**INTRODUCTION**

Data analysis is a powerful tool for learning insights on how to improve the decision making, business model and even products. This involves the construction and training of a machine learning model which faces several challenges due to lack of expert knowledge. This challenges can be overcomed by using automated machine learning(AutoML) field. AutoML refers to the process of studying a traditional machine learning model development pipeline to segment it into modules and automate each of those to accelerate workflow. With the advent of deeper models, such as the ones used in image processing, Natural Language Processing, etc., there is an increasing need for tailored models that can be crafted for specific workloads. However, such specific models require immense resources such as high capacity memory, strong GPUs, domain experts to help during the development and long wait times during training. The task gets critical as there is not much work done for creating a formal framework for deciding model parameters without the need for trial and error. These nuances emphasized the need for AutoML where automation can reduce turnaround times and also increase the accuracy of the derived models by removing human errors. In recent years, several tools and models have been proposed in the domain of AutoML. Some of these focus on particular segments of AutoML such as feature engineering or model selection, whereas some models attempt to optimize the complete pipeline. These tools have matured enough to be able to compare with human experts on Kaggle competitions and at times have beat them as well, showcasing their veracity.There are wide variety of applications based on AutoML such as autonomic cloud computing, Intelligent Vehicular networks, Block Chain,Software Defined Networking, among others. This paper aims at providing an overview of the advances seen in the realm of AutoML in recent years. We focus on individual aspects of AutoML and summarize the improvements achieved in recent years. The motivation of this paper stems from the unavailability of a compact study of the current state of AutoML. While we acknowledge the existence of other surveys, their motive is to either provide an in-depth understanding of a particular segment of AutoML, provide just an experimental comparison of various tools used or are fixated towards deep learning models.

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.

H2O architecture can be divided into different layers in which the top layer will be different APIs, and the bottom layer will be H2O JVM.

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 few others shown in the architecture, thus enabling it to be used efficiently.

In crux, we can say that H2O is an open source, in memory, distributed, fast and scalable machine learning and predictive analytics that allow building machine learning models to be an ease.

If you are using python the same method is applied in it too, from command line pip install -U h2o and h2o will be installed for your python environment. The same process will go on for Initializing 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.

Once instance is created, 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

**AutoML**

Now talking about AutoML part of H2O, 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 much less confusion. H2o AutoML does perform this task with ease and the minimal parameter passed by the user.

In both R and Python API, it uses the same data related arguments x, y, training\_frame, validation frame out of which y and training\_frame are required parameter and rest are optional. You can also configure values for max\_runtime\_sec and max\_models here max\_runtime\_sec parameter is required, and max\_model is optional if you don’t pass any parameter it takes NULL by default.

The x parameter is the vector of predictors from training\_frame if you don’t want to use all predictors from the frame you passed you can set it by passing it to x.

Now let's talk about some optional and miscellaneous parameters, try to tweak the parameters even if you don’t know about it, it will lead you to gain knowledge over some advanced topics:

validation\_frame: This parameter is used for early stopping of individual models in the automl. It is a dataframe that you pass for validation of a model or can be a part of training data if not passed by you.

leaderboard\_frame: If passed the models will be scored according to the values instead of using cross-validation metrics. Again the values are a part of training data if not passed by you.

nfolds: K-fold cross-validation by default 5, can be used to decrease the model performance.

fold\_columns: Specifies the index for cross-validation.

weights\_column: If you want to provide weights to specific columns you can use this parameter, assigning weight 0 means you are excluding the column.

ignored\_columns: Only in python, it is converse of x.

stopping\_metric: Specifies a metric for early stopping of the grid searches and models default value is logloss for classification and deviation for regression.

sort\_metric: The parameter to sort the leaderboard models at the end. This defaults to AUC for binary classification, mean\_per\_class\_error for multinomial classification, and deviance for regression.

The validation\_frame and leaderboard\_frame depend on the cross-validation parameter that is nfolds.

The following scenarios can generate in two cases:

when we are using cross-validation in the automl:

\* Only training frame is passed - Then data will split into 80-20 of training and validation frame.

\* training and leaderboard frame is passed - No change in the 80-20 split of data in training and validation frame.

\* When training and validation frame is passed - No split.

\* when all three frames are passed - No splits.

When we are not using cross-validation which will affect the leaderboard frame a lot(nfolds = 0):

\* Only training frame is passed - The data is split into 80/10/10 training, validation, and leaderboard.

\* training and leaderboard frame is passed - Data split into 80-20 of training and validation frames.

\* When training and validation frame is passed - The validation\_frame data is split into 50-50 validation and leaderboard.

\* when all three frames are passed - No splits.