Code ▾

R Notebook

Background

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. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. 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, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Data

The training data for this project are available here:

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

The test data are available here:

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

The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har). If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Report

summary

- Aim: We aim build a model to predict the quality of barbell lifts performed in 20 test cases.
- Data: The data consists of 19622 observations and 160 variables from a study by Ugulino and Others - Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements. An additional twenty observations are to be classified based on the model constructed through this exercise. These twenty observations are the validation data set
- Methods: using a cross validation approach to estimate error assess variations of three machine learning models;gradient boosting machine (GBM), distributed random forest (DRF), generalised linear model (GLM).
- Results: GBM was found to be the most accurate model with > 99% accuracy

setup

load libraries

Hide

```
library(caret)
```

Hide

```
library(tidyverse)
```

load data

Hide

```
# download data
trainurl = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testurl = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
#download.file(trainurl, "pml-training.csv")
#download.file(testurl, "pml-testing.csv")
# load data
# exclude variables containing NA
# exclude first 7 variables (not used)
not_all_na <- function(x) {!all(is.na(x))} # select columns that dont have all values NA
not_any_na <- function(x) {!any(is.na(x))} # select columns that don't have any NA
training <-
  read_csv("pml-training.csv",na=c("NA","#DIV/0!", "")) %>%
  select_if(not_any_na) %>%
  select(-c(1:7))
testing <-
  read_csv("pml-testing.csv",na=c("NA","#DIV/0!", "")) %>%
  select_if(not_any_na) %>%
  select(-c(1:7))
# recode classe as numeric - now skipped (we dont save)
ALPHA2num <- function(x) {utf8ToInt(x) - utf8ToInt("A") + 1L}
ignore <- training %>%
  rowwise() %>%
  mutate(classe = ALPHA2num(classe))
```

split training into further training/testing for cross validation

Hide

```
cv <- createDataPartition(y=training$classe, p=0.6, list = FALSE)
training_train <- training[cv, ]
training_test <- training[-cv, ]
```

Give us a quick summary of the training data

Hide

```
library(skimr)
skim(training)
```

```
-- Data Summary ------------------------
                           Values
Name                       training
Number of rows             19622
Number of columns          60
_______________________
Column type frequency:
  character                4
  numeric                  56
________________________
Group variables            None
```

	skim_variable <chr>	n_missing <int>	complete_rate <dbl>	… <int>	… <int>	em… <int>	n_unique <int>	whitespace <int>
1	user_name	0	1	5	8	0	6	0
2	cvtd_timestamp	0	1	16	16	0	20	0
3	new_window	0	1	2	3	0	2	0
4	classe	0	1	1	1	0	5	0

4 rows

	skim_variable <chr>	n_missing <int>	complete_rate <dbl>	mean <dbl>	sd <dbl>	
1	...1	0	1	9.811500e+03	5.664528e+03	1.000
2	raw_timestamp_part_1	0	1	1.322827e+09	2.049277e+05	1.322
3	raw_timestamp_part_2	0	1	5.006561e+05	2.882229e+05	2.940
4	num_window	0	1	4.306400e+02	2.479096e+02	1.000
5	roll_belt	0	1	6.440720e+01	6.275026e+01	-2.890
6	pitch_belt	0	1	3.052828e-01	2.235124e+01	-5.580
7	yaw_belt	0	1	-1.120506e+01	9.519393e+01	-1.800
8	total_accel_belt	0	1	1.131261e+01	7.742309e+00	0.000
9	gyros_belt_x	0	1	-5.592192e-03	2.073290e-01	-1.040
10	gyros_belt_y	0	1	3.958771e-02	7.823552e-02	-6.40

1-10 of 56 rows | 1-7 of 11 columns Previous **1** 2 3 4 5 6 Next

modelling

Run a quick comparison of models Referenced from H2O library code example

Hide

```
# Load library
library(h2o)
# start h2o cluster
invisible(h2o.init())
#convert variables to factors
training_train[,y] = as.factor(pull(training_train,y))
training_test[,y] = as.factor(pull(training_test,y))
# convert data as h2o type
train_h = as.h2o(training_train)
test_h = as.h2o(training_test)
# set label type
y = 'classe'
pred = setdiff(names(training_train), y)
#convert variables to factors
# Run AutoML for 20 base models
aml = h2o.automl(x = pred, y = y,
                  training_frame = train_h,
                  max_models = 20,
                  seed = 1,
                  max_runtime_secs = 20
                 )
```

Show leaderboard of models

Hide

```
# AutoML Leaderboard
lb = aml@leaderboard
print(lb, n = nrow(lb))
```

	model_id <chr>	mean_per_class_error <dbl>	logloss <dbl>	rmse <dbl>	
1	GBM_2_AutoML_2_20220703_202240	0.01086514	0.03863591	0.09264354	0.00
2	GBM_3_AutoML_2_20220703_202240	0.01181071	0.04150688	0.09628754	0.00
3	GBM_1_AutoML_2_20220703_202240	0.01351100	0.03227514	0.09290764	0.00
4	DRF_1_AutoML_2_20220703_202240	0.01832705	0.15231994	0.18467547	0.03
5	GLM_1_AutoML_2_20220703_202240	0.26887539	0.68682032	0.47114998	0.22

5 rows

```
[5 rows x 5 columns]
```

From this we see that GBM_2 gives the best performance We display performance matrices and history of the model here

Hide

```
m <- aml@leader
# create a confusion matrix
caret::confusionMatrix(training_test$classe, prediction$predict)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    A    B    C    D    E
         A 2227    5    0    0    0
         B   11 1501    6    0    0
         C    0   21 1343    4    0
         D    0    0   13 1273    0
         E    0    0    2    7 1433

Overall Statistics

               Accuracy : 0.9912
                 95% CI : (0.9889, 0.9932)
    No Information Rate : 0.2852
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9889

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: A Class: B Class: C Class: D Class: E
Sensitivity            0.9951   0.9830   0.9846   0.9914   1.0000
Specificity            0.9991   0.9973   0.9961   0.9980   0.9986
Pos Pred Value         0.9978   0.9888   0.9817   0.9899   0.9938
Neg Pred Value         0.9980   0.9959   0.9968   0.9983   1.0000
Prevalence             0.2852   0.1946   0.1738   0.1637   0.1826
Detection Rate         0.2838   0.1913   0.1712   0.1622   0.1826
Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
Balanced Accuracy      0.9971   0.9901   0.9904   0.9947   0.9993
```

Hide

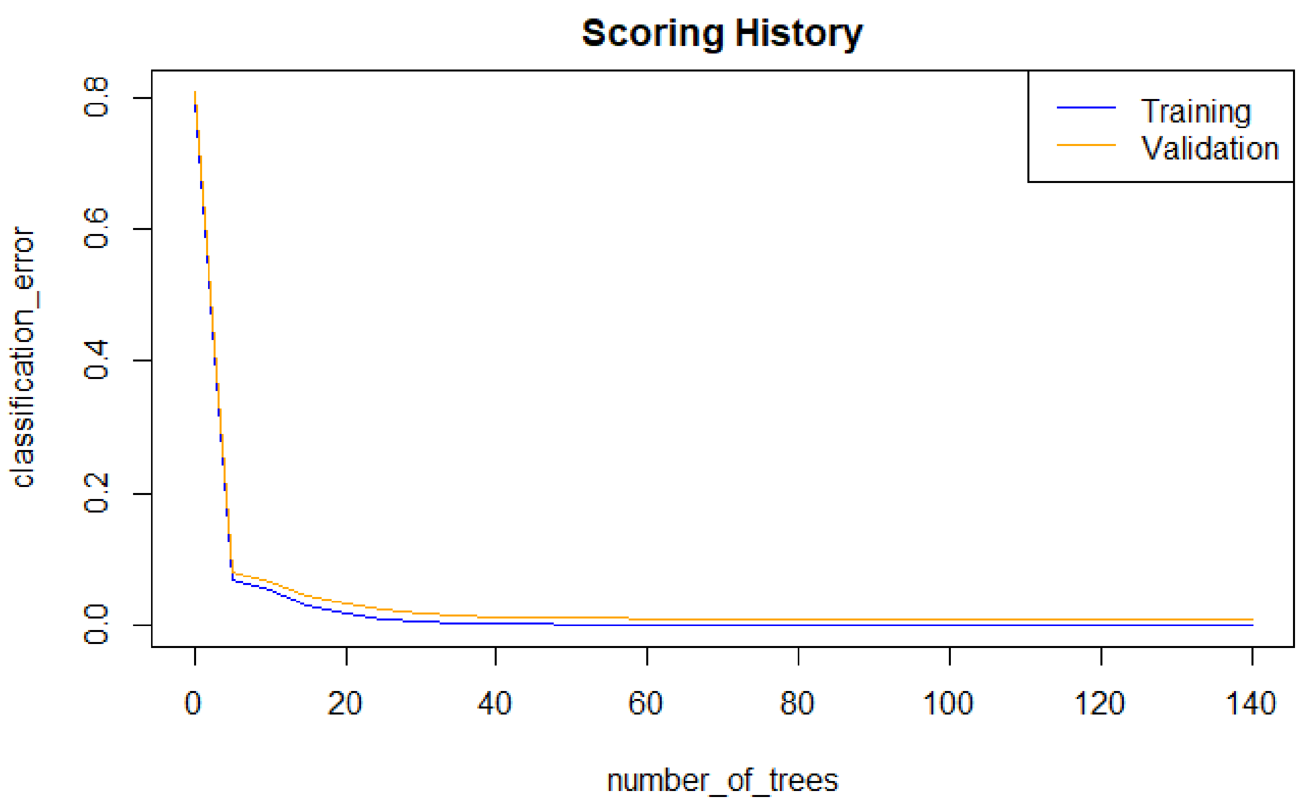
```
# close h2o connection
#h2o.shutdown(prompt = F)
m@parameters
```

```
$model_id
[1] "GBM_2_AutoML_2_20220703_202240"

$training_frame
[1] "AutoML_2_20220703_202240_training_training_train_sid_ade1_273"

$validation_frame
[1] "AutoML_2_20220703_202240_validation_training_train_sid_ade1_273"

$keep_cross_validation_models
[1] FALSE

$score_tree_interval
[1] 5

$ntrees
[1] 10000

$max_depth
[1] 7

$stopping_rounds
[1] 3

$stopping_metric
[1] "logloss"

$stopping_tolerance
[1] 0.009215122

$seed
[1] 4

$distribution
[1] "multinomial"

$sample_rate
[1] 0.8

$col_sample_rate
[1] 0.8

$col_sample_rate_per_tree
[1] 0.8

$histogram_type
[1] "UniformAdaptive"

$categorical_encoding
[1] "Enum"

$x
 [1] "roll_belt"            "pitch_belt"           "yaw_belt"             "total_accel_belt"
"gyros_belt_x"         "gyros_belt_y"         "gyros_belt_z"
 [8] "accel_belt_x"         "accel_belt_y"         "accel_belt_z"         "magnet_belt_x"
```

```
"magnet_belt_y"        "magnet_belt_z"        "roll_arm"
[15] "pitch_arm"            "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
"gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
[22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"         "magnet_arm_y"
"magnet_arm_z"         "roll_dumbbell"        "pitch_dumbbell"
[29] "yaw_dumbbell"         "total_accel_dumbbell" "gyros_dumbbell_x"     "gyros_dumbbell_y"
"gyros_dumbbell_z"     "accel_dumbbell_x"     "accel_dumbbell_y"
[36] "accel_dumbbell_z"     "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
"roll_forearm"         "pitch_forearm"        "yaw_forearm"
[43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"      "gyros_forearm_z"
"accel_forearm_x"      "accel_forearm_y"      "accel_forearm_z"
[50] "magnet_forearm_x"     "magnet_forearm_y"     "magnet_forearm_z"

$y
[1] "classe"
```

Hide

```
plot(m)
```



And finally the prediction on the test dataset (now validation dataset)

Hide

```
validation_h = as.h2o(testing)
prediction = h2o.predict(aml@leader, validation_h) %>%
                         as.data.frame()
```

Hide

```
# converted to h2o frame

prediction
```

predict <fctr>	A <dbl>	B <dbl>	C <dbl>	D <dbl>	E <dbl>
B	5.876795e-03	8.853072e-01	1.068642e-01	5.120398e-04	1.439777e-03
A	9.997455e-01	2.177900e-04	2.225265e-05	4.943333e-06	9.553789e-06
B	7.791393e-03	9.688698e-01	1.868828e-02	1.351823e-03	3.298711e-03
A	9.997204e-01	7.985877e-06	1.012960e-04	1.664220e-04	3.927235e-06
A	9.998675e-01	6.249663e-05	6.080677e-05	1.357940e-06	7.812599e-06
E	1.923637e-06	5.791374e-05	2.889833e-04	6.002039e-06	9.996452e-01
D	4.588773e-05	2.468543e-04	2.135429e-03	9.974382e-01	1.336762e-04
B	3.388855e-04	9.966069e-01	7.522652e-04	1.595089e-03	7.068835e-04
A	9.999760e-01	1.535018e-05	3.016274e-06	3.024034e-06	2.614116e-06
A	9.999519e-01	3.776395e-05	5.717074e-06	2.461197e-06	2.140006e-06

1-10 of 20 rows Previous **1** 2 Next

Hide

```
pull(prediction,predict)
```

```
 [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
```