Human activity recognition analysis

Luis Terán

14/07/2020

Description

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is 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, it is used 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. The goal of this project is to predict the manner in which they did the exercise and generate a prediction model for well or bad performed manners in excercise.

The data set was obtained from Groupware site on the Weight Lifting Exercise Dataset section at:

- http://groupware.les.inf.puc-rio.br/har

The complete data used consists two datasets (one for training and one for testing) with 19,622 observations(rows) and 160 different kind of measurements(columns). The "classe" variable contains the correct manner for training model into the prediction.

The data displays are cut for document structure purposes, we encourage you to see the whole display for better understanding

Preprocessing

The packages needed are loaded

```
library(caret)
```

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

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

```
library(e1071)
```

Initially, it is reviewed if the dataset is already downloaded. If is not, the files from training and testing activity gets downloaded.

```
if (!file.exists('pml-training.csv')) {
  download.file('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv',
                destfile = './pml-training.csv',
                method = 'curl', quiet = T)
}

if (!file.exists('pml-testing.csv')) {
  download.file('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv',
                 destfile = './pml-testing.csv',
                 method = 'curl', quiet = T)
}

train<-read.csv("pml-training.csv")
test<-read.csv("pml-testing.csv")
```

A first look of the data is displayed

```
head(train[1:5])
```

```
##   X user_name raw_timestamp_part_1 raw_timestamp_part_2   cvtd_timestamp
## 1 1  carlitos           1323084231               788290 05/12/2011 11:23
## 2 2  carlitos           1323084231               808298 05/12/2011 11:23
## 3 3  carlitos           1323084231               820366 05/12/2011 11:23
## 4 4  carlitos           1323084232               120339 05/12/2011 11:23
## 5 5  carlitos           1323084232               196328 05/12/2011 11:23
## 6 6  carlitos           1323084232               304277 05/12/2011 11:23
```

We first notice that there are a lot of variable measurement and maybe not all important for our future model. As we see, there a lot of columns that mostly contain NA vaues.

```
str(train[,20:30])
```

```
## 'data.frame':    19622 obs. of  11 variables:
##  $ max_yaw_belt        : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ min_roll_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_belt      : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_belt        : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ amplitude_roll_belt : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt: int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt  : Factor w/ 4 levels "","#DIV/0!","0.00",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ var_total_accel_belt: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_belt    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
```

The first step is to remove columns with NA values that are not significant for the model.

```
train<-train[, colSums(is.na(train)) == 0]
test<-test[, colSums(is.na(test)) == 0]
```

But still there a lot of variable measurements, other criteria for removing are factor variables. Factor variables are columns with string values in the data, with no measurement of the variable. We take a look at it

```
is.fact <- sapply(train, is.factor)
colsToRemove<-train[is.fact]
head(colsToRemove[,1:9])
```

```
##   user_name   cvtd_timestamp new_window kurtosis_roll_belt kurtosis_picth_belt
## 1  carlitos 05/12/2011 11:23         no
## 2  carlitos 05/12/2011 11:23         no
## 3  carlitos 05/12/2011 11:23         no
## 4  carlitos 05/12/2011 11:23         no
## 5  carlitos 05/12/2011 11:23         no
## 6  carlitos 05/12/2011 11:23         no
##   kurtosis_yaw_belt skewness_roll_belt skewness_roll_belt.1 skewness_yaw_belt
## 1
## 2
## 3
## 4
## 5
## 6
```

So we notice that most of them are useless for the model. The only valuable measurements are the first three rows ("user_name","cvtd_timestamp","new_window") and the last classication one ("classe"). A new data frame is created in order to show variables with no significant measurements, most of the values with no data.

```
colsToRemove<-colsToRemove[,-c(1:3, ncol(colsToRemove))]
head(colsToRemove[1:8],10)
```

```
##    kurtosis_roll_belt kurtosis_picth_belt kurtosis_yaw_belt skewness_roll_belt
## 1
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
## 10
##    skewness_roll_belt.1 skewness_yaw_belt max_yaw_belt min_yaw_belt
## 1
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
## 10
```

Now we create a new data frame with a discrete set of measurements, all of them containing values that can be processed for the model. Also, the first 6 variables like timestamp can not contribute to the prediction of a new prediction.

```
train<-train[,!names(train) %in% names(colsToRemove)]
train<-train[, 7:ncol(train)]
test<-test[,!names(test) %in% names(colsToRemove)]
test<-test[, 7:ncol(test)]
head(train[1:11])
```

```
##   num_window roll_belt pitch_belt yaw_belt total_accel_belt gyros_belt_x
## 1         11      1.41       8.07    -94.4                3         0.00
## 2         11      1.41       8.07    -94.4                3         0.02
## 3         11      1.42       8.07    -94.4                3         0.00
## 4         12      1.48       8.05    -94.4                3         0.02
## 5         12      1.48       8.07    -94.4                3         0.02
## 6         12      1.45       8.06    -94.4                3         0.02
##   gyros_belt_y gyros_belt_z accel_belt_x accel_belt_y accel_belt_z
## 1         0.00        -0.02          -21            4           22
## 2         0.00        -0.02          -22            4           22
## 3         0.00        -0.02          -20            5           23
## 4         0.00        -0.03          -22            3           21
## 5         0.02        -0.02          -21            2           24
## 6         0.00        -0.02          -21            4           21
```

Another way for verify the variables importance for the model is the variance. We call the "nearZeroVar" function and as we expected all variables now are not near zero variance.

```
head(nearZeroVar(train, saveMetrics = T))
```

```
##                  freqRatio percentUnique zeroVar   nzv
## num_window        1.000000     4.3726430   FALSE FALSE
## roll_belt         1.101904     6.7781062   FALSE FALSE
## pitch_belt        1.036082     9.3772296   FALSE FALSE
## yaw_belt          1.058480     9.9734991   FALSE FALSE
## total_accel_belt  1.063160     0.1477933   FALSE FALSE
## gyros_belt_x      1.058651     0.7134849   FALSE FALSE
```

The data is split, 70% for the training section and 30% for the test data. A seed is defined for reproducible purposes.

```
set.seed(1234)
inTrain <- createDataPartition(y = train$classe, p = 0.7, list = F)
training <- train[inTrain, ]
validation <- train[-inTrain, ]
```

Model training

Now that we have tidy and split data, we begin defining a 5 fold cross validation train control.

```
crossNum<-5
trainCtl <- trainControl(method = 'cv', number = crossNum)
```

Then we start the data modeling. Three different models are created with the train control defined before. The models correspond to:

- modrf: Random forest model
- modgbm: Gradient boosting method model
- modSvm: Support vector machine model

```
modrf<-train(classe ~., method = 'rf', data = training, trControl=trainCtl)
modgbm<-train(classe ~., method = 'gbm', data = training, verbose=F, trControl=trainCtl)
modSvm<-svm(classe ~ ., data = training, trControl=trainCtl)
```

Now the test downloaded data is predicted using all the models.

```
predrf<-predict(modrf, newdata = validation)
predgbm<-predict(modgbm, newdata = validation)
predsvm<-predict(modSvm, newdata = validation)
```

Finally, the final accuracy of the models is evaluated.

```
confusionMatrix(predrf, validation$classe)$overall[1]
```

```
## Accuracy
## 0.997791
```

```
confusionMatrix(predgbm, validation$classe)$overall[1]
```

```
##  Accuracy
## 0.9877655
```

```
confusionMatrix(predsvm, validation$classe)$overall[1]
```

```
##  Accuracy
## 0.9517417
```

A further analysis can be made by stacking models. Since we got a really high accuracy, we consider this is no longer necessary.

```
#dataStack <- data.frame(predrf, predgbm, predsvm, classe = test$classe)
#modStack <- train(classe ~., data = dataStack, method = 'gam')
#predStack <- predict(modStack, newdata = dataStack)
#confusionMatrix(predStack, dataStack$diagnosis)$overall[1]
```

Conclusions

Three models were tested in the aim of getting the best prediction for "classe" activity. Two out of the three models presented incredibly good approximations to the expected values. The data fir for the different models was:

- Random forest model accuracy: 99.7%
- Gradient boosting method model: 98.7%
- Support Vector Machine: 95.17%

Being the random forest model the best approximation

The final prediction for the test data in the random forest model:

```
predict(modrf, test)
```

```
##  [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
```

The final prediction for the test data in the grading boosting method model:

```
predict(modgbm, test)
```

```
##  [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
```

The final prediction for the test data in the support vector machinr model:

```
predict(modSvm, test)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  A  A  A  E  D  B  A  A  B  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```