Practical Machine learning: Peer graded assignment

Santiago Cárdenas

1/8/2021

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, 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 (see the section on the Weight Lifting Exercise Dataset).

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz3xsbS5bVX

##Loading packages and data

```
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(rpart)
library(ggplot2)
library(corrplot)

## corrplot 0.84 loaded

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)

## 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

library(e1071)
library(tinytex)


training_raw <- read.csv("pml-training.csv")[,-1]
testing <- read.csv("pml-testing.csv")[,-1]
```

##Creating traning and testing sets

```
NZV <- nearZeroVar(training_raw)
training <- training_raw[, -NZV]
testing <- testing[, -NZV]

NaValues <- sapply(training, function(x) mean(is.na(x))) > 0.9
training <- training[, NaValues == "FALSE"]
testing <- testing[, NaValues == "FALSE"]

training <- training[,-c(1:5)]
testing <- testing[,-c(1:5)]
```

Models

The model chosen for this data analysis are Random Forest and decision trees. I will apply the method to the training sets and then I will apply the better one to the testing set.

#Random Forest

```
set.seed(123)
cv3 <- trainControl(method="cv",number=3,allowParallel=TRUE,verboseIter=TRUE)
Randomforest<-train(classe~., data=training, method="rf",trControl=cv3)

## + Fold1: mtry= 2
## - Fold1: mtry= 2
## + Fold1: mtry=27
## - Fold1: mtry=27
```

```
## + Fold1: mtry=52
## - Fold1: mtry=52
## + Fold2: mtry= 2
## - Fold2: mtry= 2
## + Fold2: mtry=27
## - Fold2: mtry=27
## + Fold2: mtry=52
## - Fold2: mtry=52
## + Fold3: mtry= 2
## - Fold3: mtry= 2
## + Fold3: mtry=27
## - Fold3: mtry=27
## + Fold3: mtry=52
## - Fold3: mtry=52
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 2 on full training set
```

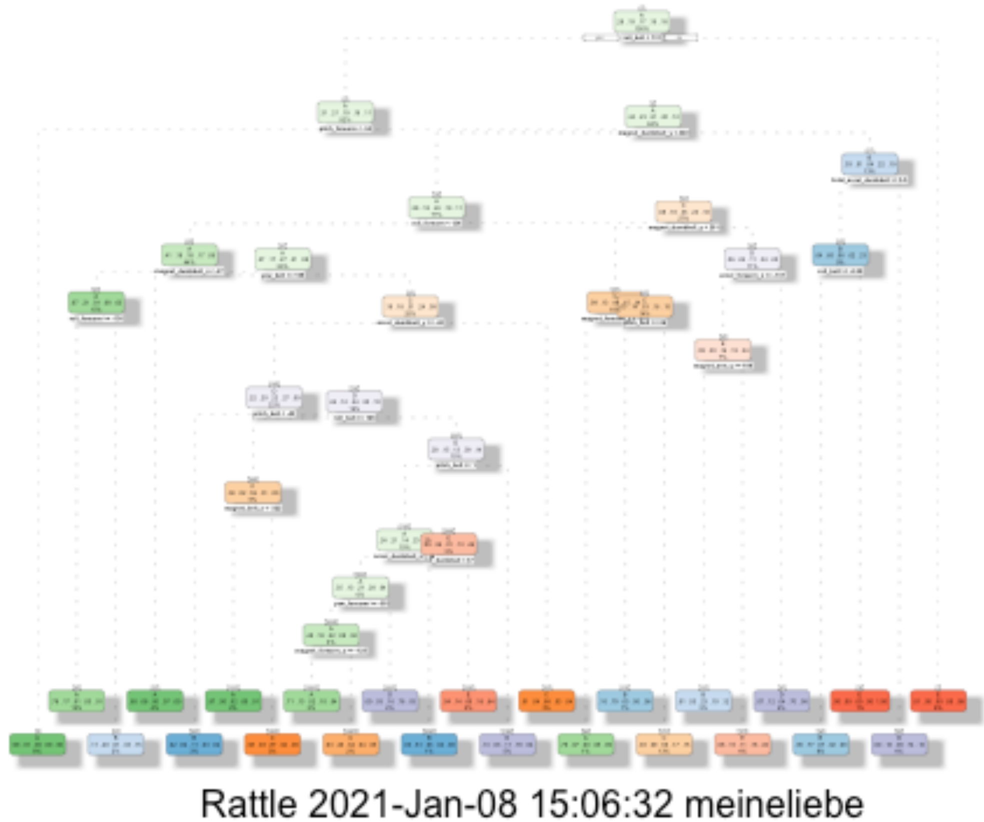
#Decision tree

```
DecisionTree<- train(classe~.,data=training,method="rpart",trControl=cv3)
```

```
## + Fold1: cp=0.03568
## - Fold1: cp=0.03568
## + Fold2: cp=0.03568
## - Fold2: cp=0.03568
## + Fold3: cp=0.03568
## - Fold3: cp=0.03568
## Aggregating results
## Selecting tuning parameters
## Fitting cp = 0.0357 on full training set
```

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

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



#Comparison

I will now use tables two compare the behavior of the two models

```r
randomF<-predict(Randomforest,training)
DecTree<-predict(DecisionTree,training)
table(randomF,training$classe)
```

```
##
## randomF    A    B    C    D    E
##       A 5580    0    0    0    0
##       B    0 3797    0    0    0
##       C    0    0 3422    0    0
##       D    0    0    0 3216    0
##       E    0    0    0    0 3607
```

```r
table(DecTree,training$classe)
```

```
##
## DecTree    A    B    C    D    E
##       A 5080 1581 1587 1449  524
##       B   81 1286  108  568  486
##       C  405  930 1727 1199  966
```

```
##          D    0    0    0    0    0
##          E   14    0    0    0 1631
```

##Conclusion

Since the Random Forest method shows a higher level of accuracy compared to the decision tree I will use it to do the final prediction.

```
finalprediction<- predict(Randomforest, testing)
finalprediction

##  [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
```