Qualitative Activity Recognition

Daria Karpova

5/1/2021

About this project

This project has been created for **Getting and Cleaning Data** course project from 'Data Science Specialization' on Coursera provided by 'Johns Hopkins University'. The goal is to predict the quality of performed activies based on the data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The utilized dataset is the **Weight Lifting Exercises Dataset** presented in the following paper:

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Read more: http:/groupware.les.inf.puc-rio.br/har#ixzz4TjtH0uqp

The dataset description can be found here:

http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har

The training data for this project is available here:

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

The test data is available here:

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

Loading data

```r
library(knitr)
library(caret)
library(randomForest)
library(mlbench)
library(stringr)
library(lubridate)
library(dplyr)

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

train_filename = "pml-training.csv"
test_filename = "pml-testing.csv"

if (!file.exists(train_filename)) {
  download.file(train_url, train_filename)
}
```

```
if (!file.exists(test_filename)) {
  download.file(test_url, test_filename)
}

train_set <- read.csv(train_filename, header=TRUE)
test_set <- read.csv(test_filename, header=TRUE)
```

Exploring and preprocessing data

The data has a lot of columns consisting mostly of NA's both in the training and the test sets. Since columns containing pretty much only NA values won't give us any valuable information let's remove those columns with more than half of the rows missing.

```
# Removing columns with more than 5000 missing values
dim(train_set)
```

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

```
dim(test_set)
```

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

```
cols_to_leave <- colSums(is.na(train_set)) < 10000
train_set <- train_set[, cols_to_leave]
test_set <- test_set[, cols_to_leave]

dim(train_set)
```

```
## [1] 19622    93
```

```
dim(test_set)
```

```
## [1] 20 93
```

Now our test consists out of 93 variables. Observing the data has shown some variables to contain empty strings instead of NA's, let's remove those ones as well.

```
cols_to_leave <- colSums(train_set != "") > 10000
train_set <- train_set[, cols_to_leave]
test_set <- test_set[, cols_to_leave]

dim(train_set)
```

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

```
dim(test_set)
```

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

Another two variables which won't help us much are X and user_name. In fact, training with those two will only hurt our test accuracy since a test set may consist of data collected from people not presented on the training set. Therefore, 'X' and 'user_name' are going to be dropped.

```
#Removing columns which do not contribute to the prediction
train_set <- subset(train_set, select = -c(X, user_name))
test_set <- subset(test_set, select = -c(X, user_name))
```

Now we have 2 char variables left: 'cvtd_timestamp' and 'new_window'. Let's convert them to numeric types.

```
#Transforming 'new_window' into a factor variable
train_set$new_window <- as.factor(train_set$new_window)
test_set$new_window <- as.factor(test_set$new_window)

train_set$new_window <- as.numeric(train_set$new_window) - 1
test_set$new_window <- as.numeric(test_set$new_window) - 1

#Converting to datetime
train_set$datetime = dmy_hm(train_set$cvtd_timestamp)
test_set$datetime = dmy_hm(test_set$cvtd_timestamp)

#Extracting valuable information for the train set
train_set$year = year(train_set$datetime)
train_set$month = month(train_set$datetime)
train_set$day_of_month = mday(train_set$datetime)
train_set$day_of_year = yday(train_set$datetime)
train_set$weekday = wday(train_set$datetime)
train_set$hours = hour(train_set$datetime)
train_set$minutes = minute(train_set$datetime)

#Extracting valuable information for the test set
test_set$year = year(test_set$datetime)
test_set$month = month(test_set$datetime)
test_set$day_of_month = mday(test_set$datetime)
test_set$day_of_year = yday(test_set$datetime)
test_set$weekday = wday(test_set$datetime)
test_set$hours = hour(test_set$datetime)
test_set$minutes = minute(test_set$datetime)

#Removing the original timastamp columns
train_set <- subset(train_set, select = -c(cvtd_timestamp, datetime))
test_set <- subset(test_set, select = -c(cvtd_timestamp, datetime))
```

Before we start training our model let's check if there are any missing values or non-numeric columns left.

```
#Checking for missing values
missing_num<-sum(is.na(train_set)) + sum(is.na(test_set))
missing_num
```

```
## [1] 0
```

```r
#Checking the column types
train_set %>% select_if(~!is.numeric(.x)) %>% head()
```

```
##   classe
## 1      A
## 2      A
## 3      A
## 4      A
## 5      A
## 6      A
```

```r
test_set %>% select_if(~!is.numeric(.x)) %>% head()
```

```
## data frame with 0 columns and 6 rows
```

The last char column is the value we predict - 'classe'. Let's factorize it before we start training.

```r
#Transforming 'new_window' into a factor variable
train_set$classe <- as.factor(train_set$classe)
```

Training

The variable we will be trying to predict is called 'Classe'. To perform that random forest classifier will be used. As R's randomForest automatically uses OOB score no validation set is needed.

```r
set.seed(123)
rf_classifier = randomForest(classe ~ ., data=train_set, ntree=100, mtry=2, importance=TRUE)
rf_classifier
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = train_set, ntree = 100,      mtry = 2, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0.3%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 5579    1    0    0    0 0.0001792115
## B   11 3784    2    0    0 0.0034237556
## C    0   11 3411    0    0 0.0032144944
## D    0    0   28 3186    2 0.0093283582
## E    0    0    0    4 3603 0.0011089548
```

OOB estimate of error rate is 0.3% which is a very good result and means 99.7% accuracy on the validation set.

Predicting

Now let’s predict the quality of the performed activities for the test set.

```r
prediction_for_table <- predict(rf_classifier, test_set)
prediction_for_table
```

```
##  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
```

Accessing project quiz has shown that all of the predictions were correct.