Lens Quality Result Interpretation (R Code)

Here, I am going to explain the code, my analysis method and the final result interpretation. The Sample Dataset consists of 28 features and 117 data rows. The response variable is Lens Quality and other are predictors. The data consist of two types of predictor: one is process setting factor that is controllable and another is questionable input factor.

I have used R programming tool to analyze this dataset.

Analysis Procedure:-

- Identification of missing data, zero-optional variables, and correlated predictors

  Firstly, we checked the missing data and zero-optional variables in the sample datasets, and we found that data is really good and we don't need to deal with missing data & zero optional variable problems. However, we found that 6 predictors (MSg, Aid1G, FSg, Aid2g, MSBCg, and Energy) are correlated to other predictors.

- Model selection, Model Training, and Model evaluation

  Here, I applied two machine learning models Partial Least Square (PLS) & Support Vector Machine (SVM) and choose the results of the SVM model at the end which is giving us high R-squared value & low root mean square error (RMSE).

- Result Interpretation

  Once I found the best model I considered its result of important factors in determining the lens quality. Radius, mold, and length are the top 3 important factors.

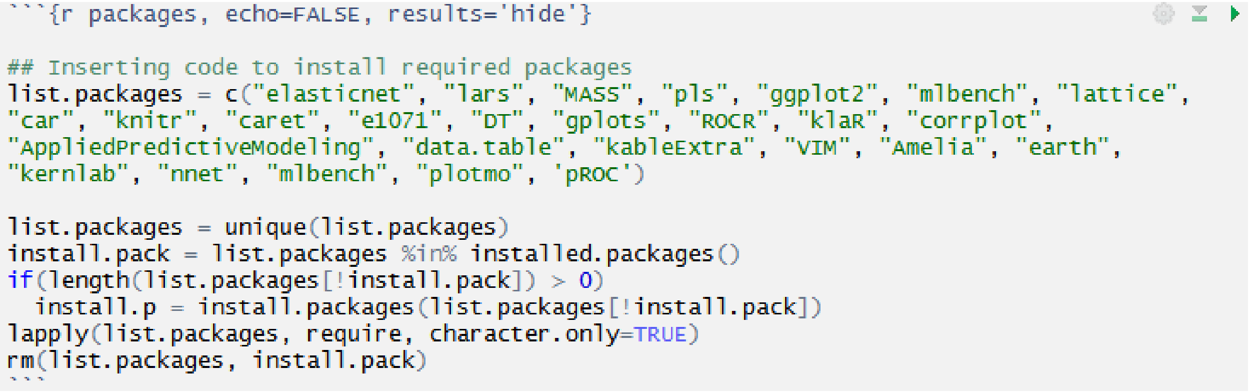
Assumptions:-

I did all the analysis on the basis of below assumptions:-

- Every factor is independent and important.
- The highest value of Lens quality tells that the lens is best among others.
- Considered both the predictors process setting and questionnaire input together for analysis.
- We haven't considered label as a predictor as it is showing distinct categorical values.

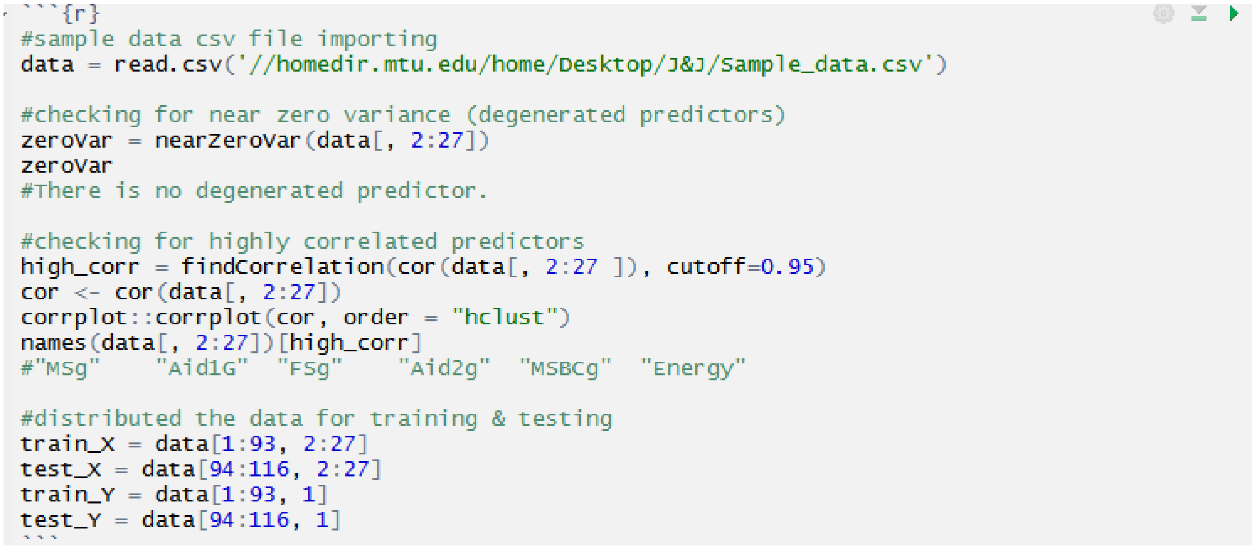
Explanation of the code:-

1) Firstly, I installed all the packages required for the analysis in Rstudio as mention in below code.



~~~
```{r packages, echo=FALSE, results='hide'}

## Inserting code to install required packages
list.packages = c("elasticnet", "lars", "MASS", "pls", "ggplot2", "mlbench", "lattice",
"car", "knitr", "caret", "e1071", "DT", "gplots", "ROCR", "klaR", "corrplot",
"AppliedPredictiveModeling", "data.table", "kableExtra", "VIM", "Amelia", "earth",
"kernlab", "nnet", "mlbench", "plotmo", 'pROC')

list.packages = unique(list.packages)
install.pack = list.packages %in% installed.packages()
if(length(list.packages[!install.pack]) > 0)
  install.p = install.packages(list.packages[!install.pack])
lapply(list.packages, require, character.only=TRUE)
rm(list.packages, install.pack)
```
~~~

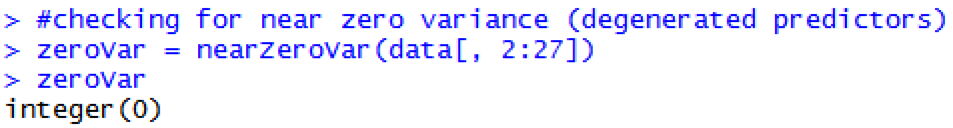
2) Then I imported the sample dataset csv file, checked the near zero variance and correlated predictors by the following R code.



~~~
```{r}
#sample data csv file importing
data = read.csv('//homedir.mtu.edu/home/Desktop/J&J/Sample_data.csv')

#checking for near zero variance (degenerated predictors)
zeroVar = nearZeroVar(data[, 2:27])
zeroVar
#There is no degenerated predictor.

#checking for highly correlated predictors
high_corr = findCorrelation(cor(data[, 2:27 ]), cutoff=0.95)
cor <- cor(data[, 2:27])
corrplot::corrplot(cor, order = "hclust")
names(data[, 2:27])[high_corr]
#"MSg"    "Aid1G"  "FSg"    "Aid2g"  "MSBCg"  "Energy"

#distributed the data for training & testing
train_X = data[1:93, 2:27]
test_X = data[94:116, 2:27]
train_Y = data[1:93, 1]
test_Y = data[94:116, 1]
```
~~~

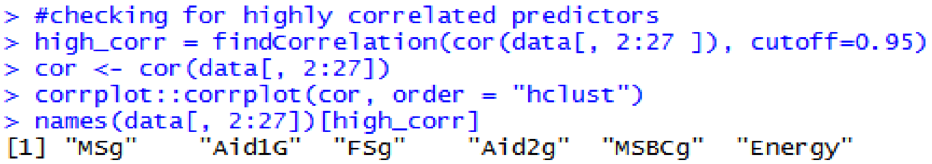
Sometimes predictors may appear with a single unique value or several such values occurring at a very low frequency. This is called near zero variance. For many models, it may lead to a collapse or instable operation. These near-zero-variance predictors must be identified and eliminated before the simulation.

Here, we don't have any near zero variance in this case as shown below.

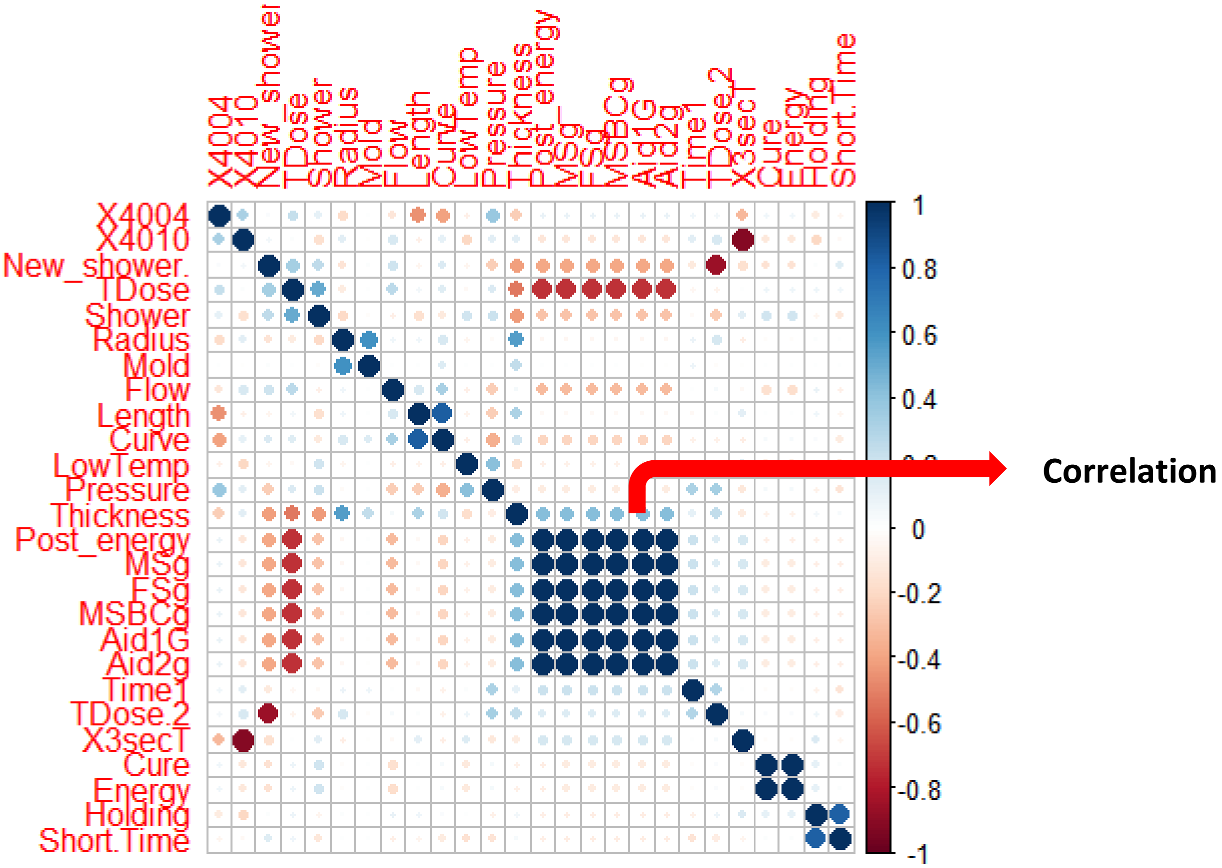


```
> #checking for near zero variance (degenerated predictors)
> zeroVar = nearZeroVar(data[, 2:27])
> zeroVar
integer(0)
```

Now, I checked highly correlated variables and we can clearly see in below figure that 6 predictors ["MSg" "Aid1G" "FSg" "Aid2g" "MSBCg" "Energy"] are correlated to other predictors. So even if we don't consider these predictors, it will not affect len's quality.



```
> #checking for highly correlated predictors
> high_corr = findCorrelation(cor(data[, 2:27 ]), cutoff=0.95)
> cor <- cor(data[, 2:27])
> corrplot::corrplot(cor, order = "hclust")
> names(data[, 2:27])[high_corr]
[1] "MSg"     "Aid1G"   "FSg"     "Aid2g"   "MSBCg"   "Energy"
```

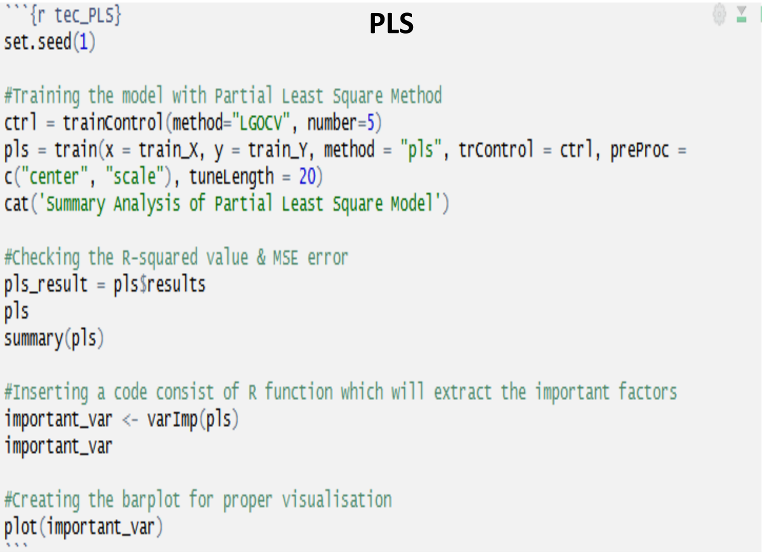


We can see here Cure & Energy are correlated to each other. Apart from this, it is clearly seen that Post energy, MSg, FSg, MSBCG, Aid1G, Aid2g are correlated to each other in the mid blue area.

3) Now, we come to the third step. Here, I used 2 models which could be suitable for this type of data. Then, I checked the R-squared value and MSE for each model.

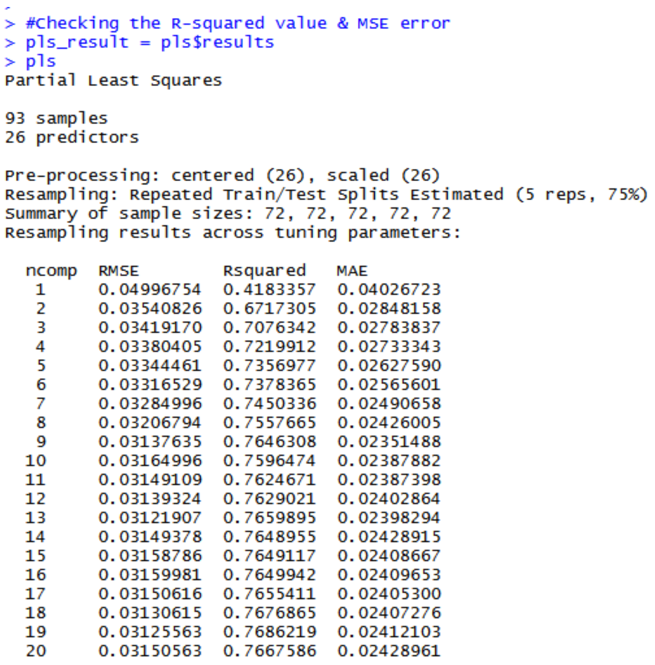
And, I found that using support vector machine, we are getting high R-squared value and low root mean square error as shown below.

This is why I preferred SVM results over Partial least square method.

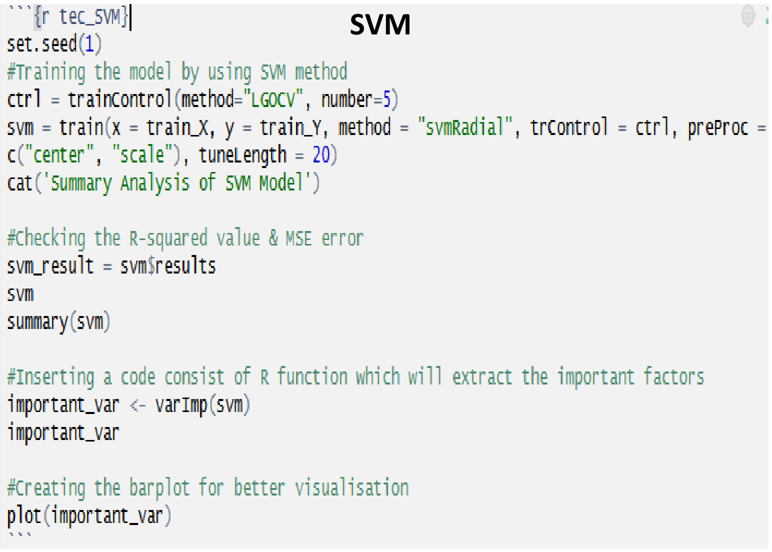


PLS

~~~
```{r tec_PLS}
set.seed(1)

#Training the model with Partial Least Square Method
ctrl = trainControl(method="LGOCV", number=5)
pls = train(x = train_X, y = train_Y, method = "pls", trControl = ctrl, preProc =
c("center", "scale"), tuneLength = 20)
cat('Summary Analysis of Partial Least Square Model')

#Checking the R-squared value & MSE error
pls_result = pls$results
pls
summary(pls)

#Inserting a code consist of R function which will extract the important factors
important_var <- varImp(pls)
important_var

#Creating the barplot for proper visualisation
plot(important_var)
```
~~~

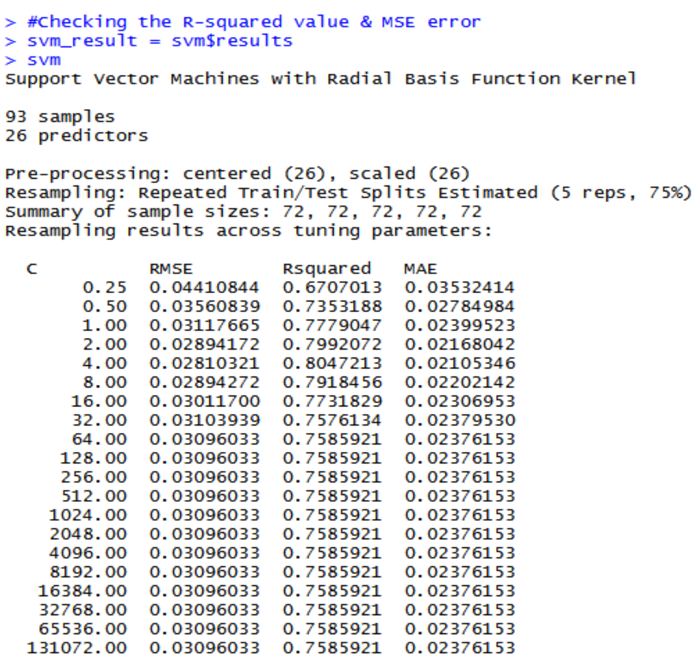


```
> #Checking the R-squared value & MSE error
> pls_result = pls$results
> pls
Partial Least Squares

93 samples
26 predictors

Pre-processing: centered (26), scaled (26)
Resampling: Repeated Train/Test Splits Estimated (5 reps, 75%)
Summary of sample sizes: 72, 72, 72, 72, 72
Resampling results across tuning parameters:

  ncomp  RMSE        Rsquared   MAE
   1     0.04996754  0.4183357  0.04026723
   2     0.03540826  0.6717305  0.02848158
   3     0.03419170  0.7076342  0.02783837
   4     0.03380405  0.7219912  0.02733343
   5     0.03344461  0.7356977  0.02627590
   6     0.03316529  0.7378365  0.02565601
   7     0.03284996  0.7450336  0.02490658
   8     0.03206794  0.7557665  0.02426005
   9     0.03137635  0.7646308  0.02351488
  10     0.03164996  0.7596474  0.02387882
  11     0.03149109  0.7624671  0.02387398
  12     0.03139324  0.7629021  0.02402864
  13     0.03121907  0.7659895  0.02398294
  14     0.03149378  0.7648955  0.02428915
  15     0.03158786  0.7649117  0.02408667
  16     0.03159981  0.7649942  0.02409653
  17     0.03150616  0.7655411  0.02405300
  18     0.03130615  0.7676865  0.02407276
  19     0.03125563  0.7686219  0.02412103
  20     0.03150563  0.7667586  0.02428961
```



SVM

~~~
```{r tec_SVM}
set.seed(1)
#Training the model by using SVM method
ctrl = trainControl(method="LGOCV", number=5)
svm = train(x = train_X, y = train_Y, method = "svmRadial", trControl = ctrl, preProc =
c("center", "scale"), tuneLength = 20)
cat('Summary Analysis of SVM Model')

#Checking the R-squared value & MSE error
svm_result = svm$results
svm
summary(svm)

#Inserting a code consist of R function which will extract the important factors
important_var <- varImp(svm)
important_var

#Creating the barplot for better visualisation
plot(important_var)
```
~~~



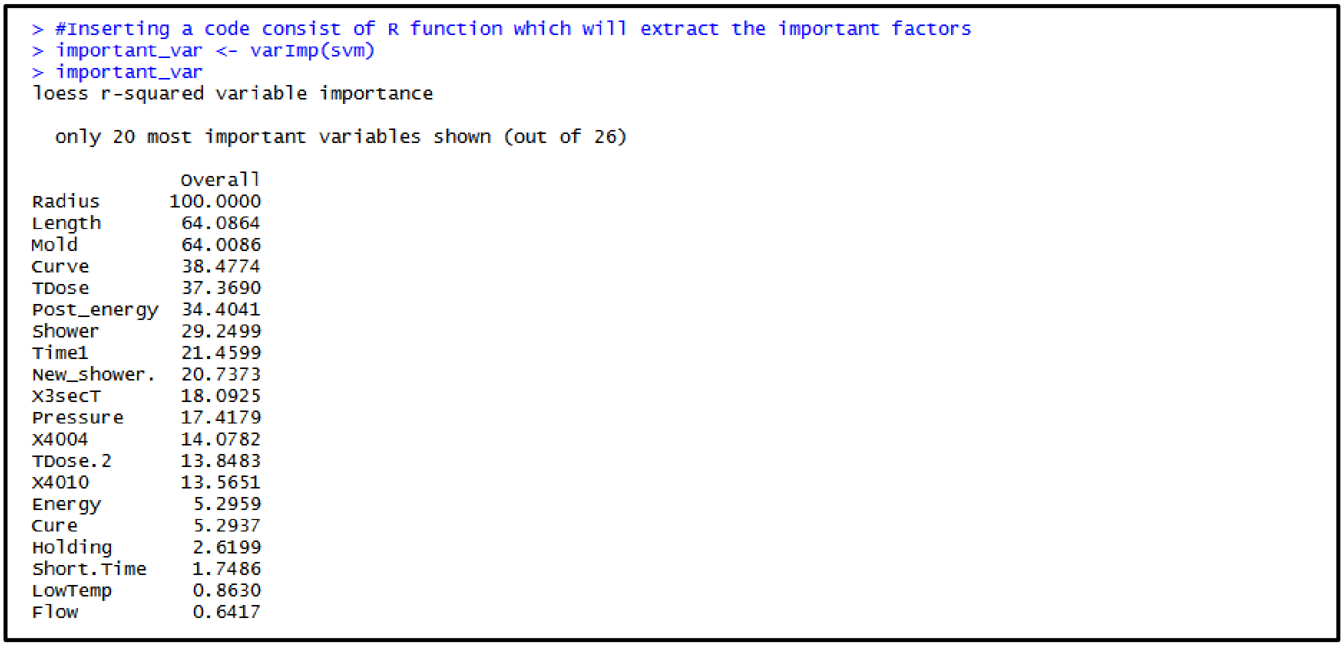
```
> #Checking the R-squared value & MSE error
> svm_result = svm$results
> svm
Support Vector Machines with Radial Basis Function Kernel

93 samples
26 predictors

Pre-processing: centered (26), scaled (26)
Resampling: Repeated Train/Test Splits Estimated (5 reps, 75%)
Summary of sample sizes: 72, 72, 72, 72, 72
Resampling results across tuning parameters:

  C          RMSE        Rsquared   MAE
       0.25  0.04410844  0.6707013  0.03532414
       0.50  0.03560839  0.7353188  0.02784984
       1.00  0.03117665  0.7779047  0.02399523
       2.00  0.02894172  0.7992072  0.02168042
       4.00  0.02810321  0.8047213  0.02105346
       8.00  0.02894272  0.7918456  0.02202142
      16.00  0.03011700  0.7731829  0.02306953
      32.00  0.03103939  0.7576134  0.02379530
      64.00  0.03096033  0.7585921  0.02376153
     128.00  0.03096033  0.7585921  0.02376153
     256.00  0.03096033  0.7585921  0.02376153
     512.00  0.03096033  0.7585921  0.02376153
    1024.00  0.03096033  0.7585921  0.02376153
    2048.00  0.03096033  0.7585921  0.02376153
    4096.00  0.03096033  0.7585921  0.02376153
    8192.00  0.03096033  0.7585921  0.02376153
   16384.00  0.03096033  0.7585921  0.02376153
   32768.00  0.03096033  0.7585921  0.02376153
   65536.00  0.03096033  0.7585921  0.02376153
  131072.00  0.03096033  0.7585921  0.02376153
```

As we can see above, PLS model is showing high root mean square value and low R-squared value in comparison to Support Vector Machine model. On the basis of this result, I am considering important factors shown by SVM model.

4) Finally, we have considered the important factor values shown by SVM and made a bar chart for the better visualization as shown below.

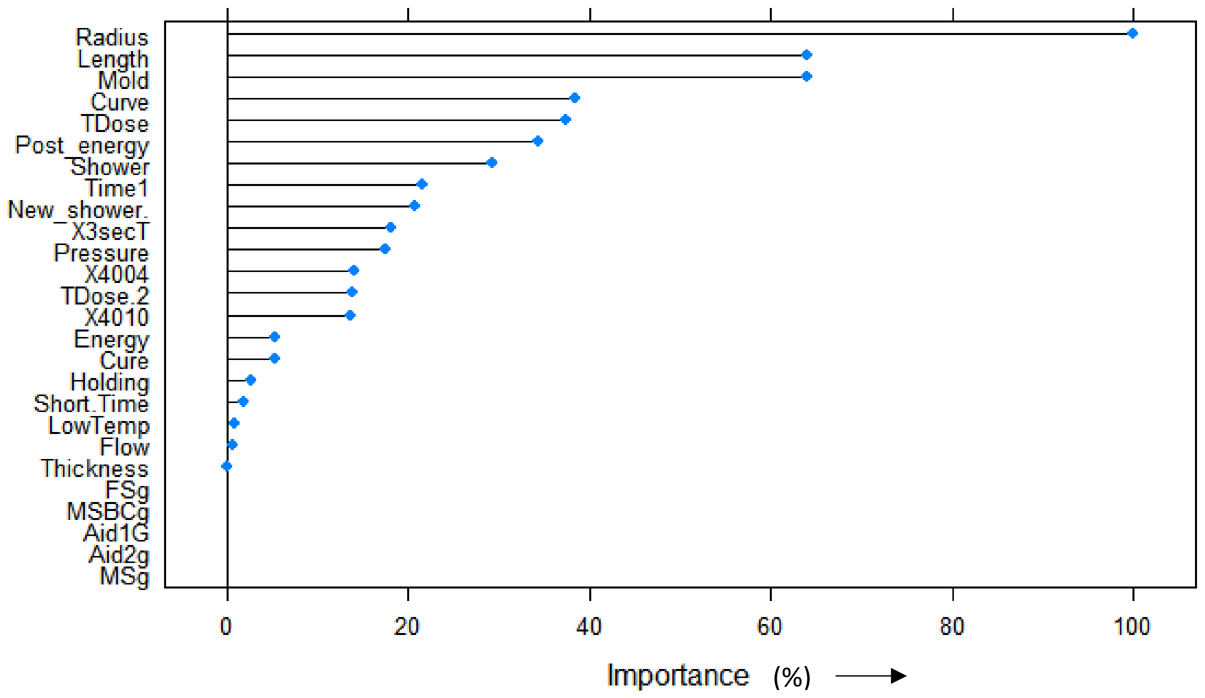
As we can see below, Top 5 most important factors in determining the lens quality are: - Radius (100 %), Length (64.08%), Mold (64.08%), Curve (38.48%), and Tdose (37.37%).

The factors that are totally useless as shown by the model are: - FSg, MSBCg, Aid1G, Aid2g, and Msg. We can also see the contribution of top 20 factors shown below.



```
> #Inserting a code consist of R function which will extract the important factors
> important_var <- varImp(svm)
> important_var
loess r-squared variable importance

  only 20 most important variables shown (out of 26)

            Overall
Radius      100.0000
Length       64.0864
Mold         64.0086
Curve        38.4774
TDose        37.3690
Post_energy  34.4041
Shower       29.2499
Time1        21.4599
New_shower.  20.7373
X3secT       18.0925
Pressure     17.4179
X4004        14.0782
TDose.2      13.8483
X4010        13.5651
Energy        5.2959
Cure          5.2937
Holding       2.6199
Short.Time    1.7486
LowTemp       0.8630
Flow          0.6417
```

All Factors Contribution (Importance) in determining Lens Quality (By Percentage)



***