HW 10.1

```
rm(list=ls())

crimedata <- read.table("C:\\Users\\Adnan Karim\\Documents\\ISYE 6501\\Homeworks\\hw6-SP22\\data 9.1\\uscrime.txt", stringsAsFactors = F, header =  T)

head(crimedata)
```

```
##      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
```

```
library(tree)

#here i am importing the data into R and displaying it
#now i will build the regression tree model

crimedatatree <- tree(Crime~., data = crimedata)

summary(crimedatatree)
```

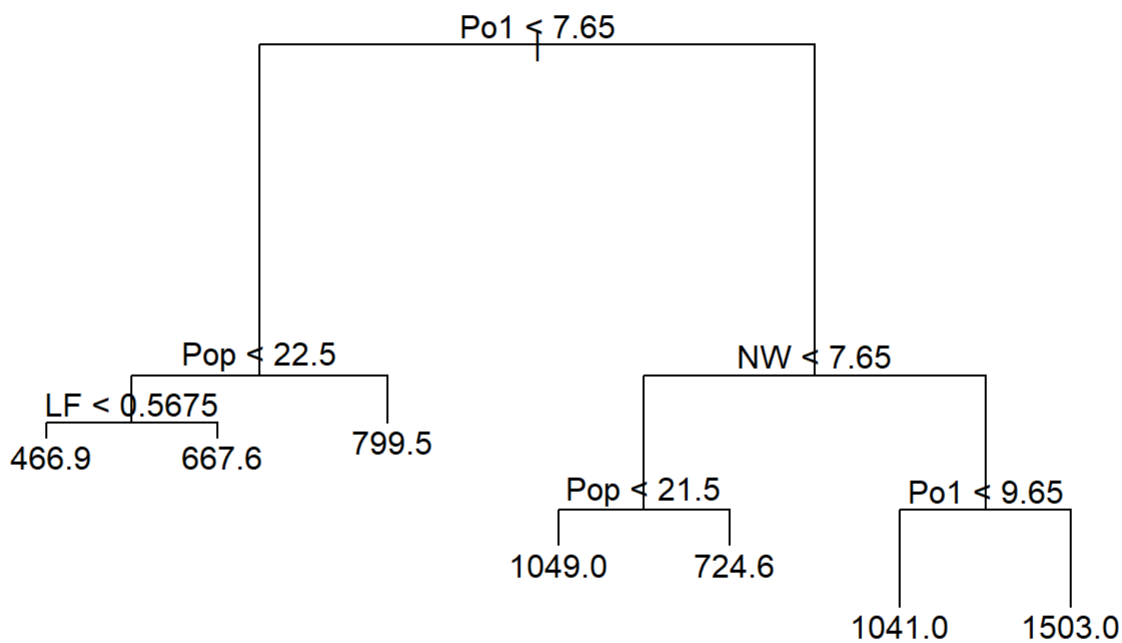
```
##
## Regression tree:
## tree(formula = Crime ~ ., data = crimedata)
## Variables actually used in tree construction:
## [1] "Po1" "Pop" "LF"  "NW"
## Number of terminal nodes:  7
## Residual mean deviance:  47390 = 1896000 / 40
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -573.900  -98.300   -1.545    0.000  110.600  490.100
```

```
#we can now use the frame callout to see how this tree was split up

crimedatatree$frame
```

```
##        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
```

```
plot(crimedatatree)
text(crimedatatree)
```

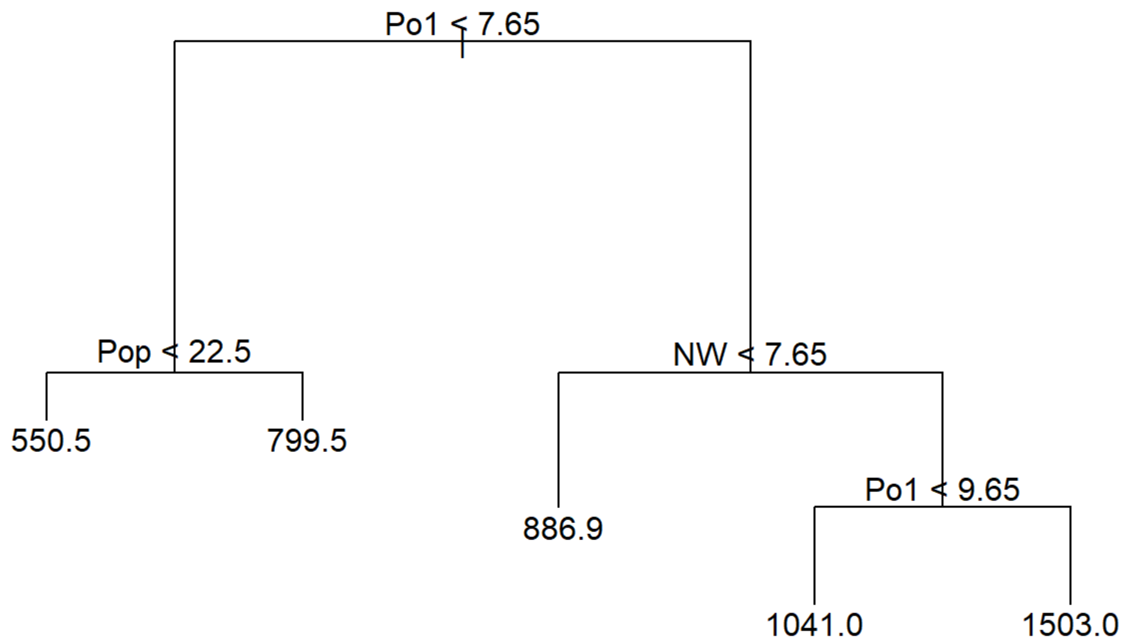


```
#we can now prune this tree by setting the nodes to 5

nodes <- 5

crimedatatreeprune <- prune.tree(crimedatatree, best = nodes)

plot(crimedatatreeprune)
text(crimedatatreeprune)
```




```
summary(crimedatatreeprune)
```

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

```
#interpreting this data shows that 5 nodes is leading to more error and overfitting in this mode
l, the original 7 nodes works just fine

nodes2 <- 7

crimedatatreeprune2 <- prune.tree(crimedatatree, best = nodes2)

plot(crimedatatreeprune2)
text(crimedatatreeprune2)
summary(crimedatatreeprune2)
```

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

```
#now we can calculate r2 to see the quality of this model

crimedataprediction <- predict(crimedatatreeprune2, data= crimedata[,1:15])
rss <- sum((crimedataprediction - crimedata[,16])^2)
tss <- sum((crimedata[,16]- mean(crimedata[,16]))^2)

r2 <- 1- rss/tss

r2
```

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

```
# the r2 at 0.724 is not the best, so even this model has a degree of over fitting


# now we can do a random forest model and compare

library(randomForest)
```

```
## randomForest 4.7-1
```

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

```
crimforest <- randomForest(Crime~. , data = crimedata, importance = T, nodesize = 5)

crimeforpredict <- predict(crimforest, data = crimedata[,-16])

rss2 <- sum((crimeforpredict - crimedata[,16])^2)

r2new <- 1 - rss2/tss

r2new
```

```
## [1] 0.4417235
```

```
# the r2 value with the forest method is even worse than the single tree , so it is not as accur
ate

#### 10.2

#I can use a logistic regression model in my job to predict the probability that a certain price
of a product (chemical) would sell in the open market. So things like volume, quality, market pr
ice could all be used as predictors.

#### 10.3

gcred <- read.table("C:\\Users\\Adnan Karim\\Documents\\ISYE 6501\\Homeworks\\hw7-SP22\\data 10.
3\\germancredit.txt", stringsAsFactors = F, header =  F)

library(caret)
```

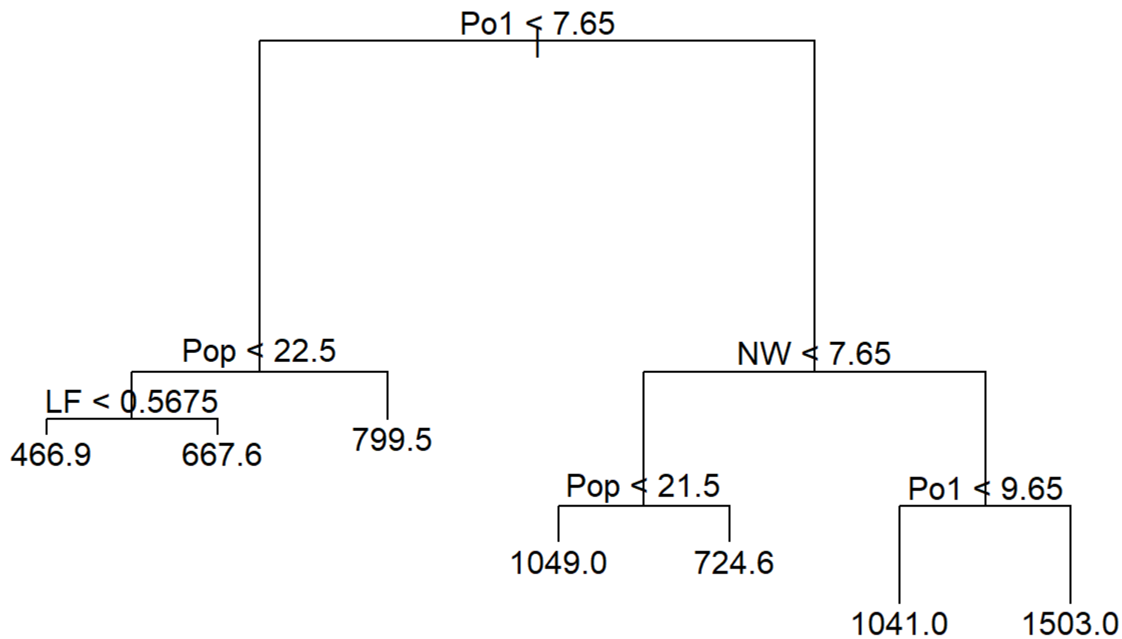
```
## Loading required package: ggplot2
```

```
## Warning in register(): Can't find generic `scale_type` in package ggplot2 to
## register S3 method.
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

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




```
#caret is so we can use the data partition function later

head(gcred)
```

```
##    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
```

```
str(gcred)
```

```
## 'data.frame':    1000 obs. of  21 variables:
##  $ V1 : chr  "A11" "A12" "A14" "A11" ...
##  $ V2 : int  6 48 12 42 24 36 24 36 12 30 ...
##  $ V3 : chr  "A34" "A32" "A34" "A32" ...
##  $ V4 : chr  "A43" "A43" "A46" "A42" ...
##  $ V5 : int  1169 5951 2096 7882 4870 9055 2835 6948 3059 5234 ...
##  $ V6 : chr  "A65" "A61" "A61" "A61" ...
##  $ V7 : chr  "A75" "A73" "A74" "A74" ...
##  $ V8 : int  4 2 2 2 3 2 3 2 2 4 ...
##  $ V9 : chr  "A93" "A92" "A93" "A93" ...
##  $ V10: chr  "A101" "A101" "A101" "A103" ...
##  $ V11: int  4 2 3 4 4 4 4 2 4 2 ...
##  $ V12: chr  "A121" "A121" "A121" "A122" ...
##  $ V13: int  67 22 49 45 53 35 53 35 61 28 ...
##  $ V14: chr  "A143" "A143" "A143" "A143" ...
##  $ V15: chr  "A152" "A152" "A152" "A153" ...
##  $ V16: int  2 1 1 1 2 1 1 1 1 2 ...
##  $ V17: chr  "A173" "A173" "A172" "A173" ...
##  $ V18: int  1 1 2 2 2 2 1 1 1 1 ...
##  $ V19: chr  "A192" "A191" "A191" "A191" ...
##  $ V20: chr  "A201" "A201" "A201" "A201" ...
##  $ V21: int  1 2 1 1 2 1 1 1 1 2 ...
```

```
#variable 21 contains the responses we want so we need to scale 1 & 2 with 0 & 1

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

# we want to split the model into training and validation sets so we can compute truepositive +
 truenegative / (total predictions + actual)

gcredspl <- createDataPartition(gcred$V21, times = 1, p = 0.7, list= F)

head(gcredspl)
```

```
##      Resample1
## [1,]         1
## [2,]         4
## [3,]         5
## [4,]         6
## [5,]         7
## [6,]         8
```

```
#create training and validation sets

gt <- gcred[gcredspl,]

gv <- gcred[-gcredspl,]

table(gt$V21)
```

```
##
##   0   1
## 480 220
```

```
table(gv$V21)
```

```
##
##   0   1
## 220  80
```

```
#show both sets

#now we can run the log model with glm

glogm <- glm(V21~., data = gt, family = binomial(link = "logit"))

summary(glogm)
```

```
##
## Call:
## glm(formula = V21 ~ ., family = binomial(link = "logit"), data = gt)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9416  -0.7180  -0.3333   0.7217   2.7415
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  7.706e-01  1.291e+00   0.597 0.550461
## V1A12       -6.201e-01  2.715e-01  -2.283 0.022401 *
## V1A13       -1.250e+00  4.730e-01  -2.644 0.008203 **
## V1A14       -1.686e+00  2.806e-01  -6.009 1.87e-09 ***
## V2           3.127e-02  1.124e-02   2.783 0.005385 **
## V3A31        1.550e-01  6.753e-01   0.230 0.818453
## V3A32       -4.830e-01  5.019e-01  -0.962 0.335895
## V3A33       -3.302e-01  5.485e-01  -0.602 0.547138
## V3A34       -1.026e+00  5.074e-01  -2.022 0.043212 *
## V4A41       -2.170e+00  4.890e-01  -4.438 9.09e-06 ***
## V4A410      -3.536e+00  1.190e+00  -2.972 0.002958 **
## V4A42       -8.265e-01  3.183e-01  -2.596 0.009422 **
## V4A43       -8.854e-01  2.984e-01  -2.968 0.003002 **
## V4A44       -3.845e-01  8.537e-01  -0.450 0.652388
## V4A45        1.917e-01  6.398e-01   0.300 0.764486
## V4A46        1.766e-01  4.686e-01   0.377 0.706313
## V4A48       -2.113e+00  1.343e+00  -1.573 0.115676
## V4A49       -9.190e-01  4.045e-01  -2.272 0.023114 *
## V5           1.602e-04  5.432e-05   2.949 0.003188 **
## V6A62       -1.445e-01  3.286e-01  -0.440 0.660123
## V6A63       -8.399e-01  5.249e-01  -1.600 0.109563
## V6A64       -1.476e+00  7.412e-01  -1.991 0.046437 *
## V6A65       -1.063e+00  3.238e-01  -3.283 0.001029 **
## V7A72       -1.147e-02  4.996e-01  -0.023 0.981680
## V7A73       -3.296e-01  4.806e-01  -0.686 0.492899
## V7A74       -5.868e-01  5.205e-01  -1.127 0.259609
## V7A75       -1.608e-01  4.926e-01  -0.326 0.744074
## V8           3.818e-01  1.067e-01   3.577 0.000348 ***
## V9A92       -4.217e-01  4.522e-01  -0.933 0.351025
## V9A93       -1.145e+00  4.433e-01  -2.582 0.009813 **
## V9A94       -4.048e-01  5.389e-01  -0.751 0.452585
## V10A102      8.723e-01  5.433e-01   1.606 0.108359
## V10A103     -1.303e+00  5.426e-01  -2.401 0.016334 *
## V11         -3.881e-02  1.035e-01  -0.375 0.707785
## V12A122      3.513e-01  3.112e-01   1.129 0.259076
## V12A123      3.237e-01  2.835e-01   1.142 0.253499
## V12A124      7.987e-01  4.979e-01   1.604 0.108680
## V13         -1.568e-02  1.169e-02  -1.342 0.179698
## V14A142     -8.240e-01  5.104e-01  -1.615 0.106416
## V14A143     -8.801e-01  2.840e-01  -3.099 0.001943 **
## V15A152     -2.719e-01  2.805e-01  -0.969 0.332378
## V15A153     -5.632e-01  5.853e-01  -0.962 0.335927
```

```
## V16           1.302e-01  2.289e-01   0.569 0.569564
## V17A172       2.240e-01  8.158e-01   0.275 0.783680
## V17A173       3.317e-01  7.781e-01   0.426 0.669939
## V17A174      -4.800e-03  7.942e-01  -0.006 0.995178
## V18           2.604e-01  3.008e-01   0.865 0.386765
## V19A192      -6.539e-02  2.412e-01  -0.271 0.786272
## V20A202      -1.010e+00  7.727e-01  -1.307 0.191262
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 871.48  on 699  degrees of freedom
## Residual deviance: 620.52  on 651  degrees of freedom
## AIC: 718.52
##
## Number of Fisher Scoring iterations: 5
```

```
#now we can use our model to predict the number of good/bad and see how good the model is

gcredpred <- predict(glogm, newdata = gv[,-21], type = "response")

table(gv$V21, round(gcredpred))
```

```
##
##       0   1
##   0 193  27
##   1  45  35
```

```
#now we can create a confusion matrix so that we can then set a threshold to hopefully get bette
r results i.e. less values that say incorrectly the data is bad credit etc.

gcredpredv2 <- predict(glogm, newdata = gv[-21], type = "response")
mat <- as.matrix(table(round(gcredpredv2), gv$V21))
names(dimnames(mat)) <- c("Predict", "observe")

mat
```

```
##        observe
## Predict   0   1
##       0 193  45
##       1  27  35
```

```
threshold <- 0.7

mat2 <- as.matrix(table(round(gcredpredv2>threshold), gv$V21))
names(dimnames(mat)) <- c("predict", "observe")

mat2
```

```
##
##       0   1
##   0 212  63
##   1   8  17
```

```
#with the threshold we have less incorrect classifications so this is good!

#now we can test accuracy

accuracy <- (mat2[1,1]+mat2[2,2])/(mat2[1,1]+mat2[1,2]+mat2[2,1]+mat2[2,2])
accuracy
```

```
## [1] 0.7633333
```

```
#so 0.7 is a pretty good threshold as it gives us 72% accuracy
```