Practical Machine Learning Project

Paul Tocto Inga

January 30, 2020

Executive Summary

GitHub Repo: https://github.com/pmtoctoi/Course-Practical-Machine-Learning

The goal of this project is to use data from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants; they were six young healthy participants, that were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions:
1) Class A - exactly according to the specification
2) Class B - throwing the elbows to the front
3) Class C - lifting the dumbbell only halfway
4) Class D - lowering the dumbbell only halfway
5) Class E - throwing the hips to the front

To create a model to predict the manner in which they did the exercise. The model was built in the following steps: a) Getting and preparing the Data b) Developing the model(training and testing) c) Executing the model

Finally the accuracy of the model is aproximatly 0.99, that classiffy to the model as very good. and also there is used cross validation

Reference

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.

Ubication of the Data

The training data are available at:

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

The test data are available at:

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

Getting and preparing the Data

Using the ubication to download the data, load to R and prepare it for
the building the model.

Load the libraries and the path of where is the data

```
library(caret)
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
library(e1071)
library(randomForest)
set.seed(1)
path <- paste(getwd(),"/", "machine", sep="")
train <- file.path(path, "pml-training.csv")
test <- file.path(path, "pml-testing.csv")
```

Fill the data with NA if there are missing data (i.e., “NA”, “#DIV/0!” and "") and put in memmory

```
traindata <- read.csv(train, na.strings=c("NA","#DIV/0!",""))
testdata <- read.csv(test, na.strings=c("NA","#DIV/0!",""))
```

Remove colums that are not needed

```
# Remove the first seven columns because they are not affect the
prediction.
traindatasecond <- traindata[,8:length(colnames(traindata))]
testdatasecond <- testdata[,8:length(colnames(testdata))]
# Remove colums with NAs
traindatasecond <- traindatasecond[, colSums(is.na(traindatasecond)) ==
0]
testdatasecond <- testdatasecond[, colSums(is.na(testdatasecond)) == 0]
# Verify near zero variance predictors
nearzero <- nearZeroVar(traindatasecond,saveMetrics=TRUE)
nearzeroind <- sum(nearzero$nearzero)
if ((nearzeroind>0)) {
        traindatasecond <- traindatasecond[,nearzero$nearzero==FALSE]
}
```

Divide the data

The training data is divided in two groups. This first group is called training set with 70% of the data that is used to train the model. The second group is a validation set used to assess the model performance.

```
datatraining <- createDataPartition(traindatasecond$classe, p=0.70,
list=F)
traindatathird <- traindatasecond[datatraining, ]
validdatafinal <- traindatasecond[-datatraining, ]
```

Model construction

At first train the model

We use a Random Forest model because it automatically selects important variables and is robust to correlated covariates & outliers in general. Also is a good algorithm, for both classification and regression problems, to produce a predictive model. Its default hyperparameters already return great results and the system is great at avoiding overfitting.

```
controlparm <- trainControl(method="cv", 5)
rf.model <- train(classe ~ ., data=traindatasecond, method="rf",
                  trControl=controlparm, ntree=251)
rf.model

## Random Forest
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 15696, 15697, 15699, 15699, 15697
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9940881  0.9925213
##   27    0.9944451  0.9929733
##   52    0.9891956  0.9863327
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

Aproximate the performance

the model generated in the training is tested with the data of validation. Comparing the values generates the accuracy that tell how is the performance of the model with other data.

```
rf.predict <- predict(rf.model, validdatafinal)
confusionMatrix(validdatafinal$classe, rf.predict)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    0 1139    0    0    0
##          C    0    0 1026    0    0
##          D    0    0    0  964    0
##          E    0    0    0    0 1082
## 
## Overall Statistics
##                                      
##                Accuracy : 1          
##                  95% CI : (0.9994, 1)
##     No Information Rate : 0.2845     
##     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             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000

accuracy <- postResample(rf.predict, validdatafinal$classe)
acc.out <- accuracy[1]
overall.ose <-
        1 - as.numeric(confusionMatrix(validdatafinal$classe, rf.predict)
                       $overall[1])
```

Performance of the model

The accuracy is **1** and the Overall Out-of-Sample error is **0**.

Execute the model

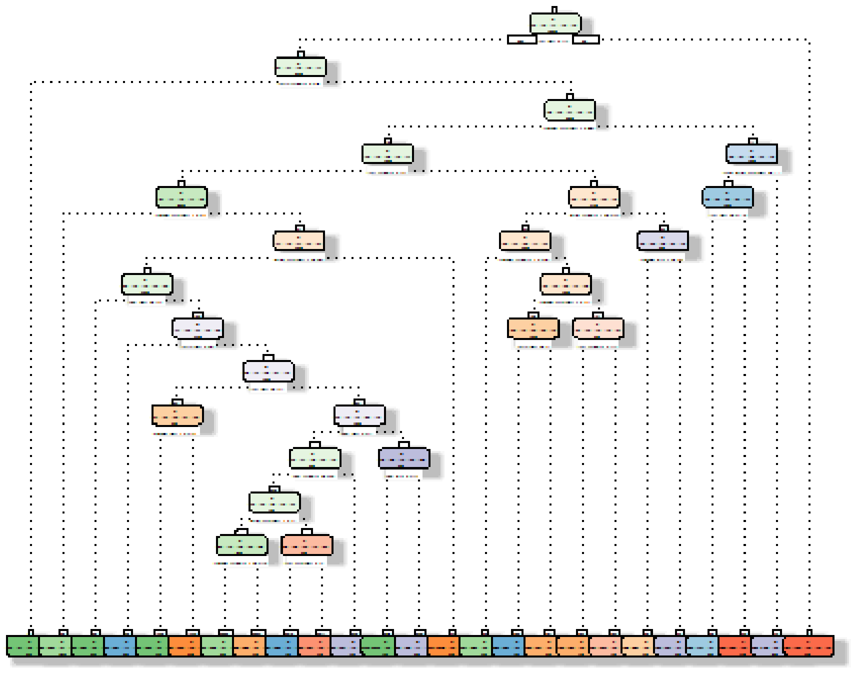
The model is used with the test data.

```
result <- predict(rf.model,
                   testdatasecond[, -length(names(testdatasecond))])
result

##  [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
```

Appendix: Decision Tree Visualization

```
treeModel <- rpart(classe ~ ., data=traindatathird, method="class")
fancyRpartPlot(treeModel)
```


Rattle 2020-ene-30 13:04:29 ptocto