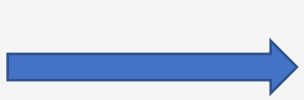
PREDICTIVE MODEL BUILDING USING LINEAR REGRESSION

- PRINCIPAL COMPONENT ANALYSIS (included)

~ Based on Life Expectancy Dataset
Done in RStudio

Let’s start by including our necessary packages..

```
library(readxl)
library(qpcR)
library(car)
library(carData)
library(nlme)
library(lmtest)
library(BSDA)
library(MASS)
library(ROCR)
library(rmarkdown)
library(pls)
library(psych)
library(Metrics)
```



These two packages are used to perform **Principal Component Analysis**

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(123)
sample <- floor(0.75 * nrow(df1))
train_ind <- sample(seq_len(nrow(df1)), size = sample)
train <- df1[train_ind, ]
test <- df1[-train_ind, ]
dim(train)
```

```
## [1] 2203   19
```

```
dim(test)
```

```
## [1] 735  19
```

Go ahead!!! Use the following code and build your own training and testing datasets….
And also build a primary linear model based on the training dataset….

Let's do some data cleansing!!!!

```
### cooks distance ###

cook = cooks.distance(Life_model)
c = cook[cook>(4/2203)]
length(c)
```

```
## [1] 145
```

```
# Influential observations

### Studentized residual###

student = studres(Life_model)
s = student[abs(student)>3]
length(s)
```

```
## [1] 22
```

```
### high leverage ###

hat = hatvalues(Life_model)
h = hat[hat>(54/2203)]
length(h)
```

```
## [1] 95
```

Cook's Distance is used to find out the influential observations. If the Cook's distance of an observation is > 4/n, then it's a potential influential observation.

Studentized Residuals is used to find out the outliers. If the absolute value of Studentized residual of a certain observation is > 3, then it's considered as an outlier.

Hat Matrix and Hat Values are used to find the high leverage values. If the hat values of a certain observation is > 3p/n, where p is the no. of covariates, then it's referred to as a high leverage value.

P.S. : Remove the impurities carefully...Data is precious and huge loss of data will be disadvantageous!!! Create another linear model based on cleansed dataset!!!!

TIME TO CHECK FOR MULTICOLLINEARITY!!!!

```
vif(Life_model1)
```

```
##         x1         x2         x3         x4         x5         x6         x7
##   1.751867 135.636949   1.573369   5.097566   1.393686   1.371874   1.697805
##         x8         x9        x10        x11        x12        x13        x14
## 136.371314   2.003802   1.165024   2.272051   1.458114   5.283758   1.455168
##        x15        x16        x17        x18
##   9.147116   9.199946   2.929374   3.245281
```

We will use the Variance Inflation Factor (VIF) to determine multicollinearity.
There's high multicollinearity present in the covariates x2 and x8 as the VIF values for these two covariates are > 10.

Time to introduce PRINCIPAL COMPONENT ANALYSIS!!!! The one who saves from the issue of multicollinearity!!!

PRINCIPAL COMPONENT ANALYSIS

Note: There's no response variable in this model, and this is not a mistake. We will just have to pass the covariates as a parameter to the prcomp function.

```
pc.fit <- prcomp(~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+x11+x12+x13+x14+x15+x16+x17+x18,data=newdata, scale=TRUE)
summary(pc.fit)
```

```
## Importance of components:
##                           PC1    PC2     PC3    PC4     PC5     PC6     PC7
## Standard deviation     2.3419 1.5822 1.32159 1.1758 1.10846 0.93407 0.91437
## Proportion of Variance 0.3047 0.1391 0.09703 0.0768 0.06826 0.04847 0.04645
## Cumulative Proportion  0.3047 0.4438 0.54080 0.6176 0.68586 0.73433 0.78078
##                           PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     0.88619 0.78262 0.75628 0.69300 0.65755 0.63083 0.54871
## Proportion of Variance 0.04363 0.03403 0.03178 0.02668 0.02402 0.02211 0.01673
## Cumulative Proportion  0.82441 0.85844 0.89021 0.91689 0.94091 0.96302 0.97975
##                           PC15    PC16    PC17    PC18
## Standard deviation     0.44834 0.31948 0.24033 0.06074
## Proportion of Variance 0.01117 0.00567 0.00321 0.00020
## Cumulative Proportion  0.99092 0.99659 0.99980 1.00000
```

These are the summaries of the 18 principal components that are created since there were 18 covariates at first.

We can select the number of principal components we want to work with based on the cumulative proportion of variance explained by these principal components.

- We will use 11 principal components instead of 18 principal components, because with the help of the first 11 principal components, we can already explain 90% of the data. That much information is more than enough to build an efficient predictive model. (# Dimension reduction). You can use a lesser number of principal components as well. Anything more than 75% will be fine.
- There's another method called the Screeplot, which also helps in choosing an appropriate number of principal components.

```r
trans_test <- as.data.frame(predict(pc.fit, test)[,1:11])
new_train <- as.data.frame(cbind(newdata$y, pc.fit$x[,1:11]))
colnames(new_train)[1]<- "Life.expectancy"

pcr_lm_model <- lm(new_train$Life.expectancy~., data=new_train)
summary(pcr_lm_model)
```

```
##
## Call:
## lm(formula = new_train$Life.expectancy ~ ., data = new_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -21.4894  -2.3332  -0.0763   2.3386  16.6319
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 69.26587    0.09094 761.657  < 2e-16 ***
## PC1          3.16884    0.03884  81.584  < 2e-16 ***
## PC2         -1.78513    0.05749 -31.051  < 2e-16 ***
## PC3          1.07189    0.06883  15.574  < 2e-16 ***
## PC4         -1.18788    0.07736 -15.354  < 2e-16 ***
## PC5          1.86202    0.08206  22.690  < 2e-16 ***
## PC6         -0.55323    0.09738  -5.681 1.52e-08 ***
## PC7         -0.59047    0.09948  -5.936 3.41e-09 ***
## PC8          1.49775    0.10264  14.592  < 2e-16 ***
## PC9         -0.21678    0.11623  -1.865   0.0623 .
## PC10        -0.47102    0.12028  -3.916 9.28e-05 ***
## PC11         0.22775    0.13126   1.735   0.0829 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.227 on 2148 degrees of freedom
## Multiple R-squared:  0.8058, Adjusted R-squared:  0.8049
## F-statistic: 810.5 on 11 and 2148 DF,  p-value: < 2.2e-16
```

Using the pc.fit model, we linearly transform the original testing dataset i.e., the new testing dataset contains just 11 principal components containing the linear combinations of each previous testing dataset observations.
Ex:- PCi = f(X1i, X2i, X3i, ….. X18i) for all I = 1, 2, …., 11 (because of 11 principal components)

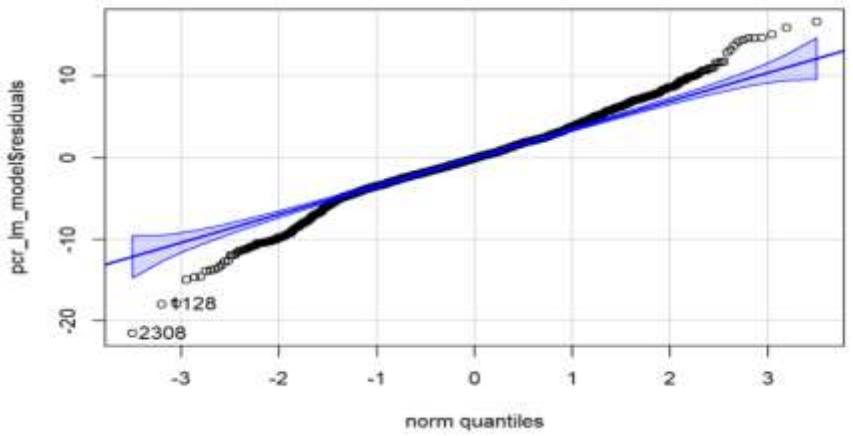
Creating a new training dataset as well which contains the 11 principal components that we have chosen along with the response variable from the training dataset.

Now, we have a new training dataset and a new testing dataset. All there's left to do is build a new linear model based on the training dataset and check the necessary tests and then move on to performing predictions.

Note: THE CREATION OF NEW DATASETS USING THE PRINCIPAL COMPONENTS IS A VERY IMPORTANT STEP. MISSING THIS ONE OUT WILL LEAD TO SERIOUS LOGICAL ERRORS WHILE PERFORMING PREDICTIONS!!!!
NOT TRYING TO SCARE YOU, JUST STATING THE FACT FROM MY OWN EXPERIENCE… ☹ ☹ ☹

The Adjusted R Square of the new linear model turns out to be 0.8049, which is quite good and it states that our new model is quite efficient as well..

```
qqPlot(pcr_lm_model$residuals) # normality assumption is satisfied
```



TEST FOR NORMALITY ASSUMPTION

We can see that most quantiles of our model fits moderately with the quantiles of any normal sample.
Thus, we can conclude, that our model somewhat satisfies the condition of normality!!! ☺ ☺
(Satisfying plot…isn't it??!!)

```
vif(pcr_lm_model) #issue of multicollinearity solved
```

```
##  PC1  PC2  PC3  PC4  PC5  PC6  PC7  PC8  PC9 PC10 PC11
##    1    1    1    1    1    1    1    1    1    1    1
```

TEST FOR MULTICOLLINEARITY

As we can see that after doing Principal Component Analysis, all the VIF values turn out to be 1, which clearly concludes that there is no multicollinearity in our model.

```
bptest(pcr_lm_model) # bptest fails. Need to opt for gls
```

```
##
##  studentized Breusch-Pagan test
##
## data:  pcr_lm_model
## BP = 189.56, df = 11, p-value < 2.2e-16
```

TEST FOR HOMOSCEDASTICITY

We can observe that p value is much less than 0.05, hence, we can say that our homoscedasticity claim is rejected!! ☹ ☹
We need to opt for GLS to fix this issue.

FINAL REVEAL OF THE PREDICTIVE LINEAR MODEL....

```
Life_model_gls = gls(Life.expectancy~., correlation = corAR1(), data=new_train)
Life_model_gls
```

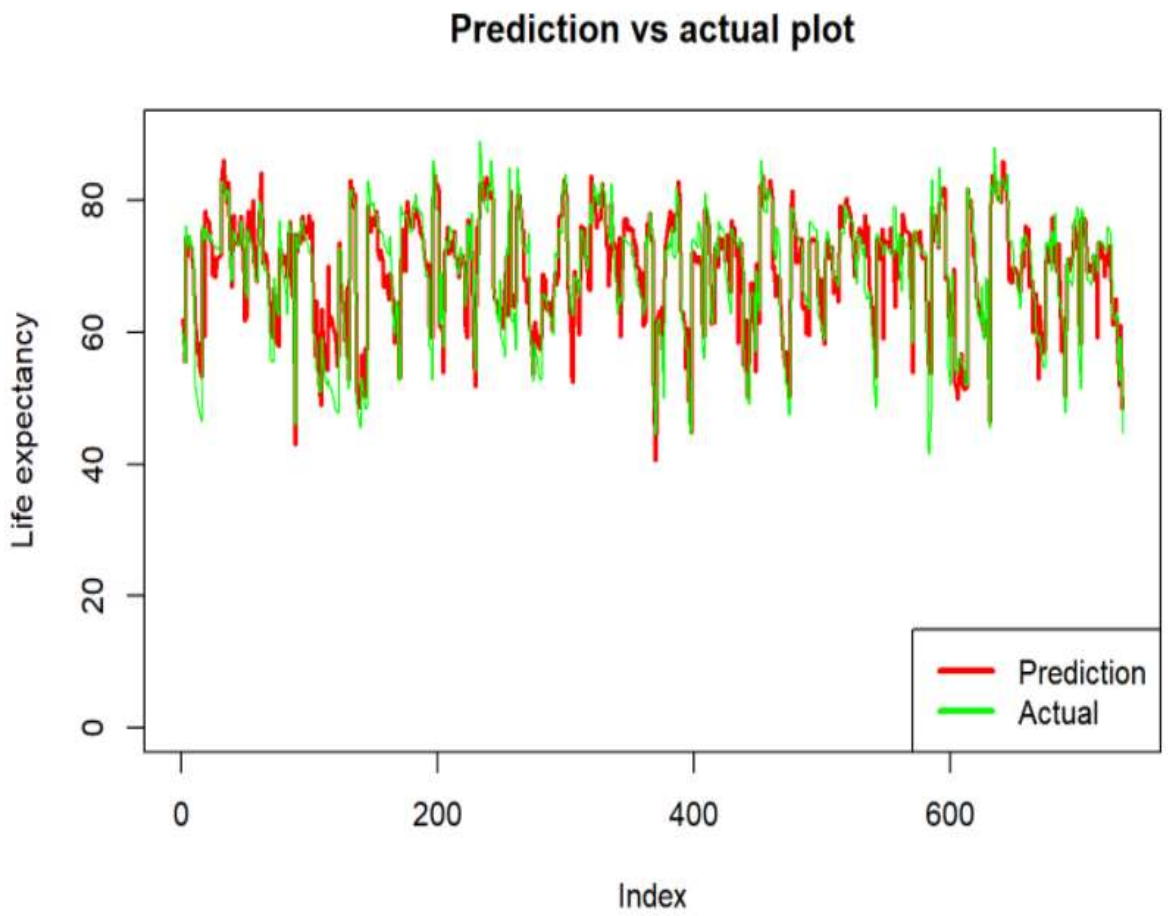
```
## Generalized least squares fit by REML
##   Model: Life.expectancy ~ .
##   Data: new_train
##   Log-restricted-likelihood: -6190.707
##
## Coefficients:
## (Intercept)         PC1         PC2         PC3         PC4         PC5
##  69.2658630   3.1685844  -1.7855303   1.0709555  -1.1894460   1.8619931
##         PC6         PC7         PC8         PC9        PC10        PC11
##  -0.5531998  -0.5907971   1.4986661  -0.2163998  -0.4707751   0.2262706
##
## Correlation Structure: AR(1)
##  Formula: ~1
##  Parameter estimate(s):
##          Phi
## -0.004442001
## Degrees of freedom: 2160 total; 2148 residual
## Residual standard error: 4.226547
```

```
Rsq.ad(Life_model_gls)
```

```
## [1] 0.8048457
```

The Adjusted R Square value is 0.8048457, which means that our predictive model is highly efficient!!! HOOORAYYYY!!!!

```r
# Predicting the fitted model on test dataset
pred1 <- predict(Life_model_gls, trans_test)
plot(pred1, col="red", type="l", lwd=2, ylab="Life expectancy", main= "Prediction vs actual plot", ylim=c(0,90))
lines(test$y, col="green", type="l", lwd="1")
legend(x="bottomright", legend = c("Prediction","Actual"), col=c("red","green"), cex=1, lty = 1, lwd=3)
```



As we can see that, the line diagram of our predictions almost overlaps the line diagram of the original response variables of the testing dataset.

Thus we can draw the one and only conclusion, which is….

Our model proves to be highly efficient as the fitting of the model is "moderate"…. ☺ ☺

Yet another satisfying plot…with a scent of success!!! XD XD