# Human Activity Recognition

*Ricardo Jorge Gil Ramos*

*02/22/2015*

## Executive Summary

The goal of this work is to predict human activities by collected data using devices such as Jawbone Up, Nike FuelBand, and Fitbit. The approach of research is based upon this paper http://groupware.les.inf.puc-rio.br/har.

The 5 possible methods include:

```
A: exactly according to the specification
B: throwing the elbows to the front
C: lifting the dumbbell only halfway
D: lowering the dumbbell only halfway
E: throwing the hips to the front
```

The main objectives of this project are as follows

```
Predict the manner in which they did the exercise depicted by the classe variable.
Build a prediction model using different features and cross-validation technique.
Calculate the out of sample error.
Use the prediction model to predict 20 different test cases provided.
```

## Analysis

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har

```
setwd('/home/user/machine_learning')
training_data <- read.csv(file="pml-training.csv", na.strings=c("", "NA"), header=TRUE, stringsAsFactors
testing_data <- read.csv(file="pml-testing.csv", , na.strings=c("", "NA"), header=TRUE, stringsAsFactors

# Create training classe as a factor
training_data$classe <- as.factor(training_data$classe)
```

We remove features that contains missing values in this way:

```
# Cleaning training and testing data
training_data_num_NA <- apply(training_data, 2, function(x) {sum(is.na(x))})
training_data_cleaned <- training_data[,which(training_data_num_NA==0)]
training_data_cleaned$classe <- as.factor(training_data_cleaned$classe)

testing_data_num_NA <- apply(testing_data, 2, function(x) {sum(is.na(x))})
testing_data_cleaned <- testing_data[,which(testing_data_num_NA==0)]
```

## Preprocessing variables

```
library(caret)
numeric_cols <- which(lapply(training_data_cleaned, class) %in% "numeric")

preObj <-preProcess(training_data_cleaned[,numeric_cols], method=c('knnImpute', 'center', 'scale'))
trainLess1 <- predict(preObj, training_data_cleaned[, numeric_cols])
trainLess1$classe <- training_data_cleaned$classe

testLess1 <-predict(preObj, testing_data_cleaned[, numeric_cols])
```

We use the function nearZeroVar in caret library to filter predictors that only have one unique value, or have few unique values relative to the number of samples and the ratio of frequency of the most common value to the frequency of second most common value is large.

```
nzv <- nearZeroVar(trainLess1, saveMetrics=TRUE)
training_data_final <- trainLess1[,nzv$nzv==FALSE]

nzv <- nearZeroVar(testLess1, saveMetrics=TRUE)
testing_data_final <- testLess1[,nzv$nzv==FALSE]
```

The final set of predictors used for classification are as follows.

```
names(training_data_final)
```

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

```
##  [1] "roll_belt"         "pitch_belt"        "yaw_belt"
##  [4] "gyros_belt_x"      "gyros_belt_y"      "gyros_belt_z"
##  [7] "roll_arm"          "pitch_arm"         "yaw_arm"
## [10] "gyros_arm_x"       "gyros_arm_y"       "gyros_arm_z"
## [13] "roll_dumbbell"     "pitch_dumbbell"    "yaw_dumbbell"
## [16] "gyros_dumbbell_x"  "gyros_dumbbell_y"  "gyros_dumbbell_z"
## [19] "magnet_dumbbell_z" "roll_forearm"      "pitch_forearm"
## [22] "yaw_forearm"       "gyros_forearm_x"   "gyros_forearm_y"
## [25] "gyros_forearm_z"   "magnet_forearm_y"  "magnet_forearm_z"
## [28] "classe"
```

## Create cross validation set

We divide the whole set in two parts, we split 70% of observations for training and the 30% remaining for crossvalidation.

```
set.seed(20150222)

inTrain <- createDataPartition(training_data_final$classe, p = 0.70, list=FALSE)
training <- training_data_final[inTrain,]
testing <- training_data_final[-inTrain,]
```

## Model and Prediction

Now, using the features in the training dataset, we will build our model using the Random Forest machine learning technique.

```
library(randomForest)
predictor <- randomForest(classe ~ ., data = training)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 0.63%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3900    6    0    0    0    0.001536
## B    8 2640   10    0    0    0.006772
## C    0    9 2365   20    2    0.012938
## D    0    0   20 2229    3    0.010213
## E    0    1    1    6 2517    0.003168
```

## In sample accuracy

Here, we calculate the in sample accuracy which is the prediction accuracy of our model on the training data set.

```
training_predictor = predict(predictor, newdata = training)
accuracy_training = confusionMatrix(training$classe, training_predictor)$overall[1]
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##         1.0000         1.0000         0.9997         1.0000         0.2843
## AccuracyPValue  McnemarPValue
##         0.0000            NaN
```

## Out sample accuracy

We calculate the out of sample accuracy of the model. In other words, this describes how accurately the model performs on the 30% testing dataset

```
testing_predictor = predict(predictor, newdata = testing)
accuracy_testing = confusionMatrix(testing$classe, testing_predictor)$overall[1]
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    1    0    0    1
##          B    4 1130    3    0    2
##          C    0    2 1020    4    0
##          D    0    0    3  958    3
##          E    0    0    0    1 1081
##
## Overall Statistics
##
##                Accuracy : 0.996
##                  95% CI : (0.994, 0.997)
##     No Information Rate : 0.285
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.995
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.998    0.997    0.994    0.995    0.994
## Specificity             1.000    0.998    0.999    0.999    1.000
## Pos Pred Value          0.999    0.992    0.994    0.994    0.999
## Neg Pred Value          0.999    0.999    0.999    0.999    0.999
## Prevalence              0.285    0.193    0.174    0.164    0.185
## Detection Rate          0.284    0.192    0.173    0.163    0.184
## Detection Prevalence    0.284    0.194    0.174    0.164    0.184
## Balanced Accuracy       0.999    0.998    0.996    0.997    0.997
```

It can be concluded that the algorithm's accuracy is equal to 99.59% and the out of sample error is 0.41%.

## Prediction Assignment

Here, we apply the machine learning algorithm we built above, to each of the 20 test cases in the testing data set provided.

```
answers <- predict(predictor, testing_data_final)
answers <- as.character(answers)
answers
```

```
##  [1] "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E" "A"
## [18] "B" "B" "B"
```

Finally, we write the answers to files as specified by the course instructor using the following code segment.

```
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
```

```
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

pml_write_files(answers)
```