Exercise Prediction Model

Felipe Frazatto

Introduction

This project aims at building and validating a machine learning model using random forest as its method. The objective is to predict how a series of exercises were performed, classifying them as "A", "B", "C", "D" or "E".

The data set comes a Puc Rio, Brazil, study, which is available at Groupware. The data is composed of a series of accelerometers measurements positioned on different body positions.

The designed random forest model had an accuracy of 98% and predicted the following sequence for the test set quiz:

B A C A A E D B A A B C B A E E A B B B

Data Processing

Setup

Load necessary libraries

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(caret)
```

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

```
library(rpart)
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
```

```
##     margin
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

Load data from provided links.

```
#Training and Test Data urls
urlTrain <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
urlTest  <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

#Download
training_raw <- read.csv(urlTrain)
testing_raw <- read.csv(urlTest)

#Backup
training <- training_raw
testing <- testing_raw
```

Set seed for Reproducibility.

```
set.seed(1234)
```

Cleaning

Explore data.

```
str(training)
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : chr  "carlitos" "carlitos" "carlitos" "carlitos" ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 1
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 484
##  $ cvtd_timestamp          : chr  "05/12/2011 11:23" "05/12/2011 11:23" "05/12/2011 11:23" "05/12/20
##  $ new_window              : chr  "no" "no" "no" "no" ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : chr  "" "" "" "" ...
##  $ kurtosis_picth_belt     : chr  "" "" "" "" ...
##  $ kurtosis_yaw_belt       : chr  "" "" "" "" ...
##  $ skewness_roll_belt      : chr  "" "" "" "" ...
##  $ skewness_roll_belt.1    : chr  "" "" "" "" ...
##  $ skewness_yaw_belt       : chr  "" "" "" "" ...
##  $ max_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt            : chr  "" "" "" "" ...
##  $ min_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_belt            : chr  "" "" "" "" ...
##  $ amplitude_roll_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt    : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt      : chr  "" "" "" "" ...
```

```
##  $ var_total_accel_belt    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_belt        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_belt_x            : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y            : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z            : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...
##  $ accel_belt_x            : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y            : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z            : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x           : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y           : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z           : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm               : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm                 : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm         : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ var_accel_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_arm_x             : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y             : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...
##  $ gyros_arm_z             : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x             : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y             : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z             : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
##  $ magnet_arm_x            : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ magnet_arm_y            : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z            : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ kurtosis_roll_arm       : chr  "" "" "" "" ...
##  $ kurtosis_picth_arm      : chr  "" "" "" "" ...
##  $ kurtosis_yaw_arm        : chr  "" "" "" "" ...
##  $ skewness_roll_arm       : chr  "" "" "" "" ...
##  $ skewness_pitch_arm      : chr  "" "" "" "" ...
##  $ skewness_yaw_arm        : chr  "" "" "" "" ...
##  $ max_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
```

```
##  $ amplitude_roll_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm       : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ roll_dumbbell           : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell          : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell            : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ kurtosis_roll_dumbbell  : chr  "" "" "" "" ...
##  $ kurtosis_picth_dumbbell : chr  "" "" "" "" ...
##  $ kurtosis_yaw_dumbbell   : chr  "" "" "" "" ...
##  $ skewness_roll_dumbbell  : chr  "" "" "" "" ...
##  $ skewness_pitch_dumbbell : chr  "" "" "" "" ...
##  $ skewness_yaw_dumbbell   : chr  "" "" "" "" ...
##  $ max_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_dumbbell        : chr  "" "" "" "" ...
##  $ min_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell        : chr  "" "" "" "" ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##    [list output truncated]
```

First 7 columns are identification and time variables, which will not be necessary for the ML model, so they can be dropped.

```
drop1 <- c(1:7)

cTraining <- select(training, -drop1)
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(drop1)` instead of `drop1` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

Search and remove any case of near zero variance, as these kind of variables do not significantly impact the prediction model.

```
drop2 <- nearZeroVar(cTraining)

cTraining <- cTraining[, -drop2]
```

Before looking for complete cases, check for columns with high number of NAs, this could potentially exclude too many observations. Look for the columns with more than 50% of its values as NAs. By taking the mean of each column with the function is.na returns the percentage of missing values.

```
#Filter threshold
fth_na <- 0.5

#Calculate proportion of NAs
drop3 <- apply(cTraining, 2, function(x) {mean(is.na(x))})

#Filter
cTraining <- cTraining[, drop3 < fth_na]
```

Get observations only with complete cases, i.e. no missing values (NA) in any column.

```
drop4 <- complete.cases(cTraining)

cTraining <- cTraining[drop4, ]
```

Finally, create training and testing data sets. Partition **training** data set into two: trainingSet (80%) and testingSet (20%) with respect to **classe** column (outcome).

```
#Data partiton
inTrain  <- createDataPartition(cTraining$classe, p = 0.8, list = FALSE)

trainingSet <- cTraining[inTrain, ]
testingSet  <- cTraining[-inTrain, ]
```

Model

Choosing Variables

Calculate correlations between variables, excluding **classe**, and get variables with high correlation, *i.e.* above 75%.

```
#Filter Threshold
fth_cr <- 0.75

cValue <- cor(trainingSet[, -53])

c_high <- findCorrelation(cValue, cutoff = fth_cr)

cnames <- c(names(trainingSet[,c_high]), "classe")

fTrain <- select(trainingSet, as.factor(cnames))
```

Model Building

This project proposes 2 models* for classification they are: Random Forest ("rf") and Linear Discriminant Analysis. Both are trained with with a 10-fold cross validation.

```
set.seed(1234)

#Random Forest
rfModel <- train(classe ~ ., data = fTrain, method = "rf", number = 10)
rfPredict <- predict(rfModel, testingSet)
rfAcc <- confusionMatrix(rfPredict, as.factor(testingSet$classe))$overall[1]


#Linear Discriminant Analysis
ldaModel <- train(classe ~ ., data = fTrain, method = "lda", number = 10)
ldaPredict <- predict(ldaModel, testingSet)
ldaAcc <- confusionMatrix(ldaPredict, as.factor(testingSet$classe))$overall[1]
```

* A priori, this project's intent was to test 4 models: Random Forest("rf"), Stochastic Gradient Boosting, Linear Discriminant Analysis ("lda") and a staked model with all previous 3 as regressors and using random forest as method, compare all 4 models and use the best one for prediction. However, my computer did not have the processing capabilities to execute it, only managing *rf* and *lda*.

Results

As noted before, the computation behind machine learning can be very demanding depending on the number of observations and regressors, one way to reduce the high computational power needed to train the model

is to use Principal Components Analysis (PCA) to reduce the regressors to only a few, highly impactful, components.

Between the two methods, random forest and Linear Discriminant Analysis, the first showed a higher accuracy (0.9824114) than the later (0.512363), so random forest is the best method for this assignment.

```
res <- predict(rfModel, newdata = testing)
```

Conclusion

After cleaning the data and reducing the number of variables from 160 to 53, an correlation analysis was done, reducing the regressors to 13. A Principal Components Analysis could be used to further reduce it.

Training and comparing the two methods, Random Forest and Linear Discriminant, Analysis, the first had a better accuracy (0.9824114), so been chosen to predict the required assignment observation, resulting in the following sequence:

B, A, C, A, A, E, D, B, A, A, B, C, B, A, E, E, A, B, B, B