Prediction Assignment Writeup

Enrique Ripoll

19th December 2017

Summary

People regularly quantify *how much* of a particular activity they do, but they rarely quantify *how well* they do it. Using data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants, the goal of this project is to predict the manner in which they did the Unilateral Dumbbell Biceps Curl excercise.

Loading the data

As first step, we should download the data and load the data sets into R

```
setwd("C:/Users/daiko/Documents/DATA_SCIENCE/000_Coursera/Course_8");

urltrain <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
urltest <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
fnametrain <- "./pml-training.csv"
fnametest <- "./pml-testing.csv"

if(!file.exists(fnametrain))   download.file(urltrain, destfile = fnametrain);
if(!file.exists(fnametest))    download.file(urltest, destfile = fnametest);

dtrain0 <- read.csv (fnametrain)
dtest0 <- read.csv (fnametest)
```

Preprocessing

If we take a look to the (train) data set, the outcome is the variable *classe*, which is a factor variable that classifies the excersice in five different fashions: 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).

The data set contains 160 variables. However, the variables [1,3,4,5,6,7] are irrelevant as predictors and there are also some variables with NA values. So we are going to remove this variables in order to clean the data.

```
dtrain <- dtrain0[,-c(1,3,4,5,6,7)]
dtrain <- dtrain[, colSums(is.na(dtrain)) == 0]
dtrain <- dtrain[,-which(sapply(dtrain, class) == "factor")] #Eliminate columns coded as factor
#re-including two removed factor variables that we need to use
dtrain$user_name <- dtrain0$user_name
dtrain$classe <- dtrain0$classe
```

Machine Learning

Data splitting and cross-validation

We are going to split the training set in two subsets: the subtraining set and the subtesting set. This will allow us to cross-validate the model, building a model on the subtraining test, evaluating on the subtesting set and repeating and averaging the estimated errors.

```
set.seed(112)
inTrain <- createDataPartition(y=dtrain$classe, p=0.7)[[1]]
dsubtrain <- dtrain[inTrain,]
dsubtest <- dtrain[-inTrain,]
```

```
control <- trainControl(method = "cv", number = 6)
```

Training the model

Since this is mainly a classification project, we are going to use *boosting* and *rain forest* as learning methods, and compare the results

Boosting was a procedure that combines the outputs of many "weak" classifiers to produce a powerful "committee". We control the tuning parameters with the function "expand.grid"

```
set.seed(112)
modelgrid <- expand.grid (interaction.depth = round(sqrt(NCOL(dsubtrain))),
                         n.trees = c(50, 100, 200),
                         shrinkage = 0.1,
                         n.minobsinnode = c(10,20))

modelBoost <- train (classe ~ ., data=dsubtrain, method="gbm", verbose=FALSE,
              preProc = c("center", "scale"),
              trControl=control,
              tuneGrid = modelgrid)
```

```
## Warning: package 'gbm' was built under R version 3.4.3

## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.3
```

```
modelBoost$results
```

```
##   shrinkage interaction.depth n.minobsinnode n.trees  Accuracy     Kappa
## 1       0.1                 7             10      50 0.9586523 0.9476763
## 4       0.1                 7             20      50 0.9581423 0.9470331
## 2       0.1                 7             10     100 0.9798353 0.9744901
## 5       0.1                 7             20     100 0.9820919 0.9773470
## 3       0.1                 7             10     200 0.9890807 0.9861887
```

```
## 6       0.1                 7             20     200 0.9910465 0.9886758
##    AccuracySD     KappaSD
## 1 0.004736931 0.006001903
## 4 0.005452266 0.006900333
## 2 0.002203148 0.002790790
## 5 0.001683415 0.002130524
## 3 0.001990898 0.002518866
## 6 0.002480278 0.003136755
```

Random forest are also a procedure for classification and operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes

```
set.seed(112)
modelgrid2 <- expand.grid(.mtry=c(5,10,15,20))
modelForest <- train(classe ~ ., data=dsubtrain, method="rf",
                   preProc = c("center", "scale"),
                   trControl=control,
                   tuneGrid = modelgrid2)
```

```
## Warning: package 'randomForest' was built under R version 3.4.3

## randomForest 4.6-12

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

##
## Attaching package: 'randomForest'

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

```
modelForest$results
```

```
##   mtry  Accuracy     Kappa  AccuracySD     KappaSD
## 1    5 0.9928659 0.9909758 0.001718496 0.002174218
## 2   10 0.9922838 0.9902398 0.002060550 0.002606057
## 3   15 0.9919199 0.9897800 0.001985569 0.002510742
## 4   20 0.9914103 0.9891351 0.002040897 0.002581196
```

Results

In order to obtain the out-sample error we use the Confussion Matrix, that compare the truth values of the subtesting set (that we have still not use) with the predictions, for both boosting and rain forest methods

```
CMBoost <- confusionMatrix(dsubtest$classe, predict(modelBoost, dsubtest))
CMForest <- confusionMatrix(dsubtest$classe, predict(modelForest, dsubtest))
```

The test accuracy is 0.9923534 for boosting model and 0.9935429 for rain forest model. Let's see what are the most important variable in both models:

Rain Forest model:

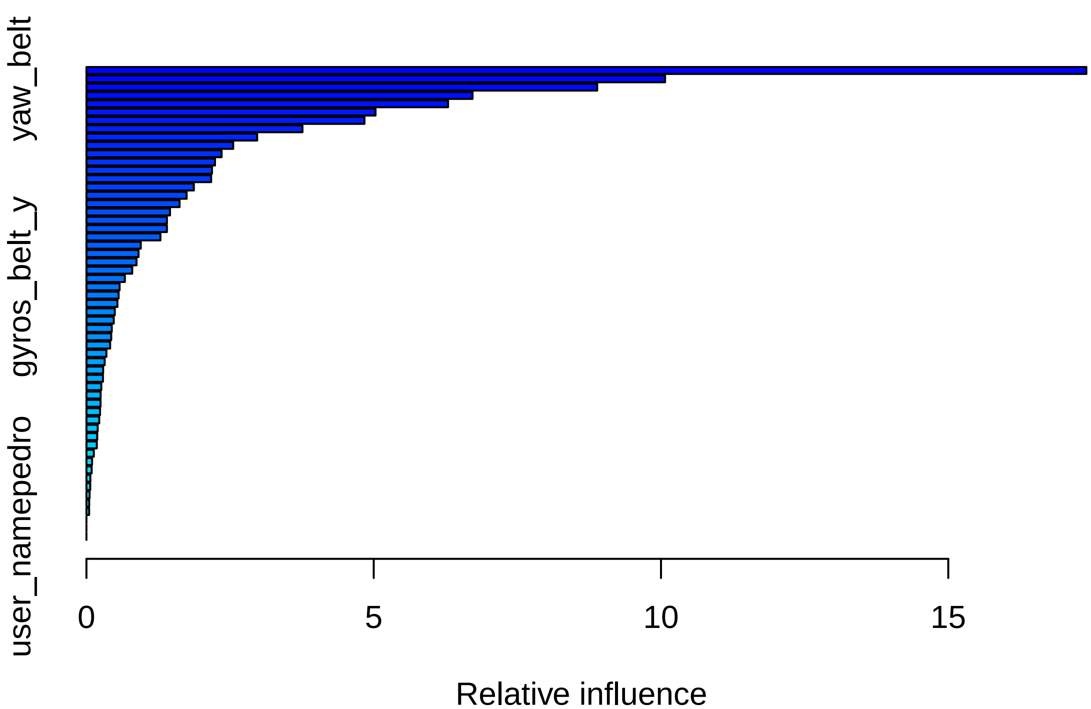
```
varImp(modelForest)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 57)
##
```

```
##                        Overall
## roll_belt               100.00
## yaw_belt                 76.75
## magnet_dumbbell_z        62.67
## pitch_forearm            60.98
## magnet_dumbbell_y        59.23
## pitch_belt               59.10
## roll_forearm             50.56
## magnet_dumbbell_x        46.06
## roll_dumbbell            40.39
## magnet_belt_z            39.73
## accel_dumbbell_y         37.06
## magnet_belt_y            36.43
## accel_belt_z             36.14
## accel_dumbbell_z         34.22
## accel_forearm_x          30.57
## gyros_belt_z             30.27
## roll_arm                 28.47
## total_accel_dumbbell     25.33
## yaw_dumbbell             25.17
## accel_dumbbell_x         25.05
```

Boosting model:

```r
head(summary(modelBoost$finalModel),10)
```




```
##                                    var    rel.inf
```

```
## roll_belt                 roll_belt 17.403429
## yaw_belt                   yaw_belt 10.071318
## pitch_forearm         pitch_forearm  8.890047
## magnet_dumbbell_z magnet_dumbbell_z  6.720207
## magnet_dumbbell_y magnet_dumbbell_y  6.294065
## roll_forearm           roll_forearm  5.031690
## pitch_belt               pitch_belt  4.838925
## magnet_belt_z         magnet_belt_z  3.760095
## gyros_belt_z           gyros_belt_z  2.971410
## accel_dumbbell_y   accel_dumbbell_y  2.554127
```

As we could see, both models aggre wiht the most relevant variables, as expected. Let's plot the outcome versus the two main predictors (roll_belt and way_belt).

```
p1 <- ggplot(dsubtrain, aes(x=roll_belt, y=pitch_forearm, colour=classe)) +
      geom_point() + theme_minimal() +
      labs (title = "SubTraining set")
```

The plot shows that the different classes are classified in different clusters.

Prediction

Once we have trained our model(s), we could use them to predict the outcome for the original test set, that have still not been used.

```
predictionBoost <- predict(modelBoost, dtest0)
predictionForest <- predict(modelForest, dtest0)
prediction <- data.frame(id=dtest0$problem_id, name=dtest0$user_name, Boosting_classe=predictionBoost,
                         RForest_classe=predictionForest)
prediction
```

```
##    id     name Boosting_classe RForest_classe
## 1   1    pedro               B              B
## 2   2   jeremy               A              A
## 3   3   jeremy               B              B
## 4   4   adelmo               A              A
## 5   5   eurico               A              A
## 6   6   jeremy               E              E
## 7   7   jeremy               D              D
## 8   8   jeremy               B              B
## 9   9 carlitos               A              A
## 10 10  charles               A              A
## 11 11 carlitos               B              B
## 12 12   jeremy               C              C
## 13 13   eurico               B              B
## 14 14   jeremy               A              A
## 15 15   jeremy               E              E
## 16 16   eurico               E              E
## 17 17    pedro               A              A
## 18 18 carlitos               B              B
## 19 19    pedro               B              B
## 20 20   eurico               B              B
```

Conclusions

From an original data set with 19622 observations and 159 variables, two predictive models have been implemented to predict the manner in which a subject was peforming an exercise. By machine learning algorithms, two different methods (boosting and random forests) have been used to train the models, using 54 different predictors.

The obtained out-sample error for both predictive models have been greater than 0.99, confirming that both models estimate the *classe* of the excercise really well. The error matrix from the test data set can be seen in the following tables.

```
kable(CMBoost$table, caption = "Boosting method")
```

Table 1: Boosting method

	A	B	C	D	E
A	1670	4	0	0	0
B	2	1132	5	0	0
C	0	9	1012	5	0
D	0	0	12	949	3
E	0	2	0	3	1077

```
kable(CMForest$table, caption = "Random Forest method")
```

Table 2: Random Forest method

	A	B	C	D	E
A	1672	2	0	0	0
B	3	1132	4	0	0
C	0	7	1018	1	0
D	0	0	18	945	1
E	0	0	0	2	1080

Finally, both models have been applied to predict the *classe* of an (unused) validation data set, showing both identical results as expected.