Practical Machine Learning Project

Darrell Hill

May 24, 2019

Overview

Fitness tools like Jawbone Up, Nike FuelBand, and Fitbit make it possible to collect a large amount of data about fitness activity relatively inexpensively. As a result, people can see what they are doing and understand a lot more about how long it takes them do it. But they don't know how well they are doing it. So this project is to evaluate performance from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants. They did the exercise correctly and incorrectly 5 different ways.

The goal is to predict the manner in which they did the exercise ("classe" variable). This report walks through how the model was built, how cross validation was used, predicted sample error, and explain why decisions were made. Then we will predict 20 test cases.

Loading the Libraries

```
if ( !require(MASS     ) ) { install.packages('MASS');      library(MASS)      }
```

```
## Loading required package: MASS
```

```
if ( !require(tidyverse) ) { install.packages('tidyverse'); library(tidyverse) }
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages ---------------------------------------------------------------
---- tidyverse 1.2.1 --
```

```
## v ggplot2 3.0.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.6
## v tidyr   0.8.1     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
```

```
## -- Conflicts --------------------------------------------------------------------- t
idyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::select() masks MASS::select()
```

```
if ( !require(broom    ) ) { install.packages('broom');     library(broom)     }
```

```
## Loading required package: broom
```

```
if ( !require(caret    ) ) { install.packages('caret');     library(caret)     }
```

```
## Loading required package: caret
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
if ( !require(rpart    ) ) { install.packages('rpart');     library(rpart)     }
```

```
## Loading required package: rpart
```

```
if ( !require(randomForest    ) ) { install.packages('randomForest');     library(randomFores
t)     }
```

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

```
## randomForest 4.6-14
```

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

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

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

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

```
if ( !require(rpart.plot    ) ) { install.packages('rpart.plot');     library(rpart.plot)
}
```

```
## Loading required package: rpart.plot
```

```
if ( !require(repmis    ) ) { install.packages('repmis');     library(repmis)     }
```

```
## Loading required package: repmis
```

```
if ( !require(rattle    ) ) { install.packages('rattle');     library(rattle)     }
```

```
## Loading required package: rattle
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.1.0 Copyright (c) 2006-2017 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
##
## Attaching package: 'rattle'
```

```
## The following object is masked from 'package:randomForest':
##
##     importance
```

```
if ( !require(corrplot    ) ) { install.packages('corrplot');     library(corrplot)     }
```

```
## Loading required package: corrplot
```

```
## corrplot 0.84 loaded
```

```
if ( !require(gbm    ) ) { install.packages('gbm');     library(gbm)     }
```

```
## Loading required package: gbm
```

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

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

```
## The following object is masked from 'package:rpart':
##
##     solder
```

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

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

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

```
## Loaded gbm 2.1.3
```

```
if ( !require(e1071    ) ) { install.packages('e1071');     library(e1071)     }
```

```
## Loading required package: e1071
```

Loading the Data

Here we load the training and test variables. The test variable is used to validate the model.

```
TrainingData <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-trainin
g.csv"),header=TRUE)
str(TrainingData)
```

```
## '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 1323084232 1
323084232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390
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 -94.4 -9
4.4 ...
##  $ 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 -0.0
3 ...
##  $ 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]
```

```
TestingData  <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testin
g.csv"),header=TRUE)
str(TestingData)
```

```
## 'data.frame':    20 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 6 5 5 1 4 5 5 5 2
3 ...
##  $ raw_timestamp_part_1    : int  1323095002 1322673067 1322673075 1322832789 1322489635 1
322673149 1322673128 1322673076 1323084240 1322837822 ...
##  $ raw_timestamp_part_2    : int  868349 778725 342967 560311 814776 510661 766645 54671 9
16313 384285 ...
##  $ cvtd_timestamp          : Factor w/ 11 levels "02/12/2011 13:33",..: 5 10 10 1 6 11 11
10 3 2 ...
##  $ new_window              : Factor w/ 1 level "no": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window              : int  74 431 439 194 235 504 485 440 323 664 ...
##  $ roll_belt               : num  123 1.02 0.87 125 1.35 -5.92 1.2 0.43 0.93 114 ...
##  $ pitch_belt              : num  27 4.87 1.82 -41.6 3.33 1.59 4.44 4.15 6.72 22.4 ...
##  $ yaw_belt                : num  -4.75 -88.9 -88.5 162 -88.6 -87.7 -87.3 -88.5 -93.7 -13.
1 ...
##  $ total_accel_belt        : int  20 4 5 17 3 4 4 4 4 18 ...
##  $ kurtosis_roll_belt      : logi  NA NA NA NA NA NA ...
##  $ kurtosis_picth_belt     : logi  NA NA NA NA NA NA ...
##  $ kurtosis_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt      : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt.1    : logi  NA NA NA NA NA NA ...
##  $ skewness_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ max_roll_belt           : logi  NA NA NA NA NA NA ...
##  $ max_picth_belt          : logi  NA NA NA NA NA NA ...
##  $ max_yaw_belt            : logi  NA NA NA NA NA NA ...
##  $ min_roll_belt           : logi  NA NA NA NA NA NA ...
##  $ min_pitch_belt          : logi  NA NA NA NA NA NA ...
##  $ min_yaw_belt            : logi  NA NA NA NA NA NA ...
##  $ amplitude_roll_belt     : logi  NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt    : logi  NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt      : logi  NA NA NA NA NA NA ...
##  $ var_total_accel_belt    : logi  NA NA NA NA NA NA ...
##  $ avg_roll_belt           : logi  NA NA NA NA NA NA ...
##  $ stddev_roll_belt        : logi  NA NA NA NA NA NA ...
##  $ var_roll_belt           : logi  NA NA NA NA NA NA ...
##  $ avg_pitch_belt          : logi  NA NA NA NA NA NA ...
##  $ stddev_pitch_belt       : logi  NA NA NA NA NA NA ...
##  $ var_pitch_belt          : logi  NA NA NA NA NA NA ...
##  $ avg_yaw_belt            : logi  NA NA NA NA NA NA ...
##  $ stddev_yaw_belt         : logi  NA NA NA NA NA NA ...
##  $ var_yaw_belt            : logi  NA NA NA NA NA NA ...
##  $ gyros_belt_x            : num  -0.5 -0.06 0.05 0.11 0.03 0.1 -0.06 -0.18 0.1 0.14 ...
##  $ gyros_belt_y            : num  -0.02 -0.02 0.02 0.11 0.02 0.05 0 -0.02 0 0.11 ...
##  $ gyros_belt_z            : num  -0.46 -0.07 0.03 -0.16 0 -0.13 0 -0.03 -0.02 -0.16 ...
##  $ accel_belt_x            : int  -38 -13 1 46 -8 -11 -14 -10 -15 -25 ...
##  $ accel_belt_y            : int  69 11 -1 45 4 -16 2 -2 1 63 ...
##  $ accel_belt_z            : int  -179 39 49 -156 27 38 35 42 32 -158 ...
##  $ magnet_belt_x           : int  -13 43 29 169 33 31 50 39 -6 10 ...
##  $ magnet_belt_y           : int  581 636 631 608 566 638 622 635 600 601 ...
##  $ magnet_belt_z           : int  -382 -309 -312 -304 -418 -291 -315 -305 -302 -330 ...
```

```
##  $ roll_arm                : num  40.7 0 0 -109 76.1 0 0 0 -137 -82.4 ...
##  $ pitch_arm               : num  -27.8 0 0 55 2.76 0 0 0 11.2 -63.8 ...
##  $ yaw_arm                 : num  178 0 0 -142 102 0 0 0 -167 -75.3 ...
##  $ total_accel_arm         : int  10 38 44 25 29 14 15 22 34 32 ...
##  $ var_accel_arm           : logi  NA NA NA NA NA NA ...
##  $ avg_roll_arm            : logi  NA NA NA NA NA NA ...
##  $ stddev_roll_arm         : logi  NA NA NA NA NA NA ...
##  $ var_roll_arm            : logi  NA NA NA NA NA NA ...
##  $ avg_pitch_arm           : logi  NA NA NA NA NA NA ...
##  $ stddev_pitch_arm        : logi  NA NA NA NA NA NA ...
##  $ var_pitch_arm           : logi  NA NA NA NA NA NA ...
##  $ avg_yaw_arm             : logi  NA NA NA NA NA NA ...
##  $ stddev_yaw_arm          : logi  NA NA NA NA NA NA ...
##  $ var_yaw_arm             : logi  NA NA NA NA NA NA ...
##  $ gyros_arm_x             : num  -1.65 -1.17 2.1 0.22 -1.96 0.02 2.36 -3.71 0.03 0.26 ...
##  $ gyros_arm_y             : num  0.48 0.85 -1.36 -0.51 0.79 0.05 -1.01 1.85 -0.02 -0.
5 ...
##  $ gyros_arm_z             : num  -0.18 -0.43 1.13 0.92 -0.54 -0.07 0.89 -0.69 -0.02 0.7
9 ...
##  $ accel_arm_x             : int  16 -290 -341 -238 -197 -26 99 -98 -287 -301 ...
##  $ accel_arm_y             : int  38 215 245 -57 200 130 79 175 111 -42 ...
##  $ accel_arm_z             : int  93 -90 -87 6 -30 -19 -67 -78 -122 -80 ...
##  $ magnet_arm_x            : int  -326 -325 -264 -173 -170 396 702 535 -367 -420 ...
##  $ magnet_arm_y            : int  385 447 474 257 275 176 15 215 335 294 ...
##  $ magnet_arm_z            : int  481 434 413 633 617 516 217 385 520 493 ...
##  $ kurtosis_roll_arm       : logi  NA NA NA NA NA NA ...
##  $ kurtosis_picth_arm      : logi  NA NA NA NA NA NA ...
##  $ kurtosis_yaw_arm        : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_arm       : logi  NA NA NA NA NA NA ...
##  $ skewness_pitch_arm      : logi  NA NA NA NA NA NA ...
##  $ skewness_yaw_arm        : logi  NA NA NA NA NA NA ...
##  $ max_roll_arm            : logi  NA NA NA NA NA NA ...
##  $ max_picth_arm           : logi  NA NA NA NA NA NA ...
##  $ max_yaw_arm             : logi  NA NA NA NA NA NA ...
##  $ min_roll_arm            : logi  NA NA NA NA NA NA ...
##  $ min_pitch_arm           : logi  NA NA NA NA NA NA ...
##  $ min_yaw_arm             : logi  NA NA NA NA NA NA ...
##  $ amplitude_roll_arm      : logi  NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm     : logi  NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm       : logi  NA NA NA NA NA NA ...
##  $ roll_dumbbell           : num  -17.7 54.5 57.1 43.1 -101.4 ...
##  $ pitch_dumbbell          : num  25 -53.7 -51.4 -30 -53.4 ...
##  $ yaw_dumbbell            : num  126.2 -75.5 -75.2 -103.3 -14.2 ...
##  $ kurtosis_roll_dumbbell  : logi  NA NA NA NA NA NA ...
##  $ kurtosis_picth_dumbbell : logi  NA NA NA NA NA NA ...
##  $ kurtosis_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_dumbbell  : logi  NA NA NA NA NA NA ...
##  $ skewness_pitch_dumbbell : logi  NA NA NA NA NA NA ...
##  $ skewness_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ max_roll_dumbbell       : logi  NA NA NA NA NA NA ...
##  $ max_picth_dumbbell      : logi  NA NA NA NA NA NA ...
```

```
##  $ max_yaw_dumbbell        : logi  NA NA NA NA NA NA ...
##  $ min_roll_dumbbell       : logi  NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell      : logi  NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell        : logi  NA NA NA NA NA NA ...
##  $ amplitude_roll_dumbbell : logi  NA NA NA NA NA NA ...
##   [list output truncated]
```

There are 160 variables. The training data has 19622 observations. The testing data has 20 observations (to do the predictions).

Cleaning the Data

Since there are a lot of NAs, we should remove them to ensure the models are as accurate as possible and run correctly.

```
CleanTrainData  <-  TrainingData[, colSums(is.na(TrainingData)) == 0]
CleanTestData  <-  TestingData[, colSums(is.na(TestingData)) == 0]
dim(CleanTrainData)
```

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

```
dim(CleanTestData)
```

```
## [1] 20 60
```

Cleaning the data leaves us with 93 variables left over out of the 19622 observations in the training data set and 60 variables left over in the testing data set.

We remove the first 7 variables because of a lack of impact on classe.

```
CleanTrainData  <-  CleanTrainData[, -c(1:7)]
CleanTestData  <-  CleanTestData[, -c(1:7)]
```

That leaves us with 86 variables of the clean training data set and 53 variables/columns of the test data set.

Data Prediction Prep

I'll be setting up the data in to a rough 2/3 split of 65% training data and 35% testing data. Splitting it up makes it possible to calculate out-of-sample errors too.

```
set.seed(1234)
TrainSet <- createDataPartition(CleanTrainData$classe, p = 0.65, list = FALSE)
WithinTrainingData <- CleanTrainData[TrainSet, ]
WithinTestData <- CleanTrainData[-TrainSet, ]
dim(WithinTrainingData)
```

```
## [1] 12757    86
```

```
dim(WithinTestData)
```

```
## [1] 6865   86
```

This leaves us with 12757 observations in the training set and 6865 in the test data set.

Removing variables/columns that have near-zero variance will also help to further get the data ready to to prediction analysis.

```
nearZeroVar <- nearZeroVar(WithinTrainingData)
WithinTrainingData <- WithinTrainingData[, -nearZeroVar]
WithinTestData  <- WithinTestData[, -nearZeroVar]
dim(WithinTrainingData)
```

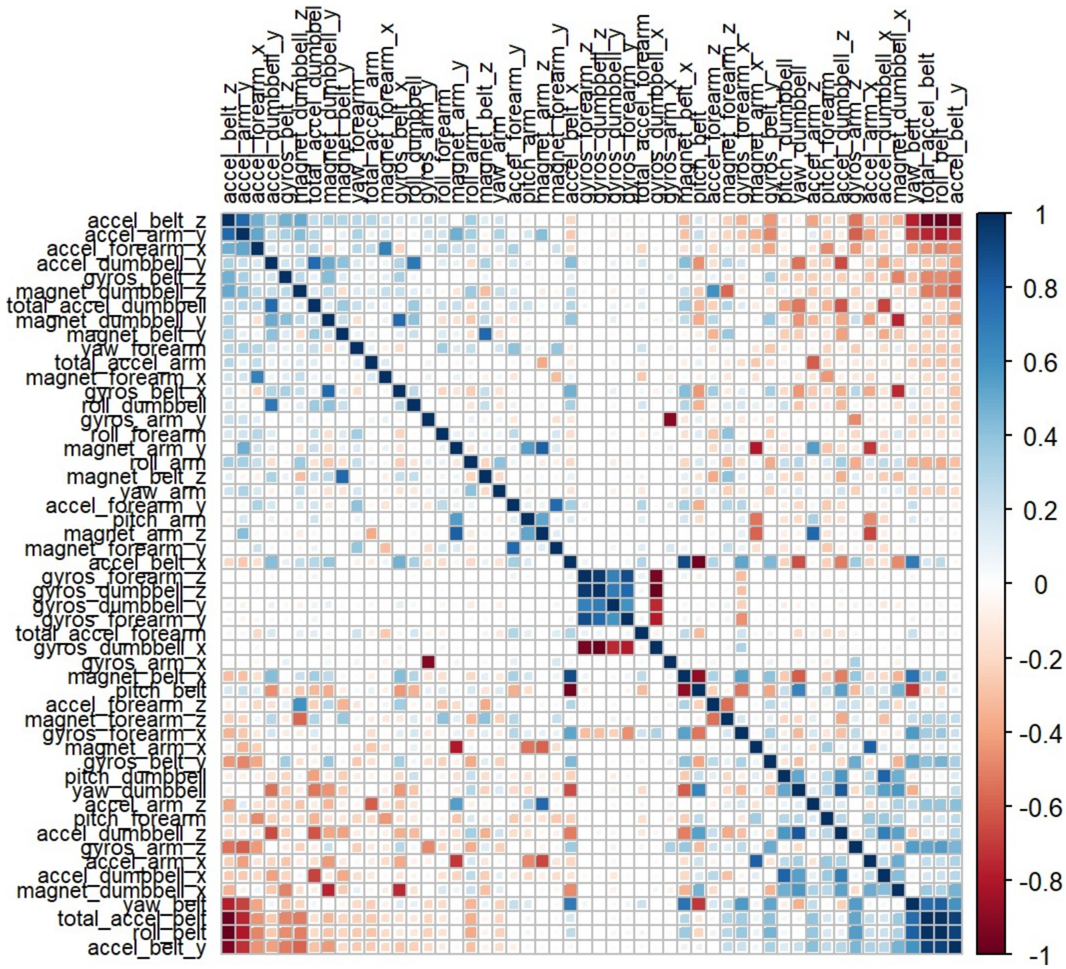
```
## [1] 12757    53
```

Within Training data set has 53 variables/columns.

Correlation Plot

The Corrplot package has several options to selection to see which variables that have relationships to each other and how they are graphed. The type is set to the default of “full”, and by seeing the colors using square it shows relationships easier. The order is set to for first principle component (FPC).

```
correlation_matrix <- cor(WithinTrainingData[, -53])
corrplot(correlation_matrix, order = "FPC", method = "square", type = "full",
         tl.cex = 0.7, tl.col = rgb(0, 0, 0))
```



Some relationships that are highly negatively correlated include roll_belt with accel_belt_z; total_accel_belt with accel_arm_y; and others. There are many variables which show positive and negative correlations based on natural expected relationships.

While that provided a good graphical relationship of the variables to see how they all relate to each other, below is an easy way to see what those are. We find 20, as follows:

```
CorrelatedVariables = findCorrelation(correlation_matrix, cutoff=0.7)
names(WithinTrainingData)[CorrelatedVariables]
```

```
##  [1] "accel_belt_z"      "roll_belt"         "accel_belt_y"
##  [4] "total_accel_belt"  "yaw_belt"          "accel_dumbbell_z"
##  [7] "accel_belt_x"      "pitch_belt"        "magnet_dumbbell_x"
## [10] "accel_dumbbell_y"  "magnet_dumbbell_y" "accel_arm_x"
## [13] "accel_dumbbell_x"  "accel_arm_z"       "magnet_arm_y"
## [16] "magnet_belt_z"     "accel_forearm_y"   "gyros_forearm_y"
## [19] "gyros_dumbbell_x"  "gyros_dumbbell_z"  "gyros_arm_x"
```

Testing the Models

In the following sections, we will test 3 different modeling techniques to see how well they perfor: classification tree, random forest, and gradient boosting method (GBM).

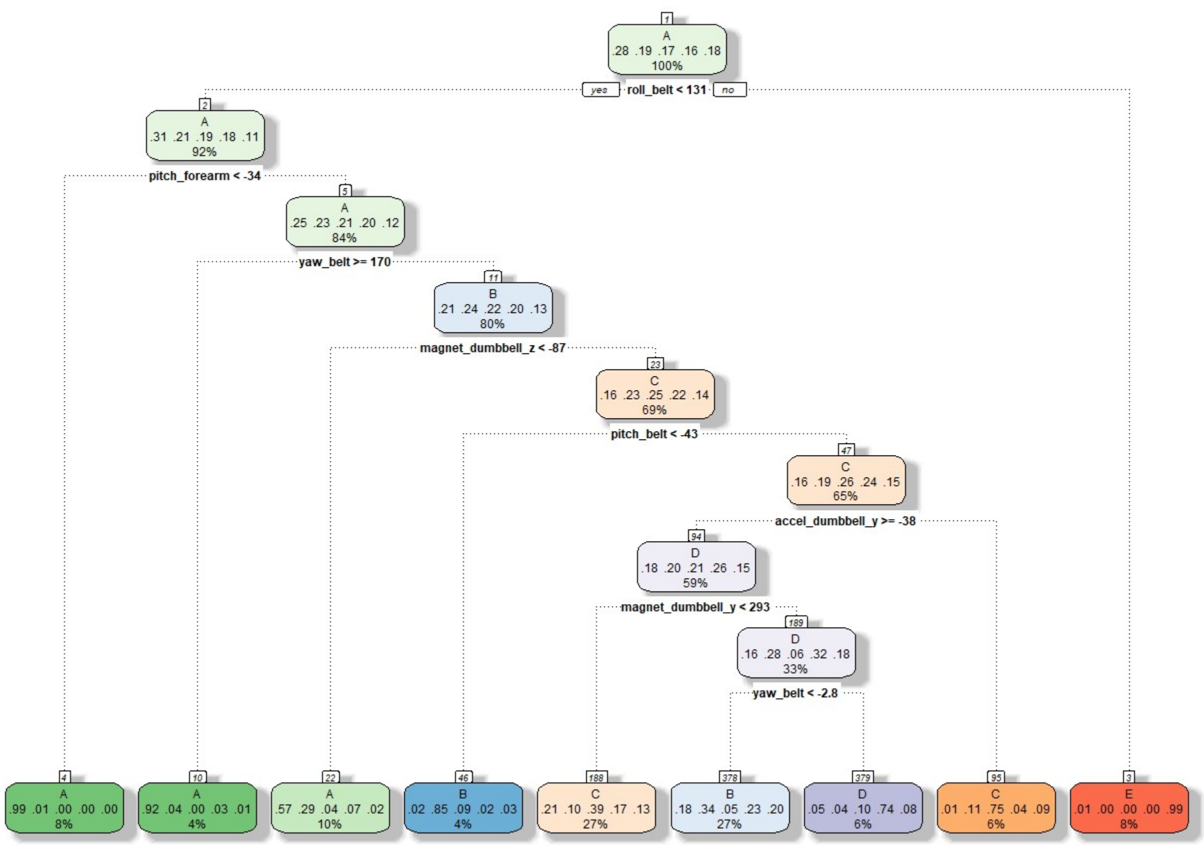
Cross-validating our performance of the different techniques will help prevent overfitting of the models. We will use K-Fold validation of 10 times to ensure it is accurate and the data set is small enough that the computer can easily handle the extra processing necessary to test the models.

Classification Tree

First we will train the classification tree.

```
ClassTrain <- trainControl(method="cv", number=10)
ClassTreeModel <- train(classe~., data=WithinTrainingData, method="rpart", trControl=ClassTra
in)
#Graph the model to see what it looks like
fancyRpartPlot(ClassTreeModel$finalModel)
```

```
## Warning: Bad 'data' field in model 'call'.
## To silence this warning:
##     Call prp with roundint=FALSE,
##     or rebuild the rpart model with model=TRUE.
```



Rattle 2019-May-26 15:49:49 Darre

We see the data is partitioned by the roll belt < 131, pitch forarm <-.34, magnet dummbell y < 427, and roll forearm <124.

```
ClassTreePred <- predict(ClassTreeModel,newdata=WithinTestData)
ClassTreeConfusionMatrix <- confusionMatrix(WithinTestData$classe, ClassTreePred)
ClassTreeConfusionMatrix
```

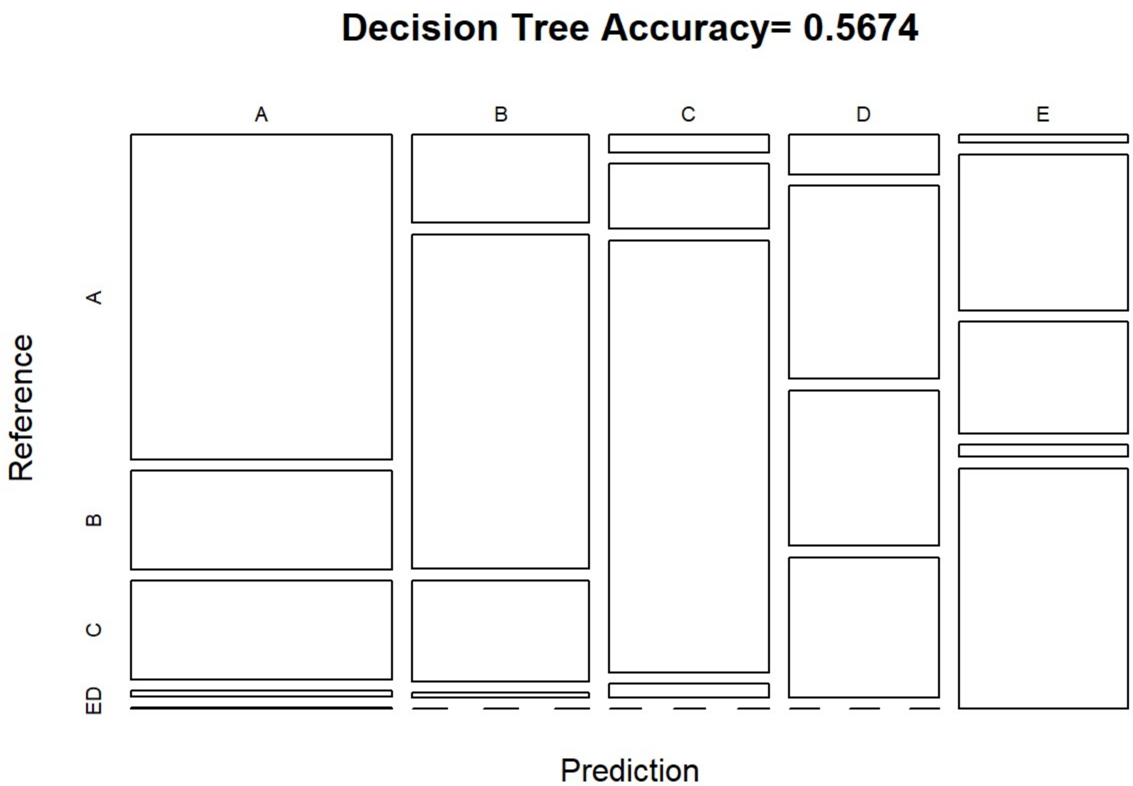
```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1201  363  365   20    4
##          B  221  842  253   12    0
##          C   40  147  979   31    0
##          D   84  411  332  298    0
##          E   19  373  267   28  575
## 
## Overall Statistics
##                                           
##                Accuracy : 0.5674          
##                  95% CI : (0.5556, 0.5791)
##     No Information Rate : 0.3199          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.4554          
##  Mcnemar's Test P-Value : < 2.2e-16       
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7674   0.3942   0.4458  0.76607  0.99309
## Specificity            0.8581   0.8972   0.9533  0.87230  0.89071
## Pos Pred Value         0.6150   0.6340   0.8179  0.26489  0.45563
## Neg Pred Value         0.9259   0.7663   0.7853  0.98415  0.99929
## Prevalence             0.2280   0.3111   0.3199  0.05666  0.08434
## Detection Rate         0.1749   0.1227   0.1426  0.04341  0.08376
## Detection Prevalence   0.2845   0.1934   0.1744  0.16387  0.18383
## Balanced Accuracy      0.8128   0.6457   0.6996  0.81918  0.94190
```

```
ClassTreeConfusionMatrix$overall[1]
```

```
##  Accuracy 
## 0.5673707
```

We see the accuracy is only approximately 50%.

```
plot(ClassTreeConfusionMatrix$table, col = ClassTreeConfusionMatrix$byClass,
     main = paste("Decision Tree Accuracy=", round(ClassTreeConfusionMatrix$overall['Accurac
y'], 4)))
```



The out of sample error rate of .5 is high.

Random Forest

Now we will train the random forest model to see how it does.

```
RandomForestControl <- trainControl(method="cv", number=3, verboseIter=FALSE)
RandomForestModel <- train(classe ~ ., data=WithinTrainingData, method="rf", trControl=Random
ForestControl)
RandomForestModel$finalModel
```

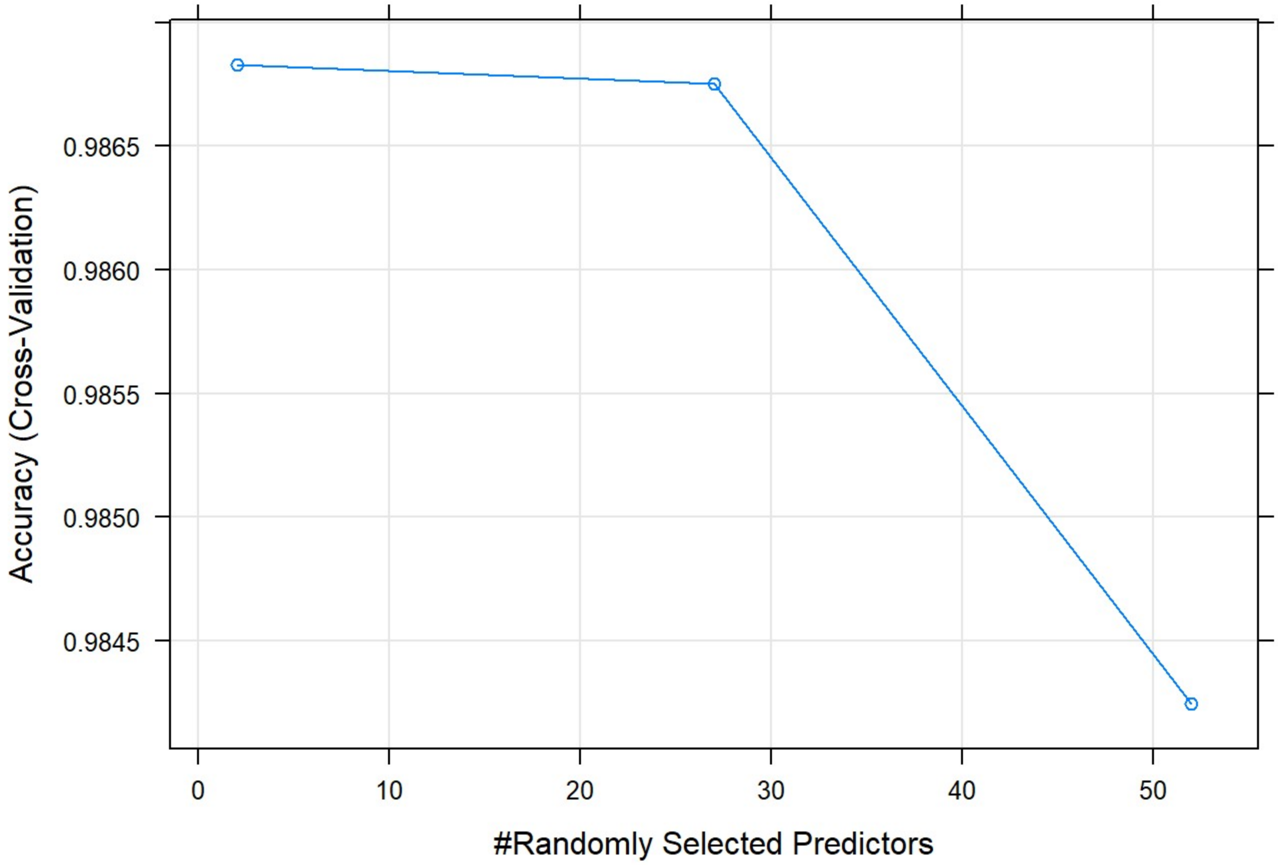
```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0.72%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3623    2    0    0    2 0.001102840
## B   13 2447    9    0    0 0.008910490
## C    0   17 2201    7    0 0.010786517
## D    0    0   36 2053    2 0.018173123
## E    0    0    1    3 2341 0.001705757
```

When we validate the model, we see class error more often in B and D.

```
RandomForestPred <- predict(RandomForestModel, newdata=WithinTestData)
RandomForestConfusionMatrix <- confusionMatrix(RandomForestPred, WithinTestData$classe)
RandomForestConfusionMatrix
```

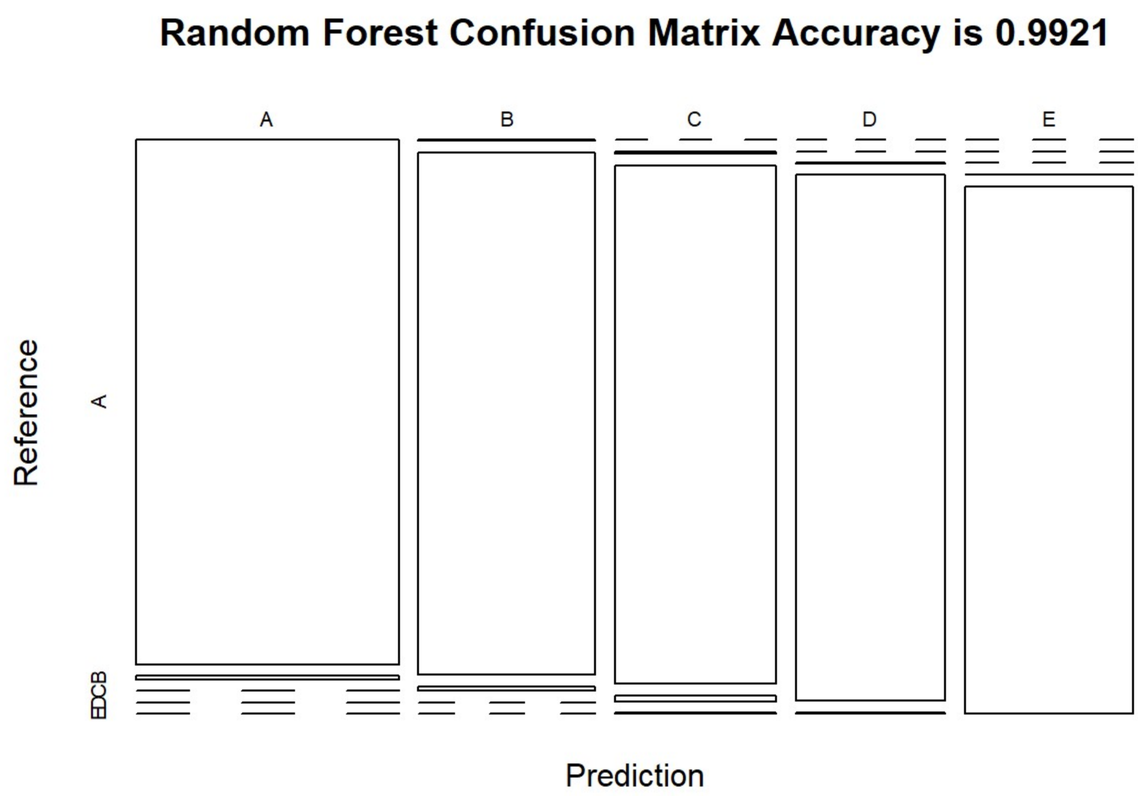
```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1951   14    0    0    0
##          B    2 1308   11    0    0
##          C    0    6 1185   14    2
##          D    0    0    1 1110    3
##          E    0    0    0    1 1257
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9921          
##                  95% CI : (0.9897, 0.9941)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.99            
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9990   0.9849   0.9900   0.9867   0.9960
## Specificity            0.9971   0.9977   0.9961   0.9993   0.9998
## Pos Pred Value         0.9929   0.9902   0.9818   0.9964   0.9992
## Neg Pred Value         0.9996   0.9964   0.9979   0.9974   0.9991
## Prevalence             0.2845   0.1934   0.1744   0.1639   0.1838
## Detection Rate         0.2842   0.1905   0.1726   0.1617   0.1831
## Detection Prevalence   0.2862   0.1924   0.1758   0.1623   0.1832
## Balanced Accuracy      0.9981   0.9913   0.9930   0.9930   0.9979
```

```
#Plotting the model to cross-validate it
plot(RandomForestModel)
```



The accuracy is 99%. This seems like a bit too high.

```
plot(RandomForestConfusionMatrix$table, col = RandomForestConfusionMatrix$byClass, main = pas
te("Random Forest Confusion Matrix Accuracy is", round(RandomForestConfusionMatrix$overall['A
ccuracy'], 4)))
```



Generalized Boosted Regression Modeling (GBM)

We set the seed again to ensure the results are consistent each time we run it. We'll run it 5 times.

```
set.seed(1234)
GBM_Control <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
GBM_Model  <- train(classe ~ ., data=WithinTrainingData, method = "gbm", trControl = GBM_Cont
rol, verbose = FALSE)
GBM_Model$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 41 had non-zero influence.
```

There were 52 predictors found and 43 that were not predictors.

```
GBM_Model
```

```
## Stochastic Gradient Boosting
##
## 12757 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 10206, 10206, 10204, 10205, 10207
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7515882  0.6849876
##   1                  100      0.8177474  0.7693766
##   1                  150      0.8525517  0.8135070
##   2                   50      0.8541201  0.8151888
##   2                  100      0.9060912  0.8811559
##   2                  150      0.9312533  0.9130055
##   3                   50      0.8969198  0.8694978
##   3                  100      0.9421486  0.9267994
##   3                  150      0.9597862  0.9491271
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

The depth at 3 levels with 150 trees has a very high accuracy of 96%.

```
GBM_predictions <- predict(GBM_Model, newdata=WithinTestData)
GBM_ConfusionMatrix <- confusionMatrix(GBM_predictions, WithinTestData$classe)
GBM_ConfusionMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1922   29    0    1    4
##          B   19 1248   46    7   18
##          C    6   49 1133   42   13
##          D    3    1   16 1067    9
##          E    3    1    2    8 1218
##
## Overall Statistics
##
##                Accuracy : 0.9597
##                  95% CI : (0.9547, 0.9642)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.949
##  Mcnemar's Test P-Value : 4.409e-07
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9841   0.9398   0.9465   0.9484   0.9651
## Specificity            0.9931   0.9837   0.9806   0.9949   0.9975
## Pos Pred Value         0.9826   0.9327   0.9115   0.9735   0.9886
## Neg Pred Value         0.9937   0.9855   0.9886   0.9899   0.9922
## Prevalence             0.2845   0.1934   0.1744   0.1639   0.1838
## Detection Rate         0.2800   0.1818   0.1650   0.1554   0.1774
## Detection Prevalence   0.2849   0.1949   0.1811   0.1597   0.1795
## Balanced Accuracy      0.9886   0.9618   0.9636   0.9717   0.9813
```

The cross-validated results of the GBM model get an accuracy rate of 96.3%. The 95% confidence interval is between 95.8% and 96.7%.

Conclusion

The Random Forest Model is the best performing model. Using it on the cleaned validation data gets:

```
Results <- predict(RandomForestModel, newdata=CleanTestData)
Results
```

```
##  [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
```