04_AutoML_example1

June 27, 2019

1 AutoML and Ensemble Learning with H2O.ai

- Introduction to H2O ai
- getting started
- AutoML workflow
- Performance and Prediction
- AutoML variable importance
- Available algorithms

```
In [1]: library( readr )
        library( dplyr )
        library( GGally )
```

```
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

Loading required package: ggplot2
Registered S3 methods overwritten by 'ggplot2':
  method         from
  [.quosures     rlang
  c.quosures     rlang
  print.quosures rlang
Registered S3 method overwritten by 'GGally':
  method from
  +.gg   ggplot2

Attaching package: GGally

The following object is masked from package:dplyr:
```

```
nasa
```

2 Funky data example

```
In [3]: funky_data <- read_csv( "funkydata.csv" ) %>%
            mutate( Y = factor( Y ) )
```

```
Parsed with column specification:
cols(
  Gaussian1 = col_double(),
  Gaussian2 = col_double(),
  Moon1 = col_double(),
  Moon2 = col_double(),
  Circle1 = col_double(),
  Circle2 = col_double(),
  Y = col_double()
)
```

```
In [4]: library( skimr )
```

```
Attaching package: skimr

The following object is masked from package:stats:

    filter
```

```
In [5]: skim_to_wide( funky_data )
```

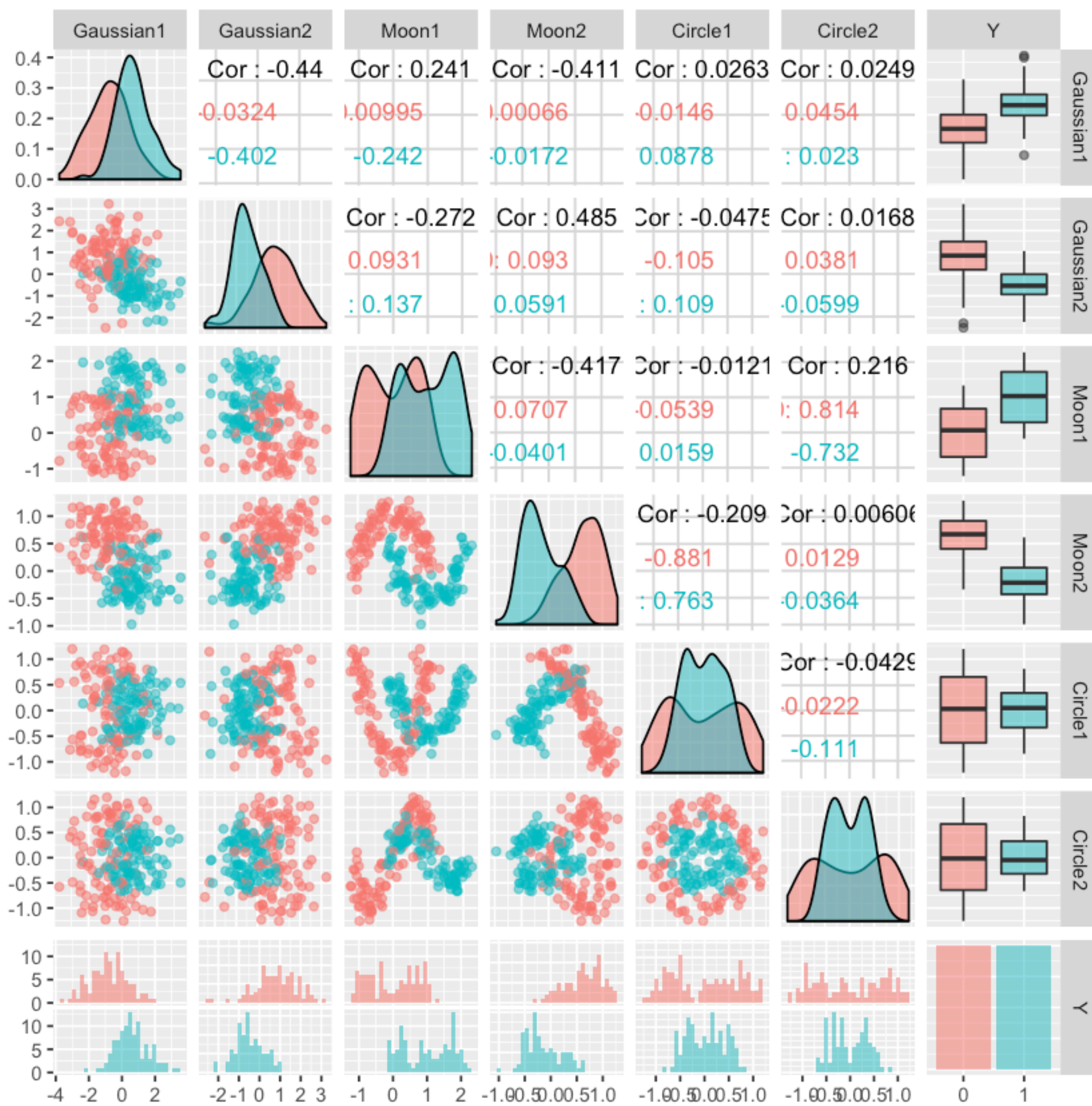
A tibble: 7 Œ 16

type <chr>	variable <chr>	missing <chr>	complete <chr>	n <chr>	n_unique <chr>	top_counts <chr>	ord <ch
factor	Y	0	200	200	2	0: 100, 1: 100, NA: 0	FA
numeric	Circle1	0	200	200	NA	NA	NA
numeric	Circle2	0	200	200	NA	NA	NA
numeric	Gaussian1	0	200	200	NA	NA	NA
numeric	Gaussian2	0	200	200	NA	NA	NA
numeric	Moon1	0	200	200	NA	NA	NA
numeric	Moon2	0	200	200	NA	NA	NA

```
In [6]: ggpairs( funky_data, aes( alpha=0.1, color=Y ) )
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
In [7]: library(h2o)
```

```
----------------------------------------------------------------------

Your next step is to start H2O:
    > h2o.init()
```

```
For H2O package documentation, ask for help:
    > ??h2o

After starting H2O, you can use the Web UI at http://localhost:54321
For more information visit http://docs.h2o.ai

----------------------------------------------------------------------


Attaching package: h2o

The following objects are masked from package:stats:

    cor, sd, var

The following objects are masked from package:base:

    &&, %*%, %in%, ||, apply, as.factor, as.numeric, colnames,
    colnames<-, ifelse, is.character, is.factor, is.numeric, log,
    log10, log1p, log2, round, signif, trunc
```

```
In [8]: h2o.init()
```

```
H2O is not running yet, starting it now...

Note:  In case of errors look at the following log files:
    /var/folders/8j/_h7w48591zq59klcljxctvcs25_0zh/T//RtmpaxYBGF/h2o_colettace_started_from_r.o
    /var/folders/8j/_h7w48591zq59klcljxctvcs25_0zh/T//RtmpaxYBGF/h2o_colettace_started_from_r.o


Starting H2O JVM and connecting: . Connection successful!

R is connected to the H2O cluster:
    H2O cluster uptime:         1 seconds 451 milliseconds
    H2O cluster timezone:       America/New_York
    H2O data parsing timezone:  UTC
    H2O cluster version:        3.24.0.5
    H2O cluster version age:    8 days
    H2O cluster name:           H2O_started_from_R_colettace_yhm690
    H2O cluster total nodes:    1
    H2O cluster total memory:   3.56 GB
    H2O cluster total cores:    8
    H2O cluster allowed cores:  8
    H2O cluster healthy:        TRUE
```

```
    H2O Connection ip:          localhost
    H2O Connection port:        54321
    H2O Connection proxy:       NA
    H2O Internal Security:      FALSE
    H2O API Extensions:         Amazon S3, XGBoost, Algos, AutoML, Core V3, Core V4
    R Version:                  R version 3.6.0 (2019-04-26)
```

```
In [9]: funky_data = as.h2o( funky_data )
```

```
  |======================================================================| 100%
```

```
In [10]: class( funky_data )
```

'H2OFrame'

```
In [11]: summary( funky_data )
```

```
Warning message in summary.H2OFrame(funky_data):
Approximated quantiles computed! If you are interested in exact quantiles, please pass the `exa
```

```
 Gaussian1          Gaussian2          Moon1              Moon2
 Min.   :-3.70164   Min.   :-2.4650   Min.   :-1.20354   Min.   :-0.9726
 1st Qu.:-0.92574   1st Qu.:-0.6316   1st Qu.:-0.09335   1st Qu.:-0.2183
 Median :-0.01343   Median : 0.1066   Median : 0.46347   Median : 0.2632
 Mean   :-0.09543   Mean   : 0.1677   Mean   : 0.50255   Mean   : 0.2487
 3rd Qu.: 0.76956   3rd Qu.: 0.9259   3rd Qu.: 1.04960   3rd Qu.: 0.6657
 Max.   : 3.38425   Max.   : 3.2245   Max.   : 2.24426   Max.   : 1.2825
 Circle1            Circle2             Y
 Min.   :-1.21187   Min.   :-1.258472   0:100
 1st Qu.:-0.44734   1st Qu.:-0.474603   1:100
 Median : 0.04368   Median :-0.044858
 Mean   : 0.01718   Mean   :-0.002766
 3rd Qu.: 0.47565   3rd Qu.: 0.432842
 Max.   : 1.19801   Max.   : 1.200726
```

2.1 Split into train val test

- H2O you give your "desired" train/val/test ratios and it gives you back approximately the proportions you want
- E.g., Say we want 500 samples for training, 100 samples for validation and 400 samples for test data:

```
In [12]: funky_splits <- h2o.splitFrame( funky_data, ratios=c(0.5) )
```

```
In [13]: funky_train <- funky_splits[[1]]
         funky_test <- funky_splits[[2]]
```

```
In [14]: dim( funky_train )
```

1. 97 2. 7

```
In [15]: dim( funky_test )
```

1. 103 2. 7

```
In [16]: library(tictoc)
```

```
In [17]: tic()
         aml_results <- h2o.automl(
             # x is omitted since we want to use all the columns except "Y" as predictors
             y = 'Y',
             training_frame = funky_train,
             leaderboard_frame = funky_test,
             max_runtime_secs = 360, # Default time is one hour!!
             exclude_algos = 'GBM',
         )
         toc()
```

```
  |======================================================================| 100%
318.586 sec elapsed
```

2.2 AutoML results

- Printing the results object shows you info from the winning "leader" model, and well as the "leaderboard" of how well the various models performed

```
In [18]: dim( aml_results@leaderboard )
```

1. 208 2. 6

```
In [22]: head( aml_results@leaderboard, 20 )
```

A df[,6]: 20 Œ 6

model_id <chr>	auc <dbl>	logloss <dbl>	m <
XGBoost_grid_1_AutoML_20190627_121001_model_143	1.0000000	6.252269e-02	0
XGBoost_grid_1_AutoML_20190627_121001_model_122	1.0000000	6.911448e-02	0
XGBoost_grid_1_AutoML_20190627_121001_model_148	1.0000000	7.210843e-02	0
XGBoost_grid_1_AutoML_20190627_121001_model_124	1.0000000	5.184913e-02	0
DeepLearning_grid_1_AutoML_20190627_121001_model_14	1.0000000	1.200763e-07	0
DeepLearning_grid_1_AutoML_20190627_121001_model_7	1.0000000	1.795007e-02	0
StackedEnsemble_BestOfFamily_AutoML_20190627_121001	1.0000000	3.869660e-02	0
XGBoost_grid_1_AutoML_20190627_121001_model_156	0.9996226	6.308121e-02	0
DeepLearning_grid_1_AutoML_20190627_121001_model_3	0.9992453	6.517710e-02	0
XGBoost_grid_1_AutoML_20190627_121001_model_85	0.9992453	7.710078e-02	0
StackedEnsemble_AllModels_AutoML_20190627_121001	0.9992453	1.011197e-01	0
DeepLearning_grid_1_AutoML_20190627_121001_model_9	0.9992453	6.464990e-02	0
XGBoost_grid_1_AutoML_20190627_121001_model_25	0.9988679	8.680839e-02	0
XGBoost_grid_1_AutoML_20190627_121001_model_173	0.9988679	5.271640e-01	0
XGBoost_grid_1_AutoML_20190627_121001_model_140	0.9988679	8.582076e-02	0
XGBoost_grid_1_AutoML_20190627_121001_model_50	0.9984906	7.613767e-02	0
XGBoost_grid_1_AutoML_20190627_121001_model_181	0.9984906	3.647458e-01	0
XGBoost_grid_1_AutoML_20190627_121001_model_4	0.9984906	7.956581e-02	0
XGBoost_grid_1_AutoML_20190627_121001_model_157	0.9981132	8.424608e-02	0
XGBoost_grid_1_AutoML_20190627_121001_model_7	0.9981132	1.473685e-01	0

```
In [20]: getParms( aml_results@leader )
         # or a synonym:
         # aml_results@leader@parameters
```

$model_id 'XGBoost_grid_1_AutoML_20190627_121001_model_143'

$training_frame 'automl_training_RTMP_sid_b932_3'

$nfolds 5

$keep_cross_validation_models FALSE

$keep_cross_validation_predictions TRUE

$fold_assignment 'Modulo'

$stopping_metric 'logloss'

$stopping_tolerance 0.05

$seed '-9118792347905751911'

$distribution 'bernoulli'

$ntrees 103

$max_depth 20

$min_rows 0.01

$learn_rate 0.05

$sample_rate 0.6

$col_sample_rate 0.8

$col_sample_rate_per_tree 0.8

$score_tree_interval 5

$booster 'dart'

$reg_lambda 0.01

$reg_alpha 0.001

$x 1. 'Gaussian1' 2. 'Gaussian2' 3. 'Moon1' 4. 'Moon2' 5. 'Circle1' 6. 'Circle2'

$y 'Y'

```
In [21]: as.data.frame( predict( aml_results@leader, funky_test ) )
```

```
  |======================================================================| 100%
```

A df[,3]: 103 Œ 3

predict	p0	p1
<fct>	<dbl>	<dbl>
1	0.003199100	0.996800900
0	0.990319550	0.009680446
0	0.996739209	0.003260799
0	0.994402945	0.005597057
0	0.996943951	0.003056029
1	0.002768338	0.997231662
0	0.996874869	0.003125152
0	0.994392037	0.005607983
1	0.082000196	0.917999804
0	0.980747581	0.019252392
0	0.111670792	0.888329208
1	0.006192982	0.993807018
1	0.002698720	0.997301280
0	0.996312141	0.003687858
0	0.965337813	0.034662213
0	0.965095460	0.034904540
1	0.003074586	0.996925414
0	0.995468318	0.004531683
0	0.990804672	0.009195301
1	0.021566689	0.978433311
0	0.994225919	0.005774109
1	0.005199194	0.994800806
0	0.990367115	0.009632888
0	0.933854520	0.066145495
0	0.933238626	0.066761374
0	0.995287836	0.004712182
0	0.992851913	0.007148073
0	0.989509702	0.010490319
0	0.996046543	0.003953473
1	0.076979578	0.923020422
1	0.102084875	0.897915125
1	0.011659980	0.988340020
1	0.004415810	0.995584190
0	0.333979487	0.666020513
1	0.051807821	0.948192179
0	0.196451485	0.803548515
0	0.946147144	0.053852841
0	0.995928228	0.004071767
0	0.945539057	0.054460917
1	0.003308177	0.996691823
0	0.972926259	0.027073767
1	0.004879534	0.995120466
1	0.002771556	0.997228444
0	0.992878377	0.007121624
1	0.003055513	0.996944487
0	0.932834148	0.067165881
1	0.004070818	0.995929182
0	0.985665262	0.014334713
0	0.978906095	0.021093879
0	0.650776029	0.349224001