Prediction Assignment

David Marquinez

Abstract

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. We will use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

Goal

The goal of this project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set: * 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).
* throwing the hips to the front (Class E).

Data

First, we will download the training and test data.

```r
if(!file.exists("pml-training.csv"))
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv","pml-training.csv")
if(!file.exists("pml-testing.csv"))
    download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv","pml-testing.cs
training <- read.csv("pml-training.csv")
testing <- read.csv("pml-testing.csv")
```

If we take a first look to the data, we can remove some unnecesary variables that will not be necessary to our model. We will only consider variables related to measurements in x,y,z axis.

```r
names(training)
```

```
##   [1] "X"                         "user_name"
##   [3] "raw_timestamp_part_1"      "raw_timestamp_part_2"
##   [5] "cvtd_timestamp"            "new_window"
##   [7] "num_window"                "roll_belt"
```

```
##   [9] "pitch_belt"               "yaw_belt"
##  [11] "total_accel_belt"         "kurtosis_roll_belt"
##  [13] "kurtosis_picth_belt"      "kurtosis_yaw_belt"
##  [15] "skewness_roll_belt"       "skewness_roll_belt.1"
##  [17] "skewness_yaw_belt"        "max_roll_belt"
##  [19] "max_picth_belt"           "max_yaw_belt"
##  [21] "min_roll_belt"            "min_pitch_belt"
##  [23] "min_yaw_belt"             "amplitude_roll_belt"
##  [25] "amplitude_pitch_belt"     "amplitude_yaw_belt"
##  [27] "var_total_accel_belt"     "avg_roll_belt"
##  [29] "stddev_roll_belt"         "var_roll_belt"
##  [31] "avg_pitch_belt"           "stddev_pitch_belt"
##  [33] "var_pitch_belt"           "avg_yaw_belt"
##  [35] "stddev_yaw_belt"          "var_yaw_belt"
##  [37] "gyros_belt_x"             "gyros_belt_y"
##  [39] "gyros_belt_z"             "accel_belt_x"
##  [41] "accel_belt_y"             "accel_belt_z"
##  [43] "magnet_belt_x"            "magnet_belt_y"
##  [45] "magnet_belt_z"            "roll_arm"
##  [47] "pitch_arm"                "yaw_arm"
##  [49] "total_accel_arm"          "var_accel_arm"
##  [51] "avg_roll_arm"             "stddev_roll_arm"
##  [53] "var_roll_arm"             "avg_pitch_arm"
##  [55] "stddev_pitch_arm"         "var_pitch_arm"
##  [57] "avg_yaw_arm"              "stddev_yaw_arm"
##  [59] "var_yaw_arm"              "gyros_arm_x"
##  [61] "gyros_arm_y"              "gyros_arm_z"
##  [63] "accel_arm_x"              "accel_arm_y"
##  [65] "accel_arm_z"              "magnet_arm_x"
##  [67] "magnet_arm_y"             "magnet_arm_z"
##  [69] "kurtosis_roll_arm"        "kurtosis_picth_arm"
##  [71] "kurtosis_yaw_arm"         "skewness_roll_arm"
##  [73] "skewness_pitch_arm"       "skewness_yaw_arm"
##  [75] "max_roll_arm"             "max_picth_arm"
##  [77] "max_yaw_arm"              "min_roll_arm"
##  [79] "min_pitch_arm"            "min_yaw_arm"
##  [81] "amplitude_roll_arm"       "amplitude_pitch_arm"
##  [83] "amplitude_yaw_arm"        "roll_dumbbell"
##  [85] "pitch_dumbbell"           "yaw_dumbbell"
##  [87] "kurtosis_roll_dumbbell"   "kurtosis_picth_dumbbell"
##  [89] "kurtosis_yaw_dumbbell"    "skewness_roll_dumbbell"
##  [91] "skewness_pitch_dumbbell"  "skewness_yaw_dumbbell"
##  [93] "max_roll_dumbbell"        "max_picth_dumbbell"
##  [95] "max_yaw_dumbbell"         "min_roll_dumbbell"
##  [97] "min_pitch_dumbbell"       "min_yaw_dumbbell"
##  [99] "amplitude_roll_dumbbell"  "amplitude_pitch_dumbbell"
## [101] "amplitude_yaw_dumbbell"   "total_accel_dumbbell"
## [103] "var_accel_dumbbell"       "avg_roll_dumbbell"
## [105] "stddev_roll_dumbbell"     "var_roll_dumbbell"
## [107] "avg_pitch_dumbbell"       "stddev_pitch_dumbbell"
## [109] "var_pitch_dumbbell"       "avg_yaw_dumbbell"
## [111] "stddev_yaw_dumbbell"      "var_yaw_dumbbell"
## [113] "gyros_dumbbell_x"         "gyros_dumbbell_y"
## [115] "gyros_dumbbell_z"         "accel_dumbbell_x"
```

```
## [117] "accel_dumbbell_y"         "accel_dumbbell_z"
## [119] "magnet_dumbbell_x"        "magnet_dumbbell_y"
## [121] "magnet_dumbbell_z"        "roll_forearm"
## [123] "pitch_forearm"            "yaw_forearm"
## [125] "kurtosis_roll_forearm"    "kurtosis_picth_forearm"
## [127] "kurtosis_yaw_forearm"     "skewness_roll_forearm"
## [129] "skewness_pitch_forearm"   "skewness_yaw_forearm"
## [131] "max_roll_forearm"         "max_picth_forearm"
## [133] "max_yaw_forearm"          "min_roll_forearm"
## [135] "min_pitch_forearm"        "min_yaw_forearm"
## [137] "amplitude_roll_forearm"   "amplitude_pitch_forearm"
## [139] "amplitude_yaw_forearm"    "total_accel_forearm"
## [141] "var_accel_forearm"        "avg_roll_forearm"
## [143] "stddev_roll_forearm"      "var_roll_forearm"
## [145] "avg_pitch_forearm"        "stddev_pitch_forearm"
## [147] "var_pitch_forearm"        "avg_yaw_forearm"
## [149] "stddev_yaw_forearm"       "var_yaw_forearm"
## [151] "gyros_forearm_x"          "gyros_forearm_y"
## [153] "gyros_forearm_z"          "accel_forearm_x"
## [155] "accel_forearm_y"          "accel_forearm_z"
## [157] "magnet_forearm_x"         "magnet_forearm_y"
## [159] "magnet_forearm_z"         "classe"
```

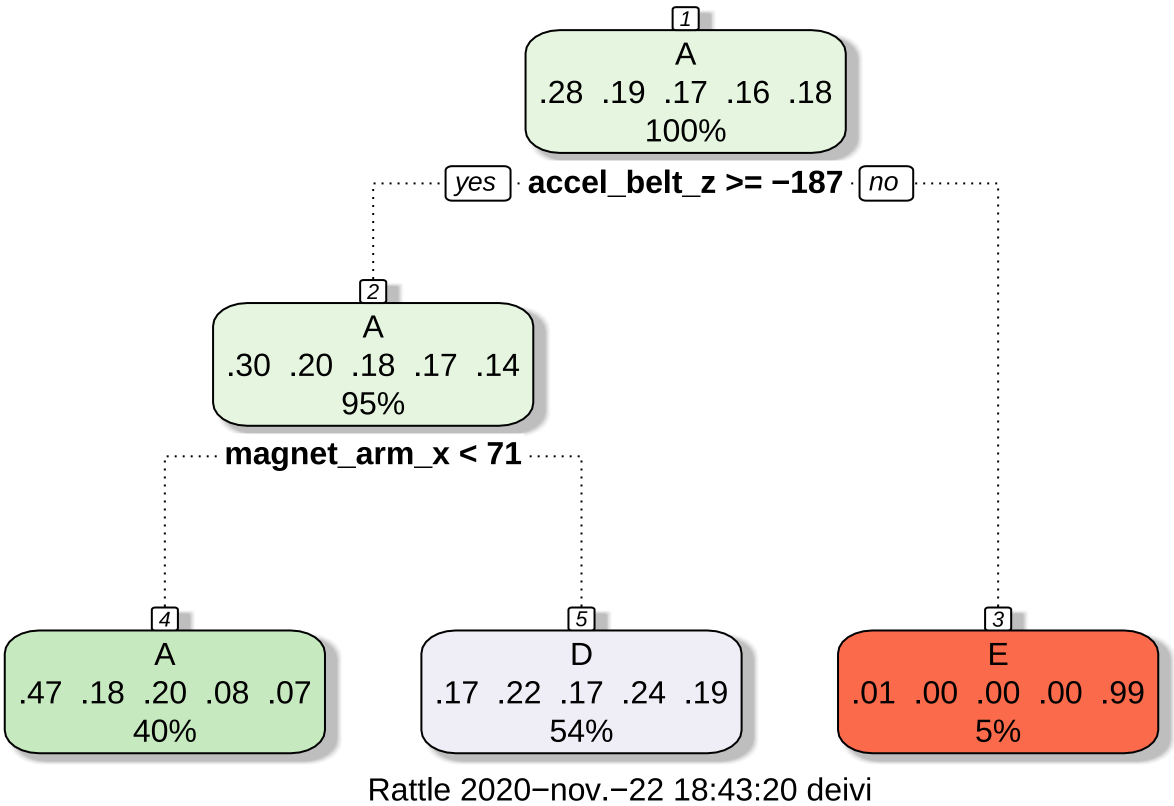
```
xyzattr <- names(training)[grepl("x$|y$|z$",names(training))]
testingData <- testing[,xyzattr]
xyzattr <- c(xyzattr,"classe")
trainingData <- training[,xyzattr]
trainingData$classe <- as.factor(trainingData$classe)
```

Models

Recursive Partitioning

Recursive partitioning is a statistical method for multivariable analysis. Recursive partitioning creates a decision tree that strives to correctly classify members of the population by splitting it into sub-populations based on several dichotomous independent variables. The process is termed recursive because each sub-population may in turn be split an indefinite number of times until the splitting process terminates after a particular stopping criterion is reached. [4]

```
tree.fit <- train(classe ~. , method="rpart", data=trainingData)
prediccion.tree <- predict(tree.fit,newdata=trainingData)
fancyRpartPlot(tree.fit$finalModel)
```



We can see the decision tree above.

```
(conf.tree <- confusionMatrix(prediccion.tree,trainingData[,"classe"]) )
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3711 1436 1560  640  560
##          B    0    0    0    0    0
##          C    0    0    0    0    0
##          D 1863 2360 1862 2576 2019
##          E    6    1    0    0 1028
##
## Overall Statistics
##
##                Accuracy : 0.3728
##                  95% CI : (0.366, 0.3796)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2025
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6651   0.0000   0.0000   0.8010  0.28500
## Specificity            0.7012   1.0000   1.0000   0.5060  0.99956
## Pos Pred Value         0.4693      NaN      NaN   0.2412  0.99324
## Neg Pred Value         0.8405   0.8065   0.8256   0.9284  0.86125
## Prevalence             0.2844   0.1935   0.1744   0.1639  0.18382
## Detection Rate         0.1891   0.0000   0.0000   0.1313  0.05239
## Detection Prevalence   0.4030   0.0000   0.0000   0.5443  0.05275
## Balanced Accuracy      0.6831   0.5000   0.5000   0.6535  0.64228
```

And some statistics about this model. For example, we have achieved a bad accuracy (`0.3727958`) and no information rate of (`0.2843747`)

Random forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees.[5]

```r
fitControl <- trainControl(method = "cv", number = 5, allowParallel = TRUE)
rf.fit <- train(classe ~ ., method="rf",data=trainingData,trControl = fitControl,ntree=500)
prediccion.rf <- predict(rf.fit,newdata=trainingData)
(conf.rf <- confusionMatrix(prediccion.rf,trainingData[,"classe"]) )
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 5580    0    0    0    0
##          B    0 3797    0    0    0
##          C    0    0 3422    0    0
##          D    0    0    0 3216    0
##          E    0    0    0    0 3607
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9998, 1)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
```

```
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2844   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

We have 'overfitted' our model, as we can see we have `1` accuracy. We can predict all the classes from every sample.

Gradient Boosting

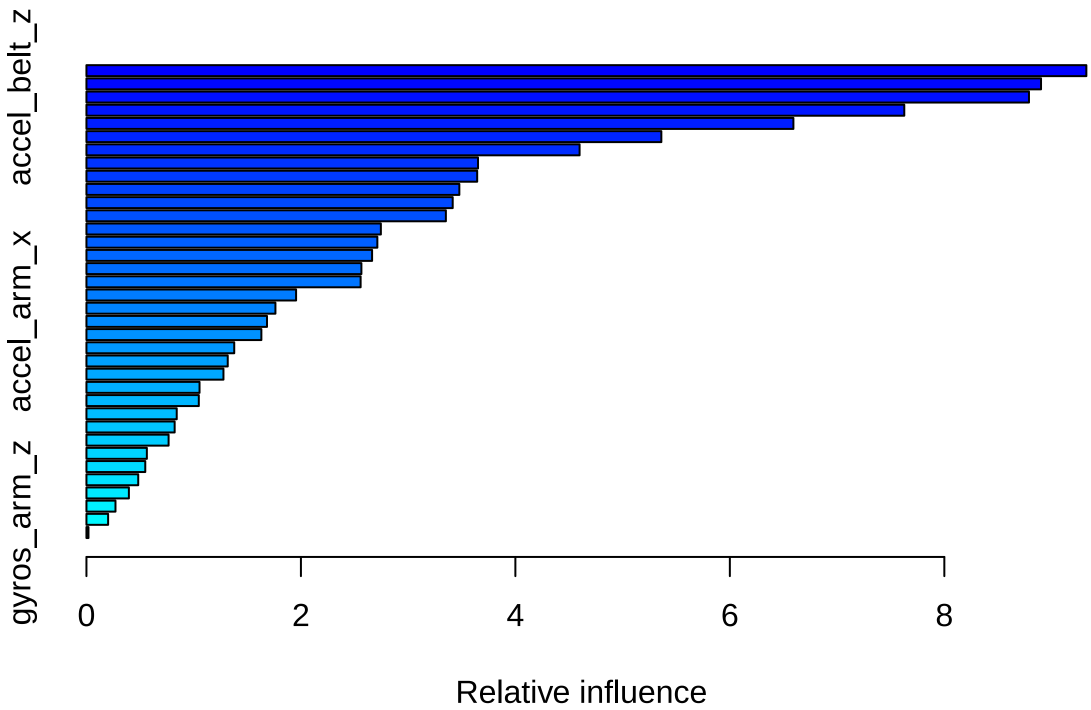
Gradient boosting is a machine learning technique for regression and classification problems. It produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.[6]

```
gbm.fit <- train(classe ~ ., method="gbm",data=trainingData,trControl = fitControl, verbose=FALSE)
prediccion.gbm <- predict(gbm.fit,newdata=trainingData)
(conf.gbm <- confusionMatrix(prediccion.gbm,trainingData[,"classe"]) )
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 5427  240   74   64   31
##          B   41 3322  141   22   74
##          C   46  191 3145  205   63
##          D   62   30   52 2880   67
##          E    4   14   10   45 3372
##
## Overall Statistics
##
##                Accuracy : 0.9248
##                  95% CI : (0.921, 0.9284)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9047
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9726   0.8749   0.9191   0.8955   0.9348
## Specificity            0.9709   0.9824   0.9688   0.9871   0.9954
## Pos Pred Value         0.9299   0.9228   0.8616   0.9317   0.9788
## Neg Pred Value         0.9889   0.9704   0.9827   0.9797   0.9855
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2766   0.1693   0.1603   0.1468   0.1718
## Detection Prevalence   0.2974   0.1835   0.1860   0.1575   0.1756
## Balanced Accuracy      0.9717   0.9287   0.9439   0.9413   0.9651
```

We will chose this model because we have a low in sample error of `7.522169` % and we have reduced the overfitting.

```r
summary(gbm.fit)
```




```
##                                   var    rel.inf
## magnet_dumbbell_z magnet_dumbbell_z 9.32450662
## magnet_dumbbell_y magnet_dumbbell_y 8.90187388
## accel_belt_z           accel_belt_z 8.78936528
## magnet_belt_z         magnet_belt_z 7.62654210
## gyros_belt_z           gyros_belt_z 6.59245267
## accel_dumbbell_y   accel_dumbbell_y 5.36126574
## accel_dumbbell_z   accel_dumbbell_z 4.59876518
## magnet_belt_y         magnet_belt_y 3.65152795
## magnet_dumbbell_x magnet_dumbbell_x 3.64371029
## accel_forearm_x     accel_forearm_x 3.47744267
## magnet_arm_z           magnet_arm_z 3.41545401
## magnet_arm_x           magnet_arm_x 3.35184285
## accel_forearm_z     accel_forearm_z 2.74555588
## magnet_forearm_x   magnet_forearm_x 2.71265775
## accel_dumbbell_x   accel_dumbbell_x 2.66370942
## magnet_belt_x         magnet_belt_x 2.56406436
## gyros_dumbbell_y   gyros_dumbbell_y 2.55691408
## magnet_arm_y           magnet_arm_y 1.95509606
## magnet_forearm_z   magnet_forearm_z 1.76226539
```

```
## accel_arm_x            accel_arm_x 1.68387059
## accel_arm_z            accel_arm_z 1.63135773
## gyros_arm_y            gyros_arm_y 1.37822610
## gyros_belt_y          gyros_belt_y 1.31770118
## magnet_forearm_y   magnet_forearm_y 1.27777620
## accel_forearm_y     accel_forearm_y 1.05468234
## gyros_belt_x          gyros_belt_x 1.04802606
## gyros_dumbbell_x   gyros_dumbbell_x 0.84207986
## accel_belt_x          accel_belt_x 0.82266317
## gyros_arm_x            gyros_arm_x 0.76615817
## gyros_forearm_y     gyros_forearm_y 0.56347584
## accel_arm_y            accel_arm_y 0.54851729
## gyros_dumbbell_z   gyros_dumbbell_z 0.48287515
## gyros_forearm_z     gyros_forearm_z 0.39584425
## accel_belt_y          accel_belt_y 0.27080797
## gyros_forearm_x     gyros_forearm_x 0.20205933
## gyros_arm_z            gyros_arm_z 0.01886658
```

We can see in the figure from above the relative influence of the variables.

Prediction

Now that we have choosen our model we will predict the classe of the testing set.

```r
predict(gbm.fit,newdata=testingData)
```

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

Bibliography

[1] 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.
[2] Mandy Sidana. Intro to types of classification algorithms in Machine Learning(Feb 28, 2017)
[3] Harshdeep Singh, Understanding Gradient Boosting Machines(Nov 3, 2018)
[4] Recursive Partitioning, Wikipedia
[5] Random forests, Wikipedia
[6] Gradient Boosting, Wikipedia