My_Homework.r

nhirata

2020-02-26

```
#10.1a
# There are only 47 points so splitting the data would be very difficult. I started by creating
 a Regression Tree and then saw that the variables actually used in the tree construction were o
nly 4 attributes.
# Then i tried pruning the tree through cross validation to see if the performance would increas
e but the best fit was to use all the nodes but I provided a sample of what the pruning would lo
ok like.
# Going forward with the unpruned tree, I calculated yhat and SSres to get my R^2. However, the
 model needs to be cross-validated because results are mostly inflated on training data due to f
itting real and random effects.



rm(list = ls())# Start Fresh
library(DAAG)
```

```
## Loading required package: lattice
```

```
library(tree)
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(1)

data <- read.table("C:/Users/nhirata/Desktop/Georgia Tech/OneDrive - Georgia Institute of Techno
logy/Georgia Tech/ISYE_6501/Week_7/data 10.1/uscrime.txt", header=TRUE, stringsAsFactors = FALSE
)

head(data)
```

```
##      M So   Ed  Po1  Po2    LF   M.F Pop   NW    U1  U2 Wealth Ineq     Prob
## 1 15.1  1  9.1  5.8  5.6 0.510  95.0  33 30.1 0.108 4.1   3940 26.1 0.084602
## 2 14.3  0 11.3 10.3  9.5 0.583 101.2  13 10.2 0.096 3.6   5570 19.4 0.029599
## 3 14.2  1  8.9  4.5  4.4 0.533  96.9  18 21.9 0.094 3.3   3180 25.0 0.083401
## 4 13.6  0 12.1 14.9 14.1 0.577  99.4 157  8.0 0.102 3.9   6730 16.7 0.015801
## 5 14.1  0 12.1 10.9 10.1 0.591  98.5  18  3.0 0.091 2.0   5780 17.4 0.041399
## 6 12.1  0 11.0 11.8 11.5 0.547  96.4  25  4.4 0.084 2.9   6890 12.6 0.034201
##      Time Crime
## 1 26.2011   791
## 2 25.2999  1635
## 3 24.3006   578
## 4 29.9012  1969
## 5 21.2998  1234
## 6 20.9995   682
```

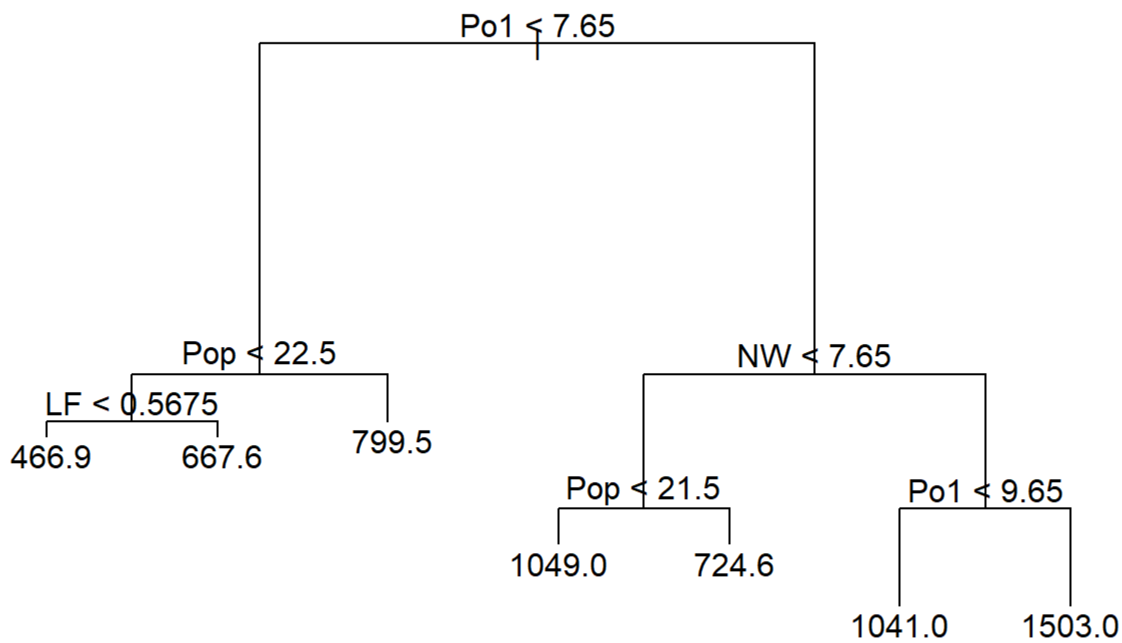
```
#Start with creating a Regression Tree.

data_tree <- tree(Crime~., data = data)
summary(data_tree)
```

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

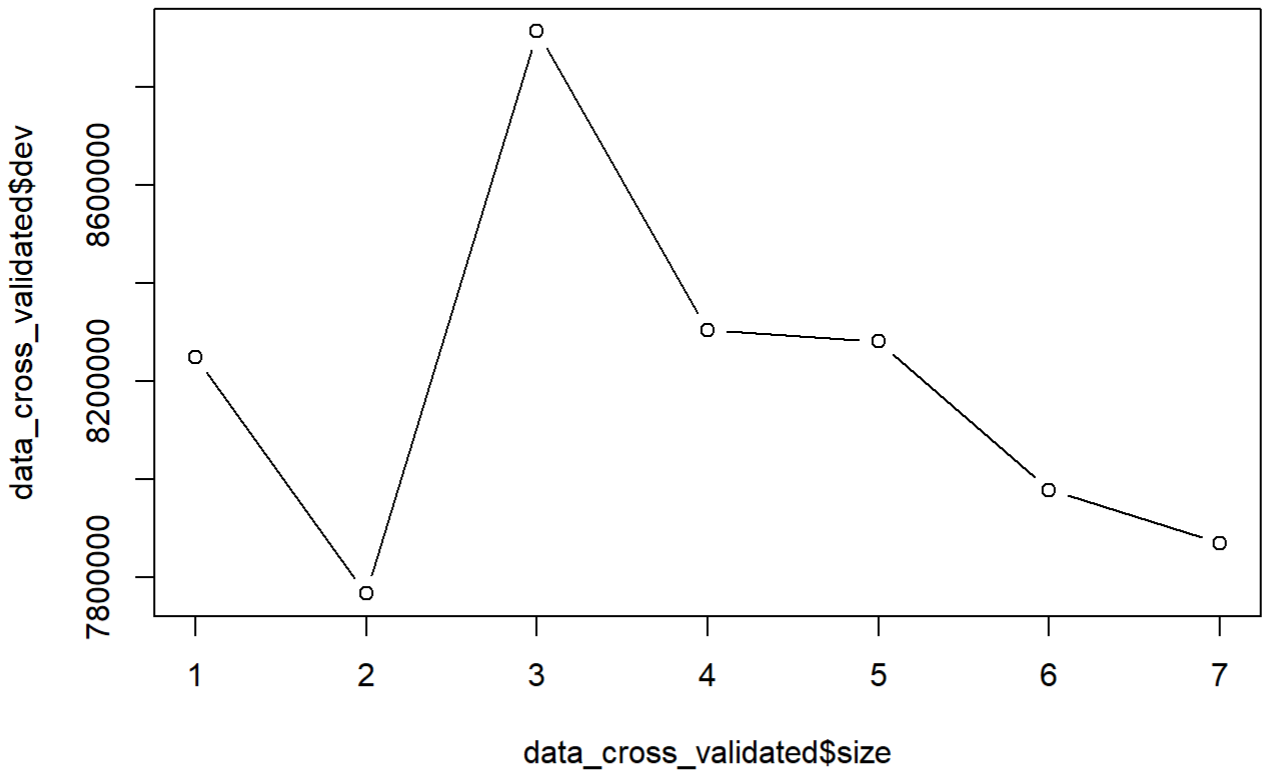
```
# Variables actually used in tree construction:"Po1" "Pop" "LF"  "NW"

data_tree$frame # The view of the tree splitting
```

```
##       var  n        dev      yval splits.cutleft splits.cutright
## 1     Po1 47 6880927.66  905.0851          <7.65           >7.65
## 2     Pop 23  779243.48  669.6087          <22.5           >22.5
## 4      LF 12  243811.00  550.5000        <0.5675         >0.5675
## 8  <leaf>  7   48518.86  466.8571
## 9  <leaf>  5   77757.20  667.6000
## 5  <leaf> 11  179470.73  799.5455
## 3      NW 24 3604162.50 1130.7500          <7.65           >7.65
## 6     Pop 10  557574.90  886.9000          <21.5           >21.5
## 12 <leaf>  5  146390.80 1049.2000
## 13 <leaf>  5  147771.20  724.6000
## 7     Po1 14 2027224.93 1304.9286          <9.65           >9.65
## 14 <leaf>  6  170828.00 1041.0000
## 15 <leaf>  8 1124984.88 1502.8750
```

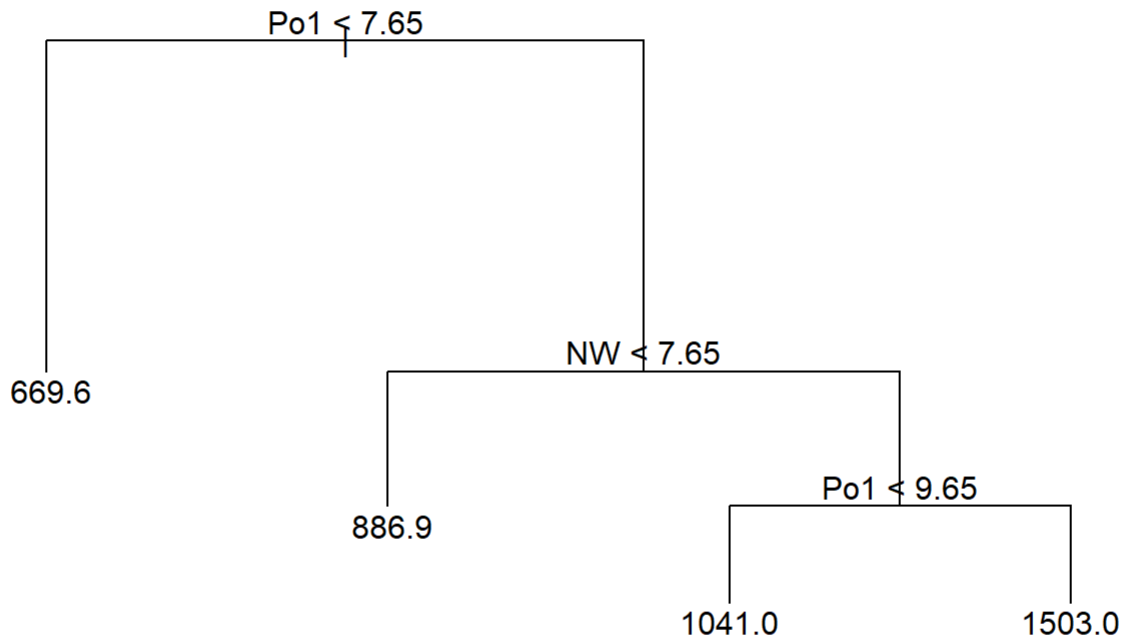
```
#Visualization
plot(data_tree)
text(data_tree)
```



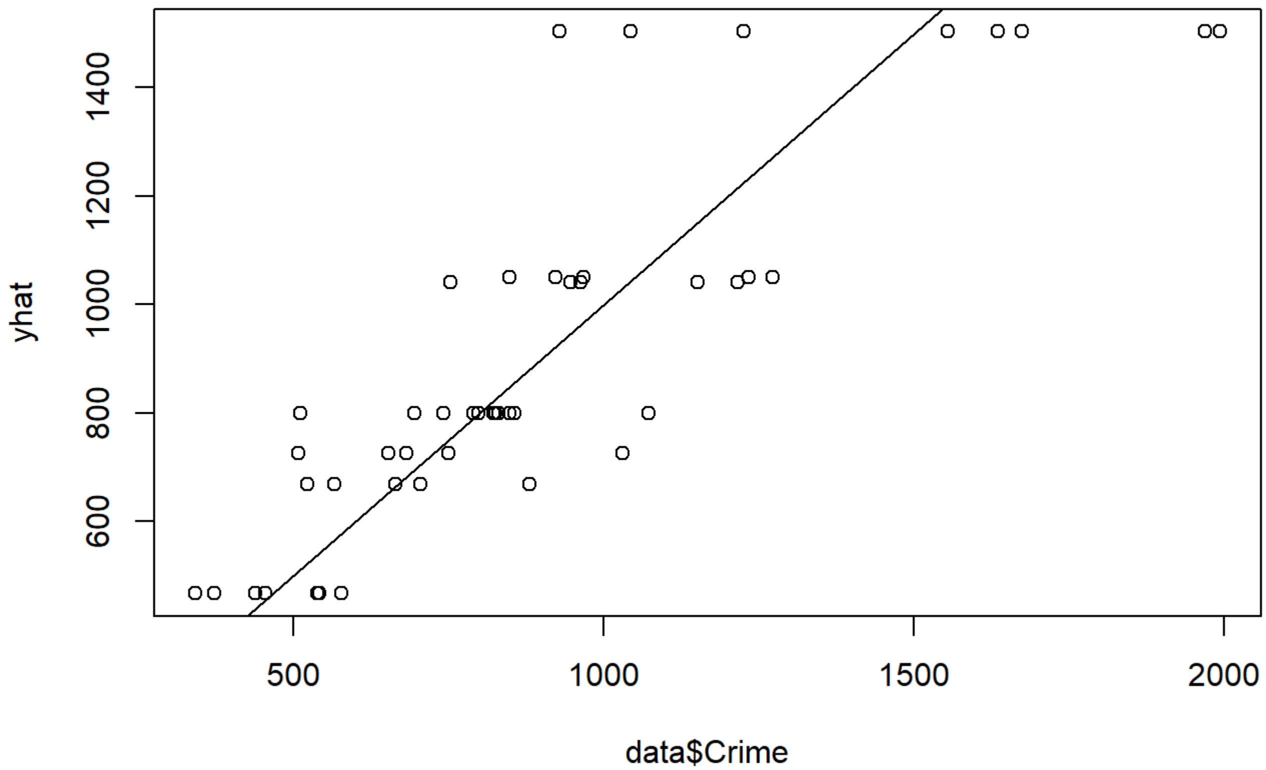

```
# Lets see if tree pruning by using cross-validation will increase performance by analyzing term
inal nodes.
# Deviance is a quality-of-fit statistic.
# The x-axis represents the number of terminal nodes.

data_cross_validated <- cv.tree(data_tree)
plot(data_cross_validated$size, data_cross_validated$dev, type = "b")
```



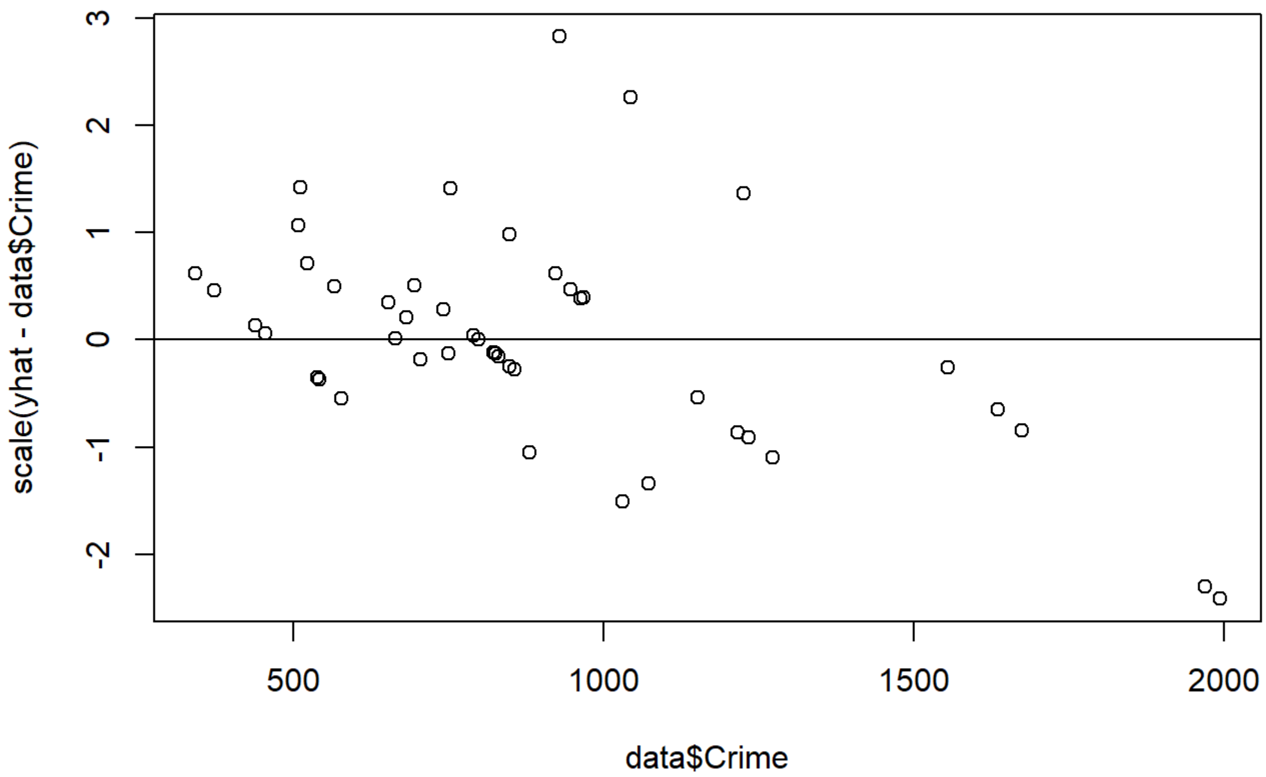

```
# This plot suggests that we get the best fit using all of the terminal nodes in the tree that w
e already plotted.

# Limit the # of tree nodes to prune the regression tree.
nodes <- 4
tree_pruned <- prune.tree(data_tree, best = nodes)

plot(tree_pruned) # Plot tree
text(tree_pruned)
```




```
# Advance with the unpruned tree
# SSres preparation

yhat <- predict(data_tree)
SSres <- sum((yhat-data$Crime)^2)

plot(data$Crime, yhat) #Predicted vs. Actual
abline(0,1)
```




```
plot(data$Crime, scale(yhat - data$Crime)) #Residuals
abline(0,0)
```




```
# Calculate R2 for the training

SStot <- sum((data$Crime - mean(data$Crime))^2)
R2 <- 1 - SSres/SStot
R2
```

```
## [1] 0.7244962
```

```
# However, we know that the model needs to be cross-validated because results are mostly inflate
d on training data due to fitting real and random effects.

# Sum of squared errors for each tree size
prune.tree(data_tree)$size
```

```
## [1] 7 6 5 4 3 2 1
```

```
prune.tree(data_tree)$dev
```

```
## [1] 1895722 2013257 2276670 2632631 3364043 4383406 6880928
```

```
#Sum of squared errors in cross validation
data_cross_validated <- cv.tree(data_tree)
data_cross_validated$size
```

```
## [1] 7 6 5 4 3 2 1
```

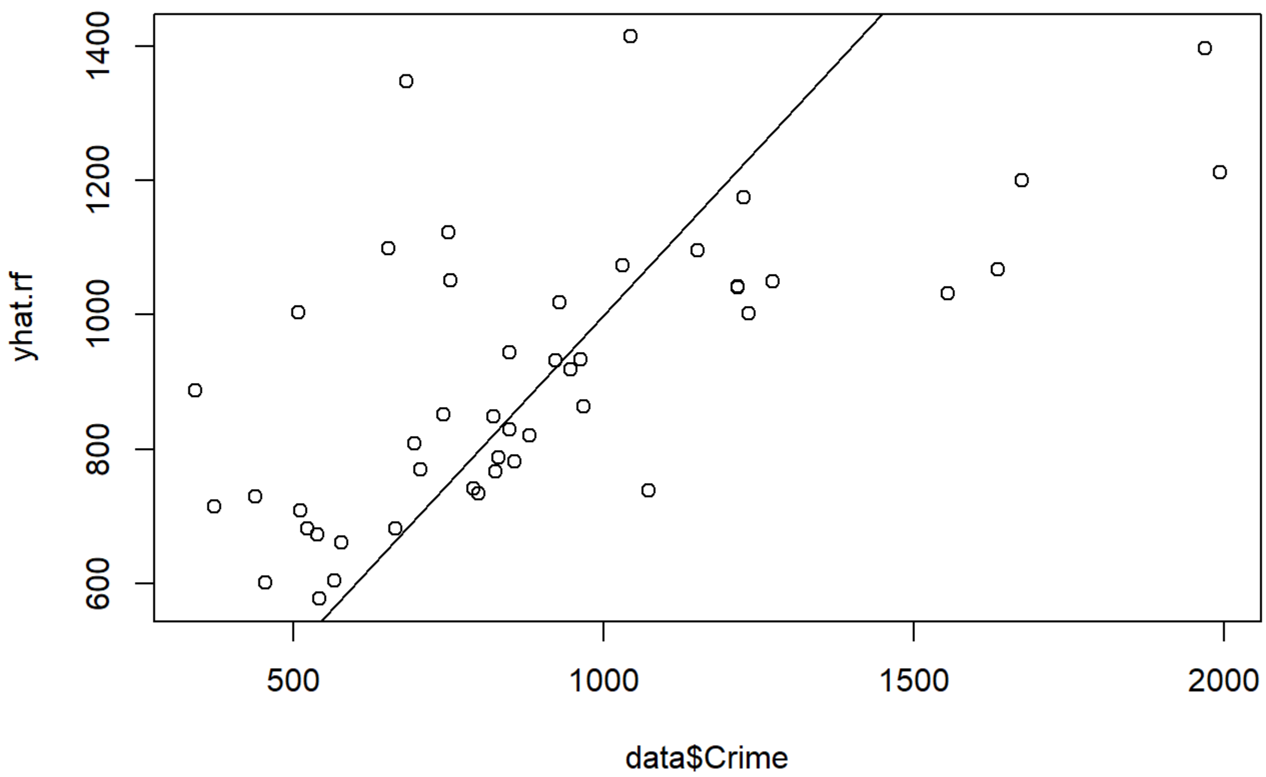
```
data_cross_validated$dev
```

```
## [1] 8114496 8384511 8369027 8554913 8458678 8396658 9298822
```

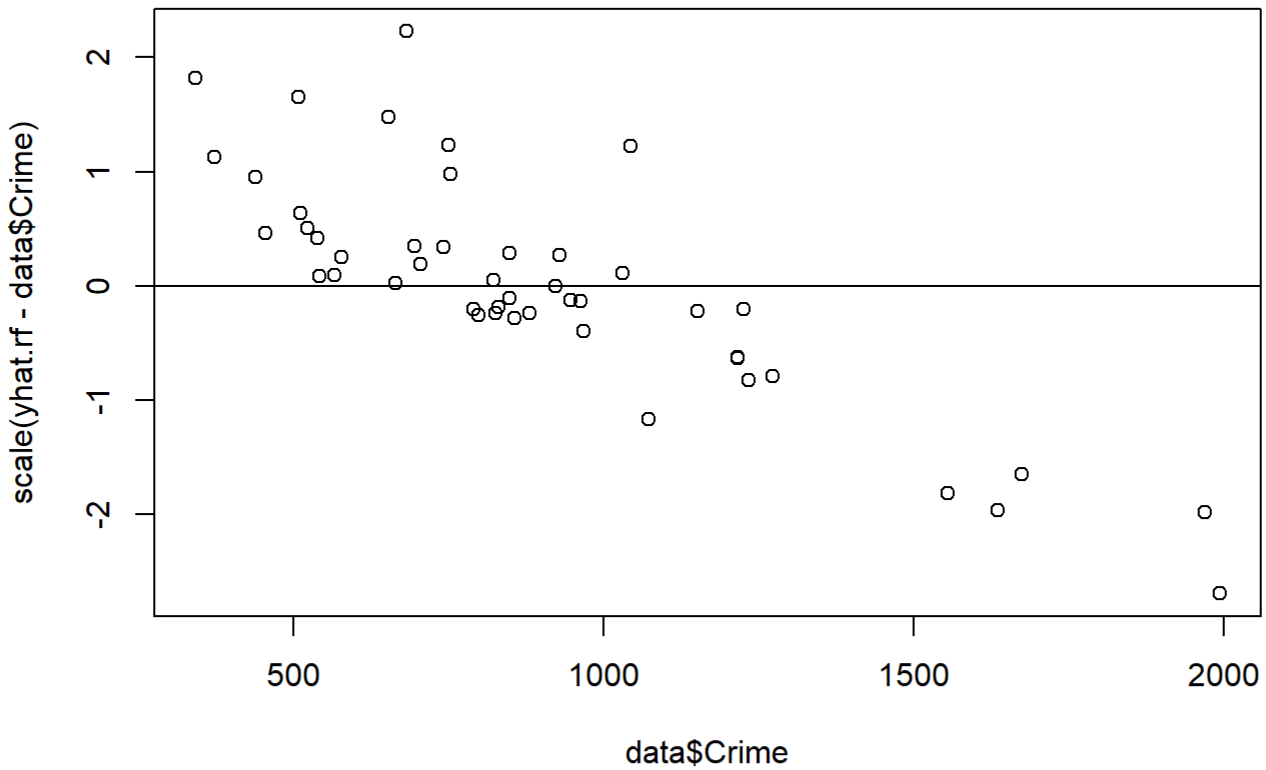
```
# The errors become a lot larger than before which means it's overfitted.
# Due to not enough data points,this was the best I could do

###########10.1b################
# I first considered the number of predictions to make and then ran the randomForest function. T
hen I calculated the SSres of the model and plotted the actual vs. predicted and residual values
to analyze my data.
# Then I calculated SStot to get my R^2 on the Training data and then compared it to R^2 through
cross validation. The R^2 came out a lot more stable and performed better than the other regress
ion tree model.



# Consider the number of predictors and run the randomForest model
number_of_predictions <- 4
randomForest_data <- randomForest(Crime~., data = data, mtry = number_of_predictions, importance
= TRUE)
randomForest_data
```

```
##
## Call:
##  randomForest(formula = Crime ~ ., data = data, mtry = number_of_predictions,      importance
= TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 84810.15
##                     % Var explained: 42.07
```

```
# Calculate SSres of the random forest model
yhat.rf <- predict(randomForest_data)
SSres <- sum((yhat.rf-data$Crime)^2)

# Plot of actual vs. predicted crime values
plot(data$Crime, yhat.rf)
abline(0,1)
```




```
# Plot residuals
plot(data$Crime, scale(yhat.rf - data$Crime))
abline(0,0)
```




```
# R^2 on Training Data
SStot <- sum((data$Crime - mean(data$Crime))^2)
R2 <- 1 - SSres/SStot
R2
```

```
## [1] 0.4207064
```

```
# R^2 on Cross Validation
SSE <- 0
for (i in 1:nrow(data)) {
  model <- randomForest(Crime~., data = data[-i,], mtry = number_of_predictions, importance = TR
UE)
  SSE = SSE + (predict(model,newdata=data[i,]) - data[i,16])^2
}
1 - SSE/SStot # Better than the regression tree model and, removed a lot of overfitting just lik
e it's supposed to.
```

```
##         1
## 0.4283111
```

```
###########10.2#################
#It would be a great time to use Logistic Regression on whether a political candidate for presid
ency wins an election (current events). The response/outcome would be (0,1) for win or lose.
#Predictors can be the amount of money spent on the campaign, time spent campaigning, type of po
litical party, education level, and gender.


###########10.3 PART 1#################

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

data<-read.table("C:/Users/nhirata/Desktop/Georgia Tech/OneDrive - Georgia Institute of Technolo
gy/Georgia Tech/ISYE_6501/Week_7/data 10.3/germancredit.txt", sep = " ")

head(data)
```

```
##    V1 V2  V3  V4   V5  V6  V7 V8  V9  V10 V11  V12 V13  V14  V15 V16  V17 V18
## 1 A11  6 A34 A43 1169 A65 A75  4 A93 A101   4 A121  67 A143 A152   2 A173   1
## 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
## 5 A11 24 A33 A40 4870 A61 A73  3 A93 A101   4 A124  53 A143 A153   2 A173   2
## 6 A14 36 A32 A46 9055 A65 A73  2 A93 A101   4 A124  35 A143 A153   1 A172   2
##    V19  V20 V21
## 1 A192 A201   1
## 2 A191 A201   2
## 3 A191 A201   1
## 4 A191 A201   1
## 5 A191 A201   2
## 6 A192 A201   1
```

```
# Transform response variable to 0's and 1's

data$V21[data$V21==1]<-0
data$V21[data$V21==2]<-1

# Separate 70% training and 30% test/validation data
m <- nrow(data)
train <- sample(1:m, size = round(m*0.7), replace = FALSE)
train_data <- data[train,]
validation_data <- data[-train,]

# Logistic regression model: Use all the available variables

reg = glm(V21 ~.,family=binomial(link = "logit"),data=train_data)
summary(reg)
```

```
##
## Call:
## glm(formula = V21 ~ ., family = binomial(link = "logit"), data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4438  -0.6861  -0.3608   0.6750   2.4540
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.823e-01  1.332e+00   0.287 0.774162
## V1A12       -5.201e-01  2.681e-01  -1.940 0.052408 .
## V1A13       -1.150e+00  4.473e-01  -2.570 0.010173 *
## V1A14       -1.675e+00  2.750e-01  -6.091 1.12e-09 ***
## V2           2.570e-02  1.159e-02   2.217 0.026647 *
## V3A31        8.440e-02  6.580e-01   0.128 0.897943
## V3A32       -8.078e-01  4.996e-01  -1.617 0.105907
## V3A33       -7.683e-01  5.372e-01  -1.430 0.152634
## V3A34       -1.446e+00  5.127e-01  -2.821 0.004784 **
## V4A41       -1.513e+00  4.479e-01  -3.379 0.000728 ***
## V4A410      -2.412e+00  1.160e+00  -2.080 0.037543 *
## V4A42       -5.496e-01  3.195e-01  -1.720 0.085354 .
## V4A43       -9.142e-01  3.024e-01  -3.023 0.002503 **
## V4A44       -4.163e-01  9.455e-01  -0.440 0.659751
## V4A45       -1.562e-01  6.742e-01  -0.232 0.816732
## V4A46       -2.569e-01  5.085e-01  -0.505 0.613382
## V4A48       -1.531e+01  4.556e+02  -0.034 0.973202
## V4A49       -5.397e-01  4.017e-01  -1.344 0.179086
## V5           1.076e-04  5.600e-05   1.922 0.054633 .
## V6A62       -3.474e-01  3.579e-01  -0.971 0.331777
## V6A63       -2.440e-01  4.761e-01  -0.513 0.608232
## V6A64       -1.379e+00  6.535e-01  -2.110 0.034823 *
## V6A65       -8.106e-01  3.223e-01  -2.515 0.011910 *
## V7A72       -1.814e-01  5.243e-01  -0.346 0.729300
## V7A73       -5.253e-01  5.001e-01  -1.050 0.293529
## V7A74       -1.129e+00  5.455e-01  -2.070 0.038431 *
## V7A75       -5.927e-01  5.052e-01  -1.173 0.240705
## V8           3.523e-01  1.094e-01   3.219 0.001284 **
## V9A92        4.849e-02  4.760e-01   0.102 0.918863
## V9A93       -4.446e-01  4.691e-01  -0.948 0.343279
## V9A94       -4.288e-01  5.837e-01  -0.735 0.462524
## V10A102      3.052e-01  5.338e-01   0.572 0.567472
## V10A103     -3.086e-01  5.237e-01  -0.589 0.555669
## V11         -1.080e-01  1.073e-01  -1.007 0.314147
## V12A122      2.219e-01  3.161e-01   0.702 0.482767
## V12A123      3.274e-01  2.922e-01   1.120 0.262504
## V12A124      1.156e+00  5.656e-01   2.044 0.040944 *
## V13         -2.257e-02  1.140e-02  -1.980 0.047667 *
## V14A142     -5.214e-01  4.925e-01  -1.059 0.289757
## V14A143     -7.780e-01  2.848e-01  -2.732 0.006299 **
## V15A152     -6.323e-01  2.870e-01  -2.203 0.027579 *
## V15A153     -6.674e-01  6.202e-01  -1.076 0.281931
## V16          2.866e-01  2.236e-01   1.282 0.199939
```

```
## V17A172      1.565e+00  8.891e-01   1.760 0.078442 .
## V17A173      1.564e+00  8.582e-01   1.823 0.068370 .
## V17A174      1.400e+00  8.772e-01   1.596 0.110563
## V18          1.645e-01  3.004e-01   0.548 0.583871
## V19A192     -3.319e-01  2.413e-01  -1.376 0.168942
## V20A202     -2.137e+00  8.573e-01  -2.493 0.012665 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 851.79  on 699  degrees of freedom
## Residual deviance: 613.21  on 651  degrees of freedom
## AIC: 711.21
##
## Number of Fisher Scoring iterations: 14
```

```
# Now use significant variables from the first.
reg = glm(V21 ~ V1+V2+V3+V4+V6+V7+V8+V12+V13+V14+V15+V20,family=binomial(link = "logit"),data=tr
ain_data)
summary(reg)
```

```
##
## Call:
## glm(formula = V21 ~ V1 + V2 + V3 + V4 + V6 + V7 + V8 + V12 +
##     V13 + V14 + V15 + V20, family = binomial(link = "logit"),
##     data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1343  -0.7151  -0.3829   0.6860   2.4613
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.954704   0.891687   2.192 0.028369 *
## V1A12        -0.576472   0.261484  -2.205 0.027481 *
## V1A13        -1.246843   0.431631  -2.889 0.003869 **
## V1A14        -1.633312   0.268529  -6.082 1.18e-09 ***
## V2            0.036245   0.009018   4.019 5.83e-05 ***
## V3A31        -0.077212   0.626544  -0.123 0.901921
## V3A32        -0.971152   0.467517  -2.077 0.037778 *
## V3A33        -0.743337   0.523613  -1.420 0.155715
## V3A34        -1.447406   0.498458  -2.904 0.003687 **
## V4A41        -1.375418   0.415248  -3.312 0.000925 ***
## V4A410       -2.141135   1.005226  -2.130 0.033171 *
## V4A42        -0.448810   0.306760  -1.463 0.143450
## V4A43        -0.939346   0.291483  -3.223 0.001270 **
## V4A44        -0.492708   0.950608  -0.518 0.604243
## V4A45        -0.238889   0.676434  -0.353 0.723969
## V4A46        -0.268251   0.499637  -0.537 0.591342
## V4A48       -15.257528 451.906122  -0.034 0.973066
## V4A49        -0.570040   0.388370  -1.468 0.142165
## V6A62        -0.307611   0.343036  -0.897 0.369862
## V6A63        -0.484692   0.466898  -1.038 0.299218
## V6A64        -1.212443   0.622420  -1.948 0.051421 .
## V6A65        -0.810257   0.313011  -2.589 0.009637 **
## V7A72         0.199964   0.461824   0.433 0.665025
## V7A73        -0.179579   0.428502  -0.419 0.675153
## V7A74        -0.874730   0.481305  -1.817 0.069154 .
## V7A75        -0.318585   0.442407  -0.720 0.471453
## V8            0.235399   0.095022   2.477 0.013237 *
## V12A122       0.282784   0.305057   0.927 0.353932
## V12A123       0.433363   0.275990   1.570 0.116365
## V12A124       1.052152   0.533417   1.972 0.048555 *
## V13          -0.022929   0.010929  -2.098 0.035904 *
## V14A142      -0.485528   0.484466  -1.002 0.316251
## V14A143      -0.716748   0.278048  -2.578 0.009944 **
## V15A152      -0.592894   0.264661  -2.240 0.025078 *
## V15A153      -0.512743   0.587424  -0.873 0.382735
## V20A202      -2.057069   0.840733  -2.447 0.014415 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 851.79  on 699  degrees of freedom
## Residual deviance: 628.99  on 664  degrees of freedom
## AIC: 700.99
##
## Number of Fisher Scoring iterations: 14
```

```
# Now use significant variables from the second.
reg = glm(V21 ~ V1+V2+V3+V4+V6+V8+V12+V13+V14+V15+V20,family=binomial(link = "logit"),data=train
_data)
summary(reg)
```

```
##
## Call:
## glm(formula = V21 ~ V1 + V2 + V3 + V4 + V6 + V8 + V12 + V13 +
##     V14 + V15 + V20, family = binomial(link = "logit"), data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1140  -0.7309  -0.3992   0.7262   2.6620
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.971739   0.746245   2.642 0.008237 **
## V1A12        -0.492608   0.254998  -1.932 0.053382 .
## V1A13        -1.156215   0.424349  -2.725 0.006436 **
## V1A14        -1.605108   0.265374  -6.048 1.46e-09 ***
## V2            0.033550   0.008783   3.820 0.000134 ***
## V3A31        -0.182160   0.623567  -0.292 0.770190
## V3A32        -1.052691   0.461930  -2.279 0.022673 *
## V3A33        -0.801449   0.518845  -1.545 0.122424
## V3A34        -1.570962   0.490694  -3.202 0.001367 **
## V4A41        -1.417883   0.414529  -3.420 0.000625 ***
## V4A410       -1.978036   0.980023  -2.018 0.043554 *
## V4A42        -0.408540   0.300536  -1.359 0.174029
## V4A43        -0.932034   0.287107  -3.246 0.001169 **
## V4A44        -0.394596   0.910382  -0.433 0.664695
## V4A45        -0.146272   0.659895  -0.222 0.824579
## V4A46        -0.211623   0.494183  -0.428 0.668485
## V4A48       -15.306333 458.685224  -0.033 0.973380
## V4A49        -0.610577   0.383411  -1.592 0.111275
## V6A62        -0.369546   0.338713  -1.091 0.275260
## V6A63        -0.494063   0.457275  -1.080 0.279942
## V6A64        -1.207293   0.604626  -1.997 0.045851 *
## V6A65        -0.879751   0.308183  -2.855 0.004309 **
## V8            0.223822   0.094210   2.376 0.017512 *
## V12A122       0.302086   0.298543   1.012 0.311602
## V12A123       0.401634   0.271930   1.477 0.139681
## V12A124       1.083056   0.526907   2.055 0.039831 *
## V13          -0.025963   0.010028  -2.589 0.009622 **
## V14A142      -0.400231   0.474423  -0.844 0.398884
## V14A143      -0.714374   0.275354  -2.594 0.009476 **
## V15A152      -0.589884   0.261522  -2.256 0.024097 *
## V15A153      -0.523675   0.584766  -0.896 0.370505
## V20A202      -2.005458   0.831808  -2.411 0.015911 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 851.79  on 699  degrees of freedom
## Residual deviance: 638.92  on 668  degrees of freedom
## AIC: 702.92
##
## Number of Fisher Scoring iterations: 14
```

```
# Bucket between 0 and 1 manually
train_data$V1A12[train_data$V1 == "A12"] <- 1
train_data$V1A12[train_data$V1 != "A12"] <- 0

train_data$V1A13[train_data$V1 == "A13"] <- 1
train_data$V1A13[train_data$V1 != "A13"] <- 0

train_data$V1A14[train_data$V1 == "A14"] <- 1
train_data$V1A14[train_data$V1 != "A14"] <- 0

train_data$V3A32[train_data$V3 == "A32"] <- 1
train_data$V3A32[train_data$V3 != "A32"] <- 0

train_data$V3A34[train_data$V3 == "A34"] <- 1
train_data$V3A34[train_data$V3 != "A34"] <- 0

train_data$V4A41[train_data$V4 == "A41"] <- 1
train_data$V4A41[train_data$V4 != "A41"] <- 0

train_data$V4A410[train_data$V4 == "A410"] <- 1
train_data$V4A410[train_data$V4 != "A410"] <- 0

train_data$V4A43[train_data$V4 == "A43"] <- 1
train_data$V4A43[train_data$V4 != "A43"] <- 0

train_data$V6A65[train_data$V6 == "A65"] <- 1
train_data$V6A65[train_data$V6 != "A65"] <- 0

train_data$V12A124[train_data$V12 == "A124"] <- 1
train_data$V12A124[train_data$V12 != "A124"] <- 0

train_data$V14A143[train_data$V14 == "A143"] <- 1
train_data$V14A143[train_data$V14 != "A143"] <- 0

train_data$V15A152[train_data$V15 == "A152"] <- 1
train_data$V15A152[train_data$V15 != "A152"] <- 0

train_data$V20A202[train_data$V20 == "A202"] <- 1
train_data$V20A202[train_data$V20 != "A202"] <- 0

# Now use significant variables from the third

reg = glm(V21 ~ V1A12 + V1A13 + V1A14 + V2 + V3A32 + V3A34 + V4A41 + V4A410 + V4A43 + V5 + V6A65
+ V8 +V12A124+ V14A143+V15A152+V20A202,family=binomial(link = "logit"),data=train_data)
summary(reg)
```

```
##
## Call:
## glm(formula = V21 ~ V1A12 + V1A13 + V1A14 + V2 + V3A32 + V3A34 +
##     V4A41 + V4A410 + V4A43 + V5 + V6A65 + V8 + V12A124 + V14A143 +
##     V15A152 + V20A202, family = binomial(link = "logit"), data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1783  -0.7529  -0.4190   0.7930   2.5906
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.902e-02  4.817e-01   0.102 0.918948
## V1A12       -5.103e-01  2.380e-01  -2.144 0.032002 *
## V1A13       -1.112e+00  4.097e-01  -2.714 0.006646 **
## V1A14       -1.739e+00  2.519e-01  -6.903  5.1e-12 ***
## V2           2.479e-02  1.061e-02   2.336 0.019497 *
## V3A32       -3.761e-01  2.465e-01  -1.526 0.127053
## V3A34       -9.854e-01  2.927e-01  -3.367 0.000760 ***
## V4A41       -1.312e+00  3.954e-01  -3.318 0.000906 ***
## V4A410      -2.304e+00  9.973e-01  -2.310 0.020880 *
## V4A43       -6.163e-01  2.305e-01  -2.674 0.007504 **
## V5           9.666e-05  5.007e-05   1.930 0.053560 .
## V6A65       -7.960e-01  2.890e-01  -2.755 0.005875 **
## V8           2.619e-01  9.837e-02   2.662 0.007758 **
## V12A124      2.108e-01  3.161e-01   0.667 0.504926
## V14A143     -5.692e-01  2.340e-01  -2.433 0.014984 *
## V15A152     -6.543e-01  2.381e-01  -2.748 0.005993 **
## V20A202     -1.556e+00  7.813e-01  -1.992 0.046388 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 851.79  on 699  degrees of freedom
## Residual deviance: 665.15  on 683  degrees of freedom
## AIC: 699.15
##
## Number of Fisher Scoring iterations: 5
```

```
#Remove V3A32, V5, V12A124
reg = glm(V21 ~ V1A12 + V1A13 + V1A14 + V2+ V3A34 + V4A41 + V4A410 + V4A43 + V6A65 + V8 + V14A14
3 + V15A152 + V20A202,family=binomial(link = "logit"),data=train_data)
summary(reg)
```

```
##
## Call:
## glm(formula = V21 ~ V1A12 + V1A13 + V1A14 + V2 + V3A34 + V4A41 +
##     V4A410 + V4A43 + V6A65 + V8 + V14A143 + V15A152 + V20A202,
##     family = binomial(link = "logit"), data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1293  -0.7635  -0.4204   0.8104   2.5703
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.065258   0.428051   0.152 0.878829
## V1A12       -0.448555   0.234168  -1.916 0.055426 .
## V1A13       -1.122845   0.408449  -2.749 0.005977 **
## V1A14       -1.683884   0.249225  -6.756 1.41e-11 ***
## V2           0.040769   0.008128   5.016 5.27e-07 ***
## V3A34       -0.698511   0.234181  -2.983 0.002856 **
## V4A41       -1.144162   0.377032  -3.035 0.002408 **
## V4A410      -1.943711   0.948136  -2.050 0.040361 *
## V4A43       -0.691870   0.227649  -3.039 0.002372 **
## V6A65       -0.755452   0.283978  -2.660 0.007808 **
## V8           0.192242   0.088846   2.164 0.030483 *
## V14A143     -0.633584   0.227941  -2.780 0.005443 **
## V15A152     -0.736912   0.207552  -3.550 0.000385 ***
## V20A202     -1.551816   0.782540  -1.983 0.047362 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 851.79  on 699  degrees of freedom
## Residual deviance: 672.59  on 686  degrees of freedom
## AIC: 700.59
##
## Number of Fisher Scoring iterations: 5
```

```
#Remove v1a12
reg = glm(V21 ~ V1A13 + V1A14 + V2+ V3A34 + V4A41 + V4A410 + V4A43 + V6A65 + V8 + V14A143 + V15A
152 + V20A202,family=binomial(link = "logit"),data=train_data)
summary(reg)
```

```
##
## Call:
## glm(formula = V21 ~ V1A13 + V1A14 + V2 + V3A34 + V4A41 + V4A410 +
##     V4A43 + V6A65 + V8 + V14A143 + V15A152 + V20A202, family = binomial(link = "logit"),
##     data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0581  -0.7678  -0.4233   0.8413   2.6010
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.185928   0.406649  -0.457 0.647513
## V1A13       -0.901858   0.392357  -2.299 0.021530 *
## V1A14       -1.466344   0.221899  -6.608 3.89e-11 ***
## V2           0.041212   0.008144   5.061 4.18e-07 ***
## V3A34       -0.696997   0.233204  -2.989 0.002801 **
## V4A41       -1.133505   0.374712  -3.025 0.002486 **
## V4A410      -2.037209   0.975433  -2.089 0.036751 *
## V4A43       -0.711609   0.226120  -3.147 0.001649 **
## V6A65       -0.795349   0.283649  -2.804 0.005047 **
## V8           0.205035   0.088325   2.321 0.020268 *
## V14A143     -0.622598   0.226520  -2.749 0.005986 **
## V15A152     -0.760406   0.206817  -3.677 0.000236 ***
## V20A202     -1.435220   0.775514  -1.851 0.064217 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 851.79  on 699  degrees of freedom
## Residual deviance: 676.28  on 687  degrees of freedom
## AIC: 702.28
##
## Number of Fisher Scoring iterations: 5
```

```
#Remove v20a202
reg = glm(V21 ~ V1A13 + V1A14 + V2+ V3A34 + V4A41 + V4A410 + V4A43 + V6A65 + V8 + V14A143 + V15A
152,family=binomial(link = "logit"),data=train_data)
summary(reg)
```

```
##
## Call:
## glm(formula = V21 ~ V1A13 + V1A14 + V2 + V3A34 + V4A41 + V4A410 +
##     V4A43 + V6A65 + V8 + V14A143 + V15A152, family = binomial(link = "logit"),
##     data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0657  -0.7733  -0.4450   0.8570   2.6154
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.289239   0.403301  -0.717 0.473263
## V1A13       -0.898192   0.390404  -2.301 0.021410 *
## V1A14       -1.471913   0.221301  -6.651 2.91e-11 ***
## V2           0.042992   0.008113   5.299 1.16e-07 ***
## V3A34       -0.717145   0.232772  -3.081 0.002064 **
## V4A41       -1.136899   0.376005  -3.024 0.002498 **
## V4A410      -2.206736   0.956408  -2.307 0.021037 *
## V4A43       -0.698407   0.225973  -3.091 0.001997 **
## V6A65       -0.807362   0.282051  -2.862 0.004203 **
## V8           0.212165   0.087871   2.415 0.015756 *
## V14A143     -0.615604   0.225951  -2.725 0.006440 **
## V15A152     -0.748453   0.205615  -3.640 0.000273 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 851.79  on 699  degrees of freedom
## Residual deviance: 680.99  on 688  degrees of freedom
## AIC: 704.99
##
## Number of Fisher Scoring iterations: 5
```

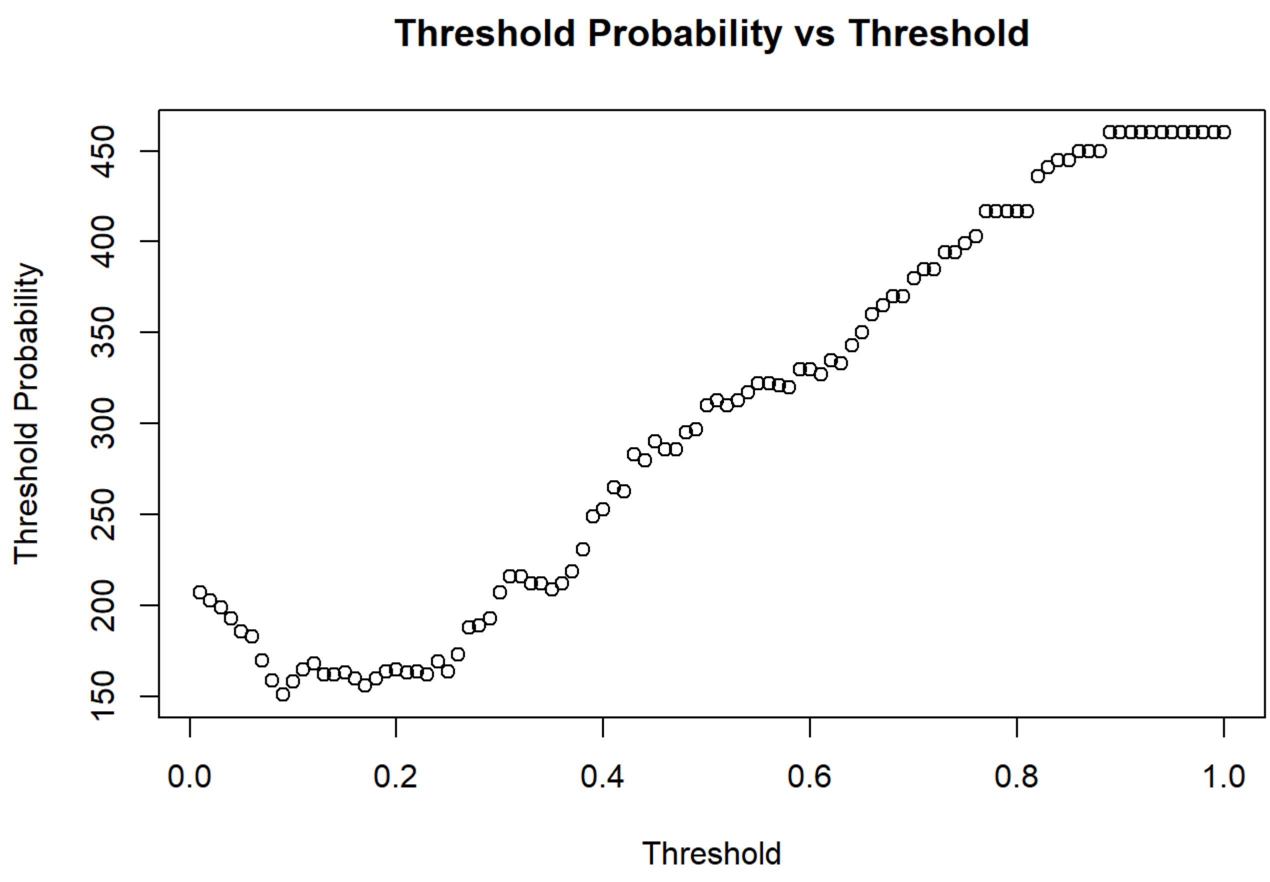
```
#ADD 0, 1 to the data set

validation_data$V1A13[validation_data$V1 == "A13"] <- 1
validation_data$V1A13[validation_data$V1 != "A13"] <- 0

validation_data$V1A14[validation_data$V1 == "A14"] <- 1
validation_data$V1A14[validation_data$V1 != "A14"] <- 0

validation_data$V3A34[validation_data$V3 == "A34"] <- 1
validation_data$V3A34[validation_data$V3 != "A34"] <- 0

validation_data$V4A41[validation_data$V4 == "A41"] <- 1
validation_data$V4A41[validation_data$V4 != "A41"] <- 0

validation_data$V4A410[validation_data$V4 == "A410"] <- 1
validation_data$V4A410[validation_data$V4 != "A410"] <- 0

validation_data$V4A43[validation_data$V4 == "A43"] <- 1
validation_data$V4A43[validation_data$V4 != "A43"] <- 0

validation_data$V6A65[validation_data$V6 == "A65"] <- 1
validation_data$V6A65[validation_data$V6 != "A65"] <- 0

validation_data$V14A143[validation_data$V14 == "A143"] <- 1
validation_data$V14A143[validation_data$V14 != "A143"] <- 0

validation_data$V15A152[validation_data$V15 == "A152"] <- 1
validation_data$V15A152[validation_data$V15 != "A152"] <- 0

# test the model

y_hat<-predict(reg,validation_data,type = "response")

# y_hat is a vector of fractions.
# Now we can use a threshold to make yes/no decisions,
# and view the confusion matrix.

rounded_y_hat <- as.integer(y_hat > 0.5)

t <- table(rounded_y_hat,validation_data$V21)
t
```

```
##
## rounded_y_hat   0   1
##             0 183  57
##             1  25  35
```

```
acc <- (t[1,1] + t[2,2]) / sum(t)
acc #Here is the accuracy value of the model
```

```
## [1] 0.7266667
```

```
############10.3 PART 2#################

# Estimate that threshold_probly identifies a bad as good is 5x worse than threshold_probly iden
tifying good as bad.
threshold_prob <- c()
for(i in 1:100)
{
  rounded_y_hat <- as.integer(y_hat > (i/100))  #threshold preds

  matrix <-as.matrix(table(rounded_y_hat,validation_data$V21))

  if(nrow(matrix)>1) { col1 <- matrix[2,1] } else { col1 <- 0 }
  if(ncol(matrix)>1) { col2 <- matrix[1,2] } else { col2 <- 0 }
  threshold_prob <- c(threshold_prob, col2*5 + col1)
}

plot(c(1:100)/100,threshold_prob,xlab = "Threshold",ylab = "Threshold Probability",main = "Thres
hold Probability vs Threshold")
```




```
which.min(threshold_prob)
```

```
## [1] 9
```

```
#Determine a good threshold probability
#The threshold probability is 9%.

#Thank you for taking the time to read my homework.
```