Homework Assignment 3

Parker Reedy

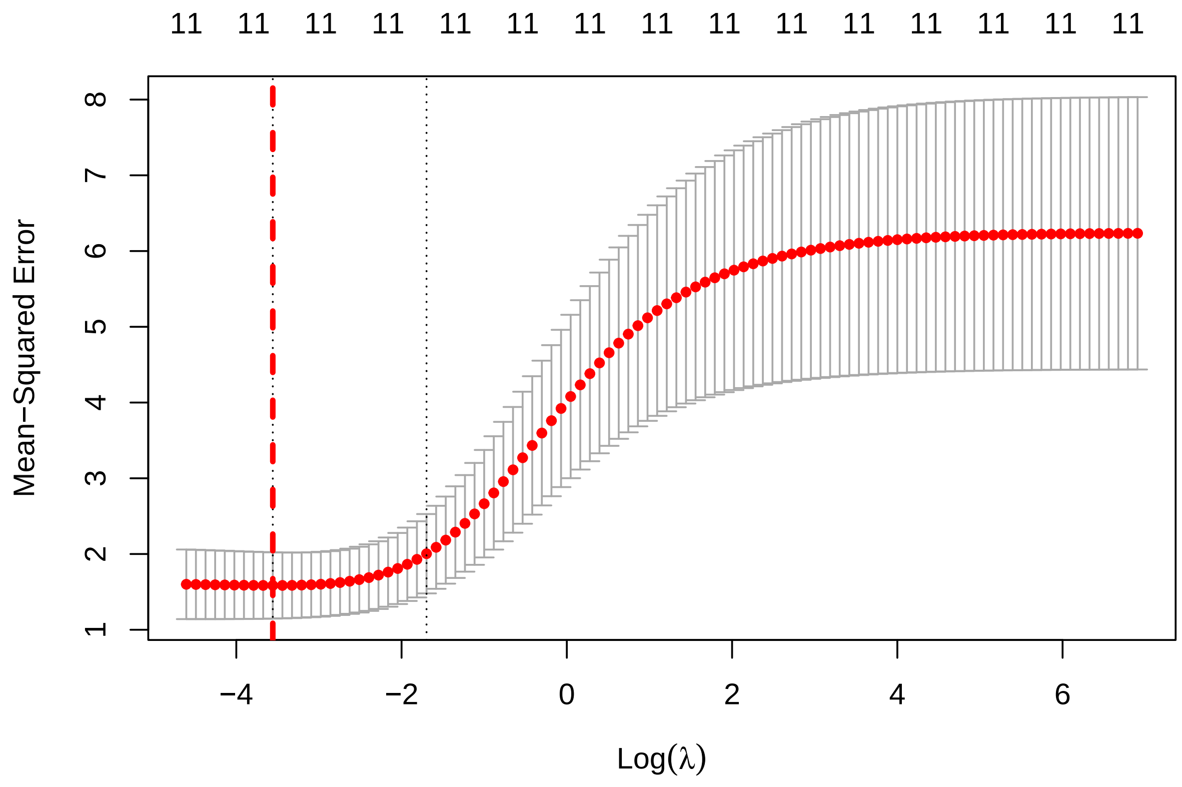
November 26, 2023

1.

```r
set.seed(123)

dat <- model.matrix(Sales~., Carseats)[, -1]
train = sample(nrow(dat), 30)
x.train = dat[train, ]
y.train = Carseats[train, ]$Sales

# The rest as test data
x.test = dat[-train, ]
y.test = Carseats[-train, ]$Sales
```

a. The best lambda is 0.02848

```r
set.seed(123)

lambda.list.ridge = 1000 * exp(seq(0, log(1e-5), length = 100))

cv.ridge.out = cv.glmnet(x.train, y.train, alpha=0, lambda = lambda.list.ridge)

plot(cv.ridge.out) + abline(v = log(cv.ridge.out$lambda.min), col="red", lwd=3, lty=2)
```



```
## integer(0)
```

```
bestlam = cv.ridge.out$lambda.min
bestlam
```

```
## [1] 0.02848
```

```
out = glmnet(x.train,y.train,alpha=0)
predict(out, type='coefficients', s=bestlam)[1:12,]
```

```
##     (Intercept)       CompPrice          Income     Advertising      Population
##       6.5730326       0.0938637       0.0075634       0.0701387      -0.0001237
##           Price   ShelveLocGood ShelveLocMedium             Age       Education
##      -0.0845304       4.0438779       1.6198782      -0.0527367      -0.0939303
##        UrbanYes           USYes
##       0.5532566      -0.0179076
```

b. The test MSE is 1.46 and the training MSE is 0.507. This is a fairly low MSE for both training and test. It is possible that we have over fit the data on the training set because the test MSE is 3 times the training MSE. but nonetheless, they are quite good values.

```r
ridge.mod = glmnet(x.train,y.train,alpha=0,lambda=bestlam)

ridge.pred=predict(ridge.mod,s=bestlam ,newx=x.test)
mean((ridge.pred-y.test)^2)
```

```
## [1] 1.46
```

```r
ridge.pred.train=predict(ridge.mod,s=bestlam ,newx=x.train)
mean((ridge.pred.train-y.train)^2)
```

```
## [1] 0.5074
```

c. The optimal lambda value is 0.0366. The lasso coefficient estimates corresponding to this lambda value are shown below. There are two coefficients set to zero which are Population, and USYes. This means that in the lasso subset selection, Population and USYes do not have an affect on the response variable Sales because they are effectively excluded from the model.

```r
set.seed(123)

lambda.list.lasso = 2 * exp(seq(0, log(1e-4), length = 100))

cv.lasso.out = cv.glmnet(x.train, y.train, alpha=1, lambda = lambda.list.lasso)

bestlamlas = cv.lasso.out$lambda.min
bestlamlas
```

```
## [1] 0.03661
```

```r
lasso.mod = glmnet(x.train, y.train, alpha=1, lambda=bestlamlas)

lasso.pred = predict(lasso.mod, s=bestlam, newx = x.train)


lasso.coef = predict(lasso.mod, type='coefficients', s=bestlamlas)
lasso.coef
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)      6.037984
## CompPrice        0.107292
## Income           0.006439
## Advertising      0.075699
## Population       .
## Price           -0.094085
## ShelveLocGood    4.112830
## ShelveLocMedium  1.662509
## Age             -0.057810
## Education       -0.082580
## UrbanYes         0.562462
## USYes            .
```

d. The training MSE for the lasso model is 0.546 while the Test MSE is 1.464. Similar to the ridge model, it could be the case that we are overfitting our data

```
lasso.pred=predict(lasso.mod,s=bestlamlas ,newx=x.test)
mean((lasso.pred-y.test)^2)
```

```
## [1] 1.464
```

```
lasso.pred.train=predict(lasso.mod,s=bestlamlas ,newx=x.train)
mean((lasso.pred.train-y.train)^2)
```

```
## [1] 0.5466
```

e. In this application, the Ridge and Lasso methods for estimates are both very similar. They have basically the same test MSE and the training MSE for lasso is only 0.04 more than the ridge training MSE. For this data, It might be better to use the Lasso because it makes the model more sparse which in turn makes it simpler.

2.

```
drug <- read_csv('drug.csv',
col_names=c('ID','Age','Gender','Education','Country',
'Ethnicity','Nscore',
'Escore','Oscore','Ascore','Cscore',
'Impulsive','SS','Alcohol','Amphet','Amyl','Benzos',
'Caff','Cannabis', 'Choc','Coke','Crack','Ecstasy',
'Heroin','Ketamine','Legalh','LSD','Meth',
'Mushrooms','Nicotine','Semer','VSA'))
```

```
## Rows: 1885 Columns: 32
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (19): Alcohol, Amphet, Amyl, Benzos, Caff, Cannabis, Choc, Coke, Crack, ...
## dbl (13): ID, Age, Gender, Education, Country, Ethnicity, Nscore, Escore, Os...
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

a.

```
drug$recent_nicotine_use <- factor(ifelse(drug$Nicotine >='CL3','Yes', 'No'), levels = c('No', 'Yes'))
head(drug$recent_nicotine_use)
```

```
## [1] No  Yes No  No  No  Yes
## Levels: No Yes
```

b.

```r
sub_drug <- drug[, c(2:13, 33)]
colnames(sub_drug)
```

```
##  [1] "Age"                 "Gender"              "Education"
##  [4] "Country"             "Ethnicity"           "Nscore"
##  [7] "Escore"              "Oscore"              "Ascore"
## [10] "Cscore"              "Impulsive"           "SS"
## [13] "recent_nicotine_use"
```

c.

```r
set.seed(123)

trainsamp = sample(nrow(sub_drug), 1000)

drug.train = sub_drug[trainsamp, ]

drug.test = sub_drug[-trainsamp, ]

head(drug.train)
```

```
## # A tibble: 6 x 13
##       Age Gender Education Country Ethnicity Nscore Escore  Oscore  Ascore
##     <dbl>  <dbl>     <dbl>   <dbl>     <dbl>  <dbl>  <dbl>   <dbl>   <dbl>
## 1 -0.952  -0.482    -1.23    0.961    -0.317 -0.580 -1.09   -0.583  -2.08
## 2 -0.952   0.482     0.455   0.961    -0.317  0.313 -0.806   1.06    0.761
## 3  1.09    0.482     0.455   0.961    -0.317  0.826 -0.948 -0.452    0.761
## 4  1.82   -0.482    -1.23    0.961    -0.317 -1.87    0.476 -1.12   -0.0173
## 5 -0.0785  0.482     0.455   0.249     0.126 -0.921  1.29   -0.0193 -0.0173
## 6 -0.952  -0.482    -0.611  -0.285    -0.317  0.417 -0.440   1.06   -1.48
## # i 4 more variables: Cscore <dbl>, Impulsive <dbl>, SS <dbl>,
## #   recent_nicotine_use <fct>
```

d.

```r
mod1 <- glm(recent_nicotine_use ~., data=drug.train, family = 'binomial')
summary(mod1)
```

```
##
## Call:
## glm(formula = recent_nicotine_use ~ ., family = "binomial", data = drug.train)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.49460    0.15657    3.16   0.0016 **
## Age         -0.44225    0.09166   -4.82  1.4e-06 ***
## Gender      -0.36206    0.16300   -2.22   0.0263 *
## Education   -0.24006    0.08186   -2.93   0.0034 **
## Country     -0.49745    0.12347   -4.03  5.6e-05 ***
## Ethnicity   -0.22305    0.42973   -0.52   0.6037
## Nscore      -0.11492    0.09117   -1.26   0.2075
## Escore      -0.10898    0.09443   -1.15   0.2485
## Oscore       0.16778    0.08888    1.89   0.0591 .
## Ascore       0.00979    0.08071    0.12   0.9034
## Cscore      -0.22607    0.08642   -2.62   0.0089 **
## Impulsive    0.10033    0.10357    0.97   0.3327
## SS           0.31365    0.11201    2.80   0.0051 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1366.1  on 999  degrees of freedom
## Residual deviance: 1131.0  on 987  degrees of freedom
## AIC: 1157
##
## Number of Fisher Scoring iterations: 4
```

e.

```
tree.drug <- tree(recent_nicotine_use~., data = drug.train)
tree.drug
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 1000 1000 Yes ( 0.43 0.57 )
##    2) SS < -0.06794 455  600 No ( 0.62 0.38 )
##      4) Education < 0.197735 207  300 Yes ( 0.48 0.52 )
##        8) Country < 0.605025 74   90 Yes ( 0.30 0.70 ) *
##        9) Country > 0.605025 133  200 No ( 0.58 0.42 )
##         18) Cscore < -0.20942 51   70 Yes ( 0.35 0.65 ) *
##         19) Cscore > -0.20942 82  100 No ( 0.72 0.28 )
##           38) Age < 0.796185 45   60 No ( 0.56 0.44 ) *
##           39) Age > 0.796185 37   20 No ( 0.92 0.08 ) *
##      5) Education > 0.197735 248  300 No ( 0.73 0.27 ) *
##    3) SS > -0.06794 545  600 Yes ( 0.27 0.73 )
##      6) Country < 0.230255 293  300 Yes ( 0.17 0.83 ) *
##      7) Country > 0.230255 252  300 Yes ( 0.39 0.61 )
##       14) Cscore < 0.671595 197  200 Yes ( 0.32 0.68 ) *
##       15) Cscore > 0.671595 55   70 No ( 0.62 0.38 ) *
```

f. The best size for our tree using 5-fold cross validation is 5

```
set.seed(2)
cv = cv.tree(tree.drug, FUN=prune.misclass, K=5)
cv
```

```
## $size
## [1] 8 7 5 2 1
##
## $dev
## [1] 332 332 327 338 429
```

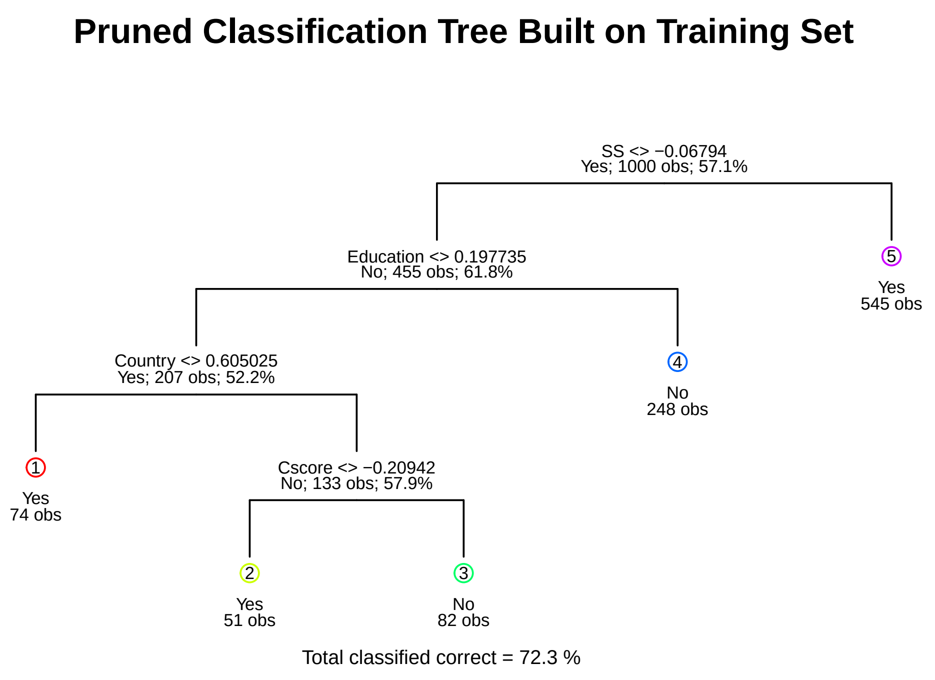
```
##
## $k
## [1]  -Inf   0.0   6.5  15.0 107.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
best.size = min(cv$size[cv$dev == min(cv$dev)])
best.size
```

```
## [1] 5
```

g.

```
pt.cv = prune.misclass(tree.drug, best=best.size)

draw.tree(pt.cv, nodeinfo=TRUE, cex = 0.55)
title("Pruned Classification Tree Built on Training Set")
```



h. The True Positive Rate is 0.80 and the False Positive Rate is 0.47

```
tree.pred = predict(pt.cv, drug.test, type = 'class')

recent.test <- drug.test$recent_nicotine_use

error = table(recent.test, tree.pred)
error
```

```
##            tree.pred
## recent.test  No Yes
##         No  208 188
##         Yes  96 393
```

```
TN = error[1]
FP = error[3]
FN = error[2]
TP = error[4]

#True Positive Rate = TP/(TP+FN)
paste('True Positive Rate: ', TP/(TP+FN))
```

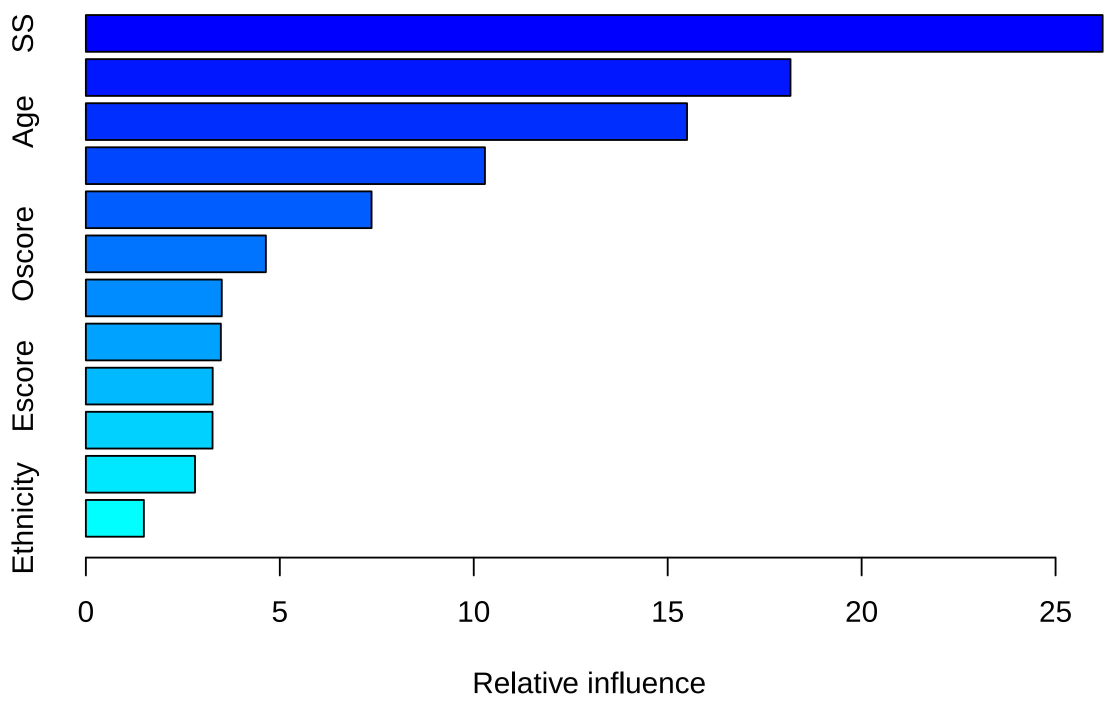
```
## [1] "True Positive Rate:  0.803680981595092"
```

```
#False Positive Rate = FP/(FP+TN)
paste('False Positive Rate: ', FP/(FP+TN))
```

```
## [1] "False Positive Rate:  0.474747474747475"
```

i. The variables that seem to be the most important are SS, Country, and Age.

```
set.seed(123)

boost.drug = gbm(ifelse(recent_nicotine_use == 'Yes', 1, 0)~., data=drug.train,
                 distribution = 'bernoulli', n.trees = 1000, shrinkage = 0.01)

summary(boost.drug)
```



```
##                 var rel.inf
## SS               SS  26.214
## Country     Country  18.166
## Age             Age  15.498
## Cscore       Cscore  10.288
## Education Education   7.365
## Oscore       Oscore   4.641
## Ascore       Ascore   3.504
## Gender       Gender   3.480
## Escore       Escore   3.270
## Impulsive Impulsive   3.266
## Nscore       Nscore   2.814
## Ethnicity Ethnicity   1.495
```

j. The OOB estimate of error rate is 30%, There were 3 variables randomly considered at each split in the trees. 500 trees were used to fit the data. Despite some differences, the order of important variables are fairly similar between the boosting and random forest models.

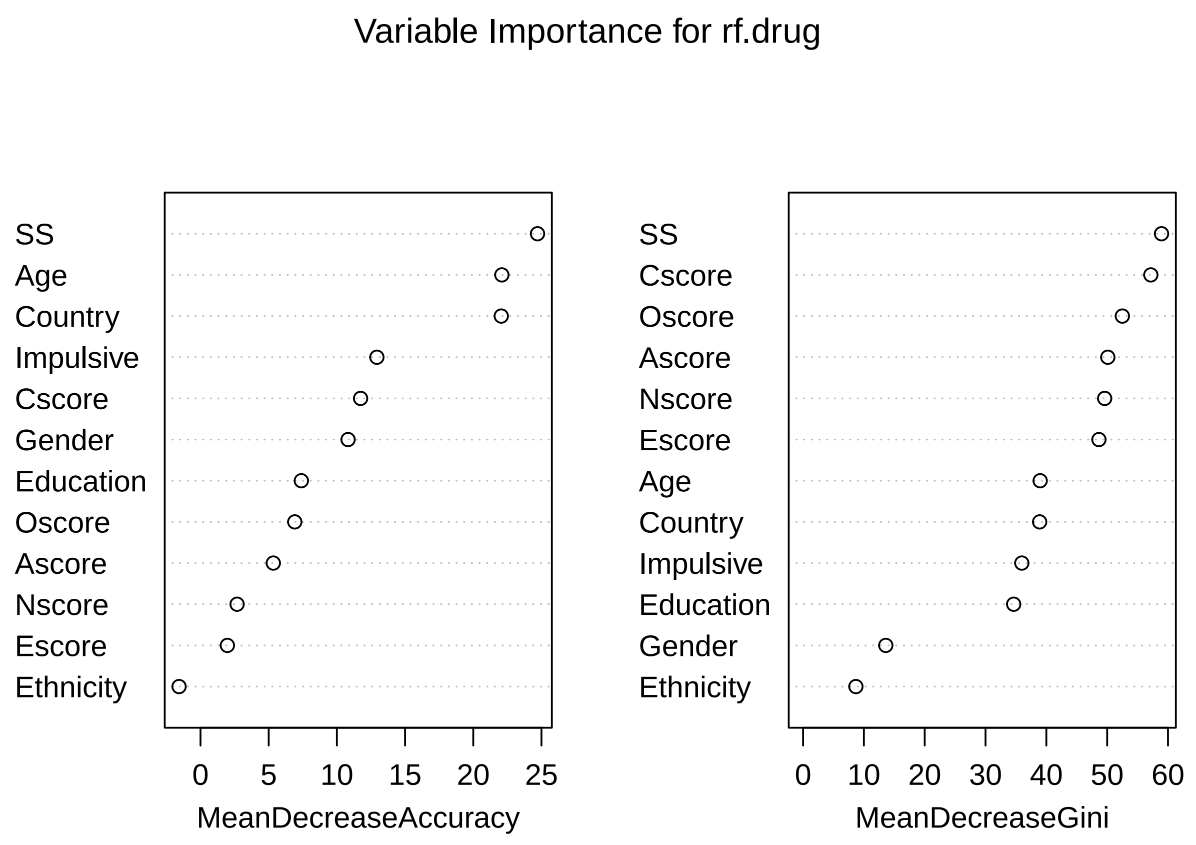
```
set.seed(123)

rf.drug <- randomForest(recent_nicotine_use~., data=drug.train,
                        importance = TRUE)

rf.drug
```

```
##
## Call:
##  randomForest(formula = recent_nicotine_use ~ ., data = drug.train,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 30%
## Confusion matrix:
##      No Yes class.error
## No  255 174      0.4056
## Yes 126 445      0.2207
```

```r
importance(rf.drug)
```

```
##                No     Yes MeanDecreaseAccuracy MeanDecreaseGini
## Age       15.4620 15.3910               22.097           38.984
## Gender    13.2275  2.2813               10.820           13.599
## Education  7.5800  2.9781                7.393           34.631
## Country   25.4077  5.0218               22.055           38.901
## Ethnicity -1.3869 -0.8032               -1.572            8.683
## Nscore    -0.7845  4.2984                2.684           49.590
## Escore     2.3166  0.4504                1.969           48.645
## Oscore     4.5145  5.0452                6.913           52.499
## Ascore     5.8128  1.5615                5.339           50.107
## Cscore    11.4552  4.5539               11.739           57.178
## Impulsive  1.5397 13.7528               12.934           35.968
## SS        16.6944 17.4728               24.712           58.939
```

```r
varImpPlot(rf.drug, sort=T, main='Variable Importance for rf.drug')
```



k.The fraction of people predicted to use nicotine recently who have in fact used nicotine recently is 0.5716.

```
prob.forest = predict(rf.drug, newdata=drug.test, type = 'prob')[, 'Yes']
pred.forest = ifelse(prob.forest >=0.2, 'Yes', 'No')

rf.matrix = table(drug.test$recent_nicotine_use, pred.forest)
rf.matrix
```

```
##      pred.forest
##        No Yes
##   No   43 353
##   Yes  18 471
```

```
response.boost <- predict(boost.drug, newdata = drug.test, type = 'response')
```

```
## Using 1000 trees...
```

```
pred.boost = ifelse(response.boost >= 0.2, 'Yes', 'No')

boost.matrix = table(drug.test$recent_nicotine_use, pred.boost)
boost.matrix
```

```
##      pred.boost
##        No Yes
##   No   24 372
##   Yes   8 481
```

```
#forest - Fraction of predicted nicotine use who actually use nicotine.
rf.matrix[4] / (rf.matrix[3]+rf.matrix[4])
```

```
## [1] 0.5716
```