# Practical Machine Learning: Prediction Assignment

*Khalaq Zaman*

## 1 Introduction

By using training devices for example Jawbone Up, Nike FuelBand, and Fitbit we can gather a large amount of data about personal activity. These devices are the part of the quantified self movement. A group of enthusiastic people take measurements about themselves regularly to improve their health, in order to find patterns in their training behavior. 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. At the time of the collecting data, 6 participants were requested to perform barbell lifts in five different ways, one of the ways was done correctly and other four were done incorrectly. In this project, our goal will be to use data from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants to build a prediction model. Afterwards, we will use this prediction model to predict the outcome of the test data set. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har.

## 2 Data Loading and Cleaning

In the following we will load the packages which are required to build the prediction models.

```
#setwd("C:/Users/Md/Desktop/machine_learning/week4/Prediction_Assignment_WriteUp")
set.seed(1977)
library(knitr)
library(caret)
library(rpart)
library(rpart.plot)
library(rattle)
library(randomForest)
```

We have collected the data set, training and testing data set as instructed by Coursera Practical Machine Learning course instructor.

```
download.file(url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
              destfile = "./pml-training1.csv")

download.file(url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
              destfile = "./pml-testing1.csv")
```

In the following we read the *.csv files. In the* .csv files, many data with "NA", "#DIV/0!", or empty space"" exist. We replace everything with "NA".

```
data_training <- read.csv("pml-training1.csv", na.strings=c("NA","#DIV/0!",""))
data_testing <- read.csv("pml-testing1.csv", na.strings=c("NA","#DIV/0!",""))
```

In the following we calculated the percentage of "NA" in every column of the data sets. If any column contains "NA"s more than 70%, we excluded that column for the model building.

```
na_tag   <- sapply(data_training, function(x) mean(is.na(x)))
na_tag1 <- na_tag>0.7
data_training <- data_training[,!na_tag1]
data_testing <- data_testing[,!na_tag1]
dim(data_training)
```

```
## [1] 19622    60
```

```
dim(data_testing)
```

```
## [1] 20 60
```

Afterwards, we calculated the variance for every column of the data set. If the variance is very small, then we excluded that column from the building the prediction model.

```
zero_var <- nearZeroVar(data_training)
data_training <- data_training[,-zero_var]
data_testing <- data_testing[,-zero_var]
str(data_training)
```

```
## 'data.frame':    19622 obs. of  59 variables:
##  $ X                    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ 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 1323084231 1323084232 1323084232 1323084232 1323
##  $ raw_timestamp_part_2: int  788290 808298 820366 120339 196328 304277 368296 440390 484323 484434
##  $ cvtd_timestamp       : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ num_window           : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt            : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt           : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ 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 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y         : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z         : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...
##  $ accel_belt_x         : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y         : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z         : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x        : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y        : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z        : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm             : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm            : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ 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 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y          : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...
##  $ gyros_arm_z          : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x          : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y          : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z          : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
##  $ magnet_arm_x         : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
```

```
##  $ magnet_arm_y        : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z        : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ roll_dumbbell       : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell      : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell        : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ 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 -0.02 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x    : int  -234 -233 -232 -232 -233 -234 -232 -234 -232 -235 ...
##  $ accel_dumbbell_y    : int  47 47 46 48 48 48 47 46 47 48 ...
##  $ accel_dumbbell_z    : int  -271 -269 -270 -269 -270 -269 -270 -272 -269 -270 ...
##  $ magnet_dumbbell_x   : int  -559 -555 -561 -552 -554 -558 -551 -555 -549 -558 ...
##  $ magnet_dumbbell_y   : int  293 296 298 303 292 294 295 300 292 291 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -69 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.9 27.9 27.8 27.7 27.7 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.8 ...
##  $ yaw_forearm         : num  -153 -153 -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  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.03 0.02 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 ...
##  $ accel_forearm_x     : int  192 192 196 189 189 193 195 193 193 190 ...
##  $ accel_forearm_y     : int  203 203 204 206 206 203 205 205 204 205 ...
##  $ accel_forearm_z     : int  -215 -216 -213 -214 -214 -215 -215 -213 -214 -215 ...
##  $ magnet_forearm_x    : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
##  $ magnet_forearm_y    : num  654 661 658 658 655 660 659 660 653 656 ...
##  $ magnet_forearm_z    : num  476 473 469 469 473 478 470 474 476 473 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

In the following we will eliminate the first 6 column from the data sets, because they are time-series type data or are not numeric.

```
data_training <- data_training[,-(1:6)]
data_testing <- data_testing[,-(1:6)]
```

## 3 Partitioning the Data Set

According to the Coursera Practical Machine Learning Instructor, we are splitting the data set into 70% (to be used for the training of the model) and 30%(to be used for validation of the model) data set.

```
set.seed(1977)
train_set_id  <- createDataPartition(data_training$classe, p=0.7, list=FALSE)
data_training_set <- data_training[train_set_id, ]
data_validation_set  <- data_training[-train_set_id, ]
```

## 4 Prediction Model Building

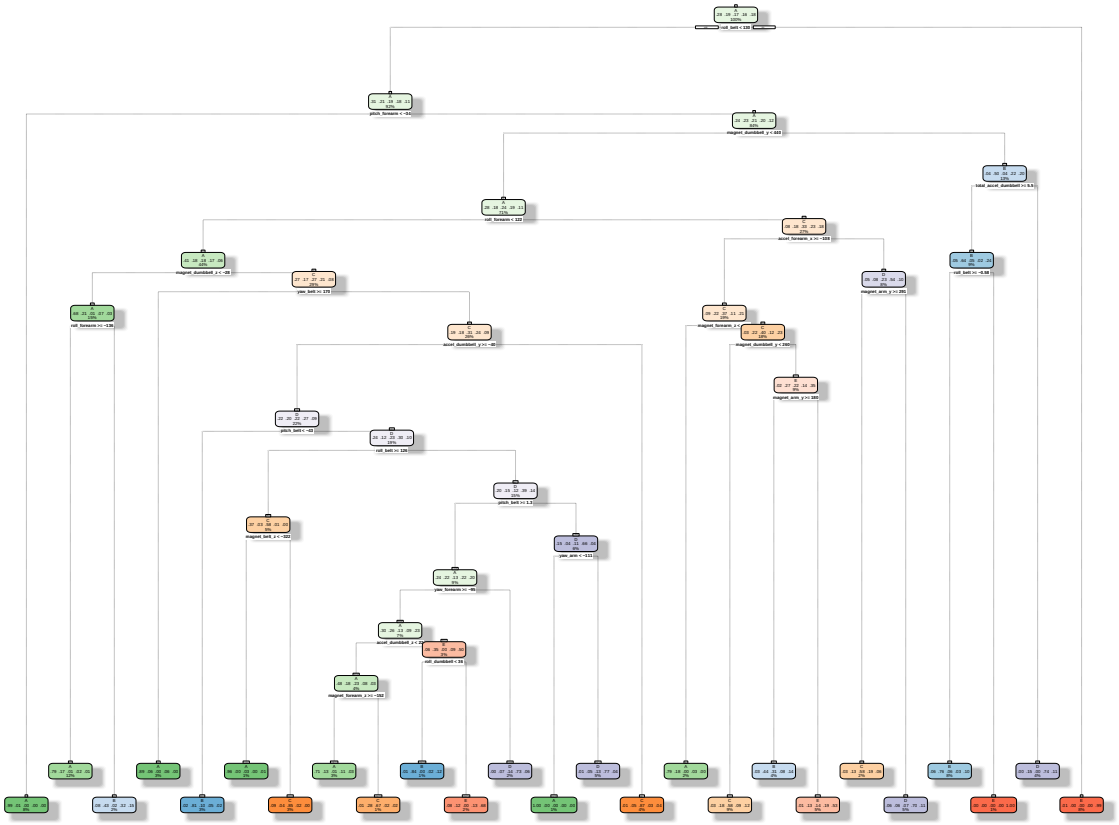
To building the prediction model for the given data set, we will use two different prediction model building algorithm, a) Decision Tree Algorithm and b) Random Forest Algorithm

### 4.1 Using Decision Tree Model

As it uses a single tree to build the prediction model, one should not expect high accuracy in the prediction model. A accuracy level in between 70 - 80% should be expected.

```
set.seed(1977)
model_fit_dt <- rpart(classe ~ ., data = data_training_set, method="class")
fancyRpartPlot(model_fit_dt)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2016−Nov−22 20:40:59 Md

```
predict_result_dt <- predict(model_fit_dt, data_validation_set, type = "class")
confusionMatrix(predict_result_dt, data_validation_set$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1526  166   15   51    6
##          B   60  688  126   93   98
##          C   39  133  777   82   77
##          D   33   81   75  653   83
```

```
##          E   16   71   33   85  818
##
## Overall Statistics
##
##                Accuracy : 0.7582
##                  95% CI : (0.747, 0.7691)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6936
##  Mcnemar's Test P-Value : 2.096e-15
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9116   0.6040   0.7573   0.6774   0.7560
## Specificity            0.9435   0.9206   0.9319   0.9447   0.9573
## Pos Pred Value         0.8651   0.6460   0.7013   0.7059   0.7996
## Neg Pred Value         0.9641   0.9064   0.9479   0.9373   0.9457
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2593   0.1169   0.1320   0.1110   0.1390
## Detection Prevalence   0.2997   0.1810   0.1883   0.1572   0.1738
## Balanced Accuracy      0.9275   0.7623   0.8446   0.8111   0.8567
```

From the model using Decision Tree, we observed that the accuracy is 76%.

## 4.2 Using Random Forest Model

As the random forest algorithm uses multiple decision trees, a improved accuracy could be expected. In the following, we will use 10 decision trees for the training of the prediction model.

```
set.seed(1977)
model_fit_rand_for <- train(classe ~ ., data=data_training_set, method="rf", ntree=10, importance = T,
predict_result_rand_for <- predict(model_fit_rand_for, data_validation_set)
confusionMatrix(predict_result_rand_for, data_validation_set$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1668    9    1    0    0
##          B    4 1126    6    1    0
##          C    1    4 1014   17    3
##          D    1    0    5  945    4
##          E    0    0    0    1 1075
##
## Overall Statistics
##
##                Accuracy : 0.9903
##                  95% CI : (0.9875, 0.9927)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.9877
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9886   0.9883   0.9803   0.9935
## Specificity            0.9976   0.9977   0.9949   0.9980   0.9998
## Pos Pred Value         0.9940   0.9903   0.9759   0.9895   0.9991
## Neg Pred Value         0.9986   0.9973   0.9975   0.9961   0.9985
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2834   0.1913   0.1723   0.1606   0.1827
## Detection Prevalence   0.2851   0.1932   0.1766   0.1623   0.1828
## Balanced Accuracy      0.9970   0.9931   0.9916   0.9891   0.9967
```

From the output above, we see that the accuracy is 99%. By increasing the number of decision trees in the random forest model, the accuracy of the prediction model can be improved.

# 5 Predicting the Test Data Set

A model using decision tree algorithm gives a accuracy of 76%, on the other hand using random forest algorithm we get 99% accuracy. For that reason we will use the model using random forest algorithm to predict the outcome the test data set.

```
predict_result_testing_data <- predict(model_fit_rand_for, data_testing)
predict_result_testing_data
```

```
##  [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
```