# Practical Machine Learning Project

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

## Data

You can download the data used here:

- The training data: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv
- The test data: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

```
descargarCSV <- function(url, NAstrings) {
    temp <- tempfile()
    download.file(url, temp, method = "curl")
    data <- read.csv(temp, na.strings = NAstrings)
    unlink(temp)
    return(data)
}
```

```
urlEntrenamiento <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
entrenamiento <- descargarCSV(urlEntrenamiento, c("NA", "#DIV/0!"))

urlPrueba <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
prueba <- descargarCSV(urlPrueba, c("", "NA", "#DIV/0!"))
```

## Exploratory data analysis

We see that there are too many NA values. They will have to be removed.

```
library(caret)
```

```
Porcentaje_maximo_NA = 90
NAmaximos <- nrow(entrenamiento) / 100 * Porcentaje_maximo_NA
removerColumnas <- which(colSums(is.na(entrenamiento) | entrenamiento == "") > NAmaximos)
entreLimpio <- entrenamiento[, -removerColumnas]
pruebaLimpio <- prueba[, -removerColumnas]
```

Doing that reduces the number of columns to 60.

```
dim(entreLimpio); dim(pruebaLimpio)
```

```
## [1] 19622    60
```

```
## [1] 20 60
```

By looking at the data. we see that the first columns have sequential numbers and time variations that we will not use. They will be removed.

```
entreOK <- entreLimpio[, -c(1:6)]
pruebaOK <- pruebaLimpio[, -c(1:6)]
dim(entreOK); dim(pruebaOK)
```

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

```
## [1] 20 54
```

Now we partition the data.

```
set.seed(134679)
enEntre <- createDataPartition(entreOK$classe, p = 3/4, list = F)
entrenar <- entreOK[enEntre, ]
validacion <- entreOK[-enEntre, ]
```

Analyzing the principal components, we got that 25 components are necessary to capture .95 of the variance. But it demands a lot of machine processing so, we decided by a .80 thresh to capture 80% of the variance using 13 components.

```
PCA <- preProcess(entrenar[, -54], method = "pca", thresh = 0.8)
PCA
```

```
## Created from 14718 samples and 53 variables
##
## Pre-processing:
##   - centered (53)
##   - ignored (0)
##   - principal component signal extraction (53)
##   - scaled (53)
##
## PCA needed 13 components to capture 80 percent of the variance
```

## Preprocessing

```
# The response class is excluded and the preProce object is created
preProce <- preProcess(entrenar[, -54], method = "pca", pcaComp = 13, thresh = 0.8)
# We apply the processing to the test and training data.
entrenarPCA <- predict(preProce, entrenar[, -54])
entrenarPCA$classe <- entrenar$classe
# entrenarPCA only has 13 principal components plus classe
PCAvalido <- predict(preProce, validacion[, -54])
PCAvalido$classe <- validacion$classe
# PCAvalid has only 13 main components plus classe
```

## Model examination

The random forest model will be used.

```
library(randomForest)
```

```
ajusteCOntrol <- trainControl(method = "cv", number = 5, allowParallel = T)

ajusteArbol <- train(classe ~ ., data = entrenarPCA, method = "rf", trControl = ajusteCOntrol)
```

```
print(ajusteArbol, digits = 4)
```

```
## Random Forest
##
## 14718 samples
##    13 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11775, 11775, 11773, 11774, 11775
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##    2    0.9608    0.9504
##    7    0.9562    0.9447
##   13    0.9491    0.9356
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
ajustePrediccion <- predict(ajusteArbol, PCAvalido)
(arbolMAtrix <- confusionMatrix(as.factor(PCAvalido$classe), ajustePrediccion))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1366    6   11   10    2
##          B   15  915   13    1    5
##          C    0   12  833   10    0
##          D    3    4   22  771    4
##          E    1   10    3    6  881
##
## Overall Statistics
##
##                Accuracy : 0.9719
##                  95% CI : (0.9668, 0.9763)
##     No Information Rate : 0.2824
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9644
##
##  Mcnemar's Test P-Value : 0.000746
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9863   0.9662   0.9444   0.9662   0.9877
## Specificity            0.9918   0.9914   0.9945   0.9920   0.9950
## Pos Pred Value         0.9792   0.9642   0.9743   0.9590   0.9778
## Neg Pred Value         0.9946   0.9919   0.9879   0.9934   0.9973
## Prevalence             0.2824   0.1931   0.1799   0.1627   0.1819
## Detection Rate         0.2785   0.1866   0.1699   0.1572   0.1796
```

```
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9890   0.9788   0.9695   0.9791   0.9913
```

```
precisionArbol <- arbolMAtrix$overall['Accuracy']
precisionArbol
```

```
##  Accuracy
## 0.9718597
```

The random forest method has a precision of 0.9718597 for this data set.

## Prediction on Testing Set

Finally we apply the random forest method to the variable "classe" in the test set.

```
pruebaPCA <- predict(preProce, pruebaOK[, -54])
pruebaPCA$problem_id <- pruebaOK$problem_id
pruebaFinal <- predict(ajusteArbol, pruebaPCA)
pruebaFinal
```

```
##  [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
```

With that we can see how they did the exercise.