Practical Machine Learning - Exercise Prediction

MahVal

October 6, 2018

Overview

People can track the quantity of thier activity usually quantify how much of a particular activity they do, but they rarely quantify how well they do it. The main goal of this project was to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 different participants, in order to predict the manner in which they did the exercise.The participants were asked to perform barbell lifts correctly and incorrectly in five different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Data Sources

The training data were available at:

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

The testing data were available at:

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

The original source of the full data can be found at: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har).

Step 1: Loading required packages

```
r = getOption("repos")
r["CRAN"] = "http://cran.us.r-project.org"
options(repos = r)
install.packages("weatherData")
```

```
## Installing package into 'C:/Users/Mah Di/Documents/R/win-library/3.4'
## (as 'lib' is unspecified)
```

```
## Warning: package 'weatherData' is not available (for R version 3.4.3)
```

```
#install required packages:
install.packages("corrplot")
```

```
## Installing package into 'C:/Users/Mah Di/Documents/R/win-library/3.4'
## (as 'lib' is unspecified)
```

```
install.packages('caret', dependencies = TRUE)
```

```
## Installing package into 'C:/Users/Mah Di/Documents/R/win-library/3.4'
## (as 'lib' is unspecified)
```

```
install.packages("gbm")
```

```
## Installing package into 'C:/Users/Mah Di/Documents/R/win-library/3.4'
## (as 'lib' is unspecified)
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.4.4
```

```
##
## 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(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.4.4
```

```
## Loaded gbm 2.1.4
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

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

```
## Warning: package 'rpart.plot' was built under R version 3.4.4
```

```r
library(RColorBrewer)
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.4.4
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
```

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

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

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

```
## The following object is masked from 'package:rattle':
## 
##     importance
```

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

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

```r
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.4.4
```

```r
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 3.4.4
```

```
## corrplot 0.84 loaded
```

Step 2: Loading and reading dataset

```r
# set the URL for the download
UrlTrain <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
UrlTest  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# download the datasets
TrainData <- read.csv(UrlTrain, header = T, na.strings = c("", "NA", "#DIV/0!"))
TestData <- read.csv(UrlTest, header = T, na.strings = c("", "NA", "#DIV/0!"))

dim(TrainData)
```

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

```r
dim(TestData)
```

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

```r
head(TrainData, n = 5)
```

	X	user_na...	raw_timestamp_part_1	raw_timestamp_part_2	cvtd_timestamp	r
	<int>	<fctr>	<int>	<int>	<fctr>	<
1	1	carlitos	1323084231	788290	05/12/2011 11:23	r
2	2	carlitos	1323084231	808298	05/12/2011 11:23	r
3	3	carlitos	1323084231	820366	05/12/2011 11:23	r
4	4	carlitos	1323084232	120339	05/12/2011 11:23	r
5	5	carlitos	1323084232	196328	05/12/2011 11:23	r

5 rows | 1-8 of 161 columns

Step 3: Cleaning and preparation of dataset

```r
# Remove variables that are mostly NA
TrainData <- Filter(function(x)!all(is.na(x)), TrainData)
TestData <- Filter(function(x)!all(is.na(x)), TestData)

# Remove all rows where new_window = Yes, since those are summary rows
TrainData <- filter(TrainData, TrainData[, 6] == "no")
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.4
```

```r
TestData <- filter(TestData, TestData[, 6] == "no")

# Remove variables with Nearly Zero Variance
NearZero <- nearZeroVar(TrainData)
TrainData <- TrainData[, -NearZero]
NearZero <- nearZeroVar(TestData)
TestData <- TestData[, -NearZero]

# Remove identification only variables (columns 1 to 7)
TrainData <- TrainData [, -(1:7)]
TestData <- TestData [, -(1:7)]

head(TrainData, n = 10)
```

	pitch_belt	yaw_belt	total_accel_belt	gyros_belt_x	gyros_belt_y	gyros_belt_
	<dbl>	<dbl>	<int>	<dbl>	<dbl>	<db
1	8.07	-94.4	3	0.00	0.00	-0.0
2	8.07	-94.4	3	0.02	0.00	-0.0
3	8.07	-94.4	3	0.00	0.00	-0.0
4	8.05	-94.4	3	0.02	0.00	-0.0
5	8.07	-94.4	3	0.02	0.02	-0.0
6	8.06	-94.4	3	0.02	0.00	-0.0
7	8.09	-94.4	3	0.02	0.00	-0.0
8	8.13	-94.4	3	0.02	0.00	-0.0
9	8.16	-94.4	3	0.02	0.00	-0.0
10	8.17	-94.4	3	0.03	0.00	0.0

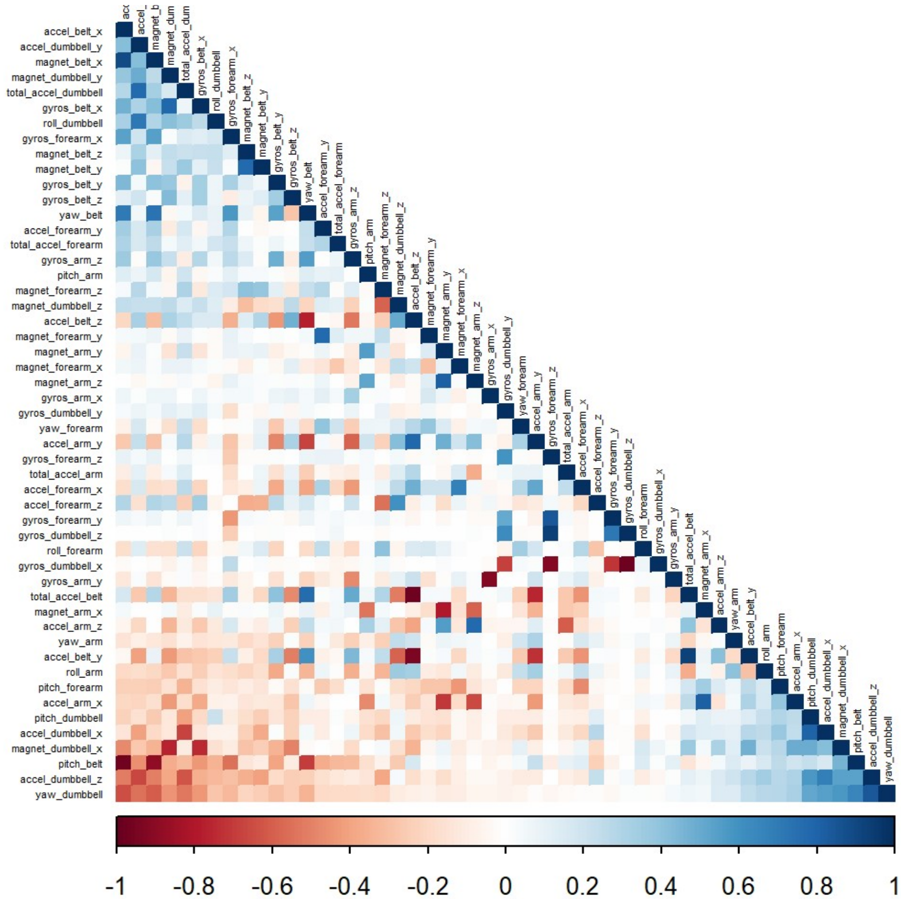
1-10 of 10 rows | 1-8 of 53 columns

```
dim(TrainData)
```

```
## [1] 19216    52
```

Correlation Analysis

```
corr <- cor(TrainData [sapply(TrainData, is.numeric)], use = "complete.obs")
corrplot(corr, order = "FPC", method = "color", type = "lower",
         tl.cex = 0.35, tl.col = rgb(0, 0, 0))
```




```
corcutoff <- findCorrelation(corr, cutoff = .90)
TrainData <- TrainData[,-corcutoff]
TestData <- TestData[,-corcutoff]
```

Cross Validation

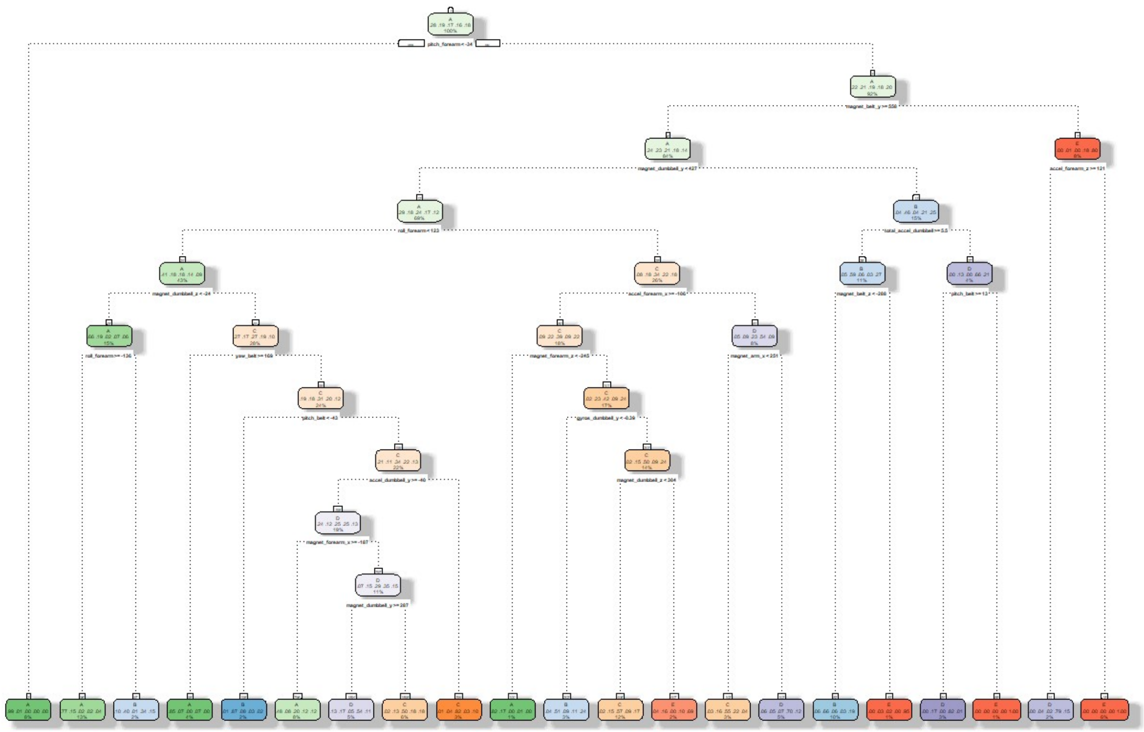
```
# create a partition with the training dataset
set.seed(11223) # For reproducibile purpose
inTrain  <- createDataPartition(TrainData$classe, p=0.7, list=FALSE)
TrainSet <- TrainData[inTrain, ]
TestSet  <- TrainData[-inTrain, ]
dim(TrainSet)
```

```
## [1] 13453    46
```

Prediction Model Building

A: Decision Trees Method

```
DecTree <- rpart(classe ~ ., data=TrainSet, method="class")
fancyRpartPlot(DecTree)
```



Rattle 2018-Oct-30 19:06:08 Mah Di

```
# prediction on Test dataset
predictDecTree <- predict(DecTree, newdata=TestSet, type="class")
confMatDecTree <- confusionMatrix(predictDecTree, TestSet$classe)
confMatDecTree
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1471  205   98   94   84
##          B   74  616   70   96  182
##          C   27  165  801  160  195
##          D   68  110   32  582   95
##          E    1   19    4   12  502
## 
## Overall Statistics
##                                           
##                Accuracy : 0.6892          
##                  95% CI : (0.6771, 0.7012)
##     No Information Rate : 0.2847          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.6044          
##  Mcnemar's Test P-Value : < 2.2e-16       
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8964   0.5525   0.7970   0.6165  0.47448
## Specificity            0.8833   0.9092   0.8850   0.9367  0.99235
## Pos Pred Value         0.7536   0.5934   0.5942   0.6561  0.93309
## Neg Pred Value         0.9554   0.8944   0.9538   0.9258  0.89359
## Prevalence             0.2847   0.1935   0.1744   0.1638  0.18358
## Detection Rate         0.2552   0.1069   0.1390   0.1010  0.08711
## Detection Prevalence   0.3387   0.1801   0.2339   0.1539  0.09335
## Balanced Accuracy      0.8899   0.7308   0.8410   0.7766  0.73341
```

B: Random Forests Method

```
controlRF <- trainControl(method="cv", number=5, verboseIter=FALSE)
modFitRandForest <- train(classe ~ ., data=TrainSet, method="rf",
                          trControl=controlRF)
modFitRandForest$finalModel

# prediction on Test dataset
predictRandForest <- predict(modFitRandForest, newdata=TestSet)
confMatRandForest <- confusionMatrix(predictRandForest, TestSet$classe)
confMatRandForest
```

C: Generalized Boosted Method

```
controlgbm <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modFitgbm  <- train(classe ~ ., data=TrainSet, method = "gbm",
                    trControl = controlgbm, verbose = FALSE)

# prediction on Test dataset
predictgbm <- predict(modFitgbm, newdata=TestSet)
confMatgbm <- confusionMatrix(predictgbm, TestSet$classe)
confMatgbm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1612   49    0    1    4
##          B   15 1032   38    4   13
##          C   10   31  947   36   17
##          D    3    1   17  892   17
##          E    1    2    3   11 1007
##
## Overall Statistics
##
##                Accuracy : 0.9526
##                  95% CI : (0.9468, 0.958)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.94
##  Mcnemar's Test P-Value : 4.838e-09
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9823   0.9256   0.9423   0.9449   0.9518
## Specificity            0.9869   0.9849   0.9802   0.9921   0.9964
## Pos Pred Value         0.9676   0.9365   0.9097   0.9591   0.9834
## Neg Pred Value         0.9929   0.9822   0.9877   0.9892   0.9892
## Prevalence             0.2847   0.1935   0.1744   0.1638   0.1836
## Detection Rate         0.2797   0.1791   0.1643   0.1548   0.1747
## Detection Prevalence   0.2891   0.1912   0.1806   0.1614   0.1777
## Balanced Accuracy      0.9846   0.9553   0.9613   0.9685   0.9741
```

```
AccuracyResults <- data.frame(
  Model = c('CART', 'GBM', 'RF'),
  Accuracy = rbind(confMatDecTree$overall[1], confMatgbm$overall[1], confMatRandForest
$overall[1])
)
print(AccuracyResults)
```

```
##   Model  Accuracy
## 1  CART 0.6892244
## 2   GBM 0.9526288
## 3    RF 0.9930592
```

Considering the accuracy of the three investigated methods, the Random Forest model was selected for validation process on the test data.

Applying the Selected Model to the Test Data

```
TestPredict <- predict(modFitRandForest, newdata=TestData)
TestPredictionResults <- data.frame(
  problem_id=TestData$problem_id,
  predicted=TestPredict
)
print(TestPredictionResults)
```

```
##    problem_id predicted
## 1           1         B
## 2           2         A
## 3           3         B
## 4           4         A
## 5           5         A
## 6           6         E
## 7           7         D
## 8           8         B
## 9           9         A
## 10         10         A
## 11         11         B
## 12         12         C
## 13         13         B
## 14         14         A
## 15         15         E
## 16         16         E
## 17         17         A
## 18         18         B
## 19         19         B
## 20         20         B
```

Conclusion

As seen, the Random Forest model resulted in a highly accurate prediction on the validation data set with accuracy of 0.993 leaving estimated out-of-sample error of 0.69%.