Code

# Week-4 Home Work - Introduction to Analytics Modeling

*Prasanta Lenka*

*June 9, 2018*

### *Question 9.1*

After running the principal compnent analysis(PCA)on the given crime data, I plotted the variance using Scree plot (See below). From the plot I determined the first five PCA have enogh features details that can be considered for the model. I ran the linear regression model using the 5 PCA and observed the **R-squared: 0.645, Adjusted R-squared: 0.602**

I transfromed the intercept and coefficients followed by unscaling those to get the originals. Then I estimated the R-squared and Adjusted R-squared using the original coefficients and intercepts . The observed **R-squared: 0.6451941, Adjusted R-squared: 0.601925**, which proved the accuracy of the estimation and model

Finally, using the given data **M = 14.0, So = 0, Ed = 10.0, Po1 = 12.0,Po2 = 15.5,LF = 0.640,M.F = 94.0,Pop = 150,NW = 1.1,U1 = 0.120,U2 = 3.6,Wealth = 3200,Ineq = 20.1,Prob = 0.04,Time = 39.0**, I estimated the prediction for the new city is **1388.926** (see below for details)

**Conclusion:** Last week I used the model **l_model <- lm(Crime ~ M + Ed + Po1 + U2 + Ineq + Prob, data=file_data)** and the model predicton was **1304.245** with **R-squared: 0.7659 and Adjusted R-squared: 0.7307**. Using the PCA, the prediction is **1388.926** with **R-squared: 0.6452 and Adjusted R-squared: 0.6019**. Camparing both R-squared, it appears that the below PCA model does not perform better than the model used in last week homework.Becasue the smaller the R-squared value, the lesser variance is explained

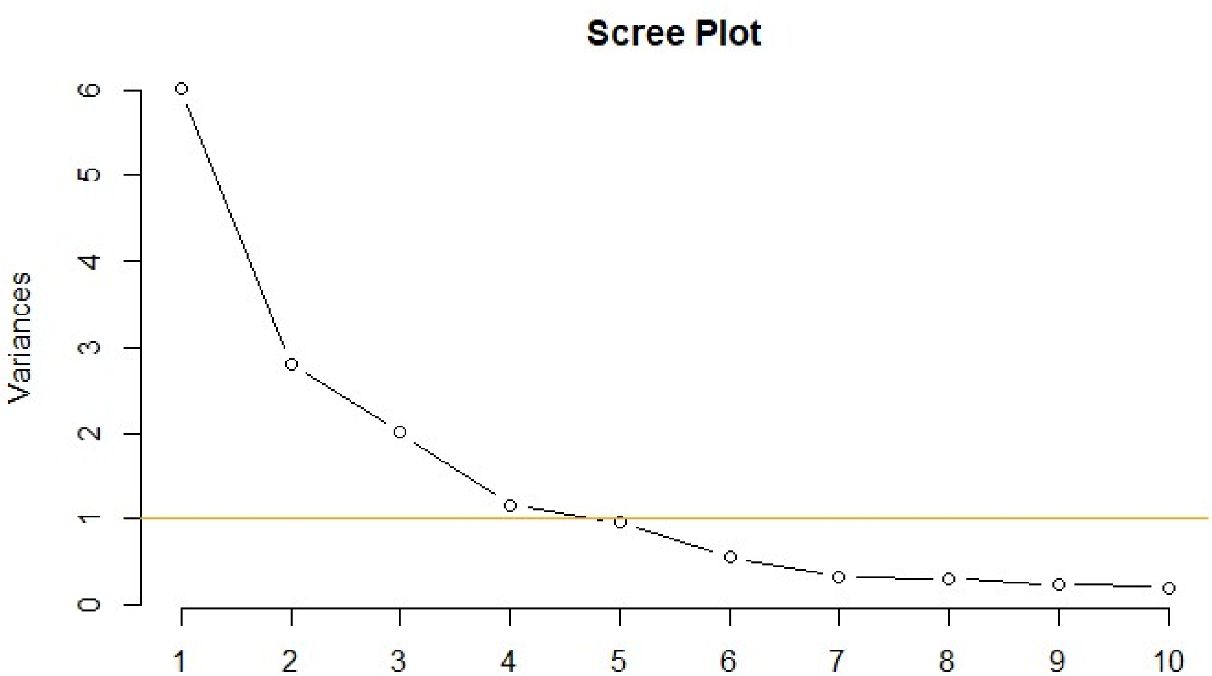
Hide

```
#R Script for Question 9.1
#Clear all data from memory
rm(list = ls())
 #Read the crime data from given file
file_data <- read.table(file ="uscrimeSummer2018.txt",header = TRUE)
#head(file_data)
set.seed(20)
#Do PCA on the crme data set
pca <- prcomp(file_data[,-16], center=T, scale=T)
#Summary of the PCA
summary(pca)
```

```
Importance of components:
                          PC1    PC2    PC3     PC4     PC5     PC6     PC7     PC
8     PC9
Standard deviation     2.4534 1.6739 1.4160 1.07806 0.97893 0.74377 0.56729 0.55444 0.
48493
Proportion of Variance 0.4013 0.1868 0.1337 0.07748 0.06389 0.03688 0.02145 0.02049 0.
01568
Cumulative Proportion  0.4013 0.5880 0.7217 0.79920 0.86308 0.89996 0.92142 0.94191 0.
95759
                          PC10    PC11    PC12    PC13   PC14    PC15
Standard deviation     0.44708 0.41915 0.35804 0.26333 0.2418 0.06793
Proportion of Variance 0.01333 0.01171 0.00855 0.00462 0.0039 0.00031
Cumulative Proportion  0.97091 0.98263 0.99117 0.99579 0.9997 1.00000
```

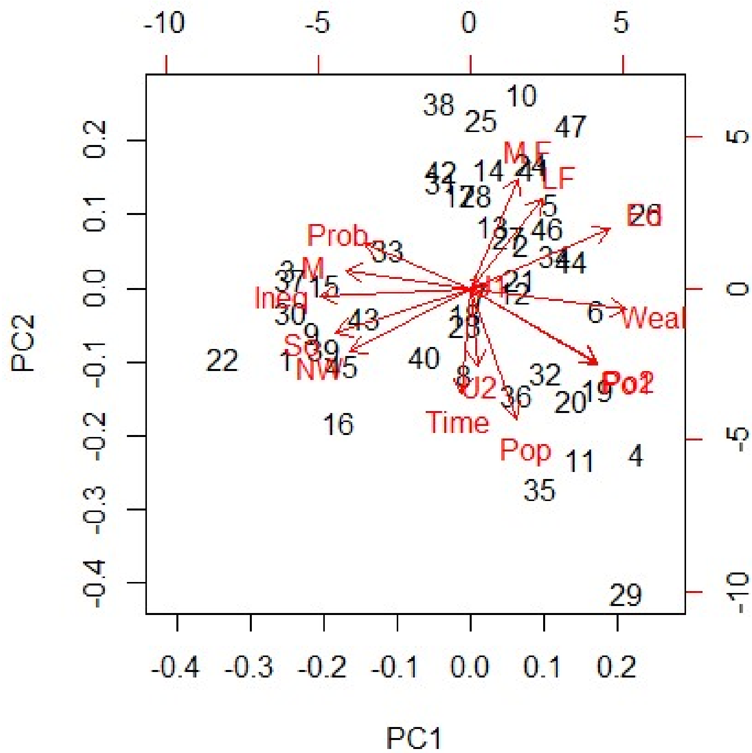
Hide

```
#Plot the standard deviation.
screeplot(pca,main = "Scree Plot", type = "line")
abline(h=1, col="orange")
```




Hide

```
biplot(pca)
```




Hide

```
#Take the first five PCA component and create a new crime data set
pca_crime <- data.frame(cbind(pca$x[,1:5],file_data[,16]))
#run the linear regression using PCA data
pc_model <- lm(V6~., data = pca_crime)
summary(pc_model)
```

```
Call:
lm(formula = V6 ~ ., data = pca_crime)

Residuals:
    Min      1Q  Median      3Q     Max 
-420.79 -185.01   12.21  146.24  447.86 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   905.09      35.59  25.428  < 2e-16 ***
PC1            65.22      14.67   4.447 6.51e-05 ***
PC2           -70.08      21.49  -3.261  0.00224 ** 
PC3            25.19      25.41   0.992  0.32725    
PC4            69.45      33.37   2.081  0.04374 *  
PC5          -229.04      36.75  -6.232 2.02e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 244 on 41 degrees of freedom
Multiple R-squared:  0.6452,	Adjusted R-squared:  0.6019 
F-statistic: 14.91 on 5 and 41 DF,  p-value: 2.446e-08
```

Hide

```
#Now do the transformation of the coeficient and the intercept
intercept <- pc_model$coefficients[1]
coef <- pc_model$coefficients[2:length(pc_model$coefficients)] %*%t(pca$rotation[,1:(l
ength(pc_model$coefficients)-1)])
#Unscale the intercept
intercept <- intercept - sum(coef*sapply(file_data[,1:15], mean)/sapply(file_data[,1:1
5],sd))
#print the original intercept
intercept
```

```
(Intercept) 
  -5933.837 
```

Hide

```
#Unscale the coefficient
coef <- coef/sapply(file_data[,1:15],sd)
#print the coeficient
coef
```

```
            M       So       Ed       Po1       Po2       LF       M.F       Pop       NW
U1
[1,] 48.37374 79.01922 17.8312 39.48484 39.85892 1886.946 36.69366 1.546583 9.537384 1
59.0115
           U2     Wealth      Ineq      Prob     Time
[1,] 38.29933 0.03724014 5.540321 -1523.521 3.838779
```

Hide

```
#Now estimate the model (Y=b0+b1x1 + b2x2+...)
estimates <- as.matrix(file_data[,1:15]) %*% t(coef) + intercept
t(estimates)
```

```
         [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]   [,9]
[,10]
[1,] 713.6803 1195.707 506.4008 1744.815 1004.322 901.3083 817.7618 1158.016 862.66 90
6.1942
        [,11]    [,12]    [,13]    [,14]    [,15]   [,16]    [,17]    [,18]    [,1
9]    [,20]
[1,] 1309.847 831.7397 668.7175 653.8079 663.3242 933.786 467.7924 1097.833 975.2212 1
238.845
        [,21]    [,22]    [,23]    [,24]    [,25]    [,26]   [,27]    [,28]    [,2
9]    [,30]
[1,] 805.7895 769.6724 768.1369 928.9523 604.2355 1845.757 480.427 1015.084 1463.794 8
01.6455
        [,31]    [,32]    [,33]    [,34]    [,35]    [,36]    [,37]    [,38]    [,3
9]    [,40]
[1,] 687.8542 969.6941 722.6822 841.7013 914.9564 977.8353 1211.689 604.2928 627.6148
1069.894
        [,41]    [,42]    [,43]    [,44]    [,45]    [,46]    [,47]
[1,] 841.4929 272.2545 1043.452 1126.343 425.4541 927.1627 1139.354
```

Hide

```
#using the estimates above, lets calculate the Sum of squared Error(SSE), total sum of
square (SST), R-square and adjusted T-Square to check the accuracy of the model
SSE = sum((estimates - file_data[,16])^2)
SStot = sum((file_data[,16] - mean(file_data[,16]))^2)
r_sq <- 1 - SSE/SStot
r_sq
```

```
[1] 0.6451941
```

Hide

```
R2_adjust <- r_sq - (1-r_sq)*5/(nrow(file_data)-5-1)
R2_adjust
```

```
[1] 0.601925
```

Hide

```
#Prepare data frame for prediction
newdata <- data.frame(M = 14.0, So = 0, Ed = 10.0, Po1 = 12.0,Po2 = 15.5,LF = 0.640,M.
F = 94.0,Pop = 150,NW = 1.1,U1 = 0.120,U2 = 3.6,Wealth = 3200,Ineq = 20.1,Prob = 0.04,
Time = 39.0)
estimates <- as.matrix(newdata[,1:15]) %*% t(coef) + intercept
t(estimates)
```

```
         [,1]
[1,] 1388.926
```

# *Question 10.1(a)*

**First,** after fiting the regression tree on the given USCrimes data, the following key information were observed and interpreted.

**No. of Terminal nodes** = 7
**Predictors used in the tree:** “Po1” “Pop” “LF” “NW”
**Terminal node interpretation (Terminal 7):** Po1 > 9.65; observations in that branch = 8 & overall prediction = 1503
**Important indicator:** Po1

**R-Squared:** 0.7244962

**Second,** I performed cross-validation and plotted the deviance to determine optimal pruning oppertunity. From the plot (See below) , the tree size of 5 appreas to be optimal pruning strtegy as it has the lowest error. The R-squared value for the pruned tree is **0.6691333**

**Conclusion:** Comparing the R-Squared for both, it appears that the unpruned tree has a better R-Squared value than the pruned tree. I think the unprunned tree is overfitting . On the otherhand the pruned tree has increased the interpretability of the tree model but introduced a bit of Bias

Hide

```
#R Script for Question 10.1(a)
#Clear all data from memory
rm(list = ls())
#load the library
library(tree)
set.seed(20)
#Read the crime data from given file
file_data <- read.table(file ="uscrimeSummer2018.txt",header = TRUE)
#head(file_data)
#Run the regression tree model
t_model <- tree(Crime~., data=file_data)
summary(t_model)
```

```
Regression tree:
tree(formula = Crime ~ ., data = file_data)
Variables actually used in tree construction:
[1] "Po1" "Pop" "LF"  "NW"
Number of terminal nodes:  7
Residual mean deviance:  47390 = 1896000 / 40
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-573.900  -98.300   -1.545    0.000  110.600  490.100
```

Hide

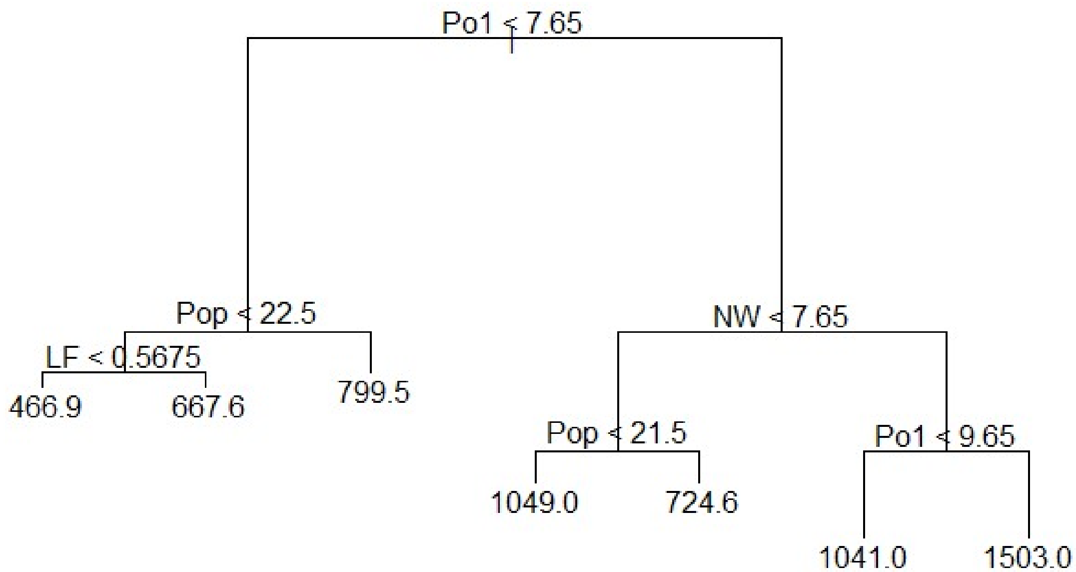
```
t_model
```

```
node), split, n, deviance, yval
      * denotes terminal node

 1) root 47 6881000  905.1
   2) Po1 < 7.65 23  779200  669.6
     4) Pop < 22.5 12  243800  550.5
       8) LF < 0.5675 7   48520  466.9 *
       9) LF > 0.5675 5   77760  667.6 *
     5) Pop > 22.5 11  179500  799.5 *
   3) Po1 > 7.65 24 3604000 1131.0
     6) NW < 7.65 10  557600  886.9
      12) Pop < 21.5 5  146400 1049.0 *
      13) Pop > 21.5 5  147800  724.6 *
     7) NW > 7.65 14 2027000 1305.0
      14) Po1 < 9.65 6  170800 1041.0 *
      15) Po1 > 9.65 8 1125000 1503.0 *
```

Hide

```
#plot the regression tree
plot(t_model)
text(t_model,pretty=0)
```



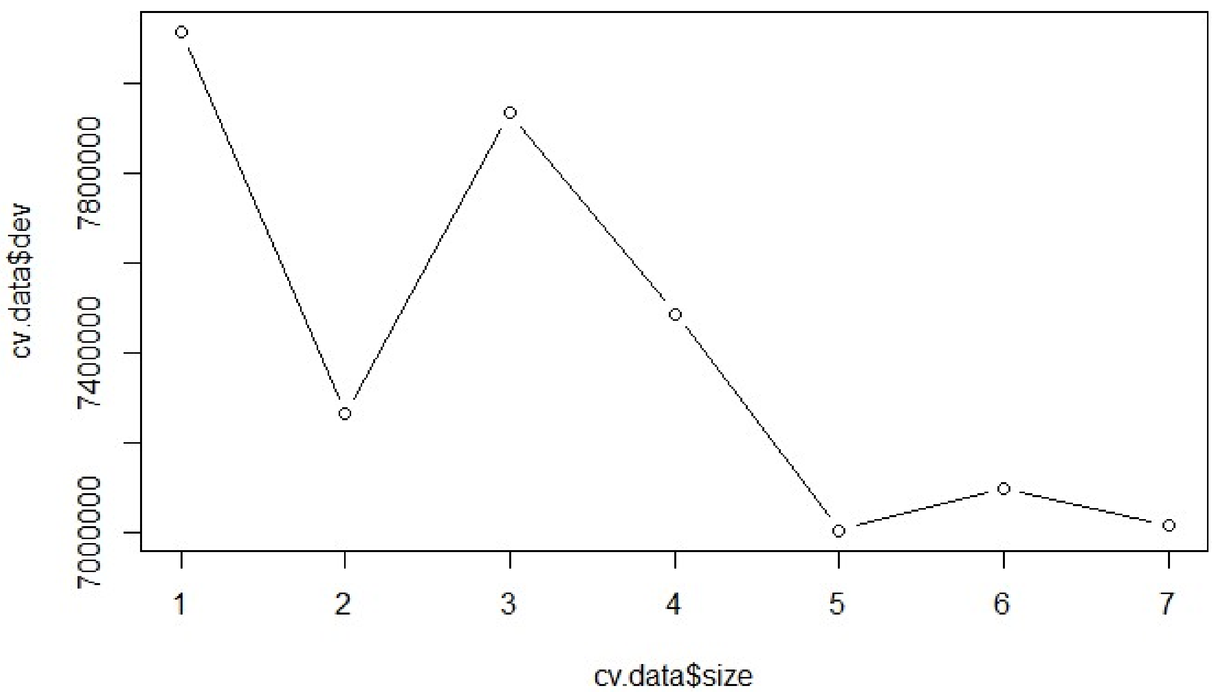

Hide

```
#compute yhat
yhat <- predict(t_model)
#compute SSE
SSE <- sum((yhat-file_data$Crime)^2)
#compute SST
SST <- sum((file_data$Crime-mean(file_data$Crime))^2)
#Compute R-Squared
r_sq <- 1-SSE/SST
r_sq
```

```
[1] 0.7244962
```

Hide

```
#Lets do the cross validation to find the pruning oppertunity
cv.data <- cv.tree(t_model)
#Plot the deviance tree. This will help detrmine pruning size. The lowest cross-valida
tion error point becomes our pruning size
plot(cv.data$size,cv.data$dev,type="b")
```




Hide

```
#print the cross validation data
cv.data
```

```
$`size`
[1] 7 6 5 4 3 2 1

$dev
[1] 7016767 7096980 7004375 7487248 7933770 7263176 8116326

$k
[1]      -Inf  117534.9  263412.9  355961.8  731412.1 1019362.7 2497521.7

$method
[1] "deviance"

attr(,"class")
[1] "prune"         "tree.sequence"
```

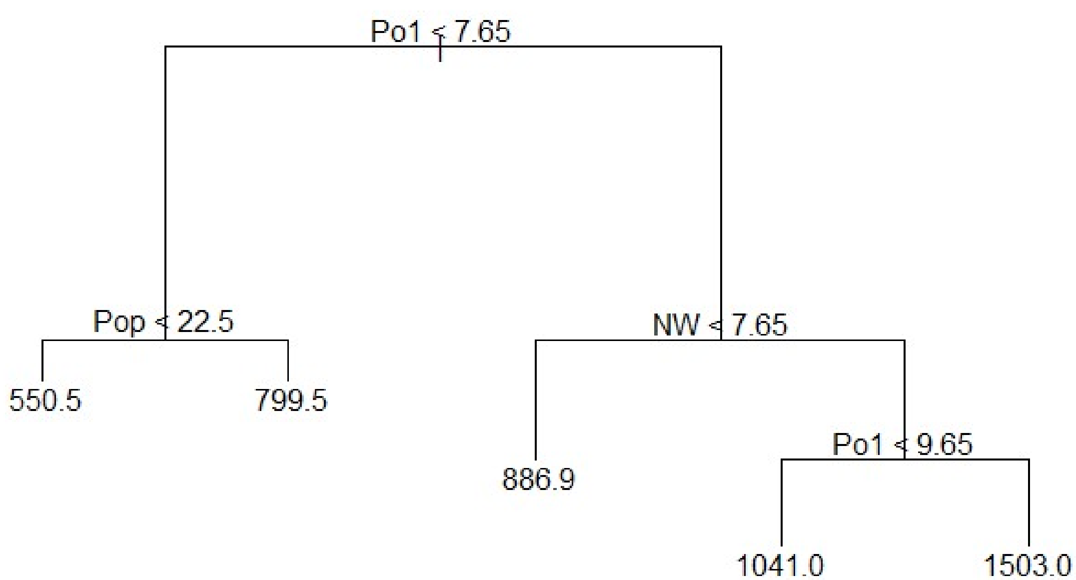
Hide

```
#Prune the tree with size =5 leaf node. We got this from the plot with lowest cross-va
lidation error
p_data <- prune.tree(t_model, best=5)
#compute yhat
yhat <- predict(p_data)
#compute SSE
SSE <- sum((yhat-file_data$Crime)^2)
#compute SST
SST <- sum((file_data$Crime-mean(file_data$Crime))^2)
r_sq <- 1-SSE/SST
r_sq
```

```
[1] 0.6691333
```

Hide

```
#plot the pruned tree
plot(p_data)
text(p_data, pretty=0)
```




## *Question 10.1(b)*

With No. of variables tried at each split=5, the R-Squared value for random forest model is **0.4181955**. Where as in the excercise 10(a), the pruned tree model has R-Squared value of *0.6691333* with tree size 5. In my opinion the pruned tree model does explain the variance better

Hide

```
#R Script for Question 10.1(b)
#Clear all data from memory
rm(list = ls())
#load the library
library(randomForest)
set.seed(20)
#Read the crime data from given file
file_data <- read.table(file ="uscrimeSummer2018.txt",header = TRUE)
#head(file_data)
#Run the Random Forest model
rf_model <- randomForest(Crime~., data=file_data, mtry=5,importance=TRUE)
rf_model
```

```

Call:
 randomForest(formula = Crime ~ ., data = file_data, mtry = 5,      importance = TRU
E)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 5

          Mean of squared residuals: 85177.76
                    % Var explained: 41.82
```

Hide

```
#compute R-Squared
yhat <- predict(rf_model)
SSE <- sum((yhat-file_data$Crime)^2)
SST <- sum((file_data$Crime-mean(file_data$Crime))^2)
r_sq <- 1-SSE/SST
r_sq
```

```
[1] 0.4181955
```

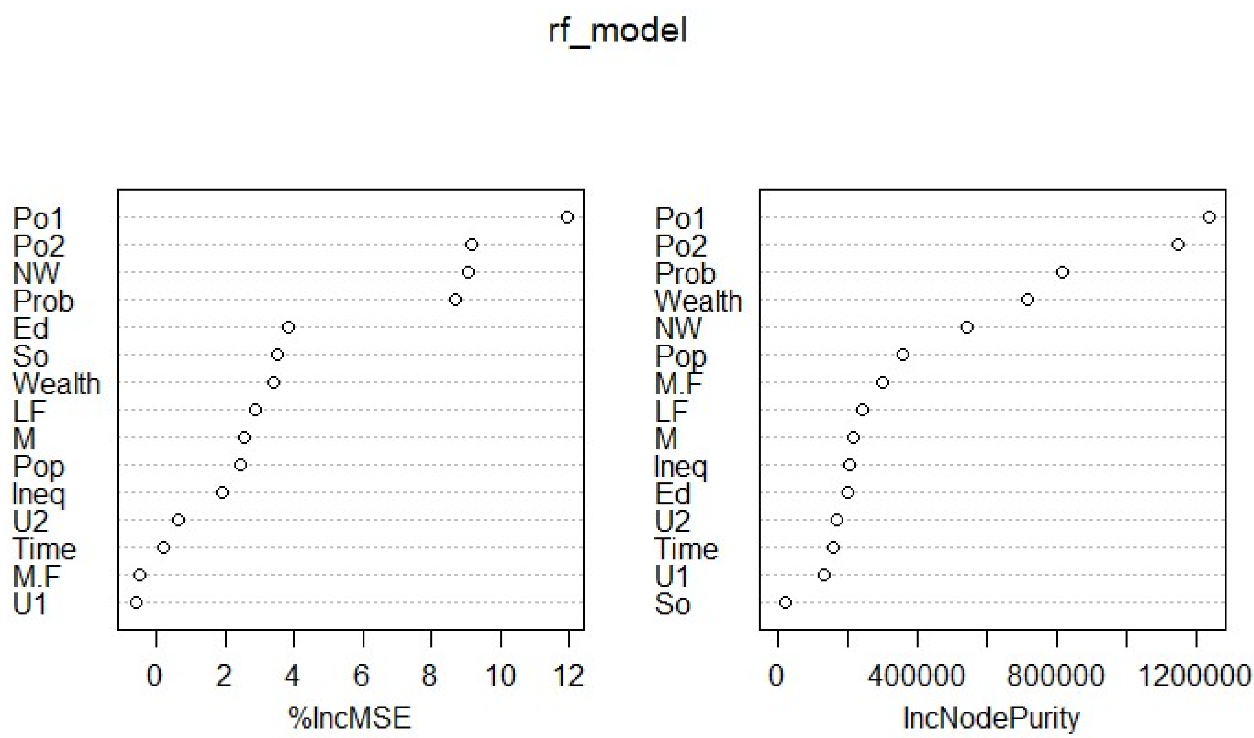
Hide

```
#Variable importance
importance(rf_model)
```

```
          %IncMSE IncNodePurity
M        2.5517094     216497.73
So       3.4817809      23935.78
Ed       3.8324213     201110.17
Po1     11.9147730    1237467.87
Po2      9.1536123    1148306.72
LF       2.8476825     244548.84
M.F     -0.4959635     302519.30
Pop      2.4272402     358648.81
NW       9.0306923     544576.54
U1      -0.6147681     131530.73
U2       0.6511779     168731.85
Wealth   3.3712368     717420.13
Ineq     1.9011985     208016.78
Prob     8.6663941     816841.95
Time     0.2009179     160964.22
```

Hide

```
#plot variable importance
varImpPlot(rf_model)
```



# *Question 10.2*

My Company is a banking and financial services company especially in Credit card and services. One scenario where the logistic regression model is appropriate is to determine the probability of default (PD) for commercial cusomers. This can help to determine during application processing if the commercial customer is going to default in future.The predictors can be

- Annual Revenue customer
- Profit
- Years in business
- industry Type
- Debt/Income Ratio
- Moody's rating

# *Question 10.3 (Part 1 & Part 2)*

## Part 1:

I didvide the data set into training and test sets (70:30 ratio). Using the stepwise method I selected optinal factors and train the model __glm(formula = V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10 + V14 +V15 + V20 ,family = binomial(link = "logit"), data = train_data). Then I used the model on test data the key findings are

**Mis-classification error Rate:** 0.1867
**Sensitivity:** 0.4588235
**Specificity:** 0.9534884
**Confusion Matrix:**

| CM | 0 | 1 |
|---|---|---|
| 0 | 205 | 46 |
| 1 | 10 | 39 |

For coefficients and other output, please see below program outputs.

## Part 2:

The optimal threshold probability is **0.5648562**. I determined this by using the function **optimalCutoff (test_data$V21, pred)[1]**. So the cost of mis-classification is **205 x 0 + 10 x 5 + 39 x 0 + 46 x 1 = 96 (incorrectly identifying a bad customer as good, is 5 times worse than incorrectly classifying a good customer as bad)**

Hide

```
#R Script for Question 10.3
#Clear all data from memory
rm(list = ls())
library(InformationValue)
set.seed(30)
#Read the credit data from given file
cr_data <- read.table(file ="germancreditSummer2018.txt",sep = " ",header=FALSE)
#convert V21 to binary
cr_data$V21[cr_data$V21==1] <- 0
cr_data$V21[cr_data$V21==2] <- 1
#Prepare training ans test data
total_rows <- nrow(cr_data)
random_sample = sample(1:total_rows, size = round(total_rows * .7))
train_data <- cr_data[random_sample,]
test_data <- cr_data[-random_sample,]
#Build the model on train data with all factors
full_model <- glm(V21~., family=binomial(link="logit"),train_data)
summary(full_model)
```

```

Call:
glm(formula = V21 ~ ., family = binomial(link = "logit"), data = train_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2999  -0.7200  -0.3634   0.7381   2.6867

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  8.864e-02  1.314e+00   0.067 0.946218
V1A12       -1.607e-01  2.571e-01  -0.625 0.532022
V1A13       -8.619e-01  4.608e-01  -1.870 0.061419 .
V1A14       -1.671e+00  2.785e-01  -5.999 1.99e-09 ***
V2           2.968e-02  1.100e-02   2.699 0.006960 **
V3A31       -2.129e-01  6.469e-01  -0.329 0.742087
V3A32       -6.299e-01  4.863e-01  -1.295 0.195185
V3A33       -8.085e-01  5.328e-01  -1.517 0.129161
V3A34       -1.442e+00  4.958e-01  -2.907 0.003646 **
V4A41       -1.626e+00  4.637e-01  -3.507 0.000454 ***
V4A410      -2.261e+00  1.050e+00  -2.153 0.031341 *
V4A42       -8.447e-01  3.130e-01  -2.698 0.006972 **
V4A43       -7.666e-01  2.948e-01  -2.600 0.009315 **
V4A44       -3.928e-01  7.731e-01  -0.508 0.611433
V4A45        1.558e-01  6.058e-01   0.257 0.797018
V4A46       -2.569e-01  4.714e-01  -0.545 0.585700
V4A48       -8.933e-01  1.281e+00  -0.697 0.485771
V4A49       -8.650e-01  4.202e-01  -2.059 0.039526 *
V5           1.253e-04  5.286e-05   2.369 0.017818 *
V6A62       -5.635e-01  3.532e-01  -1.595 0.110622
V6A63       -5.494e-02  4.540e-01  -0.121 0.903679
V6A64       -1.109e+00  5.801e-01  -1.911 0.055947 .
V6A65       -9.841e-01  3.206e-01  -3.069 0.002146 **
V7A72       -3.838e-01  5.218e-01  -0.735 0.462035
V7A73       -3.911e-01  5.021e-01  -0.779 0.436039
V7A74       -1.099e+00  5.404e-01  -2.034 0.041944 *
V7A75       -3.191e-01  5.120e-01  -0.623 0.533100
V8           3.801e-01  1.050e-01   3.619 0.000295 ***
V9A92       -1.503e-02  4.923e-01  -0.031 0.975639
V9A93       -7.733e-01  4.872e-01  -1.587 0.112447
V9A94       -2.224e-01  5.693e-01  -0.391 0.696079
V10A102      5.591e-01  5.212e-01   1.073 0.283393
V10A103     -6.886e-01  4.743e-01  -1.452 0.146590
V11          9.581e-03  1.037e-01   0.092 0.926389
V12A122      5.517e-01  3.044e-01   1.812 0.069943 .
V12A123      3.486e-01  2.939e-01   1.186 0.235676
V12A124      8.070e-01  5.122e-01   1.576 0.115088
V13         -2.281e-02  1.137e-02  -2.006 0.044831 *
```

```
V14A142     -1.710e-01  5.024e-01  -0.340 0.733550
V14A143     -8.176e-01  2.808e-01  -2.912 0.003594 **
V15A152     -3.268e-01  2.819e-01  -1.159 0.246481
V15A153     -4.350e-01  5.527e-01  -0.787 0.431244
V16          3.515e-01  2.336e-01   1.505 0.132360
V17A172      7.468e-01  8.460e-01   0.883 0.377342
V17A173      5.871e-01  8.125e-01   0.723 0.469942
V17A174      2.712e-01  8.266e-01   0.328 0.742826
V18          3.674e-01  3.088e-01   1.190 0.234137
V19A192     -1.766e-01  2.431e-01  -0.727 0.467497
V20A202     -1.382e+00  7.589e-01  -1.821 0.068552 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 863.51  on 699  degrees of freedom
Residual deviance: 638.13  on 651  degrees of freedom
AIC: 736.13

Number of Fisher Scoring iterations: 5
```

Hide

```
#use stepwise method for variable selection
step_vs <- step(full_model)
```

```
Start:  AIC=736.13
V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10 + V11 + 
    V12 + V13 + V14 + V15 + V16 + V17 + V18 + V19 + V20

       Df Deviance    AIC
- V17   3   639.79 731.79
- V15   2   639.62 733.62
- V11   1   638.14 734.14
- V12   3   642.33 734.33
- V19   1   638.66 734.66
- V18   1   639.54 735.54
- V10   2   641.66 735.66
- V7    4   645.84 735.84
<none>      638.13 736.13
- V16   1   640.43 736.43
- V20   1   642.05 738.05
- V13   1   642.26 738.26
- V5    1   643.78 739.78
- V9    3   648.43 740.43
- V4    9   660.44 740.44
- V3    4   650.88 740.88
- V14   2   647.47 741.47
- V2    1   645.51 741.51
- V6    4   651.98 741.98
- V8    1   651.77 747.77
- V1    3   684.60 776.60

Step:  AIC=731.79
V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10 + V11 + 
    V12 + V13 + V14 + V15 + V16 + V18 + V19 + V20

       Df Deviance    AIC
- V15   2   641.10 729.10
- V12   3   643.43 729.43
- V11   1   639.87 729.87
- V7    4   647.07 731.07
- V19   1   641.23 731.23
- V10   2   643.23 731.23
- V18   1   641.33 731.33
<none>      639.79 731.79
- V16   1   641.82 731.82
- V20   1   643.75 733.75
- V13   1   644.01 734.01
- V5    1   645.06 735.06
- V9    3   649.97 735.97
- V3    4   652.09 736.09
- V14   2   649.42 737.42
- V4    9   663.75 737.75
```

```
- V2    1   647.83 737.83
- V6    4   653.85 737.85
- V8    1   653.03 743.03
- V1    3   686.42 772.42

Step:  AIC=729.1
V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10 + V11 + 
    V12 + V13 + V14 + V16 + V18 + V19 + V20

       Df Deviance    AIC
- V12   3   644.73 726.73
- V11   1   641.44 727.44
- V7    4   648.11 728.11
- V19   1   642.40 728.40
- V18   1   642.57 728.57
- V10   2   644.67 728.67
<none>      641.10 729.10
- V16   1   643.26 729.26
- V20   1   644.82 730.82
- V5    1   646.50 732.50
- V13   1   646.60 732.60
- V14   2   650.28 734.28
- V3    4   654.38 734.38
- V9    3   652.57 734.57
- V2    1   648.70 734.70
- V6    4   654.72 734.72
- V4    9   664.82 734.82
- V8    1   654.08 740.08
- V1    3   688.65 770.65

Step:  AIC=726.73
V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10 + V11 + 
    V13 + V14 + V16 + V18 + V19 + V20

       Df Deviance    AIC
- V11   1   645.15 725.15
- V19   1   645.61 725.61
- V18   1   645.97 725.97
- V7    4   652.04 726.04
- V16   1   646.59 726.59
<none>      644.73 726.73
- V10   2   648.87 726.87
- V20   1   648.32 728.32
- V13   1   649.88 729.88
- V5    1   650.71 730.71
- V6    4   657.54 731.54
- V9    3   655.67 731.67
- V3    4   658.16 732.16
- V14   2   654.81 732.81
```

```
- V4    9   669.22 733.22
- V2    1   654.02 734.02
- V8    1   658.69 738.69
- V1    3   694.98 770.98

Step:  AIC=725.15
V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10 + V13 + 
    V14 + V16 + V18 + V19 + V20

       Df Deviance    AIC
- V19   1   645.99 723.99
- V18   1   646.45 724.45
- V7    4   652.58 724.58
- V16   1   647.07 725.07
<none>      645.15 725.15
- V10   2   649.37 725.37
- V20   1   648.78 726.78
- V13   1   649.97 727.97
- V5    1   651.00 729.00
- V6    4   657.93 729.93
- V9    3   656.42 730.42
- V3    4   658.46 730.46
- V14   2   655.07 731.07
- V4    9   669.58 731.58
- V2    1   654.67 732.67
- V8    1   659.09 737.09
- V1    3   696.35 770.35

Step:  AIC=723.99
V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10 + V13 + 
    V14 + V16 + V18 + V20

       Df Deviance    AIC
- V18   1   647.32 723.32
- V7    4   653.70 723.70
- V16   1   647.78 723.78
<none>      645.99 723.99
- V10   2   650.03 724.03
- V20   1   649.41 725.41
- V5    1   651.16 727.16
- V13   1   651.50 727.50
- V6    4   658.80 728.80
- V3    4   659.04 729.04
- V9    3   657.18 729.18
- V14   2   655.99 729.99
- V4    9   671.01 731.01
- V2    1   655.71 731.71
- V8    1   659.60 735.60
- V1    3   697.72 769.72
```

```
Step:  AIC=723.32
V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10 + V13 +
    V14 + V16 + V20

       Df Deviance    AIC
- V7    4   654.84 722.84
- V10   2   650.96 722.96
- V16   1   649.32 723.32
<none>      647.32 723.32
- V20   1   650.77 724.77
- V5    1   652.17 726.17
- V13   1   652.55 726.55
- V9    3   657.29 727.29
- V6    4   659.94 727.94
- V3    4   660.83 728.83
- V14   2   657.81 729.81
- V4    9   672.59 730.59
- V2    1   656.91 730.91
- V8    1   660.10 734.10
- V1    3   698.72 768.72

Step:  AIC=722.84
V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V8 + V9 + V10 + V13 + V14 +
    V16 + V20

       Df Deviance    AIC
- V16   1   656.73 722.73
<none>      654.84 722.84
- V10   2   659.22 723.22
- V20   1   658.04 724.04
- V5    1   659.01 725.01
- V13   1   659.03 725.03
- V6    4   667.66 727.66
- V9    3   666.49 728.49
- V4    9   678.76 728.76
- V14   2   665.13 729.13
- V3    4   669.41 729.41
- V2    1   663.59 729.59
- V8    1   667.58 733.58
- V1    3   709.81 771.81

Step:  AIC=722.73
V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V8 + V9 + V10 + V13 + V14 +
    V20

       Df Deviance    AIC
<none>      656.73 722.73
- V10   2   661.15 723.15
```

```
- V20   1   660.26 724.26
- V13   1   660.66 724.66
- V5    1   660.76 724.76
- V6    4   669.07 727.07
- V3    4   669.41 727.41
- V9    3   668.21 728.21
- V2    1   665.10 729.10
- V4    9   681.30 729.30
- V14   2   667.66 729.66
- V8    1   668.97 732.97
- V1    3   711.41 771.41
```

Hide

```
#step_vs
#Run the model on optimal set of factors
model <- glm(formula = V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10 + V14 +V
15 + V20 ,family = binomial(link = "logit"), data = train_data)
summary(model)
```

```
Call:
glm(formula = V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 +
    V10 + V14 + V15 + V20, family = binomial(link = "logit"),
    data = train_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1953  -0.7244  -0.3900   0.7970   2.6060

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  9.789e-01  8.773e-01   1.116 0.264463
V1A12       -2.602e-01  2.496e-01  -1.043 0.297171
V1A13       -1.007e+00  4.523e-01  -2.226 0.026002 *
V1A14       -1.741e+00  2.726e-01  -6.386  1.7e-10 ***
V2           3.232e-02  1.058e-02   3.054 0.002258 **
V3A31       -4.539e-01  6.270e-01  -0.724 0.469096
V3A32       -8.085e-01  4.631e-01  -1.746 0.080820 .
V3A33       -8.079e-01  5.283e-01  -1.529 0.126172
V3A34       -1.365e+00  4.889e-01  -2.792 0.005242 **
V4A41       -1.613e+00  4.512e-01  -3.574 0.000351 ***
V4A410      -2.372e+00  1.026e+00  -2.311 0.020840 *
V4A42       -7.501e-01  3.003e-01  -2.498 0.012480 *
V4A43       -7.802e-01  2.864e-01  -2.724 0.006445 **
V4A44       -5.302e-01  7.640e-01  -0.694 0.487707
V4A45        3.098e-01  5.963e-01   0.520 0.603381
V4A46       -1.611e-01  4.592e-01  -0.351 0.725717
V4A48       -4.254e-01  1.262e+00  -0.337 0.736079
V4A49       -8.126e-01  4.094e-01  -1.985 0.047176 *
V5           1.013e-04  4.943e-05   2.048 0.040524 *
V6A62       -3.331e-01  3.347e-01  -0.995 0.319698
V6A63       -9.777e-02  4.477e-01  -0.218 0.827119
V6A64       -9.814e-01  5.571e-01  -1.761 0.078165 .
V6A65       -9.117e-01  3.094e-01  -2.947 0.003208 **
V7A72        6.469e-02  4.569e-01   0.142 0.887413
V7A73       -2.359e-02  4.306e-01  -0.055 0.956312
V7A74       -7.078e-01  4.794e-01  -1.476 0.139891
V7A75       -8.215e-02  4.460e-01  -0.184 0.853858
V8           3.430e-01  1.003e-01   3.421 0.000625 ***
V9A92        5.433e-02  4.745e-01   0.114 0.908855
V9A93       -6.448e-01  4.667e-01  -1.382 0.167086
V9A94       -2.292e-01  5.480e-01  -0.418 0.675782
V10A102      6.221e-01  5.167e-01   1.204 0.228577
V10A103     -6.688e-01  4.594e-01  -1.456 0.145443
V14A142     -2.253e-01  4.997e-01  -0.451 0.652083
V14A143     -8.587e-01  2.748e-01  -3.124 0.001781 **
V15A152     -3.966e-01  2.601e-01  -1.524 0.127417
```

```
V15A153     -3.119e-01  3.923e-01  -0.795 0.426624
V20A202     -1.347e+00  7.386e-01  -1.824 0.068159 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 863.51  on 699  degrees of freedom
Residual deviance: 652.03  on 662  degrees of freedom
AIC: 728.03

Number of Fisher Scoring iterations: 5
```

Hide

```
#predict using the test data
pred <- predict(model, test_data)
pred <- plogis(pred)
#pred
#determine the optimal threshold probability
t_hold <- optimalCutoff(test_data$V21, pred)[1]
t_hold
```

```
[1] 0.5648562
```

Hide

```
#determine the mis classification error
misClassError(test_data$V21, pred, threshold = t_hold)
```

```
[1] 0.1867
```

Hide

```
#Determine the Sensitivity
sensitivity(test_data$V21, pred, threshold = t_hold)
```

```
[1] 0.4588235
```

Hide

```
#Determine the specificity
specificity(test_data$V21, pred, threshold = t_hold)
```

```
[1] 0.9534884
```

Hide

```
#Finally get the confusion matric. We will calculate the cost of mis classification
confusionMatrix(test_data$V21, pred, threshold = t_hold)
```

| | 0 | 1 |
|---|---|---|
| | <int> | <int> |
| 0 | 205 | 46 |
| 1 | 10 | 39 |

2 rows