# Practical Machine Learning - Assignment

*Anoop Makwana*

## Overview

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, we will use data recorded 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.

The goal of this project is to predict the manner in which the participants did the exercise. This is the classe variable of the training set, which classifies the correct and incorrect outcomes into A, B, C, D, and E categories. This report describes how the model for the project was built, its cross validation, expected out of sample error calculation, and the choices made

## Data Exploration

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

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

The next step is loading the dataset from the URL provided above.

```
library(knitr)
library(caret)
library(rpart)
library(rpart.plot)
library(randomForest)
library(corrplot)
set.seed(12345)

UrlTrain <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
UrlTest  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

training <- read.csv(url(UrlTrain), na.strings = c("NA", "#DIV/0!", ""))
testing  <- read.csv(url(UrlTest), na.strings = c("NA", "#DIV/0!", ""))
```

We take a quick look at the data and particularly at classe which is the variable we need to predict

```
str(training, list.len=20)
```

```
## 'data.frame':    19622 obs. of  160 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 1
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 484
```

```
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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 ...
##  $ kurtosis_roll_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_belt.1    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ max_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

Datasets have 160 variables. Let's first do some basic data clean-up:

- Removing all columns that are mostly NA
- Removing Near Zero Variance variables
- Removing identification and time only variables (columns 1 to 5)

```
NZV <- nearZeroVar(training)
training <- training[, -NZV]

AllNA    <- sapply(training, function(x) mean(is.na(x))) > 0.95
training <- training[, AllNA==FALSE]

training <- training[, -(1:5)]
testing  <- testing[, -(1:5)]
dim(training)
```

```
## [1] 19622    54
```

The training dataset is then partinioned in 2 sets to create a TrainSet (70% of the data) which will be used for training the model and the remaining 30% will be used for validation. Test dataset will not be touched and only used for quiz results.

```
inTrain  <- createDataPartition(training$classe, p=0.7, list=FALSE)
TrainSet <- training[inTrain, ]
TestSet  <- training[-inTrain, ]
```

With the cleaning process above, the number of variables for the analysis has been reduced to 54 only. To make an even more compact analysis, a PCA (Principal Components Analysis) could be performed as pre-processing step to the datasets (incase we dont get good results). Nevertheless, to keep the analysis simple, we will not apply it now

## Modeling

### Decision Tree

```
set.seed(12345)
model <- rpart(classe ~ ., data=TrainSet, method="class")
```

```r
# prediction on Test dataset
prediction <- predict(model, newdata=TestSet, type="class")
confusionMatrix <- confusionMatrix(prediction, TestSet$classe)
confusionMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1530  269   51   79   16
##          B   35  575   31   25   68
##          C   17   73  743   68   84
##          D   39  146  130  702  128
##          E   53   76   71   90  786
##
## Overall Statistics
##
##                Accuracy : 0.7368
##                  95% CI : (0.7253, 0.748)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6656
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9140  0.50483   0.7242   0.7282   0.7264
## Specificity            0.9014  0.96650   0.9502   0.9100   0.9396
## Pos Pred Value         0.7866  0.78338   0.7543   0.6131   0.7305
## Neg Pred Value         0.9635  0.89051   0.9422   0.9447   0.9384
## Prevalence             0.2845  0.19354   0.1743   0.1638   0.1839
## Detection Rate         0.2600  0.09771   0.1263   0.1193   0.1336
## Detection Prevalence   0.3305  0.12472   0.1674   0.1946   0.1828
## Balanced Accuracy      0.9077  0.73566   0.8372   0.8191   0.8330
```

We are getting an accuracy of 73% on validation data with decision tree. We will explore some other models to check if we can get better results

### Random Forest

```r
set.seed(12345)
model <- train(classe ~ ., data=TrainSet, method="rf", trControl=trainControl(method="cv", number=3, ve
model$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
```

```
##         OOB estimate of  error rate: 0.19%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    0    1 0.0005120328
## B    6 2651    1    0    0 0.0026335591
## C    0    6 2390    0    0 0.0025041736
## D    0    0    8 2244    0 0.0035523979
## E    0    0    0    3 2522 0.0011881188
```

```r
# prediction on Test dataset
prediction <- predict(model, newdata=TestSet)
confusionMatrix <- confusionMatrix(prediction, TestSet$classe)
confusionMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    5    0    0    0
##          B    0 1133    2    0    0
##          C    0    1 1024    7    0
##          D    0    0    0  957    4
##          E    0    0    0    0 1078
##
## Overall Statistics
##
##                Accuracy : 0.9968
##                  95% CI : (0.995, 0.9981)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9959
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9981   0.9927   0.9963
## Specificity            0.9988   0.9996   0.9984   0.9992   1.0000
## Pos Pred Value         0.9970   0.9982   0.9922   0.9958   1.0000
## Neg Pred Value         1.0000   0.9987   0.9996   0.9986   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1925   0.1740   0.1626   0.1832
## Detection Prevalence   0.2853   0.1929   0.1754   0.1633   0.1832
## Balanced Accuracy      0.9994   0.9972   0.9982   0.9960   0.9982
```

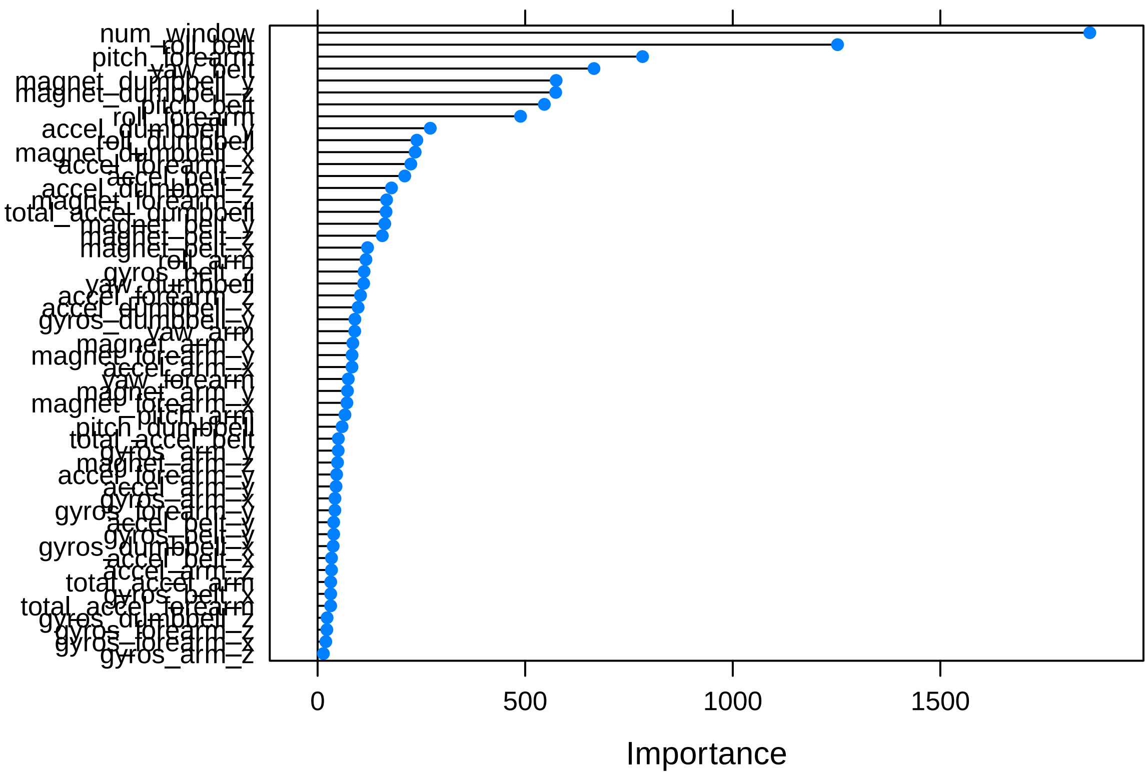
99.68% is a very impressive number for accuracy. Since we are getting a pretty high accuracy with Random Forest we will not explore other models.

### Relative importance of the variables

```r
# estimate variable importance
importance <- varImp(model, scale=FALSE)
# summarize importance
print(importance)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 53)
##
##                      Overall
## num_window            1860.1
## roll_belt             1252.5
## pitch_forearm          782.8
## yaw_belt               665.7
## magnet_dumbbell_y      574.6
## magnet_dumbbell_z      573.5
## pitch_belt             546.2
## roll_forearm           488.9
## accel_dumbbell_y       271.5
## roll_dumbbell          239.0
## magnet_dumbbell_x      235.0
## accel_forearm_x        224.5
## accel_belt_z           209.9
## accel_dumbbell_z       178.1
## magnet_forearm_z       166.2
## total_accel_dumbbell   165.0
## magnet_belt_y          161.8
## magnet_belt_z          155.9
## magnet_belt_x          120.4
## roll_arm               116.4
```

```r
# plot importance
plot(importance)
```



### Estimation of the out-of-sample error rate

The TestSet was removed and left untouched during training and optimizing of the Random Forest algorithm. Therefore this testing subset gives an unbiased estimate of the Random Forest algorithm's prediction accuracy (99.68% as calculated above). The Random Forest's out-of-sample error rate is derived by the formula 100% - Accuracy = 0.32%

## Coursera Submission (Applying the Selected Model to the Test Data)

```
predictTEST <- predict(model, newdata=testing)
predictTEST
```

```
##  [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
```