Supplementary R Syntax – Data Preprocessing

Variable Description	Variable Coding	Original Variables Used
Response Variable		
Violence	Violence	F12VIOL
Predictor Variables		
Age	Age	AGE
BIS Non-Planning Subscale	BISnp	BISPLN
BPRS Activation Subscale	BPRSa	OACTV
BPRS Hostile-Suspiciousness Subscale	BPRSh	OHOST
BPRS Total Score	BPRSt	OBPRS
Child Abuse Seriousness	ChildAbuse	Q5.5.1, Q5.5.2, Q5.5.3, Q5.5.4, Q5.5.5, Q5.5.6
Loss of Consciousness (Head Injury)	Consc	NEU2A
Father Arrest History	DadArr	Q5.20A, Q5.20B
Father's Drug Use	DadDrug	Q5.19A, Q5.19B
Drug Abuse (DSM-III-R)	DrugAbuse	DSM16A, DSM16B, DSM17A, DSM17B
Employed Prior to Hospitalization	Emp	Q4.4
Violent Fantasies: Escalating Seriousness	FantEsc	Q7.1, Q7.7
Violent Fantasies: Single Target Focus	FantSing	Q7.1, Q7.6
Violent Fantasies: Target Present	FantTarg	Q7.1, Q7.8
Level of Functioning	Function	Q9.1, Q9.2, Q9.3, Q9.4, Q9.5, Q9.6
Grandiose Delusions	GranDel	DEL03.1
Previous Head Injuries	HeadInj	NEU4B.1, NEU4B.2, NEU4B.3, NEU4B.4, NEU4B.5, NEU4B.6, NEU4B.7, NEU4B.8, NEU4B.9
Legal Status for Hospitalization	LegalStatus	LEGALR
Novaco Anger Scale Behavioral Subscale	NASb	NASBEH
Number of Negative Relationships	NegRel	Q10.10N, Q10.11N, Q10.12N, Q10.13N
Psychopathy Checklist: Screening Version	PCL	PCLTOT
MacArthur Perceived Coercion Scale	PCS	Q1.8, Q1.11, Q1.14, Q1.21, Q1.22
Prior Arrest Frequency	PriorArr	FREQARR
Property Crime Arrest	PropCrime	PROPARR
Violent Before Hospitalized	RecViol2	VIOL
Schizophrenic	Schiz	DSM2A, DSM5A
Proportion of Social Network are Mental Health Professionals	SNMHP	SNMHP
Substance Abuse	SubAbuse	DSM14A, DSM14B, DSM15A DSM15B DSM16A, DSM16B, DSM17A, DSM17B
Admission Reason: Suicide	Suicide	QREAS.02
Threat/Control Override Symptoms	tco	K1.1, K1.3, K2.1, K2.3, K3.1, K3.3, K4.1, K4.3, K8.1, K8.3, K9.1, K9.3, K10.1, K10.3, K12.1, K12.3
Threats at Admission	Threats	QREAS.20, QREAS.21

Table 1: Variables used in analyses, from the MacArthur Violence Risk Assessment Study (Monahan et al., 2001).

Table 1 above displays the variables included in the initial analyses. The first column is a brief description; the second column is the coding used in the analysis; the third column consists of the variable codes used in the original VRAS dataset. All variables come from the SPSS file `baseline.sav` except `F12VIOL` and `PCLTOT`

that were from the SPSS file `follow_up_subjects.sav`.

We begin by loading the necessary packages; this assumes the packages are installed. If not, use `install.packages()` (e.g., `install.packages("dplyr")` installs the `dplyr` package).

```
# for reading in SPSS files
require(Hmisc)
# for data frame manipulation
library(dplyr)
# for creating data tables
library(data.table)
```

Data Preprocessing

First, we bring in the two data files and sort in order of `STUDYID` and then merge them into one master file data table called `COVR`.

```
# read in data files
COVR1 = data.table(spss.get('baseline.sav'))
COVR2 = data.table(spss.get('follow_up_subjects.sav'))

# sort data, select desired variables, and merge into one master data table
COVR1 = COVR1 %>%
     select(STUDYID, BISNPLN, OACTV, OHOST, OBPRS, LEGALR, SNMHP, NOVBEH, NEU2A,
            NEU4B.1:NEU4B.9, TYPEARR, FREQARR, PROPARR, DEL03.1, VIOL, DSM2A, DSM5A,
            DSM14A:DSM17B, AGE, Q1.8, Q1.11, Q1.14, Q1.21, Q1.22, Q4.4, Q5.5.1:Q5.5.6,
            Q5.19A, Q5.19B, Q5.20A, Q5.20B, Q7.1, Q7.6:Q7.8, Q9.1:Q9.6, Q10.10N, Q10.11N,
            Q10.12N, Q10.13N, QREAS.02, QREAS.20, QREAS.21, K1.1, K1.3, K2.1, K2.3, K3.1,
            K3.3, K4.1, K4.3, K8.1, K8.3, K9.1, K9.3, K10.1, K10.3, K12.1, K12.3) %>%
     arrange(STUDYID)
COVR2 = COVR2 %>%
     mutate(STUDYID = studyid, F12VIOL = f12viol, PCLTOT = pcltot) %>%
     select(STUDYID, F12VIOL, PCLTOT) %>%
     arrange(STUDYID)
COVRdata = inner_join(COVR1, COVR2, by = 'STUDYID')
rm('COVR1', 'COVR2')

# remove data with missing outcome variable
COVRdata = filter(COVRdata, !is.na(F12VIOL))
```

Preprocess the data to create the variables used in the VRAS study.

```
ChildAbuseVars = COVRdata %>%
     select(Q5.5.1:Q5.5.6) %>%
     transmute(ChildAbuse = 5*(rowSums(cbind(Q5.5.4, Q5.5.5, Q5.5.6)) > 3) +
                   3*(rowSums(cbind(Q5.5.2, Q5.5.3)) > 2) + (as.numeric(Q5.5.1) > 1))
SubAbVars = COVRdata %>%
     select(DSM14A, DSM14B, DSM15A, DSM15B, DSM16A, DSM16B, DSM17A, DSM17B)
DrugAbVars = COVRdata %>%
     select(DSM16A, DSM16B, DSM17A, DSM17B)
HeadInjVars = COVRdata %>%
```

```
     select(NEU4B.1, NEU4B.2, NEU4B.3, NEU4B.4, NEU4B.5,
            NEU4B.6, NEU4B.7, NEU4B.8, NEU4B.9)
tcoPatient = select(COVRdata, K1.1, K2.1, K3.1, K4.1, K8.1, K9.1, K10.1, K12.1)
tcoClinical = select(COVRdata, K1.3, K2.3, K3.3, K4.3, K8.3, K9.3, K10.3, K12.3)

COVRdata = COVRdata %>%
     transmute(
          Violence = factor(ifelse(F12VIOL == 'Yes', 1, 0)),
          Age = as.numeric(AGE),
          BISnp = as.numeric(BISNPLN),
          BPRSa = as.numeric(OACTV),
          BPRSh = as.numeric(OHOST),
          BPRSt = as.numeric(OBPRS),
          ChildAbuse = ifelse(ChildAbuseVars$ChildAbuse >= 5, 3,
                              ifelse(ChildAbuseVars$ChildAbuse >= 3, 2,
                                     ifelse(ChildAbuseVars$ChildAbuse >= 1, 1, 0))),
          Consc = factor(ifelse(NEU2A == 'YES', 1, ifelse(NEU2A == 'NO', 0, NA))),

          DadDrug = factor(ifelse(((Q5.19A == 'DAILY' | Q5.19A == 'ONCE A WEEK' |
                                      Q5.19A == 'TWICE A WEEK') |
                                     (Q5.19B == 'DAILY' | Q5.19B == 'ONCE A WEEK' |
                                          Q5.19B == 'TWICE A WEEK')), 1,
                                  ifelse(((is.na(Q5.19A) | Q5.19A == 'NA') &
                                              (Q5.19B == 'NA' | Q5.19B == 'DK')),
                                         NA, 0))),
          DadArr = factor(ifelse((Q5.20A == 'NEVER' &
                                      (Q5.20B == 'NA' | Q5.20B == 'DK') |
                                      (Q5.20B == 'NEVER' &
                                           (is.na(Q5.20A) | Q5.20A == 'NA')) |
                                      (Q5.20A == 'NEVER' & Q5.20B == 'NEVER')), 0,
                                 ifelse(((is.na(Q5.20A) | Q5.20A == 'NA') &
                                             (Q5.20B == 'NA' | Q5.20B == 'DK')),
                                        NA, 1))),
          DrugAbuse = factor(ifelse(rowSums(DrugAbVars == 'UNCERTAIN') == 4, NA,
                                    ifelse(rowSums(DrugAbVars == 'PRESENT') > 0, 1, 0))),
          Emp = factor(ifelse(Q4.4 %in% c('YES - FULL-TIME', 'YES - PART-TIME'), 1,
                              ifelse(Q4.4 == 'NO', 0, NA))),
          FantEsc = factor(ifelse((Q7.1 == 'YES' & Q7.7 == 'MORE SERIOUS'), 1, 0)),
          FantSing = factor(ifelse((Q7.1 == 'YES' & Q7.6 == 'SAME'), 1, 0)),
          FantTarg = factor(ifelse((Q7.1 == 'YES' & Q7.8 == 'YES'), 1, 0)),
          Function = ifelse(rowSums(is.na(cbind(Q9.1, Q9.2,
                                                Q9.3, Q9.4, Q9.5, Q9.6))) == 6, NA,
                            rowSums(cbind(Q9.1, Q9.2, Q9.3, Q9.4, Q9.5, Q9.6) - 1,
                                    na.rm = T)),
          GranDel = factor(ifelse(DEL03.1 == 'YES - CHECKED', 1, 0)),
          HeadInj = factor(ifelse(rowSums(HeadInjVars == 'YES, HEAD INJURY',
                                          na.rm = T) > 0, 1,
                                  ifelse(rowSums(is.na(HeadInjVars)) == 9, NA, 0))),
          LegalStatus = factor(ifelse(LEGALR == 'INVOLUNTARY', 1, 0)),
          NASb = as.numeric(NOVBEH),
          NegRel = rowSums(cbind(Q10.10N, Q10.11N, Q10.12N, Q10.13N)),
          PCL = factor(ifelse(PCLTOT > 12, 1, 0)),
          PCS = ifelse(rowSums(is.na(cbind(Q1.8, Q1.11, Q1.14, Q1.21, Q1.22))) == 5, NA,
```

```
                        rowSums(-1*(cbind(Q1.8, Q1.11, Q1.14, Q1.21, Q1.22)) + 2,
                                na.rm = T)),
          PriorArr = as.numeric(FREQARR) - 1,
          PropCrime = factor(ifelse(PROPARR == 'Yes', 1, 0)),
          RecViol2 = factor(ifelse(VIOL == 'Violence', 1, 0)),
          Schiz = factor(ifelse(((DSM2A == 'ABSENT' | DSM2A == 'UNCERTAIN') &
                                   (DSM5A == 'ABSENT'| DSM5A == 'UNCERTAIN')), 0, 1)),
          SNMHP = as.numeric(SNMHP),
          SubAbuse = factor(ifelse(rowSums(SubAbVars == 'UNCERTAIN') == 8, NA,
                                   ifelse(rowSums(SubAbVars == 'PRESENT') > 0, 1, 0))),
          Suicide = factor(ifelse(QREAS.02 == 'YES - CHECKED', 1, 0)),
          tco = factor(ifelse(rowSums((tcoPatient == 'YES') == (tcoClinical == 'YES'),
                                   na.rm = T) > 0, 1, 0)),
          Threats = factor(ifelse((QREAS.20 == 'YES - CHECKED' |
                                     QREAS.21 == 'YES - CHECKED'), 1,
                                  ifelse(rowSums(is.na(cbind(QREAS.20, QREAS.21))) == 2,
                                         NA, 0)))
          )
rm(list = ls()[ls() != 'COVRdata'])
```

Save the data table for construction of classification models.

```
save.image('COVRdata.rda')
```

Compute correlations between predictor variables and the response.

```
R = data.frame(sapply(COVRdata, as.numeric))
COVRr = apply(select(R, Violence), 2, cor, select(R, -Violence), 'pairwise.complete.obs')
rownames(COVRr) = colnames(select(R, -Violence))
round(COVRr, 2)
```

```
             Violence
Age             -0.07
BISnp            0.05
BPRSa           -0.08
BPRSh            0.08
BPRSt           -0.04
ChildAbuse       0.14
Consc            0.09
DadDrug          0.14
DadArr           0.15
DrugAbuse        0.16
Emp             -0.05
FantEsc          0.13
FantSing         0.10
FantTarg         0.12
Function        -0.01
GranDel         -0.01
HeadInj          0.03
LegalStatus      0.11
NASb             0.17
NegRel           0.05
```

PCL	0.26
PCS	0.03
PriorArr	0.24
PropCrime	0.11
RecViol2	0.14
Schiz	-0.12
SNMHP	-0.10
SubAbuse	0.18
Suicide	-0.01
tco	-0.09
Threats	0.06

Supplementary R Syntax – Classification Modeling

First load the necessary packages; this assumes the packages are installed. If not, use `install.packages()` (e.g., `install.packages("dplyr")` installs the `dplyr` package).

```
# for linear discriminant analysis
library(MASS)
# for imputation missing values
require(Hmisc)
# for cross-validation of models
library(boot)
# constructing ROC plots and computing AUC
library(ROCR)
# for data frame manipulation
library(dplyr)
# for plotting
library(ggplot2)
```

Next load preprocessed data.

```
load('COVRdata.rda')
```

Calculate the base rate of violence in the sample.

```
BR = mean(select(COVRdata, Violence) == 1)
```

```
Loading required namespace: data.table
```

Logisitic Regression Model

Monahan et al. (2001) constructed a main effects logistic regression (MELR) model to predict violence that was fit with forward-stepwise variable selection with a $p < .05$-threshold for retaining predictor variables. The present analysis constructs an MELR model but fitted with only the variables from the final model given by Monahan et al. The results are similar, but not exact (see Monahan et al., 2001, Table 5.1).

Before constructing the logistic regression model, we impute missing data by replacing all missing data with the mean of the non-missing data for continuous variables and the mode of the non-missing data for categorical variables, as was done by Monahan et al. (2001).

```
# function for computing the mode
varMode <- function(x) return(factor(names(table(x))[table(x) == max(table(x))]))

# function for missing value imputation; the function impute() is from Hsmic package
imputeNA <- function(x) {
     if (is.factor(x)) { #impute mode for factor variables

          return(factor(impute(x, varMode)))

     } else {

          return(impute(x, mean))
```

```
    }

}

# impute missing data
COVRdata = COVRdata %>%
    summarise_each(funs(imputeNA))
```

First select variables used in Monahan et al.'s (2001) logistic regression model.

```
logRegData = COVRdata %>%
    select(Violence, BISnp, BPRSa, BPRSh, BPRSt, ChildAbuse, Consc, DadDrug, DrugAbuse,
           Emp, FantEsc, FantSing, GranDel, LegalStatus, NASb, PCL, PriorArr, SNMHP, tco)
```

Next construct the logistic regression model.

```
logisticModel = glm(Violence ~ ., data = logRegData, family = binomial(logit))
summary(logisticModel)
```

```
Call:
glm(formula = Violence ~ ., family = binomial(logit), data = logRegData)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.741  -0.602  -0.400  -0.230   2.857

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   -2.8990     0.6714   -4.32  1.6e-05 ***
BISnp         -0.0280     0.0129   -2.17  0.02969 *
BPRSa         -0.1511     0.0633   -2.39  0.01691 *
BPRSh          0.1175     0.0402    2.92  0.00348 **
BPRSt         -0.0331     0.0161   -2.06  0.03947 *
ChildAbuse     0.3744     0.1042    3.59  0.00033 ***
Consc1         0.5181     0.2612    1.98  0.04733 *
DadDrug1       0.7363     0.2762    2.67  0.00768 **
DrugAbuse1     0.3807     0.2296    1.66  0.09734 .
Emp1          -0.4771     0.2001   -2.38  0.01710 *
FantEsc1       0.6754     0.3262    2.07  0.03837 *
FantSing1      0.5642     0.2588    2.18  0.02922 *
GranDel1       0.7109     0.3436    2.07  0.03853 *
LegalStatus1   0.5110     0.1986    2.57  0.01006 *
NASb           0.0380     0.0151    2.52  0.01181 *
PCL1           0.8982     0.2113    4.25  2.1e-05 ***
PriorArr       0.2978     0.0820    3.63  0.00028 ***
SNMHP         -1.8550     0.7471   -2.48  0.01303 *
tco1          -0.9003     0.3420   -2.63  0.00848 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)
```

```
    Null deviance: 906.10  on 938  degrees of freedom
Residual deviance: 731.06  on 920  degrees of freedom
AIC: 769.1

Number of Fisher Scoring iterations: 5
```

We now compute the cross-validation error of logistic regression model, using leave-one-out cross-validation.

```
# function for calculating error when using classification cutscore of .5
LRcost50 = function(x, p = 0) mean(abs(x - p) > .5)
# estimated cross-validated error
cv.glm(logRegData, logisticModel, LRcost50)$delta[1]
```

```
[1] 0.1821
```

```
# resubstitution error
LRpreds = predict(logisticModel, type = 'resp')
mean((LRpreds > .5) != (select(COVRdata, Violence) == 1))
```

```
[1] 0.1693
```

```
# function for calculating error when using classification cutscore of .37
LRcost37 = function(x, p = 0) mean(abs(x - p) > .37)
# estimated cross-validated error
cv.glm(logRegData, logisticModel, LRcost37)$delta[1]
```

```
[1] 0.2407
```

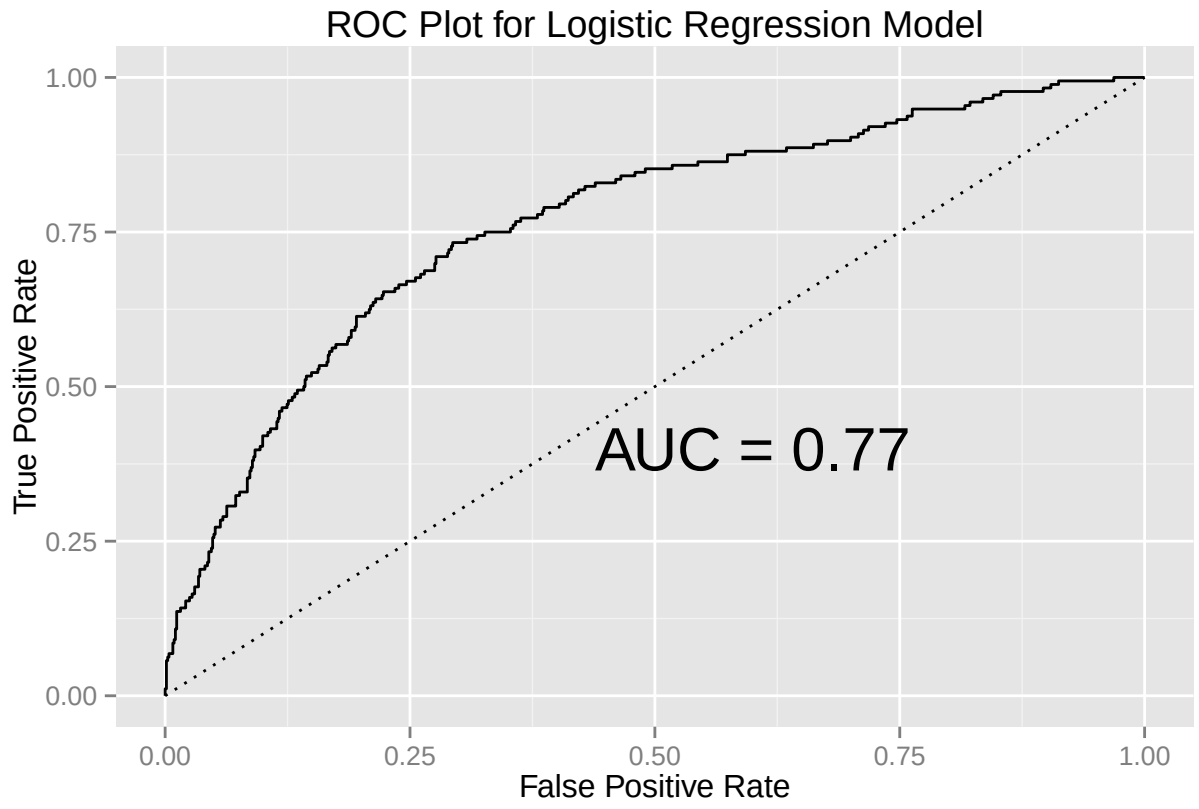
```
# resubstitution error
mean((LRpreds > 2*BR) != (select(COVRdata, Violence) == 1))
```

```
[1] 0.1789
```

Next we plot an ROC curve and calculate the AUC. We want to use cross-validated results, not the results from the model. To do so, we need to manually compute the cross-validated estimates because `cv.glm()` unfortunately does not provide this.

```
err = double()
for (k in 1:939) {
     t = glm(Violence ~ ., data = logRegData, family = binomial(logit), subset = -k)
     # predicted probabilty of being violent
     err[k] = predict(t, COVRdata[k,], type = 'resp')
}


lrPreds = prediction(err, select(COVRdata, Violence))
lrPerf = performance(lrPreds, 'tpr', 'fpr')
AUC = round(performance(lrPreds, 'auc')@y.values[[1]], 2)
ggplot(data = NULL) +
```

```
    geom_line(aes(x = lrPerf@x.values[[1]],
                              y = lrPerf@y.values[[1]])) +
    ggtitle('ROC Plot for Logistic Regression Model') +
    xlab('False Positive Rate') +
    ylab('True Positive Rate') +
    geom_segment(aes(x = 0, y = 0, xend = 1, yend = 1),
                 linetype = 'dotted') +
    geom_text(aes(x = .6, y = .4, label = paste0('AUC = ', AUC), parse = T),
              size = 8)
```



Discriminant Function Analysis

This section applies discriminant function analysis (DFA) to classify individuals as violent, for both a linear fit (i.e., assuming the covariances among the two populations—nonviolent and violent—are equal) and a quadratic fit (i.e., the covariances are allowed to be unequal). This was not done by Monahan et al. (2001) but allows comparison with the other methods used (logistic regression and classification trees).

The same data with missing value imputation that was used in the logistic regression model is used for the discriminant analyses. All the data are used.

First, a linear discriminant function is constructed, with equal costs and unequal costs. Before doing so, the workspace is cleared, the base rate for violence is calculated, and the prior probabilities are established such that unequal costs are applied.

```
# Clear workspace
rm(list = ls()[!(ls() %in% c('BR', 'COVRdata'))])
# cost of FN to FP
fnCost = (1-2*BR)/(2*BR)
# priors
priors = c((1 - BR)/(BR*fnCost + (1-BR)), BR*fnCost/(BR*fnCost + (1-BR)))
rm(fnCost)
```

Resubstitution error and cross-validation error of linear discriminant model, using leave-one-out cross-validation. This is found for both the model with equal costs and the one with unequal costs.

```
# Equal costs
# resubstitution error
ldaModel_equal = lda(Violence ~ ., data = COVRdata)
mean(predict(ldaModel_equal)$class != COVRdata$Violence)
```

```
[1] 0.1736
```

```
# estimated cross-validated error
ldaModel_equal = lda(Violence ~ ., data = COVRdata, CV = T)
mean(ldaModel_equal$class != COVRdata$Violence)
```

```
[1] 0.1885
```

```
# Unequal costs
# resubstitution error
ldaModel_unequal = lda(Violence ~ ., data = COVRdata, prior = priors)
mean(predict(ldaModel_unequal)$class != COVRdata$Violence)
```

```
[1] 0.1842
```

```
# estimated cross-validated error
ldaModel_unequal = lda(Violence ~ ., data = COVRdata, CV = T, prior = priors)
mean(ldaModel_unequal$class != COVRdata$Violence)
```

```
[1] 0.2045
```

Next a quadratic discriminant function is constructed, again with equal costs and unequal costs.

```
# Equal costs
# resubstitution error
qdaModel_equal = qda(Violence ~ ., data = COVRdata)
mean(predict(qdaModel_equal)$class != COVRdata$Violence)
```

```
[1] 0.1438
```

```
# estimated cross-validated error
qdaModel_equal = qda(Violence ~ ., data = COVRdata, CV = T)
mean(qdaModel_equal$class != COVRdata$Violence)
```

```
[1] 0.23
```

```
# Unequal costs
# resubstitution error
qdaModel_unequal = qda(Violence ~ ., data = COVRdata, prior = priors)
mean(predict(qdaModel_unequal)$class != COVRdata$Violence)
```

```
[1] 0.1523
```

```
# estimated cross-validated error
qdaModel_unequal = qda(Violence ~ ., data = COVRdata, CV = T, prior = priors)
mean(qdaModel_unequal$class != COVRdata$Violence)
```

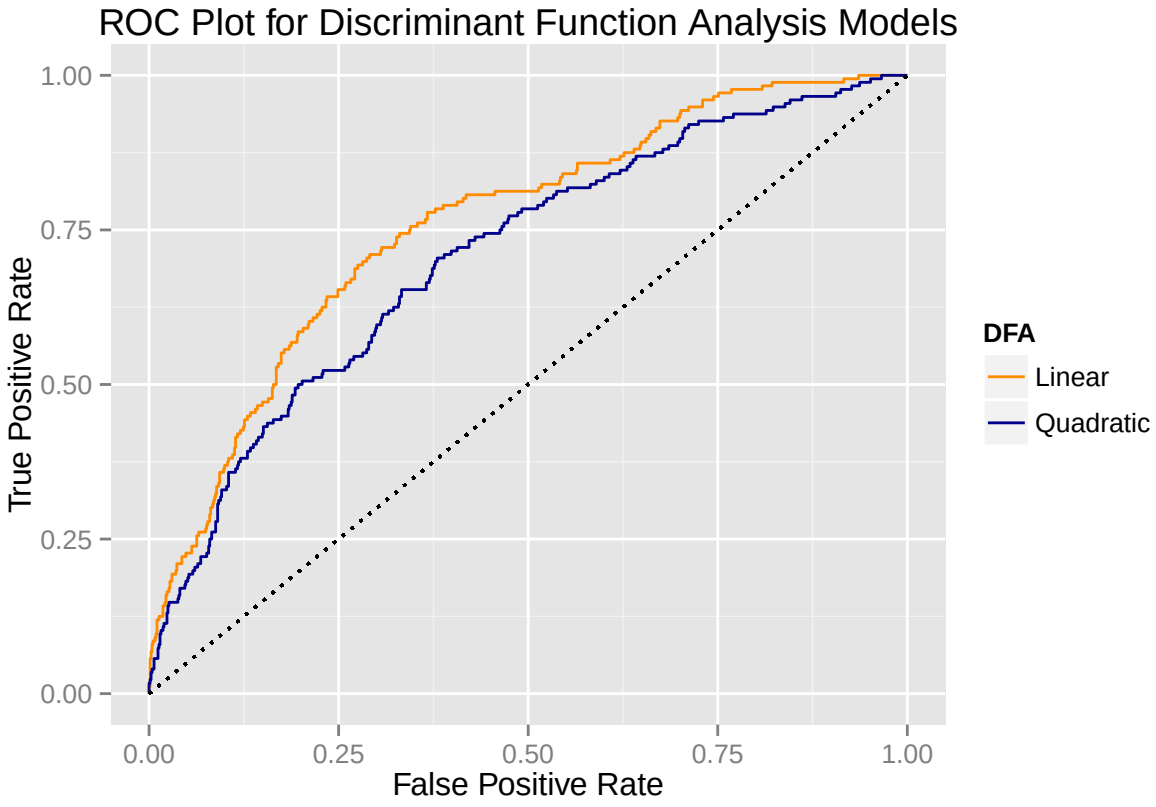
```
[1] 0.2513
```

Plotting ROC curve.

```
ldaPreds = prediction(ldaModel_equal$post[,2], select(COVRdata, Violence))
ldaPerf = performance(ldaPreds, 'tpr', 'fpr')
qdaPreds = prediction(qdaModel_equal$post[,2], select(COVRdata, Violence))
qdaPerf = performance(qdaPreds, 'tpr', 'fpr')
AUC = data_frame(lda = round(performance(ldaPreds, 'auc')@y.values[[1]], 2),
                 qda = round(performance(qdaPreds, 'auc')@y.values[[1]], 2))

# AUC for models
AUC
```

```
Source: local data frame [1 x 2]

   lda  qda
1 0.76 0.71
```

```
# ROC plot
dfaStats = data_frame(x = c(ldaPerf@x.values[[1]], qdaPerf@x.values[[1]]),
                      y = c(ldaPerf@y.values[[1]], qdaPerf@y.values[[1]]),
                      DFA = rep(c('Linear', 'Quadratic'),
                                times = c(length(ldaPerf@x.values[[1]]),
                                          length(qdaPerf@x.values[[1]]))))
ggplot(data = dfaStats) +
     geom_line(aes(x = x, y = y, color = DFA)) +
     ggtitle('ROC Plot for Discriminant Function Analysis Models') +
     xlab('False Positive Rate') +
     ylab('True Positive Rate') +
     geom_segment(aes(x = 0, y = 0, xend = 1, yend = 1),
                  linetype = 'dotted') +
     scale_color_manual(values = c('darkorange', 'darkblue'))
```


ROC Plot for Discriminant Function Analysis Models
True Positive Rate
False Positive Rate
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
DFA
Linear
Quadratic

Supplementary R Syntax – Classification Trees

First load the necessary packages; this assumes the packages are installed. If not, use `install.packages()` (e.g., `install.packages("rpart")` installs the `rpart` package).

```
# for recursive partitioning trees
library(rpart)
# for plotting rpart trees
library(rpart.plot)
# for random forest model
library(randomForest)
# for plotting
library(ggplot2)
# for data frame manipulation
library(dplyr)
# for cross-validation and other machine learning tools
library(caret)
# for constructing ROC plots and computing AUC
library(ROCR)
```

Next load preprocessed data.

```
load('COVRdata.rda')
```

Calculate the base rate of violence in the sample.

```
BR = mean(select(COVRdata, Violence) == 1)
```

Classification Trees

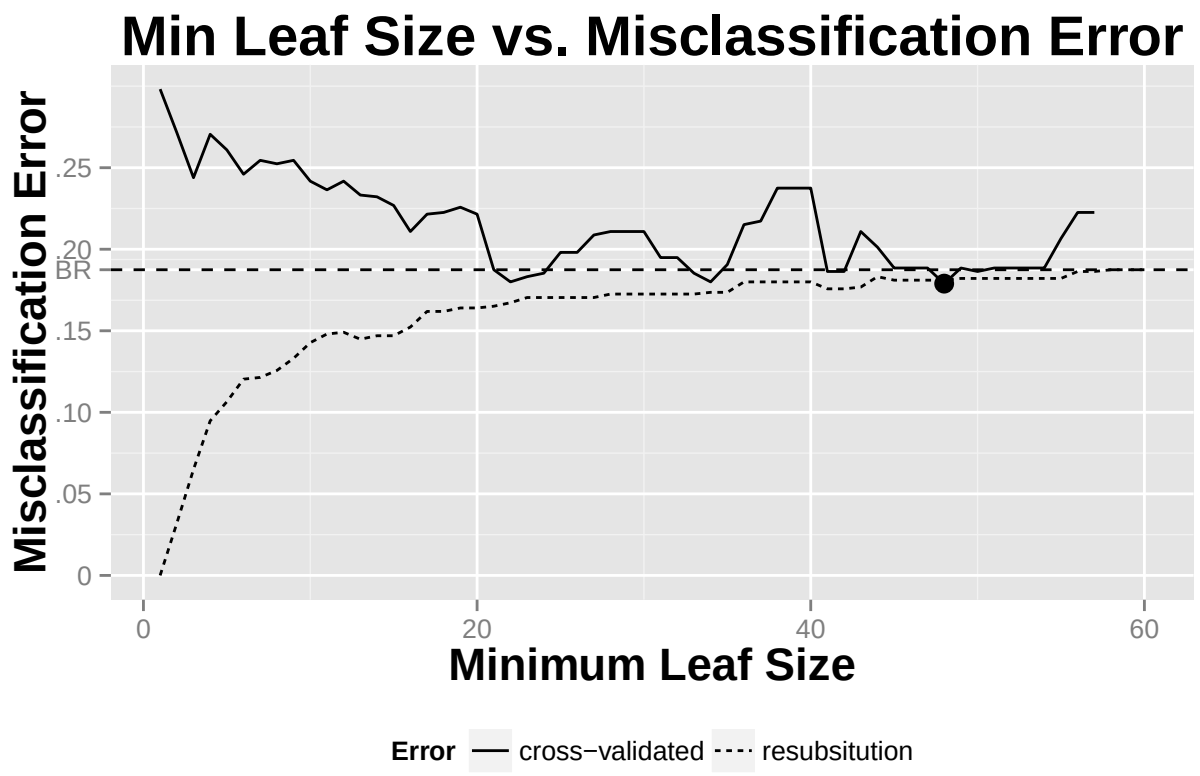
Using leave-one-out cross-validated error, we determine the minimum leaf size; the range of the minimum leaf size is $[1, 60]$. The minimum leaf size is specified with the `minbucket` statement and the `cp = 0` requests that the tree is not pruned. In addition we set the minimum split to be twice the minimum leaf (the minimum split is the minimum number needed to make a split; the default is three times the minimum leaf). Note that the relative error (`rel error`) and the expected error (`xerror`) are multiplied by the base rate giving, respectively, the estimated resubstitution and cross-validated error. The base rate is the root node error and the relative and expected errors are with respect to the root node error. Note that the cross-validated and resubstitution errors are from the *unpruned* trees. Missing values are treated as suggested by Breimen et al. (1984) and as was done by Monahan et al. (2001); this is automatically implemented in `rpart` with the `rpart.control` option `usesurrogate` set equal to 2 (default). The `xval` option is set equal to the sample size (this gives the estimate for leave-one-out cross-validation; the default is ten-fold).

```
CVerror = double() #cross-validated error
RSerror = double() #resubstitution error

for (j in 1:60) {

     t = rpart(Violence ~ ., COVRdata, control = rpart.control(minbucket = j,
                                                               minsplit = 2*j, cp = 0,
                                                               xval = nrow(COVRdata)))

     # error estimates
```

```
    CVerror[j] = t$cp[nrow(t$cp),4]*BR
    RSerror[j] = t$cp[nrow(t$cp),3]*BR

}

# create data frame with errors and leaf sizes
misclassError = data_frame('minleaf' = rep(1:60, 2), mcerror = c(RSerror, CVerror),
                           Error = rep(c('resubsitution', 'cross-validated'), each = 60))

# remove CVerror = 0
misclassError = misclassError %>%
    filter((mcerror != 0 & Error == 'cross-validated') | Error == 'resubsitution')

# optimal minimum leaf size
minLeaf = misclassError %>%
    filter(Error == 'cross-validated') %>%
    filter(rank(mcerror, ties.method = 'first') == 1)
minLeaf
```

```
Source: local data frame [1 x 3]

  minleaf mcerror           Error
1      48  0.1789 cross-validated
```

```
# plot error across minimum leaf size
ggplot(data = misclassError, aes(minleaf, mcerror, linetype = Error)) +
    geom_line() +
    geom_hline(yintercept = BR, lty = 2) +
    scale_y_continuous(breaks = c(0, .05, .1, .15, BR, .20, .25),
                       labels = c('0', '.05', '.10', '.15', 'BR', '.20','.25')) +
    theme(plot.title = element_text(size = 21, face = 'bold'),
          axis.title = element_text(size = 17, face = "bold")) +
    xlab('Minimum Leaf Size') + ylab('Misclassification Error') +
    labs(title = 'Min Leaf Size vs. Misclassification Error') +
    theme(legend.position = 'bottom') +
    geom_point(data = minLeaf, aes(y = mcerror, x = minleaf, size = 3), show_guide = F)
```



Now we construct a tree using the minimum leaf size that minimized the cross-validated error (this was found to be 48). A confusion matrix is given (the function `confsuionMatrix()` is available through the `caret` package). The tree is then plotted using the `prp()` function that is available through the `rpart.plot` package.

```
ctree = rpart(Violence ~ ., COVRdata,
                  control = rpart.control(minbucket = select(minLeaf, minleaf),
                                          minsplit = 2*select(minLeaf, minleaf), cp = 0))
# resubstitution error
mean(predict(ctree, COVRdata, type = 'class') != COVRdata$Violence)
```

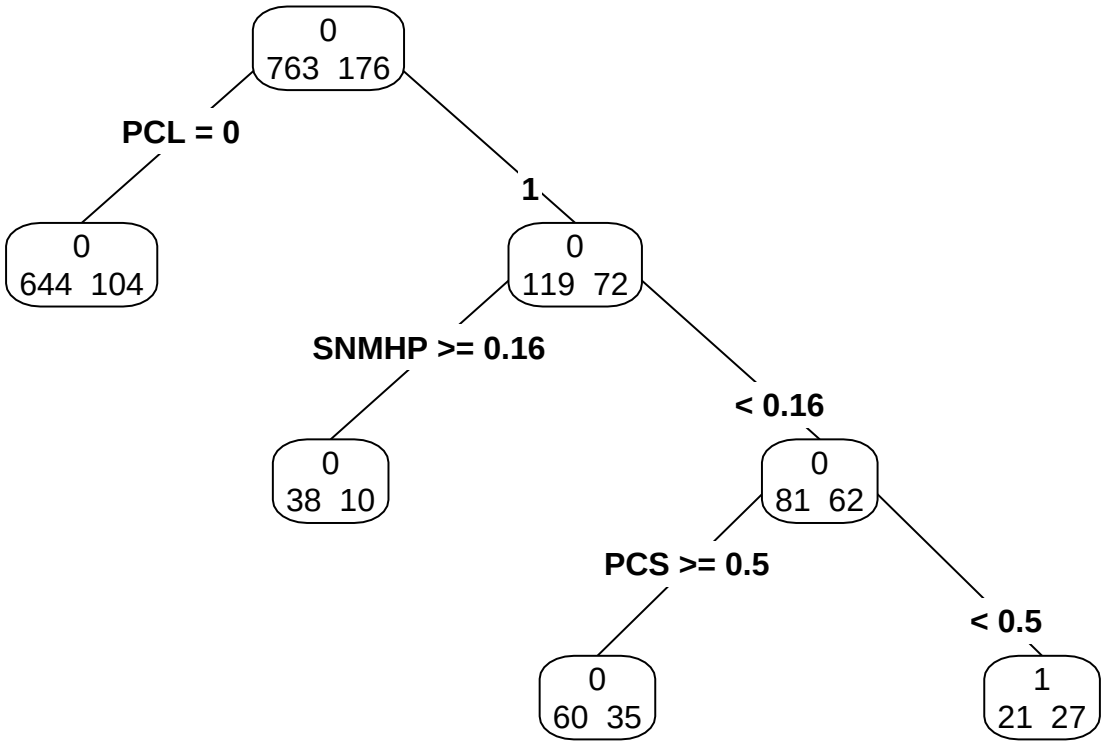
```
[1] 0.181
```

```
# confusion Matrix
confusionMatrix(predict(ctree, type = 'class'), COVRdata$Violence, positive = '1')
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 742 149
         1  21  27

               Accuracy : 0.819
                 95% CI : (0.793, 0.843)
```

```
    No Information Rate : 0.813
    P-Value [Acc > NIR] : 0.325

                  Kappa : 0.175
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.1534
            Specificity : 0.9725
         Pos Pred Value : 0.5625
         Neg Pred Value : 0.8328
             Prevalence : 0.1874
         Detection Rate : 0.0288
   Detection Prevalence : 0.0511
      Balanced Accuracy : 0.5629

       'Positive' Class : 1
```

```
# plot tree
prp(ctree, type = 4, extra = 1)
```

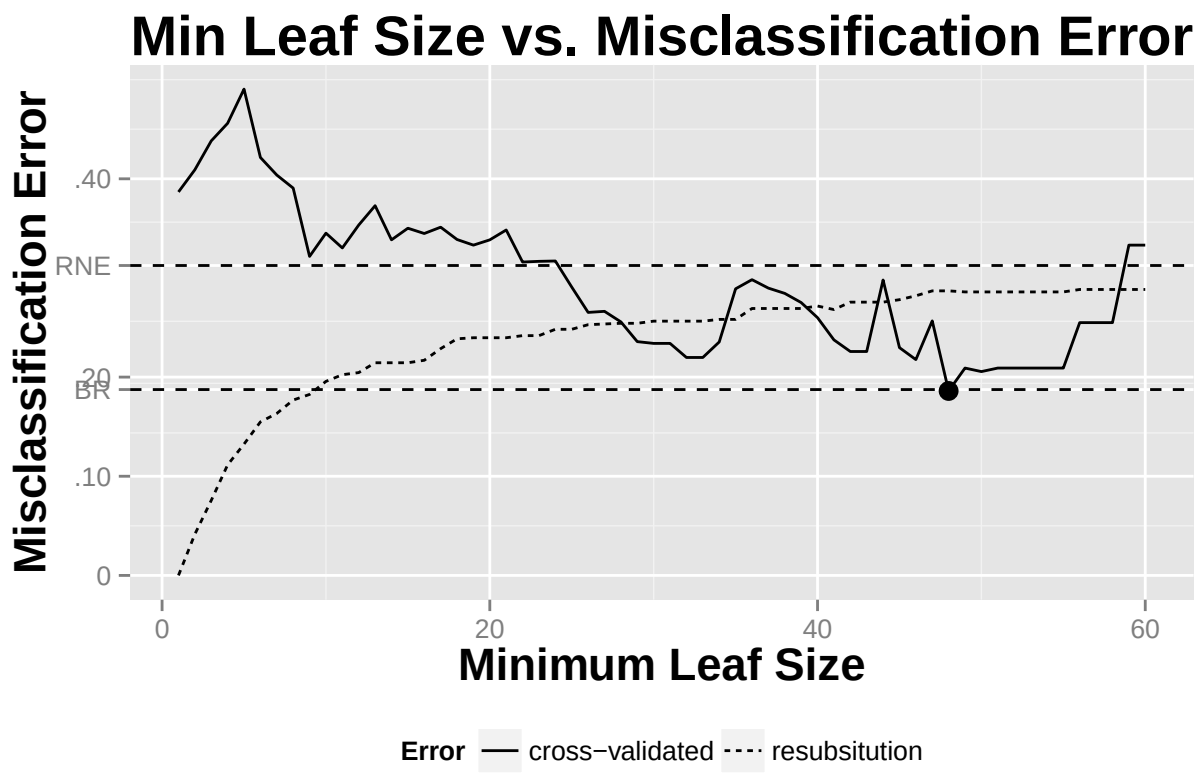


Next we implement unequal costs as implied by Monahan et al. (2001). We begin by creating a cost matrix; this is based on Monahan et al.'s (2001) choice of cutscore equal to 0.37.

```
# Clear the workspace.
rm(list = ls()[!(ls() %in% c('BR', 'COVRdata'))])
# cost matrix
twoBR = (1-2*BR)/(2*BR)
costMatrix = matrix(c(0, twoBR, 1, 0), 2)
```

We again determine the minimum leaf size, as was done with equal costs. Note that the root node error is 0.31 (= BR*twoBR; that is, the number of false negatives at the root note weighted by the cost of false negatives to false positives, 1.67).

```
CVerror = double() #cross-validated error
RSerror = double() #resubstitution error

for (j in 1:60) {

      t = rpart(Violence ~ ., COVRdata,  parms = list(loss = costMatrix),
                control = rpart.control(minbucket = j, minsplit = 2*j, cp = 0,
                                        xval = nrow(COVRdata)))

     # error estimates
     CVerror[j] = t$cp[nrow(t$cp),4]*BR*twoBR
     RSerror[j] = t$cp[nrow(t$cp),3]*BR*twoBR

}

# create data frame with error and leaf size
misclassError = data_frame('minleaf' = rep(1:60, 2), mcerror = c(RSerror, CVerror),
                           Error = rep(c('resubsitution', 'cross-validated'), each = 60))

# optimal minimum leaf size
minLeaf = misclassError %>%
     filter(Error == 'cross-validated') %>%
     filter(rank(mcerror, ties.method = 'first') == 1)

# plot error across minimum leaf size
ggplot(data = misclassError, aes(minleaf, mcerror, linetype = Error)) +
     geom_line() +
     geom_hline(yintercept = BR, linetype = 2) +
     geom_hline(yintercept = BR*twoBR, lty = 2) +
     scale_y_continuous(breaks = c(0, .1, BR, .2, BR*twoBR, .4),
                        labels = c('0', '.10', 'BR', '.20','RNE', '.40')) +
     theme(plot.title = element_text(size = 21, face = 'bold'),
           axis.title = element_text(size = 17, face = "bold")) +
     xlab('Minimum Leaf Size') + ylab('Misclassification Error') +
     labs(title = 'Min Leaf Size vs. Misclassification Error') +
     theme(legend.position = 'bottom') +
     geom_point(data = minLeaf, aes(y = mcerror, x = minleaf, size = 3), show_guide = F)
```



Next we construct and plot the tree using minimum leaf size, which was found to be 48. *This is exactly the same as the tree with equal costs* aside from one slight difference which does not affect the classification results: the non-terminal node at `SNMHP < .16` is classified as violent as opposed to nonviolent (as is the case when the costs were equal). A this node the proportion violent is 0.43 which is less than .5 but more than 0.37.

```
ctree = rpart(Violence ~ ., COVRdata, parms = list(loss = costMatrix),
                  control = rpart.control(minbucket = select(minLeaf, minleaf),
                                          minsplit = 2*select(minLeaf, minleaf), cp = 0,
                                          xval = nrow(COVRdata)))
# resubstitution error
mean(predict(ctree, COVRdata, type = 'class') != COVRdata$Violence)
```

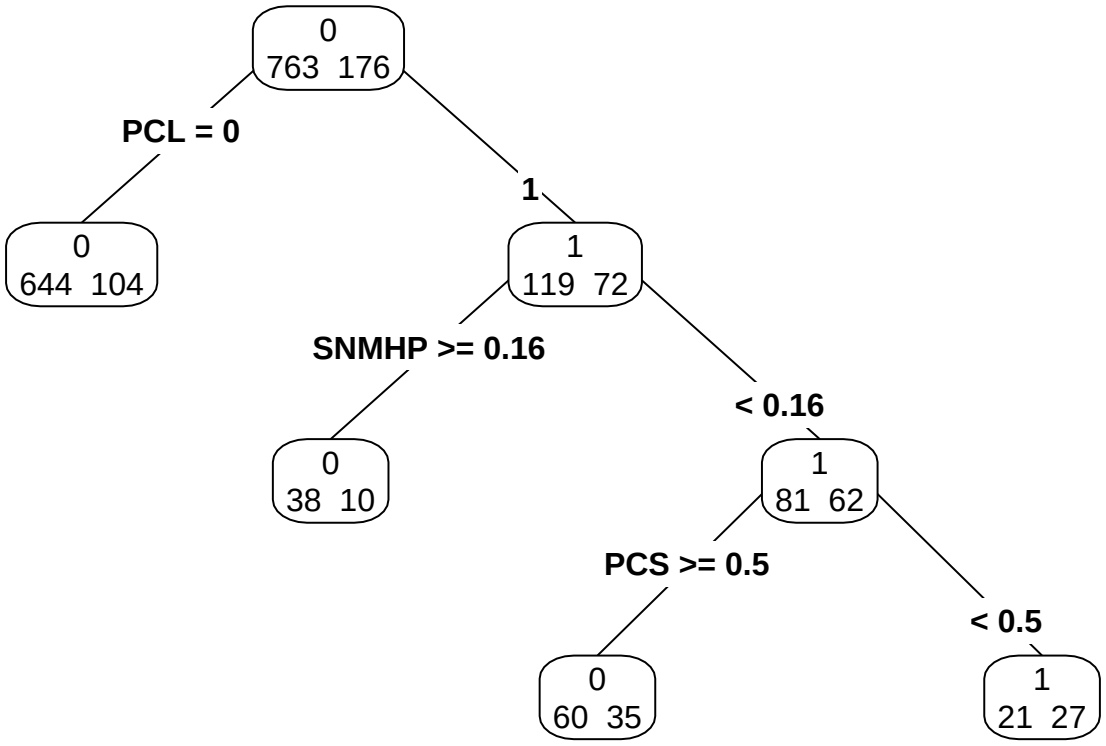
```
[1] 0.181
```

```
# confusion Matrix
confusionMatrix(predict(ctree, type = 'class'), COVRdata$Violence, positive = '1')
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 742 149
         1  21  27
```

```
               Accuracy : 0.819
                 95% CI : (0.793, 0.843)
    No Information Rate : 0.813
    P-Value [Acc > NIR] : 0.325

                  Kappa : 0.175
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.1534
            Specificity : 0.9725
         Pos Pred Value : 0.5625
         Neg Pred Value : 0.8328
             Prevalence : 0.1874
         Detection Rate : 0.0288
   Detection Prevalence : 0.0511
      Balanced Accuracy : 0.5629

       'Positive' Class : 1
```

```r
# plot tree
prp(ctree, type = 4, extra = 1)
```



Now we construct a tree using unequal costs as suggested in Berk (2012). Here, the minimum leaf size is chosen (arbitrarily) to be 30, and pruned. The relative error (`rel error`) and expected error (`xerror`) are not appropriate to use for calculating resubstitution error and cross-validated error here because they are

in relation to the root node error, which is itself in relation to the specified priors and costs. Instead, we implement the `train()` function, an extremely versatile cross-validation tool from the `caret()` package.

```
# clear workspace
rm(list = ls()[!(ls() %in% c('BR', 'COVRdata', 'costMatrix'))])
# construct tree
ctree = train(select(COVRdata, -Violence), COVRdata$Violence,
              method = 'rpart', parms = list(loss = matrix(c(0, 20, 1, 0), 2)),
              control = rpart.control(minbucket = 30, minsplit = 60),
              trControl = trainControl('cv', 'LOOCV'), tuneLength = 20)

# leave-one-out cross-validated error
1 - ctree$results$Accuracy[which.max(ctree$results$Accuracy)]
```

```
[1] 0.5857
```

```
# resubstitution error
mean(predict(ctree) != COVRdata$Violence)
```

```
[1] 0.5698
```

Finally, to demonstrate overfitting with classification trees, we fit a tree with a minimum leaf of one and without pruning.

```
rm(list = ls()[!(ls() %in% c('BR', 'COVRdata', 'costMatrix'))])
# equal costs
ctree = rpart(Violence ~ ., COVRdata, control = rpart.control(minbucket = 1, cp = 0,
                                                              xval = nrow(COVRdata)))

# leave-one-out cross-validated error
ctree$cp[nrow(ctree$cp),4]*BR
```

```
[1] 0.2961
```

```
# resubstitution error
ctree$cp[nrow(ctree$cp),3]*BR
```

```
[1] 0.00852
```

```
# Confusion Matrix
ctreePredClass = predict(ctree, COVRdata, type = 'class')
confusionMatrix(ctreePredClass, COVRdata$Violence, positive = '1')
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 761   6
         1   2 170
```

```
               Accuracy : 0.991
                 95% CI : (0.983, 0.996)
    No Information Rate : 0.813
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.972
 Mcnemar's Test P-Value : 0.289

            Sensitivity : 0.966
            Specificity : 0.997
         Pos Pred Value : 0.988
         Neg Pred Value : 0.992
             Prevalence : 0.187
         Detection Rate : 0.181
   Detection Prevalence : 0.183
      Balanced Accuracy : 0.982

       'Positive' Class : 1
```

```
# unequal costs
ctree2 = train(select(COVRdata, -Violence), COVRdata$Violence,
               method = 'rpart', parms = list(loss = costMatrix),
               control = rpart.control(minbucket = 1),
               trControl = trainControl('cv', 'LOOCV'), tuneGrid = expand.grid(.cp = 0))

# leave-one-out cross-validated error
1 - ctree2$results$Accuracy
```

```
[1] 0.279
```

```
# resubstitution error
mean(predict(ctree2) != COVRdata$Violence)
```

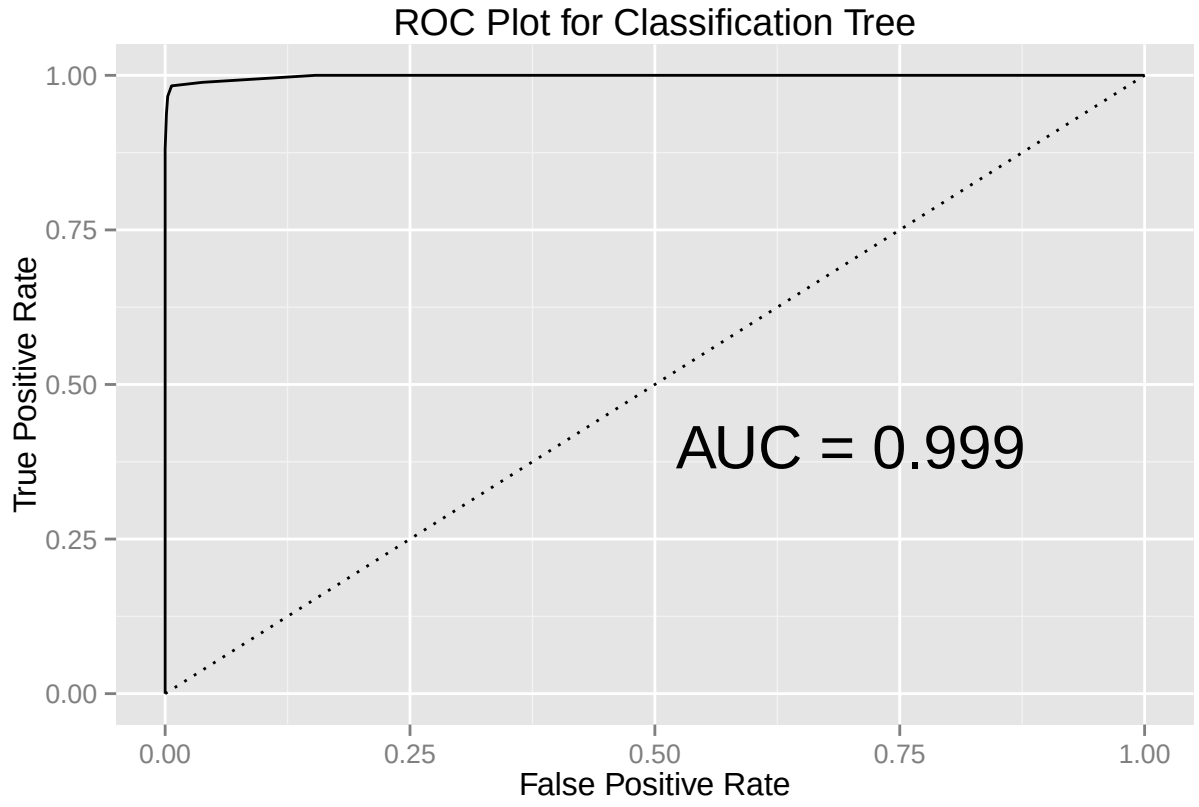
```
[1] 0.01278
```

```
# Confusion Matrix
confusionMatrix(predict(ctree2), COVRdata$Violence, positive = '1')
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 751   0
         1  12 176

               Accuracy : 0.987
                 95% CI : (0.978, 0.993)
    No Information Rate : 0.813
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.959
```

```
 Mcnemar's Test P-Value : 0.0015

            Sensitivity : 1.000
            Specificity : 0.984
         Pos Pred Value : 0.936
         Neg Pred Value : 1.000
             Prevalence : 0.187
         Detection Rate : 0.187
   Detection Prevalence : 0.200
      Balanced Accuracy : 0.992

       'Positive' Class : 1
```

```
# ROC plot
ctreePredProbs = predict(ctree, COVRdata)[,2]
ctreePreds = prediction(ctreePredProbs, select(COVRdata, Violence))
ctreePerf = performance(ctreePreds, 'tpr', 'fpr')
AUC = round(performance(ctreePreds, 'auc')@y.values[[1]], 3)
ggplot(data = NULL) +
     geom_line(aes(x = ctreePerf@x.values[[1]],
                                 y = ctreePerf@y.values[[1]])) +
     ggtitle('ROC Plot for Classification Tree') +
     xlab('False Positive Rate') +
     ylab('True Positive Rate') +
     geom_segment(aes(x = 0, y = 0, xend = 1, yend = 1),
                  linetype = 'dotted') +
     geom_text(aes(x = .7, y = .4, label = paste0('AUC = ', AUC), parse = T),
               size = 8)
```



Random Forests

The next step is to construct the random forest models. Begin by clearing the workspace. Random forest models can be generated using the `randomForest()` function from the `randomForest` package.

```
rm(list = ls()[!(ls() %in% c('BR', 'COVRdata'))])
```

First we impute missing data. The function `rfImpute()` first imputes missing data by using the mean (for continuous data) or the mode (for categorical data). Next, a random forest is fit to the new dataset that no longer contains missing data. A proximity matrix is calculated such that the (i, j)th entry contains the proportion of times that the ith and jth observation fall into the same terminal node. For continuous data, the missing values are imputed using the weighted (by proximity) average across the variable of the observations; for categorical data, the missing values are equal to the value with the largest proximity, averaged across all the observations for the variable in question. Random seeds are set for replication of results.

```
set.seed(917)
COVRdataRF = rfImpute(Violence ~ ., COVRdata)
```

```
ntree      OOB      1      2
  300:  18.32%  0.92% 93.75%
ntree      OOB      1      2
  300:  18.32%  1.18% 92.61%
ntree      OOB      1      2
  300:  18.74%  1.83% 92.05%
```

```
ntree      OOB      1      2
  300:  19.06%  2.10% 92.61%
ntree      OOB      1      2
  300:  18.85%  1.57% 93.75%
```

The data are split into a training set and testing set. The testing set contains 282 (30%) observations; the training set contains the remaining 657 (70%) observations.

```
set.seed(1983)
COVRtrain = sample_frac(COVRdataRF, .7)
COVRtest = COVRdataRF %>%
     filter(!(row_number() %in% rownames(COVRtrain)))
```

Here, the random forest model is constructed using equal costs.

```
set.seed(91783)
covrRF = randomForest(Violence ~ ., data = COVRtrain, ntree = 1000)
```

A confusion matrix for the training data is created; the misclassification error (i.e., the proportion along the off-diagonal) is the resubstitution error.

```
confusionMatrix(predict(covrRF, COVRtrain), COVRtrain$Violence, positive = '1')
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 538   0
         1   0 119

               Accuracy : 1
                 95% CI : (0.994, 1)
    No Information Rate : 0.819
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 1
 Mcnemar's Test P-Value : NA

            Sensitivity : 1.000
            Specificity : 1.000
         Pos Pred Value : 1.000
         Neg Pred Value : 1.000
             Prevalence : 0.181
         Detection Rate : 0.181
   Detection Prevalence : 0.181
      Balanced Accuracy : 1.000

       'Positive' Class : 1
```

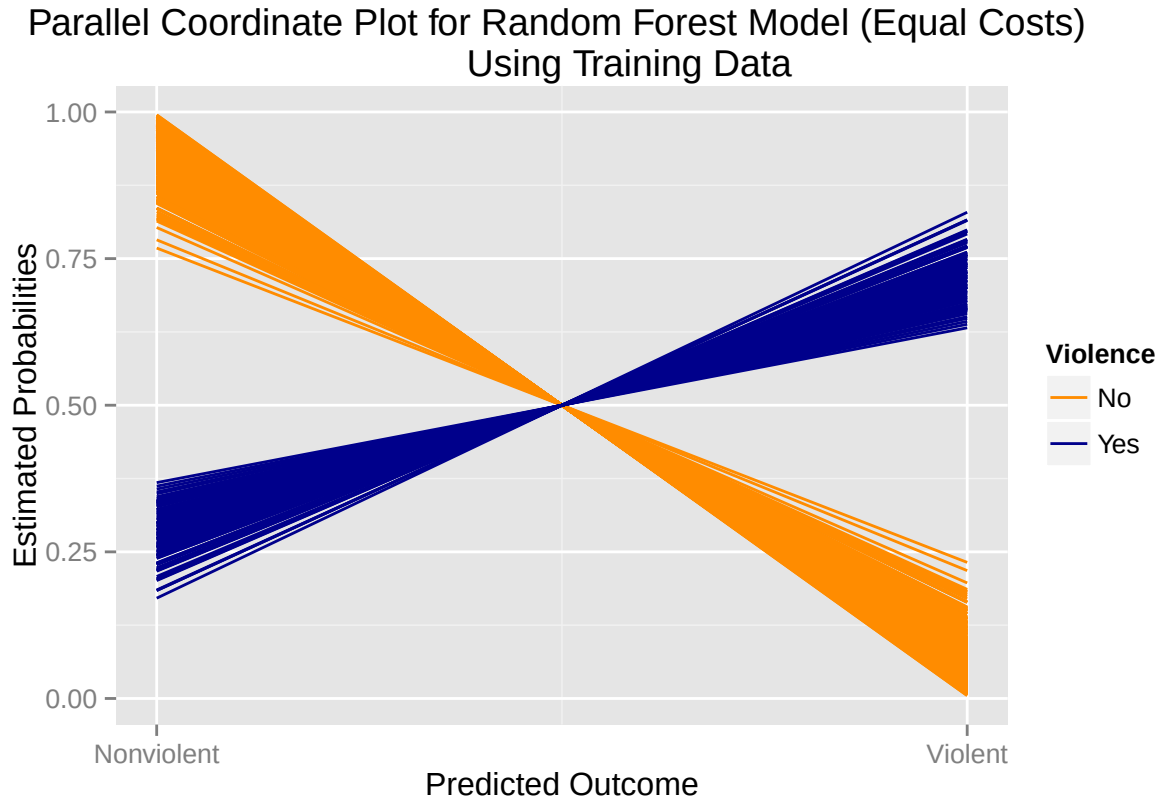
The cross-validated error can be estimated using the testing dataset.

```
confusionMatrix(predict(covrRF, COVRtest), COVRtest$Violence, positive = '1')
```

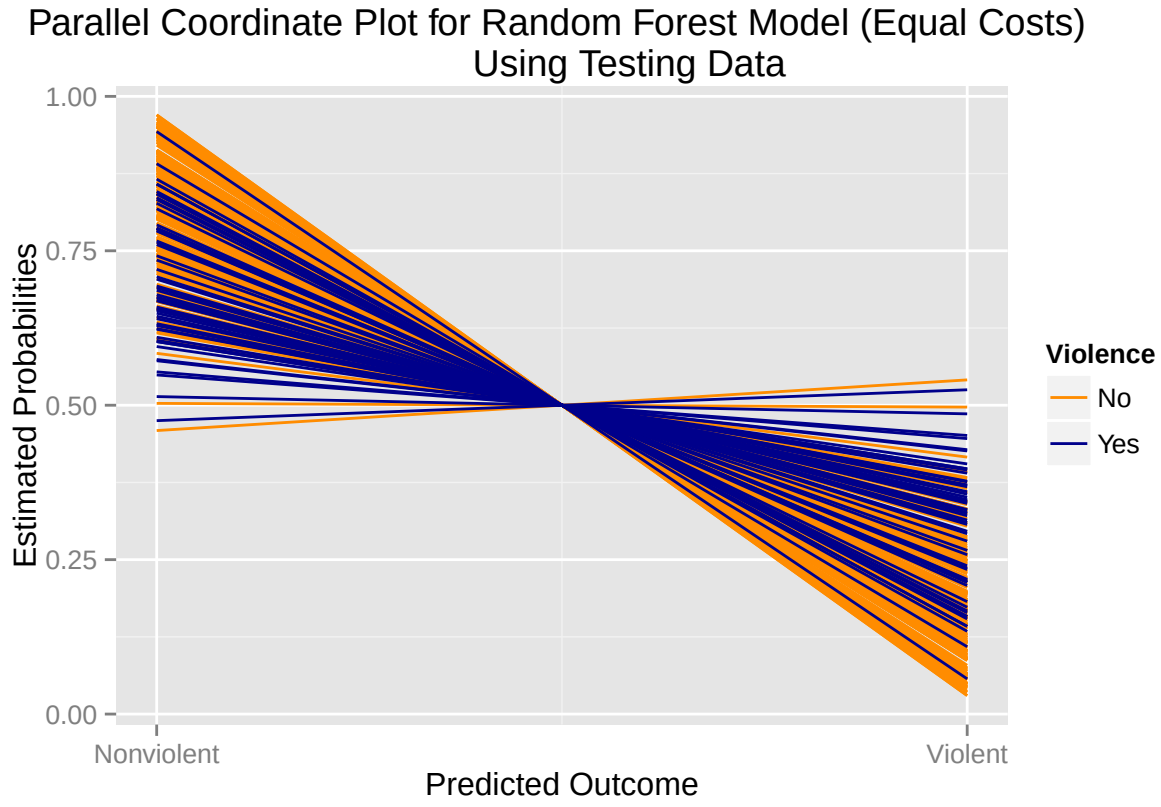
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 224  56
         1   1   1

               Accuracy : 0.798
                 95% CI : (0.746, 0.843)
    No Information Rate : 0.798
    P-Value [Acc > NIR] : 0.535

                  Kappa : 0.02
 Mcnemar's Test P-Value : 8.52e-13

            Sensitivity : 0.01754
            Specificity : 0.99556
         Pos Pred Value : 0.50000
         Neg Pred Value : 0.80000
             Prevalence : 0.20213
         Detection Rate : 0.00355
   Detection Prevalence : 0.00709
      Balanced Accuracy : 0.50655

       'Positive' Class : 1
```

To visualize the separation between the violent and non-violent individuals, we construct parallel coordinate plots for the training data and the testing data.

```
# data frame of estimated probabilities of violence and actual classification
rfPreds = data.frame(Preds = predict(covrRF, COVRtrain, type = 'prob'),
                     select(COVRtrain, Violence))
rfPreds = arrange(rfPreds, Preds.0)
ggplot(data = rfPreds) +
     geom_segment(aes(x = 0, xend = 1, y = Preds.0, yend = Preds.1, color = Violence)) +
     scale_x_continuous(breaks = c(0, 1),
                        labels = c('Nonviolent', 'Violent')) +
     xlab('Predicted Outcome') +
     ylab('Estimated Probabilities') +
     scale_color_manual(values = c('darkorange', 'darkblue'),
                        labels = c('No', 'Yes')) +
     ggtitle('Parallel Coordinate Plot for Random Forest Model (Equal Costs)
             Using Training Data')
```




```
# remove rfPreds
rm(rfPreds)

# data frame of estimated probabilities of violence and actual classification
rfPreds = data.frame(Preds = predict(covrRF, COVRtest, type = 'prob'),
                     select(COVRtest, Violence))
rfPreds = arrange(rfPreds, Preds.0)
ggplot(data = rfPreds) +
     geom_segment(aes(x = 0, xend = 1, y = Preds.0, yend = Preds.1, color = Violence)) +
     scale_x_continuous(breaks = c(0, 1),
                        labels = c('Nonviolent', 'Violent')) +
     xlab('Predicted Outcome') +
     ylab('Estimated Probabilities') +
     scale_color_manual(values = c('darkorange', 'darkblue'),
                        labels = c('No', 'Yes')) +
     ggtitle('Parallel Coordinate Plot for Random Forest Model (Equal Costs)
             Using Testing Data')
```



Now the random forest model is constructed using unequal costs.

```
rm(covrRF, rfPreds)
set.seed(917)
covrRF = randomForest(Violence ~ ., data = COVRtrain, ntree = 1000,
                      cutoff = c(1-2*BR, 2*BR))
```

The misclassification rates:

```
# resubstitution error
confusionMatrix(predict(covrRF, COVRtrain), COVRtrain$Violence, positive = '1')
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 538   0
         1   0 119

               Accuracy : 1
                 95% CI : (0.994, 1)
    No Information Rate : 0.819
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 1
```

```
 Mcnemar's Test P-Value : NA

            Sensitivity : 1.000
            Specificity : 1.000
         Pos Pred Value : 1.000
         Neg Pred Value : 1.000
             Prevalence : 0.181
         Detection Rate : 0.181
   Detection Prevalence : 0.181
      Balanced Accuracy : 1.000

       'Positive' Class : 1
```

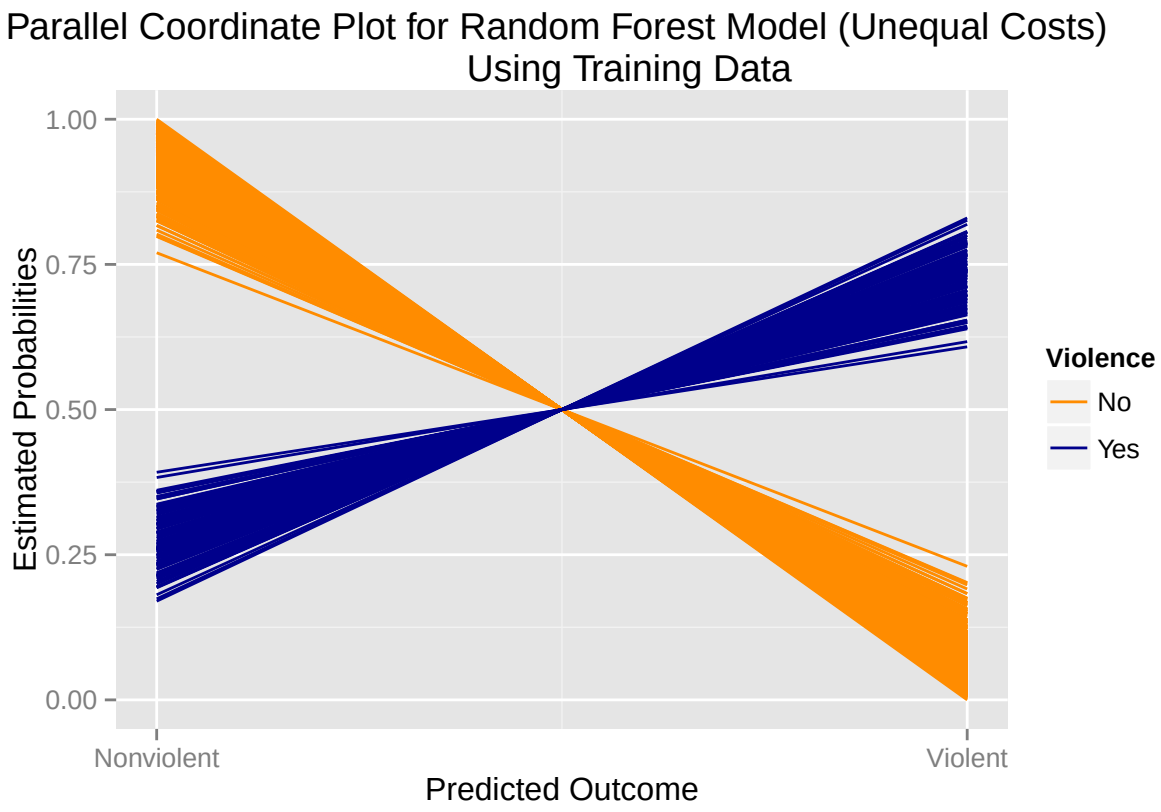
```
# cross-validated error
confusionMatrix(predict(covrRF, COVRtest), COVRtest$Violence, positive = '1')
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 217  42
         1   8  15

               Accuracy : 0.823
                 95% CI : (0.773, 0.865)
    No Information Rate : 0.798
    P-Value [Acc > NIR] : 0.168

                  Kappa : 0.293
 Mcnemar's Test P-Value : 3.06e-06

            Sensitivity : 0.2632
            Specificity : 0.9644
         Pos Pred Value : 0.6522
         Neg Pred Value : 0.8378
             Prevalence : 0.2021
         Detection Rate : 0.0532
   Detection Prevalence : 0.0816
      Balanced Accuracy : 0.6138

       'Positive' Class : 1
```

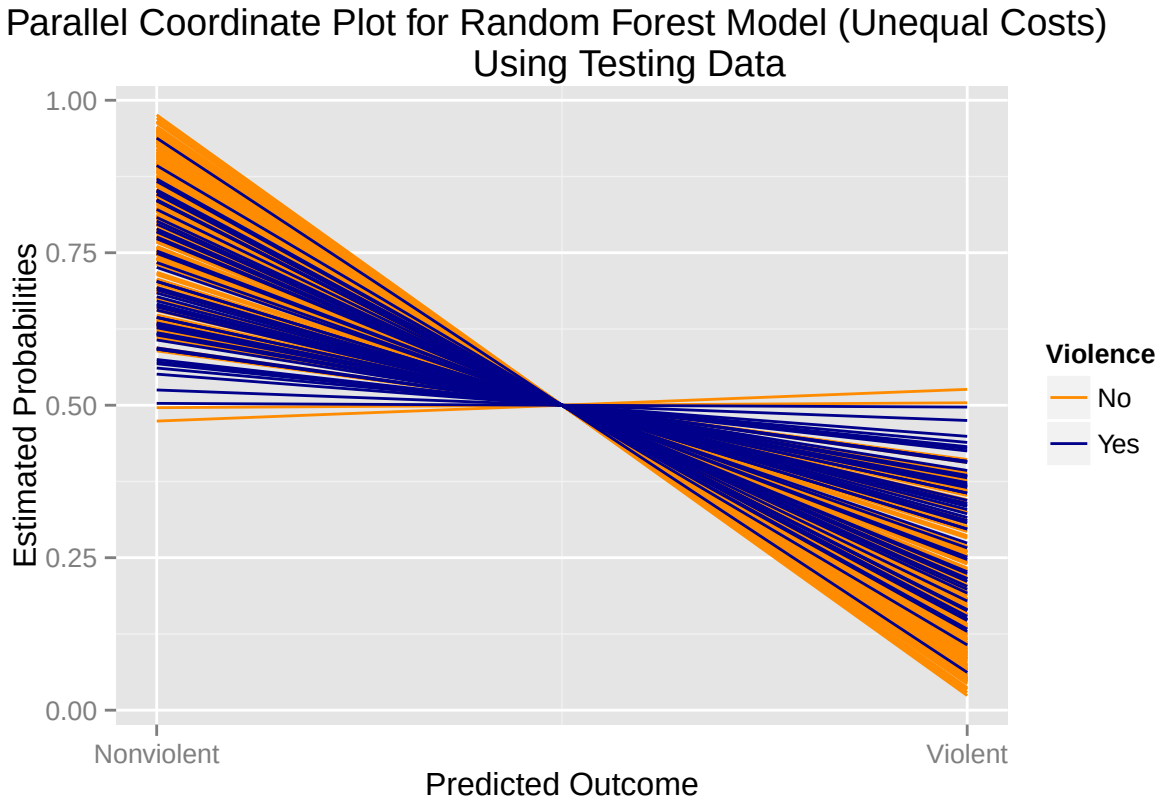
And the parallel coordinate plots:

```
# data frame of estimated probabilities of violence and actual classification
rfPreds = data.frame(Preds = predict(covrRF, COVRtrain, type = 'prob'),
                     select(COVRtrain, Violence))
rfPreds = arrange(rfPreds, Preds.0)
ggplot(data = rfPreds) +
     geom_segment(aes(x = 0, xend = 1, y = Preds.0, yend = Preds.1, color = Violence)) +
     scale_x_continuous(breaks = c(0, 1),
```

```
                        labels = c('Nonviolent', 'Violent')) +
    xlab('Predicted Outcome') +
    ylab('Estimated Probabilities') +
    scale_color_manual(values = c('darkorange', 'darkblue'),
                        labels = c('No', 'Yes')) +
    ggtitle('Parallel Coordinate Plot for Random Forest Model (Unequal Costs)
            Using Training Data')
```




```
# remove rfPreds
rm(rfPreds)

# data frame of estimated probabilities of violence and actual classification
rfPreds = data.frame(Preds = predict(covrRF, COVRtest, type = 'prob'),
                     select(COVRtest, Violence))
rfPreds = arrange(rfPreds, Preds.0)
ggplot(data = rfPreds) +
    geom_segment(aes(x = 0, xend = 1, y = Preds.0, yend = Preds.1, color = Violence)) +
    scale_x_continuous(breaks = c(0, 1),
                        labels = c('Nonviolent', 'Violent')) +
    xlab('Predicted Outcome') +
    ylab('Estimated Probabilities') +
    scale_color_manual(values = c('darkorange', 'darkblue'),
                        labels = c('No', 'Yes')) +
    ggtitle('Parallel Coordinate Plot for Random Forest Model (Unequal Costs)
            Using Testing Data')
```



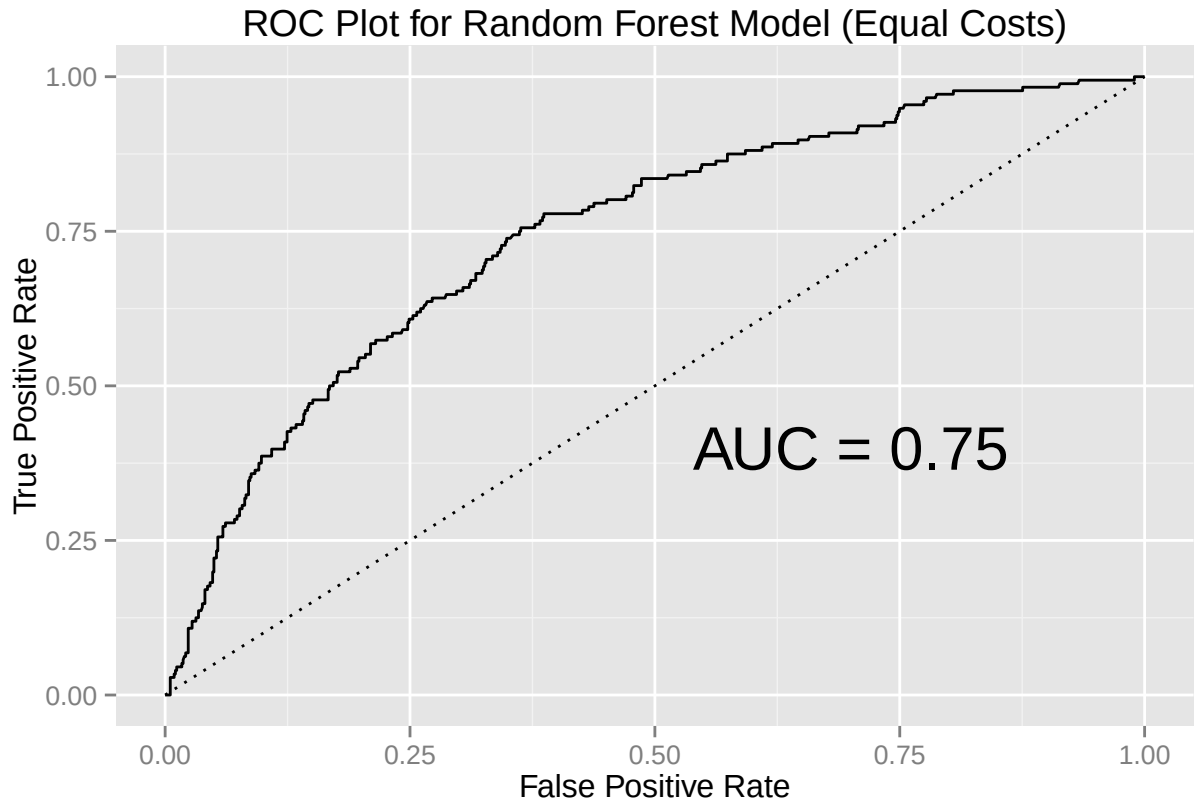
Now a random forest model with all the data and using the OOB error to estimate test error. First with equal costs.

```
# Clear workspace
rm(list = ls()[!(ls() %in% c('BR', 'COVRdata', 'COVRdataRF'))])
# Equal costs
set.seed(1983917)
covrRF = randomForest(Violence ~ ., data = COVRdataRF, ntree = 1000)
confusionMatrix((covrRF$votes > .5)[,2], COVRdataRF$Violence == 1, positive = 'TRUE')
```

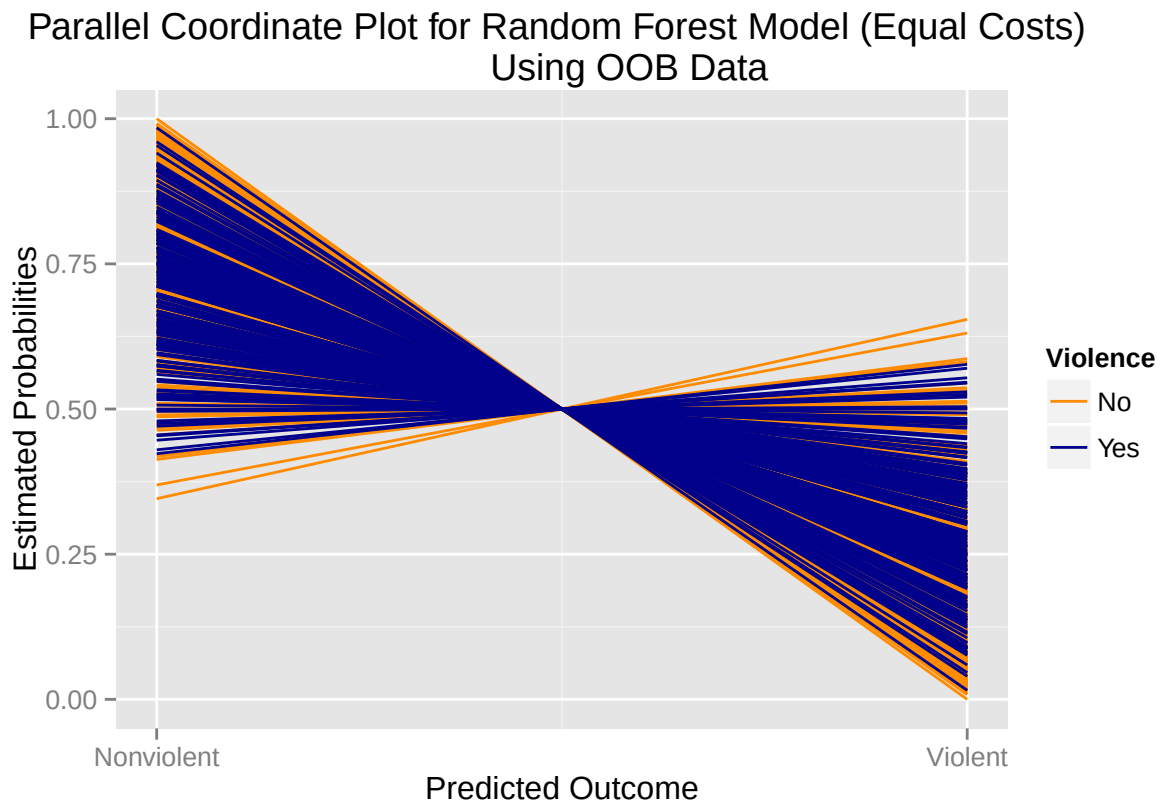
```
Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE   749  166
     TRUE     14   10

               Accuracy : 0.808
                 95% CI : (0.782, 0.833)
    No Information Rate : 0.813
    P-Value [Acc > NIR] : 0.649

                  Kappa : 0.058
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.0568
```

```
            Specificity : 0.9817
         Pos Pred Value : 0.4167
         Neg Pred Value : 0.8186
             Prevalence : 0.1874
         Detection Rate : 0.0106
   Detection Prevalence : 0.0256
      Balanced Accuracy : 0.5192

       'Positive' Class : TRUE
```

```
# ROC plot
rfPreds = prediction(covrRF$votes[,2], select(COVRdataRF, Violence))
rfPerf = performance(rfPreds, 'tpr', 'fpr')
AUC = round(performance(rfPreds, 'auc')@y.values[[1]], 2)
ggplot(data = NULL) +
     geom_line(aes(x = rfPerf@x.values[[1]],
                                y = rfPerf@y.values[[1]])) +
     ggtitle('ROC Plot for Random Forest Model (Equal Costs)') +
     xlab('False Positive Rate') +
     ylab('True Positive Rate') +
     geom_segment(aes(x = 0, y = 0, xend = 1, yend = 1),
                  linetype = 'dotted') +
     geom_text(aes(x = .7, y = .4, label = paste0('AUC = ', AUC), parse = T),
               size = 8)
```



```
# Parallel coordinate plot
rfPreds = data.frame(Preds = covrRF$votes, select(COVRdataRF, Violence))
rfPreds = arrange(rfPreds, Preds.0)
ggplot(data = rfPreds) +
     geom_segment(aes(x = 0, xend = 1, y = Preds.0, yend = Preds.1, color = Violence)) +
     scale_x_continuous(breaks = c(0, 1),
                        labels = c('Nonviolent', 'Violent')) +
     xlab('Predicted Outcome') +
     ylab('Estimated Probabilities') +
     scale_color_manual(values = c('darkorange', 'darkblue'),
                        labels = c('No', 'Yes')) +
     ggtitle('Parallel Coordinate Plot for Random Forest Model (Equal Costs)
             Using OOB Data')
```



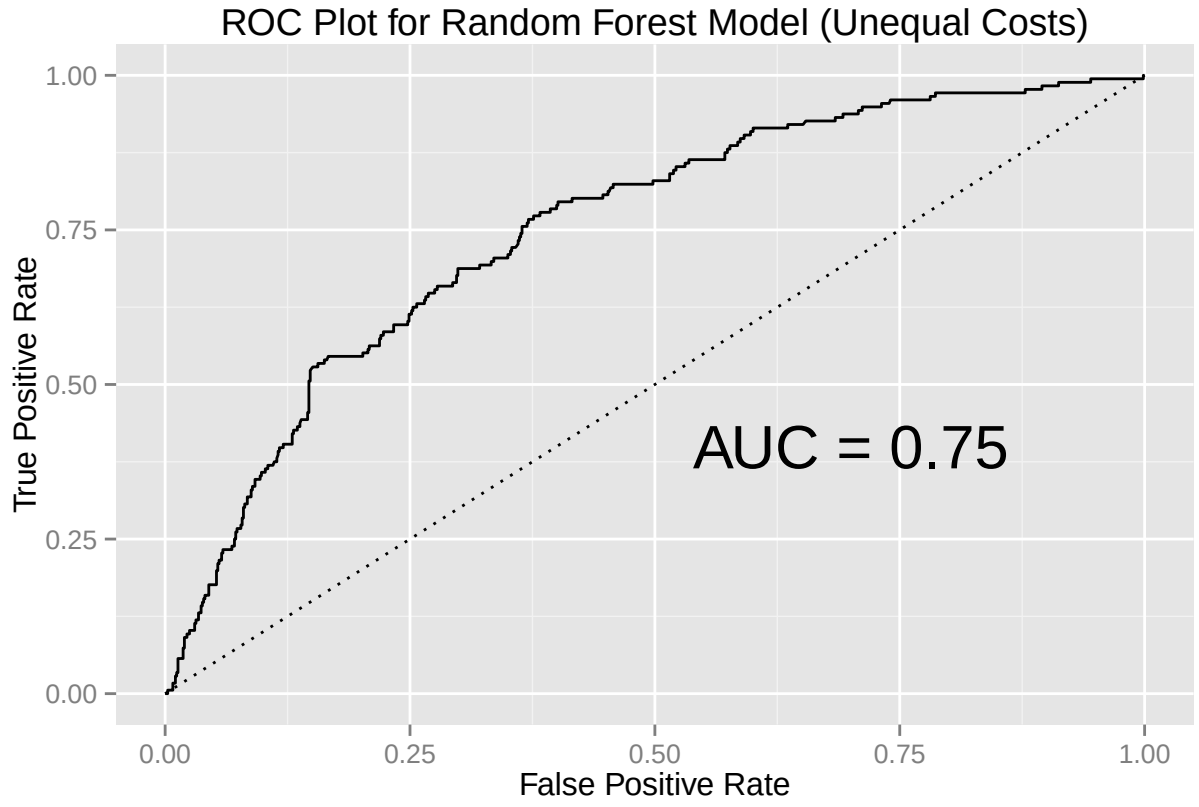
Now with unequal costs.

```
rm(covrRF, rfPreds, rfPerf, AUC)
set.seed(1983917)
covrRF = randomForest(Violence ~ ., data = COVRdataRF, ntree = 1000,
                      cutoff = c(1-2*BR, 2*BR))
confusionMatrix((covrRF$votes > .5)[,2], COVRdataRF$Violence == 1, positive = 'TRUE')
```

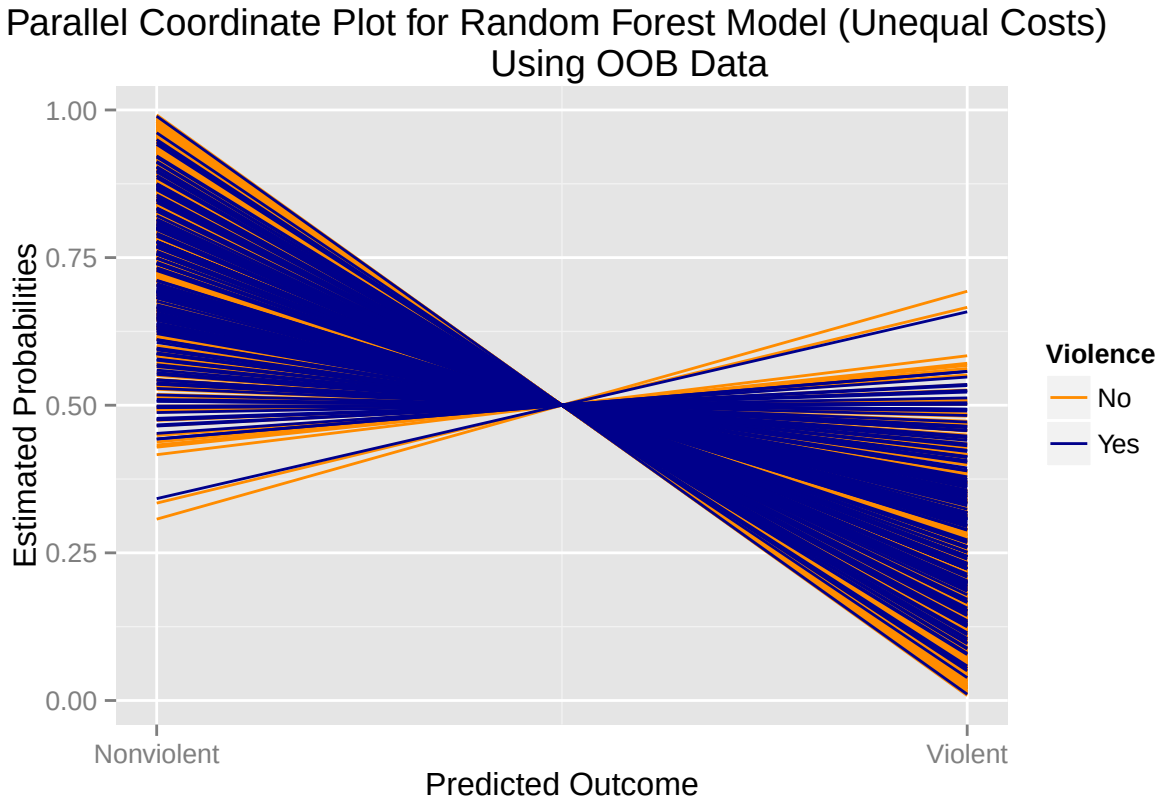
```
Confusion Matrix and Statistics

          Reference
```

```
Prediction FALSE TRUE
     FALSE   748  163
     TRUE     15   13

               Accuracy : 0.81
                 95% CI : (0.784, 0.835)
    No Information Rate : 0.813
    P-Value [Acc > NIR] : 0.586

                  Kappa : 0.08
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.0739
            Specificity : 0.9803
         Pos Pred Value : 0.4643
         Neg Pred Value : 0.8211
             Prevalence : 0.1874
         Detection Rate : 0.0138
   Detection Prevalence : 0.0298
      Balanced Accuracy : 0.5271

       'Positive' Class : TRUE
```

```
# ROC plot
rfPreds = prediction(covrRF$votes[,2], select(COVRdataRF, Violence))
rfPerf = performance(rfPreds, 'tpr', 'fpr')
AUC = round(performance(rfPreds, 'auc')@y.values[[1]], 2)
ggplot(data = NULL) +
     geom_line(aes(x = rfPerf@x.values[[1]],
                                y = rfPerf@y.values[[1]])) +
     ggtitle('ROC Plot for Random Forest Model (Unequal Costs)') +
     xlab('False Positive Rate') +
     ylab('True Positive Rate') +
     geom_segment(aes(x = 0, y = 0, xend = 1, yend = 1),
                  linetype = 'dotted') +
     geom_text(aes(x = .7, y = .4, label = paste0('AUC = ', AUC), parse = T),
               size = 8)
```

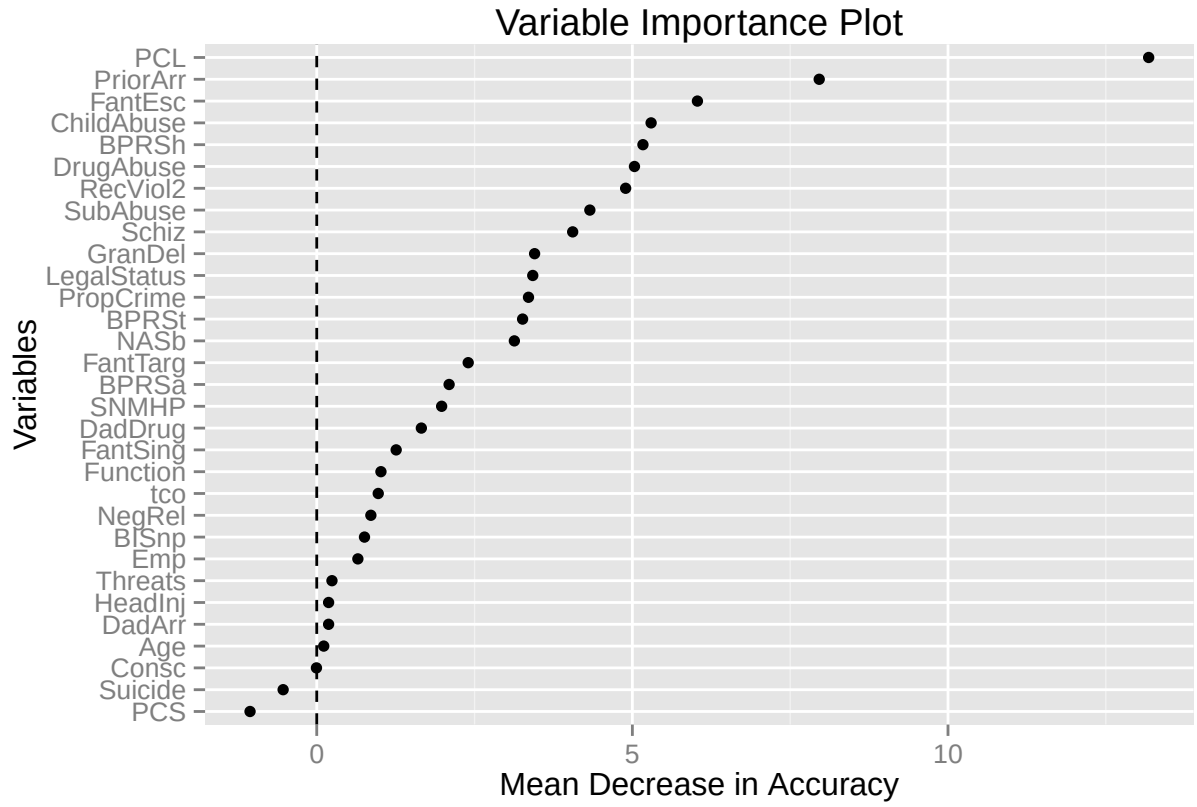



```
# Parallel coordinate plot
rfPreds = data.frame(Preds = covrRF$votes, select(COVRdataRF, Violence))
rfPreds = arrange(rfPreds, Preds.0)
ggplot(data = rfPreds) +
     geom_segment(aes(x = 0, xend = 1, y = Preds.0, yend = Preds.1, color = Violence)) +
     scale_x_continuous(breaks = c(0, 1),
                        labels = c('Nonviolent', 'Violent')) +
     xlab('Predicted Outcome') +
     ylab('Estimated Probabilities') +
     scale_color_manual(values = c('darkorange', 'darkblue'),
                        labels = c('No', 'Yes')) +
     ggtitle('Parallel Coordinate Plot for Random Forest Model (Unequal Costs)
             Using OOB Data')
```



Finally, using the OOB error to measure variable importance, we use a subset of variables for prediction, eliminating those that negatively affect or add very little to the accuracy. This is the "final model."

```
set.seed(1983917)
# measuring variable importance
covrRF = randomForest(Violence ~ ., data = COVRdataRF, ntrees = 1000, imp = T)

# variable importance plots
varImp = data_frame(Variables = rownames(importance(covrRF)),
                    Accuracy = importance(covrRF)[,3])
ggplot(data = varImp, aes(Accuracy, reorder(Variables, Accuracy))) +
     geom_point() +
     geom_vline(xintercept = 0, lty = 2) +
     xlab('Mean Decrease in Accuracy') +
     ylab('Variables') +
     ggtitle('Variable Importance Plot')
```

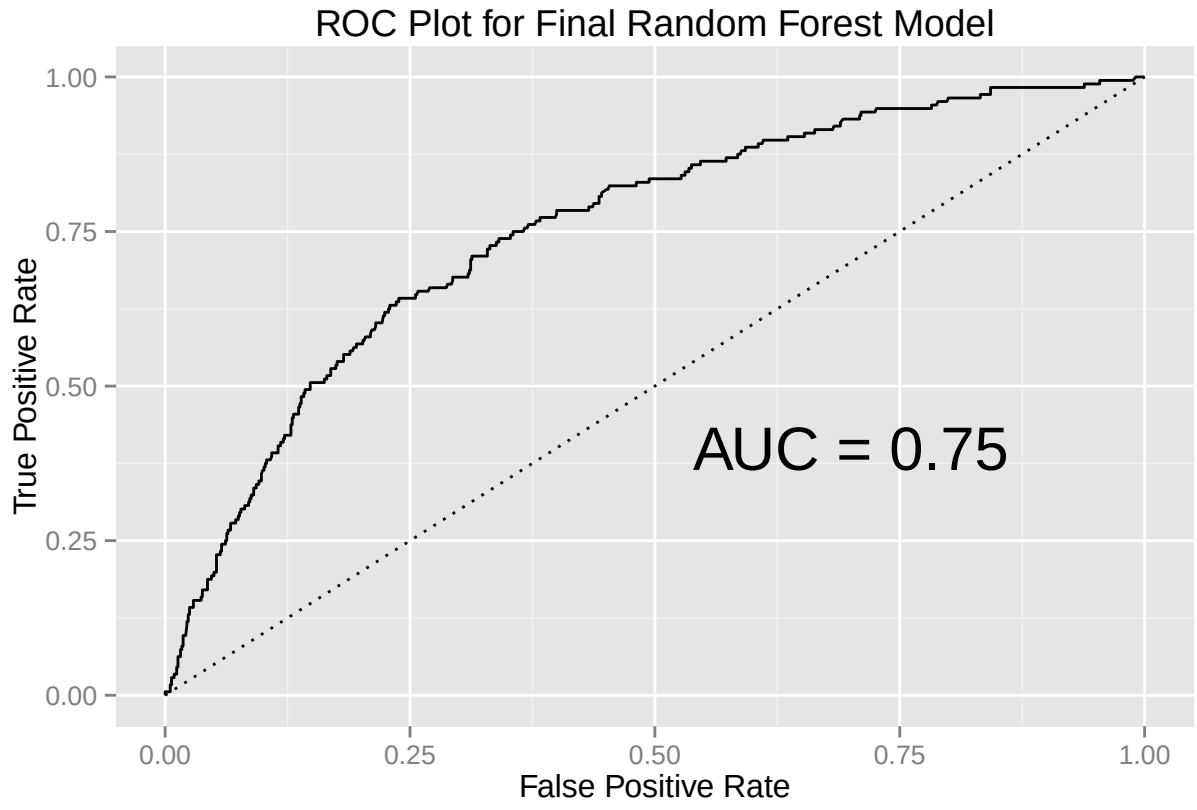



```
# select subset of variables that whose mean decrease in accuracy is greater than .5
COVRdataRF_Imp = select(COVRdataRF, Violence, (2:32)[select(varImp, Accuracy) > .5])

# final model
set.seed(1983917)
covrRF = randomForest(Violence ~ ., data = COVRdataRF_Imp, ntrees = 1000)
confusionMatrix((covrRF$votes > .5)[,2], COVRdataRF$Violence == 1, positive = 'TRUE')
```

```
Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE   747  159
     TRUE     16   17

               Accuracy : 0.814
                 95% CI : (0.787, 0.838)
    No Information Rate : 0.813
    P-Value [Acc > NIR] : 0.487

                  Kappa : 0.11
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.0966
            Specificity : 0.9790
         Pos Pred Value : 0.5152
```

```
         Neg Pred Value : 0.8245
             Prevalence : 0.1874
         Detection Rate : 0.0181
   Detection Prevalence : 0.0351
      Balanced Accuracy : 0.5378

       'Positive' Class : TRUE
```

```
# ROC plot
rfPreds = prediction(covrRF$votes[,2], select(COVRdataRF, Violence))
rfPerf = performance(rfPreds, 'tpr', 'fpr')
AUC = round(performance(rfPreds, 'auc')@y.values[[1]], 2)
ggplot(data = NULL) +
     geom_line(aes(x = rfPerf@x.values[[1]],
                                 y = rfPerf@y.values[[1]])) +
     ggtitle('ROC Plot for Final Random Forest Model') +
     xlab('False Positive Rate') +
     ylab('True Positive Rate') +
     geom_segment(aes(x = 0, y = 0, xend = 1, yend = 1),
                  linetype = 'dotted') +
     geom_text(aes(x = .7, y = .4, label = paste0('AUC = ', AUC), parse = T),
               size = 8)
```



```
# Parallel coordinate plot
rfPreds = data.frame(Preds = covrRF$votes, select(COVRdataRF, Violence))
rfPreds = arrange(rfPreds, Preds.0)
ggplot(data = rfPreds) +
     geom_segment(aes(x = 0, xend = 1, y = Preds.0, yend = Preds.1, color = Violence)) +
     scale_x_continuous(breaks = c(0, 1),
                        labels = c('Nonviolent', 'Violent')) +
     xlab('Predicted Outcome') +
     ylab('Estimated Probabilities') +
     scale_color_manual(values = c('darkorange', 'darkblue'),
                        labels = c('No', 'Yes')) +
     ggtitle('Parallel Coordinate Plot for Final Random Forest Model')
```