Coursera Machine Learning Project

Xenia Sokolova

2020, april 2

COURSERA MACHINE LEARNING PROJECT

Weight Lifting Exercises Dataset

EXECUTIVE SUMMARY

We propose a dataset with 5 classes (sitting-down, standing-up, standing, walking, and sitting) collected on 8 hours of activities of 4 healthy subjects. The goal of the project is to predict the manner in which they did the exercise.

BASIC EXPLORATORY DATA ANALYSIS

Downloading r packages:

```
library(readr)
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(dplyr)
```

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

## The following object is masked from 'package:randomForest':
##
##     combine
```

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

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

Loading data:

```
setwd("C:/Users/xssok/Documents")
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
              "pml-training.csv")
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
              "pml-testing.csv")

training_pml <-read.csv("pml-training.csv")
testing_pml <- read.csv("pml-testing.csv")
```

Checking dimensions of the training and testing data:

```
dim(training_pml)
```

```
## [1] 19622   160
```

```
dim(testing_pml)
```

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

Cross validation:

```
set.seed(1234)
inTrain <- createDataPartition(y=training_pml$classe,
                               p=0.6, list=FALSE)
training <- training_pml[inTrain,] ## subset data into training set
testing <- training_pml[-inTrain,] ## subset data into test set
```

Explore the training set:

```
dim(training)
```

```
## [1] 11776   160
```

```
dim(testing)
```

```
## [1] 7846  160
```

```r
str(training)
```

```
## 'data.frame':    11776 obs. of  160 variables:
##  $ X                       : int  2 3 4 5 6 7 8 10 11 12 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084232 1323084232 1323084232 1323084232 1
##  $ raw_timestamp_part_2    : int  808298 820366 120339 196328 304277 368296 440390 484434 500302 5283
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window              : int  11 11 12 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.45 1.45 1.43 ...
##  $ pitch_belt              : num  8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.17 8.18 8.18 ...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : Factor w/ 397 levels "","-0.016850",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_belt     : Factor w/ 317 levels "","-0.021887",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt      : Factor w/ 395 levels "","-0.003095",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt.1    : Factor w/ 338 levels "","-0.005928",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ max_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt            : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ min_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_belt            : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ amplitude_roll_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt    : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt      : Factor w/ 4 levels "","#DIV/0!","0.00",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ var_total_accel_belt    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_belt        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_belt_x            : num  0.02 0 0.02 0.02 0.02 0.02 0.02 0.03 0.03 0.02 ...
##  $ gyros_belt_y            : num  0 0 0 0.02 0 0 0 0 0 0 ...
##  $ gyros_belt_z            : num  -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 0 -0.02 -0.02 ...
##  $ accel_belt_x            : int  -22 -20 -22 -21 -21 -22 -22 -21 -21 -22 ...
##  $ accel_belt_y            : int  4 5 3 2 4 3 4 4 2 2 ...
##  $ accel_belt_z            : int  22 23 21 24 21 21 21 22 23 23 ...
##  $ magnet_belt_x           : int  -7 -2 -6 -6 0 -4 -2 -3 -5 -2 ...
##  $ magnet_belt_y           : int  608 600 604 600 603 599 603 609 596 602 ...
##  $ magnet_belt_z           : int  -311 -305 -310 -302 -312 -311 -313 -308 -317 -319 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm               : num  22.5 22.5 22.1 22.1 22 21.9 21.8 21.6 21.5 21.5 ...
##  $ yaw_arm                 : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm         : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ var_accel_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
```

```
##  $ avg_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_arm_x             : num  0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 0.02 ...
##  $ gyros_arm_y             : num  -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 -0.03 ...
##  $ gyros_arm_z             : num  -0.02 -0.02 0.02 0 0 0 0 -0.02 0 0 ...
##  $ accel_arm_x             : int  -290 -289 -289 -289 -289 -289 -289 -288 -290 -288 ...
##  $ accel_arm_y             : int  110 110 111 111 111 111 111 110 110 111 ...
##  $ accel_arm_z             : int  -125 -126 -123 -123 -122 -125 -124 -124 -123 -123 ...
##  $ magnet_arm_x            : int  -369 -368 -372 -374 -369 -373 -372 -376 -366 -363 ...
##  $ magnet_arm_y            : int  337 344 344 337 342 336 338 334 339 343 ...
##  $ magnet_arm_z            : int  513 513 512 506 513 509 510 516 509 520 ...
##  $ kurtosis_roll_arm       : Factor w/ 330 levels "","-0.02438",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_arm      : Factor w/ 328 levels "","-0.00484",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_arm        : Factor w/ 395 levels "","-0.01548",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_arm       : Factor w/ 331 levels "","-0.00051",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_arm      : Factor w/ 328 levels "","-0.00184",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_arm        : Factor w/ 395 levels "","-0.00311",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ max_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm       : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ roll_dumbbell           : num  13.1 12.9 13.4 13.4 13.4 ...
##  $ pitch_dumbbell          : num  -70.6 -70.3 -70.4 -70.4 -70.8 ...
##  $ yaw_dumbbell            : num  -84.7 -85.1 -84.9 -84.9 -84.5 ...
##  $ kurtosis_roll_dumbbell  : Factor w/ 398 levels "","-0.0035","-0.0073",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_dumbbell : Factor w/ 401 levels "","-0.0163","-0.0233",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_dumbbell   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_dumbbell  : Factor w/ 401 levels "","-0.0082","-0.0096",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_dumbbell : Factor w/ 402 levels "","-0.0053","-0.0084",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_dumbbell   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ max_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_dumbbell        : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ min_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell        : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

```
table(training$classe)
```

```
##
```

```
##    A    B    C    D    E
## 3348 2279 2054 1930 2165
```

FEATURE SELECTION

Exclude first 7 columns: "X", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2", "cvtd_timestamp", "new_window", "num_window", because we don't need them for the prediction.

```
training_ex <- select(training, -(X:num_window))
```

Select variables with missing data (more than 95%) and exclude them from the data table:

```
training_ex[training_ex==""] <- NA
NArate <- apply(training_ex, 2, function(x) sum(is.na(x)))/nrow(training_ex)
training_cl <- training_ex[!(NArate>0.95)]
```

Now we've got 53 variables:

```
str(training_cl)
```

```
## 'data.frame':    11776 obs. of  53 variables:
##  $ roll_belt           : num  1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.45 1.45 1.43 ...
##  $ pitch_belt          : num  8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.17 8.18 8.18 ...
##  $ yaw_belt            : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt    : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ gyros_belt_x        : num  0.02 0 0.02 0.02 0.02 0.02 0.02 0.03 0.03 0.02 ...
##  $ gyros_belt_y        : num  0 0 0 0.02 0 0 0 0 0 0 ...
##  $ gyros_belt_z        : num  -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 0 -0.02 -0.02 ...
##  $ accel_belt_x        : int  -22 -20 -22 -21 -21 -22 -22 -21 -21 -22 ...
##  $ accel_belt_y        : int  4 5 3 2 4 3 4 4 2 2 ...
##  $ accel_belt_z        : int  22 23 21 24 21 21 21 22 23 23 ...
##  $ magnet_belt_x       : int  -7 -2 -6 -6 0 -4 -2 -3 -5 -2 ...
##  $ magnet_belt_y       : int  608 600 604 600 603 599 603 609 596 602 ...
##  $ magnet_belt_z       : int  -311 -305 -310 -302 -312 -311 -313 -308 -317 -319 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm           : num  22.5 22.5 22.1 22.1 22 21.9 21.8 21.6 21.5 21.5 ...
##  $ yaw_arm             : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm     : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ gyros_arm_x         : num  0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 0.02 ...
##  $ gyros_arm_y         : num  -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 -0.03 ...
##  $ gyros_arm_z         : num  -0.02 -0.02 0.02 0 0 0 0 -0.02 0 0 ...
##  $ accel_arm_x         : int  -290 -289 -289 -289 -289 -289 -289 -288 -290 -288 ...
##  $ accel_arm_y         : int  110 110 111 111 111 111 111 110 110 111 ...
##  $ accel_arm_z         : int  -125 -126 -123 -123 -122 -125 -124 -124 -123 -123 ...
##  $ magnet_arm_x        : int  -369 -368 -372 -374 -369 -373 -372 -376 -366 -363 ...
##  $ magnet_arm_y        : int  337 344 344 337 342 336 338 334 339 343 ...
##  $ magnet_arm_z        : int  513 513 512 506 513 509 510 516 509 520 ...
##  $ roll_dumbbell       : num  13.1 12.9 13.4 13.4 13.4 ...
##  $ pitch_dumbbell      : num  -70.6 -70.3 -70.4 -70.4 -70.8 ...
##  $ yaw_dumbbell        : num  -84.7 -85.1 -84.9 -84.9 -84.5 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 ...
##  $ gyros_dumbbell_z    : num  0 0 -0.02 0 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x    : int  -233 -232 -232 -233 -234 -232 -234 -235 -233 -233 ...
##  $ accel_dumbbell_y    : int  47 46 48 48 48 47 46 48 47 47 ...
##  $ accel_dumbbell_z    : int  -269 -270 -269 -270 -269 -270 -272 -270 -269 -270 ...
##  $ magnet_dumbbell_x   : int  -555 -561 -552 -554 -558 -551 -555 -558 -564 -554 ...
##  $ magnet_dumbbell_y   : int  296 298 303 292 294 295 300 291 299 291 ...
##  $ magnet_dumbbell_z   : num  -64 -63 -60 -68 -66 -70 -74 -69 -64 -65 ...
##  $ roll_forearm        : num  28.3 28.3 28.1 28 27.9 27.9 27.8 27.7 27.6 27.5 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.8 -63.8 ...
##  $ yaw_forearm         : num  -153 -152 -152 -152 -152 -152 -152 -152 -152 -152 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 ...
##  $ gyros_forearm_y     : num  0 -0.02 -0.02 0 -0.02 0 -0.02 0 -0.02 0.02 ...
##  $ gyros_forearm_z     : num  -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 -0.03 ...
##  $ accel_forearm_x     : int  192 196 189 189 193 195 193 190 193 191 ...
##  $ accel_forearm_y     : int  203 204 206 206 203 205 205 205 205 203 ...
##  $ accel_forearm_z     : int  -216 -213 -214 -214 -215 -215 -213 -215 -214 -215 ...
##  $ magnet_forearm_x    : int  -18 -18 -16 -17 -9 -18 -9 -22 -17 -11 ...
##  $ magnet_forearm_y    : num  661 658 658 655 660 659 660 656 657 657 ...
##  $ magnet_forearm_z    : num  473 469 469 473 478 470 474 473 465 478 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Applying PCA Since the number of variables are over 50:

```
preProc <- preProcess(training_cl[,-53],method="pca",thresh=.8) #12 components to capture 80 percent of
preProc
```

```
## Created from 11776 samples and 52 variables
##
## Pre-processing:
##   - centered (52)
##   - ignored (0)
##   - principal component signal extraction (52)
##   - scaled (52)
##
## PCA needed 12 components to capture 80 percent of the variance
```

```
preProc <- preProcess(training_cl[,-53],method="pca",thresh=.9) #18 components to capture 90 percent of
preProc
```

```
## Created from 11776 samples and 52 variables
##
## Pre-processing:
##   - centered (52)
##   - ignored (0)
##   - principal component signal extraction (52)
##   - scaled (52)
##
## PCA needed 18 components to capture 90 percent of the variance
```

```r
preProc <- preProcess(training_cl[,-53],method="pca",thresh=.95) #24 components to capture 95 percent o
preProc
```

```
## Created from 11776 samples and 52 variables
##
## Pre-processing:
##   - centered (52)
##   - ignored (0)
##   - principal component signal extraction (52)
##   - scaled (52)
##
## PCA needed 24 components to capture 95 percent of the variance
```

```r
preProc <- preProcess(training_cl[,-53],method="pca",pcaComp=24)
training_pca <- predict(preProc,training_cl[,-53])
```

APPLYING A MODEL

Our data have got non-bionominal outcome and large sample size, that's why we're using random forest method.

```r
modFit <- randomForest(training_cl$classe ~ .,   data=training_pca, do.trace=F)
print(modFit) # view results
```

```
##
## Call:
##  randomForest(formula = training_cl$classe ~ ., data = training_pca,      do.trace = F)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 3.08%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3303   18   15    9    3  0.01344086
## B   57 2171   41    3    7  0.04738921
## C    4   34 1986   28    2  0.03310613
## D    3    1   90 1833    3  0.05025907
## E    3    8   22   12 2120  0.02078522
```

Checking on the test data:

```r
testing_cl <- select(testing, -(X:num_window))
testing_cl[testing_cl==""] <- NA
NArate <- apply(testing_cl, 2, function(x) sum(is.na(x)))/nrow(testing_cl)
testing_cl <- testing_cl[!(NArate>0.95)]
testing_pca <- predict(preProc,testing_cl[,-53])
confusionMatrix(testing_cl$classe,predict(modFit,testing_pca))
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2213    3    6   10    0
##          B   23 1470   24    0    1
##          C    3   28 1321   15    1
##          D    5    0   58 1220    3
##          E    2    8   13    9 1410
##
## Overall Statistics
##
##                Accuracy : 0.973
##                  95% CI : (0.9691, 0.9765)
##     No Information Rate : 0.2863
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9658
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9853   0.9742   0.9290   0.9729   0.9965
## Specificity            0.9966   0.9924   0.9927   0.9900   0.9950
## Pos Pred Value         0.9915   0.9684   0.9656   0.9487   0.9778
## Neg Pred Value         0.9941   0.9938   0.9844   0.9948   0.9992
## Prevalence             0.2863   0.1923   0.1812   0.1598   0.1803
## Detection Rate         0.2821   0.1874   0.1684   0.1555   0.1797
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9910   0.9833   0.9608   0.9814   0.9957
```

PREDICTING CLASSES FOR TEST DATA (SIZE OF 20)

```r
testing_pml_cl <- select(testing_pml, -(X:num_window))
testing_pml_cl[testing_pml_cl==""] <- NA
NArate <- apply(testing_pml_cl, 2, function(x) sum(is.na(x)))/nrow(testing_pml_cl)
testing_pml_cl <- testing_pml_cl[!(NArate>0.95)]
testing_pml_pca <- predict(preProc,testing_pml_cl[,-53])
testing_pml_cl$classe <- predict(modFit,testing_pml_pca)
cbind (as.character(testing_pml_cl$classe))
```

```
##       [,1]
##  [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"
```

CONCLUSION

In this analyses, 19622 observations from weight lifting exercise were used to analyze and predict correct body movement from others during the exercise. 60% of the total observations (11776 observations) were used to build a model by random forest method, and the rest of 40% of the observations (7846 observations) were used for model validation (cross-validation). The model statistics showed that the built model had the overall accuracy of 97% for the testing set, which is not overlapping with observations used to built the model. The sensitivity was in between 92%-99% and the specificity was over 99% for all classes. Overall, the model is well developed to predict the exercise classes during weight lifting. Therefore, under those condition, the model is expected to perform over 95% accuracy.