

### Question 10.1

- a) The regression tree model generated from the code in Figure 1 determined a model that is most accurate when fed all of the information from the uscrime dataset. However, given the small scope of the uscrime dataset (many variables but little observations), it can be deduced that there is definitely overfitting occurring. With that being said, going deeper into the actual results of the model in Figures 2 & 3, the decision tree made with all of the data in hand determined that the primary factor was Po1, while the 70% training data found this value to be NW. While the primary factor differed between these two models, they both did include the other at some point in the decision tree. Based on this, it would make sense to observe that these are critical factors in determining the Crime value. Furthermore, we can be sure that this tree has been optimized because of what the rpart function does to determine it. Behind the scenes rpart is automatically applying a range of cost complexity  $\alpha$  values to prune the tree. To compare the error for each  $\alpha$  value, rpart performs a 10-fold cross validation so that the error associated with a given  $\alpha$  value is computed on the hold-out validation data. Despite this, the overfitting of this model can not be ignored. As seen in the SSE values in Figures 2.1 and 3.1, the SSE was significantly reduced when  $Po1 < 7.65$  as seen in 4), 5) compared to 6), 7). This further supports the notion that Po1 is an important factor in determining Crime, and potentially validating that a good model could be generated for that specific subset (i.e when  $Po1 < 7.65$ ).
- b) The random forest model is much more difficult to interpret than the regression tree model. This model tries to account for the overfitting identified in the regression tree model by increasing the number of trees, but altering the variables that are used to separate the branches. I believe this to be true in our case as well, given that the MSE is relatively low when the number of trees is equal to 94 (Figures 5 & 6 provide more detail on this topic). With that being said, the % of variance explained (AKA R-squared) only accounted for 40.36%, meaning that strength between the predictors and the predicted Crime value is only that high. So although this method was chosen to combat overfitting, and it did seem to perform this job better than the regression tree model, it still is lacking enough to be a well performing model.

### Question 10.2

I used to work at an insurance agency that often wanted to detect cases of fraud so that they could increase their profits. However, they needed a system to which they would identify a customer potentially involved in fraud vs one who is not. I believe that running a logistic regression on the customers based on their account profile, and weighing it accordingly, would provide a basis to see which customers would be eligible for further investigation by the fraud department. The following list contains examples of predictors we could have used:

1. Amount of claims
2. Amount of money paid out

3. Amount of money they have paid us
4. Criminal history
5. Income

#### Question 10.3

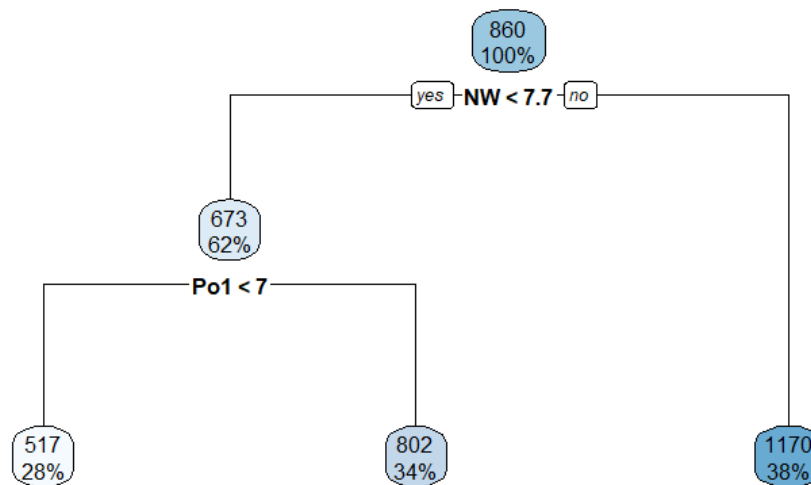
- 1) Using the GermanCredit data, logistic regression was performed to predict which credit risk would be “good” or “bad.” Firstly, the final column was changed from 1s and 2s to 0s and 1, so that the data could be read by the logistic regression model. Next, the data was split at a 70/30 percent ratio for the training and testing sets, respectively. Afterward, a logistic regression model was generated on the testing data so that the significant predictors could be determined. As seen in the previous homeworks, having too many predictors could introduce the potential for overfitting of a model. Finding the significant predictors can be seen in Figure 8, and then in Figure 9 the final model was generated using those values. After predicting the test data with the model generated from only significant predictors, any value that was greater than 0.5 was determined to be a “good” credit risk. This value of 0.5 was chosen arbitrarily and only because the conventional way in mathematics is to round up when the final digit is greater than 5. Finally, the accuracy in comparison to the real dataset was calculated, and it was found to be 74%. This value seems fairly good when relating it to the real dataset.
- 2) To determine a stricter probability threshold, as you would in a case like this where the risk of determining a 0 as a 1 is much greater, you can alter the threshold value set on line 24 in Figure 7. By increasing this value, you do sacrifice some accuracy, but it also sets a stricter guideline on which credit risks are accepted. By strengthening the constraints on which numbers get rounded up, the risk of selecting a bad credit risk as a good one decreases.

```

1 set.seed(694)
2
3 data <- read.table("uscrime.txt", header=TRUE)
4
5 library(rsample)
6 library(rpart)
7 library(rpart.plot)
8
9 #The following block of code for splitting the data is omitted as per caption in Figure 3.
10
11 split <- initial_split(data, prop=0.7) #70/30 split of the train/test data respectively
12 train <- training(split)
13 test <- testing(split)
14
15 m1 <- rpart(
16   formula = Crime ~ .,
17   data = data,
18   method = "anova"
19 )
20
21 m1
22
23 rpart.plot(m1)
24
25 plotcp(m1)

```

**Figure 1.** This is the code used to generate the decision trees visualized in part a.



**Figure 2.** This is a regression tree based on the a training split of 70% of the data of the uscrime dataset. The predictors used to determine this tree were calculated from the rpart model. This tree shows that NW was the primary factor used to differentiate between the two branches of the tree.

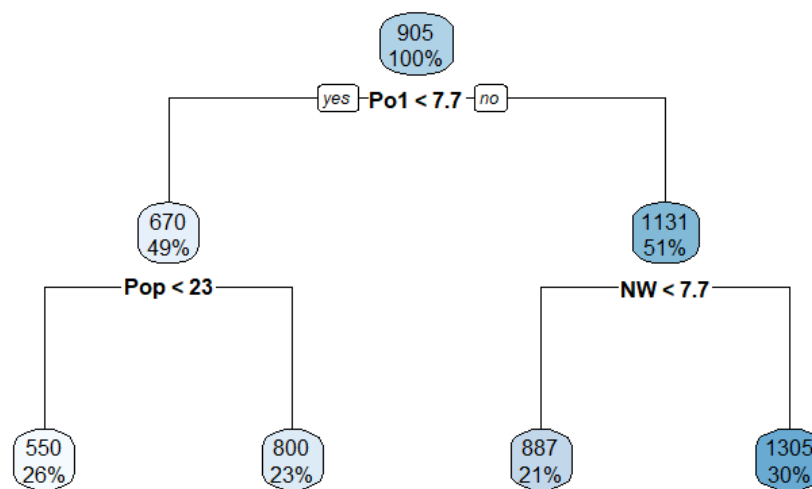
```

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

1) root 32 5209056.0 859.5312
 2) NW< 7.65 20 903983.0 673.4500
   4) Po1< 7 9 137746.2 516.5556 *
   5) Po1>=7 11 363431.6 801.8182 *
 3) NW>=7.65 12 2458341.0 1169.6670 *
>
> rpart.plot(m1)

```

**Figure 2.1.** This figure details the decision tree in Figure 1 but with the determining factor, the number of observations, the Sum of Squares Error (SSE), and the predicted average Crime value in order following the place in the tree.



**Figure 3.** This is a regression tree based on the entirety of the uscrime dataset. The predictors used to determine this tree were calculated from the rpart model. This tree shows that Po1 was the primary factor used to differentiate between the two branches of the tree.

```

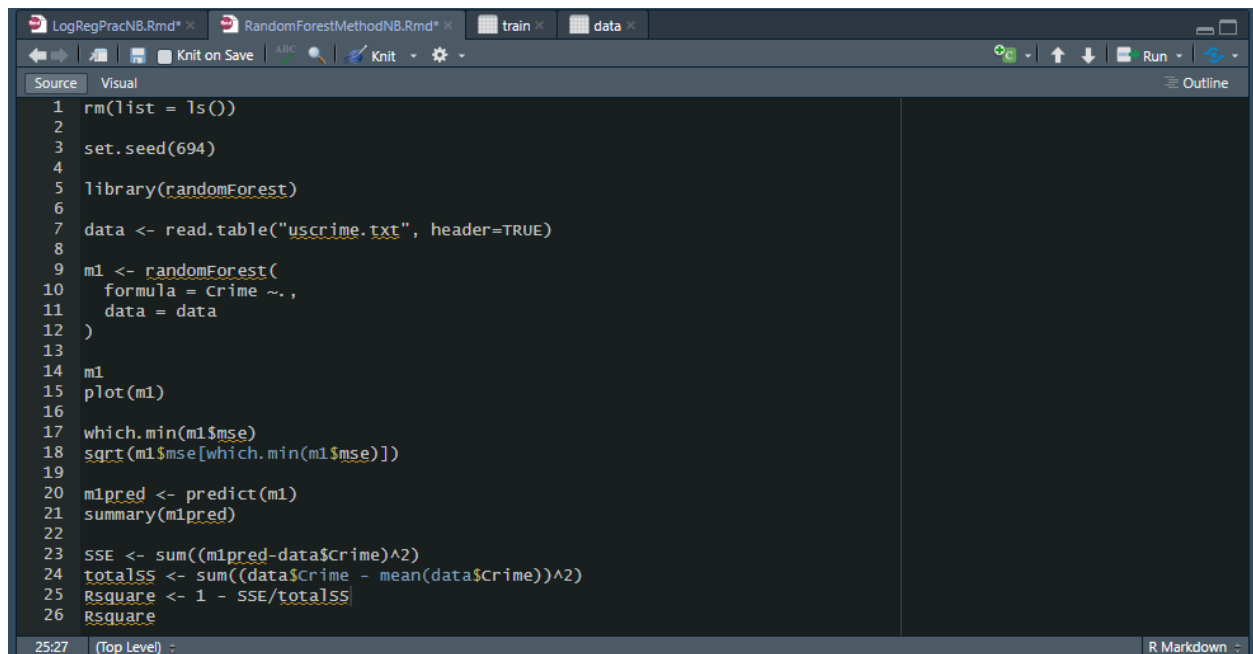
node), split, n, deviance, yval
  * denotes terminal node

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 *

>
> rpart.plot(m1)

```

**Figure 3.1.** This figure details the decision tree in Figure 3 but with the determining factor, the number of observations, the Sum of Squares Error (SSE), and the predicted average Crime value in order following the place in the tree.

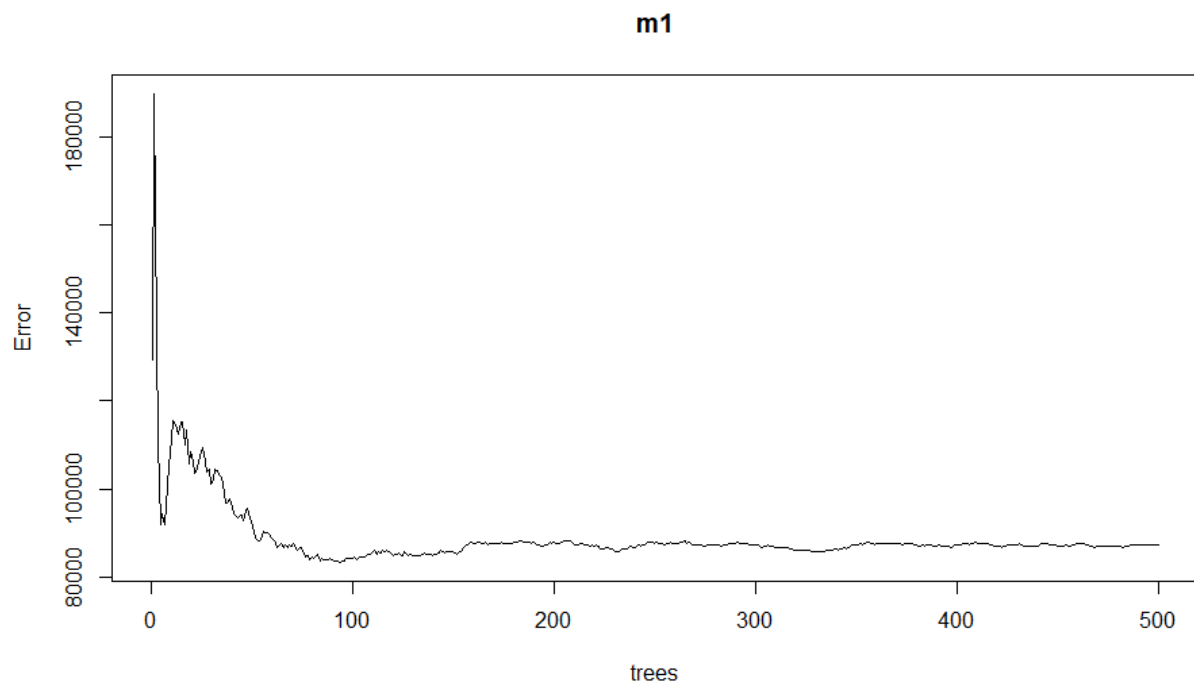


```

1 rm(list = ls())
2
3 set.seed(694)
4
5 library(randomForest)
6
7 data <- read.table("uscrime.txt", header=TRUE)
8
9 m1 <- randomForest(
10   formula = Crime ~.,
11   data = data
12 )
13
14 m1
15 plot(m1)
16
17 which.min(m1$mse)
18 sqrt(m1$mse[which.min(m1$mse)])
19
20 m1pred <- predict(m1)
21 summary(m1pred)
22
23 SSE <- sum((m1pred-data$Crime)^2)
24 totalSS <- sum((data$Crime - mean(data$Crime))^2)
25 Rsquare <- 1 - SSE/totalSS
26 Rsquare

```

**Figure 4.** This is the code used to generate the following figures based on the random forest methodology.



**Figure 5.** This graph is a depiction of the mean square error generated per number of trees in the random forest method. As will be shown in the following figure, the minimum value for this is when the number of trees equals 94.

```

Console Background Jobs
R 4.4.1 - C:/Users/Toshan/Desktop/OmsaGT/ISYE6501/LogRegressionPrac/
> m1

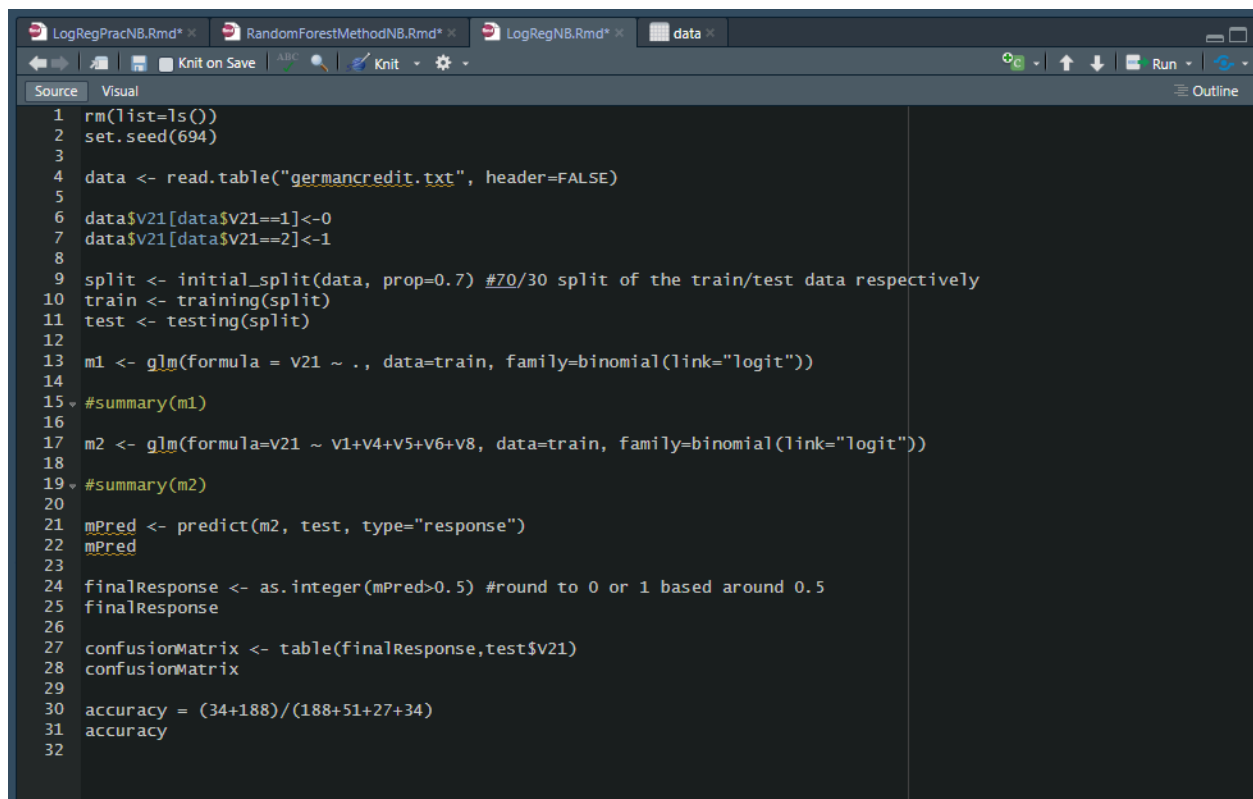
call:
 randomForest(formula = Crime ~ ., data = data)
      Type of random forest: regression
      Number of trees: 500
No. of variables tried at each split: 5

      Mean of squared residuals: 87308.38
      % Var explained: 40.36

> plot(m1)
>
> which.min(m1$mse)
[1] 94
> sqrt(m1$mse[which.min(m1$mse)])
[1] 288.8583
>
> m1pred <- predict(m1)
> summary(m1pred)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  526.5   731.9   896.0   906.0  1056.9  1346.0
>
> SSE <- sum((m1pred-data$Crime)^2)
> totalSS <- sum((data$Crime - mean(data$Crime))^2)
> Rsquare <- 1 - SSE/totalSS
> Rsquare
[1] 0.4036423
>

```

**Figure 6.** This figure is the numerical results of the code presented in Figure 4. The variables are aptly named to the values they describe, however the value of 288.8583 is the variance in error generated from the number of trees that had the least mean square error.



The image shows a screenshot of an RStudio interface. The top pane displays several open R Markdown files: 'LogRegPracNB.Rmd', 'RandomForestMethodNB.Rmd', 'LogRegNB.Rmd', and 'data'. The 'LogRegNB.Rmd' file is active, and the 'Source' pane shows the following R code:

```
1 rm(list=ls())
2 set.seed(694)
3
4 data <- read.table("germancredit.txt", header=FALSE)
5
6 data$v21[data$v21==1]<-0
7 data$v21[data$v21==2]<-1
8
9 split <- initial_split(data, prop=0.7) #70/30 split of the train/test data respectively
10 train <- training(split)
11 test <- testing(split)
12
13 m1 <- glm(formula = v21 ~ ., data=train, family=binomial(link="logit"))
14
15 #summary(m1)
16
17 m2 <- glm(formula=v21 ~ v1+v4+v5+v6+v8, data=train, family=binomial(link="logit"))
18
19 #summary(m2)
20
21 mPred <- predict(m2, test, type="response")
22 mPred
23
24 finalResponse <- as.integer(mPred>0.5) #round to 0 or 1 based around 0.5
25 finalResponse
26
27 confusionMatrix <- table(finalResponse,test$v21)
28 confusionMatrix
29
30 accuracy = (34+188)/(188+51+27+34)
31 accuracy
32
```

**Figure 7.** This is the code that was used to run the logistic regression of Question 10.3.

```

Console Background Jobs x
R 4.4.1 · C:/Users/Toshan/Desktop/OmsaGT/ISYE6501/LogRegressionPrac/ ↗
glm(formula = V21 ~ ., family = binomial(link = "logit"), data = train)

Coefficients:
Estimate Std. Error z value Pr(>|z|)
(Intercept) 5.001e-01 1.250e+00 0.400 0.689019
V1A12 -3.784e-01 2.671e-01 -1.417 0.156507
V1A13 -9.281e-01 4.778e-01 -1.943 0.052070 .
V1A14 -1.624e+00 2.784e-01 -5.834 5.40e-09 ***
V2 2.098e-02 1.086e-02 1.932 0.053338 .
V3A31 6.801e-01 6.638e-01 1.025 0.305550
V3A32 -3.687e-02 5.362e-01 -0.069 0.945174
V3A33 -3.227e-01 5.918e-01 -0.545 0.585529
V3A34 -8.611e-01 5.398e-01 -1.595 0.110673
V4A41 -1.494e+00 4.348e-01 -3.437 0.000588 ***
V4A410 -2.079e+00 9.734e-01 -2.136 0.032716 *
V4A42 -6.824e-01 3.132e-01 -2.179 0.029367 *
V4A43 -1.355e+00 3.141e-01 -4.313 1.61e-05 ***
V4A44 -5.442e-01 1.225e+00 -0.444 0.656983
V4A45 -5.685e-01 7.162e-01 -0.794 0.427342
V4A46 4.684e-04 4.813e-01 0.001 0.999223
V4A48 -1.416e+00 1.301e+00 -1.089 0.276355
V4A49 -8.911e-01 4.239e-01 -2.102 0.035514 *
V5 1.919e-04 5.419e-05 3.541 0.000398 ***
V6A62 -6.369e-01 3.703e-01 -1.720 0.085400 .
V6A63 -4.997e-01 5.069e-01 -0.986 0.324208
V6A64 -2.130e+00 7.449e-01 -2.859 0.004244 **
V6A65 -1.377e+00 3.246e-01 -4.241 2.23e-05 ***
V7A72 1.539e-01 5.069e-01 0.304 0.761405
V7A73 5.509e-02 4.894e-01 0.113 0.910377
V7A74 -5.098e-01 5.314e-01 -0.959 0.337385
V7A75 -1.749e-01 4.927e-01 -0.355 0.722535
V8 3.851e-01 1.109e-01 3.473 0.000515 ***
V9A92 -4.412e-01 4.704e-01 -0.938 0.348346
V9A93 -9.687e-01 4.617e-01 -2.098 0.035885 *
V9A94 -3.825e-01 5.672e-01 -0.674 0.500067
V10A102 2.809e-01 5.034e-01 0.558 0.576890
V10A103 -1.351e+00 5.107e-01 -2.645 0.008161 **
V11 2.629e-02 1.084e-01 0.243 0.808333
V12A122 2.348e-01 3.185e-01 0.737 0.461080
V12A123 5.148e-01 2.889e-01 1.782 0.074733 .
V12A124 1.032e+00 5.096e-01 2.025 0.042878 *
V13 -1.991e-02 1.149e-02 -1.732 0.083206 .
V14A142 -3.527e-01 5.064e-01 -0.696 0.486144
V14A143 -8.058e-01 2.907e-01 -2.772 0.005565 **
V15A152 -3.058e-01 2.878e-01 -1.063 0.287926
V15A153 -1.140e+00 5.832e-01 -1.956 0.050512 .
V16 2.782e-01 2.343e-01 1.187 0.235120
V17A172 8.289e-02 7.748e-01 0.107 0.914803
V17A173 -1.188e-01 7.384e-01 -0.161 0.872229
V17A174 -3.549e-01 7.469e-01 -0.475 0.634664
V18 2.543e-01 2.981e-01 0.853 0.393497
V19A192 -2.439e-01 2.500e-01 -0.976 0.329199
V20A202 -1.393e+00 7.482e-01 -1.861 0.062712 .
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

**Figure 8.** This is the output of m1, detailing which variables were deemed significant. The columns that contained a code of “\*\*\*” were kept and used in the next model so that overfitting would be avoided.



```

> summary(m2)

Call:
glm(formula = v21 ~ v1 + v4 + v5 + v6 + v8, family = binomial(link = "logit"),
    data = train)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.189e+00  4.011e-01  -2.963 0.003042 **
v1A12        -3.410e-01  2.331e-01  -1.463 0.143479
v1A13        -8.603e-01  4.283e-01  -2.009 0.044582 *
v1A14        -1.675e+00  2.504e-01  -6.690 2.24e-11 ***
v4A41        -1.320e+00  3.922e-01  -3.367 0.000760 ***
v4A410       -1.750e+00  8.204e-01  -2.133 0.032950 *
v4A42        -3.552e-01  2.785e-01  -1.276 0.202124
v4A43        -1.001e+00  2.734e-01  -3.662 0.000251 ***
v4A44         5.597e-01  1.015e+00   0.551 0.581428
v4A45        -2.319e-01  6.469e-01  -0.358 0.720017
v4A46         3.025e-01  4.289e-01   0.705 0.480747
v4A48        -3.884e-01  1.243e+00  -0.313 0.754653
v4A49        -2.409e-01  3.519e-01  -0.685 0.493588
v5            2.154e-04  3.934e-05   5.474 4.40e-08 ***
v6A62        -2.299e-01  3.313e-01  -0.694 0.487752
v6A63        -2.386e-01  4.654e-01  -0.513 0.608181
v6A64        -1.542e+00  6.444e-01  -2.393 0.016724 *
v6A65        -1.169e+00  2.926e-01  -3.995 6.46e-05 ***
v8            3.482e-01  9.417e-02   3.697 0.000218 ***
---
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: 693.77  on 681  degrees of freedom
AIC: 731.77

Number of Fisher Scoring iterations: 5

```

**Figure 9.** This is the second and final model from the training data.

```

      271      272      273      274      275      276      277      278      279      280
0.36432626 0.03262238 0.44132149 0.07003505 0.03412977 0.28890687 0.04468634 0.02109627 0.03068283 0.50863548
      281      282      283      284      285      286      287      288      289      290
0.29922198 0.31466617 0.17423392 0.06923250 0.13536439 0.58211302 0.06793915 0.51175621 0.43258981 0.34995008
      291      292      293      294      295      296      297      298      299      300
0.05762330 0.50194466 0.35811100 0.13438810 0.12205165 0.31195337 0.21396791 0.11045201 0.10663175 0.25920788
>
> finalResponse <- as.integer(mPred>0.5) #round to 0 or 1 based around 0.5
> finalResponse
 [1] 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 1 1 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
[55] 0 1 0 1 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0
[109] 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 1 1 1 0 1
[163] 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0
[217] 1 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0
[271] 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0
>
> confusionMatrix <- table(finalResponse,test$V21)
> confusionMatrix

finalResponse    0    1
              0 188   51
              1   27   34
>
> accuracy = (34+188)/(188+51+27+34)
> accuracy
[1] 0.74
> |

```

**Figure 10.** This figure shows the accuracy generated from the final logistic regression model, as well as the rounded final response based on a threshold of 0.5.