Predicting Barbell Lifts From Accelerometers

Isabelle Valette

17 Mar 2015

Executive Summary

One thing that people regularly do with data from sensors, like accelerometers, is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, we used data from accelerometers on the belt, forearm, arm, and dumbell from 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The goal of this project was to build a machine learning algorithm that predicted with high accuracy the manner in which the participants did the exercise. This was acheived with a Random Forest. This algorithm predicted correctly all of 20 test cases.

Data Loading Into R

First, we load the dataset into r and prepare and transform the data to create an analytical data set.

```r
# Downloading the datasets
urlTraining <-
"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
urlTest <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv"
download.file(urlTraining, "pml-training.csv", method="curl")
download.file(urlTest, "pml-testing.csv", method="curl")

## Reading data into R
train <- read.csv("./pml-training.csv", sep=",", header=T, na.strings =
c("NA", ""), stringsAsFactors=FALSE)
test <- read.csv("./pml-testing.csv", sep=",", header=T, na.strings =
c("NA", ""), stringsAsFactors=FALSE)
dim(train); dim(test)
```

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

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

Data Transformation and Preprocessing Strategy

The data is quite "dirty" with a lot of missing value and zero covariates. We choose the following strategy to improve the data quality:

1. There are a lot of missing values in 96 columns. We choose to delete them from the datasets.
2. The variable "new_window" has almost no variance. We choose to remove it from the analytical dataset.
3. A few variables have no explainatory power in order to predict our outcome "classe" (X, user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp, num_window). We choose to remove them from the dataset.
4. A lot of numeric variables are listed as integers. We convert them to numeric values.
5. We convert the outcome "classe" to a factor variable
6. There is a large number of predictors. We will run a correlation analysis to check for possible multicollinearity among predictors.

```
## Calculate the total number of missing values per column and removing them.
allmisscols <- sapply(train, function(x) sum(is.na(x)));# allmisscols
colswithallmiss <-names(allmisscols[allmisscols>0])
trainingTrs <- train[ , -which(names(train) %in% colswithallmiss)]

# Removing variables with zero covariates
library(caret)

## Warning: package 'caret' was built under R version 3.1.2

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

nsv <- nearZeroVar(trainingTrs,saveMetrics=TRUE)
trainingTrs2 <- trainingTrs[ , -6]

# Removing variables with no predictive power
trainingTrs3 <- trainingTrs2[ , -c(1:6)]

# Transforming integer to numeric variables
trainingTrs3[, c(4,8:13,17,21:26,30,34:38,43,47:50)]  <-
as.numeric(unlist(trainingTrs3[,
c(4,8:13,17,21:26,30,34:38,43,47:50)]))
# Transforming the outcome "classe" to factor variable
trainingTrs3[, 53]  <- as.factor(trainingTrs3[, 53])
dim(trainingTrs3)

## [1] 19622    53

# Run a preliminary Correlation Analysis: checking for multicollinearity
trainingCor <- findCorrelation(cor(trainingTrs3[,-53]), cutoff= 0.60);
trainingCor
```

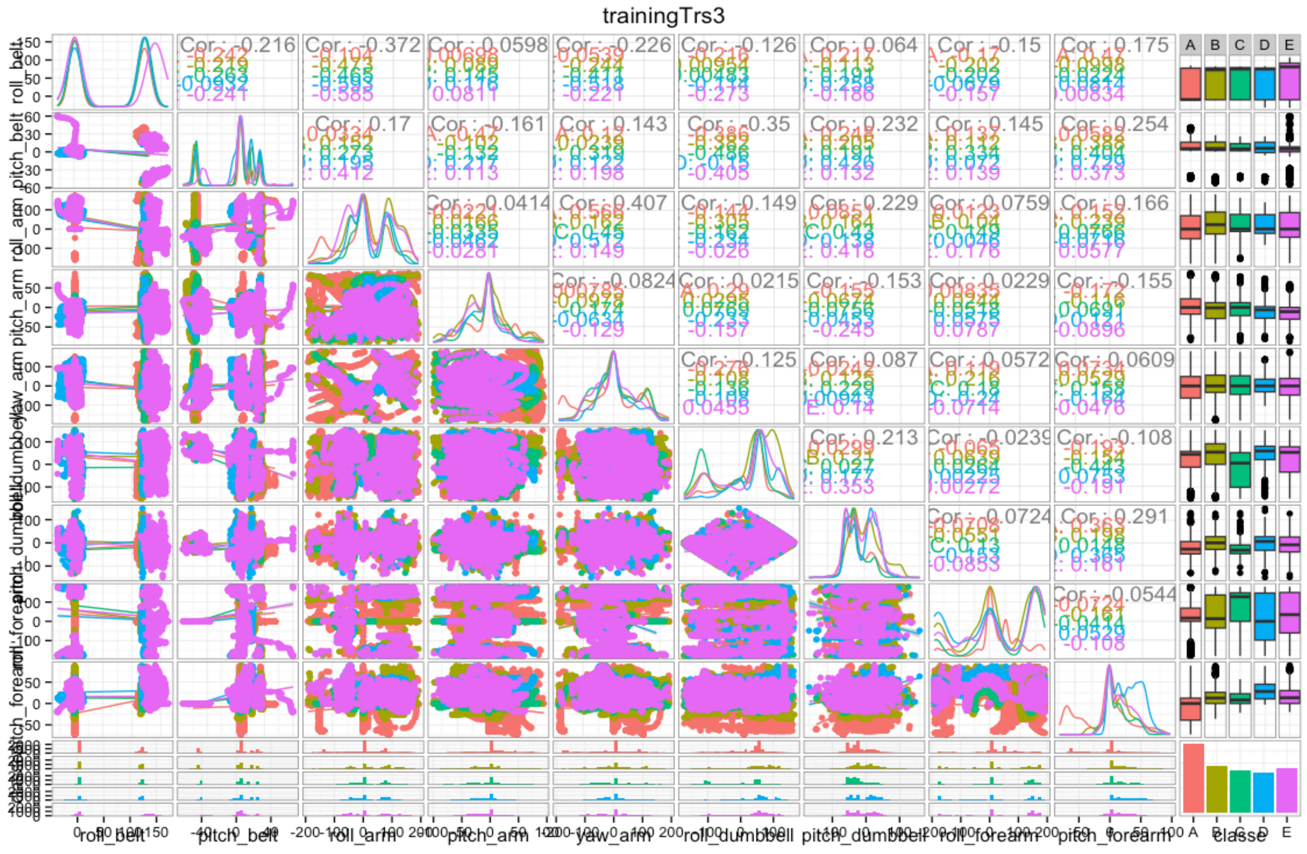
```
##  [1] 10  1  9 22  4  3 36  8  2 11 37 35 38 30 47 21 34 39 23 25 12
48 19
## [24] 32 46 45 31
```

Conclusion: The analytical dataset is now ready. However, multicollinearity is a concern and has not been delt with yet. We will deal with this by running a PCA (Principal Component Analysis) before training and tuning our models.

Exploratory Data Analysis - EDA

We get a feel of some of the data with an EDA (Exploratory Data Analysis.

```
# Running EDA
require(GGally)

## Loading required package: GGally

## Warning: package 'GGally' was built under R version 3.1.2

pairsGG=trainingTrs3[,c(1,2,14,15,16,27,28,40,41,53)]
ggpairs(pairsGG,lower =
list(continuous="smooth",theme_set(theme_bw())),title="trainingTrs3",co
lour = "classe", params=list(corSize=08))
```



Building Machine Learning Models

Data Partitioning

We first split the data into a training and a testing data set.The training set will contain 70% of the original data set while the testing set will have 30%.

Cross-Validation

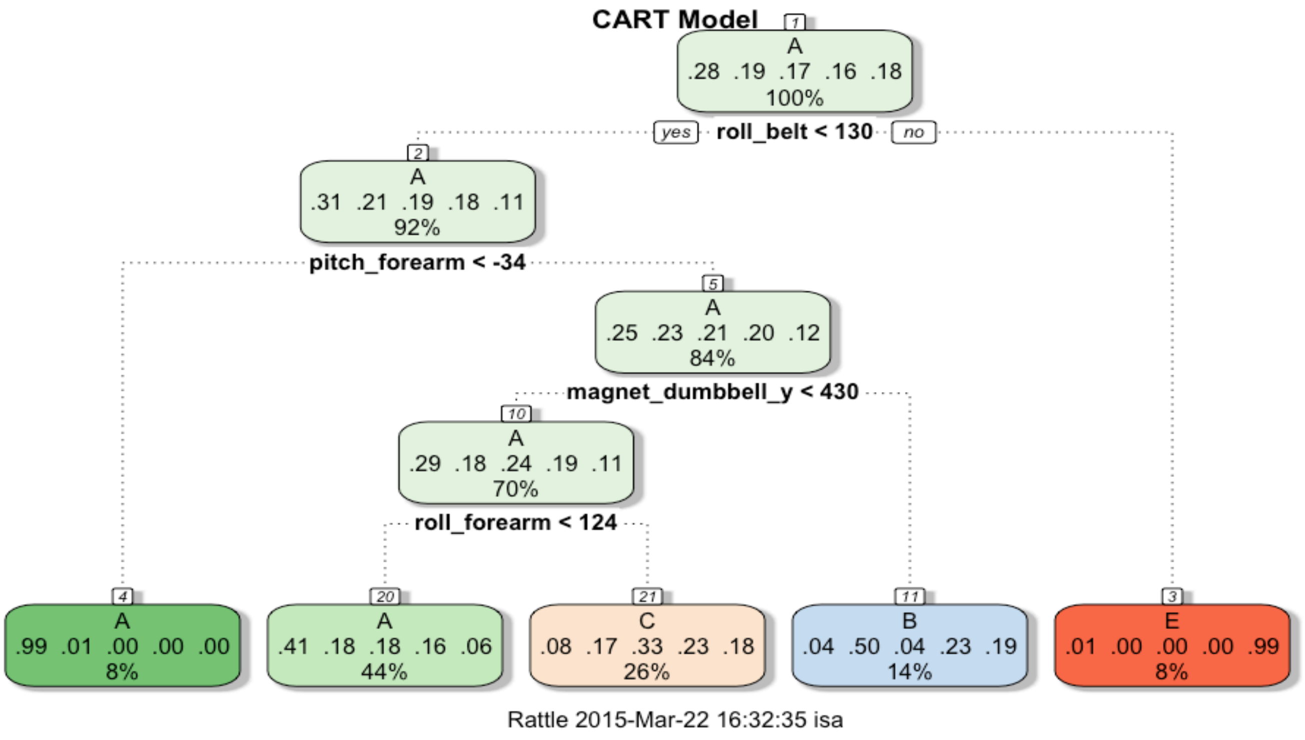
Cross-validation is a validation technique for assessing how the results of a statistical model like a random forest will generalize to an independent data set not used in training. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. In this case, we used a 10 k-folds crossvalidation repeated 5 times. We set up this step as part of the parameters setting provided by the "caret"" package in R.

Model Training and Tuning

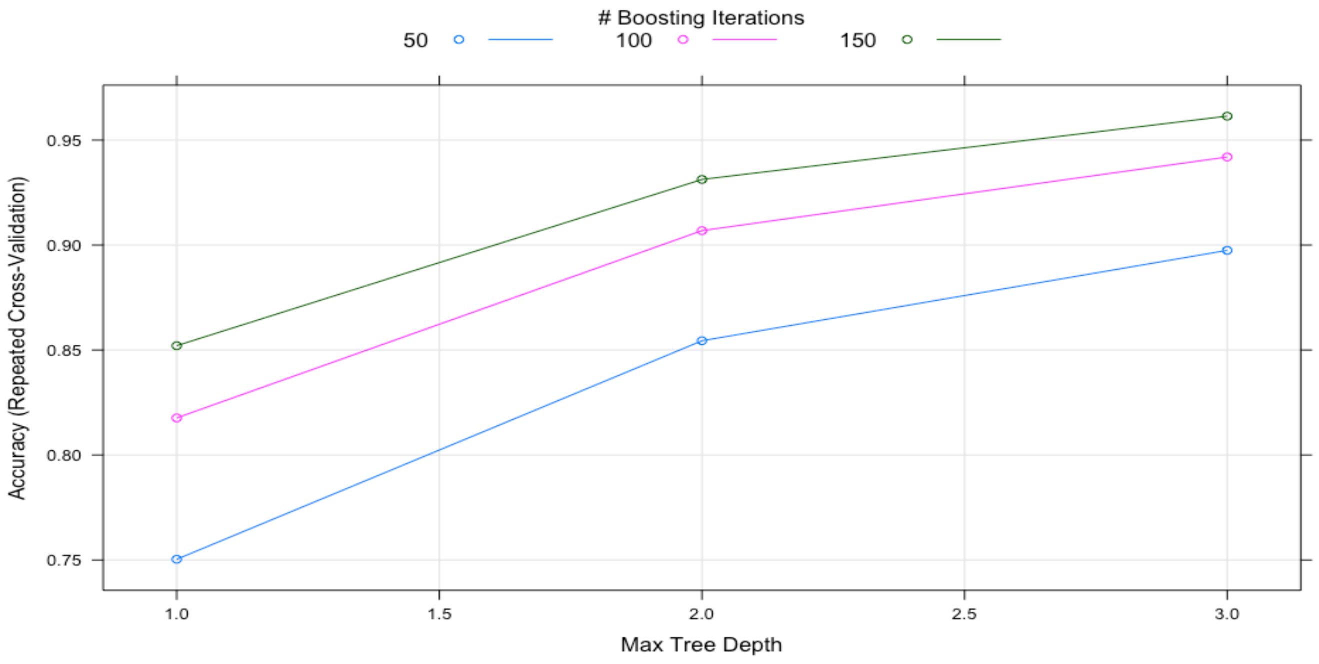
The following algorithms will be applied to the training data set: CART Classification Trees, Random Forest and Stochastic Gradient Boosting. These algorithms are covered in the Practical Machine Learning class (Data Science Specialization) on Coursera.

```
# Splitting the dataset into a training and test data set.
library(caret);
set.seed(123422)
inTrain <- createDataPartition(y=trainingTrs3$class, p=0.70,
list=FALSE)
training <- trainingTrs3[inTrain,]; dim(training)
testing <- trainingTrs3[-inTrain,]; dim(testing)

# Setting the parameters for model preprocessing and tuning from the
caret package:
fitControl <- trainControl(## 10-fold Crossvalidation
                           method = "repeatedcv",
                           number = 10,
                           ## repeated ten times
                           repeats = 5,
                           verboseIter=FALSE ,
                           # PCA Preprocessing
                           preProcOptions="pca",
                           # With parallel backend
                           allowParallel=TRUE)

# Model Training
CART <- train(classe ~ ., method="rpart",data=training, cp=0.1)

## Loading required package: rpart
```

```r
RF <- train(classe ~ ., data = training, method = "rf", trControl=
fitControl)

## Loading required package: randomForest

## Warning: package 'randomForest' was built under R version 3.1.2

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

Boosted <- train(classe ~ ., data = training, method = "gbm",
trControl= fitControl)

## Loading required package: gbm

## Warning: package 'gbm' was built under R version 3.1.3

## Loading required package: survival
## Loading required package: splines
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
##
## Loading required package: parallel
## Loaded gbm 2.1.1
## Loading required package: plyr

# Visualise the CART and Boosted Tree Models
library(rattle); library(rpart.plot)

## Warning: package 'rattle' was built under R version 3.1.2

## 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.

## Warning: package 'rpart.plot' was built under R version 3.1.2

fancyRpartPlot(CART$finalModel, main="CART Model")
```



```
plot(Boosted);
```



```
# Measuring Model Accuracy
Model <- c("CART", "Random Forest","Boosted Tree")
Accuracy <-
c(round(max(CART$results$Accuracy),4)*100,round(max(RF$results$Accuracy
),4)*100,
              round(max(Boosted$results$Accuracy),4)*100 )
Performance <- cbind(Model,Accuracy); Performance
```

```
##      Model           Accuracy
## [1,] "CART"          "49.51"
## [2,] "Random Forest" "99.2"
## [3,] "Boosted Tree"  "96.14"
```

```
# Printing the Confusion Matrix
pred_CART <- predict(CART, testing); CM_CART <- table(pred_CART,
testing$classe)
pred_RF <- predict(RF, testing); CM_RF <- table(pred_RF,
testing$classe)
pred_Boosted <- predict(Boosted, testing); CM_Boosted <-
table(pred_Boosted, testing$classe);
print("CART"); CM_CART; print("Random Forest"); CM_RF; print("Boosted
Tree"); CM_Boosted
```

```
## [1] "CART"
```

```
##
## pred_CART    A    B    C    D    E
##         A 1523  459  481  450  158
##         B   47  392   36  166  151
##         C  101  288  509  348  292
##         D    0    0    0    0    0
##         E    3    0    0    0  481
```

```
## [1] "Random Forest"
```

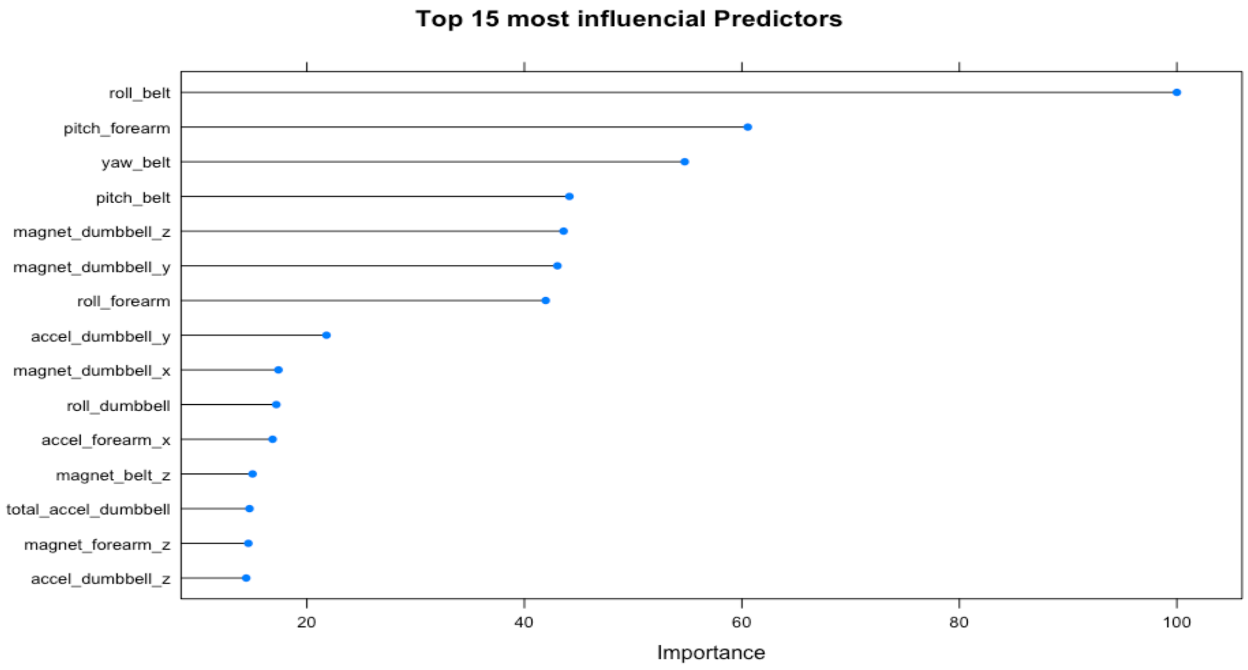
```
##
## pred_RF    A    B    C    D    E
##       A 1670    7    0    0    0
##       B    4 1130    3    0    0
##       C    0    2 1023    7    0
##       D    0    0    0  956    8
##       E    0    0    0    1 1074
```

```
## [1] "Boosted Tree"
```

```
##
## pred_Boosted    A    B    C    D    E
##            A 1646   40    0    1    2
##            B   24 1076   39    6   14
##            C    2   23  979   20    6
##            D    2    0    6  927   17
##            E    0    0    2   10 1043
```

```
# Most significant variables in the model
VarImportance <- varImp(RF)
# Plot the top 15 predictors
```

```r
plot(VarImportance, main = "Top 15 most influencial Predictors", top =
15)
```



Conclusion: We can conclude that the Random Forest model is the model with the best predictive power for accurately assessing the barbell lifts performance. Based on the results from accuracy (more the 99%), the Confusion Matrix (almost perfect classification) and the list of the top 15 predictors, we can confirm that the model has a low out of sample error. We therefore feel confident that it is a good model to use to predict the 20 cases from the "test" dataset.

Making Predictions

In this section we predict the classe category for the 20 test cases from the test dataset. All these predictions were submited and were correct.

```r
predictions <- predict(RF, newdata=test); predictions

##  [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
```