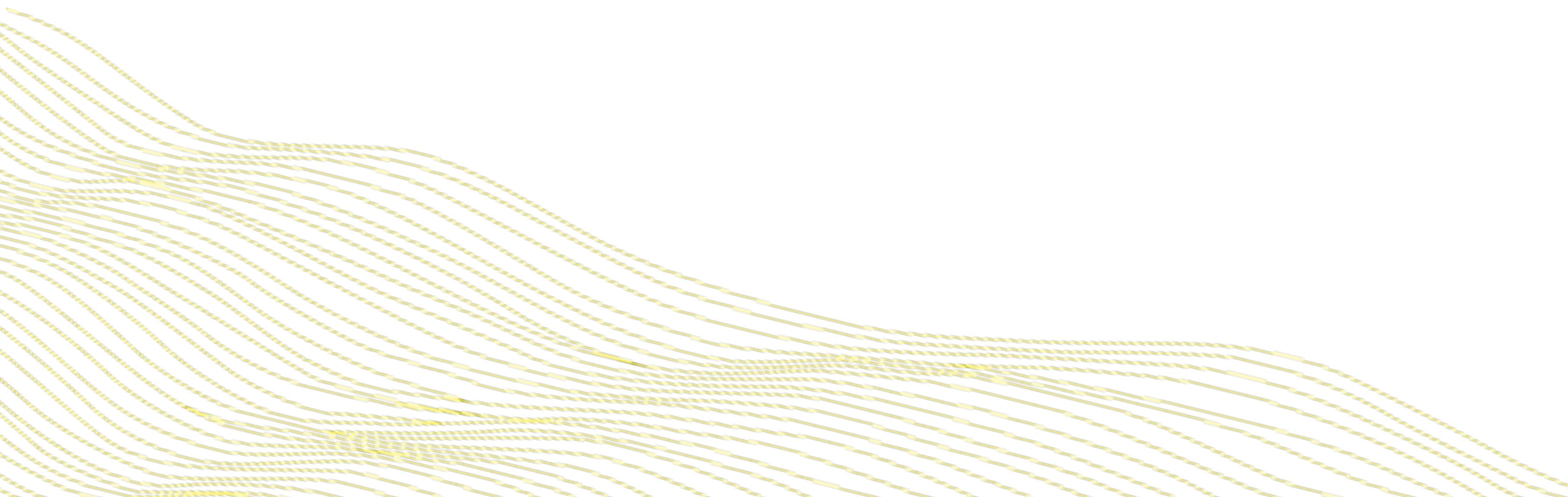


# **Architecture & Client API Overview Session**



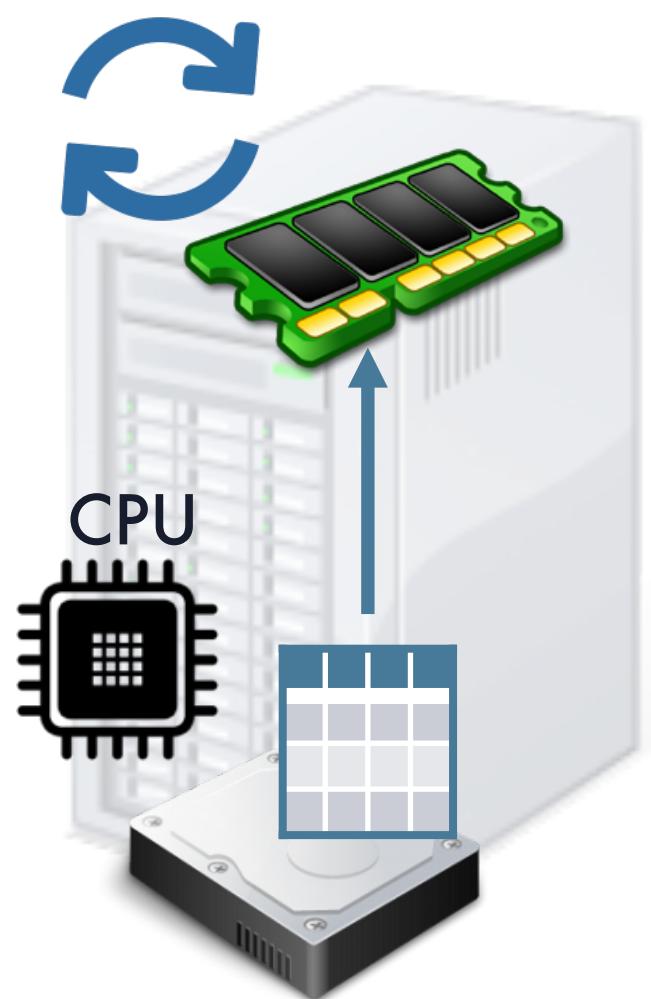
# H2O Core



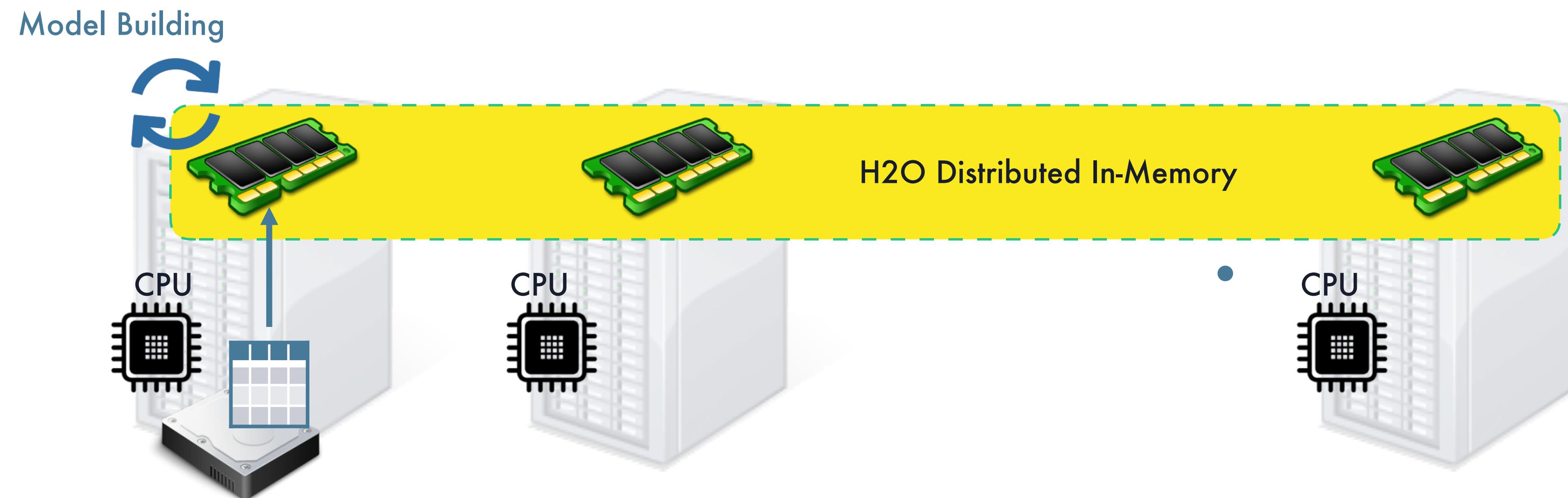
- Laptop
- Virtual Machine
- Container

# H2O Core

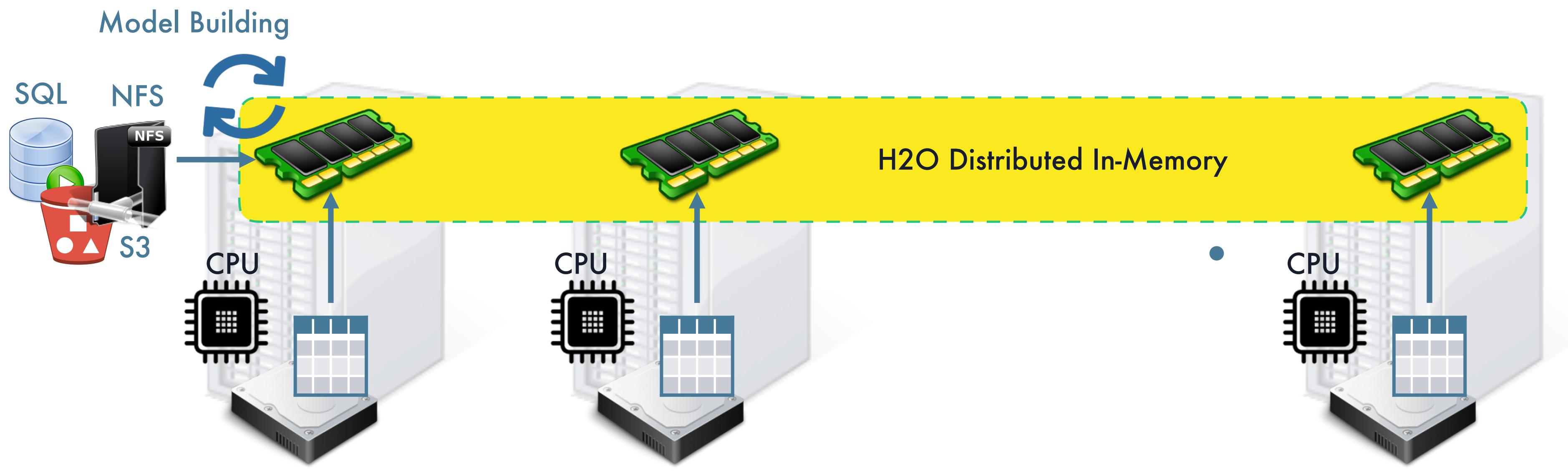
Model Building



# H2O Core



# H2O Core with Other Data Sources

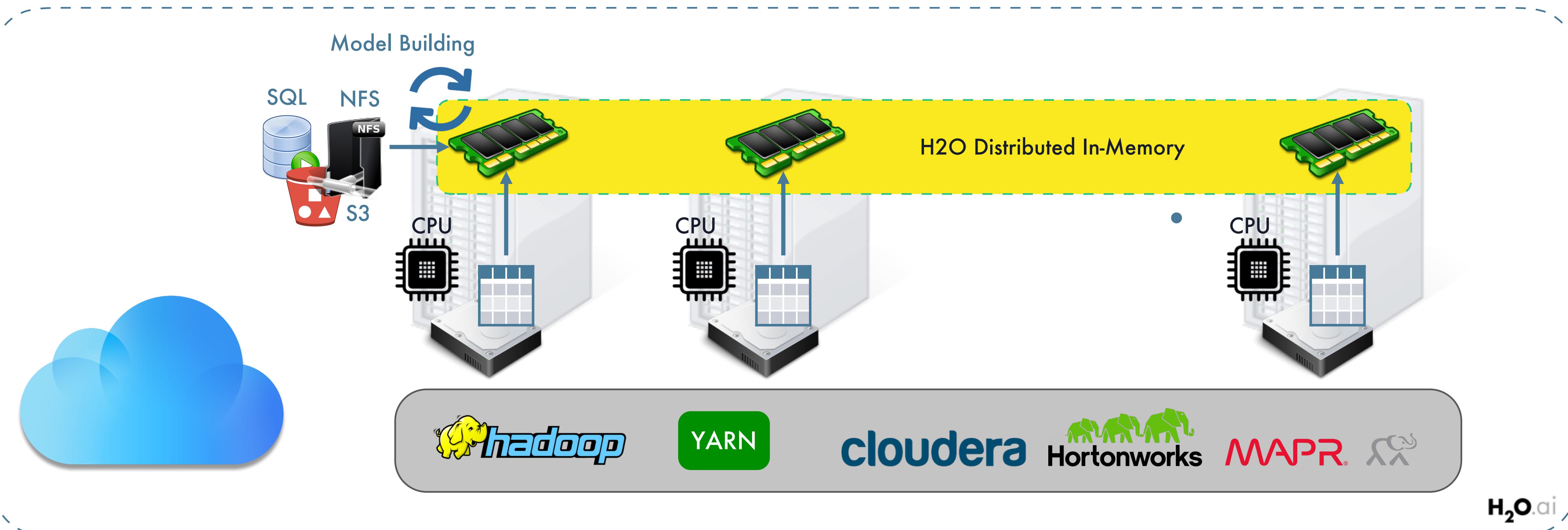


YARN

cloudera Hortonworks

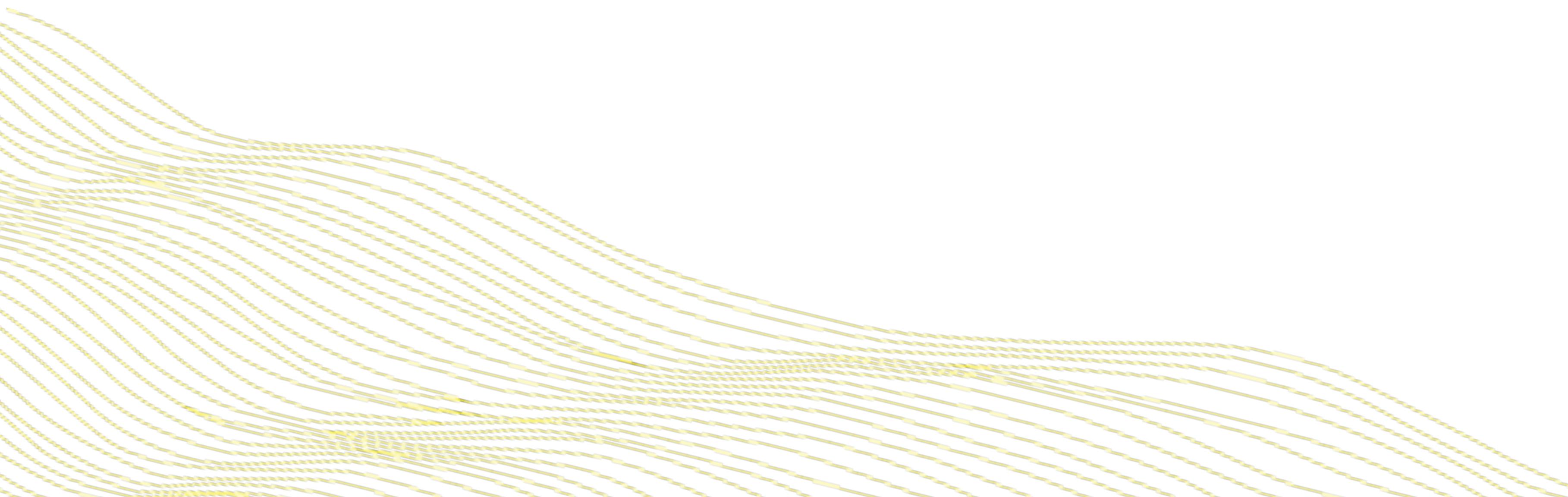
MAPR

# H2O Cluster on the Cloud



# The API Overview

*Client*  $\longleftrightarrow$  *Cluster Communication*



# First: The Client

RStudio Interface

Standard R Interface



RStudio

Project: (None)

Source

Console ~/ ↵

```
> library("h2o")
> h2o.init(ip = "localhost", port =54321)

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

Note: In case of errors look at the following log files:
/var/folders/k/_kpp1czqs3957vq2pr5qngck0000gn/T//Rtmpb16WSZ/h2o_laurend_started_from_r.out
/var/folders/k/_kpp1czqs3957vq2pr5qngck0000gn/T//Rtmpb16WSZ/h2o_laurend_started_from_r.err

java version "1.8.0_73"
Java(TM) SE Runtime Environment (build 1.8.0_73-b02)
Java HotSpot(TM) 64-Bit Server VM (build 25.73-b02, mixed mode)

Starting H2O JVM and connecting: ... Connection successful!

R is connected to the H2O cluster:
  H2O cluster uptime:      1 seconds 943 milliseconds
  H2O cluster version:     3.10.5.1
  H2O cluster version age: 2 months and 29 days
  H2O cluster name:        H2O_started_from_R_laurend_jxt477
  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
  R Version:              R version 3.3.1 (2016-06-21)

> |
```

Terminal Interface

```
Last login: Thu Sep  7 16:38:47 on ttys000
12:01:23 ~ R

R version 3.3.1 (2016-06-21) -- "Bug in Your Hair"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

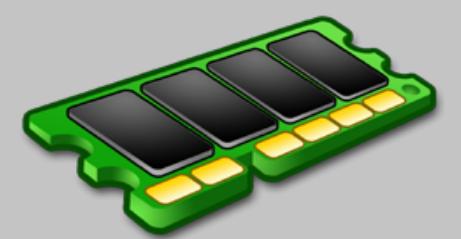
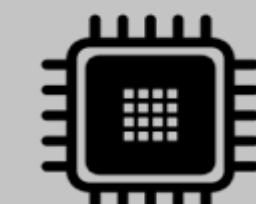
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> |
```

Local  
Machine



# Second: The Script

## R Script using H2O Package

R Script



RStudio

Untitled1\*

```
1 library("h2o")
2 h2o.init(ip ="localhost", port =54321)
3 # import the cars dataset:
4 # this dataset is used to classify whether or not a car is economical based on
5 # the car's displacement, power, weight, and acceleration, and the year it was made
6 cars <- h2o.importFile("https://s3.amazonaws.com/h2o-public-test-data/smalldata/junit/cars_20mpg.csv")
7
8 # convert response column to a factor
9 cars["economy_20mpg"] <- as.factor(cars["economy_20mpg"])
10
11 # set the predictor names and the response column name
12 predictors <- c("displacement","power","weight","acceleration","year")
13 response <- "economy_20mpg"
14
15 # split into train and validation
16 cars.splits <- h2o.splitFrame(data = cars, ratios = .8)
17 train <- cars.splits[[1]]
18 valid <- cars.splits[[2]]
19
20 # try using the `family` parameter:
21 car_glm <- h2o.glm(x = predictors, y = response, family = 'binomial', training_frame = train,
22                      validation_frame = valid)
23
24 # print the auc for your validation data
25 print(h2o.auc(car_glm, valid = TRUE))
```

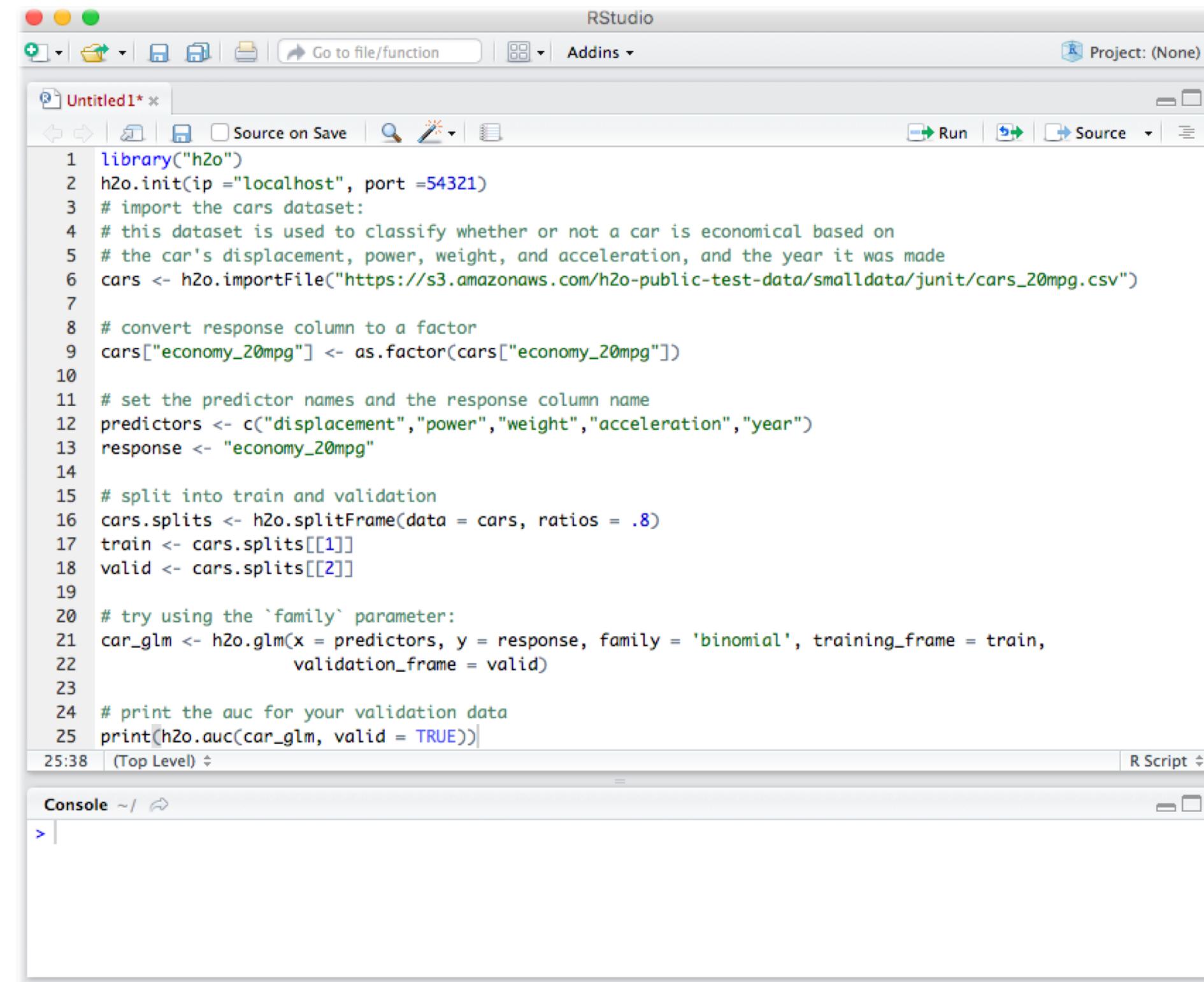
25:38 (Top Level) R Script

Console ~/ >

# Third: A Command

## Importing Big Data with R Script

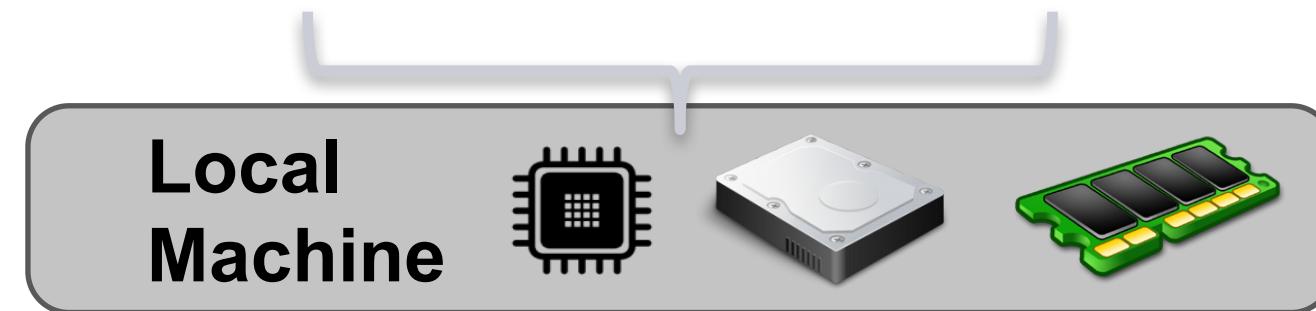
### R Commands



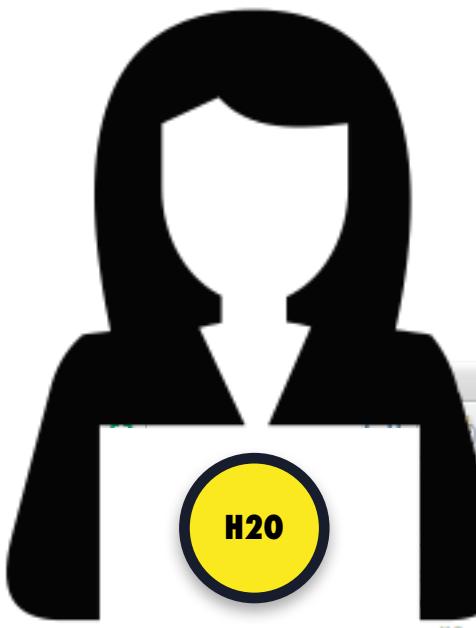
The screenshot shows an RStudio interface with an R script file named "Untitled1.R". The code imports the "cars" dataset from an S3 bucket and performs some initial data processing and splitting into training and validation sets. The RStudio interface includes a toolbar, a menu bar, and a console window below the script editor.

```
1 library("h2o")
2 h2o.init(ip = "localhost", port = 54321)
3 # import the cars dataset:
4 # this dataset is used to classify whether or not a car is economical based on
5 # the car's displacement, power, weight, and acceleration, and the year it was made
6 cars <- h2o.importFile("https://s3.amazonaws.com/h2o-public-test-data/smalldata/junit/cars_20mpg.csv")
7
8 # convert response column to a factor
9 cars["economy_20mpg"] <- as.factor(cars["economy_20mpg"])
10
11 # set the predictor names and the response column name
12 predictors <- c("displacement", "power", "weight", "acceleration", "year")
13 response <- "economy_20mpg"
14
15 # split into train and validation
16 cars.splits <- h2o.splitFrame(data = cars, ratios = .8)
17 train <- cars.splits[[1]]
18 valid <- cars.splits[[2]]
19
20 # try using the `family` parameter:
21 car_glm <- h2o.glm(x = predictors, y = response, family = 'binomial', training_frame = train,
22 validation_frame = valid)
23
24 # print the auc for your validation data
25 print(h2o.auc(car_glm, valid = TRUE))
```

**h2o.importFile(...)**



# Fourth: Communicate



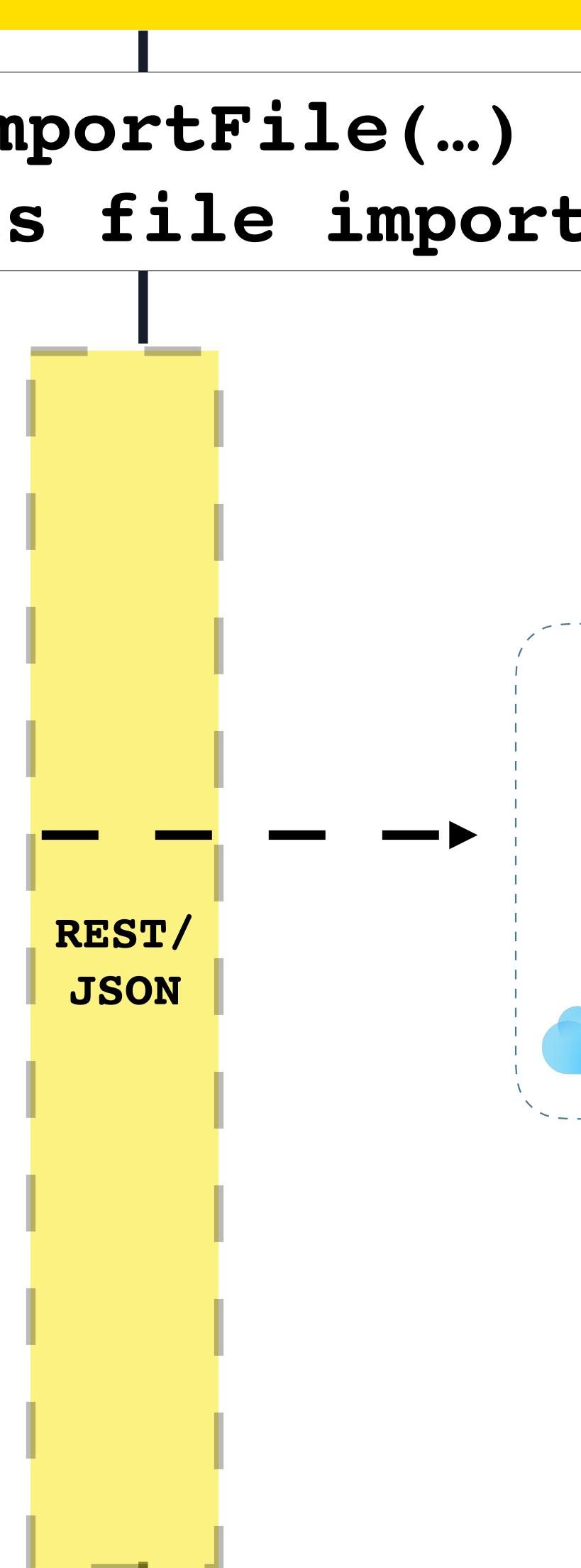
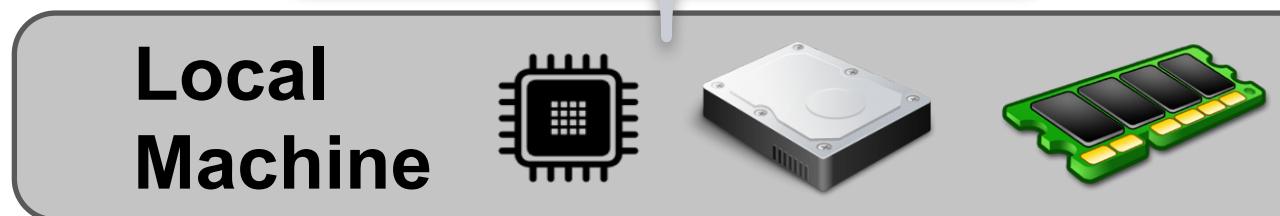
RStudio

```
h2o.importFile(...)  
requests file import
```

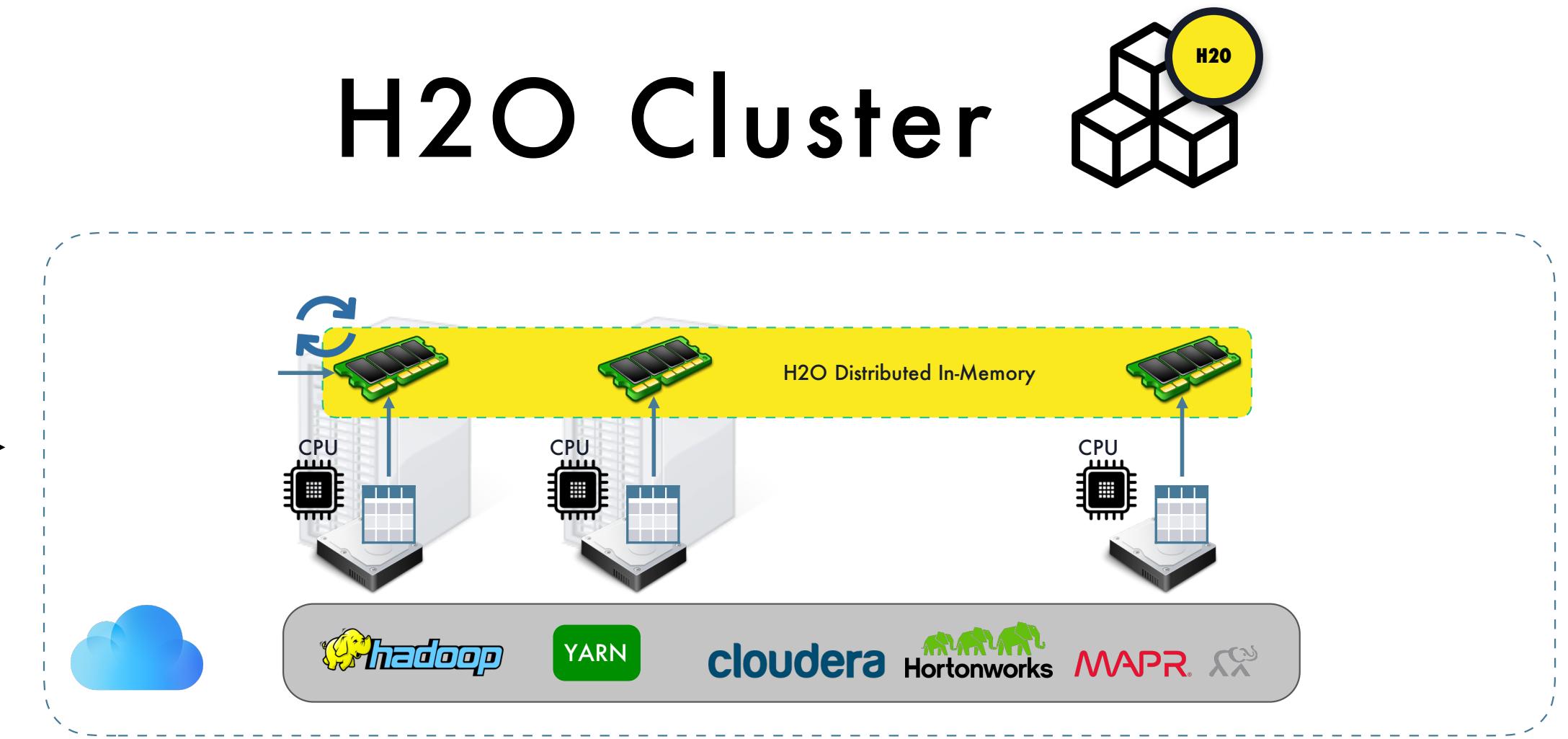
```
1 h2o.H2OJavaUtil.HOST = "localhost", port = 54321  
2  
3 # import the cars dataset:  
4 # this dataset is used to classify whether or not a car is economical based on  
5 # the car's displacement, power, weight, and acceleration, and the year it was made  
6 cars <- h2o.importFile("https://s3.amazonaws.com/h2o-public-test-data/smalldata/junit/cars_20mpg.csv")  
7  
8 # convert response column to a factor  
9 cars[["economy_20mpg"]] <- as.factor(cars[["economy_20mpg"]])  
10  
11 # set the predictor names and the response column name  
12 predictors <- c("displacement","power","weight","acceleration","year")  
13 response <- "economy_20mpg"  
14  
15 # split into train and validation  
16 cars.splits <- h2o.splitFrame(data = cars, ratios = .8)  
17 train <- cars.splits[[1]]  
18 valid <- cars.splits[[2]]  
19  
20 # try using the `family` parameter:  
21 car_glm <- h2o.glm(x = predictors, y = response, family = 'binomial', training_frame = train,  
22 validation_frame = valid)  
23  
24 # print the auc for your validation data  
25 print(h2o.auc(car_glm, valid = TRUE))
```

25:38 (Top Level) R Script

Console ~/



## H2O Cluster



# Fifth: Cluster Does Heavy Lifting



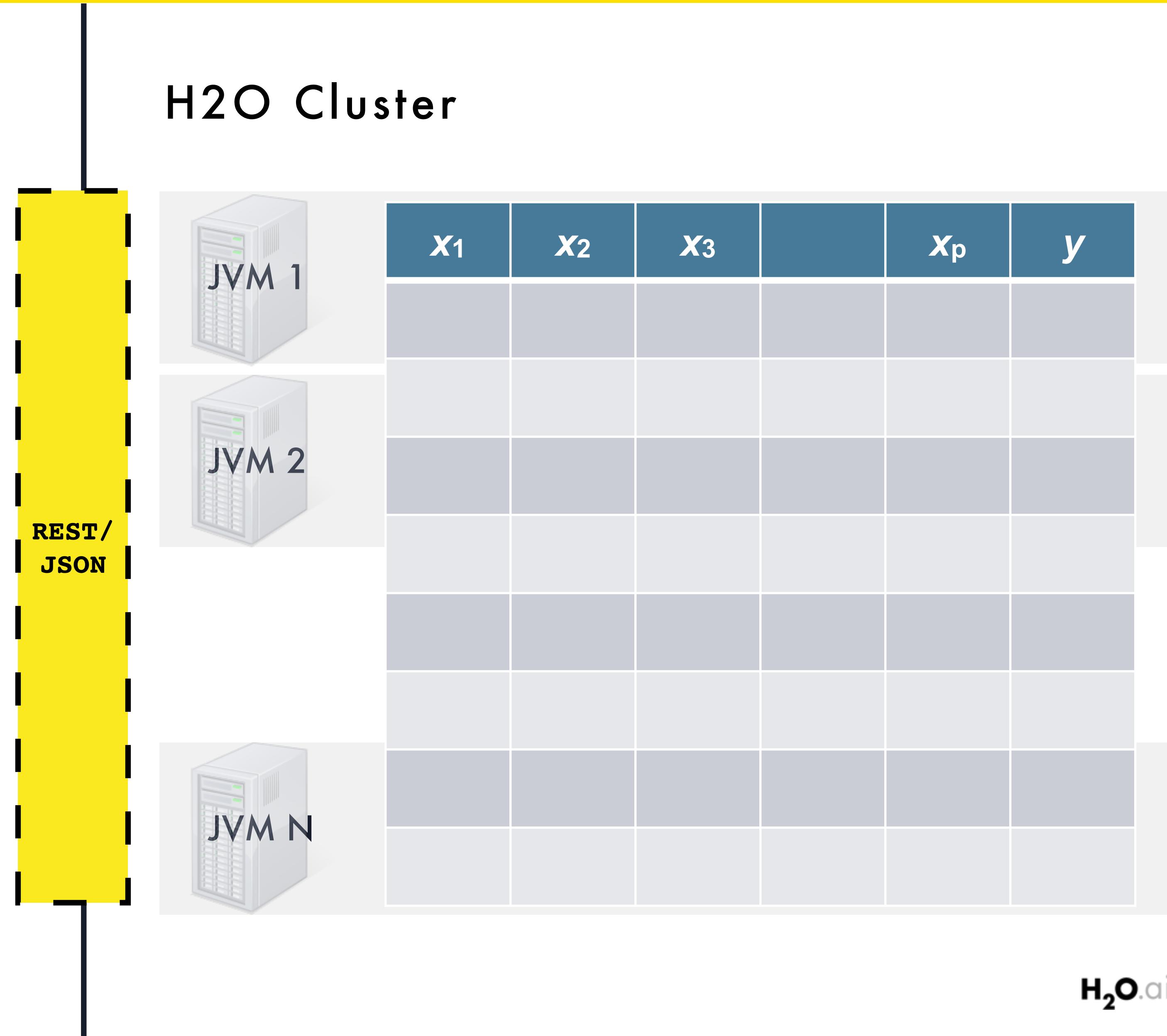
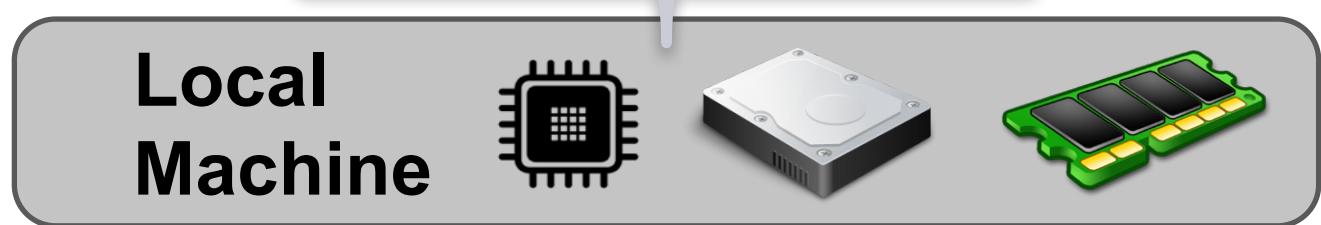
RStudio

```
1 h2o.init(ip = "localhost", port = 54321)
2 # import the cars dataset:
3 # this dataset is used to classify whether or not a car is economical based on
4 # the car's displacement, power, weight, and acceleration, and the year it was made
5 cars <- h2o.importFile("https://s3.amazonaws.com/h2o-public-test-data/smalldata/junit/cars_20mpg.csv")
6
7 # convert response column to a factor
8 cars[["economy_20mpg"]] <- as.factor(cars[["economy_20mpg"]])
9
10 # set the predictor names and the response column name
11 predictors <- c("displacement","power","weight","acceleration","year")
12 response <- "economy_20mpg"
13
14 # split into train and validation
15 cars.splits <- h2o.splitFrame(data = cars, ratios = .8)
16 train <- cars.splits[[1]]
17 valid <- cars.splits[[2]]
18
19 # try using the `family` parameter:
20 car_glm <- h2o.glm(x = predictors, y = response, family = 'binomial', training_frame = train,
21                      validation_frame = valid)
22
23 # print the auc for your validation data
24 print(h2o.auc(car_glm, valid = TRUE))
25
```

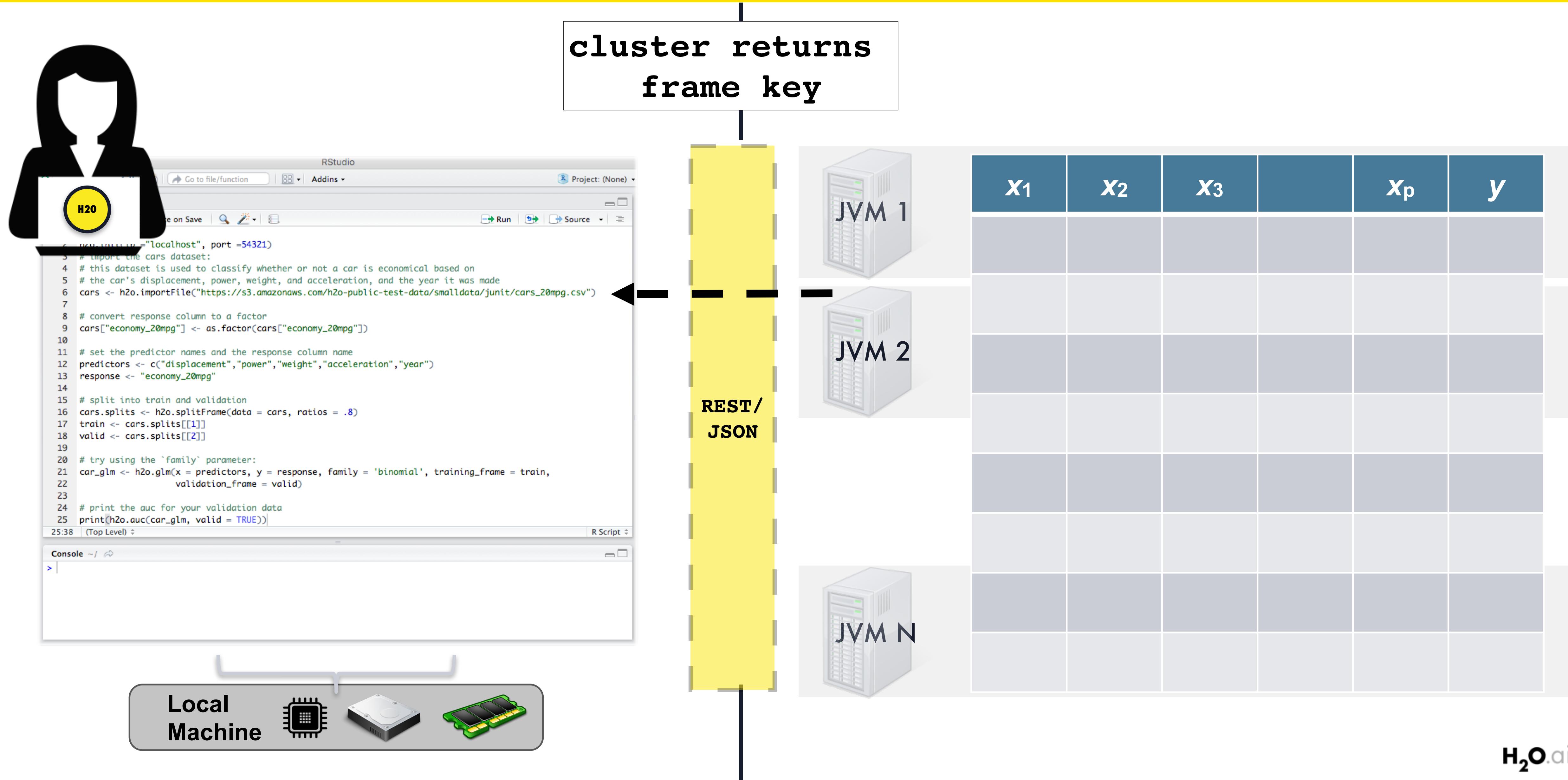
25:38 (Top Level) ▾

Console ~/ ▾

```
>
```



# H2O Clients



# Access a Frame Through its Key

```
Console ~/ 
> library(h2o)
> h2o.init()
Connection successful!

R is connected to the H2O cluster:
H2O cluster uptime:      3 minutes 54 seconds
H2O cluster version:     3.14.0.6
H2O cluster version age: 9 days
H2O cluster name:        H2O_started_from_R_laurend_msu572
H2O cluster total nodes: 1
H2O cluster total memory: 3.54 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:     XGBoost, Algos, AutoML, Core V3, Core V4
R Version:               R version 3.3.1 (2016-06-21)

> path <- "https://s3.amazonaws.com/h2o-public-test-data/smalldata/junit/cars_20mpg.csv"
> destination_frame = "cars.hex"
> cars <- h2o.importFile(path=path, destination_frame = destination_frame)
>
> head(cars)
      name economy cylinders displacement power weight acceleration year economy_20mpg
1 AMC Ambassador Brougham  13.0         8          360    175    3821        11.0    73            0
2   AMC Ambassador DPL    15.0         8          390    190    3850         8.5    70            0
3   AMC Ambassador SST    17.0         8          304    150    3672        11.5    72            0
4   AMC Concord DL 6    20.2         6          232     90    3265        18.2    79            1
5   AMC Concord DL     18.1         6          258    120    3410        15.1    78            0
6   AMC Concord DL     23.0         4          151    NaN    3035        20.5    82            1
```

# Assigning the Frame's Key

```
Console ~/ 
> library(h2o)
> h2o.init()
Connection successful!

R is connected to the H2O cluster:
  H2O cluster uptime:      3 minutes 54 seconds
  H2O cluster version:    3.14.0.6
  H2O cluster version age: 9 days
  H2O cluster name:       H2O_started_from_R_laurend_msu572
  H2O cluster total nodes: 1
  H2O cluster total memory: 3.54 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:    XGBoost, Algos, AutoML, Core V3, Core V4
  R Version:              R version 3.3.1 (2016-06-21)

> path <- "https://s3.amazonaws.com/h2o-public-test-data/smalldata/junit/cars_20mpg.csv"
> destination_frame = "cars.hex" ←
> cars <- h2o.importFile(path=path, destination_frame = destination_frame)
>
> head(cars)
      name economy cylinders displacement power weight acceleration year economy_20mpg
1 AMC Ambassador Brougham  13.0        8          360   175   3821        11.0    73            0
2   AMC Ambassador DPL   15.0        8          390   190   3850         8.5    70            0
3   AMC Ambassador SST   17.0        8          304   150   3672        11.5    72            0
4   AMC Concord DL 6   20.2        6          232    90   3265        18.2    79            1
5   AMC Concord DL   18.1        6          258   120   3410        15.1    78            0
6   AMC Concord DL   23.0        4          151   NaN   3035        20.5    82            1
```

# Local Variable vs Key

```
> cars <- 5  
> head(cars)  
[1] 5
```

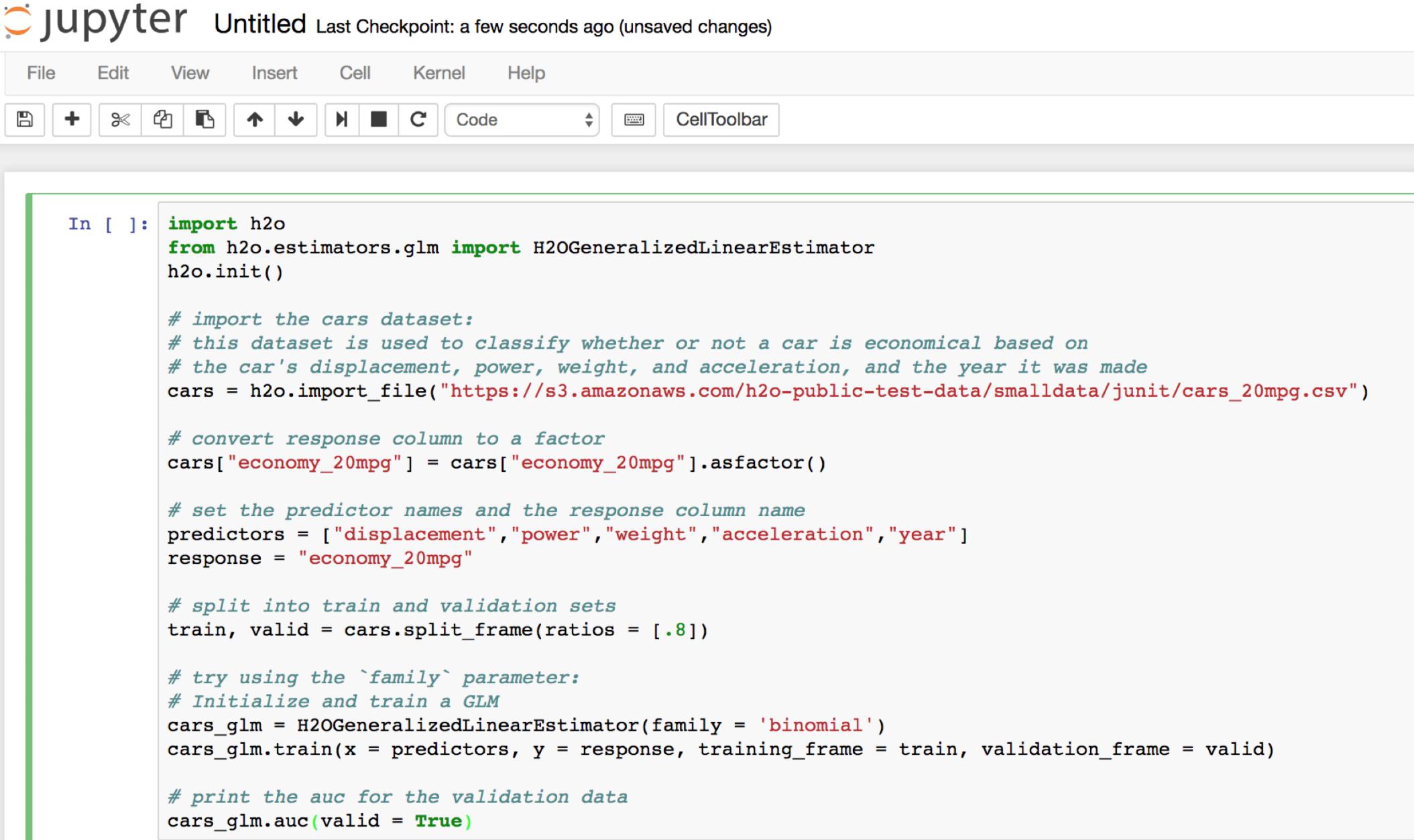
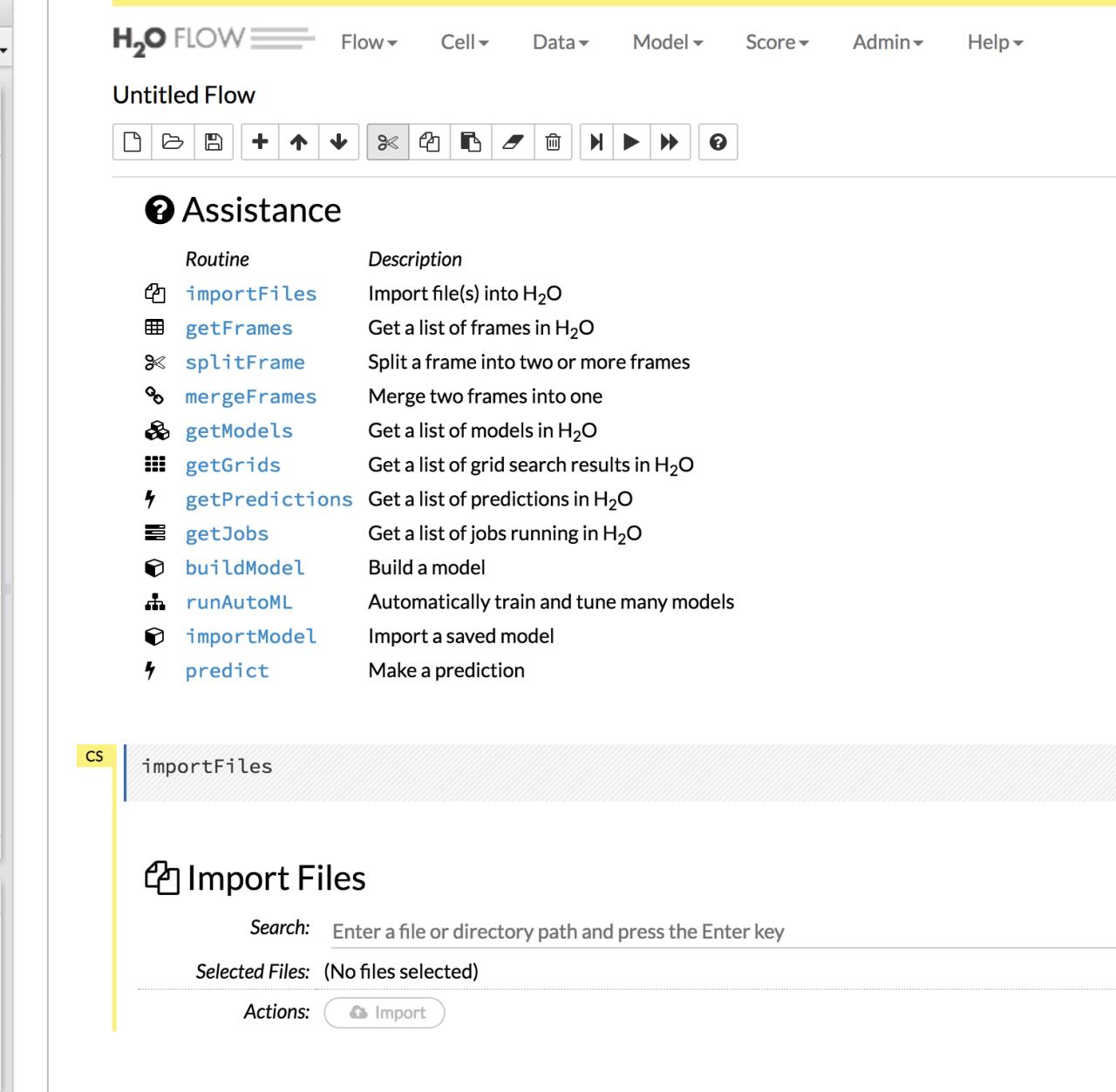
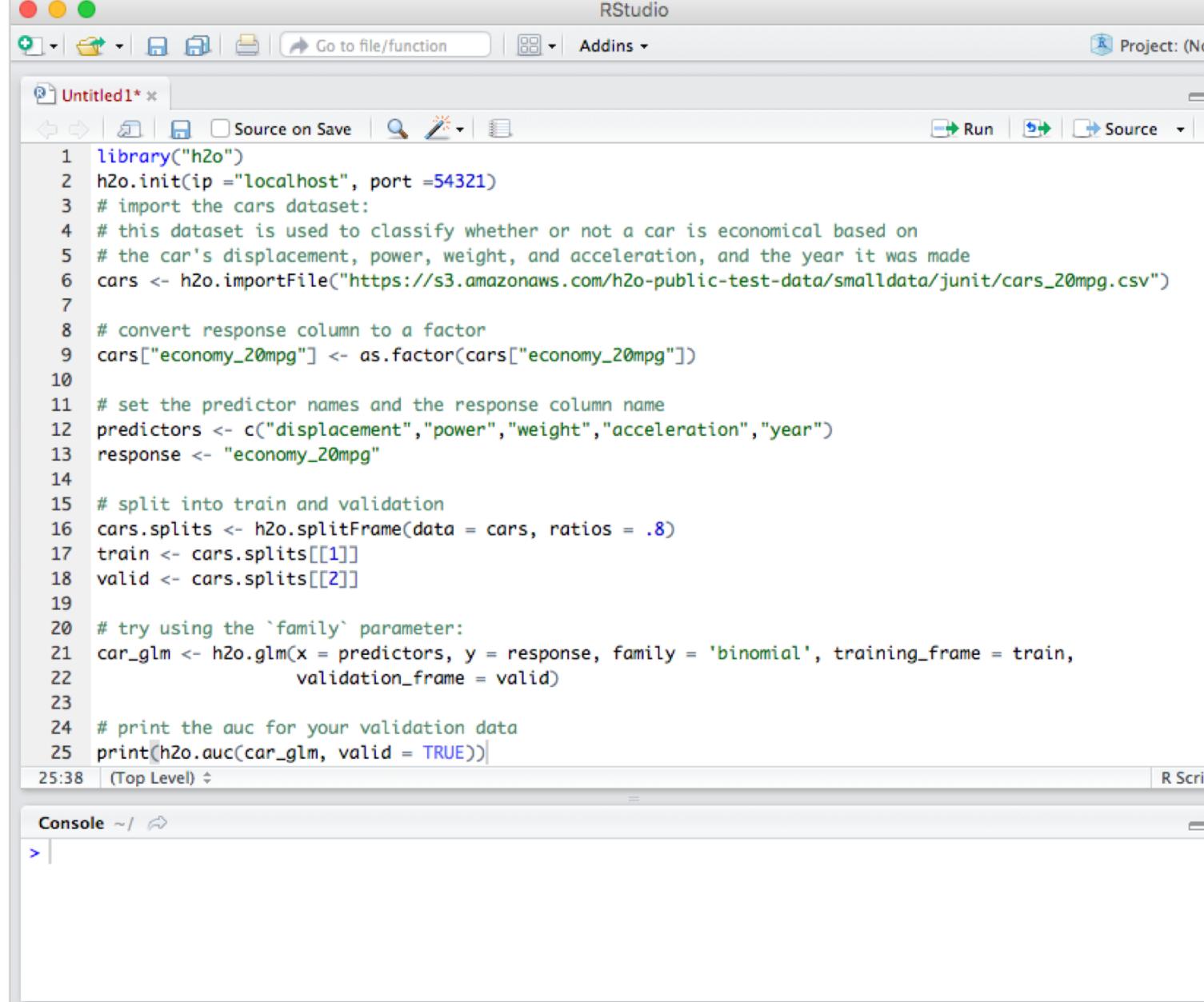
# Reassign Local Variable to the Frame

```
> cars <- h2o.getFrame('cars.hex')
> head(cars)

      name economy cylinders displacement power weight acceleration year economy_20mpg
1 AMC Ambassador Brougham 13.0          8        360    175   3821           11.0     73            0
2      AMC Ambassador DPL 15.0          8        390    190   3850            8.5     70            0
3      AMC Ambassador SST 17.0          8        304    150   3672           11.5     72            0
4      AMC Concord DL 6 20.2          6        232     90   3265           18.2     79            1
5      AMC Concord DL 18.1          6        258    120   3410           15.1     78            0
6      AMC Concord DL 23.0          4        151    NaN   3035           20.5     82            1
```

# Access a Frame From Anywhere

All you need is the **key**



The image displays three software interfaces side-by-side, illustrating the cross-platform access to H2O frames:

- RStudio:** Shows R code for loading the "cars" dataset, splitting it into training and validation sets, and fitting a GLM model. The code includes comments explaining the steps.
- H2O FLOW:** Shows a workflow titled "Untitled Flow". The first step is labeled "Import Files". A sidebar lists various H2O routines with their descriptions.
- Jupyter:** Shows Python code for the same task, using the `h2o` library and `H2OGeneralizedLinearEstimator` to fit a GLM model to the "cars" dataset.

`h2o.getFrame("cars.hex")`



`h2o.get_frame("cars.hex")`

# Who Does the Heavy Lifting?

The Client

The Cluster

# What does the Client Do?

R Only Tells the Cloud What to Do

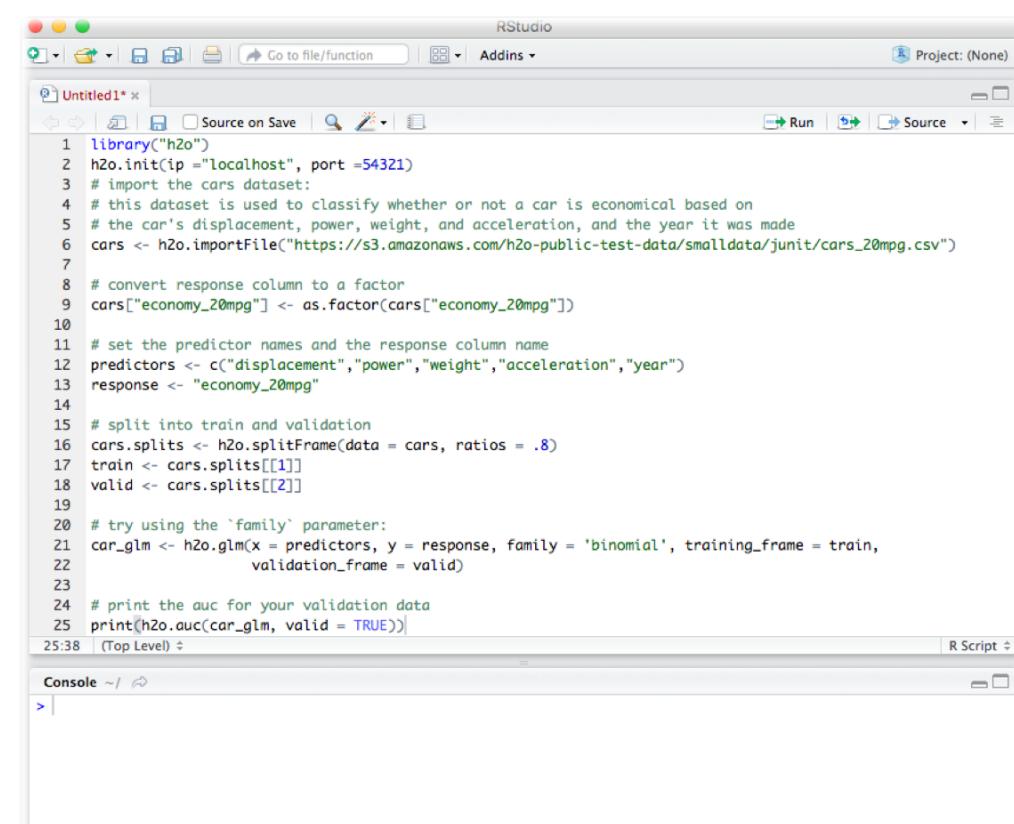


```
library("h2o")
h2o.init(ip = "localhost", port = 54321)
# import the cars dataset:
# this dataset is used to classify whether or not a car is economical based on
# the car's displacement, power, weight, and acceleration, and the year it was made
cars <- h2o.importFile("https://s3.amazonaws.com/h2o-public-test-data/smalldata/junit/cars_20mpg.csv")
# convert response column to factor
cars["economy_20mpg"] <- as.factor(cars["economy_20mpg"])
# set the predictor names and the response column name
predictors <- c("displacement","power","weight","acceleration","year")
response <- "economy_20mpg"
# split into train and validation
cars.splits <- h2o.splitFrame(data = cars, ratios = .8)
train <- cars.splits[[1]]
valid <- cars.splits[[2]]
# try using the `family` parameter:
cor_glm <- h2o.glm(x = predictors, y = response, family = 'binomial', training_frame = train,
validation_frame = valid)
# print the auc for your validation data
print(h2o.auc(cor_glm, valid = TRUE))
```

Hi Cluster,  
Can you please get me ...

# Client Only Passes Requests

Big Data Never Flows Through R  
Unless **Explicitly** Asked



The screenshot shows an RStudio interface with an 'Untitled1' script file open. The code performs the following steps:

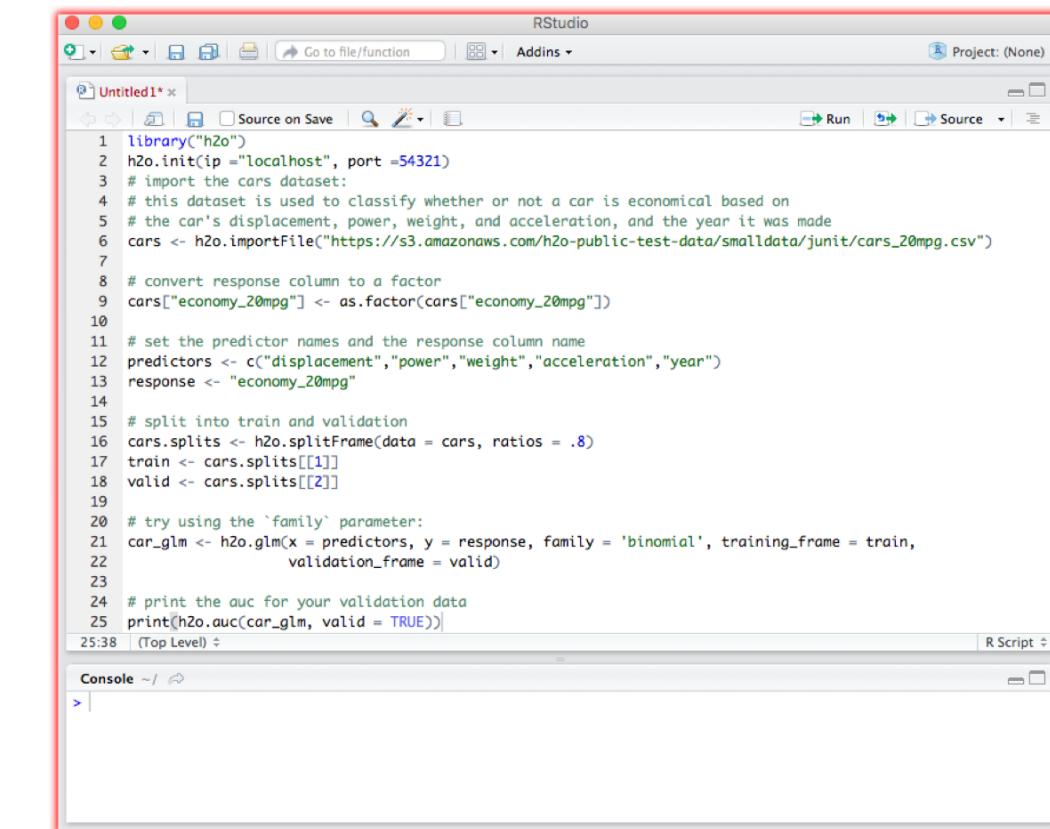
- Imports the 'h2o' library.
- Initializes H2O with port 54321.
- Imports the 'cars' dataset.
- Specifies the dataset is used to classify whether or not a car is economical based on its displacement, power, weight, and acceleration, and the year it was made.
- Imports the 'cars\_20mpg.csv' file from S3.
- Converts the 'economy\_20mpg' column to a factor.
- Sets predictor names and response column name.
- Creates splits for training and validation data.
- Attempts to fit a GLM model using the 'binomial' family.
- Prints the AUC for the validation data.

No, you take care of the  
heavy lifting

# What if?

## Pulling Big Data into R Can Overwhelm Your Session

**as.data.frame(my\_big\_dataframe)**



A screenshot of an RStudio interface. The top bar shows 'RStudio' and 'Untitled1\*'. The main area is a code editor with the following R code:

```
1 library("h2o")
2 h2o.init(ip = "localhost", port = 54321)
3 # import the cars dataset:
4 # this dataset is used to classify whether or not a car is economical based on
5 # the car's displacement, power, weight, and acceleration, and the year it was made
6 cars <- h2o.importFile("https://s3.amazonaws.com/h2o-public-test-data/smalldata/junit/cars_20mpg.csv")
7
8 # convert response column to a factor
9 cars["economy_20mpg"] <- as.factor(cars["economy_20mpg"])
10
11 # set the predictor names and the response column name
12 predictors <- c("displacement","power","weight","acceleration","year")
13 response <- "economy_20mpg"
14
15 # split into train and validation
16 cars.splits <- h2o.splitFrame(data = cars, ratios = .8)
17 train <- cars.splits[[1]]
18 valid <- cars.splits[[2]]
19
20 # try using the `family` parameter:
21 car_glm <- h2o.glm(x = predictors, y = response, family = 'binomial', training_frame = train,
22                      validation_frame = valid)
23
24 # print the auc for your validation data
25 print(h2o.auc(car_glm, valid = TRUE))
```

The bottom panel is the 'Console' window, which is currently empty.

AH!

# Documentation Resources

H<sub>2</sub>O Open Source Software  
Documentation | H<sub>2</sub>O Commercial  
Software Documentation

## Open Source Software Documentation

Getting Started & User Guides | Q & A | Algorithms | Languages | Tutorials, Examples, &  
Presentations | API & Developer Docs | For the Enterprise

### Getting Started & User Guides

The screenshot shows the main navigation menu of the H2O documentation site. It includes links for 'What is H2O?', 'H2O User Guide (Main docs)', 'H2O Book (O'Reilly)', 'Recent Changes', and 'Open Source License (Apache V2)'. Below this is a section for 'Quick Start Video - Flow Web UI', 'Quick Start Video - R', and 'Quick Start Video - Python'. At the bottom is a prominent yellow button labeled 'Download H2O'.

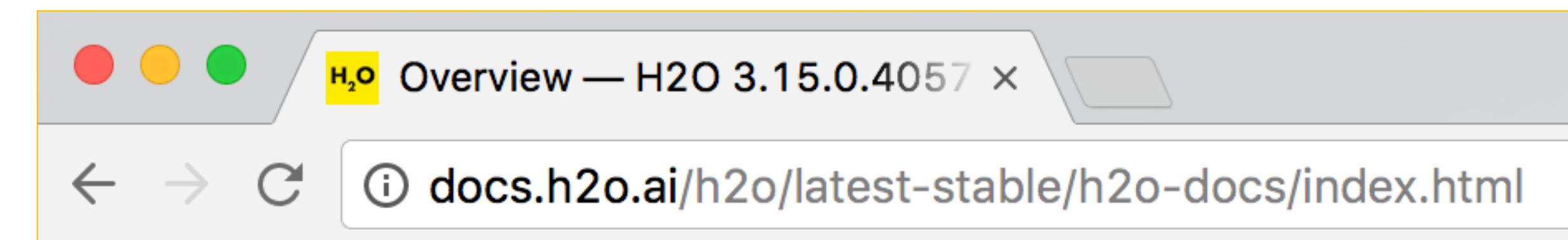
H<sub>2</sub>O

What is H2O?  
H2O User Guide (Main docs)  
H2O Book (O'Reilly)  
Recent Changes  
Open Source License (Apache V2)

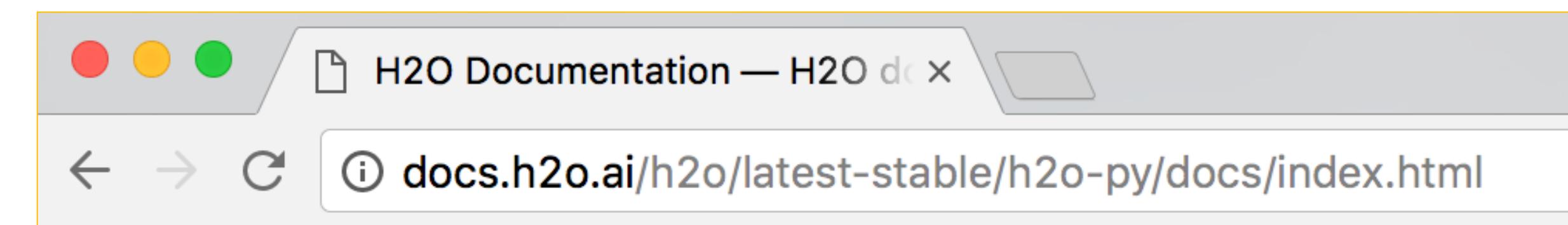
Quick Start Video - Flow Web UI  
Quick Start Video - R  
Quick Start Video - Python

Download H2O

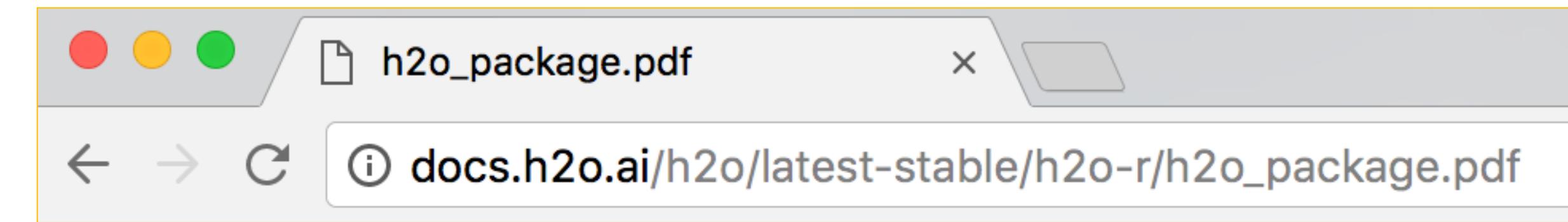
All Documentation @ <http://docs.h2o.ai/h2o/latest-stable/index.html>



Python Docs @ <http://docs.h2o.ai/h2o/latest-stable/h2o-py/docs/index.html>



R Docs @ [http://docs.h2o.ai/h2o/latest-stable/h2o-r/h2o\\_package.pdf](http://docs.h2o.ai/h2o/latest-stable/h2o-r/h2o_package.pdf)



# Where to Learn about GBM

**H<sub>2</sub>O.ai**

3.15.0.4057

Search docs

Welcome to H<sub>2</sub>O 3

Quick Start Videos

Cloud Integration

Downloading & Installing H<sub>2</sub>O

Starting H<sub>2</sub>O

Getting Data into H<sub>2</sub>O

Data Manipulation

⊖ Algorithms

- ⊕ Common
- ⊕ Supervised
- ⊕ Unsupervised
- ⊕ Miscellaneous

Cross-Validation

Grid (Hyperparameter) Search

Docs » Algorithms

[View page source](#)

## Algorithms

This section provides an overview of each algorithm available in H<sub>2</sub>O. For detailed information about the parameters that can be used for building models, refer to [Appendix A - Parameters](#).

### Common

- [Commonalities](#)

### Supervised

- [Deep Learning \(Neural Networks\)](#)
- [Distributed Random Forest \(DRF\)](#)
- [Generalized Linear Model \(GLM\)](#)
- [Gradient Boosting Machine \(GBM\)](#)
- [Naïve Bayes Classifier](#)
- [Stacked Ensembles](#)
- [XGBoost](#)

# Where to Learn about GBM in General

- Cloud Integration
- Downloading & Installing H2O
- Starting H2O
- Getting Data into H2O
- Data Manipulation

- Algorithms
  - Common
  - Supervised
    - Deep Learning (Neural Networks)
    - Distributed Random Forest (DRF)
    - Generalized Linear Model (GLM)
  - Gradient Boosting Machine (GBM)
    - Introduction
    - Quick Start
    - Defining a GBM Model
    - Interpreting a GBM Model
    - Leaf Node Assignment
    - GBM Algorithm
    - Parallel Performance in GBM
    - GBM Tuning Guide
    - References
    - FAQ

## Gradient Boosting Machine (GBM)

### Introduction

Gradient Boosting Machine (for Regression and Classification) is a forward learning ensemble method. The guiding heuristic is that good predictive results can be obtained through increasingly refined approximations. H2O's GBM sequentially builds regression trees on all the features of the dataset in a fully distributed way - each tree is built in parallel.

The current version of GBM is fundamentally the same as in previous versions of H2O (same algorithmic steps, same histogramming techniques), with the exception of the following changes:

- Improved ability to train on categorical variables (using the `nbins_cats` parameter)
- Minor changes in histogramming logic for some corner cases

There was some code cleanup and refactoring to support the following features:

- Per-row observation weights
- Per-row offsets
- N-fold cross-validation
- Support for more distribution functions (such as Gamma, Poisson, and Tweedie)

### Quick Start

## FAQ

This section describes some common questions asked by users. The questions are broken down based on one of the types below.

- [Preprocessing Steps](#)
- [Histograms and Binning](#)
- [Missing Values](#)
- [Default Values](#)
- [Building the First Tree](#)
- [Splitting](#)
- [Cross Validation](#)
- [About the Data](#)
- [Reproducibility](#)
- [Generated Metrics](#)
- [Scoring](#)
- [Tuning a GBM](#)

 [Previous](#)

[Next](#) 

# Where to Learn about GBM Parameters

## THE USER GUIDE

H<sub>2</sub>O.ai

3.15.0.4057

Search docs

Welcome to H2O 3

Quick Start Videos

Cloud Integration

Downloading & Installing H2O

Starting H2O

Getting Data into H2O

Data Manipulation

Algorithms

Cross-Validation

Grid (Hyperparameter) Search

AutoML: Automatic Machine Learning

Saving and Loading a Model

Productionizing H2O

Using Flow - H2O's Web UI

Downloading Logs

H2O Architecture

Security

FAQ

Glossary

Docs » Appendix A - Parameters

[View page source](#)

## Appendix A - Parameters

This Appendix provides detailed descriptions of parameters that can be specified in the H2O algorithms. In addition, each parameter also includes the algorithms that support the parameter, whether the parameter is a hyperparameter (can be used in grid search), links to any related parameters, and R and Python examples showing the parameter in use.

### Notes:

- This Appendix is a work in progress.
- For parameters that are supported in multiple algorithms, the included example uses the GBM or GLM algorithm.
- `alpha`
  - Description
  - Related Parameters
  - Example
- `balance_classes`
  - Description
  - Related Parameters
  - Example
- `beta_epsilon`
  - Description
  - Related Parameters
  - Example
- `binomial_double_trees`

## learn\_rate

- Available in: GBM, XGBoost
- Hyperparameter:

## Description

This option is used to specify the rate at which GBM learns when building a model. Lower learning rates are generally better, but then more trees are required (using `ntrees`) to achieve the same level of fit as if you had used a higher learning rate. This method helps avoid overfitting.

You can use this option along with the `learn_rate_annealing` option to reduce the learning rate by a specified factor for every tree. This can help speed of convergence without sacrificing too much accuracy. For faster scans, use (for example) `learn_rate=0.5` and `learn_rate_annealing=0.99`.

## Related Parameters

- `learn_rate_annealing`
- `ntrees`

## Example

H<sub>2</sub>O.ai

# Where to Learn about GBM Parameters

## learn\_rate

- Available in: GBM, XGBoost
- Hyperparameter:

## Description

This option is used to specify the rate at which GBM learns when building a model. Lower learning rates are generally better, but then more trees are required (using `ntrees`) to achieve the same level of fit as if you had used a higher learning rate. This method helps avoid overfitting.

You can use this option along with the `learn_rate_annealing` option to reduce the learning rate by a specified factor for every tree. This can help speed of convergence without sacrificing too much accuracy. For faster scans, use (for example) `learn_rate=0.5` and `learn_rate_annealing=0.99`.

## Related Parameters

- `learn_rate_annealing`
- `ntrees`

## Example

### Example

r    python

```
library(h2o)
h2o.init()
# import the titanic dataset:
# This dataset is used to classify whether a passenger will survive '1' or not '0'
# original dataset can be found at https://stat.ethz.ch/R-manual/R-devel/library/datasets
titanic <- h2o.importFile("https://s3.amazonaws.com/h2o-public-test-data/smalldata/gbm_titanic.csv")

# convert response column to a factor
titanic['survived'] <- as.factor(titanic['survived'])

# set the predictor names and the response column name
# predictors include all columns except 'name' and the response column ("survived")
predictors <- setdiff(colnames(titanic), colnames(titanic)[2:3])
response <- "survived"

# split into train and validation
titanic.splits <- h2o.splitFrame(data = titanic, ratios = .8, seed = 1234)
train <- titanic.splits[[1]]
valid <- titanic.splits[[2]]

# try using the `learn_rate` parameter:
# because we use a small learning_rate, we set ntrees to a much higher number
# early stopping makes it okay to use 'more than enough' trees
titanic.gbm <- h2o.gbm(x = predictors, y = response, training_frame = train, validation_frame = valid,
                         ntrees = 10000, learn_rate = .01,
                         # use early stopping once the validation AUC doesn't improve by at least 1e-4
                         # for 5 consecutive scoring events
                         stopping_rounds = 5,
                         stopping_tolerance = 1e-4,
                         stopping_metric = "AUC", seed = 1234)

# print the auc for the validation data
print(h2o.auc(titanic.gbm, valid = TRUE))
```

◀ Previous

Next ▶