3/26/2021

Ademayowa Omole, Hamid Kabir, Dushyanth Bommana, Shagufta Khan, Midhun Molekudiyil, Sruthi Belaganti

University of Essex

MA321 Group 18 Coursework

Contents

Abstract…………………………………………………………………………………………………………………………………………….3

Introduction………………………………………………………………………………………………………………………………………4

Question 1…………………………………………………………………………………………………………………………………………4

Question 2…………………………………………………………………………………………………………………………………………4

Question 3…………………………………………………………………………………………………………………………………………7

Question 4……………………………………………………………………………………………………………………………………….12

Appendix …………………………………………………………………………………………………………………………………………13

Word Count: 2615

**Abstract**

Every year, there is an increase in the price of the houses. This situation has created a need for a system which predicts the price and condition of a house based on the year in which it was built, number of rooms and other factors. This system could also assist the people who want to purchase a house by analysing various aspects of the house. This analysis aims to predict the overall condition(Poor, Average, Good) of a house based on the rating given using different classification algorithms. The result from this analysis proved that the logistic regression model built to predict the overall condition predicted the target with 97% accuracy and the random forest classifier predicted with an accuracy of 98.29%.

**Statement of Contributions:**

Question 1: Midhun Molekudiyil

Question 2a: Sruthi Belaganti

Question 2b: Dushyanth Bommana

Question 3a: Shagufta Khan

Question 3b: Hamid Kabir

Question 4: Ademayowa Omole

Merging and Editing: Ademayowa Omole

Introduction

The dataset given for the analysis consists of 1460 observations and 51 variables which describe the overall condition of a house depending on various factors such as lot frontage, year built, sale price, number of rooms etc.

1.

We can load data into R by using function read.csv() or by importing directly by tab. Head() and Tail () function returns the first or last parts of a vector, matrix, table, data frame or function. Summary() is a generic function used to produce result summaries of the results of various model fitting functions. Str() function is used for compactly displaying the internal structure of a R object. It can display even the internal structure of large lists which are nested. Colnames() retrieve or set the row or column names of a matrix-like object. Retrieve or set the row or column names of a matrix-like object. ggplot() is used to construct the initial plot object and is almost always followed by + to add component to the plot.

2.

The problem statement requires to classify the overall condition of a house as Poor, Good, Average based on the rating from 1 to 10. Classification is a technique for classifying data into a set of categories. The principal purpose of a classification problem is to determine which category or class new data will belong to.

For the questions 2) a & b same dataset is used to implement different classification algorithms. So, data cleaning is carried out to remove missing values from the given dataset. The following procedure is implemented to eliminate the missing values present in the data.

·       The dataset is loaded into R using read.csv () function. The libraries like dplyr, mice which are useful in building a model are loaded into R.

·       The data set is checked for missing values. Missing values cause certain problems on the decisions taken on analysing the data. However, excluding missing data from the evaluation causes loss of data and results in inaccurate conclusions. The ideal way to deal with missing values is to impute them with certain values which complete the dataset without disturbing the original structure.

·       In this analysis, MICE (Multivariate Imputation by Chained Equations) imputation technique is used to replace the missing data. This feature automatically detects the columns with missing values and imputes them accordingly using methods such as predictive mean matching, logistic regression, depending on type of the variable.

2a) The logistic regressor is fit on the given dataset as follows:

·       Logistic regression is a classification algorithm that is used to predict a binary outcome (either the event occurs or does not occur) in accordance with a set of independent variables. There are three outcomes to the dependent variables, therefore multinomial logistic regression can be applied on the data to classify the overall condition of a house. Multinomial logistic regression is an extension of binary logistic regression which enables the prediction of more than two classes of the target variable.

·       The dataset is divided into training and test sets on the overall condition variable, the model is trained on the training data and predictions are made on the test data.

·       Multinom () function from the nnet package is used to build the model on training data. Using the relevel () function, one of the levels of the target variable (Average) is set as a baseline.

·       The summary of the model describes the coefficients for Poor and Good conditions in comparison to the baseline Average.

·       The predictions of the model are calculated on the training data to check the efficiency of the model. In this analysis sensitivity of the confusion matrix is considered as a metric of evaluating the model’s performance. Sensitivity is the percentage of positives found that are expected to be positive. The sensitivity of the model for all the three levels (Poor, Average & Good) is found to be 1 which implies the model predicted with 100% accuracy on the training data.

·       The same model is applied to the test data and the results observed are as follows. The sensitivity of the class average given by the model is 99% which indicates that the model predicted the Average value accurately around 99% of the time. Likewise, the model predicted the class Good correctly 96% of the time.

·       However, the model did not classify the class Poor accurately most of the time. The sensitivity value is 0.66 which means the model predicted the class poor precisely only 66% of the time.

• The overall accuracy of the multinomial logistic regression model is 97%.

2b)The Random Forest logistic regression is fit on the given dataset as follows.

• Based on the similar study, we are using Random forest classification method to classify the house condition.

• Random forest classification creates multiple decision trees and combines them to form a forest which gives precise and dependable prediction.

• Multiple trees are created using bootstrap samples from training data, and the correlation between the trees is minimized. This method enhances the efficiency of decision trees and prevents overfitting problem.

• On the overall condition variable, the dataset is split into training and test sets, the model is trained on the training data, and predictions are made on the test data. In this analysis we are using 80% of the data as training data and 20% as test data.

• Initially the Random forest classifier is fit on the training dataset with the default parameters, random Forest () is used to build the classifier on the data.

• A confusion matrix is built with the predictions on training data, the sensitivity metric for all the three classes, Average, Good, Poor is observed to be 1. This indicates that the model is built with 100% efficiency on the training data.

• The model is applied to the test data to check the efficiency on unseen data and a confusion matrix is built to check the sensitivity metric. The observed values of sensitivity for Good and Average classes is 1 which implies that the model predicted with 100% accuracy on observation with Average and Good classes.

• The sensitivity for Poor class is 0.16 which means the model predicted the class Poor accurately 16% of the time.

• The overall accuracy of the random forest classifier is 98.29%.

• When the number of trees is reduced to 300, same results are obtained as of 500 trees.

Conclusion for 2) a & b question.

Based on the above classification techniques used, The sensitivity metric for logistic regression and random forest classifier is compared and it is observed that random forest classifier predicted the Good and Average classes more accurately when compared to logistic regression. While both the models’ predictions for the class Poor is better in the logistic regression model.

**3a**.

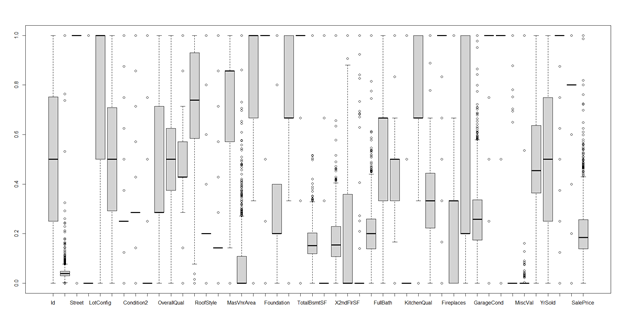
Data analysis and pre-processing:

We are provided with a house price dataset that contains 1640 observations (row) and 51 features (columns). There are 28 categorical columns and 23 numeric columns. We looked for the count of null values in each column in R.

We could see some columns have a very large number of null values so we decided to remove those columns because they would provide very little or no information at the time of training. We retained only those columns that have 85% of the not-null values. After removing columns we dropped rows that contain null values in any observation.

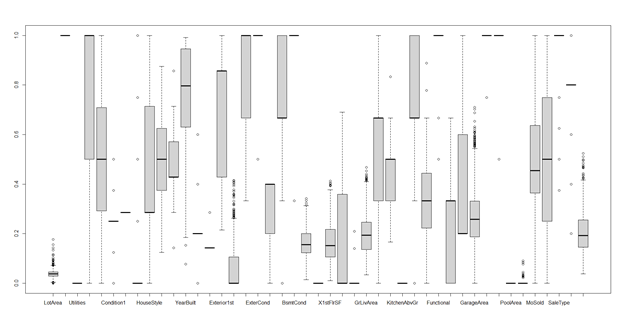
Columns in the house price dataset have different ranges so we decided to standardize our dataset and plot a box plot to analyze if there are any outliers in the dataset.

The box plot of our dataset could be shown as:



Figure

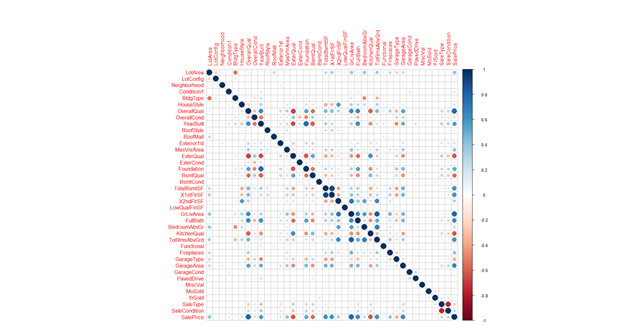
We could see there are outliers present in the dataset so we decided to remove outliers using the z-score method (remove observations with z-score>3) and plot the box plot again.



Figure

We investigated the variance in columns and found out there are some columns with zero variation so we removed those columns.

The correlation between columns can be shown in the following plot.



Figure

We could see that the columns "OverallQual", "GrLivArea",   "GarageArea",  "FullBath" could play an important role in predicting “SalePrice”.

Model Training:

Linear Model:

We split the dataset into training and testing. We used caret package for splitting data into training and testing sets. We use 80% of the data for training and 20% for testing.

We started with training a linear model on our dataset. We used 10-fold cross validation to evaluate performance of our model.

We evaluated the performance of our linear model on a test dataset.

Random Forest:

“Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees.” Wikipedia

We trained a random forest model on our preprocessed dataset and with 10-fold cross validation checked our RMSE and R-squared value for different values of mtry i.e. number of predictors to be used at the time of split.

We found out the value of 9 for mtry to produce the lowest RMSE and highest R-squared value. We trained our random forest model with this parameter value and evaluated result on test dataset as follows:

The RMSE value is low because we standardized our data at the time of preprocessing.

Whereas random forests build an ensemble of deep independent trees, GBMs build an ensemble of shallow and weak successive trees with each tree learning and improving on the previous. Many weak successive trees after combining produce powerful features.  We trained a generalized boosted model on our dataset with “gaussian” distribution and 100 trees and multiple interaction depths with 10-fold cross validation. The results are as follows:

We trained models on 9 interaction depths and evaluated results on a test dataset.

We can see that with Gradient Boosting we have r-squared value of 0.871 with cross validation and 0.895 on testing dataset so we decided to use GBM model for our dataset.

**3b**.

The output shows the average model performance across the 10 resamples and bootstrapped. RMSE (Root Mean Squared Error) and MAE(Mean Absolute Error), represent two different measures of the model prediction error. The lower the RMSE and the MAE, the better the model. The R-squared represents the proportion of variation in the outcome explained by the predictor variables included in the model. The higher the R-squared, the better the model.

Random Forest using 10 fold cross validation resampling method:

RMSE was used to select the optimal model using the smallest value.

Mtry: Number of variables is randomly collected to be sampled at each split time.

We have  root mean square error (RMSE), mean absolute error (MAE),  and  R squared for three iterations. The first time it selected 2 variables randomly and performed the split, for this RMSE = 0.414, R Squared = 0.861  and MAE = 0.247.

For the second time iteration it selected 21 variables randomly and performed the split, for this RMSE = 0.389, R Squared = 0.859 and MAE = 0.227.

For the third time iteration it selected 40 variables randomly and performed the split, for this RMSE = 0.401, R Squared = 0.849 and MAE = 0.235.

So the final value used for the model was mtry = 21.

Random Forest using Bootstrapping resampling method:

RMSE was used to select the optimal model using the smallest value.

Mtry: Number of variables is randomly collected to be sampled at each split time.

We have  root mean square error (RMSE), mean absolute error (MAE),  and  R squared for three iterations. The first time it selected 2 variables randomly and performed the split, for this RMSE = 0.418, R Squared = 0.852  and MAE = 0.250.

For the second time iteration it selected 21 variables randomly and performed the split, for this RMSE = 0.386, R Squared = 0.853 and MAE = 0.230.

For the third time iteration it selected 40 variables randomly and performed the split, for this RMSE = 0.398, R Squared = 0.842 and MAE = 0.239.

So the final value used for the model was mtry = 21.

GMB using 10 fold cross validation resampling method:

RMSE was used to select the optimal model using the smallest value.

ntree: Number of branches will grow after each time split.

The final values used for the model were n.trees = 150, interaction.depth = 3, shrinkage

 = 0.1 and n.minobsinnode = 10. So for this RMSE = 0.378, R Squared = 0.854 and MAE = 0.229.

GMB using bootstrapping resampling method:

RMSE was used to select the optimal model using the smallest value.

ntree: Number of branches will grow after each time split.

The final values used for the model were n.trees = 150, interaction.depth = 3, shrinkage

 = 0.1 and n.minobsinnode = 10. So for this RMSE = 0.406, R Squared = 0.835 and MAE = 0.236.

4.

Using the dataset we can investigate whether it is possible to predict the year a house is built. We can also investigate the relationship between the year a house is built with the overall quality and overall condition of a house. This information can be useful because if it is revealed that the variable ‘YearBuilt’ has a positive relationship with some of the negative features e.g. a poor ‘OverallQual’ score then this could be something that the government would want to look into to resolve this issue. Our initial hypothesis is that the later a house is built, the better the overall condition and overall quality.

As the dataset has a large number of variables, there will be some which provide little information in regard to our problem. To solve this we use the sapply() function twice to find the columns with a large number of missing values and the columns with little variance so we can remove them. We can now use the cor() function to get the correlation coefficient between all the variables. Regarding the variable ‘YearBuilt’, there is only one other variable in which the correlation coefficient is greater than 0.6 which is ‘foundation’ at 0.63. This would suggest that it would be extremely difficult to accurately predict the YearBuilt variable because none of the other variables share a strong positive correlation with it. If we compare this with a variable which is possible to predict in ‘SalePrice’. We can see that SalePrice has four other variables with a correlation coefficient over 0.6 and two over 0.7. We can also use the ggplot function to visualise the graphs between YearBuilt against OverallQual and OverallCond to see that there is no significant relationship disproving our initial hypothesis. These graphs are shown in the appendix.

**Appendix**

house\_data\_2=read.csv("https://raw.githubusercontent.com/sruthireddy1482/MA321\_GRP18/main/house-data.csv",stringsAsFactors = T) #reading csv file

attach(house.data)

library(tidyverse)

head(house.data)

tail(house.data)

summary(house.data)

ggplot(data = house.data) +

  geom\_point(mapping = aes(x = V13, y = V14), color="blue")

ggplot(data = house.data) +

  geom\_point(mapping = aes(x = V48, y = V25), color="blue")

library(dplyr)

library(ggplot2)

library(purrr)

library(caTools)

library(nnet)

library(mice)

library(caret)

library(data.table)

library(car)

library(MASS)

library(randomForest)

set.seed(123) #we need to set seed to achieve same results everytime

house\_data\_2=read.csv("https://raw.githubusercontent.com/sruthireddy1482/MA321\_GRP18/main/house-data.csv",stringsAsFactors = T) #reading csv file

View(house\_data\_2)

dim(house\_data\_2)

#finding missing values in each column

map(house\_data\_2, ~sum(is.na(.)))

str(house\_data\_2)

house\_data\_2 <- house\_data\_2[-c(945),]

#dividing house based on their overall condition

house\_data\_2$Overallcond <- ifelse(house\_data\_2$OverallCond>=1 & house\_data\_2$OverallCond <= 3, "Poor", "Good")

house\_data\_2$Overallcond <- ifelse(house\_data\_2$OverallCond >= 4 & house\_data\_2$OverallCond <= 6, "Average", house\_data\_2$Overallcond)

house\_data\_2$Overallcond <- ifelse(house\_data\_2$OverallCond >= 7 & house\_data\_2$OverallCond <= 10, "Good", house\_data\_2$Overallcond)

#viewing the overall condition values

table(house\_data\_2$Overallcond)

house\_data\_2$Overallcond=as.factor(house\_data\_2$Overallcond)

#removing columns with almost all missing values

data\_2 <- subset(house\_data\_2, select = -c(Alley, Fence,PoolQC,MiscFeature))

View(data\_2)

#imputing missing values of all the columns in te dataset

impute\_2= mice(data\_2[,c(2,18,22,23,37,39)])

print(impute\_2)

#impute\_2$imp$LotFrontage

new\_data\_2=complete(impute\_2,1)

new\_data\_2

#replacing columns with mising values with imputed columns

data\_2[,c(2,18,22,23,37,39)]=new\_data\_2[,c(1,2,3,4,5,6)]

sum(is.na(data\_2))

summary(data\_2)

#Question 2(a)

#splitting the dataset into training and test sets

split\_2a= sample.split(data\_2$Overallcond, SplitRatio = 0.8)

training\_set\_2a=subset(data\_2,split\_2a==TRUE)

test\_set\_2a=subset(data\_2,split\_2a==FALSE)

#fitting multinomial logistic regression on training data

training\_set\_2a$Overallcond <- relevel(training\_set\_2a$Overallcond, ref = "Average")

multinom.fit\_2a <- multinom(Overallcond~., data = training\_set\_2a)

summary(multinom.fit\_2a)

#head(probability.table <- fitted(multinom.fit\_2a))

#predicting on training data

pred\_2a <- predict(multinom.fit\_2a, newdata = training\_set\_2a, "class")

#pred\_2a

confusionMatrix(pred\_2a, training\_set\_2a$Overallcond)

#predicting on test data

pred2\_2a= predict(multinom.fit\_2a, newdata = test\_set\_2a, "class")

#pred2\_2a

cm\_2a=confusionMatrix(pred2\_2a, test\_set\_2a$Overallcond)

cm\_2a

#Question 2(b)

#Fitting Random forest model.

# sample.split() function is used to split data for training and test.

data\_2b <- copy(data\_2)

split\_2b= sample.split(data\_2b$Overallcond, SplitRatio = 0.8)

training\_set\_2b=subset(data\_2b,split\_2b == TRUE)

test\_set\_2b=subset(data\_2b,split\_2b == FALSE)

#Random forest on training data set

# by default ntree value is 500

rf\_2b <- randomForest(Overallcond~., data=training\_set\_2b)

#To achieve more accurate results, we passed ntree vales =300 to check the oob estimate of error

#rf\_2b <- randomForest(Overallcond~., data=training\_set\_2b,ntree=300)

print(rf\_2b)

attributes(rf\_2b)

prediction1\_2b <- predict(rf\_2b, training\_set\_2b)

#prediction1\_2b

confusionMatrix(prediction1\_2b, training\_set\_2b$Overallcond)

#Now lets predict test data

#Prediction & Confusion Matrix - test data

prediction2\_2b <- predict(rf\_2b, test\_set\_2b)

#prediction2\_2b

confusionMatrix(prediction2\_2b, test\_set\_2b$Overallcond)

############ PART3 (Data Analysis and Preprocessing) - START #################

df <- read.csv("https://raw.githubusercontent.com/sruthireddy1482/MA321\_GRP18/main/house-data.csv", header = TRUE)

######## Dataset summary and dimensions

summary(df)

ncol(df)

nrow(df)

####### Investigating column types (int/char) or numeric/categorical

str(df)

####### Investigating null values in all columns

sapply(df, function(x) sum(is.na(x)))

###### Loading plotting libraries

library(corrplot)

library(ggplot2)

###### Converting categorical columns to numeric

df[] <- data.matrix(df)]]

###### Retaining columns with 85% non-null values and dropping others

ncol(df)

df <- df[, which(colMeans(!is.na(df)) > 0.85)]

ncol(df)

##### Loading library for data manipulation

library(tidyr)

###### Dropping rows containing any null values.

nrow(df)

df <- drop\_na(df)

nrow(df)

###### Standardizing columns

for(name in names(df)){

df[name] <- stdize(df[name])

}

stdize <- function(x, ...) {(x - min(x, ...)) / (max(x, ...) - min(x, ...))}

###### Plotting box plot for investigating outliers

boxplot(df)

##### Removing outliers based on z-score (z-score greater than 3)

z\_scores <- as.data.frame(sapply(df, function(df) (abs(df-mean(df))/sd(df))))

z\_scores$Id <- df$Id

no\_outliers <- z\_scores[!rowSums(z\_scores[, !names(df) %in% c("Id")]>3), ]

df <- df[df$Id %in% no\_outliers$Id,]

###### Dropping Id column because it is of no use now

drop <- c("Id")

df <- df[,!names(df) %in% drop]

###### Plotting boxplot again

boxplot(df)

####### Checking for columns with zero variance

sapply(df, var)

####### Removing columns with zero variation

drops <- c("Street","Utilities","Condition2","Heating","KitchenAbvGr","PoolArea")

df<-df[ , !(names(df) %in% drops)]

###### Investigating correlation between variables

cc = cor(df)

corrplot(cc)

round(cor(df), 2)

###### Investigating effect of predictor variables on response variable.

saleprice\_cor <- sapply(df[,!(names(df) %in% c("SalePrice"))], function(x) cor(x, df$SalePrice))

saleprice\_cor <- sort(saleprice\_cor,decreasing=TRUE)

saleprice\_cor <- saleprice\_cor[saleprice\_cor>0.6]

names(saleprice\_cor)

summary(df)

############ PART 3 (Data Analysis and Preprocessing) - END #################

############ PART 3 (Model Training and Evaluation) - START #################

##### Importing libraries for model training

library(caret) # for general model fitting

library(randomForest)

library(gbm)

####### Splitting data into training and testing sets

set.seed(642)

training.samples <- createDataPartition(df$SalePrice, p = .8,

list = FALSE,

times = 1)

train.df <- df[training.samples, ]

test.df <- df[-training.samples, ]

###### Fitting random forest with different values of mtry i.e. number of predictor variables at each split and cross validation

mtry <- c(9,18,27,38)

train.control <- trainControl(method = "cv", number = 10)

tunegrid <- expand.grid(.mtry=mtry)

model1\_default <- train(SalePrice ~ ., data = train.df, method = "rf", metric="RMSE",tuneGrid=tunegrid,trControl = train.control,ntree = 100,verbose = TRUE)

###### Evaluating model performance

print(model1\_default)

##### Fitting model on selected parameters and evaluating performance on test dataset

model1 = randomForest(SalePrice ~ ., data = train.df, mtry = 9,

importance = TRUE, ntrees = 100)

predictions <-predict(model1,test.df)

RMSE(predictions, test.df$SalePrice)

R2(predictions, test.df$SalePrice)

###### Fitting gradient boosting with different values of interaction depths on train set

train.control <- trainControl(method = "cv", number = 10)

tunegrid = expand.grid(interaction.depth = c(1,3,6,9),

n.trees = 100,

shrinkage = 0.1,

n.minobsinnode = 10)

model2\_default <- train(SalePrice ~ ., data = train.df, method = "gbm", metric="RMSE",distribution="gaussian",tuneGrid=tunegrid,trControl = train.control,verbose = FALSE)

###### Evaluating model performance

print(model2\_default)

##### Fitting model on selected parameters and evaluating performance on test dataset

model2 = gbm(SalePrice ~ ., data = train.df, distribution = "gaussian",n.trees = 100,interaction.depth = 9)

predictions <- predict(model2,test.df)

RMSE(predictions, test.df$SalePrice)

R2(predictions, test.df$SalePrice)

############ PART 3 (Model Training and Evaluation) - END #################

# Loading necessary packages

# Contains the functions to streamline the model training process for

complex classification and regression problems

library(caret)

library(randomForest)

# Random Forest model:

train.control.cv <- trainControl(method = "cv", number = 10)

# Train the model

model\_rf\_cv <- train(SalePrice ~., data = df, method = "rf",

            trControl = train.control.cv)

# Summarize the results

print(model\_rf\_cv)

 # Estimating test error with boot strapping on random Forest Model

train.control.boot <- trainControl(method = "boot")

# Train the model

model\_rf\_boot <- train(SalePrice ~., data = df, method = "rf",

            trControl = train.control.boot)

# Summarize the results

print(model\_rf\_boot)

 # Estimating test error with cross validation on GBM Model

library(caret)

library(randomForest)

train.control.cv <- trainControl(method = "cv", number = 10)

# Train the model

model\_rf\_cv <- train(SalePrice ~., data = df, method = "gbm",

                  trControl = train.control.cv)

# Summarize the results

print(model\_rf\_cv)

 # Estimating test error with boot strapping on GBM Model

train.control.boot <- trainControl(method = "boot")

# Train the model

model\_rf\_boot <- train(SalePrice ~., data = df, method = "gbm",

                    trControl = train.control.boot)

# Summarize the results

print(model\_rf\_boot)

sapply(mydata, function(x) sum(is.na(x))

sapply(mydata, var)

mydata = mydata[, which(colMeans(!is.na(mydata)) > 0.85)]

mydata = subset(mydata, select = -c(GarageType,BsmtQual,GarageCond) )

res = cor(mydata)

round(res, 2)

ggplot(data = mydata) +

geom\_point(mapping = aes(x = YearBuilt, y = OverallQual), color="blue")

ggplot(data = mydata) +

geom\_point(mapping = aes(x = YearBuilt, y = OverallCond), color="blue")

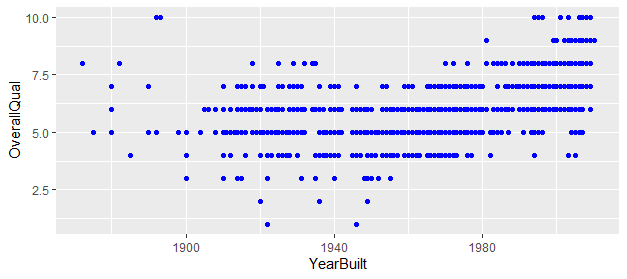


Figure showing the relationship between YearBuilt and OverallQual.

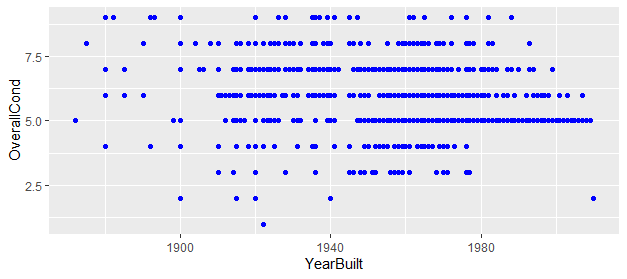


Figure 5 showing the relationship between YearBuilt and OverallCond.