Machine Learning Predictive Model from Monitored Exercise

Marcos Medeiros

1/28/2022

Synopsis

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. 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, I will use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways to predict the manner in which participants will perform a barbell lift. More information is available from the website here: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har

Possible Outcomes

The outcome variable is `classe`, a factor variable with 5 levels of precision in a set of 10 repetitions of Unilateral Dumbbell Curl:

`class A` exactly according to the specification;

`class B` throwing the elbows to the front;

`class C` lifting the dumbbell only halfway;

`class D` lowering the dumbbell only halfway;

`class E` throwing the hips to the front.

Data Loading and preparing analisys

```
library(lattice)
library(ggplot2)
library(caret)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```r
library(rpart)
library(rpart.plot)
library(rattle)
```

```
## Loading required package: tibble

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
trainURL <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testURL <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

training <- read.csv(url(trainURL))
testing <- read.csv(url(testURL))


## Creating a partition

label <- createDataPartition(training$classe, p = 0.7, list = FALSE)
train <- training[label, ]
test <- training[-label, ]


## Filtering data
### Excluding variables with nearly zero variance

NZV <- nearZeroVar(train)
train <- train[ ,-NZV]
test <- test[ ,-NZV]

### Excluding variables with more than 90% NAs

label <- apply(train, 2, function(x) mean(is.na(x))) > 0.90
train <- train[, -which(label, label == FALSE)]
test <- test[, -which(label, label == FALSE)]

### Excluding identification variables

train <- train[ , -(1:5)]
test <- test[ , -(1:5)]

dim(train)
```

```
## [1] 13737    54
```

```r
dim(test)
```

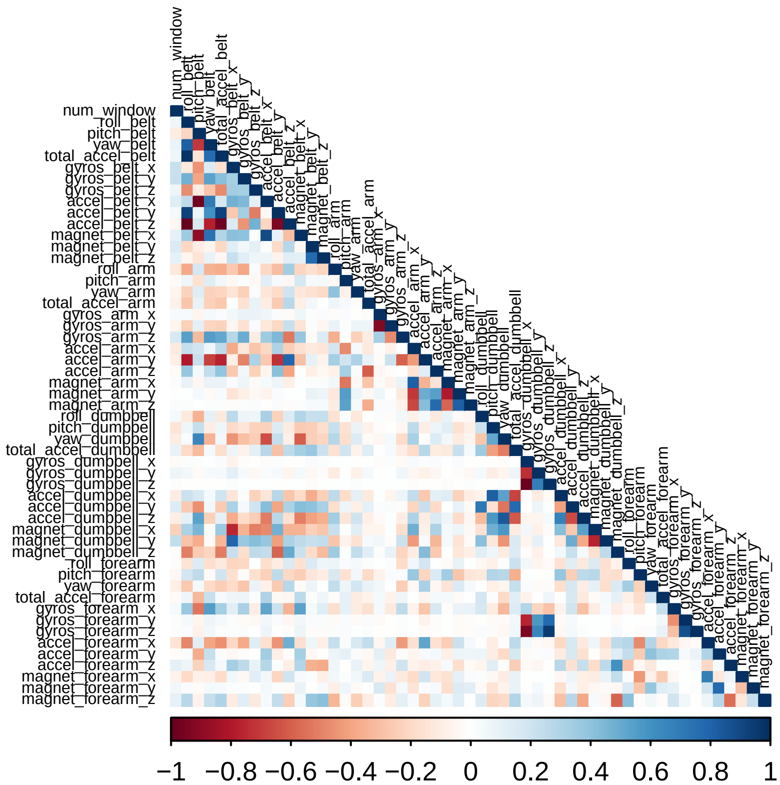
```
## [1] 5885   54
```

We reduced the dataset from 160 to 54 variables.

Exploratory Analysis

```r
## Making a correlation plot to look the dependence intensity

depend <- cor(train[,-54])
corrplot(depend, method = "color", type = "lower", tl.cex = 0.5, tl.col = rgb(0,0,0))
```



Predictive Model Selection

To choose what method provides the best accuracy in the predictive model, we will perform Random Forest, Generalized Boosted Model and Decision Tree. A confusion matrix at the end of each model will help to compare them.

Random Forest

```r
set.seed(14518)
control <- trainControl(method = "cv", number = 4, verboseIter=FALSE)
modelRF <- train(classe ~ ., data = train, method = "rf", trControl = control)
modelRF$finalModel
```

```
##
```

```
## Call:
##  randomForest(x = x, y = y, mtry = min(param$mtry, ncol(x)))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.2%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3904    1    0    0    1 0.0005120328
## B    3 2653    2    0    0 0.0018811136
## C    0    6 2390    0    0 0.0025041736
## D    0    0    8 2243    1 0.0039964476
## E    0    2    0    4 2519 0.0023762376
```

```
predictRF <- predict(modelRF, test)
confMatRF <- confusionMatrix(predictRF, as.factor(test$classe))
confMatRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    2    0    0    0
##          B    0 1136    4    0    2
##          C    0    1 1022    5    0
##          D    0    0    0  959    4
##          E    0    0    0    0 1076
##
## Overall Statistics
##
##                Accuracy : 0.9969
##                  95% CI : (0.9952, 0.9982)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9961
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9974   0.9961   0.9948   0.9945
## Specificity            0.9995   0.9987   0.9988   0.9992   1.0000
## Pos Pred Value         0.9988   0.9947   0.9942   0.9958   1.0000
## Neg Pred Value         1.0000   0.9994   0.9992   0.9990   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1930   0.1737   0.1630   0.1828
## Detection Prevalence   0.2848   0.1941   0.1747   0.1636   0.1828
## Balanced Accuracy      0.9998   0.9981   0.9974   0.9970   0.9972
```

Generalized Boosted Model

```r
set.seed(14518)
control <- trainControl(method = "repeatedcv", number = 5, repeats = 1, verboseIter = FALSE)
modelGBM <- train(classe ~ ., data = train, trControl = control, method = "gbm", verbose = FALSE)
modelGBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 53 had non-zero influence.
```

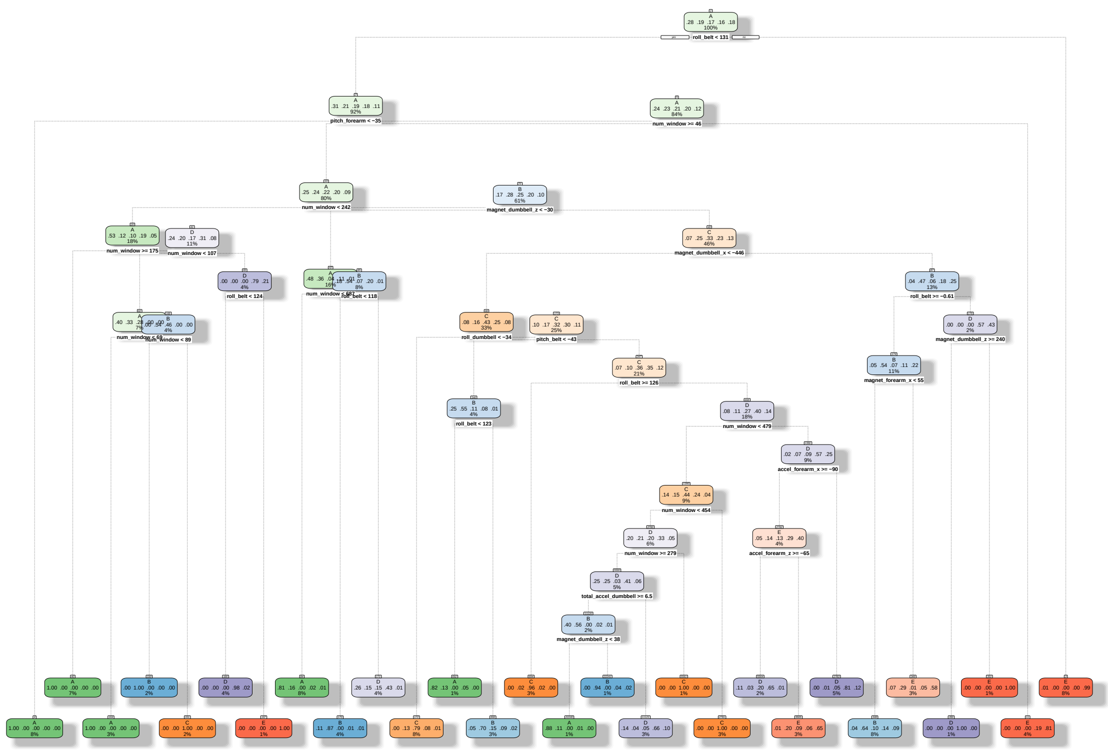
```r
predictGBM <- predict(modelGBM, test)
confMatGBM <- confusionMatrix(predictGBM, as.factor(test$classe))
confMatGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1668    9    0    0    0
##          B    5 1107   10    9    3
##          C    0   21 1013   16    2
##          D    1    2    3  938    6
##          E    0    0    0    1 1071
##
## Overall Statistics
##
##                Accuracy : 0.985
##                  95% CI : (0.9816, 0.988)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9811
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9719   0.9873   0.9730   0.9898
## Specificity            0.9979   0.9943   0.9920   0.9976   0.9998
## Pos Pred Value         0.9946   0.9762   0.9629   0.9874   0.9991
## Neg Pred Value         0.9986   0.9933   0.9973   0.9947   0.9977
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2834   0.1881   0.1721   0.1594   0.1820
## Detection Prevalence   0.2850   0.1927   0.1788   0.1614   0.1822
## Balanced Accuracy      0.9971   0.9831   0.9897   0.9853   0.9948
```

Decision Tree

```r
set.seed(14518)
modelDT <- rpart(classe ~ ., data = train, method = "class")
fancyRpartPlot(modelDT)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2022−Jan−29 00:19:21 Marcos

```r
predictDT <- predict(modelDT, test, type = "class")
confMatDT <- confusionMatrix(predictDT, as.factor(test$classe))
confMatDT
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1507   86    4   12    5
##          B   56  852   83   81   41
##          C    1   60  835   31    6
##          D   91   57   93  769   74
##          E   19   84   11   71  956
##
## Overall Statistics
##
##                Accuracy : 0.8359
##                  95% CI : (0.8261, 0.8452)
```

```
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7927
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9002   0.7480   0.8138   0.7977   0.8835
## Specificity            0.9746   0.9450   0.9798   0.9360   0.9615
## Pos Pred Value         0.9337   0.7655   0.8950   0.7094   0.8379
## Neg Pred Value         0.9609   0.9399   0.9614   0.9594   0.9734
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2561   0.1448   0.1419   0.1307   0.1624
## Detection Prevalence   0.2743   0.1891   0.1585   0.1842   0.1939
## Balanced Accuracy      0.9374   0.8465   0.8968   0.8669   0.9225
```

Random Forest Model offers the best accuracy, with 0.9968 95%CI (0.9950, 0.9981)

Predicting Results

```
predictRF <- predict(modelRF, testing)
predictRF
```

```
##  [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
```