Machine Learning Project on Dumbell Usage

Bassem Taha

February 5, 2017

Overview

In this project, we will use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants who were asked to perform barbell lifts in 5 different ways as follow:

- correct performance per 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).

The goal of this project is to predict the manner in which the 6 participants did the exercise.

Libraries

```
# Libraries
library(knitr)
library(caret)
library(rpart)
library(plyr)
library(dplyr)
library(gbm)
library(rattle)
library(randomForest)
```

Data Processing

Load the following training & testing datasets:

[pml-testing] (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

[pml-training] (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

```
download.file(url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", destfile = "
download.file(url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", destfile =
# read datasets
training <- read.csv("pml-training.csv", na.strings = c("NA", ""))
testing <- read.csv("pml-testing.csv", na.strings = c("NA", ""))
```

Datasets exploration and Cleaning

Data from the training dataset indicate that 100 columns have "NA" & spaces and columns 1:7 (i.e timestamp, date) have no bearing on the computation. Hence all these will be removed.

```
# First 7 irrelevant columns to be removed
head(training[,1:7],1)
```

```
##   X user_name raw_timestamp_part_1 raw_timestamp_part_2   cvtd_timestamp
## 1 1  carlitos           1323084231               788290 05/12/2011 11:23
##   new_window num_window
```

```
## 1          no          11
```

```r
trainingDat <- training[, 8:length(names(training))]
testingDat <- testing[, 8:length(names(testing))]

# 100 columns have NA & spaces
length(which(colSums(is.na(trainingDat))!=0)); which(colSums(is.na(trainingDat))!=0)
```

```
## [1] 100
```

```
##       kurtosis_roll_belt      kurtosis_picth_belt        kurtosis_yaw_belt
##                        5                        6                        7
##       skewness_roll_belt     skewness_roll_belt.1        skewness_yaw_belt
##                        8                        9                       10
##            max_roll_belt           max_picth_belt             max_yaw_belt
##                       11                       12                       13
##            min_roll_belt           min_pitch_belt             min_yaw_belt
##                       14                       15                       16
##      amplitude_roll_belt     amplitude_pitch_belt       amplitude_yaw_belt
##                       17                       18                       19
##     var_total_accel_belt            avg_roll_belt         stddev_roll_belt
##                       20                       21                       22
##            var_roll_belt           avg_pitch_belt        stddev_pitch_belt
##                       23                       24                       25
##           var_pitch_belt             avg_yaw_belt          stddev_yaw_belt
##                       26                       27                       28
##             var_yaw_belt            var_accel_arm             avg_roll_arm
##                       29                       43                       44
##          stddev_roll_arm             var_roll_arm            avg_pitch_arm
##                       45                       46                       47
##         stddev_pitch_arm            var_pitch_arm              avg_yaw_arm
##                       48                       49                       50
##           stddev_yaw_arm              var_yaw_arm        kurtosis_roll_arm
##                       51                       52                       62
##       kurtosis_picth_arm        kurtosis_yaw_arm        skewness_roll_arm
##                       63                       64                       65
##       skewness_pitch_arm        skewness_yaw_arm             max_roll_arm
##                       66                       67                       68
##            max_picth_arm              max_yaw_arm             min_roll_arm
##                       69                       70                       71
##            min_pitch_arm              min_yaw_arm       amplitude_roll_arm
##                       72                       73                       74
##      amplitude_pitch_arm        amplitude_yaw_arm   kurtosis_roll_dumbbell
##                       75                       76                       80
##  kurtosis_picth_dumbbell    kurtosis_yaw_dumbbell   skewness_roll_dumbbell
##                       81                       82                       83
##  skewness_pitch_dumbbell    skewness_yaw_dumbbell        max_roll_dumbbell
##                       84                       85                       86
##       max_picth_dumbbell        max_yaw_dumbbell        min_roll_dumbbell
##                       87                       88                       89
##       min_pitch_dumbbell        min_yaw_dumbbell  amplitude_roll_dumbbell
##                       90                       91                       92
## amplitude_pitch_dumbbell   amplitude_yaw_dumbbell       var_accel_dumbbell
##                       93                       94                       96
##        avg_roll_dumbbell     stddev_roll_dumbbell        var_roll_dumbbell
```

```
##                        97                        98                        99
##        avg_pitch_dumbbell     stddev_pitch_dumbbell        var_pitch_dumbbell
##                       100                       101                       102
##          avg_yaw_dumbbell       stddev_yaw_dumbbell          var_yaw_dumbbell
##                       103                       104                       105
##     kurtosis_roll_forearm    kurtosis_picth_forearm      kurtosis_yaw_forearm
##                       118                       119                       120
##     skewness_roll_forearm    skewness_pitch_forearm      skewness_yaw_forearm
##                       121                       122                       123
##          max_roll_forearm          max_picth_forearm           max_yaw_forearm
##                       124                       125                       126
##          min_roll_forearm          min_pitch_forearm           min_yaw_forearm
##                       127                       128                       129
##    amplitude_roll_forearm   amplitude_pitch_forearm     amplitude_yaw_forearm
##                       130                       131                       132
##         var_accel_forearm          avg_roll_forearm       stddev_roll_forearm
##                       134                       135                       136
##          var_roll_forearm          avg_pitch_forearm      stddev_pitch_forearm
##                       137                       138                       139
##         var_pitch_forearm           avg_yaw_forearm        stddev_yaw_forearm
##                       140                       141                       142
##           var_yaw_forearm
##                       143
```

```r
# Clean dataset by removing all columns with NA & spaces
trainingDat <- trainingDat[, colSums(is.na(trainingDat)) == 0]
testingDat <- testingDat[, colSums(is.na(testingDat)) == 0]
```

Subset training dataset into training & validation

Will subset the training dataset into 2 subsets, 75% of which is used for training the data and the remaining 25% for validating the data.

```r
set.seed(020317)
inTrain <- createDataPartition(trainingDat$classe, p = 0.75, list = FALSE)
trainingDat <- trainingDat[inTrain, ]
validatingDat <- trainingDat[-inTrain, ]
```

The dimension of the training & validating datasets are 14718 x 53 and 3681 x 53 respectively:

```r
dim(trainingDat); dim(validatingDat)
```

```
## [1] 14718    53
```

```
## [1] 3681   53
```

Model Training

We need to scale down the number of descriptors used for the model for simplicity and performance improvement yet still maintaining a minimum pof 95% covergae on variance. A few modeling techniques including decision tree, generalized boosting, k-nearest neighbor and Random Forest will be utilized to determine the best algorithm to use.

We will set k=5 for the trainControl function instead of the default setting of 10 to reduce the computing runtime. This function, used by the model fitting algorithm, set's 5-fold cross validation technique.

```r
control.parms <- trainControl(method="cv", number=5)
```

Principal component analysis (PCA) is run to identify correlated descriptors

```r
preProc <- preProcess(trainingDat[,-53],method="pca",thresh = 0.95)
preProc
```

```
## Created from 14718 samples and 52 variables
##
## Pre-processing:
##   - centered (52)
##   - ignored (0)
##   - principal component signal extraction (52)
##   - scaled (52)
##
## PCA needed 26 components to capture 95 percent of the variance
```

PCA shows that 26 components are needed to capture 95% of the variance. That is a 50% reduction from the original number of 52 descriptors, and which will use to build the model.

```r
# Creating 4 models from 26 descriptors and classe outcome
# Fitting training dataset with a decision tree model
predictedM <- predict(preProc, trainingDat)
dim(predictedM)
```

```
## [1] 14718    27
```

```r
time_st <- proc.time()
model_rpart <- train(classe ~ ., data=predictedM, method="rpart", trControl=control.parms)
# time to build rpart
(time_rpart <- proc.time() - time_st); time_st <- proc.time()
```

```
##    user  system elapsed
##   10.77    0.20   11.01
```

```r
# Fitting training dataset with a generalized boosting model
model_gbm <- train(classe ~ ., data=predictedM, method="gbm", trControl=control.parms)
```

```
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.0536
##      2        1.5743             nan     0.1000    0.0435
##      3        1.5470             nan     0.1000    0.0334
##      4        1.5256             nan     0.1000    0.0299
##      5        1.5069             nan     0.1000    0.0239
##      6        1.4912             nan     0.1000    0.0209
##      7        1.4770             nan     0.1000    0.0204
##      8        1.4643             nan     0.1000    0.0188
##      9        1.4524             nan     0.1000    0.0194
##     10        1.4397             nan     0.1000    0.0156
##     20        1.3502             nan     0.1000    0.0099
##     40        1.2383             nan     0.1000    0.0058
##     60        1.1634             nan     0.1000    0.0035
##     80        1.1062             nan     0.1000    0.0046
##    100        1.0603             nan     0.1000    0.0017
##    120        1.0221             nan     0.1000    0.0018
##    140        0.9908             nan     0.1000    0.0013
##    150        0.9757             nan     0.1000    0.0012
```

```
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.0858
##      2        1.5547             nan     0.1000    0.0618
##      3        1.5152             nan     0.1000    0.0512
##      4        1.4817             nan     0.1000    0.0389
##      5        1.4554             nan     0.1000    0.0389
##      6        1.4301             nan     0.1000    0.0325
##      7        1.4087             nan     0.1000    0.0287
##      8        1.3889             nan     0.1000    0.0285
##      9        1.3705             nan     0.1000    0.0270
##     10        1.3531             nan     0.1000    0.0243
##     20        1.2212             nan     0.1000    0.0125
##     40        1.0646             nan     0.1000    0.0095
##     60        0.9632             nan     0.1000    0.0042
##     80        0.8878             nan     0.1000    0.0038
##    100        0.8285             nan     0.1000    0.0026
##    120        0.7768             nan     0.1000    0.0015
##    140        0.7320             nan     0.1000    0.0020
##    150        0.7135             nan     0.1000    0.0015
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.0978
##      2        1.5451             nan     0.1000    0.0834
##      3        1.4922             nan     0.1000    0.0682
##      4        1.4494             nan     0.1000    0.0560
##      5        1.4127             nan     0.1000    0.0507
##      6        1.3794             nan     0.1000    0.0416
##      7        1.3516             nan     0.1000    0.0436
##      8        1.3237             nan     0.1000    0.0345
##      9        1.3001             nan     0.1000    0.0332
##     10        1.2776             nan     0.1000    0.0320
##     20        1.1218             nan     0.1000    0.0178
##     40        0.9451             nan     0.1000    0.0057
##     60        0.8340             nan     0.1000    0.0061
##     80        0.7532             nan     0.1000    0.0044
##    100        0.6859             nan     0.1000    0.0037
##    120        0.6288             nan     0.1000    0.0017
##    140        0.5836             nan     0.1000    0.0019
##    150        0.5624             nan     0.1000    0.0021
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.0556
##      2        1.5736             nan     0.1000    0.0362
##      3        1.5488             nan     0.1000    0.0385
##      4        1.5249             nan     0.1000    0.0294
##      5        1.5057             nan     0.1000    0.0265
##      6        1.4894             nan     0.1000    0.0215
##      7        1.4754             nan     0.1000    0.0193
##      8        1.4619             nan     0.1000    0.0176
##      9        1.4495             nan     0.1000    0.0170
##     10        1.4381             nan     0.1000    0.0156
##     20        1.3460             nan     0.1000    0.0109
##     40        1.2323             nan     0.1000    0.0059
```

```
##     60        1.1580             nan     0.1000    0.0041
##     80        1.1004             nan     0.1000    0.0030
##    100        1.0546             nan     0.1000    0.0019
##    120        1.0184             nan     0.1000    0.0019
##    140        0.9849             nan     0.1000    0.0013
##    150        0.9708             nan     0.1000    0.0012
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.0828
##      2        1.5582             nan     0.1000    0.0680
##      3        1.5149             nan     0.1000    0.0529
##      4        1.4812             nan     0.1000    0.0446
##      5        1.4525             nan     0.1000    0.0365
##      6        1.4274             nan     0.1000    0.0321
##      7        1.4067             nan     0.1000    0.0288
##      8        1.3872             nan     0.1000    0.0308
##      9        1.3676             nan     0.1000    0.0278
##     10        1.3503             nan     0.1000    0.0248
##     20        1.2161             nan     0.1000    0.0146
##     40        1.0630             nan     0.1000    0.0067
##     60        0.9590             nan     0.1000    0.0052
##     80        0.8823             nan     0.1000    0.0026
##    100        0.8205             nan     0.1000    0.0022
##    120        0.7725             nan     0.1000    0.0030
##    140        0.7301             nan     0.1000    0.0010
##    150        0.7114             nan     0.1000    0.0017
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1009
##      2        1.5442             nan     0.1000    0.0748
##      3        1.4963             nan     0.1000    0.0702
##      4        1.4523             nan     0.1000    0.0580
##      5        1.4154             nan     0.1000    0.0503
##      6        1.3835             nan     0.1000    0.0490
##      7        1.3525             nan     0.1000    0.0413
##      8        1.3255             nan     0.1000    0.0362
##      9        1.3019             nan     0.1000    0.0317
##     10        1.2798             nan     0.1000    0.0264
##     20        1.1232             nan     0.1000    0.0172
##     40        0.9400             nan     0.1000    0.0080
##     60        0.8294             nan     0.1000    0.0070
##     80        0.7476             nan     0.1000    0.0040
##    100        0.6834             nan     0.1000    0.0031
##    120        0.6289             nan     0.1000    0.0019
##    140        0.5848             nan     0.1000    0.0016
##    150        0.5625             nan     0.1000    0.0033
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.0520
##      2        1.5768             nan     0.1000    0.0390
##      3        1.5512             nan     0.1000    0.0334
##      4        1.5291             nan     0.1000    0.0286
##      5        1.5108             nan     0.1000    0.0256
##      6        1.4942             nan     0.1000    0.0212
```

```
##      7        1.4804             nan     0.1000    0.0202
##      8        1.4669             nan     0.1000    0.0172
##      9        1.4554             nan     0.1000    0.0165
##     10        1.4446             nan     0.1000    0.0178
##     20        1.3566             nan     0.1000    0.0092
##     40        1.2441             nan     0.1000    0.0073
##     60        1.1683             nan     0.1000    0.0043
##     80        1.1098             nan     0.1000    0.0024
##    100        1.0639             nan     0.1000    0.0022
##    120        1.0255             nan     0.1000    0.0019
##    140        0.9931             nan     0.1000    0.0006
##    150        0.9782             nan     0.1000    0.0012
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.0790
##      2        1.5602             nan     0.1000    0.0649
##      3        1.5191             nan     0.1000    0.0489
##      4        1.4877             nan     0.1000    0.0434
##      5        1.4589             nan     0.1000    0.0369
##      6        1.4344             nan     0.1000    0.0353
##      7        1.4126             nan     0.1000    0.0298
##      8        1.3922             nan     0.1000    0.0272
##      9        1.3741             nan     0.1000    0.0254
##     10        1.3575             nan     0.1000    0.0236
##     20        1.2278             nan     0.1000    0.0152
##     40        1.0712             nan     0.1000    0.0092
##     60        0.9697             nan     0.1000    0.0055
##     80        0.8909             nan     0.1000    0.0043
##    100        0.8303             nan     0.1000    0.0031
##    120        0.7814             nan     0.1000    0.0022
##    140        0.7392             nan     0.1000    0.0020
##    150        0.7193             nan     0.1000    0.0017
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1037
##      2        1.5427             nan     0.1000    0.0771
##      3        1.4949             nan     0.1000    0.0624
##      4        1.4547             nan     0.1000    0.0572
##      5        1.4181             nan     0.1000    0.0435
##      6        1.3902             nan     0.1000    0.0426
##      7        1.3620             nan     0.1000    0.0418
##      8        1.3348             nan     0.1000    0.0364
##      9        1.3117             nan     0.1000    0.0328
##     10        1.2894             nan     0.1000    0.0314
##     20        1.1281             nan     0.1000    0.0153
##     40        0.9493             nan     0.1000    0.0080
##     60        0.8337             nan     0.1000    0.0053
##     80        0.7470             nan     0.1000    0.0025
##    100        0.6849             nan     0.1000    0.0027
##    120        0.6294             nan     0.1000    0.0028
##    140        0.5814             nan     0.1000    0.0023
##    150        0.5607             nan     0.1000    0.0025
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
```

```
##      1        1.6094             nan     0.1000    0.0535
##      2        1.5747             nan     0.1000    0.0437
##      3        1.5475             nan     0.1000    0.0350
##      4        1.5245             nan     0.1000    0.0277
##      5        1.5060             nan     0.1000    0.0268
##      6        1.4884             nan     0.1000    0.0190
##      7        1.4751             nan     0.1000    0.0211
##      8        1.4608             nan     0.1000    0.0190
##      9        1.4478             nan     0.1000    0.0156
##     10        1.4369             nan     0.1000    0.0184
##     20        1.3472             nan     0.1000    0.0084
##     40        1.2347             nan     0.1000    0.0058
##     60        1.1607             nan     0.1000    0.0039
##     80        1.1028             nan     0.1000    0.0023
##    100        1.0574             nan     0.1000    0.0024
##    120        1.0194             nan     0.1000    0.0008
##    140        0.9873             nan     0.1000    0.0011
##    150        0.9728             nan     0.1000    0.0008
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.0869
##      2        1.5552             nan     0.1000    0.0631
##      3        1.5156             nan     0.1000    0.0518
##      4        1.4814             nan     0.1000    0.0446
##      5        1.4519             nan     0.1000    0.0367
##      6        1.4284             nan     0.1000    0.0347
##      7        1.4051             nan     0.1000    0.0322
##      8        1.3842             nan     0.1000    0.0289
##      9        1.3656             nan     0.1000    0.0295
##     10        1.3468             nan     0.1000    0.0248
##     20        1.2165             nan     0.1000    0.0135
##     40        1.0662             nan     0.1000    0.0087
##     60        0.9636             nan     0.1000    0.0050
##     80        0.8891             nan     0.1000    0.0035
##    100        0.8275             nan     0.1000    0.0029
##    120        0.7753             nan     0.1000    0.0031
##    140        0.7332             nan     0.1000    0.0020
##    150        0.7140             nan     0.1000    0.0015
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.0971
##      2        1.5459             nan     0.1000    0.0748
##      3        1.4966             nan     0.1000    0.0724
##      4        1.4511             nan     0.1000    0.0589
##      5        1.4135             nan     0.1000    0.0478
##      6        1.3827             nan     0.1000    0.0458
##      7        1.3531             nan     0.1000    0.0397
##      8        1.3271             nan     0.1000    0.0392
##      9        1.3009             nan     0.1000    0.0346
##     10        1.2781             nan     0.1000    0.0315
##     20        1.1204             nan     0.1000    0.0178
##     40        0.9420             nan     0.1000    0.0098
##     60        0.8272             nan     0.1000    0.0051
##     80        0.7451             nan     0.1000    0.0042
```

```
##    100        0.6825             nan     0.1000    0.0033
##    120        0.6271             nan     0.1000    0.0030
##    140        0.5776             nan     0.1000    0.0023
##    150        0.5579             nan     0.1000    0.0013
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.0555
##      2        1.5744             nan     0.1000    0.0388
##      3        1.5491             nan     0.1000    0.0375
##      4        1.5253             nan     0.1000    0.0286
##      5        1.5070             nan     0.1000    0.0247
##      6        1.4921             nan     0.1000    0.0222
##      7        1.4779             nan     0.1000    0.0224
##      8        1.4643             nan     0.1000    0.0182
##      9        1.4528             nan     0.1000    0.0176
##     10        1.4411             nan     0.1000    0.0155
##     20        1.3523             nan     0.1000    0.0098
##     40        1.2413             nan     0.1000    0.0063
##     60        1.1651             nan     0.1000    0.0032
##     80        1.1076             nan     0.1000    0.0038
##    100        1.0601             nan     0.1000    0.0020
##    120        1.0227             nan     0.1000    0.0016
##    140        0.9912             nan     0.1000    0.0012
##    150        0.9768             nan     0.1000    0.0011
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.0848
##      2        1.5544             nan     0.1000    0.0629
##      3        1.5151             nan     0.1000    0.0516
##      4        1.4808             nan     0.1000    0.0394
##      5        1.4538             nan     0.1000    0.0361
##      6        1.4306             nan     0.1000    0.0338
##      7        1.4078             nan     0.1000    0.0288
##      8        1.3890             nan     0.1000    0.0290
##      9        1.3700             nan     0.1000    0.0277
##     10        1.3525             nan     0.1000    0.0270
##     20        1.2210             nan     0.1000    0.0128
##     40        1.0637             nan     0.1000    0.0093
##     60        0.9630             nan     0.1000    0.0040
##     80        0.8886             nan     0.1000    0.0031
##    100        0.8276             nan     0.1000    0.0033
##    120        0.7758             nan     0.1000    0.0012
##    140        0.7324             nan     0.1000    0.0017
##    150        0.7144             nan     0.1000    0.0011
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1001
##      2        1.5451             nan     0.1000    0.0765
##      3        1.4968             nan     0.1000    0.0694
##      4        1.4532             nan     0.1000    0.0577
##      5        1.4169             nan     0.1000    0.0497
##      6        1.3846             nan     0.1000    0.0441
##      7        1.3560             nan     0.1000    0.0387
##      8        1.3312             nan     0.1000    0.0347
```

```
##      9        1.3069             nan     0.1000    0.0343
##     10        1.2848             nan     0.1000    0.0304
##     20        1.1274             nan     0.1000    0.0165
##     40        0.9424             nan     0.1000    0.0085
##     60        0.8308             nan     0.1000    0.0068
##     80        0.7479             nan     0.1000    0.0030
##    100        0.6835             nan     0.1000    0.0021
##    120        0.6262             nan     0.1000    0.0034
##    140        0.5802             nan     0.1000    0.0022
##    150        0.5581             nan     0.1000    0.0015
##
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        1.6094             nan     0.1000    0.1003
##      2        1.5459             nan     0.1000    0.0850
##      3        1.4919             nan     0.1000    0.0645
##      4        1.4506             nan     0.1000    0.0556
##      5        1.4154             nan     0.1000    0.0484
##      6        1.3844             nan     0.1000    0.0421
##      7        1.3572             nan     0.1000    0.0412
##      8        1.3312             nan     0.1000    0.0353
##      9        1.3078             nan     0.1000    0.0347
##     10        1.2858             nan     0.1000    0.0325
##     20        1.1287             nan     0.1000    0.0162
##     40        0.9504             nan     0.1000    0.0094
##     60        0.8391             nan     0.1000    0.0056
##     80        0.7586             nan     0.1000    0.0039
##    100        0.6934             nan     0.1000    0.0023
##    120        0.6399             nan     0.1000    0.0020
##    140        0.5928             nan     0.1000    0.0017
##    150        0.5727             nan     0.1000    0.0030
```

```r
(time_gbm <- proc.time() - time_st); time_st <- proc.time()
```

```
##    user  system elapsed
##  320.84    0.35  321.53
```

```r
# Fitting training dataset with a K-nearest model
model_knn <- train(classe ~ ., data=predictedM, method="knn", trControl=control.parms)
(time_knn <- proc.time() - time_st); time_st <- proc.time()
```

```
##    user  system elapsed
##   29.74    0.00   29.74
```

```r
# Fitting training dataset with a Random Forest model
model_rf <- train(classe ~ ., data=predictedM, method="rf", trControl=control.parms)
(time_rf <- proc.time() - time_st)
```

```
##    user  system elapsed
##  644.53    7.50  652.81
```

```r
# Random Forest has the maximum time to build the model
(max(c(time_rf[[1]], time_knn[[1]], time_gbm[[1]], time_rpart[[1]])))
```

```
## [1] 644.53
```

Cross Validation

```r
print(model_rpart, digits = 3)
```

```
## CART
##
## 14718 samples
##    26 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11772, 11775, 11775, 11774, 11776
## Resampling results across tuning parameters:
##
##   cp      Accuracy  Kappa
##   0.0259  0.381     0.1802
##   0.0574  0.350     0.1126
##   0.0702  0.326     0.0665
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.0259.
```

```r
conf_rpart <- confusionMatrix(validatingDat$classe,predict(model_rpart,predict(preProc,validatingDat[,-5
# Accuracy;
conf_rpart$overall[1]
```

```
##  Accuracy
## 0.3849497
```

```r
# Error rate;
1-conf_rpart$overall[1]
```

```
##  Accuracy
## 0.6150503
```

```r
print(model_gbm, digits = 3)
```

```
## Stochastic Gradient Boosting
##
## 14718 samples
##    26 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11774, 11775, 11773, 11776, 11774
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy  Kappa
##   1                   50      0.557     0.429
##   1                  100      0.616     0.509
##   1                  150      0.645     0.547
##   2                   50      0.658     0.563
##   2                  100      0.726     0.652
##   2                  150      0.760     0.696
##   3                   50      0.709     0.631
##   3                  100      0.774     0.714
```

```
##   3                   150      0.813     0.763
##
## 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.
```

```r
conf_gbm <- confusionMatrix(validatingDat$classe,predict(model_gbm,predict(preProc,validatingDat[,-53])
# Accuracy;
conf_gbm$overall[1]
```

```
##  Accuracy
## 0.8606357
```

```r
# Error rate;
1-conf_gbm$overall[1]
```

```
##  Accuracy
## 0.1393643
```

```r
print(model_knn, digits = 3)
```

```
## k-Nearest Neighbors
##
## 14718 samples
##    26 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11774, 11773, 11775, 11775, 11775
## Resampling results across tuning parameters:
##
##   k  Accuracy  Kappa
##   5  0.950     0.937
##   7  0.937     0.920
##   9  0.927     0.907
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was k = 5.
```

```r
conf_knn <- confusionMatrix(validatingDat$classe,predict(model_knn,predict(preProc,validatingDat[,-53])
# Accuracy;
conf_knn$overall[1]
```

```
##  Accuracy
## 0.9828851
```

```r
# Error rate;
1-conf_knn$overall[1]
```

```
##   Accuracy
## 0.01711491
```

```r
print(model_rf, digits = 3)
```

```
## Random Forest
```

```
##
## 14718 samples
##    26 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11775, 11773, 11775, 11775, 11774
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##    2    0.973     0.965
##   14    0.969     0.961
##   26    0.959     0.948
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```r
conf_rf <- confusionMatrix(validatingDat$classe,predict(model_rf,predict(preProc,validatingDat[,-53])))
# Accuracy;
conf_rf$overall[1]
```

```
## Accuracy
##        1
```

```r
# Error rate;
1-conf_rf$overall[1]
```

```
## Accuracy
##        0
```

```r
# Maximum accuracy across models goes to Random Forest
max(c(conf_rpart$overall[1], conf_gbm$overall[1], conf_knn$overall[1], conf_rf$overall[1]))
```

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

```r
conf_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1061    0    0    0    0
##          B    0  710    0    0    0
##          C    0    0  632    0    0
##          D    0    0    0  614    0
##          E    0    0    0    0  664
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.999, 1)
##     No Information Rate : 0.2882
##     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             0.2882   0.1929   0.1717   0.1668   0.1804
## Detection Rate         0.2882   0.1929   0.1717   0.1668   0.1804
## Detection Prevalence   0.2882   0.1929   0.1717   0.1668   0.1804
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Random forest method shows much better results compared to the other algorithms. We build the models using 26 predictors compared to a 2 fold performance degradation if we were to build it with all 52 variables. All the prediction outcomes fall on the diagonal in the table with a 100% accuracy rate and no out-of-sample error.

Final Testing

We will predict the outcome by running the Random Forest model on the testing dataset.

```
(conclusion <- predict(model_rf,predict(preProc,testingDat[,-53])))
```

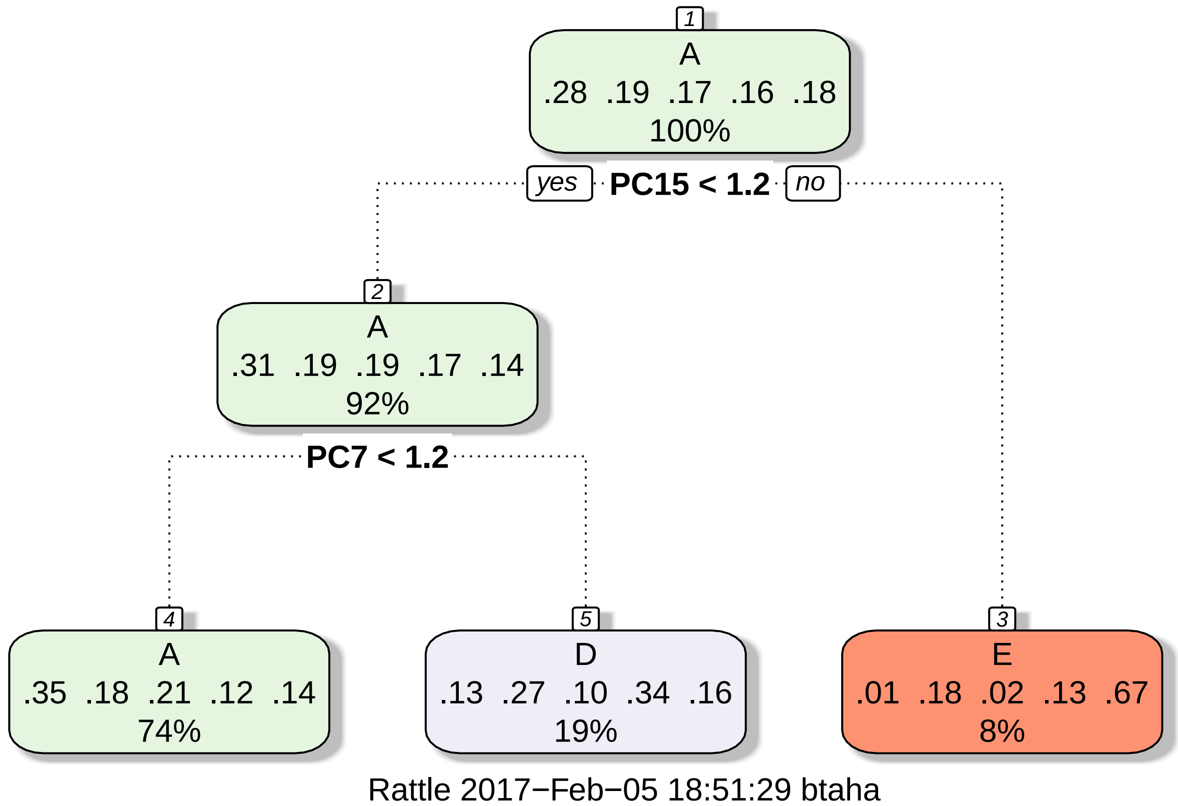
```
##  [1] B A B A A B D B A A B C B A E E A B B B
## Levels: A B C D E
```

Conclusion

We can predict the way people are performing the excercise with a very high degree of accuracy using the Random forest model. Random Forest model build took more compute time than the others but it is a secondary order effect compared to the accuracy obtained.

Appendix

```
fancyRpartPlot(model_rpart$finalModel)
```


1
A
.28 .19 .17 .16 .18
100%
yes
PC15 < 1.2
no
2
A
.31 .19 .19 .17 .14
92%
PC7 < 1.2
4
A
.35 .18 .21 .12 .14
74%
5
D
.13 .27 .10 .34 .16
19%
3
E
.01 .18 .02 .13 .67
8%
Rattle 2017−Feb−05 18:51:29 btaha