Practical Machine Learning Course Project

by Marcelo Dominguez

Contents

Requeriments.

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.

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

Goal.

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

Loading and Cleaning Data.

First step is to load all R required libraries.

```
library(caret)           # Clasification and Regression training
```

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

```
library(randomForest)    # Random fores functions-for classification and regression.
```

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

```
library(rpart)           # Recursive partitioning and regression trees

library(rpart.plot)      # Ploting Rpart Models
library(e1071)
```

Once loaded all required libraries we´re downloading training and test datasets.

Now we are going to convert raw downloaded data into a tidy data set by performing this operations:
- Remove NAs, "" and "#DIV/0!" values.

- Remove variables with time dependence values.
- Remove not needed fields this this analysis.

Cross-Validation.

Training data set was partitioned into 2 subsets to allow crossvalidation (25% - 75%).

```
set.seed(31415)
sub_training_set = createDataPartition(y=training_set$classe, p=0.75, list=FALSE)
sub_Training = training_set[sub_training_set, ]
sub_Testing = training_set[-sub_training_set, ]
```

Results.

The "classe" is an unordered factor variable, decision tree and random forest algorithms are non linear models known for their ability of iteratively split variables intro homogeneus groups and very useful for detecting the features that are important for classification. For this reason we selected both algorithms for this report.

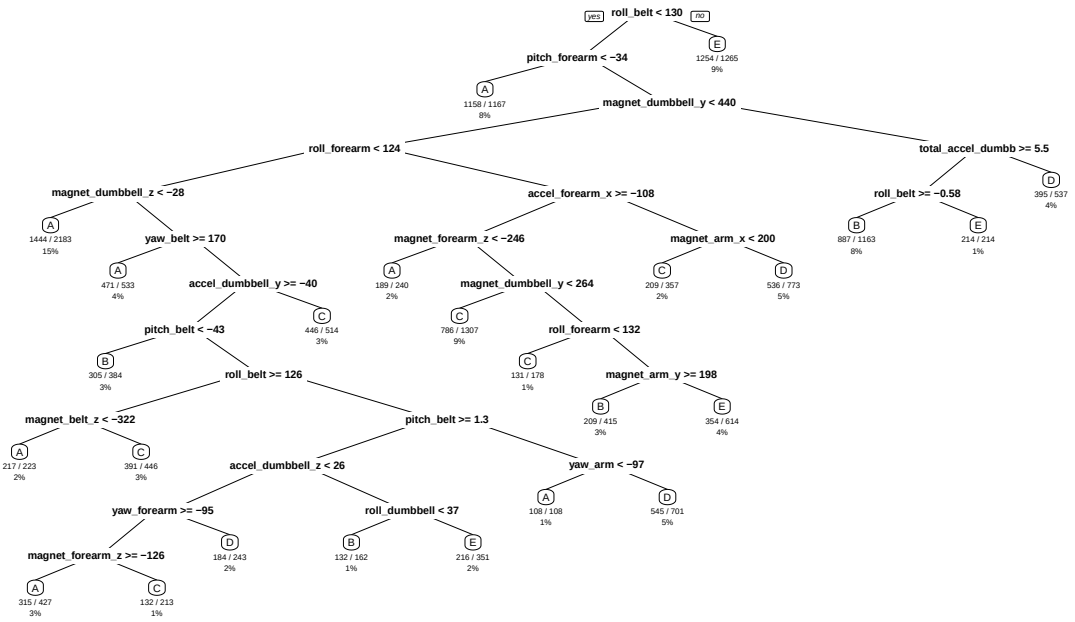
MODEL 1: Decission Trees.

```
# Decission trees
model_dec_trees = rpart(classe ~ ., data=sub_Training, method="class")
model_dec_trees_prediction = predict(model_dec_trees, sub_Testing, type = "class")
```

From Confusion Matrix we can obtain the accuracy for this model, was 0.7397 for a 95% CI : **(0.7273, 0.752)**.
Expected out-of-sample error is estimated at **0.005,a 0.5%**.

```
rpart.plot(model_dec_trees, main="Decision Tree", extra=102, under=TRUE, faclen=0)
```

Decision Tree




```
# Testing results
confusionMatrix(model_dec_trees_prediction, sub_Testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1303  187   19   82   32
##          B   23  522   66   35   64
##          C   34  144  697   77   75
##          D   22   51   58  538   76
##          E   13   45   15   72  654
##
## Overall Statistics
##
##                Accuracy : 0.7573
##                  95% CI : (0.7451, 0.7693)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6915
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9341   0.5501   0.8152   0.6692   0.7259
```

```
## Specificity            0.9088   0.9525   0.9185   0.9495   0.9638
## Pos Pred Value         0.8028   0.7352   0.6787   0.7221   0.8185
## Neg Pred Value         0.9720   0.8982   0.9592   0.9360   0.9398
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2657   0.1064   0.1421   0.1097   0.1334
## Detection Prevalence   0.3310   0.1448   0.2094   0.1519   0.1629
## Balanced Accuracy      0.9214   0.7513   0.8669   0.8093   0.8448
```

MODEL 2: Random Forest.

```
# Random forest
model_rnd_forest = randomForest(classe ~. , data=sub_Training, method="class")
model_rnd_forest_prediction = predict(model_rnd_forest, sub_Testing, type = "class")
```

From Confusion Matrix we can obtain the accuracy for this model was 0.9951 for a 95% CI : **(0.9927, 0.9969)**.
Expected out-of-sample error is estimated at **0.005,a 0.5%**.

```
# Testing results
confusionMatrix(model_rnd_forest_prediction, sub_Testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    2    0    0    0
##          B    1  946    6    0    0
##          C    0    1  849    7    1
##          D    0    0    0  796    3
##          E    0    0    0    1  897
##
## Overall Statistics
##
##                Accuracy : 0.9955
##                  95% CI : (0.9932, 0.9972)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9943
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9968   0.9930   0.9900   0.9956
## Specificity            0.9994   0.9982   0.9978   0.9993   0.9998
## Pos Pred Value         0.9986   0.9927   0.9895   0.9962   0.9989
## Neg Pred Value         0.9997   0.9992   0.9985   0.9981   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1929   0.1731   0.1623   0.1829
## Detection Prevalence   0.2847   0.1943   0.1750   0.1629   0.1831
## Balanced Accuracy      0.9994   0.9975   0.9954   0.9947   0.9977
```

Conclusions.

The Random Forest fit is clearly more accurate than Decision Tree model (0.99 vs 0.74).
For testing the 20 test-cases we select the Random Forest method.
With an accuracy above 99% on our cross-validation data, we can expect that all of the submitted test cases to be correct.

Submit.

We selected the *Random Forest Model*, we used this model for 20 test cases and submit the 20 files obtained with the next code (as required in Coursera instructions).

```
#Random Forest
selected_prediction <- predict(model_rnd_forest, testing_set, type="class")
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(selected_prediction)
#this code make 20 files, one for every test case
```