Ensemble Methods

Jered Hightower, Haniyyah Hamid

https://www.kaggle.com/datasets/vicsuperman/prediction-of-music-genre

```r
original <- read.csv("music_genre.csv")
original$key <- factor(original$key)
original$tempo <- as.numeric(original$tempo)
```

```
## Warning: NAs introduced by coercion
```

```r
original$mode <- factor(original$mode)
original$music_genre <- factor(original$music_genre)

df <- original[, -c(1,2,3,7,8,16)]

df <- df[complete.cases(df),]

df$key <- droplevels(df$key)
df$mode <- droplevels(df$mode)
df$music_genre <- droplevels(df$music_genre)

str(df)
```

```
## 'data.frame':    45020 obs. of  12 variables:
##  $ popularity      : num  27 31 28 34 32 46 43 39 22 30 ...
##  $ acousticness    : num  0.00468 0.0127 0.00306 0.0254 0.00465 0.0289 0.0297 0.00299 0.00934 0.855
##  $ danceability    : num  0.652 0.622 0.62 0.774 0.638 0.572 0.809 0.509 0.578 0.607 ...
##  $ instrumentalness: num  7.92e-01 9.50e-01 1.18e-02 2.53e-03 9.09e-01 7.74e-06 9.03e-01 2.76e-04 1.
##  $ key             : Factor w/ 12 levels "A","A#","B","C",..: 2 6 12 5 10 3 11 9 1 10 ...
##  $ liveness        : num  0.115 0.124 0.534 0.157 0.157 0.106 0.0635 0.178 0.111 0.106 ...
##  $ loudness        : num  -5.2 -7.04 -4.62 -4.5 -6.27 ...
##  $ mode            : Factor w/ 2 levels "Major","Minor": 2 2 1 1 1 1 2 2 2 2 ...
##  $ speechiness     : num  0.0748 0.03 0.0345 0.239 0.0413 0.351 0.0484 0.268 0.173 0.0345 ...
##  $ tempo           : num  101 115 128 128 145 ...
##  $ valence         : num  0.759 0.531 0.333 0.27 0.323 0.23 0.761 0.273 0.203 0.307 ...
##  $ music_genre     : Factor w/ 10 levels "Alternative",..: 6 6 6 6 6 6 6 6 6 6 ...
```

Train Test Split

```r
set.seed(1234)
i <- sample(nrow(df), .75*nrow(df), replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Decision Tree

```r
library(mltools)
library(tree)

startTime <- Sys.time()

tree <- tree(music_genre~., data = train)

endTime <- Sys.time()
print(paste("Total time: ", endTime - startTime))
```

```
## [1] "Total time:  0.16052508354187"
```

```r
tree_pred <- predict(tree, newdata=test, type="class")
table(tree_pred, test$music_genre)
```

```
##
## tree_pred     Alternative Anime Blues Classical Country Electronic Hip-Hop Jazz
##   Alternative         279     9    65        51     259         88      60  138
##   Anime                 2   497    66        19       8         39       0   17
##   Blues                 1   371   449        60       4        327       1  143
##   Classical             3   148    59       846      17         21       0  130
##   Country             249   124   223        32     572        116       7  196
##   Electronic           96    46   163        68      44        417       1  386
##   Hip-Hop             251     6    26         6      39         76     835   42
##   Jazz                  0     0     0         0       0          0       0    0
##   Rap                   0     0     0         0       0          0       0    0
##   Rock                209     3    47        18     195         50     175   56
##
## tree_pred     Rap Rock
##   Alternative  56  178
##   Anime         3    2
##   Blues         0    3
##   Classical     0    1
##   Country       2    3
##   Electronic    0    2
##   Hip-Hop     800  163
##   Jazz          0    0
##   Rap           0    0
##   Rock        302  789
```

```r
acc_dt <- mean(tree_pred==test$music_genre)
mcc_dt <- mcc(factor(tree_pred), test$music_genre)
print(paste("accuracy=", acc_dt))
```

```
## [1] "accuracy= 0.416170590848512"
```

```r
print(paste("mcc=", mcc_dt))
```

```
## [1] "mcc= 0.359839996825828"
```

Random Forest

```r
library(randomForest)
```

```
## randomForest 4.7-1.1

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

```
set.seed(1234)

startTime <- Sys.time()

rf <- randomForest(music_genre~., data=train, importance=TRUE)

endTime <- Sys.time()
print(paste("Total time: ", endTime - startTime))
```

```
## [1] "Total time:  4.21836849848429"
```

```
rf
```

```
##
## Call:
##  randomForest(formula = music_genre ~ ., data = 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: 46.27%
## Confusion matrix:
##             Alternative Anime Blues Classical Country Electronic Hip-Hop Jazz
## Alternative        1187    23    82         8     441        194     308  225
## Anime               108  2341   214       253     152        152       2   50
## Blues               177   267  1806        54     287        192      13  408
## Classical            81   102    82      2871      20         72       0  159
## Country             263    43   188         4    1875         37      55  173
## Electronic          234   165   230        34      85       1977      69  408
## Hip-Hop             180     3     1         0      48         34    1336   30
## Jazz                135    49   398       237     169        428      81 1754
## Rap                 151     1     3         0      44         17    1897   19
## Rock                512    11    61         9     333         16     124   71
##              Rap Rock class.error
## Alternative  172  765   0.6513950
## Anime          2   19   0.2890981
## Blues          2  166   0.4644128
## Classical      0   13   0.1555882
## Country       62  648   0.4399642
## Electronic    35   95   0.4066627
## Hip-Hop     1639  170   0.6117408
## Jazz          17  145   0.4860826
## Rap          893  316   0.7327148
## Rock         180 2103   0.3850877
```

```
pred <- predict(rf, newdata=test, type="response")
acc_rf <- mean(pred==test$music_genre)
mcc_rf <- mcc(factor(pred), test$music_genre)
print(paste("accuracy=", acc_rf))
```

```
## [1] "accuracy= 0.553176366059529"
```

```r
print(paste("mcc=", mcc_rf))
```

```
## [1] "mcc= 0.504103151697454"
```

boosting from adabag library

```r
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
```

```r
startTime <- Sys.time()

adab1 <- boosting(music_genre~., data=train, boos=TRUE, mfinal=20, coeflearn='Breiman')


endTime <- Sys.time()
print(paste("Total time: ", endTime - startTime))
```

```
## [1] "Total time:  21.1789329051971"
```

```r
summary(adab1)
```

```
##            Length Class   Mode
## formula         3 formula call
## trees          20 -none-  list
## weights        20 -none-  numeric
## votes      337650 -none-  numeric
## prob       337650 -none-  numeric
## class       33765 -none-  character
## importance     11 -none-  numeric
## terms           3 terms   call
## call            6 -none-  call
```

```r
pred <- predict(adab1, newdata=test, type="response")
acc_adabag <- mean(pred$class==test$music_genre)
mcc_adabag <- mcc(factor(pred$class), test$music_genre)
print(paste("accuracy=", acc_adabag))
```

```
## [1] "accuracy= 0.439982230119947"
```

```r
print(paste("mcc=", mcc_adabag))
```

```
## [1] "mcc= 0.389561644308411"
```

XGBoost

```r
library(xgboost)

genres <- df$music_genre
label <- as.integer(df$music_genre) - 1
df$music_genre = NULL

train_label <- label[i]
test_label <- label[-i]

train_matrix <- data.matrix(df[i,])
test_matrix <- data.matrix(df[-i,])

num_class = length(levels(genres))

startTime <- Sys.time()


model <- xgboost(data=train_matrix, label=train_label, nrounds=100, num_class = num_class, objective='mu
```

```
## [1]  train-mlogloss:1.830949
## [2]  train-mlogloss:1.617191
## [3]  train-mlogloss:1.474090
## [4]  train-mlogloss:1.371681
## [5]  train-mlogloss:1.292881
## [6]  train-mlogloss:1.228601
## [7]  train-mlogloss:1.178885
## [8]  train-mlogloss:1.139211
## [9]  train-mlogloss:1.103771
## [10] train-mlogloss:1.073472
## [11] train-mlogloss:1.048241
## [12] train-mlogloss:1.027538
## [13] train-mlogloss:1.008616
## [14] train-mlogloss:0.992096
## [15] train-mlogloss:0.977397
## [16] train-mlogloss:0.964583
## [17] train-mlogloss:0.953058
## [18] train-mlogloss:0.940854
## [19] train-mlogloss:0.931189
## [20] train-mlogloss:0.921321
## [21] train-mlogloss:0.912080
## [22] train-mlogloss:0.903502
## [23] train-mlogloss:0.895982
## [24] train-mlogloss:0.888791
## [25] train-mlogloss:0.881858
## [26] train-mlogloss:0.875147
## [27] train-mlogloss:0.868542
## [28] train-mlogloss:0.862615
## [29] train-mlogloss:0.856649
```

```
## [30] train-mlogloss:0.851716
## [31] train-mlogloss:0.846190
## [32] train-mlogloss:0.840060
## [33] train-mlogloss:0.833777
## [34] train-mlogloss:0.827553
## [35] train-mlogloss:0.822322
## [36] train-mlogloss:0.817467
## [37] train-mlogloss:0.812629
## [38] train-mlogloss:0.807359
## [39] train-mlogloss:0.803514
## [40] train-mlogloss:0.797398
## [41] train-mlogloss:0.793021
## [42] train-mlogloss:0.789241
## [43] train-mlogloss:0.784307
## [44] train-mlogloss:0.780050
## [45] train-mlogloss:0.776037
## [46] train-mlogloss:0.772362
## [47] train-mlogloss:0.766400
## [48] train-mlogloss:0.762245
## [49] train-mlogloss:0.758490
## [50] train-mlogloss:0.754552
## [51] train-mlogloss:0.749949
## [52] train-mlogloss:0.746131
## [53] train-mlogloss:0.741529
## [54] train-mlogloss:0.738587
## [55] train-mlogloss:0.733401
## [56] train-mlogloss:0.729130
## [57] train-mlogloss:0.726558
## [58] train-mlogloss:0.722779
## [59] train-mlogloss:0.718608
## [60] train-mlogloss:0.715014
## [61] train-mlogloss:0.711193
## [62] train-mlogloss:0.707753
## [63] train-mlogloss:0.701851
## [64] train-mlogloss:0.699117
## [65] train-mlogloss:0.695538
## [66] train-mlogloss:0.691145
## [67] train-mlogloss:0.688745
## [68] train-mlogloss:0.686185
## [69] train-mlogloss:0.682394
## [70] train-mlogloss:0.677544
## [71] train-mlogloss:0.673910
## [72] train-mlogloss:0.670411
## [73] train-mlogloss:0.665571
## [74] train-mlogloss:0.663003
## [75] train-mlogloss:0.659735
## [76] train-mlogloss:0.657078
## [77] train-mlogloss:0.654303
## [78] train-mlogloss:0.651949
## [79] train-mlogloss:0.648472
## [80] train-mlogloss:0.645396
## [81] train-mlogloss:0.641935
## [82] train-mlogloss:0.639255
## [83] train-mlogloss:0.636679
```

```
## [84] train-mlogloss:0.633110
## [85] train-mlogloss:0.630806
## [86] train-mlogloss:0.627741
## [87] train-mlogloss:0.623583
## [88] train-mlogloss:0.620868
## [89] train-mlogloss:0.617608
## [90] train-mlogloss:0.614066
## [91] train-mlogloss:0.610585
## [92] train-mlogloss:0.607184
## [93] train-mlogloss:0.603206
## [94] train-mlogloss:0.598898
## [95] train-mlogloss:0.594545
## [96] train-mlogloss:0.591701
## [97] train-mlogloss:0.588164
## [98] train-mlogloss:0.584466
## [99] train-mlogloss:0.581531
## [100]    train-mlogloss:0.579072
```

```
endTime <- Sys.time()
print(paste("Total time: ", endTime - startTime))
```

```
## [1] "Total time:  24.1045889854431"
```

```
summary(model)
```

```
##                 Length  Class              Mode
## handle                1 xgb.Booster.handle externalptr
## raw             3314421 -none-             raw
## niter                 1 -none-             numeric
## evaluation_log        2 data.table         list
## call                 15 -none-             call
## params                3 -none-             list
## callbacks             2 -none-             list
## feature_names        11 -none-             character
## nfeatures             1 -none-             numeric
```

```
probs <- predict(model, test_matrix, reshape=T)
probs <- as.data.frame(probs)
colnames(probs) <- levels(genres)

# Use the predicted label with the highest probability
pred <- apply(probs,1,function(x) colnames(probs)[which.max(x)])
test_label <- levels(genres)[test_label + 1]

acc_xg <- mean(pred==test_label)
mcc_xg <- mcc(pred, test_label)
print(paste("accuracy=", acc_xg))
```

```
## [1] "accuracy= 0.562594402487783"
```

```
print(paste("mcc=", mcc_xg))
```

```
## [1] "mcc= 0.514306852677017"
```

Analysis of Results

Decision Tree: Time-.601 seconds, Acc-.416, MCC-.360

Random Forest: Time-4.575 minutes, Acc-.553, MCC-.504

Adaboost: Time-41.386 seconds, Acc-.439, MCC-.388

XGBoost: Time-14.349 seconds, Acc-.563, MCC-.514

XGBoost I would say is the overall winner here. It achieved the highest accuracy and mcc of all the models tested and within a reasonable time (2nd fastest).

Random forest produced good results on par with XGBoost, but was very computationally expensive and took the longest by far.

Adaboost was alright, but didn't produce results as good as the above models. It's just slightly better than decision tree, but takes much more time than it and XGBoost.

Decision tree ran the absolute fastest, but consequently had the worst accuracy and mcc.