# Practical Machine Learning Project

***Chris Wilson***

***Friday, March 20, 2015***

## Background

Six young healthy participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: exactly according to the specification (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E).

Read more: http://groupware.les.inf.puc-rio.br/har#weight_lifting_exercises (http://groupware.les.inf.puc-rio.br/har#weight_lifting_exercises)

## Question

The goal is to predict the manner in which they did the exercise from the data supplied. I will initially attempt to use random forests as speed is not currently a concern. NB. Speed wasn't a concern until it took over 1 hour before I cancelled it, I used the forums for better advice on speedier solutions, see special thanks at bottom for links

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.2
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.1.2
```

```
library(memoise) # speed up reading data
```

```
## Warning: package 'memoise' was built under R version 3.1.3
```

```
library(ggplot2)
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.1.3
```

```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.1.3
```

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.1.3
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 3.4.1 Copyright (c) 2006-2014 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(plyr)
```

```
## Warning: package 'plyr' was built under R version 3.1.2
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.1.3
```

```
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
##
## The following object is masked from 'package:stats':
##
##     filter
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.1.3
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.1.3
```

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 3.1.3
```

```
library(parallel); library(doParallel)
```

```
## Warning: package 'doParallel' was built under R version 3.1.3
```

```
## Loading required package: foreach
```

```
## Warning: package 'foreach' was built under R version 3.1.2
```

```
## Loading required package: iterators
```

```
## Warning: package 'iterators' was built under R version 3.1.2
```

```
#registerDoParallel(clust <- makeForkCluster(detectCores()))


set.seed(1)
```

## Obtaining the Data

Download and load the data into R

```
getwd()

r <- memoise(read.csv)
system.time(trainingData<- r("data/pml-training.csv"))
testData<- r("data/pml-testing.csv")
dim(trainingData)
#str(trainingData, list.len = 160)
```

Although I commented out the `str(trainingData)` command I can see there are a number of columns with NA as the values. I will remove these before we continue.

```
sum((colSums(!is.na(trainingData[,-ncol(trainingData)])) < 0.5*nrow(trainingData)))
```

```
## [1] 67
```

```
trainingDataRemovedNACols <- c((colSums(!is.na(trainingData[,-ncol(trainingData)])) >= 0.5*nrow(trainingData)))
trainingData <-trainingData[,trainingDataRemovedNACols]
dim(trainingData)
```

```
## [1] 19622    93
```

Remove columns with with little variance (nearZeroVar) also

```
nzv_cols <-nearZeroVar(trainingData)
if(length(nzv_cols)>0) trainingData <-trainingData[,-nzv_cols]
#str(trainingData)
dim(trainingData)
```

```
## [1] 19622    59
```

Remove irrelevant columns

```
delete_cols <- c('X', 'user_name', 'raw_timestamp_part_1', 'raw_timestamp_part_2', 'cvtd_timestamp', 'new_window', 'num_
window')
trainingData <- trainingData[, -which(names(trainingData) %in% delete_cols)]
dim(trainingData)
```

```
## [1] 19622    53
```

## Split the data

split the training set into 2 so we can cross validate our data before we test it against the `supplied test data` using the `classe` field as y

```
inTrain <- createDataPartition(y=trainingData$classe,
                               p=0.9, list=FALSE)
training <- trainingData[inTrain,]
probe <- trainingData[-inTrain,]
dim(training)
```

```
## [1] 17662    53
```

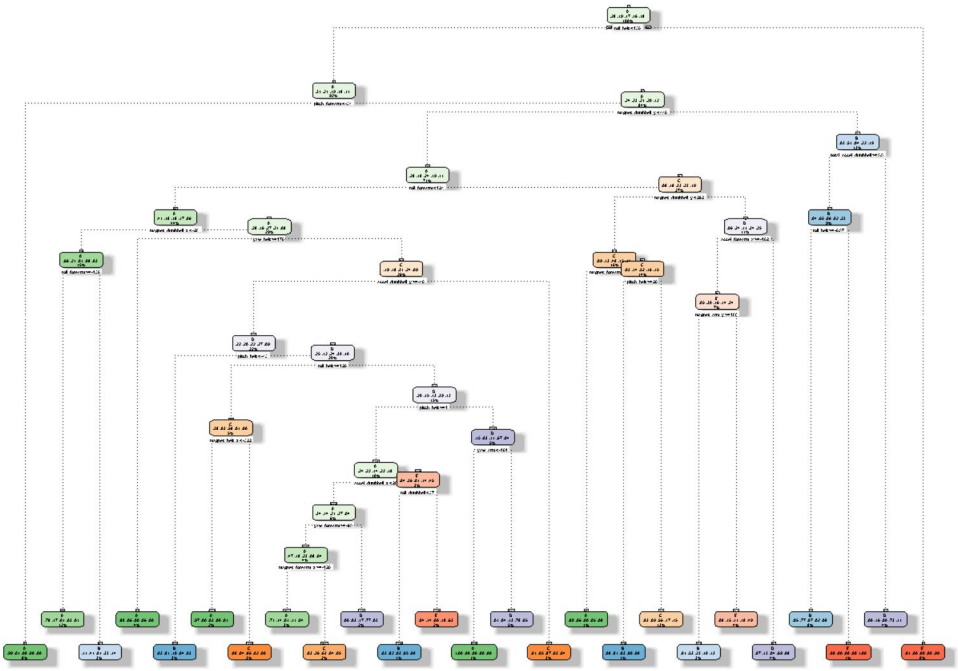
```
#str(training)
```

## Analyse the Data

Use trees for a graphical representation (similiar to decision trees)

```
fit <- rpart(classe~.,data=training)
fancyRpartPlot(fit)
```

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



Rattle 2015-Mar-22 12:18:38 ChrisKat

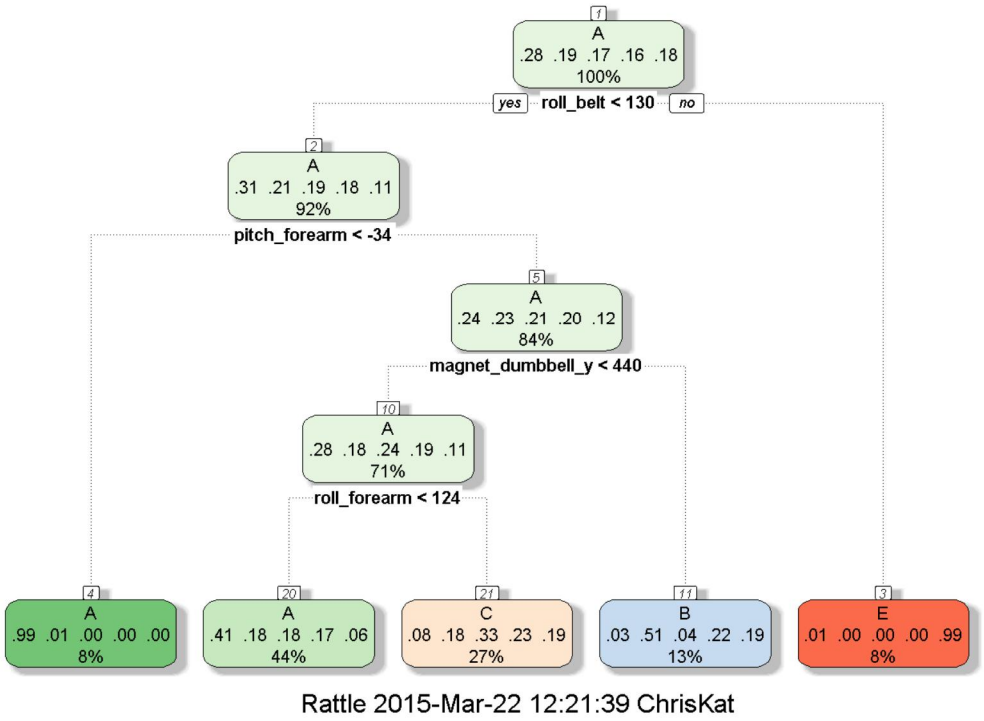
This tree is far too bushy to get any useful information, so I'll fit a model and create the tree again from that

## Build the Models

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

```
## n= 17662
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 17662 12640 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130.5 16190 11180 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -33.65 1419    14 A (0.99 0.0099 0 0 0) *
##      5) pitch_forearm>=-33.65 14771 11166 A (0.24 0.23 0.21 0.2 0.12)
##       10) magnet_dumbbell_y< 439.5 12488  8953 A (0.28 0.18 0.24 0.19 0.11)
##         20) roll_forearm< 123.5 7789  4624 A (0.41 0.18 0.18 0.17 0.06) *
##         21) roll_forearm>=123.5 4699  3146 C (0.079 0.18 0.33 0.23 0.19) *
##       11) magnet_dumbbell_y>=439.5 2283  1119 B (0.031 0.51 0.042 0.22 0.19) *
##    3) roll_belt>=130.5 1472    12 E (0.0082 0 0 0 0.99) *
```

```
fancyRpartPlot(modFit$finalModel)
```




```
rForest=randomForest(classe~.,data=training,ntree=50, importance=TRUE)
rForest
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training, ntree = 50,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.58%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 5015    6    0    1    0 0.001393867
## B   17 3394    6    0    1 0.007021650
## C    0   20 3057    3    0 0.007467532
## D    0    0   38 2855    2 0.013816926
## E    0    1    1    6 3239 0.002463813
```

## Cross validation on the models

```
confusionMatrix(predict(rForest,newdata=probe[,-ncol(probe)]),probe$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 558   1   0   0   0
##          B   0 378   2   0   0
##          C   0   0 340   4   0
##          D   0   0   0 317   0
##          E   0   0   0   0 360
##
## Overall Statistics
##
##                Accuracy : 0.9964
##                  95% CI : (0.9927, 0.9986)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9955
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9974   0.9942   0.9875   1.0000
## Specificity            0.9993   0.9987   0.9975   1.0000   1.0000
## Pos Pred Value         0.9982   0.9947   0.9884   1.0000   1.0000
## Neg Pred Value         1.0000   0.9994   0.9988   0.9976   1.0000
## Prevalence             0.2847   0.1934   0.1745   0.1638   0.1837
## Detection Rate         0.2847   0.1929   0.1735   0.1617   0.1837
## Detection Prevalence   0.2852   0.1939   0.1755   0.1617   0.1837
## Balanced Accuracy      0.9996   0.9980   0.9958   0.9938   1.0000
```

```
testingProbe <- predict(rForest, probe)
#testingProbe
```

a very high accuracy rate of over 99%, I hope this is not overfitted

### Testing Against the Supplied Test Data

```
theBigTest <- predict(rForest, testData)
theBigTest
```

```
##  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
```

### Submitting to Coursera

```
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }}

pml_write_files(theBigTest)
```

All of the tests came back correct :)

`Special Thanks to` https://class.coursera.org/predmachlearn-012/forum/thread?thread_id=61 (https://class.coursera.org/predmachlearn-012/forum/thread?thread_id=61) https://class.coursera.org/predmachlearn-012/forum/thread?thread_id=29 (https://class.coursera.org/predmachlearn-012/forum/thread?thread_id=29) http://www.statmethods.net/advstats/cart.html (http://www.statmethods.net/advstats/cart.html) http://www.quora.com/How-do-random-forests-work-in-laymans-terms (http://www.quora.com/How-do-random-forests-work-in-laymans-terms) http://www.inside-r.org/node/86995 (http://www.inside-r.org/node/86995) http://stackoverflow.com/questions/11330138/find-columns-with-all-missing-values/11330265#11330265 (http://stackoverflow.com/questions/11330138/find-columns-with-all-missing-values/11330265#11330265)

http://stackoverflow.com/questions/28043393/nearzerovar-function-in-caret (http://stackoverflow.com/questions/28043393/nearzerovar-function-in-caret)