install.packages("Boruta", repos = "http://cran.us.r-project.org")

## Installing package into 'C:/Users/Rahul/Documents/R/win-library/3.3'  
## (as 'lib' is unspecified)

## package 'Boruta' successfully unpacked and MD5 sums checked  
##   
## The downloaded binary packages are in  
## C:\Users\Rahul\AppData\Local\Temp\Rtmp8sa4W7\downloaded\_packages

library(Boruta)

## Loading required package: ranger

## read the file and remove unnecessary fields

houseData <- read.csv("file:///G:/Ryerson-BigData/capstone-R/CKME-136/data/kc\_house\_data.csv")  
houseData$date<-NULL  
houseData$id<-NULL  
colnames(houseData)

## [1] "price" "bedrooms" "bathrooms" "sqft\_living"   
## [5] "sqft\_lot" "floors" "waterfront" "view"   
## [9] "condition" "grade" "sqft\_above" "sqft\_basement"  
## [13] "yr\_built" "yr\_renovated" "zipcode" "lat"   
## [17] "long" "sqft\_living15" "sqft\_lot15"

# USING BORUTA for features selection and finally making model with the selected attributes

set.seed(1)  
boruta.train <- Boruta(price ~., data = houseData , doTrace = 2,ntree = 500)

## 1. run of importance source...

## 2. run of importance source...

## 3. run of importance source...

## 4. run of importance source...

## 5. run of importance source...

## 6. run of importance source...

## Computing permutation importance.. Progress: 98%. Estimated remaining time: 0 seconds.

## 7. run of importance source...

## 8. run of importance source...

## 9. run of importance source...

## 10. run of importance source...

## 11. run of importance source...

## After 11 iterations, +9.1 mins:

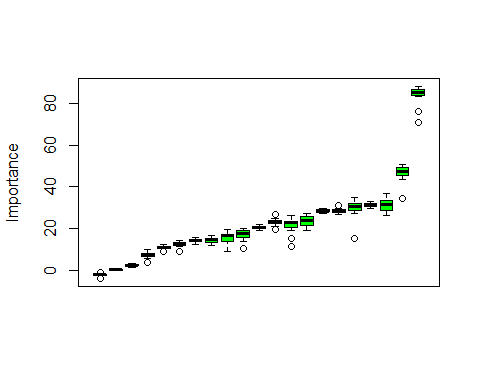
## confirmed 18 attributes: bathrooms, bedrooms, condition, floors, grade and 13 more;

## no more attributes left.

print(boruta.train)

## Boruta performed 11 iterations in 9.149673 mins.  
## 18 attributes confirmed important: bathrooms, bedrooms,  
## condition, floors, grade and 13 more;  
## No attributes deemed unimportant.

plot(boruta.train, xlab = "", xaxt = "n")



Boruta.Short <- Boruta(price ~ ., data = houseData, maxRuns = 12)

# Start calculating RMSE

newhouseData <- houseData  
  
set.seed(1)  
i=0.6  
storage <- list(c(), c(), c(),c())  
for(i in seq(from=0.6, to=0.9, by=0.01)){  
 rn\_train <- sample(nrow(newhouseData),floor(nrow(newhouseData)\*i))  
 train <- newhouseData[rn\_train,colnames(newhouseData)]  
 test <- newhouseData[-rn\_train,colnames(newhouseData)]  
 model <- lm(formula = price~.,data=train)  
 prediction <- predict(model,interval='prediction',newdata = test)  
 train\_prediction = fitted(model)  
 train\_rmse = sqrt(sum((train\_prediction-train$price)^2)/nrow(train))  
 test\_rmse = sqrt(sum((prediction - test$price)^2)/nrow(test))  
 storage[[1]]<-c(storage[[1]],i)  
 storage[[2]]<-c(storage[[2]],test\_rmse)  
 storage[[3]]<-c(storage[[3]],train\_rmse)  
}

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

##find the LM with minimun training error  
RMSE = storage[[3]]  
minimumVal = min(RMSE)  
minimumVal

## [1] 195557.8

indx = which(RMSE==min(RMSE))  
indx

## [1] 14

storage[[1]][indx]

## [1] 0.73

cat("\nMinimum Training RMSE of Regression:",storage[[3]][indx],"\nRMSE of testing :",storage[[2]][indx], "\nTraining data Percentage:",storage[[1]][indx])

##   
## Minimum Training RMSE of Regression: 195557.8   
## RMSE of testing : 659738.3   
## Training data Percentage: 0.73

## Now we come to a conclusion that 73% Training data provides the Minimum RMSE

## 1: SET training Data = 73% &

## 2: Get model with coeeficient & Intercept

## 3: Draw the error of Histogram to get confidence with model

## 4: Find out how many data have less than 25% of error

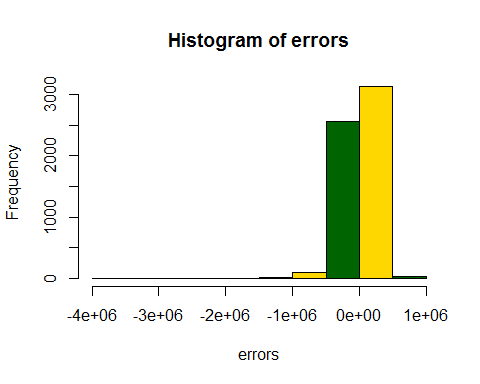
rn\_train <- sample(nrow(newhouseData),floor(nrow(newhouseData)\*storage[[1]][indx]))  
train <- newhouseData[rn\_train,colnames(newhouseData)]  
test <- newhouseData[-rn\_train,colnames(newhouseData)]  
modelXGen <- lm(price~.,data = train)  
summary(modelXGen)

##   
## Call:  
## lm(formula = price ~ ., data = train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1243266 -98314 -8876 76717 4351597   
##   
## Coefficients: (1 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 8.575e+06 3.365e+06 2.548 0.0108 \*   
## bedrooms -3.862e+04 2.267e+03 -17.037 < 2e-16 \*\*\*  
## bathrooms 4.406e+04 3.736e+03 11.794 < 2e-16 \*\*\*  
## sqft\_living 1.498e+02 5.064e+00 29.577 < 2e-16 \*\*\*  
## sqft\_lot 1.088e-01 5.228e-02 2.081 0.0374 \*   
## floors 9.588e+03 4.130e+03 2.321 0.0203 \*   
## waterfront 5.255e+05 1.972e+04 26.649 < 2e-16 \*\*\*  
## view 5.776e+04 2.477e+03 23.317 < 2e-16 \*\*\*  
## condition 2.700e+04 2.701e+03 9.997 < 2e-16 \*\*\*  
## grade 9.567e+04 2.473e+03 38.693 < 2e-16 \*\*\*  
## sqft\_above 2.521e+01 5.033e+00 5.009 5.52e-07 \*\*\*  
## sqft\_basement NA NA NA NA   
## yr\_built -2.650e+03 8.364e+01 -31.682 < 2e-16 \*\*\*  
## yr\_renovated 2.013e+01 4.226e+00 4.764 1.91e-06 \*\*\*  
## zipcode -5.832e+02 3.794e+01 -15.370 < 2e-16 \*\*\*  
## lat 5.981e+05 1.239e+04 48.276 < 2e-16 \*\*\*  
## long -2.023e+05 1.504e+04 -13.446 < 2e-16 \*\*\*  
## sqft\_living15 2.133e+01 3.962e+00 5.384 7.38e-08 \*\*\*  
## sqft\_lot15 -3.208e-01 8.134e-02 -3.944 8.06e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 197700 on 15759 degrees of freedom  
## Multiple R-squared: 0.7011, Adjusted R-squared: 0.7007   
## F-statistic: 2174 on 17 and 15759 DF, p-value: < 2.2e-16

predictionXGen <- predict(modelXGen,interval='prediction',newdata = test)

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

test\_rmseXGen = sqrt(sum((predictionXGen - test$price)^2)/nrow(test))  
errors <- predictionXGen[,'fit'] - test$price  
hist(errors,col=(c("gold","darkgreen")))



rel\_change = 1 - ((test$price - abs(errors)) / test$price)  
##Now the percentage of cases with less than 25% error.  
pred25 = table(rel\_change<0.25)["TRUE"] / nrow(test)  
pred25

## TRUE   
## 0.6094928

cat("\nConclusion:percent of data having less than 25% error:",pred25)

##   
## Conclusion:percent of data having less than 25% error: 0.6094928