

# Prediction Assignment

echf

8/9/2021

```
knitr::opts_chunk$set(echo = TRUE)
###knitr::opts_chunk$set(out.width="400px", dpi=120)
options(width=80)
```

## Synopsis

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 dumbbell 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://web.archive.org/web/20161224072740/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>

The test data are available here: <https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv>

The data for this project come from this source: <http://web.archive.org/web/20161224072740/http://groupware.les.inf.puc-rio.br/har>.

## LIBRARY'S

```
options(warn=-1)
library(caret)
```

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

```

## Loading required package: ggplot2

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(Hmisc)

## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

## Loading required package: Formula

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units

library(foreach)
library(doParallel)

## Loading required package: iterators

## Loading required package: parallel

library(rattle)

## Loading required package: tibble

## Loading required package: bitops

```

```
## Rattle: A free graphical interface for data science with R.  
## Versión 5.4.0 Copyright (c) 2006-2020 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(rpart)  
library(rpart.plot)  
library(RColorBrewer)  
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
library(plyr)
```

```
##  
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:Hmisc':  
##  
##      is.discrete, summarize
```

```
set.seed(21243)
```

## DATA LOAD

```
train_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"  
test_url  <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
```

```
training_data <- read.csv(url(train_url), na.strings = c("NA", "#DIV/0!", ""))  
testing_data  <- read.csv(url(test_url),  na.strings = c("NA", "#DIV/0!", ""))
```

```
dim(training_data)
```

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

```
dim(testing_data)
```

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

## Data Cleaning train\_url

```

# Removing Variables which are having nearly zero variance

cData <- training_data
for(i in c(8:ncol(cData)-1)) {cData[,i] = as.numeric(as.character(cData[,i]))}

# First look at the data for each column and remove variables unrelated
# to exercise (column number and time stamps)
featuresnames <- colnames(cData[colSums(is.na(cData)) == 0])[-(1:7)]
features <- cData[featuresnames]

```

## Data Partitioning

```

#### in this course it is recommended to make a partition of 60%-40%
# but in other documents it is 70%-30%. so what was seen in this
#course will be used.

inTrain <- createDataPartition(features$classe, p=0.60, list=FALSE)
training <- features[inTrain,]
testing <- features[-inTrain,]

dim(training)

```

```
## [1] 11776    53
```

```
dim(testing)
```

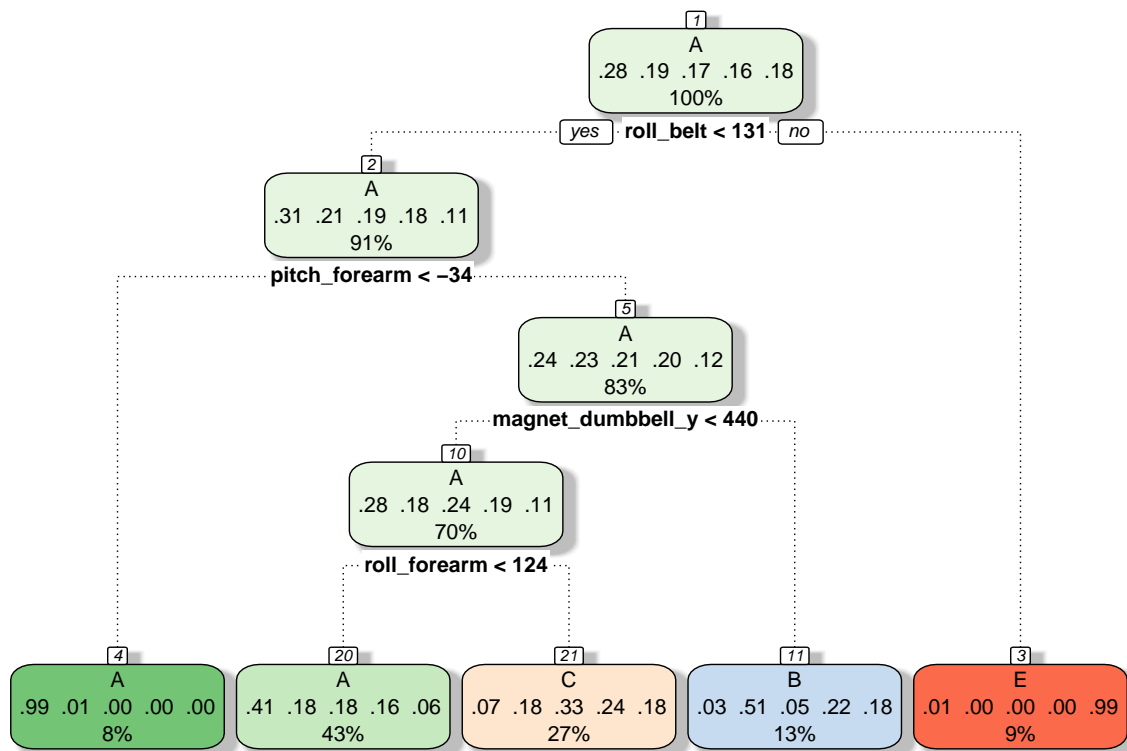
```
## [1] 7846    53
```

## Decision Tree Model and Prediction

```

DT_model<- train(classe ~. , data=training, method= "rpart")
fancyRpartPlot(DT_model$finalModel)

```



Rattle 2021-sep.-10 12:59:05 ED

```
DT_prediction<- predict(DT_model, newdata=testing)
identical(DT_prediction , testing$classe)
```

```
## [1] FALSE
```

```
confusionMatrix(table(DT_prediction, testing$classe)) ##
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##
```

```
## DT_prediction    A     B     C     D     E
```

```
##           A 2029  659  643  620  207
```

```
##           B   28  495   35  226  206
```

```
##           C  171  364  690  440  399
```

```
##           D    0    0    0    0    0
```

```
##           E    4    0    0    0  630
```

```
##
```

```
## Overall Statistics
```

```
##
```

```
##           Accuracy : 0.4899
```

```
##           95% CI : (0.4788, 0.5011)
```

```
##           No Information Rate : 0.2845
```

```
##           P-Value [Acc > NIR] : < 2.2e-16
```

```
##
```

```
##                      Kappa : 0.3325
##
## McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##          Class: A Class: B Class: C Class: D Class: E
## Sensitivity      0.9091  0.32609  0.50439  0.0000  0.43689
## Specificity      0.6208  0.92178  0.78790  1.0000  0.99938
## Pos Pred Value   0.4880  0.50000  0.33430    NaN  0.99369
## Neg Pred Value   0.9450  0.85079  0.88274  0.8361  0.88741
## Prevalence       0.2845  0.19347  0.17436  0.1639  0.18379
## Detection Rate   0.2586  0.06309  0.08794  0.0000  0.08030
## Detection Prevalence 0.5300  0.12618  0.26306  0.0000  0.08081
## Balanced Accuracy 0.7649  0.62393  0.64614  0.5000  0.71813
```

## GBM model and Prediction

```
gbm_model<- train(classe ~. , data=training, method= "gbm",
                  trControl= trainControl(method="repeatedcv",
                                          number = 5, repeats = 1),
                  verbose=FALSE)

gbm_prediction<- predict(gbm_model, newdata=testing)
identical(gbm_prediction , testing$classe)
```

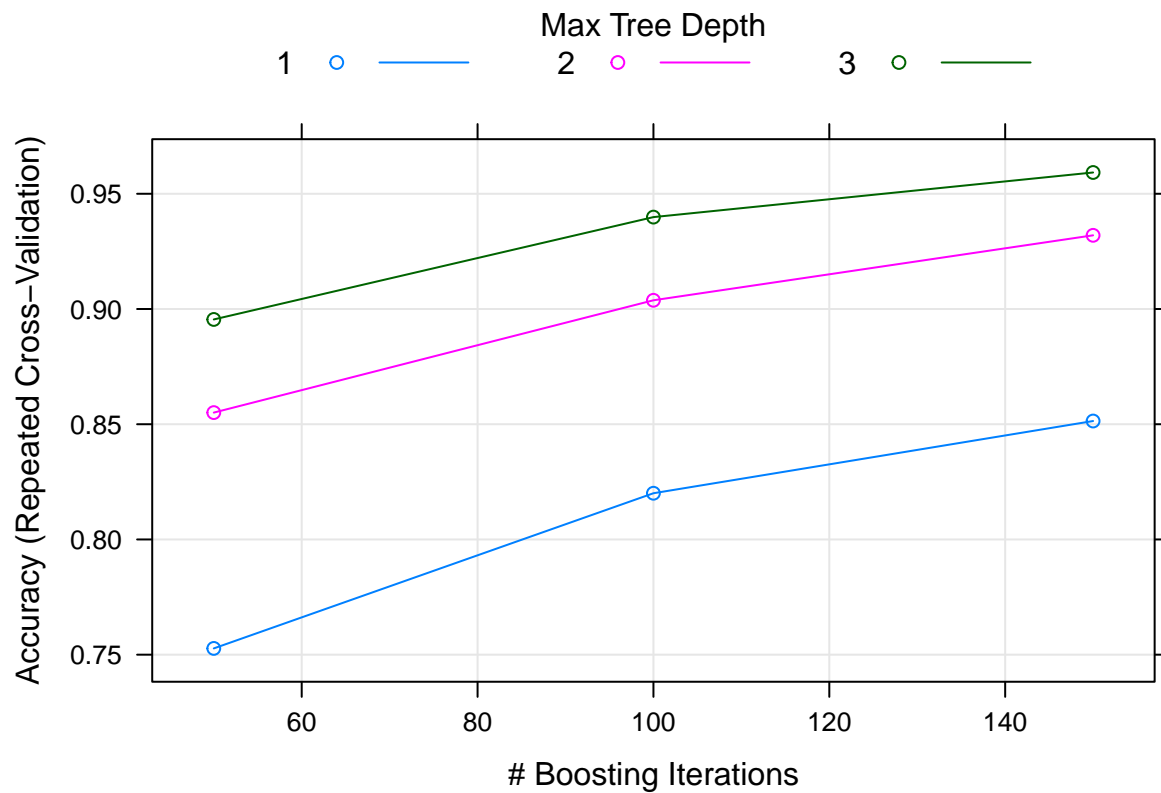
```
## [1] FALSE
```

```
confusionMatrix(table(gbm_prediction, testing$classe)) ##
```

```
## Confusion Matrix and Statistics
##
##
## gbm_prediction      A      B      C      D      E
##      A 2199    56     0     0     4
##      B   21 1407    35    13    12
##      C    7   51 1311    47    12
##      D    3    4   21 1216    20
##      E    2    0    1   10 1394
##
## Overall Statistics
##
##              Accuracy : 0.9593
##              95% CI : (0.9547, 0.9636)
##      No Information Rate : 0.2845
##      P-Value [Acc > NIR] : < 2.2e-16
##
##              Kappa : 0.9486
##
## McNemar's Test P-Value : 7.229e-11
```

```
##
## Statistics by Class:
##
##           Class: A Class: B Class: C Class: D Class: E
## Sensitivity      0.9852  0.9269  0.9583  0.9456  0.9667
## Specificity      0.9893  0.9872  0.9819  0.9927  0.9980
## Pos Pred Value   0.9734  0.9456  0.9181  0.9620  0.9908
## Neg Pred Value    0.9941  0.9825  0.9911  0.9894  0.9925
## Prevalence       0.2845  0.1935  0.1744  0.1639  0.1838
## Detection Rate   0.2803  0.1793  0.1671  0.1550  0.1777
## Detection Prevalence 0.2879  0.1897  0.1820  0.1611  0.1793
## Balanced Accuracy 0.9873  0.9570  0.9701  0.9691  0.9823
```

```
plot(gbm_model)
```



## Random Forest Model and Prediction

```
RForest <- train(classe~., data=training, method="rf",
                 trControl=trainControl(method="cv", classProbs=TRUE,
                                       savePredictions=TRUE,
                                       allowParallel=TRUE, number = 10))

predRF<-predict(RForest, testing)
```

```
CMatrixRF <- confusionMatrix(table(predRF, testing$classe))
CMatrixRF
```

```
## Confusion Matrix and Statistics
##
##
## predRF      A      B      C      D      E
##      A 2232    20      0      0      0
##      B      0 1494    13      0      1
##      C      0      4 1349    30      2
##      D      0      0      6 1255      5
##      E      0      0      0      1 1434
##
## Overall Statistics
##
##              Accuracy : 0.9895
##              95% CI : (0.987, 0.9917)
##      No Information Rate : 0.2845
##      P-Value [Acc > NIR] : < 2.2e-16
##
##              Kappa : 0.9868
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##              Class: A Class: B Class: C Class: D Class: E
## Sensitivity          1.0000   0.9842   0.9861   0.9759   0.9945
## Specificity          0.9964   0.9978   0.9944   0.9983   0.9998
## Pos Pred Value       0.9911   0.9907   0.9740   0.9913   0.9993
## Neg Pred Value       1.0000   0.9962   0.9971   0.9953   0.9988
## Prevalence           0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate       0.2845   0.1904   0.1719   0.1600   0.1828
## Detection Prevalence 0.2870   0.1922   0.1765   0.1614   0.1829
## Balanced Accuracy    0.9982   0.9910   0.9903   0.9871   0.9971
```

## Data Cleaning testing

```
### Removing Variables which are having nearly zero variance

cData_t <- testing_data
for(i in c(8:ncol(cData_t)-1)) {cData_t[,i] =
  as.numeric(as.character(cData_t[,i]))}

# First look at the data for each column and remove variables
# unrelated to exercise (column number and time stamps)
featuresnames <- colnames(cData_t[colSums(is.na(cData_t)) == 0])[-(1:7)]
features_t <- cData_t[featuresnames]
```



## Prediction on testing dataset

```
##prediction on Test dataset
predict_Test <- predict(RForest, newdata=features_t)
predict_Test
```

```
## [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
```

```
print(as.data.frame(predict_Test))
```

```
##      predict_Test
## 1                B
## 2                A
## 3                B
## 4                A
## 5                A
## 6                E
## 7                D
## 8                B
## 9                A
## 10               A
## 11               B
## 12               C
## 13               B
## 14               A
## 15               E
## 16               E
## 17               A
## 18               B
## 19               B
## 20               B
```

## Conclusion

As we can see from the three methods used, each one is improving in its precision but it also has to take into account the time it takes in each of its internal calculations. therefore we stick with rainforest for 99% accuracy.

---