# Practical Machine Learning

*Liang Pei Ling*

*11/12/2017*

## 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, data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants will be used to predict the manner in which they did the exercise. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

## Loading data and packages for analysis

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(knitr)


training <- read.csv("pml-training.csv")
testing <- read.csv("pml-testing.csv")
```

Partioning the training set into two and cleaning the data.

```
inTrain <- createDataPartition(training$classe, p=0.6, list=FALSE)
myTraining <- training[inTrain, ]
myTesting <- training[-inTrain, ]
dim(myTraining); dim(myTesting)
```

```
## [1] 11776   160
```

```
## [1] 7846  160
```

## Cleaning the data

```
nzv <- nearZeroVar(myTraining, saveMetrics=TRUE)
myTraining <- myTraining[,nzv$nzv==FALSE]

nzv<- nearZeroVar(myTesting,saveMetrics=TRUE)
myTesting <- myTesting[,nzv$nzv==FALSE]

myTraining <- myTraining[c(-1)]

## Clean variables with more than 60% NA

trainingV3 <- myTraining
for(i in 1:length(myTraining)) {
    if( sum( is.na( myTraining[, i] ) ) /nrow(myTraining) >= .7) {
        for(j in 1:length(trainingV3)) {
            if( length( grep(names(myTraining[i]), names(trainingV3)[j]) ) == 1)  {
                trainingV3 <- trainingV3[ , -j]
            }
        }
    }
}

# Set back to the original variable name
myTraining <- trainingV3
rm(trainingV3)
```

Transform the myTesting and testing data sets

```
clean1 <- colnames(myTraining)

# remove the classe column
clean2 <- colnames(myTraining[, -58])

 # allow only variables in myTesting that are also in myTraining
myTesting <- myTesting[clean1]

# allow only variables in testing that are also in myTraining
testing <- testing[clean2]


# Coerce the data into the same type

for (i in 1:length(testing) ) {
    for(j in 1:length(myTraining)) {
        if( length( grep(names(myTraining[i]), names(testing)[j]) ) == 1)  {
            class(testing[j]) <- class(myTraining[i])
        }
    }
}
```

```
# To get the same class between testing and myTraining
testing <- rbind(myTraining[2, -58] , testing)
testing <- testing[-1,]
```

## Model Building - Random Forest Model

Random Forest Model was chosen due to its known good performance and prediction. I fit the model on ptrain1, and used the ???train??? function to use 3-fold cross-validation to select optimal tuning parameters for the model.

```
set.seed(12345)
modFitB1 <- randomForest(classe ~ ., data=myTraining)
predictionB1 <- predict(modFitB1, myTesting, type = "class")
cmrf <- confusionMatrix(predictionB1, myTesting$classe)
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    1    0    0    0
##          B    0 1517    6    0    0
##          C    0    0 1359    7    0
##          D    0    0    3 1279    4
##          E    0    0    0    0 1438
##
## Overall Statistics
##
##                Accuracy : 0.9973
##                  95% CI : (0.9959, 0.9983)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9966
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9993   0.9934   0.9946   0.9972
## Specificity            0.9998   0.9991   0.9989   0.9989   1.0000
## Pos Pred Value         0.9996   0.9961   0.9949   0.9946   1.0000
## Neg Pred Value         1.0000   0.9998   0.9986   0.9989   0.9994
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1933   0.1732   0.1630   0.1833
## Detection Prevalence   0.2846   0.1941   0.1741   0.1639   0.1833
## Balanced Accuracy      0.9999   0.9992   0.9962   0.9967   0.9986
```

## Model Evaluation & Selection

The accuracy of Random Forest is 99.8%, thus my predicted accuracy for the out-of-sample error is 0.2%.

This is an excellent result, so rather than trying additional algorithms, I will use Random Forests to predict on the test set.

## Predicting Results on Test Data

```
predictionB2 <- predict(modFitB1, testing, type = "class")
predictionB2
```

```
##  1 21  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
```