Ensemble techniques

Team: Abed Ahmed (ASA190005) & Dylan Kapustka (DLK190000)

Date: 09/25/2022

HomeWork 5 - Kernel and Ensemble Methods

```
data <- read.csv(file="C:/Users/Abed/Desktop/employeeData.csv",header=TRUE)
data <- na.omit(data)
data$Attrition <- as.factor(data$Attrition)
set.seed(2)
library(caTools)

data$Attrition <- as.numeric(data$Attrition)
for(i in 1:length(data$Attrition)){
  if(data$Attrition[i] == 1){
    data$Attrition[i] = 0
  }
  else {
    data$Attrition[i] = 1
  }
}

spec <- c(train=.6, test=.4)
i <- sample(cut(1:nrow(data),
                nrow(data)*cumsum(c(0,spec)), labels=names(spec)))
train <- data[i=="train",]
test <- data[i=="test",]
```

Decision Tree

```
library(tree)

start <- Sys.time()

treeModel <- tree(Attrition ~ Age, data = train, method = "class")
pred <- predict(treeModel, newdata=test)
table <- table(test$Attrition,pred)
acc = sum(diag(table)) / sum(table)

end <- Sys.time()
taken1 <- end - start

print(paste("Accuracy of the decision tree is: ", acc))
```

```
## [1] "Accuracy of the decision tree is:  0.802053622361666"
```

```
print(paste("Time taken: ", taken1))
```

```
## [1] "Time taken:  0.016826868057251"
```

xgBoost

```
library(xgboost)

start <- Sys.time()

trainX = data.matrix(train[1])
trainY = data.matrix(train[2])

testX = data.matrix(test[1])
testY = data.matrix(test[2])

xgbTrain = xgb.DMatrix(data = trainX, label = trainY)
xgbTest = xgb.DMatrix(data = testX, label = testY)

list = list(train=xgbTrain, test=xgbTest)

model = xgb.train(data = xgbTrain, max.depth = 3, watchlist=list, nrounds = 70)
```

```
## [1]  train-rmse:0.434095 test-rmse:0.432159
## [2]  train-rmse:0.397728 test-rmse:0.394372
## [3]  train-rmse:0.378190 test-rmse:0.374098
## [4]  train-rmse:0.368111 test-rmse:0.363667
## [5]  train-rmse:0.362881 test-rmse:0.358221
## [6]  train-rmse:0.360092 test-rmse:0.355293
## [7]  train-rmse:0.358676 test-rmse:0.353410
## [8]  train-rmse:0.357852 test-rmse:0.352668
## [9]  train-rmse:0.357143 test-rmse:0.352118
## [10] train-rmse:0.356657 test-rmse:0.352155
## [11] train-rmse:0.356365 test-rmse:0.351841
## [12] train-rmse:0.356114 test-rmse:0.351822
## [13] train-rmse:0.355917 test-rmse:0.351978
## [14] train-rmse:0.355712 test-rmse:0.352066
## [15] train-rmse:0.355591 test-rmse:0.352093
## [16] train-rmse:0.355394 test-rmse:0.352234
## [17] train-rmse:0.355251 test-rmse:0.352096
## [18] train-rmse:0.355097 test-rmse:0.352216
## [19] train-rmse:0.354994 test-rmse:0.352162
## [20] train-rmse:0.354851 test-rmse:0.352282
## [21] train-rmse:0.354758 test-rmse:0.352216
## [22] train-rmse:0.354708 test-rmse:0.352240
## [23] train-rmse:0.354671 test-rmse:0.352374
## [24] train-rmse:0.354636 test-rmse:0.352441
## [25] train-rmse:0.354605 test-rmse:0.352477
## [26] train-rmse:0.354561 test-rmse:0.352545
```

```
## [27] train-rmse:0.354492 test-rmse:0.352647
## [28] train-rmse:0.354454 test-rmse:0.352554
## [29] train-rmse:0.354440 test-rmse:0.352519
## [30] train-rmse:0.354421 test-rmse:0.352564
## [31] train-rmse:0.354405 test-rmse:0.352596
## [32] train-rmse:0.354385 test-rmse:0.352692
## [33] train-rmse:0.354369 test-rmse:0.352719
## [34] train-rmse:0.354359 test-rmse:0.352761
## [35] train-rmse:0.354332 test-rmse:0.352835
## [36] train-rmse:0.354326 test-rmse:0.352861
## [37] train-rmse:0.354297 test-rmse:0.352945
## [38] train-rmse:0.354283 test-rmse:0.353017
## [39] train-rmse:0.354269 test-rmse:0.352973
## [40] train-rmse:0.354254 test-rmse:0.352927
## [41] train-rmse:0.354244 test-rmse:0.352988
## [42] train-rmse:0.354241 test-rmse:0.353031
## [43] train-rmse:0.354237 test-rmse:0.353098
## [44] train-rmse:0.354226 test-rmse:0.353112
## [45] train-rmse:0.354217 test-rmse:0.353130
## [46] train-rmse:0.354212 test-rmse:0.353172
## [47] train-rmse:0.354209 test-rmse:0.353188
## [48] train-rmse:0.354204 test-rmse:0.353238
## [49] train-rmse:0.354200 test-rmse:0.353257
## [50] train-rmse:0.354196 test-rmse:0.353236
## [51] train-rmse:0.354193 test-rmse:0.353267
## [52] train-rmse:0.354188 test-rmse:0.353283
## [53] train-rmse:0.354183 test-rmse:0.353281
## [54] train-rmse:0.354179 test-rmse:0.353297
## [55] train-rmse:0.354177 test-rmse:0.353309
## [56] train-rmse:0.354170 test-rmse:0.353337
## [57] train-rmse:0.354167 test-rmse:0.353339
## [58] train-rmse:0.354166 test-rmse:0.353370
## [59] train-rmse:0.354163 test-rmse:0.353389
## [60] train-rmse:0.354161 test-rmse:0.353385
## [61] train-rmse:0.354160 test-rmse:0.353401
## [62] train-rmse:0.354158 test-rmse:0.353405
## [63] train-rmse:0.354157 test-rmse:0.353421
## [64] train-rmse:0.354156 test-rmse:0.353433
## [65] train-rmse:0.354156 test-rmse:0.353452
## [66] train-rmse:0.354154 test-rmse:0.353430
## [67] train-rmse:0.354152 test-rmse:0.353452
## [68] train-rmse:0.354151 test-rmse:0.353458
## [69] train-rmse:0.354150 test-rmse:0.353471
## [70] train-rmse:0.354148 test-rmse:0.353484
```

```
final = xgboost(data = xgbTrain, max.depth = 3, nround = 56, verbose = 0)

pred2 <- predict(final,newdata=xgbTest)

prob2 <- ifelse(pred2>0.5,1,0)
acc2 <- mean(prob2==testY)

end <- Sys.time()
taken2 <- end - start
```

```
print(paste("Accuracy of xgBoost is: ", acc2))
```

```
## [1] "Accuracy of xgBoost is:  0.844266970907017"
```

```
print(paste("Time taken: ", taken2))
```

```
## [1] "Time taken:  0.174117088317871"
```

Random Forest

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
start <- Sys.time()
rf <- randomForest(Attrition ~ Age, data = train, importance = TRUE, proximity = TRUE,method = "class")
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

```
pred3 <- predict(rf,newdata=test)

prob3 <- ifelse(pred3>0.5,1,0)
acc3 <- mean(prob3==test$Attrition)

end <- Sys.time()
taken3 <- end - start


print(paste("Accuracy of random forest is: ", acc3))
```

```
## [1] "Accuracy of random forest is:  0.844266970907017"
```

```
print(paste("Time taken: ", taken3))
```

```
## [1] "Time taken:  4.65928912162781"
```

FAst ADA Boost

```
start <- Sys.time()

library(adabag)
```

```
## Loading required package: rpart

## Loading required package: caret

## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin

## Loading required package: lattice

## Loading required package: foreach

## Loading required package: doParallel

## Loading required package: iterators

## Loading required package: parallel
```

```
library(caret)

train$Attrition <- as.factor(train$Attrition)

model <- boosting(Attrition ~ Age, data=train)

pred4 <- predict(model,newdata=test)


acc4 <- mean(pred4$class==test$Attrition)


print(paste("Accuracy of adaBoost is: ", acc4))
```

```
## [1] "Accuracy of adaBoost is:  0.844837421563035"
```

```
end <- Sys.time()
taken4 <- end - start
print(paste("Time taken: ", taken4))
```

```
## [1] "Time taken:  5.09910678863525"
```

Summary and Analysis of Ensembles

Ranking in Accuracy:

1 - Tie between Random Forest, and xgBoost at 84.4% 2 - Decision Trees came in at 80% 3 - FastAdaBoost came in at 14%

Time:

1 - Decision Trees: 0.016826868057251s

2 - xgBoost: 0.174117088317871s

3 - Random Forest:: 4.65928912162781s

4 - fastAdaBoost 5.09910678863525s‘