Machine Learning Course Project

Anik

9/30/2020

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. 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, my goal is 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.

We have analyzed and interpreted our findings with help of machine learning algorithms.We cleaned our data for our analysis for relevant study.

```
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
library(rpart.plot)
```

```
## Loading required package: rpart
```

```
library(rpart)
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
```

```
training <- read.csv('training_data.csv', na.strings = c("NA", "#DIV/0!", ""))
testing <- read.csv('testing_data.csv', na.strings = c("NA", "#DIV/0!", ""))
```

Data Cleaning

Step 1: We first remove those columns that has more than 95% of the observation as NA. We will filter out those records.

```
cleancolumn <- colSums(is.na(training))/nrow(training) < 0.95
cleantraining <- training[,cleancolumn]

# Verifying whether we have removed correctly

colSums(is.na(cleantraining))/nrow(cleantraining)
```

```
##                    X            user_name raw_timestamp_part_1
##                    0                    0                    0
## raw_timestamp_part_2       cvtd_timestamp           new_window
##                    0                    0                    0
##           num_window            roll_belt           pitch_belt
##                    0                    0                    0
##             yaw_belt     total_accel_belt         gyros_belt_x
##                    0                    0                    0
##         gyros_belt_y         gyros_belt_z         accel_belt_x
##                    0                    0                    0
##         accel_belt_y         accel_belt_z        magnet_belt_x
##                    0                    0                    0
##        magnet_belt_y        magnet_belt_z             roll_arm
##                    0                    0                    0
##            pitch_arm              yaw_arm      total_accel_arm
##                    0                    0                    0
##          gyros_arm_x          gyros_arm_y          gyros_arm_z
##                    0                    0                    0
##          accel_arm_x          accel_arm_y          accel_arm_z
##                    0                    0                    0
##         magnet_arm_x         magnet_arm_y         magnet_arm_z
##                    0                    0                    0
```

```
##        roll_dumbbell       pitch_dumbbell         yaw_dumbbell
##                    0                    0                    0
## total_accel_dumbbell     gyros_dumbbell_x     gyros_dumbbell_y
##                    0                    0                    0
##     gyros_dumbbell_z     accel_dumbbell_x     accel_dumbbell_y
##                    0                    0                    0
##     accel_dumbbell_z    magnet_dumbbell_x    magnet_dumbbell_y
##                    0                    0                    0
##    magnet_dumbbell_z         roll_forearm        pitch_forearm
##                    0                    0                    0
##          yaw_forearm  total_accel_forearm      gyros_forearm_x
##                    0                    0                    0
##      gyros_forearm_y      gyros_forearm_z      accel_forearm_x
##                    0                    0                    0
##      accel_forearm_y      accel_forearm_z     magnet_forearm_x
##                    0                    0                    0
##     magnet_forearm_y     magnet_forearm_z               classe
##                    0                    0                    0
```

```r
colSums(is.na(cleantraining))
```

```
##                    X            user_name raw_timestamp_part_1
##                    0                    0                    0
## raw_timestamp_part_2       cvtd_timestamp           new_window
##                    0                    0                    0
##           num_window            roll_belt           pitch_belt
##                    0                    0                    0
##             yaw_belt     total_accel_belt         gyros_belt_x
##                    0                    0                    0
##         gyros_belt_y         gyros_belt_z         accel_belt_x
##                    0                    0                    0
##         accel_belt_y         accel_belt_z        magnet_belt_x
##                    0                    0                    0
##        magnet_belt_y        magnet_belt_z             roll_arm
##                    0                    0                    0
##            pitch_arm              yaw_arm      total_accel_arm
##                    0                    0                    0
##          gyros_arm_x          gyros_arm_y          gyros_arm_z
##                    0                    0                    0
##          accel_arm_x          accel_arm_y          accel_arm_z
##                    0                    0                    0
##         magnet_arm_x         magnet_arm_y         magnet_arm_z
##                    0                    0                    0
##        roll_dumbbell       pitch_dumbbell         yaw_dumbbell
##                    0                    0                    0
## total_accel_dumbbell     gyros_dumbbell_x     gyros_dumbbell_y
##                    0                    0                    0
##     gyros_dumbbell_z     accel_dumbbell_x     accel_dumbbell_y
##                    0                    0                    0
##     accel_dumbbell_z    magnet_dumbbell_x    magnet_dumbbell_y
##                    0                    0                    0
##    magnet_dumbbell_z         roll_forearm        pitch_forearm
##                    0                    0                    0
##          yaw_forearm  total_accel_forearm      gyros_forearm_x
```

```
##                    0                    0                    0
##       gyros_forearm_y      gyros_forearm_z      accel_forearm_x
##                    0                    0                    0
##       accel_forearm_y      accel_forearm_z     magnet_forearm_x
##                    0                    0                    0
##      magnet_forearm_y     magnet_forearm_z               classe
##                    0                    0                    0
```

Step 2:

(i)We will remove unnecessary columns (ii)Partition the trainingdata properly (iii)Will do same for testing data

```
cleantraining <- cleantraining [,-c(1:7)]
cleantest <- testing[,-c(1:7)]
set.seed(34)
inTrainIndex <- caret::createDataPartition(cleantraining$classe,p=0.75,list=FALSE)
trainingdata <- cleantraining[inTrainIndex,]
trainingcrossvalue  <- cleantraining[-inTrainIndex,]
allNames <- names(cleantraining)
cleantest <- testing[,allNames[1:52]]
```

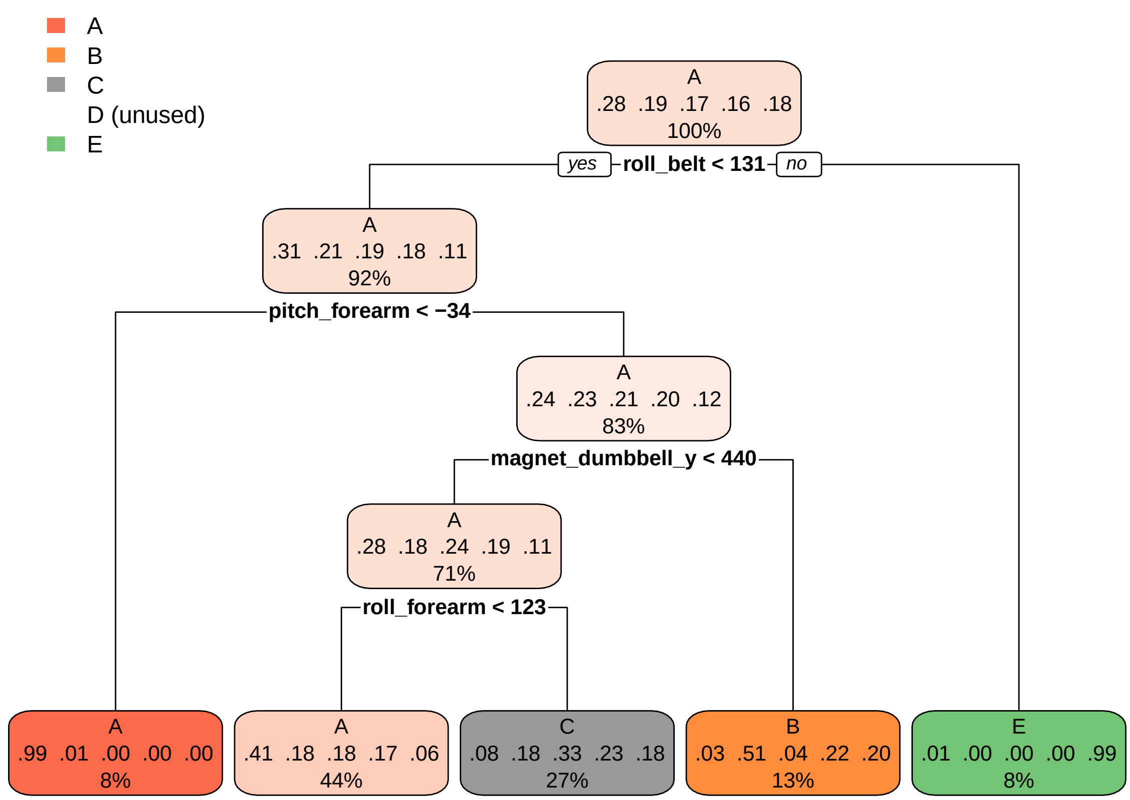
ML Algorithm-Decision Tree

```
decisionTree <- train(classe ~., method='rpart', data=trainingdata)
decisionTreePrediction <- predict(decisionTree, trainingcrossvalue)
trainingcrossvalue$classe <- as.factor(trainingcrossvalue$classe)
confusionMatrix(trainingcrossvalue$classe, decisionTreePrediction)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    A    B    C    D    E
##          A 1263   28  102    0    2
##          B  381  320  248    0    0
##          C  411   25  419    0    0
##          D  368  143  293    0    0
##          E  154  109  241    0  397
##
## Overall Statistics
##
##                Accuracy : 0.4892
##                  95% CI : (0.4751, 0.5033)
##     No Information Rate : 0.5255
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3319
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4901  0.51200  0.32157       NA  0.99499
## Specificity            0.9433  0.85300  0.87892   0.8361  0.88812
## Pos Pred Value         0.9054  0.33720  0.49006       NA  0.44062
## Neg Pred Value         0.6255  0.92288  0.78167       NA  0.99950
## Prevalence             0.5255  0.12745  0.26570   0.0000  0.08136
## Detection Rate         0.2575  0.06525  0.08544   0.0000  0.08095
## Detection Prevalence   0.2845  0.19352  0.17435   0.1639  0.18373
## Balanced Accuracy      0.7167  0.68250  0.60024       NA  0.94156
```

```r
rpart.plot(decisionTree$finalModel)
```



ML Algorithm-Random Forest

```r
randomforest <- train(classe ~., method='rf', data=trainingdata, ntree=50)
rfPrediction <- predict(randomforest, trainingcrossvalue)
confusionMatrix(trainingcrossvalue$classe, rfPrediction)
```

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

```
## Prediction    A    B    C    D    E
##          A 1392    1    1    0    1
##          B    4  943    1    0    1
##          C    0    4  849    2    0
##          D    1    0   12  791    0
##          E    0    0    2    1  898
##
## Overall Statistics
##
##                Accuracy : 0.9937
##                  95% CI : (0.991, 0.9957)
##     No Information Rate : 0.2849
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.992
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9947   0.9815   0.9962   0.9978
## Specificity            0.9991   0.9985   0.9985   0.9968   0.9993
## Pos Pred Value         0.9978   0.9937   0.9930   0.9838   0.9967
## Neg Pred Value         0.9986   0.9987   0.9960   0.9993   0.9995
## Prevalence             0.2849   0.1933   0.1764   0.1619   0.1835
## Detection Rate         0.2838   0.1923   0.1731   0.1613   0.1831
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9978   0.9966   0.9900   0.9965   0.9985
```

```r
predict(randomforest, cleantest)
```

```
##  [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
```

Conclusion

We have used two machine learning algorithms.Among them random forest worked much better than the other one.So inspite of decision tree algorithm,we should use randomforest algorithm.