Course Project Machine Learning Week 4

Darima Butitova

6/4/2019

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. The goal of the project is to predict the manner in which 6 participants did the exercise, using data from accelerometers on the belt, forearm, arm, and dumbell. Participants 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).

Data

Two datasets are given with training and testing data. The dependent variable is "classe" - factor variable with 5 levels corresponding to the way participants performed barbell lifts. After cleaning the data, the training data has 19622 observations and 59 variables (including "classe"), testing data has 20 observations and 58 variables. 100 variables were dropped, as 98% were missing values.

```
# Loading Data

fileUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
download.file(fileUrl, destfile = "/Users/darimabutitova/Desktop/Coursera/pml-training.csv", method="cu


fileUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(fileUrl, destfile = "/Users/darimabutitova/Desktop/Coursera/pml-testing.csv", method="curl

training <- read.csv("/Users/darimabutitova/Desktop/Coursera/pml-training.csv", stringsAsFactors = F, na
testing  <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", stringsAsFa

#Cleaning training data

anymissing <- colSums(is.na(training)) # counting number of missing values in each column
        missing <- anymissing[anymissing>0] # looking only at columns with missing values
                missing #19216 missing values in 100 columns
```

```
##       kurtosis_roll_belt      kurtosis_picth_belt        kurtosis_yaw_belt
##                    19216                    19216                    19216
##       skewness_roll_belt     skewness_roll_belt.1        skewness_yaw_belt
##                    19216                    19216                    19216
##            max_roll_belt           max_picth_belt             max_yaw_belt
##                    19216                    19216                    19216
##            min_roll_belt           min_pitch_belt             min_yaw_belt
##                    19216                    19216                    19216
##      amplitude_roll_belt     amplitude_pitch_belt       amplitude_yaw_belt
##                    19216                    19216                    19216
##     var_total_accel_belt            avg_roll_belt         stddev_roll_belt
```

```
##                   19216                   19216                   19216
##            var_roll_belt          avg_pitch_belt       stddev_pitch_belt
##                   19216                   19216                   19216
##           var_pitch_belt            avg_yaw_belt         stddev_yaw_belt
##                   19216                   19216                   19216
##             var_yaw_belt           var_accel_arm            avg_roll_arm
##                   19216                   19216                   19216
##          stddev_roll_arm            var_roll_arm           avg_pitch_arm
##                   19216                   19216                   19216
##         stddev_pitch_arm           var_pitch_arm             avg_yaw_arm
##                   19216                   19216                   19216
##           stddev_yaw_arm             var_yaw_arm       kurtosis_roll_arm
##                   19216                   19216                   19216
##       kurtosis_picth_arm        kurtosis_yaw_arm       skewness_roll_arm
##                   19216                   19216                   19216
##       skewness_pitch_arm        skewness_yaw_arm            max_roll_arm
##                   19216                   19216                   19216
##            max_picth_arm             max_yaw_arm            min_roll_arm
##                   19216                   19216                   19216
##            min_pitch_arm             min_yaw_arm      amplitude_roll_arm
##                   19216                   19216                   19216
##      amplitude_pitch_arm       amplitude_yaw_arm   kurtosis_roll_dumbbell
##                   19216                   19216                   19216
##  kurtosis_picth_dumbbell   kurtosis_yaw_dumbbell   skewness_roll_dumbbell
##                   19216                   19216                   19216
##  skewness_pitch_dumbbell   skewness_yaw_dumbbell        max_roll_dumbbell
##                   19216                   19216                   19216
##       max_picth_dumbbell        max_yaw_dumbbell        min_roll_dumbbell
##                   19216                   19216                   19216
##       min_pitch_dumbbell        min_yaw_dumbbell  amplitude_roll_dumbbell
##                   19216                   19216                   19216
## amplitude_pitch_dumbbell  amplitude_yaw_dumbbell       var_accel_dumbbell
##                   19216                   19216                   19216
##        avg_roll_dumbbell     stddev_roll_dumbbell        var_roll_dumbbell
##                   19216                   19216                   19216
##       avg_pitch_dumbbell    stddev_pitch_dumbbell       var_pitch_dumbbell
##                   19216                   19216                   19216
##         avg_yaw_dumbbell      stddev_yaw_dumbbell         var_yaw_dumbbell
##                   19216                   19216                   19216
##    kurtosis_roll_forearm   kurtosis_picth_forearm     kurtosis_yaw_forearm
##                   19216                   19216                   19216
##    skewness_roll_forearm   skewness_pitch_forearm     skewness_yaw_forearm
##                   19216                   19216                   19216
##         max_roll_forearm        max_picth_forearm         max_yaw_forearm
##                   19216                   19216                   19216
##         min_roll_forearm        min_pitch_forearm         min_yaw_forearm
##                   19216                   19216                   19216
##   amplitude_roll_forearm  amplitude_pitch_forearm    amplitude_yaw_forearm
##                   19216                   19216                   19216
##        var_accel_forearm         avg_roll_forearm      stddev_roll_forearm
##                   19216                   19216                   19216
##         var_roll_forearm        avg_pitch_forearm     stddev_pitch_forearm
##                   19216                   19216                   19216
##        var_pitch_forearm          avg_yaw_forearm       stddev_yaw_forearm
```

```
##                    19216                    19216                    19216
##          var_yaw_forearm
##                    19216
```

```
19216/19622 #[1] 0.9793089 decided to drop since 98% is missing no point to impute... 100 variables to
```

```
## [1] 0.9793089
```

```
training <- training[, colSums(is.na(training))==0] # dropping columns with missing values
training <- training[,-1] # don't need this variable, as it's basically just a row number

#Recoding variables
training$cvtd_timestamp <- as.POSIXct(training$cvtd_timestamp, format="%d/%m/%Y  %H:%M")
training$user_name <- as.factor(training$user_name)
training$new_window <- as.factor(training$new_window)
training$classe <- as.factor(training$classe)


# Cleaning testing data
anymissing <- colSums(is.na(testing))
testing <- testing[, colSums(is.na(testing))==0]
testing <- testing[, -c(1, 60)] # don't need these variables, as it's basically just a row number
testing$cvtd_timestamp <- as.POSIXct(testing$cvtd_timestamp, format="%d/%m/%Y  %H:%M")
testing$user_name <- as.factor(testing$user_name)
testing$new_window <- as.factor(testing$new_window)

#Comparing training and testing data
a <- names(training)
b <- names(testing)
setdiff(b, a)
```

```
## character(0)
```

```
setdiff(a, b)
```

```
## [1] "classe"
```

```
#Making sure the predictors are the same across two datasets
testing <- rbind(training[1, -59], testing) # adding first row of training data to testing while droppi
testing <- testing[-1,] # deleting the row that we just added
```

Data Analysis

Given that our response variable is a factor with 5 levels, and the goal of the project is a classification problem, we choose randomForest. The training data is divided into training1 and validation datasets to compare error rates and predictions on the testing data.

```
# Create training and validation set
library(caret)
```

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

```
## Loading required package: ggplot2
```

```
valid <- createDataPartition(y=training$classe,
                             p=0.8, list=F)
training1 <- training[valid,]
```

```r
validation <- training[-valid,]

#Random Forest
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
```

```r
rftrain <- randomForest(x=training1[,-59], y=training1$classe, data=training1)
rfvalid <- randomForest(x=validation[,-59], y=validation$classe, data=validation)
rftrain
```

```
##
## Call:
##  randomForest(x = training1[, -59], y = training1$classe, data = training1)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.1%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4464    0    0    0    0 0.0000000000
## B    2 3036    0    0    0 0.0006583278
## C    0    3 2735    0    0 0.0010956903
## D    0    0    6 2566    1 0.0027205597
## E    0    0    0    3 2883 0.0010395010
```

```r
rfvalid
```

```
##
## Call:
##  randomForest(x = validation[, -59], y = validation$classe, data = validation)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.89%
## Confusion matrix:
##      A   B   C   D   E class.error
## A 1116   0   0   0   0 0.000000000
## B    6 750   3   0   0 0.011857708
## C    0  10 673   1   0 0.016081871
## D    0   0   8 634   1 0.013996890
## E    0   0   0   6 715 0.008321775
```

```r
#Predicting testing using training1 and validation datasets
pred <- predict(rftrain, testing)
predval <- predict(rfvalid, testing)
pred
```

```
##  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
##  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
```

```
predval
```

```
##  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
##  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
```

Results

500 trees were built with 7 variables at each split. The error rate for training1 dataset is a bit lower, than the error rate for validaiton dataset, overall the error rates are quite low. The results look good and we procede with predictions. We have similar prediction values after using training1 and validation dataset on testing data.

B A B A A E D B A A B C B A E E A B B B

Thus, randomForest worked great on this classification problem.