HW7

2022-10-10

Question 10.1

The first step is to set up the enviroment and load the data. After that, I used the tree funcction to tain the tree model,visualized it,and calculated the R2:

```
# Clear the environment
rm(list = ls())

# Comment in set.seed(33) to repeat results
set.seed(33)

# Load tree lib
require(tree)
```

```
## Loading required package: tree
```

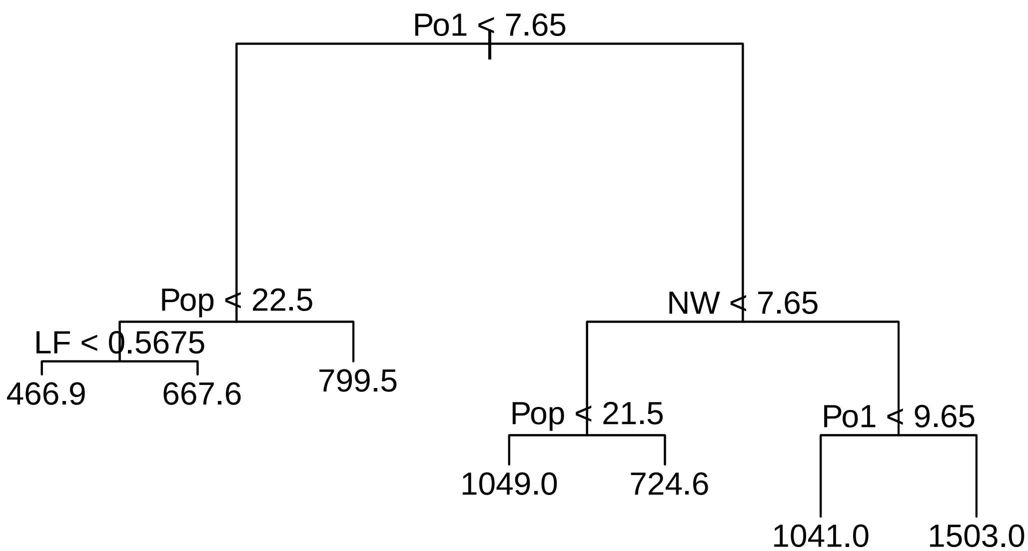
```
require(randomForest)
```

```
## Loading required package: randomForest

## randomForest 4.7-1.1

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

```
# Load crime data into a data frame
data_df <- read.table("uscrime.txt", header=TRUE)

# Train tree model
tree_model <- tree(Crime ~., data_df)
summary(tree_model)
```

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

```r
# Visualize tree model
plot(tree_model)
text(tree_model)
```




```r
# Function to calculate R^2
ComputeR2 <- function(yhat_df, data_df) {
  SSres <- sum((yhat_df - data_df$Crime)^2)
  SStot <- sum((data_df$Crime - mean(data_df$Crime))^2)
  R2 <- 1 - SSres/SStot
  return(R2)
}

# Calculate R^2
tree_yhat <- predict(tree_model)
tree_r2 <- ComputeR2(tree_yhat, data_df)
tree_r2
```

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

According to the models, Po1 is the main branching factor, and when Po1 is less than 7.65, we can use PCA to develop a regression model that can explain around 30% of the variability. We don't have a strong linear regression model for Po1 > 7.65, though, as none of the variables are important.

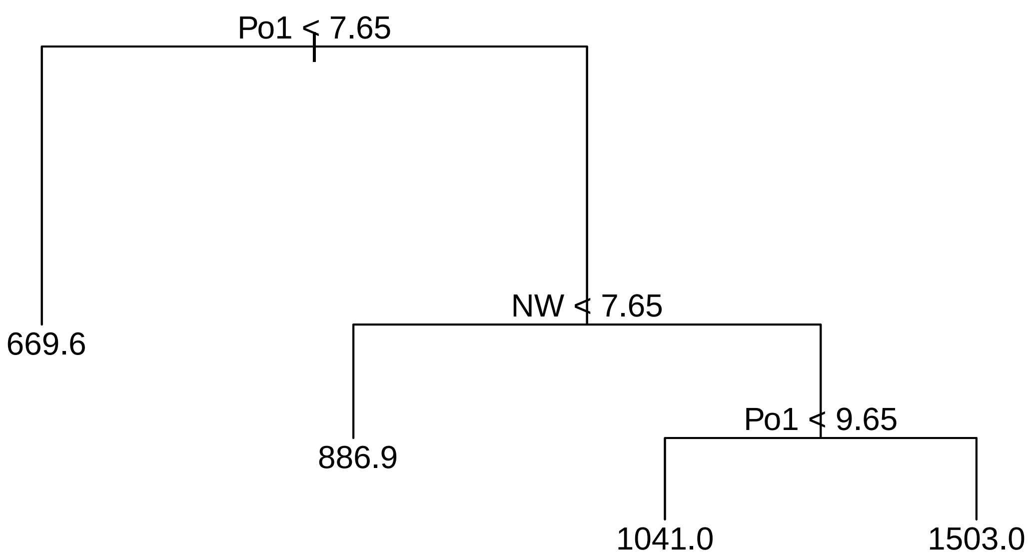
My takeaway from the regression tree model: 1. Due to overfitting, the original regression tree model is incredibly flawed. The model quality can be improved by trimming the tree and deleting variables from the regression models of the resulting leaves. Regression tree models, however, don't seem to be designed to handle such tiny data sets, especially when there are so many variables influencing the size of the data set.

As the dataset is not so big, we can prune the tree to a smaller size.

```
# Manually prune tree
tree_model_pruned <- prune.tree(tree_model,best = 4)
summary(tree_model_pruned)
```

```
##
## Regression tree:
## snip.tree(tree = tree_model, nodes = c(6L, 2L))
## Variables actually used in tree construction:
## [1] "Po1" "NW"
## Number of terminal nodes:  4
## Residual mean deviance:  61220 = 2633000 / 43
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -573.90 -152.60   35.39    0.00  158.90  490.10
```

```
# Visualize pruned tree
plot(tree_model_pruned)
text(tree_model_pruned)
```



```r
# and then we calculate the R^2
# Calc R^2
pruned_yhat <- predict(tree_model_pruned)
pruned_r2 <- ComputeR2(pruned_yhat, data_df)
pruned_r2
```

```
## [1] 0.6174017
```

Cross-validation demonstrates that the random forest method outperforms the earlier models we discovered for this data set by reducing some of the potential for overfitting. The graph displays a similar qualitative pattern to what we saw in the regression tree model. Po1 is regarded as the most significant predicting factor for Crime according to the random forest model as well.

I created a variable to set my factor set to 1 log(n) and then trained a forest model using randomForestO. Later. we calculated R^2 visualized variable importance for the forest model:

```r
# Use 1+log(n) standard to pick number of factors in each set
factor_set <- round(1 + log(ncol(data_df)))

# Train forest model
forest_model <- randomForest(Crime ~., data_df, mtry = factor_set, importance = TRUE, ntree = 500)
forest_model
```

```
##
## Call:
##  randomForest(formula = Crime ~ ., data = data_df, mtry = factor_set,      importance = TRUE, ntree =
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 84735.92
##                     % Var explained: 42.12
```

```r
# Use R2 function to calculate
forest_model_yhat <- predict(forest_model)
ComputeR2(forest_model_yhat, data_df)
```

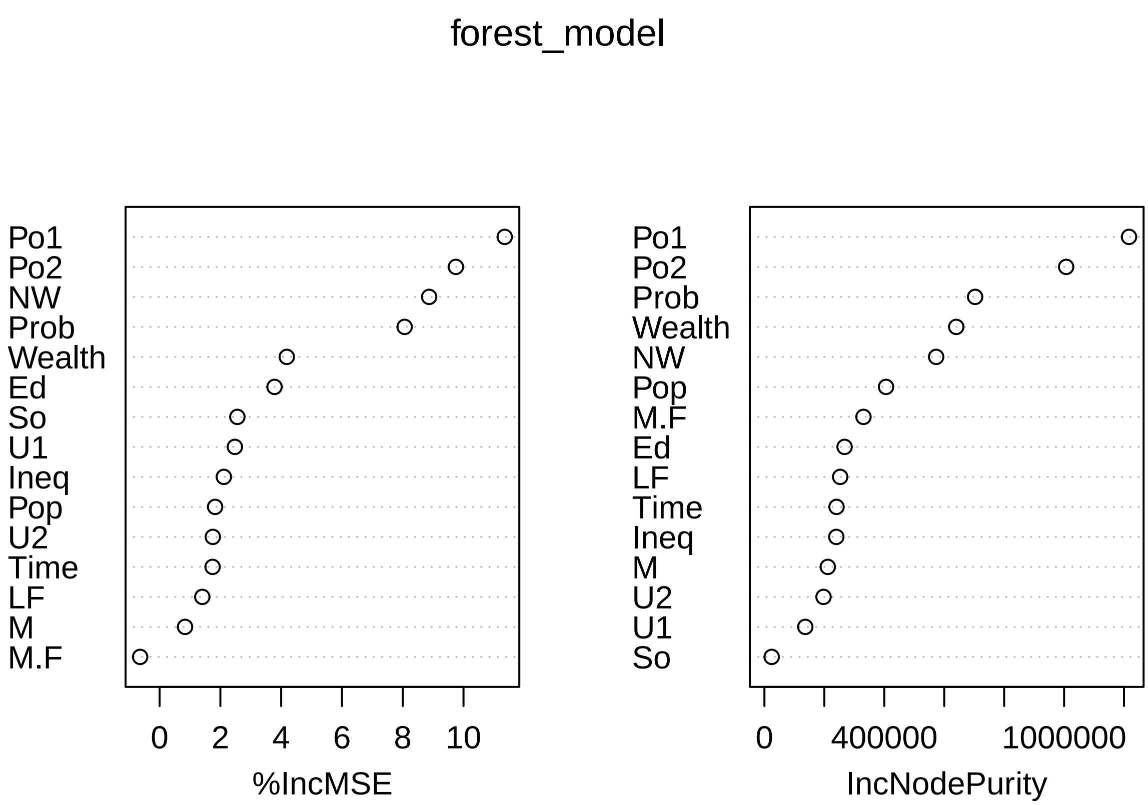
```
## [1] 0.4212135
```

```r
# Visualize variable importance
importance(forest_model)
```

```
##           %IncMSE IncNodePurity
## M       0.8371873     211468.40
## So      2.5592540      24342.06
## Ed      3.7818692     267536.99
## Po1    11.3561686    1216641.54
## Po2     9.7487025    1006931.65
## LF      1.4052189     252824.54
## M.F    -0.6400580     330551.12
## Pop     1.8266879     405866.27
```

```
## NW      8.8691496     573160.13
## U1      2.4773654     136393.12
## U2      1.7511554     197200.97
## Wealth  4.1849440     640225.32
## Ineq    2.1134413     239915.28
## Prob    8.0623962     703096.44
## Time    1.7430770     240622.24
```

```
varImpPlot(forest_model)
```



It is interesting to see the Po1 and Po2 variables at the top.

Question 10.2 A lot of businesses depend on clients recurring membership fees. Retaining members is essential because each client who cancels their subscription significantly reduces our earnings. To assist our relationship managers in concentrating on the high-risk clients, I would apply a logistic regression model to calculate the probability that a customer will terminate their subscription. As predictors, I would look at the number of client contacts, participation in our webinars and meetings, customer margin from the previous quarter, and the proportion of website active users to all staff.

HW7 Q3

2022-10-08

The data was split into training and test sets and then created logistic regression model of the data. When reviewing the summary of the original model, many of the factor have high p-values and do not seem relevant. Therefore, all factors with p-values >.05 were removed and a new model was created. This was repeated, and the formula for the final model was calculated.

```
# Clear the environment
rm(list = ls())

# Comment in set.seed(33) to repeat results
set.seed(33)

# Load data
data <- read.table("/Users/xiaofanjiao/Desktop/germancredit.txt", header=FALSE)

# Change Response (V21) to 1 (good) or 0 (bad)
data$V21[data$V21==1]<-0
data$V21[data$V21==2]<-1

# Create training and test datasets
m <- nrow(data)
trn <- sample(1:m, size = round(m*0.7), replace = FALSE)
d.learn <- data[trn,]
d.valid <- data[-trn,]

# Build model with all factors to determine significant factors
reg = glm(V21 ~.,family=binomial(link = "logit"),data=d.learn)
summary(reg)
```

```
##
## Call:
## glm(formula = V21 ~ ., family = binomial(link = "logit"), data = d.learn)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1059  -0.7221  -0.3863   0.7307   2.5428
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.204e-01  1.276e+00   0.329  0.74190
## V1A12       -4.428e-01  2.587e-01  -1.711  0.08699 .
## V1A13       -6.850e-01  3.938e-01  -1.739  0.08201 .
## V1A14       -1.777e+00  2.722e-01  -6.528 6.67e-11 ***
## V2           2.598e-02  1.108e-02   2.345  0.01905 *
## V3A31        6.869e-02  6.556e-01   0.105  0.91655
```

```
## V3A32       -4.769e-01  5.031e-01  -0.948  0.34319
## V3A33       -9.296e-01  5.521e-01  -1.684  0.09221 .
## V3A34       -1.413e+00  5.117e-01  -2.762  0.00575 **
## V4A41       -2.137e+00  4.977e-01  -4.293 1.77e-05 ***
## V4A410      -1.049e+00  8.689e-01  -1.207  0.22741
## V4A42       -6.450e-01  3.182e-01  -2.027  0.04265 *
## V4A43       -6.544e-01  2.947e-01  -2.221  0.02638 *
## V4A44       -1.540e-01  8.702e-01  -0.177  0.85951
## V4A45       -8.044e-01  7.482e-01  -1.075  0.28230
## V4A46        5.863e-02  4.760e-01   0.123  0.90197
## V4A48       -1.916e+00  1.234e+00  -1.553  0.12033
## V4A49       -5.369e-01  3.987e-01  -1.347  0.17811
## V5           1.253e-04  5.239e-05   2.392  0.01676 *
## V6A62       -1.845e-01  3.554e-01  -0.519  0.60361
## V6A63       -2.105e-01  4.666e-01  -0.451  0.65196
## V6A64       -1.919e+00  7.025e-01  -2.731  0.00631 **
## V6A65       -9.981e-01  3.071e-01  -3.250  0.00115 **
## V7A72       -3.833e-01  5.108e-01  -0.751  0.45295
## V7A73       -3.860e-01  4.830e-01  -0.799  0.42424
## V7A74       -9.249e-01  5.257e-01  -1.759  0.07851 .
## V7A75       -6.927e-01  4.830e-01  -1.434  0.15150
## V8           3.169e-01  1.063e-01   2.983  0.00286 **
## V9A92       -7.059e-02  4.382e-01  -0.161  0.87201
## V9A93       -3.939e-01  4.340e-01  -0.908  0.36402
## V9A94        1.406e-02  5.233e-01   0.027  0.97857
## V10A102     -6.025e-02  5.222e-01  -0.115  0.90816
## V10A103     -8.688e-01  4.887e-01  -1.778  0.07547 .
## V11          1.128e-02  9.910e-02   0.114  0.90937
## V12A122     -1.856e-02  2.973e-01  -0.062  0.95020
## V12A123     -1.006e-01  2.793e-01  -0.360  0.71864
## V12A124      7.153e-01  5.154e-01   1.388  0.16517
## V13         -1.255e-02  1.079e-02  -1.163  0.24470
## V14A142     -3.530e-01  5.089e-01  -0.694  0.48786
## V14A143     -6.302e-01  2.940e-01  -2.144  0.03206 *
## V15A152     -3.941e-01  2.832e-01  -1.392  0.16396
## V15A153     -1.145e+00  6.076e-01  -1.885  0.05946 .
## V16          3.976e-01  2.221e-01   1.791  0.07336 .
## V17A172      4.904e-01  8.128e-01   0.603  0.54628
## V17A173      6.451e-01  7.866e-01   0.820  0.41214
## V17A174      7.430e-01  8.083e-01   0.919  0.35799
## V18          1.351e-01  2.966e-01   0.455  0.64883
## V19A192     -3.406e-01  2.404e-01  -1.417  0.15660
## V20A202     -1.301e+00  6.546e-01  -1.988  0.04679 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 865.13  on 699  degrees of freedom
## Residual deviance: 639.39  on 651  degrees of freedom
## AIC: 737.39
##
## Number of Fisher Scoring iterations: 5
```

```r
# 2nd iteration: Use all the variables found significant in
# the 1st iteration.
reg = glm(V21 ~ V1+V2+V3+V4+V5+V6+V7+V8+V9+V10+V12+V14+V16+V20,family=binomial(link = "logit"),data=d.le
summary(reg)
```

```
##
## Call:
## glm(formula = V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 + V9 +
##     V10 + V12 + V14 + V16 + V20, family = binomial(link = "logit"),
##     data = d.learn)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1266  -0.7401  -0.4025   0.7682   2.5350
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.382e-01  9.449e-01   0.146 0.883744
## V1A12       -4.753e-01  2.545e-01  -1.868 0.061816 .
## V1A13       -7.693e-01  3.861e-01  -1.993 0.046315 *
## V1A14       -1.787e+00  2.698e-01  -6.626 3.45e-11 ***
## V2           2.559e-02  1.075e-02   2.381 0.017253 *
## V3A31       -3.865e-02  6.435e-01  -0.060 0.952112
## V3A32       -5.362e-01  4.966e-01  -1.080 0.280228
## V3A33       -9.572e-01  5.460e-01  -1.753 0.079579 .
## V3A34       -1.468e+00  5.040e-01  -2.913 0.003582 **
## V4A41       -2.089e+00  4.892e-01  -4.270 1.95e-05 ***
## V4A410      -1.222e+00  8.376e-01  -1.459 0.144686
## V4A42       -5.256e-01  3.092e-01  -1.700 0.089192 .
## V4A43       -5.811e-01  2.870e-01  -2.024 0.042928 *
## V4A44       -1.537e-01  8.636e-01  -0.178 0.858741
## V4A45       -8.279e-01  7.163e-01  -1.156 0.247733
## V4A46        2.693e-02  4.622e-01   0.058 0.953540
## V4A48       -1.867e+00  1.227e+00  -1.521 0.128215
## V4A49       -5.457e-01  3.922e-01  -1.391 0.164118
## V5           1.122e-04  4.868e-05   2.305 0.021187 *
## V6A62       -4.900e-02  3.477e-01  -0.141 0.887909
## V6A63       -3.194e-01  4.646e-01  -0.687 0.491770
## V6A64       -1.899e+00  6.847e-01  -2.773 0.005548 **
## V6A65       -1.005e+00  3.004e-01  -3.347 0.000818 ***
## V7A72       -3.217e-02  4.557e-01  -0.071 0.943718
## V7A73       -6.648e-02  4.279e-01  -0.155 0.876523
## V7A74       -6.191e-01  4.772e-01  -1.297 0.194523
## V7A75       -4.891e-01  4.413e-01  -1.108 0.267770
## V8           3.134e-01  1.033e-01   3.032 0.002428 **
## V9A92        1.012e-03  4.255e-01   0.002 0.998102
## V9A93       -3.713e-01  4.199e-01  -0.884 0.376561
## V9A94        7.056e-02  5.116e-01   0.138 0.890308
## V10A102      1.669e-01  5.036e-01   0.331 0.740338
## V10A103     -8.402e-01  4.843e-01  -1.735 0.082787 .
## V12A122      1.836e-02  2.921e-01   0.063 0.949873
## V12A123     -2.874e-02  2.697e-01  -0.107 0.915124
## V12A124      1.893e-01  3.776e-01   0.501 0.616226
```

```
## V14A142     -3.974e-01  4.995e-01  -0.796 0.426246
## V14A143     -6.510e-01  2.900e-01  -2.244 0.024807 *
## V16          3.512e-01  2.155e-01   1.629 0.103226
## V20A202     -1.158e+00  6.435e-01  -1.800 0.071839 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 865.13  on 699  degrees of freedom
## Residual deviance: 649.88  on 660  degrees of freedom
## AIC: 729.88
##
## Number of Fisher Scoring iterations: 5
```

```
# 3rd iteration: Use only the significant variables obtained in the 2nd iteration.

reg = glm(V21 ~ V1+V2+V3+V4+V5+V6+V8+V9+V10+V14+V20,family=binomial(link = "logit"),data=d.learn)
summary(reg)
```

```
##
## Call:
## glm(formula = V21 ~ V1 + V2 + V3 + V4 + V5 + V6 + V8 + V9 + V10 +
##     V14 + V20, family = binomial(link = "logit"), data = d.learn)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1187  -0.7420  -0.4013   0.7805   2.7370
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  7.192e-01  7.458e-01   0.964 0.334929
## V1A12       -4.734e-01  2.497e-01  -1.896 0.057994 .
## V1A13       -8.244e-01  3.806e-01  -2.166 0.030295 *
## V1A14       -1.797e+00  2.665e-01  -6.741 1.57e-11 ***
## V2           2.530e-02  1.049e-02   2.413 0.015824 *
## V3A31       -3.279e-01  6.206e-01  -0.528 0.597273
## V3A32       -8.492e-01  4.727e-01  -1.796 0.072422 .
## V3A33       -1.109e+00  5.391e-01  -2.057 0.039708 *
## V3A34       -1.567e+00  4.961e-01  -3.160 0.001579 **
## V4A41       -2.091e+00  4.800e-01  -4.357 1.32e-05 ***
## V4A410      -1.191e+00  8.255e-01  -1.442 0.149220
## V4A42       -5.699e-01  3.006e-01  -1.896 0.057921 .
## V4A43       -6.090e-01  2.819e-01  -2.160 0.030754 *
## V4A44       -1.241e-01  8.325e-01  -0.149 0.881497
## V4A45       -6.763e-01  7.038e-01  -0.961 0.336546
## V4A46        8.708e-02  4.459e-01   0.195 0.845155
## V4A48       -2.113e+00  1.259e+00  -1.679 0.093145 .
## V4A49       -5.830e-01  3.864e-01  -1.509 0.131377
## V5           1.117e-04  4.765e-05   2.344 0.019072 *
## V6A62       -6.327e-02  3.407e-01  -0.186 0.852693
## V6A63       -3.627e-01  4.572e-01  -0.793 0.427554
## V6A64       -1.865e+00  6.778e-01  -2.752 0.005929 **
## V6A65       -1.020e+00  2.958e-01  -3.449 0.000562 ***
```

```
## V8           2.966e-01  1.008e-01   2.942 0.003266 **
## V9A92        2.743e-04  4.205e-01   0.001 0.999479
## V9A93       -4.193e-01  4.123e-01  -1.017 0.309194
## V9A94        1.057e-01  5.047e-01   0.209 0.834104
## V10A102      3.021e-01  4.932e-01   0.613 0.540177
## V10A103     -9.054e-01  4.752e-01  -1.905 0.056757 .
## V14A142     -2.231e-01  4.867e-01  -0.458 0.646651
## V14A143     -6.412e-01  2.847e-01  -2.253 0.024283 *
## V20A202     -1.136e+00  6.470e-01  -1.755 0.079219 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 865.13  on 699  degrees of freedom
## Residual deviance: 657.43  on 668  degrees of freedom
## AIC: 721.43
##
## Number of Fisher Scoring iterations: 5
```

```
#create a binary variable for each significant factor:
d.learn$V1A13[d.learn$V1 == "A13"] <- 1
d.learn$V1A13[d.learn$V1 != "A13"] <- 0

d.learn$V1A14[d.learn$V1 == "A14"] <- 1
d.learn$V1A14[d.learn$V1 != "A14"] <- 0

d.learn$V3A32[d.learn$V3 == "A32"] <- 1
d.learn$V3A32[d.learn$V3 != "A32"] <- 0

d.learn$V3A33[d.learn$V3 == "A33"] <- 1
d.learn$V3A33[d.learn$V3 != "A33"] <- 0

d.learn$V3A34[d.learn$V3 == "A34"] <- 1
d.learn$V3A34[d.learn$V3 != "A34"] <- 0

d.learn$V4A41[d.learn$V4 == "A41"] <- 1
d.learn$V4A41[d.learn$V4 != "A41"] <- 0

d.learn$V4A410[d.learn$V4 == "A410"] <- 1
d.learn$V4A410[d.learn$V4 != "A410"] <- 0

d.learn$V4A42[d.learn$V4 == "A42"] <- 1
d.learn$V4A42[d.learn$V4 != "A42"] <- 0

d.learn$V4A43[d.learn$V4 == "A43"] <- 1
d.learn$V4A43[d.learn$V4 != "A43"] <- 0

d.learn$V4A48[d.learn$V4 == "A48"] <- 1
d.learn$V4A48[d.learn$V4 != "A48"] <- 0

d.learn$V4A49[d.learn$V4 == "A49"] <- 1
d.learn$V4A49[d.learn$V4 != "A49"] <- 0
```

```r
d.learn$V6A63[d.learn$V6 == "A63"] <- 1
d.learn$V6A63[d.learn$V6 != "A63"] <- 0

d.learn$V6A65[d.learn$V6 == "A65"] <- 1
d.learn$V6A65[d.learn$V6 != "A65"] <- 0

d.learn$V9A93[d.learn$V9 == "A93"] <- 1
d.learn$V9A93[d.learn$V9 != "A93"] <- 0

d.learn$V10A103[d.learn$V10 == "A103"] <- 1
d.learn$V10A103[d.learn$V10 != "A103"] <- 0

d.learn$V14A143[d.learn$V14 == "A143"] <- 1
d.learn$V14A143[d.learn$V14 != "A143"] <- 0

d.learn$V20A202[d.learn$V20 == "A202"] <- 1
d.learn$V20A202[d.learn$V20 != "A202"] <- 0

# Next round model:

reg = glm(V21 ~ V1A13 + V1A14 + V2 + V3A32 + V3A33 + V3A34 + V4A41 + V4A410 + V4A42 + V4A43 + V4A48 + V4
summary(reg)
```

```
##
## Call:
## glm(formula = V21 ~ V1A13 + V1A14 + V2 + V3A32 + V3A33 + V3A34 +
##     V4A41 + V4A410 + V4A42 + V4A43 + V4A48 + V4A49 + V5 + V6A63 +
##     V6A65 + V8 + V9A93 + V10A103 + V14A143 + V20A202, family = binomial(link = "logit"),
##     data = d.learn)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0578  -0.7653  -0.4246   0.8422   2.5792
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  7.391e-02  5.122e-01   0.144 0.885266
## V1A13       -5.507e-01  3.589e-01  -1.534 0.124944
## V1A14       -1.569e+00  2.311e-01  -6.786 1.15e-11 ***
## V2           2.694e-02  1.025e-02   2.628 0.008597 **
## V3A32       -6.589e-01  3.368e-01  -1.956 0.050425 .
## V3A33       -9.704e-01  4.302e-01  -2.256 0.024097 *
## V3A34       -1.395e+00  3.676e-01  -3.795 0.000148 ***
## V4A41       -2.050e+00  4.610e-01  -4.446 8.76e-06 ***
## V4A410      -1.004e+00  8.034e-01  -1.250 0.211263
## V4A42       -4.838e-01  2.674e-01  -1.809 0.070387 .
## V4A43       -5.469e-01  2.527e-01  -2.164 0.030430 *
## V4A48       -1.815e+00  1.204e+00  -1.507 0.131716
## V4A49       -6.362e-01  3.570e-01  -1.782 0.074714 .
## V5           1.116e-04  4.658e-05   2.396 0.016585 *
## V6A63       -3.226e-01  4.507e-01  -0.716 0.474050
## V6A65       -9.754e-01  2.835e-01  -3.441 0.000580 ***
## V8           2.794e-01  9.800e-02   2.851 0.004355 **
```

```
## V9A93       -3.756e-01  2.013e-01  -1.866 0.061994 .
## V10A103     -8.155e-01  4.615e-01  -1.767 0.077220 .
## V14A143     -5.027e-01  2.445e-01  -2.056 0.039799 *
## V20A202     -9.724e-01  6.303e-01  -1.543 0.122884
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 865.13  on 699  degrees of freedom
## Residual deviance: 675.94  on 679  degrees of freedom
## AIC: 717.94
##
## Number of Fisher Scoring iterations: 5
```

```
# Remove V4A48 and V6A63 (p-value above 0.05) and V20A202 (p-value above 0.1)

reg = glm(V21 ~ V1A13 + V1A14 + V2 + V3A32 + V3A33 + V3A34 + V4A41 + V4A410 + V4A42 + V4A43 + V4A49 + V5
summary(reg)
```

```
##
## Call:
## glm(formula = V21 ~ V1A13 + V1A14 + V2 + V3A32 + V3A33 + V3A34 +
##     V4A41 + V4A410 + V4A42 + V4A43 + V4A49 + V5 + V6A65 + V8 +
##     V9A93 + V10A103 + V14A143, family = binomial(link = "logit"),
##     data = d.learn)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0316  -0.7863  -0.4344   0.8779   2.6469
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.658e-01  4.975e-01  -0.333 0.738896
## V1A13       -5.253e-01  3.566e-01  -1.473 0.140752
## V1A14       -1.602e+00  2.283e-01  -7.015 2.31e-12 ***
## V2           2.925e-02  1.017e-02   2.876 0.004027 **
## V3A32       -5.566e-01  3.271e-01  -1.701 0.088863 .
## V3A33       -8.461e-01  4.229e-01  -2.001 0.045446 *
## V3A34       -1.282e+00  3.573e-01  -3.588 0.000334 ***
## V4A41       -1.982e+00  4.581e-01  -4.326 1.52e-05 ***
## V4A410      -9.636e-01  8.175e-01  -1.179 0.238532
## V4A42       -4.166e-01  2.643e-01  -1.576 0.114955
## V4A43       -4.892e-01  2.494e-01  -1.961 0.049841 *
## V4A49       -5.464e-01  3.534e-01  -1.546 0.122005
## V5           1.106e-04  4.641e-05   2.384 0.017143 *
## V6A65       -9.441e-01  2.809e-01  -3.361 0.000776 ***
## V8           2.798e-01  9.721e-02   2.878 0.004001 **
## V9A93       -3.673e-01  1.995e-01  -1.841 0.065668 .
## V10A103     -8.790e-01  4.530e-01  -1.940 0.052362 .
## V14A143     -5.144e-01  2.432e-01  -2.116 0.034385 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 865.13  on 699  degrees of freedom
## Residual deviance: 682.02  on 682  degrees of freedom
## AIC: 718.02
##
## Number of Fisher Scoring iterations: 5
```

```r
# Now add the binary variables to the validation set
d.valid$V1A13[d.valid$V1 == "A13"] <- 1
d.valid$V1A13[d.valid$V1 != "A13"] <- 0

d.valid$V1A14[d.valid$V1 == "A14"] <- 1
d.valid$V1A14[d.valid$V1 != "A14"] <- 0

d.valid$V3A32[d.valid$V3 == "A32"] <- 1
d.valid$V3A32[d.valid$V3 != "A32"] <- 0

d.valid$V3A33[d.valid$V3 == "A33"] <- 1
d.valid$V3A33[d.valid$V3 != "A33"] <- 0

d.valid$V3A34[d.valid$V3 == "A34"] <- 1
d.valid$V3A34[d.valid$V3 != "A34"] <- 0

d.valid$V4A41[d.valid$V4 == "A41"] <- 1
d.valid$V4A41[d.valid$V4 != "A41"] <- 0

d.valid$V4A410[d.valid$V4 == "A410"] <- 1
d.valid$V4A410[d.valid$V4 != "A410"] <- 0

d.valid$V4A42[d.valid$V4 == "A42"] <- 1
d.valid$V4A42[d.valid$V4 != "A42"] <- 0

d.valid$V4A43[d.valid$V4 == "A43"] <- 1
d.valid$V4A43[d.valid$V4 != "A43"] <- 0

d.valid$V4A49[d.valid$V4 == "A49"] <- 1
d.valid$V4A49[d.valid$V4 != "A49"] <- 0

d.valid$V6A65[d.valid$V6 == "A65"] <- 1
d.valid$V6A65[d.valid$V6 != "A65"] <- 0

d.valid$V9A93[d.valid$V9 == "A93"] <- 1
d.valid$V9A93[d.valid$V9 != "A93"] <- 0

d.valid$V10A103[d.valid$V10 == "A103"] <- 1
d.valid$V10A103[d.valid$V10 != "A103"] <- 0

d.valid$V14A143[d.valid$V14 == "A143"] <- 1
d.valid$V14A143[d.valid$V14 != "A143"] <- 0

# test the model

y_hat<-predict(reg,d.valid,type = "response")
```

```
y_hat
```

```
##         11          12          14          17          18          20 
## 0.524146712 0.718377956 0.407178090 0.036437256 0.575982746 0.145005731 
##         21          22          27          28          33          35 
## 0.090091810 0.256250883 0.193190173 0.387148735 0.532645440 0.492194109 
##         45          47          49          50          53          55 
## 0.443487714 0.219477590 0.073796577 0.164057808 0.085870715 0.738635307 
##         56          61          74          82          85          86 
## 0.031801520 0.427455517 0.611254478 0.184130886 0.330589874 0.033479200 
##         89          90          95          98         102         108 
## 0.572034999 0.632990002 0.502920320 0.446451262 0.583109640 0.494685515 
##        118         120         126         127         128         131 
## 0.096672146 0.375199917 0.348735438 0.455489221 0.484146119 0.608634544 
##        133         134         135         139         140         142 
## 0.140628822 0.223933084 0.527187919 0.093844323 0.243635053 0.726378810 
##        146         149         151         152         159         165 
## 0.789752371 0.242607054 0.091400327 0.051665009 0.636511623 0.282264654 
##        166         170         173         178         180         182 
## 0.055324277 0.383819712 0.456771706 0.184436938 0.369850196 0.633417112 
##        192         194         198         200         202         203 
## 0.794813592 0.086314349 0.607365867 0.544378667 0.589617154 0.136267234 
##        205         211         216         227         228         234 
## 0.091499053 0.011193677 0.087247170 0.788725271 0.579184203 0.296553787 
##        238         247         253         254         257         262 
## 0.683626876 0.053617462 0.653149763 0.100549211 0.090940826 0.567641166 
##        270         271         272         282         286         288 
## 0.062527890 0.098870179 0.066932953 0.121514894 0.832530651 0.486234793 
##        289         291         292         296         297         304 
## 0.266150797 0.086638239 0.300167259 0.756219055 0.009789358 0.309403651 
##        307         310         312         318         319         324 
## 0.036231141 0.543876924 0.336882723 0.243337524 0.084454431 0.479751704 
##        328         333         334         339         343         344 
## 0.299690599 0.921039841 0.143617180 0.565003755 0.362519765 0.416080397 
##        348         349         350         351         356         358 
## 0.418564181 0.032494948 0.305484877 0.103515707 0.704560123 0.211018478 
##        359         363         371         376         378         382 
## 0.145315079 0.406564884 0.162837145 0.669935590 0.022075710 0.397099861 
##        383         393         396         397         406         407 
## 0.132605150 0.707261333 0.751665906 0.455628615 0.373122487 0.008220193 
##        409         419         428         429         430         433 
## 0.193317171 0.245579323 0.066143300 0.050493882 0.322050349 0.223708463 
##        435         443         444         446         448         449 
## 0.335839907 0.287510843 0.241996520 0.047981895 0.127512240 0.144916525 
##        452         454         455         457         459         460 
## 0.094865586 0.044303438 0.547406295 0.265802209 0.523889907 0.139310607 
##        463         464         465         466         470         471 
## 0.467698345 0.269966018 0.201565001 0.110466155 0.031725932 0.539217186 
##        473         478         479         482         487         491 
## 0.462690121 0.571177857 0.300352399 0.586544360 0.079870669 0.035719807 
##        492         493         499         505         508         512 
## 0.739375179 0.030450143 0.306690273 0.672091080 0.605571269 0.044919297 
##        515         517         520         524         525         532 
```

```
## 0.111116388 0.190972440 0.018124254 0.052482919 0.302285715 0.513105421
##         534         541         542         548         549         558
## 0.203967279 0.283803325 0.190083671 0.187299460 0.707699138 0.220271713
##         559         564         566         572         573         582
## 0.728002837 0.538672698 0.424932400 0.114585995 0.019800822 0.301846380
##         585         588         594         596         602         603
## 0.111397079 0.216350327 0.646810788 0.596864819 0.457708876 0.865110778
##         608         609         618         621         622         623
## 0.677290459 0.139123338 0.187343463 0.214503143 0.132221937 0.297622194
##         626         628         631         632         636         641
## 0.054045069 0.483551169 0.485069148 0.704257192 0.255936011 0.620487884
##         644         646         648         650         658         660
## 0.051855705 0.126640002 0.144964796 0.485389558 0.384409113 0.277650682
##         661         664         668         669         679         681
## 0.285428832 0.404907226 0.442504303 0.353603520 0.623814777 0.098581407
##         682         685         687         692         695         702
## 0.051323090 0.495758154 0.061377054 0.582295713 0.165021085 0.252004772
##         703         707         709         710         714         715
## 0.430917688 0.796525353 0.447596321 0.261858688 0.212432211 0.943828127
##         718         720         723         729         731         747
## 0.132045274 0.489149718 0.591222343 0.881428408 0.333402262 0.440135299
##         750         752         755         756         759         762
## 0.020746274 0.420020099 0.135497870 0.445765820 0.064815784 0.377999281
##         763         765         768         769         771         774
## 0.406506183 0.186924332 0.007604557 0.152520617 0.069759791 0.137196825
##         775         776         777         793         795         797
## 0.219948625 0.448117590 0.202965828 0.012946322 0.188072536 0.052496753
##         805         808         812         815         817         821
## 0.326601782 0.022068498 0.264575101 0.794724184 0.096632599 0.178519486
##         823         825         826         827         829         830
## 0.600165916 0.100089929 0.487127041 0.449524124 0.152395276 0.469503401
##         831         832         833         839         850         852
## 0.123687576 0.632550330 0.830454819 0.103109964 0.492376866 0.007625786
##         853         858         859         861         866         872
## 0.062995992 0.081672510 0.617540566 0.030775362 0.118043468 0.095280028
##         873         874         877         879         880         883
## 0.288672212 0.106396143 0.635993635 0.414759485 0.039871532 0.261998980
##         886         893         894         897         898         907
## 0.579596623 0.473864098 0.144621401 0.573331326 0.068257713 0.277232908
##         908         909         911         912         917         932
## 0.460585653 0.018094603 0.278382983 0.480332246 0.013380939 0.460319941
##         934         940         943         949         951         953
## 0.052533275 0.007856161 0.106385299 0.191557485 0.321797344 0.227478715
##         954         956         959         960         965         971
## 0.532698980 0.378942029 0.621808816 0.534893508 0.457140801 0.319107528
##         973         975         976         977         984         986
## 0.606688304 0.148911733 0.360995956 0.143469619 0.256669049 0.340237765
##         987         989         990         991         993         994
## 0.675159684 0.261668809 0.309022231 0.027382558 0.202615107 0.643186844
```

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

```r
y_hat_round <- as.integer(y_hat > 0.5)

t <- table(y_hat_round,d.valid$V21)
t
```

```
##
## y_hat_round   0   1
##           0 188  40
##           1  28  44
```

```r
# Model's accuracy is (183 + 43) / (183 + 43 + 22 + 52) = 75%.

acc <- (t[1,1] + t[2,2]) / sum(t)
acc
```

```
## [1] 0.7733333
```

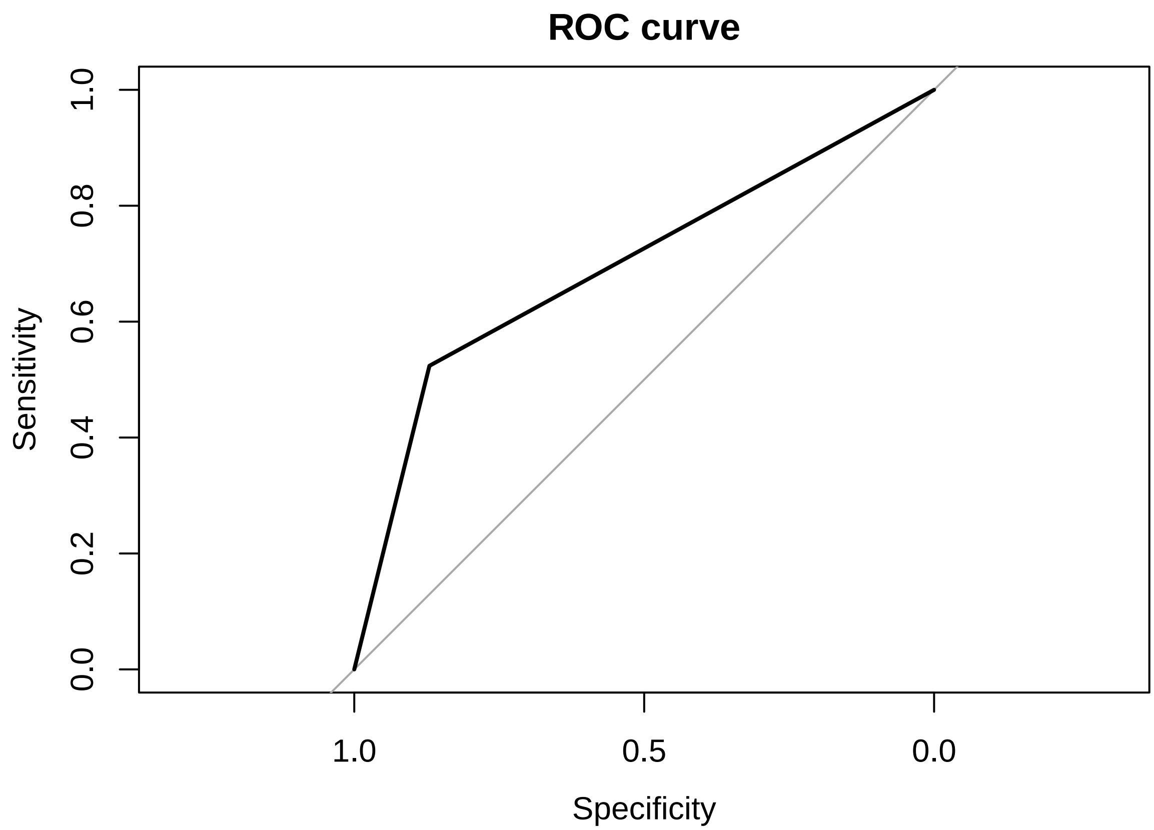
```r
# Import the library for developing ROC curve

library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
# Develop ROC curve to determine the quality of fit

r<-roc(d.valid$V21,y_hat_round)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```r
# Plot the ROC curve

plot(r,main="ROC curve")
```




```
r
```

```
##
## Call:
## roc.default(response = d.valid$V21, predictor = y_hat_round)
##
## Data: y_hat_round in 216 controls (d.valid$V21 0) < 84 cases (d.valid$V21 1).
## Area under the curve: 0.6971
```

67% of the curve's surface is under it. This indicates that the model will correctly categorise both samples 67% of the time when one sample is taken from the response group and another sample is picked from the non-response group.