# Practical Machine Learning

Executive Summary

GitHub Repo: https://github.com/wj9379/Practical-Machine-Learning-Project/tree/master

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 is to use data from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants as they perform barbell lifts correctly and incorrectly 5 different ways.

Six young healthy participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions:

First item Class A - exactly according to the specification

First itemClass B - throwing the elbows to the front

First itemClass C - lifting the dumbbell only halfway

First itemClass D - lowering the dumbbell only halfway

First item 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. Researchers made sure that all participants could easily simulate the mistakes in a safe and controlled manner by using a relatively light dumbbell (1.25kg).

Get Datasets

Load libraries

```
library(caret)
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
library(e1071)
library(randomForest)
library(VIM)
library(dplyr)
library(tidyr)
library(rpart.plot)
```

```
set.seed(1234)
```

Load datasets

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

train <- file.path( "machine-train-data.csv")
test <- file.path("machine-test-data.csv")
if (!file.exists(train)) {
        download.file(train.url, destfile=train)
}
if (!file.exists(test)) {
        download.file(test.url, destfile=test)
}

train_raw=read.csv(train,na.strings=c('NA','#DIV/0!',""))
test_raw=read.csv(test,na.strings=c('NA','#DIV/0!',""))
```

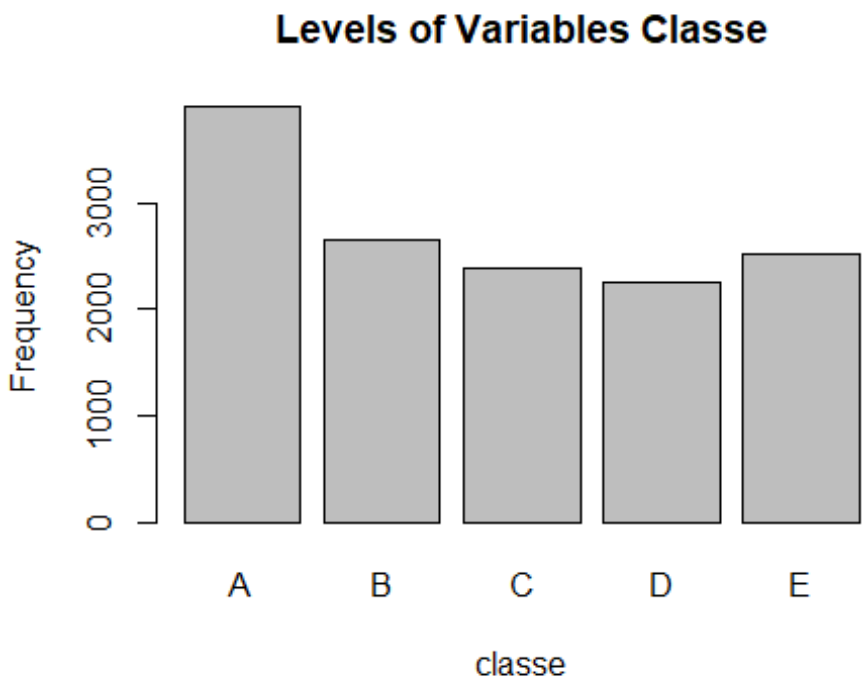
Clean Datasets

Remove columnns with NAs and irrelevant variables

```
train_clean=train_raw[,colSums(is.na(train_raw))==0]
test_clean=test_raw[,colSums(is.na(test_raw))==0]
train_clean=train_clean[,-c(1:7)]
test_clean=test_clean[,-c(1:7)]
```

Slice the data, 70% training dataset and 30% testing dataset. And we could see from the histogram below that level A is most frequency and level D is the least frequent.

```
trainingset <- createDataPartition(y=train_clean$classe, p=0.7, list=FALSE)
train_trainingset <- train_clean[trainingset, ]
test_trainingset <- train_clean[-trainingset, ]
plot(train_trainingset$classe,xlab='classe',ylab='Frequency',main='Levels of
Variables Classe')
```




Estimate Model

Model 1: Decision Tree: Fit a classfication tree

```
model1=rpart(classe~.,data=train_trainingset,method="class")
prediction1=predict(model1,test_trainingset,type='class')
rpart.plot(model1,main='Classification Tree',extra=103,under=T,faclen=0)

## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```

Model1 Results Testing:The accuracy for decision tree model is 0.7215(95%,CI:(0.7098, 0.7329))

```
confusionMatrix(prediction1,test_trainingset$classe)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1522  167   12   49   13
##          B   58  706  100   79   96
##          C   47  109  819  148  139
##          D   25   94   67  609   52
##          E   22   63   28   79  782
##
## Overall Statistics
##
##                Accuracy : 0.7541
```

```
##                  95% CI : (0.7429, 0.7651)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6885
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9092   0.6198   0.7982   0.6317   0.7227
## Specificity            0.9428   0.9298   0.9088   0.9516   0.9600
## Pos Pred Value         0.8633   0.6795   0.6490   0.7190   0.8029
## Neg Pred Value         0.9631   0.9106   0.9552   0.9295   0.9389
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2586   0.1200   0.1392   0.1035   0.1329
## Detection Prevalence   0.2996   0.1766   0.2144   0.1439   0.1655
## Balanced Accuracy      0.9260   0.7748   0.8535   0.7917   0.8414
```

Model2:Random Forest: Fit a random forest model since it automatically selects important variables and reduce the variance as well

```
model2=randomForest(classe~.,data=train_trainingset,method='class')
prediction2=predict(model2,test_trainingset,type='class')
```

Model2 Results Testing:The accuracy for random forest is 0.9952 (95%, CI:(0.9931, 0.9968))

```
confusionMatrix(prediction2,test_trainingset$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    2    0    0    0
##          B    1 1133   11    0    0
##          C    0    4 1014    6    0
##          D    0    0    1  957    0
##          E    0    0    0    1 1082
##
## Overall Statistics
##
##                Accuracy : 0.9956
##                  95% CI : (0.9935, 0.9971)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9944
##
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9947   0.9883   0.9927   1.0000
## Specificity            0.9995   0.9975   0.9979   0.9998   0.9998
## Pos Pred Value         0.9988   0.9895   0.9902   0.9990   0.9991
## Neg Pred Value         0.9998   0.9987   0.9975   0.9986   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1925   0.1723   0.1626   0.1839
## Detection Prevalence   0.2846   0.1946   0.1740   0.1628   0.1840
## Balanced Accuracy      0.9995   0.9961   0.9931   0.9963   0.9999
```

Model Selection :The random forest model is chosen since it has a higher accuracy.

Submission

Use Random forest algorithm to predict outcome levels on the original testing dataset.

```
predictfinal <- predict(model2, test_clean, type="class")
predictfinal
```

```
##  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
```