Homework 7

Code

This is an R script with the purpose of comparing various ensemble methods on a day-trading data set

Ramesh Kanakala

Loading and Cleaning Classification Data (https://www.kaggle.com/dawerty/cleaned-daytrading-training-data (https://www.kaggle.com/dawerty/cleaned-daytrading-training-data))

Hide

```
#load the data
st <- read.csv("stock.csv")
st <- subset(st, select = -c(sym, datetime))

#converting to binary
st$is_profit <- as.integer(as.factor(st$is_profit)) - 1
st$is_profit <- as.factor(st$is_profit)

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

Random Forest

Hide

```
library(randomForest)
memory.limit(20000)
```

```
[1] 20000
```

Hide

```
start <- Sys.time()
rf1 <- randomForest(is_profit~., data=train, importance=TRUE)
#rf1 <- randomForest(x = train[,2:21], y = train$is_profit, training_frame = train)
end <- Sys.time()
start - end
```

```
Time difference of -10.95372 mins
```

Hide

```
summary(rf1)
```

```
                Length Class  Mode
call                 4 -none- call
type                 1 -none- character
predicted       194046 factor numeric
err.rate          1500 -none- numeric
confusion            6 -none- numeric
votes           388092 matrix numeric
oob.times       194046 -none- numeric
classes              2 -none- character
importance          80 -none- numeric
importanceSD        60 -none- numeric
localImportance      0 -none- NULL
proximity            0 -none- NULL
ntree                1 -none- numeric
mtry                 1 -none- numeric
forest              14 -none- list
y               194046 factor numeric
test                 0 -none- NULL
inbag                0 -none- NULL
terms                3 terms  call
```

Hide

```
pred1 <- predict(rf1, newdata=test, type="response")
acc_rf <- mean(pred1==test$is_profit)
mcc_rf <- mcc(factor(pred1), test$is_profit)
print(paste("accuracy=", acc_rf))
```

```
[1] "accuracy= 0.702456596014409"
```

Hide

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

```
[1] "mcc= 0.34830343290132"
```

Boosting (adaboost from adabag)

Hide

```
library(adabag)
library(mltools)
start <- Sys.time()
adab1 <- boosting(is_profit~., data=train, boos=TRUE, mfinal=20, coeflearn="Breiman")
end <- Sys.time()
start - end
```

```
Time difference of -11.3259 mins
```

Hide

```
summary(adab1)
```

```
           Length Class   Mode     
formula         3 formula call     
trees          20 -none-  list     
weights        20 -none-  numeric  
votes      388092 -none-  numeric  
prob       388092 -none-  numeric  
class      194046 -none-  character
importance     20 -none-  numeric  
terms           3 terms   call     
call            6 -none-  call     
```

Hide

```
pred2 <- predict(adab1, newdata=test, type="response")
acc_adabag <- mean(pred2$class==test$is_profit)
mcc_adabag <- mcc(factor(pred2$class), test$is_profit)
print(paste("accuracy=", acc_adabag))
```

```
[1] "accuracy= 0.695144010018088"
```

Hide

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

```
[1] "mcc= 0.327761895194885"
```

AdaBoost (from fastAdaboost)

Hide

```
library(fastAdaboost)
start <- Sys.time()
fadab1 <- adaboost(is_profit~., train, 10)
end <- Sys.time()
start - end
```

```
Time difference of -4.720586 mins
```

Hide

```
summary(fadab1)
```

```
                   Length Class   Mode
formula             3     formula call
trees              10     -none-  list
weights            10     -none-  numeric
classnames          2     -none-  character
dependent_variable  1     -none-  character
call                4     -none-  call
```

Hide

```
pred3 <- predict(fadab1, newdata=test, type="response")
acc_fadab <- mean(pred3$class==test$is_profit)
mcc_fadab <- mcc(pred3$class, test$is_profit)
print(paste("accuracy=", acc_fadab))
```

```
[1] "accuracy= 0.654499636689702"
```

Hide

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

```
[1] "mcc= 0.26781341148625"
```

XGBoost

Hide

```
train_label <- ifelse(train$is_profit==1, 1, 0)
train_matrix <- data.matrix(train[, -1])
test_label <- ifelse(test$is_profit==1, 1, 0)
test_matrix <- data.matrix(test[, -1])


library(xgboost)
```

```
package 惖祂xgboost惖祚 was built under R version 4.0.5
```

Hide

```
start <- Sys.time()
xgb1 <- xgboost(data=train_matrix, label=train_label, nrounds=100, objective="binary:logistic",
 verbose  = 0)
```

```
[21:02:12] WARNING: amalgamation/../src/learner.cc:1061: Starting in XGBoost 1.3.0, the default
evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'loglos
s'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

Hide

```
end <- Sys.time()
start - end
```

```
Time difference of -6.266878 secs
```

Hide

```
probs <- predict(xgb1, test_matrix)
pred4 <- ifelse(probs>0.5, 1, 0)
acc_xg <- mean(pred4==test_label)
mcc_xg <- mcc(pred4, test_label)
print(paste("accuracy=", acc_xg))
```

```
[1] "accuracy= 0.697231111729512"
```

Hide

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

```
[1] "mcc= 0.337638934292441"
```

Original Project Models for comparison

Hide

```
#logistic regression model
glm1 <- glm(is_profit~., data=train, family=binomial)
probs <- predict(glm1, newdata=test, type="response")
glmpred <- ifelse(probs>0.5, 1, 0)
glmacc <- mean(glmpred==test$is_profit)
print(paste("acc: ", glmacc))
```

```
[1] "acc:  0.695144010018088"
```

Hide

```
#naive bayes model
library(e1071)
nb1 <- naiveBayes(is_profit~., data=train)
nbpred <- predict(nb1, newdata=test, type="class")
library(caret)
nbacc <- mean(nbpred==test$is_profit)
print(paste("acc: ", nbacc))
```

```
[1] "acc:  0.682868759952383"
```

Hide

```
#decision tree model
library(tree)
tree2 <- tree(is_profit~., data=train)
tree_pred2 <- predict(tree2, newdata=test, type="class")
treeacc <- mean(tree_pred2 == test$is_profit)
print(paste("acc: ", treeacc))
```

```
[1] "acc:  0.678416276301347"
```

Discussion

The accuracies of the original algorithms of the project, logistic regression, naive bayes, and a decision tree, were 0.695, 0.683, and 0.678 respectively. Logistic regression performed the best and the decision tree the worst. The ensemble methods used in this notebook all improved upon the highest accuracy, 0.695 by logistic regression. Here are the model ranked by accuracy: random forest (0.702), xgboost (0.697), adaboost (0.695), and fastadaboost (0.654). The mccs are also ranked the same. Random forest, though it had the highest accuracy, ran the second-longest at 10.954 minutes, adaboost was highest at 11.3259 minutes, and fastadaboost and xgboost followed both much fast at 4.721 minutes and 6.266878 seconds respectively. It's interesting how xgboost was had the second highest accuracy but ran much over a hundred times faster than the random forest. Fast adaboost was worse than adaboost in accuracy but faster in time.

Random foresting most likely took the a long time as it trains multiple trees on subsets of the data and chose the best model after trying many of them but this also probably led to it's success. Fastadaboost uses C++ code to run about 100 times fast than adaboost, though in this case it was more about 1.3x faster, and interestingly received a slightly lower accuracy. Finally, XGBoost is known for its extreme scalability and ran faster by a huge margin. This is done because the C++ computation utilizes multithreading processing. Of course, the data had to be preprocessed before, however.