Assignment

Maciej Gielnik

17/01/2021

Introduction

Smart devices are getting more and more popular. They can be used to monitor our activities and can help us to improve our training routine.

The main goal of this Assignment was to predict how participants performed the Unilateral Dumbbell Biceps Curl using data from sensors placed on arm, glove, belt and dumbbell (Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises).

In the data set participants performed the Unilateral Dumbbell Biceps Curl 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).

The main question is, if we use the data from sensors, can we define how the Unilateral Dumbbell Biceps Curl was performed?

Downloading and preprocessing the data

The files were downloaded using following commands:

```
if (!file.exists("pml-training.csv")){
    URLtraining <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
    download.file(URLtraining, destfile = "./pml-training.csv")
}

if (!file.exists("pml-testing.csv")){
    URLtesting <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
    download.file(URLtesting, destfile = "./pml-testing.csv")
}
```

Downloaded filed were loaded using following commands:

```
training <- read.csv("./pml-training.csv", na.strings = c("", "NA"))
testing <- read.csv("./pml-testing.csv")
```

The **training** data frame contains data for building and testing a model, where **testing** data frame contains data for prediction to answer the final Quiz questions.

The **training** data frame has a lot of missing values, therefore to build the model for prediction, columns that have more than 50% NA were removed using the following script:

```
{
j <- h <- training1 <- trainingtemp <- 0 #initial parameters

dimmension <- (matrix(dim(training))[1,1])
```

```
columnNames <- names(training)

for (i in 1:160){

    addition <- is.na(training[, i])
    k <- (sum(addition)) #adding the number of NA`s in a i`th column

        if (k < dimmension/2){ #only columns with less than 50% NA`s goes here

                    if (j == 0) { #first column
                        wybrana <- columnNames[i]
                        training1 <- subset(training, select = c(wybrana))
                    }

                    if (j > 0){  #binding rest of columns to the first column
                        wybrana <- columnNames[i]
                        trainingtemp <- subset(training, select = c(wybrana))
                        training1 <-cbind(training1, trainingtemp)
                    }
                   j <- 1
        }
    }
}
```

The first seven columns were also removed as the contain data with username, timestamps, etc.

```
trainfin <- training1[, 8:60]
```

Building the trainig and testing set.

To build training set and test set I have loaded the caret package. I have also loaded the data.table package because it allows multithreading.

```
library(data.table)
library(caret)
```

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

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

Creating training and testing data set. I have also set the seed to make operation reproducable.

```
RNGversion("3.0.0.")
```

```
## Warning in RNGkind("Mersenne-Twister", "Inversion", "Rounding"): non-uniform
## 'Rounding' sampler used
```

```
set.seed(123)
inTrain <- createDataPartition(trainfin$classe, p = 0.7, list = FALSE)
dataTraining <- trainfin[inTrain,]
dataTesting <- trainfin[-inTrain,]
```

Predicting with trees

```
modelRpart <- train(classe~., data = dataTraining, method = "rpart")
confusionMatrix(dataTesting$classe, predict(modelRpart,dataTesting))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1061  163  341  102    7
##          B  235  631  230   43    0
##          C   27   42  819  138    0
##          D   64  133  509  258    0
##          E   13  281  247   60  481
##
## Overall Statistics
##
##                Accuracy : 0.5523
##                  95% CI : (0.5394, 0.565)
##     No Information Rate : 0.3647
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4373
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7579   0.5048   0.3816  0.42928  0.98566
## Specificity            0.8633   0.8904   0.9446  0.86639  0.88864
## Pos Pred Value         0.6338   0.5540   0.7982  0.26763  0.44455
## Neg Pred Value         0.9195   0.8696   0.7269  0.93030  0.99854
## Prevalence             0.2379   0.2124   0.3647  0.10212  0.08292
## Detection Rate         0.1803   0.1072   0.1392  0.04384  0.08173
## Detection Prevalence   0.2845   0.1935   0.1743  0.16381  0.18386
## Balanced Accuracy      0.8106   0.6976   0.6631  0.64784  0.93715
```

Predicting with random forest

```
modelrf <- train(classe~., data = dataTraining, method = "rf")
confusionMatrix(dataTesting$classe, predict(modelrf,dataTesting))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    2    0    0    0
##          B    8 1128    2    1    0
##          C    0    9 1014    3    0
##          D    0    0   16  947    1
##          E    0    0    2    1 1079
```

```
##
## Overall Statistics
##
##                Accuracy : 0.9924
##                  95% CI : (0.9898, 0.9944)
##     No Information Rate : 0.2855
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9903
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9952   0.9903   0.9807   0.9947   0.9991
## Specificity            0.9995   0.9977   0.9975   0.9966   0.9994
## Pos Pred Value         0.9988   0.9903   0.9883   0.9824   0.9972
## Neg Pred Value         0.9981   0.9977   0.9959   0.9990   0.9998
## Prevalence             0.2855   0.1935   0.1757   0.1618   0.1835
## Detection Rate         0.2841   0.1917   0.1723   0.1609   0.1833
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9974   0.9940   0.9891   0.9957   0.9992
```

Predicting with bagging

```
modeltreebag <- train(classe~., data = dataTraining, method = "treebag")
confusionMatrix(dataTesting$classe, predict(modeltreebag,dataTesting))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1667    6    0    0    1
##          B   13 1110   12    2    2
##          C    1   15 1006    4    0
##          D    0    2   22  939    1
##          E    1    3    3    1 1074
##
## Overall Statistics
##
##                Accuracy : 0.9849
##                  95% CI : (0.9814, 0.9878)
##     No Information Rate : 0.2858
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9809
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
```

```
## Sensitivity            0.9911   0.9771   0.9645   0.9926   0.9963
## Specificity            0.9983   0.9939   0.9959   0.9949   0.9983
## Pos Pred Value         0.9958   0.9745   0.9805   0.9741   0.9926
## Neg Pred Value         0.9964   0.9945   0.9924   0.9986   0.9992
## Prevalence             0.2858   0.1930   0.1772   0.1607   0.1832
## Detection Rate         0.2833   0.1886   0.1709   0.1596   0.1825
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9947   0.9855   0.9802   0.9938   0.9973
```

Predicting with boosting

```
modelGBM <- train(classe~., data = dataTraining, method = "gbm", verbose = FALSE)
confusionMatrix(dataTesting$classe, predict(modelGBM,dataTesting))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1642   17   12    2    1
##          B   43 1064   29    2    1
##          C    0   40  972   14    0
##          D    1    2   44  911    6
##          E    5   11    8   10 1048
##
## Overall Statistics
##
##                Accuracy : 0.9579
##                  95% CI : (0.9524, 0.9628)
##     No Information Rate : 0.2873
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9467
##
##  Mcnemar's Test P-Value : 2.479e-09
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9710   0.9383   0.9127   0.9702   0.9924
## Specificity            0.9924   0.9842   0.9888   0.9893   0.9930
## Pos Pred Value         0.9809   0.9342   0.9474   0.9450   0.9686
## Neg Pred Value         0.9884   0.9853   0.9809   0.9943   0.9983
## Prevalence             0.2873   0.1927   0.1810   0.1596   0.1794
## Detection Rate         0.2790   0.1808   0.1652   0.1548   0.1781
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9817   0.9612   0.9507   0.9797   0.9927
```

Predictions - summary

The random forest had the best accuracy of ~99%.

Predicting on testing data set

```
print(predict(modelrf,testing))
```

```
##  [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
```