The H2O Package

W. Evan Johnson, Ph.D.
Professor, Division of Infectious Disease
Director, Center for Data Science
Rutgers University – New Jersey Medical School

11/27/2023

Describing H2O

There is a lot of buzz for machine learning algorithms as well as a requirement for its experts. We all know that there is a significant gap in the skill requirement. The motive of H2O is to provide a platform which made easy for the non-experts to do experiments with machine learning.

Describing H2O

H2O's core code is written in Java that enables the whole framework for multi-threading. Although it is written in Java, it provides interfaces for R, Python and others, thus enabling it to be used efficiently.

In short, we can say that H2O is an open source, in memory, distributed, fast and scalable machine learning and predictive analytics toolkit that facilitates the building and application of machine learning models.

Using H2O

```
# install.packages("h2o")
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 https://docs.h2o.ai
##
## ----------------------------------------------------------------------

##
## Attaching package: 'h2o'

## The following objects are masked from 'package:lubridate':
##
##     day, hour, month, week, year

## 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,
```

Using H2O

```
h2o.init()
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         3 hours 38 minutes
##     H2O cluster timezone:       America/Denver
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.42.0.2
##     H2O cluster version age:    4 months and 2 days
##     H2O cluster name:           H2O_started_from_R_evan_oqw544
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   4.67 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
##     R Version:                  R version 4.3.2 (2023-10-31)

## Warning in h2o.clusterInfo():
## Your H2O cluster version is (4 months and 2 days) old. There may be a newer vers
```

Using H2O

Note: Initializing H2O might throw an error in your system in the case where you don't have Jdk of 64 bit. If such issue arises, please install latest Jdk of 64 bits here, it should work without issue afterward.

Using H2O

The h2o.init() command is pretty smart and does a lot of work. At first, it looks for any active H2O instance before starting a new one and then starts a new one when instance are not present.

It does have arguments which helps to accommodate resources to the H2O instance frequently used are:

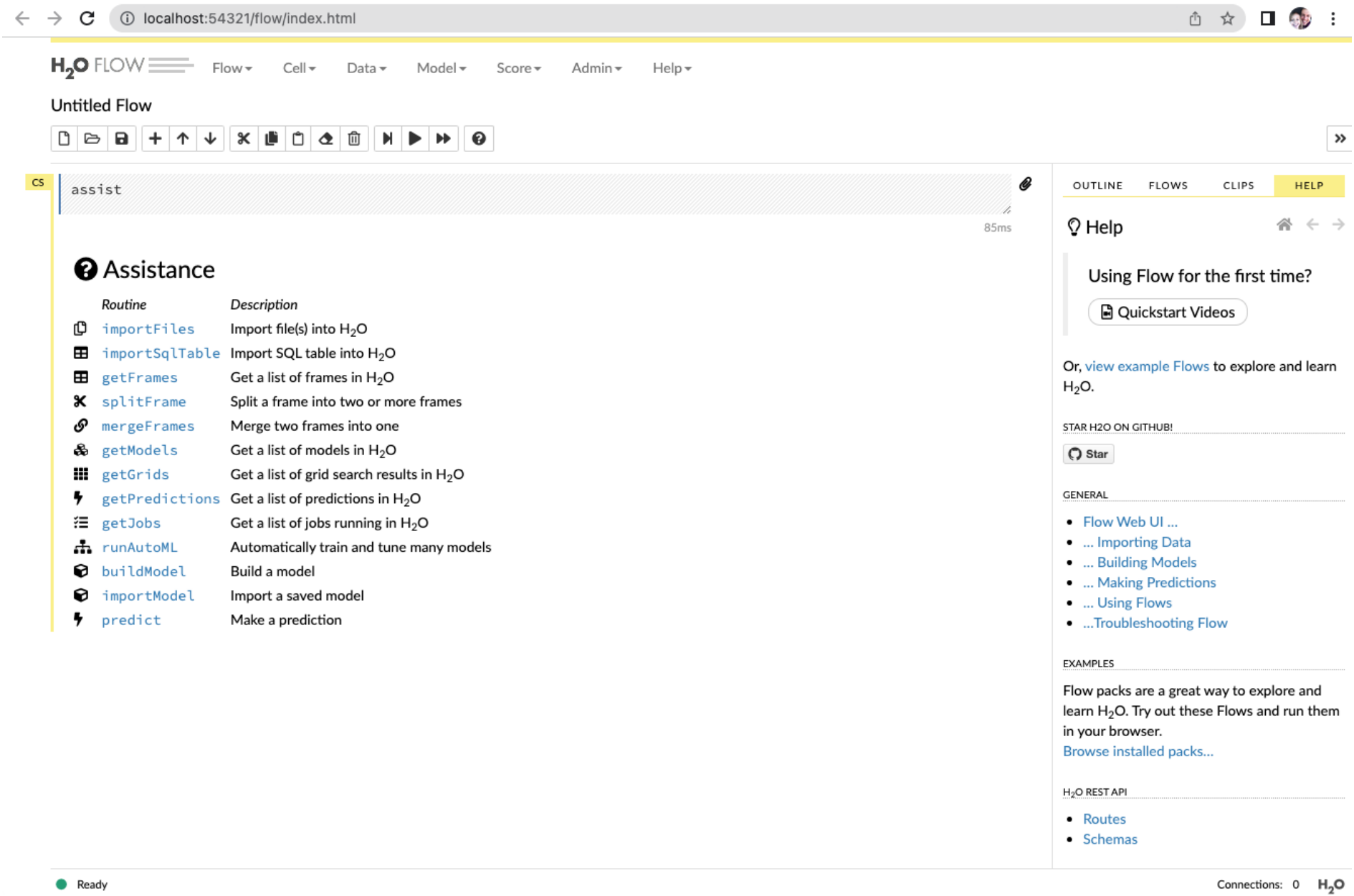
- **nthreads:** By default, the value of nthreads will be -1 which means the instance can use all the cores of the CPU, we can set the number of cores utilized by passing the value to the argument.
- **max_mem_size:** By passing a value to this argument you can restrict the maximum memory allocated to the instance. Its od string type can pass an argument as '2g' or '2G' for 2 GBs of memory, same when you want to allocate in MBs.

Using H2O

You can access the flow by typing http://localhost:54321 in your browser.

Flow is the name of the web interface that is part of H2O which does not require any extra installations which is written in CoffeeScript (a JavaScript like language). You can use it for doing the following things:

- Upload data directly
- View data uploaded by the client
- Create models directly
- View models created by you or your client
- view predictions
- Run predictions directly


localhost:54321/flow/index.html
H2O FLOW
Flow
Cell
Data
Model
Score
Admin
Help
Untitled Flow
CS
assist
85ms
Assistance
Routine
Description
importFiles
Import file(s) into H2O
importSqlTable
Import SQL table into H2O
getFrames
Get a list of frames in H2O
splitFrame
Split a frame into two or more frames
mergeFrames
Merge two frames into one
getModels
Get a list of models in H2O
getGrids
Get a list of grid search results in H2O
getPredictions
Get a list of predictions in H2O
getJobs
Get a list of jobs running in H2O
runAutoML
Automatically train and tune many models
buildModel
Build a model
importModel
Import a saved model
predict
Make a prediction
OUTLINE
FLOWS
CLIPS
HELP
Help
Using Flow for the first time?
Quickstart Videos
Or, view example Flows to explore and learn H2O.
STAR H2O ON GITHUB!
Star
GENERAL
Flow Web UI ...
... Importing Data
... Building Models
... Making Predictions
... Using Flows
...Troubleshooting Flow
EXAMPLES
Flow packs are a great way to explore and learn H2O. Try out these Flows and run them in your browser.
Browse installed packs...
H2O REST API
Routes
Schemas
Ready
Connections: 0
H2O

H2O AutoML

AutoML helps in automatic training and tuning of many models within a user-specified time limit.

The current version of AutoML function can train and cross-validate a Random Forest, an Extremely-Randomized Forest, a random grid of Gradient Boosting Machines (GBMs), a random grid of Deep Neural Nets, and then trains a Stacked Ensemble using all of the models.

When we say AutoML, it should cater to the aspects of data preparation, Model generation, and Ensembles and also provide few parameters as possible so that users can perform tasks with little confusion.

H2O AutoML

AutoML inputs required arguments **y** and the **training_frame**, with the **x** and **validation frame** as optional arguments. The user can also configure values for **max_runtime_sec** and **max_models**.

Additional optional parameters include:

- leaderboard_frame
- nfolds
- fold_columns
- weights_column
- ignored_columns
- stopping_metric
- sort_metric

H2O Kmeans on the iris data

```
iris_h2o <- as.h2o(iris)
```

```
##   |
```

```
iris_h2o['Species'] <- as.factor(iris_h2o['Species'])
predictors <- colnames(iris_h2o)[-length(iris_h2o)]

iris_splits <- h2o.splitFrame(data = iris_h2o,
                              ratios = 0.7, seed = 1234)
train <- iris_splits[[1]]
valid <- iris_splits[[2]]
```

H2O Kmeans on the iris data

```
kmeans_model <- h2o.kmeans(training_frame = train,
                           x = predictors, k = 3,
                           seed = 1)
```

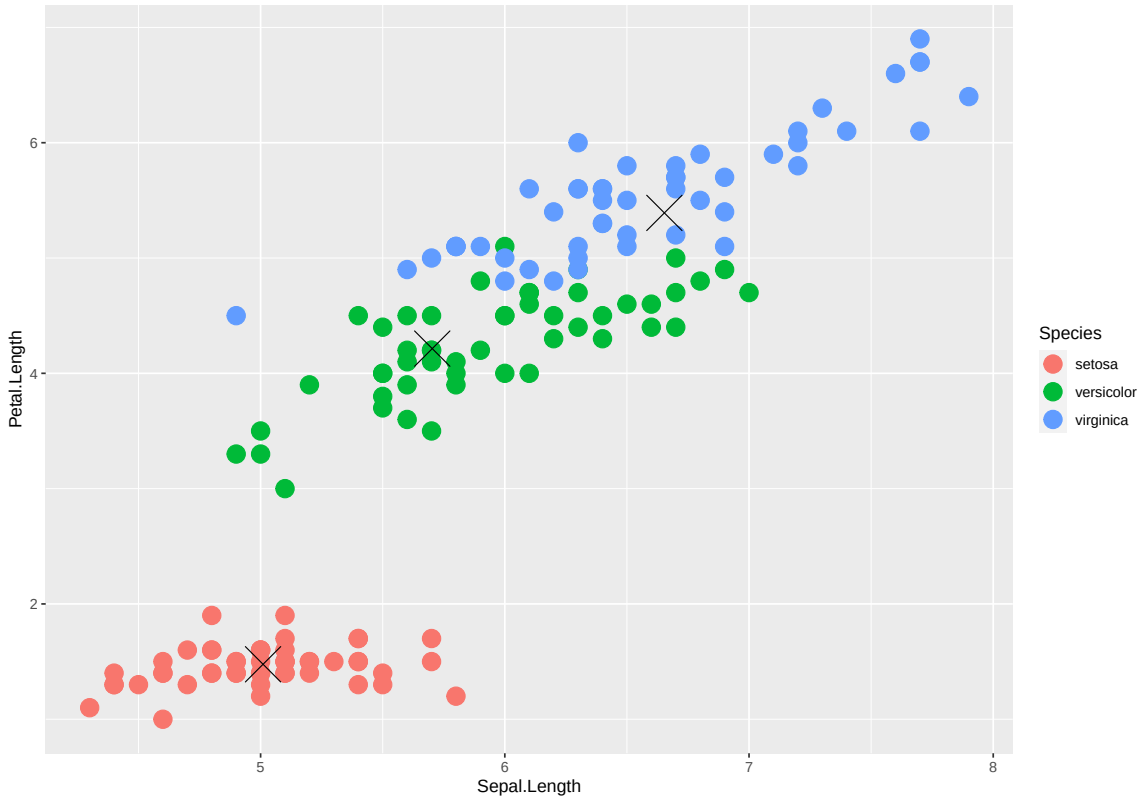
```
##   |
```

```
centers <- h2o.centers(kmeans_model)
centers
```

```
##   sepallength sepalwidth petallength petalwidth
## 1    5.702857   2.637143    4.214286   1.340000
## 2    6.653333   3.051111    5.391111   1.937778
## 3    5.010000   3.436667    1.476667   0.250000
```

H2O Kmeans on the iris data

```
iris %>% ggplot(aes(Sepal.Length,Petal.Length)) +
  geom_point(aes(col=Species), size=5) +
  geom_point(aes(sepallength,petallength),
             col=1, size=10, pch = 4, data=centers)
```



H2O AutoML on the iris data

```
iris_automl <- h2o.automl(x = predictors, y = "Species",
                          training_frame = train,
                          max_runtime_secs = 20, seed = 1,
                          validation_frame = valid)
```

```
##   |
## 11:40:11.415: User specified a validation frame with cross-validation still enabl
## 11:40:13.210: _min_rows param, The dataset size is too small to split for min_ro
```

```
iris_automl
```

```
## AutoML Details
## ==============
## Project Name: AutoML_30_20231127_114011
## Leader Model ID: GBM_4_AutoML_30_20231127_114011
## Algorithm: gbm
##
## Total Number of Models Trained: 17
## Start Time: 2023-11-27 11:40:11 UTC
## End Time: 2023-11-27 11:40:32 UTC
## Duration: 20 s
##
## Leaderboard
## ===========
```

H2O AutoML on the iris data

```
lb <- h2o.get_leaderboard(iris_automl)
head(lb)
```

```
##                                 model_id mean_per_class_error    logloss      rmse
## 1     GBM_4_AutoML_30_20231127_114011           0.02595453 0.11458244 0.1640160
## 2 XGBoost_3_AutoML_30_20231127_114011           0.02595453 0.15362192 0.1827537
## 3     XRT_1_AutoML_30_20231127_114011           0.02595453 0.10908611 0.1743630
## 4     DRF_1_AutoML_30_20231127_114011           0.02595453 0.12166596 0.1780753
## 5     GBM_2_AutoML_30_20231127_114011           0.02595453 0.10981022 0.1623032
## 6     GLM_1_AutoML_30_20231127_114011           0.02595453 0.05974883 0.1381892
##           mse
## 1 0.02690125
## 2 0.03339893
## 3 0.03040245
## 4 0.03171080
## 5 0.02634232
## 6 0.01909625
```

H2O AutoML on the iris data

```
iris_automl@leader
```

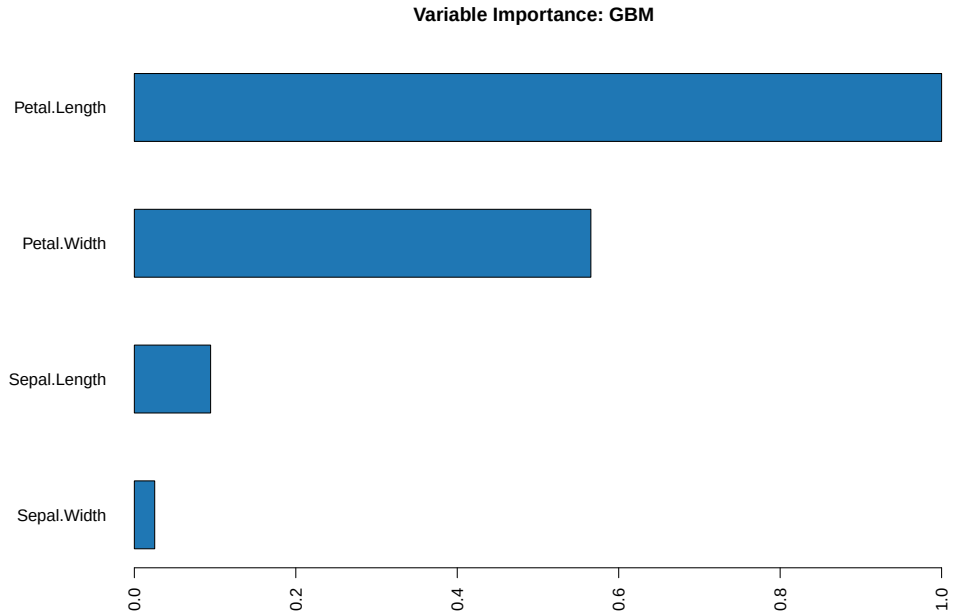
```
## Model Details:
## ==============
##
## H2OMultinomialModel: gbm
## Model ID:  GBM_4_AutoML_30_20231127_114011
## Model Summary:
##   number_of_trees number_of_internal_trees model_size_in_bytes min_depth
## 1             149                      447                64694         1
##   max_depth mean_depth min_leaves max_leaves mean_leaves
## 1         6    4.09620          2          8     6.82774
##
##
## H2OMultinomialMetrics: gbm
## ** Reported on training data. **
##
## Training Set Metrics:
## =====================
##
## Extract training frame with `h2o.getFrame("AutoML_30_20231127_114011_training_RT
## MSE: (Extract with `h2o.mse`) 3.313642e-07
## RMSE: (Extract with `h2o.rmse`) 0.0005756425
## Logloss: (Extract with `h2o.logloss`) 0.0001854509
```

H2O AutoML on the iris data

```
h2o.varimp(iris_automl@leader)
```

```
## Variable Importances:
##       variable relative_importance scaled_importance percentage
## 1 Petal.Length          187.907913          1.000000   0.593410
## 2  Petal.Width          106.264732          0.565515   0.335582
## 3 Sepal.Length           17.747568          0.094448   0.056047
## 4  Sepal.Width            4.737470          0.025212   0.014961
```

```
h2o.varimp_plot(iris_automl@leader)
```



H2O AutoML on the iris data

```
pred <- h2o.predict(iris_automl@leader, valid)
```

```
##   |
```

```
pred
```

```
##   predict    setosa   versicolor    virginica
## 1  setosa 1.0000000 3.973320e-09 1.010332e-09
## 2  setosa 0.9999999 5.665938e-08 2.272198e-08
## 3  setosa 0.9999999 7.060713e-08 2.775642e-08
## 4  setosa 0.9999998 2.053704e-07 2.294094e-09
## 5  setosa 1.0000000 5.701943e-09 9.974891e-10
## 6  setosa 0.9999992 8.326854e-07 2.641224e-09
##
## [40 rows x 4 columns]
```

H2O AutoML on the prostate cancer dataset

```
prostate_path <- system.file("extdata", "prostate.csv",
                             package = "h2o")
prostate <- h2o.importFile(path = prostate_path, header = TRUE)
```

```
##   |
```

```
y <- "CAPSULE"
prostate[,y] <- as.factor(prostate[,y])
knitr::kable(prostate)
```

ID	CAPSULE	AGE	RACE	DPROS	DCAPS	PSA	VOL	GLEASON
1	0	65	1	2	1	1.40	0.00	6
2	0	72	1	3	2	6.70	0.00	7
3	0	70	1	1	2	4.90	0.00	6
4	0	76	2	2	1	51.20	20.00	7
5	0	69	1	1	1	12.30	55.90	6
6	1	71	1	3	2	3.30	0.00	8
7	0	68	2	4	2	31.90	0.00	7
8	0	61	2	4	2	66.70	27.20	7
9	0	69	1	1	1	3.90	24.00	7
10	0	68	2	1	2	13.00	0.00	6
11	1	68	2	4	2	4.00	0.00	7
12	1	72	1	2	2	21.20	0.00	7

H2O AutoML on the prostate cancer dataset

```
aml <- h2o.automl(y = y, training_frame = prostate,
                  max_runtime_secs = 20, seed = 1)
```

```
##   |
```

```
lb <- h2o.get_leaderboard(aml)
head(lb)
```

```
##                                                 model_id       auc   logloss
## 1 StackedEnsemble_BestOfFamily_4_AutoML_31_20231127_114034 0.8165040 0.5169198
## 2           GBM_grid_1_AutoML_31_20231127_114034_model_11 0.8144309 0.5289923
## 3    StackedEnsemble_AllModels_1_AutoML_31_20231127_114034 0.8141142 0.5186176
## 4 StackedEnsemble_BestOfFamily_2_AutoML_31_20231127_114034 0.8133944 0.5173770
## 5            GBM_grid_1_AutoML_31_20231127_114034_model_4 0.8123290 0.5222005
## 6 StackedEnsemble_BestOfFamily_3_AutoML_31_20231127_114034 0.8116380 0.5213914
##       aucpr mean_per_class_error      rmse       mse
## 1 0.7310594            0.2336817 0.4129932 0.1705634
## 2 0.7171983            0.2298379 0.4190206 0.1755782
## 3 0.7300851            0.2301978 0.4151250 0.1723288
## 4 0.7338738            0.2346031 0.4150536 0.1722695
## 5 0.7152434            0.2413550 0.4169814 0.1738735
## 6 0.7306238            0.2335378 0.4165257 0.1734936
```

Session Info

```
sessionInfo()
```

```
## R version 4.3.2 (2023-10-31)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS Ventura 13.5.1
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib;  LAPACK version 3
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: America/Denver
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] h2o_3.42.0.2    lubridate_1.9.3 forcats_1.0.0   stringr_1.5.1
##  [5] dplyr_1.1.3     purrr_1.0.2     readr_2.1.4     tidyr_1.3.0
##  [9] tibble_3.2.1    tidyverse_2.0.0 caret_6.0-94    lattice_0.22-5
## [13] ggplot2_3.4.4
##
## loaded via a namespace (and not attached):
##  [1] gtable_0.3.4         xfun_0.41            recipes_1.0.8
##  [4] tzdb_0.4.0           bitops_1.0-7         vctrs_0.6.4
##  [7] tools_4.3.2          generics_0.1.3       curl_5.1.0
## [10] stats4_4.3.2         parallel_4.3.2       fansi_1.0.5
## [13] pkgconfig_2.0.3      ModelMetrics_1.2.2.2 Matrix_1.6-3
## [16] data.table_1.14.8    lifecycle_1.0.4      farver_2.1.1
```