Coursera Practical Machine Learning Project

Masamichi Kawarabayashi

Dec 30, 2018

This is a report of final project of **Practical Machine Learning** course in Coursera.

Introduction

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. In this project.

The goal will be to 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. More information is available from the website here: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

```r
library(caret)
```

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

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

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.5.2
```

```
## 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
```

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

```
## Warning: package 'rattle' was built under R version 3.5.2
```

```
## Rattle: A free graphical interface for data science with R.
## バージョン 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## 'rattle()' と入力して、データを多角的に分析します。
```

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

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

Read and clean up data

Read data from the provided URL, replacing blanks and unexpected values to NA. Then create data set which columns has 5% less NA percentage. Also first 7 columns are removed because they are not used in this analysis.

```
training.org <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmach
learn/pml-training.csv"), header=TRUE, na.strings=c("NA","#DIV/0!",""))
testing.org  <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmach
learn/pml-testing.csv"), header=TRUE, na.strings=c("NA","#DIV/0!",""))
str(training.org$class)
```

```
##  Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
dim(training.org); dim(testing.org)
```

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

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

```
training.org.clean <- select(training.org, which(as.logical(sapply(training.
org, function(y) sum(is.na(y)))/dim(training.org)[1]<0.05)))
testing.org.clean  <- select(testing.org, which(as.logical(sapply(testing.or
g, function(y) sum(is.na(y)))/dim(testing.org)[1]<0.05)))
training.org.clean <- training.org.clean[, -c(1:7)]
testing.org.clean <- testing.org.clean[, -c(1:7)]
dim(training.org.clean); dim(testing.org.clean)
```

```
## [1] 19622    53
```

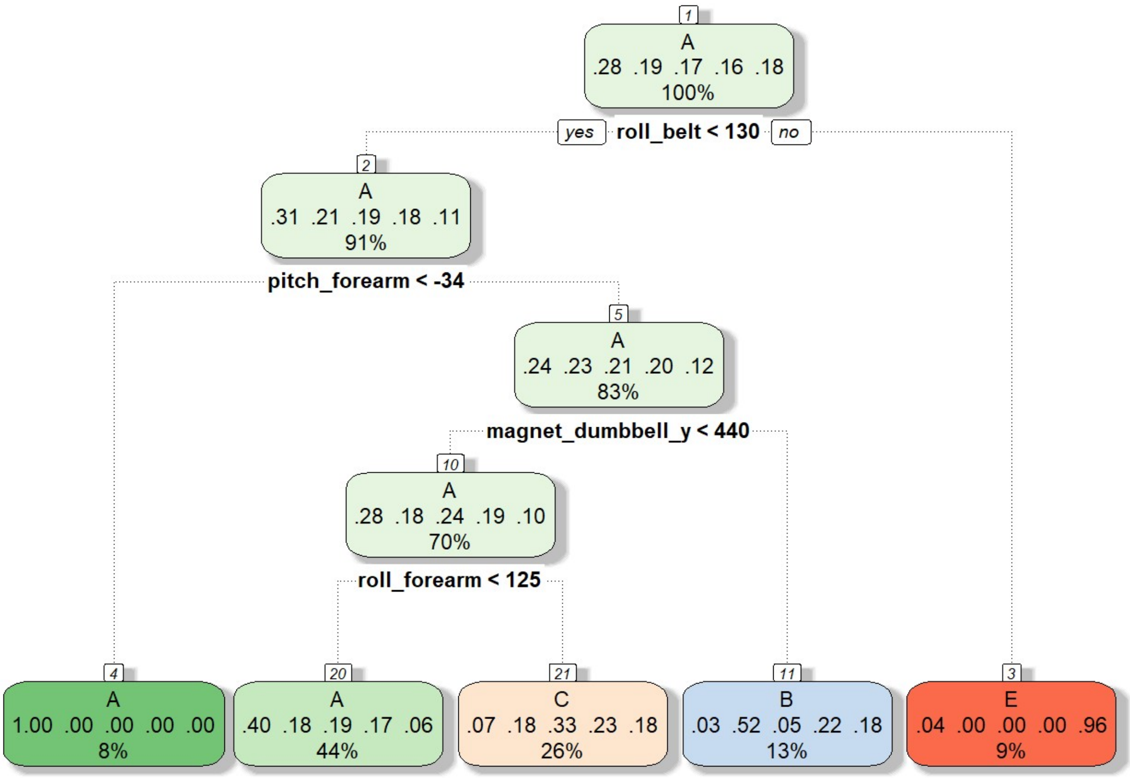
```
## [1] 20 53
```

```
inTrain <- createDataPartition(training.org.clean$classe, p=0.8, list=FALSE)
training <- training.org.clean[inTrain,]
testing <- training.org.clean[-inTrain,]
```

Now the cleaned training/test data have 53 columns. And training and test set are prepared.

Analysis with classification tree

```
set.seed(1960)
model.ct <- train(classe~., data=training, method="rpart", trControl=trainCo
ntrol(method="cv", number=5))
fancyRpartPlot(model.ct$finalModel)
```



Rattle 2018-12-30 23:41:29 a5031286

```
pred.ct <- predict(model.ct, testing)
confmt <- confusionMatrix(testing$classe, pred.ct)
confmt$table
```

```
##             Reference
## Prediction    A   B   C   D   E
##          A 989  13  95   0  19
##          B 317 257 185   0   0
##          C 308  18 358   0   0
##          D 293 126 224   0   0
##          E  99 104 191   0 327
```

```
confmt$overall["Accuracy"]
```

```
##  Accuracy
## 0.4922253
```

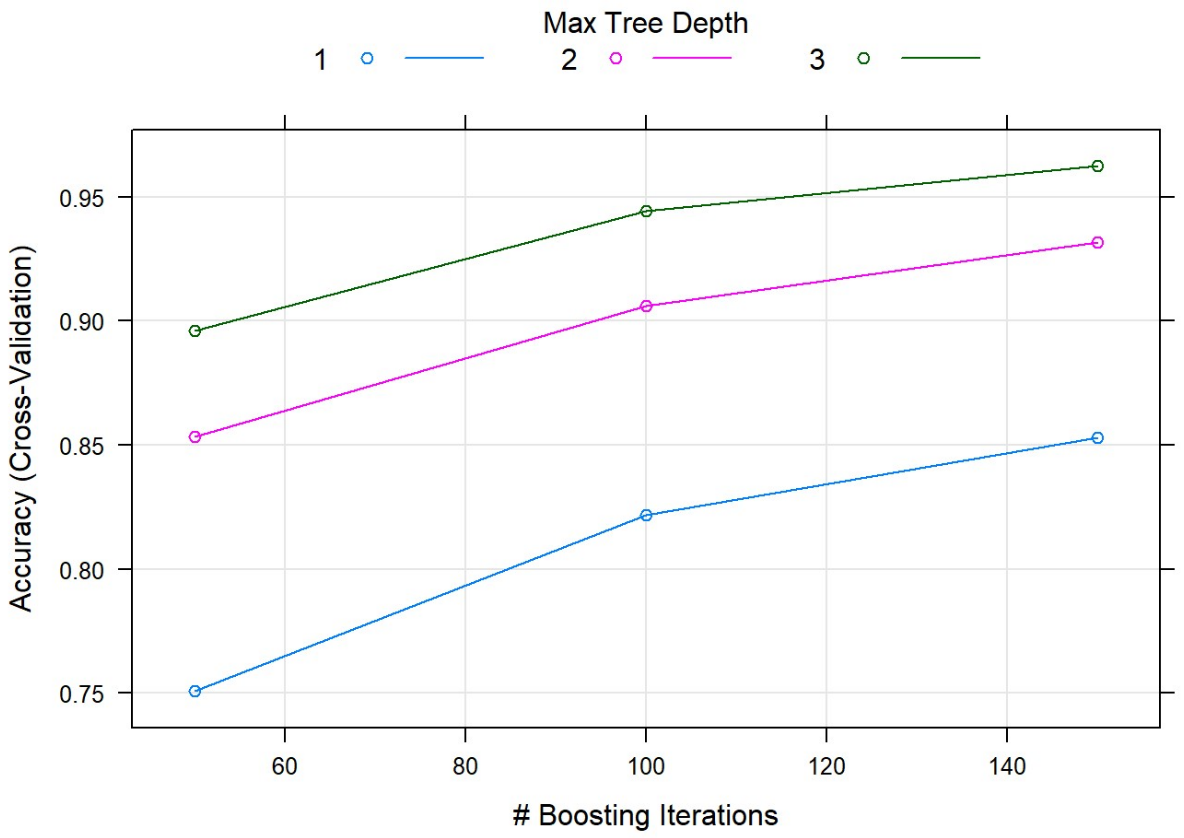
The accuracy is only 49%, so this model can not predict the outcome **classe** well.

Analysis with gradient boosting method

```
set.seed(1960)
model.gbm <- train(classe~., data=training, method="gbm", trControl=trainCon
trol(method="cv", number=5), verbose=FALSE)
model.gbm
```

```
## Stochastic Gradient Boosting
##
## 15699 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 12560, 12558, 12557, 12561, 12560
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7507463  0.6840178
##   1                  100      0.8219642  0.7746678
##   1                  150      0.8528558  0.8138367
##   2                   50      0.8534925  0.8143617
##   2                  100      0.9059798  0.8810026
##   2                  150      0.9317134  0.9135767
##   3                   50      0.8959795  0.8683358
##   3                  100      0.9443897  0.9296276
##   3                  150      0.9624172  0.9524479
##
## 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.
```

```
plot(model.gbm)
```



```
pred.gbm <- predict(model.gbm, testing)
confmt <- confusionMatrix(testing$classe, pred.gbm)
confmt$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1095   14    5    2    0
##          B   19  715   23    2    0
##          C    0   19  657    7    1
##          D    0    5   29  605    4
##          E    4   10    7   11  689
```

```
confmt$overall["Accuracy"]
```

```
##  Accuracy
## 0.9587051
```

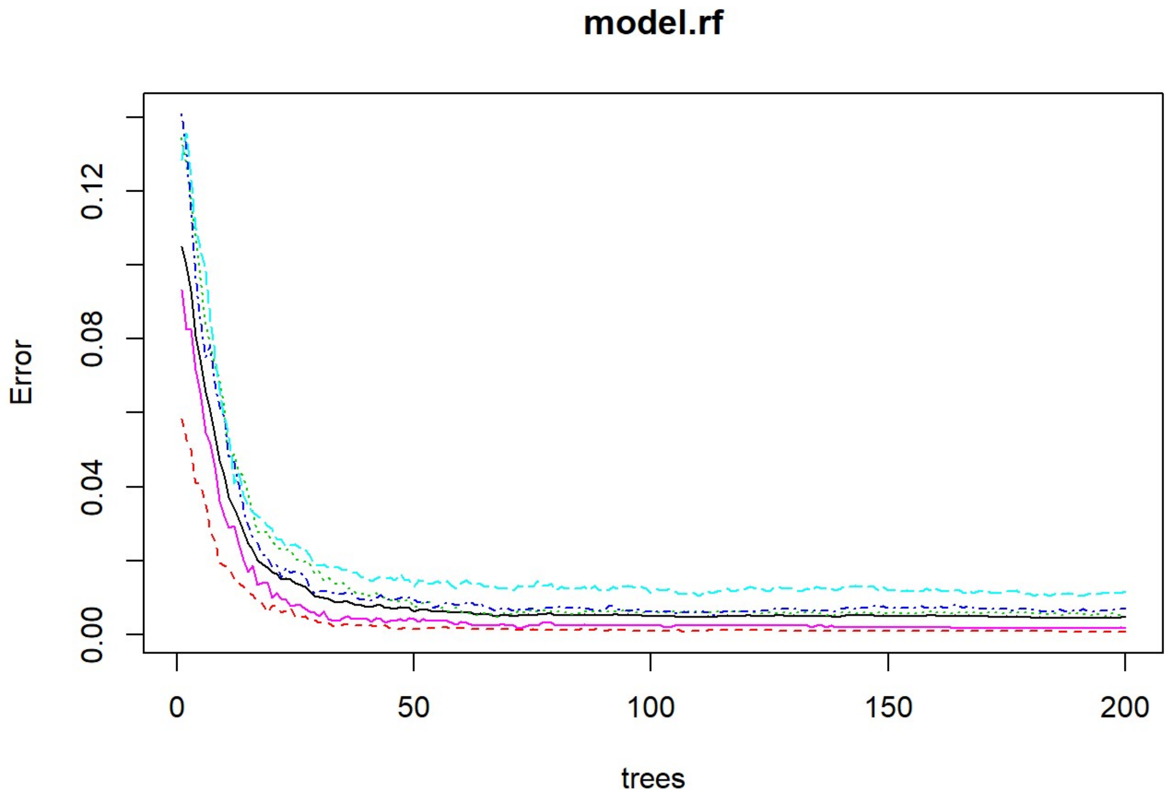
The accuracy is 96%, so this model can predict the outcome **classe** well.

Analysis with random forest

```
set.seed(1960)
model.rf <- randomForest(classe~.,data=training,ntree=200,importance=TRUE)
model.rf
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training, ntree = 200,      im
portance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 200
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.47%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4461    2    0    0    1 0.000672043
## B   13 3021    4    0    0 0.005595787
## C    0   15 2719    4    0 0.006939372
## D    0    0   28 2543    2 0.011659541
## E    0    0    2    3 2881 0.001732502
```

```
plot(model.rf)
```




```
pred.rf <- predict(model.rf, testing)
confmt <- confusionMatrix(testing$classe, pred.rf)
confmt$table
```

```
##             Reference
## Prediction    A    B    C    D    E
##            A 1116    0    0    0    0
##            B    1  756    2    0    0
##            C    0    1  683    0    0
##            D    0    0    2  639    2
##            E    0    0    0    5  716
```

```
confmt$overall["Accuracy"]
```

```
##  Accuracy
## 0.9966862
```

The accuracy is 99.6%, so this model can predict the outcome **classe** well.

Important variables

The importance of the variables are shown below.

```
varImp(model.rf)
```

```
##                              A        B         C         D         E
## roll_belt            23.646850 26.06038 27.904326 27.757281 26.730684
## pitch_belt           19.568734 32.17015 26.001407 21.953129 21.903986
## yaw_belt             29.535222 29.55428 31.207846 31.596539 24.127844
## total_accel_belt      8.093842 10.18173  8.472693  8.618195  9.156145
## gyros_belt_x         11.947743 11.61141 12.281680  8.963509  9.711916
## gyros_belt_y          6.809356 11.14313 11.252702  8.652334 10.635152
## gyros_belt_z         13.419486 17.75496 17.039032 13.477530 17.054721
## accel_belt_x          9.967855 12.01545 11.566798  8.906671  9.539169
## accel_belt_y          9.986120 12.72311  9.298537 10.215625  8.230196
## accel_belt_z         15.542653 17.97151 17.506655 14.341672 12.696413
## magnet_belt_x        12.168719 18.21943 16.021707 15.220771 14.894789
## magnet_belt_y        14.371426 17.44835 17.319835 17.310882 16.238022
## magnet_belt_z        14.084587 15.86296 14.709075 17.594504 14.062241
## roll_arm             13.979049 21.27786 18.327590 17.561760 14.359072
## pitch_arm            14.082142 15.87867 15.861693 13.327157 11.485843
## yaw_arm              14.330518 17.64171 15.728978 16.683617 11.705515
## total_accel_arm       7.559544 15.24054 13.948417 13.102138 12.794688
## gyros_arm_x          12.183780 16.56456 16.962912 16.883764 14.096681
## gyros_arm_y          14.081644 21.12851 15.914677 17.860645 13.142503
## gyros_arm_z           8.953416 10.09192 10.934105  9.832125  7.161455
## accel_arm_x          10.638184 12.14769 12.035009 13.481810  9.774446
## accel_arm_y          11.623622 14.00730 10.878198 11.440424 12.075734
## accel_arm_z           7.793306 11.56236 13.426963 13.305773 12.531113
## magnet_arm_x         11.237484 10.43510 11.779654 11.929499 10.233181
## magnet_arm_y          7.639705 10.65991 11.481156 13.751240  9.674266
## magnet_arm_z         15.400356 18.64638 16.557933 15.271325 13.661427
## roll_dumbbell        16.111444 17.82545 18.533913 18.378051 17.092101
## pitch_dumbbell        8.424021 12.79061 11.366916  8.340915  9.352962
## yaw_dumbbell         11.144873 13.76591 13.831007 12.909346 14.807507
## total_accel_dumbbell 12.044093 14.64275 13.205121 14.018378 16.359459
## gyros_dumbbell_x     11.634500 18.16268 14.084777 14.956471 13.476616
## gyros_dumbbell_y     13.275800 15.02854 17.241546 14.903674 13.461721
## gyros_dumbbell_z     12.017996 17.34367 13.535593 11.146685 10.993782
## accel_dumbbell_x     10.631245 14.19931 12.632657 11.880459 12.552219
## accel_dumbbell_y     16.487717 17.60288 20.297019 18.083093 18.512914
## accel_dumbbell_z     14.046763 16.20681 16.238067 16.197503 19.053651
## magnet_dumbbell_x    15.166885 17.03835 19.383258 16.782638 13.829157
## magnet_dumbbell_y    24.493642 23.24523 27.860629 23.161252 20.176702
## magnet_dumbbell_z    26.829429 26.19571 31.023640 23.834283 22.855098
## roll_forearm         18.179846 16.69997 19.742737 15.039391 15.725379
## pitch_forearm        19.554739 22.67000 25.065905 21.457274 21.912198
## yaw_forearm          12.814449 13.92601 13.880489 12.186595 13.131799
## total_accel_forearm  11.723735 13.62380 14.263923 10.729528 11.104177
## gyros_forearm_x       9.024761 12.31116 11.468527 12.424978  9.228086
## gyros_forearm_y      12.616397 18.64946 17.149274 15.616780 13.531652
## gyros_forearm_z      10.886130 16.35829 15.815199 11.738507 13.136722
## accel_forearm_x      11.469211 15.69424 15.209703 17.042583 13.814170
## accel_forearm_y      12.208144 15.01002 14.750909 11.684624 14.625100
## accel_forearm_z      11.294911 13.90804 16.741209 14.399026 15.081610
## magnet_forearm_x     10.304391 16.40070 13.710748 13.244713 16.838743
## magnet_forearm_y     12.598711 13.53023 15.472127 13.292635 12.807140
## magnet_forearm_z     15.283628 18.75752 18.310507 17.188564 15.807670
```

Conclusion

Based on the above trials, **random forest** model is the best one to predict it. So predict **classe** with original test (validation) data.

```
pred.testing <- predict(model.rf, testing.org.clean)
pred.testing
```

```
##  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
```