# Coursera - Practical Machine Learning - Final Project

Daniel Morales

2020-08-14

## Introduction

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, our goal will be to use data from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways: exactly according to the specification (class A), throwing the elbows to the front (class B), lifting the dumbbell only halfway (class C), lowering the dumbbell only halfway (class D) and throwing the hips to the front (class E). 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).

This data will be used to train a model that predicts which way (A, B, C, D or E) the exercise is being done, given the measurements.

## Data

The data used a courtesy of:

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13). Stuttgart, Germany: ACM SIGCHI, 2013.

The labeled data for this project are available here (this is the data provided for us to train and test our model in this project):

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The unlabeled data are available here (and the main goal of the project is to correctly label these observations with our trained model):

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

### Loading Libraries and Importing Data

We start loading packages and reading data directly from the links provided. Note that the first row contains headers and the first column contains row numbers.

```r
library(caret)
library(corrplot)
library(e1071)
library(rattle)
```

```r
labeled <- read.csv(
  url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"),
  header = TRUE, row.names = 1
)
unlabeled <- read.csv(
  url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"),
  header = TRUE, row.names = 1
)
dim(labeled)
```

```
## [1] 19622   159
```

```r
dim(unlabeled)
```

```
## [1]  20 159
```

### Cleaning Data

After importing the datasets with 160 columns each (159 if we do not count the column with row numbers), matching every column except for the last one, which is `classe`, our target variable for the labeled set, and `problem_id` for the unlabeled set, an ID for submitting the predictions. We also have 19622 rows (observations) for the labeled set and 20 rows for the unlabeled set.

**Missing Values** A simple `str` applied to the data (due to number of variables, this step is not shown) shows us that there are variables (columns) with most values as `NA` or empty `""`. These variables will be disconsidered.

```r
labeled_clean <- labeled[, colSums(is.na(labeled) | labeled == "") < 0.95 * nrow(labeled)]
unlabeled_clean <- unlabeled[, colSums(is.na(unlabeled)) < 0.95 * nrow(unlabeled)]
dim(labeled_clean)
```

```
## [1] 19622    59
```

```r
dim(unlabeled_clean)
```

```
## [1] 20 59
```

**Identification Variables** The first 6 columns will also be disconsidered, as they are identification variables, and our goal is to predict based entirely on the movements captured by the accelerometers.

```r
labeled_clean <- labeled_clean[, -c(1:6)]
unlabeled_clean <- unlabeled_clean[, -c(1:6)]
dim(labeled_clean)
```

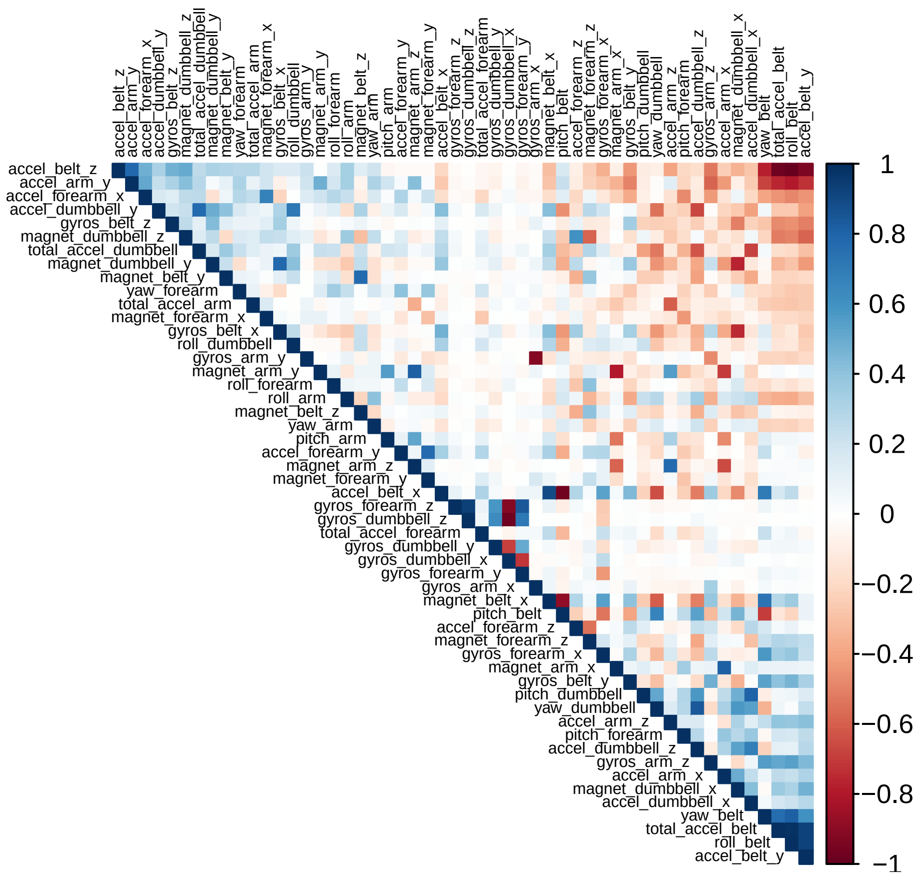
```
## [1] 19622    53
```

```r
dim(unlabeled_clean)
```

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

**High Correlations** Let us now analyze the correlations between the observations.

```r
corrplot(cor(labeled_clean[, -53]), order = "FPC", method = "color",
         type = "upper", tl.cex = 0.5, tl.col = rgb(0, 0, 0))
```



Being the correlations quite few, we will refrain from doing a more profound analysis on the correlations, like PCA (Principal Component Analysis) and just proceed to the next steps.

### Partition Data

We divide our labeled data in training and testing sets at the ratio of 80% for training and 20% for testing, followed by a 5-fold cross validation. Both values have empirically shown performance that balance well bias and variance.

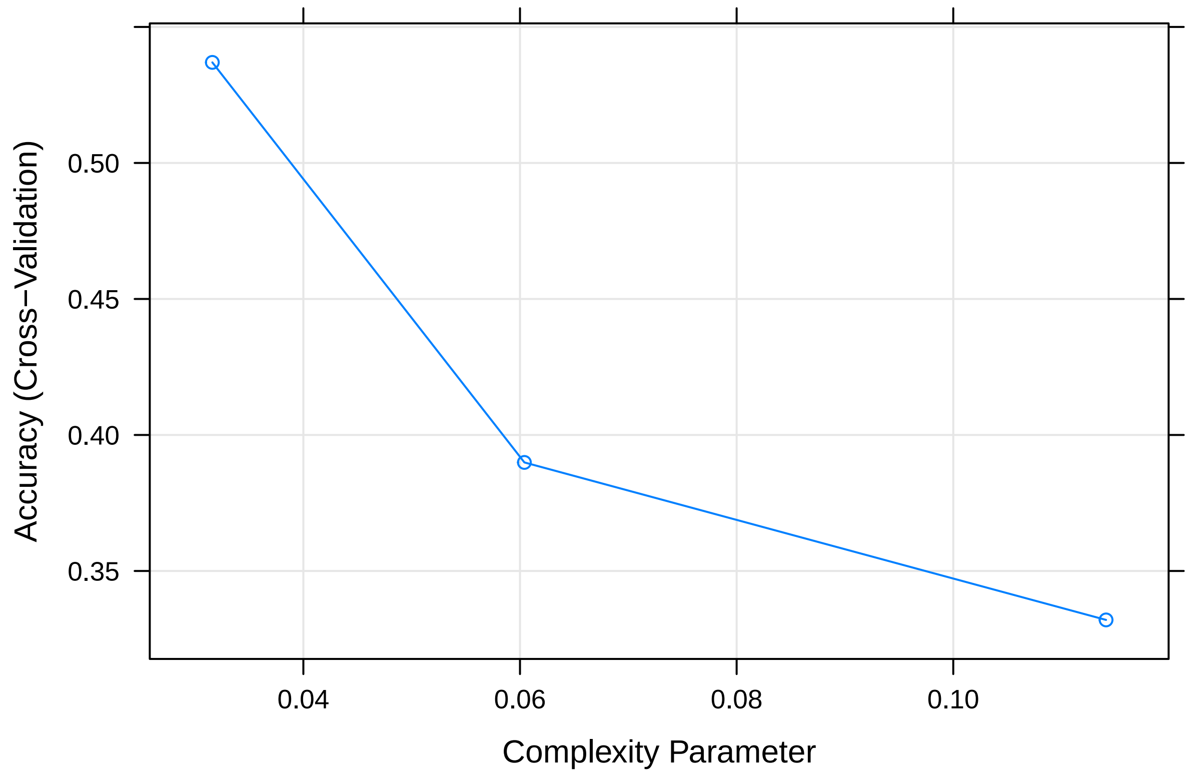
```r
set.seed(8226)
in_train <- createDataPartition(labeled_clean$classe, p = 0.8, list = FALSE)
training <- labeled_clean[in_train, ]
testing <- labeled_clean[-in_train, ]
tr_control <- trainControl(method = "cv", number = 5)
```

## Prediction Algorithms

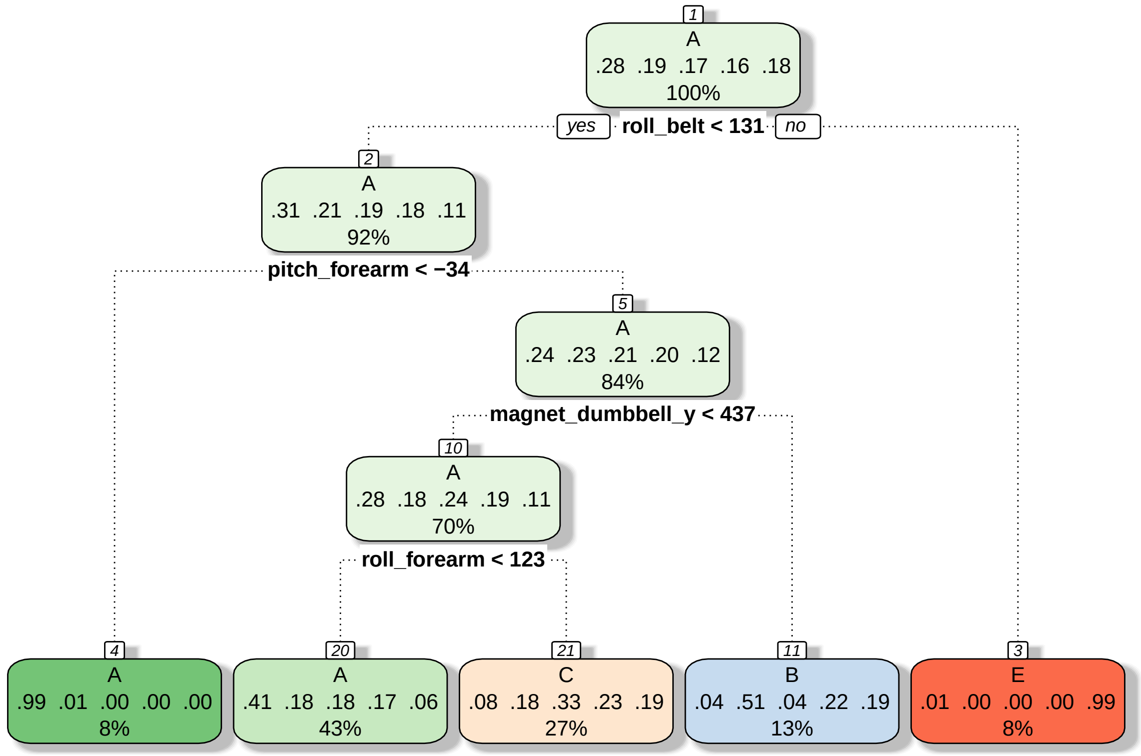
We will be using three different algorithms for our multiclass classification: a simple classification tree for reference, random forest and GBM (Gradient Boosting Machine).

### Classification Tree

```r
rpart_fit <- train(classe ~ ., data = training,
                   method = "rpart", trControl = tr_control)
plot(rpart_fit)
```



```r
fancyRpartPlot(rpart_fit$finalModel, sub = NULL)
```



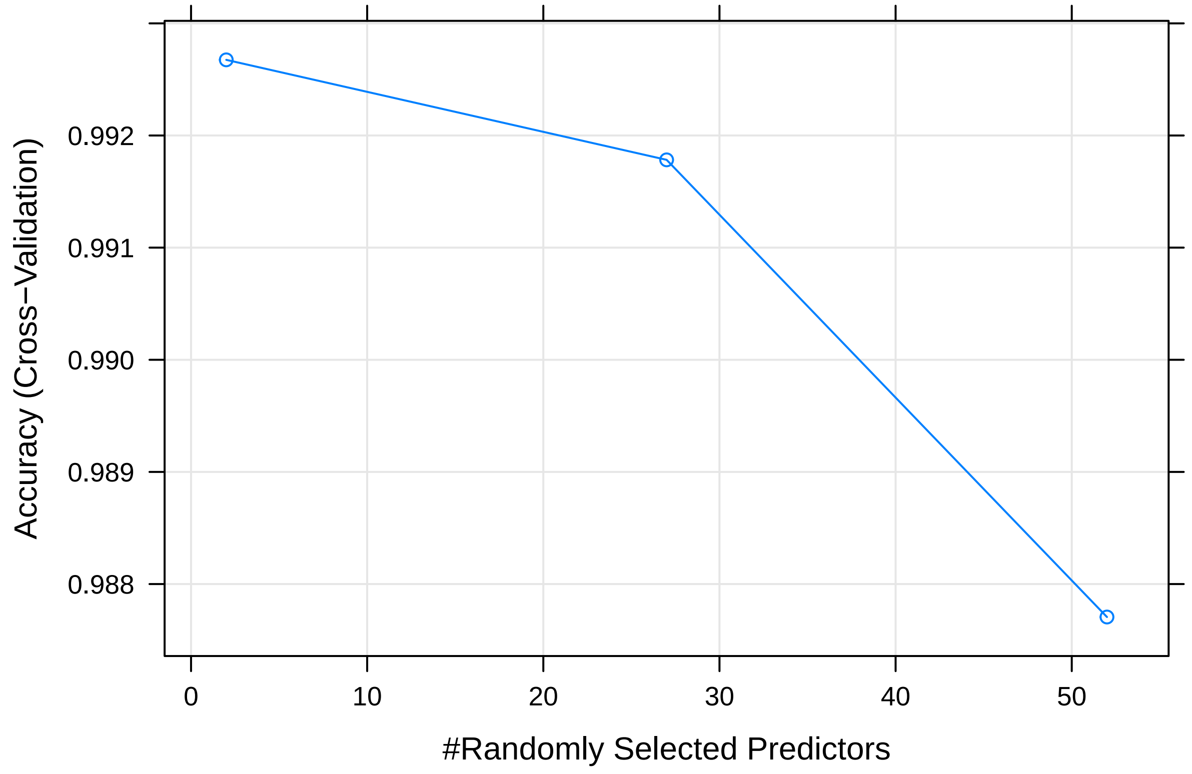
```r
rpart_predict <- predict(rpart_fit, newdata = testing)
rpart_conf_mat <- confusionMatrix(rpart_predict, factor(testing$classe))
rpart_conf_mat$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1023  321  329  279  113
##          B   16  250   22  113  103
##          C   75  188  333  251  168
##          D    0    0    0    0    0
##          E    2    0    0    0  337
```

Note that this simple classification tree never predicts class D!

### Random Forest

```r
rf_fit <- train(classe ~ ., data = training,
                method = "rf", trControl = tr_control)
plot(rf_fit)
```

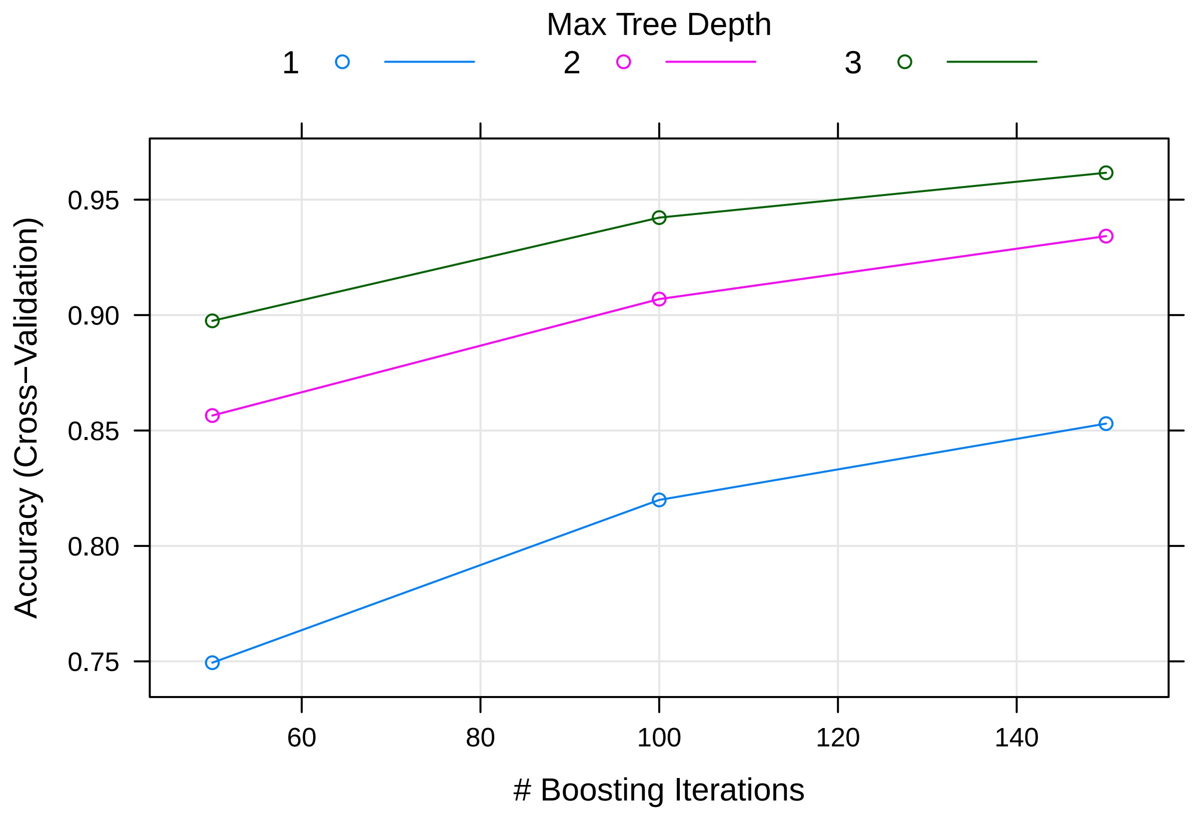


```r
rf_predict <- predict(rf_fit, newdata = testing)
rf_conf_mat <- confusionMatrix(rf_predict, factor(testing$classe))
rf_conf_mat$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    1    0    0    0
##          B    0  758   10    0    0
##          C    0    0  672    8    0
##          D    0    0    2  635    4
##          E    0    0    0    0  717
```

### Gradient Boosting Machine

```r
gbm_fit <- train(classe ~ ., data = training,
                method = "gbm", trControl = tr_control, verbose = FALSE)
plot(gbm_fit)
```



```r
gbm_predict <- predict(gbm_fit, newdata = testing)
gbm_conf_mat <- confusionMatrix(gbm_predict, factor(testing$classe))
gbm_conf_mat$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1100   20    0    0    3
##          B   11  724   22    0    6
##          C    4   15  649   20    5
##          D    1    0   13  619   14
##          E    0    0    0    4  693
```

## Prediction

Since we have quite balanced classes on our target variable `classe`, we can use accuracy on the testing set as a measure to select a prediction algorithm. Having:

| Algorithm | Accuracy |
|---|---|
| Classification Tree | 0.4952842 |
| Random Forest | 0.9936273 |
| GBM | 0.9648228 |

So the selected algorithm for prediction is random forest, that gives the following predictions for our unlabeled dataset:

```r
predict(rf_fit, newdata = unlabeled)
```

```
##  [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
```

Note that the second best algorithm (according to accuracy), GBM, produces the same predictions:

```r
predict(gbm_fit, newdata = unlabeled)
```

```
##  [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
```

Due to the high accuracy from the random forests we can keep ourselves within these somewhat simple algorithms and within the scope of the course.