Practical Machine Learning

Emil Bena

3/24/2022

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://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(tidyverse)
library(caret)
library(rpart)
library(rpart.plot)
library(randomForest)

traindata <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv")
testdata <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv")
```

Loading training data and test data

```
dim(traindata)
```

Getting dimensions and heading traindata to see the structure of data

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

```
dim(testdata)
```

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

Cross validation

```
train <- createDataPartition(y=traindata$classe,p=0.8, list=F)
training <- traindata[train,]
testing <- traindata[-train,]
```

Splitting training data into training and testing set

```
train2 <- training %>% select(-(1:7))
test2 <- testing %>% select(-(1:7))
```

Removing columns unecessary for ML models, as they do not contain necessary data

```
train2 <- train2[ , colSums(is.na(train2)) == 0]
test2 <- test2[ , colSums(is.na(train2)) == 0]
```

Removing columns with NA data

```
trainfin <- train2 %>% select(-contains("kurtosis"),-contains("skewness"),-contains("amplitude"),-conta
testfin <- test2 %>% select(-contains("kurtosis"),-contains("skewness"),-contains("amplitude"),-contains
```

Removing columns with very low number of values that need to be removed and storing them as final sets for ML model

I decided to create two different models and use better one on the final prediction

Decission Tree

```
dtree <- rpart(classe ~ ., data=trainfin, method="class")
pred1<- predict(dtree, testfin, type = "class")


confusionMatrix(pred1,as.factor(testfin$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1007  123   17   40    9
##          B   41  436   32   51   64
##          C   24   76  553   89   89
##          D   14   54   39  416   25
##          E   30   70   43   47  534
##
## Overall Statistics
##
##                Accuracy : 0.751
##                  95% CI : (0.7371, 0.7644)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6843
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9023   0.5744   0.8085   0.6470   0.7406
## Specificity            0.9327   0.9406   0.9142   0.9598   0.9407
## Pos Pred Value         0.8420   0.6987   0.6655   0.7591   0.7376
## Neg Pred Value         0.9600   0.9021   0.9576   0.9327   0.9415
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2567   0.1111   0.1410   0.1060   0.1361
## Detection Prevalence   0.3049   0.1591   0.2118   0.1397   0.1846
## Balanced Accuracy      0.9175   0.7575   0.8613   0.8034   0.8407
```

Decission Tree has accuracy of 0.7624

Random Forest

```
rforest <- randomForest(as.factor(classe) ~. , data=trainfin, method="class")
pred2 <- predict(rforest, testfin, type = "class")


confusionMatrix(pred2, as.factor(testfin$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    3    0    0    0
##          B    0  754    1    0    0
##          C    0    2  683    6    0
##          D    0    0    0  635    1
##          E    0    0    0    2  720
##
## Overall Statistics
##
##                Accuracy : 0.9962
##                  95% CI : (0.9937, 0.9979)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9952
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9934   0.9985   0.9876   0.9986
## Specificity            0.9989   0.9997   0.9975   0.9997   0.9994
## Pos Pred Value         0.9973   0.9987   0.9884   0.9984   0.9972
## Neg Pred Value         1.0000   0.9984   0.9997   0.9976   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1922   0.1741   0.1619   0.1835
## Detection Prevalence   0.2852   0.1925   0.1761   0.1621   0.1840
## Balanced Accuracy      0.9995   0.9965   0.9980   0.9936   0.9990
```

Random Forest has accuracy of 0.9952

Final prediction on test data with Random Forest model

```
predict(rforest, testdata, type = "class")
```

```
##  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
```