Classification of five behaviors based on quantified self movement data

Yiyun

March 11, 2016

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 is 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://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Contents

1. How to build the model

2. How to use the cross-validation

3. Estemated out of sample error

4. Compare the prediction accuracy of different models

5. Validation of the predicted results using test data

Initiliazation: update your variables here

```
library(lattice)
library(ggplot2)
library(caret)
library(rpart)
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(MASS)
library(mlbench)
```

```r
library(class)
library(randomForest)

## 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

library(klaR)
Training_Link <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
Test_Link <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
Traning_File_Name <- "pml-training.csv"
Test_File_Name <- "pml-testing.csv"
DataFolder <- "./Human Activity Project/" # folder where all your data are saved
```

Data Acquisition: all data are saved under the DataFolder

```r
if (!file.exists(DataFolder)) dir.create(DataFolder)
setwd(DataFolder)
if (!file.exists(Traning_File_Name)) download.file(url = Training_Link, destfile = paste(DataFolder, Traning_File_Name, sep = ""))
if (!file.exists(Test_File_Name)) download.file(url = Test_Link, destfile = paste(DataFolder, Test_File_Name, sep = ""))
Training_Data <- read.csv(file = Traning_File_Name)
Test_Data <- read.csv(file = Test_File_Name)
```

Preprocessing 1: remove non-features and transform data to numeric type

```r
Training_Data[, 7:159] <- sapply(Training_Data[, 7:159], as.numeric)
Test_Data[, 7:159] <- sapply(Test_Data[, 7:159], as.numeric)
Training_Data <- Training_Data[8:160]
Test_Data <- Test_Data[8:160]
```

Preprocessing 2: remove NA features

```r
nas <- is.na(apply(Test_Data,2,sum))
Test_Data <- Test_Data[,!nas]
dim(Test_Data)

## [1] 20 53

Training_Data <- Training_Data[,!nas]
```

Create crossvalidation set in variable validation, the rest is training set in variable buildData

70% of the training data will be used to train the model, 30% will be used for prediction and estimate

the out of sample error.

```
inBuild <- createDataPartition(Training_Data$classe, p = 0.7, list = FALSE)
validation <- Training_Data[-inBuild[,1],]
buildData <- Training_Data[inBuild[,1],]
```

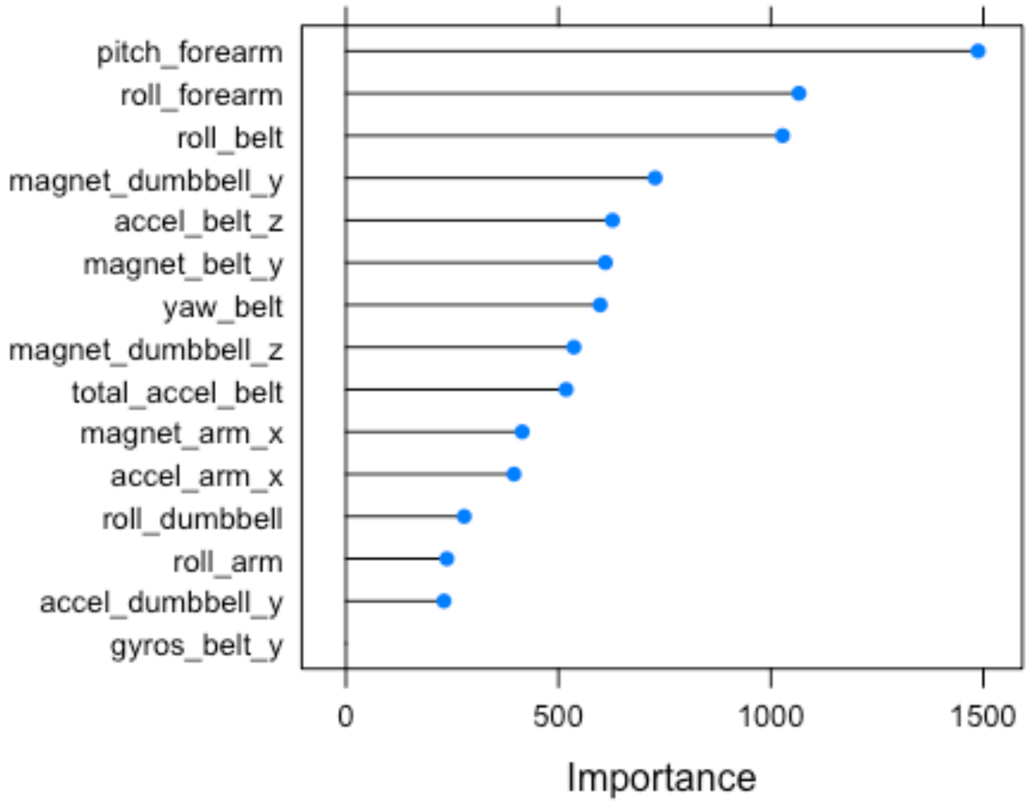
Rank feature importance and select important features for model training

In this case, there are less than 20 features

```
model <- train(classe~., data = buildData, method = "rpart")
importance <- varImp(model, scale = FALSE)
print(importance, top = 15)
```

```
## rpart variable importance
##
##   only 15 most important variables shown (out of 52)
##
##                     Overall
## pitch_forearm        1487.7
## roll_forearm         1066.1
## roll_belt            1027.6
## magnet_dumbbell_y     727.6
## accel_belt_z          627.3
## magnet_belt_y         610.5
## yaw_belt              598.3
## magnet_dumbbell_z     536.5
## total_accel_belt      518.0
## magnet_arm_x          414.7
## accel_arm_x           395.4
## roll_dumbbell         278.3
## roll_arm              237.4
## accel_dumbbell_y      230.6
## accel_belt_y            0.0
```

```
plot(importance, top = 15)
```




```
ImpVariables <- c("pitch_forearm",
                  "roll_forearm",
                  "roll_belt",
                  "magnet_dumbbell_y",
                  "accel_belt_z",
                  "magnet_belt_y",
                  "yaw_belt",
                  "magnet_dumbbell_z",
                  "total_accel_belt",
                  "magnet_arm_x",
                  "accel_arm_x",
                  "roll_dumbbell",
                  "accel_dumbbell_y",
                  "magnet_dumbbell_x",
                  "total_accel_dumbbell",
                  "pitch_belt",
                  "accel_dumbbell_x",
                  "accel_forearm_x")

important_features <- buildData[, colnames(buildData) %in% c(ImpVariables,
"classe")]
```

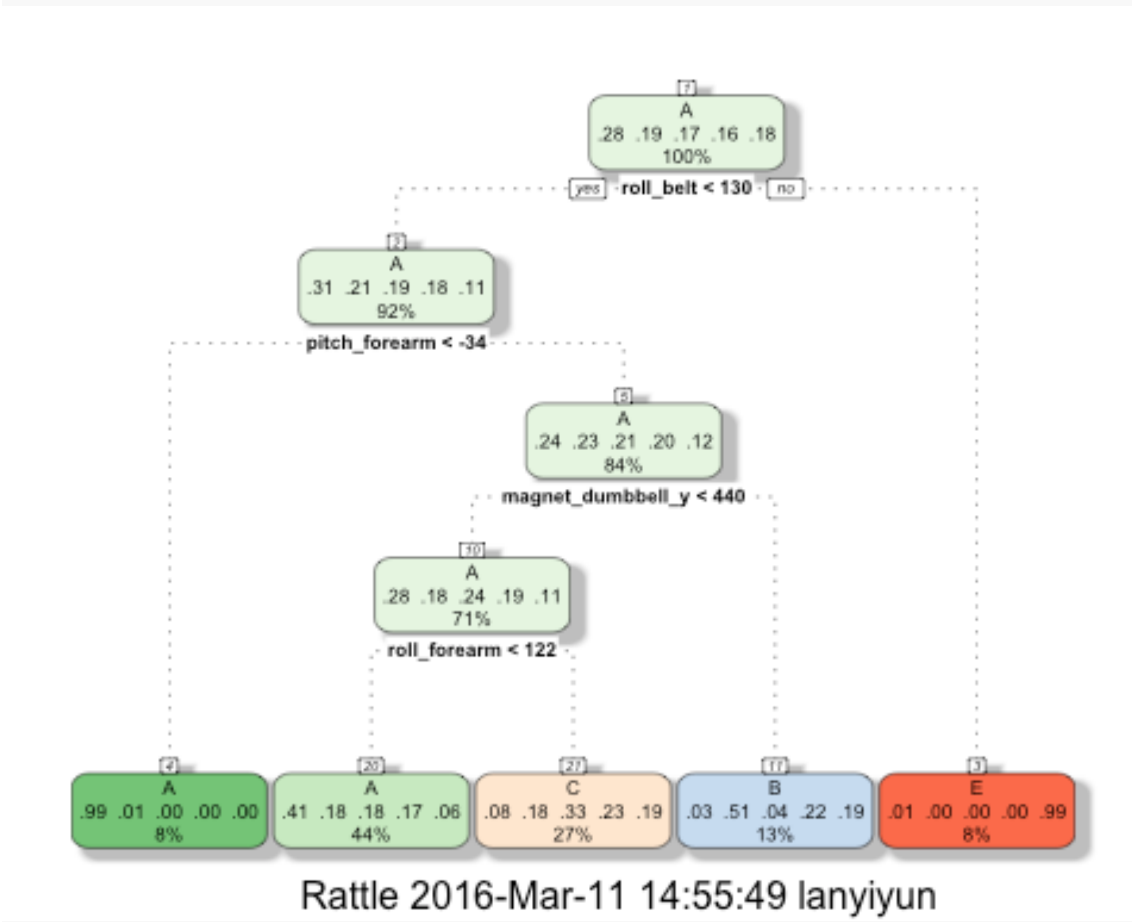
```
validation_features <- validation[, colnames(validation) %in% c(ImpVariables,
"classe")]
```

Use selected features to 1) train the the models and predict the outcome with validation dataset

Models used: 1) Recursive partitioning 2) linear discriminant analysis 3) random forest

1) Recursive partitioning

```
modFit_rpart <- train(classe~., method = "rpart", data = important_features)
fancyRpartPlot(modFit_rpart$finalModel)
```




```
pred_Rpart <- predict(modFit_rpart, validation_features)
Matrix_Rpart <- confusionMatrix(pred_Rpart, validation$classe)
```

2) linear discriminant analysis

```
modFit_lda <- train(classe~., method = "lda", data = important_features, prox
= TRUE)
pred_lda <- predict(modFit_lda, newdata = validation_features)
Matrix_lda <- confusionMatrix(pred_lda, validation$classe)
```

3) random forest

```r
modFit_rf <- train(classe~., method = "rf", data = important_features)
pred_rf <- predict(modFit_rf, newdata = validation_features)
Matrix_rf <- confusionMatrix(pred_rf, validation$classe)
```

Now we compare the prediction accuracy of out of sample errors among the 3 different models below.

It is clear that the random forest model has the lowest out of smaple errors and is selected as the

optimal model to predict the test dataset

1) Recursive partitioning

```r
Matrix_Rpart
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1517  476  473  413  156
##          B   29  395   36  184  150
##          C  124  268  517  367  280
##          D    0    0    0    0    0
##          E    4    0    0    0  496
##
## Overall Statistics
##
##                Accuracy : 0.497
##                  95% CI : (0.4842, 0.5099)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3429
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9062  0.34680  0.50390   0.0000  0.45841
## Specificity            0.6395  0.91593  0.78617   1.0000  0.99917
## Pos Pred Value         0.4998  0.49748  0.33226      NaN  0.99200
## Neg Pred Value         0.9449  0.85386  0.88242   0.8362  0.89118
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2578  0.06712  0.08785   0.0000  0.08428
## Detection Prevalence   0.5157  0.13492  0.26440   0.0000  0.08496
## Balanced Accuracy      0.7729  0.63136  0.64503   0.5000  0.72879
```

2) Linear discriminant analysis

```r
Matrix_lda
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1213  296  283   94  145
##          B   81  448   74   43  128
##          C   97  217  552   74  151
##          D  250  148   98  689  131
##          E   33   30   19   64  527
##
## Overall Statistics
##
##                Accuracy : 0.5827
##                  95% CI : (0.5699, 0.5953)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4693
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7246  0.39333   0.5380   0.7147  0.48706
## Specificity            0.8057  0.93131   0.8891   0.8726  0.96960
## Pos Pred Value         0.5972  0.57881   0.5060   0.5236  0.78306
## Neg Pred Value         0.8804  0.86480   0.9011   0.9398  0.89351
## Prevalence             0.2845  0.19354   0.1743   0.1638  0.18386
## Detection Rate         0.2061  0.07613   0.0938   0.1171  0.08955
## Detection Prevalence   0.3451  0.13152   0.1854   0.2236  0.11436
## Balanced Accuracy      0.7652  0.66232   0.7135   0.7937  0.72833
```

3) random forest

```
Matrix_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1669   14    0    0    0
##          B    5 1120    9    1    2
##          C    0    5 1016    9    0
##          D    0    0    1  954    3
##          E    0    0    0    0 1077
##
## Overall Statistics
##
##                Accuracy : 0.9917
##                  95% CI : (0.989, 0.9938)
##     No Information Rate : 0.2845
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9895
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9833   0.9903   0.9896   0.9954
## Specificity            0.9967   0.9964   0.9971   0.9992   1.0000
## Pos Pred Value         0.9917   0.9850   0.9864   0.9958   1.0000
## Neg Pred Value         0.9988   0.9960   0.9979   0.9980   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2836   0.1903   0.1726   0.1621   0.1830
## Detection Prevalence   0.2860   0.1932   0.1750   0.1628   0.1830
## Balanced Accuracy      0.9968   0.9899   0.9937   0.9944   0.9977
```

Prediction outcome using the test dataset

```
predict(modFit_rf, newdata = Test_Data)
```

```
##  [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
```

Conclusion

After comparing four different types of models, our result shows random forest provides the most accurate outcome and reached 100% prediction rate in the test dataset.

Among 153 features collected, we ranked and chose the most important 18 features and fed them the training models. By doing this, I believe the out of sample error is reduced, as well as the computational intensity.

However, the random forest require a lot more time for training than the others and won't be ideal for a quick prediction. Our results are limited to four models due to the limited time and future work should focus on other models to test both accuracy and efficiency.