# Practical Machine Learning Assessment

*Tianxing Li*

*2015/4/23*

## 1. 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, the 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 (see the section on the Weight Lifting Exercise Dataset).

## 2. Load data

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

The packages used in the report are:

```
library(caret)
library(rpart)
library(randomForest)
```

Load the datasets from local. If the datasets don't exist, download them first.

```
# Download the datasets in case they don't exist.
  if (!file.exists("data/pml-training.csv")) {
  dir.create("data")
  download.file(
    "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
    "data/pml-training.csv")
}
if (!file.exists("data/pml-testing.csv")) {
  download.file(
    "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
    "data/pml-testing.csv")
}
```

```
pml_training <- read.csv("data/pml-training.csv", na.strings=c("NA","#DIV/0!",""))
pml_testing <- read.csv("data/pml-testing.csv", na.strings=c("NA","#DIV/0!",""))
```

## 3. Data cleaning

Set seed first.

```r
set.seed(10010)
```

The `pml_training` has 160 variables. Some variables are filled with `NA`s. Pick the variables with 90% `NA`s and drop them from our training dataset.

```r
na_rate <- colSums(is.na(pml_training)) / nrow(pml_training)
drops <- names(na_rate)[na_rate > 0.9]
pml_training <- pml_training[, !(names(pml_training) %in% drops)]
```

Some variables have near zero variance, drop them as well.

```r
pml_training <- pml_training[, !nearZeroVar(pml_training, saveMetrics=TRUE)$nzv]
```

Also drop the index, name and time variables. After we get the cleaned data, split it into training and testing dataset.

```r
drop_col_names <- c("X", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2", "cvtd_timestamp")
pml_training <- pml_training[, !(names(pml_training) %in% drop_col_names)]

split <- createDataPartition(y=pml_training$classe, p=0.75, list=FALSE)
split_training <- pml_training[split, ]
split_testing <- pml_training[-split, ]
```

## 4. Build model

### 4.1. Setup cross validation

With the data cleaned, build the decision tress.

```r
cv <- trainControl(method = "cv", number = 5, verboseIter=FALSE , allowParallel=TRUE)
```

### 4.2. Build decision tree

Build decision tree as folllow:

```r
dt <- rpart(classe ~ ., data=split_training, method="class")
```

Test the model with testing dataset.

```r
predict_dt <- predict(dt, split_testing, type = "class")
confusionMatrix(predict_dt, split_testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1266  199   46   78   25
##          B   38  519   25   25   18
##          C   17   57  683  122   68
##          D   63  124   49  498  103
##          E   11   50   52   81  687
##
## Overall Statistics
##
##                Accuracy : 0.7449
##                  95% CI : (0.7325, 0.7571)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6759
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9075   0.5469   0.7988   0.6194   0.7625
## Specificity            0.9008   0.9732   0.9348   0.9173   0.9515
## Pos Pred Value         0.7844   0.8304   0.7212   0.5950   0.7798
## Neg Pred Value         0.9608   0.8995   0.9565   0.9248   0.9468
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2582   0.1058   0.1393   0.1015   0.1401
## Detection Prevalence   0.3291   0.1274   0.1931   0.1707   0.1796
## Balanced Accuracy      0.9042   0.7600   0.8668   0.7684   0.8570
```

### 4.3. Build random forest

Build random forest as folllow:

```r
rf <- train(classe ~ ., data = split_training, method = "rf", trControl= cv)
```

Test the model with testing dataset.

```r
predict_rf <- predict(rf, split_testing)
confusionMatrix(predict_rf, split_testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    3    0    0    0
##          B    0  942    2    0    2
##          C    0    3  853    5    0
##          D    0    1    0  799    3
##          E    1    0    0    0  896
##
## Overall Statistics
##
##                Accuracy : 0.9959
##                  95% CI : (0.9937, 0.9975)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                  Kappa : 0.9948
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9926   0.9977   0.9938   0.9945
## Specificity            0.9991   0.9990   0.9980   0.9990   0.9998
## Pos Pred Value         0.9979   0.9958   0.9907   0.9950   0.9989
## Neg Pred Value         0.9997   0.9982   0.9995   0.9988   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1921   0.1739   0.1629   0.1827
## Detection Prevalence   0.2849   0.1929   0.1756   0.1637   0.1829
## Balanced Accuracy      0.9992   0.9958   0.9978   0.9964   0.9971
```

## 5. Decision

The test result above shows that random forest has better performance than random forest.

Choose random forest to predict the `pml_testing`.

```
answers <- predict(rf, pml_testing)

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(answers)
```