Prediction Assignment

John Econ

May 20, 2016

1 Intro

1.1 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:

- exactly according to the specification (Class A)
- throwing the elbows to the front (Class B)
- lifting the dumbbell only halfway (Class C)
- lowering the dumbbell only halfway (Class D)
- throwing the hips to the front (Class E)

More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

1.2 Data

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv.

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv.

2 Analysis

2.1 Packages required for this analysis

```
library(caret);library(randomForest)
```

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

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

```
## 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
```

2.2 Reading the data

After downloading the training and testing datasets into a folder named data under the current working directory, we read them, treating blanks, division by zeros and NAs as NAs.

```
setwd("C:/Users/johnecon/Projects/datasciencecoursera/Practical Machine Learning/assignment/")
pml_data <- read.csv("./data/pml-training.csv", na.strings=c("", "NA","#DIV/0!"))
```

2.3 Partitioning training and testing sets

Since the pml-testing.csv has only 20 records, we split the pml-training.csv into the training and the testing datasets on the varriable of interest (classe). We will later use the pml-testing.csv as a validation dataset.

```
set.seed(34233)
inTrain <- createDataPartition(y=pml_data$classe, list=FALSE, p=0.7)
training <- pml_data[inTrain,]
testing <- pml_data[-inTrain,]
validation <- read.csv("./data/pml-testing.csv")
```

```
dim(training)
```

```
## [1] 13737   160
```

```
dim(testing)
```

```
## [1] 5885  160
```

```
dim(validation)
```

```
## [1]  20 160
```

2.4 Cleaning up variables from the training set that are mostly NAs

By reviewing the training data we can identify varriables that are mostly NAs. To clean them up we take 70% as a thresshold, meaning if a variable appears more than 70% NA then it needs to be removed.

```
training <- training[,colSums(is.na(training))/nrow(training)<0.7]
```

2.5 Cleaning up time and user related variables (columns 1 to 7)

Predicting how well one performs an activity has nothing to do with the time or the name of the person. Thus the first seven columns have to be removed.

```r
training <- training[,-c(1:7)]
```

```r
dim(training)
```

```
## [1] 13737    53
```

2.6 Building a Random Forest

Because of the characteristic noise in the sensor data, the recommended approach for the model method is the Random Forest. This algorithm is characterized by a subset of features, selected in a random and independent manner with the same distribution for each of the trees in the forest.

```r
set.seed(34233)
mod_fit_rf <- train(classe ~ ., data = training, method = "rf", ntree=10)
predRF <- predict(mod_fit_rf, testing)
confusionMatrix(predRF, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    6    1    1    2
##          B    0 1118    5    2    4
##          C    3   14 1018   12    2
##          D    0    0    2  947    5
##          E    0    1    0    2 1069
##
## Overall Statistics
##
##                Accuracy : 0.9895
##                  95% CI : (0.9865, 0.9919)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9867
##  Mcnemar's Test P-Value : 0.001505
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9816   0.9922   0.9824   0.9880
## Specificity            0.9976   0.9977   0.9936   0.9986   0.9994
## Pos Pred Value         0.9941   0.9903   0.9704   0.9927   0.9972
## Neg Pred Value         0.9993   0.9956   0.9983   0.9966   0.9973
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1900   0.1730   0.1609   0.1816
## Detection Prevalence   0.2856   0.1918   0.1782   0.1621   0.1822
## Balanced Accuracy      0.9979   0.9896   0.9929   0.9905   0.9937
```

```r
# Accuracy
confusionMatrix(predRF, testing$classe)$overall[1]
```

```
##  Accuracy
## 0.9894647
```

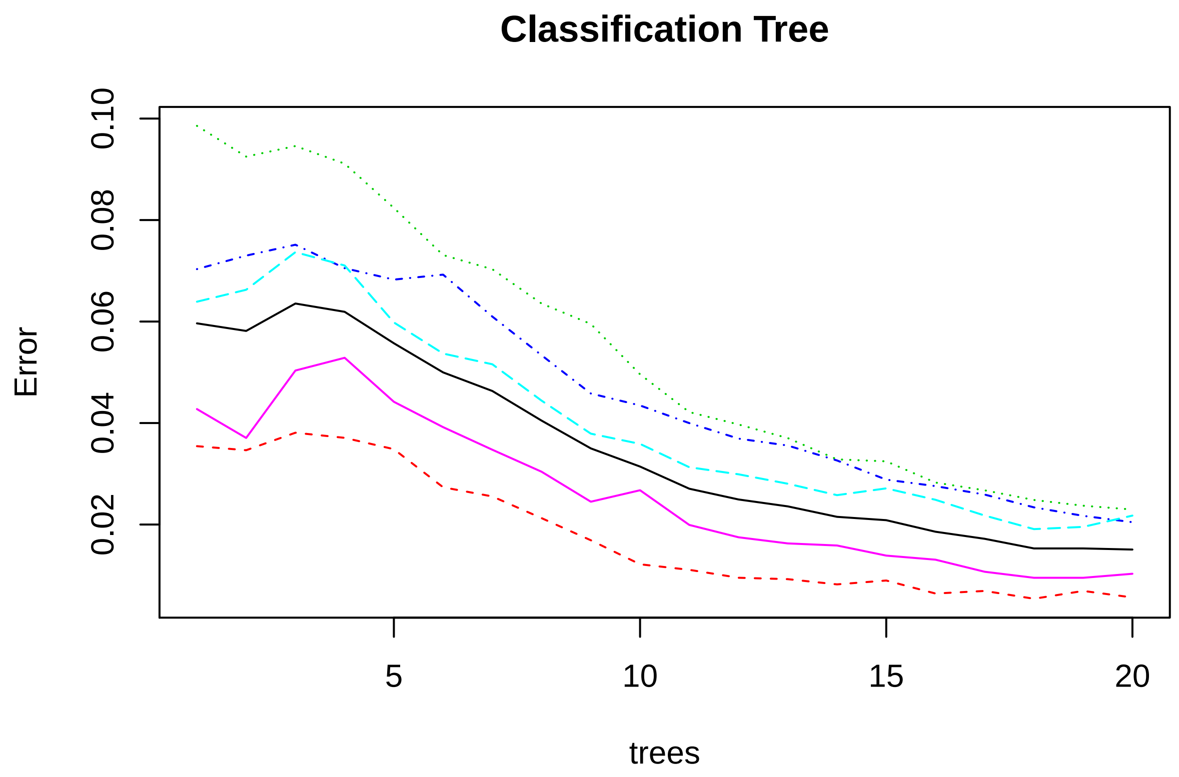
2.7 Improving the model

Increasing the number of trees to 20 we expect to improve the accuracy of the model but that is not actually the case since it leads to overfitting.

```r
set.seed(34233)
mod_fit_rf_20 <- train(classe ~ ., data = training, method = "rf", ntree=20)
pred_rf_20 <- predict(mod_fit_rf_20, testing)
confusionMatrix(pred_rf_20, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1670   10    1    0    1
##          B    1 1119    5    2    6
##          C    3    7 1014   15    2
##          D    0    1    3  945    2
##          E    0    2    3    2 1071
##
## Overall Statistics
##
##                Accuracy : 0.9888
##                  95% CI : (0.9858, 0.9913)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9858
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9824   0.9883   0.9803   0.9898
## Specificity            0.9972   0.9971   0.9944   0.9988   0.9985
## Pos Pred Value         0.9929   0.9876   0.9741   0.9937   0.9935
## Neg Pred Value         0.9990   0.9958   0.9975   0.9961   0.9977
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2838   0.1901   0.1723   0.1606   0.1820
## Detection Prevalence   0.2858   0.1925   0.1769   0.1616   0.1832
## Balanced Accuracy      0.9974   0.9897   0.9914   0.9895   0.9942
```

```r
plot(mod_fit_rf_20$finalModel, main="Classification Tree")
```



Droping downping down to 15 trees seems gives us the final model.

```
set.seed(34233)
mod_fit_rf_15 <- train(classe ~ ., data = training, method = "rf", ntree=15)
pred_rf_15 <- predict(mod_fit_rf_15, testing)
confusionMatrix(pred_rf_15, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1670    7    0    1    1
##          B    1 1119    4    2    3
##          C    2   11 1018   11    2
##          D    1    2    3  949    2
##          E    0    0    1    1 1074
##
## Overall Statistics
##
##                Accuracy : 0.9907
##                  95% CI : (0.9879, 0.993)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9882
##  Mcnemar's Test P-Value : 0.0402
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9824   0.9922   0.9844   0.9926
## Specificity            0.9979   0.9979   0.9946   0.9984   0.9996
## Pos Pred Value         0.9946   0.9911   0.9751   0.9916   0.9981
## Neg Pred Value         0.9990   0.9958   0.9983   0.9970   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2838   0.1901   0.1730   0.1613   0.1825
## Detection Prevalence   0.2853   0.1918   0.1774   0.1626   0.1828
## Balanced Accuracy      0.9977   0.9902   0.9934   0.9914   0.9961
```

3 Conclusions and Final Predictions

3.1 Conclusions

A Random Forest model with 15 trees provided accuracy 99% and thus we expect that very few of the test samples will be missclassified.

3.2 Predictions on the official test dataset

```r
set.seed(34233)
predict(mod_fit_rf_15, validation)
```

```
##  [1] B A B A A E D D A A B C B A E E A B B B
## Levels: A B C D E
```

4 Credits

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