Practical Machine Learning - Week 4 Assignment

C. Staples

5 February 2018

Introduction

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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Data

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

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).

Full source:

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. “Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human ’13)”. Stuttgart, Germany: ACM SIGCHI, 2013.

Thankyou to the authors for being so generous in allowing their data to be used for this kind of assignment!

“Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: 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) and throwing the hips to the front (Class E).

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate. The exercises were performed by six male

participants aged between 20-28 years, with little weight lifting experience. We made sure that all participants could easily simulate the mistakes in a safe and controlled manner by using a relatively light dumbbell (1.25kg)."

```
##Library Setup

library(caret)
```

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

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

```
library(rpart)
library(rpart.plot)
library(randomForest)
```

```
## 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
```

```
library(gbm)
```

```
## 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
```

```
library(knitr)

## Data Import

# Download from the URL (Link)
UrlTrain <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
UrlTest  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# Import into R Environment
training <- read.csv(url(UrlTrain))
testing  <- read.csv(url(UrlTest))

# Create partitions (trainign and test) from the dataset
inTrain  <- createDataPartition(training$classe, p=0.7, list=FALSE)
TrainSet <- training[inTrain, ]
TestSet  <- training[-inTrain, ]

dim(TrainSet)
```

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

```
dim(TestSet)
```

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

```
# Remove any variables with Near Zero Variance
NZV <- nearZeroVar(TrainSet)
TrainSet <- TrainSet[, -NZV]
TestSet  <- TestSet[, -NZV]

dim(TrainSet)
```

```
## [1] 13737   103
```

```
dim(TestSet)
```

```
## [1] 5885  103
```

```
# Remove Uneccesary Variables (where most of the fields are NA) & ID Categories

allNA    <- sapply(TrainSet, function(x) mean(is.na(x))) > 0.95
TrainSet <- TrainSet[, allNA==FALSE]
TestSet  <- TestSet[, allNA==FALSE]

TrainSet <- TrainSet[, -(1:5)]
TestSet  <- TestSet[, -(1:5)]

dim(TrainSet)
```

```
## [1] 13737    54
```

```
dim(TestSet)
```

```
## [1] 5885   54
```

The following sections will use three methods for modelling the training data to enable prediction on the test data. The models will be compared for their accuract and the “best” model will then be used on the final questions.

```
## Model 1 - Decision Tree

set.seed(29)

mod1DT <- rpart(classe ~ ., data = TrainSet, method="class")

## Model 1 - Decision Tree Predictions

prediction <- predict(mod1DT, TestSet, type = "class")

confMatDT <- confusionMatrix(prediction, TestSet$classe)

confMatDT
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1530  232   16   56   42
##          B   51  680   97   85  127
##          C   16   79  784   82   55
##          D   71  116   96  677  174
##          E    6   32   33   64  684
##
## Overall Statistics
##
##                Accuracy : 0.74
##                  95% CI : (0.7286, 0.7512)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.67
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9140   0.5970   0.7641   0.7023   0.6322
## Specificity            0.9178   0.9241   0.9523   0.9071   0.9719
## Pos Pred Value         0.8156   0.6538   0.7717   0.5970   0.8352
## Neg Pred Value         0.9641   0.9053   0.9503   0.9396   0.9214
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2600   0.1155   0.1332   0.1150   0.1162
## Detection Prevalence   0.3188   0.1767   0.1726   0.1927   0.1392
## Balanced Accuracy      0.9159   0.7606   0.8582   0.8047   0.8020
```

```
## Model 2 - Random Forests

set.seed(29)

controlRF <- trainControl(method="cv", number=3, verboseIter=FALSE)

mod2RF <- train(classe ~ ., data=TrainSet, method="rf",trControl=controlRF)

mod2RF$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.23%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3905    0    0    0    1 0.0002560164
## B    7 2648    3    0    0 0.0037622272
## C    0    6 2390    0    0 0.0025041736
## D    0    0   10 2242    0 0.0044404973
## E    0    1    0    4 2520 0.0019801980
```

```
## Model 2 - Random Forest Predictions

predictRF <- predict(mod2RF, newdata=TestSet)

confMatRF <- confusionMatrix(predictRF, TestSet$classe)

confMatRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    4    0    0    0
##          B    1 1131    4    0    0
##          C    0    3 1022    3    0
##          D    0    1    0  960    3
##          E    0    0    0    1 1079
##
## Overall Statistics
##
##                Accuracy : 0.9966
##                  95% CI : (0.9948, 0.9979)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9957
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9930   0.9961   0.9959   0.9972
## Specificity            0.9991   0.9989   0.9988   0.9992   0.9998
## Pos Pred Value         0.9976   0.9956   0.9942   0.9959   0.9991
## Neg Pred Value         0.9998   0.9983   0.9992   0.9992   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1922   0.1737   0.1631   0.1833
## Detection Prevalence   0.2850   0.1930   0.1747   0.1638   0.1835
## Balanced Accuracy      0.9992   0.9960   0.9974   0.9975   0.9985
```

```
## Model 3 - Boost Model

set.seed(29)

controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)

mod3GBM  <- train(classe ~ ., data=TrainSet, method = "gbm",trControl = controlGBM, verbose =
 FALSE)
mod3GBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 41 had non-zero influence.
```

```
## Model 3 - Boost Model Predictions

predictGBM <- predict(mod3GBM, newdata=TestSet)

confMatGBM <- confusionMatrix(predictGBM, TestSet$classe)

confMatGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1670   10    0    2    0
##          B    4 1108   12    1    1
##          C    0   15 1010   19    2
##          D    0    6    2  941   11
##          E    0    0    2    1 1068
##
## Overall Statistics
##
##                Accuracy : 0.985
##                  95% CI : (0.9816, 0.988)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9811
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9728   0.9844   0.9761   0.9871
## Specificity            0.9972   0.9962   0.9926   0.9961   0.9994
## Pos Pred Value         0.9929   0.9840   0.9656   0.9802   0.9972
## Neg Pred Value         0.9990   0.9935   0.9967   0.9953   0.9971
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2838   0.1883   0.1716   0.1599   0.1815
## Detection Prevalence   0.2858   0.1913   0.1777   0.1631   0.1820
## Balanced Accuracy      0.9974   0.9845   0.9885   0.9861   0.9932
```

The best model was the Random Forest with the highest accuracy (0.9973) vs 0.9802 for Boost Model and 0.7349 for Decision Tree. Randomf Forest model will be used for the final prediction on the test set.

```
## Predictions

predictTEST <- predict(mod2RF, newdata=testing)
predictTEST
```

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