Personal Activity Prediction Modeling

Chris Coultas

September 10, 2017

`{r setup, include=FALSE} knitr::opts_chunk$set(echo = TRUE)` ###Step 1: Install packages and set up training, test, and cross-validation sets

```
library(pacman)
p_load(xgboost, nlme, parallel, doParallel, MASS, Hmisc, randomForest, GGally, Applied
PredictiveModeling, caret, ElemStatLearn, pgmm, rpart, gbm, lubridate, forecast, e107
1)



#Download data
download.file(url="https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.cs
v",
                      destfile="C:/Users/Chris/Desktop/Data Science Courses/Practical
Machine Learning/pml-training.csv")
train <- read.csv("C:/Users/Chris/Desktop/Data Science Courses/Practical Machine Learn
ing/pml-training.csv")


download.file(url="https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.cs
v",
              destfile="C:/Users/Chris/Desktop/Data Science Courses/Practical Machine
Learning/pml-testing.csv")
test <- read.csv("C:/Users/Chris/Desktop/Data Science Courses/Practical Machine Learni
ng/pml-testing.csv")

#REMOVE NA Columns and PreProcess data

train <- train[, colSums(is.na(train))==0]
inTrain <- createDataPartition(train$classe, p=.75, list=FALSE)
train2 <- train[inTrain,]
trainTest <- train[-inTrain,]
```

Step 2: Prepare data for modeling with preprocessing for data reduction

```
preProc <- preProcess(train2[,-1],method=c("BoxCox", "center","scale","pca"))
trainProc <- predict(preProc,train2[,-1])
df <- trainProc[,37:63]

testProc <- predict(preProc,trainTest[,-1])
dftest <- testProc[,37:63]
```

Step 3: Begin initial predictive modeling

Using the PCA factors extracted in Step 2, let's build several models using several different methods.

```
#Predictive Modeling
set.seed(1)
ldamod <- train(classe ~ ., data=df, method="lda")
qdamod <- train(classe ~ ., data=df, method="qda")
gbmod <- train(classe ~ ., data=df, method="gbm")
xgbmod <- train(classe~ ., data=df, method="xgbTree")
rfmod <- train(classe ~ ., data=df, method="rf", ntree=300)

ldaPRED <- predict(ldamod,df)
gbPRED <- predict(gbmod,df)
rfPRED <- predict(rfmod,df)
qdaPRED <- predict(qdamod, df)
xgbPRED <- predict(xgbmod,df)
```

Now, let's check the accuracies of each in the training set.

```
#Check accuracies in training set
confusionMatrix(ldaPRED,df$classe)
confusionMatrix(gbPRED,df$classe)
confusionMatrix(rfPRED,df$classe)
confusionMatrix(qdaPRED,df$classe)
confusionMatrix(xgbPRED,df$classe)
```

It appears the random forest model is the strongest, with Extreme Gradient Boosting a close second.

Step 4: Test the models and ensemble in the test set

We'll first check the accuracy of our training models in the test set.

```
ldaTest <- predict(ldamod,dftest)
qdaTest <- predict(qdamod,dftest)
gbTest <- predict(gbmod,dftest)
rfTest <- predict(rfmod,dftest)
xgbTest <- predict(xgbmod, dftest)

confusionMatrix(ldaTest, dftest$classe)
confusionMatrix(qdaTest, dftest$classe)
confusionMatrix(gbTest, dftest$classe)
confusionMatrix(rfTest, dftest$classe)
confusionMatrix(xgbTest, dftest$classe)
```

Again, it appears that Random Forest is the strongest (97.9% accurate), followed by Extreme Gradient Boosting (93.8% accurate). Now, let's put the models together and see if accuracy improves.

```
predtestDF <- data.frame(rfTest, xgbTest, gbTest, qdaTest, ldaTest, classe=dftest$clas
se)
combTest <- train(classe ~ ., method="rf", data=predtestDF)

comPRED <- predict(combTest,dftest)

confusionMatrix(comPRED,dftest$classe)
```

It appears we get a slight bump by ensembling the models together, moving from 97.9% accurate (with Random forests alone) to 98.3% accurate (97.9% LCI, 98.6% UCI) This suggests the out-of-sample error rate will be somewhere between 1.4% and 2.1%.