HW7

10.1

First, I split the dataset into training and test data. As shown in the decision tree graph, it looks like only one attribute matters the most in predicting the decision variable. However, the only plausible explanation I can give for this behavior is the limited number of data points. I feel that we need a larger dataset to make meaningful conclusions regarding the fit of a decision tree regression model. It has to be noted there are other variables that are important as well in the following order (Po1, Po2, Wealth, Prob, M, M.F, Ineq), however Po1 seems to be the most important at each node.

Also, the RMSE of this model for the test data is 310.7. The mean Crime obtained is 913.3.

Next, I fit a random forest model for the same data to observe the RMSE drop to 263.8. This tells me that it is better to use multiple decision trees rather than a single decision tree model. The effect of the random forest is well shown in the lower RMSE. When I look at the importance parameters I can see that M, So, Ed, Po1 and Po2 to be the top parameters.

10.2

Logistic Regression can be used to predict the probability that someone will get a heart attack. This would be better than just classifying if someone would get a heart attack or not. The probability of getting one will be more informative than a simple yes or no.

The attributes that can be useful are as follows: 1) Age 2) Sex 3) Height 4) Weight 5) BMI 6) Lower blood pressure 7) Higher blood pressure 8) A1C level 9) Good cholesterol level 10) Bad cholesterol level 11) Triglyceride level 12) heartbeats per minute 13) Pulse rate.

10.3

First, I transformed the data using one hot encoder to transform the categorical values of some attributes into numerical values. Also, I transformed the values of the response variable to just 0 or 1. I then fit a logistic regression model to the data and analyzed the z values of the different attributes to filter out anything that has a z value greater than 0.05. The idea is to select attributes that are indeed useful in explaining the variability of the data to the best. I then fit another logistic regression model using these reduced attributes and obtained a prediction accuracy of 77%.

It must be noted that I used a threshold of 0.5 in classifying the data points to achieve the above-mentioned accuracy. I then used different threshold values to observe the changes in the accuracy and concluded that a threshold of 0.5 gives the best accuracy.

Also, I created an equation as follows to determine the cost for different threshold values to find out that the best threshold value is about 0.6 i.e. the threshold that has the lowest cost, after which the drop in the threshold is not worth the drop in the cost observed. The equation is as follows:

Cost = Good - correctly predicted * 0 + Bad - correctly predicted * 0 + Good in reality but bad in the model * 1.0 + Bad in reality but Good in model * 5.0

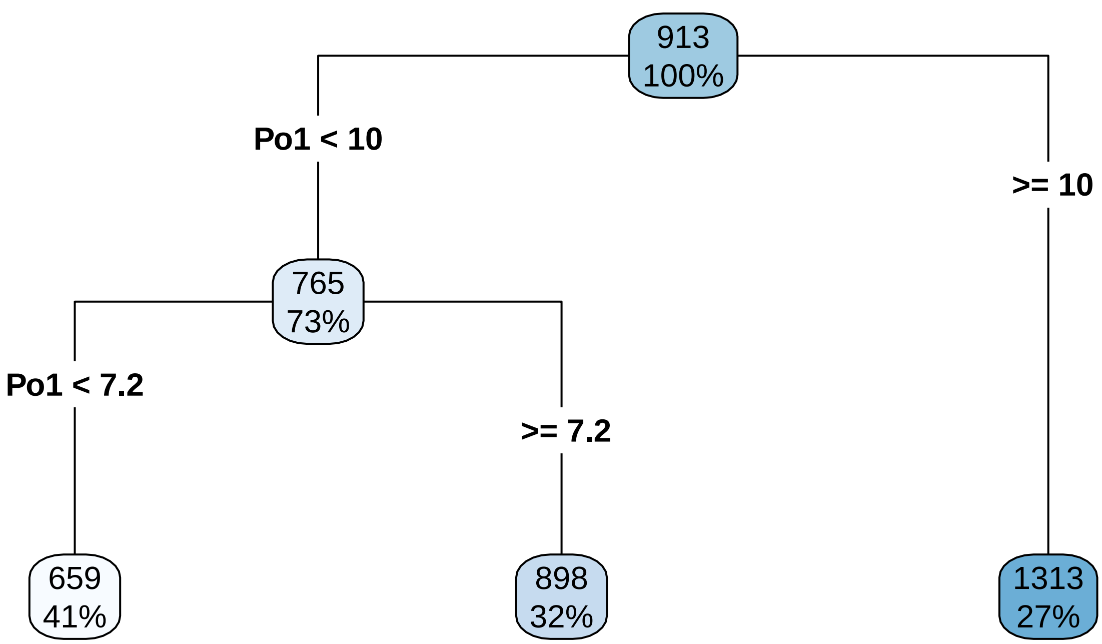
```r
library(rpart)
library(rpart.plot)
library(mltools)
```

```
## Warning: package 'mltools' was built under R version 3.4.4
```

```r
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 3.4.4
```

```r
#library(Metrics)

crime_data = read.table("uscrime.txt", header=TRUE)

#### Split the dataset into training and test data
set.seed(100)
samples = sort(sample(nrow(crime_data), nrow(crime_data)*0.80))
train_data = crime_data[samples,]
test_data = crime_data[-samples,]
## Regression Tree Model
mod = rpart(Crime ~., data=train_data)
rpart.plot(mod, type=4, extra=100, fallen.leaves = TRUE)
```



```r
summary(mod)
```

```
## Call:
## rpart(formula = Crime ~ ., data = train_data)
##   n= 37
##
##           CP nsplit rel error    xerror      xstd
## 1 0.4011410      0 1.0000000 1.0679573 0.3248226
## 2 0.0692256      1 0.5988590 0.7405842 0.1790819
## 3 0.0100000      2 0.5296334 0.7056235 0.1799912
##
## Variable importance
##    Po1    Po2 Wealth   Prob      M    M.F   Ineq
##     28     25     22     10      7      5      2
##
## Node number 1: 37 observations,    complexity param=0.401141
##   mean=913.3784, MSE=147742.7
##   left son=2 (27 obs) right son=3 (10 obs)
##   Primary splits:
##       Po1    < 10        to the left,  improve=0.4011410, (0 missing)
##       Po2    < 9.3       to the left,  improve=0.3805235, (0 missing)
##       Wealth < 6230      to the left,  improve=0.2552668, (0 missing)
##       Prob   < 0.042399  to the right, improve=0.2477063, (0 missing)
##       NW     < 7.65      to the left,  improve=0.2179881, (0 missing)
##   Surrogate splits:
##       Po2    < 9.3       to the left,  agree=0.973, adj=0.9, (0 split)
##       Wealth < 6230      to the left,  agree=0.946, adj=0.8, (0 split)
##       Prob   < 0.0198995 to the right, agree=0.811, adj=0.3, (0 split)
##       M      < 12.2      to the right, agree=0.784, adj=0.2, (0 split)
##       M.F    < 94.3      to the right, agree=0.784, adj=0.2, (0 split)
##
## Node number 2: 27 observations,    complexity param=0.0692256
##   mean=765.2222, MSE=44349.65
##   left son=4 (15 obs) right son=5 (12 obs)
##   Primary splits:
##       Po1  < 7.15      to the left,  improve=0.3160244, (0 missing)
##       Prob < 0.053001  to the right, improve=0.2807898, (0 missing)
##       Po2  < 7.2       to the left,  improve=0.2724502, (0 missing)
##       LF   < 0.541     to the left,  improve=0.2183302, (0 missing)
##       Pop  < 22.5      to the left,  improve=0.2155764, (0 missing)
##   Surrogate splits:
##       Po2    < 6.5       to the left,  agree=0.926, adj=0.833, (0 split)
##       Wealth < 4875      to the left,  agree=0.889, adj=0.750, (0 split)
##       Prob   < 0.043598  to the right, agree=0.889, adj=0.750, (0 split)
##       M      < 13.25     to the right, agree=0.815, adj=0.583, (0 split)
##       Ineq   < 23.05     to the right, agree=0.778, adj=0.500, (0 split)
##
## Node number 3: 10 observations
##   mean=1313.4, MSE=207621
##
## Node number 4: 15 observations
##   mean=659.3333, MSE=35219.42
##
```

```
## Node number 5: 12 observations
##   mean=897.5833, MSE=24227.41
```

```r
##Predict
pred = predict(mod, test_data)

cat("The RMSE is for a Regression Tree model is ", rmse(test_data[,16],pred))
```

```
## The RMSE is for a Regression Tree model is  310.6628
```

```r
########################################################################################################
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
```

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

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

```r
rf = randomForest(Crime ~., data=train_data, importance=TRUE)
summary(rf)
```

```
##                 Length Class  Mode
## call              4    -none- call
## type              1    -none- character
## predicted        37    -none- numeric
## mse             500    -none- numeric
## rsq             500    -none- numeric
## oob.times        37    -none- numeric
## importance       30    -none- numeric
## importanceSD     15    -none- numeric
## localImportance   0    -none- NULL
## proximity         0    -none- NULL
## ntree             1    -none- numeric
## mtry              1    -none- numeric
## forest           11    -none- list
## coefs             0    -none- NULL
## y                37    -none- numeric
## test              0    -none- NULL
## inbag             0    -none- NULL
## terms             3    terms  call
```

```r
importance(rf)
```

```
##           %IncMSE IncNodePurity
## M       1.5107972     115017.48
## So      1.3067615      13026.14
## Ed      2.3430988     145911.87
## Po1    10.3794578    1005500.22
## Po2    10.8088570     902928.09
```

```
## LF      5.2172306     271884.30
## M.F     1.4260296     328212.25
## Pop     0.3573072     291006.19
## NW      4.1130729     261526.57
## U1     -0.8486389      72669.93
## U2     -1.2219522     124575.63
## Wealth  4.5499614     631531.24
## Ineq    0.7754722     150485.01
## Prob    5.3694680     535788.91
## Time   -0.6178077     147117.95
```

```
pred = predict(rf, test_data)

cat("The RMSE for Random Forest model is ", rmse(test_data[,16],pred))
```

```
## The RMSE for Random Forest model is  268.2567
```

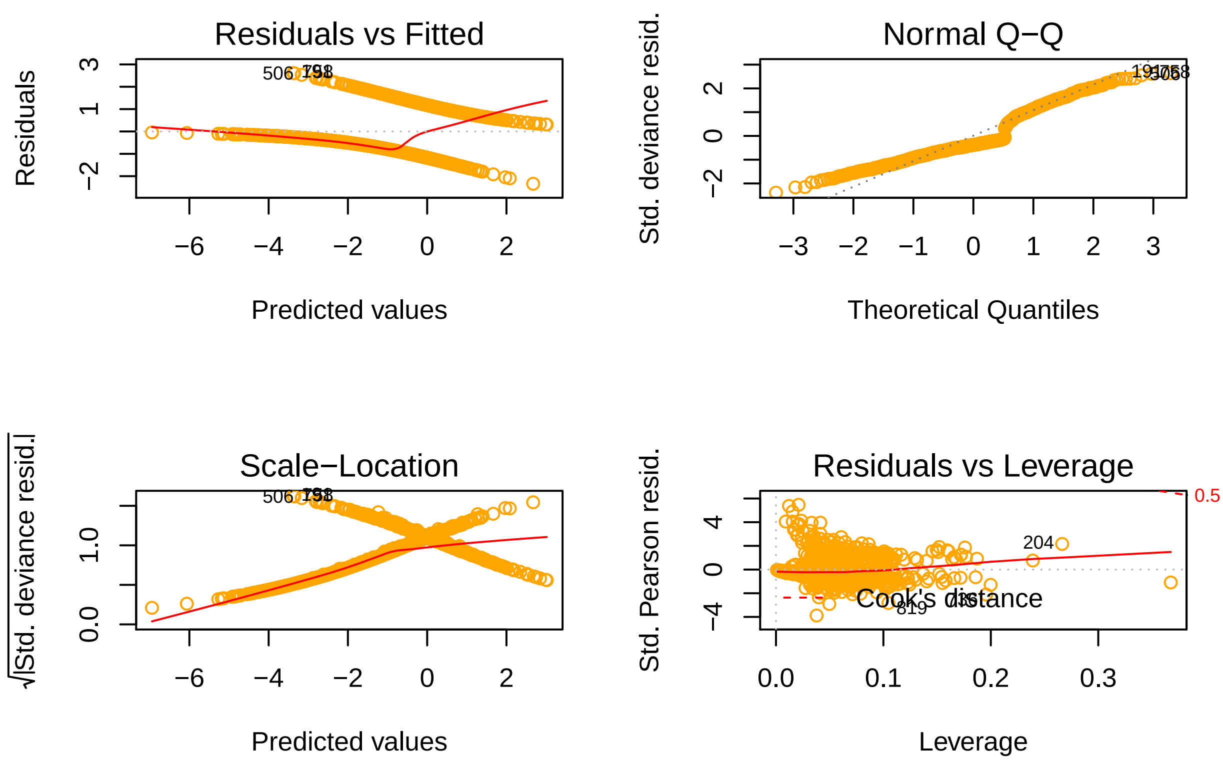
```
########################################################################################################

german_credit = read.table("germancredit.txt", header=FALSE)

new_data <- one_hot(as.data.table(german_credit))
new_data[,"V21"] = new_data[,"V21"] - 1


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

```
##
## Call:
## glm(formula = V21 ~ ., family = binomial(link = "logit"), data = new_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3410  -0.6994  -0.3752   0.7095   2.6116
##
## Coefficients: (13 not defined because of singularities)
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -7.869e+00  1.279e+00  -6.152 7.66e-10 ***
## V1_A11       1.712e+00  2.322e-01   7.373 1.66e-13 ***
## V1_A12       1.337e+00  2.325e-01   5.752 8.83e-09 ***
## V1_A13       7.462e-01  3.831e-01   1.948 0.051419 .
## V1_A14              NA         NA      NA       NA
## V2           2.786e-02  9.296e-03   2.997 0.002724 **
## V3_A30       1.436e+00  4.399e-01   3.264 0.001099 **
## V3_A31       1.579e+00  4.381e-01   3.605 0.000312 ***
## V3_A32       8.497e-01  2.587e-01   3.284 0.001022 **
## V3_A33       5.826e-01  3.345e-01   1.742 0.081540 .
## V3_A34              NA         NA      NA       NA
## V4_A40       7.401e-01  3.339e-01   2.216 0.026668 *
## V4_A41      -9.264e-01  4.409e-01  -2.101 0.035645 *
## V4_A410     -7.487e-01  7.998e-01  -0.936 0.349202
## V4_A42      -5.152e-02  3.543e-01  -0.145 0.884391
```

```
## V4_A43      -1.515e-01  3.370e-01  -0.450 0.653002
## V4_A44       2.173e-01  8.041e-01   0.270 0.786976
## V4_A45       5.237e-01  5.933e-01   0.883 0.377428
## V4_A46       7.764e-01  4.660e-01   1.666 0.095718 .
## V4_A48      -1.319e+00  1.233e+00  -1.070 0.284625
## V4_A49              NA         NA      NA       NA
## V5           1.283e-04  4.444e-05   2.887 0.003894 **
## V6_A61       9.467e-01  2.625e-01   3.607 0.000310 ***
## V6_A62       5.889e-01  3.493e-01   1.686 0.091805 .
## V6_A63       5.706e-01  4.492e-01   1.270 0.203940
## V6_A64      -3.925e-01  5.644e-01  -0.695 0.486765
## V6_A65              NA         NA      NA       NA
## V7_A71       2.766e-01  4.134e-01   0.669 0.503410
## V7_A72       2.097e-01  2.947e-01   0.712 0.476718
## V7_A73       9.379e-02  2.510e-01   0.374 0.708653
## V7_A74      -5.544e-01  3.007e-01  -1.844 0.065230 .
## V7_A75              NA         NA      NA       NA
## V8           3.301e-01  8.828e-02   3.739 0.000185 ***
## V9_A91       3.671e-01  4.537e-01   0.809 0.418448
## V9_A92       9.162e-02  3.118e-01   0.294 0.768908
## V9_A93      -4.490e-01  3.152e-01  -1.424 0.154345
## V9_A94              NA         NA      NA       NA
## V10_A101     9.786e-01  4.243e-01   2.307 0.021072 *
## V10_A102     1.415e+00  5.685e-01   2.488 0.012834 *
## V10_A103            NA         NA      NA       NA
## V11          4.776e-03  8.641e-02   0.055 0.955920
## V12_A121    -7.304e-01  4.245e-01  -1.721 0.085308 .
## V12_A122    -4.490e-01  4.130e-01  -1.087 0.277005
## V12_A123    -5.359e-01  4.017e-01  -1.334 0.182211
## V12_A124            NA         NA      NA       NA
## V13         -1.454e-02  9.222e-03  -1.576 0.114982
## V14_A141     6.463e-01  2.391e-01   2.703 0.006871 **
## V14_A142     5.231e-01  3.754e-01   1.393 0.163501
## V14_A143            NA         NA      NA       NA
## V15_A151     6.839e-01  4.770e-01   1.434 0.151657
## V15_A152     2.402e-01  4.503e-01   0.534 0.593687
## V15_A153            NA         NA      NA       NA
## V16          2.721e-01  1.895e-01   1.436 0.151109
## V17_A171    -4.795e-01  6.623e-01  -0.724 0.469086
## V17_A172     5.666e-02  3.501e-01   0.162 0.871450
## V17_A173     7.524e-02  2.845e-01   0.264 0.791419
## V17_A174            NA         NA      NA       NA
## V18          2.647e-01  2.492e-01   1.062 0.288249
## V19_A191     3.000e-01  2.013e-01   1.491 0.136060
## V19_A192            NA         NA      NA       NA
## V20_A201     1.392e+00  6.258e-01   2.225 0.026095 *
## V20_A202            NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1221.73  on 999  degrees of freedom
## Residual deviance:  895.82  on 951  degrees of freedom
```

```
## AIC: 993.82
##
## Number of Fisher Scoring iterations: 5
```

```r
par(mfrow=c(2,2))
plot(mod, col="orange")
```

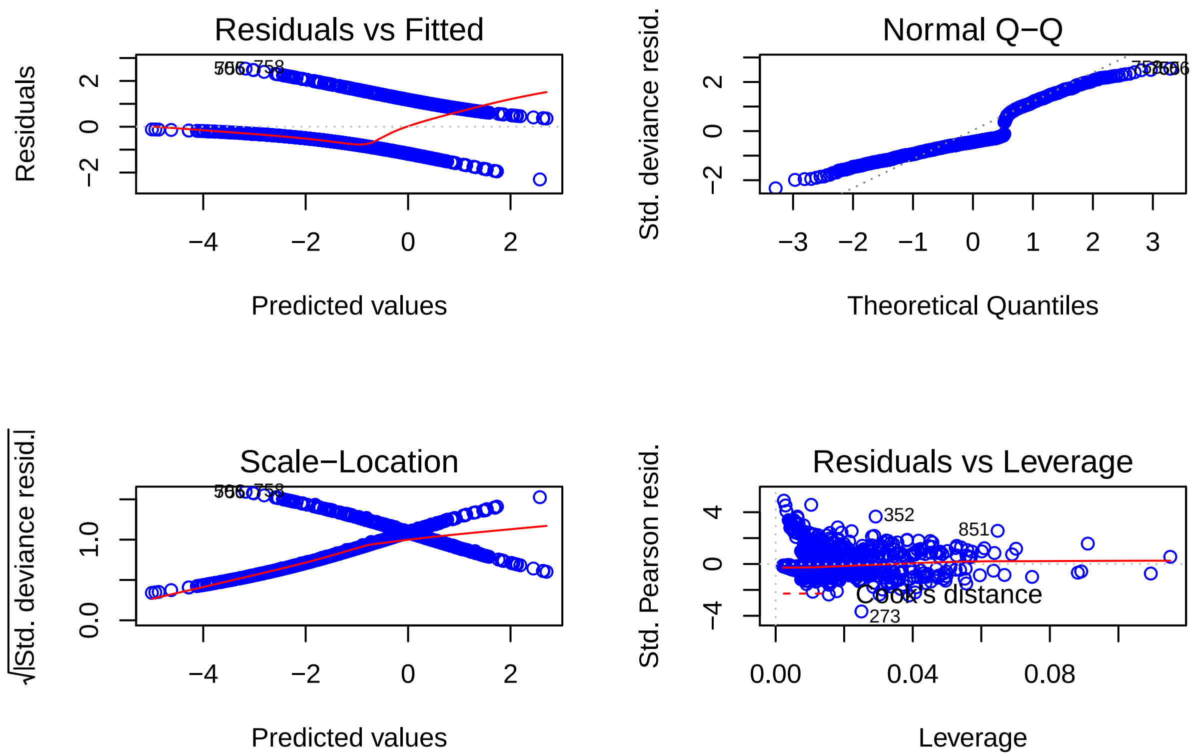



```r
p = summary(mod)$coefficients[, "Pr(>|z|)"]

p = p[p <0.05]
pnames = names(p)
pnames = c(pnames, "V21")

pnames = pnames[c(2:length(pnames))]

new_data = as.data.frame(new_data)
new_data_2 = new_data[,pnames]

mod2 = glm(V21 ~.,data=new_data_2, family=binomial(link="logit"))
summary(mod2)
```

```
##
## Call:
## glm(formula = V21 ~ ., family = binomial(link = "logit"), data = new_data_2)
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3006  -0.7489  -0.4430   0.8168   2.5353
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.956e+00  8.126e-01  -8.560  < 2e-16 ***
## V1_A11       1.631e+00  2.025e-01   8.056 7.89e-16 ***
## V1_A12       1.305e+00  2.025e-01   6.443 1.17e-10 ***
## V2           2.963e-02  8.466e-03   3.499 0.000467 ***
## V3_A30       1.411e+00  3.953e-01   3.569 0.000359 ***
## V3_A31       1.383e+00  3.722e-01   3.715 0.000203 ***
## V3_A32       6.141e-01  1.783e-01   3.444 0.000573 ***
## V4_A40       6.219e-01  1.881e-01   3.307 0.000944 ***
## V4_A41      -9.761e-01  3.261e-01  -2.993 0.002764 **
## V5           8.228e-05  3.900e-05   2.110 0.034878 *
## V6_A61       7.278e-01  1.788e-01   4.070 4.71e-05 ***
## V8           2.244e-01  8.038e-02   2.791 0.005253 **
## V10_A101     1.172e+00  3.973e-01   2.949 0.003185 **
## V10_A102     1.509e+00  5.417e-01   2.785 0.005346 **
## V14_A141     4.861e-01  2.246e-01   2.164 0.030463 *
## V20_A201     1.390e+00  6.127e-01   2.269 0.023296 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1221.73  on 999  degrees of freedom
## Residual deviance:  961.04  on 984  degrees of freedom
## AIC: 993.04
##
## Number of Fisher Scoring iterations: 5
```

```
par(mfrow=c(2,2))
plot(mod2, col="blue")
```




```
pred = predict(mod2, newdata = new_data_2, type="response")
predicted_classes = ifelse(pred > 0.5, "good", "bad")

true_classes = ifelse(new_data_2[,16]==1, "good", "bad")
tab = table(predicted_classes,true_classes)
cat("Confusion Matrix is \n")
```

```
## Confusion Matrix is
```

```
print(tab)
```

```
##                    true_classes
## predicted_classes bad good
##              bad   627  157
##              good   73  143
```

```
cat("Accuracy is ", sum(diag(tab)/sum(tab)))
```

```
## Accuracy is  0.77
```

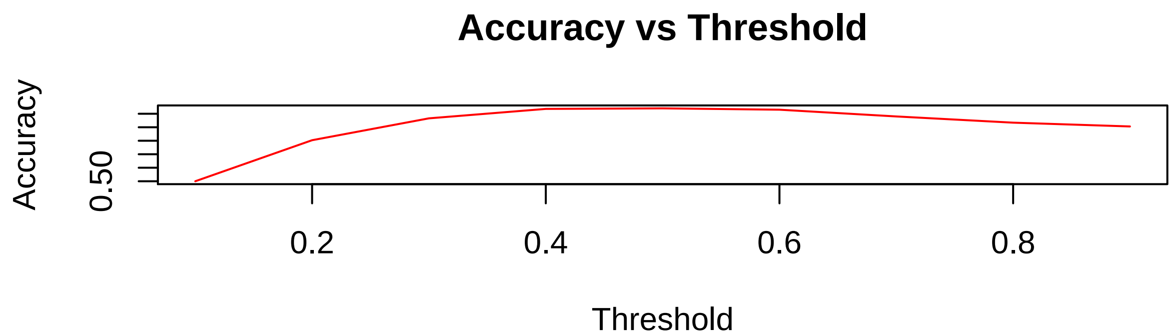
```
ac = c()
th = c()
cost = c()
```

```r
for (i in seq(0.1, 0.9, by = 0.1))
{
  predicted_classes = ifelse(pred > i, "good", "bad")

  tab = table(predicted_classes,true_classes)
  cat("Confusion Matrix for a threshold of", i, "is \n")
  print(tab)
  print(tab[2,2])

  cat("Accuracy for a threshold of ", i, "is", sum(diag(tab)/sum(tab)), "\n")
  cat("\n")
  ac = c(ac, sum(diag(tab)/sum(tab)))
  th = c(th, i)
  c = tab[1,1]*0.0 + tab[2,2] * 0.0 + tab[1,2] *1.0 + tab[2,1] * 5.0
  cost = c(cost,c )
}
```

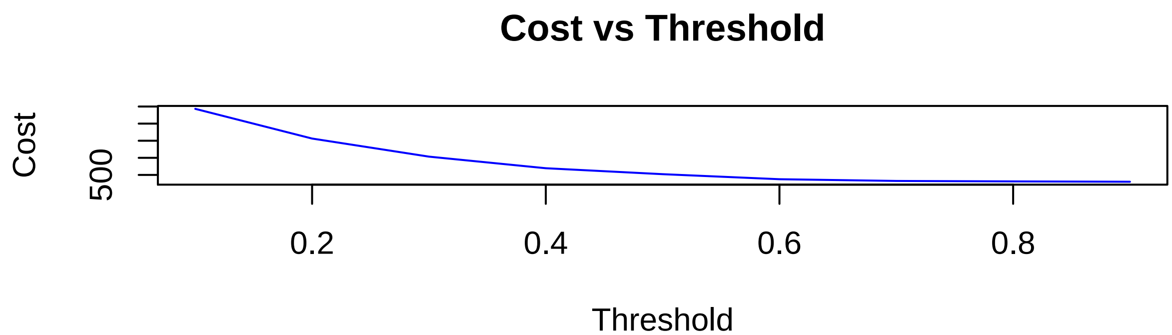
```
## Confusion Matrix for a threshold of 0.1 is
##                  true_classes
## predicted_classes bad good
##              bad  217   17
##              good 483  283
## [1] 283
## Accuracy for a threshold of  0.1 is 0.5
##
## Confusion Matrix for a threshold of 0.2 is
##                  true_classes
## predicted_classes bad good
##              bad  396   44
##              good 304  256
## [1] 256
## Accuracy for a threshold of  0.2 is 0.652
##
## Confusion Matrix for a threshold of 0.3 is
##                  true_classes
## predicted_classes bad good
##              bad  508   75
##              good 192  225
## [1] 225
## Accuracy for a threshold of  0.3 is 0.733
##
## Confusion Matrix for a threshold of 0.4 is
##                  true_classes
## predicted_classes bad good
##              bad  584  116
##              good 116  184
## [1] 184
## Accuracy for a threshold of  0.4 is 0.768
##
## Confusion Matrix for a threshold of 0.5 is
##                  true_classes
## predicted_classes bad good
##              bad  627  157
```

```
##              good  73  143
## [1] 143
## Accuracy for a threshold of  0.5 is 0.77
##
## Confusion Matrix for a threshold of 0.6 is
##                  true_classes
## predicted_classes bad good
##              bad  665  200
##              good  35  100
## [1] 100
## Accuracy for a threshold of  0.6 is 0.765
##
## Confusion Matrix for a threshold of 0.7 is
##                  true_classes
## predicted_classes bad good
##              bad  684  244
##              good  16   56
## [1] 56
## Accuracy for a threshold of  0.7 is 0.74
##
## Confusion Matrix for a threshold of 0.8 is
##                  true_classes
## predicted_classes bad good
##              bad  693  276
##              good   7   24
## [1] 24
## Accuracy for a threshold of  0.8 is 0.717
##
## Confusion Matrix for a threshold of 0.9 is
##                  true_classes
## predicted_classes bad good
##              bad  699  296
##              good   1    4
## [1] 4
## Accuracy for a threshold of  0.9 is 0.703
```

```r
par(mfrow = c(2, 1))
plot(th, ac, xlab = "Threshold", ylab = "Accuracy", main="Accuracy vs Threshold", type="l", col="red")

plot(th, cost, xlab = "Threshold", ylab = "Cost", main="Cost vs Threshold", type="l", col="blue")
```



Accuracy vs Threshold
Accuracy
0.50
0.2
0.4
0.6
0.8
Threshold



Cost vs Threshold
Cost
500
0.2
0.4
0.6
0.8
Threshold