Summary

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. The goal of this project would 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 following 5 different ways-

- Class A - exactly according to the specification
- Class B - throwing the elbows to the front
- Class C - lifting the dumbbell only halfway
- Class D - lowering the dumbbell only halfway
- Class E - throwing the hips to the front

Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. Participants were supervised by an experienced weight lifter to make sure the execution complied to the manner they were supposed to simulate. The exercises were performed by six male participants aged between 20-28 years, with little weight lifting experience. We made sure that all participants could easily simulate the mistakes in a safe and controlled manner by using a relatively light dumbbell (1.25kg).

Data

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Getting and cleaning the Data

As part of this step, we will download the data, load it into R and clean it for further processing.

Load required libraries and set static variables

We load all required libraries in this section and set few variables to download Training and Testing data.

```
library(caret)
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
library(e1071)
library(randomForest)
set.seed(1)

training.url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testing.url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

path <- paste(getwd(),"/", "", sep = "")
training.file <- file.path(path, "training-data.csv")
testing.file <- file.path(path, "testing-data.csv")
```

Download the data files and clean unwanted data

In this section, we will download the training and testing data files. We will also replace the missing data to "NA" for further processing.

```
if (!file.exists(training.file)) {
        download.file(training.url, destfile = training.file)
}
if (!file.exists(testing.file)) {
        download.file(testing.url, destfile = testing.file)
}

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

Eliminate unwanted columns and data

Columns 1 to 7 are not relevant to this project, so we will drop them from data set. We will also get rid of column with NAs as part of cleaning.

```
# Drop columns 1 to 7 as they are not required for processing
training.data.clean1 <- training.data.raw[,8:length(colnames(training.data.raw))]
testing.data.clean1 <- testing.data.raw[,8:length(colnames(testing.data.raw))]

# Drop columns with NAs as they will also not be required
training.data.clean1 <- training.data.clean1[, colSums(is.na(training.data.clean1)) == 0]
testing.data.clean1 <- testing.data.clean1[, colSums(is.na(testing.data.clean1)) == 0]
```

Partitioning the training data set to allow cross-validation

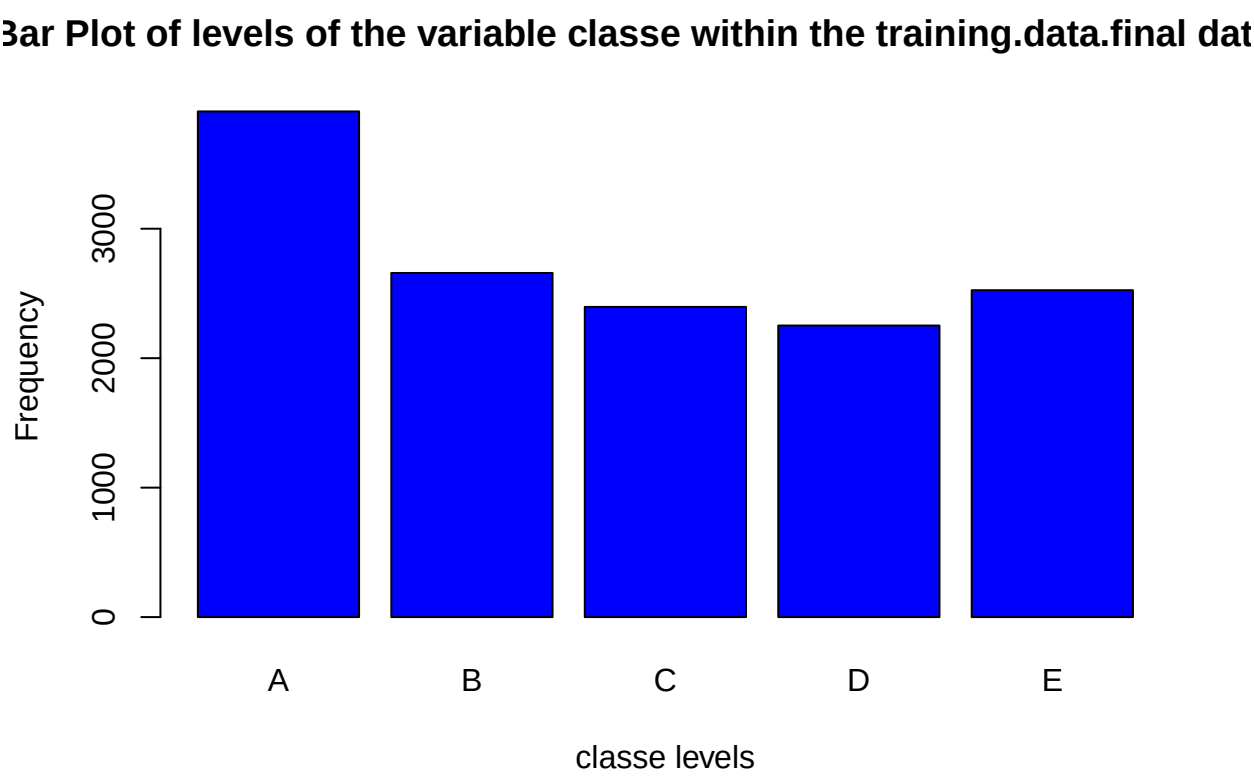
The training data will be divided into two sets. The first set will be 70% of the data which will be used to train the model. The second set is rest 30% to be used for validation

```
in.training <- createDataPartition(training.data.clean1$classe, p = 0.70, list=F)
training.data.final <- training.data.clean1[in.training, ]
testing.data.final <- training.data.clean1[-in.training, ]
```

A look at data

The variable "classe" contains 5 levels: A, B, C, D and E. A plot of the outcome variable will allow us to see the frequency of each levels in the "training.data.final" data set and compare one another.

```
plot(training.data.final$classe, col="blue", main="Bar Plot of levels of the variable classe within the
```



From the graph above, we can see that each level frequency is within the same order of magnitude of each other. Level A is the most frequent with more than 4000 occurrences while level D is the least frequent with about 2500 occurrences.

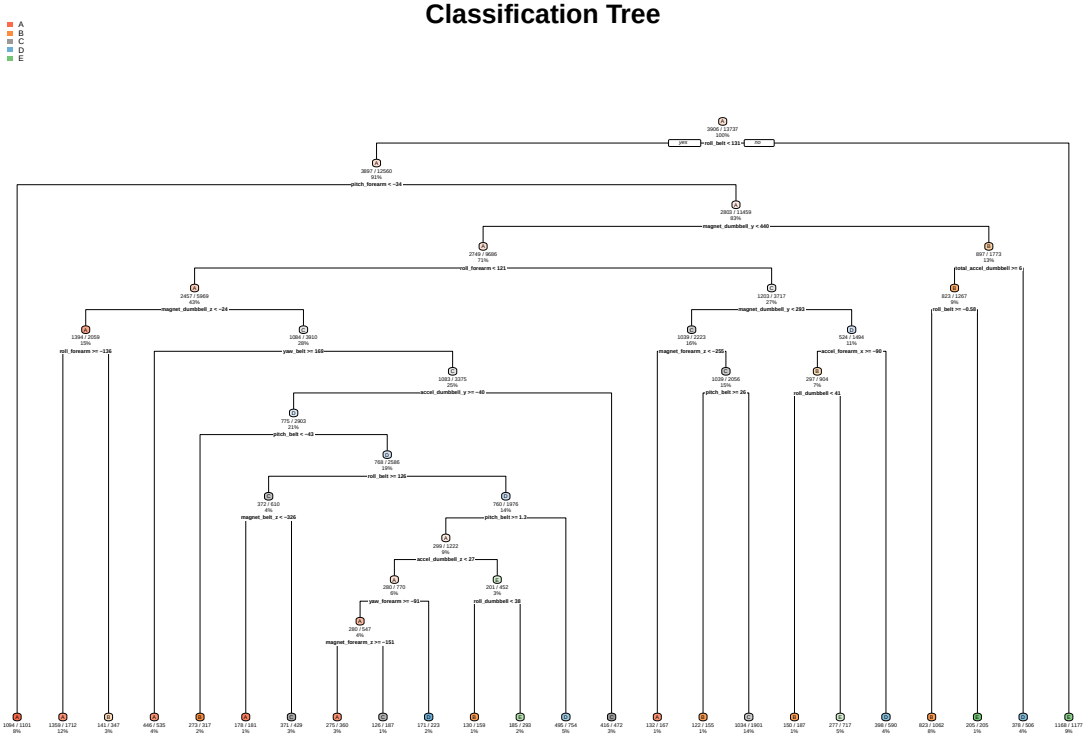
First prediction model: Decision Tree

We will use the Decision Tree prediction model in this section.

```
model1 <- rpart(classe ~ ., data = training.data.final, method = "class")

# Prediction
prediction1 <- predict(model1, testing.data.final, type = "class")

# Plot Decision Tree
rpart.plot(model1, main = "Classification Tree", extra = 102, under = TRUE, faclen = 0)
```




```
# Test results on our testing.data.final data set
confusionMatrix(prediction1, testing.data.final$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1473  160   24   44   15
##          B   55  699   56   73  101
##          C   57  121  818  161  132
##          D   56   82   60  609   53
##          E   33   77   68   77  781
##
## Overall Statistics
##
##                Accuracy : 0.7443
##                  95% CI : (0.7329, 0.7554)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6764
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
```

```
## Sensitivity            0.8799   0.6137   0.7973   0.6317   0.7218
## Specificity            0.9423   0.9399   0.9031   0.9490   0.9469
## Pos Pred Value         0.8584   0.7104   0.6346   0.7081   0.7539
## Neg Pred Value         0.9518   0.9102   0.9547   0.9294   0.9379
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2503   0.1188   0.1390   0.1035   0.1327
## Detection Prevalence   0.2916   0.1672   0.2190   0.1461   0.1760
## Balanced Accuracy      0.9111   0.7768   0.8502   0.7904   0.8344
```

Second prediction model: Random Forest

We will use the Random Forest prediction model in this section.

```
model2 <- randomForest(classe ~ . , data = training.data.final, method = "class")

# Prediction
prediction2 <- predict(model2, testing.data.final, type = "class")

# Test results on our testing.data.final data set
confusionMatrix(prediction2, testing.data.final$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    2    0    0    0
##          B    0 1136    7    0    0
##          C    0    1 1018    3    0
##          D    0    0    1  959    2
##          E    0    0    0    2 1080
##
## Overall Statistics
##
##                Accuracy : 0.9969
##                  95% CI : (0.9952, 0.9982)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9961
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9974   0.9922   0.9948   0.9982
## Specificity            0.9995   0.9985   0.9992   0.9994   0.9996
## Pos Pred Value         0.9988   0.9939   0.9961   0.9969   0.9982
## Neg Pred Value         1.0000   0.9994   0.9984   0.9990   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1930   0.1730   0.1630   0.1835
## Detection Prevalence   0.2848   0.1942   0.1737   0.1635   0.1839
## Balanced Accuracy      0.9998   0.9979   0.9957   0.9971   0.9989
```

Decision

As per the results, Random Forest algorithm performed better than Decision Tree. Accuracy for Random Forest model was 0.995 compared to 0.725 for Decision Tree model. The random Forest model is choosen as the accuracy of the model is 0.995.