Machine Learning on data from Fitness Trackers Devices

Shan Dey

Sunday, August 23, 2015

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 Â- 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 data set, the participants 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 (see the section on the Weight Lifting Exercise Dataset).

In this project, the goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants toto predict the manner in which praticipants did the exercise.

The dependent variable or response is the "classe" variable in the training set.

Data

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://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.

Please download the data if you want to try it out.

```
FALSE Loading required package: lattice
FALSE Loading required package: ggplot2
FALSE Loading required package: RGtk2
FALSE Rattle: A free graphical interface for data mining with R.
FALSE Version 3.5.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
FALSE Type 'rattle()' to shake, rattle, and roll your data.
FALSE randomForest 4.6-10
FALSE Type rfNews() to see new features/changes/bug fixes.
```

Get data into Training and Testing sets. Keep the testing set aside for later use.

```r
# download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", destfile = "./p
# download.file("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", destfile = "./pml

set.seed(12345)

trainData = read.csv("pml-training.csv", na.strings=c("", "NA", "NULL"))

testData = read.csv("pml-testing.csv", na.strings=c("", "NA", "NULL"))
dim(trainData)
```

```
## [1] 19622   160
```

```r
dim(testData)
```

```
## [1]  20 160
```

Clean Up Data, remove columns / variables with #NAs or near Zero Variance

```r
trainData <- trainData[,colSums(is.na(trainData)) == 0]
nearZeroVar <- nearZeroVar(trainData, saveMetrics = TRUE)
trainData <- trainData[, !nearZeroVar$nzv]
trainData <- trainData[,-(1:6)]

dim(trainData)
```

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

Split the Train data into Training and Test for Cross Validation before applying on actual test data

```r
inTrain <- createDataPartition(trainData$classe, p =0.75, list=FALSE)
trainDataForTrain <- trainData[inTrain,]
trainDataForTest <- trainData[-inTrain,]

dim(trainDataForTrain)
```

```
## [1] 14718    53
```

```r
dim(trainDataForTest)
```

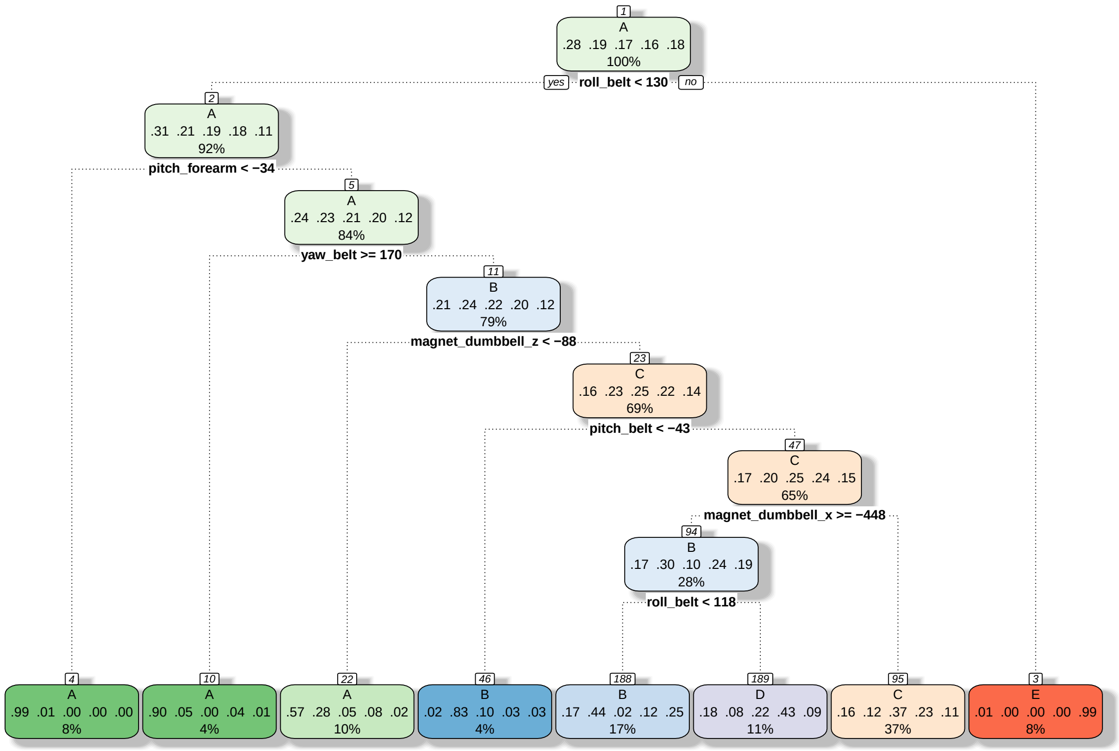
```
## [1] 4904   53
```

Out of sample error

We will use the model accuracy in the cross validation data.Accuracy is determined by the total number of correct predictions against the test dataset genrated from the original datset. Out of sample error is 1 minus accuracy i.e the expected number of misclassified data.

Model 1 - Classification Tree

```r
modelCF <- train(classe ~., method="rpart", data=trainDataForTrain)
fancyRpartPlot(modelCF$finalModel)
```



Rattle 2015−Aug−23 12:16:39 sdey

Applying the Model

Apply the model on the same training dataset and check for accuracy.

```r
predictionCF <- predict(modelCF, trainDataForTrain)
confusionMatrix(predictionCF, trainDataForTrain$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2569  457   69  141   38
##          B  433 1627  119  334  655
##          C  893  631 2037 1263  621
##          D  281  133  342  674  150
##          E    9    0    0    0 1242
##
## Overall Statistics
```

```
##
##                Accuracy : 0.5537
##                  95% CI : (0.5456, 0.5617)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4402
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6139   0.5713   0.7935  0.27944  0.45898
## Specificity            0.9331   0.8702   0.7195  0.92638  0.99925
## Pos Pred Value         0.7847   0.5136   0.3741  0.42658  0.99281
## Neg Pred Value         0.8588   0.8943   0.9428  0.86771  0.89129
## Prevalence             0.2843   0.1935   0.1744  0.16388  0.18386
## Detection Rate         0.1745   0.1105   0.1384  0.04579  0.08439
## Detection Prevalence   0.2224   0.2152   0.3700  0.10735  0.08500
## Balanced Accuracy      0.7735   0.7207   0.7565  0.60291  0.72912
```

The model accuracy is not so good. So, let's try other model.

# Model 2 - Random Forest

```
modelRF <- randomForest(classe ~. , data=trainDataForTrain, method=class)
```

Applying the Model

Apply the model on the same training dataset and check for accuracy.

```
predictionRF <- predict(modelRF, trainDataForTrain)
confusionMatrix(predictionRF, trainDataForTrain$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
```

Accuracy looks good, let's test with test /validation set…

Now try with validation set

Apply the model on the test / validation set from the same training dataset and check for accuracy.

```
predictionRF2 <- predict(modelRF, trainDataForTest)
confusionMatrix(predictionRF2, trainDataForTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    5    0    0    0
##          B    0  939    3    0    0
##          C    0    5  851    7    1
##          D    0    0    1  797    4
##          E    0    0    0    0  896
##
## Overall Statistics
##
##                Accuracy : 0.9947
##                  95% CI : (0.9922, 0.9965)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9933
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9895   0.9953   0.9913   0.9945
## Specificity            0.9986   0.9992   0.9968   0.9988   1.0000
## Pos Pred Value         0.9964   0.9968   0.9850   0.9938   1.0000
## Neg Pred Value         1.0000   0.9975   0.9990   0.9983   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1915   0.1735   0.1625   0.1827
## Detection Prevalence   0.2855   0.1921   0.1762   0.1635   0.1827
## Balanced Accuracy      0.9993   0.9944   0.9961   0.9950   0.9972
```

The random Forest model has produced very high accuracy (over 99%) and hence very little out of sample errors. Let's use the random forest model and apply this to our original test dataset.

Applying the Model to the test data and product output files

Now apply the model on the test data

```r
predictionsRFOnTest <- predict(modelRF, testData, type = "class")
```

Create the output files per test data

```r
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)
  }
}

pml_write_files(predictionsRFOnTest)
```