PREDICTIVE MODEL BUILDING USING LINEAR REGRESSION

~ Based on a House Pricing Dataset
Done in RStudio

Let's start by including some packages first!!!

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

P.S. Extremely sorry for the image quality..kindly deal with it!! :(

After loading the dataset, remember to split it into Training and Testing datasets... And this should be done randomly!!!

P.S. - Predictive models are build based on training 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 traini
ng 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
```

Go ahead!! Build a primary model on the training data!!

OOPPSSS!!!! Some cleansing is necessary... For more efficient predictions!!!

Let's start by finding out some influential observations...
Pro Tip:- The concept of Cook's Distance is used to do so..

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.

Let's find the <u>outliers</u> and <u>high leverage values</u> as well... This is done by the concepts of <u>Studentized Residuals</u> and <u>Hat Matrix</u>..These all are data impurities!!

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
```

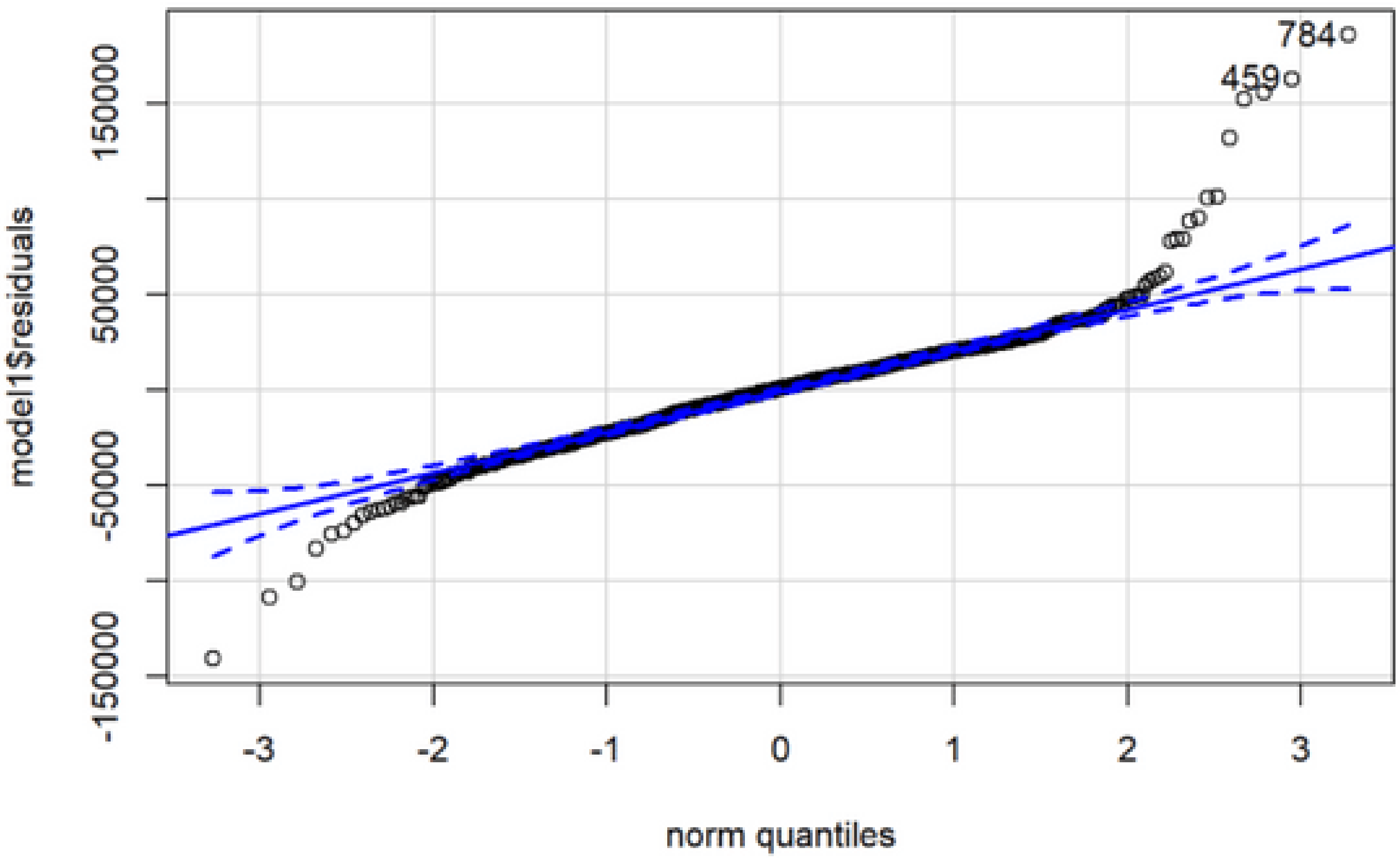
We know, that the potential high leverage values are the values that have hat values more than (3p/n), where n is the no. of observations, that is 953 in this case and p is the no. of covariates, that is 63 in this case.

WARNING:- Remove the impurities carefully...Data is precious and huge loss of data will be disadvantageous!!!

- After data cleansing, build another model based on the new dataset

Now let's check whether our model satisfies the condition of normality... It is done using the QQPlot

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



Hence, normality condition is satisfied!!!!

Isn't the plot highly satisfying??!!!

Now, let's check for presence of autocorrelation.

Durbin Watson test is used to determine this..

```
dwtest(model1)
```

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

Yayyy!!! No autocorrelation is present in the model as p value > 0.05!!!!

Now, let's find out whether our model is homoscedastic or not.

Breusch Pagan test is used to determine this

```
bptest(model1)
```

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

Alas!!! Our model turns out to be heteroscedastic as p value < 0.05!!!!!!

Don't worry!!! We will fix this in a bit using GLS!!!!

Let's find out whether our model has high multicollinearity or not!!!!

This is going to be interesting... **Variance Inflation Factor (VIF)** is gonna help us do that!!!

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
```

Another good news!!!! Since VIF values aren't >10, that means absence of high multicollinearity!!!!

As mentioned earlier... Let's use GLS to fix the issue of heteroscedasticity and build a new and improved predictive model using the significant covariates!!

```
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.

Time for the final reveal!!!!
The final predictive model looks like this.....

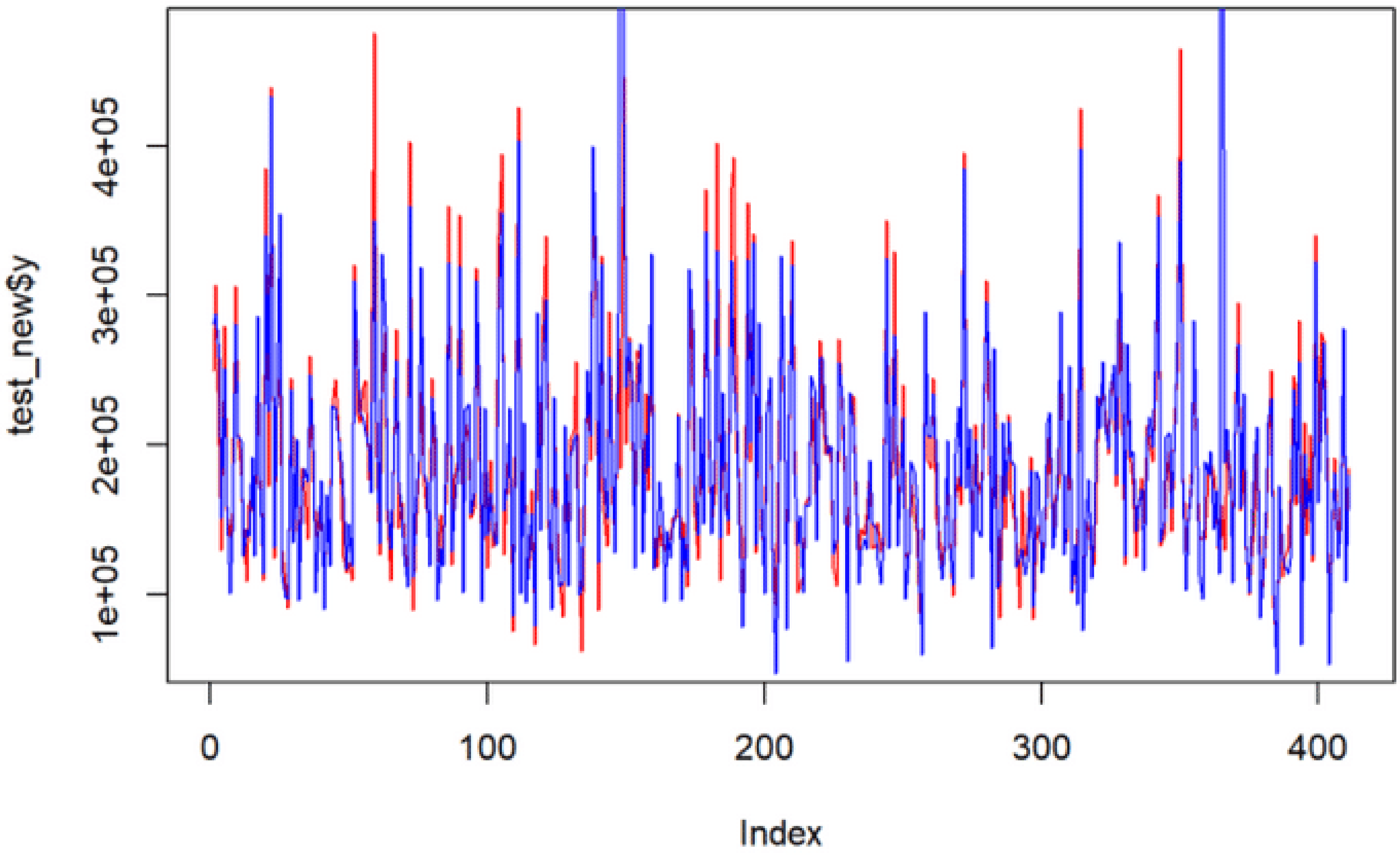
```
model3=gls(y~x3+x4+x5+x12+x13+x14+x15+x17+x18+x20+x21+x22+x26+x27+x29+x31+x32+x34+x35+x41+x42+x43+x45+x47+x48+x63, data = tr
ain_new2, correlation = corAR1())

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
```

The model may look complicated...but believe me it's not!!!

But our job isn't done yet!!! We need to make use of our model. Let's perform prediction with the help of our model on the testing dataset and compare the results with the original values. Now that sounds like fun!!!!

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



Another satisfying plot!!! We can call this a moderate fitting!!!
And our job is done!!!