ST301_s16_315

s16315

10/12/2020

Introduction

A real estate agent wants to develop a model to predict the sale price of a house using data collected from 5575 sold houses. He assumes that the following variables which may useful in predicting the sale price of a given house.

1. soldPrice - sold price of house
2. sqftLiving - square footage of living area
3. sqftLand - square footage of land
4. sqftAbove - square footage of area above ground
5. sqftBasement - square footage of basement
6. numBedRooms - number of bed rooms
7. numBathRooms - number of bath rooms
8. numFloors - number of floors
9. builtYear - year of construction
10. grade - construction quality ranked from 1 to 4 where 1 is the lowest grade
11. waterFront- whether the house has a waterfront (1) or not (0)
12. condition - condition of the house (Excellent, Good, Average)

Import the data set.

```
housePrice <- read.csv("D:/3rd_year/ST301/housePrice.csv")
```

The data set

```
head(housePrice)
```

```
##   soldPrice sqftLiving sqftLand sqftAbove sqftBasement numBedRooms numBathRooms
## 1    221900       1180     5650      1180            0           3            1
## 2    180000        770    10000       770            0           2            1
## 3    604000       1960     5000      1050          910           4            3
## 4    510000       1680     8080      1680            0           3            2
## 5    229500       1780     7470      1050          730           3            1
```

```
## 6    468000       1160     6000       860          300           2            1
##   numFloors builtYear grade waterFront condition
## 1         1      1955     2          0   average
## 2         1      1933     1          0   average
## 3         1      1965     2          0 excellent
## 4         1      1987     2          0   average
## 5         1      1960     2          0   average
## 6         1      1942     2          0      good
```

Check for the missing values:

```
sum(is.na(housePrice)==TRUE)
```

```
## [1] 0
```

This implies that there are no missing values. Next we have to check for duplicate values.

Checking for duplicate values:

```
nrow(housePrice)-nrow(unique(housePrice))
```

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

This implies there is a one duplicate value.

Removing the duplicate values:

```
new_housePrice=unique(housePrice)
```

The structure of the data set:

```
str(new_housePrice)
```

```
## 'data.frame':    5574 obs. of  12 variables:
##  $ soldPrice   : num  221900 180000 604000 510000 229500 ...
##  $ sqftLiving  : int  1180 770 1960 1680 1780 1160 2950 1890 1200 1250 ...
##  $ sqftLand    : int  5650 10000 5000 8080 7470 6000 5000 14040 9850 9774 ...
##  $ sqftAbove   : int  1180 770 1050 1680 1050 860 1980 1890 1200 1250 ...
##  $ sqftBasement: int  0 0 910 0 730 300 970 0 0 0 ...
##  $ numBedRooms : int  3 2 4 3 3 2 4 3 2 3 ...
##  $ numBathRooms: int  1 1 3 2 1 1 3 2 1 1 ...
##  $ numFloors   : int  1 1 1 1 1 1 2 2 1 1 ...
##  $ builtYear   : int  1955 1933 1965 1987 1960 1942 1979 1994 1921 1969 ...
##  $ grade       : int  2 1 2 2 2 2 2 2 2 2 ...
##  $ waterFront  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ condition   : chr  "average" "average" "excellent" "average" ...
```

Exploratory Data Analysis

Explanatory data analytics focuses on all the parts of context, mainly the why and how. An outcome can be statistically calculated, modeled, or visualized to tell you the likelihood of certain events based on preconceived variables.

In this data set "condition" is a qualitative variable. It need to converted in to a quantitative variable.

```
l<-c()
for(n in new_housePrice$condition){if(n=="good"){n=2};if(n=="excellent"){n=3};if(n=="average"){n=1}; l<
new_housePrice$condition=l
```

```
head(new_housePrice)
```

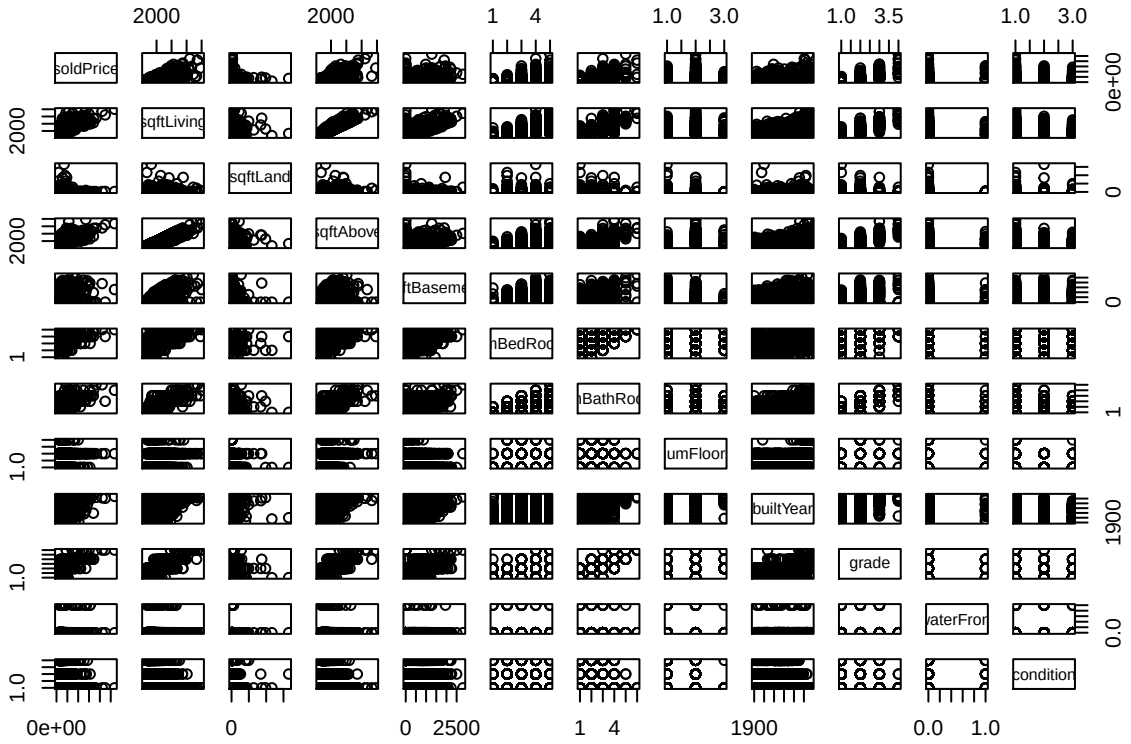
```
##   soldPrice sqftLiving sqftLand sqftAbove sqftBasement numBedRooms numBathRooms
## 1    221900       1180     5650      1180            0           3            1
## 2    180000        770    10000       770            0           2            1
## 3    604000       1960     5000      1050          910           4            3
## 4    510000       1680     8080      1680            0           3            2
## 5    229500       1780     7470      1050          730           3            1
## 6    468000       1160     6000       860          300           2            1
##   numFloors builtYear grade waterFront condition
## 1         1      1955     2          0         1
## 2         1      1933     1          0         1
## 3         1      1965     2          0         3
## 4         1      1987     2          0         1
## 5         1      1960     2          0         1
## 6         1      1942     2          0         2
```

Summary of the converted data set:

```
summary(new_housePrice)
```

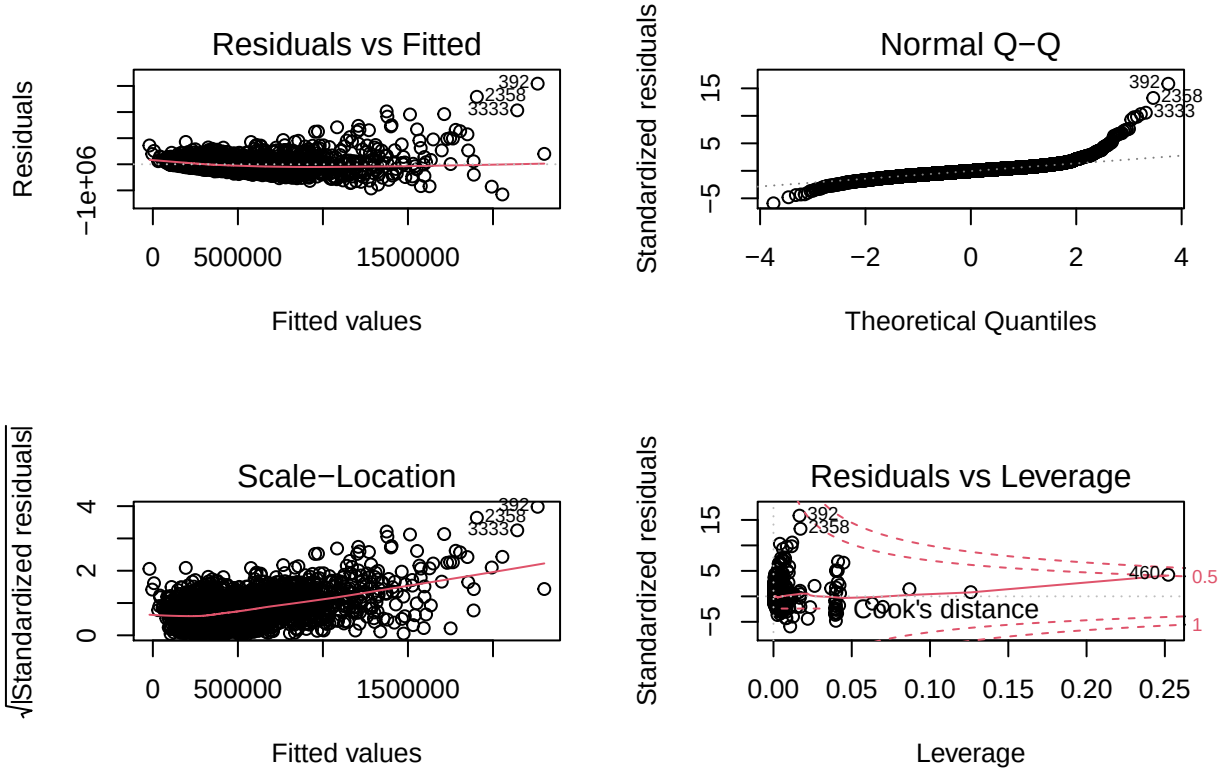
```
##    soldPrice         sqftLiving      sqftLand         sqftAbove
##  Min.   :  78000   Min.   : 390   Min.   :    600   Min.   : 390
##  1st Qu.: 253601   1st Qu.:1020   1st Qu.:   5120   1st Qu.: 950
##  Median : 356100   Median :1340   Median :   7425   Median :1160
##  Mean   : 427813   Mean   :1565   Mean   :  13575   Mean   :1352
##  3rd Qu.: 499943   3rd Qu.:1820   3rd Qu.:   9826   3rd Qu.:1510
##  Max.   :5350000   Max.   :8000   Max.   :1651359   Max.   :7850
##   sqftBasement     numBedRooms     numBathRooms     numFloors      builtYear
##  Min.   :   0.0   Min.   :1.000   Min.   :1.000   Min.   :1.00   Min.   :1900
##  1st Qu.:   0.0   1st Qu.:2.000   1st Qu.:1.000   1st Qu.:1.00   1st Qu.:1942
##  Median :   0.0   Median :3.000   Median :1.000   Median :1.00   Median :1955
##  Mean   : 212.7   Mean   :2.921   Mean   :1.581   Mean   :1.19   Mean   :1958
##  3rd Qu.: 320.0   3rd Qu.:3.000   3rd Qu.:2.000   3rd Qu.:1.00   3rd Qu.:1975
##  Max.   :2810.0   Max.   :5.000   Max.   :6.000   Max.   :3.00   Max.   :2015
##      grade         waterFront         condition
##  Min.   :1.000   Min.   :0.000000   Min.   :1.000
##  1st Qu.:1.000   1st Qu.:0.000000   1st Qu.:1.000
##  Median :2.000   Median :0.000000   Median :1.000
##  Mean   :1.758   Mean   :0.004665   Mean   :1.483
##  3rd Qu.:2.000   3rd Qu.:0.000000   3rd Qu.:2.000
##  Max.   :4.000   Max.   :1.000000   Max.   :3.000
```

```
plot(new_housePrice)
```




```
fitm <- lm(soldPrice ~ sqftLiving + sqftLand + sqftAbove + sqftBasement+numBedRooms+numBathRooms+numFlo

par(mfrow=c(2,2))
plot(fitm)
```

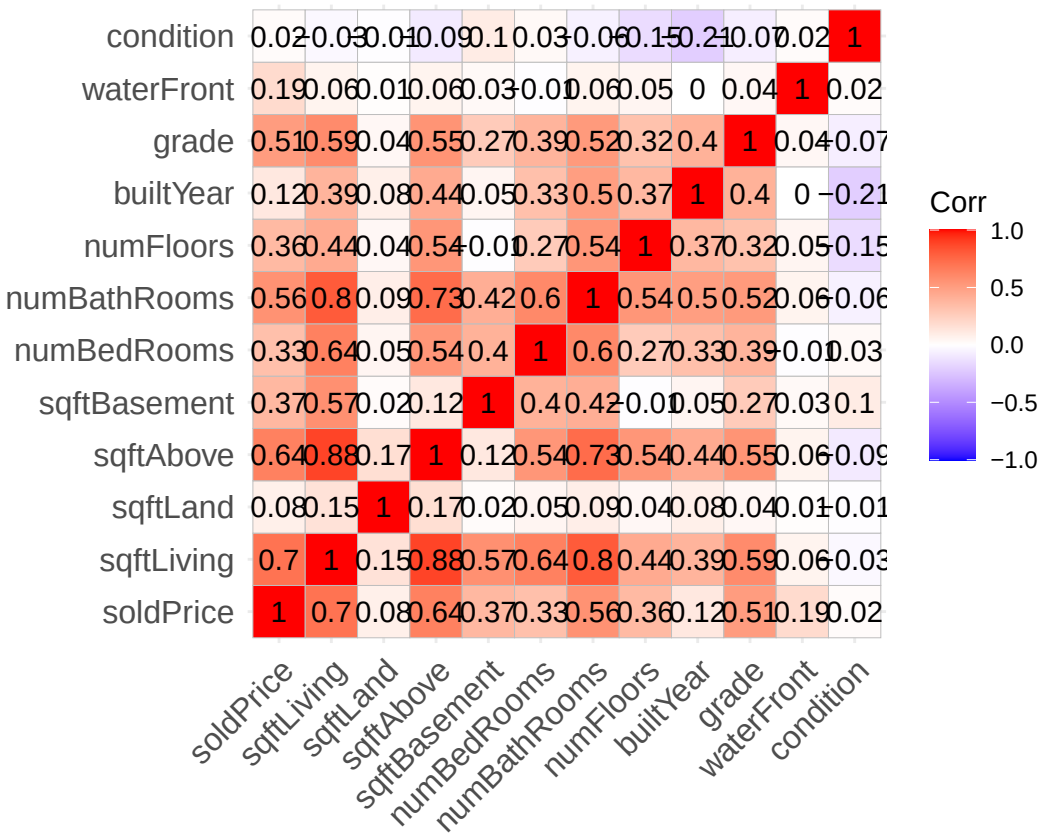


Correlation among the variables:

```
library(ggcorrplot)
```

```
## Loading required package: ggplot2
```

```
correlation=cor(new_housePrice)
ggcorrplot(correlation,lab=T)
```



According to this plot variables, cor(sqftAbove,sqftLiving)=0.88 cor(numBathRooms,sqftLiving)=0.8 has high correlation values. cor(numBathRooms,sqftAbove)=0.73 has moderate correlation value.

The relationship between high correlation variables

The relationship between SoldPrice & sqftLiving variables.

```
plot(new_housePrice$sqftAbove,new_housePrice$soldPrice,,xlab="sqftAbove",ylab = "soldPrice",main="soldP
```

The relationship between SoldPrice & sqftLiving variables.

```
plot(new_housePrice$sqftLiving,new_housePrice$soldPrice,,xlab="sqftLiving",ylab = "soldPrice",main="sol
```

Model fitting

Scaling the variables:

scaling is a method used to normalize the range of independent variables or features of data.

```
new_housePrice2=data.frame(cbind(soldPrice=new_housePrice[,1],scale(new_housePrice[,-1])))

head(new_housePrice2)
```

```
##   soldPrice sqftLiving    sqftLand  sqftAbove sqftBasement numBedRooms
## 1    221900 -0.4701279 -0.17660171 -0.2539532   -0.5540308  0.09135439
## 2    180000 -0.9712089 -0.07966798 -0.8594754   -0.5540308 -1.06070303
## 3    604000  0.4831483 -0.19108605 -0.4459481    1.8160596  1.24341181
## 4    510000  0.1409466 -0.12245252  0.4844885   -0.5540308  0.09135439
## 5    229500  0.2631615 -0.13604553 -0.4459481    1.3472505  0.09135439
## 6    468000 -0.4945708 -0.16880244 -0.7265559    0.2273176 -1.06070303
##   numBathRooms  numFloors   builtYear      grade  waterFront  condition
## 1   -0.7350245 -0.4391482 -0.09887001  0.4686515 -0.06845095 -0.7302748
## 2   -0.7350245 -0.4391482 -0.92308698 -1.4692293 -0.06845095 -0.7302748
## 3    1.7955664 -0.4391482  0.27577407  0.4686515 -0.06845095  2.2961578
## 4    0.5302709 -0.4391482  1.09999104  0.4686515 -0.06845095 -0.7302748
## 5   -0.7350245 -0.4391482  0.08845203  0.4686515 -0.06845095 -0.7302748
## 6   -0.7350245 -0.4391482 -0.58590731  0.4686515 -0.06845095  0.7829415
```

The data set is splitted in to two data sets called Train set and test set.

Training data set, which is a set of examples used to fit the parameters of the model.

The test data set is a data set used to provide an unbiased evaluation of a final model fit on the training data set.

```
library(caret)
```

```
## Loading required package: lattice
```

```
data1=createDataPartition(new_housePrice2$soldPrice,p=0.8,list = FALSE)
```

```
train.data=new_housePrice2[data1,]
dim(train.data)
```

```
## [1] 4462   12
```

```
test.data=new_housePrice2[-data1,]
dim(test.data)
```

```
## [1] 1112   12
```

Model Fitting

Model Fitting is a measure of how well a machine learning model generalizes to similar data to that on which it was trained. A model that is well-fitted produces more accurate outcomes. A model that is overfitted matches the data too closely.

In here we use Backward selection method:

```
model1=lm(train.data$soldPrice~.,data=train.data)
summary(model1)
```

```
##
## Call:
## lm(formula = train.data$soldPrice ~ ., data = train.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1163359  -101598    -7868    84264  3060731
##
## Coefficients: (1 not defined because of singularities)
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    429460       2980 144.115  < 2e-16 ***
## sqftLiving     174405       8166  21.357  < 2e-16 ***
## sqftLand       -15408       3771  -4.086 4.47e-05 ***
## sqftAbove       42477       6998   6.070 1.38e-09 ***
## sqftBasement       NA         NA      NA       NA
## numBedRooms    -51077       3998 -12.777  < 2e-16 ***
## numBathRooms    31151       5669   5.495 4.14e-08 ***
## numFloors       17518       3816   4.590 4.54e-06 ***
## builtYear      -77552       3720 -20.850  < 2e-16 ***
## grade           61623       3789  16.264  < 2e-16 ***
```

```
## waterFront      42509       2988  14.228  < 2e-16 ***
## condition        7899       3091   2.555   0.0106 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 199000 on 4451 degrees of freedom
## Multiple R-squared:  0.5928, Adjusted R-squared:  0.5919
## F-statistic:   648 on 10 and 4451 DF,  p-value: < 2.2e-16
```

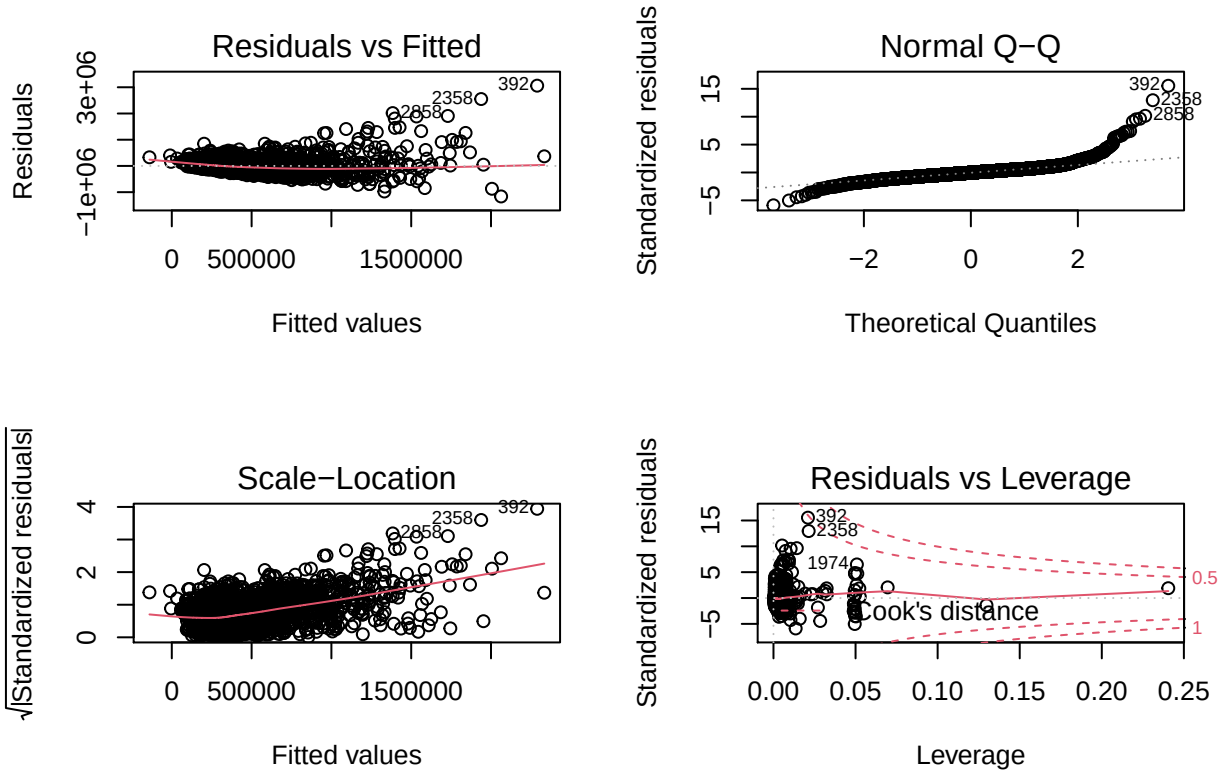
According to the summary of the model1, sqftBasement variable is not significant. Therefore we can remove the sqftBasement variable under 5% significant.

Model after sqftBasement variable removed:

```
new_model1=lm(soldPrice~.,data=train.data[,-5])
summary(new_model1)
```

```
##
## Call:
## lm(formula = soldPrice ~ ., data = train.data[, -5])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1163359  -101598    -7868    84264  3060731
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    429460       2980 144.115  < 2e-16 ***
## sqftLiving     174405       8166  21.357  < 2e-16 ***
## sqftLand       -15408       3771  -4.086 4.47e-05 ***
## sqftAbove       42477       6998   6.070 1.38e-09 ***
## numBedRooms    -51077       3998 -12.777  < 2e-16 ***
## numBathRooms    31151       5669   5.495 4.14e-08 ***
## numFloors       17518       3816   4.590 4.54e-06 ***
## builtYear      -77552       3720 -20.850  < 2e-16 ***
## grade           61623       3789  16.264  < 2e-16 ***
## waterFront      42509       2988  14.228  < 2e-16 ***
## condition        7899       3091   2.555   0.0106 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 199000 on 4451 degrees of freedom
## Multiple R-squared:  0.5928, Adjusted R-squared:  0.5919
## F-statistic:   648 on 10 and 4451 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(new_model1)
```



VIF Value:

The Variance Inflation Factor (VIF) is 1/Tolerance, it is always greater than or equal to 1. There is no formal VIF value for determining presence of multicollinearity. Values of VIF that exceed 10 are often regarded as indicating multicollinearity, but in weaker models values above 2.5 may be a cause for concern.

VIF values with sqftBasement

```
library("olsrr")
```

```
##
## Attaching package: 'olsrr'
```

```
## The following object is masked from 'package:datasets':
##
##     rivers
```

```
ols_vif_tol(model1)
```

```
##       Variables Tolerance      VIF
## 1    sqftLiving 0.0000000      Inf
## 2      sqftLand 0.9450133 1.058186
## 3     sqftAbove 0.0000000      Inf
## 4  sqftBasement 0.0000000      Inf
## 5   numBedRooms 0.5573164 1.794313
## 6  numBathRooms 0.2805673 3.564208
```

```
## 7     numFloors 0.6145929 1.627093
## 8     builtYear 0.6452754 1.549726
## 9         grade 0.6134832 1.630036
## 10   waterFront 0.9856974 1.014510
## 11    condition 0.9306835 1.074479
```

An infinite VIF value indicates that the corresponding variable may be expressed exactly by a linear combination of other variables

VIF values without sqftBasement

```
ols_vif_tol(new_model1)
```

```
##       Variables Tolerance      VIF
## 1    sqftLiving 0.1358924 7.358761
## 2      sqftLand 0.9450133 1.058186
## 3     sqftAbove 0.1800314 5.554585
## 4   numBedRooms 0.5573164 1.794313
## 5  numBathRooms 0.2805673 3.564208
## 6     numFloors 0.6145929 1.627093
## 7     builtYear 0.6452754 1.549726
## 8         grade 0.6134832 1.630036
## 9    waterFront 0.9856974 1.014510
## 10    condition 0.9306835 1.074479
```

After removing low significant variable(sqftBasement) VIF decreases.Multicollinearity decreases.

According to these VIF values sqftLiving & sqftAbove has multicollinearity.

Lets take the new_model2 as the model which removed sqftLiving.

```
new_model2=lm(soldPrice~.,data=train.data[,-c(2,5)])
summary(new_model2)
```

```
##
## Call:
## lm(formula = soldPrice ~ ., data = train.data[, -c(2, 5)])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1038188  -110881    -8384    86234  3288654
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    429189       3129 137.184  < 2e-16 ***
## sqftLand       -10111       3951  -2.559  0.01052 *
## sqftAbove      149171       5144  28.998  < 2e-16 ***
## numBedRooms    -29648       4062  -7.298 3.44e-13 ***
## numBathRooms    85830       5311  16.162  < 2e-16 ***
## numFloors        2554       3938   0.649  0.51666
## builtYear      -92886       3832 -24.242  < 2e-16 ***
## grade           77591       3900  19.897  < 2e-16 ***
## waterFront      44653       3135  14.244  < 2e-16 ***
## condition       11618       3240   3.586  0.00034 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 208900 on 4452 degrees of freedom
## Multiple R-squared:  0.5511, Adjusted R-squared:  0.5502
## F-statistic: 607.2 on 9 and 4452 DF,  p-value: < 2.2e-16
```

VIF value for the new_model2

```
ols_vif_tol(new_model2)
```

```
##      Variables Tolerance      VIF
## 1     sqftLand 0.9491196 1.053608
## 2    sqftAbove 0.3671841 2.723429
## 3  numBedRooms 0.5947920 1.681260
## 4 numBathRooms 0.3524452 2.837321
## 5    numFloors 0.6360340 1.572243
## 6    builtYear 0.6702507 1.491979
## 7        grade 0.6383405 1.566562
## 8   waterFront 0.9868123 1.013364
## 9    condition 0.9336472 1.071068
```

Lets take the new_model3 as the model which removed sqftAbove.

```
new_model3=lm(soldPrice~.,data=train.data[,-c(4,5)])
summary(new_model3)
```

```
##
## Call:
## lm(formula = soldPrice ~ ., data = train.data[, -c(4, 5)])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1154051   -99029    -8686    83205  3106751
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    429565       2992 143.576  < 2e-16 ***
## sqftLiving     209795       5741  36.542  < 2e-16 ***
## sqftLand       -13374       3771  -3.546 0.000395 ***
## numBedRooms    -52303       4008 -13.048  < 2e-16 ***
## numBathRooms    28362       5674   4.999 5.98e-07 ***
## numFloors       24892       3632   6.853 8.23e-12 ***
## builtYear      -74236       3694 -20.096  < 2e-16 ***
## grade           62934       3798  16.570  < 2e-16 ***
## waterFront      42561       3000  14.189  < 2e-16 ***
## condition        6563       3096   2.120 0.034038 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 199800 on 4452 degrees of freedom
## Multiple R-squared:  0.5894, Adjusted R-squared:  0.5886
## F-statistic: 710.2 on 9 and 4452 DF,  p-value: < 2.2e-16
```

VIF value for the new_model3

```
ols_vif_tol(new_model3)
```

```
##      Variables Tolerance      VIF
## 1   sqftLiving 0.2771602 3.608022
## 2     sqftLand 0.9525327 1.049833
## 3  numBedRooms 0.5587433 1.789731
## 4 numBathRooms 0.2824223 3.540797
## 5    numFloors 0.6838747 1.462256
## 6    builtYear 0.6595072 1.516284
## 7        grade 0.6154825 1.624742
## 8   waterFront 0.9857056 1.014502
## 9    condition 0.9354237 1.069034
```

If we were to remove both sqftLiving and sqftAbove,take that model as the new_model5. The summary is as follows:

```
new_model5=lm(soldPrice~.,data=train.data[,-c(2,4,5)])
summary(new_model5)
```

```
##
## Call:
## lm(formula = soldPrice ~ ., data = train.data[, -c(2, 4, 5)])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -903543 -119269  -11890   84336 3949545
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    429541       3411 125.934  < 2e-16 ***
## sqftLand        12157       4225   2.877  0.00403 **
## numBedRooms     -7284       4348  -1.675  0.09397 .
## numBathRooms   145325       5340  27.213  < 2e-16 ***
## numFloors       33616       4132   8.136 5.27e-16 ***
## builtYear      -91417       4177 -21.886  < 2e-16 ***
## grade          110188       4071  27.066  < 2e-16 ***
## waterFront      48145       3415  14.097  < 2e-16 ***
## condition        7457       3529   2.113  0.03464 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 227800 on 4453 degrees of freedom
## Multiple R-squared:  0.4663, Adjusted R-squared:  0.4653
## F-statistic: 486.3 on 8 and 4453 DF,  p-value: < 2.2e-16
```

When removing both sqftLiving and sqftAbove, adjusted r squared value has decreased significantly.

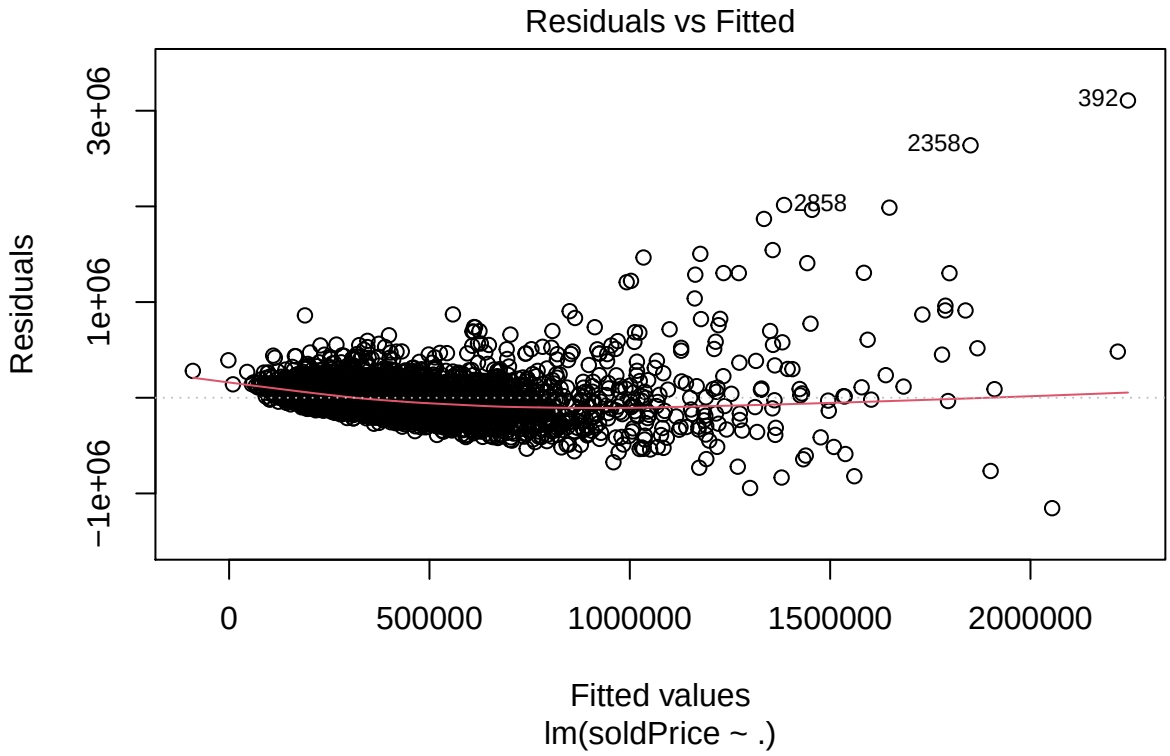
Between new_model2 & new_model3, new_model3(whithout sqftAbove and sqftBasement) has the highest adjusted r value. So we keep the new_model3.

The validation of the fitted model

by checking assumptions we can validate the model.

Linearity & the constant variance of the model can be check by Residuals vs Fitted plot.

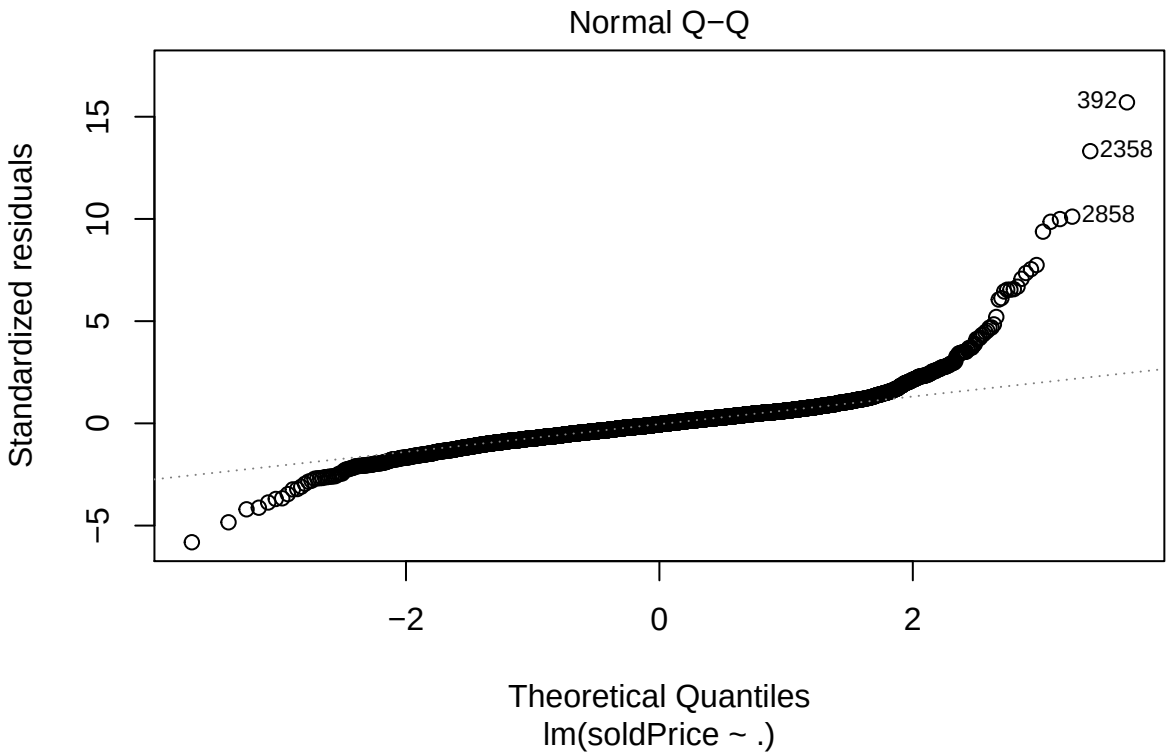
```
plot(new_model3,1)
```



In this plot there is no pattern.only a random pattern exist. This plot indicates the Linearity & the constant variance of the model.

We can check the normality of the error term by Normal Q-Q plot.

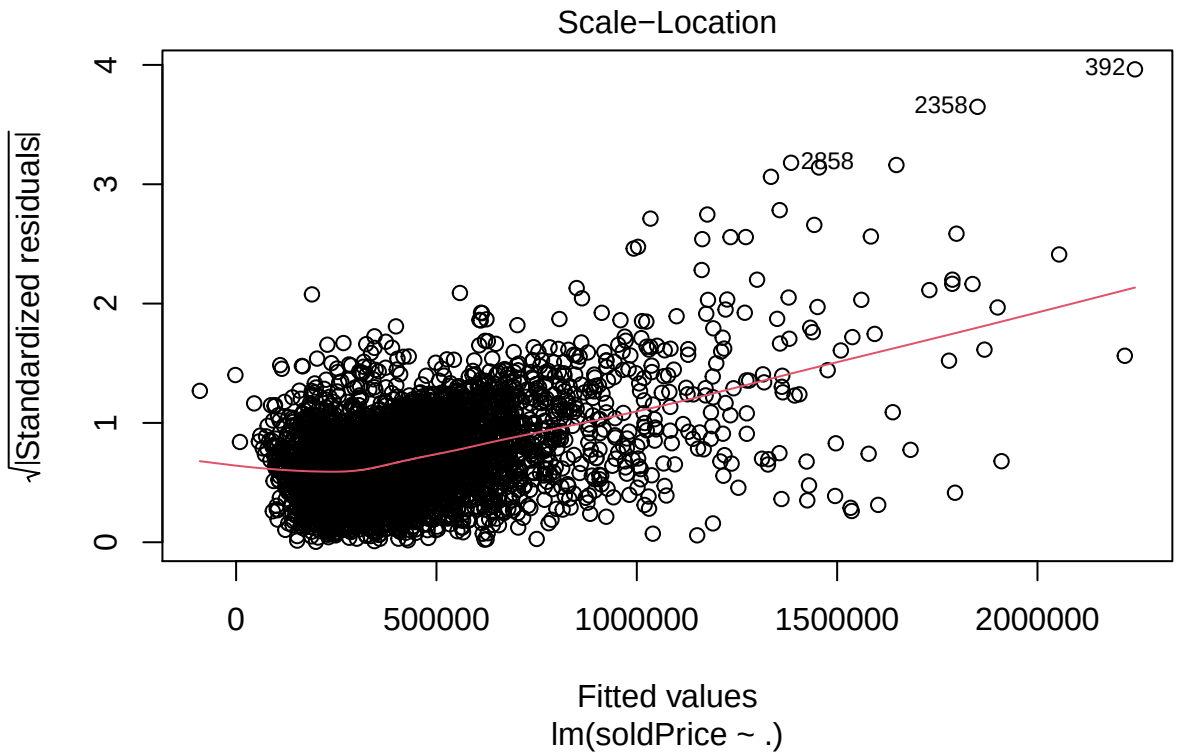
```
plot(new_model3,2)
```



In this plot majority of data points are in the straight line. Therefore it can be taken as a not violation of the normality of the error term.

we can also use the Scale−Location plot.

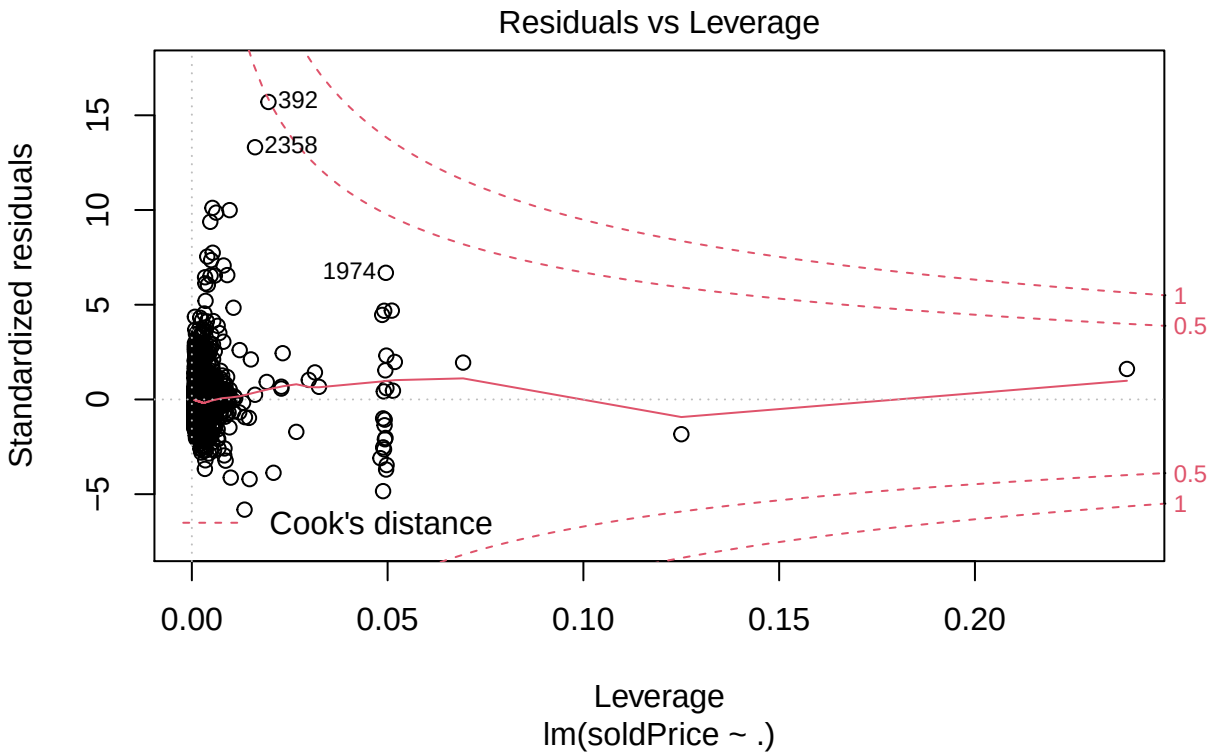
```
plot(new_model3,3)
```



This plot shows if residuals are spread equally along the ranges of predictors. This is how you can check the assumption of equal variance (homoscedasticity).

Residuals vs Leverage plot

```
plot(new_model3,5)
```



In this plot we see that there are several points that have high residual and high leverage. The points that lie close to or outside of the dashed red curves are worth investigating further.

Predicting the values using model.

Using test data we can compare the predicted values and actual values:

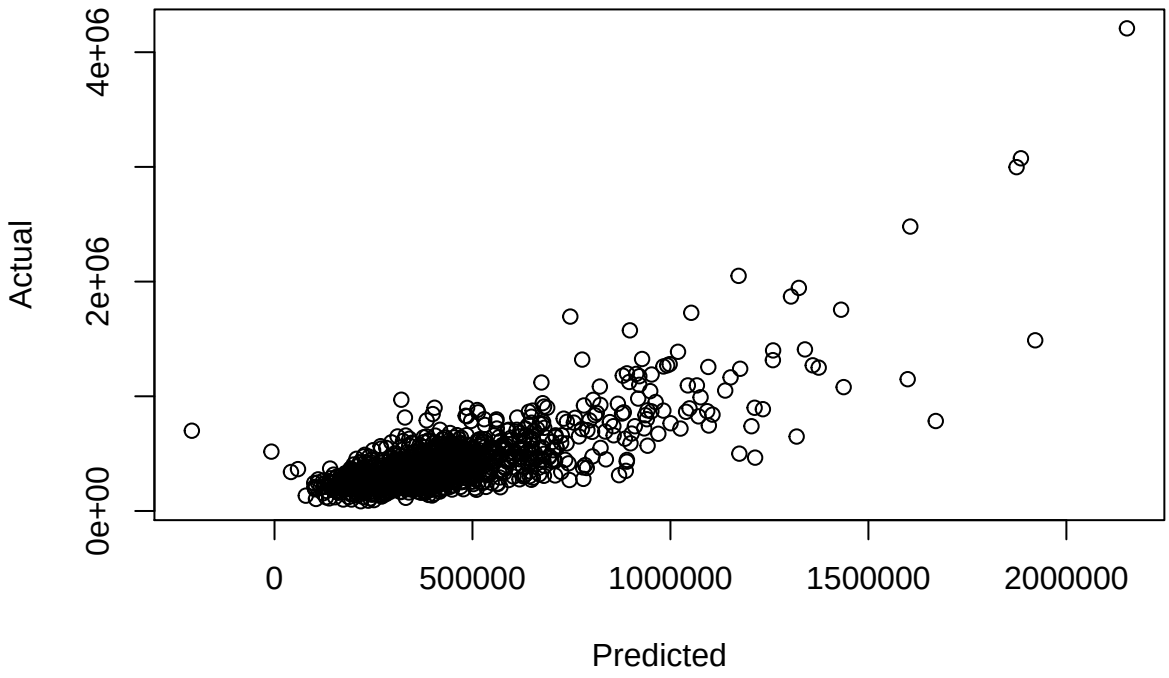
```
predictedV=predict(new_model3,test.data)
head(predictedV)
```

```
##         3        16        34        44        46        48
##  529621.1  123742.3  425710.4  404501.8  445633.3 1066781.1
```

```
compareV=cbind(test.data$soldPrice,predictedV)
head(compareV)
```

```
##            predictedV
## 3   604000   529621.1
## 16  210490   123742.3
## 34  205000   425710.4
## 44  371500   404501.8
## 46  330000   445633.3
## 48 1095000  1066781.1
```

```
plot(predictedV,test.data$soldPrice,xlab="Predicted",ylab="Actual")
```



The accuracy of the model.

The accuracy of the model can be checked by using Normalized Root Mean Squared Error(NRMSE).

```
NRMSE=RMSE(test.data$soldPrice,predictedV)/mean(test.data$soldPrice)
NRMSE
```

```
## [1] 0.4371687
```

NRMSE value of the current model is 0.4327762 The error of the model is 43.28% .

```
Accuracy = 1 - NRMSE
Accuracy
```

```
## [1] 0.5628313
```

Accuracy of the current model is 0.5672238. The accuracy of the model is 56.72% .

Discussion and Conclusion:

Here we created a new model using variables which is useful in predicting the sale price of a given house.

According to the correlation plot, variables (sqftAbove & sqftLiving=0.88), (numBathRooms,sqftLiving=0.8) has high correlation values. (numBathRooms,sqftAbove=0.73) has moderate correlation value.

To select the best fitted model we used Backward selection method.Therefore we took the summary of the model1. According to the summary of the model1, sqftBasement variable is not significant. Therefore sqftBasement variable removed under 5% significant.

VIF values were used to identify the multicollinearity.

When removing both sqftLiving and sqftAbove, adjusted r squared value has decreased significantly.Therefore only variable sqftAbove was removed due to the high multicollinearity.

The best fitted model:

SoldPrice= 429204 + 204546(sqftLiving) - 2584(sqftLand) -53100(numBedRooms) +28500(numBathRooms) + 26200(numFloors) - 71624(builtYear) + 63604(grade) + 32753(waterFront) + 6506(condition)

Adjusted R-squared value of the best fitted model is 0.5844 R-squared value of the best fitted model is 0.5853. 58.53% of the variability in response soldPrice is explained by the regression model.

The validation of the fitted model was done under plots of Residuals vs Fitted, Normal Q−Q, Scale−Location, Residuals vs Leverage. (Checked the Assumptions of the fitted model)

Predicted the values using model for test data.

Under the accuracy of the model,Normalized Root Mean Squared Error (NRMSE) was calculated. value of NRMSE is 0.4327762 The error of the model is 43.28% .

Accuracy of the current model is 0.5672238. The accuracy of the model is 56.72% .