AutoML: Automatic Machine Learning

In recent years, the demand for machine learning experts has outpaced the supply, despite the surge of people entering the field. To address this gap, there have been big strides in the development of user-friendly machine learning software that can be used by non-experts. The first steps toward simplifying machine learning involved developing simple, unified interfaces to a variety of machine learning algorithms (e.g. H2O).

Although H2O has made it easy for non-experts to experiment with machine learning, there is still a fair bit of knowledge and background in data science that is required to produce high-performing machine learning models. Deep Neural Networks in particular are notoriously difficult for a non-expert to tune properly. In order for machine learning software to truly be accessible to non-experts, we have designed an easy-to-use interface which automates the process of training a large selection of candidate models. H2O's AutoML can also be a helpful tool for the advanced user, by providing a simple wrapper function that performs a large number of modeling-related tasks that would typically require many lines of code, and by freeing up their time to focus on other aspects of the data science pipeline tasks such as data-preprocessing, feature engineering and model deployment.

H2O's AutoML can be used for automating the machine learning workflow, which includes automatic training and tuning of many models within a user-specified time-limit. Stacked Ensembles – one based on all previously trained models, another one on the best model of each family – will be automatically trained on collections of individual models to produce highly predictive ensemble models which, in most cases, will be the top performing models in the AutoML Leaderboard.

AutoML Interface

The H2O AutoML interface is designed to have as few parameters as possible so that all the user needs to do is point to their dataset, identify the response column and optionally specify a time constraint or limit on the number of total models trained.

In both the R and Python API, AutoML uses the same data-related arguments, `x`, `y`, `training_frame`, `validation_frame`, as the other H2O algorithms. Most of the time, all you'll need to do is specify the data arguments. You can then configure values for `max_runtime_secs` and/or `max_models` to set explicit time or number-of-model limits on your run.

Required Parameters

Required Data Parameters

- y: This argument is the name (or index) of the response column.
- training_frame: Specifies the training set.

Required Stopping Parameters

One of the following stopping strategies (time or number-of-model based) must be specified. When both options are set, then the AutoML run will stop as soon as it hits one of either of these limits.

- max_runtime_secs: This argument controls how long the AutoML will run at the most, before training the final Stacked Ensemble models. Defaults to 3600 seconds (1 hour).
- **max_models**: Specify the maximum number of models to build in an AutoML run, excluding the Stacked Ensemble models. Defaults to `NULL/None`.

Optional Parameters

Optional Data Parameters

- x: A list/vector of predictor column names or indexes. This argument only needs to be specified if the user wants to exclude columns from the set of predictors. If all columns (other than the response) should be used in prediction, then this does not need to be set.
- validation_frame: This argument is ignored unless `nfolds == 0`, in which a validation frame can be specified and used for early stopping of individual models and early stopping of the grid searches (unless `max_models` or `max_runtime_secs` overrides metric-based early stopping). By default and when `nfolds > 1`, cross-validation metrics will be used for early stopping and thus `validation_frame` will be ignored.
- **leaderboard_frame**: This argument allows the user to specify a particular data frame use to score & rank models on the leaderboard. This frame will not be used for anything besides leaderboard scoring. If a leaderboard frame is not specified by the user, then the leaderboard will use cross-validation metrics instead (or if cross-validation is turned off by setting `nfolds = 0`, then a leaderboard frame will be generated automatically from the validation frame (if provided) or the training frame).
- fold_column: Specifies a column with cross-validation fold index assignment per observation. This is used to override the default, randomized, 5-fold cross-validation scheme for individual models in the AutoML run.
- weights_column: Specifies a column with observation weights. Giving some observation a weight of zero is equivalent to excluding it from the dataset; giving an observation a relative weight of 2 is equivalent to repeating that row twice. Negative weights are not allowed.
- ignored_columns: (Optional, Python only) Specify the column or columns (as a list/vector) to be excluded from the model. This is the converse of the `x` argument.

Optional Miscellaneous Parameters

- nfolds: Number of folds for k-fold cross-validation of the models in the AutoML run. Defaults to 5. Use 0 to disable cross-validation; this will also disable Stacked Ensembles (thus decreasing the overall best model performance).
- balance_classes: Specify whether to oversample the minority classes to balance the class distribution. This option is not enabled by default and can increase the data frame size. This option is only applicable for classification. Majority classes can be undersampled to satisfy the **max_after_balance_size** parameter.
- class_sampling_factors: Specify the per-class (in lexicographical order) over/under-sampling ratios. By default, these ratios are automatically computed during training to obtain the class balance. Note that this requires `balance_classes=true`.
- max_after_balance_size: Specify the maximum relative size of the training data after balancing class counts (**balance_classes** must be enabled). Defaults to 5.0. (The value can be less than 1.0).
- stopping_metric: Specifies the metric to use for early stopping of the grid searches and individual models. Defaults to `"AUTO"`. The available options are:
    - `AUTO`: This defaults to `logloss` for classification, `deviance` for regression
    - `deviance` (mean residual deviance)
    - `logloss`
    - `MSE`
    - `RMSE`
    - `MAE`
    - `RMSLE`
    - `AUC`
    - `lift_top_group`
    - `misclassification`
    - `mean_per_class_error`

- stopping_tolerance: This option specifies the relative tolerance for the metric-based stopping criterion to stop a grid search and the training of individual models within the AutoML run. This value defaults to 0.001 if the dataset is at least 1 million rows; otherwise it defaults to a bigger value determined by the size of the dataset and the non-NA-rate. In that case, the value is computed as 1/sqrt(nrows * non-NA-rate).
- stopping_rounds: This argument is used to stop model training when the stopping metric (e.g. AUC) doesn't improve for this specified number of training rounds, based on a simple moving average. In the context of AutoML, this controls early stopping both within the random grid searches as well as the individual models. Defaults to 3 and must be an non-negative integer. To disable early stopping altogether, set this to 0.

- sort_metric: Specifies the metric used to sort the Leaderboard by at the end of an AutoML run. Available options include:
    - `AUTO`: This defaults to `AUC` for binary classification, `mean_per_class_error` for multinomial classification, and `deviance` for regression.
    - `deviance` (mean residual deviance)
    - `logloss`
    - `MSE`
    - `RMSE`
    - `MAE`
    - `RMSLE`
    - `AUC`
    - `mean_per_class_error`

- seed: Integer. Set a seed for reproducibility. AutoML can only guarantee reproducibility under certain conditions. H2O Deep Learning models are not reproducible by default for performance reasons, so if the user requires reproducibility, then `exclude_algos` must contain `"DeepLearning"`. In addition `max_models` must be used because `max_runtime_secs` is resource limited, meaning that if the available compute resources are not the same between runs, AutoML may be able to train more models on one run vs another. Defaults to `NULL/None`.
- **project_name**: Character string to identify an AutoML project. Defaults to `NULL/None`, which means a project name will be auto-generated based on the training frame ID. More models can be trained and added to an existing AutoML project by specifying the same project name in muliple calls to the AutoML function (as long as the same training frame is used in subsequent runs).
- **exclude_algos**: List/vector of character strings naming the algorithms to skip during the model-building phase. An example use is `exclude_algos = ["GLM", "DeepLearning", "DRF"]` in Python or `exclude_algos = c("GLM", "DeepLearning", "DRF")` in R. Defaults to `None/NULL`, which means that all appropriate H2O algorithms will be used, if the search stopping criteria allow. The algorithm names are:
    - `DRF` (This includes both the Random Forest and Extremely Randomized Trees (XRT) models. Refer to the Extremely Randomized Trees section in the DRF chapter and the histogram_type parameter description for more information.)
    - `GLM`
    - `XGBoost` (XGBoost GBM)
    - `GBM` (H2O GBM)
    - `DeepLearning` (Fully-connected multi-layer artificial neural network)
    - `StackedEnsemble`

- keep_cross_validation_predictions: Specify whether to keep the predictions of the cross-validation predictions. This needs to be set to TRUE if running the same AutoML object for repeated runs because CV predictions are required to build additional Stacked Ensemble models in AutoML. This option defaults to FALSE.
- keep_cross_validation_models: Specify whether to keep the cross-validated models. Keeping cross-validation models may consume significantly more memory in the H2O cluster. This option defaults to FALSE.
- keep_cross_validation_fold_assignment: Enable this option to preserve the cross-validation fold assignment. Defaults to FALSE.

Notes

If the user sets `nfolds == 0`, then cross-validation metrics will not be available to populate the leaderboard. In this case, we need to make sure there is a test frame (aka. the "leaderboard frame") to score the models on so that we can generate model performance metrics for the leaderboard. If the user does not specify the `leaderboard_frame` argument, AutoML will automatically split 10% of the training frame to use for scoring models on the leaderboard.

Code Examples

Here's an example showing basic usage of the `h2o.automl()` function in *R* and the `H2OAutoML` class in *Python*. For demonstration purposes only, we explicitly specify the the *x* argument, even though on this dataset, that's not required. With this dataset, the set of predictors is all columns other than

the response. Like other H2O algorithms, the default value of `x` is “all columns, excluding `y`”, so that will produce the same result.

r python

```r
library(h2o)

h2o.init()

# Import a sample binary outcome train/test set into H2O
train <- h2o.importFile("https://s3.amazonaws.com/erin-data/higgs/higgs_train_10k.csv")
test <- h2o.importFile("https://s3.amazonaws.com/erin-data/higgs/higgs_test_5k.csv")

# Identify predictors and response
y <- "response"
x <- setdiff(names(train), y)

# For binary classification, response should be a factor
train[,y] <- as.factor(train[,y])
test[,y] <- as.factor(test[,y])

# Run AutoML for 20 base models (limited to 1 hour max runtime by default)
aml <- h2o.automl(x = x, y = y,
                  training_frame = train,
                  max_models = 20,
                  seed = 1)

# View the AutoML Leaderboard
lb <- aml@leaderboard
print(lb, n = nrow(lb))  # Print all rows instead of default (6 rows)

#                                                 model_id       auc   logloss
mean_per_class_error      rmse       mse
# 1     StackedEnsemble_AllModels_AutoML_20181210_150447 0.7895453 0.5516022
0.3250365 0.4323464 0.1869234
# 2  StackedEnsemble_BestOfFamily_AutoML_20181210_150447 0.7882530 0.5526024
0.3239841 0.4328491 0.1873584
# 3                    XGBoost_1_AutoML_20181210_150447 0.7846510 0.5575305
0.3254707 0.4349489 0.1891806
# 4        XGBoost_grid_1_AutoML_20181210_150447_model_4 0.7835232 0.5578542
0.3188188 0.4352486 0.1894413
# 5        XGBoost_grid_1_AutoML_20181210_150447_model_3 0.7830043 0.5596125
0.3250808 0.4357077 0.1898412
# 6                    XGBoost_2_AutoML_20181210_150447 0.7813603 0.5588797
0.3470738 0.4359074 0.1900153
# 7                    XGBoost_3_AutoML_20181210_150447 0.7808475 0.5595886
0.3307386 0.4361295 0.1902090
# 8                        GBM_5_AutoML_20181210_150447 0.7808366 0.5599029
0.3408479 0.4361915 0.1902630
# 9                        GBM_2_AutoML_20181210_150447 0.7800361 0.5598060
0.3399258 0.4364149 0.1904580
# 10                       GBM_1_AutoML_20181210_150447 0.7798274 0.5608570
0.3350957 0.4366159 0.1906335
# 11                       GBM_3_AutoML_20181210_150447 0.7786685 0.5617903
0.3255378 0.4371886 0.1911339
# 12       XGBoost_grid_1_AutoML_20181210_150447_model_2 0.7744105 0.5750165
0.3228112 0.4427003 0.1959836
# 13                       GBM_4_AutoML_20181210_150447 0.7714260 0.5697120
0.3374203 0.4410703 0.1945430
# 14           GBM_grid_1_AutoML_20181210_150447_model_1 0.7697524 0.5725826
0.3443314 0.4424524 0.1957641
# 15           GBM_grid_1_AutoML_20181210_150447_model_2 0.7543664 0.9185673
0.3558550 0.4966377 0.2466490
# 16                       DRF_1_AutoML_20181210_150447 0.7428924 0.5958832
0.3554027 0.4527742 0.2050045
# 17                       XRT_1_AutoML_20181210_150447 0.7420910 0.5993457
0.3565826 0.4531168 0.2053148
# 18  DeepLearning_grid_1_AutoML_20181210_150447_model_2 0.7388505 0.6012286
0.3695292 0.4555318 0.2075092
# 19       XGBoost_grid_1_AutoML_20181210_150447_model_1 0.7257836 0.6013126
0.3820490 0.4565541 0.2084417
# 20               DeepLearning_1_AutoML_20181210_150447 0.6979292 0.6339217
0.3979403 0.4692373 0.2201836
# 21  DeepLearning_grid_1_AutoML_20181210_150447_model_1 0.6847773 0.6694364
0.4081802 0.4799664 0.2303678
# 22           GLM_grid_1_AutoML_20181210_150447_model_1 0.6826481 0.6385205
0.3972341 0.4726827 0.2234290
#
# [22 rows x 6 columns]



# The leader model is stored here
aml@leader

# If you need to generate predictions on a test set, you can make
# predictions directly on the `"H2OAutoML"` object, or on the leader
# model object directly

pred <- h2o.predict(aml, test)  # predict(aml, test) also works
```

```
# or:
pred <- h2o.predict(aml@leader, test)
```

The code above is the quickest way to get started, however to learn more about H2O AutoML we recommend taking a look at our more in-depth AutoML tutorial (available in R and Python).

AutoML Output

The AutoML object includes a "leaderboard" of models that were trained in the process, including the 5-fold cross-validated model performance (by default). The number of folds used in the model evaluation process can be adjusted using the `nfolds` parameter. If the user would like to score the models on a specific dataset, they can specify the `leaderboard_frame` argument, and then the leaderboard will show scores on that dataset instead.

The models are ranked by a default metric based on the problem type (the second column of the leaderboard). In binary classification problems, that metric is AUC, and in multiclass classification problems, the metric is mean per-class error. In regression problems, the default sort metric is deviance. Some additional metrics are also provided, for convenience.

Here is an example leaderboard for a binary classification task:

model_id	auc	logloss	mean_p
StackedEnsemble_AllModels_AutoML_20181212_105540	0.7898014	0.5511086	0.33317
StackedEnsemble_BestOfFamily_AutoML_20181212_105540	0.7884246	0.5521454	0.32319
XGBoost_1_AutoML_20181212_105540	0.7846510	0.5575305	0.32547
XGBoost_grid_1_AutoML_20181212_105540_model_4	0.7835232	0.5578542	0.31881
XGBoost_grid_1_AutoML_20181212_105540_model_3	0.7830043	0.5596125	0.32508
XGBoost_2_AutoML_20181212_105540	0.7813603	0.5588797	0.34707
XGBoost_3_AutoML_20181212_105540	0.7808475	0.5595886	0.33073
GBM_5_AutoML_20181212_105540	0.7808366	0.5599029	0.34084
GBM_2_AutoML_20181212_105540	0.7800361	0.5598060	0.33992
GBM_1_AutoML_20181212_105540	0.7798274	0.5608570	0.33509
GBM_3_AutoML_20181212_105540	0.7786685	0.5617903	0.32553
XGBoost_grid_1_AutoML_20181212_105540_model_2	0.7744105	0.5750165	0.32281
GBM_4_AutoML_20181212_105540	0.7714260	0.5697120	0.33742
GBM_grid_1_AutoML_20181212_105540_model_1	0.7697524	0.5725826	0.34433
GBM_grid_1_AutoML_20181212_105540_model_2	0.7543664	0.9185673	0.35585
DRF_1_AutoML_20181212_105540	0.7428924	0.5958832	0.35540
XRT_1_AutoML_20181212_105540	0.7420910	0.5993457	0.35658
DeepLearning_grid_1_AutoML_20181212_105540_model_2	0.7417952	0.6014974	0.36829
XGBoost_grid_1_AutoML_20181212_105540_model_1	0.6935538	0.6207021	0.40588
DeepLearning_1_AutoML_20181212_105540	0.6913704	0.6379538	0.40935
DeepLearning_grid_1_AutoML_20181212_105540_model_1	0.6900835	0.6617941	0.41846
GLM_grid_1_AutoML_20181212_105540_model_1	0.6826481	0.6385205	0.39723

Experimental Features

XGBoost

AutoML now includes XGBoost GBMs (Gradient Boosting Machines) among its set of algorithms. This feature is currently provided with the following restrictions:

- XGBoost is used only if it is available globally and if it hasn't been explicitly disabled.
- XGBoost is disabled by default in AutoML when running H2O-3 in multi-node due to current limitations. XGBoost can however be enabled experimentally in multi-node by setting the environment variable `-Dsys.ai.h2o.automl.xgboost.multinode.enabled=true` (when launching the H2O process from the command line) for every node of the H2O cloud.
- You can check if XGBoost is available by using the `h2o.xgboost.available()` in R or `h2o.estimators.xgboost.H2OXGBoostEstimator.available()` in Python.

FAQ

- **Which models are trained in the AutoML process?**

  The current version of AutoML trains and cross-validates the following algorithms (in the following order): A default Random Forest (DRF), an Extremely Randomized Forest (XRT), a fixed grid of GLMs, three pre-specified XGBoost GBM (Gradient Boosting Machine) models, five pre-specified H2O GBMs, a near-default Deep Neural Net, a random grid of XGBoost GBMs, a random grid of H2O GBMs, and lastly if there is time, a random grid of Deep Neural Nets. AutoML then trains two Stacked Ensemble models. Particular algorithms (or groups of algorithms) can be switched off using the `exclude_algos` argument. This is useful if you already have some idea of the algorithms that will do well on your dataset. As a recommendation, if you have really wide or sparse data, you may consider skipping the tree-based algorithms (GBM, DRF, XGBoost).

  A list of the hyperparameters searched over for each algorithm in the AutoML process is included in the appendix below. More details about the hyperparamter ranges for the models in addition to the hard-coded models will be added to the appendix at a later date.

  Both of the ensembles should produce better models than any individual model from the AutoML run with the exception of some rare cases. One ensemble contains all the models, and the second ensemble contains just the best performing model from each algorithm class/family. The "Best of Family" ensemble is optimized for production use since it only contains six (or fewer) base models. It should be relatively fast to use (to generate predictions on new data) without much degredation in model performance when compared to the "All Models" ensemble.

- **How do I save AutoML runs?**

  Rather than saving an AutoML object itself, currently, the best thing to do is to save the models you want to keep, individually. A utility for saving all of the models at once, along with a way to save the AutoML object (with leaderboard), will be added in a future release.

- **Why don't I see XGBoost models when using AutoML in a multi-node H2O cluster?**

  XGBoost is turned off by default for multi-node H2O clusters.

Resources

- AutoML Tutorial (R and Python notebooks)
- Intro to AutoML + Hands-on Lab (1 hour video) (slides)
- Scalable Automatic Machine Learning in H2O (1 hour video) (slides)
- AutoML Roadmap

Appendix: Random Grid Search Parameters

AutoML performs hyperparameter search over a variety of H2O algorithms in order to deliver the best model. In AutoML, the following hyperparameters are supported by grid search. Random Forest and Extremely Randomized Trees are not grid searched (in the current version of AutoML), so they are not included in the list below.

GLM Hyperparameters

- `alpha`
- `missing_values_handling`

XGBoost Hyperparameters

- `ntrees`
- `max_depth`
- `min_rows`
- `min_sum_hessian_in_leaf`
- `sample_rate`
- `col_sample_rate`
- `col_sample_rate_per_tree`
- `booster`
- `reg_lambda`
- `reg_alpha`

GBM Hyperparameters

- `histogram_type`
- `ntrees`
- `max_depth`
- `min_rows`
- `learn_rate`
- `sample_rate`
- `col_sample_rate`
- `col_sample_rate_per_tree`
- `min_split_improvement`

Deep Learning Hyperparameters

- `epochs`
- `adaptivate_rate`
- `activation`
- `rho`
- `epsilon`
- `input_dropout_ratio`
- `hidden`
- `hidden_dropout_ratios`

Additional Information

- AutoML development is tracked here. This page lists all open or in-progress AutoML JIRA tickets.
- AutoML is currently in experimental mode ("V99" in the REST API). This means that, although unlikely, the API (REST, R, Python or otherwise) may be subject to breaking changes.