# Practical Machine Learning Course Project Write-up

*Sylvia Seow*

*August 19, 2015*

## Executive Summary

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 (see the section on the Weight Lifting Exercise Dataset).

## Input Data

We will initialise by loading neccessary library

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.1

## Loading required package: lattice
## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.2.1
```

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.2.1

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

```r
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.2.1
```

```r
set.seed(8888)
```

High level description of the raw data and detailed data description for this project has come from source: http://groupware.les.inf.puc-rio.br/har.

We will load the csv file downloaded into R

```
train<-read.csv("pml-training.csv",na.strings=c("NA",""), strip.white = T)
test <-read.csv("pml-testing.csv", na.strings=c("NA",""), strip.white = T)
```

### Formatting and cleaning data

Below code fragment is to clean and prepare the dataset for further processing, that step including the treatment of null value for data

```
isNA.train <- apply(train, 2, function(x) { sum(is.na(x)) })
isNA.test <- apply(test, 2, function(x) { sum(is.na(x)) })
training <- subset(train[, which(isNA.train == 0)],
                   select=-c(X, user_name, new_window, num_window, raw_timestamp_part_1, raw_timestamp_p
testing <- subset(test[, which(isNA.test == 0)],
                              select=-c(X, user_name, new_window, num_window, raw_timestamp_part_1, raw_
```

### Cross Validation/data spilting

We will create data partitition 60% for training and testing data. The method we use here is just simple hold-out, by spilting data into 2 set, which is training and another for testing

```
pml.training.index <- createDataPartition(y=training$classe,p=0.6,list=FALSE)
pml.training.train <- training[pml.training.index,]
pml.training.test <- training[-pml.training.index,]

dim(pml.training.train)
```

```
## [1] 11776    53
```

```
dim(pml.training.test)
```

```
## [1] 7846   53
```

### Analyse (Model Testing & Selection)

Below model used as shown:

#### Linear Discriminative Analysis

```
model.lda <- train(classe ~., method="lda", data=pml.training.train)
```

```
## Loading required package: MASS
```

```
## Warning: package 'MASS' was built under R version 3.2.1
```

```r
confusionMatrix(pml.training.train$classe, predict(model.lda, pml.training.train))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2757   67  267  247   10
##          B  345 1445  292   88  109
##          C  188  213 1364  242   47
##          D  110   73  250 1413   84
##          E   77  372  213  198 1305
##
## Overall Statistics
##
##                Accuracy : 0.7035
##                  95% CI : (0.6951, 0.7117)
##     No Information Rate : 0.2953
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6248
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7929   0.6659   0.5717   0.6458   0.8392
## Specificity            0.9288   0.9132   0.9265   0.9461   0.9159
## Pos Pred Value         0.8235   0.6341   0.6641   0.7321   0.6028
## Neg Pred Value         0.9146   0.9237   0.8949   0.9213   0.9740
## Prevalence             0.2953   0.1843   0.2026   0.1858   0.1320
## Detection Rate         0.2341   0.1227   0.1158   0.1200   0.1108
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.8609   0.7895   0.7491   0.7959   0.8775
```

**Trees**

```r
model.tree <- train(classe ~., method="rpart", data=pml.training.train)
confusionMatrix(pml.training.train$classe, predict(model.tree, pml.training.train))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2071   10  646  611   10
##          B  359  402  396 1122    0
##          C   48   49 1375  582    0
##          D  114   13  550 1253    0
##          E   29   15  409  731  981
##
## Overall Statistics
##
```

```
##                Accuracy : 0.5165
##                  95% CI : (0.5074, 0.5255)
##     No Information Rate : 0.3651
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3981
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7902  0.82209   0.4073   0.2915  0.98991
## Specificity            0.8605  0.83370   0.9192   0.9095  0.89022
## Pos Pred Value         0.6186  0.17639   0.6694   0.6492  0.45312
## Neg Pred Value         0.9347  0.99084   0.7942   0.6906  0.99896
## Prevalence             0.2226  0.04153   0.2867   0.3651  0.08415
## Detection Rate         0.1759  0.03414   0.1168   0.1064  0.08331
## Detection Prevalence   0.2843  0.19353   0.1744   0.1639  0.18385
## Balanced Accuracy      0.8253  0.82789   0.6632   0.6005  0.94006
```

#### Random Forest

```
model.randForest <- train(classe ~., model=FALSE, method="rf", data=pml.training.train,ntree=100,prox=T
confusionMatrix(pml.training.train$classe, predict(model.randForest, pml.training.train))
```

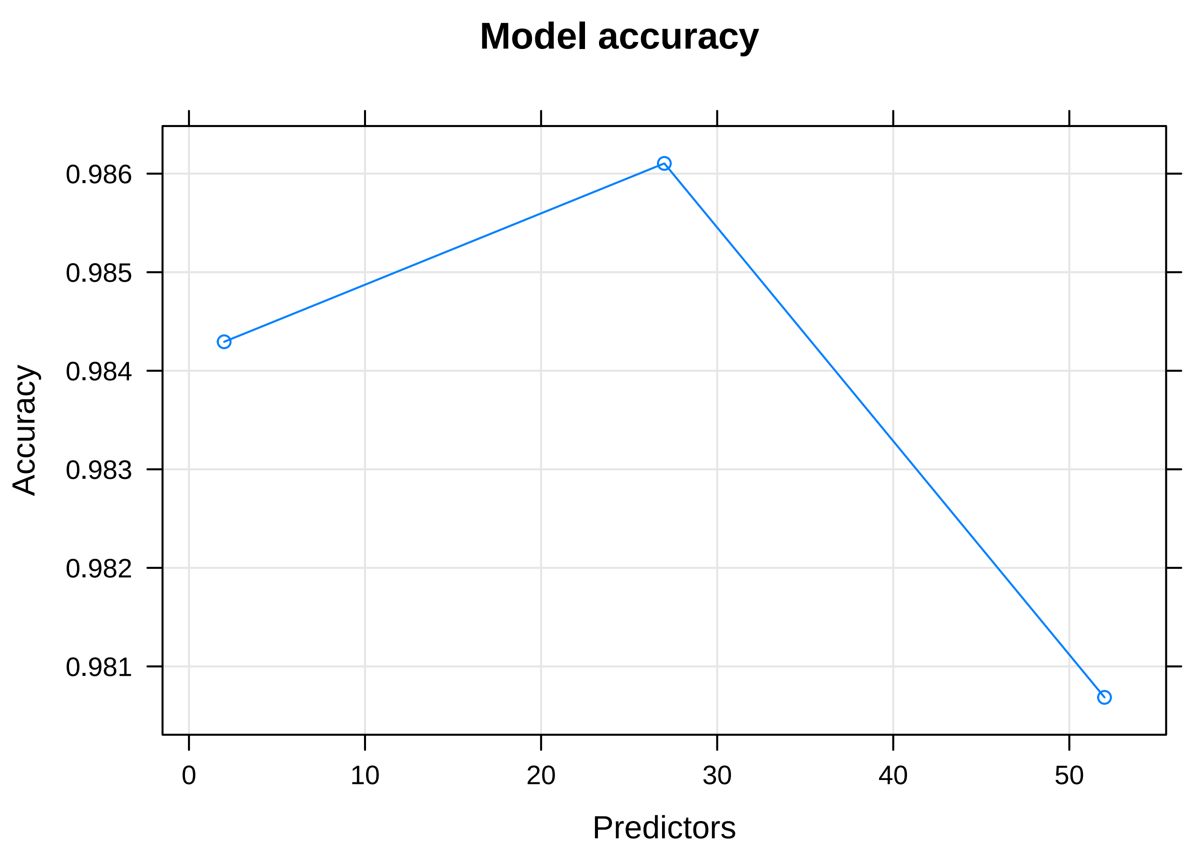
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3348    0    0    0    0
##          B    0 2279    0    0    0
##          C    0    0 2054    0    0
##          D    0    0    0 1930    0
##          E    0    0    0    0 2165
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
```

```
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

### Details on Random Forest Model

It seems that random forest provide the result with best "accuracy". the We then use the model to predict the classe value for the 6 participants in the testing dataset. We also apply the model on the validation dataset to determine the accuracy of the selected model. The OOB estimate of error is 0.65% which is excellent, the Confusion matrix looks good too. Next, we will take the look at the variable importance.

```
plot(model.randForest, log = "y",
     main = "Model accuracy",
     xlab = "Predictors",
     ylab = "Accuracy")
```




```
var.imp <- varImp(model.randForest)
var.imp
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
```

```
##                         Overall
## roll_belt                100.00
## pitch_forearm             59.70
## yaw_belt                  56.73
## magnet_dumbbell_z         48.26
## pitch_belt                43.52
## magnet_dumbbell_y         41.50
## roll_forearm              38.33
## accel_dumbbell_y          21.62
## roll_dumbbell             18.66
## magnet_dumbbell_x         18.38
## accel_forearm_x           17.13
## total_accel_dumbbell      15.36
## accel_belt_z              15.30
## accel_dumbbell_z          14.16
## magnet_forearm_z          12.85
## gyros_belt_z              12.77
## magnet_belt_z             12.38
## magnet_belt_x             11.32
## yaw_arm                   11.29
## magnet_belt_y             11.05
```

```
##var.imp$variable_name <- row.names(var.imp)
##var.imp[order(var.imp$Overall, decreasing=TRUE),]
```

we will apply the model to validation dataset and testing dataset from csv file

```
pml.val <- predict(model.randForest,newdata=pml.training.test)
pml.pred <- predict(model.randForest,newdata=testing)
result.test <-predict(model.randForest,testing)
```

## Testing & Result

Let's calculate the Out of Sample Error rate, or generalisation error, and the accuracy of the model based on the validation sub set of data that was used.

```
#calculate error rate and accuracy of the validation
ose.acc <- sum(pml.val == pml.training.test$classe)/length(pml.val)
ose.err <- (1 - ose.acc)
##show confusion matrix
confusionMatrix(pml.training.test$classe,pml.val)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2229    2    1    0    0
##          B   21 1493    4    0    0
##          C    0    9 1356    3    0
##          D    0    0   13 1271    2
##          E    0    0    5    3 1434
##
```

```
## Overall Statistics
##
##                Accuracy : 0.992
##                  95% CI : (0.9897, 0.9938)
##     No Information Rate : 0.2868
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9898
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9907   0.9927   0.9833   0.9953   0.9986
## Specificity            0.9995   0.9961   0.9981   0.9977   0.9988
## Pos Pred Value         0.9987   0.9835   0.9912   0.9883   0.9945
## Neg Pred Value         0.9963   0.9983   0.9964   0.9991   0.9997
## Prevalence             0.2868   0.1917   0.1758   0.1628   0.1830
## Detection Rate         0.2841   0.1903   0.1728   0.1620   0.1828
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9951   0.9944   0.9907   0.9965   0.9987
```

```r
## Accuracy
ose.acc
```

```
## [1] 0.9919704
```

```r
## Error Rate
ose.err
```

```
## [1] 0.008029569
```

### Evaluation

The achieved error value is below 5% and the prediction accurary close to 100%. So this is the best model to be used, although it does a long time to generate the model.

The final result on the testing dataset (test csv) is 20 correct prediction out of 20. So the accuracy is 100%

### Submission for grading

```r
## evaluate 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(result.test)
```