PREDICTIVE MODEL BUILDING USING LINEAR REGRESSION ANALYSIS

AGNIVA KONAR, UPASYA BOSE, TIYASHA SAMANTA

29/08/2021

```
library(readxl)
library(qpcR)
library(car)
library(carData)
library(nlme)
library(lmtest)
library(BSDA)
library(MASS)
library(ROCR)
library(writexl)
library(WriteXLS)
library(caTools)
```

The necessary packages are installed.

```
data=read_excel(file.choose())
View(data)
df=data[,-c(1,3,5,8,10,12,14,16,18,20,22,28,30,32,34,36,38,40,42,44,46,48,50,53,57,68,71,74,76,79,83,90,94,96)]
View(df)
dim(df)
```

```
## [1] 1379   64
```

The unnecessary columns,i.e., the columns which had maximum number of single type observations and the columns which had non numeric data, are dropped from the previous dataset. The new dataset is named as “df”. This dataset contains only numeric data which will be useful for further calculations. This new dataset contains 1379 rows of data and 64 columns.

Now, we name the variables accordingly for our ease and create a new dataframe.

```
#naming the response variable and the covariates
y=df$SalePrice
x1=df$`MSSubClass Indicator`
x2=df$`MSZoning Indicator`
x3=df$`LotFrontage Converted`
x4=df$LotArea
x5=df$`Alley Indicator`
x6=df$`LotShape Indicator`
x7=df$`LandContour Indicator`
x8=df$`LotConfig Indicator`
x9=df$`Neighbourhood Indicator`
x10=df$`Condition1 Indicator`
x11=df$`BldgType Indicator`
x12=df$`HouseStyle Indicator`
x13=df$OverallQual
x14=df$OverallCond
x15=df$YearBuilt
x16=df$YearRemodAdd
x17=df$`RoofStyle Indicator`
x18=df$`Exterior1st Indicator`
x19=df$`Exterior2nd Indicator`
x20=df$`MasVnrType Indicator`
x21=df$`MasVnrArea Converted`
x22=df$`ExterQual Indicator`
x23=df$`ExterCond Indicator`
x24=df$`Foundation Indicator`
x25=df$`BsmtQual Indicator`
x26=df$`BsmtCond Indicator`
x27=df$`BsmtExposure Indicator`
x28=df$`BsmtFinType1 Indicator`
x29=df$BsmtFinSF1
x30=df$`BsmtFinType2 Indicator`
x31=df$BsmtFinSF2
x32=df$BsmtUnfSF
x33=df$`Electrical Indicator`
x34=df$`1stFlrSF`
x35=df$`2ndFlrSF`
x36=df$LowQualFinSF
x37=df$BsmtFullBath
x38=df$BsmtHalfBath
x39=df$FullBath
x40=df$HalfBath
x41=df$BedroomAbvGr
x42=df$KitchenAbvGr
x43=df$`KitchenQual Indicator`
x44=df$TotRmsAbvGrd
x45=df$`Functional Indicator`
x46=df$Fireplaces
x47=df$`FireplaceQU Indicator`
x48=df$`GarageType Indicator`
x49=df$GarageYrBlt
x50=df$`GarageFinish Indicator`
x51=df$GarageCars
x52=df$GarageArea
x53=df$`PavedDrive Indicator`
x54=df$WoodDeckSF
x55=df$OpenPorchSF
x56=df$EnclosedPorch
x57=df$`3SsnPorch`
x58=df$ScreenPorch
x59=df$`Fence Indicator`
x60=df$MoSold
x61=df$YrSold
x62=df$`SaleType Indicator`
x63=df$`SaleCondition Indicator`

df1=data.frame(y,x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15,x16,x17,x18,x19,x20,x21,x22,x23,x24,x25,x26,x27,x28,x29,x30,x31,x32,x33,x34,x35,x36,x37,x38,x39,x40,x41,x42,x43,x44,x45,x46,x47,x48,x49,x50,x51,x52,x53,x54,x55,x56,x57,x58,x59,x60,x61,x62,x63)
```

After the original dataset is imported, we need to split it into two datasets, naming them as, train and test dataset. And this splitting must be done randomly, in the ratio of 70:30. We need to build our predictive model based on the train dataset and evaluate the model based on the testing dataset.

```
set.seed(100)

traindf=sort(sample(nrow(df1),nrow(df1)*0.7)) #randomly picking 70% of the observations from our data set to form the training data

train=df1[traindf,] #new training data set
test=df1[-traindf,] #new testing data set

rownames(train)=1:nrow(train) #changing the indexes of the train dataset
rownames(test)=1:nrow(test) #changing the indexes of the test dataset

#omitting if any NA values are present
train_new=na.omit(train)
test_new=na.omit(test)

dim(train_new)
```

```
## [1] 953  64
```

```
dim(test_new)
```

```
## [1] 411  64
```

```
View(train_new)
View(test_new)
```

We create a primary linear model based on the train dataset.

```
model=lm(y~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+x11+x12+x13+x14+x15+x16+x17+x18+x19+x20+x21+x22+x23+x24+x25+x26+x27+x28+x29+x30+x31+x32+x33+x34+x35+x36+x37+x38+x39+x40+x41+x42+x43+x44+x45+x46+x47+x48+x49+x50+x51+x52+x53+x54+x55+x56+x57+x58+x59+x60+x61+x62+x63, data = train_new)

summary(model)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 +
##     x10 + x11 + x12 + x13 + x14 + x15 + x16 + x17 + x18 + x19 +
##     x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28 + x29 +
##     x30 + x31 + x32 + x33 + x34 + x35 + x36 + x37 + x38 + x39 +
##     x40 + x41 + x42 + x43 + x44 + x45 + x46 + x47 + x48 + x49 +
##     x50 + x51 + x52 + x53 + x54 + x55 + x56 + x57 + x58 + x59 +
##     x60 + x61 + x62 + x63, data = train_new)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -140621  -14704     615   14078  188552
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.044e+06  1.439e+06   0.726 0.468268
## x1          -1.732e+02  6.199e+02  -0.279 0.780027
## x2          -7.189e+02  1.087e+03  -0.661 0.508667
## x3           1.598e+02  5.505e+01   2.903 0.003793 **
## x4           3.261e-01  9.533e-02   3.420 0.000654 ***
## x5           6.287e+03  2.697e+03   2.331 0.019985 *
## x6           7.188e+02  1.762e+03   0.408 0.683491
## x7          -9.667e+01  1.789e+03  -0.054 0.956917
## x8           7.323e+02  1.212e+03   0.604 0.545703
## x9          -1.660e+02  1.650e+02  -1.006 0.314694
## x10         -1.285e+03  1.006e+03  -1.276 0.202139
## x11         -3.402e+03  2.216e+03  -1.535 0.125169
## x12         -1.054e+03  8.532e+02  -1.235 0.217049
## x13          1.169e+04  1.356e+03   8.616  < 2e-16 ***
## x14          5.441e+03  1.217e+03   4.470 8.85e-06 ***
## x15          3.174e+02  8.891e+01   3.570 0.000375 ***
## x16          5.989e+01  7.742e+01   0.773 0.439438
## x17          2.967e+03  1.186e+03   2.502 0.012535 *
## x18         -1.443e+03  5.045e+02  -2.861 0.004319 **
## x19          6.595e+02  5.203e+02   1.268 0.205302
## x20          6.410e+03  1.041e+03   6.154 1.14e-09 ***
## x21          4.582e+01  6.719e+00   6.819 1.69e-11 ***
## x22         -1.048e+04  2.755e+03  -3.804 0.000152 ***
## x23          2.858e+03  3.053e+03   0.936 0.349446
## x24          3.353e+03  1.795e+03   1.868 0.062036 .
## x25         -3.254e+03  2.236e+03  -1.455 0.145918
## x26          1.076e+04  2.759e+03   3.899 0.000104 ***
## x27         -5.568e+03  1.140e+03  -4.884 1.23e-06 ***
## x28          7.058e+02  6.904e+02   1.022 0.306935
## x29          5.174e+01  6.238e+00   8.295 4.00e-16 ***
## x30         -1.078e+03  1.671e+03  -0.645 0.519026
## x31          3.119e+01  9.677e+00   3.223 0.001314 **
## x32          2.731e+01  5.513e+00   4.953 8.73e-07 ***
## x33          4.489e+03  2.645e+03   1.698 0.089932 .
## x34          6.164e+01  6.453e+00   9.552  < 2e-16 ***
## x35          7.259e+01  5.083e+00  14.281  < 2e-16 ***
## x36          4.156e+01  2.416e+01   1.720 0.085746 .
## x37         -1.379e+03  2.711e+03  -0.508 0.611242
## x38          9.307e+02  4.105e+03   0.227 0.820699
## x39         -1.315e+03  2.879e+03  -0.457 0.647832
## x40         -8.571e+02  2.718e+03  -0.315 0.752538
## x41         -9.456e+03  1.829e+03  -5.169 2.91e-07 ***
## x42         -2.240e+04  6.316e+03  -3.547 0.000411 ***
## x43         -7.329e+03  2.226e+03  -3.292 0.001034 **
## x44          2.053e+03  1.254e+03   1.638 0.101870
## x45         -6.723e+03  1.579e+03  -4.257 2.29e-05 ***
## x46          4.699e+03  2.891e+03   1.625 0.104415
## x47          2.348e+03  1.074e+03   2.186 0.029059 *
## x48          2.634e+03  6.714e+02   3.922 9.44e-05 ***
## x49         -1.408e+02  8.069e+01  -1.746 0.081219 .
## x50         -1.129e+03  1.556e+03  -0.726 0.468297
## x51          4.216e+03  2.939e+03   1.434 0.151789
## x52          1.065e+01  1.040e+01   1.024 0.306052
## x53          4.159e+02  2.476e+03   0.168 0.866617
## x54          5.451e+00  8.041e+00   0.678 0.498026
## x55          2.962e+00  1.518e+01   0.195 0.845359
## x56         -1.711e+01  1.788e+01  -0.957 0.338628
## x57          2.011e+00  2.984e+01   0.067 0.946277
## x58          2.633e+01  1.607e+01   1.639 0.101632
## x59         -1.852e+02  7.919e+02  -0.234 0.815091
## x60         -4.538e+02  3.400e+02  -1.335 0.182296
## x61         -7.826e+02  7.093e+02  -1.103 0.270166
## x62          5.762e+00  8.254e+02   0.007 0.994432
## x63          2.276e+03  7.865e+02   2.894 0.003891 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 27140 on 889 degrees of freedom
## Multiple R-squared:  0.8942, Adjusted R-squared:  0.8867
## F-statistic: 119.3 on 63 and 889 DF,  p-value: < 2.2e-16
```

```
RSS(model)
```

```
## [1] 654759362143
```

```
AIC(model)
```

```
## [1] 22226.06
```

Now that we have splitted our dataset into train and test, we must proceed with our model building using train dataset. But, before moving with our model building, we must remove the impurities, such as outliers, influential observations and high leverage values from the train dataset. This process is called Data Cleansing. While performing data cleansing, we must make sure that we don’t lose too much information in this process.

We use Cook’s Distance to find out the influential observations from the data.

```
cook = cooks.distance(model)
c = cook[cook>(4/953)] #sorting out potential influential observations.
c
```

```
##           5          42          56         108         117         119
## 0.006496926 0.008562686 0.005724470 0.008439562 0.004715284 0.004302250
##         132         156         157         165         174         181
## 0.011893322 0.007508394 0.011332998 0.007212348 0.005253426 0.005814620
##         193         208         216         223         248         253
## 0.020855284 0.005823357 0.006032043 0.017308381 0.006152650 0.005552164
##         266         269         286         308         324         330
## 0.006969248 0.004395845 0.020617098 0.010657000 0.004342246 0.004788936
##         385         388         404         415         417         438
## 0.009822633 0.032371461 0.005100716 0.010083875 0.016856153 0.010570426
##         439         459         475         477         512         514
## 0.010271649 0.093237372 0.007344220 0.004356159 0.041123718 0.007386519
##         532         588         600         631         632         659
## 0.047701796 0.006792937 0.044981724 0.004267031 0.009688364 0.004812571
##         689         701         702         773         775         783
## 0.015632610 0.004438708 0.034493543 0.007541256 0.033105131 0.018810256
##         784         789         802         811         817         827
## 0.181727914 0.005718070 0.020856412 0.009362059 0.007602273 0.004561052
##         840         876         919         920         938
## 0.006885672 0.018846858 0.006272004 0.012508284 0.005798899
```

```
length(c) #total no. of potential influential observations.
```

```
## [1] 59
```

We know, that the potential influential observations are the values that have Cook’s distance more than (4/n), where n is the no. of observations, that is 953 in this case.

To find out the outliers from our data, the concept of Studentized residual is used.

```
student = studres(model)
s = student[abs(student)>3] #sorting out the potential outliers
s
```

```
##       132       286       308       388       417       459       512       514
##  3.037065  3.152830  3.885908 -3.934498 -4.203158  6.646794  3.712988  3.342702
##       532       600       702       775       783       784       802       876
##  6.166638  5.242724  5.924668  4.005920  3.071273  7.869528 -3.271101 -5.374302
```

```
length(s) #total no. of potential outliers.
```

```
## [1] 16
```

The observations, for which the absolute value of the studentized residual is greater than 3, gets the tag of being a potential outlier.

We use the concept of Hat Matrix and Hat Values to find out the high leverage values from the data.

```
hat = hatvalues(model)
h = hat[hat>(189/953)] #sorting out the potential high leverage values
h
```

```
##        34       103       156       157       205       223       264       447
## 0.2102171 0.2285490 0.2399208 0.2757952 0.4252388 0.3045922 0.2465346 0.2169015
##       588       625       877
## 0.2535756 0.2986185 0.2505759
```

```
length(h) #total no. of potential high leverage values.
```

```
## [1] 11
```

We know, that the potential high leverage values are the values that have hat values more than (3p/n), where n is the no. of observations, that is 953 in this case and p is the no. of covariates, that is 63 in this case.

Now that we have found out the anomalies in the data, we need to remove them tactically.

```
influential=as.numeric(names(c)) #storing the indexes of the values that are influential observations
outliers=as.numeric(names(s)) #storing the indexes of the values that are outliers
highleverage=as.numeric(names(h)) #storing the indexes of the values that are high leverage values

a=intersect(influential,outliers)
b=intersect(outliers,highleverage)
c=intersect(highleverage,influential)
d=intersect(influential,b) #common values in all three of them

a
```

```
##  [1] 132 286 308 388 417 459 512 514 532 600 702 775 783 784 802 876
```

```
b
```

```
## numeric(0)
```

```
c
```

```
## [1] 156 157 223 588
```

```
d
```

```
## numeric(0)
```

Now that we have found out the data impurities, which is 20 in our case. We remove them from the train dataset and create a clean dataset. We also remove the NA values from the data, if any. Based on this dataset, we carry out our further modeling.

```
train_new1=train_new[-c(a,c),] #creating a new dataframe after data cleansing
train_new2=na.omit(train_new1)

dim(train_new2)
```

```
## [1] 933  64
```

```
View(train_new2)
```

Now that we have a cleaned data set, we create a new linear model and check for its efficiency.

```
model1=lm(y~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+x11+x12+x13+x14+x15+x16+x17+x18+x19+x20+x21+x22+x23+x24+x25+x26+x27+x28+x29+x30+x31+x32+x33+x34+x35+x36+x37+x38+x39+x40+x41+x42+x43+x44+x45+x46+x47+x48+x49+x50+x51+x52+x53+x54+x55+x56+x57+x58+x59+x60+x61+x62+x63, data = train_new2)

summary(model1)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 +
##     x10 + x11 + x12 + x13 + x14 + x15 + x16 + x17 + x18 + x19 +
##     x20 + x21 + x22 + x23 + x24 + x25 + x26 + x27 + x28 + x29 +
##     x30 + x31 + x32 + x33 + x34 + x35 + x36 + x37 + x38 + x39 +
##     x40 + x41 + x42 + x43 + x44 + x45 + x46 + x47 + x48 + x49 +
##     x50 + x51 + x52 + x53 + x54 + x55 + x56 + x57 + x58 + x59 +
##     x60 + x61 + x62 + x63, data = train_new2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -141020  -14872     811   14020  185943
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.068e+06  1.465e+06   0.729 0.466019
## x1          -1.327e+02  6.306e+02  -0.210 0.833383
## x2          -6.857e+02  1.101e+03  -0.623 0.533549
## x3           1.638e+02  5.552e+01   2.950 0.003259 **
## x4           3.249e-01  9.605e-02   3.382 0.000751 ***
## x5           6.049e+03  2.748e+03   2.201 0.027987 *
## x6           1.095e+03  1.798e+03   0.609 0.542608
## x7          -2.893e+02  1.816e+03  -0.159 0.873494
## x8           7.045e+02  1.225e+03   0.575 0.565359
## x9          -1.858e+02  1.680e+02  -1.106 0.269111
## x10         -1.302e+03  1.013e+03  -1.285 0.198973
## x11         -3.411e+03  2.252e+03  -1.515 0.130256
## x12         -1.140e+03  8.671e+02  -1.315 0.188845
## x13          1.155e+04  1.373e+03   8.410  < 2e-16 ***
## x14          5.431e+03  1.235e+03   4.396 1.24e-05 ***
## x15          3.187e+02  8.979e+01   3.549 0.000407 ***
## x16          5.829e+01  7.859e+01   0.742 0.458474
## x17          3.041e+03  1.207e+03   2.520 0.011920 *
## x18         -1.426e+03  5.087e+02  -2.802 0.005191 **
## x19          6.707e+02  5.251e+02   1.277 0.201806
## x20          6.404e+03  1.055e+03   6.072 1.88e-09 ***
## x21          4.567e+01  6.794e+00   6.722 3.25e-11 ***
## x22         -1.088e+04  2.800e+03  -3.885 0.000110 ***
## x23          3.324e+03  3.126e+03   1.063 0.287948
## x24          3.127e+03  1.814e+03   1.724 0.085149 .
## x25         -3.612e+03  2.269e+03  -1.592 0.111849
## x26          1.102e+04  2.797e+03   3.940 8.80e-05 ***
## x27         -5.709e+03  1.162e+03  -4.915 1.06e-06 ***
## x28          7.058e+02  7.022e+02   1.005 0.315061
## x29          5.183e+01  6.363e+00   8.146 1.30e-15 ***
## x30         -1.067e+03  1.683e+03  -0.634 0.526311
## x31          3.132e+01  9.765e+00   3.207 0.001388 **
## x32          2.743e+01  5.611e+00   4.889 1.21e-06 ***
## x33          4.546e+03  2.673e+03   1.701 0.089366 .
## x34          6.198e+01  6.553e+00   9.457  < 2e-16 ***
## x35          7.326e+01  5.176e+00  14.153  < 2e-16 ***
## x36          4.147e+01  2.432e+01   1.705 0.088590 .
## x37         -1.565e+03  2.767e+03  -0.566 0.571792
## x38          4.951e+02  4.161e+03   0.119 0.905322
## x39         -1.413e+03  2.911e+03  -0.485 0.627475
## x40         -7.277e+02  2.756e+03  -0.264 0.791800
## x41         -9.684e+03  1.859e+03  -5.209 2.37e-07 ***
## x42         -2.210e+04  6.482e+03  -3.409 0.000682 ***
## x43         -7.611e+03  2.248e+03  -3.386 0.000742 ***
## x44          1.987e+03  1.274e+03   1.561 0.118977
## x45         -6.679e+03  1.594e+03  -4.191 3.06e-05 ***
## x46          4.542e+03  2.917e+03   1.557 0.119824
## x47          2.341e+03  1.086e+03   2.156 0.031330 *
## x48          2.668e+03  6.861e+02   3.890 0.000108 ***
## x49         -1.471e+02  8.169e+01  -1.800 0.072190 .
## x50         -9.718e+02  1.583e+03  -0.614 0.539477
## x51          3.891e+03  2.972e+03   1.309 0.190841
## x52          1.241e+01  1.055e+01   1.176 0.239886
## x53          2.801e+02  2.500e+03   0.112 0.910835
## x54          5.495e+00  8.148e+00   0.674 0.500215
## x55          3.479e+00  1.532e+01   0.227 0.820423
## x56         -1.843e+01  1.840e+01  -1.002 0.316752
## x57          2.103e+00  3.003e+01   0.070 0.944198
## x58          2.572e+01  1.618e+01   1.590 0.112282
## x59         -2.860e+02  8.014e+02  -0.357 0.721272
## x60         -4.743e+02  3.445e+02  -1.377 0.168969
## x61         -7.868e+02  7.220e+02  -1.090 0.276164
## x62         -3.683e+02  8.668e+02  -0.425 0.671013
## x63          2.391e+03  8.059e+02   2.967 0.003087 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 27290 on 869 degrees of freedom
## Multiple R-squared:  0.8943, Adjusted R-squared:  0.8866
## F-statistic: 116.7 on 63 and 869 DF,  p-value: < 2.2e-16
```

```
RSS(model1)
```

```
## [1] 647411693584
```

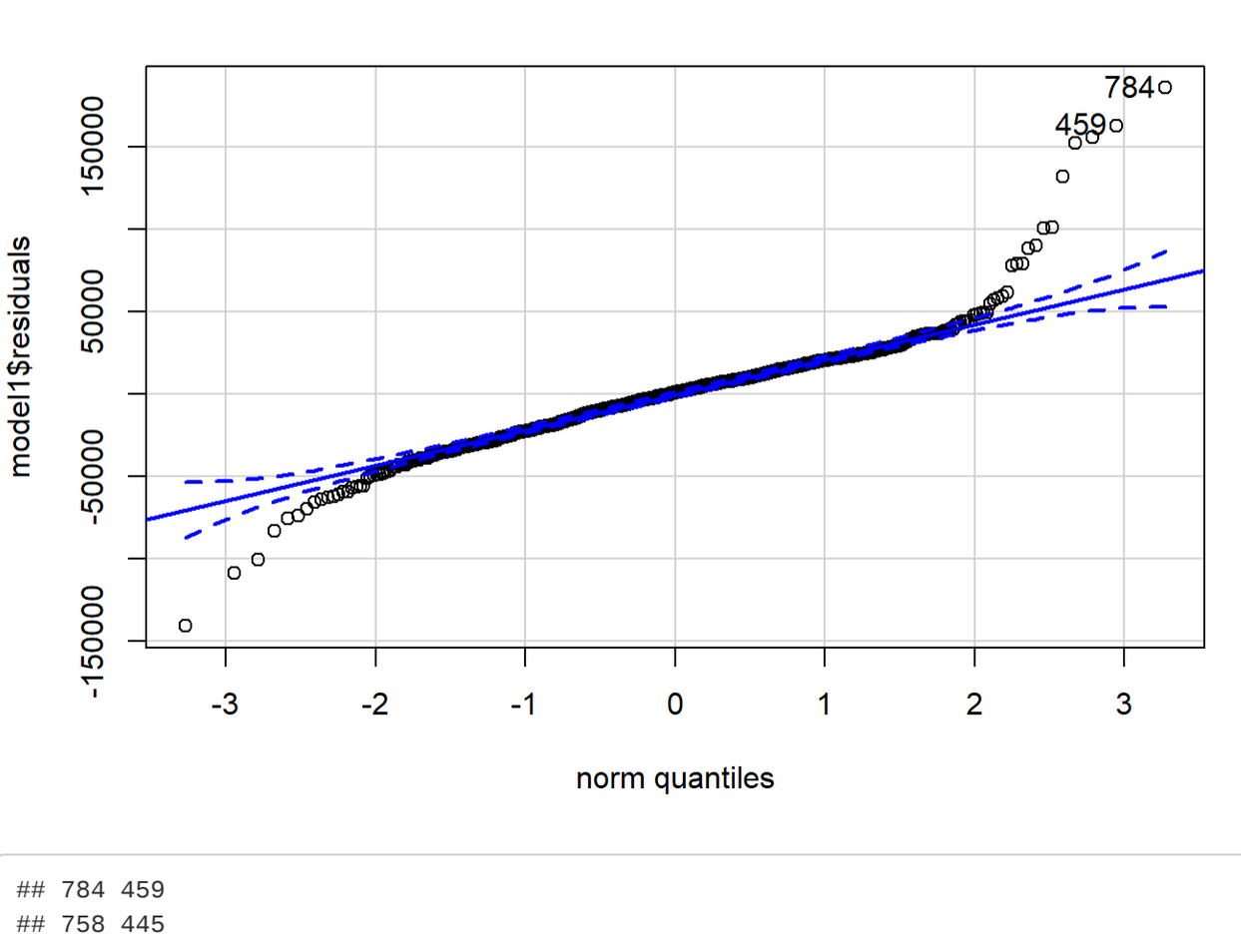
```
AIC(model1)
```

```
## [1] 21771.61
```

We see that the adjusted R Square value increases a bit and at the same time the AIC value drops down as well. This indicates that our model has improved a bit.

Now we check for the normality assumption of this model. We will use histogram and QQPlot to check for normality.

```
qqPlot(model1$residuals)
```




```
## 784 459
## 758 445
```

From the QQPlot, we can observe that the quantiles of the data was moderately fitting with the quantiles of a normal data.

Hence the normality assumption is restored.

Now, we check for autocorrelation. We use Durbin Watson Test to check this.

```
dwtest(model1)
```

```
##
##  Durbin-Watson test
##
## data:  model1
## DW = 2.0391, p-value = 0.7256
## alternative hypothesis: true autocorrelation is greater than 0
```

The Value of the test statistic is nearly equals to 2, which suggests that there is no autocorrelation is the dataset. This conclusion is evident even from the p value of the test. The null hypothesis of the test is that there is no autocorrelation and since the p value of the test turns out to be greater than 0.05 (5% level of significance) and hence, null hypothesis is accepted.

Now, we need to check if our data is homoscedastic or not. We use Breusch Pagan test to check for homoscedasticity in our dataset.

```
bptest(model1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  model1
## BP = 274.44, df = 63, p-value < 2.2e-16
```

The p value of the test turns out to be much lesser than 0.05 and hence that suggests our data is heteroscedastic. Hence, we need to find some other method to solve this issue.

Now, we need to check for multicollinearity in the dataset. We use Variance Inflation Factor to check which covariates have high multicollinearity.

```
vif(model1)
```

```
##       x1       x2       x3       x4       x5       x6       x7       x8
## 8.607972 1.526831 1.676028 1.529429 1.309004 1.392394 1.428043 1.156591
##       x9      x10      x11      x12      x13      x14      x15      x16
## 1.298096 1.122592 6.708097 3.672226 4.080977 2.075974 8.730581 3.161467
##      x17      x18      x19      x20      x21      x22      x23      x24
## 1.306359 5.187784 5.123236 1.591527 1.901525 3.191180 1.258551 2.106329
##      x25      x26      x27      x28      x29      x30      x31      x32
## 4.649462 2.545442 1.894112 2.706943 9.556699 3.011436 3.955825 7.844222
##      x33      x34      x35      x36      x37      x38      x39      x40
## 1.232631 7.948481 6.511998 1.139208 2.552397 1.269888 3.218698 2.398204
##      x41      x42      x43      x44      x45      x46      x47      x48
## 2.590653 1.952115 2.785475 5.028808 1.231910 4.578061 4.807685 1.891560
##      x49      x50      x51      x52      x53      x54      x55      x56
## 5.127843 2.094336 4.400363 4.645188 1.366848 1.305917 1.298265 1.336342
##      x57      x58      x59      x60      x61      x62      x63
## 1.075643 1.150837 1.239959 1.077858 1.125961 1.602706 1.831000
```

We find out that the VIF value each and every covariates are lesser than 10, which indicates that there is moderate multicollinearity present in the model which isn’t an issue. Thus we can proceed further with our predictive model building.

But still, the issue of heteroscedasticity still persists. And hence, we need to use the GLS function to solve this issue.

```
model2=gls(y~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+x11+x12+x13+x14+x15+x16+x17+x18+x19+x20+x21+x22+x23+x24+x25+x26+x27+x28+x29+x30+x31+x32+x33+x34+x35+x36+x37+x38+x39+x40+x41+x42+x43+x44+x45+x46+x47+x48+x49+x50+x51+x52+x53+x54+x55+x56+x57+x58+x59+x60+x61+x62+x63,data = train_new2,correlation = corAR1())
```

After passing the model as a parameter to the summary(), we found out that the covariates x3,x4,x5,x12,x13,x14,x15,x17,x18,x20,x21,x22,x26,x27,x29,x31,x32,x34,x35,x41,x42,x43,x45,x47,x48,x63 are the significant covariates.

Thus, LotFrontage, LotArea, Alley, Housestyle, OverallQual, OverallCond, YearBuilt, RoofStyle, Exterior1st, MasVnrType, MasVnrArea, ExterQual, BsmtCond, BsmtExposure, BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, 1stFlrSF, 2ndFlrSF, BedroomAbvGr, KitchenAbvGr, KitchenQual, Functional, FireplaceQU, GarageType, SaleCondition are our significant factors in determining the price of the house.

Hence, we need to build our predictive model based on these covariates. Hence, our predictive model is given by:-

```
model3=gls(y~x3+x4+x5+x12+x13+x14+x15+x17+x18+x20+x21+x22+x26+x27+x29+x31+x32+x34+x35+x41+x42+x43+x45+x47+x48+x63, data = train_new2, correlation = corAR1())

model3
```

```
## Generalized least squares fit by REML
##   Model: y ~ x3 + x4 + x5 + x12 + x13 + x14 + x15 + x17 + x18 + x20 +      x21 + x22 + x26 + x27 + x29 + x31 + x32 + x34 + x35 + x41 +      x42 + x43 + x45 + x47 + x48 + x63
##   Data: train_new2
##   Log-restricted-likelihood: -10688.48
##
## Coefficients:
##   (Intercept)            x3            x4            x5           x12
## -6.834697e+05  2.599412e+02  3.772257e-01  5.695243e+03 -1.064007e+03
##           x13           x14           x15           x17           x18
##  1.138925e+04  5.692319e+03  3.175505e+02  3.311524e+03 -5.496840e+02
##           x20           x21           x22           x26           x27
##  7.142737e+03  4.839669e+01 -1.165855e+04  1.132460e+04 -6.270452e+03
##           x29           x31           x32           x34           x35
##  5.495424e+01  3.823496e+01  3.358001e+01  6.526766e+01  7.923491e+01
##           x41           x42           x43           x45           x47
## -7.321179e+03 -2.650223e+04 -7.906382e+03 -6.343883e+03  5.797173e+02
##           x48           x63
##  2.484910e+03  2.594036e+03
##
## Correlation Structure: AR(1)
##  Formula: ~1
##  Parameter estimate(s):
##         Phi
## -0.03488849
## Degrees of freedom: 933 total; 906 residual
## Residual standard error: 27667.37
```

```
pred=predict(model3,test_new)
pred
```

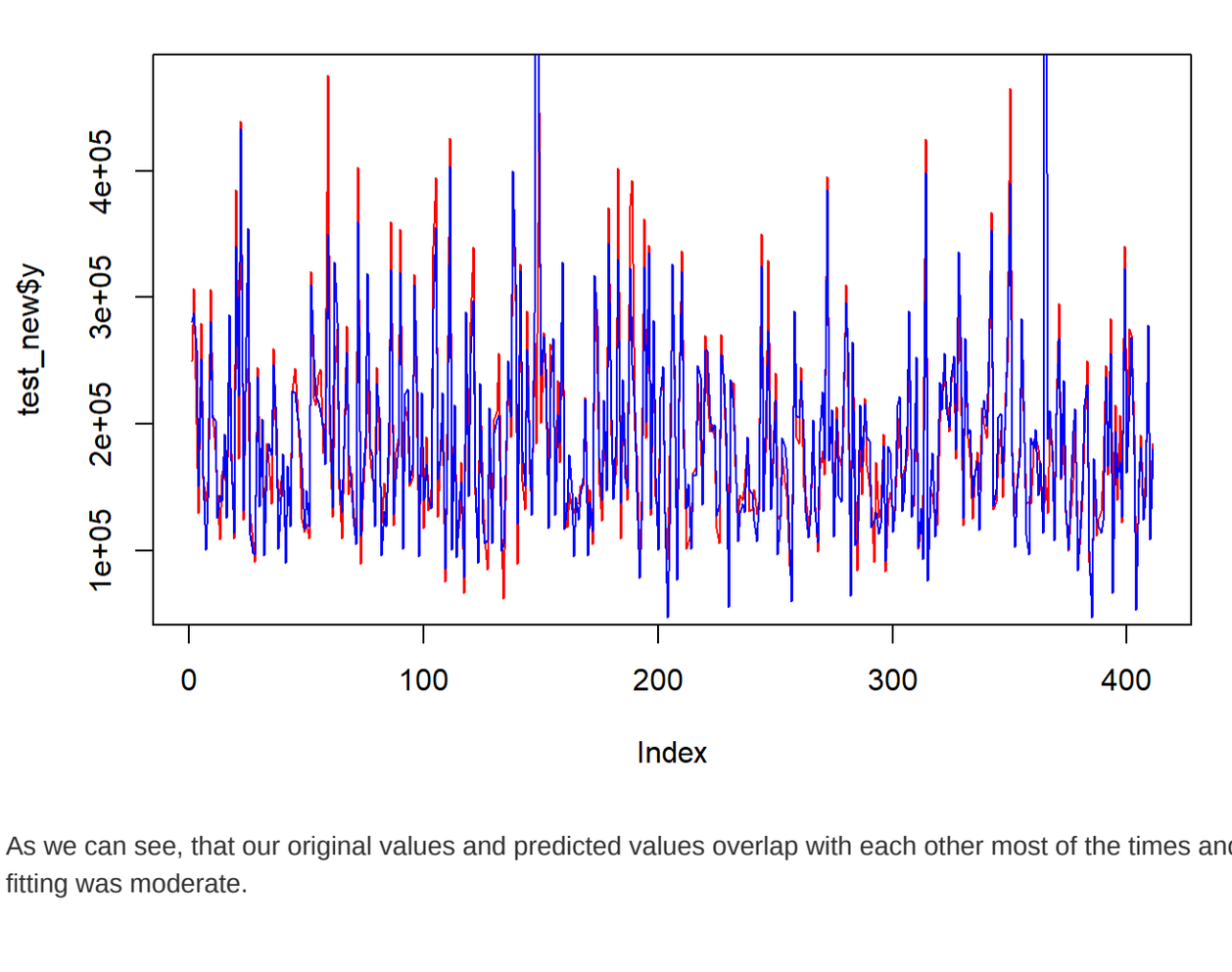
```
##   [1] 280943.33 287548.97 258937.40 151380.98 251183.73 159196.66 101226.94
##   [8] 141384.81 281172.38 205518.60 203176.81 126179.67 144110.12 139416.28
##  [15] 191873.36 126207.79 286427.07 171435.80 114055.36 340288.77 222383.33
##  [22] 433084.08 132053.98 225786.72 354470.22 113090.02  99004.21  97464.71
##  [29] 237539.51 134643.19 204208.69  96374.14 172202.00 184644.12 175872.63
##  [36] 247241.84 197212.12 102003.75 142295.31 176172.60  90312.45 166895.07
##  [43] 119503.36 225851.40 224696.31 209667.22 189826.34 166039.07 117384.09
##  [50] 147318.13 119174.86 310300.18 251228.98 222171.18 217398.25 209719.05
##  [57] 197371.53 168412.10 350254.81 215504.19 134478.95 328123.14 280542.90
##  [64] 157110.81 130340.65 182578.42 256715.50 175898.71 150542.07 126198.98
##  [71] 105168.78 359773.74 112069.71 154285.61 188769.40 319043.96 181120.02
##  [78] 175461.77 119258.50 231960.99 205268.45  96393.48 131610.69 119795.31
##  [85] 201188.62 322175.91 129391.57 156172.77 182529.59 319476.43 101778.69
##  [92] 223927.14 227524.76 154086.94 167518.01 309849.06 226084.72  96059.84
##  [99] 224881.21 140016.60 168429.86 142416.71 133637.76 290656.22 355045.68
## [106] 156463.97 178285.44 224861.31  85976.14 171405.10 403380.59 100841.65
## [113] 215049.10  94753.54 140560.92 154465.51  79583.83 288730.61 142999.91
## [120] 255983.86 297482.55 150512.72  90283.05 232078.67 121842.54 106016.15
## [127] 108664.67 212735.06 106407.64 193043.09 203205.75 206947.30  99901.83
## [134] 105375.43 186741.02 249937.23 206284.24 400007.76 310376.13 121366.87
## [141] 320728.04 178886.72 151433.21 258708.89 172998.77 128423.20 287590.96
## [148] 700839.89 414992.30 238595.39 269384.31 244020.52 117630.28 228710.02
## [155] 267813.56 128426.37 207998.39 185258.73 327909.05 116904.90 137807.19
## [162] 175783.42 146344.82  95598.91 142087.46 124321.34 144423.43 157572.78
## [169] 219544.90  96667.95 138470.88 115554.91 317118.97 278459.29 173299.90
## [176] 140313.89 219086.83 147414.94 342243.36 204220.50 140899.67 190000.08
## [183] 330105.29 137615.23 234971.60 157076.11 150107.81 323361.56 246724.59
## [190] 209829.48 147083.08  78765.75 124139.06 324215.42 201970.32 334844.10
## [197] 133754.57 281787.32 157764.43 100891.81 221443.59 245538.30 129971.76
## [204]  47800.49 121006.16 326233.99 246699.41  77448.43 220436.91 320352.46
## [211] 180797.70 133858.85 152746.43 101842.59 159877.52 159724.74 245987.82
## [218] 236956.59 136151.92 257862.96 257857.70 206737.70 194148.47 199264.99
## [225] 127935.11 136653.26 255328.27 232964.38 193581.27  55331.54 235462.69
## [232] 213874.46 176328.78 107232.74 130172.57 137179.70 130239.49 189870.00
## [239] 148410.75 145291.77 123809.80 107427.30 137619.96 324593.48 130960.51
## [246] 219041.54 273827.81 132912.34 168028.57 218661.57  97375.59 126989.69
## [253] 189087.82 182316.39 165477.08 101549.36  60311.68 289188.23 206748.54
## [260] 205806.65 234511.69 189117.39 130485.98 110886.03 128623.18 203189.56
## [267] 131590.10 106956.82 194612.11 225549.99 172556.70 384738.71 171675.79
## [274] 211407.01 111424.63 203601.38 143894.31 138571.15 214256.44 295686.88
## [281] 256996.43  64747.10 264746.02 104251.45 108532.31 215270.85 165879.67
## [288] 215473.57 189992.73 186635.75 118545.52 123679.53 130643.00 113179.84
## [295] 121826.19 183368.21  91859.38 182614.46 178230.72 114859.69 131495.85
## [302] 215147.29 221543.53 131481.19 144621.67 197979.98 289103.89 126950.23
## [309] 143963.83 252620.51 103240.60 133325.04  93277.49 397980.53  76233.00
## [316] 150680.61 177315.77 110972.42 146313.74 232601.88 211117.20 256104.58
## [323] 212075.67 197808.87 237327.66 253916.29 180078.00 335978.55 269228.53
## [330] 126032.66 267761.14 192261.17 196110.54 142337.91 159703.26 170532.37
## [337] 116219.18 205329.26 218754.91 198679.76 249242.09 353026.96 135835.71
## [344] 148165.87 205276.56 208484.91 157701.18 213608.33 268218.98 390324.65
## [351] 174996.55 103160.51 149124.71 173473.58 283161.54 186153.58 112716.58
## [358]  97366.46 189138.22 180862.90 195452.64 144250.90 170153.56 113971.78
## [365] 976946.18 188292.25 210359.47 170823.75 108448.44 208125.12 267888.16
## [372] 157718.10 234505.54 132605.82 101737.42 139209.06 194654.57 212357.98
## [379]  84297.52 110098.62 155262.20 214723.63 231174.49 100030.72  47249.99
## [386] 172603.43 125726.81 119347.30 114394.99 123734.48 237326.09 167100.19
## [393] 256181.22  66875.43 194243.31 157634.58 170221.34 127158.96 322371.27
## [400] 161392.66 222184.61 268419.45 176871.13  53117.63 156487.02 181535.17
## [407] 124602.57 142685.44 277724.27 109293.92 180080.93
## attr(,"label")
## [1] "Predicted values"
```

```
length(pred)
```

```
## [1] 411
```

Now, we need to compare our predicted values with the original values to check the efficiency of the model and to make a comment on how good our fit is.

```
plot(test_new$y,type="l",lty=1.8,col="red")
lines(pred,type="l",lty=1.8,col="blue")
```



As we can see, that our original values and predicted values overlap with each other most of the times and hence, we can conclude that our model fitting was moderate.