Practical Machine Learning Project

Gerald Yeo

Tuesday, March 10, 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 - 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, we will 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 (see the section on the Weight Lifting Exercise Dataset).

Aim

The goal of this project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set.

Data Preparation

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.

Our data preparation work includes: 1. Loading the required libraries 2. Reading the training and testing datasets 3. Cleaning data with 'NA' or blank entries

```r
#install.packages("caret")
#install.packages("randomForest")
#install.packages("rpart")
#install.packages("rpart.plot")

library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.2
```

```
## Loading required package: lattice
## Loading required package: ggplot2
## Warning: package 'ggplot2' was built under R version 3.1.2
```

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.1.2
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```r
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.1.2
```

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

```
## Warning: package 'rpart.plot' was built under R version 3.1.2
```

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

trainingdata <- read.csv("pml-training.csv", header = TRUE, na.strings = c("NA", ""))
testingdata <- read.csv("pml-testing.csv", header = TRUE, na.strings = c("NA", ""))
dim(trainingdata)
```

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

```r
dim(testingdata)
```

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

We trimmed the data by doing the following: 1. Delete variables with all missing values 2. Remove variables not relevant for this project

```r
#Delete variables with all missing values
trainingdena <- trainingdata[ , colSums(is.na(trainingdata)) == 0]
testingdena <- testingdata[ , colSums(is.na(testingdata)) == 0]

#Remove variables not relevant for this project
trainingcleaned <- trainingdena[,-c(1:7)]
testingcleaned <- testingdena[,-c(1:7)]

dim(trainingcleaned)
```

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

```r
dim(testingcleaned)
```

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

Splitting the Data

We split the cleaned training dataset into two datasets: 70% for training the model and 30% for validation.

```r
intrain <- createDataPartition(trainingcleaned$classe, p=0.7, list=FALSE)
training <- trainingcleaned[intrain,]
validation <- trainingcleaned[-intrain,]
dim(training)
```

```
## [1] 13737    53
```

```r
dim(validation)
```

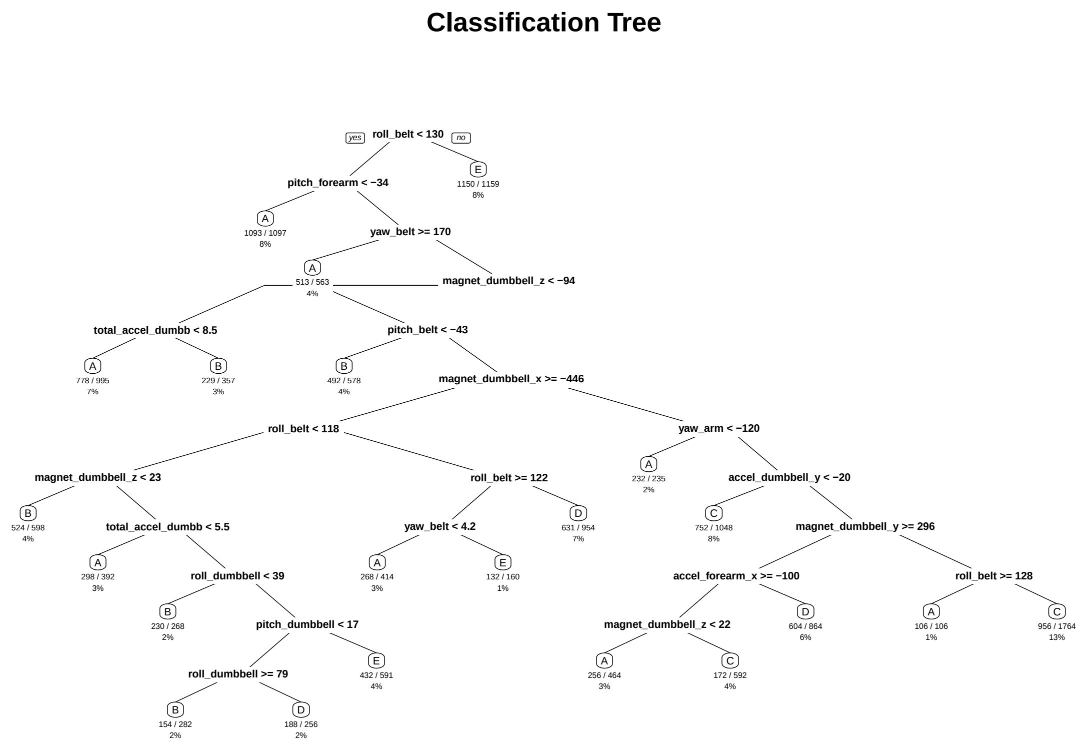
```
## [1] 5885   53
```

Modeling

Decision Tree

We explore the decision tree approach in training a model.

```r
trainingdt <- rpart(classe ~ ., data=training, method="class")
rpart.plot(trainingdt, main="Classification Tree", extra=102, under=TRUE, faclen=0)
```



```r
trainingdt2 <- predict(trainingdt, validation, type = "class")
confusionMatrix(trainingdt2, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1482  154   40   83   40
##          B   45  707   63   36   78
##          C   27  178  802  208  213
##          D   92   72  117  606   43
##          E   28   28    4   31  708
##
## Overall Statistics
##
##                Accuracy : 0.7315
##                  95% CI : (0.72, 0.7428)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6599
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8853   0.6207   0.7817   0.6286   0.6543
## Specificity            0.9247   0.9532   0.8712   0.9342   0.9811
## Pos Pred Value         0.8238   0.7610   0.5616   0.6516   0.8861
## Neg Pred Value         0.9530   0.9128   0.9497   0.9277   0.9265
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2518   0.1201   0.1363   0.1030   0.1203
## Detection Prevalence   0.3057   0.1579   0.2427   0.1580   0.1358
## Balanced Accuracy      0.9050   0.7870   0.8264   0.7814   0.8177
```

Based on the confusion matrix, the decision tree model has a prediction accuracy of about 73%.

Random Forests

We then explore the random forest approach in training a model.

```r
require(randomForest)
set.seed(123)

trainingrf <- randomForest(classe~.,data=training,ntree=100, importance=TRUE)
trainingrf
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training, ntree = 100,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.58%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3902    3    0    0    1 0.001024066
## B   13 2637    8    0    0 0.007900677
## C    0   16 2376    4    0 0.008347245
## D    0    0   25 2226    1 0.011545293
## E    0    2    1    5 2517 0.003168317
```

```
trainingrf2 <- predict(trainingrf, newdata=validation)
confusionMatrix(trainingrf2, validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    3    0    0    0
##          B    0 1136    3    0    0
##          C    0    0 1021   15    0
##          D    0    0    2  948    3
##          E    0    0    0    1 1079
##
## Overall Statistics
##
##                Accuracy : 0.9954
##                  95% CI : (0.9933, 0.997)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9942
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9974   0.9951   0.9834   0.9972
## Specificity            0.9993   0.9994   0.9969   0.9990   0.9998
## Pos Pred Value         0.9982   0.9974   0.9855   0.9948   0.9991
## Neg Pred Value         1.0000   0.9994   0.9990   0.9968   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1930   0.1735   0.1611   0.1833
## Detection Prevalence   0.2850   0.1935   0.1760   0.1619   0.1835
## Balanced Accuracy      0.9996   0.9984   0.9960   0.9912   0.9985
```

Based on the confusion matrix, the random forest model has a prediction accuracy of over 99% which is significantly better than the decision tree model.

Predicted Results

Finally, we apply the random forest model to the initial testing dataset and display the predicted results.

```r
answers <- predict(trainingrf, testingdata)
answers
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```