Practical Machine Learning - Course Project

Fausto Rubino

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. The purpose of this project is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants, in order to predict the manner in which they did the exercise.

Load data and libraries

First of all, let's import the R libraries and data that will be used throughout the analysis.

```r
library(caret)
```

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

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

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
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
```

We also set the seed for reproducibility purposes

```r
set.seed(82637)
```

Getting the data

Download the training and test sets

```
trainingSetUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testingSetUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training <- read.csv(url(trainingSetUrl), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(testingSetUrl), na.strings=c("NA","#DIV/0!",""))
```

Partitioning the data sets

We can now partition the training set into two (60% and 40% size respectively)

```
inTrain <- createDataPartition(training$classe, p=0.6, list=FALSE)
trainingDS <- training[inTrain, ]
testingDS <- training[-inTrain, ]
dim(trainingDS);
```

```
## [1] 11776   160
```

```
dim(testingDS)
```

```
## [1] 7846  160
```

Cleaning the data sets

The following procedures are applied to clean the dataset

Procedure 1: Remove near zero covariates

```
#Remove near zero covariates
nsv <- nearZeroVar(trainingDS, saveMetrics = T)
trainingDS <- trainingDS[, !nsv$nzv]

nzv<- nearZeroVar(testingDS,saveMetrics=TRUE)
testingDS <- testingDS[,nzv$nzv==FALSE]

#Removing first ID variable
trainingDS <- trainingDS[c(-1)]
```

Procedure 2: Remove variables with a high number of missing values

```
trainingBuffer <- trainingDS
for(i in 1:length(trainingDS)) {
    if( sum( is.na( trainingDS[, i] ) ) / nrow(trainingDS) >= .7) {
        for(i1 in 1:length(trainingBuffer)) {
            if( length( grep(names(trainingDS[i]), names(trainingBuffer)[i1]) ) == 1)  {
                trainingBuffer <- trainingBuffer[ , -i1]
            }
        }
```

```
    }
}
#Repopulate the original training dataframe with removed variables with high level of NAs
trainingDS <- trainingBuffer
```

Procedure 3: Remove variables which are not relevant for prediction "user_name" "raw_timestamp_part_1" "raw_timestamp_part_2" "cvtd_timestamp"

```
c1 <- colnames(trainingDS)
c2 <- colnames(trainingDS[, -c(1:4, 58)]) #removing "user_name"  "raw_timestamp_part_1" "raw_timestamp_
testingDS <- testingDS[c1]
testing <- testing[c2]
```

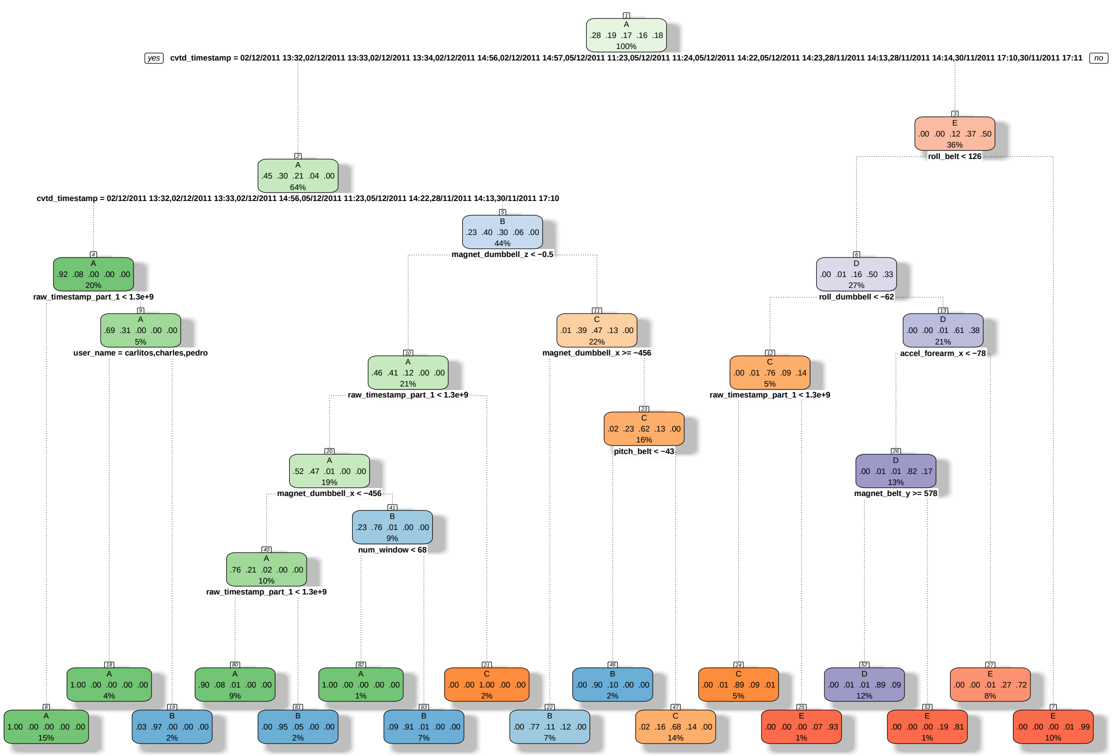
Building the model

Using Decision trees

```
set.seed(91919)
model1Fit <- rpart(classe ~ ., data=trainingDS, method="class")
```

Let's print the decision tree using the fancyRpartPlot library

```
fancyRpartPlot(model1Fit)
```



Rattle 2016−Mar−20 17:50:10 Fausto

Let's apply the model to the testing data frame and print the confusion matrix to test the accuracy of results

```
predictionsModel1 <- predict(model1Fit, testingDS, type = "class")
print(confusionMatrix(predictionsModel1, testingDS$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2148   71    5    1    0
##          B   66 1252   84   57    0
##          C   18  184 1250  207    4
##          D    0   11   14  832   91
##          E    0    0   15  189 1347
##
## Overall Statistics
##
##                Accuracy : 0.8704
##                  95% CI : (0.8627, 0.8777)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.836
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9624   0.8248   0.9137   0.6470   0.9341
## Specificity            0.9863   0.9673   0.9362   0.9823   0.9681
## Pos Pred Value         0.9654   0.8581   0.7517   0.8776   0.8685
## Neg Pred Value         0.9851   0.9584   0.9809   0.9342   0.9849
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2738   0.1596   0.1593   0.1060   0.1717
## Detection Prevalence   0.2836   0.1860   0.2120   0.1208   0.1977
## Balanced Accuracy      0.9743   0.8960   0.9250   0.8146   0.9511
```

The decision tree generated a model with accuracy = 0.8704. ###Using Random Forest

```
set.seed(91919)
model2Fit <- randomForest(classe ~ ., data=trainingDS)
```

```
predictionsModel2 <- predict(model2Fit, testingDS, type = "class")
print(confusionMatrix(predictionsModel2, testingDS$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2230    3    0    0    0
##          B    2 1515    2    0    0
##          C    0    0 1366    2    0
##          D    0    0    0 1284    3
##          E    0    0    0    0 1439
```

```
##
## Overall Statistics
##
##                Accuracy : 0.9985
##                  95% CI : (0.9973, 0.9992)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9981
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9980   0.9985   0.9984   0.9979
## Specificity            0.9995   0.9994   0.9997   0.9995   1.0000
## Pos Pred Value         0.9987   0.9974   0.9985   0.9977   1.0000
## Neg Pred Value         0.9996   0.9995   0.9997   0.9997   0.9995
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2842   0.1931   0.1741   0.1637   0.1834
## Detection Prevalence   0.2846   0.1936   0.1744   0.1640   0.1834
## Balanced Accuracy      0.9993   0.9987   0.9991   0.9990   0.9990
```

The decision tree yialded better results than the decision tree model # with accuracy = 0.9985.

The expected out of sample error is 1-0.9985 = 0.15%.

Run against 20 testing set

print(predict(model2Fit, newdata=testing))