# Quantifying how well participants will do against barnbell lifting activity

by Mustafa Houd

7/14/2020

## Introduction

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. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har. The goal of this project is to predict the manner in which they did the execise (see the section on the Weight Lifting Exercise Dataset).

## Data

The training data for this project are available here:

[https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv]

The test data are available here:

[https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv]

## Goal Of this Project:

1. Predicting the manner in which the participants did the exercise. Refer to the "classe" variable in the training set. All other variables can be used as predictor.

2.Show how the model was built, performed cross validation, and expectation of the sample error and reasons of choices made.

3.Use the prediction model to predict 20 different test cases.

## Data Preprocessing:

```
library(caret)
## Warning: package 'caret' was built under R version 4.0.2
## Loading required package: lattice
## Loading required package: ggplot2
## Warning: package 'ggplot2' was built under R version 4.0.2
library(rpart)
library(knitr)
## Warning: package 'knitr' was built under R version 4.0.2
library(randomForest)
## Warning: package 'randomForest' was built under R version 4.0.2
## 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(corrplot)
## Warning: package 'corrplot' was built under R version 4.0.2
## corrplot 0.84 loaded
set.seed(888) # For research reproducibility purpose
trainUrl <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-tra
ining.csv"
testUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-tes
ting.csv"
trainFile <- "./data/pml-training.csv"
testFile  <- "./data/pml-testing.csv"
if (!file.exists("./data")) {
  dir.create("./data")
}
if (!file.exists(trainFile)) {
  download.file(trainUrl, destfile=trainFile)
}
```

```
if (!file.exists(testFile)) {
  download.file(testUrl, destfile=testFile)
}
```

## Read the data:

```
trainRaw <- read.csv("./data/pml-training.csv",header=T,sep=",",na.stri
ngs=c("NA",""))
testRaw <- read.csv("./data/pml-testing.csv",header=T,sep=",",na.string
s=c("NA",""))
dim(trainRaw); dim(testRaw)
## [1] 19622   160
## [1]  20 160
```

The training data set contains 19622 observations and 160 variables, while the testing data set contains 20 observations and 160 variables. The “classe” variable in the training set is the outcome to predict.

## Data Sets Partitioning Definitions

The data partitions of training and validating data sets are created as below:

```
trainRaw <- trainRaw[,-1] # Remove the first column that represents a I
D Row
inTrain = createDataPartition(trainRaw$classe, p=0.60, list=F)
training = trainRaw[inTrain,]
validating = trainRaw[-inTrain,]
```

## Data Cleaning

Since a random forest model is chosen and the data set must first be checked on possibility of columns without data.

The decision is made whereby all the columns that having less than 60% of data filled are removed.

```
sum((colSums(!is.na(training[,-ncol(training)])) < 0.6*nrow(training)))
# Number of columns with less than 60% of data
## [1] 100
```

Next, the criteria to remove columns that do not satisfy is applied before applying to the model.

```
Keep <- c((colSums(!is.na(training[,-ncol(training)])) >= 0.6*nrow(trai
ning)))

training <-  training[,Keep]

validating <- validating[,Keep]
```

## Modeling:

In random forests, there is no need for cross-validation or a separate test set to get an unbiased estimate of the test set error. It is estimated internally, during the execution. Therefore, the training of the model (Random Forest) is proceeded using the training data set.

```
training$classe <- as.character(training$classe)

training$classe <- as.factor(training$classe)

model <- randomForest(classe~.,data=training)

model
##

## Call:

##  randomForest(formula = classe ~ ., data = training)

##                Type of random forest: classification

##                      Number of trees: 500

## No. of variables tried at each split: 7

##

##         OOB estimate of  error rate: 0.2%

## Confusion matrix:

##      A    B    C    D    E class.error

## A 3348    0    0    0    0 0.000000000

## B    3 2276    0    0    0 0.001316367

## C    0    8 2044    2    0 0.004868549

## D    0    0    7 1922    1 0.004145078

## E    0    0    0    3 2162 0.001385681
```

## Model Evaluation:

Next, the model results is evaluated through the confusion Matrix.

The accurancy for the validating data set is calculated with the following formula:

```
acrcy<-c(as.numeric(predict(model,newdata=validating[,-ncol(validating)
])==validating$classe))

acrcy<-sum(acrcy)*100/nrow(validating)
```

Model Accuracy as tested over Validation set = 99.8725465% The out-of-sample error is 0.13%, which is pretty low.

## Model Test:

For the model testing, the new values are predicted using the testing dataset provided which was loaded earlier. Data cleaning was first performed and all columns of Testing data set are coerced for the same class of previous data set.

```
testRaw <- testRaw[,-1] # Remove the first column that represents a ID
Row

testRaw <- testRaw[ , Keep] # Keep the same columns of testing dataset

testRaw <- testRaw[,-ncol(testRaw)] # Remove the problem ID
```

### Transformations and Coercing of Testing Dataset

```
# Coerce testing dataset to same class and structure of training datase
t

testing <- rbind(training[100, -59] , testRaw)


# Apply the ID Row to row.names and 100 for dummy row from testing data
set

row.names(testing) <- c(100, 1:20)
```

## Prediction with the Testing Dataset

```
predictions <- predict(model,newdata=testing[-1,])

predictions
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20

##  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
```

### *Hope You enjoyed it*