Weight lifting sensor data analysis

Author: "Salvatore Lenza"

Rome, "Sunday, December 27, 2015"

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 dumbell of 6 participants to predict the manner in which praticipants did the exercise.

We propose random forest based model because no need cross-validation or a separate test set to get an unbiased estimate of the test set error.

The dependent variable or response is the "classe" variable in the training set.

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

```r
training <- read.csv("pml-training.csv");
testing <-  read.csv("pml-testing.csv");

str(training)
```

Data getting and cleaning

```
## '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 1323084232
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 484
##  $ 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 -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]
```

```r
dim(testing)
```

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

```r
irr_V <- grep("X|timestamp|user_name|new_window", names(training));
training <- training[,-irr_V];
testing <- testing[,-irr_V];

training[is.na(training)] <- 0

n_V<- nearZeroVar(training);
training <- training[,-n_V];
testing <- testing[,-n_V];
```

```r
set.seed(1968);
in_Train <- createDataPartition(training$classe, p=0.7, list = FALSE);
new_training <- training[in_Train,];
new_validation <- training[-in_Train,];
```

Split the preprocessed training data into training set and validation set

Check the correlations There doesn't seem to be any predictors strongly correlated with the outcome variable, so linear regression model may not be a good option. Random forest model may be more robust for this data.

```r
cor <- abs(sapply(colnames(new_training[, -ncol(training)]), function(x) cor(as.numeric(new_training[, 

cor
```

```
##           num_window            roll_belt           pitch_belt
##         0.0042309478         0.1269828075         0.0457276986
##             yaw_belt     total_accel_belt         gyros_belt_x
##         0.0718554663         0.0852095475         0.0019411607
##         gyros_belt_y         gyros_belt_z         accel_belt_x
##         0.0017295759         0.0016520645         0.0405468895
##         accel_belt_y         accel_belt_z        magnet_belt_x
##         0.0171893316         0.1355909260         0.0017815510
##        magnet_belt_y        magnet_belt_z             roll_arm
##         0.1950122741         0.1362964376         0.0563142577
##            pitch_arm              yaw_arm      total_accel_arm
##         0.1845562942         0.0316015862         0.1493382448
##          gyros_arm_x          gyros_arm_y          gyros_arm_z
##         0.0254306845         0.0331709786         0.0135874084
##          accel_arm_x          accel_arm_y          accel_arm_z
##         0.2588471723         0.0810985735         0.0969412939
##         magnet_arm_x         magnet_arm_y         magnet_arm_z
##         0.2828523358         0.2659390781         0.1640101352
##        roll_dumbbell       pitch_dumbbell         yaw_dumbbell
##         0.0907666304         0.0998789814         0.0086422174
## total_accel_dumbbell      gyros_dumbbell_x     gyros_dumbbell_y
##         0.0145960288         0.0062670313         0.0179900499
##     gyros_dumbbell_z     accel_dumbbell_x     accel_dumbbell_y
##         0.0167054902         0.1293439352         0.0122107618
##     accel_dumbbell_z    magnet_dumbbell_x    magnet_dumbbell_y
##         0.0817389615         0.1485777163         0.0436729696
##    magnet_dumbbell_z         roll_forearm        pitch_forearm
##         0.2011403744         0.0571866769         0.3219191157
##          yaw_forearm  total_accel_forearm      gyros_forearm_x
##         0.0519096134         0.1199433870         0.0077698236
##      gyros_forearm_y      gyros_forearm_z      accel_forearm_x
##         0.0031574012         0.0050085335         0.2053942452
##      accel_forearm_y      accel_forearm_z     magnet_forearm_x
##         0.0211072849         0.0005669254         0.1961128708
##     magnet_forearm_y     magnet_forearm_z
##         0.1122917869         0.0507961046
```

```r
model_Fit <- randomForest(classe ~., data=new_training, do.trace = 10)
```

Model Fitting

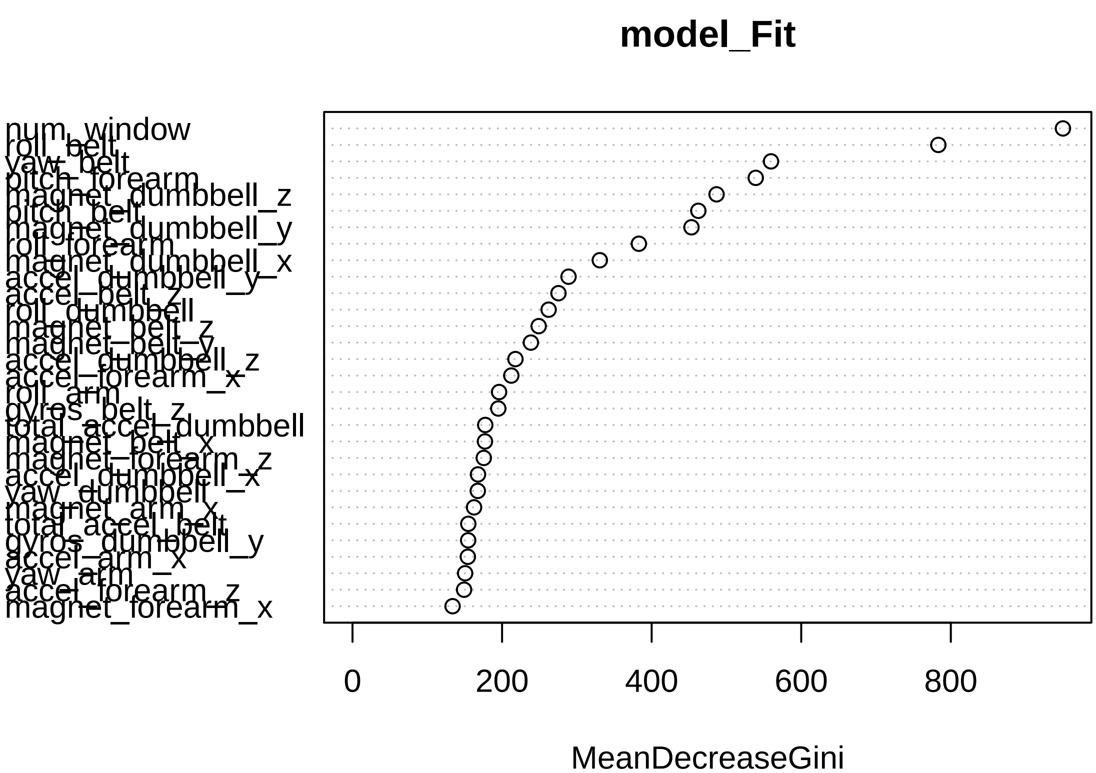
```
## ntree      OOB      1      2      3      4      5
##    10:   3.77%  1.84%  5.33%  5.60%  4.17%  3.03%
##    20:   1.16%  0.44%  1.88%  1.67%  1.42%  0.83%
##    30:   0.74%  0.15%  1.13%  1.09%  1.11%  0.55%
##    40:   0.53%  0.03%  0.68%  1.04%  0.89%  0.36%
##    50:   0.51%  0.05%  0.68%  0.96%  0.84%  0.32%
##    60:   0.42%  0.03%  0.41%  0.83%  0.89%  0.24%
##    70:   0.41%  0.00%  0.41%  0.75%  0.93%  0.28%
##    80:   0.33%  0.00%  0.38%  0.54%  0.75%  0.24%
##    90:   0.31%  0.00%  0.38%  0.46%  0.71%  0.20%
##   100:   0.31%  0.00%  0.38%  0.58%  0.62%  0.20%
##   110:   0.33%  0.00%  0.41%  0.54%  0.71%  0.20%
##   120:   0.31%  0.00%  0.30%  0.54%  0.71%  0.20%
##   130:   0.31%  0.00%  0.30%  0.58%  0.71%  0.20%
##   140:   0.30%  0.00%  0.26%  0.58%  0.67%  0.20%
##   150:   0.30%  0.00%  0.30%  0.54%  0.67%  0.20%
##   160:   0.32%  0.00%  0.34%  0.58%  0.71%  0.20%
##   170:   0.31%  0.00%  0.38%  0.54%  0.62%  0.20%
##   180:   0.30%  0.00%  0.38%  0.50%  0.62%  0.20%
##   190:   0.28%  0.00%  0.34%  0.46%  0.58%  0.20%
##   200:   0.28%  0.00%  0.34%  0.42%  0.62%  0.20%
##   210:   0.27%  0.00%  0.30%  0.50%  0.53%  0.20%
##   220:   0.28%  0.00%  0.30%  0.50%  0.62%  0.16%
##   230:   0.28%  0.00%  0.30%  0.50%  0.62%  0.20%
##   240:   0.26%  0.00%  0.30%  0.42%  0.58%  0.20%
##   250:   0.25%  0.00%  0.30%  0.42%  0.53%  0.16%
##   260:   0.25%  0.00%  0.34%  0.42%  0.49%  0.16%
##   270:   0.25%  0.00%  0.34%  0.42%  0.49%  0.16%
##   280:   0.25%  0.00%  0.34%  0.42%  0.49%  0.16%
##   290:   0.25%  0.00%  0.30%  0.46%  0.49%  0.20%
##   300:   0.26%  0.00%  0.34%  0.46%  0.49%  0.20%
##   310:   0.24%  0.00%  0.26%  0.46%  0.49%  0.16%
##   320:   0.24%  0.00%  0.26%  0.46%  0.49%  0.16%
##   330:   0.23%  0.00%  0.26%  0.42%  0.49%  0.16%
##   340:   0.23%  0.00%  0.26%  0.42%  0.49%  0.16%
##   350:   0.24%  0.00%  0.26%  0.46%  0.49%  0.16%
##   360:   0.23%  0.00%  0.26%  0.46%  0.44%  0.16%
##   370:   0.23%  0.00%  0.26%  0.46%  0.44%  0.16%
##   380:   0.23%  0.00%  0.26%  0.46%  0.44%  0.16%
##   390:   0.24%  0.00%  0.26%  0.50%  0.44%  0.16%
##   400:   0.23%  0.00%  0.26%  0.46%  0.44%  0.16%
##   410:   0.24%  0.00%  0.26%  0.50%  0.44%  0.16%
##   420:   0.25%  0.00%  0.30%  0.46%  0.44%  0.20%
##   430:   0.23%  0.00%  0.30%  0.42%  0.44%  0.16%
##   440:   0.25%  0.00%  0.34%  0.42%  0.49%  0.16%
##   450:   0.24%  0.00%  0.30%  0.42%  0.49%  0.16%
##   460:   0.24%  0.00%  0.26%  0.50%  0.44%  0.16%
##   470:   0.25%  0.00%  0.34%  0.46%  0.44%  0.16%
```

```
##   480:   0.25%  0.00%  0.34%  0.50%  0.44%  0.16%
##   490:   0.25%  0.00%  0.26%  0.50%  0.49%  0.16%
##   500:   0.23%  0.00%  0.26%  0.46%  0.44%  0.16%
```

```
model_Fit
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = new_training, do.trace = 10)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.23%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3906    0    0    0    0 0.000000000
## B    6 2651    1    0    0 0.002633559
## C    0   10 2385    1    0 0.004590985
## D    0    0   10 2242    0 0.004440497
## E    0    0    0    4 2521 0.001584158
```

```
varImpPlot(model_Fit)
```



```r
model_Pred <- predict(model_Fit,new_validation);
confusionMatrix(model_Pred, new_validation$classe);
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    9    0    0    0
##          B    0 1128    1    0    0
##          C    0    2 1025    3    0
##          D    0    0    0  961    3
##          E    1    0    0    0 1079
##
## Overall Statistics
##
##                Accuracy : 0.9968
##                  95% CI : (0.995, 0.9981)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9959
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9903   0.9990   0.9969   0.9972
## Specificity            0.9979   0.9998   0.9990   0.9994   0.9998
## Pos Pred Value         0.9946   0.9991   0.9951   0.9969   0.9991
## Neg Pred Value         0.9998   0.9977   0.9998   0.9994   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1917   0.1742   0.1633   0.1833
## Detection Prevalence   0.2858   0.1918   0.1750   0.1638   0.1835
## Balanced Accuracy      0.9986   0.9951   0.9990   0.9981   0.9985
```

The random forest algorithm generates a model with accuracy 0.9968.

Results

Now running the model on the test set (20 test cases).

```r
pred_Final <- predict(model_Fit, testing);
pred_Final;
```

```
##  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
```

The algorithm does correctly predicts the way in which the exercises were carried out. The outputs are to be saved to files for submission.

```r
results <- as.vector(pred_Final)

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)
  }
}
write_files(results)
```