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, 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.

Downloading and loading the data into R

First, the data is downloaded from the web and stored in a R dataset:

```
trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
trainingData <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
testingData <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))
```

Cleaning the datasets

Once the data is stored, unnecesary variables should be ommited, such as *name, timestamp, window and X* related variables, as well as the null values:

```
trainingSet<-trainingData[,colSums(is.na(trainingData)) == 0]
testingSet <-testingData[,colSums(is.na(testingData)) == 0]

unnecesaryVariables <- grep("name|timestamp|window|X", colnames(trainingSet), value=F)
trainingTidy <- trainingSet[,-unnecesaryVariables]
```

Splitting the training Data set (for cross validation)

In order to perform a cross validation, the training set is split in 2 parts: Training(70%) and Testing (30%)

```
index <- createDataPartition(y=trainingTidy$classe, p=0.7, list=FALSE)
myTraining <- trainingTidy[index, ]
myTesting <- trainingTidy[-index, ]
myTraining$classe <- as.factor(myTraining$classe)
myTesting$classe <- as.factor(myTesting$classe)
```

Random Forest

The first machine learning algorithm to be tested is Random Forest, using the class as the outcome and all the variables remaining as the predictors:

```
rfModel <- randomForest(classe ~ . , data=myTraining, method="class")
rfPred <- predict(rfModel, myTesting, type = "class")
confusionMatrix(rfPred, myTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674   10    0    0    0
##          B    0 1126   12    0    0
##          C    0    3 1014    8    3
##          D    0    0    0  956    1
##          E    0    0    0    0 1078
##
## Overall Statistics
##
##                Accuracy : 0.9937
##                  95% CI : (0.9913, 0.9956)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.992
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9886   0.9883   0.9917   0.9963
## Specificity            0.9976   0.9975   0.9971   0.9998   1.0000
## Pos Pred Value         0.9941   0.9895   0.9864   0.9990   1.0000
## Neg Pred Value         1.0000   0.9973   0.9975   0.9984   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1913   0.1723   0.1624   0.1832
## Detection Prevalence   0.2862   0.1934   0.1747   0.1626   0.1832
## Balanced Accuracy      0.9988   0.9930   0.9927   0.9957   0.9982
```

Decision Trees

The second machine learning algorithm to be tested is Decision Trees, using the class as the outcome and all the variables remaining as the predictors:

```
dtModel <- rpart(classe ~ ., data=myTraining, method="class")
dtPred <- predict(dtModel, myTesting, type = "class")
confusionMatrix(dtPred, myTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1521  212   73  130   55
##          B   49  644  107   82   81
##          C   38   88  742  130   90
##          D   36   79   69  537   48
##          E   30  116   35   85  808
##
## Overall Statistics
```

```
##
##                Accuracy : 0.7225
##                  95% CI : (0.7109, 0.7339)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6465
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9086   0.5654   0.7232  0.55705   0.7468
## Specificity            0.8884   0.9328   0.9288  0.95286   0.9446
## Pos Pred Value         0.7639   0.6687   0.6820  0.69831   0.7523
## Neg Pred Value         0.9607   0.8994   0.9408  0.91654   0.9430
## Prevalence             0.2845   0.1935   0.1743  0.16381   0.1839
## Detection Rate         0.2585   0.1094   0.1261  0.09125   0.1373
## Detection Prevalence   0.3383   0.1636   0.1849  0.13067   0.1825
## Balanced Accuracy      0.8985   0.7491   0.8260  0.75495   0.8457
```

Analysing the information given by the confussion Matrix, it is stated that the accuracy is sufficient to acknowledge the Random Forest model as more reliable, compared to the decision trees.

Using original Testing Data

Using the fitted model from the Random Forest algorithm, now it is going to be tested for the original test dataset (20 sets):

```r
testPred <- predict(rfModel, testingSet, type="class")
testPred
```

```
##  1  2  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
```