Human Activity Recognition

Omar Marquina

June 21, 2016

This is a study to identify the efectiveness of an individual performing dumbbel biceps curls, using werable devices. The base analysis information is found in http://groupware.les.inf.puc-rio.br/har. The goal is to develop a machine learning model that based on four sensors (belt, arm, forearm and dumbbell) can predict if the individual is executing the excersise in a proper manner (Classe A), or the opposite (Classes B to E).

1. Exploratory Analysis

a) Data Cleaning

After loading the information from the source, we can identify that there are several variables that have NA values with few observations valuables or not necessary like time and windows related. For such, these are considered to be removed from the model.

```
training <- read.table("pml-training.csv", header = TRUE, sep=",", na.strings = c(NA, ""))
testing  <- read.table("pml-testing.csv", header = TRUE, sep=",", na.strings = c(NA, ""))

# High number of NAs in some variables, provides low value to the model
table(training$user_name, is.na(training$max_roll_belt))
```

```
##
##            FALSE TRUE
##   adelmo      83 3809
##   carlitos    56 3056
##   charles     81 3455
##   eurico      54 3016
##   jeremy      77 3325
##   pedro       55 2555
```

```
# Mantain predictors without NAs and the outcome (classe)
dataCols <- c("user_name",
              "roll_belt",         "pitch_belt",        "yaw_belt",     "total_accel_belt",
              "gyros_belt_x",      "gyros_belt_y",      "gyros_belt_z",
              "accel_belt_x",      "accel_belt_y",      "accel_belt_z",
              "magnet_belt_x",     "magnet_belt_y",     "magnet_belt_z",
              "roll_arm",          "pitch_arm",         "yaw_arm", "total_accel_arm",
              "gyros_arm_x",       "gyros_arm_y",       "gyros_arm_z",
              "accel_arm_x",       "accel_arm_y",       "accel_arm_z",
              "magnet_arm_x",      "magnet_arm_y",      "magnet_arm_z",
              "roll_dumbbell",     "pitch_dumbbell",    "yaw_dumbbell", "total_accel_dumbbell",
              "gyros_dumbbell_x",  "gyros_dumbbell_y",  "gyros_dumbbell_z",
              "accel_dumbbell_x",  "accel_dumbbell_y",  "accel_dumbbell_z",
              "magnet_dumbbell_x", "magnet_dumbbell_y", "magnet_dumbbell_z",
              "roll_forearm",      "pitch_forearm",     "yaw_forearm",  "total_accel_forearm",
              "gyros_forearm_x",   "gyros_forearm_y",   "gyros_forearm_z",
              "accel_forearm_x",   "accel_forearm_y",   "accel_forearm_z",
```

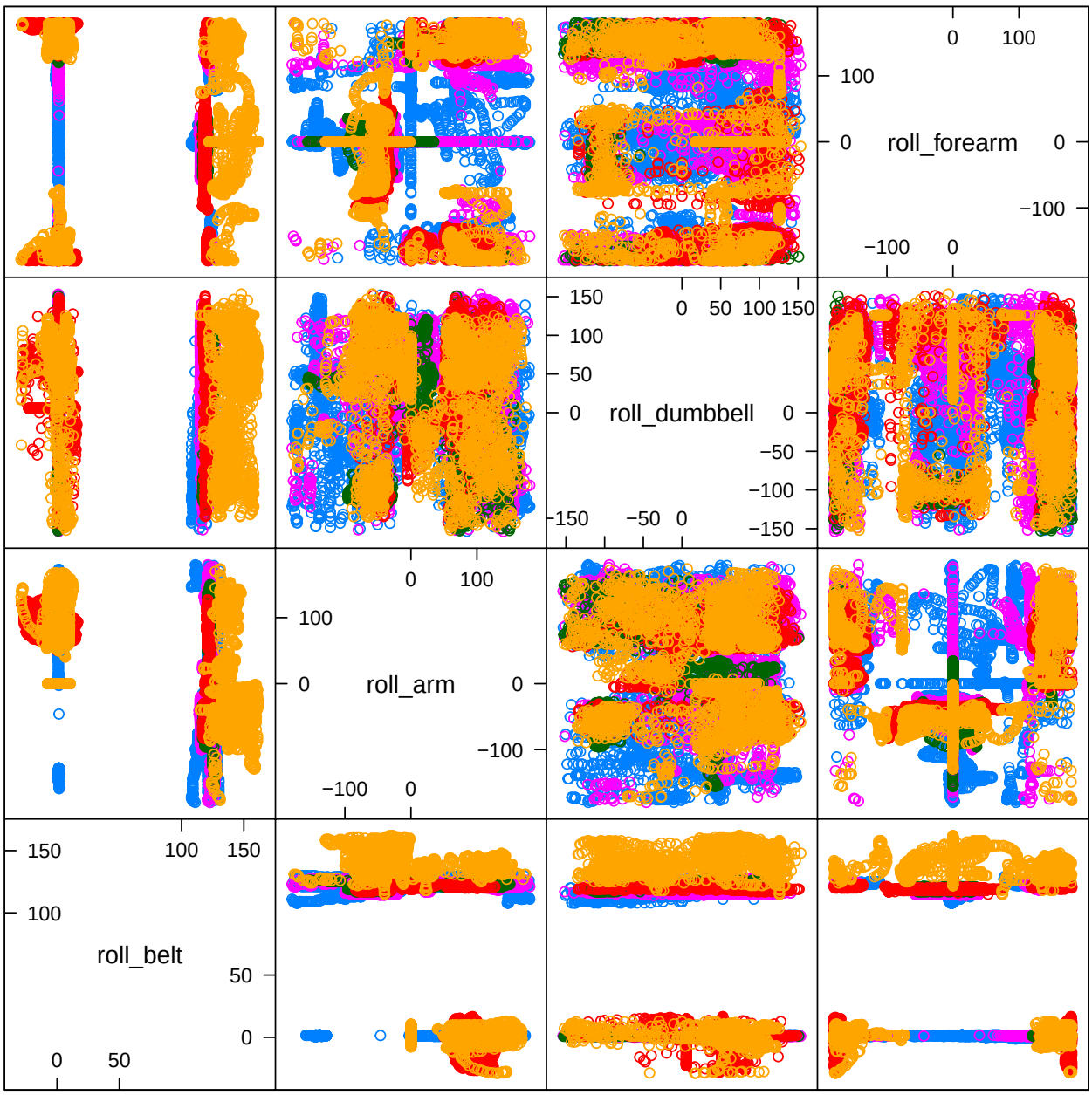
```
              "magnet_forearm_x",  "magnet_forearm_y",  "magnet_forearm_z",
              "classe")

# Remove none significant variabes
training <- training[,dataCols]
# Testing does not include outcome variable
testing <- testing[,dataCols[1:length(dataCols)-1]]
```

b) Plotting Trainig

Next ploting shows that it is hard to identify a clear patter on the data obtained of the 4 sensors and its corresponding indicators:

```
library(caret)
featurePlot(x=training[, c("roll_belt", "roll_arm","roll_dumbbell", "roll_forearm")],
            y=training$classe, plot = "pairs")
```



Scatter Plot Matrix

2. Prediction Study Design

a) Training Split

Training data will be splitted in 75% training and 25% testing samples size. The Testing data will be used as validation.

```
#Data splitting
set.seed(23987)
inTrain <- createDataPartition(y=training$classe, p=0.75, list = FALSE)
new.training   <- training[inTrain,]
new.testing    <- training[-inTrain,]
```

```
new.validation <- testing
dim(new.training) ; dim(new.testing) ; dim(new.validation)
```

```
## [1] 14718    54
```

```
## [1] 4904   54
```

```
## [1] 20 53
```

b) Method

The method used for this model will be Random Forest due the following considerations:

- It is very accurate for categorical outcomes among current algorithms
- Uses Bootstraps samples
- Grow multiple trees and vote
- It is harder to overfit
- It gives estimates of what variables are important in the classification

c) Cross validation

As this is a physical excersise that is executed over time, it can be considered as time series and for that the cross validation uses K-folds for model learning. Grouping the new.training set and identifying the mean size of observation for each class, we determine that having a fold of **5** will produce a good 100 observations per fold per classe in order to learn the model.

```
library(dplyr)
# Determine K-folds
mean(summarize(group_by(new.training, user_name, classe), count=n())$count)
```

```
## [1] 490.6
```

d) Model Learning

Using *caret* library for training the fitModel, using cross validation with 5 k-folds over all predictors and classe as the outcome.

```
set.seed(23955)
system.time (fitModel <- train(classe ~ ., data = new.training, method = "rf",
                               trControl=trainControl("cv", number = 5)
        )
)
```

```
##    user  system elapsed
## 900.698  10.437 942.792
```

```
fitModel
```

```
## Random Forest
##
## 14718 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11775, 11775, 11774, 11775, 11773
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9904199  0.9878802
##   29    0.9919827  0.9898582
##   57    0.9832854  0.9788524
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 29.
```

```
fitModel$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: 29
##
##         OOB estimate of  error rate: 0.69%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4175    8    1    0    1 0.002389486
## B   25 2816    6    1    0 0.011235955
## C    0   12 2546    9    0 0.008180756
## D    0    1   27 2381    3 0.012852405
## E    0    1    2    4 2699 0.002586844
```

Resulting in an accurracy of 99.2%

3. Predictions

Now using the *new.testing* for predicting and getting the Confusion Matrix:

```
predT <- predict(fitModel, new.testing)
(cMat <- confusionMatrix(predT, new.testing$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    4    0    0    0
##          B    1  939    3    0    0
```

```
##          C    0    6  851    4    2
##          D    0    0    1  797    1
##          E    0    0    0    3  898
##
## Overall Statistics
##
##                Accuracy : 0.9949
##                  95% CI : (0.9925, 0.9967)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9936
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9895   0.9953   0.9913   0.9967
## Specificity            0.9989   0.9990   0.9970   0.9995   0.9993
## Pos Pred Value         0.9971   0.9958   0.9861   0.9975   0.9967
## Neg Pred Value         0.9997   0.9975   0.9990   0.9983   0.9993
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1915   0.1735   0.1625   0.1831
## Detection Prevalence   0.2851   0.1923   0.1760   0.1629   0.1837
## Balanced Accuracy      0.9991   0.9942   0.9962   0.9954   0.9980
```

Resulting in an accurracy of 99.49%

4. Sample Errors

Calculating the sample errors (in and out of training and testing sets):

```
# Calculating in-sample error
accuracy.training <- fitModel$results[2,2]
in.sample.error <- 1 - accuracy.training

# Calculating out-sample error
accuracy.testing <- cMat$overall[1]
out.sample.error <- 1 - accuracy.testing

# Summary of accurracy and errors
data.frame(
        training = round(c(accuracy.training, in.sample.error), 4),
        testing  = round(c(accuracy.testing,out.sample.error),  4),
        row.names = c("Accuracy","In/Out Sample Error")
        )
```

```
##                     training testing
## Accuracy               0.992  0.9949
## In/Out Sample Error    0.008  0.0051
```

5. Validation

Using the validation data set, we can canculate only the predictions and estimate the error.

```
data.frame (predicted=predict(fitModel, new.validation))
```

```
##    predicted
## 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 estimated out-sample error would be very close to the obtained in the testing data set of 99.49%