Machine Learning Project

DP

25/04/2015

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, 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; **A B C D E**. The type **A** is the only **correct**. 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).

What we should submit

The goal of the project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. We can use any of the other variables to predict with. We should create a report describing how we built our model, how we used cross validation, what we think the expected out of sample error is, and why we made the choices we did. We will also use our prediction model to predict 20 different test cases.

Getting and Cleaning data

First of all I load the two *.csv files. They contains the training set and the 20 cases we will use the model predictor on.

```
library(caret)
```

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

```
library(rpart)
library(rattle)
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 3.4.1 Copyright (c) 2006-2014 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest)
```

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

```
training<-read.csv("pml-training.csv",header=TRUE,sep=",",na.strings=c("","NA","NUL
L"))
testing<-read.csv("pml-testing.csv",header=TRUE,sep=",",na.strings=c("","NA","NULL"
))
```

Exploratory Data Analysis

We will use the training set to build and test a model predictor. To better perform these steps we only need useful variables. For example we delete variables where all the observations are Not Available:

```
training_clean <- training[, colSums(is.na(training))==FALSE]
testing_clean <- testing[, colSums(is.na(testing))==FALSE]
```

Furthermore we can delete columns that not contain informations on the movements of the users; for example the user names or timestamps...

```
training_clean <- training_clean[, c(8:60)]
testing_clean <- testing_clean[, c(8:60)]

dim(training_clean)
```

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

```
dim(testing_clean)
```

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

Data Slicing

Accuracy of the model built on the training set will be optimistic; infact a better estimate come from an indipendent set (test set). So we slice training data to perform **Cross Validation** : in other words:
1. Take the training data
2. Use 75% of the observations to build the model
3. Calculate the Accuracy on the remaining 25%

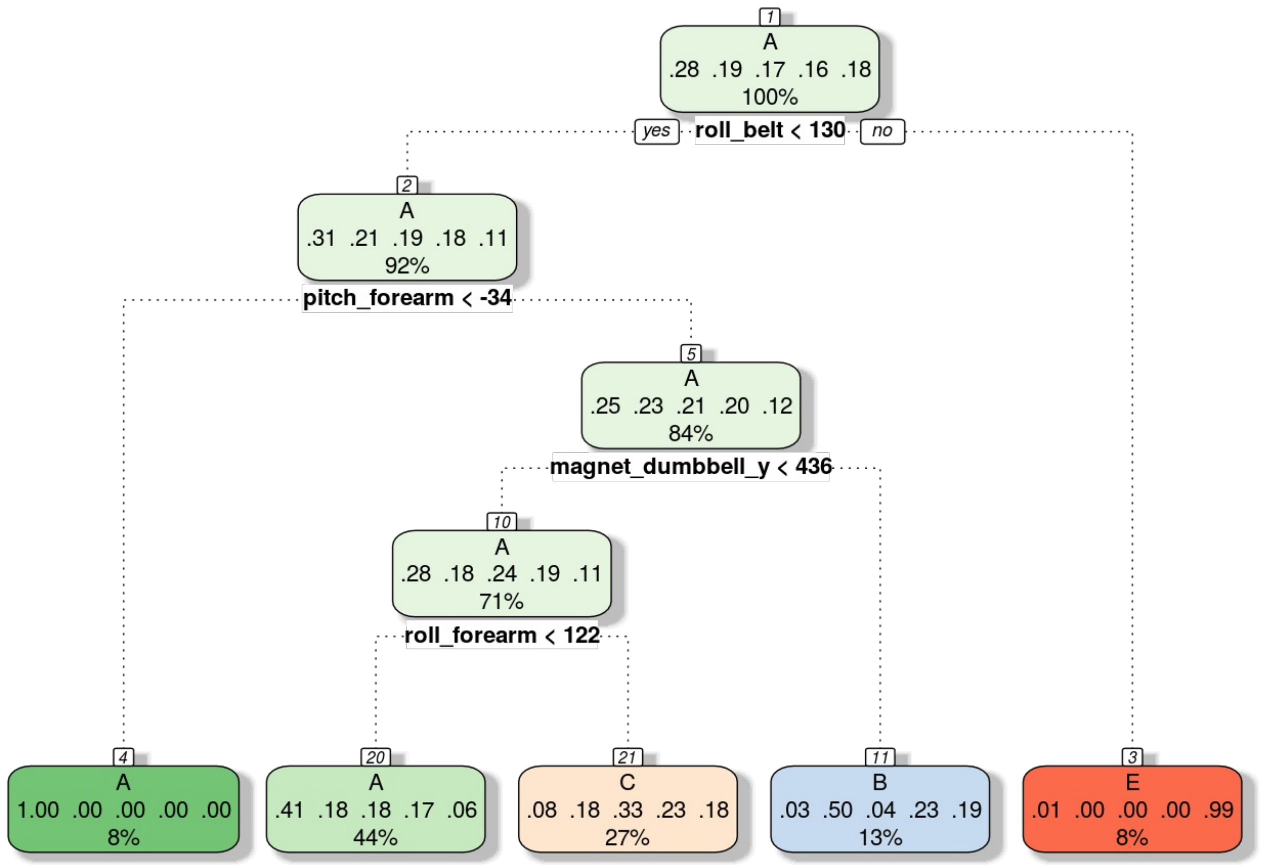
```
inTrain <- createDataPartition(y=training_clean$classe, p=.75, list=FALSE)
train <- training_clean[inTrain,]
test <- training_clean[-inTrain,]
```

Build a good Prediction Model

Decision Trees

The first model we can use is the **Decision Trees**:

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



Rattle 2015-apr-25 20:37:50 leone

```
confusionMatrix( predict(modFit_tree, newdata = test) , test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1252  373  415  351  125
##          B   29  342   25  141  131
##          C  107  234  415  312  251
##          D    0    0    0    0    0
##          E    7    0    0    0  394
##
## Overall Statistics
##
##                Accuracy : 0.49
##                  95% CI : (0.4759, 0.5041)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.334
##  Mcnemar's Test P-Value : NA
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8975  0.36038  0.48538   0.0000  0.43729
## Specificity            0.6398  0.91757  0.77673   1.0000  0.99825
## Pos Pred Value         0.4976  0.51198  0.31463      NaN  0.98254
## Neg Pred Value         0.9401  0.85670  0.87727   0.8361  0.88741
## Prevalence             0.2845  0.19352  0.17435   0.1639  0.18373
## Detection Rate         0.2553  0.06974  0.08462   0.0000  0.08034
## Detection Prevalence   0.5131  0.13622  0.26896   0.0000  0.08177
## Balanced Accuracy      0.7686  0.63898  0.63106   0.5000  0.71777
```

The **Accuracy** is too low to accept this model.

Random Forests

Another model is **Random Forests**:

```
# I tried to use Random Forests with caret package but it never ends
#modFit_rf<-train(classe ~ .,method="rf",data=train)

# Using the randomForest() function is fast enough:
modFit_rf <- randomForest(classe ~ ., data=train)
CM<-confusionMatrix( predict(modFit_rf, newdata = test) , test$classe)
CM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    1    0    0    0
##          B    0  947    2    0    0
##          C    0    1  853   11    0
##          D    0    0    0  793    3
##          E    0    0    0    0  898
##
## Overall Statistics
##
##                Accuracy : 0.9963
##                  95% CI : (0.9942, 0.9978)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9954
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9979   0.9977   0.9863   0.9967
## Specificity            0.9997   0.9995   0.9970   0.9993   1.0000
## Pos Pred Value         0.9993   0.9979   0.9861   0.9962   1.0000
## Neg Pred Value         1.0000   0.9995   0.9995   0.9973   0.9993
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1931   0.1739   0.1617   0.1831
## Detection Prevalence   0.2847   0.1935   0.1764   0.1623   0.1831
## Balanced Accuracy      0.9999   0.9987   0.9973   0.9928   0.9983
```

The **Accuracy** is very good; we accept this model to predict the 20 cases.

Out of sample errors

The Accuracy is:

```
sum(diag( CM$table )) / sum( CM$table )
```

```
## [1] 0.9963295
```

then the out of sample error is:

```
1-(sum(diag( CM$table )) / sum( CM$table ))
```

```
## [1] 0.003670473
```

Predictions

Finally we can predict the type of “classe” for the 20 test cases:

```
predict(modFit_rf, testing_clean)
```

```
##  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
```