PSTAT 131 Final Project

Patrick Omens and Drew Cockerham

2023-11-10

#Data Description

The data on drug consumption records information about 1885 respondents, who are primarily from English-speaking countries.

For each respondent there are 12 attributes recorded in the original dataset: age, gender, level of education, country of residence and ethnicity, as well as personality information (measures of neuroticism, extraversion, openness to experience, agreeableness, conscientiousness, impulsivity, and sensation seeking). The personality variables are all numeric while the others listed are categorical. Furthermore, information is recorded for each participant on their use of 18 different drugs. These variables are categorical and have 6 levels: CL0 - never used, CL1 - used over a decade ago, CL2 - used in last decade,CL3 - used in last year, CL4 - used in last month, CL5 - used in last week, CL6 - used in last day. One of these variables is the recorded response of each participant to a question asking if they have taken a fictitious drug 'Semeron'. Participants who answered that they had taken this drug should be dropped from the data. The data contains no missing values.

#variables in the original dataset: age, gender, education, country, ethnicity, nscore, escore, oscore, ascore, cscore, impulsiveness, ss (sensation seeking), alcohol, amphet, amyl (amyl nitrate), benzos caff, cannabis, coke, crack, ecstasy, heroin, ketamine, legalh, lsd, meth, mushrooms, nicotine, VSA (volatile substance abuse)

For this project, we are only concerned about whether or not a respondent has taken any 'hard drugs' at least once in the past month. This is the list of drugs we are looking at: amphetamine, amyl nitrite, benzodiazepine, cocaine, crack cocaine, ecstasy, heroin, ketamine, legal highs, methamphetamine, and volatile substances. If a participant has used any of these drugs at least once in the past month, then we classify him or her as 'Yes' under the 'User' column, which in this case we use as our response variable. Otherwise, the participant is classified as 'No' under the 'User' column. If a participant is classified with the value 'Yes', then we say he or she is at risk of being a drug user.

For our purposes we are only interested in twelve variables: User, Age, Gender, Education, Country, Ethnicity, Nscore, Escore, Oscore, Ascore, Cscore, and Impulsiveness. We will later delve into these twelve variables in detail.

Next, we will import all of the needed libraries, import the data,
and clean the data.

Each variable value, regardless if it was continuous or categorical, was
originally represented by a double with six digits. We convert many of the
variable values in the original dataset into their real-life meaning. We do not,
however, change the values of the personality scores in the original dataset
because they are supposed to be continuous anyway, and they are already
conveniently standardized for us.

Additional variable information can be found at
https://archive.ics.uci.edu/dataset/373/drug+consumption+quantified

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- 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(ROCR)
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.3.2
```

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

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.3.2
```

```
## 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
```

```r
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 4.3.2
```

```
## Loaded gbm 2.1.8.1
```

```r
drugs <- read.table('drug_consumption.data', sep = ',')

drugs <- drugs[,-1] # We get rid of the ID number of each participant

colnames(drugs)[1] = 'Age'
colnames(drugs)[2] = 'Gender'
colnames(drugs)[3] = 'Education'
colnames(drugs)[4] = 'Country'
colnames(drugs)[5] = 'Ethnicity'
colnames(drugs)[6] = 'Nscore'
colnames(drugs)[7] = 'Escore'
colnames(drugs)[8] = 'Oscore'
colnames(drugs)[9] = 'Ascore'
colnames(drugs)[10] = 'Cscore'
colnames(drugs)[11] = 'Impulsiveness'
colnames(drugs)[12] = 'SS'
colnames(drugs)[13] = 'Alcohol'
colnames(drugs)[14] = 'Amphet'
colnames(drugs)[15] = 'Amyl'
colnames(drugs)[16] = 'Benzos'
colnames(drugs)[17] = 'Caff'
colnames(drugs)[18] = 'Cannabis'
colnames(drugs)[19] = 'Choc'
colnames(drugs)[20] = 'Coke'
colnames(drugs)[21] = 'Crack'
colnames(drugs)[22] = 'Ecstasy'
colnames(drugs)[23] = 'Heroin'
colnames(drugs)[24] = 'Ketamine'
colnames(drugs)[25] = 'Legalh'
colnames(drugs)[26] = 'LSD'
colnames(drugs)[27] = 'Meth'
colnames(drugs)[28] = 'Mushrooms'
colnames(drugs)[29] = 'Nicotine'
colnames(drugs)[30] = 'Semer'
colnames(drugs)[31] = 'VSA'

drugs$Education_og <- drugs$Education
```

```r
# Get rid of rows where participants claim to have used 'Semer'. 'Semer' is a
# fictitious drug which was introduced to identify over-claimers.

semerIndices <- which(drugs[,30] != 'CL0')

drugs <- drugs[-semerIndices, ]

# We now get rid of the column 'Semer'
drugs <- drugs[,-30]

# We also get rid of the column 'Choc', as we are uninterested in participants'
# use of chocolate as it is irrelevant to the questions we want answered.
drugs <- drugs[,-19]

nrow(drugs)
```

```
## [1] 1877
```

```r
# converting variable values of 'Age' column
for (i in 1:nrow(drugs)) {
  if (drugs$Age[i] == -0.95197) {
    drugs$Age[i] = '18-24'
  } else if (drugs$Age[i] == -0.07854) {
    drugs$Age[i] = '25-34'
  } else if (drugs$Age[i] == 0.49788) {
    drugs$Age[i] = '35-44'
  } else if (drugs$Age[i] == 1.09449) {
    drugs$Age[i] = '45-54'
  } else if (drugs$Age[i] == 1.82213) {
    drugs$Age[i] = '55-64'
  } else if (drugs$Age[i] == 2.59171) {
    drugs$Age[i] = '65+'
  }
}

# converting variable values of 'Gender' column
for (i in 1:nrow(drugs)) {
  if (drugs$Gender[i] == 0.48246) {
    drugs$Gender[i] = 'Female'
  } else if (drugs$Gender[i] == -0.48246) {
    drugs$Gender[i] = 'Male'
  }
}


# converting variable values of 'Education' column
for (i in 1:nrow(drugs)) {
  if (drugs$Education[i] == -2.43591 | drugs$Education[i] == -1.73790 |
      drugs$Education[i] == -1.43719 | drugs$Education[i] == 1.22751 |
      drugs$Education[i] == -1.22751) {
    drugs$Education[i] = 'Low'
  } else {
    drugs$Education[i] = 'High'
```

```
  }
}

# converting variable values of 'Country' column
for (i in 1:nrow(drugs)) {
  if (drugs$Country[i] == -0.09765) {
    drugs$Country[i] = 'Australia'
  } else if (drugs$Country[i] == 0.24923) {
    drugs$Country[i] = 'Canada'
  } else if (drugs$Country[i] == -0.46841 | drugs$Country[i] == -0.28519 |
             drugs$Country[i] == 0.21128) {
    drugs$Country[i] = 'Other'
  } else if (drugs$Country[i] == 0.96082) {
    drugs$Country[i] = 'UK'
  } else if (drugs$Country[i] == -0.57009) {
    drugs$Country[i] = 'USA'
  }
}

# converting variable values of 'Ethnicity' column
for (i in 1:nrow(drugs)) {
  if (drugs$Ethnicity[i] == -0.50212) {
    drugs$Ethnicity[i] = 'Asian'
  } else if (drugs$Ethnicity[i] == -1.10702) {
    drugs$Ethnicity[i] = 'Black'
  } else if (drugs$Ethnicity[i] == -0.31685) {
    drugs$Ethnicity[i] = 'White'
  } else {
    drugs$Ethnicity[i] = 'Other'
  }
}

# binary classification

# Hard drugs: Amphet, Amyl, Benzos, Coke, Crack, Ecstasy, Heroin, Ketamine,
# Legalh, Meth, VSA
# these correspond to columns 14, 15, 16, 19, 20, 21, 22, 23, 24, 26, 29

hardDrugs  <- drugs[,c(14, 15, 16, 19, 20, 21, 22, 23, 24, 26, 29)]

usersIndices <- apply(hardDrugs, 1, function(r) any(r %in%
                                                    c('CL4', 'CL5', 'CL6')))




# creating response variable 'User'
for (i in 1:length(usersIndices)) {
  if (usersIndices[i] == TRUE) {
    drugs$User[i] = 'Yes'
  } else {
    drugs$User[i] = 'No'
```

```
  }
}
```

```
dim(drugs)
```

```
## [1] 1877   31
```

```
names(drugs)[-c(6:12)]
```

```
##  [1] "Age"          "Gender"       "Education"    "Country"      "Ethnicity"
##  [6] "Alcohol"      "Amphet"       "Amyl"         "Benzos"       "Caff"
## [11] "Cannabis"     "Coke"         "Crack"        "Ecstasy"      "Heroin"
## [16] "Ketamine"     "Legalh"       "LSD"          "Meth"         "Mushrooms"
## [21] "Nicotine"     "VSA"          "Education_og" "User"
```

```
# converting all categorical variable values into factors
drugs[,-c(6:12)] <- lapply(drugs[,-c(6:12)], factor)
drugs$Country = factor(drugs$Country)
drugs$Ethnicity = factor(drugs$Ethnicity)

# relevel() re-orders the levels of a factor so that the level specified is
# first and the others are moved down
drugs$Country = relevel(drugs$Country, ref = 'USA')
drugs$Ethnicity = relevel(drugs$Ethnicity, 'White')

View(drugs)
```

#Explaining the variables

For this project, we are only concerned with the response variable 'User' and the following 11 predictors.

'User' - binary classification - has the person used a hard drug at least once in the past month? 'Yes' or 'No'

'Age' is the age of the participant.
Age25-34: participant is of the age range 25-34,
Age35-44 means participant is of the age range 35-34, etc. The '18-24' age range is the baseline level.

'Gender' is the gender of the participant. 'Female' is the baseline level.

'Education' is the education of the participant.
'Low' Education refers to anyone who left school at 18 years or younger.
'High' education (baseline value) refers to anyone who has attended some college or has any kind of diploma or degree.

Country is the country of current residence of the participant.

'USA' is the baseline level.

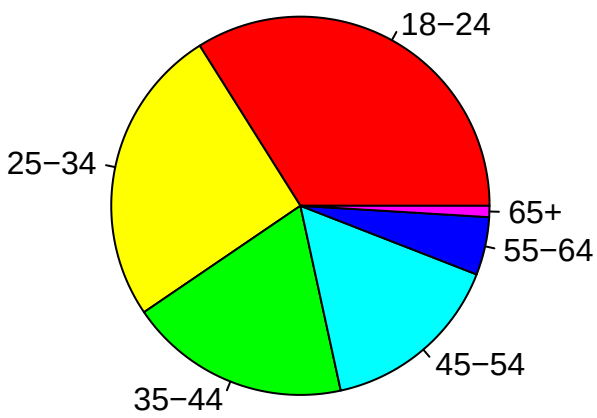
Ethnicity is the ethnicity of the participant. 'White' is the baseline level.

Nscore represents a participant's standardized neuroticism score on the
NEO-FFI-R assessment.
Escore represents a participant's standardized extraversion score on the
NEO-FFI-R assessment.
Ascore represents a participant's standardized agreeableness score on the
NEO-FFI-R assessment.
Oscore represents a participant's standardized openness score on the
NEO-FFI-R assessment.
Cscore represents a participant's standardized conscientiousness score on
the NEO-FFI-R assessment.
Impulsiveness represents a participant's standardized impulsiveness
score measured by BIS-11.

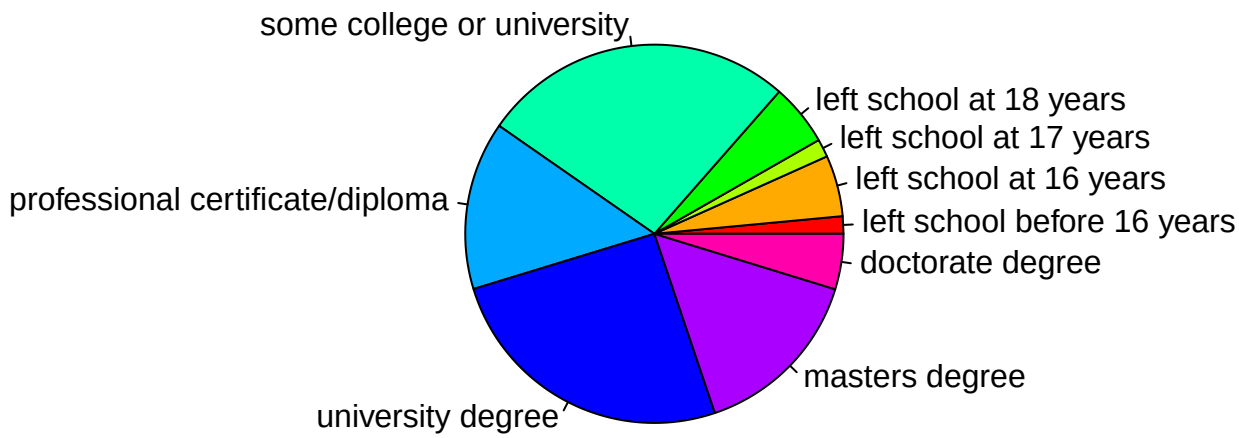
```
# Exploratory analysis


summary(drugs[,c(2,4,5,31)])
```

```
##     Gender          Country     Ethnicity     User
##  Female:937   USA       : 551   White:1715   No :1173
##  Male  :940   Australia:  52   Asian:  25   Yes: 704
##               Canada    :  87   Black:  33
##               Other     : 143   Other: 104
##               UK        :1044
```

```
# The respondents of this survey are close to evenly split among gender.
# Most of them are not drug users, most come from the UK, and they are
# overwhelmingly white.

# Pie chart of age of respondents
pie(table(drugs$Age), names(table(drugs$Age)), main = "Pie Chart", col =
      rainbow(length(names(table(drugs$Age)))))
```




```
# The respondents in this survey skew young. Most of them are younger
# than the age of 35

labels2 <- c('left school before 16 years', 'left school at 16 years',
'left school at 17 years', 'left school at 18 years',
'some college or university','professional certificate/diploma',
'university degree', 'masters degree','doctorate degree')

# Pie chart of education of respondents
pie(table(drugs$Education_og), labels2, main = "Pie Chart",
    col = rainbow(length(labels2)))
```




```
# The respondents in this survey are relatively educated. The vast majority of
# them would have 'High' education by our metrics.
```

```
# Training/test split

set.seed(1)
print(1885 * 0.25) # we will use a training/test split of around 75/25.
```

```
## [1] 471.25
```

```
test.indices = sample(1:nrow(drugs), 471)
drugs.train = drugs[-test.indices,]
drugs.test = drugs[test.indices,]
```

```
# Fitting the model
glm.fit <- glm(User ~ Age + Gender + Education + Country + Ethnicity + Nscore +
                 Escore + Oscore + Ascore + Cscore + Impulsiveness,
               data = drugs.train, family = binomial)

summary(glm.fit)
```

```
##
## Call:
## glm(formula = User ~ Age + Gender + Education + Country + Ethnicity +
##     Nscore + Escore + Oscore + Ascore + Cscore + Impulsiveness,
##     family = binomial, data = drugs.train)
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)       0.16617    0.16566   1.003 0.315823
## Age25-34          0.02448    0.17201   0.142 0.886852
## Age35-44         -0.74817    0.20602  -3.632 0.000282 ***
## Age45-54         -0.79327    0.22442  -3.535 0.000408 ***
## Age55-64         -1.11584    0.37775  -2.954 0.003138 **
## Age65+           -0.59696    0.76106  -0.784 0.432815
## GenderMale        0.63560    0.14351   4.429 9.47e-06 ***
## EducationLow      0.57755    0.20510   2.816 0.004863 **
## CountryAustralia  0.95424    0.42058   2.269 0.023275 *
## CountryCanada    -0.36002    0.30617  -1.176 0.239642
## CountryOther     -0.70436    0.24026  -2.932 0.003371 **
## CountryUK        -1.63244    0.16382  -9.965  < 2e-16 ***
## EthnicityAsian   -1.34762    0.81878  -1.646 0.099787 .
## EthnicityBlack   -0.42469    0.59311  -0.716 0.473971
## EthnicityOther   -0.03182    0.28872  -0.110 0.912248
## Nscore            0.19984    0.08098   2.468 0.013592 *
## Escore            0.07480    0.07874   0.950 0.342141
## Oscore            0.22263    0.07791   2.858 0.004269 **
## Ascore           -0.15686    0.07104  -2.208 0.027238 *
## Cscore           -0.17408    0.08004  -2.175 0.029631 *
## Impulsiveness     0.31105    0.08122   3.830 0.000128 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1862.1  on 1405  degrees of freedom
## Residual deviance: 1371.2  on 1385  degrees of freedom
## AIC: 1413.2
##
## Number of Fisher Scoring iterations: 5
```

#Which variables are significant?

With a p-value of 0.05, the variables that are significant are:
Age35-44, Age45-54, Age55-64, GenderMale, EducationLow, CountryAustralia,
CountryOther, CountryUK, Nscore, Oscore, Ascore, Cscore, and Implulsiveness.
It is notable that many of the coefficients are not significant with a p-value
of 0.05. We'll keep an eye on this, and later we'll perform lasso regression
to see which coefficients it will set to zero.

#Interpreting coefficients

The coefficient estimates of Age35-44, Age45-54, and Age55-64 are all
significant and have negative values. We interpret this to mean if a

participant's age is from 35-64, they are less likely to be a drug user than someone aged 18-24 (which is the baseline level).

Males are far more likely to become drug users than females. This is because the coefficient estimate for 'GenderMale' is positive, and the baseline level is represented by females.

Individuals who have not gone to college or who have dropped out of high school are likelier to be drug users. This is because the coefficient estimate for EducationLow is positive.

Using the same logic, Australia is the only other country in this dataset that seems to have a higher percentage of drug users than America.

Neuroticism, openness, and impulsiveness are positively correlated with drug use. Agreeableness and conscientiousness seem to be negatively correlated with drug use. Impulsiveness seems to be the most significant of the personality traits. This makes inuitive sense - someone who is more impulsive is less likely to think twice about engaging in risky behavior such as drug use.

```
# Finding training error and test error with our training/test data split

predicted_train <- ifelse(predict(glm.fit, drugs.train,
                                  type = 'response') > 0.5, 'Yes', 'No')

predicted_test <- ifelse(predict(glm.fit, drugs.test,
                                 type = 'response') > 0.5, 'Yes', 'No')
true_train <- drugs.train[,'User']
true_test <- drugs.test[,'User']

calc_error_rate <- function(predicted.value, true.value){
return(mean(true.value!=predicted.value))
}

train.error <- calc_error_rate(predicted_train,true_train)
train.error # training error rate: 0.235
```

```
## [1] 0.2354196
```

```
test.error <- calc_error_rate(predicted_test,true_test)
test.error # test error rate: 0.212
```

```
## [1] 0.2123142
```

```
# Let's use the method of determining the best threshold value!
# We want the false negative rate and the false positive rate to be as small
# as possible simultaneously.

prob.training = predict(glm.fit, drugs.train, type = 'response')


pred = prediction(prob.training, drugs.train$User)
```

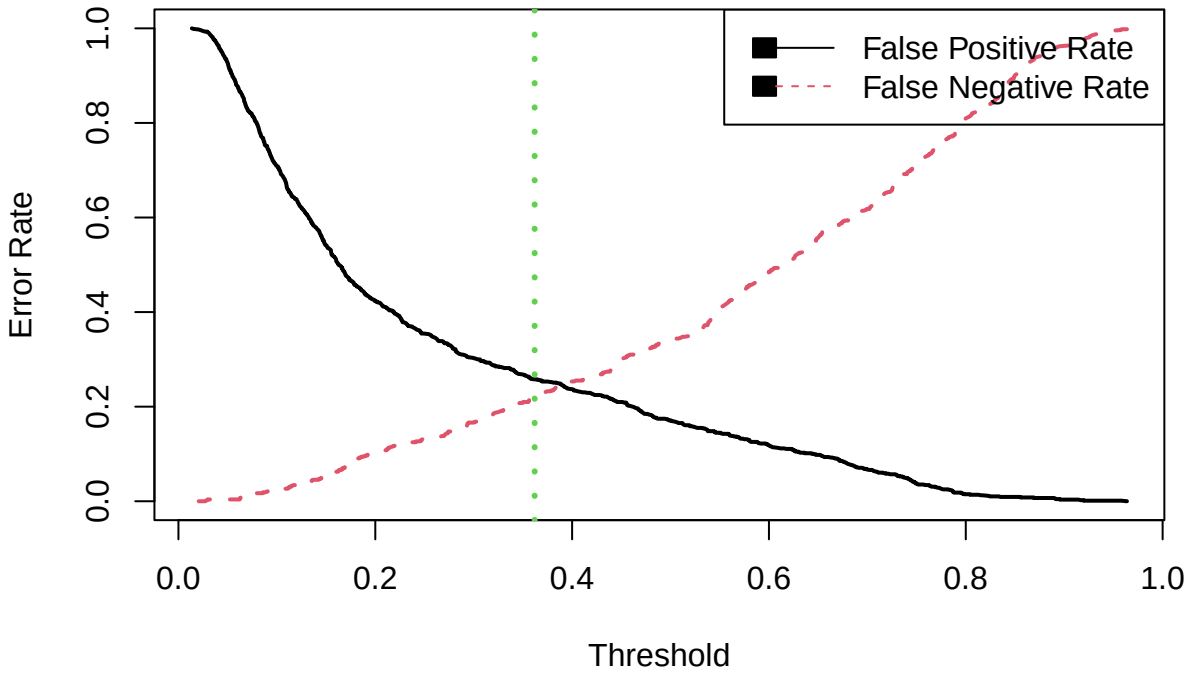
```
perf = performance(pred, measure = 'tpr', x.measure = 'fpr')

fpr = performance(pred, 'fpr')@y.values[[1]]
cutoff = performance(pred, 'fpr')@x.values[[1]]
fnr = performance(pred, 'fnr')@y.values[[1]]

# Calculating the euclidean distance between (FPR,FNR) and (0,0)

rate = as.data.frame(cbind(Cutoff = cutoff, FPR = fpr, FNR = fnr))
rate$distance = sqrt((rate[,2])^2 + (rate[,3])^2)
index = which.min(rate$distance)
index
```

```
## [1] 642
```

```
best = rate$Cutoff[index]
best
```

```
## [1] 0.3619286
```

```
# our best cutoff value is 0.3619

# probability of being a user being less than 0.3619 should be predicted as no,
# and higher than 0.3619 should be predicted as yes.

matplot(cutoff, cbind(fpr, fnr), type = 'l', lwd = 2, xlab = 'Threshold',
        ylab = 'Error Rate')
legend('topright',0.2, 1, legend = c('False Positive Rate',
                                     'False Negative Rate'),
       col = c(1,2), lty = c(1,2))
abline(v = best, col = 3, lty = 3, lwd = 3)
```



This is a more conservative approach - we end up trading a lower false negative rate for a higher false positive rate. This makes more sense for the question we're trying to answer, which is whether or not someone is at risk of being a drug user. It is far more preferable that we erroneously predict someone is at risk of becoming a drug user when they're not (false positive) than us predicting someone will not become a drug user when they are indeed at risk of becoming one (false negative). It's less harmful to classify someone who isn't at risk to be at risk (and erroneously notifying them of such nonexistent risk) since it would merely result in them exercising a bit more caution than necessary. However, classifying someone who is at risk to be not at risk (and erroneously notifying them of such lack of risk) is far more problematic. This would enable someone to continue to uphold a lifestyle which may lead them to drug abuse, and giving them the false impression that they don't contain the traits that put them at risk.

```
# We will now use the new 'best' threshold value and determine if it helps
# lower the training error rate or the test error rate.

predicted_train2 <- ifelse(predict(glm.fit, drugs.train,
                                   type = 'response') > best, 'Yes', 'No')
predicted_test2 <- ifelse(predict(glm.fit, drugs.test,
                                  type = 'response') > best, 'Yes', 'No')
```

```
train.error2 <- calc_error_rate(predicted_train2, true_train)
train.error2 # training error rate: 0.243
```

```
## [1] 0.242532
```

```
test.error2 <- calc_error_rate(predicted_test2, true_test)
test.error2 # training error rate: 0.212
```

```
## [1] 0.2123142
```

```
# Our training error rate is now a bit larger, but our test error rate is
# exactly the same. This is most likely worth it, as we've determined that a
# slightly more conservative approach makes more sense for this data.

# How does the confusion matrix look like?
confMatrixNew = table(pred = predicted_train2, true = true_train)
confMatrixNew
```

```
##      true
## pred   No Yes
##   No  651 115
##   Yes 226 414
```

```
# We will now try lasso regression, and see if we see any improvements on the
# test error rate.

# Lasso regression performs variable selection. This may help us in this
# scenario because we've used many variables in our logistic model,
# many of which are not significant.

x = model.matrix(User ~ Age + Gender + Education + Country + Ethnicity +
                   Nscore + Escore + Oscore + Ascore + Cscore +
                   Impulsiveness, drugs)
y = drugs$User
x.train = x[-test.indices, ]
y.train = y[-test.indices]
x.test = x[test.indices, ]
y.test = y[test.indices]

grid = 10^seq(10, -2, length = 100)
lasso.mod <- glmnet(x.train, y.train, alpha = 1, lambda = grid,
                    family = binomial)
# cross-validation to choose the best tuning parameter

cv.out.lasso = cv.glmnet(x.train, y.train, alpha = 1, family = binomial)
bestLambda = cv.out.lasso$lambda.min
lasso.pred = ifelse(predict(lasso.mod, s = bestLambda,
                            newx = x[test.indices, ]) > 0.5, 'Yes', 'No')


calc_error_rate(lasso.pred, y.test) # Test error rate is 0.242.
```

```
## [1] 0.2420382
```

```
# How about if we use the cutoff value we previously found (0.3619)
# to see if our error rate changes?

BestLasso.pred = ifelse(predict(lasso.mod, s = bestLambda, newx =
                                  x[test.indices, ]) > best, 'Yes', 'No')
calc_error_rate(BestLasso.pred, y.test) # Test error rate is 0.227.
```

```
## [1] 0.2271762
```

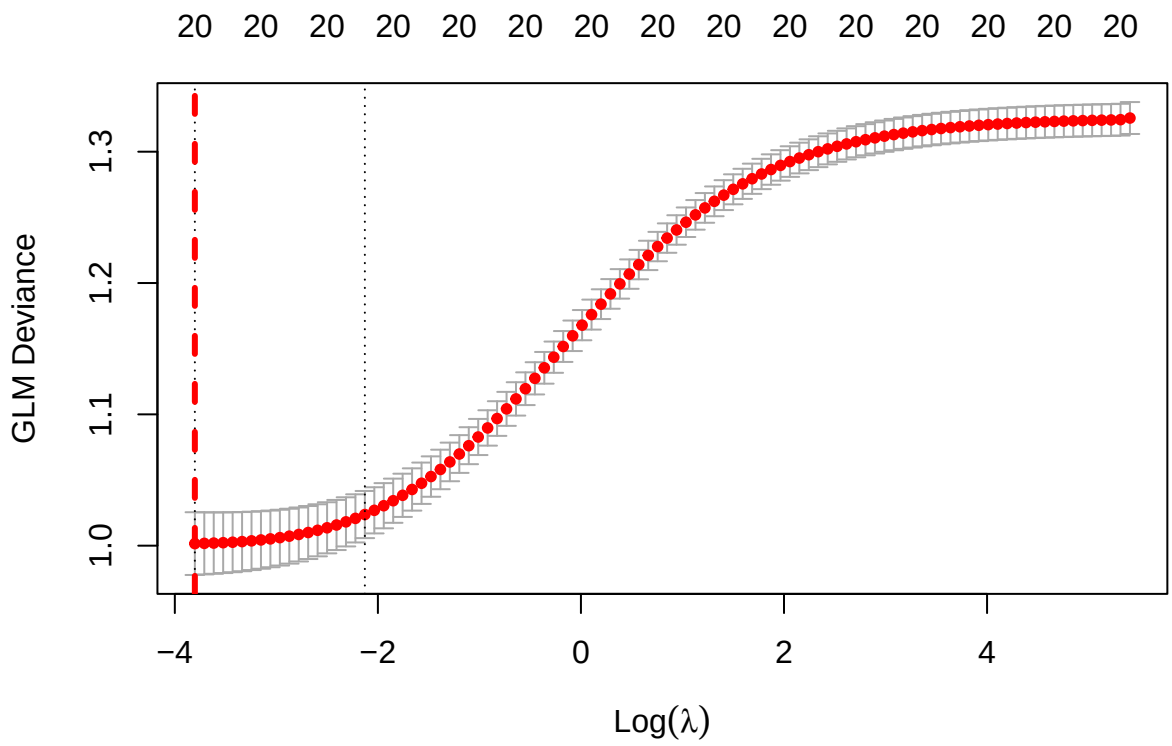
```
# Using a better cutoff value only gave us a slight improvement. However, this
# error rate is still higher than the test error rate we obtained using normal
# logistic regression with the best cutoff value based on the TPR and FPR.

# How sparse is the model? How many of the coefficients are equal to zero?

out = glmnet(x, y, alpha = 1,lambda = grid, family = binomial)
lasso.coef = predict(out, type = "coefficients", s = bestLambda)
lasso.coef
```

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                            s1
## (Intercept)       0.11873928
## (Intercept)       .
## Age25-34          .
## Age35-44         -0.50763558
## Age45-54         -0.56752397
## Age55-64         -0.88330218
## Age65+            .
## GenderMale        0.52529161
## EducationLow      0.36724892
## CountryAustralia  0.14410935
## CountryCanada    -0.08522108
## CountryOther     -0.39646774
## CountryUK        -1.53090897
## EthnicityAsian   -0.41478509
## EthnicityBlack   -0.11840758
## EthnicityOther    .
## Nscore            0.12647546
## Escore            .
## Oscore            0.21119868
## Ascore           -0.11290166
## Cscore           -0.18101628
## Impulsiveness     0.29865192
```

```
# Four of the twenty coefficients are exactly zero. The model is not too sparse.
# The coefficients that converged to 0 are Age25-34, Age65+, EthnicityOther,
# and Escore.
```

```
# Would there be an improvement with ridge regression?
# Let's use cross-validation to choose the best tuning parameter
```

```
ridge.mod = glmnet(x.train, y.train, alpha = 0,lambda = grid, family = binomial)
cv.out.ridge = cv.glmnet(x.train, y.train, alpha = 0, family = binomial)

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




```
# the lambda we will choose will be very small. The closer lambda is to zero,
# the closer the model comes to the least squared estimates.


bestLambda2 = cv.out.ridge$lambda.min
bestLambda2
```

```
## [1] 0.02230634
```

```
ridge.pred = ifelse(predict(ridge.mod, s = bestLambda2,
                            newx = x[test.indices, ]) > 0.5, 'Yes', 'No')
calc_error_rate(ridge.pred, y.test) # Test error rate is 0.248. This has a
```

```
## [1] 0.2484076
```

```
# larger test error rate than the one we got from using lasso regression.
```

```r
# How about if we use the better cutoff value (0.3619) again?

BestRidge.pred = ifelse(predict(ridge.mod, s = bestLambda2,
                                newx = x[test.indices, ]) > best, 'Yes', 'No')
calc_error_rate(BestRidge.pred, y.test)
```

```
## [1] 0.2271762
```

```r
# Test error rate is 0.227. This is equivalent to the test error rate we had
# gotten with lasso regression when using 0.362 as a cutoff. Although the test
# error rates are equivalent, it would be preferable to use lasso regression
# over ridge regression because it is more interpretable. The data is more
# simple when some coefficients are reduced to zero.
```

```r
# Now let's look at classification trees. Will we get a lower test error
# rate?

library(ISLR)
library(tree)
```

```
## Warning: package 'tree' was built under R version 4.3.2
```

```r
library(maptree)
```

```
## Warning: package 'maptree' was built under R version 4.3.2

## Loading required package: cluster

## Warning: package 'cluster' was built under R version 4.3.2

## Loading required package: rpart
```

```r
set.seed(1)


# Bagging
# Will bagging give us a lower test error rate?
bag.drugs = randomForest(User ~ Age + Gender + Education + Country + Ethnicity
                         + Nscore + Escore + Oscore + Ascore + Cscore +
                           Impulsiveness, data = drugs.train, mtry = 11,
                         importance = TRUE)
bag.drugs
```

```
##
## Call:
##  randomForest(formula = User ~ Age + Gender + Education + Country +      Ethnicity + Nscore + Escore
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 11
##
```

```
##         OOB estimate of  error rate: 26.17%
## Confusion matrix:
##      No Yes class.error
## No  702 175   0.1995439
## Yes 193 336   0.3648393
```

```
# It seems like the 'yes' error rate is far higher than the 'no' error rate.
#The OOB estimate of the error rate is 26.17%, which is not good compared to the
#previous machine learning methods we have already used.
predict.bag = predict(bag.drugs, newdata = drugs.test)

test.bag.err = mean(predict.bag != y.test)
test.bag.err # The test error rate is 0.236.
```

```
## [1] 0.2356688
```

```
# We will compare this model to a random forest model.
```

```
# Will random forests give us a better test error rate?
# Usually, m ~ sqrt(p). In this case, m will equal 3.

set.seed(1)

rf.drugs = randomForest(User ~ Age + Gender + Education + Country + Ethnicity
                          + Nscore + Escore + Oscore + Ascore + Cscore +
                            Impulsiveness, data = drugs.train, mtry = 3,
                          importance = TRUE)
rf.drugs
```

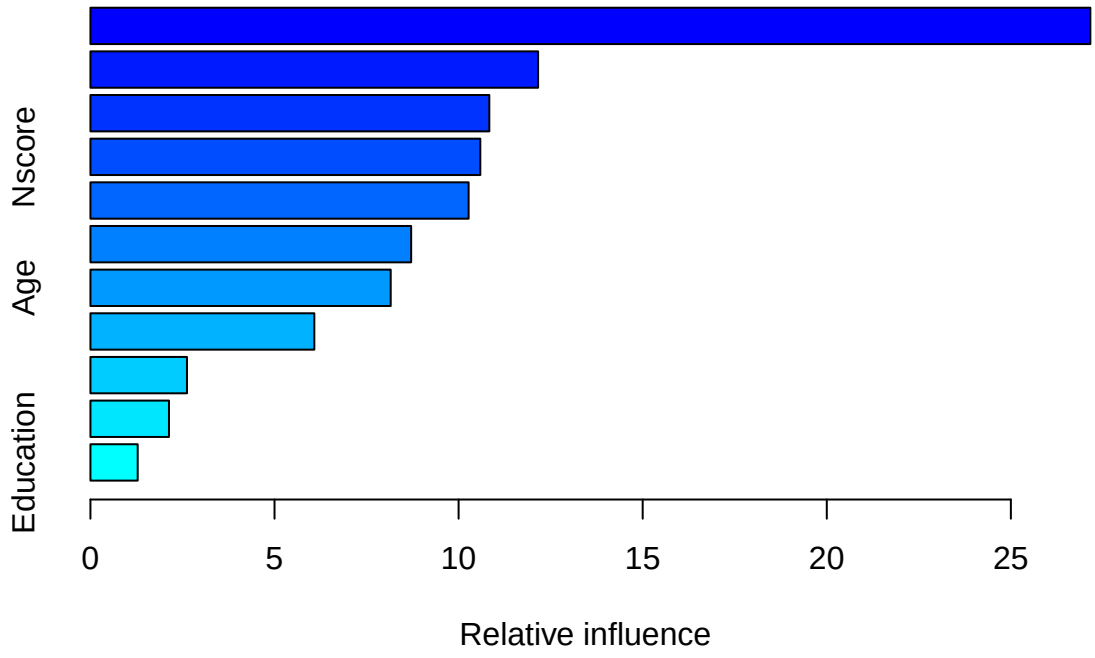
```
##
## Call:
##  randomForest(formula = User ~ Age + Gender + Education + Country +      Ethnicity + Nscore + Escore
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 25.11%
## Confusion matrix:
##      No Yes class.error
## No  714 163   0.1858609
## Yes 190 339   0.3591682
```

```
yhat.rf = predict(rf.drugs, newdata = drugs.test)
test.rf.error = mean(yhat.rf != y.test)
test.rf.error # Test error rate is 0.231. This is a slight improvement
```

```
## [1] 0.2314225
```

```
# over bagging
```

```r
# Analyzing the data with boosting

boost.drugs = gbm(ifelse(User == 'Yes', 1, 0) ~ Age + Gender + Education +
                    Country + Ethnicity + Nscore + Escore + Oscore + Ascore +
                    Cscore + Impulsiveness, data = drugs.train, distribution =
                    'bernoulli', n.trees = 500, interaction.depth = 2)
summary(boost.drugs)
```




```
##                             var   rel.inf
## Country                 Country 27.159309
## Cscore                   Cscore 12.158612
## Escore                   Escore 10.833991
## Nscore                   Nscore 10.589565
## Oscore                   Oscore 10.271898
## Ascore                   Ascore  8.711430
## Age                         Age  8.155529
## Impulsiveness Impulsiveness  6.081072
## Gender                   Gender  2.621745
## Ethnicity             Ethnicity  2.133232
## Education             Education  1.283615
```

```r
# This shows us that Country is the most important variable by far.
# This is because it has the highest influence out of all the other predictors.
# This also shows us that gender, ethnicity, and education are relatively
# uninfluential.
```

```
# What is the error rate using this boosted model?

yhat.boost = predict(boost.drugs, newdata = drugs.test, n.trees = 500, type =
                       'response')

# We will convert the probability to labels
yhat.boost = ifelse(yhat.boost > 0.5, 'Yes', 'No')

test.boost.err = mean(yhat.boost != y.test)
test.boost.err # Test error rate is 0.246.
```

```
## [1] 0.2462845
```

```
# Amongst the tree-based methods we've used, random forests is only slightly
# better than bagging and boosting, in terms of test error rate.
```

“‘{}

SUMMARY

Out of all the methods we have used, the method that has given us the best results in terms of test error rate was using the best threshold value to minimize the false negative rate and the false positive rate. We did this by calculating the euclidean distance between each point of (FPR, FNR) and (0,0).

Out of the tree based methods, it seemed like random forests performed the best. Random forests has certain advantages that we may enjoy: it decorrelates the bagged trees, and it makes the resulting prediction less variable.

Out of the regularization methods used on this dataset, lasso regression gets a slight edge. It has an advantage on this dataset, since we used 11 predictors, and there are a substantial amount of them which were not significant nor impactful. Minimizing the coefficients of these variables to zero can make for more interpretable results.

If we were to do this project again, we perhaps would have done the training/test data split a bit differently. Setting aside 25% of our data as test data may not have been the right move. We could have perhaps done a 85/15 split or a 90/10 split to train the data more accurately. Another possible blunder was that our question was too vague. When looking into ‘hard drugs’ and who may use them, we may have overlooked that there are so many different kinds of hard drugs, each with different effects, cultures and aspects that may make each one appealing in vastly different ways. It may have been far easier to predict who used one specific drug, or maybe we could have tried to analyze the use of drugs that fall in the same family. The kind of people who use crack may differ from the kind of people who use meth, and they both may differ vastly from someone who uses heroin. If the question we posed was less vague and more specific, our training and test error rates may have been lower.