Week 4 Homework : Intro to Analytics

Source code is provided at the end

Question 9.1

Using the same crime data set `uscrime.txt` as in Question 8.2, apply Principal Component Analysis and then create a regression model using the first few principal components. Specify your new model in terms of the original variables (not the principal components), and compare its quality to that of your solution to Question 8.2. You can use the R function `prcomp` for PCA. (**Note** that to first scale the data, you can include `scale. = TRUE` to scale as part of the PCA function. Don't forget that, to make a prediction for the new city, you'll need to unscale the coefficients (i.e., do the scaling calculation in reverse)!)

Answer:

```
US_Crimes_Data <- read.table("../UScrimeSummer2018.txt", header=T)
View(US_Crimes_Data)

crimes_pca <- prcomp(US_Crimes_Data, scale. = T)
crimes_pca
```

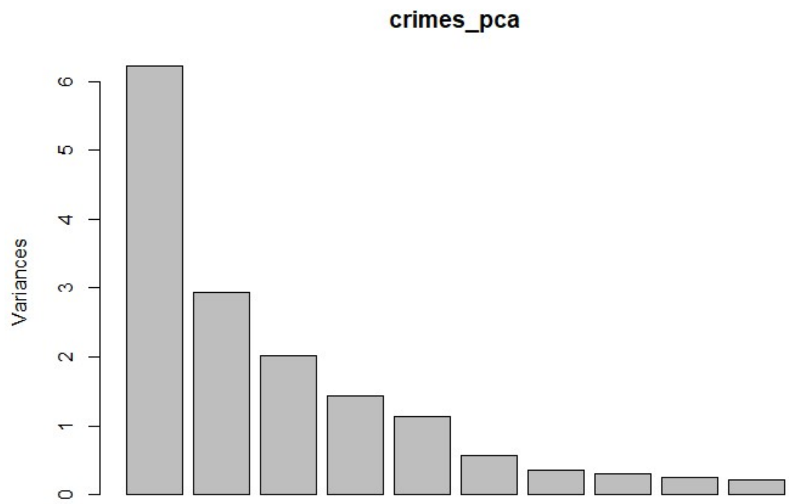
```
> crimes_pca
Standard deviations (1, .., p=16):
 [1] 2.49443367 1.71114001 1.42083523 1.19585483 1.06341246 0.75086767
0.60237227 0.55502694 0.49243978
[10] 0.47036049 0.43856093 0.41777035 0.29147362 0.26063133 0.21812568
0.06584351
```

PC5 standard deviation of 1 on the scaled model is a reasonable choice.

```
plot(crimes_pca)
summary(crimes_pca)
```

```
> summary(crimes_pca)
Importance of components:
                          PC1    PC2    PC3     PC4     PC5     PC6     PC7
PC8     PC9    PC10    PC11
Standard deviation     2.4944 1.7111 1.4208 1.19585 1.06341 0.75087 0.60237
0.55503 0.49244 0.47036 0.43856
Proportion of Variance 0.3889 0.1830 0.1262 0.08938 0.07068 0.03524 0.02268
0.01925 0.01516 0.01383 0.01202
Cumulative Proportion  0.3889 0.5719 0.6981 0.78744 0.85812 0.89336 0.91603
0.93529 0.95044 0.96427 0.97629
                          PC12    PC13    PC14    PC15    PC16
Standard deviation     0.41777 0.29147 0.26063 0.21813 0.06584
Proportion of Variance 0.01091 0.00531 0.00425 0.00297 0.00027
Cumulative Proportion  0.98720 0.99251 0.99676 0.99973 1.00000

pcr(formula = Crime ~ ., data = US_Crimes_Data, scale = T, validation = "CV")
> summary(crimes_pcr)
Data:   X dimension: 47 15
        Y dimension: 47 1
```

```
Fit method: svdpc
Number of components considered: 15
```




```
VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7
comps  8 comps  9 comps  10 comps
CV           390.9    364.4    358.2    363.3    359.4    264.7    263.1
260.0    271.6    277.9     280.8
adjCV        390.9    363.6    356.8    361.7    365.7    262.6    260.6
254.7    269.0    275.0     277.8
       11 comps  12 comps  13 comps  14 comps  15 comps
CV        309.6     267.7     277.8     273.7     269.0
adjCV     307.0     263.4     273.8     268.3     263.7

TRAINING: % variance explained
       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
9 comps  10 comps  11 comps
X        40.13    58.81    72.17    79.92    86.31    90.00    92.14    94.19
95.76     97.09     98.26
Crime    17.11    26.31    27.16    30.91    64.52    65.86    68.82    68.99
69.20     69.63     69.74
       12 comps  13 comps  14 comps  15 comps
X         99.12     99.58     99.97    100.00
Crime     76.93     77.24     79.11     80.31

> test1_df <- 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,
+                        Crime = 0)
>
> output1 <- predict(crimes_pcr, test1_df)
> output1

1  comps: Crime = 984.9123
2  comps: Crime = 1178.877
3  comps: Crime = 1192.321
4  comps: Crime = 1112.678
5  comps: Crime = 1388.926
6  comps: Crime = 1248.427
7  comps: Crime = 1230.418
8  comps: Crime = 1190.455
9  comps: Crime = 1136.169
10 comps: Crime = 1110.684
11 comps: Crime = 1100.079
12 comps: Crime = 1581.932
13 comps: Crime = 1433.792
14 comps: Crime = 957.264
15 comps: Crime = 155.4349
```

15 component solution of 155.4349 is the same as the prediction made by the lm function in earlier homework exercise. That prediction was out of min-max bounds in the training data range as indicated by the table below. The crime prediction with components 5 (standard deviation near 1 for PCA) or average of the (4,5, and 6) is a better prediction. Too many components cause over-fitting by the model.

	M	So	Ed	Po1	Po2	LF	M.F	Pop	NW	U1	U2	Wealth	Ineq	Prob	Time	Crime
Max	17.7	1	12.2	16.6	15.7	0.641	107.1	168	42.3	0.142	5.8	6890	27.6	0.119804	44.0004	1993.0000
Prediction	14.0	0	10.0	12.0	15.5	0.640	94.0	150	1.1	0.120	3.6	3200	20.1	0.040000	39.0000	155.4349
Min	11.9	0	8.7	4.5	4.1	0.480	93.4	3	0.2	0.070	2.0	2880	12.6	0.006900	12.1996	342.0000

Question 10.1

Using the same crime data set `uscrime.txt` as in Questions 8.2 and 9.1, find the best model you can using

(a) a regression tree model, and

(b) a random forest model.

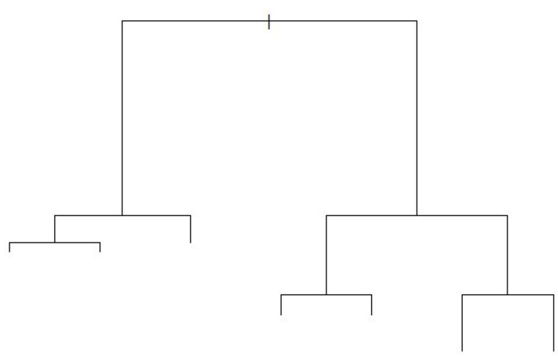
In R, you can use the `tree` package or the `rpart` package, and the `randomForest` package. For each model, describe one or two qualitative takeaways you get from analyzing the results (i.e., don't just stop when you have a good model, but interpret it too).

<u>Answer:</u>

US_Crimes_Data <- read.table("../UScrimeSummer2018.txt", header=T)
View(US_Crimes_Data)

#part a: Tree model
Crime_tree <- tree(Crime~., US_Crimes_Data)
Crime_tree
summary(Crime_tree)

```
> Crime_tree
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 *
> summary(Crime_tree)

Regression tree:
tree(formula = Crime ~ ., data = US_Crimes_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
```



Crime_rpart <- rpart(Crime~., data= US_Crimes_Data, method= "anova")
Crime_rpart
summary(Crime_rpart)

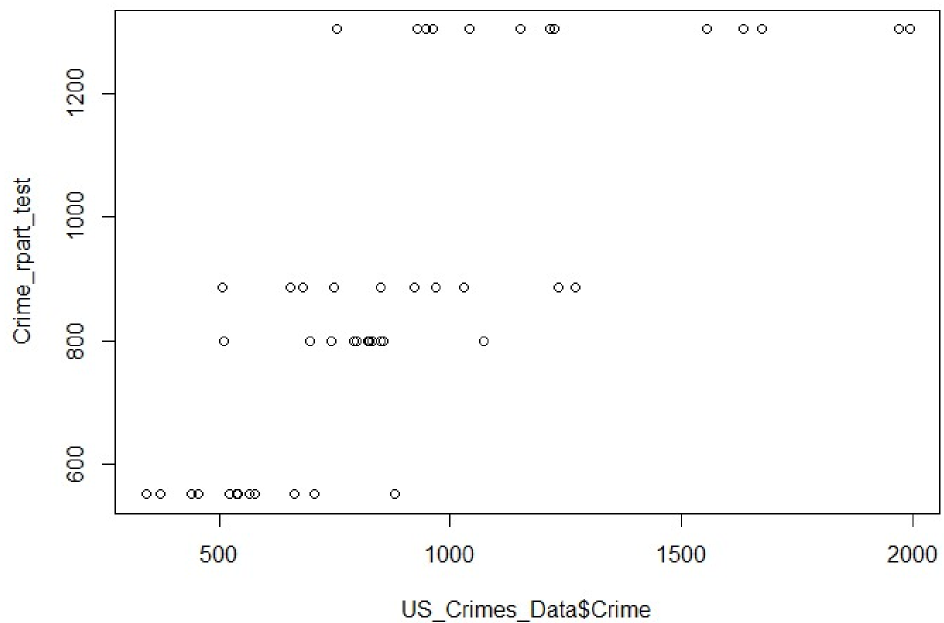
```
> Crime_rpart
n= 47

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

1) root 47 6880928.0  905.0851
  2) Po1< 7.65 23  779243.5  669.6087
    4) Pop< 22.5 12  243811.0  550.5000 *
    5) Pop>=22.5 11  179470.7  799.5455 *
  3) Po1>=7.65 24 3604162.0 1130.7500
    6) NW< 7.65 10  557574.9  886.9000 *
    7) NW>=7.65 14 2027225.0 1304.9290 *
> summary(Crime_rpart)
Call:
rpart(formula = Crime ~ ., data = US_Crimes_Data, method = "anova")
  n= 47

          CP nsplit rel error   xerror      xstd
1 0.36296293      0 1.0000000 1.059739 0.2589546
2 0.14814320      1 0.6370371 1.024087 0.2227001
3 0.05173165      2 0.4888939 1.160607 0.2416903
4 0.01000000      3 0.4371622 1.058472 0.2346708

Variable importance
   Po1    Po2 Wealth   Ineq   Prob      M     NW    Pop   Time     Ed     LF
So
    17     17     11     11     10     10      9      5      4      4      1
1

Node number 1: 47 observations,    complexity param=0.3629629
  mean=905.0851, MSE=146402.7
  left son=2 (23 obs) right son=3 (24 obs)
  Primary splits:
      Po1    < 7.65      to the left,  improve=0.3629629, (0 missing)
      Po2    < 7.2       to the left,  improve=0.3629629, (0 missing)
      Prob   < 0.0418485 to the right, improve=0.3217700, (0 missing)
      NW     < 7.65      to the left,  improve=0.2356621, (0 missing)
      Wealth < 6240      to the left,  improve=0.2002403, (0 missing)
  Surrogate splits:
      Po2    < 7.2       to the left,  agree=1.000, adj=1.000, (0 split)
      Wealth < 5330      to the left,  agree=0.830, adj=0.652, (0 split)
      Prob   < 0.043598  to the right, agree=0.809, adj=0.609, (0 split)
      M      < 13.25     to the right, agree=0.745, adj=0.478, (0 split)
      Ineq   < 17.15     to the right, agree=0.745, adj=0.478, (0 split)

Node number 2: 23 observations,    complexity param=0.05173165
  mean=669.6087, MSE=33880.15
  left son=4 (12 obs) right son=5 (11 obs)
  Primary splits:
      Pop < 22.5      to the left,  improve=0.4568043, (0 missing)
      M   < 14.5      to the left,  improve=0.3931567, (0 missing)
      NW  < 5.4       to the left,  improve=0.3184074, (0 missing)
      Po1 < 5.75      to the left,  improve=0.2310098, (0 missing)
      U1  < 0.093     to the right, improve=0.2119062, (0 missing)
  Surrogate splits:
```

```
      NW   < 5.4       to the left,  agree=0.826, adj=0.636, (0 split)
      M    < 14.5      to the left,  agree=0.783, adj=0.545, (0 split)
      Time < 22.30055  to the left,  agree=0.783, adj=0.545, (0 split)
      So   < 0.5       to the left,  agree=0.739, adj=0.455, (0 split)
      Ed   < 10.85     to the right, agree=0.739, adj=0.455, (0 split)

Node number 3: 24 observations,    complexity param=0.1481432
  mean=1130.75, MSE=150173.4
  left son=6 (10 obs) right son=7 (14 obs)
  Primary splits:
      NW   < 7.65      to the left,  improve=0.2828293, (0 missing)
      M    < 13.05     to the left,  improve=0.2714159, (0 missing)
      Time < 21.9001   to the left,  improve=0.2060170, (0 missing)
      M.F  < 99.2      to the left,  improve=0.1703438, (0 missing)
      Po1  < 10.75     to the left,  improve=0.1659433, (0 missing)
  Surrogate splits:
      Ed   < 11.45     to the right, agree=0.750, adj=0.4, (0 split)
      Ineq < 16.25     to the left,  agree=0.750, adj=0.4, (0 split)
      Time < 21.9001   to the left,  agree=0.750, adj=0.4, (0 split)
      Pop  < 30        to the left,  agree=0.708, adj=0.3, (0 split)
      LF   < 0.5885    to the right, agree=0.667, adj=0.2, (0 split)

Node number 4: 12 observations
  mean=550.5, MSE=20317.58

Node number 5: 11 observations
  mean=799.5455, MSE=16315.52

Node number 6: 10 observations
  mean=886.9, MSE=55757.49

Node number 7: 14 observations
  mean=1304.929, MSE=144801.8
```

```
> output_rpart
    1
886.9
```

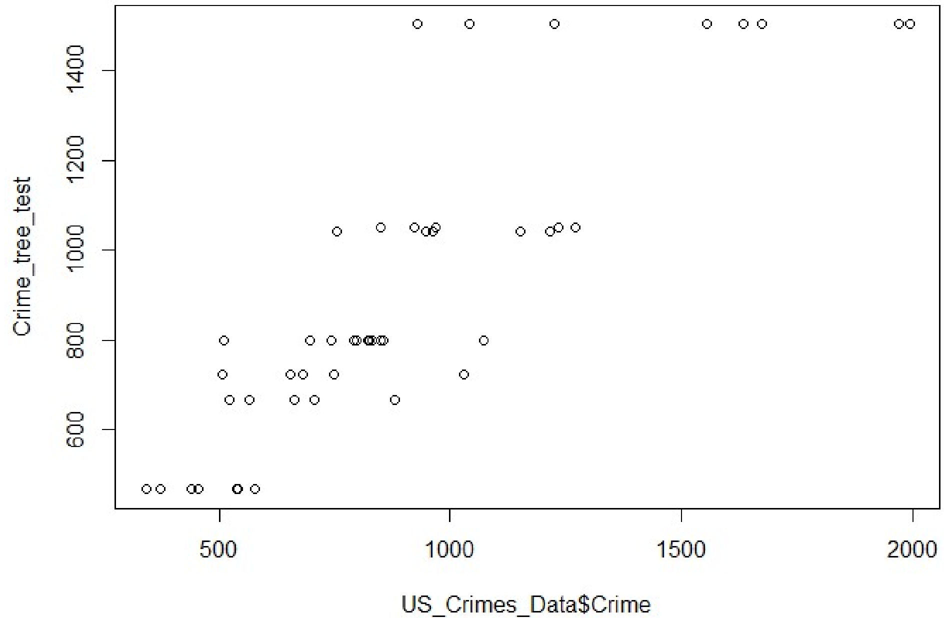
```
Crime_rpart_test <- predict(Crime_rpart,US_Crimes_Data[,1:15])
plot(US_Crimes_Data$Crime, Crime_rpart_test)
```

Model Prediction does not capture the actual Crime data.

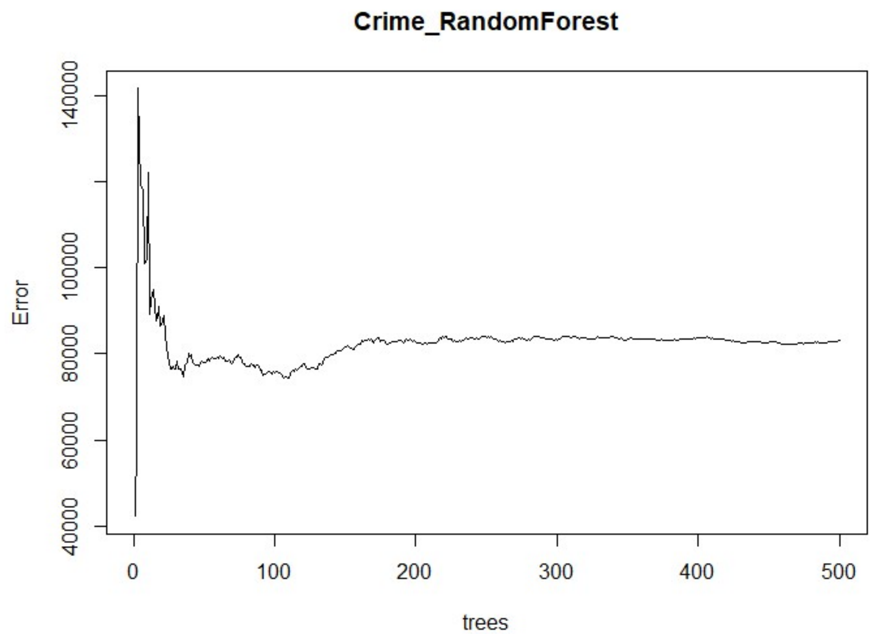



```
> output_tree
     1
724.6
```

Crime_tree_test <- predict(Crime_tree,US_Crimes_Data[,1:15])
plot(US_Crimes_Data$Crime, Crime_tree_test)



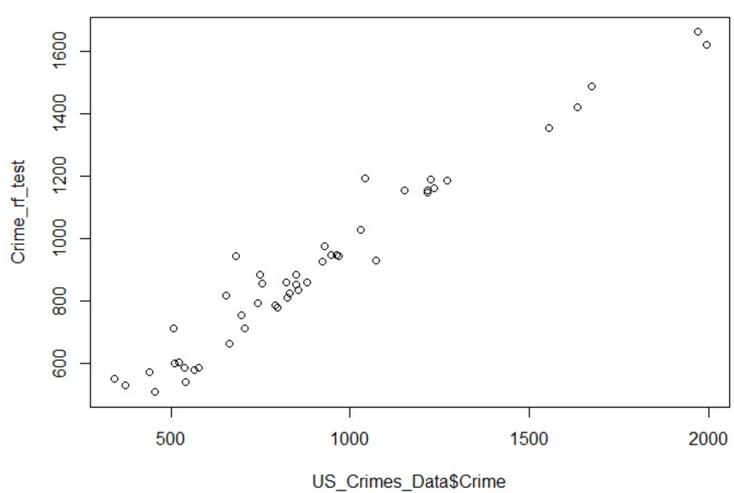

```
> #part b: Random forest
> Crime_RandomForest <- randomForest(Crime~., US_Crimes_Data)
> Crime_RandomForest

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

          Mean of squared residuals: 82993.23
                    % Var explained: 43.31
> summary(Crime_RandomForest)
                Length Class  Mode
call              3    -none- call
type              1    -none- character
predicted        47    -none- numeric
mse             500    -none- numeric
rsq             500    -none- numeric
oob.times        47    -none- numeric
importance       15    -none- numeric
importanceSD      0    -none- NULL
localImportance   0    -none- NULL
proximity         0    -none- NULL
ntree             1    -none- numeric
mtry              1    -none- numeric
forest           11    -none- list
coefs             0    -none- NULL
y                47    -none- numeric
test              0    -none- NULL
inbag             0    -none- NULL
terms             3    terms  call
```




```
> output_forest <- predict(Crime_RandomForest, test1_df)
> output_forest
       1
1208.885
```

<u>Plot of the model prediction- RandomForest is much better than previous models discussed in part a</u>

Crime_rf_test <- predict(Crime_RandomForest,US_Crimes_Data[,1:15])
Crime_rf_test
plot(US_Crimes_Data$Crime, Crime_rf_test)



Question 10.2
Describe a situation or problem from your job, everyday life, current events, etc., for which a logistic regression model would be appropriate. List some (up to 5) predictors that you might use.

Answer:
Logistic Regression can be used to predict if a person will buy (dependent variable = 1) or will not buy (dependent variable = 0) a product based on demographic and individual behaviors.
Some of the independent variables are: time since last purchase, how many orders placed last year, total monetary value purchased last year, number of searches on the product, number of the product sold in that zip code.

Question 10.3

1. **Using the GermanCredit data set `germancredit.txt` from http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german / (description at http://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29), use logistic regression to find a good predictive model for whether credit applicants are good credit risks or not. Show your model (factors used and their coefficients), the software output, and the quality of fit. You can use the `glm` function in R. To get a logistic regression (logit) model on data where the response is either zero or one, use `family=binomial(link="logit")` in your `glm` function call.**

Answer:

GermanCredit_Data <- read.table("../GermanCreditSummer2018.txt", header=T)
View(GermanCredit_Data)

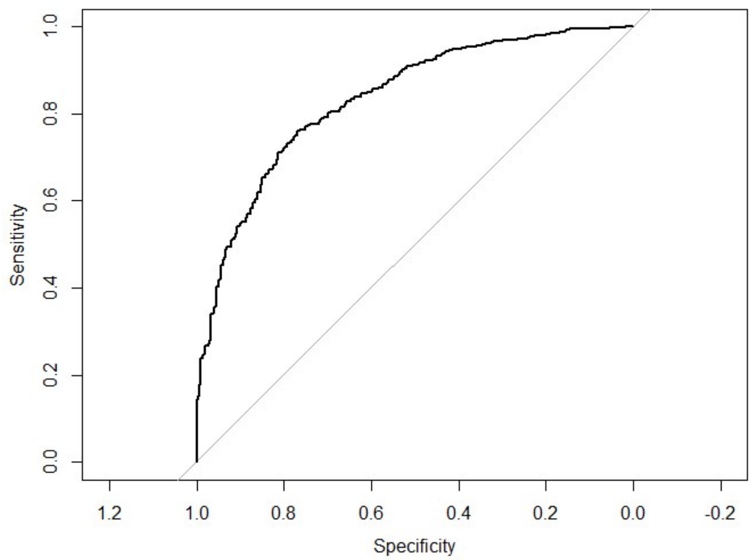
#Convert column X1.1 to binary 0 and 1
#X1.1 1-good, 2-bad
GermanCredit_Data$X1.1_binary <- ifelse(GermanCredit_Data$X1.1 == 2,0,1)

#part 1
GermanCredit_glm <- glm(X1.1_binary~. -X1.1, GermanCredit_Data, family= binomial(link = "logit"))
GermanCredit_glm
summary(GermanCredit_glm)

```
glm(formula = X1.1_binary ~ . - X1.1, family = binomial(link = "logit"),
    data = GermanCredit_Data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6116  -0.7121   0.3760   0.6988   2.3408

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.987e-01  1.084e+00  -0.368 0.713081
A11A12       3.757e-01  2.179e-01   1.724 0.084697 .
A11A13       9.667e-01  3.692e-01   2.619 0.008828 **
A11A14       1.713e+00  2.322e-01   7.376 1.63e-13 ***
```

```
X6          -2.783e-02  9.297e-03  -2.993 0.002762 **
A34A31      -1.432e-01  5.488e-01  -0.261 0.794169
A34A32       5.866e-01  4.305e-01   1.363 0.172988
A34A33       8.535e-01  4.716e-01   1.810 0.070347 .
A34A34       1.435e+00  4.399e-01   3.262 0.001105 **
A43A41       1.666e+00  3.743e-01   4.451 8.54e-06 ***
A43A410      1.488e+00  7.762e-01   1.917 0.055256 .
A43A42       7.909e-01  2.610e-01   3.031 0.002441 **
A43A43       8.899e-01  2.472e-01   3.601 0.000318 ***
A43A44       5.231e-01  7.622e-01   0.686 0.492559
A43A45       2.161e-01  5.500e-01   0.393 0.694385
A43A46      -3.661e-02  3.964e-01  -0.092 0.926423
A43A48       2.059e+00  1.212e+00   1.698 0.089435 .
A43A49       7.395e-01  3.339e-01   2.215 0.026782 *
X1169       -1.283e-04  4.443e-05  -2.887 0.003887 **
A65A62       3.569e-01  2.861e-01   1.247 0.212254
A65A63       3.761e-01  4.011e-01   0.938 0.348468
A65A64       1.339e+00  5.248e-01   2.552 0.010718 *
A65A65       9.443e-01  2.626e-01   3.596 0.000324 ***
A75A72       6.651e-02  4.269e-01   0.156 0.876200
A75A73       1.829e-01  4.104e-01   0.446 0.655928
A75A74       8.310e-01  4.454e-01   1.866 0.062069 .
A75A75       2.762e-01  4.133e-01   0.668 0.503926
X4          -3.301e-01  8.827e-02  -3.739 0.000184 ***
A93A92       2.751e-01  3.865e-01   0.712 0.476530
A93A93       8.152e-01  3.799e-01   2.146 0.031889 *
A93A94       3.670e-01  4.536e-01   0.809 0.418493
A101A102    -4.356e-01  4.101e-01  -1.062 0.288116
A101A103     9.790e-01  4.242e-01   2.308 0.021003 *
X4.1        -4.798e-03  8.639e-02  -0.056 0.955709
A121A122    -2.801e-01  2.534e-01  -1.106 0.268940
A121A123    -1.935e-01  2.360e-01  -0.820 0.412293
A121A124    -7.289e-01  4.245e-01  -1.717 0.085946 .
X67          1.446e-02  9.227e-03   1.568 0.116992
A143A142     1.232e-01  4.119e-01   0.299 0.764790
A143A143     6.459e-01  2.391e-01   2.701 0.006911 **
A152A152     4.435e-01  2.347e-01   1.890 0.058768 .
A152A153     6.841e-01  4.769e-01   1.434 0.151435
X2          -2.720e-01  1.895e-01  -1.435 0.151174
A173A172    -5.362e-01  6.795e-01  -0.789 0.430035
A173A173    -5.554e-01  6.548e-01  -0.848 0.396331
A173A174    -4.793e-01  6.622e-01  -0.724 0.469134
X1          -2.640e-01  2.492e-01  -1.059 0.289416
A192A192     2.992e-01  2.013e-01   1.486 0.137186
A201A202     1.392e+00  6.257e-01   2.225 0.026051 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1221.01  on 998  degrees of freedom
Residual deviance:  895.75  on 950  degrees of freedom
AIC: 993.75

Number of Fisher Scoring iterations: 5
```

```
#Plot ROC
#rocplot(GermanCredit_glm)
prob1=predict(GermanCredit_glm,type=c("response"))
GermanCredit_Data$prob1=prob1

graph_roc <- roc(X1.1_binary ~ prob1, data = GermanCredit_Data)
plot(graph_roc)
graph_roc
auc(graph_roc)
```




```
> auc(graph_roc)
Area under the curve: 0.8335
```

```
> table(GermanCredit_Data$X1.1_binary)

  0   1
300 699
```

#Add a column with predicted values with threshold = 0.5

```
Credit_fitted <- fitted(GermanCredit_glm) #same output as predict
Credit_fitted_binary <- as.data.frame(round(Credit_fitted))
names(Credit_fitted_binary)[1] <- "fitted"
View(Credit_fitted_binary)

table(Credit_fitted_binary$fitted)
table(GermanCredit_Data$X1.1_binary)

#use confusion matrix to check accuracy
confusion_matrix <- confusionMatrix(factor(GermanCredit_Data$X1.1_binary, levels = 0:1),
                        factor(Credit_fitted_binary$fitted, levels = 0:1))
```

confusion_matrix
confusion_matrix$table[1,1]

<u>Set threshold to 0.5 for first estimate</u>

```
> confusion_matrix
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 160 140
         1  74 625

               Accuracy : 0.7858
                 95% CI : (0.759, 0.8109)
    No Information Rate : 0.7658
    P-Value [Acc > NIR] : 0.07152

                  Kappa : 0.4561
 Mcnemar's Test P-Value : 8.859e-06

            Sensitivity : 0.6838
            Specificity : 0.8170
         Pos Pred Value : 0.5333
         Neg Pred Value : 0.8941
             Prevalence : 0.2342
         Detection Rate : 0.1602
   Detection Prevalence : 0.3003
      Balanced Accuracy : 0.7504

       'Positive' Class : 0
```

2. **Because the model gives a result between 0 and 1, it requires setting a threshold probability to separate between "good" and "bad" answers. In this data set, they estimate that incorrectly identifying a bad customer as good, is 5 times worse than incorrectly classifying a good customer as bad. Determine a good threshold probability based on your model.**

<u>Answer:</u>

#Part 2: In this data set, they estimate that incorrectly identifying a bad customer as good, is 5 times worse than incorrectly classifying a good customer as bad.
#Set weight factors, 1 is good, 0 is bad
FN_cost <- 1
FP_cost <- 5
TP_cost <- 0
TN_cost <- 0

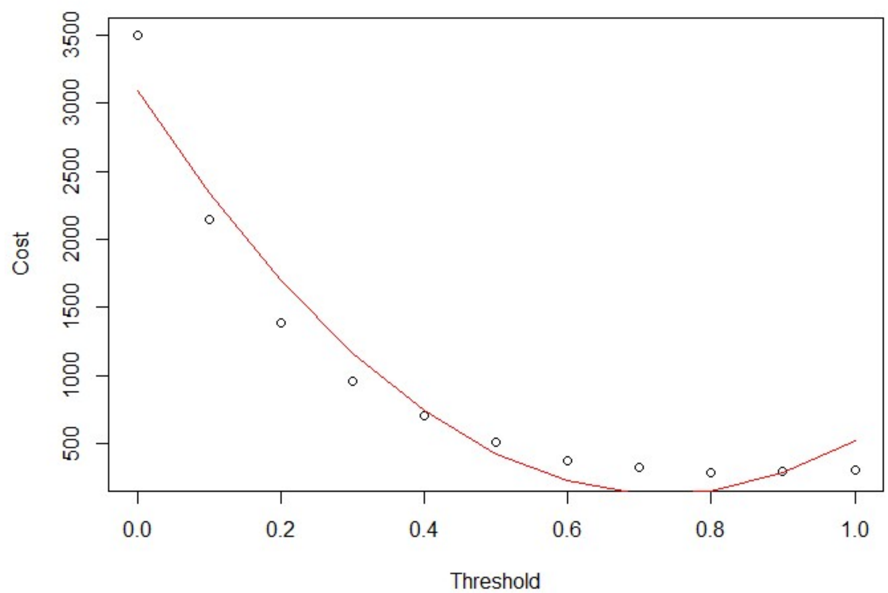
Total_cost <- confusion_matrix$table[1,1]*TP_cost + confusion_matrix$table[1,2]*FN_cost +
confusion_matrix$table[2,1]*FP_cost + confusion_matrix$table[2,2]*TN_cost

Total_cost

```
#Find the optimized Threshold
#Create a column with different threshold and corresponding cost

threshold_df <- data.frame(Threshold=numeric(), Cost=numeric())
#View(threshold_df)
```

Make a table of Threshold and Cost

```
for (index1 in 0:10)
{
  #index1 <-0 #for debugging
  Threshold1 <- index1/10

  Credit_fitted_bin1 <- as.data.frame(round(Credit_fitted-0.5+Threshold1))
  names(Credit_fitted_bin1)[1] <- "fitted"

  confusion_matrix1 <- confusionMatrix(factor(GermanCredit_Data$X1.1_binary, levels = 0:1),
                                  factor(Credit_fitted_bin1$fitted, levels = 0:1))
  confusion_matrix1

  Total_cost1 <- confusion_matrix1$table[1,1]*TP_cost + confusion_matrix1$table[1,2]*FN_cost +
    confusion_matrix1$table[2,1]*FP_cost + confusion_matrix1$table[2,2]*TN_cost

  threshold_df[nrow(threshold_df)+1,] <- c(Threshold1,Total_cost1)
}

View(threshold_df)
plot(threshold_df)
curve_Fit1 <- nls(Cost~a*Threshold+b*Threshold^2+c, data=threshold_df, start = list (a=0.01, b=0.01,
c=0.01))
curve_Fit1
lines(threshold_df$Threshold, predict(curve_Fit1), col="red")
nls_params <- curve_Fit1$m$getAllPars()
```




```
> curve_Fit1
Nonlinear regression model
  model: Cost ~ a * Threshold + b * Threshold^2 + c
   data: threshold_df
    a     b     c 
-8105  5532  3095 
 residual sum-of-squares: 469647

Number of iterations to convergence: 1 
Achieved convergence tolerance: 9.223e-08
```

```
nls_Function <- function(x) {nls_params[1]*x + nls_params[2]*x^2 + nls_params[3]}
optimize(nls_Function, threshold_df$Threshold, maximum = F)
> optimize(nls_Function, threshold_df$Threshold, maximum = F)
$minimum
[1] 0.732521

$objective
       a 
126.2443 
```

```
#Optimized solution
optimizedThreshold <- 0.732521
Credit_fitted_bin2 <- as.data.frame(round(Credit_fitted-0.5+optimizedThreshold))
names(Credit_fitted_bin2)[1] <- "fitted"

confusion_matrix2 <- confusionMatrix(factor(GermanCredit_Data$X1.1_binary, levels = 0:1),
                       factor(Credit_fitted_bin2$fitted, levels = 0:1))
confusion_matrix2
> confusion_matrix2
Confusion Matrix and Statistics
```

```
          Reference
Prediction   0   1
         0  63 237
         1  15 684

               Accuracy : 0.7477
                 95% CI : (0.7196, 0.7744)
    No Information Rate : 0.9219
    P-Value [Acc > NIR] : 1

                  Kappa : 0.239
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.80769
            Specificity : 0.74267
         Pos Pred Value : 0.21000
         Neg Pred Value : 0.97854
             Prevalence : 0.07808
         Detection Rate : 0.06306
   Detection Prevalence : 0.30030
      Balanced Accuracy : 0.77518

       'Positive' Class : 0
```

Total_cost2 <- confusion_matrix2$table[1,1]*TP_cost + confusion_matrix2$table[1,2]*FN_cost + confusion_matrix2$table[2,1]*FP_cost + confusion_matrix2$table[2,2]*TN_cost

Total_cost2

```
> Total_cost2
[1] 312
```

	Threshold	Cost
1	0.0	3495
2	0.1	2149
3	0.2	1382
4	0.3	953
5	0.4	708
6	0.5	510
7	0.6	371
8	0.7	321
9	0.8	281
10	0.9	294
11	1.0	300

Curve fitted solution of 312 is higher than the actual data of 281 for Threshold of 0.8. So, Threshold of 0.8 is recommended.

Complete Source Code:

```
# eDX Intro to Analytics HW week 4- Principal Component Analysis

# Clear environment
rm(list = ls())
set.seed(1)
library(pls)

US_Crimes_Data <- read.table("../UScrimeSummer2018.txt", header=T)
View(US_Crimes_Data)

crimes_pca <- prcomp(US_Crimes_Data, scale. = T)
crimes_pca
plot(crimes_pca)
summary(crimes_pca)

#Principal Component Regression, PCR in pls package
crimes_pcr <- pcr(Crime~., data= US_Crimes_Data, scale=T, validation= "CV")
crimes_pcr
summary(crimes_pcr)

test1_df <- 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,
            Crime = 0)

output1 <- predict(crimes_pcr, test1_df)
output1
```

```
# eDX Intro to Analytics HW week 4- Regression Tree & Random Forest

# Clear environment
```

```
rm(list = ls())
set.seed(1)

library(tree)
library(randomForest)
library(rpart)

US_Crimes_Data <- read.table("../UScrimeSummer2018.txt", header=T)
View(US_Crimes_Data)

#part a: Tree model
Crime_tree <- tree(Crime~., US_Crimes_Data)
Crime_tree
summary(Crime_tree)
plot(Crime_tree)

Crime_rpart <- rpart(Crime~., data= US_Crimes_Data, method= "anova")
Crime_rpart
summary(Crime_rpart)

Crime_rpart_test <- predict(Crime_rpart,US_Crimes_Data[,1:15])
plot(US_Crimes_Data$Crime, Crime_rpart_test)

test1_df <- 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,
                Crime = 0)

output_rpart <- predict(Crime_rpart, test1_df)
output_rpart

output_tree <- predict(Crime_tree, test1_df)
output_tree

Crime_tree_test <- predict(Crime_tree,US_Crimes_Data[,1:15])
plot(US_Crimes_Data$Crime, Crime_tree_test)
```

```
#----------------------
#part b: Random forest
Crime_RandomForest <- randomForest(Crime~., US_Crimes_Data)
Crime_RandomForest
summary(Crime_RandomForest)
plot(Crime_RandomForest)

output_forest <- predict(Crime_RandomForest, test1_df)
output_forest

Crime_rf_test <- predict(Crime_RandomForest,US_Crimes_Data[,1:15])
Crime_rf_test
plot(US_Crimes_Data$Crime, Crime_rf_test)
```

```
# eDX Intro to Analytics HW week 4- Logistic Regression

# Clear environment
rm(list = ls())
set.seed(1)
library(pROC)
#library(Deducer)
library(caret)

GermanCredit_Data <- read.table("../GermanCreditSummer2018.txt", header=T)
View(GermanCredit_Data)

#Convert column X1.1 to binary 0 and 1
#X1.1 1-good, 2-bad
GermanCredit_Data$X1.1_binary <- ifelse(GermanCredit_Data$X1.1 == 2,0,1)

#part 1
GermanCredit_glm <- glm(X1.1_binary~. -X1.1, GermanCredit_Data, family= binomial(link = "logit"))
GermanCredit_glm
summary(GermanCredit_glm)

#Plot ROC
#rocplot(GermanCredit_glm)
prob1=predict(GermanCredit_glm,type=c("response"))
GermanCredit_Data$prob1=prob1

graph_roc <- roc(X1.1_binary ~ prob1, data = GermanCredit_Data)
plot(graph_roc)
graph_roc
auc(graph_roc)
```

```
#Add a column with predicted values with threshold = 0.5
Credit_fitted <- fitted(GermanCredit_glm) #same output as predict
Credit_fitted_binary <- as.data.frame(round(Credit_fitted))
names(Credit_fitted_binary)[1] <- "fitted"
View(Credit_fitted_binary)

table(Credit_fitted_binary$fitted)
table(GermanCredit_Data$X1.1_binary)

#Need to use factor for the confusion matrix input
test1 <- factor(GermanCredit_Data$X1.1_binary, levels = 0:1)
View(test1)

#use confusion matrix to check accuracy
confusion_matrix <- confusionMatrix(factor(GermanCredit_Data$X1.1_binary, levels = 0:1),
                   factor(Credit_fitted_binary$fitted, levels = 0:1))
confusion_matrix
confusion_matrix$table[1,1]


#Part 2: In this data set, they estimate that incorrectly identifying a bad customer as good, is 5 times worse than incorrectly classifying a good customer as bad.
#Set weight factors, 1 is good, 0 is bad
FN_cost <- 1
FP_cost <- 5
TP_cost <- 0
TN_cost <- 0

Total_cost <- confusion_matrix$table[1,1]*TP_cost + confusion_matrix$table[1,2]*FN_cost +
              confusion_matrix$table[2,1]*FP_cost + confusion_matrix$table[2,2]*TN_cost

Total_cost

#Find the optimized Threshold
#Create a column with different threshold and corresponding cost

threshold_df <- data.frame(Threshold=numeric(), Cost=numeric())
#View(threshold_df)

for (index1 in 0:10)
{
  #index1 <-0 #for debugging
  Threshold1 <- index1/10

  Credit_fitted_bin1 <- as.data.frame(round(Credit_fitted-0.5+Threshold1))
  names(Credit_fitted_bin1)[1] <- "fitted"

  confusion_matrix1 <- confusionMatrix(factor(GermanCredit_Data$X1.1_binary, levels = 0:1),
```

```
                     factor(Credit_fitted_bin1$fitted, levels = 0:1))
  confusion_matrix1

  Total_cost1 <- confusion_matrix1$table[1,1]*TP_cost + confusion_matrix1$table[1,2]*FN_cost +
    confusion_matrix1$table[2,1]*FP_cost + confusion_matrix1$table[2,2]*TN_cost

  threshold_df[nrow(threshold_df)+1,] <- c(Threshold1,Total_cost1)
}

View(threshold_df)
plot(threshold_df)
curve_Fit1 <- nls(Cost~a*Threshold+b*Threshold^2+c, data=threshold_df, start = list (a=0.01, b=0.01,
c=0.01))
curve_Fit1
lines(threshold_df$Threshold, predict(curve_Fit1), col="red")
nls_params <- curve_Fit1$m$getAllPars()


nls_Function <- function(x) {nls_params[1]*x + nls_params[2]*x^2 + nls_params[3]}
optimize(nls_Function, threshold_df$Threshold, maximum = F)

#Optimized solution
optimizedThreshold <- 0.732521
Credit_fitted_bin2 <- as.data.frame(round(Credit_fitted-0.5+optimizedThreshold))
names(Credit_fitted_bin2)[1] <- "fitted"

confusion_matrix2 <- confusionMatrix(factor(GermanCredit_Data$X1.1_binary, levels = 0:1),
                     factor(Credit_fitted_bin2$fitted, levels = 0:1))
confusion_matrix2

Total_cost2 <- confusion_matrix2$table[1,1]*TP_cost + confusion_matrix2$table[1,2]*FN_cost +
  confusion_matrix2$table[2,1]*FP_cost + confusion_matrix2$table[2,2]*TN_cost

Total_cost2
```