Practical Machine Learning: Coursera Course Project

Helmut Dittrich

June 2016

Practical Machine Learning Project: Prediction Assignment Writeup

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://groupware.les.inf.puc-rio.br/har (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
(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).
The data has been downloaded into the current working directory for easy access

Objective

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.

Load Libraries

```
options(warn = -1)
library(caret)
```

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

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

```
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(Hmisc) # to support the data analysis
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: Formula
```

```
##
## Attaching package: 'Hmisc'
```

```
## The following object is masked from 'package:randomForest':
##
##     combine
## The following objects are masked from 'package:base':
##
##     format.pval, round.POSIXt, trunc.POSIXt, units
```

```
library(foreach) # to reduce processing time of randowm forests
library(doParallel) # by optimizing operation
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
library(rpart)
library(rpart.plot)
```

Read the Data

```
set.seed(1234) # set seed to be reproductible
training_data <- read.csv("pml-training.csv")
testing_data <- read.csv("pml-testing.csv")
```

Analyze the Data

Hide results because they need a lot of space

```
str(training_data)
head(training_data)
describe(training_data)
```

The analysis shows that the training_data:
1. Contain a number of numeric data as factors because of characters like “#DIV/0!” and blanks.
2. A lot of missing data.

Clean the Data

Load the data again and indicate the dubious data as NA

```
training_data <- read.csv("pml-training.csv", na.strings=c("#DIV/0!"," ", "", "NA",
"NAs", "NULL"))
testing_data <- read.csv("pml-testing.csv", na.strings=c("#DIV/0!"," ", "", "NA", "
NAs", "NULL"))
```

We ignore the first 7 columns because they contain names, time_stamps etc. and change the columns 8 to end into numeric values.

```
train_c <- training_data
for(i in c(8:ncol(train_c)-1)) {train_c[,i] = as.numeric(as.character(train_c[,i]))
}
test_c <- testing_data
for(i in c(8:ncol(test_c)-1)) {test_c[,i] = as.numeric(as.character(test_c[,i]))}
```

As predictors we only want the columns without any missing values and we don't want any useless columns as predictors like the first 7. These are "X", "user_name", timestamps, "new_window" and "num_window".
Get the names of the predictor columns

```
predictor_names_tr <- colnames(train_c[colSums(is.na(train_c)) == 0])[-(1:7)]
predictor_names_te <- colnames(test_c[colSums(is.na(test_c)) == 0])[-(1:7)]
predictor_names_tr
```

```
##  [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"
```

Subset the primary dataset train_c to include only the predictors and the outcome variable classe.

```
predictors_tr <- train_c[predictor_names_tr]
pml_test_clean <- test_c[predictor_names_te]
```

Data Partitioning and Prediction Process

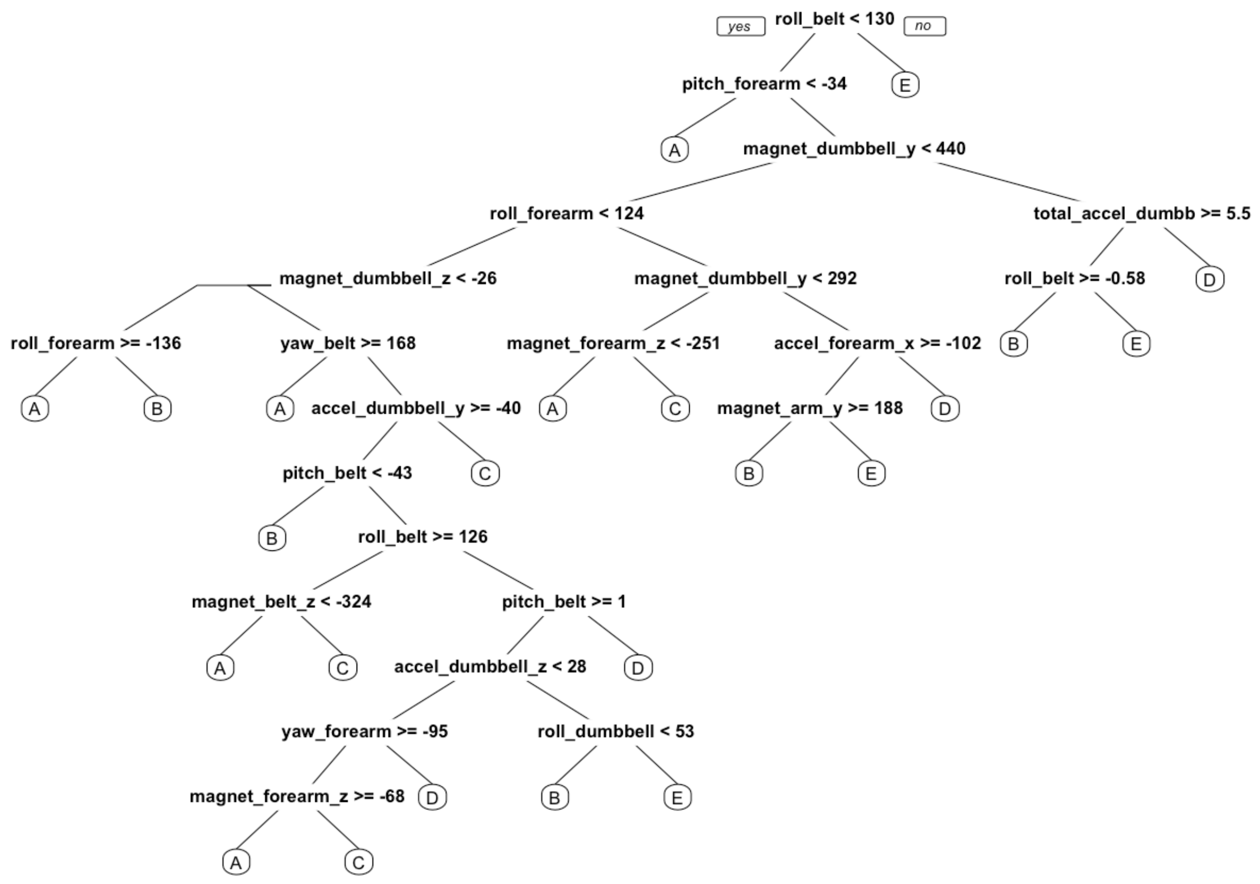
```
inTraining <- createDataPartition(y=predictors_tr$classe, p=3/4, list=FALSE)
training <- predictors_tr[inTraining,]
testing <- predictors_tr[-inTraining,]
```

We now build 5 random forests with 150 trees each. We make use of parallel processing to build this model. Several examples were found of how to perform parallel processing with random forests in R. This provided a great speedup.

```
registerDoParallel()
model <- foreach(ntree=rep(150, 4), .combine=randomForest::combine) %dopar% randomF
orest(training[-ncol(training)], training$classe, ntree=ntree)
```

Decision Tree

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



Evaluate the Model

We use the confusionmatrix method to evaluate the model focusing on accuracy.
Applying to the training data.

```
predictions_tr <- predict(model, newdata=training)
confusionMatrix(predictions_tr,training$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4185    0    0    0    0
##          B    0 2848    0    0    0
##          C    0    0 2567    0    0
##          D    0    0    0 2412    0
##          E    0    0    0    0 2706
##
## 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   0.1839
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1839
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1839
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Applying to the testing data

```
predictions_te <- predict(model, newdata=testing)
confusionMatrix(predictions_te,testing$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    4    0    0    0
##          B    0  944   11    0    0
##          C    0    1  842    7    0
##          D    0    0    2  797    0
##          E    0    0    0    0  901
## 
## Overall Statistics
## 
##                Accuracy : 0.9949          
##                  95% CI : (0.9925, 0.9967)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
## 
##                   Kappa : 0.9936          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9848   0.9913   1.0000
## Specificity            0.9989   0.9972   0.9980   0.9995   1.0000
## Pos Pred Value         0.9971   0.9885   0.9906   0.9975   1.0000
## Neg Pred Value         1.0000   0.9987   0.9968   0.9983   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1925   0.1717   0.1625   0.1837
## Detection Prevalence   0.2853   0.1947   0.1733   0.1629   0.1837
## Balanced Accuracy      0.9994   0.9960   0.9914   0.9954   1.0000
```

As seen as a result of the confusionmatrix, the model for the testing data is fine and efficient with an accuracy of 0.9963 and good values for sensitivity (lowest value 0.9926 for class B) and specificity (lowest value 0.9980 for class A).

Preparation for the Submission

Using the code provided by COURSERA

```
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}
```

Applying the Prediction Model

Get the result for the pml-tesing dataset

```
pred_testing <- predict(model, newdata=pml_test_clean)
#pred_testing
```

Write the result for each case into files

```
#pml_write_files(pred_testing)
```

Conclusion

Applying the Random Forest model to the pml-testing dataset to predict the 20 quiz results shows a 100% accuracy.