R Notebook

Code

Hide

```
df <- read.csv("picks.csv")
str(df)
```

```
'data.frame':	16035 obs. of  17 variables:
 $ date          : chr  "2020-03-18" "2020-03-18" "2020-03-18" "2020-03-17" ...
 $ team_1        : chr  "TeamOne" "Rugratz" "New England Whalers" "Complexity" ...
 $ team_2        : chr  "Recon 5" "Bad News Bears" "Station7" "forZe" ...
 $ inverted_teams: int  1 0 0 1 0 1 0 0 1 1 ...
 $ match_id      : int  2340454 2340453 2340461 2340279 2340456 2340397 2340130 2340131 2340443 
2340444 ...
 $ event_id      : int  5151 5151 5243 5226 5247 5226 5243 5243 5236 5236 ...
 $ best_of       : chr  "3" "3" "1" "3" ...
 $ system        : int  123412 123412 121212 123412 123412 123412 121212 121212 123412 123412 
...
 $ t1_removed_1  : chr  "Vertigo" "Dust2" "Mirage" "Inferno" ...
 $ t1_removed_2  : chr  "Train" "Nuke" "Dust2" "Nuke" ...
 $ t1_removed_3  : chr  "0.0" "0.0" "Vertigo" "0.0" ...
 $ t2_removed_1  : chr  "Nuke" "Mirage" "Nuke" "Overpass" ...
 $ t2_removed_2  : chr  "Overpass" "Train" "Train" "Vertigo" ...
 $ t2_removed_3  : chr  "0.0" "0.0" "Overpass" "0.0" ...
 $ t1_picked_1   : chr  "Dust2" "Vertigo" "0.0" "Dust2" ...
 $ t2_picked_1   : chr  "Inferno" "Inferno" "0.0" "Train" ...
 $ left_over     : chr  "Mirage" "Overpass" "Inferno" "Mirage" ...
```

Hide

```
df <- df[,c(7,9,10,12,13)] #grab best_of, t1_removed_1, t1_removed_2, t2_removed_1, t2_removed_
2, left_over



df$best_of <- factor(df$best_of)
df$t1_removed_1 <- factor(df$t1_removed_1)
df$t1_removed_2 <- factor(df$t1_removed_2)
df$t2_removed_1 <- factor(df$t2_removed_1)
df$t2_removed_2 <- factor(df$t2_removed_2)

levels(df$best_of)[levels(df$best_of)=="1(Online)"] <- "1"
levels(df$best_of)[levels(df$best_of)=="2(Online)"] <- "2"
levels(df$best_of)[levels(df$best_of)=="3(LAN)"] <- "3"
levels(df$best_of)[levels(df$best_of)=="3(Online)"] <- "3"
levels(df$best_of)[levels(df$best_of)=="3."] <- "3"
levels(df$best_of)[levels(df$best_of)=="of"] <- "3"

sapply(df, function(x) sum(is.na(x)==TRUE))
```

```
     best_of t1_removed_1 t1_removed_2 t2_removed_1 t2_removed_2
           0            0            0            0            0
```

Hide

```
set.seed(1234)
i <- sample(1:nrow(df), .8*nrow(df), replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Hide

```
summary(train$best_of)
```

```
   1    2    3
4539  171 8118
```

Decision tree baseline

Hide

```
library(tree)
library(mltools)
tree1 <- tree(best_of~., data=train)
pred <- predict(tree1, newdata=test, type="class")
table(pred, test$best_of)
```

```

pred    1    2    3
   1    0    0    0
   2    0    0    0
   3 1119   54 2034
```

Hide

```
acc_base <- mean(pred==test$best_of)
mcc_base <- mcc(pred, test$best_of)
```

Random Forest

Hide

```
library(randomForest)
set.seed(1234)
rf <- randomForest(best_of~., data=train, importance=TRUE)
rf
```

```
Call:
 randomForest(formula = best_of ~ ., data = train, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 36.92%
Confusion matrix:
     1 2    3 class.error
1 1494 1 3044   0.6708526
2   49 0  122   1.0000000
3 1518 2 6598   0.1872382
```

Hide

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

adabag boosting

Hide

```
library(adabag)
adab1 <- boosting(best_of~., data=train, boos=TRUE, mfinal=20, coeflearn='Breiman')
summary(adab1)
```

```
           Length Class   Mode
formula        3  formula call
trees         20  -none-  list
weights       20  -none-  numeric
votes      38484  -none-  numeric
prob       38484  -none-  numeric
class      12828  -none-  character
importance     4  -none-  numeric
terms          3  terms   call
call           6  -none-  call
```

Hide

```
pred <- predict(adab1, newdata=test, type="response")
acc_adabag <- mean(pred$class==test$best_of)
mcc_adabag <- mcc(factor(pred$class), test$best_of)
```

XGBoost

For XGBoost, we'll need to clean up the best_of so that it's binary => we're going to go ahead and combine any best_of = 2 into best_of = 1 since bo2 is most similar to bo1; this is why we put XGBoost last

Hide

```
library(xgboost)
levels(df$best_of)[levels(df$best_of)=="2"] <- "1"
train_label <- ifelse(train$best_of=="1", 1, 0)
train_matrix <- data.matrix(train[,c(2,3,4,5)])
model <- xgboost(data=train_matrix, label=train_label,
                 nrounds=100, objective='binary:logistic')
```

```
[1] train-logloss:0.662679
[2] train-logloss:0.644108
[3] train-logloss:0.634438
[4] train-logloss:0.626266
[5] train-logloss:0.621254
[6] train-logloss:0.615564
[7] train-logloss:0.611387
[8] train-logloss:0.609864
[9] train-logloss:0.606123
[10]    train-logloss:0.603464
[11]    train-logloss:0.602154
[12]    train-logloss:0.601091
[13]    train-logloss:0.599223
[14]    train-logloss:0.598494
[15]    train-logloss:0.597655
[16]    train-logloss:0.596962
[17]    train-logloss:0.595265
[18]    train-logloss:0.593905
[19]    train-logloss:0.593486
[20]    train-logloss:0.592867
[21]    train-logloss:0.591481
[22]    train-logloss:0.590993
[23]    train-logloss:0.590634
[24]    train-logloss:0.589787
[25]    train-logloss:0.588578
[26]    train-logloss:0.588197
[27]    train-logloss:0.587182
[28]    train-logloss:0.586309
[29]    train-logloss:0.585105
[30]    train-logloss:0.584780
[31]    train-logloss:0.583765
[32]    train-logloss:0.583087
[33]    train-logloss:0.582175
[34]    train-logloss:0.581478
[35]    train-logloss:0.580945
[36]    train-logloss:0.580432
[37]    train-logloss:0.579897
[38]    train-logloss:0.579511
[39]    train-logloss:0.579005
[40]    train-logloss:0.578645
[41]    train-logloss:0.578468
[42]    train-logloss:0.577972
[43]    train-logloss:0.577300
[44]    train-logloss:0.576947
[45]    train-logloss:0.576625
[46]    train-logloss:0.576268
[47]    train-logloss:0.575810
[48]    train-logloss:0.575501
[49]    train-logloss:0.575273
[50]    train-logloss:0.574593
[51]    train-logloss:0.574050
[52]    train-logloss:0.573655
```

```
[53]	train-logloss:0.573546
[54]	train-logloss:0.573173
[55]	train-logloss:0.572606
[56]	train-logloss:0.572059
[57]	train-logloss:0.571575
[58]	train-logloss:0.571119
[59]	train-logloss:0.570748
[60]	train-logloss:0.570342
[61]	train-logloss:0.569917
[62]	train-logloss:0.569557
[63]	train-logloss:0.569190
[64]	train-logloss:0.568827
[65]	train-logloss:0.568463
[66]	train-logloss:0.568146
[67]	train-logloss:0.567876
[68]	train-logloss:0.567380
[69]	train-logloss:0.567030
[70]	train-logloss:0.566762
[71]	train-logloss:0.566420
[72]	train-logloss:0.566025
[73]	train-logloss:0.565692
[74]	train-logloss:0.565496
[75]	train-logloss:0.565201
[76]	train-logloss:0.564926
[77]	train-logloss:0.564631
[78]	train-logloss:0.564367
[79]	train-logloss:0.564151
[80]	train-logloss:0.563864
[81]	train-logloss:0.563649
[82]	train-logloss:0.563355
[83]	train-logloss:0.563097
[84]	train-logloss:0.562887
[85]	train-logloss:0.562597
[86]	train-logloss:0.562427
[87]	train-logloss:0.562079
[88]	train-logloss:0.561909
[89]	train-logloss:0.561714
[90]	train-logloss:0.561475
[91]	train-logloss:0.561177
[92]	train-logloss:0.561053
[93]	train-logloss:0.560885
[94]	train-logloss:0.560750
[95]	train-logloss:0.560608
[96]	train-logloss:0.560480
[97]	train-logloss:0.560235
[98]	train-logloss:0.560039
[99]	train-logloss:0.559822
[100]	train-logloss:0.559593
```

Hide

```
test_label <- ifelse(test$best_of==1, 1, 0)
test_matrix <- data.matrix(test[,c(2,3,4,5)])
probs <- predict(model, test_matrix)
pred <- ifelse(probs>0.5, 1, 0)
acc_xg <- mean(pred==test_label)
mcc_xg <- mcc(pred, test_label)
```

Accuracies and MCC

Hide

```
print(paste("Decision Tree Accuracy:",acc_base))
```

```
[1] "Decision Tree Accuracy: 0.634237605238541"
```

Hide

```
print(paste("Decision Tree MCC:",mcc_base))
```

```
[1] "Decision Tree MCC: 0"
```

Hide

```
print(paste("Random Forest Accuracy:",acc_rf))
```

```
[1] "Random Forest Accuracy: 0.618334892422825"
```

Hide

```
print(paste("Random Forest MCC:",mcc_rf))
```

```
[1] "Random Forest MCC: 0.12512180829025"
```

Hide

```
print(paste("XGBoost Accuracy:",acc_xg))
```

```
[1] "XGBoost Accuracy: 0.640162145307141"
```

Hide

```
print(paste("XGBoost MCC:",mcc_xg))
```

```
[1] "XGBoost MCC: 0.129107566114014"
```

Hide

```
print(paste("adabag boost Accuracy:",acc_adabag))
```

```
[1] "adabag boost Accuracy: 0.658871219207983"
```

Hide

```
print(paste("adabag boost MCC:",mcc_adabag))
```

```
[1] "adabag boost MCC: 0.201557744569221"
```

Analysis

Most of the accuracies and MCC are fairly close together, with the adabag boost being the most accurate. This also aligns with the SVM we did for classification on this same dataset - the method of classification we used didn't drastically affect the results of being able to link best_of series with the other columns. In terms of runtime, they were all fairly quick, with only RF taking a bit longer to process.