Homework Assignment 3

Nicholas Axl Andrian

November 21, 2023

Loading needed libraries

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(ISLR)
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-8
```

```
library(tree)
library(maptree)
```

```
## Loading required package: cluster
## Loading required package: rpart
```

```
library(randomForest)
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(ROCR)
library(gbm)
```

```
## Loaded gbm 2.1.8.1
```

Predicting carseats sales using regularized regression methods

```
set.seed(123)
dat <- model.matrix(Sales~., Carseats)
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) finding the best lambda

```
set.seed(123)
lambda.list.ridge = 1000 * exp(seq(0, log(1e-5), length = 100))
cv_ridge_mod = cv.glmnet(x.train, y.train, alpha = 0, lambda = lambda.list.ridge, nfolds = 5)
bestlam = cv_ridge_mod$lambda.min
bestlam
```

```
## [1] 0.08111
```

refitting into a ridge regression model with the optimal best_lam

```
ridge_mod = glmnet(x.train, y.train, alpha=0, lambda = bestlam)
coef(ridge_mod)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                          s0
## (Intercept)      6.1039242
## (Intercept)      .
## CompPrice        0.1017457
## Income           0.0085167
## Advertising      0.0686428
## Population       0.0001441
## Price           -0.0903424
## ShelveLocGood    4.2185125
## ShelveLocMedium  1.7347806
## Age             -0.0569933
## Education       -0.0902160
## UrbanYes         0.6012834
## USYes            0.1212527
```

(b) finding train mse

```
ridge_train_pred <- predict(ridge_mod, newx = x.train, s = bestlam)
train_mse <- mean((y.train - ridge_train_pred)^2)
train_mse
```

```
## [1] 0.5567
```

finding test mse

```
ridge_test_pred <- predict(ridge_mod, newx = x.test, s = bestlam)
test_mse <- mean((y.test - ridge_test_pred)^2)
test_mse
```

```
## [1] 1.419
```

It is significantly larger than the training MSE, this could be due to overfiting or a bad split in the train test process

(C) fitting lasso and finding best lambda with 10-fold cv

```
set.seed(123)
lambda.list.lasso = 2 * exp(seq(0, log(1e-4), length = 100))
cv_lasso_mod = cv.glmnet(x.train, y.train, alpha = 1, lambda = lambda.list.lasso, nfolds = 10)
bestlam_lasso = cv_lasso_mod$lambda.min
bestlam_lasso
```

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

refitting lasso

```
lasso_mod = glmnet(x.train, y.train, alpha=1, lambda = bestlam_lasso)
coef(lasso_mod)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                         s0
## (Intercept)      6.037984
## (Intercept)      .
## 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            .
```

The USYes and population are set to 0, they are interpreted to be insignificant parameters in finding the predicted sales when using the current lambda value.

(d)Train MSE

```
lasso_train_pred <- predict(lasso_mod, newx = x.train, s = bestlam_lasso)
train_mse_lasso <- mean((y.train - lasso_train_pred)^2)
train_mse_lasso
```

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

Test MSE

```
lasso_test_pred <- predict(lasso_mod, newx = x.test, s = bestlam_lasso)
test_mse_lasso <- mean((y.test - lasso_test_pred)^2)
test_mse_lasso
```

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

The test MSE was larger, similar to the case in Ridge Regression

(e)I find that despite LASSO undergoing variable selection in removing the US and population parameters, they still end up pretty similar. With the current seed (123), Ridge still has the more acdcurate model for the test dataset due to lower MSE though

Analyzing Drug Use

```
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.
```

```
head(drug)
```

```
## # A tibble: 6 x 32
##      ID     Age Gender Education Country Ethnicity Nscore Escore  Oscore Ascore
##   <dbl>   <dbl>  <dbl>     <dbl>   <dbl>     <dbl>  <dbl>  <dbl>   <dbl>  <dbl>
## 1     1  0.498   0.482   -0.0592   0.961     0.126  0.313 -0.575 -0.583  -0.917
## 2     2 -0.0785 -0.482    1.98     0.961    -0.317 -0.678  1.94   1.44    0.761
## 3     3  0.498  -0.482   -0.0592   0.961    -0.317 -0.467  0.805 -0.847  -1.62
## 4     4 -0.952   0.482    1.16     0.961    -0.317 -0.149 -0.806 -0.0193  0.590
## 5     5  0.498   0.482    1.98     0.961    -0.317  0.735 -1.63  -0.452  -0.302
## 6     6  2.59    0.482   -1.23     0.249    -0.317 -0.678 -0.300 -1.56    2.04
## # i 22 more variables: Cscore <dbl>, Impulsive <dbl>, SS <dbl>, Alcohol <chr>,
## #   Amphet <chr>, Amyl <chr>, Benzos <chr>, Caff <chr>, Cannabis <chr>,
## #   Choc <chr>, Coke <chr>, Crack <chr>, Ecstasy <chr>, Heroin <chr>,
## #   Ketamine <chr>, Legalh <chr>, LSD <chr>, Meth <chr>, Mushrooms <chr>,
## #   Nicotine <chr>, Semer <chr>, VSA <chr>
```

(a) creating a new factor response

```
drug <- drug %>%
mutate(recent_nicotine_use = factor(ifelse(Nicotine >= "CL3", "Yes", "No"),levels = c("No", "Yes")));
```

(b)

```
sub_drug <- drug %>%
  select(Age:SS, recent_nicotine_use)

head(sub_drug)
```

```
## # A tibble: 6 x 13
##       Age Gender Education Country Ethnicity Nscore Escore  Oscore Ascore
##     <dbl>  <dbl>     <dbl>   <dbl>     <dbl>  <dbl>  <dbl>   <dbl>  <dbl>
## 1  0.498   0.482   -0.0592   0.961     0.126  0.313 -0.575 -0.583  -0.917
## 2 -0.0785 -0.482    1.98     0.961    -0.317 -0.678  1.94   1.44    0.761
```

```
## 3  0.498  -0.482   -0.0592   0.961    -0.317 -0.467  0.805 -0.847  -1.62
## 4 -0.952   0.482    1.16     0.961    -0.317 -0.149 -0.806 -0.0193  0.590
## 5  0.498   0.482    1.98     0.961    -0.317  0.735 -1.63  -0.452  -0.302
## 6  2.59    0.482   -1.23     0.249    -0.317 -0.678 -0.300 -1.56    2.04
## # i 4 more variables: Cscore <dbl>, Impulsive <dbl>, SS <dbl>,
## #   recent_nicotine_use <fct>
```

(c)

```
set.seed(123)
train = sample(nrow(sub_drug), 1000)
drug.train = sub_drug[train, ]
drug.test = sub_drug[-train, ]
```

(d)

```
drug_logr <- glm(recent_nicotine_use ~ . , data = drug.train, family = "binomial")
summary(drug_logr)
```

```
##
## 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)

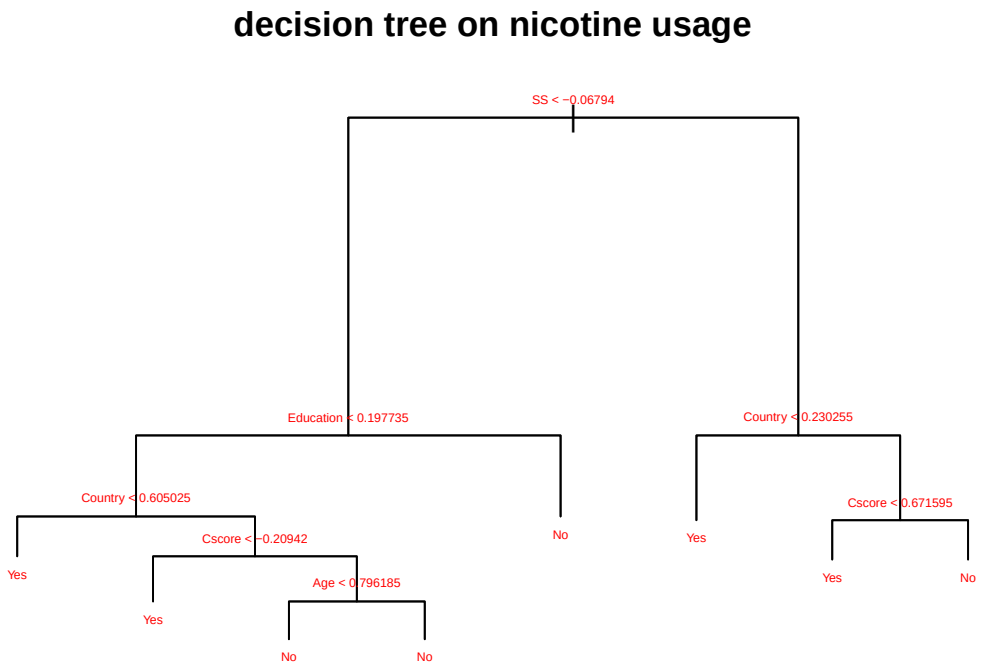
```
tree.drugs = tree(recent_nicotine_use  ~ . , data = drug.train)

summary(tree.drugs)
```

```
##
## Classification tree:
```

```
## tree(formula = recent_nicotine_use ~ ., data = drug.train)
## Variables actually used in tree construction:
## [1] "SS"        "Education" "Country"   "Cscore"    "Age"
## Number of terminal nodes:  8
## Residual mean deviance:  1.12 = 1120 / 992
## Misclassification error rate: 0.264 = 264 / 1000
```

```
plot(tree.drugs)
text(tree.drugs, pretty = 0, cex = .4, col = "red")
title("decision tree on nicotine usage", cex = 0.8)
```



(f)

```
set.seed(123)
drug_tree.cv = cv.tree(tree.drugs, FUN=prune.misclass, K=5)
drug_tree.cv$size
```

```
## [1] 8 7 5 2 1
```

```
drug_tree.cv$dev
```

```
## [1] 308 308 313 323 429
```

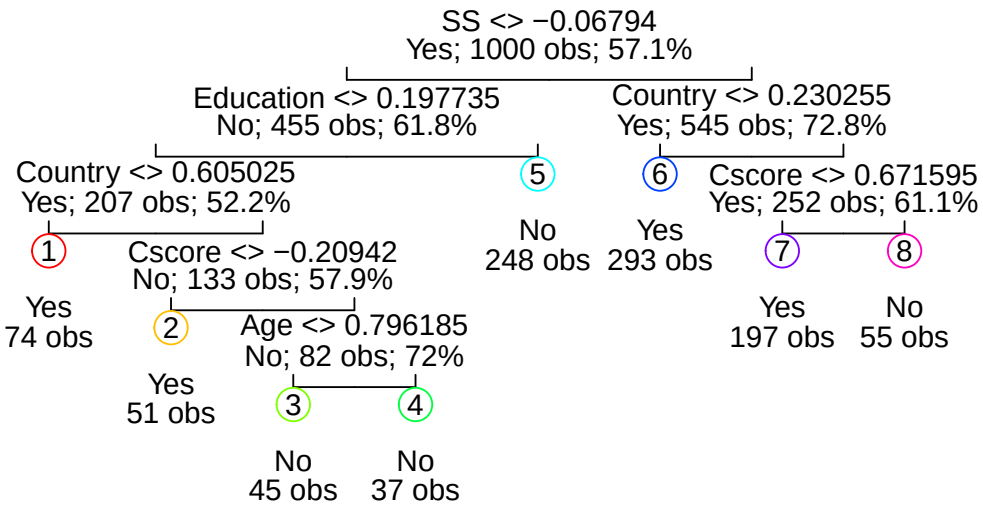
```
best.cv = min(drug_tree.cv$size[drug_tree.cv$dev == min(drug_tree.cv$dev)])
best.cv
```

```
## [1] 7
```

The best size is 7

(g)

```
pt.cv = prune.misclass(tree.drugs, best=best.cv)
draw.tree(tree.drugs, nodeinfo=TRUE)
```



SS was split first, followed by education/country

(h)

```
tree.pred = predict(tree.drugs, drug.test, type="class")
true.test = drug.test$recent_nicotine_use
error = table(tree.pred, true.test)
error
```

```
##          true.test
## tree.pred  No Yes
##       No  238 114
##       Yes 158 375
```

test error rate

```
1-sum(diag(error))/sum(error)
```

```
## [1] 0.3073
```

TPR and FPR

```
TP <- error[2, 2]
FP <- error[1, 2]
FN <- error[2, 1]
```

```
TN <- error[1, 1]
TPR <- TP / (TP + FN)
FPR <- FP / (FP + TN)
TPR
```

```
## [1] 0.7036
```

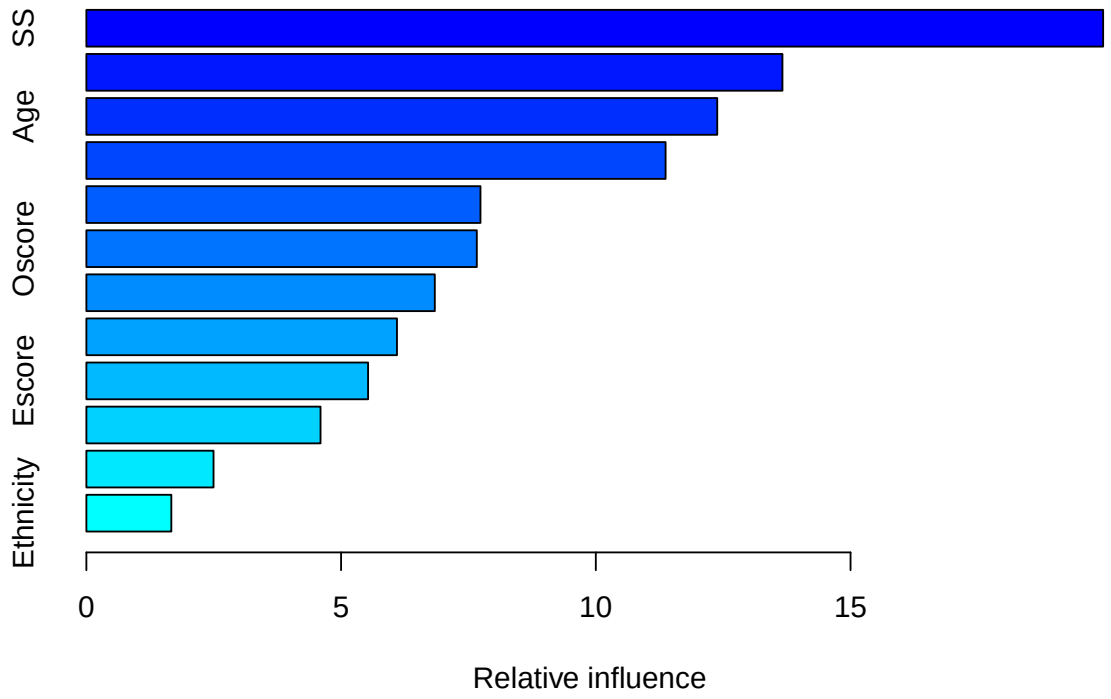
```
FPR
```

```
## [1] 0.3239
```

In the case of TPR, we calculate the percentages of true positive values out of the total between true positives and false negatives. (total number of positives) For FPR, it is the same concept but over the total of false positives and true negatives (total number of negatives)

(i)

```
set.seed(123)
boost.nicotine = gbm(ifelse(recent_nicotine_use=="Yes",1,0)~., data=drug.train,
distribution="bernoulli", n.trees=1000, interaction.depth=2, shrinkage = 0.01)
summary(boost.nicotine)
```




```
##                 var rel.inf
## SS               SS  19.958
## Country     Country  13.663
## Age             Age  12.383
## Cscore       Cscore  11.369
## Education Education   7.736
```

```
## Oscore       Oscore   7.664
## Ascore       Ascore   6.839
## Nscore       Nscore   6.097
## Escore       Escore   5.530
## Impulsive Impulsive   4.596
## Gender       Gender   2.496
## Ethnicity Ethnicity   1.668
```

SS country and age seem to be the most important (in that order)

(j)

```
rf.drugs = randomForest(recent_nicotine_use ~ ., data=drug.train, importance=TRUE)
rf.drugs
```

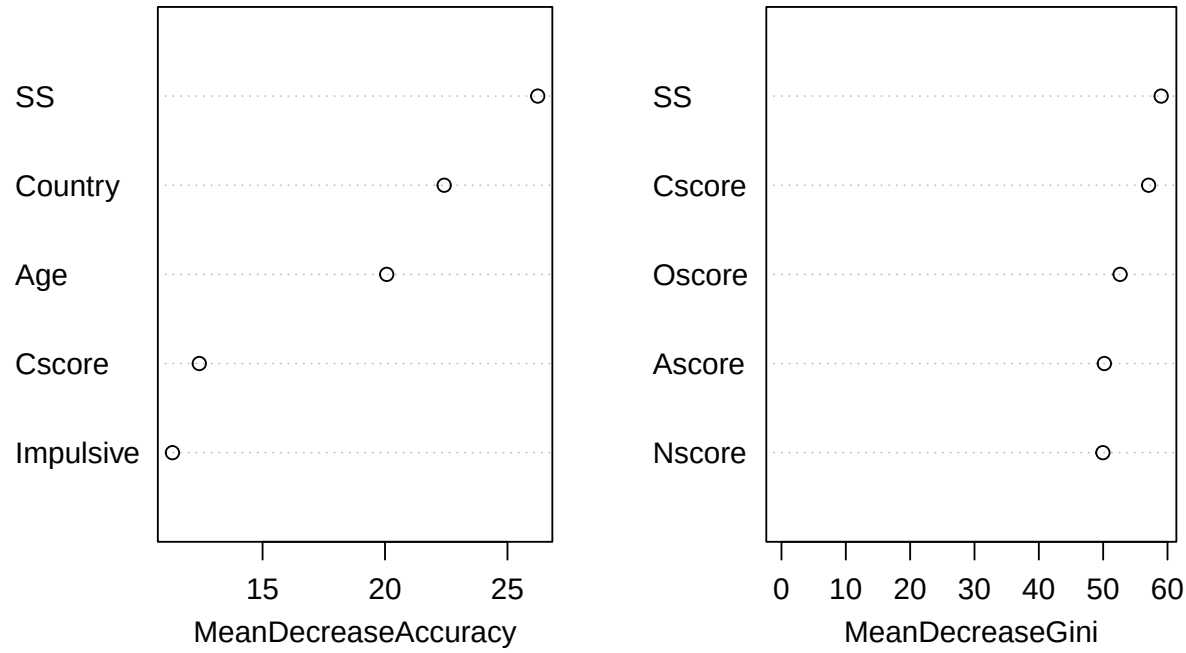
```
##
## Call:
##  randomForest(formula = recent_nicotine_use ~ ., data = drug.train,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 29.7%
## Confusion matrix:
##      No Yes class.error
## No  256 173      0.4033
## Yes 124 447      0.2172
```

oob error is 29.7% 3 variables considered at each split in the trees 500 trees used

```
importance(rf.drugs)
```

```
##                 No     Yes MeanDecreaseAccuracy MeanDecreaseGini
## Age       14.99877 12.8167              20.0698           38.706
## Gender    11.27140  2.5607               9.9950           14.171
## Education  9.40704  4.4487              10.1441           34.655
## Country   25.03406  5.8043              22.4165           37.248
## Ethnicity -1.14151  0.2183              -0.6107            8.611
## Nscore    -0.07185  2.9401               2.0607           49.958
## Escore     2.75827 -0.4205               1.5990           48.878
## Oscore     2.98058  6.7652               7.2115           52.633
## Ascore     5.73845  1.8643               5.5300           50.186
## Cscore    11.04314  6.6751              12.4159           57.063
## Impulsive  3.78407 10.4688              11.3197           35.955
## SS        17.03198 17.0259              26.2365           59.018
```

```
varImpPlot(rf.drugs, sort=T,
main="Variable Importance for rf.drugs", n.var=5)
```




```
4
```

```
## [1] 4
```

Yeah, SS country and age seem to be the most important too. Similar to boosting

(k) Boosting matrix

```
yhat.boost = predict(boost.nicotine, newdata = drug.test,
n.trees=1000, type = "response")
# then convert the probability to labels
yhat.boost = ifelse(yhat.boost > 0.2, 'Yes', 'No')
table(drug.test$recent_nicotine_use, yhat.boost)
```

```
##       yhat.boost
##         No Yes
##   No    54 342
##   Yes   18 471
```

RF matrix

```
yhat.rf = predict(rf.drugs, newdata = drug.test, type='Prob')
yhat.rf2 <- ifelse(yhat.rf[, 2] > 0.2, "Yes", "No")
table(drug.test$recent_nicotine_use, yhat.rf2)
```

```
##       yhat.rf2
##         No Yes
##   No    44 352
##   Yes   18 471
```

823/489 = 1/6 1.6x more people were predicted using the 20% metric