Coursera Machine Learning Project

MLovejoy

July 17, 2016

SUMMARY

This project attempts to predict the manner in which individuals did a weight lifting exercise. We know the "classe" variable is the target variable, to be predicted by all other variables. Ultimately we created a Random Forest model that predicts results with over 99% accuracy

LOAD DATA

First we will load the data set and explore it a bit.

```
pmltraining <- read.csv("pml-training.csv", header=TRUE)
pmltesting <- read.csv("pml-testing.csv", header=TRUE)
dim(pmltraining)
```

```
## [1] 19622   160
```

```
dim(pmltesting)
```

```
## [1]  20 160
```

```
head(pmltraining$classe)
```

```
## [1] A A A A A A
## Levels: A B C D E
```

```
str(pmltraining)
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                        : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name                : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2
 2 2 2 ...
##  $ raw_timestamp_part_1     : int  1323084231 1323084231 1323084231 1323084232 1323084
232 1323084232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2     : int  788290 808298 820366 120339 196328 304277 368296 44
0390 484323 484434 ...
##  $ cvtd_timestamp           : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9
 9 9 9 ...
##  $ new_window               : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window               : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt                : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45
 ...
##  $ pitch_belt               : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17
 ...
##  $ yaw_belt                 : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -9
4.4 -94.4 ...
##  $ total_accel_belt         : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt       : Factor w/ 397 levels "","-0.016850",..: 1 1 1 1 1 1 1 1
 1 1 ...
##  $ kurtosis_picth_belt      : Factor w/ 317 levels "","-0.021887",..: 1 1 1 1 1 1 1 1
 1 1 ...
##  $ kurtosis_yaw_belt        : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt       : Factor w/ 395 levels "","-0.003095",..: 1 1 1 1 1 1 1 1
 1 1 ...
##  $ skewness_roll_belt.1     : Factor w/ 338 levels "","-0.005928",..: 1 1 1 1 1 1 1 1
 1 1 ...
##  $ skewness_yaw_belt        : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ max_roll_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt           : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt             : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1
 1 1 ...
##  $ min_roll_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_belt           : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_belt             : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1
 1 1 ...
##  $ amplitude_roll_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt     : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt       : Factor w/ 4 levels "","#DIV/0!","0.00",..: 1 1 1 1 1 1 1
 1 1 1 ...
##  $ var_total_accel_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_belt        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_belt             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_belt             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_belt_x             : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y             : num  0 0 0 0 0.02 0 0 0 0 0 ...
```

```
##  $ gyros_belt_z            : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.
02 0 ...
##  $ accel_belt_x            : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y            : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z            : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x           : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y           : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z           : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308
 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128
 ...
##  $ pitch_arm               : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm                 : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161
 ...
##  $ total_accel_arm         : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ var_accel_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_arm_x             : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y             : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -
0.03 ...
##  $ gyros_arm_z             : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x             : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288
 ...
##  $ accel_arm_y             : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z             : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124
 ...
##  $ magnet_arm_x            : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376
 ...
##  $ magnet_arm_y            : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z            : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ kurtosis_roll_arm       : Factor w/ 330 levels "","-0.02438",..: 1 1 1 1 1 1 1 1 1
 1 ...
##  $ kurtosis_picth_arm      : Factor w/ 328 levels "","-0.00484",..: 1 1 1 1 1 1 1 1 1
 1 ...
##  $ kurtosis_yaw_arm        : Factor w/ 395 levels "","-0.01548",..: 1 1 1 1 1 1 1 1 1
 1 ...
##  $ skewness_roll_arm       : Factor w/ 331 levels "","-0.00051",..: 1 1 1 1 1 1 1 1 1
 1 ...
##  $ skewness_pitch_arm      : Factor w/ 328 levels "","-0.00184",..: 1 1 1 1 1 1 1 1 1
 1 ...
##  $ skewness_yaw_arm        : Factor w/ 395 levels "","-0.00311",..: 1 1 1 1 1 1 1 1 1
 1 ...
##  $ max_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
```

```
##  $ min_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm       : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ roll_dumbbell           : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell          : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell            : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ kurtosis_roll_dumbbell  : Factor w/ 398 levels "","-0.0035","-0.0073",..: 1 1 1 1
 1 1 1 1 1 1 ...
##  $ kurtosis_picth_dumbbell : Factor w/ 401 levels "","-0.0163","-0.0233",..: 1 1 1 1
 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_dumbbell   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_dumbbell  : Factor w/ 401 levels "","-0.0082","-0.0096",..: 1 1 1 1
 1 1 1 1 1 1 ...
##  $ skewness_pitch_dumbbell : Factor w/ 402 levels "","-0.0053","-0.0084",..: 1 1 1 1
 1 1 1 1 1 1 ...
##  $ skewness_yaw_dumbbell   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ max_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_dumbbell        : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1
 1 1 ...
##  $ min_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell        : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1
 1 1 ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##    [list output truncated]
```

We can see that it has 160 fields and under 20,000 records. Some of the fields seem to have many NAs and some are identifying fields that should not be used to predict.

We will load the several packages needed:

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.5
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.2.4
```

```
library(ggplot2)
library(rattle)
```

```
## Warning: Failed to load RGtk2 dynamic library, attempting to install it.
```

```
## Please install GTK+ from http://r.research.att.com/libs/GTK_2.24.17-X11.pkg
```

```
## If the package still does not load, please ensure that GTK+ is installed and that it
 is on your PATH environment variable
```

```
## IN ANY CASE, RESTART R BEFORE TRYING TO LOAD THE PACKAGE AGAIN
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

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

```
## Warning: package 'rpart.plot' was built under R version 3.2.5
```

```
library(gbm)
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: splines
```

```
## Loading required package: parallel
```

```
## Loaded gbm 2.1.1
```

DATA PREPARATION

We can see that the first columns of data are index fields, names, and timestamps, so we will remove those unnecessary fields. However, we still have 154 possible predictor variables, so we need to perform some dimension reduction on this data set. This dimension reduction will improve processing time for the models and reduce multi-collinearity from correlated variables. To accomplish this, we will use the Near Zero Variance function in caret to remove these unnecessary variables.

```
pmltraining <- pmltraining[, -(1:5)]
manyNA <- sapply(pmltraining, function(x) mean(is.na(x))) > 0.95
pmltraining <- pmltraining[ , manyNA == F]

nzv <- nearZeroVar(pmltraining, saveMetrics = TRUE)
pmltraining_nzv <-  pmltraining[ , nzv$nzv==FALSE]
dim(pmltraining_nzv)
```

```
## [1] 19622    54
```

```
summary(pmltraining_nzv$classe)
```

```
##    A    B    C    D    E 
## 5580 3797 3422 3216 3607
```

```
str(pmltraining_nzv)
```

```
## 'data.frame':    19622 obs. of  54 variables:
##  $ num_window          : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt           : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt          : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt            : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -
94.4 ...
##  $ total_accel_belt    : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ gyros_belt_x        : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y        : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z        : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0
 ...
##  $ accel_belt_x        : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y        : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z        : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x       : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y       : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z       : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm           : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm             : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm     : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ gyros_arm_x         : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y         : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03
 ...
##  $ gyros_arm_z         : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x         : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y         : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z         : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
##  $ magnet_arm_x        : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ magnet_arm_y        : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z        : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ roll_dumbbell       : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell      : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell        : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -
0.02 ...
##  $ gyros_dumbbell_z    : num  0 0 0 -0.02 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x    : int  -234 -233 -232 -232 -233 -234 -232 -234 -232 -235 ...
##  $ accel_dumbbell_y    : int  47 47 46 48 48 48 47 46 47 48 ...
##  $ accel_dumbbell_z    : int  -271 -269 -270 -269 -270 -269 -270 -272 -269 -270 ...
##  $ magnet_dumbbell_x   : int  -559 -555 -561 -552 -554 -558 -551 -555 -549 -558 ...
##  $ magnet_dumbbell_y   : int  293 296 298 303 292 294 295 300 292 291 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -69 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.9 27.9 27.8 27.7 27.7 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -
63.8 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -152 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.03 0.02 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 ...
```

```
## $ accel_forearm_x     : int  192 192 196 189 189 193 195 193 193 190 ...
## $ accel_forearm_y     : int  203 203 204 206 206 203 205 205 204 205 ...
## $ accel_forearm_z     : int  -215 -216 -213 -214 -214 -215 -215 -213 -214 -215 ...
## $ magnet_forearm_x    : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
## $ magnet_forearm_y    : num  654 661 658 658 655 660 659 660 653 656 ...
## $ magnet_forearm_z    : num  476 473 469 469 473 478 470 474 476 473 ...
## $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1
 ...
```

No variables had zero variance, but 60 variables had near-zero variance and are now removed from the training data set. We are left with 54 variables, including the classe target variable.

Next we will partition the pmltraining data into training and testing sets. This pmltraining data set is meant for both training and testing models, as the separate pmltesting data set is reserved for evaluating the models for the purposes of the project.

```
inTrain <- createDataPartition(y = pmltraining_nzv$classe, p=0.7, list=FALSE)
mytrain <- pmltraining_nzv[inTrain, ]
mytest <- pmltraining_nzv[-inTrain, ]

dim(mytrain); dim(mytest)
```

```
## [1] 13737    54
```

```
## [1] 5885   54
```

We now have the training set (consisting of 13,737 records) and the testing set (consisting of 5,885 records). Because of these fairly large partition sizes, I expect to have fairly small Out of Sample Error rates.

MACHINE LEARNING ALGORITHMS

1. Decision Tree

We'll start with applying a Decision Tree because it can handle missing variables and outliers, and identify the most important variables for predicting Class E.

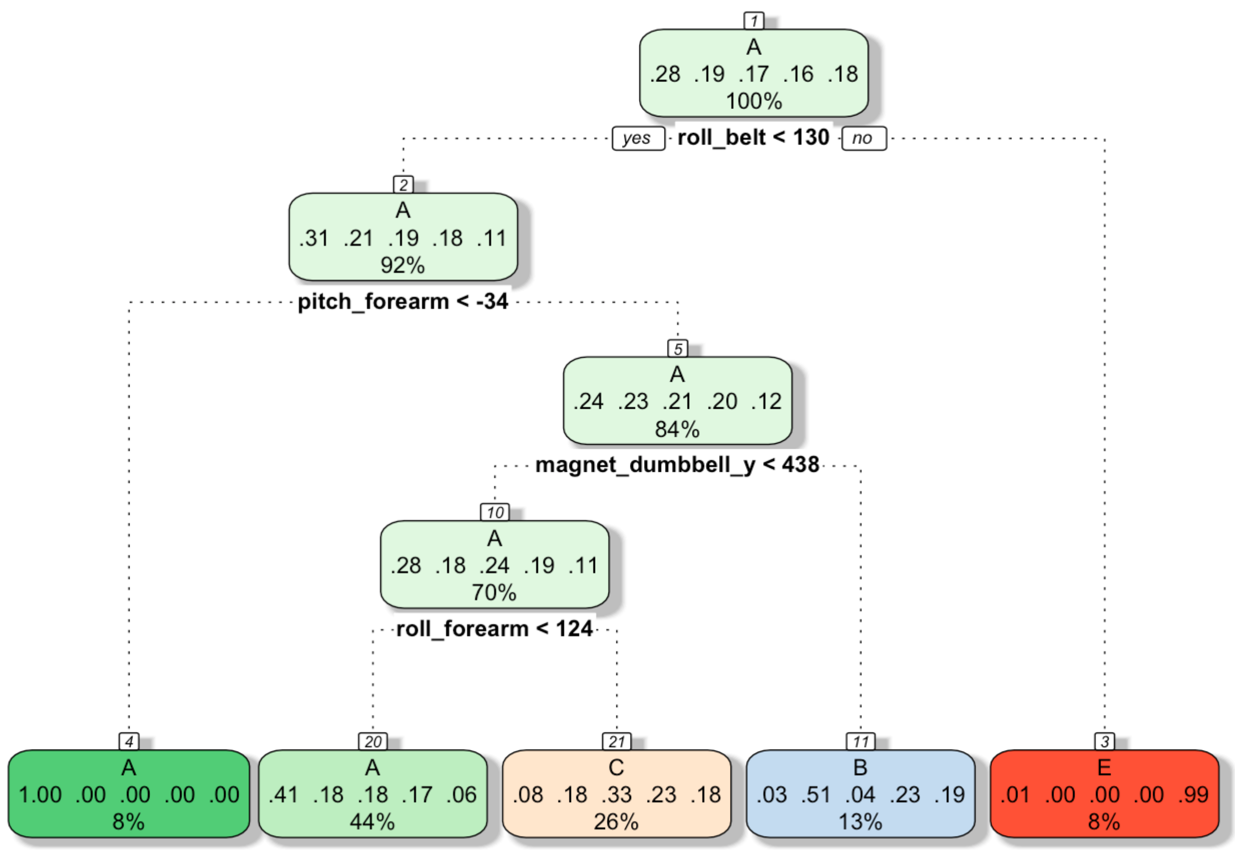
Training

```
set.seed(62384)

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

```
## n= 13737
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 13737 9831 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130.5 12576 8679 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -33.95 1105    5 A (1 0.0045 0 0 0) *
##      5) pitch_forearm>=-33.95 11471 8674 A (0.24 0.23 0.21 0.2 0.12)
##       10) magnet_dumbbell_y< 438.5 9670 6936 A (0.28 0.18 0.24 0.19 0.11)
##         20) roll_forearm< 123.5 6038 3590 A (0.41 0.18 0.18 0.17 0.063) *
##         21) roll_forearm>=123.5 3632 2420 C (0.079 0.18 0.33 0.23 0.18) *
##       11) magnet_dumbbell_y>=438.5 1801  880 B (0.035 0.51 0.042 0.23 0.19) *
##    3) roll_belt>=130.5 1161    9 E (0.0078 0 0 0 0.99) *
```

```
fancyRpartPlot(modFit_dt$finalModel)
```



Rattle 2016-Jul-17 20:58:28 michaellovejoy

Testing

```
pred_dt <- predict(modFit_dt, newdata = mytest)

confusionMatrix(pred_dt, mytest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1528  477  478  440  141
##          B   24  372   33  165  156
##          C  117  290  515  359  306
##          D    0    0    0    0    0
##          E    5    0    0    0  479
##
## Overall Statistics
##
##                Accuracy : 0.4918
##                  95% CI : (0.4789, 0.5046)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3357
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9128  0.32660  0.50195   0.0000  0.44270
## Specificity            0.6352  0.92035  0.77938   1.0000  0.99896
## Pos Pred Value         0.4987  0.49600  0.32451      NaN  0.98967
## Neg Pred Value         0.9482  0.85063  0.88111   0.8362  0.88835
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2596  0.06321  0.08751   0.0000  0.08139
## Detection Prevalence   0.5206  0.12744  0.26967   0.0000  0.08224
## Balanced Accuracy      0.7740  0.62348  0.64066   0.5000  0.72083
```

We can see that this model has very poor results - just better than tossing a fair coin. This model yields a very large Out of Sample Error, which is most likely due to overfitting. Let's try another model.

2. Random Forest

We'll now run a Random Forest model, using Train Control to reduce processing time.

Training

```
set.seed(62384)

modFit_rf <- train(classe ~ ., data = mytrain, method = "rf", ntree = 10, trControl = tr
ainControl(method = "cv"))
```

```
## Loading required package: randomForest
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
print(modFit_rf$finalModel)
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 10, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 10
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 2.14%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3834   20    5    6    1 0.008277289
## B   35 2540   35   11   10 0.034587609
## C    6   47 2296   14    6 0.030814690
## D    9   11   24 2178    8 0.023318386
## E    4    9   10   20 2456 0.017206883
```

Testing

```
pred_rf <- predict(modFit_rf, newdata = mytest)

confusionMatrix(pred_rf, mytest$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    3    0    0    0
##          B    2 1133    8    0    0
##          C    0    3 1018    6    0
##          D    0    0    0  958    3
##          E    0    0    0    0 1079
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9958          
##                  95% CI : (0.9937, 0.9972)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9946          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9947   0.9922   0.9938   0.9972
## Specificity            0.9993   0.9979   0.9981   0.9994   1.0000
## Pos Pred Value         0.9982   0.9913   0.9912   0.9969   1.0000
## Neg Pred Value         0.9995   0.9987   0.9984   0.9988   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1925   0.1730   0.1628   0.1833
## Detection Prevalence   0.2846   0.1942   0.1745   0.1633   0.1833
## Balanced Accuracy      0.9990   0.9963   0.9952   0.9966   0.9986
```

This model has over 99% accuracy, which is pretty good! But let's still try another one.

3. Generalized Boosted Regression

We'll also run a Generalized Boosted Regression Model (GBM).

Training

```
set.seed(62384)

fitControl <- trainControl(method = "repeatedcv", number = 10, repeats = 10)

modFit_gbm <- train(classe ~ ., data = mytrain, method = "gbm", trControl = fitControl,
verbose = FALSE)
```

```
## Loading required package: plyr
```

```
print(modFit_gbm$finalModel)
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 41 had non-zero influence.
```

Testing

```
pred_gbm <- predict(modFit_gbm, newdata = mytest)

confusionMatrix(pred_gbm, mytest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1668   18    0    0    0
##          B    4 1103    9    1    9
##          C    0   15 1014   12    2
##          D    1    0    2  951   18
##          E    1    3    1    0 1053
##
## Overall Statistics
##
##                Accuracy : 0.9837
##                  95% CI : (0.9801, 0.9868)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9794
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9684   0.9883   0.9865   0.9732
## Specificity            0.9957   0.9952   0.9940   0.9957   0.9990
## Pos Pred Value         0.9893   0.9796   0.9722   0.9784   0.9953
## Neg Pred Value         0.9986   0.9924   0.9975   0.9974   0.9940
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2834   0.1874   0.1723   0.1616   0.1789
## Detection Prevalence   0.2865   0.1913   0.1772   0.1652   0.1798
## Balanced Accuracy      0.9961   0.9818   0.9912   0.9911   0.9861
```

This accuracy is also very good. But as we can see from the 3 models, the Random Forest model yielded the most accurate results, with GBM very close behind.

```
## pred_20
## A B C D E
## 7 8 1 1 3
```

Citation:

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz4EXCjRVDj (http://groupware.les.inf.puc-rio.br/har#ixzz4EXCjRVDj)