# Human Activity Recognition based on Data Retrieved from Activity Monitoring Devices (Accelerometers)

## Background and Problem Statement

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. The goal for this project is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which they did the exercise. Participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available here: [http://groupware.les.inf.puc-rio.br/har] (see the section on the Weight Lifting Exercise Dataset).

## Data

The training data subset for this project is available here: [https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv]

The test data subset is available here: [https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv]

Data Source: [http://groupware.les.inf.puc-rio.br/har].

## Goal

The goal of this project is to find the model that more accurately predicts the "classe" variable in the training set using the rest of the variables available in the dataset.

## Preliminary Work

### Reproduceability

An overall pseudo-random number generator seed was set at 876. In order for the results below to be reproduced, the same seed should be used. Required libraries will need to be installed prior use (you can use "install.packages("nameofpackage")".

```
#Preliminaries
# set working directory
directory <- "C:\\Users\\Constantina\\Google Drive\\Data Science and Business Analytics\\coursera\\Data
setwd(directory)
set.seed(876)
```

## How the model was built

The outcome variable is **classe**, a factor variable with 5 levels. For this data set, participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in 5 different fashions:

- exactly 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)
- throwing the hips to the front (Class E)

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes." [1] Prediction evaluations will be based on maximizing the accuracy and minimizing the out-of-sample error. All other available variables after cleaning will be used for prediction. Two models will be tested using decision tree and random forest algorithms. The model with the highest accuracy will be chosen as the final model.

### Approach

1. Load the data set and briefly learn the characteristics of the data
2. Use cross-validation method to built a valid model; 75% of the original data is used for model building (training data) while the rest of 25% of the data is used for testing (testing data)
3. Since the number of variables in the training data is too large, clean the data by a) excluding variables which apparently cannot be explanatory variables, and b) reducing variables with little information.
4. Apply Principle component Analysis (PCA) to reduce the number of variables
5. Apply random forest method to build a model
6. Check the model with the testing data set
7. Apply the model to estimate classes of 20 observations

### Loading data

```
#download & load data
#trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
#testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
trainUrl <- "pml-training.csv"
testUrl <- "pml-testing.csv"
training <- read.csv(trainUrl, na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(testUrl, na.strings=c("NA","#DIV/0!",""))

library(caret)
library(lattice)
library(ggplot2)
```

## Cross-validation

Cross-validation will be performed by subsampling the training data set randomly without replacement into 2 subsamples: sub-training data (75% of the original training data set) and sub-testing data (25%). The models will be fitted on the sub-training data set, and tested on the sub-testing data. Once the most accurate model is choosen, it will be tested on the original testing data set.

```
in_train <- createDataPartition(y=training$classe, p=0.75, list=FALSE)
my_training <- training[in_train, ]; my_testing <- training[-in_train, ]
dim(my_training); dim(my_testing)
```

## Cleaning the training data

The following transformations were used to clean the data:

Transformation 1: Cleaning near-zero-variance variables. This kind of predictors are non-informative (most are constant across sample), and can possibly break models we may want to fit in the data.

The created variable "my_NZV" contains all the variables that have near zero variance

```
my_NZV <- nearZeroVar(my_training, saveMetrics=TRUE)

#checking the number of variables with near zero variance
library(plyr)
count(my_NZV$nzv)

#creating a second my_training variable, so that we don't loose the original training set
my_training2 <- my_training

#removing near zero variance variables from training dataset
my_training2 <- my_training2[,my_NZV$nzv == FALSE]

#checking new dimensions of data
dim(my_training2)
```

Transformation 2: Removing first ID variable so that it does not interfere with ML Algorithms:

```
my_training2 <- my_training2[c(-1)]
```

Transformation 3: Variables containing more than 60% of NA's are left out:

```
training_V3 <- my_training2 #creating another subset to iterate in loop
for(i in 1:length(my_training2)) { #for every column in the training dataset
        if( sum( is.na( my_training2[, i] ) ) /nrow(my_training2) >= .6 ) { #if n?? NAs > 60% of total
        for(j in 1:length(training_V3)) {
            if( length( grep(names(my_training2[i]), names(training_V3)[j]) ) ==1)  { #if the columns a
                training_V3 <- training_V3[ , -j] #Remove that column
            }
        }
    }
}
#checking new dimensions of data
dim(training_V3)

#Setting back to my_training2 set:
my_training2 <- training_V3
```

Transformations for my_testing and testing data sets.

```
clean1 <- colnames(my_training2)
clean2 <- colnames(my_training2[, -58]) #already with classe column removed
my_testing <- my_testing[clean1]
testing <- my_testing[clean2]
```

```
#To check the new N?? of observations
dim(my_testing)
dim(testing)
```

Data Coercion:

```
for (i in 1:length(testing) ) {
        for(j in 1:length(my_training2)) {
        if( length( grep(names(my_training2[i]), names(testing)[j]) ) ==1)  {
            class(testing[j]) <- class(my_training2[i])
        }
    }
}
#And to make sure Coercion really worked:
testing <- rbind(my_training2[2, -58] , testing)
testing <- testing[-1,]
```

## Using ML algorithms for prediction: Decision Tree

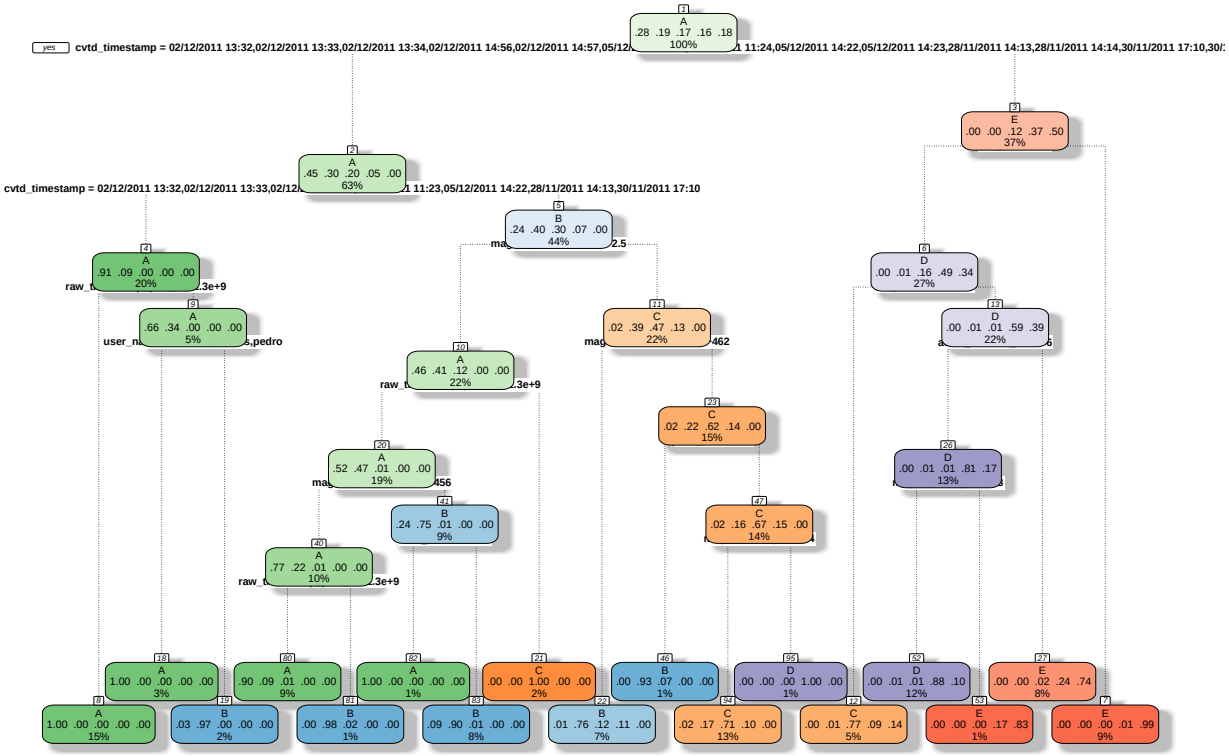
```
library(rpart)
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
mod_Fit_A1 <- rpart(classe ~ ., data=my_training2, method="class")
```

Note: Viewing the decision tree.

```
#varImpPlot(mod_Fit_A1)
#fancyRpartPlot(mod_Fit_A1)
library(rpart)
library(rpart.plot)
#plot(mod_Fit_A1)
#text(mod_Fit_A1)
fancyRpartPlot(mod_Fit_A1, cex = 0.3, sub="")
```



Predicting:

```
predictions_A1 <- predict(mod_Fit_A1, my_testing, type = "class")
```

Using confusion Matrix to test results:

```
confusionMatrix(predictions_A1, my_testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1348   41    4    1    0
##          B   35  783   51   41    0
##          C   12  120  783   84   31
##          D    0    5   11  550   48
##          E    0    0    6  128  822
##
## Overall Statistics
##
##                Accuracy : 0.874
##                  95% CI : (0.8644, 0.8831)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
```

```
##                   Kappa : 0.8405
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9663   0.8251   0.9158   0.6841   0.9123
## Specificity            0.9869   0.9679   0.9390   0.9844   0.9665
## Pos Pred Value         0.9670   0.8604   0.7602   0.8958   0.8598
## Neg Pred Value         0.9866   0.9584   0.9814   0.9408   0.9800
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2749   0.1597   0.1597   0.1122   0.1676
## Detection Prevalence   0.2843   0.1856   0.2100   0.1252   0.1949
## Balanced Accuracy      0.9766   0.8965   0.9274   0.8342   0.9394
```

## Using ML algorithms for prediction: Random Forests

```
library(randomForest)
mod_Fit_B1 <- randomForest(classe ~. , data=my_training2)
```

Predicting in-sample error:

```
predictions_B1 <- predict(mod_Fit_B1, my_testing, type = "class")
```

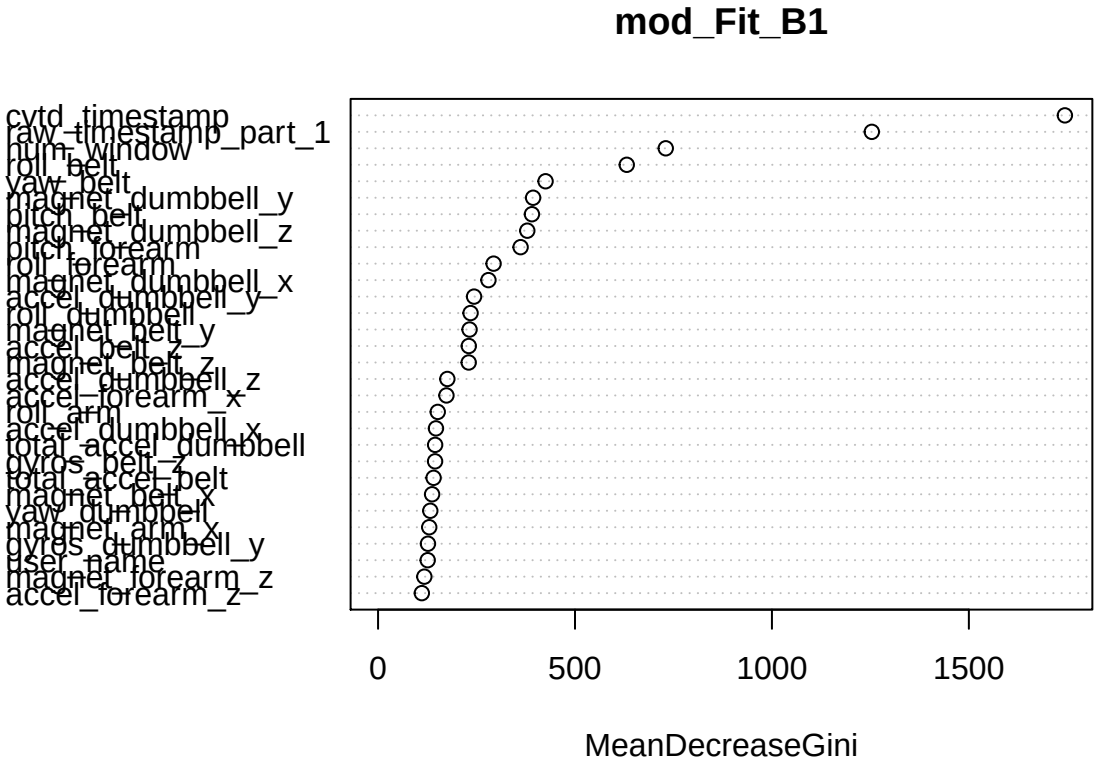
Testing model's accuracy

```
#checking accuracy
confusionMatrix(predictions_B1, my_testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    1    0    0    0
##          B    0  948    0    0    0
##          C    0    0  855    1    0
##          D    0    0    0  803    0
##          E    0    0    0    0  901
##
## Overall Statistics
##
##                Accuracy : 0.9996
##                  95% CI : (0.9985, 1)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9995
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9989   1.0000   0.9988   1.0000
## Specificity            0.9997   1.0000   0.9998   1.0000   1.0000
## Pos Pred Value         0.9993   1.0000   0.9988   1.0000   1.0000
## Neg Pred Value         1.0000   0.9997   1.0000   0.9998   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1933   0.1743   0.1637   0.1837
## Detection Prevalence   0.2847   0.1933   0.1746   0.1637   0.1837
## Balanced Accuracy      0.9999   0.9995   0.9999   0.9994   1.0000
```

```r
#viewing the importance for each factor
varImpPlot(mod_Fit_B1)
```




## Generating Files

Finally, we test the model using the provided test set out-of-sample error.

We use the random forest model, which yielded a much better prediction in in-sample:

```r
predictions_B2 <- predict(mod_Fit_B1, testing, type = "class")
confusionMatrix(predictions_B2, my_testing$classe)
```

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

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    1    0    0    0
##          B    0  948    0    0    0
##          C    0    0  855    1    0
##          D    0    0    0  803    0
##          E    0    0    0    0  901
##
## Overall Statistics
##
##                Accuracy : 0.9996
##                  95% CI : (0.9985, 1)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9995
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9989   1.0000   0.9988   1.0000
## Specificity            0.9997   1.0000   0.9998   1.0000   1.0000
## Pos Pred Value         0.9993   1.0000   0.9988   1.0000   1.0000
## Neg Pred Value         1.0000   0.9997   1.0000   0.9998   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1933   0.1743   0.1637   0.1837
## Detection Prevalence   0.2847   0.1933   0.1746   0.1637   0.1837
## Balanced Accuracy      0.9999   0.9995   0.9999   0.9994   1.0000
```

Generating related files with predictions

```
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(predictions_B2)
```