Machine Learning Project

yohan

30 de agosto de 2019

Summary

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.

Objective:

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. Predict the outcome of the variable classe in the training set.

Set of libraries

```
options(tinytex.verbose = TRUE)
library(caret)
## Loading required package: lattice
## Loading required package: ggplot2
library(rpart)
library(rpart.plot)
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
library(rattle)
## Rattle: A free graphical interface for data science with R.
```

```
## VersiÃ³n 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Escriba 'rattle()' para agitar, sacudir y  rotar sus datos.
##
## Attaching package: 'rattle'
## The following object is masked from 'package:randomForest':
##
##     importance
```

```
library(knitr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
## The following object is masked from 'package:randomForest':
##
##     combine
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Load Data

```
options(tinytex.verbose = TRUE)


Train <- read.csv(file='pml-training.csv', header = TRUE, sep = ',',na.st
rings=c( "", "#DIV/0!","NA"))
Test <- read.csv(file='pml-testing.csv', header = TRUE, sep = ',' ,na.str
ings=c( "", "#DIV/0!","NA"))
```

dimention of sets of Data

```
options(tinytex.verbose = TRUE)


dim(Train)
```

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

```
dim(Test)
```

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

Dimentions of trainning data are Rows=19622 columns=160 esting data = Rows = 20 columns=160

Clean Data

If we make a summary rain and Test we can see a lot NA data so we have to clean Changin NA BY 0

```
options(tinytex.verbose = TRUE)


Train[is.na(Train)] <- 0
Test[is.na(Test)] <- 0
```

Dataset with 5 classes (sitting-down, standing-up, standing, walking, and sitting) collected on 8 hours of activities of 4 healthy subjects

```
options(tinytex.verbose = TRUE)


summary(Train$classe)
```

```
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

Lets selects variables that are useful

```
options(tinytex.verbose = TRUE)


Train <- select(Train,(8:160))
Test <- select(Test,(8:160))
```

Lests takeout de columns with “total” since might be dirty data for the porpouse.

```
options(tinytex.verbose = TRUE)


CleanTrain <-  select (Train, -"total_accel_belt", -"total_accel_arm", -"
total_accel_dumbbell", -"total_accel_forearm")
```

```
CleanTest <- select (Test, -"total_accel_belt", -"total_accel_arm", -"tot
al_accel_dumbbell", -"total_accel_forearm")
```

Partitioning of Data

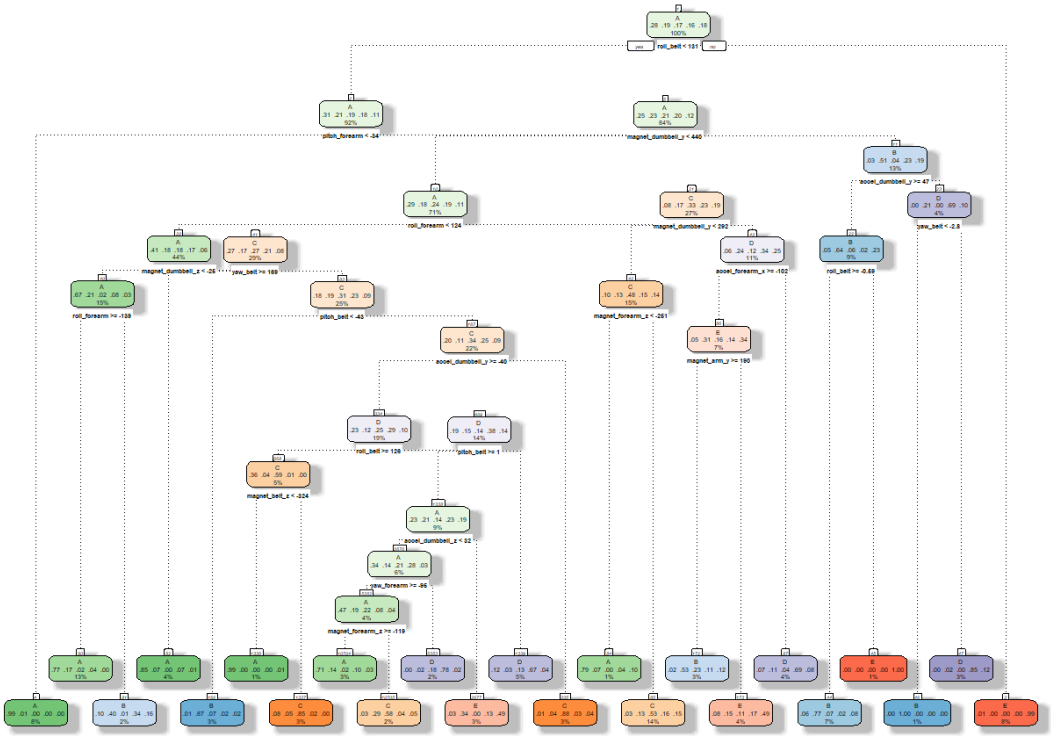
```
options(tinytex.verbose = TRUE)


Traini<-createDataPartition(y=CleanTrain$classe, p=0.75, list=FALSE)
Trainsub<-CleanTrain[Traini,]
Testsub<-CleanTrain[-Traini,]
```

dimention of sets of subData

```
options(tinytex.verbose = TRUE)


dim(Trainsub)
## [1] 14718   149
dim(Testsub)
## [1] 4904  149
```

Decision Tree

```
options(tinytex.verbose = TRUE)


set.seed(2019)
Treefit <- rpart(classe ~ ., data=Trainsub, method="class")
fancyRpartPlot(Treefit)
```



Rattle 2019-sept.-04 19:48:33 Yohan

###Lets predict with decision tree

```
options(tinytex.verbose = TRUE)


set.seed(2019)
Treefitest <- predict(Treefit, Testsub, type = "class")
confusionMatrix(Treefitest,Testsub$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1244  158   22   45   17
##          B   34  509   63   64   70
##          C   47  147  703  123  125
##          D   52   45   44  506   51
##          E   18   90   23   66  638
##
## Overall Statistics
```

```
##
##                Accuracy : 0.7341
##                  95% CI : (0.7215, 0.7464)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6631
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8918   0.5364   0.8222   0.6294   0.7081
## Specificity            0.9310   0.9416   0.8908   0.9532   0.9508
## Pos Pred Value         0.8371   0.6878   0.6140   0.7249   0.7641
## Neg Pred Value         0.9558   0.8943   0.9596   0.9291   0.9354
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2537   0.1038   0.1434   0.1032   0.1301
## Detection Prevalence   0.3030   0.1509   0.2335   0.1423   0.1703
## Balanced Accuracy      0.9114   0.7390   0.8565   0.7913   0.8294
```

This method result in a Accuracy : 0.7612 and a error rate: 0.53%, so lets try with another one.

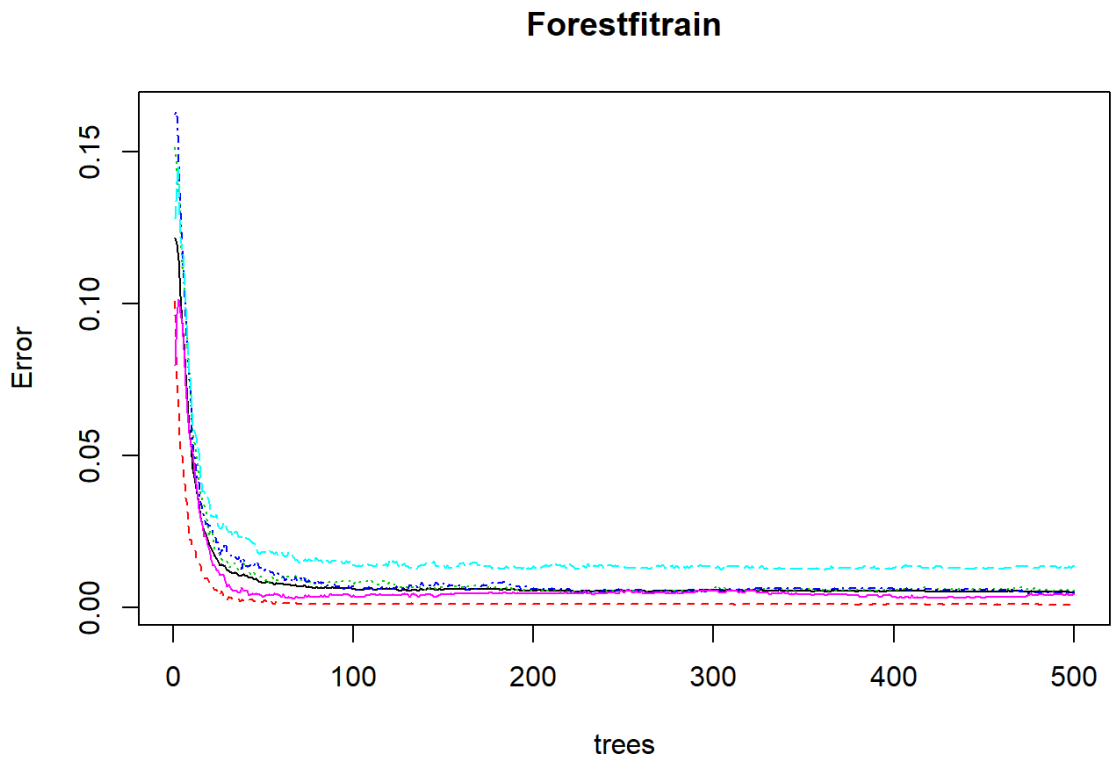
Ramdon forest Train

```
options(tinytex.verbose = TRUE)


Forestfitrain <- randomForest(classe ~ ., data=Trainsub)
Forestfitrain
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = Trainsub)
##                Type of random forest: classification
##                      Number of trees: 500
```

```
## No. of variables tried at each split: 12
##
##         OOB estimate of  error rate: 0.52%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4181    4    0    0    0 0.0009557945
## B   12 2832    4    0    0 0.0056179775
## C    0   11 2555    1    0 0.0046747176
## D    0    0   31 2379    2 0.0136815920
## E    0    0    5    7 2694 0.0044345898
```

```
options(tinytex.verbose = TRUE)


plot(Forestfitrain)
```



In this graphic we cn see the error, is not precisly under 0,05 as I hoped but the accuracy is over 99,5%

```
options(tinytex.verbose = TRUE)
```

```
predforest <- predict(Forestfitrain, Testsub, type = "class")
Pred <- confusionMatrix(predforest, Testsub$classe)
Pred
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    8    0    0    0
##          B    0  940    3    0    0
##          C    0    1  852    7    0
##          D    0    0    0  797    1
##          E    0    0    0    0  900
##
## Overall Statistics
##
##                Accuracy : 0.9959
##                  95% CI : (0.9937, 0.9975)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9948
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9905   0.9965   0.9913   0.9989
## Specificity            0.9977   0.9992   0.9980   0.9998   1.0000
## Pos Pred Value         0.9943   0.9968   0.9907   0.9987   1.0000
## Neg Pred Value         1.0000   0.9977   0.9993   0.9983   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1917   0.1737   0.1625   0.1835
## Detection Prevalence   0.2861   0.1923   0.1754   0.1627   0.1835
```

```
## Balanced Accuracy      0.9989   0.9949   0.9973   0.9955   0.9994
```

This method is Accuracy : 0.9955 although error is above 0,05 random forest is a better model than Clasiification Trees model

Let make a prediction with the data delivered at the begining of the exercise

```
options(tinytex.verbose = TRUE)


finalPred <- predict(Forestfitrain, CleanTest, type = "class")
finalPred
```

```
##  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
```

it has been successfully predicted the outcome of the test data using random forest model used on inicial training data with 99.55% of accuracy