Machine Learning

Jesus

27/1/2021

Executive Summary

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 (see the section on the Weight Lifting Exercise Dataset).

Data

The training data for this project are available here:

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

The test data are available here:

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

The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Goal

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases. ##Steps 1. Loadind data. 2. Split data in a train and test dataset. 3. Data processing in order to eliminate columns with no significance or few values. 4. Apply a random forest fit model on the train data (we will see we can make this assumption due to results obtained) 5. Apply a cross validation in order to certify the model fit.

Imports

Preliminaries

```
library(tidyr)
require(knitr)
library(dplyr)
library(caret)
library(randomForest)
```

Data Processing

1. Load the Data

We load the dataset with the information into the train an test datasets (the files were downloaded previously from original web via "downloadcsv", in order to don't need internet connection to execute the code):

```
train_dataset <- read.csv("pml-training.csv",na.strings=c("NA","#DIV/0!",""))
test_dataset  <- read.csv("pml-testing.csv", na.strings =c("NA","#DIV/0!",""))
```

2. We exclude columns without implication:

```
train_dataset <- train_dataset[,-c(1,3,4,5,6,7)]
test_dataset <- test_dataset[,-c(1,3,4,5,6,7)]
```

3. We separate original train dataset in two datasets for training and validation:

```
set.seed(122)
part <- createDataPartition(train_dataset$classe, p = 0.7, list = FALSE)
TrainSet <- train_dataset[part, ]
ValSet <- train_dataset[-part, ]
```

4. We want to select which columns apply on the model. Using an auxiliar variable, first We exclude columns with near 0 var:

```
# Auxiliar variable 2
TrainSet2<-TrainSet
TrainSet2 <- TrainSet2[, -nearZeroVar(TrainSet2)]
```

5. We want to exclude columns with >85% NA values for the train dataset:

```
ratioCols <- colSums(is.na(TrainSet2))/nrow(TrainSet2)
colsgreat85 <- which(ratioCols > 0.85)
TrainSet2 <- TrainSet2[,-colsgreat85]
```

6. We want to exclude the user name column:

```
TrainSet2 <- TrainSet2[,-grep("user_name",names(TrainSet2))]
```

7. Final columns to use in the model to fit:

```
fitCol<-names(TrainSet2)
fitCol
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"     "classe"
```

MODEL

1. We apply a random Forest model over the original Train Dataset over the final columns (we assume it a good fit model to use due to the quantity of existing data as we will see).

We could include the use of other models as Gradient boosted via function: train(classe ~ ., data=TrainSet, method="gbm") but the best adjustment obtained is via Random Forest (we won't add this information of model "gbm"in* *order to make the document shorter).*

```
TrainSet <- TrainSet[,fitCol]

TrainSet$classe = factor(TrainSet$classe)
rfModel <- randomForest(classe ~ ., data = TrainSet, importance = TRUE, ntrees = 10)
```

2. We apply it over the same dataset to observe accuracy:

```
predicttraining <- predict(rfModel, TrainSet)
confusionMatrix(predicttraining, TrainSet$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3906    0    0    0    0
##          B    0 2658    0    0    0
##          C    0    0 2396    0    0
##          D    0    0    0 2252    0
##          E    0    0    0    0 2525
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

As we expected we get a good accuracy, we must certify with a cross validation.

3. We apply it over validation dataset to observe accuracy on the cross validation:

```
ValSet$classe = factor(ValSet$classe)
predicttraining <- predict(rfModel, ValSet)
confusionMatrix(predicttraining, ValSet$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
```

```
##          A 1674   15    0    0    0
##          B    0 1121    5    0    0
##          C    0    3 1017   10    1
##          D    0    0    4  953    1
##          E    0    0    0    1 1080
##
## Overall Statistics
##
##                Accuracy : 0.9932
##                  95% CI : (0.9908, 0.9951)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9914
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9842   0.9912   0.9886   0.9982
## Specificity            0.9964   0.9989   0.9971   0.9990   0.9998
## Pos Pred Value         0.9911   0.9956   0.9864   0.9948   0.9991
## Neg Pred Value         1.0000   0.9962   0.9981   0.9978   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1905   0.1728   0.1619   0.1835
## Detection Prevalence   0.2870   0.1913   0.1752   0.1628   0.1837
## Balanced Accuracy      0.9982   0.9916   0.9942   0.9938   0.9990
```

Result

With the cross validation of the model we get a 99.32 % accuracy on the validation set. Its quite a good value as was expected by the assumption on point 1. The model adjust to the data of the experiment.

The same result is obtained if the model is applied to dataset test_dataset (not added in this document).