Practical Machine Learning

Tarik Bouchnayaf

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 (see the section on the Weight Lifting Exercise Dataset).

Introduction

The purpose behind this project is to predict the manner the exercises are done, which is the classe variable in our case, of course we can use any other variable. So we're going pull the data, load, and clean it, then test few models and analyse the accuracy of each one.

Data Processing

load the appropriate packages

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.5.1

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 3.5.1

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.5.1
```

```r
library(ggplot2)
library(rpart)
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.5.1
```

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.5.1

## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

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

Download & Import Data

```
URL_training <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
URL_Testing <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
Training_Data <- read.csv(URL_training, na.strings = c("", "NA"))
Testing_Data <- read.csv(URL_Testing, na.strings = c("", "NA"))
```

Data Cleaning

This step includes the NA removal, and filter the columns needed for our exercise, so we avoid errors and optimize the analysis.

```
Training_Data<-Training_Data[,colSums(is.na(Training_Data)) == 0]
Testing_Data <-Testing_Data[,colSums(is.na(Testing_Data)) == 0]
#Delete the 7 first columns due to their irrelevance
Training_Data <- Training_Data[,8:dim(Training_Data)[2]]
Testing_Data <- Testing_Data[,8:dim(Testing_Data)[2]]
```

Partition the Training Data

Partition training dataset using caret into training and test with a ration on .7,.3 respectively

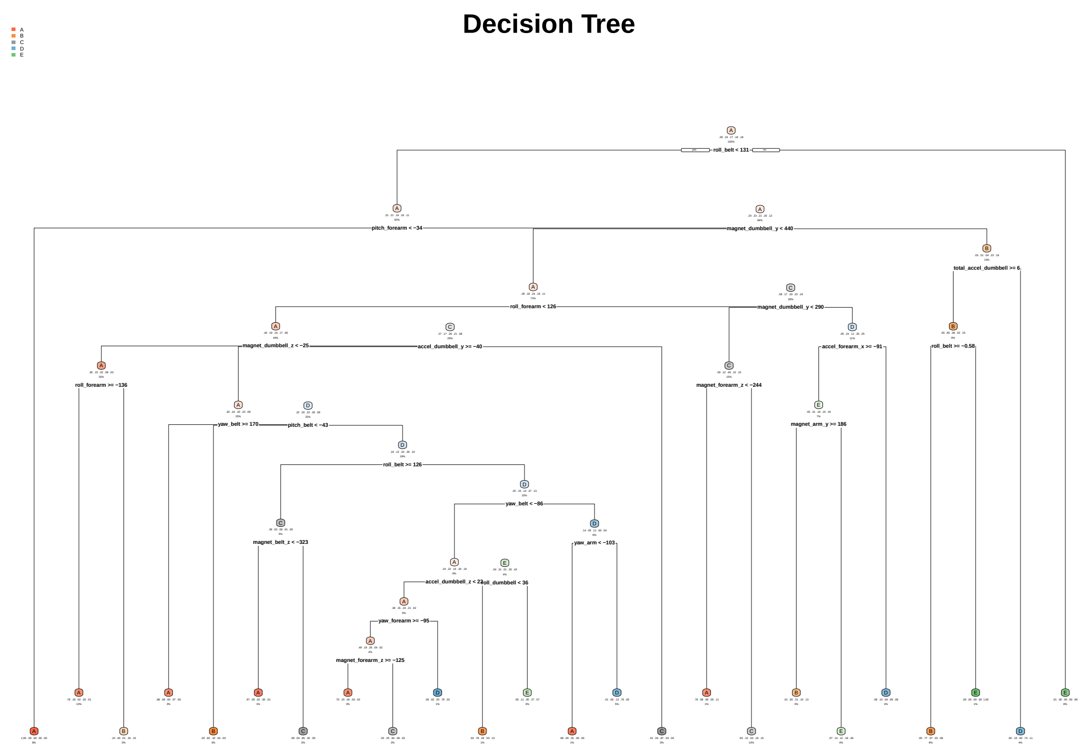
```
partition_data  <- createDataPartition(Training_Data$classe, p=0.7, list=FALSE)
Training_Dataset <- Training_Data[partition_data, ]
Testing_dataset  <- Training_Data[-partition_data, ]
```

Models

1-Decision Tree

```
DTModel <- rpart(classe ~ ., data = Training_Dataset, method = "class")
DTPrediction <- predict(DTModel, Testing_dataset, type = "class")
rpart.plot(DTModel, main = "Decision Tree", under = T, faclen = 0)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



```r
#Testing the result
confusionMatrix(DTPrediction, Testing_dataset$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1525  171   30   40   18
##          B   47  676   91   80   98
##          C   40  146  828  138  121
##          D   22   87   58  614   60
##          E   40   59   19   92  785
##
## Overall Statistics
##
##                Accuracy : 0.7524
##                  95% CI : (0.7412, 0.7634)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6862
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9110   0.5935   0.8070   0.6369   0.7255
## Specificity            0.9385   0.9334   0.9084   0.9539   0.9563
## Pos Pred Value         0.8548   0.6815   0.6504   0.7301   0.7889
## Neg Pred Value         0.9637   0.9054   0.9571   0.9306   0.9393
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2591   0.1149   0.1407   0.1043   0.1334
## Detection Prevalence   0.3031   0.1686   0.2163   0.1429   0.1691
## Balanced Accuracy      0.9247   0.7635   0.8577   0.7954   0.8409
```

1-Random Forest

```
RFModel <- randomForest(classe ~. , data = Training_Dataset, method = "class")
RFPrediction <- predict(RFModel, Testing_dataset, type = "class")
confusionMatrix(RFPrediction, Testing_dataset$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    3    0    0    0
##          B    0 1131   10    0    0
##          C    0    5 1016    6    0
##          D    0    0    0  958    7
##          E    1    0    0    0 1075
##
## Overall Statistics
##
##                Accuracy : 0.9946
##                  95% CI : (0.9923, 0.9963)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9931
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9930   0.9903   0.9938   0.9935
## Specificity            0.9993   0.9979   0.9977   0.9986   0.9998
## Pos Pred Value         0.9982   0.9912   0.9893   0.9927   0.9991
## Neg Pred Value         0.9998   0.9983   0.9979   0.9988   0.9985
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1922   0.1726   0.1628   0.1827
## Detection Prevalence   0.2848   0.1939   0.1745   0.1640   0.1828
## Balanced Accuracy      0.9993   0.9954   0.9940   0.9962   0.9967
```

Conclusion.

The Random Forest is higher in terms of accuracy, so it's the appropriate one to use.