Ensemble Methods

Jered Hightower, Haniyyah Hamid

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

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

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

```
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

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

Logistic Regression Baseline

```
library(nnet)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.2      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
model <- multinom(music_genre~., data = train)
```

```
## # weights:  230 (198 variable)
## initial  value 77746.785665
## iter  10 value 67168.433345
## iter  20 value 65260.642120
## iter  30 value 56000.181934
## iter  40 value 52744.779833
## iter  50 value 47291.408867
## iter  60 value 45682.405372
## iter  70 value 44607.482915
## iter  80 value 43939.879454
## iter  90 value 43528.645055
## iter 100 value 43378.838235
## final  value 43378.838235
## stopped after 100 iterations
```

```
summary(model)
```

```
## Call:
## multinom(formula = music_genre ~ ., data = train)
##
## Coefficients:
##            (Intercept)  popularity acousticness danceability instrumentalness
## Anime        11.788988 -0.27329183   1.29824679    -4.028378        1.7138201
## Blues         6.535135 -0.16788814   1.12668878    -3.104663       -0.6146215
## Classical     5.569906 -0.17706009   3.17316047    -7.929667        1.5265082
## Country       1.009485 -0.05152201   1.27664502     1.190020       -5.1376114
## Electronic    1.834373 -0.13115779  -1.11466367     5.426443        2.8831725
## Hip-Hop     -12.029383  0.10618761   0.25691745     8.070587       -1.1408478
## Jazz          0.535040 -0.08623975   2.43090611     1.229938        2.1662601
## Rap         -12.765089  0.13640149  -0.07164067     7.247705       -1.7100462
## Rock         -6.770276  0.13517921  -0.16525879    -2.011927       -0.3023107
##                  keyA#        keyB        keyC      keyC#        keyD
## Anime       0.07345559  0.13012586  0.25416704  0.4426582  0.26816717
## Blues      -0.49661430 -0.28905797 -0.20320774 -0.5649137  0.03434801
## Classical   0.19359219 -0.11944464  0.06455740  0.3334889  0.40763849
## Country    -0.05948387  0.04980049 -0.30780691 -0.2030529  0.01421242
## Electronic  0.22818893 -0.03117974 -0.03703149  0.3451220 -0.08629558
## Hip-Hop     0.19547476 -0.24076502  0.03846586  0.3729724 -0.26367966
## Jazz        0.60425872 -0.16044059  0.18371930  0.3114141  0.08577498
## Rap         0.21593797 -0.18391441  0.12146436  0.3882362 -0.08293701
```

```
## Rock       -0.20692640 -0.24498055 -0.06024065 -0.2807591 -0.09230792
##                   keyD#        keyE         keyF       keyF#         keyG
## Anime       0.252047196  0.11144692  0.166220534  0.15295876  0.161211490
## Blues      -0.591018912 -0.14100655 -0.366558701 -0.71056929 -0.067740728
## Classical   0.457609400  0.23063883  0.001924632  0.28284578  0.074157038
## Country     0.004243883  0.28208387 -0.262488516 -0.08626081  0.002552225
## Electronic  0.087774344 -0.23605963 -0.016661392  0.06656550  0.045215918
## Hip-Hop    -0.425849761 -0.31990018 -0.031820172 -0.14563409 -0.205377731
## Jazz        0.315013649  0.03418819  0.352276955 -0.05317297  0.200849382
## Rap         0.020336389 -0.23916635 -0.054171832 -0.13535078 -0.009281665
## Rock       -0.244379471  0.14346058 -0.306689316 -0.35056250 -0.189260912
##                  keyG#   liveness     loudness   modeMinor speechiness
## Anime       0.4341145 -0.8849344  0.074672030  0.22450081  -6.2744937
## Blues      -0.4257908  0.7944619 -0.119351642 -0.13633898  -5.1776704
## Classical   0.3945598  0.1653169 -0.218665839 -0.04955369  -0.6577311
## Country    -0.1473948  0.2129294 -0.028286924 -1.19389089 -13.3956330
## Electronic  0.1432608  0.6926290  0.026922616  0.46191522   0.4455714
## Hip-Hop     0.3845333  1.1808896 -0.033534014  0.35374679   7.6146828
## Jazz        0.4950303  0.0719771 -0.104576296  0.47160616   0.1507638
## Rap         0.3194984  0.9440703 -0.005632251  0.38941790   7.2169878
## Rock       -0.2634193  0.3563773 -0.077303958 -0.50178831 -11.3064443
##                    tempo   valence
## Anime       2.411271e-03  1.150237
## Blues      -2.615603e-03  3.808033
## Classical  -2.967387e-03  2.302714
## Country     5.975087e-03  1.696532
## Electronic  8.405227e-03 -2.201357
## Hip-Hop     2.818687e-03 -1.619983
## Jazz       -6.184847e-03  2.171284
## Rap         2.253831e-03 -1.902966
## Rock       -3.137353e-05  1.887560
##
## Std. Errors:
##             (Intercept)  popularity acousticness danceability instrumentalness
## Anime         0.2942366 0.003813213    0.1449513    0.2596536        0.1316190
## Blues         0.2605323 0.003275016    0.1265705    0.2307063        0.1374889
## Classical     0.3252530 0.003923535    0.1652313    0.3158020        0.1388254
## Country       0.2519980 0.002899674    0.1200351    0.2172171        0.3691710
## Electronic    0.2703443 0.003270044    0.1523961    0.2250716        0.1189565
## Hip-Hop       0.3118113 0.003484377    0.1424283    0.2250167        0.2149326
## Jazz          0.2599929 0.003104976    0.1194446    0.2154241        0.1168388
## Rap           0.3176141 0.003578601    0.1475345    0.2247977        0.2581651
## Rock          0.2772073 0.003378835    0.1321709    0.2174016        0.1577729
##                 keyA#      keyB      keyC     keyC#      keyD     keyD#
## Anime      0.1706734 0.1550651 0.1415165 0.1465681 0.1431875 0.2089715
## Blues      0.1487720 0.1319178 0.1179003 0.1317584 0.1179889 0.1922738
## Classical  0.1918653 0.1963459 0.1644903 0.1808937 0.1661796 0.2261215
## Country    0.1379398 0.1220056 0.1117180 0.1193980 0.1110234 0.1675513
## Electronic 0.1467015 0.1339106 0.1302222 0.1259527 0.1322281 0.1948638
## Hip-Hop    0.1430544 0.1327181 0.1291605 0.1223671 0.1330032 0.2121700
## Jazz       0.1362767 0.1350078 0.1214954 0.1266159 0.1261224 0.1758605
## Rap        0.1461220 0.1356923 0.1318906 0.1254191 0.1341514 0.2013610
## Rock       0.1415867 0.1250952 0.1131394 0.1228038 0.1141886 0.1820773
##                 keyE      keyF     keyF#      keyG     keyG#  liveness
```

```
## Anime      0.1561601 0.1485417 0.1648940 0.1397088 0.1636148 0.2056131
## Blues      0.1302592 0.1267203 0.1476696 0.1153081 0.1446341 0.1695672
## Classical  0.1816998 0.1724079 0.2049953 0.1638588 0.1950346 0.2367502
## Country    0.1210079 0.1226132 0.1278536 0.1090843 0.1307787 0.1749574
## Electronic 0.1417310 0.1351085 0.1404587 0.1260731 0.1473959 0.1804579
## Hip-Hop    0.1449360 0.1346785 0.1403564 0.1312157 0.1394269 0.1855825
## Jazz       0.1345735 0.1244126 0.1413469 0.1207363 0.1385785 0.1821904
## Rap        0.1473457 0.1389139 0.1439684 0.1325407 0.1436758 0.1904585
## Rock       0.1235542 0.1268547 0.1331988 0.1139923 0.1379596 0.1845092
##               loudness  modeMinor speechiness        tempo   valence
## Anime      0.011341019 0.06996096   0.5273099 0.0010837663 0.1619553
## Blues      0.009684562 0.06274035   0.4418551 0.0009619578 0.1431814
## Classical  0.010457885 0.08270099   0.5681792 0.0012287362 0.2028972
## Country    0.009887059 0.06606020   0.6306054 0.0009196003 0.1281846
## Electronic 0.010992275 0.06154893   0.3438751 0.0010391915 0.1415290
## Hip-Hop    0.011270126 0.06080174   0.2753588 0.0009882909 0.1372813
## Jazz       0.009294279 0.05849396   0.3335881 0.0009583321 0.1353293
## Rap        0.011735546 0.06159452   0.2805962 0.0009973171 0.1397939
## Rock       0.010219268 0.06093197   0.5902811 0.0009277310 0.1331937
##
## Residual Deviance: 86757.68
## AIC: 87153.68
```

```
pred <- model %>% predict(test)

# Accuracy
mean(pred == test$music_genre)
```

```
## [1] 0.5350511
```

Random Forest

```
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:dplyr':
##
##     combine
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
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:  4.18517526785533"
```

```
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
```

```
library(mltools)
```

```
##
## Attaching package: 'mltools'
```

```
## The following object is masked from 'package:tidyr':
##
##     replace_na
```

```
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"
```

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

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

boosting from adabag library

```
library(adabag)
```

```
## Loading required package: rpart
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
## Loading required package: foreach
```

```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
## Loading required package: doParallel
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
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:  20.8812239170074"
```

```
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
```

```
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"
```

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

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

XGBoost

```
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
```

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

```
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:  24.0367519855499"
```

```
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
```

```
library(mltools)

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"
```