Machine Learning Course Project

Anton Medvedev

January 6, 2018

Background: Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Data: The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har).

Objective: The goal of this project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. This is a report describing how the model was built, how cross validation was performed, and how the out-of-sample error rate was estimated.

```
rm(list=ls())
library(caret)
```

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

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

```
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", destfile =
 "training.csv")
InTrain = read.csv("~/training.csv")

download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", destfile =
"testing.csv")
InTest = read.csv("~/testing.csv")
```

The above code clears the global environment, loads the caret package and downloads the training and testing data to be used for this study.

```
InTrain <- InTrain[sapply(InTrain, function(x) !any(is.na(x)))]
InTrain <- InTrain[!sapply(InTrain, function(x) any(x == ""))]
InTrain <- InTrain[,-c(1:7)]

InTest <- InTest[sapply(InTest, function(x) !any(is.na(x)))]
InTest <- InTest[!sapply(InTest, function(x) any(x == ""))]
InTest <- InTest[,-c(1:7)]
```

The above chunk eliminates columns which contain fields with NAs and blanks from the sets in order to: (i)Minimize computational capacity needed to perform the analysis; and (ii)Remove any obstacles such variables introduce in running the necesarry models. The first seven columns of the data frames are also removed, this is done for the first reason above.

```
stage <- createDataPartition(InTrain$classe, p = 0.7, list = F)
training <- InTrain[stage,]
testing <- InTrain[-stage,]
```

This chunk further breaks down the training data set into a training set and a testing set. We can think of the testing set in this context as a validation set; this will be used to gauge the effectiveness of the model out-of sample and calculate an out-of-sample error rate.

```
set.seed(117)
rf <- train(classe ~ ., method = "rf", data = training)
```

```
## randomForest 4.6-12
```

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
predtest_rf <- predict(rf, newdata = testing)
confusionMatrix(testing$classe, predtest_rf)$overall[1]
```

```
##  Accuracy
## 0.9938828
```

The above chunk sets the seed, performs a random forest analysis, applies the generated algorithm to the validation dataframe, and calculates the accuracy of the generated algorithm (99.4%).

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
## Loading required package: splines
```

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

```
## Loaded gbm 2.1.3
```

```
confusionMatrix(testing$classe, predtest_gbm)$overall[1]
```

```
##  Accuracy
## 0.9641461
```

The above two chunks perform a generalized boosting analysis, apply the generated algorithm to the validation dataframe, and calculate the accuracy of the generated algorithm (96.3%).

```
df <- data.frame(predtest_rf, predtest_gbm, classe = testing$classe)
comb <- train(classe~., method = "rf", data = df)
predtest_comb <- predict(comb, data = df)
confusionMatrix(df$classe, predtest_comb)$overall[1]
```

```
##  Accuracy
## 0.9938828
```

The above block of code stacks the two predictions together using the random forest method and calculates the accuracy of this model, which turns out to be equivalent to the one generated by random forests (99.4%). We can thus estimate the out-of-sample error rate to be 0.6% utilizing cross-validation between the testing and validation sets.

```
library(rpart)

rp <- rpart(classe ~ ., data=training, method="class")
predtest_rp <- predict(rp, testing, type = "class")
cmtree <- confusionMatrix(predtest_rp, testing$classe)
cmtree$overall[1]
```

```
##  Accuracy
## 0.7291419
```

The above chunk uses a prediction tree to generate an algorithm as a possible alternaive to the random forest algorithm, but we see that the accuracy of 74.9% is significantly lower, so our earlier model is more suitable.

```
cn1 <- as.data.frame(colnames(InTrain));cn2 <- as.data.frame(colnames(InTest))
cn <- merge(cn1, cn2, by.x = 'colnames(InTrain)',by.y = 'colnames(InTest)')
```

The chunk above is meant to validate that the columns dropped from the training and testing data sets at the beginning of the study are identical, so the algorithm developed during the training phase runs on identical variables in the testing phase. The dataframe contains 52 out of 53 columns, which means one column was mis-matched (the classe column) in the testing set we are trying to predict.

```
test_rf <- as.data.frame(predict(rf, newdata = InTest))
test_rf
```

```
##    predict(rf, newdata = InTest)
## 1                              B
## 2                              A
## 3                              B
## 4                              A
## 5                              A
## 6                              E
## 7                              D
## 8                              B
## 9                              A
## 10                             A
## 11                             B
## 12                             C
## 13                             B
## 14                             A
## 15                             E
## 16                             E
## 17                             A
## 18                             B
## 19                             B
## 20                             B
```

The prediction for the testing set is stored in the test_rf dataframe.