Homework 3

PSTAT131-231

Background

The substance use data set you'll be working with is publicly available on the UCI Machine Learning Repository here, and includes information on 5 demographic variables, 7 personality trait indices, and use status of 18 substances for 1885 respondents.

The demographic variables are as follows.

- Age: Age of the participant
- Gender: Gender of the participant (M/F)
- Education: Level of education of the participant
- Country: Country of current residence of the participant
- Ethnicity: Ethnicity of the participant

The personality measurements comprise the following standardized indices:

- Nscore: NEO- FFI- R Neuroticism (Index ranging from 12 to 60)
- Escore: NEO- FFI- R Extraversion (Index ranging from 16 to 59)
- Oscore: NEO- FFI- R Openness (Index ranging from 24 to 60)
- Ascore: NEO- FFI- R Agreeableness (Index ranging from 12 to 60)
- Cscore: NEO- FFI- R Conscientiousness (Index ranging from 17 to 59)
- Impulsive: Impulsiveness measured by BIS-11
- SS: Sensation Seeking measured by ImpSS

Study participants were questioned concerning their use of 18 substances: alcohol, amphetamines, amyl nitrite, benzodiazepine, cannabis, chocolate, cocaine, caffeine, crack, ecstasy, heroin, ketamine, legal highs, LSD, methadone, mushrooms, nicotine, volatile substances, and one fictitious drug (Semeron) which was introduced to identify over-claimers. Their responses were recorded separately for each substance as a factor with the following 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"

Part 1: Exploratory plots

Question 1 (a)

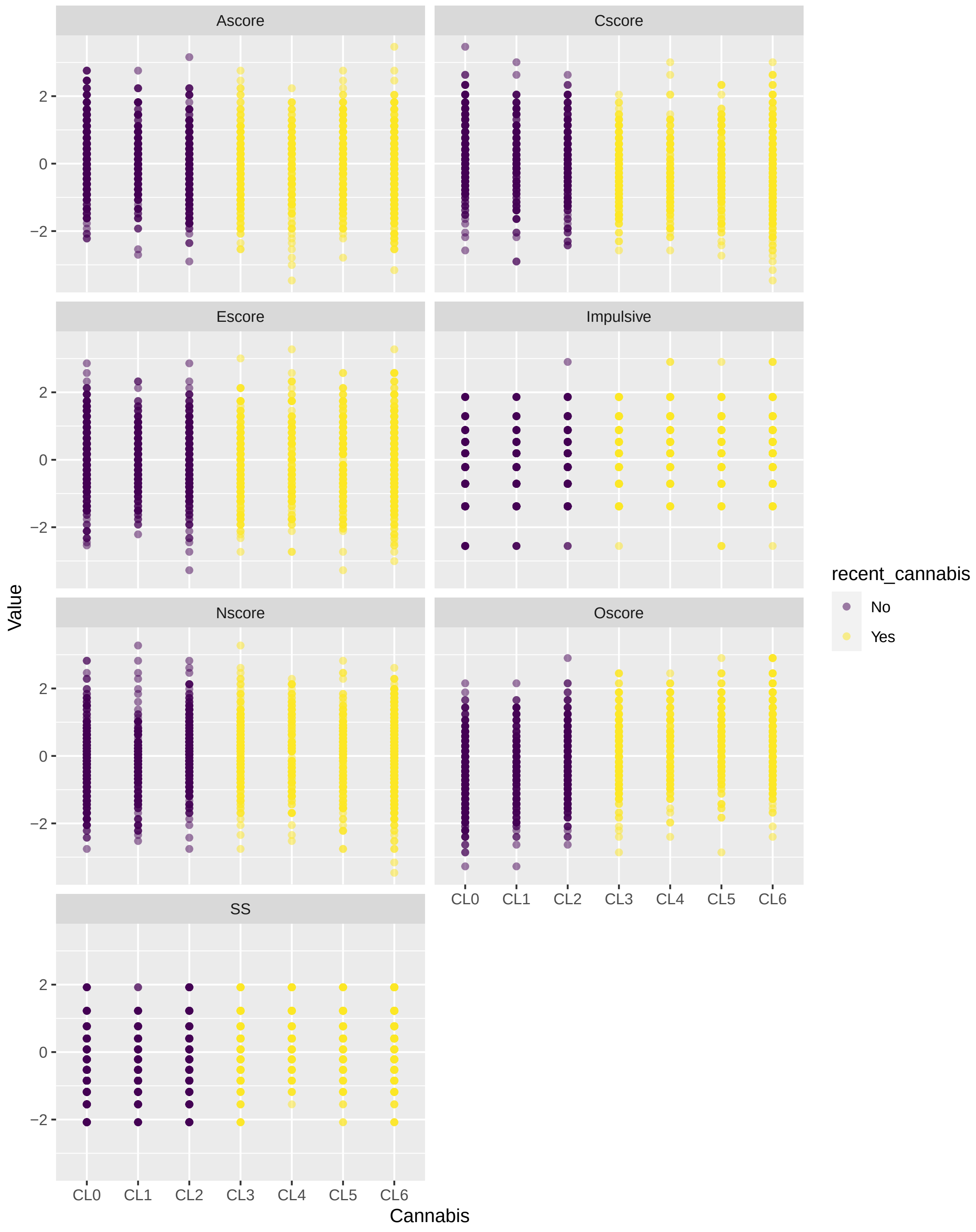
Define a new factor response variable `recent_cannabis` which is "Yes" if a respondent has used cannabis within a year, and "No" otherwise. (Hint: `fct_collapse`; see Lab 3.)

```
# define factor
substance$recent_cannabis = fct_collapse(substance$Cannabis, Yes = c('CL3','CL4','CL5','CL6'), No = c('
```

Question 1 (b)

i) Make an exploratory plot that helps to visualize information in the covariates about recent cannabis use. This is completely open-ended, but your plot should compare covariates between classes for some subset of the variables, and it should be informative and legible with properly labeled axes and aesthetics.

```
# exploratory plot
substance %>%
  gather(Nscore:SS, key = 'Scores', value = 'Value') %>%
  ggplot(aes(x=Cannabis, y = Value)) +
  geom_point(aes(color = recent_cannabis), alpha = 0.5)  +
  facet_wrap(~Scores, nrow = 4, ncol = 2)
```



ii) Interpret your plot – describe what it shows in 1-3 sentences.

Answer Each plot shows the range and spread of each personality measurement. Interestingly, the personality measurement did not vary across frequency of cannabis use. If anything, the Cscore was slightly higher for the No category and the Oscore was slightly higher for the Yes category.

Part 2: Classification tree

Question 2 (a,echo = T)

Exclude the other substance use variables and partition the data into a 70% training set and a 30% test set using `resample_partition`. Set the random number generator seed to `20121` for reproducibility.

Question 2 (b)

i) On the training partition, grow a large initial decision tree $\mathcal{T}_0$ using $n_{min} = 5$. Display a table of the number of splits on each variable.

```
# fit tree
nmin = 5
tree_opts <- tree.control(nobs = nrow(substance_part$train),
                          minsize = nmin,
                          mindev = exp(-6))
t_0 <- tree(recent_cannabis ~ ., data = substance_part$train,
                control = tree_opts, split = 'deviance')
# construct table of splits

# print table
pander(table(t_0$frame$var))
```

Table 1: Table continues below

	Age	Gender	Education	Country	Ethnicity	Nscore	Escore
81	7	2	5	3	1	10	9

Oscore	Ascore	Cscore	Impulsive	SS
14	8	13	3	5

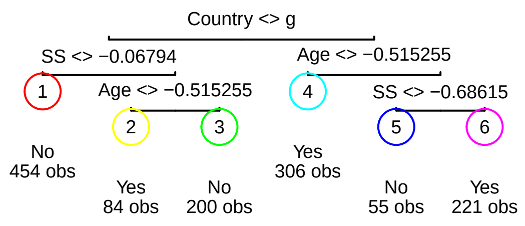
ii) Which three variables are most often used to split the data?

Answer The three variables that are most often used to split the data are Nscore, Cscore, and Oscore

Question 2 (c)

Apply cost-complexity pruning to $\mathcal{T}_0$: select the optimal tuning parameter via 10-fold cross validation (`cv.tree`); then prune $\mathcal{T}_0$ using the optimal tuning parameter (`prune.tree`). Choose the *largest* alpha minimizing the cost criterion. Plot the resulting tree.

```
# select tuning parameter
nfolds = 10
cv_out <- cv.tree(t_0, K = nfolds)

# prune initial tree
cv_df <- tibble(alpha = cv_out$k,
                impurity = cv_out$dev,
                size = cv_out$size)
best_alpha <- slice_min(cv_df, impurity) %>%
  slice_max(size)
t_opt <- prune.tree(t_0, k = best_alpha$alpha)

# plot
draw.tree(t_opt, cex = 0.6, size = 2.5, digits = 2)
```



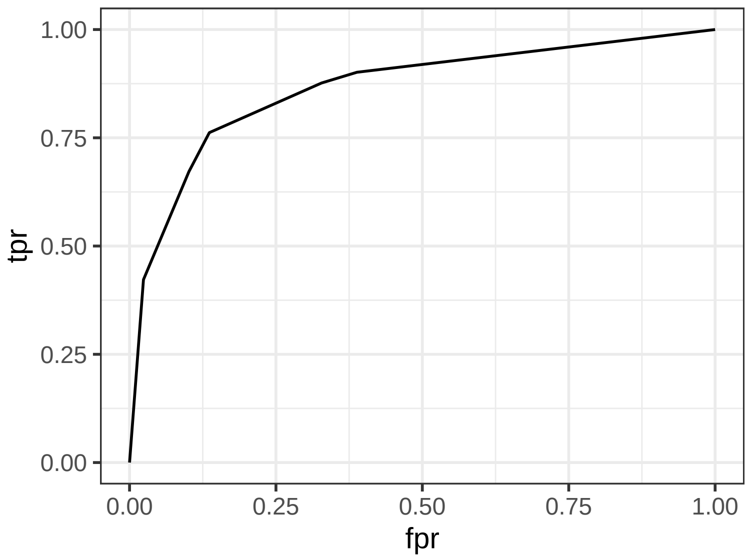
Question 2 (d)

i) From the estimated probabilities at each leaf node, construct and plot an ROC curve.

```
# compute estimated probabilities for each observation in the training set
prob_train = predict(t_opt, newdata = substance_part$train)

# create prediction object (ROCR)
prediction_train = prediction(predictions = prob_train[,2], labels = train$recent_cannabis)

# create performance object (ROCR)
perf_train = performance(prediction.obj = prediction_train, 'tpr','fpr')

# find TPR, FPR
rates_train <- tibble(fpr = slot(perf_train, 'x.values')[[1]],
                      tpr = slot(perf_train, 'y.values')[[1]],
                      thresh = slot(perf_train, 'alpha.values')[[1]])
rates_train <- rates_train %>%
  mutate(youden = tpr-fpr)
rates_train%>%
  filter(near(thresh, 0.5, tol = 0.01))
```

```
## # A tibble: 0 x 4
## # ... with 4 variables: fpr <dbl>, tpr <dbl>, thresh <dbl>, youden <dbl>
```

```
optimal_thresh <- rates_train%>%
  slice_max(youden)

# construct plot
rates_train %>%
  ggplot(aes(x = fpr, y = tpr)) +
  geom_line() +
  theme_bw()
```



ii) Should the probability threshold should be adjusted away from 0.5?

```
# codes to explore the question
optimal_thresh
```

```
## # A tibble: 1 x 4
##     fpr   tpr thresh youden
##   <dbl> <dbl>  <dbl>  <dbl>
## 1 0.136 0.762  0.738  0.626
```

Answer The best threshold value is around 0.738. So, yes, the probability threshold should be adjusted away from 0.5.

Question 2 (e)

Estimate class probabilities on the test partition, convert to predicted classifications (determine the threshold based on your answer in 2 (d)ii), and compute the misclassification error rates. Display the error rates in a 2x2 table.

```
# class probabilities on test partition
prob_test = predict(t_opt, test)
```

```r
# predicted class labels
pred_label = factor(prob_test[,2] >= optimal_thresh$thresh, labels = c('No','Yes'))

# misclassification error rates
errors_test<- table(class = test$recent_cannabis,
                        pred = pred_label)
errors_test/rowSums(errors_test)
```

```
##      pred
## class        No       Yes
##   No  0.8470588 0.1529412
##   Yes 0.2677419 0.7322581
```

Part 3: Ensemble methods

In this part, use the same partitioning of the data from Part 2.

Question 3 (a)

From the training partition, grow a random forest comprising 100 trees using 5 candidate variables at each split to grow the trees. Display a summary of the fitted model.

```
# grow forest
rf = randomForest(recent_cannabis ~ ., data=train, ntree = 100, mtry = 5, importance = TRUE)

# show summary
rf
```

```
##
## Call:
##  randomForest(formula = recent_cannabis ~ ., data = train, ntree = 100,      mtry = 5, importance = 
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 18.56%
## Confusion matrix:
##      No Yes class.error
## No  513 118   0.1870048
## Yes 127 562   0.1843251
```

Question 3 (b)

i) Compute the variable importance scores based on accuracy decrease, and display the result in a table with variables arranged by decreasing accuracy.

```
# compute importance scores
imp = rf$importance

# arrange in table
imp = imp %>% as.data.frame() %>% arrange(desc(MeanDecreaseAccuracy))
# display table
imp %>% pander()
```

	No	Yes	MeanDecreaseAccuracy	MeanDecreaseGini
Country	0.1048	0.04815	0.07499	150.2
SS	0.05907	0.03602	0.04701	84.25
Age	0.04794	0.03353	0.0403	74.22
Oscore	0.02681	0.0301	0.02856	69.49
Cscore	0.02346	0.008415	0.01564	59.87
Education	0.01395	0.008372	0.01099	37.55
Impulsive	0.01953	-0.001574	0.008516	30.87
Escore	0.01438	0.002767	0.008217	43.51
Ascore	0.004896	0.003934	0.004381	42.32

	No	Yes	MeanDecreaseAccuracy	MeanDecreaseGini
Gender	0.003446	0.003862	0.003703	12.02
Nscore	0.002269	0.00188	0.002054	46.42
Ethnicity	-0.0002461	0.001725	0.0007729	7.787

ii) Which variables seem most important? How does this compare with the set of variables that figured most often in splitting the data for the decision tree in Part 2?

Answer The variables that seem the most important are Country, SS, Age, and Oscore. This is pretty different from the variables that split the most often in Part 2. The only variable that matches up is Oscore.

Question 3 (c)

i) Calculate predictions on the test partition and cross-classify with the true labels. Display the misclassification rates in a 2x2 table.

```
# calculate predictions
pred_test = predict(rf, test, type = 'class')

# compute misclassification errors
errors_test1<- table(class = test$recent_cannabis,
                     pred = pred_test)
# print table
errors_test1/rowSums(errors_test1)
```

```
##      pred
## class        No       Yes
##   No  0.8078431 0.1921569
##   Yes 0.1774194 0.8225806
```

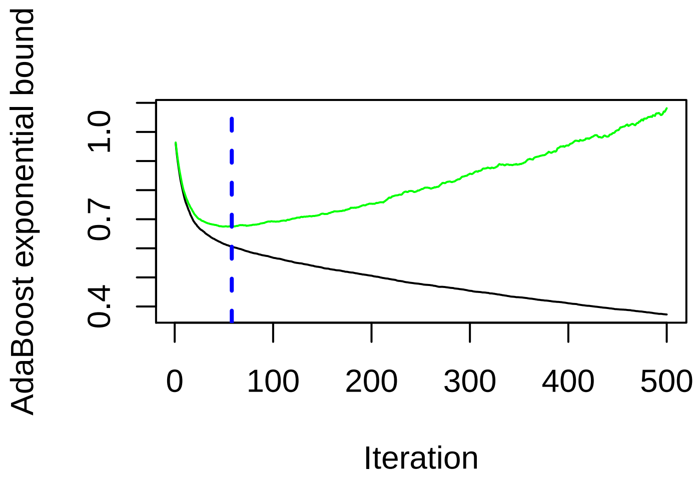
ii) Does the random forest offer an improvement over a single classification tree?

Answer Yes, the random forest offers an improvement over a single classification tree. The overall errors have decreased.

Question 3 (d)

i) Fit a boosted model to the training partition using 100 trees with up to order 3 interaction terms, and use 5-fold cross validation to select the number of boosting iterations that optimize estimated prediction error. Compute the predictions on the test partition and cross-classify the results with the true labels. Show the result in a 2x2 table.

```
# fit boosted model
fit_gbm = gbm(unclass(recent_cannabis)-1~ .,
              distribution = 'adaboost',
              data = train,
              interaction.depth = 3,
              n.trees = 500,
              cv.folds = 5
              )
# select boosting iterations
best_m = gbm.perf(fit_gbm, method = 'cv')
```

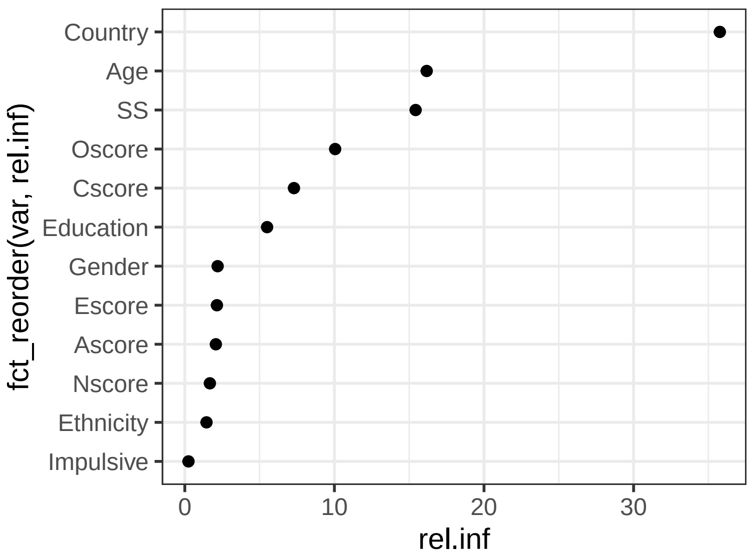


```
# compute predictions on test set
preds = predict(fit_gbm, test, n.trees = best_m)
prob = 1/(1+exp(-preds))
y_hat = factor(prob > 0.5, labels = c('No','Yes'))
y = factor(test$recent_cannabis, labels = c('No','Yes'))

# compute misclassification errors
errors_test2 = table(class = y, pred  = y_hat)

# print table
errors_test2/rowSums(errors_test2)
```

```
##      pred
## class        No       Yes
##   No  0.7921569 0.2078431
##   Yes 0.1709677 0.8290323
```

ii) Compute and plot variable importance measures.

```
##               rel.inf
## Country   35.7655015
## Age       16.1697613
## SS        15.4307074
## Oscore    10.0504572
## Cscore     7.2967956
## Education  5.5003156
## Gender     2.1938740
## Escore     2.1465815
## Ascore     2.0756626
## Nscore     1.6754065
## Ethnicity  1.4499714
## Impulsive  0.2449654
```



iii) How do these results compare with those of the previous models?

Answer This matched the previous random forest model well. The variables Country, SS, Age, and Oscore matched the previous model in importance values. Our first model had the highest error rates. This current model had slightly higher error rates than the previous random forest model.

OPTIONAL

Use 5-fold cross-validation to select the shrinkage parameter λ from a grid of values between 0.01 and 0.2. To do this, you will need to cross-validate the *entire procedure*, including selection of the best number of iterations M. Repeat the previous part using the selected parameter. Does the optimal number of iterations change? Does the classification accuracy improve?

Codes

```
library(tidyverse)
library(modelr)
library(randomForest)
library(gbm)
library(ROCR)
library(pander)
library(tree)
library(maptree)
knitr::opts_chunk$set(echo=F,
                      eval=T,
                      cache=F,
                      results='markup',
                      message=F,
                      warning=F,
                      fig.height=3,
                      fig.width=4,
                      fig.align='center')
load('substance.RData')
# define factor
substance$recent_cannabis = fct_collapse(substance$Cannabis, Yes = c('CL3','CL4','CL5','CL6'), No = c('
# exploratory plot
substance %>%
  gather(Nscore:SS, key = 'Scores', value = 'Value') %>%
  ggplot(aes(x=Cannabis, y = Value)) +
  geom_point(aes(color = recent_cannabis), alpha = 0.5)  +
  facet_wrap(~Scores, nrow = 4, ncol = 2)
# set rng
set.seed(20121)

# partition data
substance = substance %>%
  select(-(Alcohol:VSA))
substance = substance %>%
  select(-(ID))
substance_part = resample_partition(substance, c(test = 0.3, train = 0.7))
train = as.tibble(substance_part$train)
test = as.tibble(substance_part$test)
# fit tree
nmin = 5
tree_opts <- tree.control(nobs = nrow(substance_part$train),
                          minsize = nmin,
                          mindev = exp(-6))
t_0 <- tree(recent_cannabis ~ ., data = substance_part$train,
            control = tree_opts, split = 'deviance')
# construct table of splits

# print table
pander(table(t_0$frame$var))
# select tuning parameter
nfolds = 10
cv_out <- cv.tree(t_0, K = nfolds)
```

```r
# prune initial tree
cv_df <- tibble(alpha = cv_out$k,
                impurity = cv_out$dev,
                size = cv_out$size)
best_alpha <- slice_min(cv_df, impurity) %>%
  slice_max(size)
t_opt <- prune.tree(t_0, k = best_alpha$alpha)

# plot
draw.tree(t_opt, cex = 0.6, size = 2.5, digits = 2)

# compute estimated probabilities for each observation in the training set
prob_train = predict(t_opt, newdata = substance_part$train)

# create prediction object (ROCR)
prediction_train = prediction(predictions = prob_train[,2], labels = train$recent_cannabis)

# create performance object (ROCR)
perf_train = performance(prediction.obj = prediction_train, 'tpr','fpr')

# find TPR, FPR
rates_train <- tibble(fpr = slot(perf_train, 'x.values')[[1]],
                      tpr = slot(perf_train, 'y.values')[[1]],
                      thresh = slot(perf_train, 'alpha.values')[[1]])
rates_train <- rates_train %>%
  mutate(youden = tpr-fpr)
rates_train%>%
  filter(near(thresh, 0.5, tol = 0.01))

optimal_thresh <- rates_train%>%
  slice_max(youden)

# construct plot
rates_train %>%
  ggplot(aes(x = fpr, y = tpr)) +
  geom_line() +
  theme_bw()

# codes to explore the question
optimal_thresh
# class probabilities on test partition
prob_test = predict(t_opt, test)

# predicted class labels
pred_label = factor(prob_test[,2] >= optimal_thresh$thresh, labels = c('No','Yes'))

# misclassification error rates
errors_test<- table(class = test$recent_cannabis,
                        pred = pred_label)
errors_test/rowSums(errors_test)

# grow forest
rf = randomForest(recent_cannabis ~ ., data=train, ntree = 100, mtry = 5, importance = TRUE)
```

```r
# show summary
rf
# compute importance scores
imp = rf$importance

# arrange in table
imp = imp %>% as.data.frame() %>% arrange(desc(MeanDecreaseAccuracy))
# display table
imp %>% pander()
# calculate predictions
pred_test = predict(rf, test, type = 'class')

# compute misclassification errors
errors_test1<- table(class = test$recent_cannabis,
                        pred = pred_test)
# print table
errors_test1/rowSums(errors_test1)
# fit boosted model
fit_gbm = gbm(unclass(recent_cannabis)-1~ .,
              distribution = 'adaboost',
              data = train,
              interaction.depth = 3,
              n.trees = 500,
              cv.folds = 5
              )
# select boosting iterations
best_m = gbm.perf(fit_gbm, method = 'cv')

# compute predictions on test set
preds = predict(fit_gbm, test, n.trees = best_m)
prob = 1/(1+exp(-preds))
y_hat = factor(prob > 0.5, labels = c('No','Yes'))
y = factor(test$recent_cannabis, labels = c('No','Yes'))

# compute misclassification errors
errors_test2 = table(class = y, pred  = y_hat)

# print table
errors_test2/rowSums(errors_test2)
# compute importance measures
import_gbm <- summary(fit_gbm, n.trees = best_m, plotit = F)
import_gbm[2]
# construct plot
summary(fit_gbm, n.trees = best_m, plotit = F) %>%
  ggplot(aes(y=fct_reorder(var,rel.inf), x = rel.inf)) +geom_point() + theme_bw()
```