Practical Machine Learning - Course Project

Antonio A. Malanda

July, 2017

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).

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. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Objective

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 preprocessing the data

Data Loading

First of all we load the required libraries:

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.3
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.2.5
```

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.2.5
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.2.3
```

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

```
## Warning: package 'rpart.plot' was built under R version 3.2.4
```

```
library(RColorBrewer)
```

```
## Warning: package 'RColorBrewer' was built under R version 3.2.3
```

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.2.4
```

```
## Rattle: A free graphical interface for data mining with R.
## Versión 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Escriba 'rattle()' para agitar, sacudir y  rotar sus datos.
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.2.5
```

```
## 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
```

```
library(knitr)
```

Now, we download the training and testing datasets from the given URLs:

```
train_Url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
test_Url <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

training_data <- read.csv(url(train_Url), na.strings=c("NA","#DIV/0",""))
testing_data <- read.csv(url(test_Url), na.strings=c("NA","#DIV/0",""))

dim(training_data)
```

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

```
dim(testing_data)
```

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

Data Cleaning

In order to clean the data we have followed the next steps:

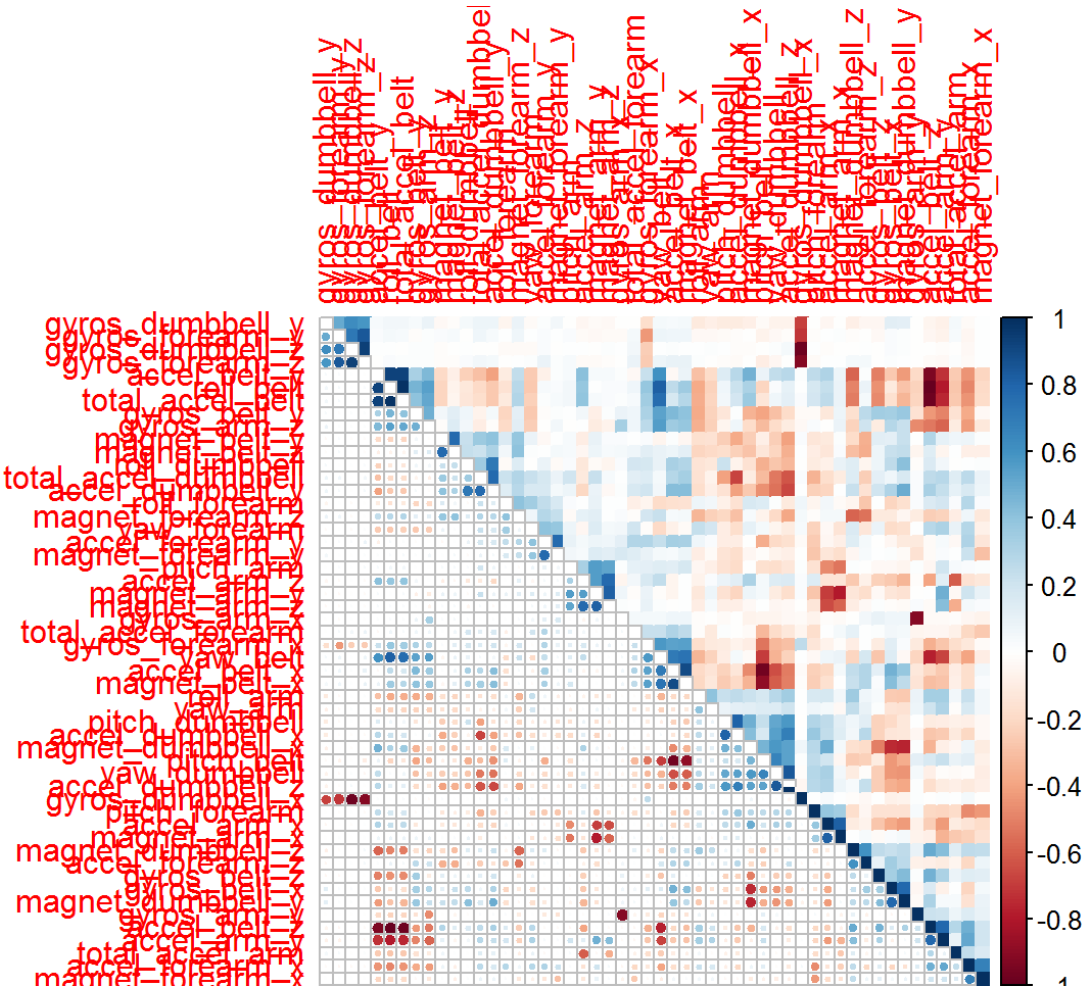
1. Remove the predictors with missing values.
2. Remove the predictors zero or near to zero variance.
3. Remove unrelevant variables (like time stamp, window variables or user name).

```
# Remove predictors that contain any missing values
cols_without_missing_values <- colSums(is.na(testing_data)) == 0
training_data <- training_data[, cols_without_missing_values]
testing_data <- testing_data[, cols_without_missing_values]

# Remove NearZeroVariance variables
nzv <- nearZeroVar(testing_data, saveMetrics=TRUE)
training_data <- training_data[,nzv$nzv==FALSE]
testing_data <- testing_data[,nzv$nzv==FALSE]

# Remove remove unrelevant variables
training_data <- training_data[, -c(1:6)]
testing_data <- testing_data[, -c(1:6)]
```

Exploratory Analysis

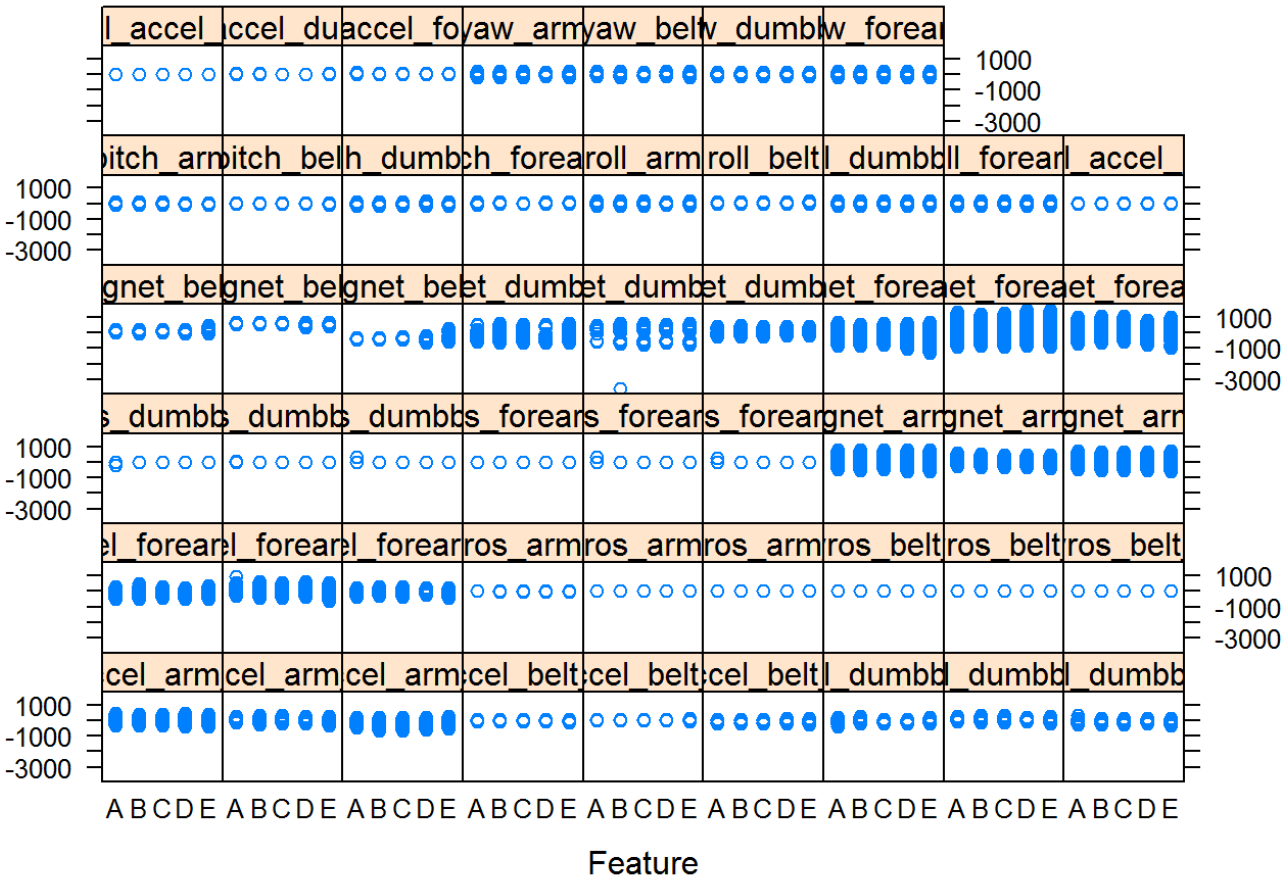
At this point, we have 52 predictor variables. The next figure show de correlation among all these predictors.

```
corrplot.mixed(cor(training_data[,-c(53)]), lower="circle", upper="color",
               tl.pos="lt", diag="n", order="hclust", hclust.method="complete")
```



In the same way, the figure below shows the relation between the 52 predictors and the outcome variable.

```
featurePlot(training_data[,-c(53)], training_data[,c(53)], "strip")
```



Data Splitting

In order to get out-of-sample errors, we split the resulting training dataset into a training set (70%) and a validation set (30%).

```
set.seed(282828)
inTrain <- createDataPartition(training_data$classe, p = 0.7, list = FALSE)
train <- training_data[inTrain, ]
valid <- training_data[-inTrain, ]
```

Prediction Algorithms

In order to predict the outcome we have used two differente algorithms:

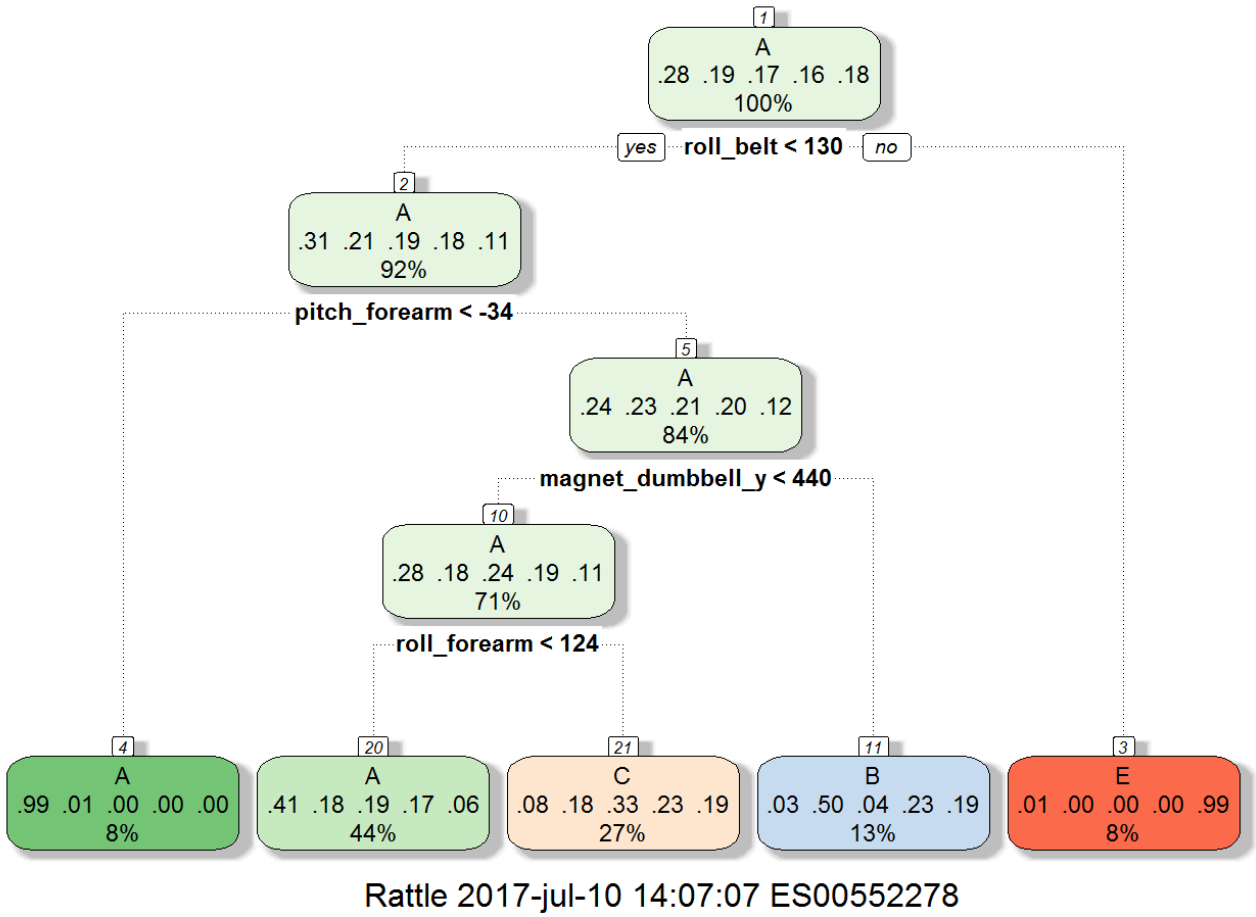
1. Decision Trees.
2. Random Forests.

Algorithm 1: Decision Trees

```
set.seed(282828)
train_control <- trainControl(method = "cv", number = 10)
mod_DT <- train(classe ~., method="rpart", data=train, trControl = train_control)
print(mod_DT, digits = 4)
```

```
## CART
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 12363, 12365, 12364, 12362, 12363, 12362, ...
## Resampling results across tuning parameters:
##
##   cp       Accuracy  Kappa    Accuracy SD  Kappa SD
##   0.03509  0.5054    0.36203  0.01792      0.03099
##   0.05883  0.4130    0.20417  0.06084      0.10350
##   0.11372  0.3152    0.04693  0.03990      0.06069
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03509.
```

```
fancyRpartPlot(mod_DT$finalModel)
```



Once we have developed the model, we validate its accuracy using the validation dataset as follows:

```
Conf_Mat_DT <- confusionMatrix(valid$classe, predict(mod_DT, newdata = valid))
Conf_Mat_DT
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1521   25  123    0    5
##          B  459  402  278    0    0
##          C  461   30  535    0    0
##          D  434  158  372    0    0
##          E  164  144  270    0  504
##
## Overall Statistics
##
##                Accuracy : 0.5033
##                  95% CI : (0.4905, 0.5162)
##     No Information Rate : 0.5164
##     P-Value [Acc > NIR] : 0.9784
##
##                   Kappa : 0.3512
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.5005  0.52964  0.33904       NA  0.99018
## Specificity            0.9462  0.85622  0.88600   0.8362  0.89249
## Pos Pred Value         0.9086  0.35294  0.52144       NA  0.46580
## Neg Pred Value         0.6395  0.92478  0.78535       NA  0.99896
## Prevalence             0.5164  0.12897  0.26814   0.0000  0.08649
## Detection Rate         0.2585  0.06831  0.09091   0.0000  0.08564
## Detection Prevalence   0.2845  0.19354  0.17434   0.1638  0.18386
## Balanced Accuracy      0.7234  0.69293  0.61252       NA  0.94133
```

Using the validation dataset we conclude that the accuracy obtained with this model is too low, so using Decision Trees to predict the outcome does not seem to be a suitable method.

Algorithm 2: Random Forests

```
set.seed(282828)
mod_RF <- train(classe ~ ., method = "rf", data = train, trControl = train_control
)
print(mod_RF, digits = 4)
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 12363, 12365, 12364, 12362, 12363, 12362, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa   Accuracy SD  Kappa SD
##    2    0.9916    0.9894  0.003114     0.003940
##   27    0.9921    0.9901  0.002192     0.002774
##   52    0.9879    0.9847  0.002621     0.003317
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
Conf_Mat_RF <- confusionMatrix(valid$classe, predict(mod_RF, newdata = valid))
Conf_Mat_RF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    3 1133    2    1    0
##          C    0    2 1022    2    0
##          D    0    1   13  949    1
##          E    0    0    0    2 1080
##
## Overall Statistics
##
##                Accuracy : 0.9954
##                  95% CI : (0.9933, 0.997)
##     No Information Rate : 0.285
##     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            0.9982   0.9974   0.9855   0.9948   0.9991
## Specificity            1.0000   0.9987   0.9992   0.9970   0.9996
## Pos Pred Value         1.0000   0.9947   0.9961   0.9844   0.9982
## Neg Pred Value         0.9993   0.9994   0.9969   0.9990   0.9998
## Prevalence             0.2850   0.1930   0.1762   0.1621   0.1837
## Detection Rate         0.2845   0.1925   0.1737   0.1613   0.1835
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9991   0.9980   0.9924   0.9959   0.9993
```

As we can see, the Random Forest based methods is much better than the Decision Tree based Method. The accuracy obtained in the validation dataset is 0.991.

Prediction on Test Dataset

Once we have selected the model, we run our model on the Test Dataset to make our predictions.

```
prediction_test <- predict(mod_RF, testing_data)
prediction_test
```

```
##  [1] 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
```