ISYE 6051 : Homework 7
3/10/2021

Table of Contents

10.1a Regression Tree model

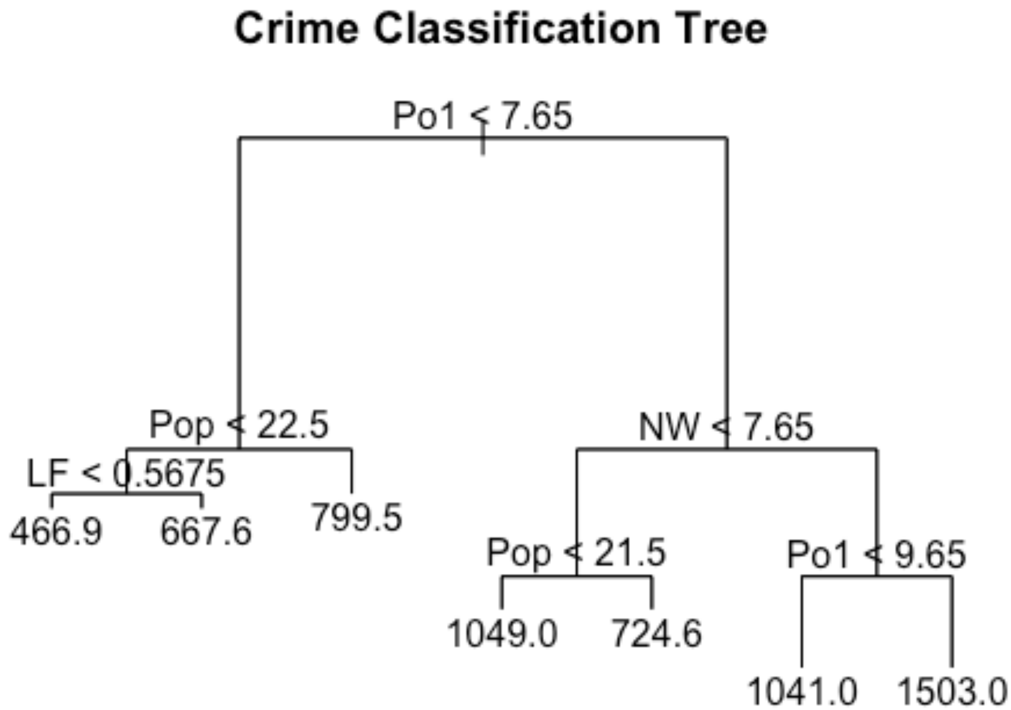
Using the tree library, I initially developed a regression tree using all of the factors in uscrime.txt. The data was not split into training and testing subsets due to the fact that the total number of data points available is only 47.

4 variables are used in the creation of the tree- Po1, Pop, LF and NW. There are 7 terminal nodes. These can be seen either printing out the vector (treefit) or in the summary data of the vector.

```
#Homework Week 7 Data Transformatin and PCA Analysis
# Question 10.1 a) Regression tree model and b) random forest model

rm(list = ls())
set.seed(17)
setwd("~/Documents/ISYE6501 Intro to Analytics Modeling/FA_SP_hw7")

#library uploads
install.packages("tree")
install.packages("randomForest")

library(tree)
library(randomForest)


#Read in Data
crimedf <- read.table("uscrime.txt", header = TRUE)
head(crimedf)


# Create a regression tree using tree
# Did not create training and test datasets since the original only includes 47
observations
```

```
treefit <- tree(Crime~.,data=crimedf, method = 'class')
treefit

summary(treefit)
```

```
treefit <- tree(Crime~.,data=crimedf, method = 'class')
> treefit
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(treefit)

Regression tree:
tree(formula = Crime ~ ., data = crimedf, method = "class")
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
```

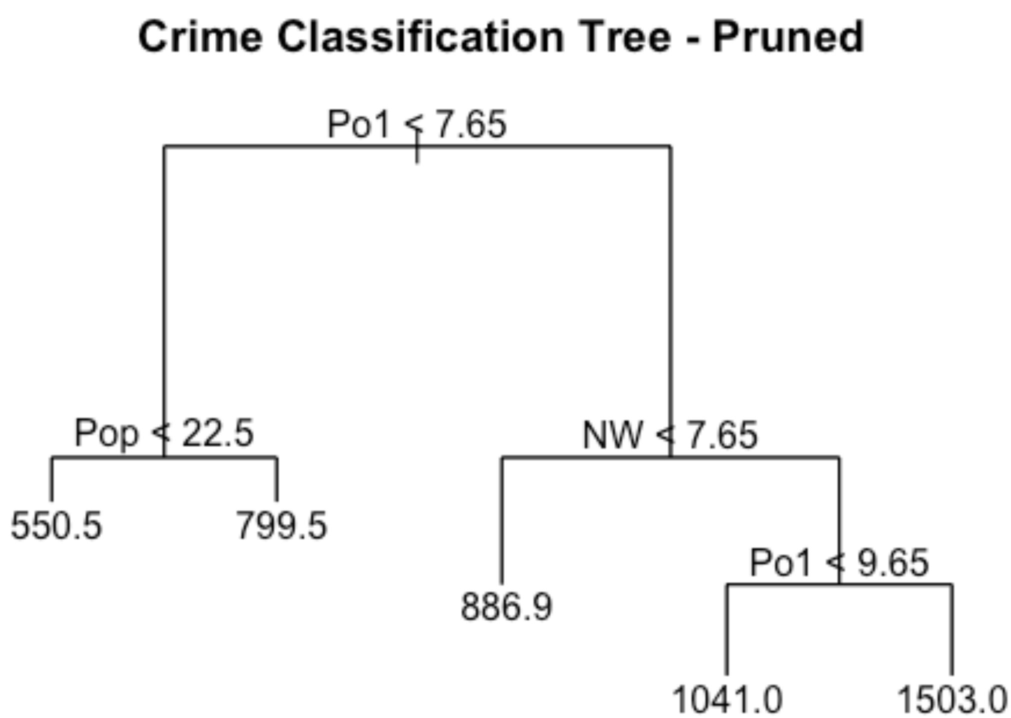


To evaluate the fit of this model, the regression tree is now pruned to determine if having fewer nodes provides a better fit for the model. I selected 5 terminal nodes for this model.

```
#Next prune the tree down to 5 nodes

treenode <- 5
prune.treefit <- prune.tree(treefit, best = treenode)
summary(prune.treefit)
plot(prune.treefit)
text(prune.treefit)
title("Crime Classification Tree - Pruned")
```

```
prune.treefit
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 *
    5) Pop > 22.5 11  179500  799.5 *
  3) Po1 > 7.65 24 3604000 1131.0
    6) NW < 7.65 10  557600  886.9 *
    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(prune.treefit)

Regression tree:
snip.tree(tree = treefit, nodes = c(4L, 6L))
Variables actually used in tree construction:
[1] "Po1" "Pop" "NW"
Number of terminal nodes:  5
Residual mean deviance:  54210 = 2277000 / 42
Distribution of residuals:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -573.9  -107.5    15.5     0.0   122.8   490.1
```

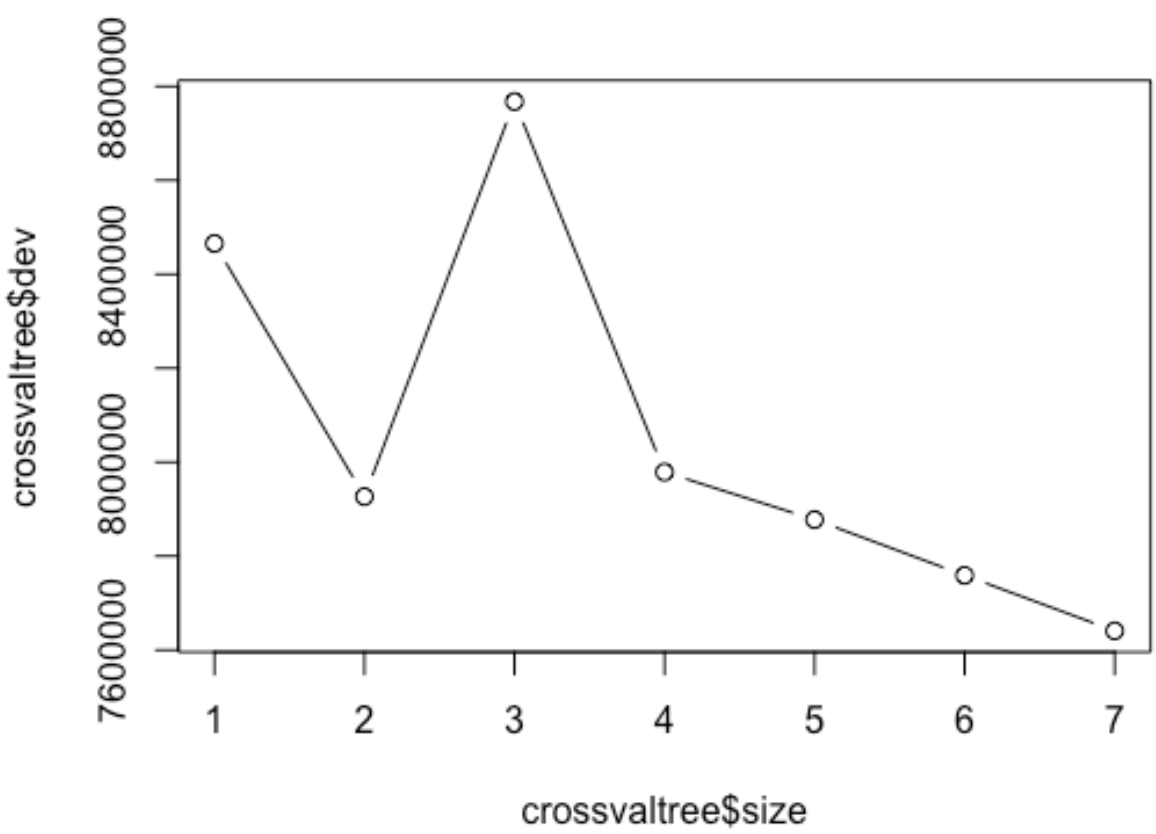


The Residual mean deviance is higher in the pruned tree (54210 versus 47390) which may indicate overfitting and having fewer nodes with this dataset is not a good fit.

Cross validation can be used with regression trees to validate the fit of the model. Based on the deviations, size and plot it appears that using 7 terminal nodes is best since it has the fewest errors. Pruning the regression tree to 5 models did not provide value in the reduction of errors.

```
#Cross Validation and view the deviation to determine the best # of nodes
crossvaltree <-cv.tree(treefit)
crossvaltree$dev
plot(crossvaltree$size, crossvaltree$dev, type = "b")
```

```
crossvaltree$dev
[1] 7640892 7759046 7877708 7978826 8767659 7926751 8465636
plot(crossvaltree$size, crossvaltree$dev, type = "b")
```



I then used predict() to determine the quality of fit for the regression tree model. R-squared is .7244962 which indicates approximately 72% of the data fit the model. However, since the Regression mean value increased on the pruned tree there may in fact be overfitting.

<u>10.1b Random Forest model</u>

Using randomforest regression trees, I attempted different values of mtry to determine what may provide the best R-squared value. Mtry = 4 (factors) provided the best value – 42.85% even though is is significantly lower than the 72% achieved using a regression tree. The randomforest functionality produced 500 trees.

```
#Evaluating Crime data using randomforest trees
```

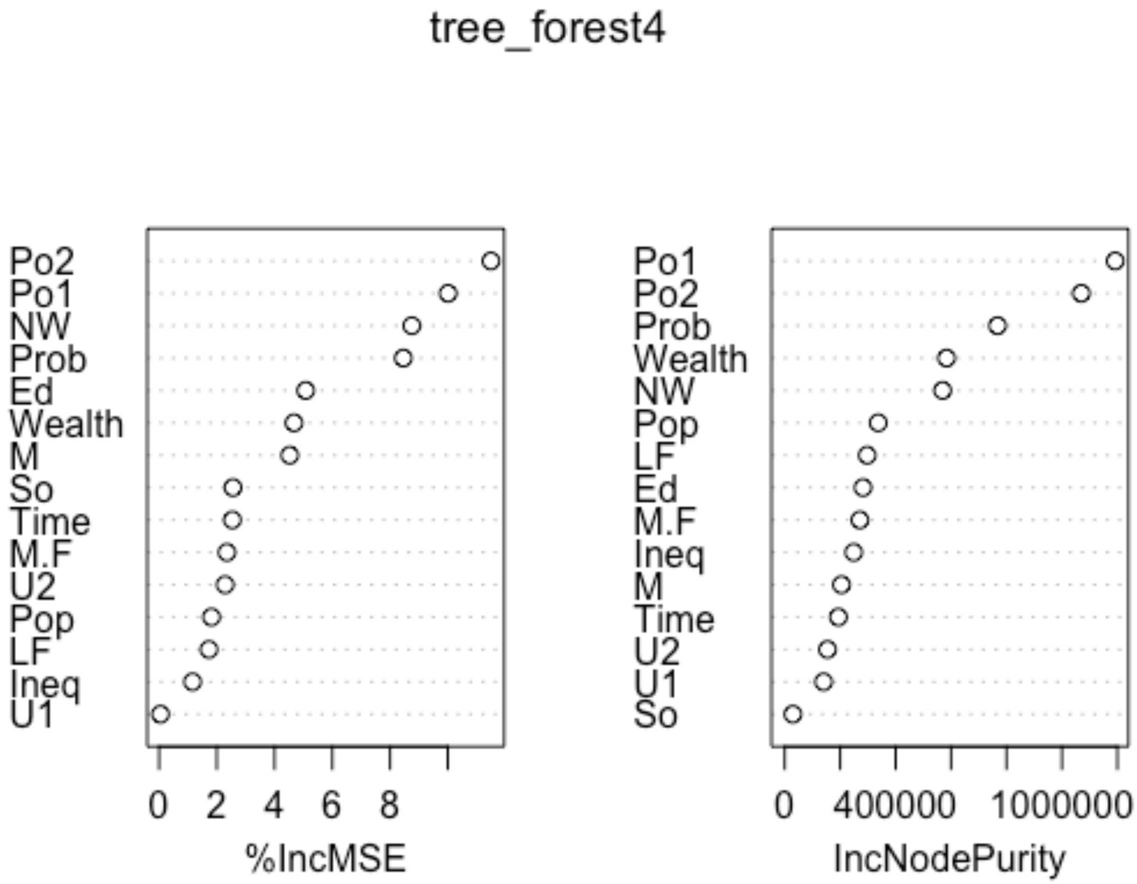
```
> tree_forest7 <- randomForest(Crime~., data = crimedf, importance = TRUE, mtry = 7)
> tree_forest7

Call:
 randomForest(formula = Crime ~ ., data = crimedf, importance = TRUE,      mtry = 7)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 7

          Mean of squared residuals: 89708.06
                    % Var explained: 38.73
> tree_forest7.predict <- predict(tree_forest7, data = crimedf[,1:15])
>
> tree_forest5 <- randomForest(Crime~., data = crimedf, importance = TRUE, mtry = 5)
> tree_forest5

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

          Mean of squared residuals: 84047.57
                    % Var explained: 42.59
> tree_forest5.predict <- predict(tree_forest5, data = crimedf[,1:15])
>
>
> tree_forest4 <- randomForest(Crime~., data = crimedf, importance = TRUE, mtry = 4)
> tree_forest4

Call:
```

```
 randomForest(formula = Crime ~ ., data = crimedf, importance = TRUE,      mtry = 4)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 4

          Mean of squared residuals: 83675.34
                    % Var explained: 42.85
> tree_forest4.predict <- predict(tree_forest4, data = crimedf[,1:15])
>
>
> tree_forest3 <- randomForest(Crime~., data = crimedf, importance = TRUE, mtry = 3)
> tree_forest3

Call:
 randomForest(formula = Crime ~ ., data = crimedf, importance = TRUE,      mtry = 3)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 3

          Mean of squared residuals: 86280.59
                    % Var explained: 41.07
```

Looking at the importance() of the data the top 4 factors are – Po2, Po1, NW and Prob based on %IncMSE.

```
importance(tree_forest4)
           %IncMSE IncNodePurity
M       4.52626758     204871.31
So      2.55875094      29620.86
Ed      5.07821017     282415.36
Po1    10.00875290    1192258.35
Po2    11.49427680    1070043.03
LF      1.72792931     297750.46
M.F     2.35055137     271258.01
Pop     1.81591520     337266.58
NW      8.75473116     569578.76
U1      0.04885049     140328.68
U2      2.28792819     154475.31
Wealth 4.67313115      584010.19
Ineq    1.16012959     248854.14
Prob    8.45817937     767842.87
Time    2.53949544     193533.12
```

The best value for R-squared shows that in this example that the randomforest methodology



10.2 Examples of Logistic Regression

My real-world example of a situation where Logistic Regression would be of use is in determining the probability that an individual who contracts Covid-19 will end up hospitalized. 5 predictors could be: 1) Presence of high-risk comorbidities, 2) In-person worker (versus having the ability to work remotely), 3) Housing density, 4) Blood oxygen saturation levels and 5) socio-economic level.

10.3 Logistic Regression model for Credit Applicants

Data is read in for German credit data. The good or bad variables are then converted to 0 and 1s per the homework assignment.

```
#Homework 10.3 Logistic Regression

#Read in credit data
set.seed(17)
creditdf <- read.table("germancredit.txt", header = FALSE)
head(creditdf)

install.packages("caTools")
library(caTools)

#V21 reflects 1= good and 2=bad for credit risks
#Remember that it is 5 times as bad to misclassify an applicant as good
#as it is to classify them as bad

table(creditdf$V21)

#convert the data to 0 and 1 for good and bad
creditdf$V21[creditdf$V21==1] <- 0
creditdf$V21[creditdf$V21==2] <- 1
```

```
table(creditdf$V21)

  0   1
700 300
```

This data indicates that there are 700 good applicants and 300 bad applicants. Data is then split into Training and Testing 70% and 30%.

```
#Split the data into training and validation datasets 70% and 30%
samplemetric = sample.split(creditdf, SplitRatio = 0.70)
credittrain <- subset(creditdf, samplemetric == TRUE)
credittest <- subset(creditdf, samplemetric == FALSE)

table(credittrain$V21)
table(credittest$V21)
head(credittest)
```

```
table(credittrain$V21)

  0   1
```

```
471 195
> table(credittest$V21)

  0   1
229 105
> head(credittest)
   V1 V2  V3  V4    V5  V6  V7 V8  V9  V10 V11  V12 V13  V14  V15 V16  V17 V18
2  A12 48 A32 A43  5951 A61 A73  2 A92 A101   2 A121  22 A143 A152   1 A173   1
3  A14 12 A34 A46  2096 A61 A74  2 A93 A101   3 A121  49 A143 A152   1 A172   2
4  A11 42 A32 A42  7882 A61 A74  2 A93 A103   4 A122  45 A143 A153   1 A173   2
8  A12 36 A32 A41  6948 A61 A73  2 A93 A101   2 A123  35 A143 A151   1 A174   1
12 A11 48 A32 A49  4308 A61 A72  3 A92 A101   4 A122  24 A143 A151   1 A173   1
19 A12 24 A32 A41 12579 A61 A75  4 A92 A101   2 A124  44 A143 A153   1 A174   1
   V19  V20 V21
2  A191 A201   1
3  A191 A201   0
4  A191 A201   0
8  A192 A201   0
12 A191 A201   1
19 A192 A201   1
```

Create a model using all of the predictors.

```
creditmodel <- glm(V21~., data = credittrain, family = binomial(link="logit"))
> summary(creditmodel)

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

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.8413  -0.6945  -0.3454   0.6955   2.5671

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  4.565e-01  1.403e+00   0.325 0.744949
V1A12       -3.243e-01  2.674e-01  -1.213 0.225142
V1A13       -1.550e+00  4.719e-01  -3.284 0.001022 **
V1A14       -2.007e+00  2.997e-01  -6.698 2.12e-11 ***
V2           3.419e-02  1.181e-02   2.896 0.003776 **
V3A31       -2.318e-01  7.040e-01  -0.329 0.741952
V3A32       -1.883e-01  5.309e-01  -0.355 0.722749
V3A33       -5.493e-01  5.787e-01  -0.949 0.342490
V3A34       -7.620e-01  5.339e-01  -1.427 0.153478
V4A41       -2.132e+00  5.667e-01  -3.762 0.000169 ***
```

```
V4A410      -1.024e+00  8.606e-01  -1.190 0.234208
V4A42       -7.933e-01  3.183e-01  -2.493 0.012677 *
V4A43       -7.705e-01  2.993e-01  -2.575 0.010035 *
V4A44       -1.019e+00  1.023e+00  -0.996 0.319118
V4A45        7.135e-03  8.020e-01   0.009 0.992902
V4A46        4.131e-02  5.090e-01   0.081 0.935319
V4A48       -1.684e+00  1.262e+00  -1.335 0.181968
V4A49       -8.031e-01  4.298e-01  -1.869 0.061678 .
V5           8.226e-05  5.418e-05   1.518 0.128973
V6A62       -3.114e-01  3.628e-01  -0.858 0.390698
V6A63       -1.006e+00  5.654e-01  -1.780 0.075137 .
V6A64       -1.861e+00  6.902e-01  -2.696 0.007009 **
V6A65       -8.509e-01  3.315e-01  -2.567 0.010269 *
V7A72       -1.524e-01  5.802e-01  -0.263 0.792846
V7A73        4.303e-02  5.566e-01   0.077 0.938385
V7A74       -6.571e-01  5.966e-01  -1.101 0.270715
V7A75       -4.023e-01  5.587e-01  -0.720 0.471488
V8           2.780e-01  1.089e-01   2.554 0.010645 *
V9A92       -5.046e-01  4.608e-01  -1.095 0.273503
V9A93       -8.243e-01  4.491e-01  -1.835 0.066466 .
V9A94       -3.051e-01  5.371e-01  -0.568 0.569991
V10A102     -1.571e-01  5.075e-01  -0.310 0.756866
V10A103     -1.406e+00  5.525e-01  -2.544 0.010945 *
V11         -1.120e-01  1.076e-01  -1.040 0.298347
V12A122      6.235e-01  3.162e-01   1.972 0.048607 *
V12A123      5.043e-01  2.999e-01   1.682 0.092586 .
V12A124      1.096e+00  5.158e-01   2.125 0.033579 *
V13         -1.343e-02  1.198e-02  -1.121 0.262333
V14A142      1.189e-01  5.071e-01   0.234 0.814600
V14A143     -5.050e-01  3.029e-01  -1.667 0.095453 .
V15A152     -4.169e-01  2.926e-01  -1.425 0.154248
V15A153     -8.037e-01  5.928e-01  -1.356 0.175178
V16          4.452e-01  2.434e-01   1.829 0.067388 .
V17A172      4.076e-01  9.443e-01   0.432 0.666045
V17A173      5.004e-01  9.193e-01   0.544 0.586260
V17A174      1.284e-01  9.019e-01   0.142 0.886811
V18         -1.422e-01  3.340e-01  -0.426 0.670156
V19A192     -2.247e-01  2.554e-01  -0.880 0.379012
V20A202     -9.065e-01  6.925e-01  -1.309 0.190481
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 805.37  on 665  degrees of freedom
Residual deviance: 583.87  on 617  degrees of freedom
```

```
AIC: 681.87

Number of Fisher Scoring iterations: 5

>
> #Evaluate the creditmodel to determine the predictions to determine
> #the confusion matrix
>
> predictdata <- predict(creditmodel, newdata=credittest[,-21], type="response")
> table(credittest$V21, round (predictdata))

     0   1
  0 205  24
  1  52  53
```

The confusion matrix shows that the number of FPs are higher than acceptable when classifying an applicant as good has 5x the cost of a FN.
The next step is to evaluate the results and remove some predictors in order to get to an acceptable level of FPs.

```
creditmodel2 <-glm(V21 ~ V1+V2+V3+V4+V5+V6+V8+V9+V10+V20, data = credittrain,
family = binomial(link="logit"))
> summary(creditmodel2)

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

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1588  -0.7289  -0.3924   0.7880   2.5187

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.757e-01  7.213e-01   0.244 0.807574
V1A12       -3.263e-01  2.547e-01  -1.281 0.200221
V1A13       -1.457e+00  4.498e-01  -3.239 0.001198 **
V1A14       -1.975e+00  2.861e-01  -6.901 5.15e-12 ***
V2           3.813e-02  1.112e-02   3.428 0.000608 ***
V3A31       -4.121e-01  6.449e-01  -0.639 0.522835
V3A32       -6.719e-01  4.814e-01  -1.396 0.162777
V3A33       -7.693e-01  5.493e-01  -1.400 0.161393
V3A34       -1.007e+00  5.047e-01  -1.996 0.045953 *
V4A41       -2.155e+00  5.598e-01  -3.850 0.000118 ***
V4A410      -1.274e+00  8.225e-01  -1.549 0.121415
V4A42       -5.762e-01  2.958e-01  -1.948 0.051427 .
```

```
V4A43       -7.181e-01  2.802e-01  -2.563 0.010391 *
V4A44       -1.117e+00  9.828e-01  -1.136 0.255867
V4A45       -2.591e-02  7.453e-01  -0.035 0.972269
V4A46        1.705e-01  4.873e-01   0.350 0.726368
V4A48       -1.695e+00  1.211e+00  -1.399 0.161728
V4A49       -7.100e-01  4.054e-01  -1.751 0.079865 .
V5          6.385e-05  5.003e-05   1.276 0.201808
V6A62       -1.423e-01  3.388e-01  -0.420 0.674469
V6A63       -1.043e+00  5.505e-01  -1.895 0.058124 .
V6A64       -1.770e+00  6.511e-01  -2.719 0.006558 **
V6A65       -8.417e-01  3.128e-01  -2.691 0.007128 **
V8          2.610e-01  1.028e-01   2.538 0.011153 *
V9A92       -3.154e-01  4.281e-01  -0.737 0.461324
V9A93       -7.980e-01  4.161e-01  -1.918 0.055104 .
V9A94       -1.892e-01  5.039e-01  -0.375 0.707362
V10A102      3.418e-02  4.861e-01   0.070 0.943952
V10A103     -1.294e+00  5.134e-01  -2.521 0.011711 *
V20A202     -7.266e-01  6.466e-01  -1.124 0.261086
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 805.37  on 665  degrees of freedom
Residual deviance: 611.31  on 636  degrees of freedom
AIC: 671.31

Number of Fisher Scoring iterations: 5
```

The prediction will now be recalculated to determine if an updated confusion matrix will reflect a lower number of FPs.

```
predictdata2 <- predict(creditmodel2, newdata=credittest[,-21], type="response")
> summary(predictdata2)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.002574 0.074165 0.187889 0.273440 0.470090 0.900388
```

```
predicted_dataround <- round(predictdata2)
> confusion_matrix <- as.matrix(table(predicted_dataround,credittest$V21))
> confusion_matrix

predicted_dataround   0   1
                  0 204  54
                  1  25  51
>
```

```
> accuracy <- (confusion_matrix[1,1] + confusion_matrix[2,2]) / sum(confusion_matrix)
> accuracy
[1] 0.7634731
>
> sensitivity <- (confusion_matrix[1,1]) / (confusion_matrix[1,1] + confusion_matrix[2,1])
> sensitivity
[1] 0.8908297
>
> specificity <- (confusion_matrix[2,2]) / (confusion_matrix[2,2] + confusion_matrix[2,1])
> specificity
[1] 0.6710526
```

The number of FPs has now dropped to a lower number that is acceptable. The accuracy of the model is 76% with specificity of 67%.