HomeWork 7

ISYE 6501

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(kernlab)
library(kknn)
library(lattice)
library(ggplot2)

library(caret)# an aggregator package for performing many machine learning mo
dels

library(randomForest)

library(ranger)# a faster implementation of randomForest

library(h2o)# an extremely fast java-based platform

library(rpart)
library(tree)
library(corrplot) #graphical display of correlation matrix

library(rsample)      # data splitting
```

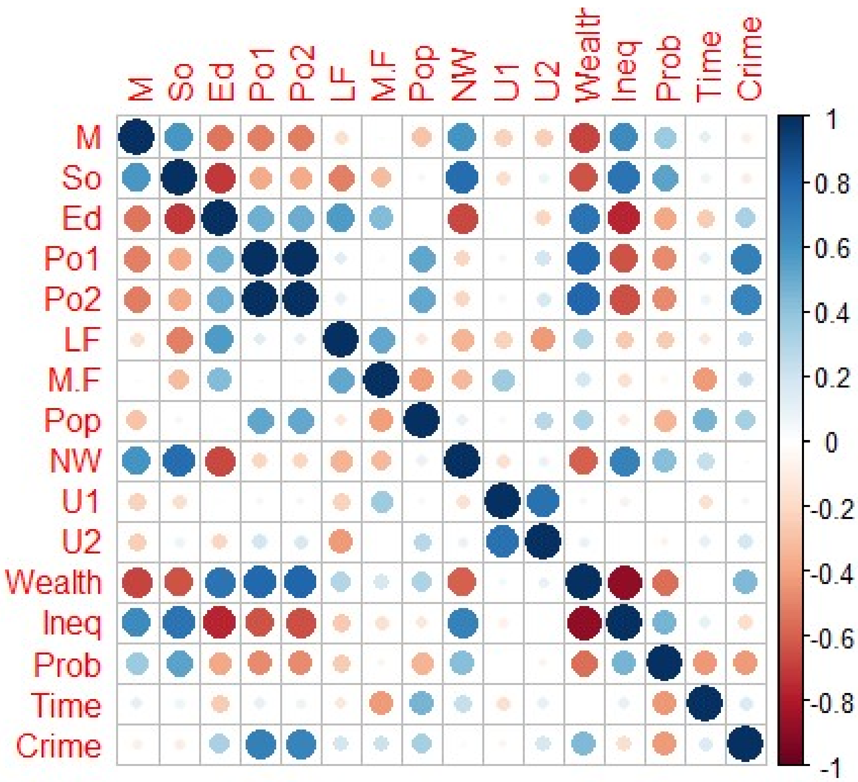
Next we will load the data and look at the data structure.

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

##      M So   Ed  Po1  Po2    LF   M.F Pop   NW    U1  U2 Wealth Ineq     Pr
ob
## 1 15.1  1  9.1  5.8  5.6 0.510  95.0  33 30.1 0.108 4.1   3940 26.1 0.0846
02
## 2 14.3  0 11.3 10.3  9.5 0.583 101.2  13 10.2 0.096 3.6   5570 19.4 0.0295
99
## 3 14.2  1  8.9  4.5  4.4 0.533  96.9  18 21.9 0.094 3.3   3180 25.0 0.0834
01
## 4 13.6  0 12.1 14.9 14.1 0.577  99.4 157  8.0 0.102 3.9   6730 16.7 0.0158
01
## 5 14.1  0 12.1 10.9 10.1 0.591  98.5  18  3.0 0.091 2.0   5780 17.4 0.0413
99
## 6 12.1  0 11.0 11.8 11.5 0.547  96.4  25  4.4 0.084 2.9   6890 12.6 0.0342
01
```

```
##      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
```

Lets examine the data for correlations using a visualisation. This will help us understand which features are most useful to us.

```
#uscrime$So <- NULL
#head(uscrime)
cormatrix <- cor(uscrime) #calculate correlation matrix
corrplot(cormatrix, method = "circle") #plot correlation matrix
```

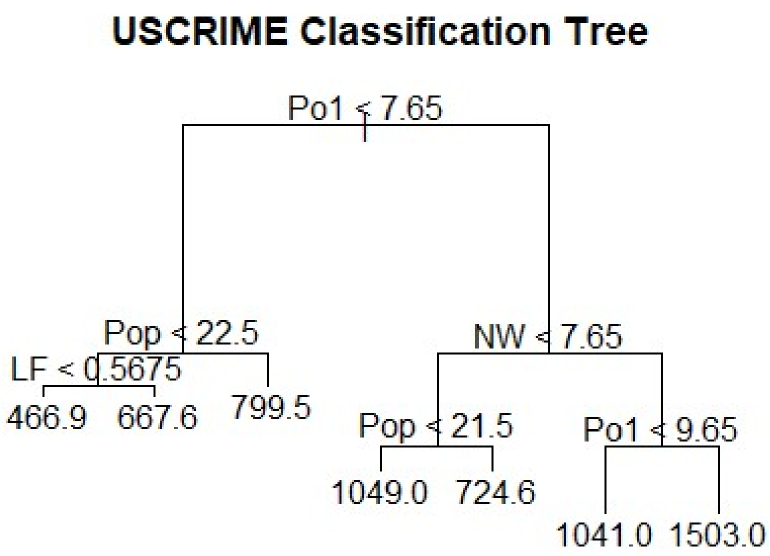


The correlation plot may not be needed here but it gives us an idea for which features to look out for tree branching. For instance Po1/Po2(but not both) are very important for Crime.

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

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

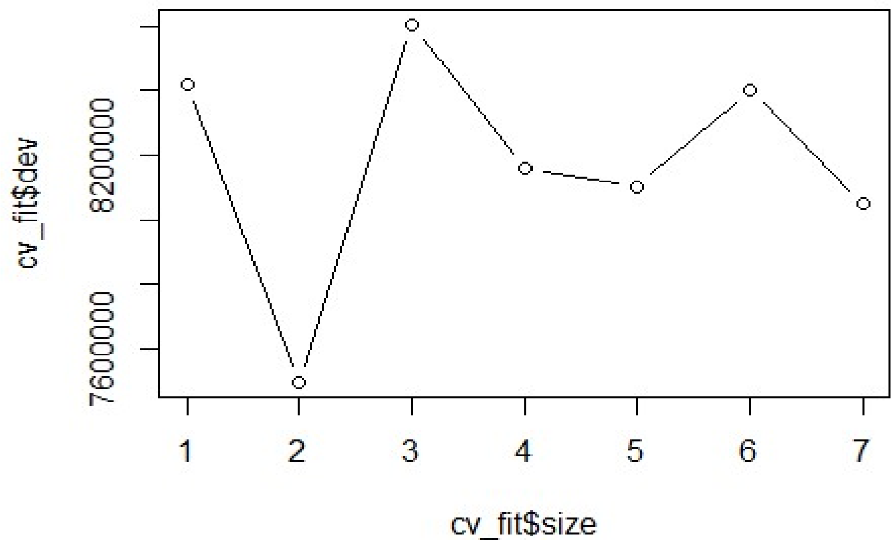
```r
plot(tree.uscrime)
text(tree.uscrime)
title("USCRIME Classification Tree")
```




```r
summary(tree.uscrime)

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

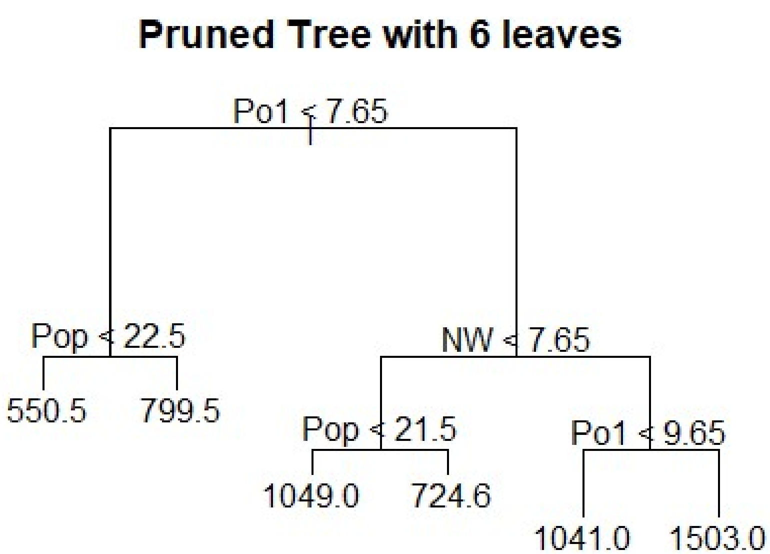
So we have created a tree with 7 leaves. But but we cant be certain if the tree above will give us the lowest error rate. Lets try doing some cross validation to see the number of splits we want.

```r
cv_fit = cv.tree(tree.uscrime)
plot(cv_fit$size, cv_fit$dev, type = 'b')
```

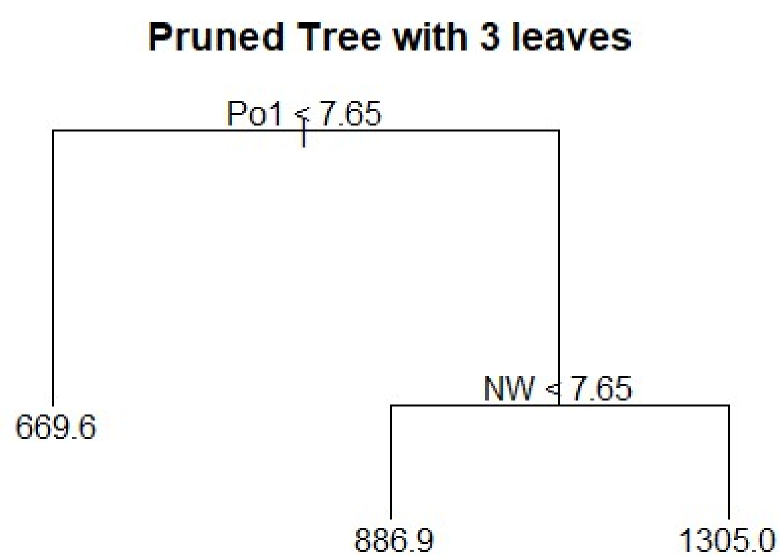


This indicates that it's best to use the terminal nodes 2,6 or 7, as it has the least amount of error. Lets get the summary to decide using the Residual mean deviance for the three models.

```
# Prune the tree with 5 nodes
library(RColorBrewer)
prune.tree6 <- prune.tree(tree.uscrime, best = 6)
plot(prune.tree6)
text(prune.tree6)
title("Pruned Tree with 6 leaves")
```



```r
summary(prune.tree6)

##
## Regression tree:
## snip.tree(tree = tree.uscrime, nodes = 4L)
## Variables actually used in tree construction:
## [1] "Po1" "Pop" "NW"
## Number of terminal nodes:  6
## Residual mean deviance:  49100 = 2013000 / 41
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -573.900  -99.520   -1.545    0.000  122.800  490.100

# Prune the tree with 3 nodes

prune.tree3 <- prune.tree(tree.uscrime, best = 3)
plot(prune.tree3)
text(prune.tree3)
title("Pruned Tree with 3 leaves")
```

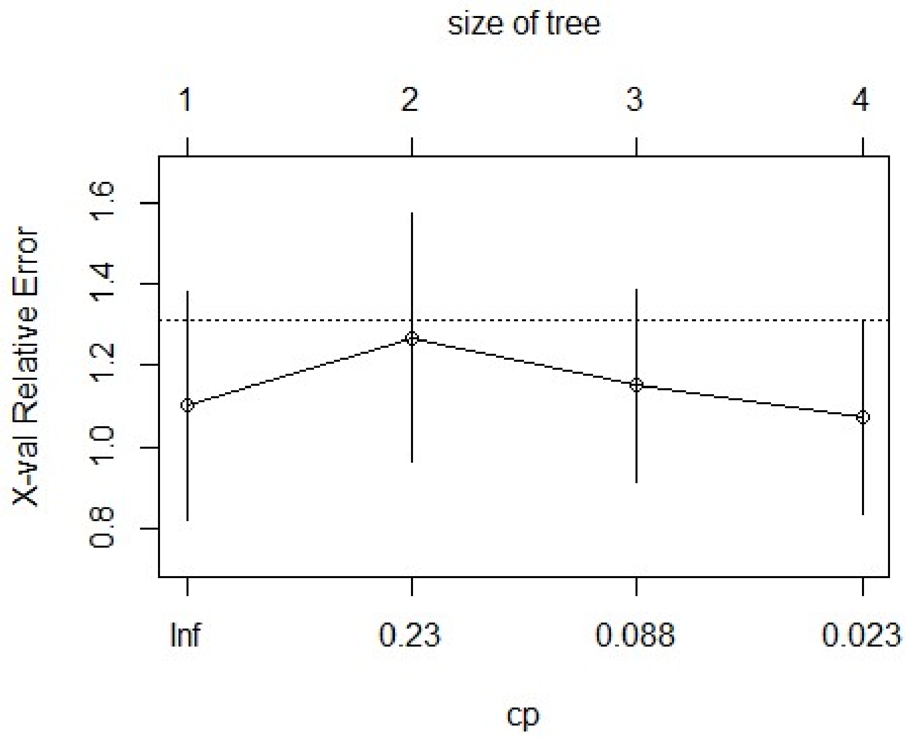



```r
summary(prune.tree3)

##
## Regression tree:
## snip.tree(tree = tree.uscrime, nodes = c(6L, 2L, 7L))
## Variables actually used in tree construction:
## [1] "Po1" "NW"
## Number of terminal nodes:  3
## Residual mean deviance:  76460 = 3364000 / 44
## Distribution of residuals:
```

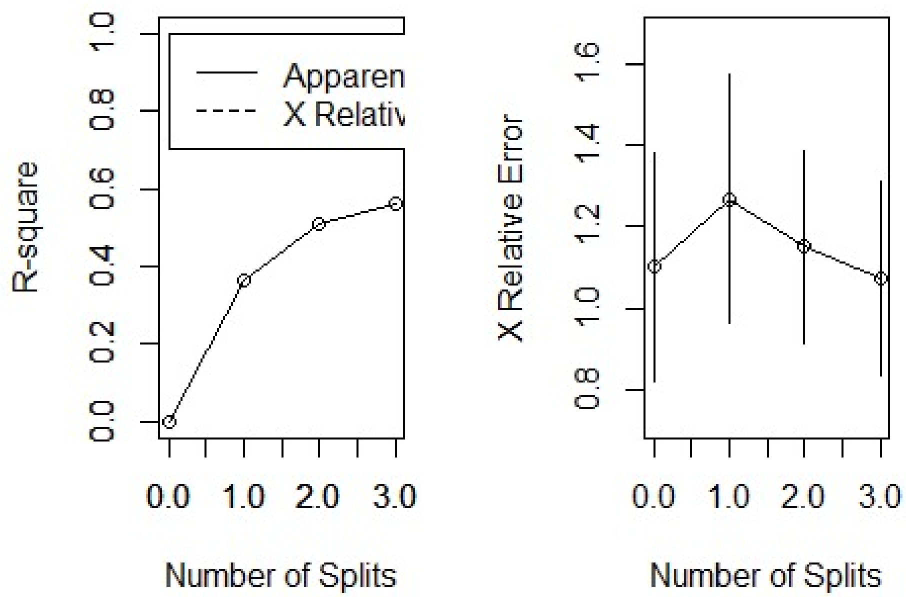
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -550.9  -181.8   -37.9     0.0   158.9   688.1
```

I actually ran the summary for all the models ranging from 2 nodes to 7. And the tree with the max node 7 had the least Residual mean deviance. Lets try to verify our observations by estimating our quality of fit using R2. I thought maybe using another tree model may yeild better results. I tried using rparts which gave a result with 4 nodes.

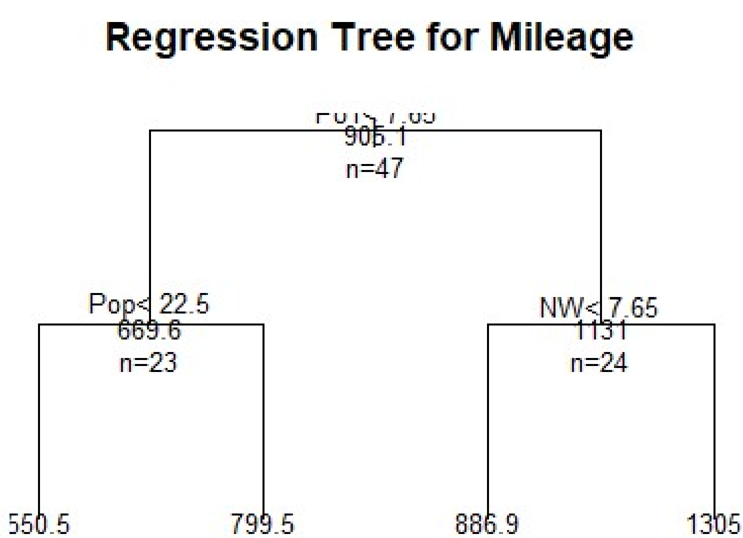
```
# Regression Tree Example
library(rpart)

# grow tree
fit <- rpart(Crime~.,
   method="anova", data=uscrime)

printcp(fit) # display the results

##
## Regression tree:
## rpart(formula = Crime ~ ., data = uscrime, method = "anova")
##
## Variables actually used in tree construction:
## [1] NW  Po1 Pop
##
## Root node error: 6880928/47 = 146403
##
## n= 47
##
##         CP nsplit rel error xerror    xstd
## 1 0.362963      0   1.00000 1.0999 0.28014
## 2 0.148143      1   0.63704 1.2679 0.30698
## 3 0.051732      2   0.48889 1.1508 0.23559
## 4 0.010000      3   0.43716 1.0710 0.23869

plotcp(fit) # visualize cross-validation results
```




```
#summary(fit) # detailed summary of splits

# create additional plots
par(mfrow=c(1,2)) # two plots on one page
rsq.rpart(fit) # visualize cross-validation results

##
## Regression tree:
## rpart(formula = Crime ~ ., data = uscrime, method = "anova")
##
## Variables actually used in tree construction:
## [1] NW  Po1 Pop
##
## Root node error: 6880928/47 = 146403
##
## n= 47
##
##         CP nsplit rel error xerror    xstd
## 1 0.362963      0   1.00000 1.0999 0.28014
## 2 0.148143      1   0.63704 1.2679 0.30698
## 3 0.051732      2   0.48889 1.1508 0.23559
## 4 0.010000      3   0.43716 1.0710 0.23869
```




```r
# plot tree
plot(fit, uniform=TRUE,
   main="Regression Tree for Mileage ")
text(fit, use.n=TRUE, all=TRUE, cex=.8)
```



Comparing the R2 of all the above models.

```r
# Calculate quality of fit for model with 7 nodes
Tree7_predict <- predict(tree.uscrime, data = uscrime[,1:15])
RSS7 <- sum((Tree7_predict - uscrime[,16])^2)
TSS7 <- sum((uscrime[,16] - mean(uscrime[,16]))^2)
R27 <- 1 - RSS7/TSS7
#R27
#prediction7 <- predict(tree.uscrime, test_point) # gives the probability for
each class


# Calculate quality of fit for model with 6 nodes
Tree6_predict <- predict(prune.tree6, data = uscrime[,1:15])
RSS6 <- sum((Tree6_predict - uscrime[,16])^2)
TSS6 <- sum((uscrime[,16] - mean(uscrime[,16]))^2)
R26 <- 1 - RSS6/TSS6
#prediction6 <- predict(Tree6_predict, test_point) # gives the probability fo
r each class


# Calculate quality of fit for model with 4 nodes
Tree4_predict <- predict(fit, data = uscrime[,1:15])
RSS4 <- sum((Tree4_predict - uscrime[,16])^2)
TSS4 <- sum((uscrime[,16] - mean(uscrime[,16]))^2)
R24 <- 1 - RSS4/TSS4
#prediction4 <- predict(Tree4_predict, test_point)

# Calculate quality of fit for model with 3 nodes
Tree3_predict <- predict(prune.tree3, data = uscrime[,1:15])
RSS3 <- sum((Tree3_predict - uscrime[,16])^2)
TSS3 <- sum((uscrime[,16] - mean(uscrime[,16]))^2)
R23 <- 1 - RSS3/TSS3

R2 <- c(R27, R26, R24,R23)
Model <- c('7 nodes', '6 nodes', '4 nodes' , '3 nodes' )
compData <- data.frame(Model, R2)
compData
```

```
##     Model        R2
## 1 7 nodes 0.7244962
## 2 6 nodes 0.7074149
## 3 4 nodes 0.5628378
## 4 3 nodes 0.5111061
```

By far the model with 7 nodes has performed better than the other models with a higer R2 and lower Residual mean deviance. Which means the residual errors are not as spread out. And the model explains 72% of the variation in the response. Lets try using the model to predict on unseen data.

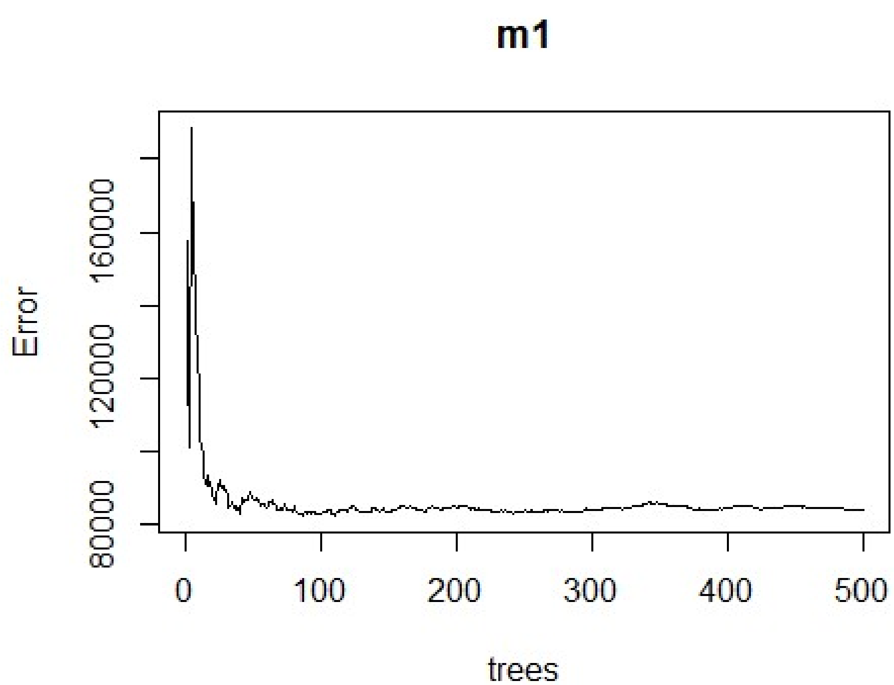
```r
# Create the test datapoint mannually ising the data.frame() function for the
new city
test_point <- data.frame(M = 14.0,So=0, Ed = 10.0 ,Po1 = 12.0 ,Po2 = 15.5,LF
= 0.640 ,M.F = 94.0 ,Pop = 150 ,NW = 1.1 ,U1= 0.120 ,U2 = 3.6 ,Wealth = 3200,
Ineq = 20.1,Prob = 0.04 ,Time = 39.0)
```

```
prediction7 <- predict(tree.uscrime, test_point) # gives the probability for
each class
prediction7

##     1
## 724.6
```

Random Forest

This is a reasonable estimate for our unseen data. Now lets try to create a random forest

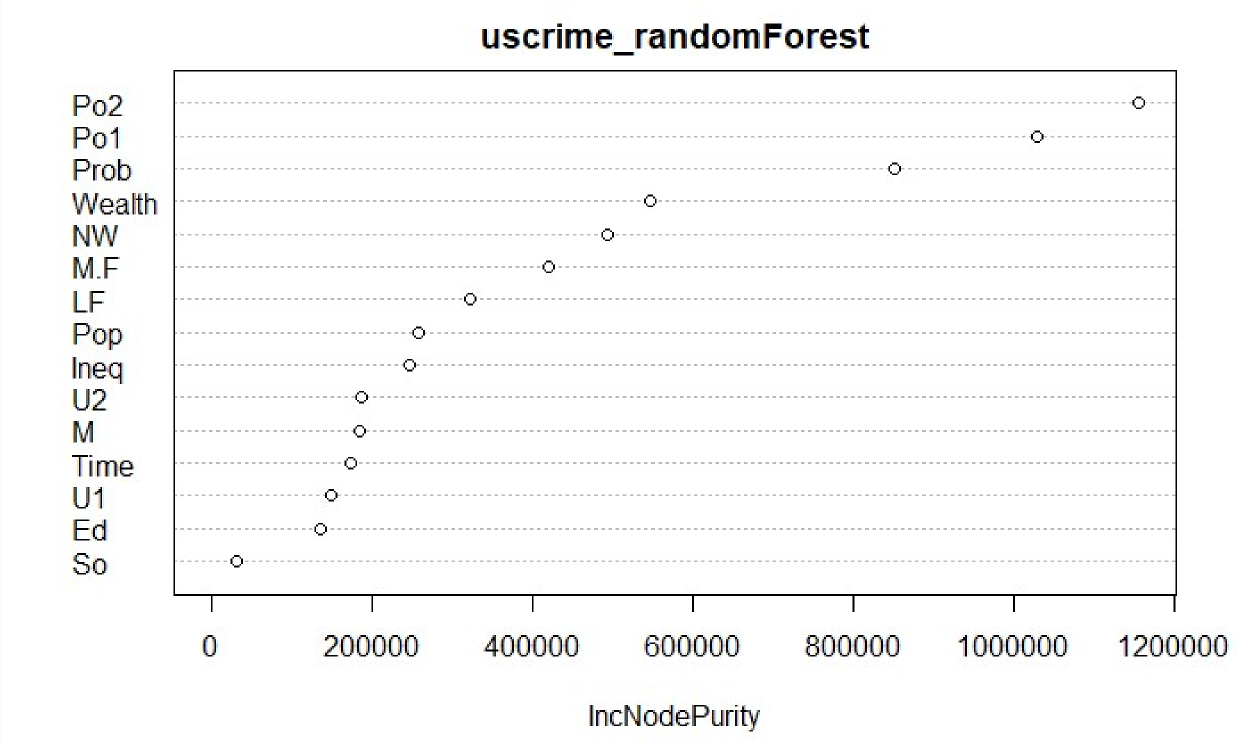
```
# for reproduciblity
set.seed(678)

# default RF model
m1 <- randomForest(
  formula = Crime~.,
  data    = uscrime
)

plot(m1)
```




```
# number of trees with lowest MSE
which.min(m1$mse)

## [1] 87

# RMSE of this optimal random forest
sqrt(m1$mse[which.min(m1$mse)])

## [1] 286.7039
```

Looks like we perform best at about 100 trees. Below we create a random forest model for ntree = 100. We also plot the important variables.

```
# randomForest speed


system.time(

uscrime_randomForest <- randomForest(
  formula = Crime~.,
  data    = uscrime, ntree = 100, mtry = floor(length(features) / 3)

      )
)

#Run prediction to see estimated value for test data
pred_randomForest <- predict(m1, test_point)
pred_randomForest

##       1
## 1226.47
```



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.

Answer

My cousin recently started a company to sell head phones in India. He is having a hard time figuring out his market demographic. He has redacted data about people who viewed his product on amazon, age, gender, purchase habits, city etc. Using information, he already has about people who purchased his product he can create a logistic regression model to predict whether prospective customers will buy his headphones. This can help target his marketing and sales tactics. Example say he sees teenage girls are more likely to buy his colored headphones, he can advertise using an Instagram influencer popular with teenage girls.

Question 10.3

Using the GermanCredit data set germancredit.txt, use logistic regression to find a good predictive model for whether credit applicants are good credit risks or not. Show your model (factors used and their coefficients), the software output, and the quality of fit.

Answer

```
#reading the german credit data

german <- read.table("germancredit.txt", header = F)
german$V21[german$V21 == 2] <- 0
head(german)
```

```
##    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   0
## 3 A191 A201   1
## 4 A191 A201   1
## 5 A191 A201   0
## 6 A192 A201   1
```

```
# training and test data sets, 80-20 split
partition <- createDataPartition(german$V21, p = .8, list = F)
# training and test data sets
train_credit <- german[partition,]
test_credit <- german[-partition,]
```

```r
# define training control
train_control <- trainControl(method = "cv", number = 10)

# train the model on training set
model <- train(V21 ~ .,
               data = train_credit,
               trControl = train_control,
               method = "glm",
               family=binomial())

summary(model)

##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7030  -0.6375   0.3349   0.6529   2.2619
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.943e-01  1.315e+00  -0.376  0.70705
## V1A12        1.198e-01  2.516e-01   0.476  0.63382
## V1A13        8.141e-01  4.351e-01   1.871  0.06132 .
## V1A14        1.807e+00  2.732e-01   6.614 3.73e-11 ***
## V2          -2.767e-02  1.127e-02  -2.455  0.01410 *
## V3A31       -3.168e-01  6.706e-01  -0.472  0.63667
## V3A32        7.412e-01  5.352e-01   1.385  0.16604
## V3A33        9.916e-01  5.754e-01   1.723  0.08487 .
## V3A34        1.567e+00  5.407e-01   2.898  0.00376 **
## V4A41        2.305e+00  4.462e-01   5.166 2.39e-07 ***
## V4A410       2.211e+00  8.938e-01   2.474  0.01337 *
## V4A42        9.322e-01  2.977e-01   3.131  0.00174 **
## V4A43        1.363e+00  2.895e-01   4.706 2.52e-06 ***
## V4A44        3.697e-01  8.652e-01   0.427  0.66914
## V4A45        5.010e-01  6.031e-01   0.831  0.40618
## V4A46        1.336e-01  4.433e-01   0.301  0.76314
## V4A48        1.475e+00  1.159e+00   1.273  0.20311
## V4A49        1.059e+00  3.870e-01   2.735  0.00623 **
## V5          -1.333e-04  5.247e-05  -2.539  0.01110 *
## V6A62        5.105e-01  3.411e-01   1.497  0.13442
## V6A63        6.777e-01  5.081e-01   1.334  0.18223
## V6A64        1.391e+00  5.587e-01   2.490  0.01279 *
## V6A65        8.894e-01  2.977e-01   2.987  0.00281 **
## V7A72        7.966e-01  5.203e-01   1.531  0.12575
## V7A73        8.230e-01  4.986e-01   1.651  0.09880 .
## V7A74        1.452e+00  5.350e-01   2.714  0.00665 **
## V7A75        7.195e-01  5.001e-01   1.439  0.15022
## V8          -3.235e-01  9.993e-02  -3.237  0.00121 **
## V9A92        1.327e-01  4.773e-01   0.278  0.78100
## V9A93        8.371e-01  4.689e-01   1.785  0.07422 .
## V9A94        1.153e-01  5.541e-01   0.208  0.83513
## V10A102     -5.364e-01  4.633e-01  -1.158  0.24690
```

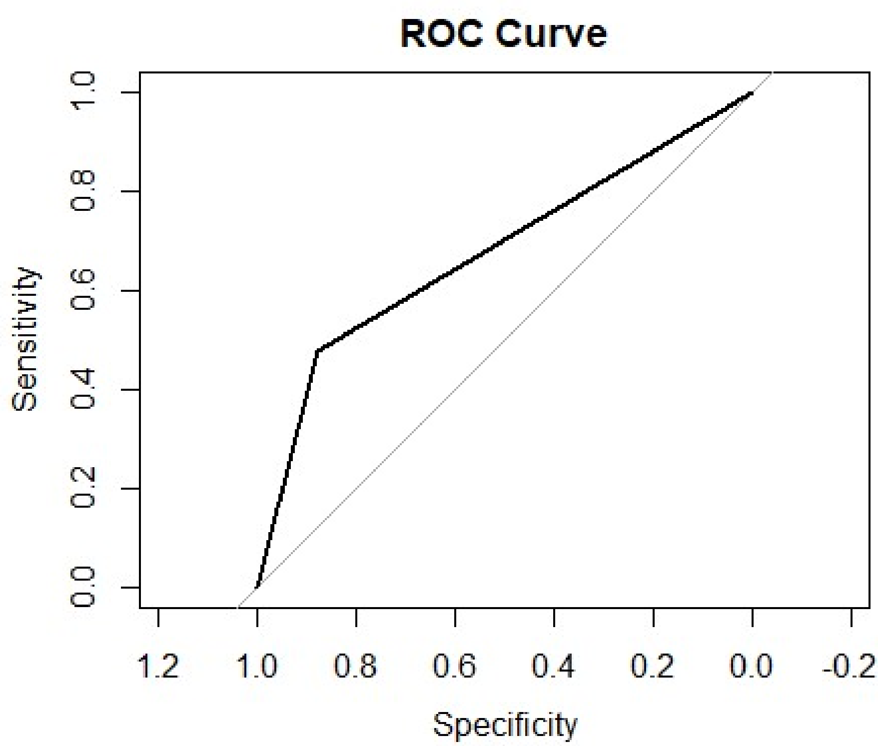
```
## V10A103      6.670e-01  4.438e-01   1.503  0.13290
## V11          6.064e-02  1.016e-01   0.597  0.55073
## V12A122     -1.314e-01  2.886e-01  -0.455  0.64887
## V12A123     -2.003e-01  2.753e-01  -0.728  0.46690
## V12A124     -8.476e-01  4.863e-01  -1.743  0.08135 .
## V13          1.453e-02  1.056e-02   1.375  0.16900
## V14A142     -2.924e-01  4.869e-01  -0.600  0.54818
## V14A143      5.813e-01  2.723e-01   2.135  0.03277 *
## V15A152      2.902e-01  2.639e-01   1.099  0.27155
## V15A153      8.119e-01  5.388e-01   1.507  0.13184
## V16         -4.154e-01  2.167e-01  -1.917  0.05528 .
## V17A172     -8.324e-01  8.616e-01  -0.966  0.33400
## V17A173     -9.030e-01  8.369e-01  -1.079  0.28058
## V17A174     -1.032e+00  8.510e-01  -1.212  0.22543
## V18         -3.788e-01  2.847e-01  -1.331  0.18328
## V19A192      1.407e-01  2.340e-01   0.601  0.54768
## V20A202      9.533e-01  6.757e-01   1.411  0.15832
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 975.68  on 799  degrees of freedom
## Residual deviance: 683.69  on 751  degrees of freedom
## AIC: 781.69
##
## Number of Fisher Scoring iterations: 5
```

```r
# print cv scores
coef(model$finalModel)
```

```
##   (Intercept)         V1A12         V1A13         V1A14            V2
## -0.4942674228  0.1198333257  0.8140986788  1.8072660868 -0.0276663464
##         V3A31         V3A32         V3A33         V3A34         V4A41
## -0.3167700947  0.7412052069  0.9915539036  1.5668124275  2.3051577363
##        V4A410         V4A42         V4A43         V4A44         V4A45
##  2.2112134459  0.9322449205  1.3626602113  0.3697158883  0.5009790313
##         V4A46         V4A48         V4A49            V5         V6A62
##  0.1335891722  1.4745677145  1.0586075436 -0.0001332511  0.5105372920
##         V6A63         V6A64         V6A65         V7A72         V7A73
##  0.6777262616  1.3910411924  0.8893618394  0.7966499797  0.8229982670
##         V7A74         V7A75            V8         V9A92         V9A93
##  1.4518465324  0.7195219575 -0.3234770463  0.1327046401  0.8371469431
##         V9A94        V10A102       V10A103           V11       V12A122
##  0.1153206949 -0.5364093747  0.6669749969  0.0606352152 -0.1314140575
##       V12A123       V12A124           V13       V14A142       V14A143
## -0.2002841564 -0.8476012104  0.0145285982 -0.2923618008  0.5813361680
##       V15A152       V15A153           V16       V17A172       V17A173
##  0.2901514902  0.8119140708 -0.4153818579 -0.8323877030 -0.9030154607
##       V17A174           V18       V19A192       V20A202
## -1.0316150541 -0.3788381113  0.1406864969  0.9532860400
```

```r
yhat_logit <- predict(model$finalModel, test_credit, type = "response")
yhat1 <- as.integer(yhat_logit > 0.5)
```

```r
table(yhat1, test_credit$V21)

##
## yhat1   0   1
##     0 183  48
##     1  25  44

require(pROC)

AUC <- roc(test_credit$V21, yhat1)

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

## Setting direction: controls < cases

plot(AUC, main = "ROC Curve")
```

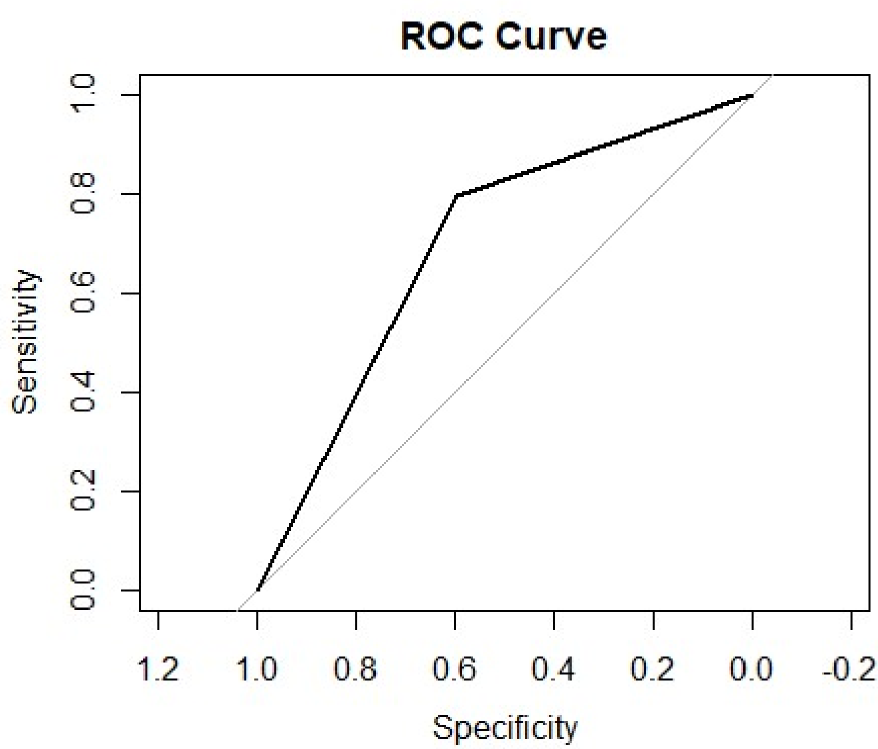



```r
AUC
## Call:
## roc.default(response = test_credit$V21, predictor = yhat1)
##
## Data: yhat1 in 208 controls (test_credit$V21 0) < 92 cases (test_credit$V2
1 1).
## Area under the curve: 0.679
```

Since we have a lot of redundent features, we will create a new logistic regression model only using important features. And as approving a bad loan is 5 times worse than deniing a good loan, we will change out threshold value to .22.

```r
#Create logistic model important features
reg_imp <- glm(V21~V1+V2+V3+V4+V8+V12+V13+V14+V20+V17+V7,family=binomial(link
```

```
= 'logit'), data =train_credit )

yhat_logit_imp <- predict(reg_imp, test_credit, type = "response")
yhat1_imp <- as.integer(yhat_logit_imp > 0.22)

table(yhat1_imp, test_credit$V21)

##
## yhat1_imp   0   1
##         0 124  19
##         1  84  73

AUC_imp <- roc(test_credit$V21, yhat1_imp)

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

## Setting direction: controls < cases

plot(AUC_imp, main = "ROC Curve")
```




```
AUC_imp

##
## Call:
## roc.default(response = test_credit$V21, predictor = yhat1_imp)
##
## Data: yhat1_imp in 208 controls (test_credit$V21 0) < 92 cases (test_credi
t$V21 1).
## Area under the curve: 0.6948

# print cv scores for final model
coef(reg_imp)
```

```
##  (Intercept)        V1A12        V1A13        V1A14           V2           V3
A31
##   0.85501504  -0.71593511  -1.27762694  -1.84872379   0.03584184  -0.21559
341
##        V3A32        V3A33        V3A34        V4A41       V4A410           V4
A42
##  -1.10097859  -0.90948599  -1.53181261  -1.35567874  -2.15463870  -0.43123
636
##        V4A43        V4A44        V4A45        V4A46        V4A48           V4
A49
##  -0.92997078  -0.63716931  -0.13092203  -0.21249392 -15.32054213  -0.56945
382
##           V8      V12A122      V12A123      V12A124          V13         V14A
142
##   0.19708549   0.18301749   0.39052156   1.05280470  -0.02863517  -0.52205
221
##      V14A143      V20A202      V17A172      V17A173      V17A174           V7
A72
##  -0.66345477  -1.94259785   1.28748749   1.25020310   1.24495706  -0.03587
521
##        V7A73        V7A74        V7A75
##  -0.48726744  -1.20969829  -0.64892301
```

Fortunately, our final logistic model has a higher AUC of 0.6948 and has a much lower false positive vs false negative.