Homework 7

4375 Machine Learning with Dr. Mazidi

Anthony Martinez | amm180005

10/31/21

I decided to use my classification data set from the R Project. The dataset includes a variatey of team statistics from the 2004-2015 NBA seasons. The goal is to predicted the winner of NBA games using the team's field goal percentage for 2 and 3 pointers. These metrics are an indication of how efficient a team is at scoring points.

Load/Preprocess the data set.

Load data

```
# load data
df0<-read.csv("data/games.csv", header=TRUE)
```

Data Cleaning

```
#count NAs
sapply(df0, function(x) sum(is.na(x)))
```

```
##    GAME_DATE_EST          GAME_ID GAME_STATUS_TEXT     HOME_TEAM_ID
##                0                0                 0                0
##  VISITOR_TEAM_ID           SEASON     TEAM_ID_home         PTS_home
##                0                0                 0               99
##      FG_PCT_home      FT_PCT_home     FG3_PCT_home         AST_home
##               99               99                99               99
##         REB_home     TEAM_ID_away         PTS_away      FG_PCT_away
##               99                0                99               99
##      FT_PCT_away     FG3_PCT_away         AST_away         REB_away
##               99               99                99               99
##   HOME_TEAM_WINS
##                0
```

```
# remove columns that are unnecessary
df <- df0[-c(2,3,4,5,7,14)]

# removing those columns coincidentally made it easier to replace NAs in the columns tha
t contained NAs
df$PTS_home[is.na(df$PTS_home)] <- mean(df$PTS_home, na.rm=TRUE)
df$FG_PCT_home[is.na(df$FG_PCT_home)] <- mean(df$FG_PCT_home, na.rm=TRUE)
df$FT_PCT_home[is.na(df$FT_PCT_home)] <- mean(df$FT_PCT_home, na.rm=TRUE)
df$FG3_PCT_home[is.na(df$FG3_PCT_home)] <- mean(df$FG3_PCT_home, na.rm=TRUE)
df$AST_home[is.na(df$AST_home)] <- mean(df$AST_home, na.rm=TRUE)
df$REB_home[is.na(df$REB_home)] <- mean(df$REB_home, na.rm=TRUE)
df$PTS_away[is.na(df$PTS_away)] <- mean(df$PTS_away, na.rm=TRUE)
df$FG_PCT_away[is.na(df$FG_PCT_away)] <- mean(df$FG_PCT_away, na.rm=TRUE)
df$FT_PCT_away[is.na(df$FT_PCT_away)] <- mean(df$FT_PCT_away, na.rm=TRUE)
df$FG3_PCT_away[is.na(df$FG3_PCT_away)] <- mean(df$FG3_PCT_away, na.rm=TRUE)
df$AST_away[is.na(df$AST_away)] <- mean(df$AST_away, na.rm=TRUE)
df$REB_away[is.na(df$REB_away)] <- mean(df$REB_away, na.rm=TRUE)

#show na's are deleted
sapply(df, function(x) sum(is.na(x)))
```

```
##  GAME_DATE_EST         SEASON       PTS_home    FG_PCT_home    FT_PCT_home
##              0              0              0              0              0
##   FG3_PCT_home       AST_home       REB_home       PTS_away    FG_PCT_away
##              0              0              0              0              0
##    FT_PCT_away   FG3_PCT_away       AST_away       REB_away HOME_TEAM_WINS
##              0              0              0              0              0
```

Split Data Into Test Train

```
set.seed(1234)
i <- sample(1:nrow(df), .75*nrow(df), replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Random Forest

```
library(randomForest)
```

```
## randomForest 4.6-14
```

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

```
set.seed(1234)

train$HOME_TEAM_WINS <- as.factor(train$HOME_TEAM_WINS)
test$HOME_TEAM_WINS <- as.factor(test$HOME_TEAM_WINS)
rf <- randomForest(HOME_TEAM_WINS~FG_PCT_home+FG_PCT_away+FG3_PCT_home+FG3_PCT_away, dat
a=train, importance=TRUE)
```

Random Forest Preformance Metrics

```
library(mltools)
rf
```

```
##
## Call:
##  randomForest(formula = HOME_TEAM_WINS ~ FG_PCT_home + FG_PCT_away +      FG3_PCT_hom
e + FG3_PCT_away, 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: 20.5%
## Confusion matrix:
##      0    1 class.error
## 0 5553 2005   0.2652818
## 1 1789 9160   0.1633939
```

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

```
## [1] "accuracy= 0.807617504051864"
```

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

```
## [1] "mcc= 0.602852357263745"
```

Bagging

```
#If mtry is set to the number of predictors, then bagging is performed instead of the Ra
ndom Forest.

bagging <- randomForest(HOME_TEAM_WINS~FG_PCT_home+FG_PCT_away+FG3_PCT_home+FG3_PCT_awa
y, data=train, mtry=4,importance=TRUE)
```

Bagging Preformance Metrics

```
pred_bag <- predict(rf, newdata=test, type="response")
acc_bag <- mean(pred_bag==test$HOME_TEAM_WINS)
mcc_bag <- mcc(factor(pred_bag), test$HOME_TEAM_WINS)
print(paste("accuracy=", acc_bag))
```

```
## [1] "accuracy= 0.807293354943274"
```

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

```
## [1] "mcc= 0.602211118317174"
```

AdaBoost

```
library(adabag)
```

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

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

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

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'ggplot2'
```

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

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

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

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

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

```
adab1 <- boosting(HOME_TEAM_WINS~FG_PCT_home+FG_PCT_away+FG3_PCT_home+FG3_PCT_away, 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       37014  -none-  numeric
## prob        37014  -none-  numeric
## class       18507  -none-  character
## importance      4  -none-  numeric
## terms           3  terms   call
## call            6  -none-  call
```

AbaBoost Preformance Metrics

```
# your code here
pred_boost <- predict(adab1, newdata=test, type="response")
acc_boost <- mean(pred_boost$class==test$HOME_TEAM_WINS)
mcc_boost <- mcc(factor(pred_boost$class), test$HOME_TEAM_WINS)
print(paste("accuracy=", acc_boost))
```

```
## [1] "accuracy= 0.811993517017828"
```

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

```
## [1] "mcc= 0.612174868122985"
```

XGBoost

```
library(xgboost)
train_label <- ifelse(train$HOME_TEAM_WINS==1, 1, 0)
train_matrix <- data.matrix(train[, -15])
model <- xgboost(data=train_matrix, label=train_label,
                 nrounds=100, objective='binary:logistic')
```

```
## [14:01:12] WARNING: amalgamation/../src/learner.cc:1095: Starting in XGBoost 1.3.0, t
he default evaluation metric used with the objective 'binary:logistic' was changed from
'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavi
or.
## [1]  train-logloss:0.462030
## [2]  train-logloss:0.328934
## [3]  train-logloss:0.245032
## [4]  train-logloss:0.186771
## [5]  train-logloss:0.143605
## [6]  train-logloss:0.109405
## [7]  train-logloss:0.084041
## [8]  train-logloss:0.066109
## [9]  train-logloss:0.052560
## [10] train-logloss:0.043064
## [11] train-logloss:0.035222
## [12] train-logloss:0.030815
## [13] train-logloss:0.025805
## [14] train-logloss:0.023234
## [15] train-logloss:0.020539
## [16] train-logloss:0.017819
## [17] train-logloss:0.016504
## [18] train-logloss:0.014654
## [19] train-logloss:0.013696
## [20] train-logloss:0.012628
## [21] train-logloss:0.011662
## [22] train-logloss:0.010469
## [23] train-logloss:0.010119
## [24] train-logloss:0.009594
## [25] train-logloss:0.008744
## [26] train-logloss:0.008362
## [27] train-logloss:0.008046
## [28] train-logloss:0.007505
## [29] train-logloss:0.007128
## [30] train-logloss:0.006882
## [31] train-logloss:0.006455
## [32] train-logloss:0.006210
## [33] train-logloss:0.005856
## [34] train-logloss:0.005712
## [35] train-logloss:0.005507
## [36] train-logloss:0.005397
## [37] train-logloss:0.005143
## [38] train-logloss:0.004930
## [39] train-logloss:0.004706
## [40] train-logloss:0.004527
## [41] train-logloss:0.004301
## [42] train-logloss:0.004181
## [43] train-logloss:0.003997
## [44] train-logloss:0.003933
## [45] train-logloss:0.003841
## [46] train-logloss:0.003687
## [47] train-logloss:0.003556
## [48] train-logloss:0.003441
```

```
## [49] train-logloss:0.003314
## [50] train-logloss:0.003196
## [51] train-logloss:0.003147
## [52] train-logloss:0.003091
## [53] train-logloss:0.002983
## [54] train-logloss:0.002934
## [55] train-logloss:0.002838
## [56] train-logloss:0.002752
## [57] train-logloss:0.002680
## [58] train-logloss:0.002600
## [59] train-logloss:0.002567
## [60] train-logloss:0.002539
## [61] train-logloss:0.002471
## [62] train-logloss:0.002393
## [63] train-logloss:0.002368
## [64] train-logloss:0.002318
## [65] train-logloss:0.002286
## [66] train-logloss:0.002235
## [67] train-logloss:0.002193
## [68] train-logloss:0.002170
## [69] train-logloss:0.002146
## [70] train-logloss:0.002128
## [71] train-logloss:0.002095
## [72] train-logloss:0.002032
## [73] train-logloss:0.002011
## [74] train-logloss:0.001967
## [75] train-logloss:0.001920
## [76] train-logloss:0.001873
## [77] train-logloss:0.001823
## [78] train-logloss:0.001790
## [79] train-logloss:0.001770
## [80] train-logloss:0.001730
## [81] train-logloss:0.001692
## [82] train-logloss:0.001672
## [83] train-logloss:0.001634
## [84] train-logloss:0.001614
## [85] train-logloss:0.001591
## [86] train-logloss:0.001567
## [87] train-logloss:0.001545
## [88] train-logloss:0.001527
## [89] train-logloss:0.001507
## [90] train-logloss:0.001486
## [91] train-logloss:0.001473
## [92] train-logloss:0.001440
## [93] train-logloss:0.001424
## [94] train-logloss:0.001408
## [95] train-logloss:0.001393
## [96] train-logloss:0.001379
## [97] train-logloss:0.001365
## [98] train-logloss:0.001346
## [99] train-logloss:0.001335
## [100]    train-logloss:0.001327
```

XGBoost Preformance Metrics

```
test_label <- ifelse(test$HOME_TEAM_WINS==1, 1, 0)
test_matrix <- data.matrix(test[, -15])
probs_xgb <- predict(model, test_matrix)
pred_xgb <- ifelse(probs_xgb>0.5, 1, 0)
acc_xgb <- mean(pred_xgb==test_label)
mcc_xgb <- mcc(pred_xgb, test_label)
print(paste("accuracy=", acc_xgb))
```

```
## [1] "accuracy= 0.99837925445705"
```

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

```
## [1] "mcc= 0.996671251322924"
```

Conclusion

Write a summary of your results comparing how fast or slow the algorithms were versus their accuracies

- Random Forest ran the 2nd slowest and had an accuracy of about 81%. The model had a MCC of .603
- Bagging ran the slowest out of all the models and had an accuracy of 81%. Bagging had a MCC ever so slightly less than the Random Forest model at .602
- The Boosting model ran slightly faster than the Bagging model and about as fast as the Random Forest model. The Boosting model had a accuracy higher than both Random Forest and Bagging at 81.2% but had a slightly higher MCC score of .612
- XGBoost ran the fastest and had the highest accuracy of 98% but also had the highest MCC at .997
- The models on my R project were Logistic Regression, Naive Bayes and SVM. Their accuracies were 50.2%, 81% and 81.4% respectivly. The Random Forest, Bagging and Boosting ensemble models that I created here preformed about the same as my best preforming model (SVM) from the Project. However, the XGBoost model heavily outperformed all other models with an accuracy of 98%.