House Price Prediction using Machine Learning

MA321 Group Coursework

Department of Mathematical Sciences, University of Essex,

CO4 3SQ Colchester, United Kingdom

Group 11

Smruti Das	- sd20396
Jhansi Rani Choutapalem	- jc20168
Venkata Naga Sai Pooja Kommasani	- vk20242
Mamatha Sai Yarabarla	- my20474
Raja Sumanth Dulam	- rd20618
Kotla Madhav Srinivas	-mk20955

Abstract:

Housing and real estate play a major role in business sector and economy of the world. Some realtors provided housing market data to predict housing prices in Ames, we will perform appropriate statistical analysis of housing data and develop machine learning models based on classification and regression. R is the language of choice for statistical analysis. All models in this project are implemented using the R language. The housing data provided will be used in a way that uses several R statistical methods to generate an accurate forecast of the selling price of a home.

Keywords: House price prediction, Sale Price, R programming, Classification, Regression.

Contents

Word Count:2589

1. Introduction:

The market prices of the houses have a crucial replication on the economy, All the sellers and buyers have a great interest in house prices. In this project the prediction of house prices taken place by considering all the explanatory variables that play an important role in setting a price for house.

The dataset used in this project consists of data of houses from the city Ames. Some of the features of the data are

- Neighborhood,
- Utilities,
- Housing Style,
- OverallQual – Rates the Overall quality of the materials,
- OverallCond – Rages the overall condition of the house,
- Year built – Original construction date, etc.

Continuous home prices are predicted using various regression techniques such as lasso, ridge, SVM regression, and random forest regression. Classification models used for predicting the overall condition of the houses. The goal of this project is to create regression models that are ready to accurately estimate the price of a featured house.

1.1 Numerical and Graphical Summaries:

Summarizing the data will help us to understand the data distribution and exploratory data analysis helps you analyze data features and explore all your data insights. EDA plays an important role in building predictive models because it helps to understand the characteristics of the data.

1.2 Imputation of Missing Data:

In machine learning projects, missing values can affect model predictions, so you need to impute them. Missing value imputations are useful for statistical analysis of data to build machine learning models. In this project all the missing data is imputed using appropriate imputation techniques.

1.3 Classification for Over All condition of the house:

We use classification techniques of machine learning to classify Overall condition of the house.

1.4 Regression for Sale Price Prediction:

Building machine learning models for predicting the sale price of the house.

1.5 Research Question on Housing Data

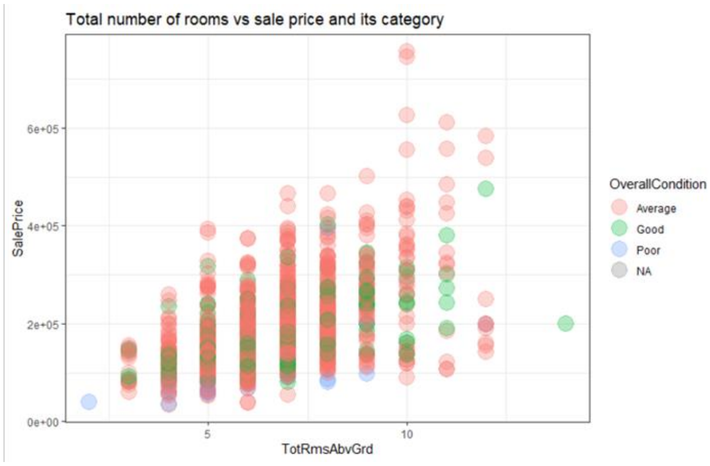
- Does reducing dimensionality has any effect on model prediction?

2. Numerical and Graphical Summaries:

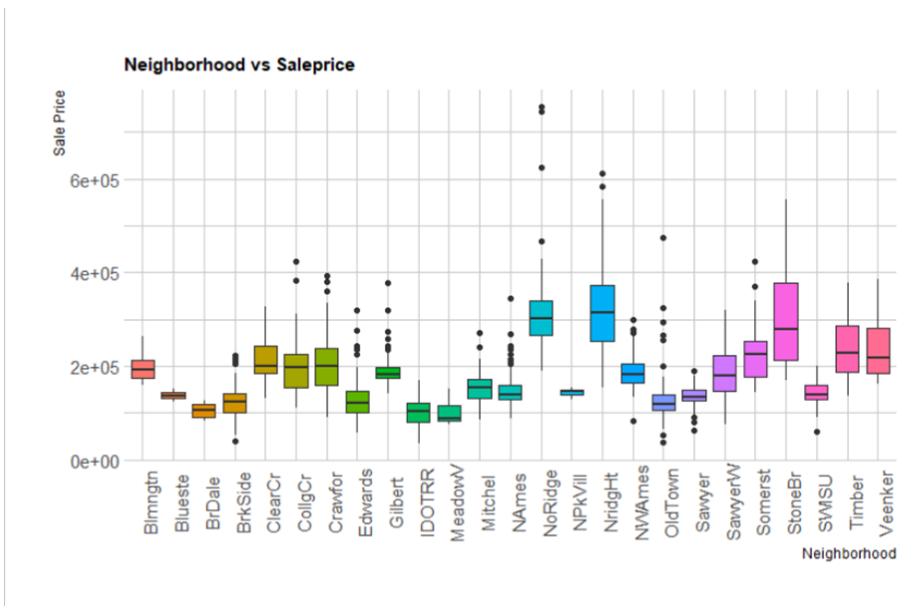
Here we are going to use the function summary () to display the statistical or numerical summary of the dataset. Graphical data or qualitative data: The data in the form of Non numerical, can be grouped or sorted by category. It can be ordinal data and nominal data. Ordinal data: The graphical or qualitative data that can be ranked or ordered. Example: Good or poor, effective, or non-effective and agree or disagree. Nominal data: The graphical or qualitative data that cannot be ranked or ordered. Example: Color, texture, names, gender etc.

To summarize the numerical data, we are going to present them using plots. There are many types of plots which can be used to display the relation between the variables in the dataset.

Here we are visualizing how sale price of a house depends on number of rooms and overall condition of the house.



We know that there are many factors which plays important roles while buying a house and Neighborhood is one of them. Here in the below graph, we are visualizing Sale Price according to Neighborhood.



Housing Market Analysis can be visualized as below with few most important factors such as Area in sqare feet including Lot Area and Neighborhood. Most of the sale prices depend on these three factors, so we are visualizing the relation between these values and how sale price depends on them.

3. Imputation of Missing Data:

Firstly, the main issue with any data is that the dataset having missing values which may reduce the accuracy of the model for predictions. There are many ways to deal with missing values such as imputing mean, median, mode values, removing the missing values or using multiple imputations with the help of "mice" package. If we delete the missing values the predictions will be biased so we are using median imputation for numerical values and replacing the missing values of categorical variables.

We can display the number of missing values present in each column with the help of is.na () and sapply ().

The main issue with any data is that the dataset having missing values which may reduce the accuracy of the model for predictions. There are many ways to deal with missing values such as imputing mean, median, mode values, removing the missing values or using multiple imputations with the help of "mice" package. If we delete the missing values the predictions will be biased so we are using median imputation for numerical values and replacing the missing values of categorical variables.

For column LotFrontage we replaced missing values with median due to lots of outliers.

For column MasVnrArea we replaced missing values with 0.

For column Alley we replaced missing values with "No alley access".

For column BsmtCond we replaced missing values with “No Basement”.

For column BsmtQual we replaced missing values with “No Basement”

For columns GarageCond and GarageType we replaced missing values with “No Garage”

For column PoolQC we replaced missing values with “No Pool”

For column Fence we replaced missing values with “No Fence”

For column MiscFeature we replaced missing values with “None”

4. Classification for Over All condition of the house:

We use classification techniques of machine learning to classify Overall condition of the house.

4.1 Logistic Regression:

Now we are going to predict the overall condition of the house and categorize them into Poor, Average and Good. To compare after prediction is done, we have to categorize the data priorly. So we are creating a column “OverallCondition” in the dataset and categorizing the “OverallCond” values as follows.

If the value is >=1 and <=3 then it is categorized as Poor, if the value is >=4 and <=6 then it is categorized as Average and if the value is >=7 and <=10 then it is categorized as Good. After categorizing, we use table () to display the number of times each value comes.

To create a model, we are converting the categorical values of a variable to numerical values using unclass () function. Then we are splitting the data into train data and test data with train data having 1000 rows and test data having 460 rows out of 1460 rows.

Logistic regression is the most renown name for statistical modelling. Here we are going to use Logistic Regression to train our model and Perform predictions. We have taken the necessary columns in x-axis and “OverallCond” in y-axis as we are going to make predictions about the Overall condition. After training, we can visualize the summary of the trained model where we can find the p values which lets us decide which columns to remove and AIC (Akaike information criterion) which tells how good our model is.

After that, we now start our predictions on test data using predict () function and round the decimal values present in the output. Then, categorize the predicted output into Poor, Average and Good as before and draw a confusion matrix. In our case, our model predicted all the values correctly with 0 error rate.

4.2 Naïve Bayes:

Now, we are going to perform classification method to perform predictions on test data. There are many classification methods such as Linear Discriminant Analysis, Quadratic Discriminant Analysis, Naïve Bayes, Decision Tree etc., that can be used and here we are opting to use Naïve Bayes Classification method to build and train our model. Naïve Bayes is a classification model that works with the help of Bayes theorem. To perform classification using Naïve Bayes classifier in R we need to install a package called “e1071” and load it. This package has an inbuilt function called naiveBayes () that can be used to build the model. Using the required variables and the inbuilt naiveBayes () function we are building the model using train data. After training the model we now start making predictions on the test data. After completing predictions, we categorize the values predicted into Poor, Average and Good. Then, we calculate the accuracy of the model which is 64% in our case.

5. Regression for Sale Price Prediction

‘Random Forest’, ‘Gradient Boosting’ and ‘SVM’ algorithms were chosen to construct machine learning models to predict the sale price of a houses based on given predictors.

5.1 Random Forests: Random Forest builds several decision trees based on bootstrapped training samples and while constructing these trees, each time a split is considered, a random sample of m predictors are considered from a full set of p predictors. Every time a fresh set of m predictors are chosen at each split, and typically $m = \sqrt{p}$. As a result, the constructed trees are decorrelated and less variable and are more reliable.

5.2 Boosting: In Boosting method, the trees are grown sequentially. Each tree is formed using information from previous tree constructed. This approach does not involve bootstrap sampling. Given a current model, the boosting algorithm fits a decision tree to the residuals from the model. The tree is fitted on current residuals rather than outcome Y, as the response. We then update the residuals by adding this new decision tree to the fitted function. This process learns slowly, and the trees can be small.

5.3 SVM: Support Vector machine is a flexible algorithm. It allows to discover and model non-linear relationships in the data. By employs kernel trick, the data is projected into high dimensional space such that the linearly non-separable data can be separated when projected in high dimensional space by simply drawing a hyperplane. Choice of kernels likes linear, radial and polynomial helps to solve complex non-linear problems.

5.4 Resampling methods

Resampling techniques involves taking samples repeatedly from a training dataset and refitting the model of interest on each sample in order to obtain additional information about the fitted model.

For instance, in order to evaluate the variability of a Random Forest Regression fit, we need to repeat the process multiple times by selecting different training and test sample, every time we do train test split of whole data and repeat fitting the same Random Forest model on these different training samples and check to what extent the resulting fits differ.

This approach allows us to obtain additional information about the model, which would not otherwise available if we fit on single training sample.

k-fold Cross Validation and bootstrap are two popular resampling techniques, we have employed these two methods for estimating the test error associated with fitting these models on the training data.

5.5 k-fold cross validation: This method is used to estimate the test error associated with a given statistical learning method to evaluate its performance. In this approach the data is divided randomly into k groups, or folds, of approximately same size. The first fold is considered as validation set, and the model is fitted on k-1 folds. This procedure is repeated k times; every time, a different set of observations is taken as a validation set, the MSE is then calculated on this validation set. This approach results in k estimates of the test error and the K-fold CV estimate is calculated by taking the average of these values.

$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\mathrm{MSE}_i.$$

5.6 Bootstrap:

In bootstrap the sampling is performed by drawing multiple observations from original data with replacement. It is widely used method quantify the uncertainty associated with a given estimator.

We can measure the variability of regression coefficients using bootstrap or measure the test error associated with fitting different estimators on same data.

$$\mathrm{SE}_B(\hat{\alpha}) = \sqrt{\frac{1}{B-1}\sum_{r=1}^{B}\left(\hat{\alpha}^{*r} - \frac{1}{B}\sum_{r'=1}^{B}\hat{\alpha}^{*r'}\right)^2}.$$

	RMSE, R^2 (once only training)	**Cross validation (RMSE, R^2)**	**Bootstrap (RMSE, R^2)**
Random Forest	33737.56, 0.869	30085.82, 0.858	31666.14, 0.848
Gradient Boosting	33737.56, 0.852	29680.43, 0.861	31198.14, 0.845
SVM Linear	32960.11, 0.858	39057.77, 0.759	43366.72, 0.704

6. Research Question on Housing Data

6.1 Does reducing dimensionality has any effect on model prediction?

Solution:

In the housing data we have 51 columns which means data distribution is in 51 dimensions. In machine learning we dimensionality reduction techniques which will help in reducing the dimensions of the data and improve the accuracy of the predictions. Principle Component Analysis is a dimensionality reduction technique used in machine learning.

Principle Component Analysis:

Principal Component Analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set. Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity. Because smaller data sets are easier to explore and visualize and make analysing data much easier and faster for machine learning algorithms without extraneous variables to process.

By applying PCA to the data, Based on Cumulative proportion of variance we selected first 35 principle components to train random forest model and predict sale price of the house from validation set.

Accuracy of Random Forest Model:

The R^2 score for Random Forest Model after dimensionality reduction is 0.8880039

7. Conclusion:

By the above models we can say that Logistic regression is the best method to make predictions about the condition of the house.We had chosen random forest over decision trees and bagging methods because it reduces the variance associated with estimating test error and boosting methods learn slowly compared to ensemble methods, they are generally well known to perform when compared to traditional machine learning methods because they implement gradient descent. The results also show that Gradient descent worked better when compared to other methods. We have done model assessment or selection using k fold cross validation and bootstrap resampling techniques. Using PCA the data is transformed into a low dimensional space. The compressed data i.e principal components are used for modelling and an equal accuracy is achived compared to algorithms such as RandomForest. After dimensionality reduction the accuracy of the Random forest model increased to 88%

8. References:

1. Brownlee, J., 2021. Bagging and Random Forest Ensemble Algorithms for Machine Learning. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/

2. Built In. 2021. A Step-by-Step Explanation of Principal Component Analysis. [online] Available at: https://builtin.com/data-science/step-step-explanation-principal-component-analysis

3. James, G., Witten, D., Hastie, T. and Tibshirani, R., n.d. An introduction to statistical learning.
4. Cran.r-project.org. 2021. A Short Introduction to the caret Package. [online] Available at: https://cran.r-project.org/web/packages/caret/vignettes/caret.html

9. Individual Contributions:

Name	Part done as per question	Individual Contribution	Additional Contribution
Smruti Das	3(a)	Code, Report, and presentation Part for 3(b)	• Helped in making changes to report and presentation.
Jhansi Rani Choutapalem	3(b)	Code, Report, and presentation Part for 4 (a)	• Adding Slides to presentation • Missing value Imputation
Venkata Naga Sai Pooja Kommasani	4	Code, Report, and presentation Part for 4	• Have collated whole report • Have collated the presentation. • Written Abstract, Introduction,

			conclusion.
Mamatha Sai Yarabarla	2(b)	Code, Report, and presentation Part for 2(b)	
Raja Sumanth Dulam	2(a)	Code, Report, and presentation Part for 2(a)	
Kotla Madhav Srinivas	1	Code, Report, and presentation Part for 1	

10. Appendix

```
#loading required libraries
library(data.table)
library(ggplot2)
library(dplyr)
library(Amelia)
library(e1071)
library(hrbrthemes)
library(gganimate)
library(corrplot)
```

```
> #loading required libraries
> library(data.table)
> library(ggplot2)
> library(dplyr)
> library(Amelia)
> library(e1071)
> library(hrbrthemes)
> library(gganimate)
> library(corrplot)
> 
```

```
df <- read.csv("D:\\New folder\\Applied Statistics\\house-data.csv", header = T)
```

```
Console  Terminal ×  Jobs ×
~/
> #loading the dataset
> df <- read.csv("D:\\New folder\\Applied Statistics\\house-data.csv", header = T)
> 
```

```
summary(df)
```

```
Console   Terminal ×   Jobs ×
~/
> #numerical summary statistics
> summary(df)
       Id          LotFrontage        LotArea          Street             Alley            Utilities
 Min.   :   1.0   Min.   : 21.00   Min.   :  1300   Length:1460        Length:1460        Length:1460
 1st Qu.: 365.8   1st Qu.: 59.00   1st Qu.:  7554   Class :character   Class :character   Class :character
 Median : 730.5   Median : 69.00   Median :  9478   Mode  :character   Mode  :character   Mode  :character
 Mean   : 730.5   Mean   : 70.05   Mean   : 10517
 3rd Qu.:1095.2   3rd Qu.: 80.00   3rd Qu.: 11602
 Max.   :1460.0   Max.   :313.00   Max.   :215245
                  NA's   :259
  LotConfig         Neighborhood        Condition1         Condition2          BldgType
 Length:1460        Length:1460        Length:1460        Length:1460        Length:1460
 Class :character   Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character


  HouseStyle         OverallQual      OverallCond      YearBuilt     RoofStyle           RoofMatl
 Length:1460        Min.   : 1.000   Min.   :1.000   Min.   :1872   Length:1460        Length:1460
 Class :character   1st Qu.: 5.000   1st Qu.:5.000   1st Qu.:1954   Class :character   Class :character
 Mode  :character   Median : 6.000   Median :5.000   Median :1973   Mode  :character   Mode  :character
                    Mean   : 6.099   Mean   :5.575   Mean   :1971
                    3rd Qu.: 7.000   3rd Qu.:6.000   3rd Qu.:2000
                    Max.   :10.000   Max.   :9.000   Max.   :2010
 Exterior1st          MasVnrArea       ExterQual          ExterCond          Foundation
 Length:1460        Min.   :   0.0   Length:1460        Length:1460        Length:1460
 Class :character   1st Qu.:   0.0   Class :character   Class :character   Class :character
 Mode  :character   Median :   0.0   Mode  :character   Mode  :character   Mode  :character
                    Mean   : 103.7
                    3rd Qu.: 166.0
                    Max.   :1600.0
                    NA's   :8
   BsmtQual           BsmtCond          TotalBsmtSF       Heating            X1stFlrSF      X2ndFlrSF
 Length:1460        Length:1460        Min.   :   0.0   Length:1460        Min.   : 334   Min.   :   0
 Class :character   Class :character   1st Qu.: 795.8   Class :character   1st Qu.: 882   1st Qu.:   0
 Mode  :character   Mode  :character   Median : 991.5   Mode  :character   Median :1087   Median :   0
                                       Mean   :1057.4                      Mean   :1163   Mean   : 347
                                       3rd Qu.:1298.2                      3rd Qu.:1391   3rd Qu.: 728
                                       Max.   :6110.0                      Max.   :4692   Max.   :2065
  LowQualFinSF        GrLivArea       FullBath      BedroomAbvGr    KitchenAbvGr   KitchenQual
 Min.   :  0.000   Min.   : 334   Min.   :0.000   Min.   :0.000   Min.   :0.000   Length:1460
 1st Qu.:  0.000   1st Qu.:1130   1st Qu.:1.000   1st Qu.:2.000   1st Qu.:1.000   Class :character
 Median :  0.000   Median :1464   Median :2.000   Median :3.000   Median :1.000   Mode  :character
 Mean   :  5.845   Mean   :1515   Mean   :1.565   Mean   :2.866   Mean   :1.047
 3rd Qu.:  0.000   3rd Qu.:1777   3rd Qu.:2.000   3rd Qu.:3.000   3rd Qu.:1.000
 Max.   :572.000   Max.   :5642   Max.   :3.000   Max.   :8.000   Max.   :3.000
```

```
  TotRmsAbvGrd     Functional           Fireplaces      GarageType          GarageArea       GarageCond
 Min.   : 2.000   Length:1460        Min.   :0.000   Length:1460        Min.   :   0.0   Length:1460
 1st Qu.: 5.000   Class :character   1st Qu.:0.000   Class :character   1st Qu.: 334.5   Class :character
 Median : 6.000   Mode  :character   Median :1.000   Mode  :character   Median : 480.0   Mode  :character
 Mean   : 6.518                      Mean   :0.613                      Mean   : 473.0
 3rd Qu.: 7.000                      3rd Qu.:1.000                      3rd Qu.: 576.0
 Max.   :14.000                      Max.   :3.000                      Max.   :1418.0
  PavedDrive           PoolArea           PoolQC             Fence            MiscFeature
 Length:1460        Min.   :  0.000   Length:1460        Length:1460        Length:1460
 Class :character   1st Qu.:  0.000   Class :character   Class :character   Class :character
 Mode  :character   Median :  0.000   Mode  :character   Mode  :character   Mode  :character
                    Mean   :  2.759
                    3rd Qu.:  0.000
                    Max.   :738.000
    MiscVal             MoSold           YrSold       SaleType          SaleCondition        SalePrice
 Min.   :    0.00   Min.   : 1.000   Min.   :2006   Length:1460        Length:1460        Min.   : 34900
 1st Qu.:    0.00   1st Qu.: 5.000   1st Qu.:2007   Class :character   Class :character   1st Qu.:129975
 Median :    0.00   Median : 6.000   Median :2008   Mode  :character   Mode  :character   Median :163000
 Mean   :   43.49   Mean   : 6.322   Mean   :2008                                         Mean   :180921
 3rd Qu.:    0.00   3rd Qu.: 8.000   3rd Qu.:2009                                         3rd Qu.:214000
 Max.   :15500.00   Max.   :12.000   Max.   :2010                                         Max.   :755000
> |
```

#dimensions of dataset

dim(df)

```
> #dimensions of dataset
> dim(df)
[1] 1460   51
>
```

#displaying the columns having null values and number of missing values in each column
colSums(sapply(df, is.na))

```
> #displaying the columns having null values and number of missing values in each column
> colSums(sapply(df, is.na))
           Id   LotFrontage       LotArea        Street         Alley     Utilities     LotConfig
            0           259             0             0          1369             0             0
 Neighborhood    Condition1    Condition2      BldgType    HouseStyle   OverallQual   OverallCond
            0             0             0             0             0             0             0
    YearBuilt     RoofStyle      RoofMatl   Exterior1st    MasVnrArea     ExterQual     ExterCond
            0             0             0             0             8             0             0
   Foundation      BsmtQual      BsmtCond   TotalBsmtSF       Heating     X1stFlrSF     X2ndFlrSF
            0            37            37             0             0             0             0
 LowQualFinSF     GrLivArea      FullBath  BedroomAbvGr  KitchenAbvGr   KitchenQual  TotRmsAbvGrd
            0             0             0             0             0             0             0
   Functional    Fireplaces    GarageType    GarageArea    GarageCond    PavedDrive      PoolArea
            0             0            81             0            81             0             0
       PoolQC         Fence   MiscFeature       MiscVal        MoSold        YrSold      SaleType
         1453          1179          1406             0             0             0             0
SaleCondition     SalePrice
            0             0
>
```

#Imputatuion method for the columns having missing values i.e., LotFrontage and MasVnrArea by using median

df$LotFrontage[which(is.na(df$LotFrontage))] <- median(df$LotFrontage,na.rm = TRUE)
df$MasVnrArea[which(is.na(df$MasVnrArea))] <- 0

print(sum(is.na(df$MasVnrArea)))
print(sum(is.na(df$LotFrontage)))

```
            0             0
> #Imputatuion method for the columns having missing values i.e., LotFrontage and MasVnrArea by using median
> df$LotFrontage[which(is.na(df$LotFrontage))] <- median(df$LotFrontage,na.rm = TRUE)
> df$MasVnrArea[which(is.na(df$MasVnrArea))] <- 0
>
> print(sum(is.na(df$MasVnrArea)))
[1] 0
> print(sum(is.na(df$LotFrontage)))
[1] 0
>
```

########### Treating missing values in categorical columns #################

replacing NA's with "No alley access" in Alley column
df$Alley <- as.character(df$Alley)
df$Alley[which(is.na(df$Alley))] <- "No alley access"
df$Alley <- as.factor(df$Alley)

```
# replacing NA's with "No Basement" in BsmtCond, BsmtQual columns
df$BsmtCond <- as.character(df$BsmtCond)
df$BsmtCond[is.na(df$BsmtCond)] <- "No Basement"
df$BsmtCond <- as.factor(df$BsmtCond)
print(table(df$BsmtCond))
df$BsmtQual <- as.character(df$BsmtQual)
df$BsmtQual[is.na(df$BsmtQual)] <- "No Basement"
df$BsmtQual <- as.factor(df$BsmtQual)
print(table(df$BsmtQual))
```

```
> ################################ Treating missing values in categorical columns ##############################
>
> # replacing NA's with "No alley access" in Alley column
> df$Alley <- as.character(df$Alley)
> df$Alley[which(is.na(df$Alley))] <- "No alley access"
> df$Alley <- as.factor(df$Alley)
>
>
> # replacing NA's with "No Basement" in BsmtCond, BsmtQual columns
> df$BsmtCond <- as.character(df$BsmtCond)
> df$BsmtCond[is.na(df$BsmtCond)] <- "No Basement"
> df$BsmtCond <- as.factor(df$BsmtCond)
> print(table(df$BsmtCond))

         Fa          Gd No Basement          Po          TA 
         45          65          37           2        1311 
> df$BsmtQual <- as.character(df$BsmtQual)
> df$BsmtQual[is.na(df$BsmtQual)] <- "No Basement"
> df$BsmtQual <- as.factor(df$BsmtQual)
> print(table(df$BsmtQual))

         Ex          Fa          Gd No Basement          TA 
        121          35         618          37         649 
```

```
# replacing NA's with ""No Garage" in GarageCond , GarageType columns
df$GarageCond <- as.character(df$GarageCond)
df$GarageCond[is.na(df$GarageCond)] <- "No Garage"
df$GarageCond <- as.factor(df$GarageCond)
print(table(df$GarageCond))
df$GarageType <- as.character(df$GarageType)
df$GarageType[is.na(df$GarageType)] <- "No Garage"
df$GarageType <- as.factor(df$GarageType)
print(table(df$GarageType))

# replacing NA's with "No Pool" in PoolQC column
df$PoolQC <- as.character(df$PoolQC)
df$PoolQC[is.na(df$PoolQC)] <- "No Pool"
df$PoolQC <- as.factor(df$PoolQC)
print(table(df$PoolQC ))
```

```
> # replacing NA's with ""No Garage" in GarageCond , GarageType columns
> df$GarageCond <- as.character(df$GarageCond)
> df$GarageCond[is.na(df$GarageCond)] <- "No Garage"
> df$GarageCond <- as.factor(df$GarageCond)
> print(table(df$GarageCond))

       Ex        Fa        Gd No Garage        Po        TA 
        2        35         9        81         7      1326 
> df$GarageType <- as.character(df$GarageType)
> df$GarageType[is.na(df$GarageType)] <- "No Garage"
> df$GarageType <- as.factor(df$GarageType)
> print(table(df$GarageType))

   2Types    Attchd   Basment   BuiltIn   CarPort    Detchd No Garage 
        6       870        19        88         9       387        81 
> 
> # replacing NA's with "No Pool" in PoolQC column
> df$PoolQC <- as.character(df$PoolQC)
> df$PoolQC[is.na(df$PoolQC)] <- "No Pool"
> df$PoolQC <- as.factor(df$PoolQC)
> print(table(df$PoolQC ))

     Ex      Fa      Gd No Pool 
      2       2       3    1453 
> |
```

```
# replacing NA's with  "No Fence" in Fence column
df$Fence <- as.character(df$Fence)
df$Fence[is.na(df$Fence)] <- "No Fence"
df$Fence <- as.factor(df$Fence)
print(table(df$Fence))

# replacing NA's with "None" in Fence column
df$MiscFeature <- as.character(df$MiscFeature)
df$MiscFeature[which(is.na(df$MiscFeature))] <- "None"
df$MiscFeature <- as.factor(df$MiscFeature)
print(table(df$MiscFeature))
```

```
> # replacing NA's with  "No Fence" in Fence column
> df$Fence <- as.character(df$Fence)
> df$Fence[is.na(df$Fence)] <- "No Fence"
> df$Fence <- as.factor(df$Fence)
> print(table(df$Fence))

   GdPrv     GdWo    MnPrv     MnWw No Fence 
      59       54      157       11     1179 
> 
> # replacing NA's with "None" in Fence column
> df$MiscFeature <- as.character(df$MiscFeature)
> df$MiscFeature[which(is.na(df$MiscFeature))] <- "None"
> df$MiscFeature <- as.factor(df$MiscFeature)
> print(table(df$MiscFeature))

Gar2 None Othr Shed TenC 
   2 1406    2   49    1 
```

```
################### Factorizing #############################
df$Alley <-  as.factor(df$Alley)
```

```
df$BsmtCond <- as.factor(df$BsmtCond)
df$BsmtQual <- as.factor(df$BsmtQual)
df$GarageCond<- as.factor(df$GarageCond)
df$GarageType<- as.factor(df$GarageType)
df$PoolQC<- as.factor(df$PoolQC)
df$Fence<- as.factor(df$Fence)
df$MiscFeature<- as.factor(df$MiscFeature)
df$Street <- as.factor(df$Street)
df$Utilities <- as.factor(df$Utilities)
df$LotConfig <- as.factor(df$LotConfig)
df$Neighborhood <- as.factor(df$Neighborhood)
df$Condition1<- as.factor(df$Condition1)
df$Condition2<- as.factor(df$Condition2)
df$BldgType<- as.factor(df$BldgType)
df$HouseStyle<- as.factor(df$HouseStyle)
df$RoofStyle<- as.factor(df$RoofStyle)
df$RoofMatl <- as.factor(df$RoofMatl)
df$Exterior1st <- as.factor(df$Exterior1st)
df$ExterQual  <- as.factor(df$ExterQual)
df$ExterCond  <- as.factor(df$ExterCond)
df$Foundation <- as.factor(df$Foundation)
df$Heating  <- as.factor(df$Heating)
df$KitchenQual <- as.factor(df$KitchenQual)
df$Functional <- as.factor(df$Functional)
df$PavedDrive <- as.factor(df$PavedDrive)
df$SaleType <- as.factor(df$SaleType)
df$SaleCondition <- as.factor(df$SaleCondition)
```

```
> ################################  Factorizing ############################################################
> df$Alley <-  as.factor(df$Alley)
> df$BsmtCond <- as.factor(df$BsmtCond)
> df$BsmtQual <- as.factor(df$BsmtQual)
> df$GarageCond<- as.factor(df$GarageCond)
> df$GarageType<- as.factor(df$GarageType)
> df$PoolQC<- as.factor(df$PoolQC)
> df$Fence<- as.factor(df$Fence)
> df$MiscFeature<- as.factor(df$MiscFeature)
> df$Street <- as.factor(df$Street)
> df$Utilities <- as.factor(df$Utilities)
> df$LotConfig <- as.factor(df$LotConfig)
> df$Neighborhood <- as.factor(df$Neighborhood)
> df$Condition1<- as.factor(df$Condition1)
> df$Condition2<- as.factor(df$Condition2)
> df$BldgType<- as.factor(df$BldgType)
> df$HouseStyle<- as.factor(df$HouseStyle)
> df$RoofStyle<- as.factor(df$RoofStyle)
> df$RoofMatl <- as.factor(df$RoofMatl)
> df$Exterior1st <- as.factor(df$Exterior1st)
> df$ExterQual  <- as.factor(df$ExterQual)
> df$ExterCond  <- as.factor(df$ExterCond)
> df$Foundation <- as.factor(df$Foundation)
> df$Heating  <- as.factor(df$Heating)
> df$KitchenQual <- as.factor(df$KitchenQual)
> df$Functional <- as.factor(df$Functional)
> df$PavedDrive <- as.factor(df$PavedDrive)
> df$SaleType <- as.factor(df$SaleType)
> df$SaleCondition <- as.factor(df$SaleCondition)
> 
```

#Checking for the missing values and can be observed that there are no missing values present after imputation

colSums(sapply(df, is.na))

```
> #Checking for the missing values and can be observed that there are no missing values present after imputation
> colSums(sapply(df, is.na))
           Id   LotFrontage       LotArea        Street         Alley     Utilities     LotConfig 
            0             0             0             0             0             0             0 
 Neighborhood    Condition1    Condition2      BldgType    HouseStyle   OverallQual   OverallCond 
            0             0             0             0             0             0             0 
    YearBuilt     RoofStyle      RoofMatl   Exterior1st    MasVnrArea     ExterQual     ExterCond 
            0             0             0             0             0             0             0 
   Foundation      BsmtQual      BsmtCond   TotalBsmtSF       Heating     X1stFlrSF     X2ndFlrSF 
            0             0             0             0             0             0             0 
 LowQualFinSF     GrLivArea      FullBath  BedroomAbvGr  KitchenAbvGr   KitchenQual  TotRmsAbvGrd 
            0             0             0             0             0             0             0 
   Functional    Fireplaces    GarageType    GarageArea    GarageCond    PavedDrive      PoolArea 
            0             0             0             0             0             0             0 
       PoolQC         Fence   MiscFeature       MiscVal        MoSold        YrSold      SaleType 
            0             0             0             0             0             0             0 
SaleCondition     SalePrice 
            0             0 
> 
```

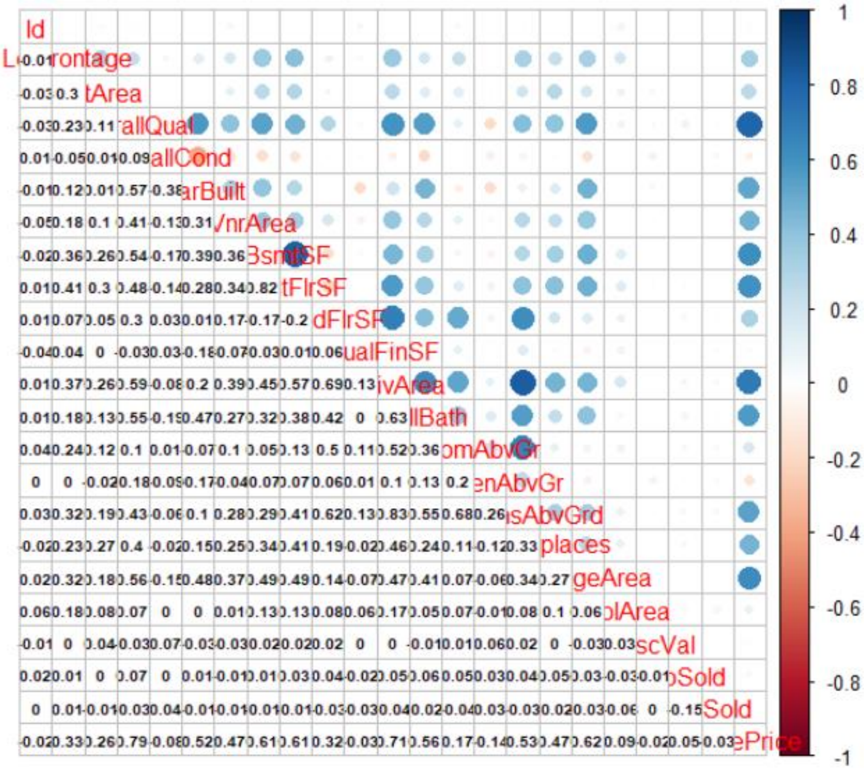
#correlation for the variables present in dataset which has numerical values

df_new <- df %>% select_if(is.numeric)

df_new.corr<-cor(df_new)

corrplot.mixed(df_new.corr, lower.col = "black", number.cex = .6)

```
> #correlation for the variables present in dataset which has numerical values
> df_new <- df %>% select_if(is.numeric)
> df_new.corr<-cor(df_new)
> corrplot.mixed(df_new.corr, lower.col = "black", number.cex = .6)
> 
```

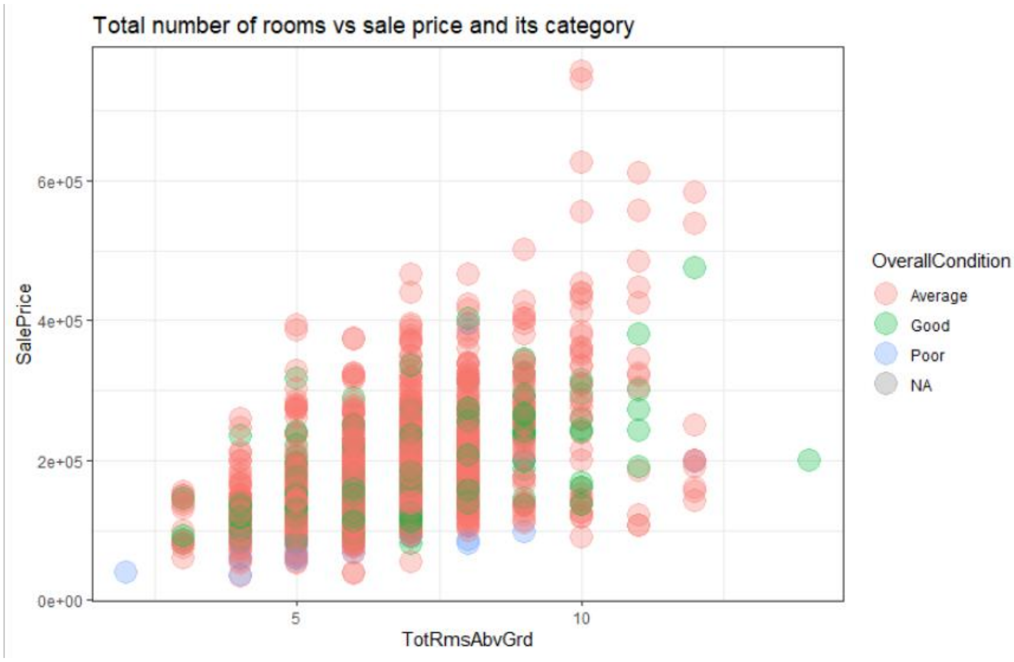


#creating a column "OverallCondition" in the data set to categorize the overall condition of the house w.r.t "OverallCond and categorizing as Poor, Average and Good"
table(df$OverallCond)
setDT(df)[OverallCond >1 & OverallCond <=3, OverallCondition := "Poor"]
df[OverallCond >3 & OverallCond <=6, OverallCondition := "Average"]
df[OverallCond >6 & OverallCond <=10, OverallCondition := "Good"]
#displaying the total number of times a unique value comes in the created column
df[,table(OverallCondition)]

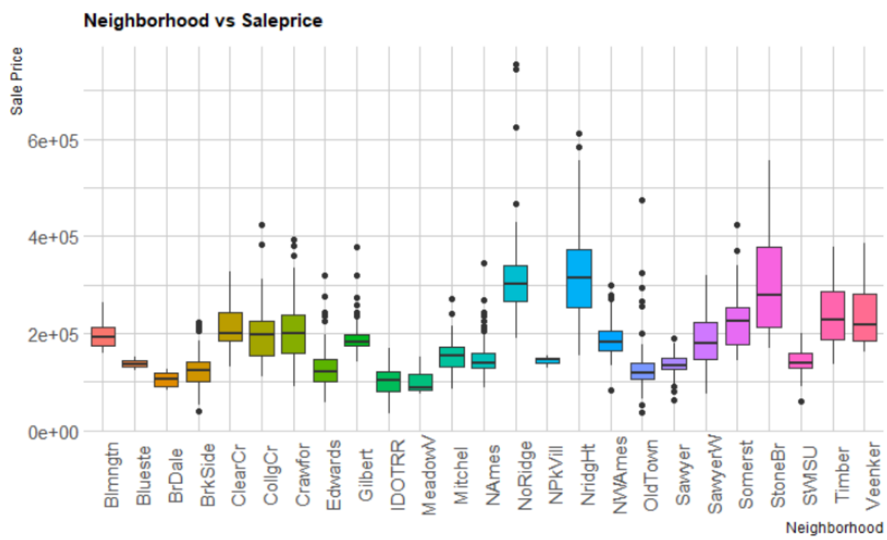
```
> table(df$OverallCond)

  1   2   3   4   5   6   7   8   9
  1   5  25  57 821 252 205  72  22
> setDT(df)[OverallCond >1 & OverallCond <=3, OverallCondition := "Poor"]
> df[OverallCond >3 & OverallCond <=6, OverallCondition := "Average"]
> df[OverallCond >6 & OverallCond <=10, OverallCondition := "Good"]
> #displaying the total number of times a unique value comes in the created column
> df[,table(OverallCondition)]
OverallCondition
Average    Good    Poor
   1130     299      30
>
```

#Plot for total rooms vs sale price and its overall condition
ggplot(df, aes(x=TotRmsAbvGrd, y=SalePrice, color=OverallCondition)) +

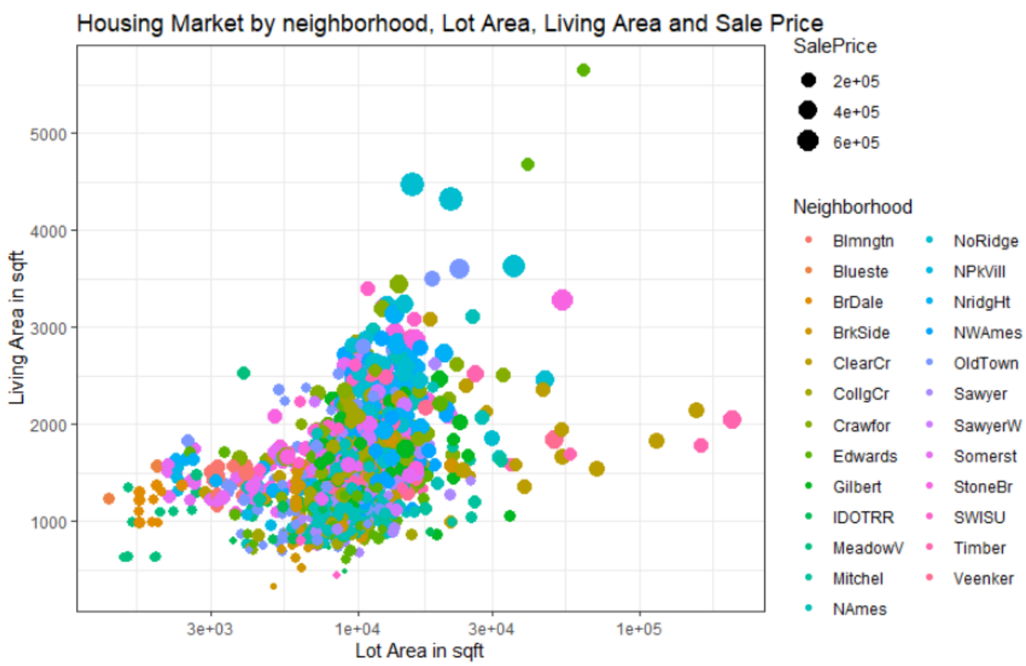
```
  geom_point(size=6, alpha = 0.3) + theme_bw() + labs(title = "Total number of rooms vs sale price and its category", xlab = "Total rooms", ylab = "Sale Price")
```




```
#Plotting Neighborhood vs Sale Price using boxplot
ggplot(df, aes(x=Neighborhood, y=SalePrice, fill=Neighborhood)) +
  geom_boxplot() +
  theme_ipsum() +
  theme(
    legend.position="none",
    plot.title = element_text(size=11)
  ) +
  ggtitle("Neighborhood vs Saleprice") +
  xlab("Neighborhood") + ylab("Sale Price") + theme(axis.text.x = element_text(angle = 90))
```




```
#Plotting Housing market analysis using Area, Sale Price and Neighborhood
ggplot(df, aes(LotArea,GrLivArea, size = SalePrice, color = Neighborhood)) +
  geom_point() +
  scale_x_log10() +
  theme_bw() + labs(title = "Housing Market by neighborhood, Lot Area, Living Area and Sale Price", x = "Lot Area in sqft", y = "Living Area in sqft")
```



```
#Dividing the dataset into train data and test data with train data consisting 1000 rows and test data with 460 rows
set.seed(52)
ids <- sample(x = 1460, size = 1000, replace = F)
train <- df[ids,]
test <- df[-ids,]
```

```
> #Dividing the dataset into train data and test data with train data consisting 1000 rows and test data with 460
 rows
> set.seed(52)
> ids <- sample(x = 1460, size = 1000, replace = F)
> train <- df[ids,]
> test <- df[-ids,]
>
```

```
#using Logistic Regression to train the model using train data and few columns adjusting according to the AIC Value
log.fit <- glm(OverallCond ~  TotRmsAbvGrd + FullBath + LotFrontage + BedroomAbvGr + MasVnrArea + TotalBsmtSF + SalePrice + MiscVal + OverallCondition + Neighborhood, data = train)
```

```
> log.fit <- glm(OverallCond ~  TotRmsAbvGrd + FullBath + LotFrontage + BedroomAbvGr + MasVnrArea + TotalBsmtSF +
 SalePrice + MiscVal + OverallCondition + Neighborhood, data = train)
>
```

```
#Displaying the statistical summary of the trained model
summary(log.fit)
```

```
Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.46785  -0.26605  -0.07028   0.09202   1.73284

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           5.267e+00  1.720e-01  30.624  < 2e-16 ***
TotRmsAbvGrd         -3.608e-02  1.723e-02  -2.094  0.03656 *
FullBath             -1.169e-01  4.401e-02  -2.656  0.00803 **
LotFrontage          -1.047e-03  8.125e-04  -1.288  0.19796
BedroomAbvGr          2.861e-02  2.989e-02   0.957  0.33861
MasVnrArea           -1.430e-05  1.084e-04  -0.132  0.89510
TotalBsmtSF          -9.195e-05  4.807e-05  -1.913  0.05606 .
SalePrice             1.598e-06  3.954e-07   4.042 5.71e-05 ***
MiscVal              -2.038e-05  9.884e-05  -0.206  0.83671
OverallConditionGood  2.092e+00  4.324e-02  48.378  < 2e-16 ***
OverallConditionPoor -2.328e+00  1.083e-01 -21.493  < 2e-16 ***
NeighborhoodBlueste   8.513e-01  5.203e-01   1.636  0.10212
NeighborhoodBrDale    1.625e-01  2.100e-01   0.774  0.43940
NeighborhoodBrkSide   2.510e-01  1.742e-01   1.441  0.14987
NeighborhoodClearCr  -1.007e-01  1.942e-01  -0.519  0.60410
NeighborhoodCollgCr   5.367e-02  1.568e-01   0.342  0.73219
NeighborhoodCrawfor   3.907e-01  1.727e-01   2.262  0.02390 *
NeighborhoodEdwards   1.948e-01  1.649e-01   1.181  0.23786
NeighborhoodGilbert   5.756e-02  1.652e-01   0.348  0.72765
NeighborhoodIDOTRR   -2.332e-03  1.907e-01  -0.012  0.99025
NeighborhoodMeadowV   1.234e-01  2.096e-01   0.589  0.55615
NeighborhoodMitchel   4.855e-02  1.715e-01   0.283  0.77710
NeighborhoodNAmes     2.346e-01  1.600e-01   1.466  0.14302
NeighborhoodNoRidge  -4.320e-02  1.801e-01  -0.240  0.81049
NeighborhoodNPkVill   4.996e-01  2.295e-01   2.177  0.02972 *
NeighborhoodNridgHt  -1.027e-01  1.664e-01  -0.617  0.53725
NeighborhoodNWAmes    4.948e-01  1.696e-01   2.918  0.00361 **
NeighborhoodOldTown   3.046e-01  1.638e-01   1.860  0.06320 .
NeighborhoodSawyer    7.936e-02  1.702e-01   0.466  0.64117
NeighborhoodSawyerW   5.276e-02  1.668e-01   0.316  0.75182
NeighborhoodSomerst  -3.688e-02  1.613e-01  -0.229  0.81923
NeighborhoodStoneBr  -1.378e-01  1.990e-01  -0.692  0.48886
NeighborhoodSWISU     3.581e-01  2.006e-01   1.786  0.07448 .
NeighborhoodTimber    4.805e-02  1.735e-01   0.277  0.78182
NeighborhoodVeenker   6.338e-02  2.415e-01   0.262  0.79301
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.2472979)

    Null deviance: 1245.6  on 998  degrees of freedom
Residual deviance:  238.4  on 964  degrees of freedom
  (1 observation deleted due to missingness)
AIC: 1475.6
```

```
#Predicting the output using test data after training the model
log.pred <- predict(log.fit, test, type = "response")
log.pred <- round(log.pred)
log.pred <- as.numeric(log.pred)

#categorize the output predicted as Poor, Average and Good
log.class <- ifelse(log.pred >=1 & log.pred <= 3, "Poor",
             ifelse(log.pred >= 4 & log.pred <=6, "Average",
                 ifelse(log.pred >=7 & log.pred<=10, "Good", NA)))

#displaying the confusion matrix
table(log.class, test$OverallCondition)
```

```
> log.pred <- predict(log.fit, test, type = "response")
> log.pred <- round(log.pred)
> log.pred <- as.numeric(log.pred)
>
> #categorize the output predicted as Poor, Average and Good
> log.class <- ifelse(log.pred >=1 & log.pred <= 3, "Poor",
+                     ifelse(log.pred >= 4 & log.pred <=6, "Average",
+                            ifelse(log.pred >=7 & log.pred<=10, "Good", NA)))
>
> #displaying the confusion matrix
> table(log.class, test$OverallCondition)

log.class Average Good Poor
  Average     363    0    0
  Good          0   91    0
  Poor          0    0    6
>
```

#Using Naivebayes to train the model
Bayes.fit <- naiveBayes(OverallCond ~ TotRmsAbvGrd + FullBath + LotFrontage + BedroomAbvGr + MasVnrArea + TotalBsmtSF + SalePrice + MiscVal + OverallCondition + Neighborhood, data = train)

#Predicting the output for test data after completing the training
testPred=predict(Bayes.fit, newdata=test, type="class")
testPred <- as.numeric(testPred)

#categorize the output predicted as Poor, Average and Good
testclass <- ifelse(testPred >=1 & testPred <= 3, "Poor",
ifelse(testPred >= 4 & testPred <=6, "Average",
ifelse(testPred >=7 & testPred<=10, "Good", NA)))

#creating a table for predicted output and original values in test data
testTable=table(test$OverallCondition, testclass)
testTable

```
> #Using Naivebayes to train the model
> Bayes.fit <- naiveBayes(OverallCond ~  TotRmsAbvGrd + FullBath + LotFrontage + BedroomAbvGr + MasVnrArea + Tota
lBsmtSF + SalePrice + MiscVal + OverallCondition + Neighborhood, data = train)
> #Predicting the output for test data after completing the training
> testPred=predict(Bayes.fit, newdata=test, type="class")
> testPred <- as.numeric(testPred)
> #categorize the output predicted as Poor, Average and Good
> testclass <- ifelse(testPred >=1 & testPred <= 3, "Poor",
+                     ifelse(testPred >= 4 & testPred <=6, "Average",
+                            ifelse(testPred >=7 & testPred<=10, "Good", NA)))
>
> #creating a table for predicted output and original values in test data
> testTable=table(test$OverallCondition, testclass)
> testTable
         testclass
          Average Good Poor
  Average     201   61  101
  Good          0   89    2
  Poor          0    0    6
> |
```

#Calculating the accuracy of the trained model

testAcc=(testTable[1,1]+testTable[2,2]+testTable[3,3])/sum(testTable)

testAcc

```
> #Calculating the accuracy of the trained model
> testAcc=(testTable[1,1]+testTable[2,2]+testTable[3,3])/sum(testTable)
> testAcc
[1] 0.6434783
> |
```

################# Regression for Sales Price Prediction#################

Installing & Loading required packages

##

install.packages('caret')

install.packages('randomForest')

install.packages('gbm')

install.packages('e1071')

library(caret)

```
library(randomForest)

library(gbm)

library(e1071)


################################################  Loading the data
#############################################################################


data <- read.csv("E:/academics/Applied statistics/house-data.csv",header=TRUE)

head(data)

str(data)


########################################## Description of column names
#############################################################################

#LotFrontage: Linear feet of street connected to property

#LotArea: Lot size in square feet

#Street: Type of road access to property

#Utilities: Type of utilities available

#LotConfig: Lot configuration

#Neighborhood: Physical locations within Ames city limits

#Condition1: Proximity to various conditions

#Condition2: Proximity to various conditions (if more than one is present)

#BldgType: Type of dwelling

#HouseStyle: Style of dwelling

#OverallQual: Rates the overall material and finish of the house
```

#OverallCond: Rates the overall condition of the house

#YearBuilt: Original construction date

#RoofStyle: Type of roof

##RoofMatl: Roof material

#Exterior1st: Exterior covering on house

#MasVnrArea: Masonry veneer area in square feet

#ExterQual: Evaluates the quality of the material on the exterior

#ExterCond: Evaluates the present condition of the material on the exterior

#Foundation: Type of foundation

#BsmtQual: Evaluates the height of the basement

#BsmtCond: Evaluates the general condition of the basement

#TotalBsmtSF: Total square feet of basement area

#Heating: Type of heating

#1stFlrSF: First Floor square feet

#2ndFlrSF: Second floor square feet

#LowQualFinSF: Low quality finished square feet (all floors)

#GrLivArea: Above grade (ground) living area square feet

#FullBath: Full bathrooms above grade

#Bedroom: Bedrooms above grade (does NOT include basement bedrooms)

#Kitchen: Kitchens above grade

#KitchenQual: Kitchen quality

#TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)

#Functional: Home functionality (Assume typical unless deductions are warranted)

```
#Fireplaces: Number of fireplaces

#GarageType: Garage location

#GarageArea: Size of garage in square feet

#GarageQual: Garage quality

#GarageCond: Garage condition

#PavedDrive: Paved driveway

#PoolArea: Pool area in square feet

#PoolQC: Pool quality

#Fence: Fence quality

#MiscFeature: Miscellaneous feature not covered in other categories

#MiscVal: $Value of miscellaneous feature

#MoSold: Month Sold (MM)

#YrSold: Year Sold (YYYY)

#SaleType: Type of sale

#SaleCondition: Condition of sale

#SalePrice: Price of the property

#################################################################################
#################################


############################## Treating missing values in numerical columns
####################################

names(data)

data$LotFrontage[which(is.na(data$LotFrontage))] <- median(data$LotFrontage,na.rm = TRUE)

data$MasVnrArea[which(is.na(data$MasVnrArea))] <- 0
```

```
print(sum(is.na(data$MasVnrArea)))

print(sum(is.na(data$LotFrontage)))


################################# Treating missing values in categorical columns
###############################


# replacing NA's with "No alley access" in Alley column

data$Alley <- as.character(data$Alley)

data$Alley[which(is.na(data$Alley))] <- "No alley access"

data$Alley <- as.factor(data$Alley)




# replacing NA's with "No Basement" in BsmtCond, BsmtQual columns

data$BsmtCond <- as.character(data$BsmtCond)

data$BsmtCond[is.na(data$BsmtCond)] <- "No Basement"

data$BsmtCond <- as.factor(data$BsmtCond)

print(table(data$BsmtCond))


data$BsmtQual <- as.character(data$BsmtQual)

data$BsmtQual[is.na(data$BsmtQual)] <- "No Basement"

data$BsmtQual <- as.factor(data$BsmtQual)

print(table(data$BsmtQual))
```

```r
# replacing NA's with ""No Garage" in GarageCond , GarageType columns
data$GarageCond <- as.character(data$GarageCond)
data$GarageCond[is.na(data$GarageCond)] <- "No Garage"
data$GarageCond <- as.factor(data$GarageCond)
print(table(data$GarageCond))

data$GarageType <- as.character(data$GarageType)
data$GarageType[is.na(data$GarageType)] <- "No Garage"
data$GarageType <- as.factor(data$GarageType)
print(table(data$GarageType))

# replacing NA's with "No Pool" in PoolQC column
data$PoolQC <- as.character(data$PoolQC)
data$PoolQC[is.na(data$PoolQC)] <- "No Pool"
data$PoolQC <- as.factor(data$PoolQC)
print(table(data$PoolQC ))

# replacing NA's with  "No Fence" in Fence column
data$Fence <- as.character(data$Fence)
data$Fence[is.na(data$Fence)] <- "No Fence"
data$Fence <- as.factor(data$Fence)
print(table(data$Fence))
```

```
# replacing NA's with "None" in Fence column

data$MiscFeature <- as.character(data$MiscFeature)

data$MiscFeature[which(is.na(data$MiscFeature))] <- "None"

data$MiscFeature <- as.factor(data$MiscFeature)

print(table(data$MiscFeature))


##################################  Factorizing the categorical columns
################################################################

data$Alley <-  as.factor(data$Alley)

data$BsmtCond <- as.factor(data$BsmtCond)

data$BsmtQual <- as.factor(data$BsmtQual)

data$GarageCond<- as.factor(data$GarageCond)

data$GarageType<- as.factor(data$GarageType)

data$PoolQC<- as.factor(data$PoolQC)

data$Fence<- as.factor(data$Fence)

data$MiscFeature<- as.factor(data$MiscFeature)

data$Street <- as.factor(data$Street)

data$Utilities <- as.factor(data$Utilities)

data$LotConfig <- as.factor(data$LotConfig)

data$Neighborhood <- as.factor(data$Neighborhood)

data$Condition1<- as.factor(data$Condition1)

data$Condition2<- as.factor(data$Condition2)
```

```
data$BldgType<- as.factor(data$BldgType)

data$HouseStyle<- as.factor(data$HouseStyle)

data$RoofStyle<- as.factor(data$RoofStyle)

data$RoofMatl <- as.factor(data$RoofMatl)

data$Exterior1st <- as.factor(data$Exterior1st)

data$ExterQual  <- as.factor(data$ExterQual)

data$ExterCond  <- as.factor(data$ExterCond)

data$Foundation <- as.factor(data$Foundation)

data$Heating  <- as.factor(data$Heating)

data$KitchenQual <- as.factor(data$KitchenQual)

data$Functional <- as.factor(data$Functional)

data$PavedDrive <- as.factor(data$PavedDrive)

data$SaleType <- as.factor(data$SaleType)

data$SaleCondition <- as.factor(data$SaleCondition)


##################### Removing the 'ID' column ######################

data <- subset(data,select=-c(Id))
dim(data)


############## Data partitioning for training and testing ##############

# splitting data for training and testing
set.seed(125)
inTraining <- createDataPartition(data$SalePrice, p = .80, list = FALSE)
```

```
train_set <- data[inTraining,]
validate_set  <- data[-inTraining,]

dim(train_set)
dim(validate_set)


##################### Model building ########################
##################### Building a Random Forest Model ####################

set.seed(825)
forest_model <- randomForest(SalePrice~.,
                        data = train_set,
                        importance=TRUE)
forest_model
plot(forest_model)

### Feature Importance ###
varImpPlot(forest_model)



########### Building a Gradient Boosting Model #############
set.seed(825)
model_gbm <- gbm(SalePrice ~., data=train_set)
model_gbm

##################### Building SVM model #####################
set.seed(825)
model_svm <- svm(SalePrice ~., data=train_set)
model_svm
```

```
### prediction on validation set: Computing RMSE and R^2 scores ###
predicted_prices_forest <- predict(forest_model, newdata=validate)
predicted_prices_gbm <- predict(model_gbm, newdata=validate)
predicted_prices_svm <- predict(model_svm, newdata=validate)

RMSE <- function(actual,predicted) {sqrt(mean((actual-predicted)^2))}
rmse_RandomForest <- RMSE(validate$SalePrice,predicted_prices_forest)
rmse_gbm <- RMSE(validate$SalePrice,predicted_prices_gbm)
rmse_svm <- RMSE(validate$SalePrice,predicted_prices_svm)

rmse_mat <- matrix(c(rmse_RandomForest,rmse_gbm,rmse_svm), nrow = 1, ncol = 3,
byrow = TRUE,
          dimnames = list(c("RMSE_validation"),
                 c("  RandomForest  ", "  GBM  ", "  SVM  ")))
rmse_mat

mean_saleprice <- mean(validate$SalePrice)
R2 <- function(predicted) { 1 - (sum((validate$SalePrice-
predicted)^2)/sum((validate$SalePrice-mean_saleprice)^2))}

R2_RandomForest <- R2(predicted_prices_forest)
R2_gbm <- R2(predicted_prices_gbm)
R2_svm <- R2(predicted_prices_svm)

R2_mat <- matrix(c(R2_RandomForest,R2_gbm,R2_svm), nrow = 1, ncol = 3, byrow =
TRUE,
            dimnames = list(c("R^2_validation"),
                   c("  RandomForest  ", "  GBM  ", "  SVM  ")))
R2_mat
######################### Random Forest ###########################
```

k- fold cross validation with k = 10

```
# The function trainControl can be used to specify the type of resampling:
fitControl <- trainControl(method = "cv",
                           number = 10)

RandomForest.cv <- train(SalePrice ~. ,
                  data= data,
                  method = 'rf',
                  trControl = fit_ctrl
                  )
print(RandomForest.cv)
plot(RandomForest.cv)
```

Bootstrapping with n=25

```
RandomForest.boot <-  train(SalePrice ~. ,
                     data= data,
                     method = 'rf')
print(RandomForest.boot)
plot(RandomForest.boot)
```

############### (Gradient Boosting) #################

```
set.seed(825)
### k-fold crossvalidation with k=10 ###
gbmFit1_cv <- train(SalePrice~ .,
                data = data,
                method = "gbm",
                trControl = trainControl("cv", number = 10),
                verbose = FALSE)
```

```
gbmFit1_cv
plot(gbmFit1_cv)
### bootstrap with n=25 ###
gbmFit1_boot <- train(SalePrice~ .,
                      data = data,
                      method = "gbm",
                      verbose = FALSE)
plot(gbmFit1_boot)
gbmFit1_boot


### For a gradient boosting machine (GBM) model, there are three main tuning
parameters: ###

# number of iterations, i.e. trees, (called n.trees in the gbm function)
# complexity of the tree, called interaction.depth
# learning rate: how quickly the algorithm adapts, called shrinkage
# the minimum number of training set samples in a node to commence splitting
(n.minobsinnode)

gbmGrid <-  expand.grid(interaction.depth = c(1, 5, 9),
                        n.trees = (1:30)*50,
                        shrinkage = 0.1,
                        n.minobsinnode = 20)

nrow(gbmGrid)

set.seed(825)
gbmFit2 <- train(SalePrice ~ ., data = data,
                 method = "gbm",
                 trControl = fitControl,
```

```
            verbose = FALSE,
            tuneGrid = gbmGrid)
gbmFit2

trellis.par.set(caretTheme())
plot(gbmFit2)

trellis.par.set(caretTheme())
plot(gbmFit2, metric = "Rsquared")

par(mfrow=c(1,2))
ggplot(gbmFit1_cv)
ggplot(gbmFit2)
```

############### Support Vector Machine (SVM) ####################

Linear kernel

```
svmFit_linear_cv <- train(SalePrice ~ ., data = data,
                   method = "svmLinear",
                   trControl = fitControl, # 10 fold cross validation
                   metric = "RMSE")
svmFit_linear_cv
ggplot(svmFit_linear_cv)

svmFit_linear_boot <- train(SalePrice ~ ., data = data,
                    method = "svmLinear",    # bootstrap with n=25
                    metric = "RMSE")
svmFit_linear_boot
ggplot(svmFit_linear_boot)

### rbf kernel ###
```

```
svmFit_rbf <- train(SalePrice ~ ., data = data,
                    method = "svmRadial",
                    trControl = fitControl, # 10 fold cross validation
                    metric = "RMSE")
svmFit_rbf

svmFit_rbf_boot <- train(SalePrice ~ ., data = data,
                         method = "svmRadial",
                         metric = "RMSE")  # bootstrap with n=25
svmFit_rbf_boot
```

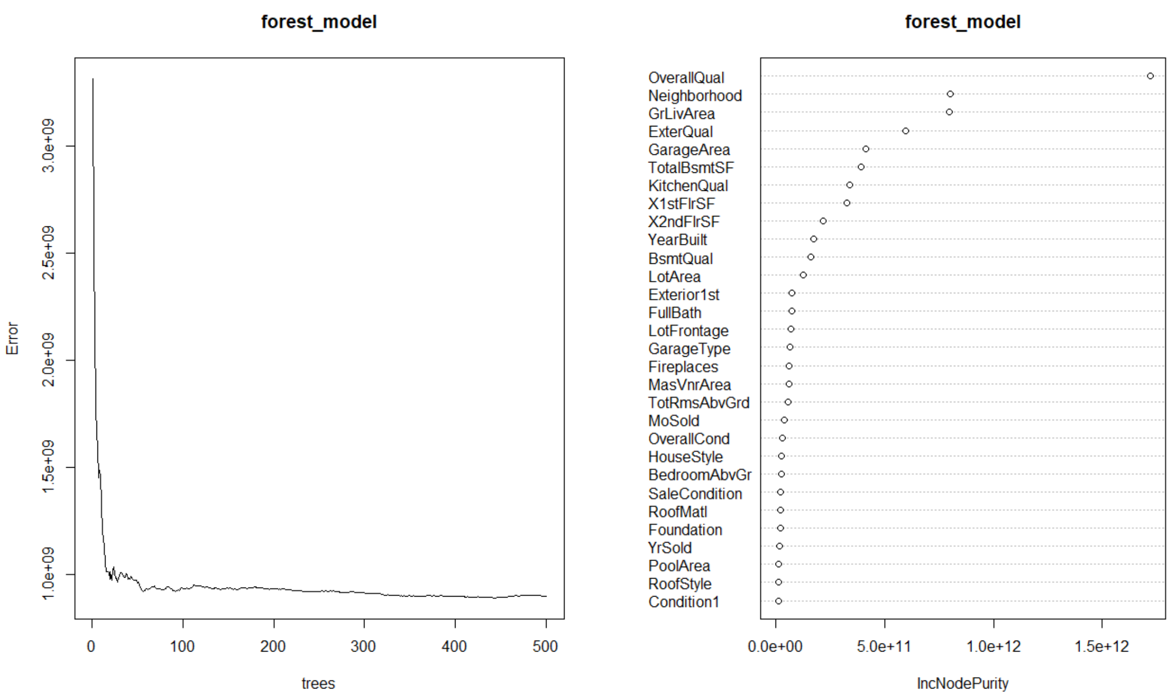
Models’ comparison before cross validation and bootstrapping:

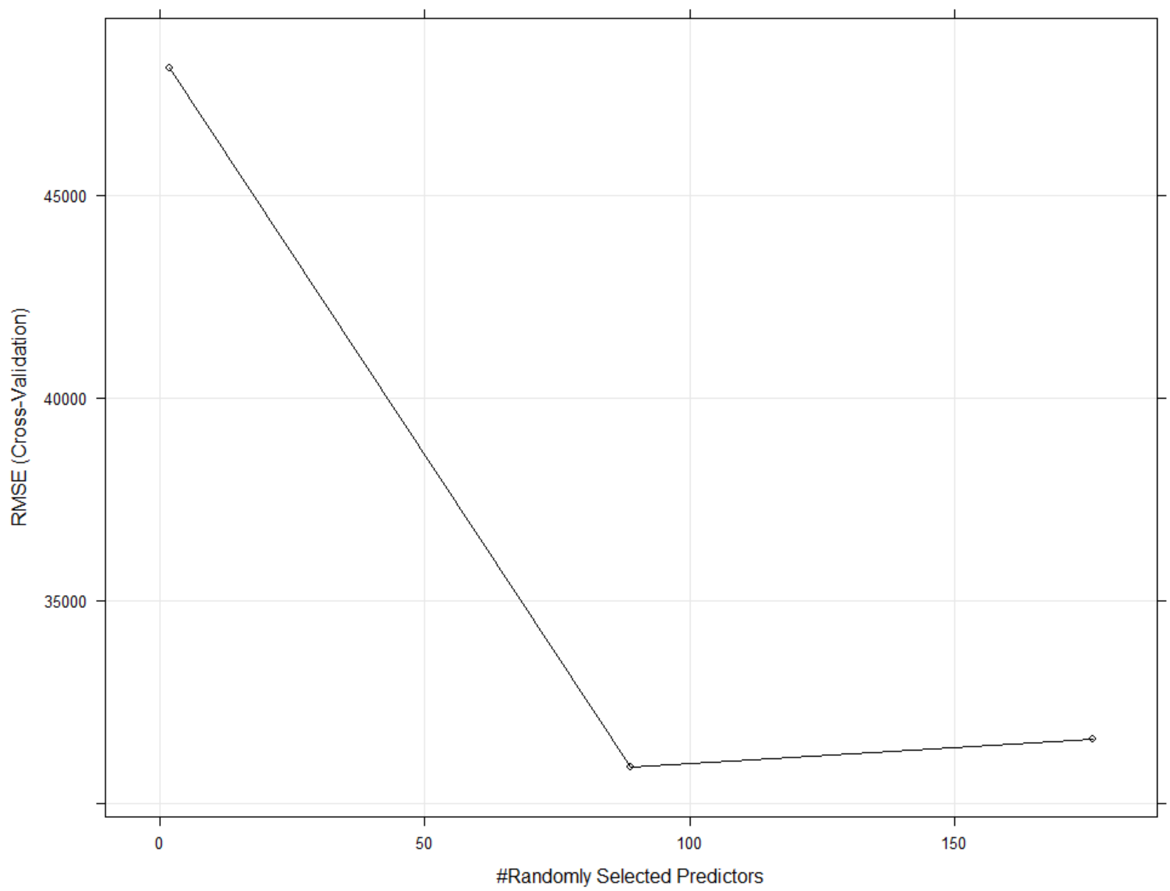
```
> rmse_mat
                 RandomForest      GBM      SVM
RMSE_validation      31653.87 33737.56 32960.11
> R2_mat
                RandomForest      GBM       SVM
R^2_validation     0.8698737 0.852178 0.8589124
>
```

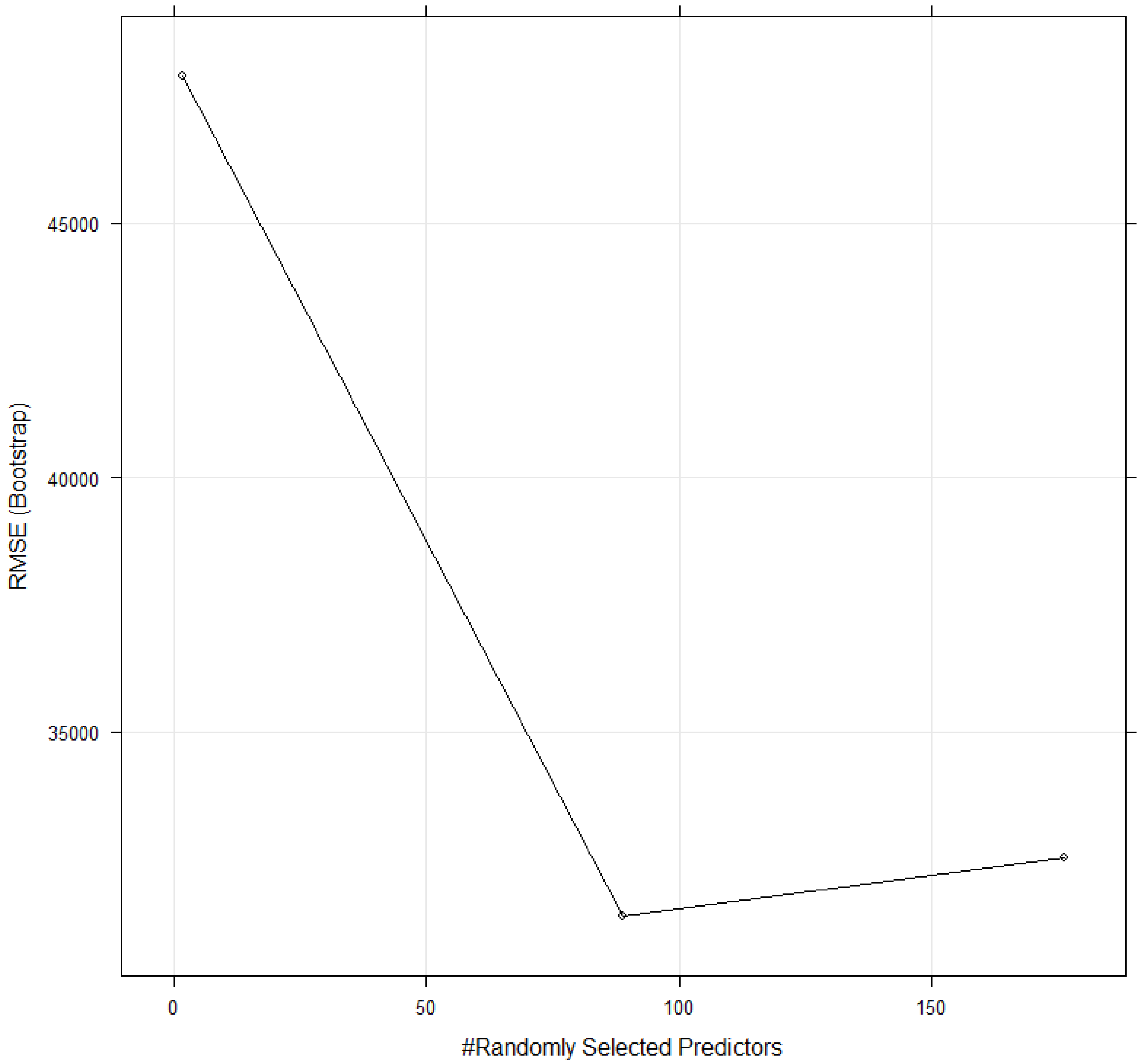
Models’ comparison after performing cross validation and bootstrapping:

Random Forest Model:

```
> print(RandomForest.cv)
Random Forest

1460 samples
  49 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 1167, 1168, 1168, 1168, 1169
Resampling results across tuning parameters:

  mtry  RMSE      Rsquared   MAE
    2   48156.68  0.7775713  30623.88
   89   30878.51  0.8538530  18202.37
  176   31593.22  0.8453337  18780.34

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 89.
> print(RandomForest.boot)
Random Forest

1460 samples
  49 predictor

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 1460, 1460, 1460, 1460, 1460, 1460, ...
Resampling results across tuning parameters:

  mtry  RMSE      Rsquared   MAE
    2   47894.64  0.7664077  30590.14
   89   31369.35  0.8422499  18377.49
  176   32534.83  0.8285532  19241.16

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 89.
```

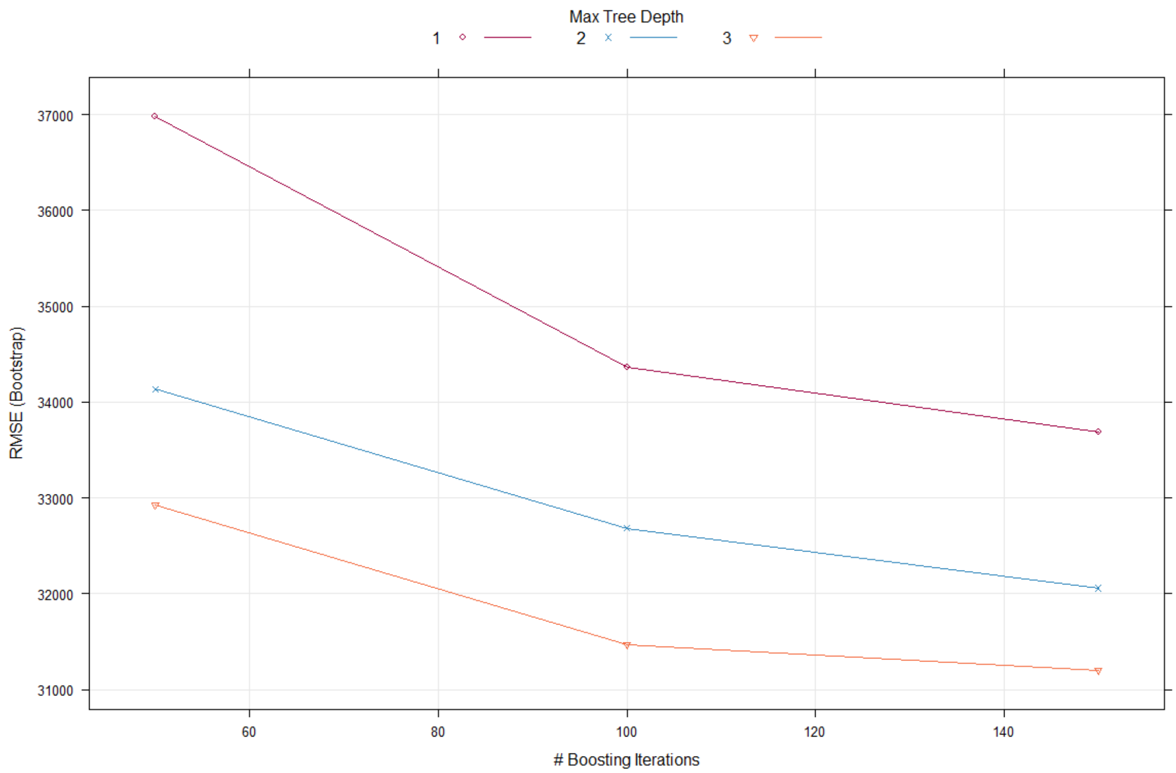



45000
40000
35000
RMSE (Cross-Validation)
0
50
100
150
#Randomly Selected Predictors

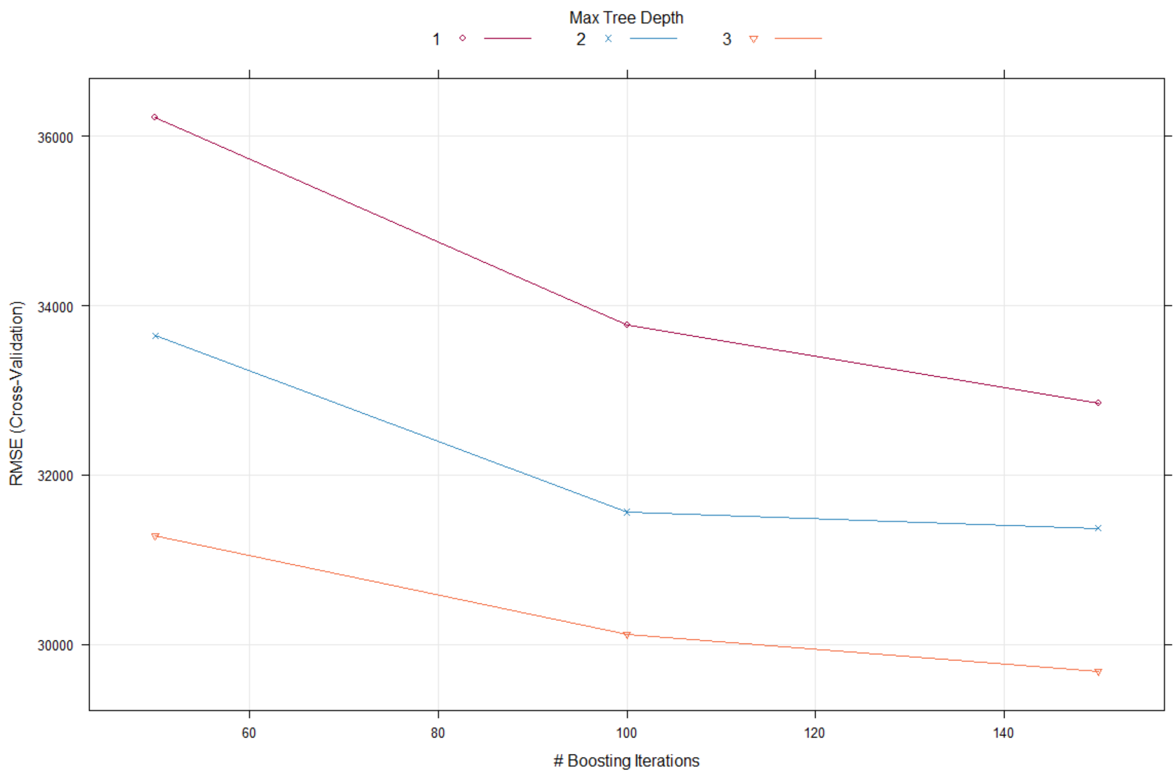


Gradient Boosting:

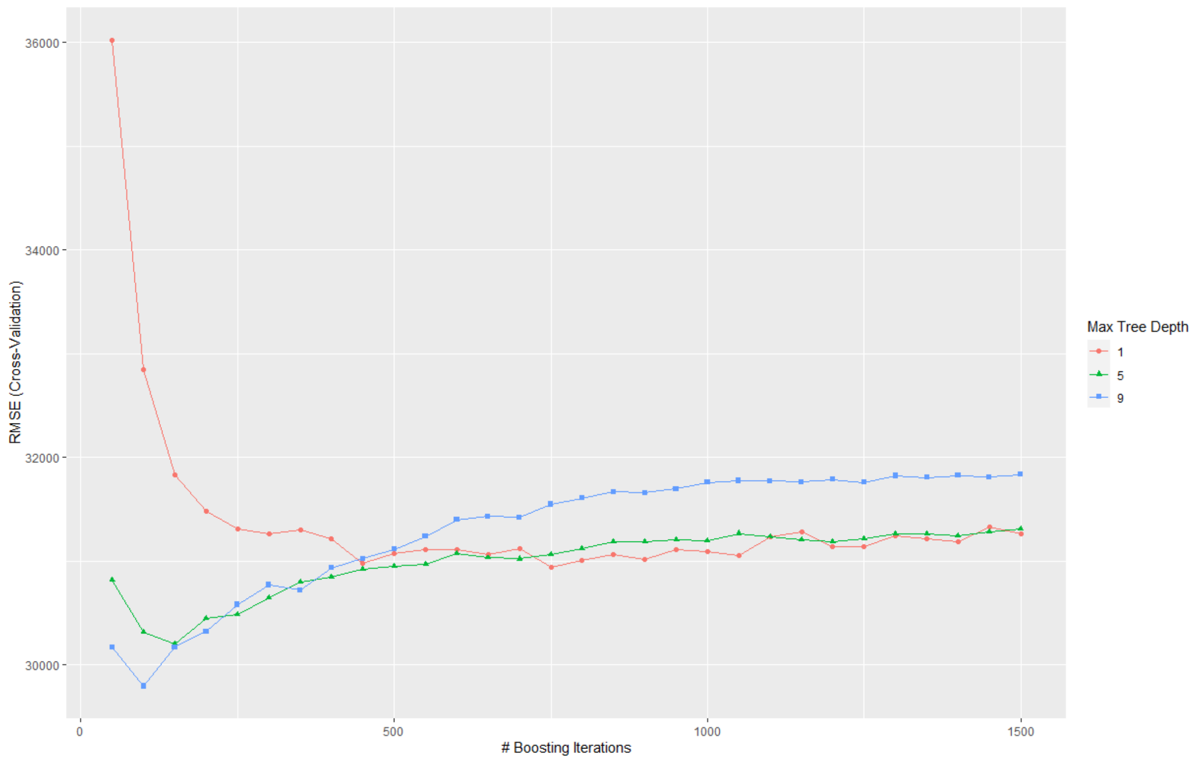
```
> gbmFit1_cv
Stochastic Gradient Boosting

1460 samples
  49 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1314, 1315, 1314, 1315, 1313, 1314, ...
Resampling results across tuning parameters:

  interaction.depth  n.trees  RMSE      Rsquared   MAE
  1                   50      36226.33  0.8063432  23804.11
  1                  100      33769.91  0.8203704  21573.78
  1                  150      32854.59  0.8295881  20800.78
  2                   50      33646.08  0.8231197  21506.69
  2                  100      31560.10  0.8424150  19616.30
  2                  150      31373.90  0.8444784  19238.08
  3                   50      31286.44  0.8471846  20181.50
  3                  100      30122.12  0.8571606  18864.93
  3                  150      29680.43  0.8612641  18309.56

Tuning parameter 'shrinkage' was held constant at a value of 0.1
Tuning parameter 'n.minobsinnode' was held constant at a value of 10
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were n.trees = 150, interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```



Max Tree Depth
1
2
3
37000
36000
35000
34000
33000
32000
31000
RMSE (Bootstrap)
60
80
100
120
140
Boosting Iterations



Max Tree Depth
1
2
3
36000
34000
32000
30000
RMSE (Cross-Validation)
60
80
100
120
140
Boosting Iterations



Support Vector Machine :

```
> svmFit_linear_cv
Support Vector Machines with Linear Kernel

1460 samples
  49 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1314, 1314, 1314, 1315, 1313, 1314, ...
Resampling results:

  RMSE      Rsquared   MAE
  39057.77  0.7594577  22750.04

Tuning parameter 'C' was held constant at a value of 1
> svmFit_linear_boot
Support Vector Machines with Linear Kernel

1460 samples
  49 predictor

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 1460, 1460, 1460, 1460, 1460, 1460, ...
Resampling results:

  RMSE      Rsquared   MAE
  43366.72  0.7049402  25160.48

Tuning parameter 'C' was held constant at a value of 1
```

```
###################Research Question##########################

#load package
library(HSAUR2)
library(ISLR)
library(xtable)
library(randomForest)
############# LOADING DATA ################
data <- read.csv('E:/academics/Applied statistics/house-data.csv',header=TRUE)
############# IMPUTING MISSING VALUES ###############
data$LotFrontage[is.na(data$LotFrontage)] <- median(data$LotFrontage,na.rm = TRUE)
data$MasVnrArea[is.na(data$MasVnrArea)] <- 0
data$Alley[is.na(data$Alley)] <- "No alley access"
data$BsmtCond[is.na(data$BsmtCond)] <- "No Basement"
data$BsmtQual[is.na(data$BsmtQual)] <- "No Basement"
data$GarageCond[is.na(data$GarageCond)] <- "No Garage"
data$GarageType[is.na(data$GarageType)] <- "No Garage"
data$PoolQC[is.na(data$PoolQC)] <- "No Pool"
data$Fence[is.na(data$Fence)] <- "No Fence"
data$MiscFeature[is.na(data$MiscFeature)] <- "None"
############### CONVERTING TO FACTORS ##################
```

```
data$Alley <-  as.factor(data$Alley)

data$BsmtCond <- as.factor(data$BsmtCond)

data$BsmtQual <- as.factor(data$BsmtQual)

data$GarageCond<- as.factor(data$GarageCond)

data$GarageType<- as.factor(data$GarageType)

data$PoolQC<- as.factor(data$PoolQC)

data$Fence<- as.factor(data$Fence)

data$MiscFeature<- as.factor(data$MiscFeature)

data$Street <- as.factor(data$Street)

data$Utilities <- as.factor(data$Utilities)

data$LotConfig <- as.factor(data$LotConfig)

data$Neighborhood <- as.factor(data$Neighborhood)

data$Condition1<- as.factor(data$Condition1)

data$Condition2<- as.factor(data$Condition2)

data$BldgType<- as.factor(data$BldgType)

data$HouseStyle<- as.factor(data$HouseStyle)

data$RoofStyle<- as.factor(data$RoofStyle)

data$RoofMatl <- as.factor(data$RoofMatl)

data$Exterior1st <- as.factor(data$Exterior1st)

data$ExterQual  <- as.factor(data$ExterQual)

data$ExterCond  <- as.factor(data$ExterCond)
```

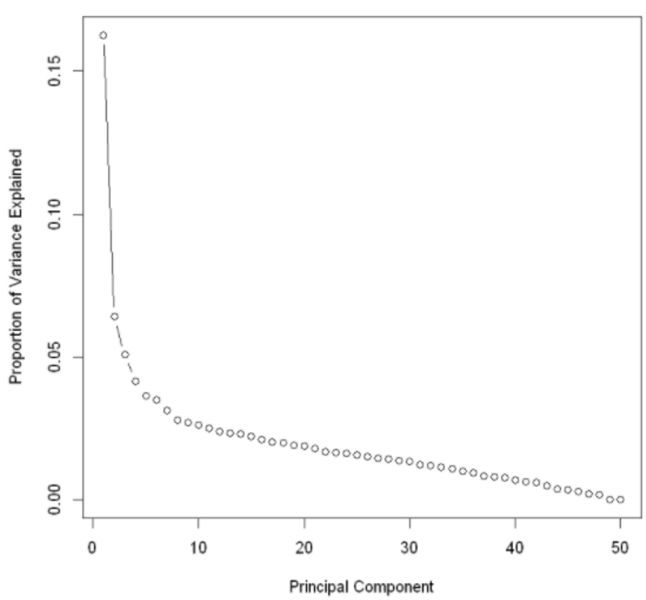
```
data$Foundation <- as.factor(data$Foundation)

data$Heating  <- as.factor(data$Heating)

data$KitchenQual <- as.factor(data$KitchenQual)

data$Functional <- as.factor(data$Functional)

data$PavedDrive <- as.factor(data$PavedDrive)

data$SaleType <- as.factor(data$SaleType)

data$SaleCondition <- as.factor(data$SaleCondition)

############### CONVERTING TO NUMERIC  ##########################


data$Street <- as.numeric(data$Street)

data$Alley <- as.numeric(data$Alley)

data$Fence <- as.numeric(data$Fence)

data$Utilities <- as.numeric(data$Utilities)

data$LotConfig <- as.numeric(data$LotConfig)

data$Neighborhood <- as.numeric(data$Neighborhood)

data$Condition1 <- as.numeric(data$Condition1)

data$Condition2 <- as.numeric(data$Condition2)

data$BldgType <- as.numeric(data$BldgType)

data$HouseStyle <- as.numeric(data$HouseStyle)

data$RoofStyle <- as.numeric(data$RoofStyle)

data$RoofMatl <- as.numeric(data$RoofMatl)
```

```
data$Exterior1st <- as.numeric(data$Exterior1st)

data$ExterQual <- as.numeric(data$ExterQual)


data$ExterCond <- as.numeric(data$ExterCond)

data$Foundation <- as.numeric(data$Foundation)

data$BsmtQual <- as.numeric(data$BsmtQual)

data$BsmtCond <- as.numeric(data$BsmtCond)

data$Heating <- as.numeric(data$Heating)

data$KitchenQual <- as.numeric(data$KitchenQual)

data$Functional <- as.numeric(data$Functional)

data$GarageType <- as.numeric(data$GarageType)

data$GarageCond <- as.numeric(data$GarageCond)

data$PavedDrive <- as.numeric(data$Neighborhood)

data$PoolQC <- as.numeric(data$PoolQC)

data$MiscFeature <- as.numeric(data$MiscFeature)

data$SaleType <- as.numeric(data$SaleType)

data$SaleCondition <- as.numeric(data$SaleCondition)

########################## REMOVING ID COLUMN #########################

data <- subset(data,select=-c(Id))

dim(data)

######################### SPLITTING DATA######################
```

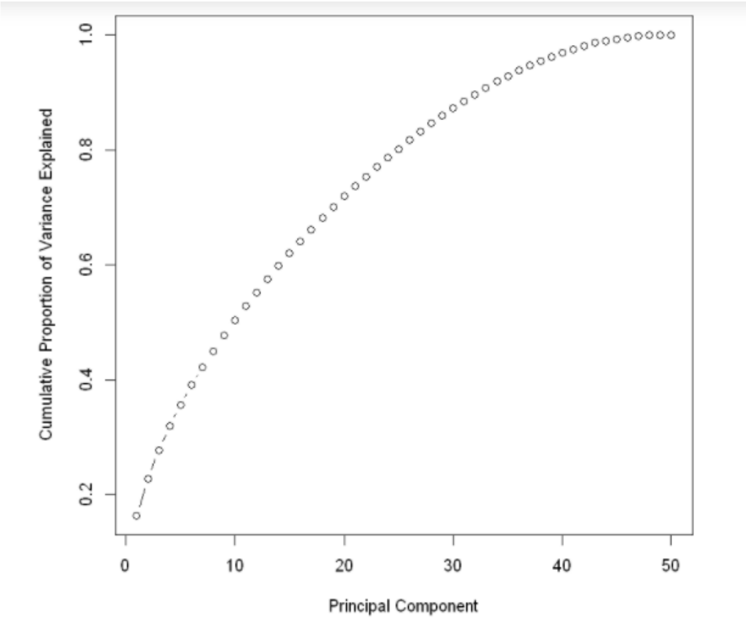
```
set.seed(1)

samp <- sample(nrow(data), nrow(data)*0.75)

house.train <- data[samp,]

house.valid <- data[-samp,]

############################## PCA ###################################

prin_comp <- prcomp(house.train, scale = T)
```

```
set.seed(1)
samp <- sample(nrow(data), nrow(data)*0.75)
house.train <- data[samp,]
house.valid <- data[-samp,]
prin_comp <- prcomp(house.train, scale = T)
```

```
#scree plot

plot(prin_comp, xlab = "Principal Component",

        ylab = "Proportion of Variance Explained",

        type = "b")
```



```
#cumulative scree plot

plot(cumsum(prop_varex), xlab = "Principal Component",

        ylab = "Cumulative Proportion of Variance Explained",

        type = "b")
```

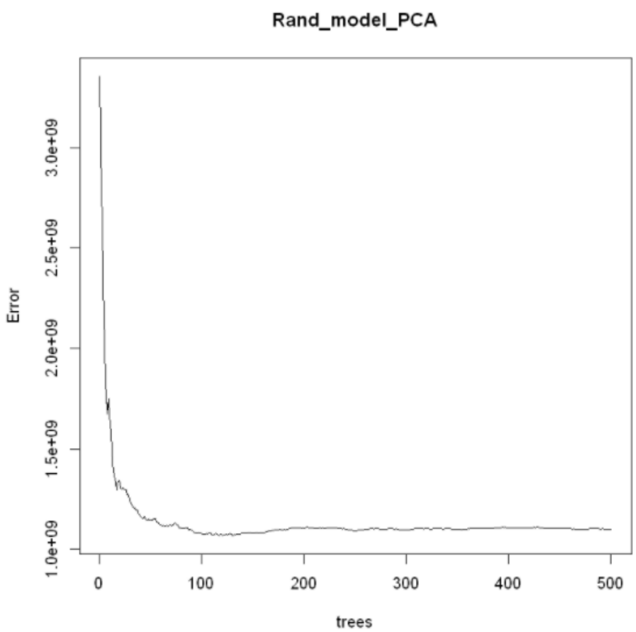



```
############# PRINCIPLE COMPONENTS FOR TRAINING DATA ###############

train.data <- data.frame(SalePrice = house.train$SalePrice, prin_comp$x)

#Selecting first  35 PCAs

train.data <- train.data[,1:35]

##########  BUILDING A RANDOM FOREST MODEL ##################

Rand_model_PCA <- randomForest(SalePrice~.,

                data = train.data,

                importance=TRUE)
```

Rand_model_PCA

```
> Rand_model_PCA

Call:
 randomForest(formula = SalePrice ~ ., data = train.data, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 11

          Mean of squared residuals: 1101610983
                    % Var explained: 82.23
```

plot(Rand_model_PCA)



############PRINCIPLE COMPONENTS FOR VALIDATION DATA ################

test.data <- predict(prin_comp,newdata=house.valid)

SELECTING FIRST 35 PCA's

test.data <- test.data[,1:35]

################### PREDICTING ON VALIDATION SET #######################

Predicted_Sale_Prices_forest_PCA <- predict(Rand_model_PCA, newdata=test.data)

```
################### CALCULATING R2 SCORE###########################

mean_sale_price_house <- mean(house.valid$SalePrice)

R2_score <- function(predicted) { 1 - (sum((house.valid$SalePrice-predicted)^2)/sum((house.valid$SalePrice-mean_sale_price_house)^2))}

R2_Score_Rand_Forest <- R2_score(Predicted_Sale_Prices_forest_PCA)

R2_Score_Rand_Forest
```