Practical Machine Learning - Course Project

Jorge Gervasio Pereira

26 de novembro de 2016

Introduction

The goal of this project is develop a routine to deploy a machine-learning algorithm that can correctly identify the quality of barbell bicep curls by using data collected from enthusiasts who take measurements about themselves regularly to improve their health.

Objectives

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.

The goal of this project is to build a machine learning algorithm to predict activity quality (classe) from activity monitors.

On line Repositories

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)

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har (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.

```
set.seed(837642)

trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"


#Reading taining and testing set without NA variable
training <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))
```

Preparing Data:

Removing zero value data from datasets

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

Removing first seven columns, since them harbor no information

```
trainData <- training[, -c(1:7)]
testData <- testing[, -c(1:7)]

#Show size of training and testing data sets
dim(trainData)
```

```
## [1] 19622    53
```

```
dim(testData)
```

```
## [1] 20 53
```

Data Splitting

In order to verify training datataset for errors, let's split our sample in 70/30 prediction and validation sets

```
inTrain <- createDataPartition(trainData$classe, p = 0.7, list = FALSE)
train <- trainData[inTrain, ]
valid <- trainData[-inTrain, ]
```

``` ## Choosing algorithm for prediction and Data Modeling

My options for ML algorithms to be used was Random Forest (RF) and Classification Trees (CART).

RF is a very good choice when non-linearies are present on sample data, but CART are more flexible.

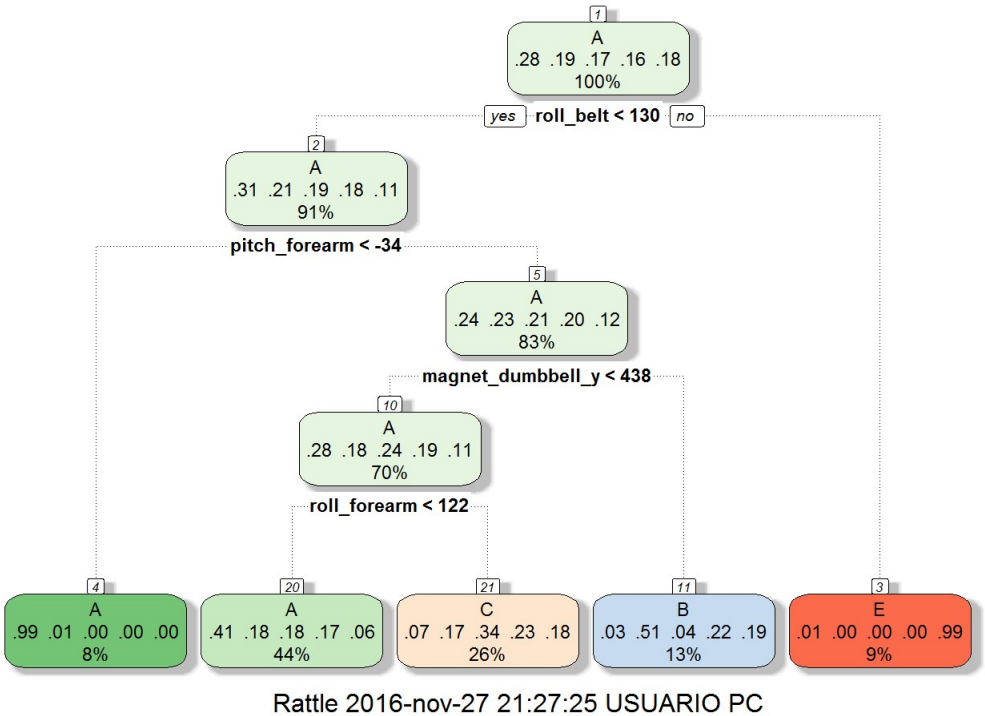
## Classification Trees

In order to economize computer time, let's use 5-fold cross validatio when implementing the algorithm.

```
varcrtl <- trainControl(method = "cv", number = 5)
rpart_ajusted <- train(classe ~ ., data = train, method = "rpart",trControl = varcrtl)
print(rpart_ajusted, digits = 4)
```

```
## CART
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10988, 10990, 10990, 10990
## Resampling results across tuning parameters:
##
##   cp       Accuracy  Kappa
##   0.03560  0.5140    0.36536
##   0.06205  0.4462    0.25707
##   0.11749  0.3332    0.07438
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.0356.
```

```
fancyRpartPlot(rpart_ajusted$finalModel)
```



Rattle 2016-nov-27 21:27:25 USUARIO PC

Let's see the acuracy of our predictions unsing the validation dataset
```

```
rpart_predicted <- predict(rpart_ajusted, valid)

# Aplying confusion matrix
(Matrix_conf <- confusionMatrix(valid$classe, rpart_predicted))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1491   32  144    0    7
##          B  464  367  308    0    0
##          C  481   37  508    0    0
##          D  424  173  367    0    0
##          E  169  147  297    0  469
##
## Overall Statistics
##
##                Accuracy : 0.4817
##                  95% CI : (0.4689, 0.4946)
##     No Information Rate : 0.5147
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3232
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4922  0.48545  0.31281       NA  0.98529
## Specificity            0.9359  0.84948  0.87843   0.8362  0.88667
## Pos Pred Value         0.8907  0.32221  0.49513       NA  0.43346
## Neg Pred Value         0.6348  0.91804  0.77032       NA  0.99854
## Prevalence             0.5147  0.12846  0.27596   0.0000  0.08088
## Detection Rate         0.2534  0.06236  0.08632   0.0000  0.07969
## Detection Prevalence   0.2845  0.19354  0.17434   0.1638  0.18386
## Balanced Accuracy      0.7141  0.66747  0.59562       NA  0.93598
```

Lets now verify the accuracy of CART prediction

```
(verf_acc <- Matrix_conf$overall[1])
```

```
##  Accuracy
## 0.4817332
```

Analyzing the result of confusion matrix, we can see that we have a very poor accuracy result, so we must choose anther kind of algorithm to look for a better outcomo for our prediction

Random Forest

Let's start the use of Random Forest algorithm preparing the dataset

```
fit_rf <- train(classe ~ ., data = train, method = "rf",
                  trControl = varcrtl)
print(fit_rf, digits = 4)
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10990, 10988, 10990, 10991
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa
##    2    0.9903    0.9878
##   27    0.9915    0.9892
##   52    0.9832    0.9787
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Let's now verify the accuracy of this new model

```
# predict outcomes using validation set
predict_rf <- predict(fit_rf, valid)
# Show prediction result
(conf_rf <- confusionMatrix(valid$classe, predict_rf))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1670    4    0    0    0
##          B    8 1125    6    0    0
##          C    0    9 1015    2    0
##          D    1    1    7  954    1
##          E    0    0    2    5 1075
##
## Overall Statistics
##
##                Accuracy : 0.9922
##                  95% CI : (0.9896, 0.9943)
##     No Information Rate : 0.2853
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9901
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9946   0.9877   0.9854   0.9927   0.9991
## Specificity            0.9990   0.9971   0.9977   0.9980   0.9985
## Pos Pred Value         0.9976   0.9877   0.9893   0.9896   0.9935
## Neg Pred Value         0.9979   0.9971   0.9969   0.9986   0.9998
## Prevalence             0.2853   0.1935   0.1750   0.1633   0.1828
## Detection Rate         0.2838   0.1912   0.1725   0.1621   0.1827
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9968   0.9924   0.9916   0.9953   0.9988
```

Algorithm Avaliation and Out-of-Sample Error

For this dataset, the Random Forest (RF) ML algorithm brings the best accuracy on predicting the outcome, far beyeond that on brought by CART. The reason is that Random forests chooses a subset of predictors at each split and decorrelate the trees and this leads to high accuracy.

The accuracy rate is 0.991, and so the out-of-sample error rate is 0.009.

Let's use this algoritm then on our Testing set and analyze the results..

Applying Random Forest algoritm on Testing Set

The output variable is class and let’s see the results

```
r  (predict(fit_rf, testData))
```

```
##  [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
```

Conclusion

Our exercise has displyed a method to compare two diffent types of Machine Leaning algorithms and the necessary steps to choose the best one to an especific data set