Practical Machine Learning Course project

Galina Chtcherbakova

November 26, 2017

Summary

The dataset used in this project is called Weight Lifting Exercise Dataset and comes from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har).

The goal of this project is to create a prediction model on common incorrect gestures during barbell lifts based on the data collected by devices such as Jawbone Up, Nike FuelBand, and Fitbit.

To find the best prediction model we clean the data by eliminating missing values. Then data is divided into training set and test set. We use the training set by three different methods: decision trees, random forest and generalized boosted models to find the better predicting method. Chosen method is applied to the test set.

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 (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Data Analysis

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

FIrst, let’s set up an enviroment for our analysis and upload all needed libraries.

```
knitr::opts_chunk$set(echo = TRUE)
rm(list=ls())
library(caret)
library(rpart)
library(randomForest)
library(knitr)
library(ggplot2)
library(rattle)
library(rpart.plot)
```

Loading of training and test data

```
train_Url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_Url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training <- read.csv(url(train_Url), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(test_Url), na.strings=c("NA","#DIV/0!",""))
dim(training)
```

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

```
dim(testing)
```

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

Cleaning data of NA values

```
training <- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]
dim(training)
```

```
## [1] 19622    60
```

```
dim(testing)
```

```
## [1] 20 60
```

Splitting training data into training and testing sets

```
trainingSet <- createDataPartition(training$classe, p=0.6, list=FALSE)
trainSet1 <- training[trainingSet, ]
testSet1 <- training[-trainingSet, ]
dim(trainSet1)
```

```
## [1] 11776    60
```

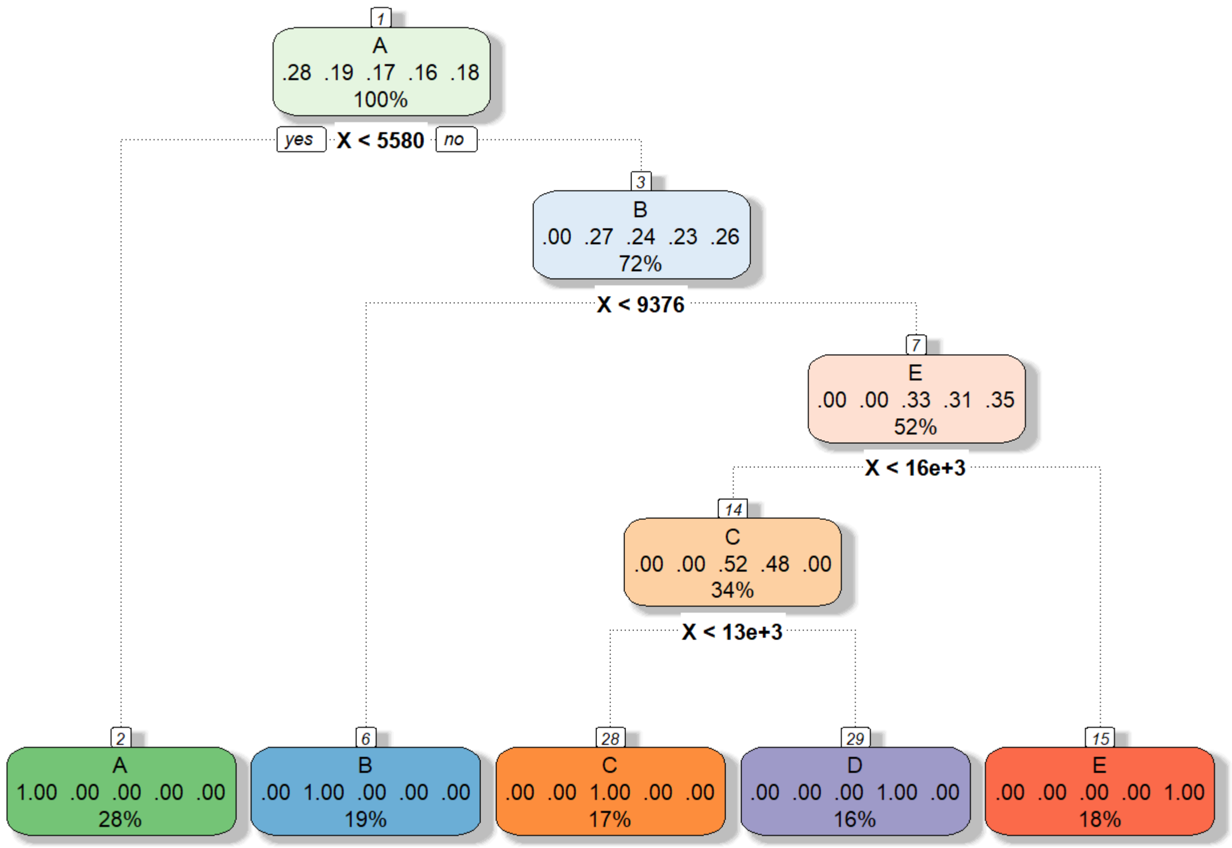
```
dim(testSet1)
```

```
## [1] 7846   60
```

Choosing Prediction Model

Method - Decision Trees

```
set.seed(9876)
modelFitDT <- rpart(classe ~ ., data=trainSet1, method="class")
fancyRpartPlot(modelFitDT)
```



Rattle 2017-Nov-27 00:03:20 Galka

```
predictDT <- predict(modelFitDT, testSet1, type = "class")
confMatDT <- confusionMatrix(predictDT, testSet1$classe)
confMatDT
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    0    0    0    0
##          B    0 1517    0    0    0
##          C    0    1 1367    0    0
##          D    0    0    1 1286    0
##          E    0    0    0    0 1442
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9997          
##                  95% CI : (0.9991, 1)     
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9997          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9993   0.9993   1.0000   1.0000
## Specificity            1.0000   1.0000   0.9998   0.9998   1.0000
## Pos Pred Value         1.0000   1.0000   0.9993   0.9992   1.0000
## Neg Pred Value         1.0000   0.9998   0.9998   1.0000   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1933   0.1742   0.1639   0.1838
## Detection Prevalence   0.2845   0.1933   0.1744   0.1640   0.1838
## Balanced Accuracy      1.0000   0.9997   0.9996   0.9999   1.0000
```

Method - Random Forest

```
set.seed(9876)
controlRF <- trainControl(method="cv", number=5, verbose=FALSE)
modelFitRF <- train(classe ~ ., data=trainSet1, method="rf", trControl=controlRF)
modelFitRF$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 41
##
##         OOB estimate of  error rate: 0.02%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3347    1    0    0    0 0.0002986858
## B    1 2278    0    0    0 0.0004387889
## C    0    0 2054    0    0 0.0000000000
## D    0    0    0 1930    0 0.0000000000
## E    0    0    0    0 2165 0.0000000000
```

```
predictRF <- predict(modelFitRF, newdata=testSet1)
confMatRF <- confusionMatrix(predictRF, testSet1$classe)
confMatRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    0    0    0    0
##          B    0 1517    0    0    0
##          C    0    1 1368    0    0
##          D    0    0    0 1286    0
##          E    0    0    0    0 1442
##
## Overall Statistics
##
##                Accuracy : 0.9999
##                  95% CI : (0.9993, 1)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9998
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9993   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   0.9998   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   0.9993   1.0000   1.0000
## Neg Pred Value         1.0000   0.9998   1.0000   1.0000   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1933   0.1744   0.1639   0.1838
## Detection Prevalence   0.2845   0.1933   0.1745   0.1639   0.1838
## Balanced Accuracy      1.0000   0.9997   0.9999   1.0000   1.0000
```

Method - Generalized Boosted Model

```
set.seed(9876)
controlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modelFitGBM <- train(classe ~ ., data= trainSet1, method="gbm",trControl= controlGBM, verbose=FA
LSE)
modelFitGBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 50 iterations were performed.
## There were 81 predictors of which 6 had non-zero influence.
```

```
predictGBM <- predict(modelFitGBM, newdata=testSet1)
confMatGBM <- confusionMatrix(predictGBM, testSet1$classe)
confMatGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    0    0    0    0
##          B    0 1517    0    0    0
##          C    0    1 1367    0    0
##          D    0    0    1 1286    0
##          E    0    0    0    0 1442
##
## Overall Statistics
##
##                Accuracy : 0.9997
##                  95% CI : (0.9991, 1)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9997
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9993   0.9993   1.0000   1.0000
## Specificity            1.0000   1.0000   0.9998   0.9998   1.0000
## Pos Pred Value         1.0000   1.0000   0.9993   0.9992   1.0000
## Neg Pred Value         1.0000   0.9998   0.9998   1.0000   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1933   0.1742   0.1639   0.1838
## Detection Prevalence   0.2845   0.1933   0.1744   0.1640   0.1838
## Balanced Accuracy      1.0000   0.9997   0.9996   0.9999   1.0000
```

Choosing The Model For Prediction

The accuracy of the 3 regression modeling methods above is very good.

For our project the Random Forest model will be applied to predict the 20 test cases (testing dataset) as shown below.

```
predictTest <- predict(modelFitRF, newdata=testing)
predictTest
```

```
##  [1] A A A A A A A A A A A A A A A A A A A A
## Levels: A B C D E
```