Project Machine Learning

Auriane

4 augustus 2017

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

Data

The training data for this project are available here:

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

The test data are available here:

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

Libraries

In order to reproduce my work, you need to load the following libraries.

```
library(caret)
```

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

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

```
library(rpart)
library(rpart.plot)
library(RColorBrewer)
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(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
```

Load the data

```
trainURL <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
validURL <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

training <- read.csv(url(trainURL), na.strings=c("NA","#DIV/0!",""))
validationset  <- read.csv(url(validURL), na.strings=c("NA","#DIV/0!",""))
```

Cleaning

Remove columns containing values equal to zero and the first seven columns.

```
training <- training[, colSums(is.na(training)) == 0]
validationset <- validationset[, colSums(is.na(validationset)) == 0]

trainData <- training[, -c(1:7)]
ValidationData <- validationset[, -c(1:7)]
```

Create a training and a test set

Partition the data into a training set and testing set. We use 70% for training set and 30% for testing set.

```
set.seed(6789)
inTrain <- createDataPartition(trainData$classe, p = 0.7, list = FALSE)
train <- trainData[inTrain, ]
testingset<- trainData[-inTrain, ]

dim(train)
```

```
## [1] 13737    53
```

```
dim(testingset)
```

```
## [1] 5885   53
```

Prediction Algorithms

In this project we will make use of three predicting models: a classification tree and a random forest. We do not need to make data transformations as it is less important in non-linear models, we do not transform any variables. We set the number of resampling iterations to 5.

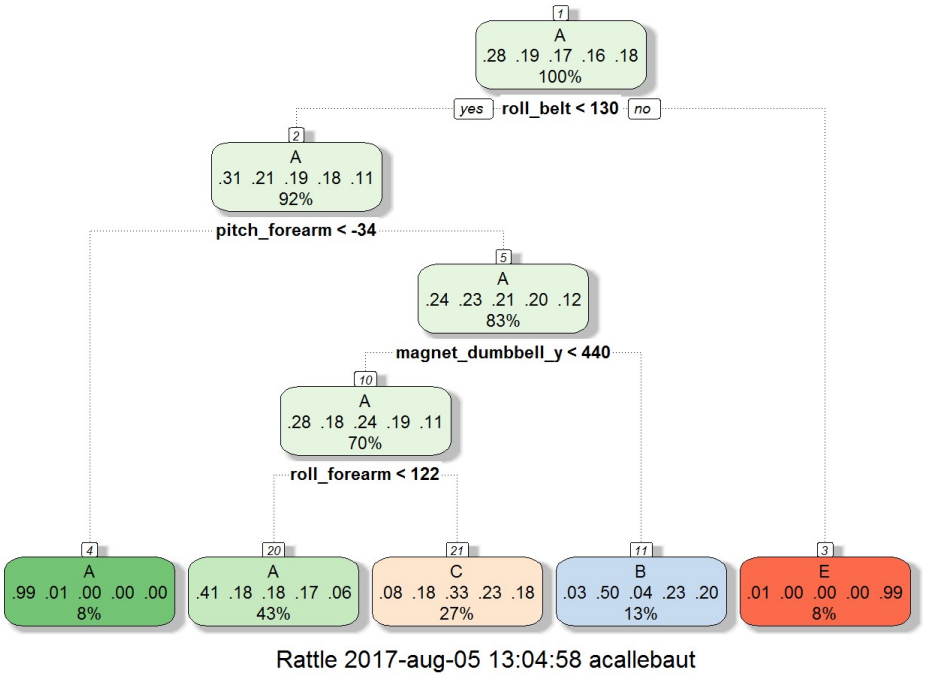
1. Classification Trees

The idea here is to predict with trees, we iteratively split the variables into groups, evaluate the homogeneity within each group and slit again. It's easy to interpret and nice and pretty plots can be made with the function *fancyRpartPlot* from the *rattle* package. We put the number of resampling iterations to 5.

```
mod_CT <- train(classe ~ ., data = train, method = "rpart", trControl = trainControl(method = "cv", number
= 5))
print(mod_CT, digits = 4)
```

```
## CART
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10989, 10989, 10990, 10991
## Resampling results across tuning parameters:
##
##   cp       Accuracy  Kappa
##   0.03286  0.5130    0.36459
##   0.05947  0.4390    0.24790
##   0.11474  0.3159    0.04813
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03286.
```

```
fancyRpartPlot(mod_CT$finalModel)
```




```
predict_CT <- predict(mod_CT, testingset)
ConfusionMatrix_CT <- confusionMatrix(testingset$classe, predict_CT)
```

We apply the model on the testing set and check the accuracy of the prediction.

```
accuracy_CT <- ConfusionMatrix_CT$overall[1]
accuracy_CT
```

```
##  Accuracy
## 0.4987256
```

The accuracy is not very high meaning we can try another model.

2. Random Forest

Random forest are also a kind of classification method but here we bootstrap the variables at each split. We grow multiple trees and vote. It's known to have a high accuracy. We expect thus more predicting power with random forests than with a classical classification tree. We put the number of resampling iterations to 5.

```
mod_RF <- train(classe ~ ., data = train, method = "rf", trControl = trainControl(method = "cv", number =
5))
print(mod_RF, digits = 4)
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10990, 10991, 10989, 10989
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##    2    0.9913    0.9890
##   27    0.9916    0.9894
##   52    0.9859    0.9821
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
predict_RF <- predict(mod_RF, testingset)
ConfusionMatrix_RF <- confusionMatrix(testingset$classe, predict_RF)
```

We apply the model on the testing set and check the accuracy of the prediction.

```
accuracy_RF <- ConfusionMatrix_RF$overall[1]
accuracy_RF
```

```
##  Accuracy
## 0.9925234
```

The accuracy here is higher. We select this model to predict the *classe* variable.

Predictions on the validation data set

Now that we have selected the random forest model, we can apply it on the validation data and check what are the values that the model predicts. Find below the 20 cases and their classes.

```
predict(mod_RF, ValidationData)
```

```
##  [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
```

Thank you !