# Practical Machine Learning Course Project

Sammy

6/9/2021

## Introduction

In this project we will be building a model to predict the method in which unilateral dumbbell bicep curls were performed. The methods are determined by classes (labelled “classe” in the dataset) and include the following:

- 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

Data was collected from 6 participants’ measurements from accelerometers on the belt, forearm, arm, and dumbell.

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

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

## I. Environment Preparation

```
library(ggplot2)
library(caret)
library(lattice)
library(rpart)
library(rattle)
library(randomForest)
library(gbm)
set.seed(1004)
```

## II. Data Processing and Cleaning

### II.A. Download Data

Download both training and testing CSV files.

Training

```r
trainURL <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
download.file(trainURL, destfile = "pmltraining.csv", method = "curl")
training <- read.csv("pmltraining.csv", sep = ",", header = TRUE, na.strings = "NA")
```

Testing

```r
testURL <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(testURL, destfile = "pmltesting.csv", method = "curl")
testing <- read.csv("pmltesting.csv", sep = ",", header = TRUE, na.strings = "NA")
```

### II.B. Cleaning the Data

The training dataset will be partitioned 70% to create a validation set with the outcome of interest "classe". The testing set will only be used in the final prediction in order to prevent bias.

```r
inTrain <- createDataPartition(y = training$classe, p = 0.7, list = FALSE)
training <- training[inTrain, ]
validation <- training[-inTrain, ]
```

The dataset has 160 variables. There are 13737, 4123 and 20 observations for the training, validation and testing set.

```r
data.frame(dim(training), dim(validation), dim(testing), row.names = c("observations", "variables"))
```

```
##              dim.training. dim.validation. dim.testing.
## observations         13737            4123           20
## variables              160             160          160
```

In order to simplify the training and validation dataset we will first remove variables that mostly (90%) have NA values.

```r
training <- training[,colMeans(is.na(training)) < 0.9]
validation <- validation[,colMeans(is.na(validation)) < 0.9]
data.frame(dim(training), dim(validation), dim(testing), row.names = c("observations", "variables"))
```

```
##              dim.training. dim.validation. dim.testing.
## observations         13737            4123           20
## variables               93              93          160
```

Next, we will remove near zero variance variable.

```r
NZV <- nearZeroVar(training)
training <- training[,-NZV]
validation <- validation[,-NZV]
data.frame(dim(training), dim(validation),dim(testing), row.names = c("observations", "variables"))
```

```
##              dim.training. dim.validation. dim.testing.
## observations         13737            4123           20
## variables               59              59          160
```

Additionally, we will remove the first seven columns that are used for sample identification which are, as expected, not necessary for further prediction analysis.

```
training <- training[, -(1:7)]
validation  <- validation[, -(1:7)]
data.frame(dim(training), dim(validation), dim(testing), row.names = c("observations", "variables"))
```

```
##              dim.training. dim.validation. dim.testing.
## observations         13737            4123           20
## variables               52              52          160
```

The final dataset has 52 variables for the training and validation dataset. Additionally, observations in each dataset has reduced to 9619 and 2896 for training and validation datasets, respectively. The testing set has not been further processed and will be used only at the final prediction.

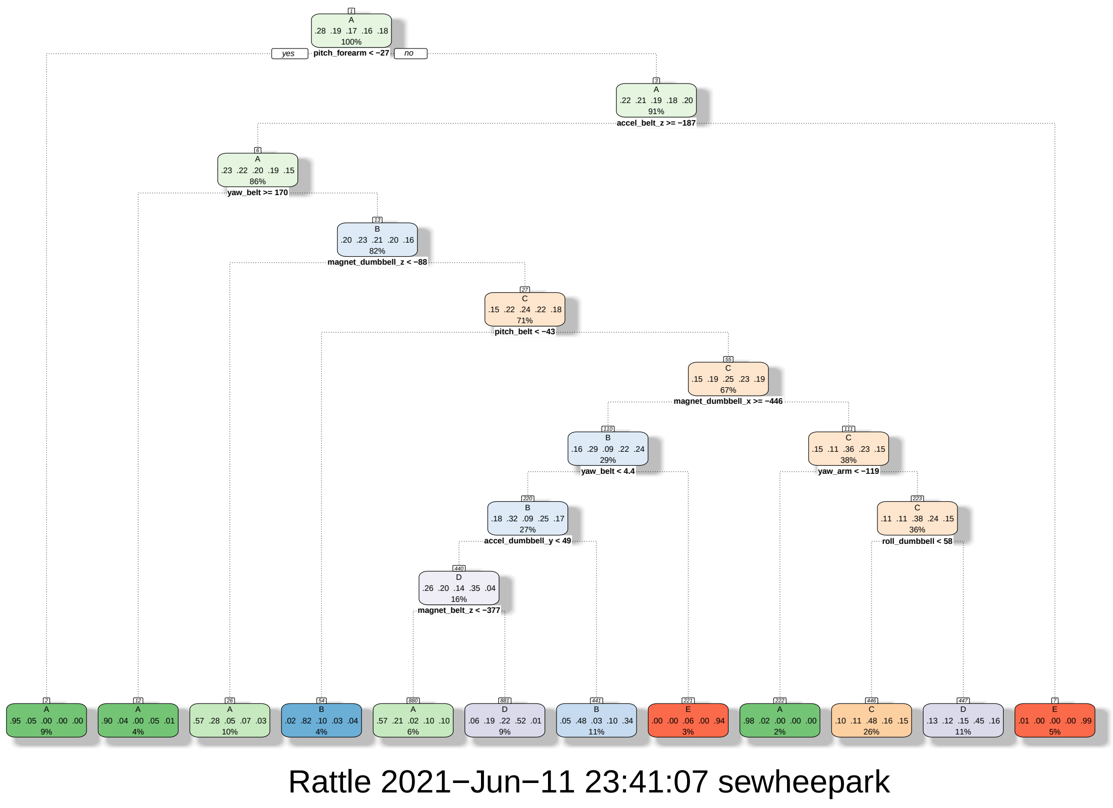
## III. Prediction Model Building

In this section, we will test out different prediction models in order to determine the model with the highest accuracy. The models that we will build are Classification Tree(RP), Random Forest (RF) and Gradient Boosting (GBM)

First, we train the dataset against the observation classe. Next, we will predict the model against the validation dataset and retrieve an "Accuracy" measurement.

### III.A Classification/Decision Tree

```
fitRP <- train(classe ~., method = "rpart", data = training)
fancyRpartPlot(fitRP$finalModel)
```



### III.A.a. Training

```r
predRP <- predict(fitRP, newdata = validation)
cmRP <- confusionMatrix(predRP, factor(validation$classe))
cmRPA <- cmRP$overall[["Accuracy"]]
cmRPA
```

### III.A.b. Predicting

```
## [1] 0.6090226
```

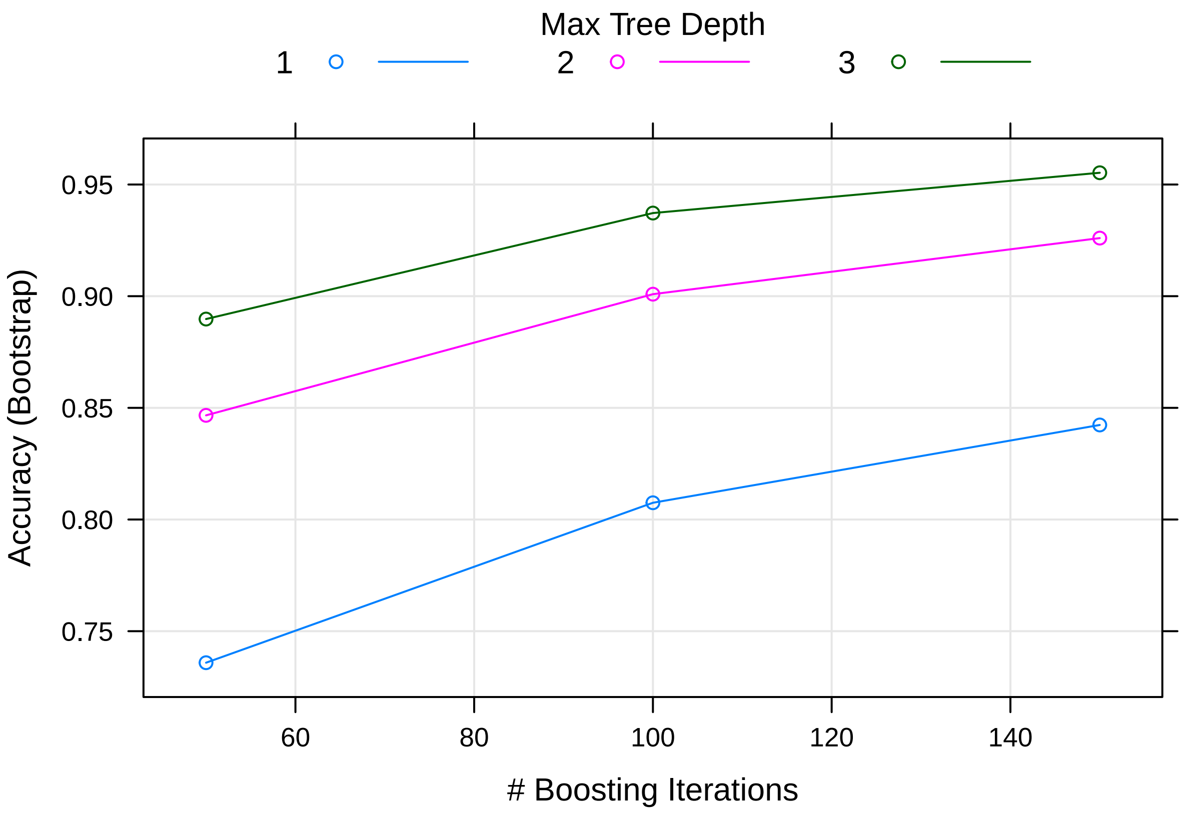
```r
cmRPo <- 1-cmRPA
cmRPo
```

```
## [1] 0.3909774
```

The accuracy for the decision tree model is 0.6090226 and the out-of sample error is 0.3909774.

### III.B. Gradient Boosting (GBM)

```r
fitGBM <- train(classe ~., method = "gbm", data = training, verbose = FALSE)
plot(fitGBM)
```



#### III.B.a Training

```
predGBM <- predict(fitGBM, newdata = validation)
cmGBM <- confusionMatrix(predGBM, factor(validation$classe))
cmGBMA <- cmGBM$overall[["Accuracy"]]
cmGBMA
```

#### III.B.b. Predicting

```
## [1] 0.9716226
```

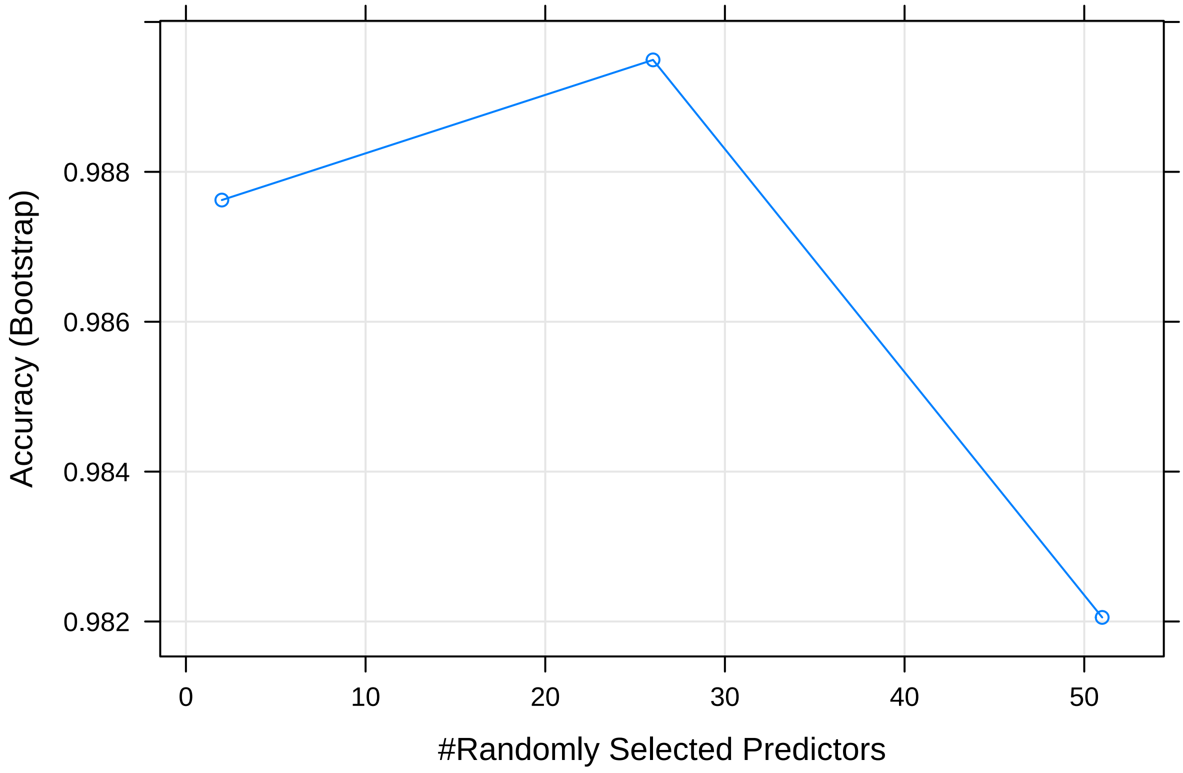
```
cmGBMo <- 1-cmGBMA
cmGBMo
```

```
## [1] 0.0283774
```

The accuracy for the GBM model is 0.9716226. There is an out-of-sample error rate of 0.0283774.

#### Random Forest

```
fitRF <- train(classe ~., method = "rf", data = training)
plot(fitRF)
```



#### III.C.a. Training

```r
predRF <- predict(fitRF, newdata = validation)
cmRF <- confusionMatrix(predRF, factor(validation$classe))
cmRFA <- cmRF$overall[["Accuracy"]]
cmRFA
```

#### III.C.b. Predicting

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

```r
cmRFo <- 1-cmRFA
cmRFo
```

```
## [1] 0
```

The accuracy for the Random Forest Model is very high at 1. This could partially be due to overfitting. The out-of-sample error is therefore 0.

**III.D. The Best Fitting Model** Here, we have identified that the model with the highest Accuracy score is the Random Forest model. Thus, we will use this model in order to predict on the test dataset.

```r
DecisionTree <- cmRP$overall[["Accuracy"]]
GBM <- cmGBM$overall[["Accuracy"]]
RandomForest <- cmRF$overall[["Accuracy"]]
data.frame(DecisionTree, GBM, RandomForest, row.names = "Accuracy")
```

```
##          DecisionTree       GBM RandomForest
## Accuracy    0.6090226 0.9716226            1
```

## IV. Prediction on the Test Dataset

```r
testpred <- predict(fitRF, newdata = testing)
print(testpred)
```

```
##  [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
```

The prediction was performed using the RandomForest model on the untouched dataset "testing". The predictions for the 20 observations are: A B A A E D B A A B C B A E E A B B B .