HW7

Question 10.1

Using the same crime data set uscrime.txt as in Questions 8.2 and 9.1, find the best model you can using (a) a regression tree model, and (b) a random forest model.  
In R, you can use the tree package or the rpart package, and the randomForest package. For each model, describe one or two qualitative takeaways you get from analyzing the results (i.e., don’t just stop when you have a good model, but interpret it too).

library(rpart)  
library(rpart.plot)  
library(randomForest)

## randomForest 4.6-14

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

library(rsample)   
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following object is masked from 'package:randomForest':  
##   
## combine

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(rpart.plot)   
library(ipred)   
library(caret)

## Loading required package: ggplot2

##   
## Attaching package: 'ggplot2'

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

## Loading required package: lattice

Bring in the data after clearing memory

rm(list = ls())  
uscrime <- read.table("uscrime.txt", stringsAsFactors = FALSE, header = TRUE)  
summary(uscrime)

## M So Ed Po1   
## Min. :11.90 Min. :0.0000 Min. : 8.70 Min. : 4.50   
## 1st Qu.:13.00 1st Qu.:0.0000 1st Qu.: 9.75 1st Qu.: 6.25   
## Median :13.60 Median :0.0000 Median :10.80 Median : 7.80   
## Mean :13.86 Mean :0.3404 Mean :10.56 Mean : 8.50   
## 3rd Qu.:14.60 3rd Qu.:1.0000 3rd Qu.:11.45 3rd Qu.:10.45   
## Max. :17.70 Max. :1.0000 Max. :12.20 Max. :16.60   
## Po2 LF M.F Pop   
## Min. : 4.100 Min. :0.4800 Min. : 93.40 Min. : 3.00   
## 1st Qu.: 5.850 1st Qu.:0.5305 1st Qu.: 96.45 1st Qu.: 10.00   
## Median : 7.300 Median :0.5600 Median : 97.70 Median : 25.00   
## Mean : 8.023 Mean :0.5612 Mean : 98.30 Mean : 36.62   
## 3rd Qu.: 9.700 3rd Qu.:0.5930 3rd Qu.: 99.20 3rd Qu.: 41.50   
## Max. :15.700 Max. :0.6410 Max. :107.10 Max. :168.00   
## NW U1 U2 Wealth   
## Min. : 0.20 Min. :0.07000 Min. :2.000 Min. :2880   
## 1st Qu.: 2.40 1st Qu.:0.08050 1st Qu.:2.750 1st Qu.:4595   
## Median : 7.60 Median :0.09200 Median :3.400 Median :5370   
## Mean :10.11 Mean :0.09547 Mean :3.398 Mean :5254   
## 3rd Qu.:13.25 3rd Qu.:0.10400 3rd Qu.:3.850 3rd Qu.:5915   
## Max. :42.30 Max. :0.14200 Max. :5.800 Max. :6890   
## Ineq Prob Time Crime   
## Min. :12.60 Min. :0.00690 Min. :12.20 Min. : 342.0   
## 1st Qu.:16.55 1st Qu.:0.03270 1st Qu.:21.60 1st Qu.: 658.5   
## Median :17.60 Median :0.04210 Median :25.80 Median : 831.0   
## Mean :19.40 Mean :0.04709 Mean :26.60 Mean : 905.1   
## 3rd Qu.:22.75 3rd Qu.:0.05445 3rd Qu.:30.45 3rd Qu.:1057.5   
## Max. :27.60 Max. :0.11980 Max. :44.00 Max. :1993.0

Run the model, and let’s take a look

set.seed(1)  
tree\_model <-rpart(Crime ~ ., data = uscrime)  
summary(tree\_model)

## Call:  
## rpart(formula = Crime ~ ., data = uscrime)  
## n= 47   
##   
## CP nsplit rel error xerror xstd  
## 1 0.36296293 0 1.0000000 1.0303899 0.2549076  
## 2 0.14814320 1 0.6370371 0.8900680 0.2149365  
## 3 0.05173165 2 0.4888939 0.9096979 0.2393384  
## 4 0.01000000 3 0.4371622 0.8893049 0.2346618  
##   
## Variable importance  
## Po1 Po2 Wealth Ineq Prob M NW Pop Time Ed LF   
## 17 17 11 11 10 10 9 5 4 4 1   
## So   
## 1   
##   
## Node number 1: 47 observations, complexity param=0.3629629  
## mean=905.0851, MSE=146402.7   
## left son=2 (23 obs) right son=3 (24 obs)  
## Primary splits:  
## Po1 < 7.65 to the left, improve=0.3629629, (0 missing)  
## Po2 < 7.2 to the left, improve=0.3629629, (0 missing)  
## Prob < 0.0418485 to the right, improve=0.3217700, (0 missing)  
## NW < 7.65 to the left, improve=0.2356621, (0 missing)  
## Wealth < 6240 to the left, improve=0.2002403, (0 missing)  
## Surrogate splits:  
## Po2 < 7.2 to the left, agree=1.000, adj=1.000, (0 split)  
## Wealth < 5330 to the left, agree=0.830, adj=0.652, (0 split)  
## Prob < 0.043598 to the right, agree=0.809, adj=0.609, (0 split)  
## M < 13.25 to the right, agree=0.745, adj=0.478, (0 split)  
## Ineq < 17.15 to the right, agree=0.745, adj=0.478, (0 split)  
##   
## Node number 2: 23 observations, complexity param=0.05173165  
## mean=669.6087, MSE=33880.15   
## left son=4 (12 obs) right son=5 (11 obs)  
## Primary splits:  
## Pop < 22.5 to the left, improve=0.4568043, (0 missing)  
## M < 14.5 to the left, improve=0.3931567, (0 missing)  
## NW < 5.4 to the left, improve=0.3184074, (0 missing)  
## Po1 < 5.75 to the left, improve=0.2310098, (0 missing)  
## U1 < 0.093 to the right, improve=0.2119062, (0 missing)  
## Surrogate splits:  
## NW < 5.4 to the left, agree=0.826, adj=0.636, (0 split)  
## M < 14.5 to the left, agree=0.783, adj=0.545, (0 split)  
## Time < 22.30055 to the left, agree=0.783, adj=0.545, (0 split)  
## So < 0.5 to the left, agree=0.739, adj=0.455, (0 split)  
## Ed < 10.85 to the right, agree=0.739, adj=0.455, (0 split)  
##   
## Node number 3: 24 observations, complexity param=0.1481432  
## mean=1130.75, MSE=150173.4   
## left son=6 (10 obs) right son=7 (14 obs)  
## Primary splits:  
## NW < 7.65 to the left, improve=0.2828293, (0 missing)  
## M < 13.05 to the left, improve=0.2714159, (0 missing)  
## Time < 21.9001 to the left, improve=0.2060170, (0 missing)  
## M.F < 99.2 to the left, improve=0.1703438, (0 missing)  
## Po1 < 10.75 to the left, improve=0.1659433, (0 missing)  
## Surrogate splits:  
## Ed < 11.45 to the right, agree=0.750, adj=0.4, (0 split)  
## Ineq < 16.25 to the left, agree=0.750, adj=0.4, (0 split)  
## Time < 21.9001 to the left, agree=0.750, adj=0.4, (0 split)  
## Pop < 30 to the left, agree=0.708, adj=0.3, (0 split)  
## LF < 0.5885 to the right, agree=0.667, adj=0.2, (0 split)  
##   
## Node number 4: 12 observations  
## mean=550.5, MSE=20317.58   
##   
## Node number 5: 11 observations  
## mean=799.5455, MSE=16315.52   
##   
## Node number 6: 10 observations  
## mean=886.9, MSE=55757.49   
##   
## Node number 7: 14 observations  
## mean=1304.929, MSE=144801.8

If you take a look at: Node number 4: 12 observations mean=550.5, MSE=20317.58

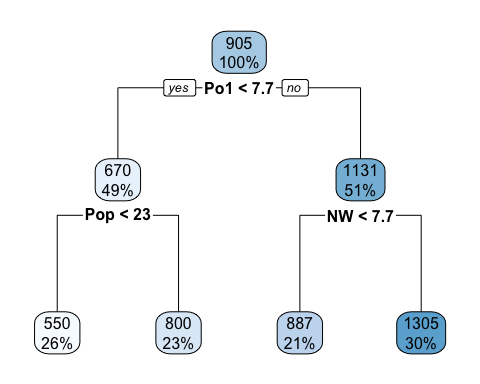
Node number 5: 11 observations mean=799.5455, MSE=16315.52

Node number 6: 10 observations mean=886.9, MSE=55757.49

Node number 7: 14 observations mean=1304.929, MSE=144801.8

You can see that the model tried out all of the above and chose the model with only 4 leaves. Rpart package does pruning along with tuning in the backround. With the lowest real error of 0.437162 and a MSE of 20317.58.

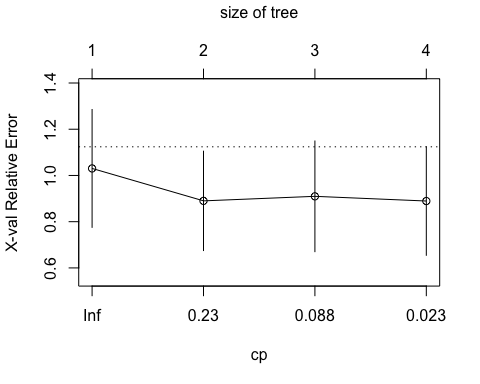
rpart.plot(tree\_model)

 Its important to note that the model initialy split the data on Po1 because of the large variance that it would be cutting down on. This is follow Pop and NW. I would like to point out that the prediction coming out of NW is probably going to be more accurate from the rest.

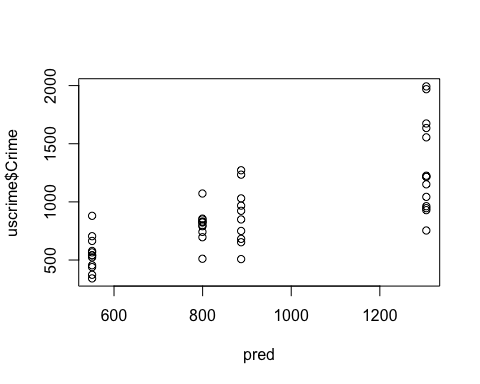
tree\_model$frame

## var n wt dev yval complexity ncompete nsurrogate  
## 1 Po1 47 47 6880927.7 905.0851 0.36296293 4 5  
## 2 Pop 23 23 779243.5 669.6087 0.05173165 4 5  
## 4 <leaf> 12 12 243811.0 550.5000 0.01000000 0 0  
## 5 <leaf> 11 11 179470.7 799.5455 0.01000000 0 0  
## 3 NW 24 24 3604162.5 1130.7500 0.14814320 4 5  
## 6 <leaf> 10 10 557574.9 886.9000 0.01000000 0 0  
## 7 <leaf> 14 14 2027224.9 1304.9286 0.01000000 0 0

plotcp(tree\_model)



pred <-predict(tree\_model)  
plot(pred,uscrime$Crime)



Here is the R^2 for basic regresion tree

ssres <-sum((pred-uscrime$Crime)^2)  
totalss <-sum((uscrime$Crime-mean(uscrime$Crime))^2)  
R\_square <-1-(ssres/totalss)  
R\_square

## [1] 0.5628378

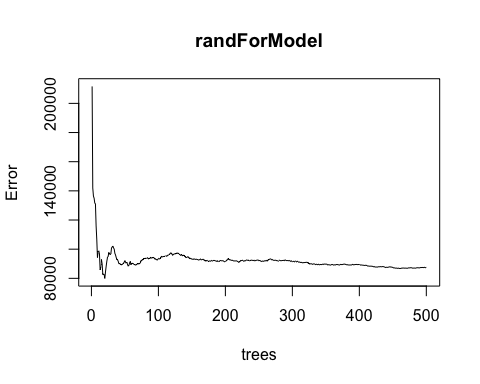
Here we are running the random forest model. This blanket random forest model can account for 40% of the data. Also one should not that the random forest package tried 5 different splits at each branch to optimize model.

set.seed(1)  
randForModel <-randomForest(Crime ~ ., data = uscrime)  
randForModel

##   
## Call:  
## randomForest(formula = Crime ~ ., data = uscrime)   
## Type of random forest: regression  
## Number of trees: 500  
## No. of variables tried at each split: 5  
##   
## Mean of squared residuals: 87461.6  
## % Var explained: 40.26

At 20 trees we have a low error rate, so im going to take that value and plug it into another random forest model to see if it gives a better outcome. Perhaps with a lower Mean of squared residuals alond with a higher % Var explained.

plot(randForModel)



which.min(randForModel$mse)

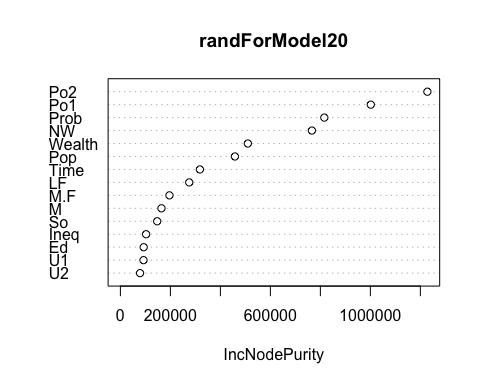
## [1] 20

This model gives us a lower Mean of squared residual and a higher % Var explained. Which is what we were hoping for.

set.seed(1)  
randForModel20 <-randomForest(Crime ~ ., data = uscrime, ntree=20)  
randForModel20

##   
## Call:  
## randomForest(formula = Crime ~ ., data = uscrime, ntree = 20)   
## Type of random forest: regression  
## Number of trees: 20  
## No. of variables tried at each split: 5  
##   
## Mean of squared residuals: 79954.56  
## % Var explained: 45.39

varImpPlot(randForModel20)



importance(randForModel20)

## IncNodePurity  
## M 164305.21  
## So 147172.54  
## Ed 93200.17  
## Po1 1001756.80  
## Po2 1228206.84  
## LF 275426.15  
## M.F 196282.05  
## Pop 458054.69  
## NW 766053.93  
## U1 92423.59  
## U2 78617.68  
## Wealth 509884.36  
## Ineq 102874.77  
## Prob 815847.72  
## Time 317996.81

predictTest<-predict(randForModel20, data = uscrime$Crime)  
predictTest

## 1 2 3 4 5 6 7 8   
## 859.5667 1249.6429 507.4333 1372.6963 985.8896 556.3333 1171.0917 1130.6717   
## 9 10 11 12 13 14 15 16   
## 814.1278 694.9727 1266.2381 700.3883 698.8800 742.4048 708.9200 916.9333   
## 17 18 19 20 21 22 23 24   
## 684.8519 1189.1200 1213.5000 1008.4667 717.7438 692.7600 1241.2929 709.6333   
## 25 26 27 28 29 30 31 32   
## 621.8417 1197.5583 815.7455 1072.8500 1440.9250 795.5500 744.9667 1233.5593   
## 33 34 35 36 37 38 39 40   
## 731.7071 892.5933 1012.2563 952.9650 702.2000 543.4708 711.9000 1277.7500   
## 41 42 43 44 45 46 47   
## 742.7000 691.9347 868.0417 890.4056 637.1600 1009.7306 1082.2407

randForestPred <-predict(randForModel20)  
ssres <-sum((randForestPred-uscrime$Crime)^2)  
totalss <-sum((uscrime$Crime-mean(uscrime$Crime))^2)  
R\_square <-1-(ssres/totalss)  
R\_square

## [1] 0.4538725

Question 10.2

Describe a situation or problem from your job, everyday life, current events, etc., for which a logistic regression model would be appropriate. List some (up to 5) predictors that you might use.

logistic regression model would be appropriate for a bank analyst working along side a loan officer. Trying to figure out to whom to give loans too.

This could also be useful at an admissions office to determine which students to accept.

germancredit <- read.table("germancredit.txt", header = FALSE)   
head(germancredit)

## 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

library(pROC)

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

##   
## Attaching package: 'pROC'

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

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

training <-sample(nrow(germancredit),0.8 \* nrow(germancredit), replace = FALSE)  
germancredit\_training <-germancredit[training, ]  
germancredit\_test <-germancredit[-training, ]

set.seed(1)  
model <-glm(V21~.,family =binomial(link ="logit"),data =germancredit\_training)  
summary(model)

##   
## Call:  
## glm(formula = V21 ~ ., family = binomial(link = "logit"), data = germancredit\_training)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.2948 -0.7060 -0.3538 0.7264 2.7216   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.225e+00 1.351e+00 -0.907 0.364464   
## V1A12 -3.146e-01 2.417e-01 -1.302 0.192954   
## V1A13 -9.779e-01 4.067e-01 -2.405 0.016181 \*   
## V1A14 -1.809e+00 2.655e-01 -6.814 9.48e-12 \*\*\*  
## V2 3.234e-02 1.057e-02 3.061 0.002207 \*\*   
## V3A31 4.125e-01 6.427e-01 0.642 0.521020   
## V3A32 -4.984e-01 5.015e-01 -0.994 0.320317   
## V3A33 -8.790e-01 5.487e-01 -1.602 0.109173   
## V3A34 -1.193e+00 5.076e-01 -2.351 0.018728 \*   
## V4A41 -1.837e+00 4.309e-01 -4.263 2.01e-05 \*\*\*  
## V4A410 -1.183e+00 8.633e-01 -1.371 0.170528   
## V4A42 -7.286e-01 2.863e-01 -2.545 0.010918 \*   
## V4A43 -8.990e-01 2.739e-01 -3.282 0.001030 \*\*   
## V4A44 -4.186e-01 8.959e-01 -0.467 0.640348   
## V4A45 8.335e-01 6.919e-01 1.205 0.228386   
## V4A46 -3.170e-01 4.757e-01 -0.666 0.505140   
## V4A48 -1.985e+00 1.293e+00 -1.535 0.124726   
## V4A49 -6.346e-01 3.947e-01 -1.608 0.107910   
## V5 1.033e-04 4.999e-05 2.067 0.038742 \*   
## V6A62 -3.882e-01 3.172e-01 -1.224 0.221062   
## V6A63 -7.996e-02 4.227e-01 -0.189 0.849953   
## V6A64 -2.089e+00 7.445e-01 -2.807 0.005008 \*\*   
## V6A65 -1.091e+00 2.993e-01 -3.644 0.000269 \*\*\*  
## V7A72 3.181e-01 4.882e-01 0.652 0.514665   
## V7A73 1.502e-01 4.684e-01 0.321 0.748507   
## V7A74 -4.869e-01 5.110e-01 -0.953 0.340743   
## V7A75 2.492e-02 4.718e-01 0.053 0.957881   
## V8 2.940e-01 9.820e-02 2.994 0.002753 \*\*   
## V9A92 1.336e-01 4.283e-01 0.312 0.755086   
## V9A93 -6.705e-01 4.255e-01 -1.576 0.115047   
## V9A94 8.063e-02 5.040e-01 0.160 0.872902   
## V10A102 3.181e-01 4.490e-01 0.708 0.478725   
## V10A103 -1.210e+00 4.864e-01 -2.488 0.012828 \*   
## V11 1.345e-02 9.607e-02 0.140 0.888616   
## V12A122 4.461e-01 2.860e-01 1.560 0.118800   
## V12A123 1.160e-01 2.641e-01 0.439 0.660548   
## V12A124 5.710e-01 5.232e-01 1.092 0.275041   
## V13 -1.084e-02 1.060e-02 -1.022 0.306880   
## V14A142 2.924e-01 4.747e-01 0.616 0.537938   
## V14A143 -3.572e-01 2.759e-01 -1.295 0.195305   
## V15A152 -4.810e-01 2.644e-01 -1.820 0.068826 .   
## V15A153 -4.683e-01 5.715e-01 -0.819 0.412529   
## V16 1.838e-01 2.167e-01 0.848 0.396204   
## V17A172 1.218e+00 8.691e-01 1.401 0.161127   
## V17A173 1.413e+00 8.496e-01 1.663 0.096359 .   
## V17A174 1.233e+00 8.702e-01 1.417 0.156384   
## V18 2.697e-01 2.854e-01 0.945 0.344736   
## V19A192 -2.692e-01 2.275e-01 -1.183 0.236616   
## V20A202 -1.572e+00 7.159e-01 -2.196 0.028119 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 988.95 on 799 degrees of freedom  
## Residual deviance: 713.69 on 751 degrees of freedom  
## AIC: 811.69  
##   
## Number of Fisher Scoring iterations: 5

preds <-predict(model, germancredit\_test,type = "response")

roc\_val <-roc(germancredit\_test$V21, round(preds))

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

## Setting direction: controls < cases

roc\_val

##   
## Call:  
## roc.default(response = germancredit\_test$V21, predictor = round(preds))  
##   
## Data: round(preds) in 147 controls (germancredit\_test$V21 0) < 53 cases (germancredit\_test$V21 1).  
## Area under the curve: 0.6678

In this data set, they estimate that incorrectly identifying a bad customer as good, is 5 times worse than incorrectly classifying a good customer as bad. Determine a good threshold probability based on your model.

Cost of mislabeling customer $0 for correct classifiation $1 for incorrectly classifying a good customer as bad $5 for incorrectly identifying a bad customer as good

threshold8 <- 0.8  
pred\_threshold8 <- as.integer(preds > threshold8)

conf\_matrix8 <- as.matrix(table(pred\_threshold8,germancredit\_test$V21))  
conf\_matrix8

##   
## pred\_threshold8 0 1  
## 0 146 46  
## 1 1 7

costAt0.8 <- (146 \* 0) + (46 \* 1) + (1 \* 5) + (7 \* 0)  
costAt0.8

## [1] 51

threshold6 <- 0.6  
pred\_threshold6 <- as.integer(preds > threshold6)  
  
conf\_matrix6 <- as.matrix(table(pred\_threshold6,germancredit\_test$V21))  
conf\_matrix6

##   
## pred\_threshold6 0 1  
## 0 131 36  
## 1 16 17

costAt0.6 <- (131 \* 0) + (36 \* 1) + (16 \* 5) + (17 \* 0)  
costAt0.6

## [1] 116

threshold9 <- 0.9  
pred\_threshold9 <- as.integer(preds > threshold9)  
  
conf\_matrix9 <- as.matrix(table(pred\_threshold9,germancredit\_test$V21))  
conf\_matrix9

##   
## pred\_threshold9 0 1  
## 0 147 50  
## 1 0 3

WE got a winner, but only by one dollar.

costAt0.9 <- (147 \* 0) + (50 \* 1) + (0 \* 5) + (3 \* 0)  
costAt0.9

## [1] 50