

# Practical Machine Learning Course Project

## Background

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 dumbbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from [the website](#): (see the section on the Weight Lifting Exercise Dataset).

## Data

The training data for this project are available [here](#). The test data are available [here](#).

The data for this project come from [this source](#). Please cite them if you use this data to any further extent.

## Goal

The goal of this project is to predict the manner in which they did the exercise, which is the “classe” variable in the training set. This is a report detailing the building of the model, cross-validating and showing expected error, and using the model to predict 20 different test cases.

## Building the Model

First, we set seed and load the data from the working directory and some packages that we'll use:

```
# Set seed
set.seed(101010)

# Load the training and testing data
pml_training <- read.csv("pml-training.csv", na.strings=c('NA', '', ' '))
pml_testing <- read.csv("pml-testing.csv", na.strings=c('NA', '', ' '))

# Load required packages
require(caret)
```

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

```
require(randomForest)
```

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

Remove new window records, prepare modeling and predicting data:

```
model_data <- pml_training[pml_training$new_window == 'no', ]
predict_data <- pml_testing
```

Remove header and NA-ful predictors:

```
# Remove the header fields (index, user name, timestamps, window definitions)
header_fields <- 1:7
model_data <- model_data[-header_fields]
predict_data <- predict_data[-header_fields]

# Remove fields with NAs
na_fields <- which(sapply(model_data, function(x) sum(is.na(x)) > 0))
names(na_fields) <- rep(NULL, length(na_fields))
model_data <- model_data[-na_fields]
predict_data <- predict_data[-na_fields]
```

Create training and testing sets from the training data and preprocess training data:

```
# Partition pml_training into model training and testing data
in_training <- createDataPartition(y = model_data$classe, p = .8, list = FALSE)

# Pull in fully defined numeric fields (no username); we add classe back in later
training <- model_data[in_training, ]
testing <- model_data[-in_training, ]
```

Create model from training data and test against testing data:

```
# Create model from training data
pred_model <- randomForest(classe ~ ., data = training)

# Test model with testing data
testing_predictions <- predict(pred_model, testing)

# Print expected error
print(table(testing_predictions, testing$classe))
```

```
##
## testing_predictions      A      B      C      D      E
##           A 1094      5      0      0      0
##           B   0  735      2      0      0
##           C   0   3  667      3      0
##           D   0   0   1  626      0
##           E   0   0   0   0  705
```

## Use Model

Make predictions:

```
predictions <- predict(pred_model, predict_data)
```

Write predictions to file:

```
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)
```