Predicting Motions

Dongying

9/13/2020

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.
In this project, our 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: 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) and throwing the hips to the front (Class E).

Getting and cleaning the data

We first download the data from provided URL and load them into R.

```
## Download the CSV file and then unzip it. Check if the files exist before processing.
trainingURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
if (!file.exists("training.csv")){
        download.file(trainingURL, "training.csv", method = "curl")
}
testingURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
if (!file.exists("testing.csv")){
        download.file(testingURL, "testing.csv", method = "curl")
}

## Read data into R
training <- read.csv("training.csv", na.strings=c("NA","#DIV/0!", ""))
testing <- read.csv("testing.csv", na.strings=c("NA","#DIV/0!", ""))
```

We can take a look at the data and cleaning them by exclude the NA columns and irrelevant columns such as usernames and so on.

```
library(caret)
```

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

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

```
## Warning: replacing previous import 'vctrs::data_frame' by 'tibble::data_frame'
## when loading 'dplyr'
```

```
training <- training[,colSums(is.na(training)) == 0]
testing <- testing[,colSums(is.na(testing)) == 0]
training   <-training[,-c(1:7)]
testing <-testing[,-c(1:7)]
training$classe <- as.factor(training$classe)
```

Prediction by using random forest

Create partition using the training data so we can have 30% data for validation:

```
set.seed(123)
inValid <- createDataPartition(training$classe, p = 0.3, list = FALSE)
validation <- training[inValid,]
training_train <- training[-inValid,]
dim(validation)
```

```
## [1] 5889   53
```

```
dim(training_train)
```

```
## [1] 13733    53
```

Train the actual training data using "classe" as outcome using random forest method. Use the fitted model to predict for the validation group and than compare the results.

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
modFit_rf <- randomForest(classe ~., data = training_train, method = "class")
pred_rf <- predict(modFit_rf, newdata = validation)
confusionMatrix(pred_rf, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    8    0    0    0
##          B    0 1130    7    0    0
##          C    0    2 1020    8    0
##          D    0    0    0  956    1
##          E    0    0    0    1 1082
##
## Overall Statistics
##
##                Accuracy : 0.9954
##                  95% CI : (0.9933, 0.997)
##     No Information Rate : 0.2843
```

```
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9942
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9912   0.9932   0.9907   0.9991
## Specificity            0.9981   0.9985   0.9979   0.9998   0.9998
## Pos Pred Value         0.9952   0.9938   0.9903   0.9990   0.9991
## Neg Pred Value         1.0000   0.9979   0.9986   0.9982   0.9998
## Prevalence             0.2843   0.1936   0.1744   0.1639   0.1839
## Detection Rate         0.2843   0.1919   0.1732   0.1623   0.1837
## Detection Prevalence   0.2856   0.1931   0.1749   0.1625   0.1839
## Balanced Accuracy      0.9991   0.9949   0.9956   0.9952   0.9994
```

The accuracy is as high as 0.9954. We can use this model to predict the testing data.

Predicting Motions in Testing data

Here we predict the motions in the testing group.

```
pred_testing <- predict(modFit_rf, newdata = testing)
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
```

Apendix: Training data with "rpart" (less accurate)

Here is and example of other model fitting with "rpart" method, which gives us lower accuracy by comparing the prediction results with validation group. We choose random forest from all the methods.

```
library(rpart)
library(rattle)
```

```
## Loading required package: tibble
```

```
## Loading required package: bitops
```

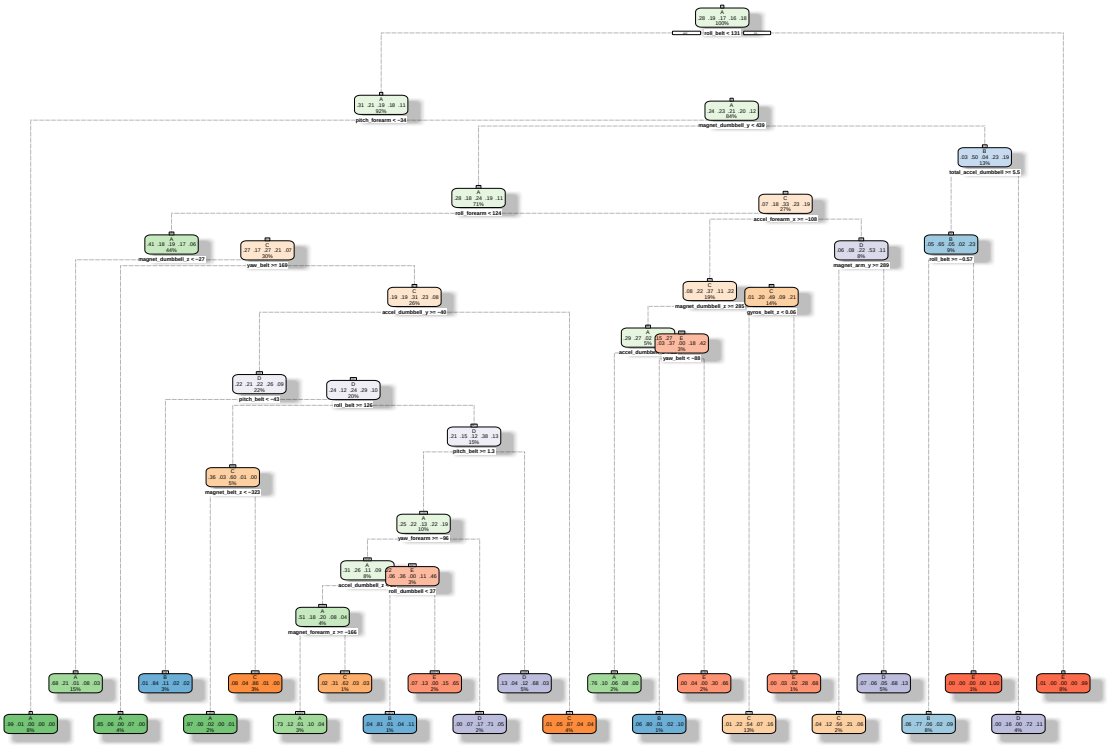
```
## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

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

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

```
modFit_rpart <- rpart(classe ~., data = training_train, method = "class")
fancyRpartPlot(modFit_rpart)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2020−Sep−13 18:40:08 dongyingwang

```
pred_rpart <- predict(modFit_rpart, newdata = validation, type = "class")
confusionMatrix(pred_rpart, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1528  258   21  107   40
##          B   36  595   53   25   51
##          C   42  197  876   84  135
##          D   56   67   76  671   74
##          E   12   23    1   78  783
##
## Overall Statistics
##
##                Accuracy : 0.7562
##                  95% CI : (0.745, 0.7671)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6901
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
```

```
## Sensitivity            0.9128   0.5219   0.8530   0.6953   0.7230
## Specificity            0.8989   0.9653   0.9058   0.9446   0.9763
## Pos Pred Value         0.7820   0.7829   0.6567   0.7108   0.8729
## Neg Pred Value         0.9629   0.8937   0.9668   0.9405   0.9399
## Prevalence             0.2843   0.1936   0.1744   0.1639   0.1839
## Detection Rate         0.2595   0.1010   0.1488   0.1139   0.1330
## Detection Prevalence   0.3318   0.1291   0.2265   0.1603   0.1523
## Balanced Accuracy      0.9059   0.7436   0.8794   0.8199   0.8496
```