Practical Machine Learning - Prediction Assignment

Mehran Behzad

26 6 2020

Intro

Data from fitness tracking devices were gathered for subjects doing curls in 4 different manners (see [http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har]). The goal of this assignment is to build a training model based on these data capable of predicting the tpye of exersise based of the movement data measured by these devices.

Data exploration and cleanup

```r
# load 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
```

```r
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```r
# read data
training <- read.csv(file = 'data/pml-training.csv')
testing <- read.csv(file = 'data/pml-testing.csv')
```

The table has 19622 rows of data, each has 160 features.

```r
#
head(training[(1:10)])
```

```
##   X user_name raw_timestamp_part_1 raw_timestamp_part_2   cvtd_timestamp
## 1 1  carlitos           1323084231               788290 05/12/2011 11:23
## 2 2  carlitos           1323084231               808298 05/12/2011 11:23
## 3 3  carlitos           1323084231               820366 05/12/2011 11:23
## 4 4  carlitos           1323084232               120339 05/12/2011 11:23
## 5 5  carlitos           1323084232               196328 05/12/2011 11:23
## 6 6  carlitos           1323084232               304277 05/12/2011 11:23
##   new_window num_window roll_belt pitch_belt yaw_belt
## 1         no         11      1.41       8.07    -94.4
```

```
## 2          no          11       1.41        8.07     -94.4
## 3          no          11       1.42        8.07     -94.4
## 4          no          12       1.48        8.05     -94.4
## 5          no          12       1.48        8.07     -94.4
## 6          no          12       1.45        8.06     -94.4
```

It seems the first 7 columns are user specific or some time stamps and not task specific. We can remove them for the training.

```
# remove first columns which are not relevant for the excercise
training <- training[-(1:7)]
```

Furtermore, there are alot of features that are almost N/A for all obesrvations. We will remove them:

```
# remove column which have at least 90% NAs
training <- training[ , apply(training, 2, function(y) length(which(is.na(y))) < .90 * dim(training)[1]]
# same for test set
testing <- testing[-(1:7)]
testing <- testing[ , apply(testing, 2, function(y) length(which(is.na(y))) < .90 * dim(testing)[1])]
```

We will also remove any feature/column which is missing the Training or Testing data set which won't help with prediction or training.

```
# memorize the calsse column which is delibertly missing in the tetesting dataset and has to be predict
classe <- training$classe

# remove those which are only present in one dataset
testing <- testing[, intersect(colnames(training), colnames(testing))]
training <- training[, intersect(colnames(training), colnames(testing))]
training$classe <- classe
```

This gives us 53 features which we will feed the models with.

Training

Now we will split our training data, to allow testing its prediction with known observations:

```
inTrain <- createDataPartition(y= training$classe, p=0.7, list=FALSE)
trainingSub <- training[inTrain,]
testingSub <- training[-inTrain,]
```

Using the Random Forest (rf) algorithm to train the model

```
model_rf <- train(classe  ~ ., method="rf", data = trainingSub,
               trControl = trainControl(method = "cv"),
               number = 3, na.action=na.exclude)
```

We build a second model using lda technique:

```
model_lda <- train(classe~., method="lda", data = trainingSub,
                trControl=  trainControl(method="cv", number=10),
                na.action=na.exclude)
```

Evaluation

```
confusionMatrix(testingSub$classe, predict(model_rf, testingSub))
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    6 1132    1    0    0
##          C    0    8 1018    0    0
##          D    0    0   13  950    1
##          E    0    0    1    2 1079
##
## Overall Statistics
##
##                Accuracy : 0.9946
##                  95% CI : (0.9923, 0.9963)
##     No Information Rate : 0.2855
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9931
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9930   0.9855   0.9979   0.9991
## Specificity            1.0000   0.9985   0.9984   0.9972   0.9994
## Pos Pred Value         1.0000   0.9939   0.9922   0.9855   0.9972
## Neg Pred Value         0.9986   0.9983   0.9969   0.9996   0.9998
## Prevalence             0.2855   0.1937   0.1755   0.1618   0.1835
## Detection Rate         0.2845   0.1924   0.1730   0.1614   0.1833
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9982   0.9958   0.9919   0.9975   0.9992
```

```r
confusionMatrix(testingSub$classe, predict(model_lda, testingSub))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1389   36  120  126    3
##          B  182  743  133   37   44
##          C  119   99  672  116   20
##          D   69   42  108  705   40
##          E   52  192   93   95  650
##
## Overall Statistics
##
##                Accuracy : 0.7067
##                  95% CI : (0.6949, 0.7183)
##     No Information Rate : 0.3077
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6282
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7670   0.6682   0.5968   0.6534   0.8587
## Specificity            0.9300   0.9170   0.9256   0.9461   0.9158
## Pos Pred Value         0.8297   0.6523   0.6550   0.7313   0.6007
## Neg Pred Value         0.8998   0.9223   0.9066   0.9240   0.9777
## Prevalence             0.3077   0.1890   0.1913   0.1833   0.1286
## Detection Rate         0.2360   0.1263   0.1142   0.1198   0.1105
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.8485   0.7926   0.7612   0.7997   0.8872
```

It seems that the random forest method gives us a model with a hight (~99%) accuracy.

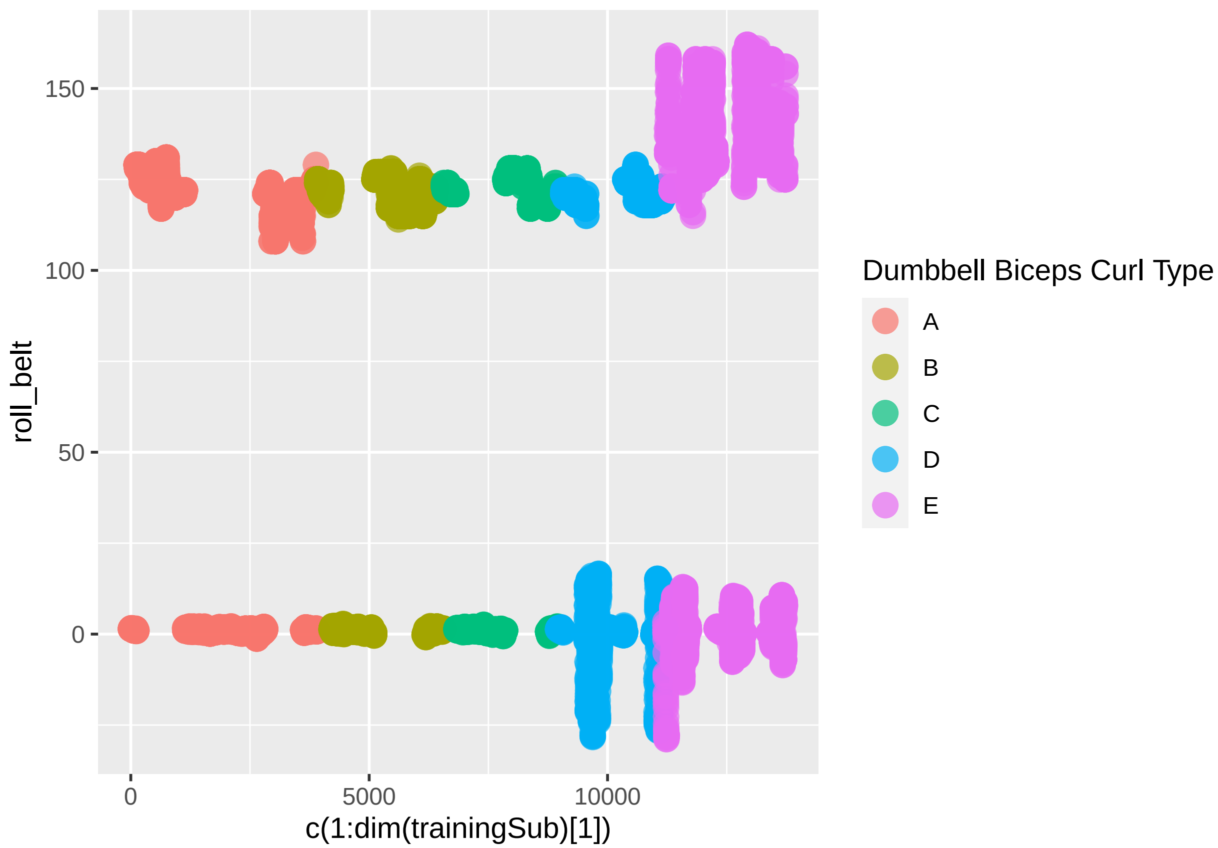
Let's see which parameters have the mose influence:

```r
varImp(model_rf)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 52)
##
##                      Overall
## roll_belt             100.00
## yaw_belt               79.70
## magnet_dumbbell_z      66.34
## magnet_dumbbell_y      60.41
## pitch_belt             59.82
## pitch_forearm          54.47
## magnet_dumbbell_x      52.32
## roll_forearm           50.07
## accel_belt_z           44.97
## magnet_belt_z          43.23
## accel_dumbbell_y       41.54
## roll_dumbbell          41.52
## magnet_belt_y          39.90
## accel_dumbbell_z       35.37
## roll_arm               33.25
## accel_forearm_x        32.90
## accel_arm_x            28.32
## total_accel_dumbbell   28.25
## accel_dumbbell_x       27.39
## yaw_dumbbell           27.26
```

And the one with most effect plotted:

```r
ggplot(trainingSub, aes(y=roll_belt, x = c(1:dim(trainingSub)[1]), colour = factor(classe))) +
  geom_point(size=4, alpha=0.7) +
  labs(color = "Dumbbell Biceps Curl Type")
```



Predicting testing data

```r
predict(model_rf, testing)
```

```
##  [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
```