Ensemble Techniques

Load the data

Dataset: The Spotify Hit Predictor Dataset (1960-2019) via Kaggle.

```
df_00 <- read.csv("dataset-of-00s.csv", header=TRUE)
df_10 <- read.csv("dataset-of-10s.csv", header=TRUE)
df <- rbind(df_00, df_10)

str(df)
```

```
## 'data.frame':    12270 obs. of  19 variables:
##  $ track           : chr  "Lucky Man" "On The Hotline" "Clouds Of Dementia" "Heavy Metal, Raise Hell
##  $ artist          : chr  "Montgomery Gentry" "Pretty Ricky" "Candlemass" "Zwartketterij" ...
##  $ uri             : chr  "spotify:track:4GiXBCUF7H6YfNQsnBRIzl" "spotify:track:1zyqZONW985Cs4osz9wls
##  $ danceability    : num  0.578 0.704 0.162 0.188 0.63 0.726 0.365 0.726 0.481 0.647 ...
##  $ energy          : num  0.471 0.854 0.836 0.994 0.764 0.837 0.922 0.631 0.786 0.324 ...
##  $ key             : int  4 10 9 4 2 11 1 11 10 7 ...
##  $ loudness        : num  -7.27 -5.48 -3.01 -3.75 -4.35 ...
##  $ mode            : int  1 0 1 1 1 0 1 0 1 1 ...
##  $ speechiness     : num  0.0289 0.183 0.0473 0.166 0.0275 0.0965 0.071 0.0334 0.0288 0.0377 ...
##  $ acousticness    : num  3.68e-01 1.85e-02 1.11e-04 7.39e-06 3.63e-01 3.73e-01 2.85e-03 2.20e-01 5.3
##  $ instrumentalness: num  0 0 0.00457 0.0784 0 0.268 0 0 0 0 ...
##  $ liveness        : num  0.159 0.148 0.174 0.192 0.125 0.136 0.321 0.193 0.0759 0.115 ...
##  $ valence         : num  0.532 0.688 0.3 0.333 0.631 0.969 0.29 0.746 0.389 0.344 ...
##  $ tempo           : num  133 93 87 148 112 ...
##  $ duration_ms     : int  196707 242587 338893 255667 193760 192720 89427 239240 253640 314286 ...
##  $ time_signature  : int  4 4 4 4 4 4 4 4 4 3 ...
##  $ chorus_hit      : num  30.9 41.5 65.3 58.6 22.6 ...
##  $ sections        : int  13 10 13 9 10 10 4 10 11 16 ...
##  $ target          : int  1 1 0 0 1 0 0 1 1 0 ...
```

```
df <- df[-c(1:3)]                                    # don't need the first three columns, so we can drop th
```

Split Train and Test

```
set.seed(1234)
i <- sample(1:nrow(df), 0.75*nrow(df), replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Decision Tree (Baseline)

```
library(tree)
df$key <- as.factor(df$key)
df$target <- as.factor(df$target)

start_time_tree <- Sys.time()
d_tree <- tree(target~., data=train)
end_time_tree <- Sys.time()
summary(d_tree)
```

```
##
## Regression tree:
## tree(formula = target ~ ., data = train)
## Variables actually used in tree construction:
## [1] "instrumentalness" "danceability"     "energy"           "acousticness"
## Number of terminal nodes:  8
## Residual mean deviance:  0.1413 = 1299 / 9194
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.81390 -0.09933 -0.02281  0.00000  0.18610  0.97720
```

```
run_time_tree <- end_time_tree - start_time_tree
print(paste("run time = ", run_time_tree))
```

```
## [1] "run time =  0.211658000946045"
```

Now we will evaluate the decision tree on test data.

```
probs <- predict(d_tree, newdata=test)
pred <- ifelse(probs>0.5, 1, 0)
print(paste("cor= ", cor(probs, test$target)))
```

```
## [1] "cor=  0.641852267784561"
```

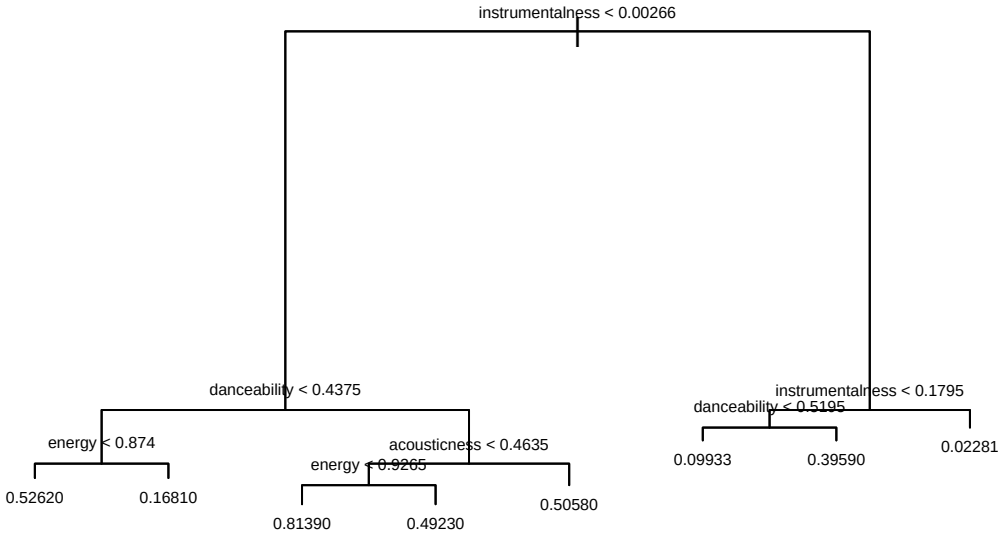
```
rmse_tree <- sqrt(mean((probs-test$target)^2))
print(paste("rmse= ", rmse_tree))
```

```
## [1] "rmse=  0.383483096254877"
```

```
acc_tree <- mean(pred==test$target)
print(paste("acc= ", acc_tree))
```

```
## [1] "acc=  0.792698826597132"
```

```
plot(d_tree)
text(d_tree, cex=0.5, pretty=0)
```



Train Random Forest

```
library(randomForest)
```

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

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

```
df$key <- as.factor(df$key)
df$target <- as.factor(df$target)

set.seed(1234)
start_time_rf <- Sys.time()
rf <- randomForest(target~danceability+energy+key+loudness+speechiness+acousticness+instrumentalness+li
```

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

```
end_time_rf <- Sys.time()
rf
```

```
##
## Call:
##  randomForest(formula = target ~ danceability + energy + key +      loudness + speechiness + acoustic
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 0.1141937
##                     % Var explained: 54.32
```

```
run_time_rf <- end_time_rf - start_time_rf
print(paste("run time = ", run_time_rf))
```

```
## [1] "run time =  3.07990119854609"
```

Predict Random Forest

```
library(mltools)
probs <- predict(rf, newdata=test, type="response")
pred <- ifelse(probs>0.5, 1, 0)
print(paste("cor= ", cor(probs, test$target)))
```

```
## [1] "cor=  0.748996378654668"
```

```
rmse_xgb <- sqrt(mean((probs-test$target)^2))
print(paste("rmse= ", rmse_xgb))
```

```
## [1] "rmse=  0.332299085039935"
```

```
acc_rf <- mean(pred==test$target)
#mcc_rf <- mcc(factor(pred), test$target)
paste("acc= ", acc_rf)
```

```
## [1] "acc=  0.855280312907432"
```

```
#paste("MCC: ", mcc_rf)
```

Train XGBoost

```
library(xgboost)
train_label <- ifelse(train$target==1, 1, 0)
train_matrix <- data.matrix(train[, -16])
start_time_xgb <- Sys.time()
model <- xgboost(data=train_matrix, label=train_label,
                 nrounds=100, objective='binary:logistic')
```

```
## [1]  train-logloss:0.561928
## [2]  train-logloss:0.486778
## [3]  train-logloss:0.438784
## [4]  train-logloss:0.404848
## [5]  train-logloss:0.379529
## [6]  train-logloss:0.362276
## [7]  train-logloss:0.344573
## [8]  train-logloss:0.331236
## [9]  train-logloss:0.322536
## [10] train-logloss:0.313637
## [11] train-logloss:0.306621
## [12] train-logloss:0.299427
## [13] train-logloss:0.294783
## [14] train-logloss:0.291359
## [15] train-logloss:0.287835
## [16] train-logloss:0.281583
## [17] train-logloss:0.278820
## [18] train-logloss:0.274677
## [19] train-logloss:0.266587
## [20] train-logloss:0.264546
## [21] train-logloss:0.263056
## [22] train-logloss:0.261287
## [23] train-logloss:0.259807
## [24] train-logloss:0.258036
## [25] train-logloss:0.255691
## [26] train-logloss:0.250689
## [27] train-logloss:0.248788
## [28] train-logloss:0.245167
## [29] train-logloss:0.242622
## [30] train-logloss:0.240600
## [31] train-logloss:0.240043
## [32] train-logloss:0.239466
## [33] train-logloss:0.235221
## [34] train-logloss:0.230120
## [35] train-logloss:0.228845
## [36] train-logloss:0.228252
## [37] train-logloss:0.227233
## [38] train-logloss:0.225181
## [39] train-logloss:0.222859
## [40] train-logloss:0.221549
## [41] train-logloss:0.220670
## [42] train-logloss:0.217131
## [43] train-logloss:0.215159
## [44] train-logloss:0.211563
## [45] train-logloss:0.210389
## [46] train-logloss:0.208893
## [47] train-logloss:0.207533
## [48] train-logloss:0.206459
## [49] train-logloss:0.203614
## [50] train-logloss:0.202799
## [51] train-logloss:0.201041
## [52] train-logloss:0.199515
## [53] train-logloss:0.198808
## [54] train-logloss:0.196607
```

```
## [55] train-logloss:0.195952
## [56] train-logloss:0.194998
## [57] train-logloss:0.193768
## [58] train-logloss:0.190107
## [59] train-logloss:0.189199
## [60] train-logloss:0.185512
## [61] train-logloss:0.182205
## [62] train-logloss:0.180905
## [63] train-logloss:0.178496
## [64] train-logloss:0.176008
## [65] train-logloss:0.174476
## [66] train-logloss:0.171087
## [67] train-logloss:0.170261
## [68] train-logloss:0.168404
## [69] train-logloss:0.165908
## [70] train-logloss:0.164692
## [71] train-logloss:0.163314
## [72] train-logloss:0.162533
## [73] train-logloss:0.159924
## [74] train-logloss:0.158536
## [75] train-logloss:0.157222
## [76] train-logloss:0.156149
## [77] train-logloss:0.155329
## [78] train-logloss:0.153840
## [79] train-logloss:0.151146
## [80] train-logloss:0.149026
## [81] train-logloss:0.146975
## [82] train-logloss:0.143519
## [83] train-logloss:0.140159
## [84] train-logloss:0.137972
## [85] train-logloss:0.136018
## [86] train-logloss:0.135347
## [87] train-logloss:0.134256
## [88] train-logloss:0.133908
## [89] train-logloss:0.133057
## [90] train-logloss:0.130309
## [91] train-logloss:0.129217
## [92] train-logloss:0.128594
## [93] train-logloss:0.128020
## [94] train-logloss:0.127336
## [95] train-logloss:0.125609
## [96] train-logloss:0.124588
## [97] train-logloss:0.121900
## [98] train-logloss:0.121117
## [99] train-logloss:0.119753
## [100]    train-logloss:0.118103
```

```
end_time_xgb <- Sys.time()
run_time_xgb <- end_time_xgb - start_time_xgb
print(paste("run time = ", run_time_xgb))
```

```
## [1] "run time =  2.97735905647278"
```

Test XGBoost

```
test_label <- ifelse(test$target==1, 1, 0)
test_matrix <- data.matrix(test[, -16])
probs <- predict(model, test_matrix)
pred <- ifelse(probs>0.5, 1, 0)
print(paste("cor= ", cor(probs, test$target)))
```

```
## [1] "cor=  0.741474120261494"
```

```
rmse_xgb <- sqrt(mean((probs-test$target)^2))
print(paste("rmse= ", rmse_xgb))
```

```
## [1] "rmse=  0.336909993604144"
```

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

```
## [1] "acc= 0.847457627118644"
```

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

```
## [1] "mcc= 0.696437159701238"
```

Train with boosting from adabag library

```
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
```

```
#train$target <- factor(train$target, levels = c(0,1)) #encode target as a factor
#start_time_adab <- Sys.time()
#adab <- boosting(target~., data=train, boos=TRUE, mfinal=20, coeflearn='Breiman')
#end_time_adab <- Sys.time()
#summary(adab)
#run_time_adab <- end_time_adab - start_time_adab
#print(paste("run time = ", run_time_adab))
```

Test with boosting from adabag library

```
#test$target <- factor(test$target, levels = c(0,1)) #encode target as a factor
#probs <- predict(adab, newdata=test, type="response")
#probs_num <- as.numeric(unlist(probs))
#pred <- ifelse(probs_num>0.5, 1, 0)
#print(paste("cor= ", cor(probs, test$target)))
#rmse_adab <- sqrt(mean((probs-test$target)^2))
#print(paste("rmse= ", rmse_adab))
#acc_adab <- mean(pred$class==test$Result)
#mcc_adabag <- mcc(factor(pred$class), test$Result)
#print(paste("acc=", acc_adab))
#print(paste("mcc=", mcc_adabag))
```