Practical Machine Learning - Project

Wilfredo

August 19, 2018

Background

A dataset logging the activity of several individuals doing 5 activities was analyzed. The original data comes from the reference below. The loggin was conducted by using an on-body sensing approach.

The activities were: Class A: Correctly performing Unilateral Dumbbell Biceps Curl Class B: throwing the elbows to the front Class C: lifting the dumbbell only halfway Class D: lowering the dumbbell only halfway Class E: throwing the hips to the front

Reference: Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements. Proceedings of 21st Brazilian Symposium on Artificial Intelligence. Advances in Artificial Intelligence - SBIA 2012. In: Lecture Notes in Computer Science. , pp. 52-61. Curitiba, PR: Springer Berlin / Heidelberg, 2012. ISBN 978-3-642-34458-9. DOI: 10.1007/978-3-642-34459-6_6.

Processing and model building

```
set.seed(1234)
setwd("~/Personal/Coursera/Practical Machine Learning/Course project")
library(lattice)
library(ggplot2)
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library("caret")
library("MLmetrics")
```

```
##
## Attaching package: 'MLmetrics'
```

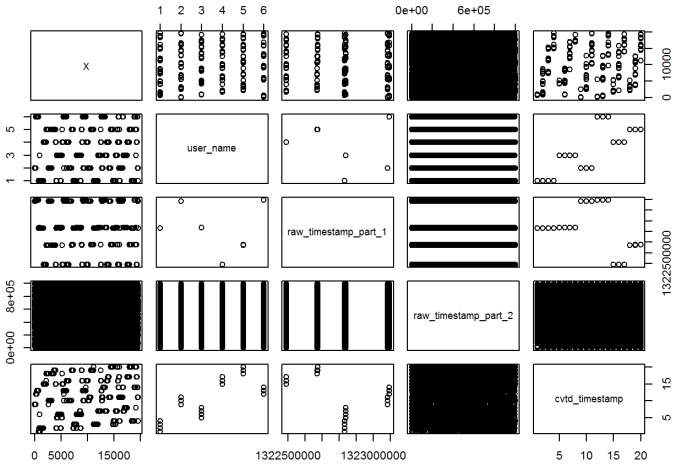
```
## The following objects are masked from 'package:caret':
##
##     MAE, RMSE
```

```
## The following object is masked from 'package:base':
##
##     Recall
```

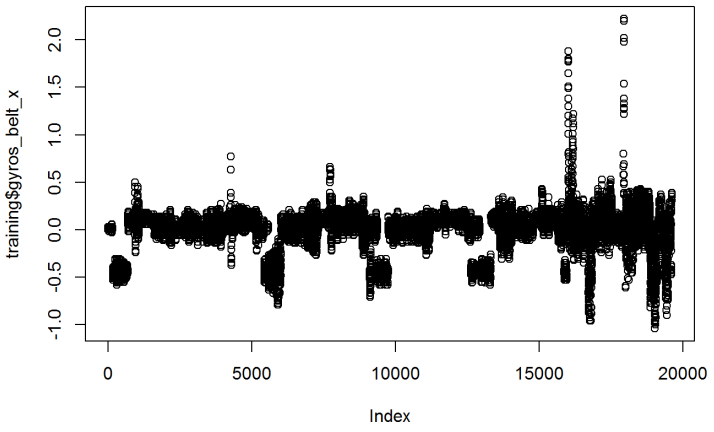
```
#Load data
training <- read.csv("pml-training.csv")
testing <- read.csv("pml-testing.csv")

#keep the 'y' variable
y <- training$classe

##Exploratory analysis
plot(training[1:5])
```



```
plot(training$gyros_belt_x)
```




```
#variables with missing data
na <- as.data.frame(sapply(training, function(x) sum(is.na(x)/length(training$X))))
table(na)
```

```
## na
##                 0 0.979308938946081
##                93                67
```

```
#variables with little variation
var <- as.data.frame(sapply(training, function(x) length(unique(x))))
rownames <- row.names(var)
var$variable <- rownames
colnames(var) <- c("unique","variable")
sort(x = var$unique, decreasing = F)
```

```
##   [1]     2     2     2     2     2     2     2     3     3     4     5
##  [12]     6    14    17    20    23    29    39    43    44    45    45
##  [23]    52    52    59    64    66    66    68    68    69    70    70
##  [34]    73    73    97   140   143   146   149   156   164   169   172
##  [45]   184   185   192   196   206   215   241   241   248   264   270
##  [56]   272   278   279   291   291   294   295   298   298   299   307
##  [67]   307   317   319   321   321   322   323   323   323   323   323
##  [78]   324   325   325   325   327   328   328   328   328   330   331
##  [89]   331   331   331   331   331   331   331   333   338   339   340
## [100]   357   376   384   385   388   392   392   392   392   392   392
## [111]   395   395   395   396   397   398   398   398   398   400   401
## [122]   401   402   410   425   457   466   537   580   643   676   741
## [133]   777   792   794   837   844   858   872  1003  1128  1265  1330
## [144]  1339  1524  1683  1840  1872  1957  1991  2176  2654  2876  2915
## [155]  3087 16040 16381 16523 16783 19622
```

```
##Feature selection
#Eliminate the columns with 97%
training <- training[, colMeans(is.na(training)) <= .97]
#Eliminate names and time stamps
training <- training[,-(1:7)]

#Eliminate variables that have less than 100 unique values.
#100 in the 19,622-observation dataset represents only a 0.5% variation
to.keep <- filter(var, unique > 100)
training <- select(training, one_of(to.keep$variable)) # select from the list of "to.keep"
```

```
## Warning: Unknown columns: `X`, `raw_timestamp_part_1`,
## `raw_timestamp_part_2`, `num_window`, `max_roll_belt`, `min_roll_belt`,
## `amplitude_roll_belt`, `avg_roll_belt`, `avg_pitch_belt`, `avg_yaw_belt`,
## `var_yaw_belt`, `var_accel_arm`, `avg_roll_arm`, `stddev_roll_arm`,
## `var_roll_arm`, `avg_pitch_arm`, `stddev_pitch_arm`, `var_pitch_arm`,
## `avg_yaw_arm`, `stddev_yaw_arm`, `var_yaw_arm`, `max_roll_arm`,
## `max_picth_arm`, `min_roll_arm`, `min_pitch_arm`, `amplitude_roll_arm`,
## `amplitude_pitch_arm`, `max_roll_dumbbell`, `max_picth_dumbbell`,
## `min_roll_dumbbell`, `min_pitch_dumbbell`, `amplitude_roll_dumbbell`,
## `amplitude_pitch_dumbbell`, `var_accel_dumbbell`, `avg_roll_dumbbell`,
## `stddev_roll_dumbbell`, `var_roll_dumbbell`, `avg_pitch_dumbbell`,
## `stddev_pitch_dumbbell`, `var_pitch_dumbbell`, `avg_yaw_dumbbell`,
## `stddev_yaw_dumbbell`, `var_yaw_dumbbell`, `max_roll_forearm`,
## `max_picth_forearm`, `min_roll_forearm`, `min_pitch_forearm`,
## `amplitude_roll_forearm`, `amplitude_pitch_forearm`, `var_accel_forearm`,
## `avg_roll_forearm`, `stddev_roll_forearm`, `var_roll_forearm`,
## `avg_pitch_forearm`, `stddev_pitch_forearm`, `var_pitch_forearm`,
## `avg_yaw_forearm`, `stddev_yaw_forearm`, `var_yaw_forearm`
```

```
##Principal Component analysis
#first change all variables to numeric
training <- sapply(X = training, FUN = as.numeric)
training <- as.data.frame(training)
#Based on summary function, it looks like the variables are across 3 logs 10^1 - 10^3. A log10 transformation might be appro
priate. We need to scale the variables
pca <- prcomp(x = training, scale. = T)
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6
## Standard deviation     3.7636 2.7594 2.6496 2.13085 2.02926 1.88327
## Proportion of Variance 0.2179 0.1171 0.1080 0.06985 0.06335 0.05456
## Cumulative Proportion  0.2179 0.3351 0.4431 0.51292 0.57628 0.63084
##                            PC7     PC8     PC9    PC10    PC11    PC12
## Standard deviation     1.67874 1.47700 1.36205 1.27814 1.18914 1.02741
## Proportion of Variance 0.04336 0.03356 0.02854 0.02513 0.02175 0.01624
## Cumulative Proportion  0.67420 0.70776 0.73630 0.76143 0.78319 0.79943
##                           PC13   PC14    PC15    PC16   PC17    PC18
## Standard deviation     0.96631 0.9121 0.89739 0.86873 0.7860 0.78060
## Proportion of Variance 0.01437 0.0128 0.01239 0.01161 0.0095 0.00937
## Cumulative Proportion  0.81379 0.8266 0.83898 0.85059 0.8601 0.86947
##                           PC19   PC20    PC21    PC22    PC23    PC24
## Standard deviation     0.71929 0.7077 0.65692 0.61828 0.61741 0.61105
## Proportion of Variance 0.00796 0.0077 0.00664 0.00588 0.00586 0.00574
## Cumulative Proportion  0.87743 0.8851 0.89177 0.89765 0.90352 0.90926
##                           PC25    PC26    PC27    PC28   PC29    PC30
## Standard deviation     0.57508 0.56229 0.54479 0.54187 0.5286 0.51807
## Proportion of Variance 0.00509 0.00486 0.00457 0.00452 0.0043 0.00413
## Cumulative Proportion  0.91435 0.91921 0.92378 0.92830 0.9326 0.93673
##                           PC31    PC32    PC33    PC34    PC35    PC36
## Standard deviation     0.51278 0.50313 0.48466 0.48260 0.47808 0.46503
## Proportion of Variance 0.00405 0.00389 0.00361 0.00358 0.00352 0.00333
## Cumulative Proportion  0.94077 0.94467 0.94828 0.95186 0.95538 0.95871
##                           PC37    PC38    PC39    PC40    PC41    PC42
## Standard deviation     0.46054 0.45109 0.43950 0.41826 0.41305 0.40614
## Proportion of Variance 0.00326 0.00313 0.00297 0.00269 0.00262 0.00254
## Cumulative Proportion  0.96197 0.96510 0.96807 0.97076 0.97339 0.97592
##                           PC43    PC44    PC45   PC46    PC47   PC48
## Standard deviation     0.40201 0.39774 0.39305 0.3779 0.36330 0.3513
## Proportion of Variance 0.00249 0.00243 0.00238 0.0022 0.00203 0.0019
## Cumulative Proportion  0.97841 0.98084 0.98322 0.9854 0.98745 0.9893
##                           PC49   PC50    PC51    PC52    PC53    PC54
## Standard deviation     0.31751 0.2908 0.27110 0.25633 0.23673 0.23102
## Proportion of Variance 0.00155 0.0013 0.00113 0.00101 0.00086 0.00082
## Cumulative Proportion  0.99090 0.9922 0.99333 0.99434 0.99520 0.99602
##                           PC55    PC56    PC57    PC58    PC59    PC60
## Standard deviation     0.19879 0.19224 0.18423 0.17572 0.16826 0.15699
## Proportion of Variance 0.00061 0.00057 0.00052 0.00048 0.00044 0.00038
## Cumulative Proportion  0.99663 0.99720 0.99772 0.99820 0.99863 0.99901
##                           PC61   PC62   PC63    PC64    PC65
## Standard deviation     0.14964 0.1388 0.1146 0.08378 0.04836
## Proportion of Variance 0.00034 0.0003 0.0002 0.00011 0.00004
## Cumulative Proportion  0.99936 0.9997 0.9999 0.99996 1.00000
```

```
#looks like the first 16 PC explain at least > 1% of the variation
#pca <- prcomp(x = training, scale. = T, rank. = 16)

training$classe <- y #add the y variable back

#Extract the variable names to match the testing dataset
features <- colnames(training)

#train model
#Cross validation was done by dividing data in 10 portions and using a grid approach for tuning parameters

control <- trainControl(method = "cv", number = 10, search = "grid")
set.seed(1234)
#Model was marked to avoid delaying the file creation
#rf <- train(classe~.,
#            data = training,
#            preProcess = c("scale", "pca"),
#            pcaComp = 16,
#            na.remove = T,
#            trControl = control,
#            method = "rf")

#saveRDS(rf, "rf.rds") #save model

my_model <- readRDS("rf.rds")

my_model
```

```
## Random Forest
##
## 19622 samples
##    65 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: scaled (65), principal component signal extraction
##  (65), centered (65)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 17659, 17661, 17659, 17661, 17659, 17659, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9776780  0.9717543
##   33    0.9711040  0.9634452
##   65    0.9692692  0.9611266
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

The model has an accuracy of 0.972. Hence, the expected out of sample error is at least 0.02.

Test the model

```
setwd("~/Personal/Coursera/Practical Machine Learning/Course project")
my_model <- readRDS("rf.rds")
testing$classe <- NULL
test <- select(testing, one_of(features)) # select from the list of features from the model
```

```
## Warning: Unknown columns: `classe`
```

```
#make variables numeric
test <- sapply(X = test, FUN = as.numeric)
test <- as.data.frame(test)
test1 <- test
test1[is.na(test1)] <- 0

#predict the testing dataset

predicted <- predict(object = my_model, newdata = test1)

predicted
```

```
##  [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
```