practical machine learning : Assignment

daniel

13 April 2018

Executive 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. 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.

The goal of this project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set. You may use any of the

initialisation

Load the libary that will be used

```
library(ggplot2)
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

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

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.4.3
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.1.0 Copyright (c) 2006-2017 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

set the seed

```
set.seed(42)
```

Load the data

```
dTraining <- read.csv("pml-training.csv")
dTesting <- read.csv("pml-testing.csv")
```

analyse the data

```
#creation of a function that will clean the training & test dataset
fPrepareData <- function(dataset){
    #remove row with more than 90% na
    dim(dataset)
    naRow <- sapply(names(dataset), function(x){sum(is.na(dataset[x]))/nrow(dataset) })
    datasetc1 <- dataset[,naRow <.9]
    dim(datasetc1)


    #remove near zera variable
    nzvId <- nearZeroVar(datasetc1)
    datasetc2 <- datasetc1[,-nzvId]
    dim(datasetc2)

    #remove First Column
    datasetc3 <- datasetc2[,-1]
    dim(datasetc3)
    return(datasetc3)
}

dTrainingClean <- fPrepareData(dTraining)
dTestingClean <- fPrepareData(dTesting)
```

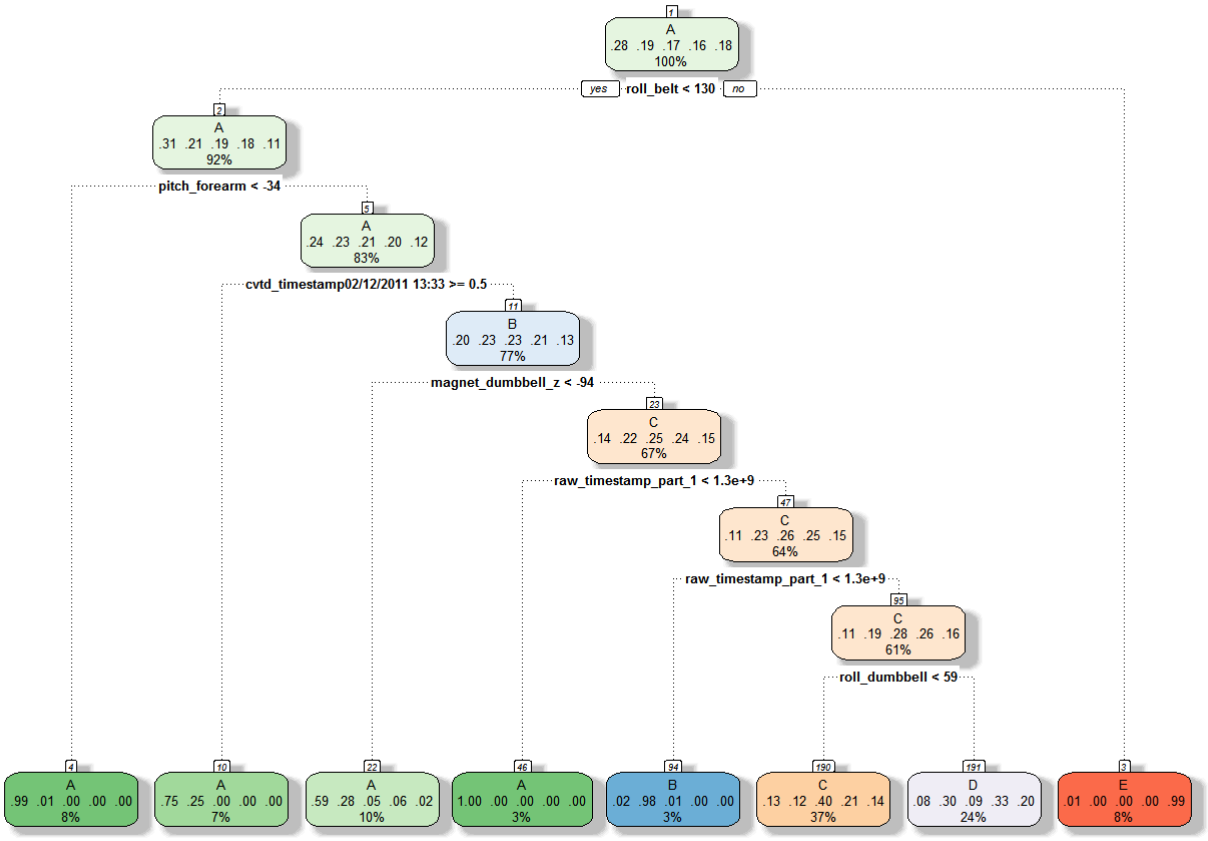
split data into trainnig and testing

```
dPartition <- createDataPartition(dTrainingClean$classe, p=.6, list = FALSE)
dTrainingCleanTr <- dTrainingClean[dPartition, ]
dTrainingCleanTe <- dTrainingClean[-dPartition, ]
rm(dPartition)
```

build model

model 1: decision tree

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



Rattle 2018-Apr-13 21:03:22 bruled

model 2: random Forest

```
modelRF <- train(classe ~ ., data=dTrainingCleanTr, method="rf", do.trace=FALSE)
```

Evaluate Model

```
pDT <- predict(modelDT, dTrainingCleanTe)
pRF <- predict(modelRF, dTrainingCleanTe)

confusionMatrix(pDT, dTrainingCleanTe$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1661  355   33   37   23
##          B    4  242    0    0    0
##          C  387  356 1181  574  406
##          D  174  565  154  675  359
##          E    6    0    0    0  654
##
## Overall Statistics
##
##                Accuracy : 0.5625
##                  95% CI : (0.5514, 0.5735)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4512
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7442  0.15942   0.8633  0.52488  0.45354
## Specificity            0.9202  0.99937   0.7340  0.80915  0.99906
## Pos Pred Value         0.7876  0.98374   0.4067  0.35029  0.99091
## Neg Pred Value         0.9005  0.83211   0.9622  0.89677  0.89034
## Prevalence             0.2845  0.19347   0.1744  0.16391  0.18379
## Detection Rate         0.2117  0.03084   0.1505  0.08603  0.08335
## Detection Prevalence   0.2688  0.03135   0.3701  0.24560  0.08412
## Balanced Accuracy      0.8322  0.57939   0.7987  0.66701  0.72630
```

```
confusionMatrix(pRF, dTrainingCleanTe$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    2    0    0    0
##          B    0 1516    0    0    0
##          C    0    0 1368    6    0
##          D    0    0    0 1279    0
##          E    0    0    0    1 1442
##
## Overall Statistics
##
##                Accuracy : 0.9989
##                  95% CI : (0.9978, 0.9995)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9985
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9987   1.0000   0.9946   1.0000
## Specificity            0.9996   1.0000   0.9991   1.0000   0.9998
## Pos Pred Value         0.9991   1.0000   0.9956   1.0000   0.9993
## Neg Pred Value         1.0000   0.9997   1.0000   0.9989   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1932   0.1744   0.1630   0.1838
## Detection Prevalence   0.2847   0.1932   0.1751   0.1630   0.1839
## Balanced Accuracy      0.9998   0.9993   0.9995   0.9973   0.9999
```

Prediction

As random forest is better we will use it for the prediction

```
predict(modelRF, dTestingClean)
```

```
##  [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
```