dummy variables can be used in correlation matrices and can be more easily modelled using specific algorithms.

## 3 Analysis

### *3.1 Normally Distributed SalePrice variable*

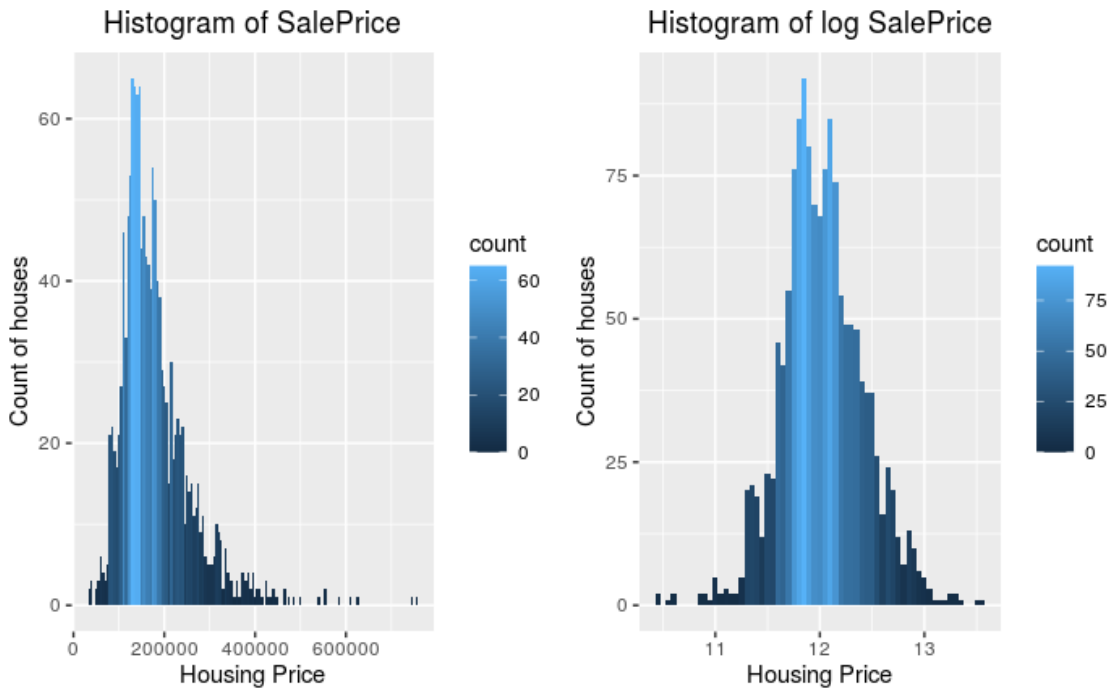


**Figure 2:** Two Histogram plots on SalePrice variable which are raw data and have taken a log on the variable.

Two histogram graphs above, the left of graph shows that the distribution of target variable (Sale Price) is skewed to right. Therefore, take a log on the variable can fix this problem. The histogram graph on the right indicates that sale price variable become a normally distributed and it will be use to fit Random Forest and Stochastic Gradient Boosting models later.

### *3.2 Boruta Feature Selection for house sale price*

#### *3.2.1 Why need to feature selection and the benefits.*

Feature selection is an important step in a predictive model task. It is also called 'Variable Selection'[1]. There are too many predictors for predictive model but not every observation is necessary. Hence, it is essential to investigate which features are benefit for our model. Here are three reasons why feature selection is significant. First, improve the model accuracy by remain

relevant variables and remove some unimportant features. Secondly, to avoid the result overfitting by use too many variables. Finally, variable selection not only can reduce computation time but also cut some hardware requirement such as memory and CPU resource. There are many packages for feature selection in R. In this studying, Boruta method is used on the feature selection.

#### *3.2.2 Reason to use Boruta package*

Basic idea of Boruta algorithm. It is an improvement of Random Forest feature selection and works as a wrapper algorithm around Random Forest. It useful on classification and regression problems. Boruta can handle multi-variable relationships and interactions between variables.

#### *3.2.3 How Boruta Algorithm works*

The steps working of Boruta algorithm is below[1]:

1. Firstly, creating shuffled copies of all features (shadow features) adds randomness to the given dataset.

2. Next, a random forest classifier is trained on the extended data set and use a feature importance measure to identify the importance of each feature.

3. In each iteration, the algorithm marks if the real features have higher importance than the best of shadow feature then remove features which are deemed highly unimportant.

4. Lastly, Boruta method stops when it gets the result of all features are important set or unimportant set.

### ***3.3 Regression Analysis***

#### *3.3.1 Multinomial Logistic Regression*

Multinomial logistic regression is used to fit the logistic regression model since the response variable ***"OverallCond"*** is divided into three categories, namely poor, average and good. The category, poor, is used as the reference type for comparing with average and good categories. In order to reduce the features, ten important variables are chosen based on the previous steps. The result is shown as Table 4 in the appendix.

It is clear that if *"SalePrice"* increases one unit, the logit coefficient for average relative to poor

and good relative to poor will rise by the amount of 2.06 and 4.68, respectively. Likewise, the coefficient of *"OverallQual"* are both positive, which suggest that higher scores are more likely to be average or good than poor. In contrast to *"GarageArea"*, the chances of staying in poor category are higher by 0.0004 compared to average and by 0.002 compared to good category for one unit increase in *"GarageArea."*

As for *"KitchenQual"*, the majority of coefficients are positive, indicating that *Fa* (Fair) and *Gd* (Good) kitchen quality are more likely to have an average or good house condition. While *TA* (Typical/Average) is less likely to be in good house condition than in other categories. Regarding to the types of *"Foundation"*, the coefficients of *CBlock* (Cinder Block), *Stone* and *Wood* are positive, suggesting that the houses used these three materials of foundation are more likely to be in average or good condition. Using *Slab* for foundation is less likely to be average or good when the reference category is poor.

### *3.3.2 Linear Discriminant Analysis*

The method of Linear Discriminant Analysis (LDA) was used to classify the variable *'OverallCond'*. The Boruta algorithm was used to determine Variable Importance and the graph depicting the same can be found in the Appendix. The top 9 important variables were used to form the model of LDA. Following were the prior probabilities of occurrence of each class -

**Table 2:** Prior Probabilities of the classes in OverallCond

| Poor | Average | Good |
|---|---|---|
| 0.02123288 | 0.77397260 | 0.20479452 |

As understood from Table 3, most observations ($\approx$ 77%) were originally classified as having an Average house condition. Good and Poor followed with 20.47% and 2.12%.

The model output also displayed 'Group Means', which had the average value corresponding to each class under each variable considered. These values can be seen in Figure 7 of the Appendix.

Finally, the model also displayed the Coefficients of linear discriminants. These coefficients can be

used to form a linear combination with the values of each variable, the output of which will be a decision boundary. This decision boundary may be used to classify the house condition into the 3 classes. These values can also be found in Figure 7 of the Appendix.

Proportion of Trace shows that LD1 seems to give a more accurate prediction of class values as compared to LD2. These values can also be found in Figure 7 of the Appendix.

### *3.4 The result of SalePrice feature selection in Boruta Algorithm*

There are total 174 variables in our pre-process data set and separate the dataset to train and test datasets and put the train dataset into Boruta algorithm. The result of Boruta method shows that 77 attributes confirmed important:

*Boruta performed 99 iterations in 22.03427 mins. Tentatives roughfixed over the last 99 iterations.**77 attributes confirmed important;** 94 attributes confirmed unimportant.*

### *3.5 Correlation Matrix with SalePrice feature*

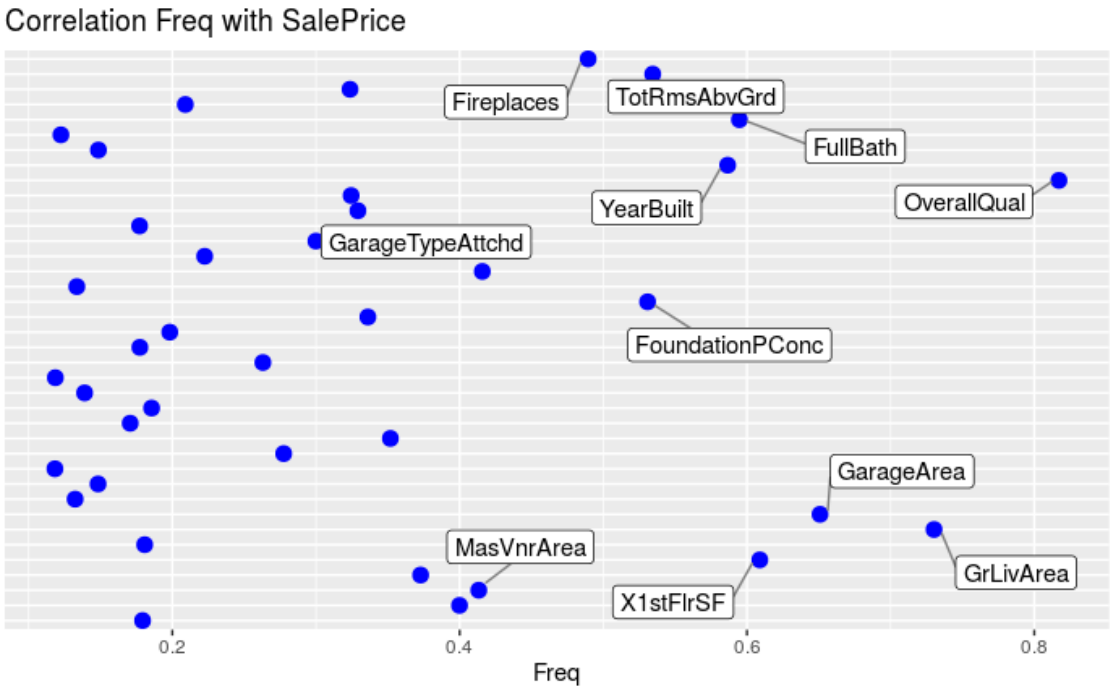


**Figure 3:** plot correlation value of features and labeled the high correlation variables with target variable (sale price)

On the above graph provides that the variables of correlation values which are over 0.1. We can see that 0.4 of correlation value separate the features into two groups on the plot. There are about 27

features under the rate. On the other side, OverallQual feature has a stronger relation with saleprice variable, which is the highest on the plot, at 0.817. In the next stage (training model), we are going to use these 11 features which the correlation value is over 0.4 to train our predictive model and evaluate the prediction error.

The 11 features are

***"X1stFlrSF","GrLivArea","GarageArea","FoundationPConc","OverallQual",YearBuilt",***
***"FullBath","TotRmsAbvGrd","MasVnrArea","GarageTypeAttchd","Fireplaces"***

Finally, we have reduced the total features of in our preprocess dataset from 174 to 77 by using Boruta algorithm, for purpose of diminshing more variables then compute the Correlation Matrix to cut the number of variables to 11. As the reasons mention before it has already showed that why feature selection is significant and necessary since it will improve accuracy and efficiency when sale price prediction.

### *3.6 Predicting house sale prices*

This section is predicting house sale prices, Random Forest and Gradient Boosting Machines Methods were used in order to predict the house prices. Validation set approach and Leave-one-out cross validation were used to calculate the test error with Random Forest and Gradient Boosting Machines on a set of observations. Root Mean Square Error (RMSE) was used to estimate test error and compared to each algorithms and methods. Hence, briefly explain of Root Mean Square Error (RMSE), RMSE is a measure of how spread out between the regression line and actual data point. It is usually used in forecasting, regression analysis. The formula is:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(f-o)^2},$$

*where f = forecasts (expected values or unknown results), o = observed values (known results)*

From using Boruta algorithm and correlation matrix to find feature important to Sale Prices, eleven variables were selected to employ in Random Forest and Gradient Boosting Machines with two different methods of cross-validation. After the experiment, table 3 shows the RMSE results of Random Forest and Gradient Boosting Machines and we can see that Random Forest method can

perform better than Gradient Boosting Machines which RMSE are 0.1396 and 0.1425 respectively. Next, looking at Figure 4 the illustration shows that for the Random Forest with leave-one-out cross-validation method, RMSE is the least of 0.1629 when it used 6 randomly selected predictors from set of predictors available. For Gradient Boosting Machines with leave-one-out cross-validation method, it can be seen that the final values used for the model are n.trees = 300, and the best result of RMSE is 0.16328. After n.trees = 300, RMSE score continues increase.

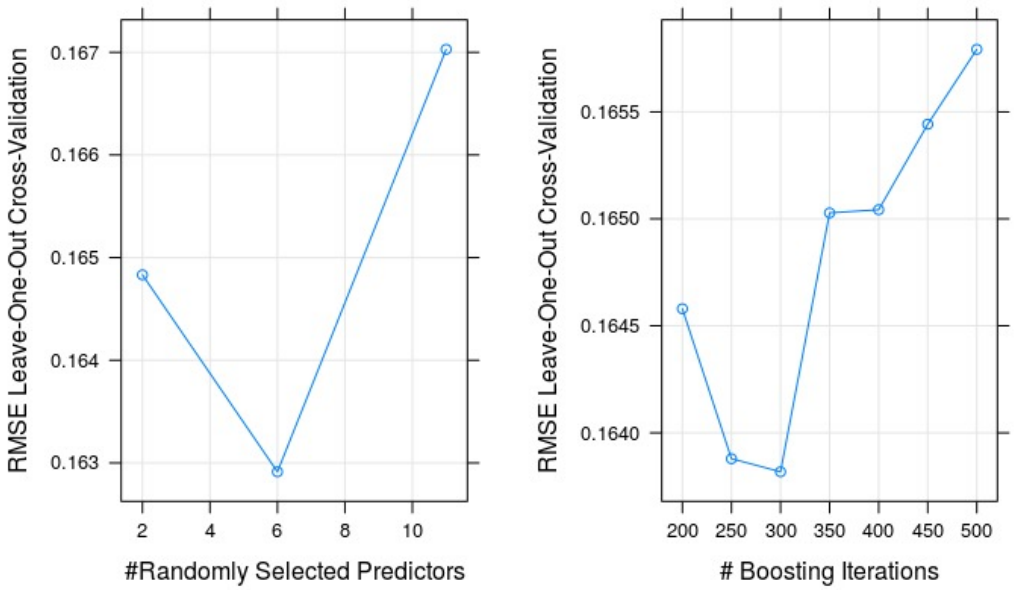


**Figure 4:** RMSE score of Random Forest and Gradient Boosting Machines with Leave-one-out Cross-Validation.

**Table 3:** Evaluating RMSE and R-square of Random Forest and Gradient Boosting Machines by using validation set approach.

| Model | RMSE | R-square |
|---|---|---|
| *Random Forest* | 0.1396 | 0.87813 |
| *Gradient Boosting Machines* | 0.1425 | 0.8681 |

### *3.7 Research Question*

We wanted to know how the different Neighbourhoods performed with respect to Overall Quality and Sale Price. The grouped mean of both *OverallQual* and *SalePrice* was taken for this analy-

sis. The following 10 results were chosen with optimum Sale Price for each Neighborhood -

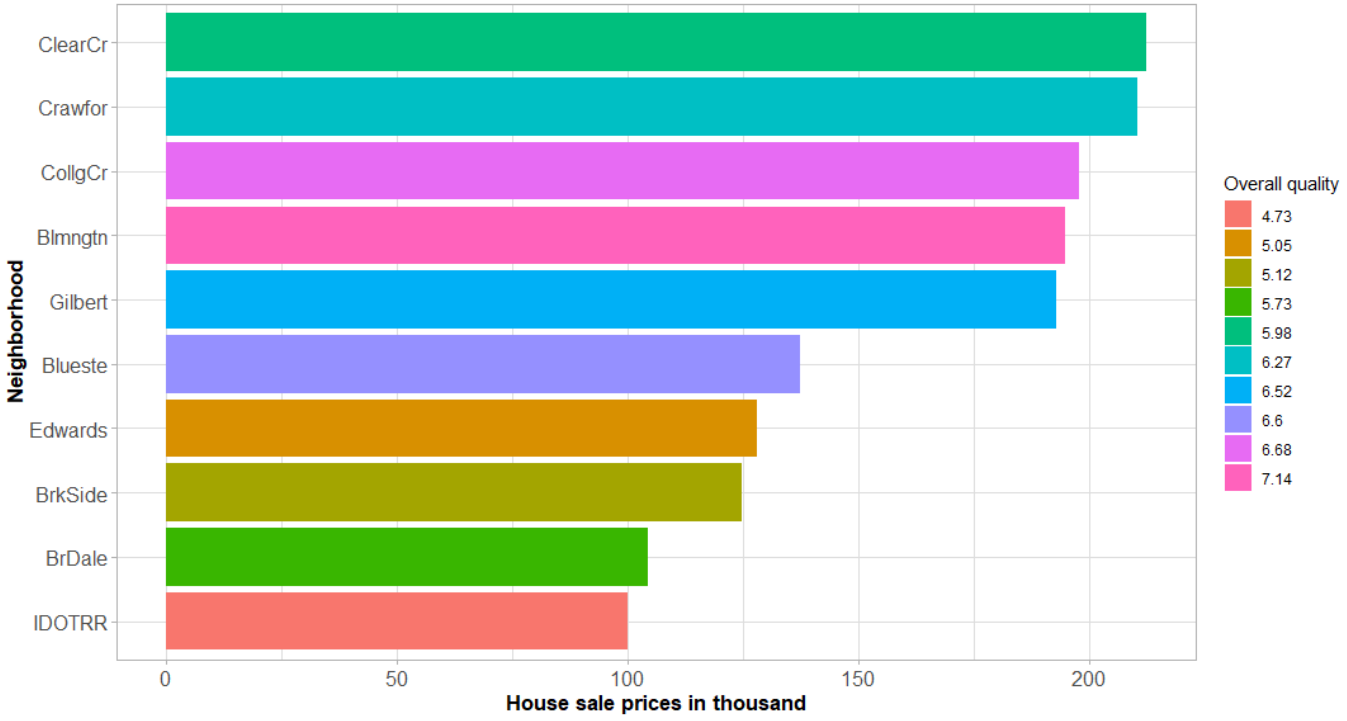


**Figure 5:** Analysis of Neighborhoods

It may be assumed that a customer would want to buy a house which is ranked high in quality and is offered at an affordable price. According to this assumption, the neighborhoods *CollgCr*, *Blmngtn*, *Blueste*, and *Gilbert* seem to be the best choices for buying a house. With Overall Quality between 6 and 8, and sale prices in the range of 130,00 and 200,000, these four neighborhoods can be viewed as having the optimum features for an affordable house with food quality.

## 4 Discussion

From the above analysis of two different algorithms with two different methods of cross-validation, it can be interpreted that for validation set approach, Random Forest can perform better as compared to Gradient Boosting Machines. This is due to its lower RMSE. By using leave-one-out cross validation, Random Forest again outperforms Gradient Boosting Machines. Their RMSEs are 0.1629 and 0.16328 respectively. Since Random Forest has a better accuracy with both cross validation methods, we need only one model with a specific cross validation method. Random Forest with validation set approach has the lowest RMSE and so it can be used to predict house sale

prices using the test dataset. Moreover, figure 6 shows **line of best fit** of predicted and actual sale prices in terms of logarithmic values by employing Random Forest model with validation set approach. We can say that predicted sale prices can explain variance of actual sale prices by 87.81% which can be seen in table. 3.

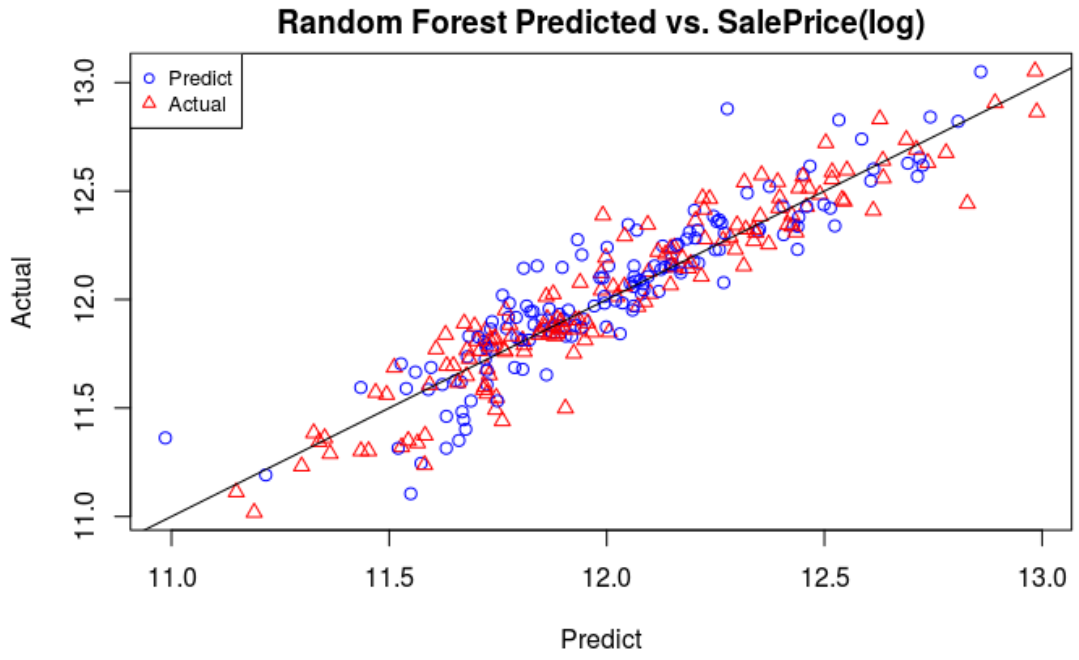


**Figure 6:** Illustration of Predicted and Actual Sale Price in log by Random Forest method with validation set approach

## 5 Conclusion

Buying a house involves going through multiple features, which makes the task difficult. However, advanced machine learning algorithms and predictive statistics can be used to ease the process and find the best options as per one's choices. The algorithms can also be used to classify the houses based on their overall condition. One can also predict the sale price of any new listing added to this dataset, based on the already present data. The dataset may have some preexisting issues such as missing values, irrelevant features, ambiguous entries and others. All these issues must be tackled in the beginning so that the important analysis can be done on a cleaned dataset.

**Contributions:**

- **1907611:** *Question 1, preliminary analysis*
- **1900716, 1900906:** *Question 2. Report 3.3*
- **1908015, 1901094:** *Question 3 code. Report 3.1,3.2, 3.4-3.6 and 4*
- **All members:** *Question 4*

## 6 Appendix

**Table 4:** Regression Results

| | *Dependent variable:* | | | | |
|---|---|---|---|---|---|
| | Average | Good | | Average | Good |
| | (1) | (2) | | (1) | (2) |
| YearBuilt | 0.002*** | −0.041*** | FoundationWood | 10.547*** | 13.425*** |
| | (0.0003) | (0.0003) | | (0.004) | (0.004) |
| KitchenQualFa | 0.859*** | 0.224** | BsmtQualFa | −0.368*** | 2.453*** |
| | (0.111) | (0.096) | | (0.050) | (0.046) |
| KitchenQualGd | 2.372*** | 2.443*** | BsmtQualGd | 1.662*** | 3.819*** |
| | (0.097) | (0.096) | | (0.074) | (0.067) |
| KitchenQualTA | 0.982*** | −0.228*** | BsmtQualNone | 13.452*** | 16.152*** |
| | (0.091) | (0.079) | | (0.054) | (0.035) |
| SalePrice | 2.063*** | 4.680*** | BsmtQualTA | 1.331*** | 3.384*** |
| | (0.025) | (0.021) | | (0.076) | (0.067) |
| ExterCondFa | 16.951*** | −13.527*** | GarageArea | −0.0004 | −0.002** |
| | (0.032) | (0.018) | | (0.001) | (0.001) |
| ExterCondGd | 18.947*** | −8.607*** | ExterQualFa | −10.266*** | −12.419*** |
| | (0.054) | (0.054) | | (0.026) | (0.014) |
| ExterCondPo | −9.831*** | −33.414*** | ExterQualGd | −10.916*** | −12.411*** |
| | (0.000) | (0.000) | | (0.069) | (0.068) |
| ExterCondTA | 18.483*** | −10.799*** | ExterQualTA | −10.555*** | −11.984*** |
| | (0.060) | (0.055) | | (0.075) | (0.069) |
| FoundationCBlock | 0.188** | 0.763*** | FullBath | −0.263*** | −1.321*** |
| | (0.093) | (0.083) | | (0.090) | (0.085) |
| FoundationPConc | 0.320*** | −0.373*** | OverallQual | 0.341*** | 0.189*** |
| | (0.117) | (0.105) | | (0.054) | (0.060) |
| FoundationSlab | −12.609*** | −12.746*** | | | |
| | (0.041) | (0.023) | | | |
| FoundationStone | 12.270*** | 13.999*** | Constant | −35.641*** | 47.683*** |
| | (0.006) | (0.006) | | (0.003) | (0.002) |
| | | | Akaike Inf. Crit. | 1,270.912 | 1,270.912 |

*Note:* *p<0.1; **p<0.05; ***p<0.01

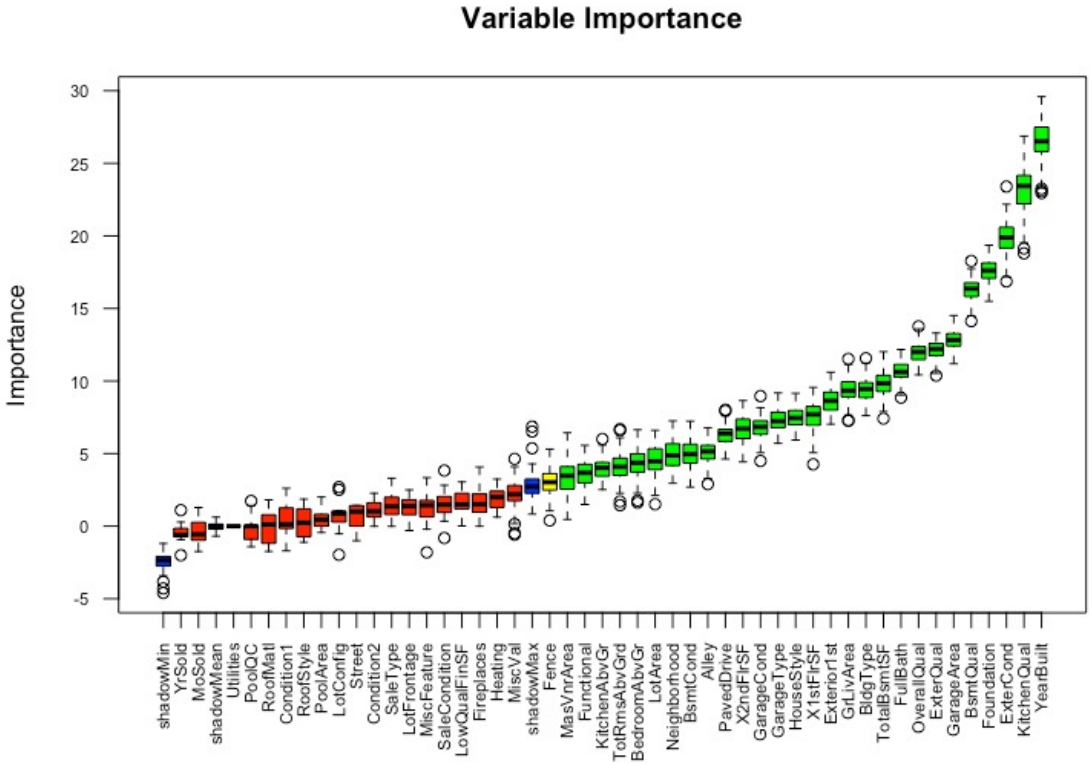


**Figure 7:** Boruta Feature Selection for ”OverallCond

```
Prior probabilities of groups:
      Poor    Average       Good
0.02123288 0.77397260 0.20479452

Group means:
        YearBuilt KitchenQualFa KitchenQualGd KitchenQualTA SalePrice ExterCondFa
Poor     1945.645    0.19354839    0.03225806     0.7419355  11.44990  0.25806452
Average  1978.457    0.01769912    0.41858407     0.4884956  12.06233  0.01504425
Good     1946.756    0.04347826    0.37458194     0.5351171  11.93890  0.01003344
        ExterCondGd ExterCondPo ExterCondTA FoundationCBlock FoundationPConc
Poor     0.03225806  0.03225806   0.6774194        0.4516129       0.1290323
Average  0.05132743  0.00000000   0.9336283        0.3884956       0.5300885
Good     0.29096990  0.00000000   0.6889632        0.6053512       0.1471572
        FoundationSlab FoundationStone FoundationWood BsmtQualFa BsmtQualGd
Poor       0.129032258    0.0000000000    0.000000000 0.16129032 0.09677419
Average    0.015929204    0.0008849558    0.001769912 0.01150442 0.48495575
Good       0.006688963    0.0167224080    0.003344482 0.05685619 0.22408027
        BsmtQualNone BsmtQualTA GarageArea ExterQualFa ExterQualGd ExterQualTA
Poor      0.12903226  0.5806452   310.8710 0.129032258  0.06451613   0.8064516
Average   0.02477876  0.3743363   499.0035 0.007079646  0.38495575   0.5672566
Good      0.01672241  0.6956522   391.4381 0.006688963  0.17056856   0.8026756
        OverallQual FullBath
Poor       4.419355 1.225806
Average    6.260177 1.637168
Good       5.665552 1.327759

Coefficients of linear discriminants:
                          LD1           LD2
YearBuilt        -0.030197278  0.0021793503
KitchenQualFa    -0.575211573 -0.2645830042
KitchenQualGd     0.074079693 -0.4627471150
KitchenQualTA    -0.990438780 -0.4341697481
SalePrice         1.502375893 -1.2323655799
ExterCondFa      -3.527746040  3.5782672556
ExterCondGd      -1.087877209 -0.2546260842
ExterCondPo      -4.009899379 17.7057355298
ExterCondTA      -2.666978547 -0.1146435157
FoundationCBlock  0.300561486 -0.3945660061
FoundationPConc  -0.341159074 -0.4496749545
FoundationSlab    0.136331631  1.9280226590
FoundationStone   1.471430393 -1.4214359438
FoundationWood    1.888550557 -0.4645569726
BsmtQualFa        0.785050454  1.1254218836
BsmtQualGd        0.335877056 -0.4498313669
BsmtQualNone      0.392057722 -1.3410273173
BsmtQualTA        0.183861331 -0.4682107467
GarageArea       -0.001284127  0.0004383141
ExterQualFa      -0.506545689  1.8896728154
ExterQualGd      -0.249109836 -0.0019393016
ExterQualTA      -0.128953793 -0.3328779787
OverallQual      -0.070259879 -0.1592315669
FullBath         -0.579306431  0.1329910121

Proportion of trace:
  LD1   LD2
0.725 0.275
```

**Figure 8:** Linear discriminant analysis results

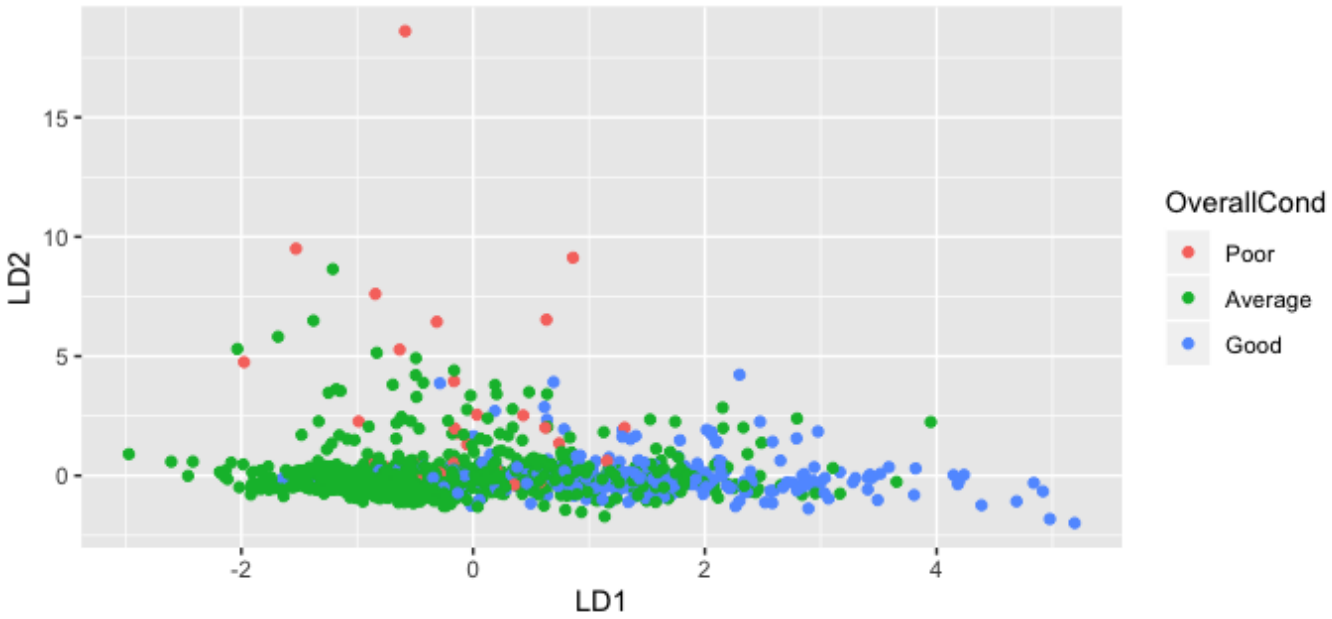


**Figure 9:** Classify the house condition

```
##########################
####### MA 321 #########
##### Group Project ####

#loading libraries
library(knitr)
library(ggplot2)
library(plyr)
library(dplyr)
library(corrplot)
library(caret)
library(gridExtra)
library(scales)
library(Rmisc)
library(ggrepel)
```

```
library(psych)
library(Boruta)
library(dplyr)
library(tidyr)
library(mice)
library(nnet)
library(MASS)
library(tidyverse)
library(reshape2)
library(car)
library(ggrepel)
library(psych)
library(devtools)
library(ggbiplot)
library(mlbench)
library(tidyr)
library(mice)
library(corrplot)
library(randomForest)
library(xgboost)
library(tidyverse)
library(wordcloud)
library(visNetwork)
library(networkD3)
library(magrittr)
library(data.table)
library(lattice)
```

```r
library(gbm)

####### Question 1 #########
#### Missing data
df_house = read.csv('house_data.csv')

df_house = subset(df_house, select = -c(Id))
df_house = subset(df_house, select = -c(SalePrice))
#treat missing value (character) have meaning from description of
  ↪ data
#ref:https://stackoverflow.com/questions/34718771/r-change-na-values
#PoolQC
df_house$PoolQC <- as.character(df_house$PoolQC)
df_house$PoolQC[is.na(df_house$PoolQC)] <- "None"
df_house$PoolQC <- as.factor(df_house$PoolQC)
#Misfeature
df_house$MiscFeature <- as.character(df_house$MiscFeature)
df_house$MiscFeature[is.na(df_house$MiscFeature)] <- "None"
df_house$MiscFeature <- as.factor(df_house$MiscFeature)
#Alley
df_house$Alley <- as.character(df_house$Alley)
df_house$Alley[is.na(df_house$Alley)] <- "None"
df_house$Alley <- as.factor(df_house$Alley)
#Fence
df_house$Fence <- as.character(df_house$Fence)
df_house$Fence[is.na(df_house$Fence)] <- "None"
df_house$Fence <- as.factor(df_house$Fence)
```

```
#GarageCond
df_house$GarageCond <- as.character(df_house$GarageCond)
df_house$GarageCond[is.na(df_house$GarageCond)] <- "None"
df_house$GarageCond <- as.factor(df_house$GarageCond)
#GarageType
df_house$GarageType <- as.character(df_house$GarageType)
df_house$GarageType[is.na(df_house$GarageType)] <- "No Garage"
df_house$GarageType <- as.factor(df_house$GarageType)
#BsmtCond
df_house$BsmtCond <- as.character(df_house$BsmtCond)
df_house$BsmtCond [is.na(df_house$BsmtCond)] <- "None"
df_house$BsmtCond <- as.factor(df_house$BsmtCond)
#BsmtQual
df_house$BsmtQual <- as.character(df_house$BsmtQual)
df_house$BsmtQual [is.na(df_house$BsmtQual)] <- "None"
df_house$BsmtQual <- as.factor(df_house$BsmtQual)
##MoSold
df_house$MoSold = as.factor(df_house$MoSold)
##YrSold
df_house$YrSold = as.factor(df_house$YrSold)
# no meaning
#Utilities
df_house$Utilities  <- as.character(df_house$Utilities)
df_house$Utilities  [is.na(df_house$Utilities)] <- "AllPub"
df_house$Utilities  <- as.factor(df_house$Utilities)
#Functional
df_house$Functional  <- as.character(df_house$Functional)
```

```
df_house$Functional  [is.na(df_house$Functional)] <- "Typ"
df_house$Functional  <- as.factor(df_house$Functional)
#Exterior1st
df_house$Exterior1st  <- as.character(df_house$Exterior1st)
df_house$Exterior1st  [is.na(df_house$Exterior1st)] <- "Other"
df_house$Exterior1st  <- as.factor(df_house$Exterior1st)
#SaleType
df_house$SaleType  <- as.character(df_house$SaleType)
df_house$SaleType  [is.na(df_house$SaleType)] <- "WD"
df_house$SaleType  <- as.factor(df_house$SaleType)
#KitchenQual
df_house$KitchenQual  <- as.character(df_house$KitchenQual)
df_house$KitchenQual  [is.na(df_house$KitchenQual)] <- "TA"
df_house$KitchenQual  <- as.factor(df_house$KitchenQual)
df_house$OverallCond = cut(df_house$OverallCond,breaks =
  ↪  c(0,3,6,10), labels = c('Poor', 'Average', 'Good'))
#replace by mean (numeric)
#ref:https://subscription.packtpub.com/book/big_data_and_business_intelliger
#ref:https://statisticsglobe.com/r-replace-na-with-0/
#LotFrontage
#df_houseLotFrontage <- ifelse(is.na(df_houseLotFrontage),
  ↪  mean(df_houseLotFrontage, na.rm = TRUE), df_houseLotFrontage)
df_house$LotFrontage[is.na(df_house$LotFrontage)] <- 0
#GarageArea
df_house$GarageArea[is.na(df_house$GarageArea)] <- 0
#TotalBsmtSF
df_house$TotalBsmtSF[is.na(df_house$TotalBsmtSF)] <- 0
```

```
#MasVnrArea (character)
df_house$MasVnrArea[is.na(df_house$MasVnrArea)] <- 0
numericVars <- which(sapply(df_house, is.numeric)) #index vector
    ↪ numeric variables
numericVarNames <- names(numericVars) #saving names vector for
    ↪ use later on
numericVarNames <- numericVarNames[!(numericVarNames %in%
    ↪ c('YearBuilt', 'OverallQual', 'FullBath','BedroomAbvGr',
    ↪ 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces'))]
DFnumeric <- df_house[, names(df_house) %in% numericVarNames]
for(i in 1:ncol(DFnumeric)){if
    ↪ (abs(skew(DFnumeric[,i]))>0.8){DFnumeric[,i] <-
    ↪ log(DFnumeric[,i] +1)}}
PreNum <- preProcess(DFnumeric, method=c("center", "scale"))
DFnorm <- predict(PreNum, DFnumeric)
DFfactors <- df_house[, !(names(df_house) %in% numericVarNames)]
DFordinal = DFfactors[,c('OverallQual',
    ↪ 'YearBuilt','ExterQual','ExterCond', 'BsmtQual',
    ↪ 'BsmtCond','FullBath', 'BedroomAbvGr',
    ↪ 'KitchenAbvGr',"KitchenQual","GarageCond", 'TotRmsAbvGrd',
    ↪ 'Fireplaces',"PoolQC")]
for (i in 1:length(DFordinal)) {DFordinal[,i] =
    ↪ as.numeric(DFordinal[,i])}
DFfactors <- DFfactors[, !(names(DFfactors) %in%
    ↪ names(DFordinal))]
df_house = subset(df_house, select = -c(OverallCond))
DFdummies <- as.data.frame(model.matrix(~.-1, DFfactors))
```

```r
dim(DFdummies)

combined <- cbind(DFnorm, DFordinal)
#write.csv(combined, file = "pre_processed_data.csv",
  ↪ row.names=FALSE)
combined = cbind(combined, DFdummies)

#take log with Saleprice
df_house$SalePrice <- log(df_house$SalePrice)

####### Question 2 #########
# Select 10 important variables #
# Fit Logistic regression#
# Linear discriminant analysis #
############################
#separate train/test and unseen data
df_house_tt = df_house[1:1460,]
df_house_unseen = df_house[-(1:1460),]

#feature important
boruta_output <- Boruta(OverallCond ~ ., data=df_house_tt,
  ↪ doTrace=2)
boruta_signif <-
  ↪ names(boruta_output$finalDecision[boruta_output$finalDecision
  ↪ %in% c("Confirmed")])
plot(boruta_output, cex.axis=.7, las=2, xlab="", main="Variable
  ↪ Importance")
```

```r

#Logistic Regression
#ref:https://stackoverflow.com/questions/28551633/error-in-r-mice-package-to
test <- multinom(OverallCond ~ YearBuilt + KitchenQual +
  ↪ SalePrice + ExterCond + Foundation + BsmtQual + GarageArea +
  ↪ ExterQual + FullBath + OverallQual, data =
  ↪ df_house_tt,MaxNWts = 1515)
summary(test)

#Linear discriminant analysis
#ref:http://www.sthda.com/english/articles/36-classification-methods-essenti
boruta_lda <- Boruta(OverallCond ~ ., data = df_house_tt,
  ↪ doTrace=2)
plot(boruta_output, cex.axis=.7, las=2, xlab="", main="Variable
  ↪ Importance")
model <- lda(OverallCond ~ YearBuilt + KitchenQual + SalePrice +
  ↪ ExterCond + Foundation + BsmtQual + GarageArea + ExterQual +
  ↪ OverallQual + FullBath , data = df_house_tt)
model
plot(model)

####### Question 3 #########
# RandomFoest and Boosting predict #
# Validation set approach and #####
# Leave one out cross validation #
#########################

```

```
####code reference
#ref:https://rstudio-pubs-static.s3.amazonaws.com/252976_5361c17c408b491198
#ref:https://www.kaggle.com/c/house-prices-advanced-regression-techniques/ke
####

#loading libraries
library(tidyverse)
library(knitr)
library(scales)
library(wordcloud)
library(visNetwork)
library(networkD3)
library(knitr)
library(magrittr)
library(dplyr)
library(tidyverse)
library(ggplot2)
library(randomForest)
library(data.table)
library(caret)
library(lattice)

#load the dataset house_data and pre_process_data
df_house = read.csv("house_data.csv")
df_pre = read.csv("pre_processed_data.csv")
#inset Saleprice variable in preprocess dataset
df_pre$SalePrice = df_house$SalePrice
```

```
205
206 #Histogram saleprice graph
207 options(scipen=10000)
208 plot1 = ggplot(df_pre, aes(x = SalePrice, fill = ..count..)) +
209 geom_histogram(binwidth = 5000) +
210 ggtitle("Histogram of SalePrice") +
211 ylab("Count of houses") +
212 xlab("Housing Price") +
213 theme(plot.title = element_text(hjust = 0.5))
214
215 #log term of SalePrice
216 df_pre$SalePrice <- log(df_pre$SalePrice)
217
218 #### code reference
219 ##ref:https://www.kaggle.com/shaoyingzhang/data-exploration-and-prediction-c
220 ###
221 #plot a saleprice Histogram graph which take a log
222 plot2 = ggplot(df_pre, aes(x = SalePrice, fill = ..count..)) +
223 geom_histogram(binwidth = 0.05) +
224 ggtitle("Histogram of log SalePrice") +
225 ylab("Count of houses") +
226 xlab("Housing Price") +
227 theme(plot.title = element_text(hjust = 0.5))
228
229 #plot two Histogram saleprice
230 require(gridExtra)
231 grid.arrange(plot1,plot2 ,ncol=2)
```

```
232
233 #splite dataset to train and test.
234 #if the SalePrice column values is empyt then put it to test
    ↪ dataset.
235 df_train <- df_pre[!is.na(df_pre$SalePrice), ]
236 test <- df_pre[is.na(df_pre$SalePrice),]
237
238 # feature selection (Boruta) and plot the barchart
239 library(ranger)
240 library(Boruta)
241 #Feature select for SalePrice
242 # Decide if a variable is important or not using Boruta
243 boruta_output <- Boruta(SalePrice ~ ., data=df_train,
    ↪ doTrace=2,maxRuns = 100)
244 boruta_signif <-
    ↪ names(boruta_output$finalDecision[boruta_output$finalDecision
    ↪ %in% c("Confirmed", "Tentative")])
245 print(boruta_signif)
246 plot(boruta_output, cex.axis=.6, las=2, xlab="",
    ↪ main="SalePrice(Log) Variable Importance")
247
248 ### code reference
249 #ref:https://www.analyticsvidhya.com/blog/2016/03/select-important-variables
250 ###
251 #Show Boruta Feature selection result (important and unimportand)
252 boruta.df <- attStats(final.boruta)
253 class(boruta.df)
```

```
print(boruta.df)
final.boruta <- TentativeRoughFix(boruta_output)
print(final.boruta)

# list important features
getSelectedAttributes(final.boruta, withTentative = F)

# remain these 77 variables and SalePrice, drop the columns which
  ↪ are not important
df_boruta <-df_train[
  ↪ ,c("LotFrontage","LotArea","MasVnrArea","TotalBsmtSF",
"X1stFlrSF","X2ndFlrSF","GrLivArea","GarageArea","AlleyNone",
"LotConfigCulDSac","NeighborhoodBrDale","NeighborhoodBrkSide",
"NeighborhoodClearCr","NeighborhoodCollgCr",
"NeighborhoodCrawfor","NeighborhoodEdwards","NeighborhoodGilbert",
"NeighborhoodIDOTRR","NeighborhoodMeadowV","NeighborhoodNAmes",
"NeighborhoodNoRidge","NeighborhoodNPkVill","NeighborhoodNridgHt",
"NeighborhoodNWAmes","NeighborhoodOldTown","NeighborhoodSawyer",
"NeighborhoodSomerst","NeighborhoodStoneBr","NeighborhoodTimber",
"Condition1Feedr","Condition1Norm","BldgTypeDuplex","BldgTypeTwnhs",
"BldgTypeTwnhsE","HouseStyle1Story","HouseStyle2Story",
"HouseStyleSFoyer","HouseStyleSLvl","OverallCondAverage",
"OverallCondGood","RoofStyleGable","RoofStyleHip","Exterior1stBrkFace",
"Exterior1stCemntBd","Exterior1stHdBoard","Exterior1stMetalSd",
"Exterior1stPlywood","Exterior1stVinylSd","Exterior1stWd.Sdng",
"FoundationCBlock","FoundationPConc","FoundationSlab",
"HeatingWall","FunctionalTyp","GarageTypeAttchd","GarageTypeBuiltIn",
```

```
"GarageTypeDetchd","GarageTypeNo.Garage","PavedDriveY","FenceNone",
"SaleTypeNew","SaleTypeWD","SaleConditionNormal","SaleConditionPartial",
"OverallQual","YearBuilt","ExterQual","ExterCond","BsmtQual","BsmtCond",
"FullBath","BedroomAbvGr","KitchenAbvGr","KitchenQual","GarageCond",
"TotRmsAbvGrd","Fireplaces","SalePrice")]


### code reference
##ref:https://stackoverflow.com/questions/7074246/show-correlations-as-an-o
#ref:https://www.statmethods.net/management/subset.html
###
## Plot a correlation matrix
z = cor(df_boruta)
zdf <- as.data.frame(as.table(z))
# show the correlation matrix value with SalePrice and the
  ↪  correlation value
# freq > 0.1 has 38 variables,
# freq > 0.2 has 23 variables
# freq > 0.3 has 18 variables
# freq > 0.4 has 11 variables
# freq > 0.5 has 8  variables
# freq > 0.6 has 4  variables

## list the correlation values with Saleprice over 0.2.
df_correlation <- subset(zdf,Var2 =="SalePrice" & Freq >0.1)

#delete "Saleprice" in Var1
```

```r
df_correlation  <- df_correlation [df_correlation$Var1 != "SalePrice", ]
df_correlation

# plot  and label the freq varables are over 0.4
nbaplot <- ggplot(df_correlation,yaxt = 'n', aes(x= df_correlation$Freq, y = df_correlation$Var1)) +
geom_point(color = "blue", size = 3) +  theme(axis.title.y=element_blank(),
axis.text.y=element_blank(),axis.ticks.y=element_blank()) +
labs(title="Correlation Freq with SalePrice",x ="Freq")


#### code reference
#ref:https://stackoverflow.com/questions/15624656/label-points-in-geom-point
#ref:https://stackoverflow.com/questions/35090883/remove-all-of-x-axis-label  remove y axis in ggplot
####
#remove y axis in ggplot
library(ggplot2)
library(ggrepel)
### geom_label_repel
nbaplot +  geom_label_repel(aes(label=ifelse(df_correlation$Freq > 0.4,as.character(df_correlation$Var1),'')),box.padding  = 0.35, point.padding = 0.5,segment.color = 'grey50')

```

```r
# final these 11 features to use on RF and Boosting for predicting Saleprice:
# X1stFlrSF + GrLivArea + GarageArea + FoundationPConc + OverallQual +YearBuilt+ FullBath + TotRmsAbvGrd +MasVnrArea + GarageTypeAttchd + Fireplaces

# validation set approach
library(caTools)
set.seed(101)
sample = sample.split(df_boruta, SplitRatio = .80)
s_train = subset(df_boruta, sample == TRUE) # 75% from train data
s_test  = subset(df_boruta, sample == FALSE) #25% from train data


#########
#RandomForest
#ref:#https://rstudio-pubs-static.s3.amazonaws.com/252976_5361c17c408b491198
#########

set.seed(222)
model.rf <- randomForest(SalePrice ~ X1stFlrSF + GrLivArea + GarageArea + FoundationPConc + OverallQual +YearBuilt+ FullBath +TotRmsAbvGrd + MasVnrArea + GarageTypeAttchd + Fireplaces,data = s_train, CV=TRUE)
print(model.rf)
plot(model.rf)

```

```r
#vaildation set approach (s_test) to compute RMSE
preds.rf <- predict(model.rf, s_test)
rmse.rf <- RMSE(preds.rf,s_test$SalePrice)
rmse.rf

#########
#Gradient Boosting
#########
set.seed(123)
model.gbm <- gbm(SalePrice ~ X1stFlrSF + GrLivArea + GarageArea +
    FoundationPConc + OverallQual +YearBuilt+ FullBath
    +TotRmsAbvGrd +MasVnrArea + GarageTypeAttchd +
    Fireplaces,data =s_train,distribution = "gaussian",  # SSE
    loss functionn.trees = 5000,shrinkage = 0.1,interaction.depth
    = 3,n.minobsinnode = 10,cv.folds = 10)

# find index for number trees with minimum CV error
best <- which.min(model.gbm$cv.error)

# get MSE and compute RMSE
rmse.gbm <- sqrt(model.gbm$cv.error[best])
gbm.perf(model.gbm, method = "cv")

###############
# Randonforest with LOOCV
###############
library(tidyverse)
```

```
368 library(caret)
369 set.seed(122)
370 LOOCV <- trainControl(method = "LOOCV")
371
372 # Randomforest for Loocv
373 model.rf.loocv <- train(SalePrice ~ X1stFlrSF + GrLivArea +
    ↪  GarageArea + FoundationPConc + OverallQual +YearBuilt+
    ↪  FullBath +TotRmsAbvGrd +        MasVnrArea + GarageTypeAttchd
    ↪  + Fireplaces, data = s_train, method = "rf",trControl =
    ↪  LOOCV,
374 nodesize= 10,  # 10 data-points/node. Speeds modeling
375 ntree =500,    # Default 500. Reduced to speed up modeling
376 )
377
378 # Summarize the RF LOOCV results
379 print(model.rf.loocv)
380 preds.rf.loocv <- predict(model.rf.loocv, s_test)
381 plot(model.rf.loocv)
382
383 #compute RMSE from s_test dataset Saleprice
384 rmse.rf.loocv <- RMSE(preds.rf.loocv ,s_test$SalePrice)
385 rmse.rf.loocv
386
387 ##################
388 # Stochastic Gradient Boosting  with LOOCV
389 #ref:https://www.kaggle.com/aniruddhachakraborty/lasso-gbm-xgboost-top-20-0
390 #ref:https://www.kaggle.com/xdurana/gbm-simple-model
```

```
391 ##################
392 library(iterators)
393 library(parallel)
394 set.seed(222)
395
396 #train the gbm model.
397 LOOCV <- trainControl(method = "LOOCV")
398 model.gbm.loocv <- train(SalePrice ~ X1stFlrSF + GrLivArea +
    ↪  GarageArea + FoundationPConc + OverallQual +YearBuilt+
    ↪  FullBath +TotRmsAbvGrd +        MasVnrArea + GarageTypeAttchd
    ↪  + Fireplaces,data = s_train, method = "gbm",metric="RMSE",
    ↪  tuneGrid = expand.grid(n.trees = 500, interaction.depth =
    ↪  5,shrinkage = 0,05, n.minobsinnode = 10) ,trControl = LOOCV)
399
400 # Summarize the GBM LOOCV results
401 print(model.gbm.loocv)
402 preds.gbm.loocv <- predict(model.gbm.loocv, s_test)
403 plot(model.gbm.loocv)
404
405 #compute RMSE from s_test dataset Saleprice
406 rmse.gbm.loocv <- RMSE(preds.gbm.loocv ,s_test$SalePrice)
407 rmse.gbm.loocv
408
409 #Plot RMSE Loocv with Randomforest and boosting
410 p1 <- plot(model.rf.loocv)
411 p2 <- plot(model.gbm.loocv)
412 grid.arrange(p1, p2, nrow = 1)
```

```
413
414 # show RMSE and R2 between RandomForest and GBM
415 r2.rf <- R2(preds.rf, s_test$SalePrice)
416 r2.gbm <- R2(preds.gbm.loocv,s_test$SalePrice)
417 paste0("RandomForest RMSE:",rmse.rf.loocv)
418 paste0("RandomForest R2:",r2.rf)
419 paste0("GBM RMSE:",rmse.gbm.loocv)
420 paste0("GBM R2:",r2.gbm)
421
422 #Because the RandomForest RMSE is better than GBM's therefore, we
    ↪  used the Randomforest to fill SalePrice column in test
    ↪  dataset and output the csv file
423 #predic the saleprice and fill in test dataset
424 pred.rf <- predict(model.rf.loocv,test)
425 #exp the predicting saleprice as it took a log before.
426 test$SalePrice <- exp(pred.rf)
427
428 #Output a csv file
429 write.csv(test, "predict SalePrice.csv")
430
431
432 #Plot predicted (RandomForest) and actual saleprice scatter plot.
433 plot(preds.rf.loocv, s_test$SalePrice, main = "Random Forest
    ↪  Predicted vs.
434 SalePrice(log)",ylab = "Actual",xlab =
    ↪  "Predict",pch=c(1,2),col=c("blue","red"))
```

```
435 abline(0,1)legend("topleft", legend=c("Predict",
    ↪  "Actual"),col=c("blue", "red"), pch=c(1,2), cex=0.8)
```

## References

[1] D Dutta. How to perform feature selection (ie pick important variables) using boruta package in r, 2016.