

Democratising Machine learning with H2O

Overview of H2O: the open source, distributed in-memory machine learning platform



Parul Pandey

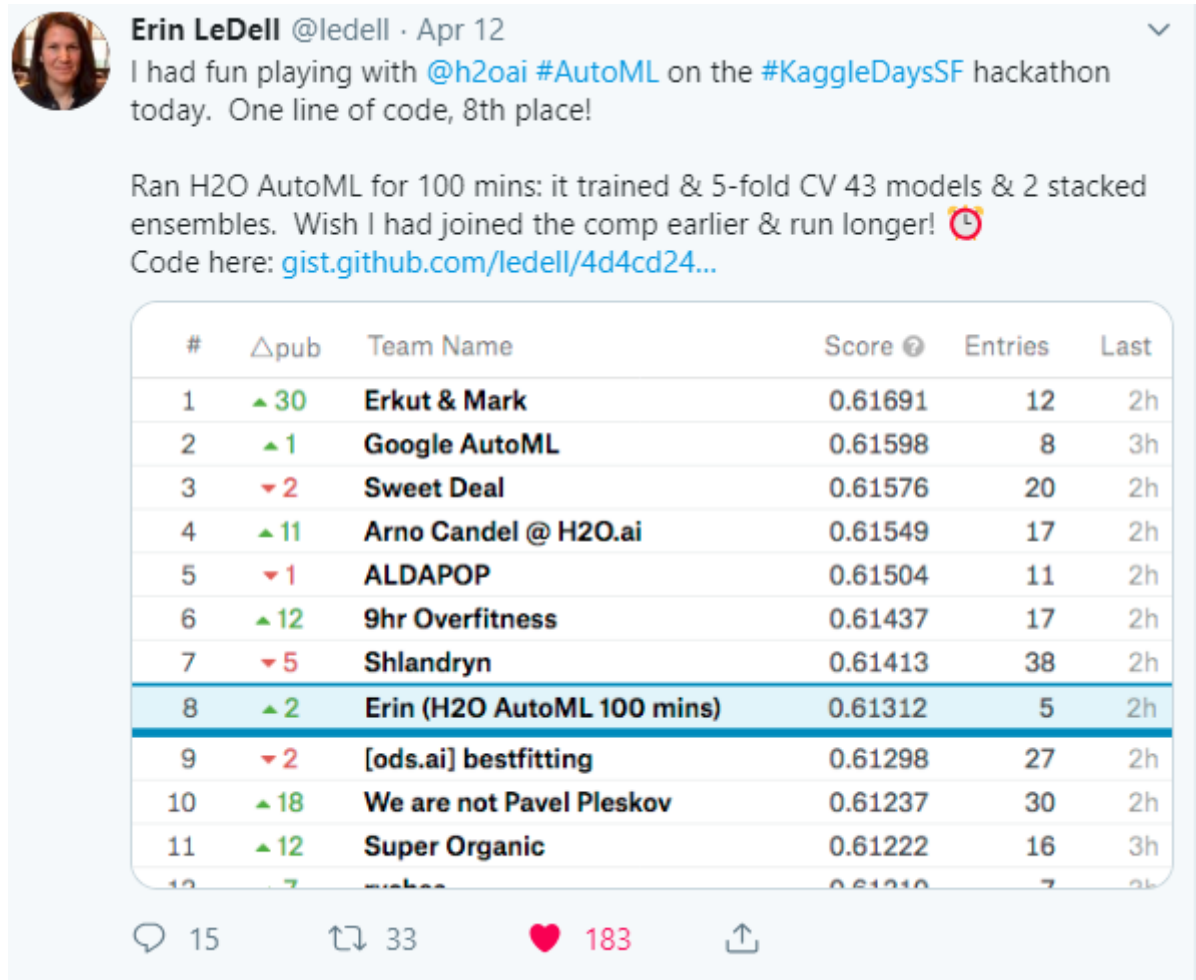
Apr 20 · 10 min read



Photo by Pixabay from Pexels

It is important to make AI accessible to everyone for the sake of social and economic stability.

Kaggle days is a two-day event where data science enthusiasts can talk to each other face to face, exchange knowledge, and compete together. Kaggle days San Francisco just concluded and as is customary, Kaggle also organised a hackathon for the participants. I had been following Kaggle days on Twitter and the following tweet from **Erin LeDell** (Chief Machine Learning Scientist at H2O.ai) caught my eye.



Erin LeDell @ledell · Apr 12

I had fun playing with @h2oai #AutoML on the #KaggleDaysSF hackathon today. One line of code, 8th place!

Ran H2O AutoML for 100 mins: it trained & 5-fold CV 43 models & 2 stacked ensembles. Wish I had joined the comp earlier & run longer! 🕒

Code here: gist.github.com/ledell/4d4cd24...

#	Δpub	Team Name	Score 🏆	Entries	Last
1	▲30	Erkut & Mark	0.61691	12	2h
2	▲1	Google AutoML	0.61598	8	3h
3	▼2	Sweet Deal	0.61576	20	2h
4	▲11	Arno Candel @ H2O.ai	0.61549	17	2h
5	▼1	ALDAPOP	0.61504	11	2h
6	▲12	9hr Overfitness	0.61437	17	2h
7	▼5	Shlandryn	0.61413	38	2h
8	▲2	Erin (H2O AutoML 100 mins)	0.61312	5	2h
9	▼2	[ods.ai] bestfitting	0.61298	27	2h
10	▲18	We are not Pavel Pleskov	0.61237	30	2h
11	▲12	Super Organic	0.61222	16	3h

15 replies, 33 retweets, 183 likes

Source: Twitter

I have been experimenting with H2O for quite some time and found it really seamless and intuitive for solving ML problems. Seeing it perform so well on Leaderboard, I thought it was time that I wrote an article on the same to make it easy for others to make a transition into the world of H2O.

...

H2O.ai: The company behind H2O

H2O.ai is based in Mountain View, California and offers a suite of Machine Learning platforms. H2O's core strength is its high-performing ML components, which are

tightly integrated. H2O.ai is a Visionary in the Gartner Magic Quadrant for Data Science Platforms in its report released in Jan'2019.



Let's take a brief look at the offerings of H2O.ai:





H2O.ai Products and Solutions

H2O

H2O is an open-source, distributed in-memory machine learning platform with linear scalability. H2O supports the most widely used statistical & machine learning algorithms and also has an AutoML functionality. H2O's core code is written in Java and its REST API allows access to all the capabilities of H2O from an external program or script. The platform includes interfaces for R, Python, Scala, Java, JSON and CoffeeScript/JavaScript, along with a built-in web interface, Flow,

Since the main focus of this article is about H2O, we shall get to know more about it later in the article.

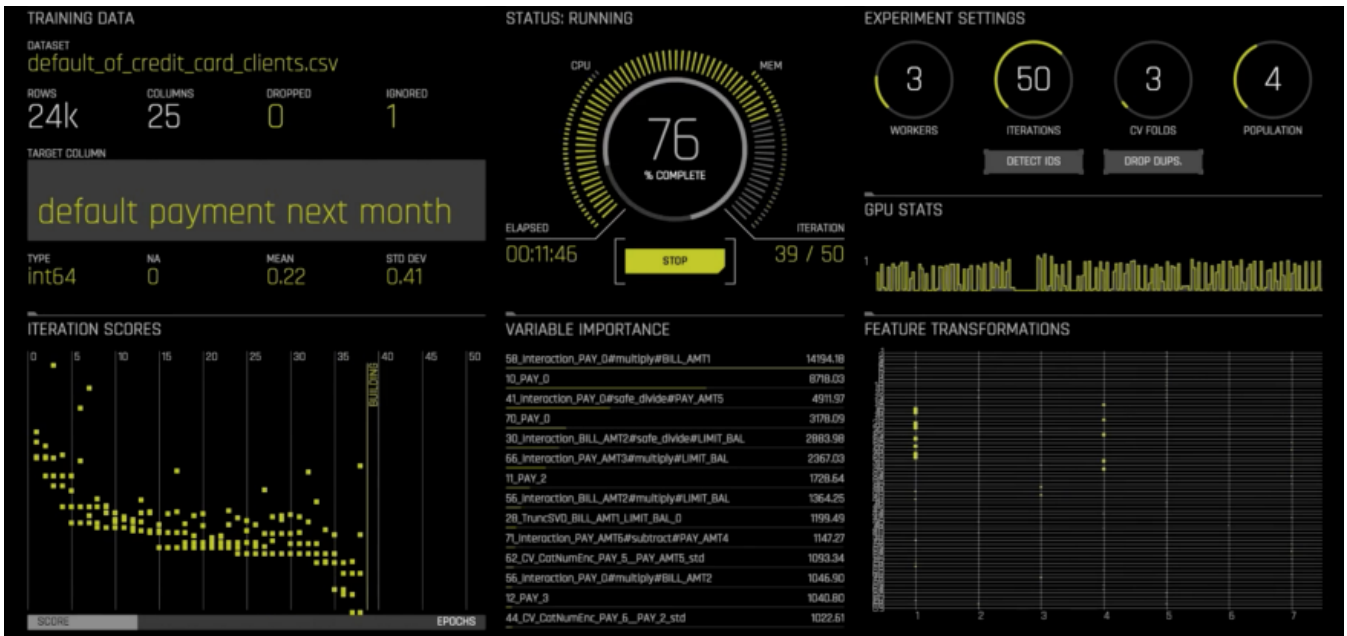
H2O Sparkling Water

Sparkling Water allows users to combine the fast, scalable machine learning algorithms of H2O with the capabilities of Spark. Sparkling Water is ideal for H2O users who need to manage large clusters for their data processing needs and want to transfer data from Spark to H2O (or vice versa).

H2O4GPU

H2O4GPU is an open-source, GPU-accelerated machine learning package with APIs in Python and R that allows anyone to take advantage of GPUs to build advanced machine learning models.

H2O Driverless AI



Driverless AI's UI

H2O Driverless AI is H2O.ai's flagship product for automatic machine learning. It fully automates some of the most challenging and productive tasks in applied data science such as feature engineering, model tuning, model ensembling and model deployment. With Driverless AI, data scientists of all proficiency levels can train and deploy modelling pipelines with just a few clicks from the GUI. Driverless AI is a commercially licensed product with a 21-day free trial version.

...

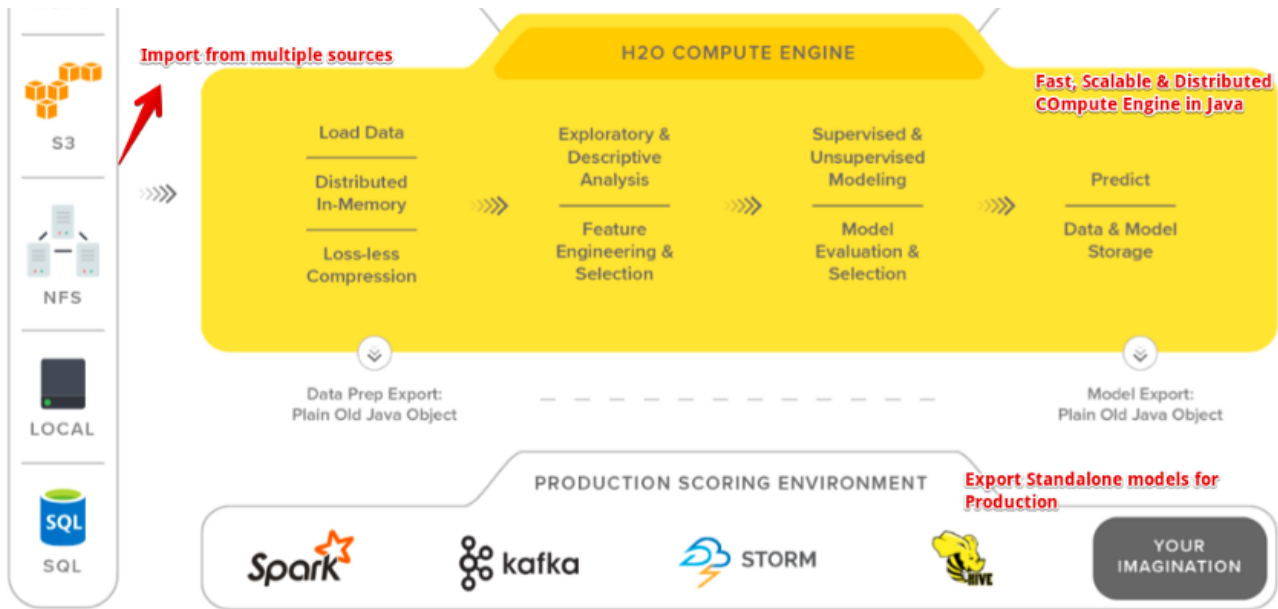
What is H2O

The latest version called H2O-3 is the third incarnation of H2O. H2O uses familiar interfaces like R, Python, Scala, Java, JSON and the Flow notebook/web interface, and works seamlessly with big data technologies like Hadoop and Spark. H2O can easily and quickly derive insights from the data through faster and better predictive modelling.

High-Level Architecture

H2O makes it possible to import data from multiple sources and has a fast, Scalable & Distributed Compute Engine Written in Java. Here is a high-level overview of the platform.





A High-Level architecture of h2o

Supported Algorithms

H2O supports a lot of commonly used algorithms of Machine Learning.

Supervised Learning

Statistical Analysis

- **Generalized Linear Models:** Binomial, Gaussian, Gamma, Poisson and Tweedie
- **Naïve Bayes**

Ensembles

- **Distributed Random Forest:** Classification or regression models
- **Gradient Boosting Machine:** Produces an ensemble of decision trees with increasing refined approximations

Deep Neural Networks

- **Deep learning:** Create multi-layer feed forward neural networks starting with an input layer followed by multiple layers of nonlinear transformations

Unsupervised Learning

Clustering

- **K-means:** Partitions observations into k clusters/groups of the same spatial size. Automatically detect optimal k

Dimensionality Reduction

- **Principal Component Analysis:** Linearly transforms correlated variables to independent components
- **Generalized Low Rank Models:** extend the idea of PCA to handle arbitrary data consisting of numerical, Boolean, categorical, and missing data

Anomaly Detection

- **Autoencoders:** Find outliers using a nonlinear dimensionality reduction using deep learning

Algorithms supported by H2O

Installation

H2O offers an **R package** that can be installed from CRAN and a **python package** that can be installed from PyPI. In this article, I shall be working with only the Python implementation. Also, you may want to look at the documentation for complete details.

Pre-requisites

- Python

- Java 7 or later, which you can get at the [Java download page](#). To build H2O or run H2O tests, the 64-bit JDK is required. To run the H2O binary using either the command line, R or Python packages, only 64-bit JRE is required.

Dependencies :

```
pip install requests
pip install tabulate
pip install "colorama>=0.3.8"
pip install future
```

- pip install

```
pip install -f http://h2o-
release.s3.amazonaws.com/h2o/latest_stable_Py.html h2o
```

- conda

```
conda install -c h2oai h2o=3.22.1.2
```

Note: When installing H2O from `pip` in OS X El Capitan, users must include the `--user` flag. For example -

```
pip install -f http://h2o-
release.s3.amazonaws.com/h2o/latest_stable_Py.html h2o --user
```

For R installation please refer to the official documentation [here](#).

Testing installation

Every new python session begins by initializing a connection between the python client and the H2O cluster. A cluster is a group of H2O nodes that work together; when a job is submitted to a cluster, all the nodes in the cluster work on a portion of the job.

To check if everything is in place, open your Jupyter Notebooks and type in the following:


```
import h2o
h2o.init()
```

This is a local H2O cluster. On executing the cell, some information will be printed on the screen in a tabular format displaying amongst other things, the number of nodes, total memory, Python version etc. In case you need to report a bug, make sure you include all this information. Also, the `h2o.init()` makes sure that no prior instance of H2O is running.

```
Checking whether there is an H2O instance running at http://localhost:54321 ..... not found.
Attempting to start a local H2O server...
; Java HotSpot(TM) 64-Bit Server VM (build 25.201-b09, mixed mode)
Starting server from c:\users\parul\appdata\local\programs\python\python37\lib\site-packages\h2o\backend\bin\h2o.jar
Ice root: C:\Users\Parul\AppData\Local\Temp\tmp24io0168
JVM stdout: C:\Users\Parul\AppData\Local\Temp\tmp24io0168\h2o_Parul_started_from_python.out
JVM stderr: C:\Users\Parul\AppData\Local\Temp\tmp24io0168\h2o_Parul_started_from_python.err
Server is running at http://127.0.0.1:54321
Connecting to H2O server at http://127.0.0.1:54321 ... successful.
```

H2O cluster uptime:	04 secs
H2O cluster timezone:	Asia/Kolkata
H2O data parsing timezone:	UTC
H2O cluster version:	3.24.0.1
H2O cluster version age:	15 days
H2O cluster name:	H2O_from_python_Parul_uc80wu
H2O cluster total nodes:	1
H2O cluster free memory:	1.757 Gb
H2O cluster total cores:	4
H2O cluster allowed cores:	4
H2O cluster status:	accepting new members, healthy
H2O connection url:	http://127.0.0.1:54321
H2O connection proxy:	None
H2O internal security:	False
H2O API Extensions:	Amazon S3, Algos, AutoML, Core V3, Core V4
Python version:	3.7.2 final

Running `h2o.init()` (in Python)

By default, H2O instance uses all the cores and about 25% of the system's memory. However, in case you wish to allocate it a fixed chunk of memory, you can specify it in the `init` function. Let's say we want to give the H2O instance 4GB of memory and it should only use 2 cores.

```
#Allocate resources
```

```
h2o.init(nthreads=2,max_mem_size=4)
```

H2O cluster free memory:	3.556 Gb
H2O cluster total cores:	4
H2O cluster allowed cores:	2

Now our H2O instance is using only 2 cores and around 4GB of memory. However, we will go with the default method.

. . .

Importing Data with H2O in Python

After the installation is successful, it's time to get our hands dirty by working on a real-world dataset. We will be working on a Regression problem using the famous wine dataset. The task here is to *predict the quality of white wine* on a scale of 0–10 given a set of features as inputs.

Here is a link to the Github Repository in case you want to follow along or you can view it on my binder by clicking the image below.



Data

The data belongs to the white variants of the Portuguese “Vinho Verde” wine.

- **Source:** <https://archive.ics.uci.edu/ml/datasets/Wine+Quality>
- **CSV File :** (<https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv>)

Data Import

Importing data from a local CSV file. The command is very similar to `pandas.read_csv` and the data is stored in memory as a H2OFrame.

```
wine_data = h2o.import_file("winequality-white.csv")
wine_data.head(5) # The default head() command displays the first 10
rows.
```

fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
---------------	------------------	-------------	----------------	-----------	---------------------	----------------------	---------	----	-----------	---------	---------

acidity	acidity	acidity	sugar	acidity	acidity	acidity	acidity	acidity	acidity	acidity	acidity
7	0.27	0.36	20.7	0.045	45	170	1.001	3	0.45	8.8	6
6.3	0.3	0.34	1.6	0.049	14	132	0.994	3.3	0.49	9.5	6
8.1	0.28	0.4	6.9	0.05	30	97	0.9951	3.26	0.44	10.1	6
7.2	0.23	0.32	8.5	0.058	47	186	0.9956	3.19	0.4	9.9	6
7.2	0.23	0.32	8.5	0.058	47	186	0.9956	3.19	0.4	9.9	6

Displaying the first 5 rows of the dataset

EDA

Let us explore the dataset to get some insights.

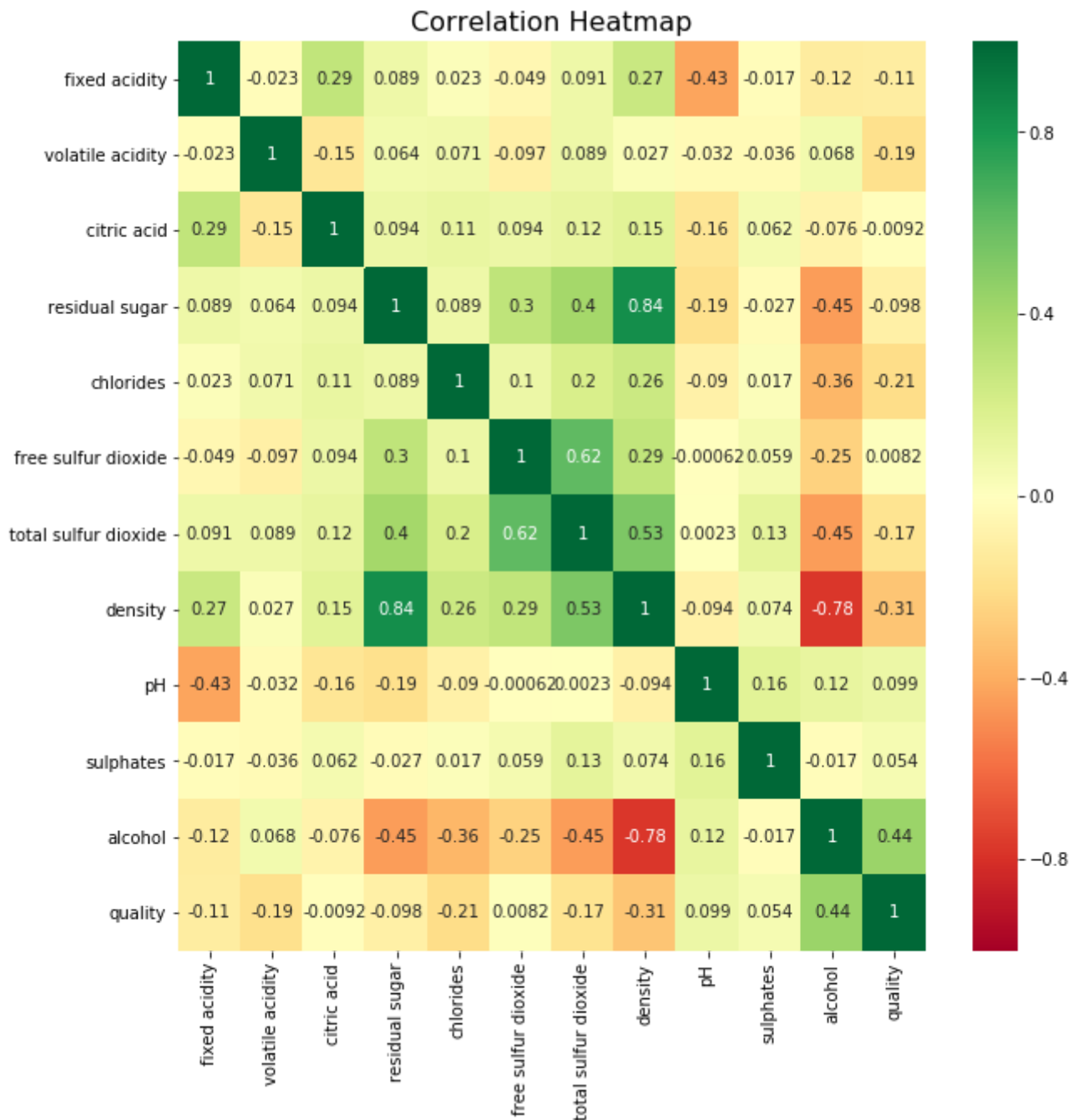
```
wine_data.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide
type	real	real	real	real	real	real	real
mins	3.8	0.08	0.0	0.6	0.009	2.0	9.0
mean	6.854787868438097	0.27824111882400978	0.33419150673744388	6.391414863209474	0.045772358063699476	35.30808493262556	138.38065741118824
maxs	14.2	1.1	1.66	85.8	0.346	289.0	440.0
sigma	0.8438682278875131	0.10079454842486535	0.1210198042029825	5.072057784014881	0.021847968093728794	17.007137325232588	42.498064554142914
zeros	0	0	19	0	0	0	0
missing	0	0	0	0	0	0	0
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0
1	6.3	0.3	0.34	1.6	0.049	14.0	132.0
2	8.1	0.28	0.4	6.9	0.05	30.0	97.0
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0
5	8.1	0.28	0.4	6.9	0.05	30.0	97.0
6	6.2	0.32	0.16	7.0	0.045	30.0	136.0
7	7.0	0.27	0.36	20.7	0.045	45.0	170.0
8	6.3	0.3	0.34	1.6	0.049	14.0	132.0
9	8.1	0.22	0.43	1.5	0.044	28.0	129.0

Exploring some of the columns of the dataset

All the features here are numbers and there aren't any categorical variables. Now let us also look at the correlation of the individual features.

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(10,10))
corr = wine_data.corr().as_data_frame()
corr.index = wine_data.columns
sns.heatmap(corr, annot = True, cmap='RdYlGn', vmin=-1, vmax=1)
plt.title("Correlation Heatmap", fontsize=16)
plt.show()
```



Modeling with H2O

We shall build a regression model to predict the **Quality** of the wine. There are a lot of algorithms available in the H2O module both for Classification as well as Regression problems.

Splitting data into Test and Training sets

Since we have only one dataset, let's split it into training and Testing part, so that we can evaluate the model's performance. We shall use the `split_frame()` function.

```
wine_split = wine_data.split_frame(ratios = [0.8], seed = 1234)

wine_train = wine_split[0] # using 80% for training
wine_test = wine_split[1] #rest 20% for testing

print(wine_train.shape, wine_test.shape)
(3932, 12) (966, 12)
```

Defining Predictor Variables

```
predictors = list(wine_data.columns)
predictors.remove('quality') # Since we need to predict quality
predictors
```

```
Out[43]: ['fixed acidity',
          'volatile acidity',
          'citric acid',
          'residual sugar',
          'chlorides',
          'free sulfur dioxide',
          'total sulfur dioxide',
          'density',
          'pH',
          'sulphates',
          'alcohol']
```

Generalized Linear Model

We shall build a Generalized Linear Model (GLM) with default settings. Generalized Linear Models (GLM) estimate regression models for outcomes following exponential distributions. In addition to the Gaussian (i.e. normal) distribution, these include Poisson, binomial, and gamma distributions. You can read more about GLM in the documentation.

```
# Import the function for GLM
from h2o.estimators.glm import H2OGeneralizedLinearEstimator

# Set up GLM for regression
glm = H2OGeneralizedLinearEstimator(family = 'gaussian', model_id =
'glm_default')

# Use .train() to build the model
glm.train(x = predictors,
          y = 'quality',
          training_frame = wine_train)

print(glm)
```


glm prediction progress:  100%

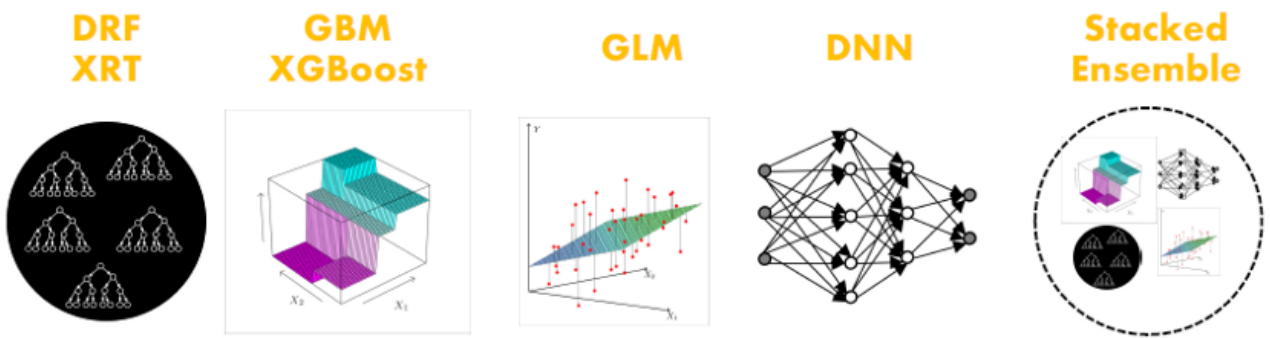
predict
5.76109
5.76721
5.64325
5.85764
5.77967

Similarly, you could use other supervised algorithms like **Distributed Random Forest**, **Gradient Boosting Machines**, and even **Deep Learning**.you could also tune in the hyperparameters.

. . .

H2OAutoML: Automatic Machine Learning

Automated machine learning (AutoML) is the process of automating the end-to-end process of applying machine learning to real-world problems. AutoML makes machine learning available in a true sense, even to people with no major expertise in this field. H2O's AutoML tends to automate the training and the tuning part of the models.



In this section, we shall be using the AutoML capabilities of H2O to work on the same regression problem of predicting wine quality.

Importing the AutoML Module

```
from h2o.automl import H2OAutoML
aml = H2OAutoML(max_models = 20, max_runtime_secs=100, seed = 1)
```

Here AutoML will run for 20 base models for 100 seconds. The default runtime is 1 Hour.

Training

```
aml.train(x=predictors, y='quality', training_frame=wine_train,
validation_frame=wine_test)
```

Leaderboard

Now let us look at the automl leaderboard.

```
print(aml.leaderboard)
```

	model_id	mean_residual_deviance	rmse	mse	mae	rmsle
	StackedEnsemble_BestOfFamily_AutoML_20190418_113327	0.392908	0.626823	0.392908	0.455143	0.0939202
	StackedEnsemble_AllModels_AutoML_20190418_113327	0.392992	0.626891	0.392992	0.455512	0.0939226
	XRT_1_AutoML_20190418_113327	0.404961	0.636366	0.404961	0.463282	0.0953964
	DRF_1_AutoML_20190418_113327	0.413018	0.642664	0.413018	0.466174	0.0963974
	GBM_4_AutoML_20190418_113327	0.417493	0.646137	0.417493	0.486555	0.0966819
	GBM_3_AutoML_20190418_113327	0.426668	0.653198	0.426668	0.499098	0.097667
	GBM_1_AutoML_20190418_113327	0.435121	0.659637	0.435121	0.505029	0.0984655
	GBM_2_AutoML_20190418_113327	0.440507	0.663707	0.440507	0.512223	0.0990434
	GBM_5_AutoML_20190418_113327	0.473033	0.687774	0.473033	0.533694	0.102412
	DeepLearning_1_AutoML_20190418_113327	0.52709	0.72601	0.52709	0.565672	0.108326

AutoML Leaderboard

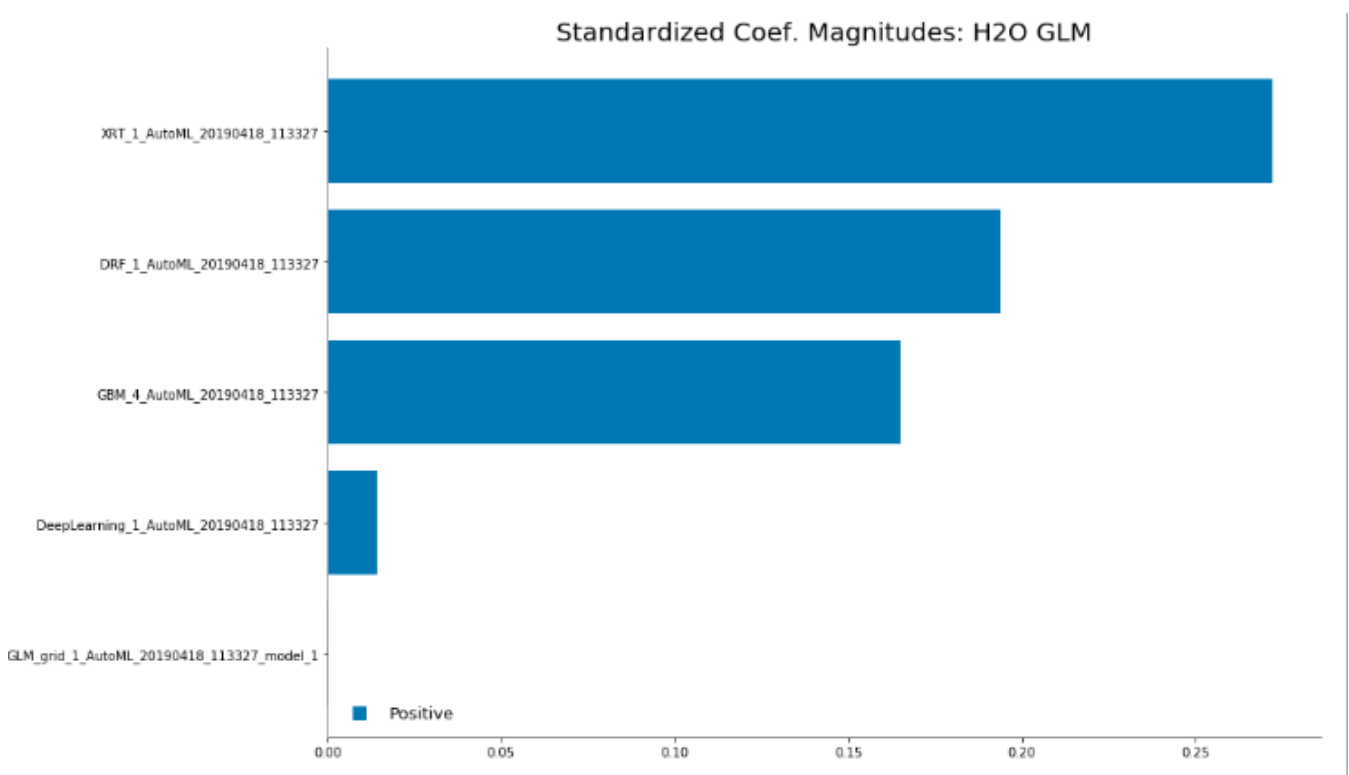
The leaderboard displays the top 10 models built by AutoML with their parameters. The best model is placed on the top is a Stacked Ensemble.

The leader model is stored as `aml.leader`

Contribution of Individual Models

Let us look at the contribution of the individual models for this meta-learner.

```
metalearner = h2o.get_model(aml.leader.metalearner()['name'])
metalearner.std_coef_plot()
```



XRT(Extremely Randomized Trees) has the maximum contribution followed by Distributed Random Forests.

Predictions

```
preds = aml.leader.predict(wine_test)
```

The code above is the quickest way to get started, however, to learn more about H2O AutoML it is worth taking a look at the in-depth AutoML tutorial (available in R and

Python).

Shutting Down

```
h2o.shutdown()
```

. . .

Using Flow — H2O's Web UI

In the final leg of this article, let us have a quick overview of H2O's open source Web UI called **Flow**. Flow is a web-based interactive computational environment where you can combine code execution, text, mathematics, plots and rich media into a single document, much like Jupyter Notebooks.

Launching Flow

Once H2O is up and running all you need to do is point your browser to <http://localhost:54321> and you'll see our very nice user interface called **Flow**.

```
In [1]: import h2o

In [2]: h2o.init()

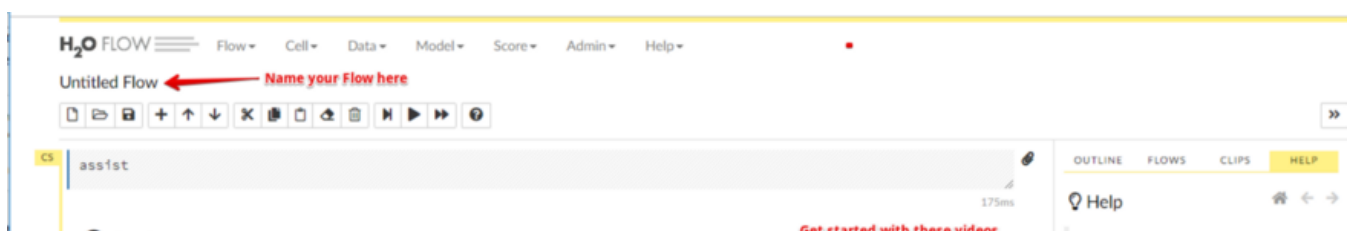
Checking whether there is an H2O instance running at http://localhost:54321 ..... not found.
Attempting to start a local H2O server...
; Java HotSpot(TM) 64-Bit Server VM (build 25.201-b09, mixed mode)
Starting server from c:\users\parul\AppData\Local\programs\python\python37\lib\site-packages\h2o\backend\bin\h2o.jar
Ice root: C:\Users\Parul\AppData\Local\Temp\tmp241o0168
JVM stdout: C:\Users\Parul\AppData\Local\Temp\tmp241o0168\h2o_Parul_started_from_python.out
JVM stderr: C:\Users\Parul\AppData\Local\Temp\tmp241o0168\h2o_Parul_started_from_python.err
Server is running at http://127.0.0.1:54321
Connecting to H2O server at http://127.0.0.1:54321 ... successful.
```

H2O cluster uptime:	04 secs
H2O cluster timezone:	Asia/Kolkata
H2O data parsing timezone:	UTC

Launching H2O flow

Flow Interface

Here is a quick glance over the flow interface. You can read more about using and working with it [here](#).



Assistance

Routine	Description
<code>importFiles</code>	Import file(s) into H ₂ O
<code>importSqlTable</code>	Import SQL table into H ₂ O
<code>getFrames</code>	Get a list of frames in H ₂ O
<code>splitFrame</code>	Split a frame into two or more frames
<code>mergeFrames</code>	Merge two frames into one
<code>getModels</code>	Get a list of models in H ₂ O
<code>getGrids</code>	Get a list of grid search results in H ₂ O
<code>getPredictions</code>	Get a list of predictions in H ₂ O
<code>getJobs</code>	Get a list of jobs running in H ₂ O
<code>runAutoML</code>	Automatically train and tune many models
<code>buildModel</code>	Build a model
<code>importModel</code>	Import a saved model
<code>predict</code>	Make a prediction

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

H2O's flow interface

Flow is designed to help data scientists rapidly and easily create models, import files, split data frames and do all the things that would normally require quite a bit of typing in other environments.

Working

Let's work through our same wine example but this time with Flow. The following video explains the model building and prediction using flow and it is kind of self-explanatory.



3 min

Demonstration of H2O Flow

For a deep dive into FLOW and its capabilities, refer to the article below:

Getting started with H2O using Flow

A look into H2O's open-source UI for combining code execution, text, plots, and rich media in a...

towardsdatascience.com

. . .

Conclusion

H2O is a powerful tool and given its capabilities, it can really transform the Data Science process for good. The capabilities and advantages of AI should be made available to everybody and not a select few. This is the real essence of Democratisation and Democratising Data Science should be essential for resolving Real problems threatening our planet.

[Machine Learning](#)

[Artificial Intelligence](#)

[H2oai](#)

[Automl](#)

[Data Science](#)

[About](#) [Help](#) [Legal](#)