Random Forest

Carrie Cheng

2023-04-30

```
library(dplyr)
dat <- read.csv('brfss_final.csv')
outcome <- data.frame(dat$X,dat$MICHD,dat$CVDINFR4,dat$CVDCRHD4)
outcome %>% group_by(dat.MICHD) %>% summarise(count=n())
```

```
## # A tibble: 2 x 2
##   dat.MICHD count
##       <int> <int>
## 1         1 14580
## 2         2 14580
```

```
outcome %>% group_by(dat.CVDINFR4) %>% summarise(count=n())
```

```
## # A tibble: 4 x 2
##   dat.CVDINFR4 count
##          <int> <int>
## 1            1  9188
## 2            2 19802
## 3            7   160
## 4            9    10
```

```
outcome %>% group_by(dat.CVDCRHD4) %>% summarise(count=n())
```

```
## # A tibble: 4 x 2
##   dat.CVDCRHD4 count
##          <int> <int>
## 1            1  9729
## 2            2 18874
## 3            7   550
## 4            9     7
```

```
## remove the ones that responded don't know & not sure in CVDINFR4 & CVDCRHD4
dat <- dat[-which(dat$CVDINFR4 == 7 | dat$CVDINFR4 == 9),]
dat <- dat[-which(dat$CVDCRHD4 == 7 | dat$CVDCRHD4 == 9),]
# remove columns that has only 1 value for all rows
dat <- dat[ , -which(names(dat) %in% c("MEDSHEPB","TOLDCFS", "HAVECFS", "WORKCFS"))]
```

Drop columns with more than 5% data missing, impute the rest using KNN

```r
# convert outcome variables
dat$MICHD <- factor(2-dat$MICHD)
dat$CVDINFR4 <- factor(2-dat$CVDINFR4)
dat$CVDCRHD4 <- factor(2-dat$CVDCRHD4)
# i believe X is the index column, not needed
# remove weights
dat <- dat[, !colnames(dat) %in% c('X', 'LLCPWT2', 'LLCPWT', 'CLLCPWT','STRWT','WT2RAKE')]
dat <- dat[, !colnames(dat) %in% c('QSTVER', 'STSTR','RAWRAKE')] # remove based on knowledge
threshold <- .05
ncol(dat) # 190
```

```
## [1] 187
```

```r
dat <- dat[, colMeans(is.na(dat)) <= threshold]
ncol(dat) # 52 columns left
```

```
## [1] 49
```

```r
columns_to_impute <- colnames(dat)[colSums(is.na(dat)) > 0]
columns_to_impute
```

```
##  [1] "CPDEMO1B" "VETERAN3" "EMPLOY1"  "INCOME3"  "DEAF"     "BLIND"
##  [7] "DECIDE"   "DIFFWALK" "DIFFDRES" "DIFFALON" "USENOW3"  "METSTAT"
## [13] "URBSTAT"  "MSCODE"   "DRDXAR3"
```

```r
str(dat[,columns_to_impute])
```

```
## 'data.frame':    28433 obs. of  15 variables:
##  $ CPDEMO1B: int  1 1 8 1 1 8 8 1 1 2 ...
##  $ VETERAN3: int  2 2 2 2 1 2 1 2 2 2 ...
##  $ EMPLOY1 : int  8 7 2 7 7 7 7 8 7 7 ...
##  $ INCOME3 : int  77 3 99 77 7 99 5 77 5 10 ...
##  $ DEAF    : int  2 2 2 2 2 2 1 2 2 2 ...
##  $ BLIND   : int  1 2 2 2 2 2 2 2 2 2 ...
##  $ DECIDE  : int  1 2 1 2 1 2 2 2 2 2 ...
##  $ DIFFWALK: int  1 2 2 2 2 1 1 1 2 2 ...
##  $ DIFFDRES: int  2 2 2 2 2 1 2 2 2 2 ...
##  $ DIFFALON: int  1 2 2 2 2 1 1 2 2 2 ...
##  $ USENOW3 : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ METSTAT : int  1 1 1 1 1 2 1 2 1 1 ...
##  $ URBSTAT : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ MSCODE  : int  2 1 3 1 3 2 2 5 2 3 ...
##  $ DRDXAR3 : int  1 2 1 1 2 1 1 2 1 1 ...
```

```r
complete_columns <- colnames(dat)[colSums(is.na(dat)) == 0 &
                                      !colnames(dat) %in% c('MICHD', 'CVDINFR4','CVDCRHD4')]
for (c in columns_to_impute) {
    col <- dat[[c]]
    scaled <- scale(dat[, complete_columns])
    knn <- knn(
```

```
        train = scaled[!is.na(col), complete_columns],
        test  = scaled[is.na(col), complete_columns],
        cl    = dat[!is.na(col), c]
        )

    dat[is.na(col), c] = knn
}
colSums(is.na(dat))
```

```
##  GENHLTH PHYSHLTH MENTHLTH PRIMINSR PERSDOC3 MEDCOST1 CHECKUP1 CVDINFR4
##        0        0        0        0        0        0        0        0
## CVDCRHD4 CVDSTRK3 CHCSCNCR CHCOCNCR CHCCOPD3 ADDEPEV3 CHCKDNY2 DIABETE4
##        0        0        0        0        0        0        0        0
##  MARITAL RENTHOM1 NUMHHOL3 CPDEMO1B VETERAN3  EMPLOY1  INCOME3     DEAF
##        0        0        0        0        0        0        0        0
##    BLIND   DECIDE DIFFWALK DIFFDRES DIFFALON  USENOW3  QSTLANG  METSTAT
##        0        0        0        0        0        0        0        0
##  URBSTAT   MSCODE  DUALUSE  TOTINDA  RFHYPE6  CHOLCH3    MICHD  ASTHMS1
##        0        0        0        0        0        0        0        0
##  DRDXAR3     RACE      SEX    AGE80  CHLDCNT   EDUCAG  SMOKER3  CURECI1
##        0        0        0        0        0        0        0        0
## DROCDY3_
##        0
```

```
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
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:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(ggplot2)
library(ROCR)
set.seed(263)
train_index <- createDataPartition(dat$MICHD, p = 0.8, list = FALSE)
train <- dat[train_index, ]
test <- dat[-train_index, ]
```

Parameter Tuning

Let's tune number of trees ntrees and number of features selected to place split mtry. In the following, let's use 10-fold cross-validation.

```r
## get index of the other two outcomes

index_michd <- which(names(train) == "MICHD")
index_infr <- which(names(train) == "CVDINFR4")
index_crhd <- which(names(train) == "CVDCRHD4")
```

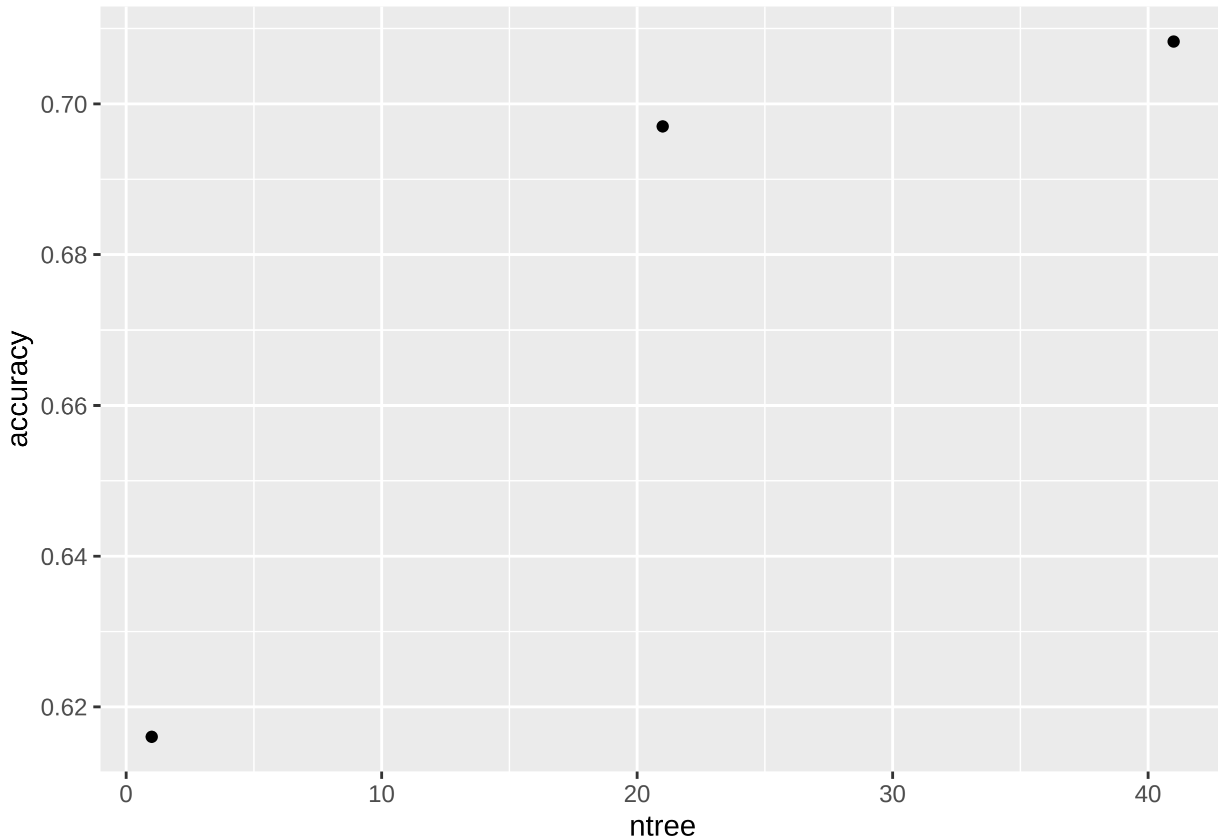
Tune number of trees

Let's set mtry = 10.

```r
ntree <- seq(1, 51, by = 20)
accuracy <- sapply(ntree, function(n){
  train(as.factor(MICHD) ~ ., method = "rf",
        data = train[, -c(index_infr, index_crhd)],
        tuneGrid = data.frame(mtry = 10),
        ntree = n, trControl = trainControl(method = "cv", number = 10))$results$Accuracy
})

qplot(ntree, accuracy)
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



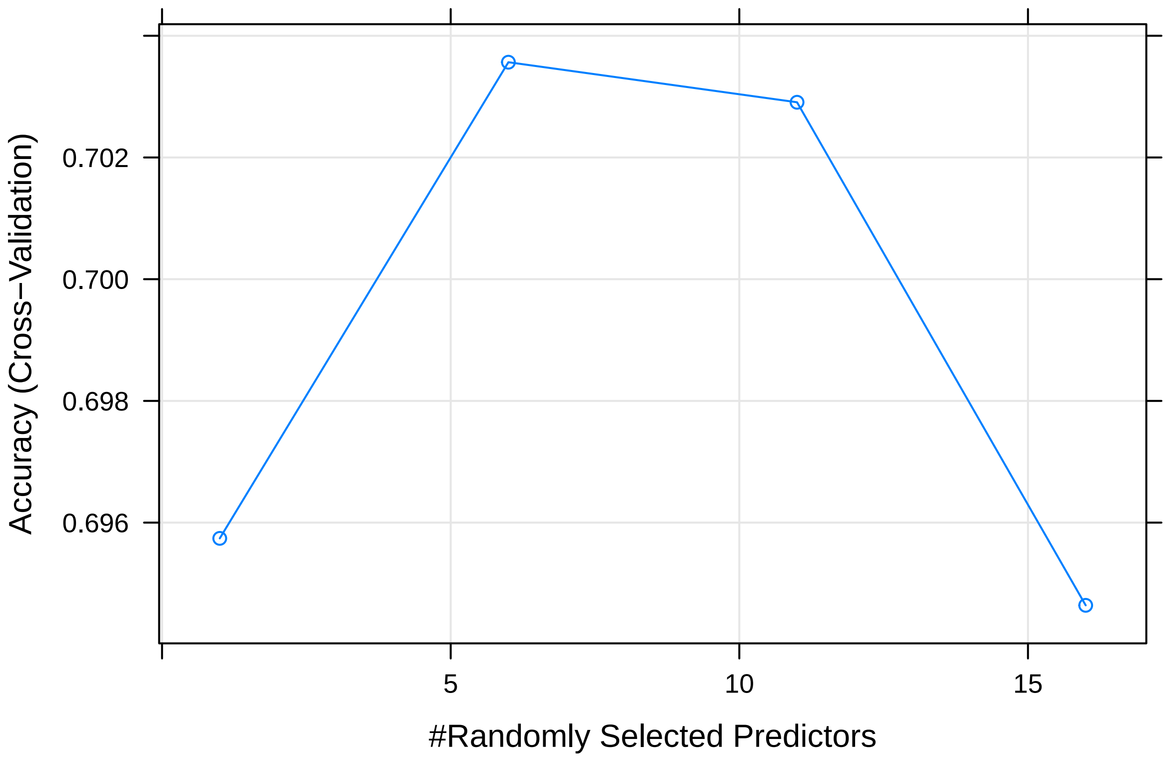

```
best_ntree <- ntree[which(accuracy == max(accuracy))]
print(paste("The best ntree is", best_ntree))
```

```
## [1] "The best ntree is 41"
```

Tune mtry

```
train_rf <- train(as.factor(MICHD) ~ ., method = "rf",
                  data = train[, -c(index_infr, index_crhd)],
                  tuneGrid = data.frame(mtry = seq(1, 20, by = 5)),
                  ntree = 20,
                  nodesize = 10, trControl = trainControl(method = "cv", number = 10))

plot(train_rf)
```




```r
best_mtry <- train_rf$bestTune

result_cv <- train_rf$results

print(paste("The best mtry is ", best_mtry))
```

```
## [1] "The best mtry is  6"
```

Use the best model to train random forest

The below is the confusion matrix on the test set.

```r
rf_best <- randomForest(as.factor(MICHD) ~.,
                        data = train[, -c(index_infr, index_crhd)],
                        mtry = best_mtry[[1]], ntree = best_ntree, nodesize = 10)

pred_test <- predict(rf_best, newdata = test)
cm_test <- confusionMatrix(pred_test, as.factor(test$MICHD))

cm_test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2050  743
##          1  866 2027
```

```
##
##                Accuracy : 0.717
##                  95% CI : (0.7051, 0.7287)
##     No Information Rate : 0.5128
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4343
##
##  Mcnemar's Test P-Value : 0.002354
##
##             Sensitivity : 0.7030
##             Specificity : 0.7318
##          Pos Pred Value : 0.7340
##          Neg Pred Value : 0.7007
##              Prevalence : 0.5128
##          Detection Rate : 0.3605
##    Detection Prevalence : 0.4912
##       Balanced Accuracy : 0.7174
##
##        'Positive' Class : 0
##
```

```
metric_test <- c(cm_test$overall[["Accuracy"]],
                 cm_test$byClass[c("Sensitivity","Specificity")])

cat(paste("The overall accuracy using the best tuned random forest model is",
      metric_test[1], "\n",
      "Sensitivity is", metric_test[2], "\n",
      "Specificity is", metric_test[3]))
```

```
## The overall accuracy using the best tuned random forest model is 0.717024270137179
##  Sensitivity is 0.703017832647462
##  Specificity is 0.731768953068592
```