Intro to H2O

Lauren DiPerna // Data Scientist @ H2O

Charlotte, NC // June 11, 2018

H2O.ai

The Agenda

- What is H2O
- How does H2O Core work?
- The Python API
- The Use Case Dataset
- How to Learn From Data
- How to use AutoML?

What is H2O.ai?

The AI Company that builds ML Products

H2O.ai: Company Overview

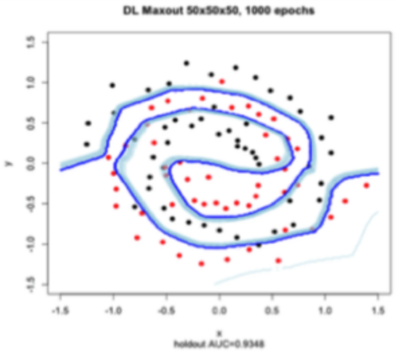
Founded	2011 Venture-backed, Debuted in 2012				
Products	**H20-3**	**E. Steam**	**Sparkling Water**	**DAI**	**H204GPU**
Mission	Operationalize Data Science & Platform to Build Beautiful Data Products				
Team	75+ employees: Distributed Systems Engineers , ML Experts, World-class Visualization Designers				
Headquarters	Mountain View, CA				

What is H2O-3?

Powerful Platform for End-to-End Machine Learning

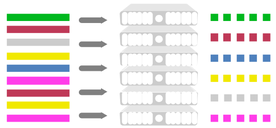
What is H2O-3?

Java-Based Software for **In-Memory** **Data Modeling**

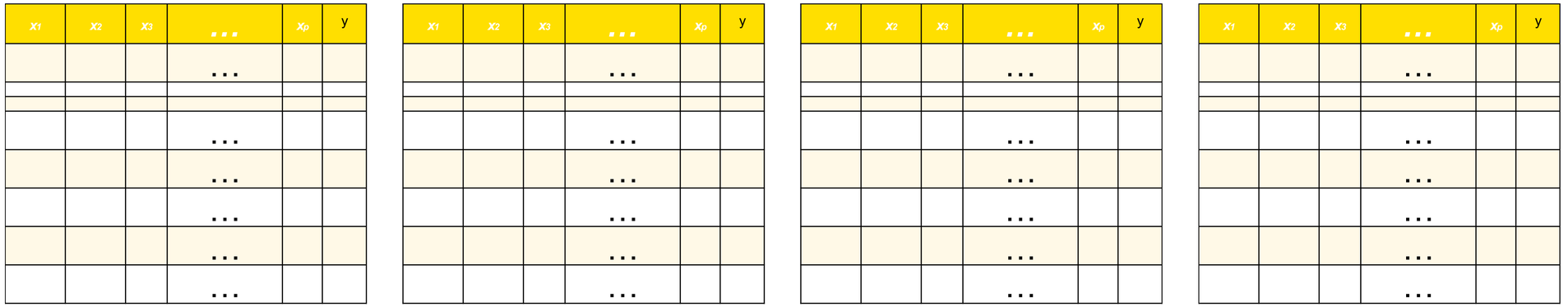
Open Source	Big Data Ecosystem	Flexible Interface	Smart and Fast Algorithms
			

What is H2O-3?

Java-Based Software for **In-Memory** **Data Modeling**

Scalability and Performance	Rapid Model Deployment	GPU Enablement*	Cloud Integration
	• Highly portable models deployed in Java (POJO) • Automated and streamlined scoring service deployment with Rest API*		 
• Distributed In-Memory Computing Platform • Distributed Algorithms • Fine-Grain MapReduce			

Why H2O-3?



No Data Size Limit

Why H2O-3?

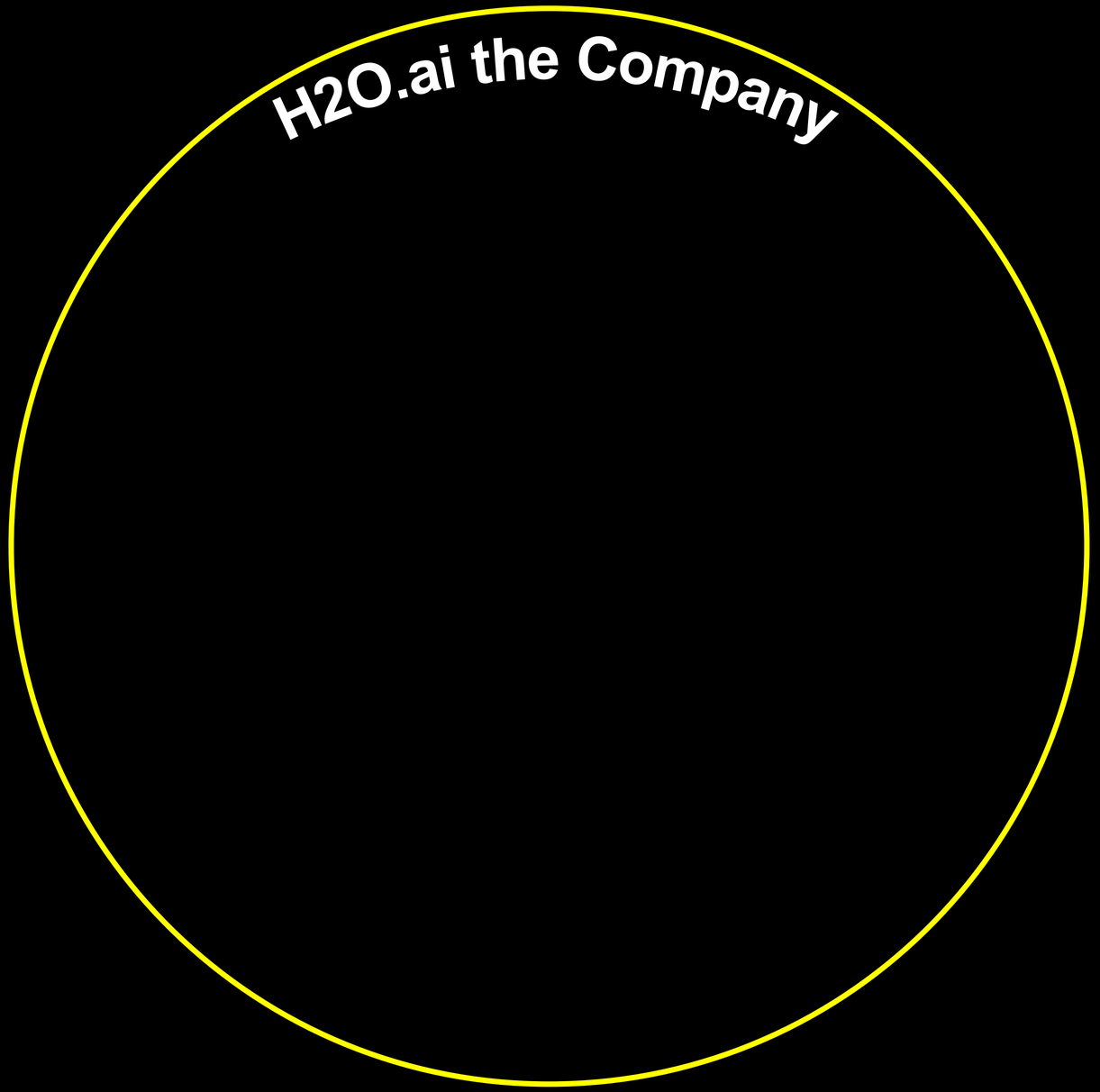
💛 R + 💛 Py. + 💛UI

Why H2O-3?

EASY

Live Results!

How to think about H2O-3?

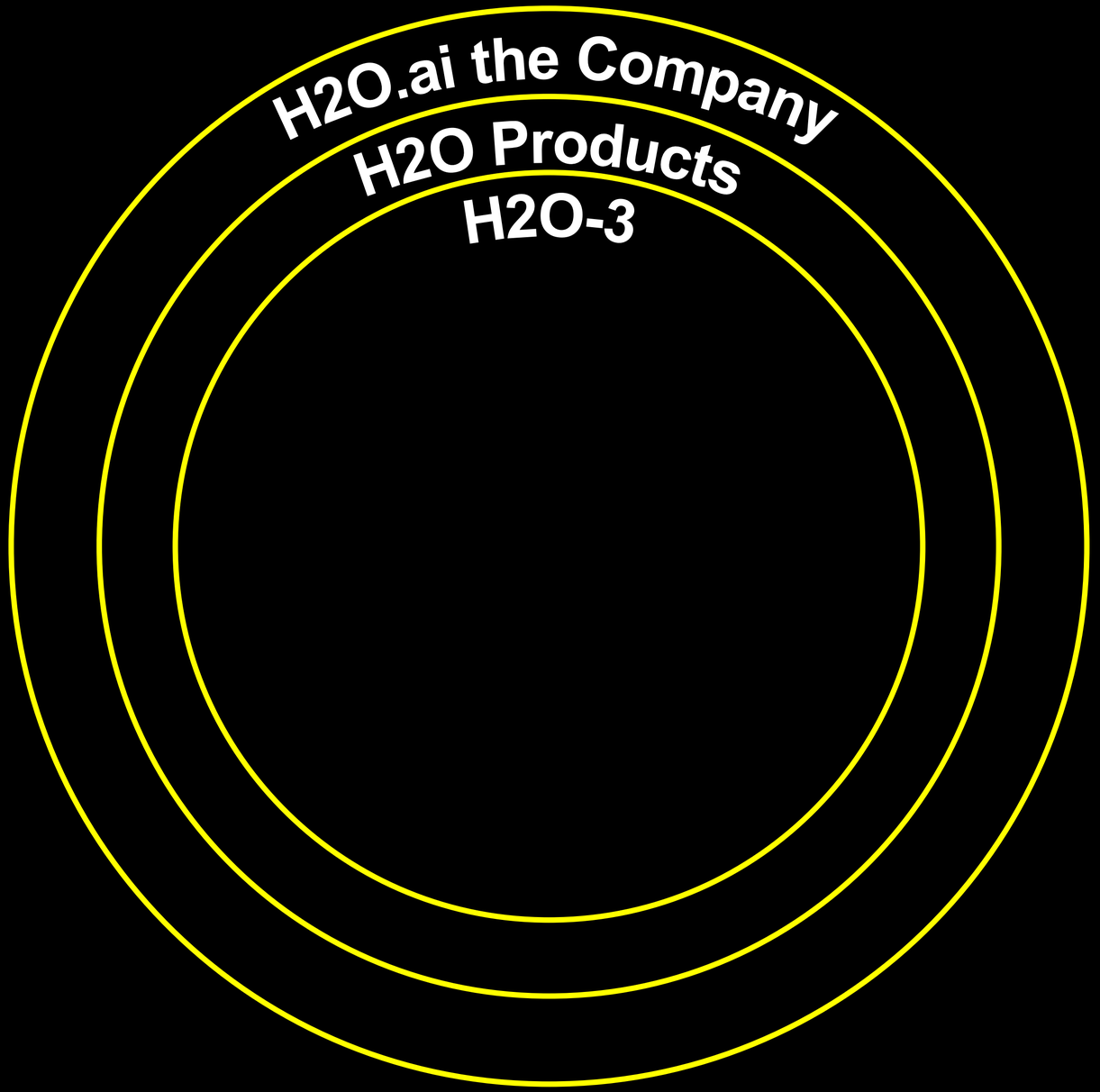

H2O.ai the Company


H2O.ai the Company
All H2O Products

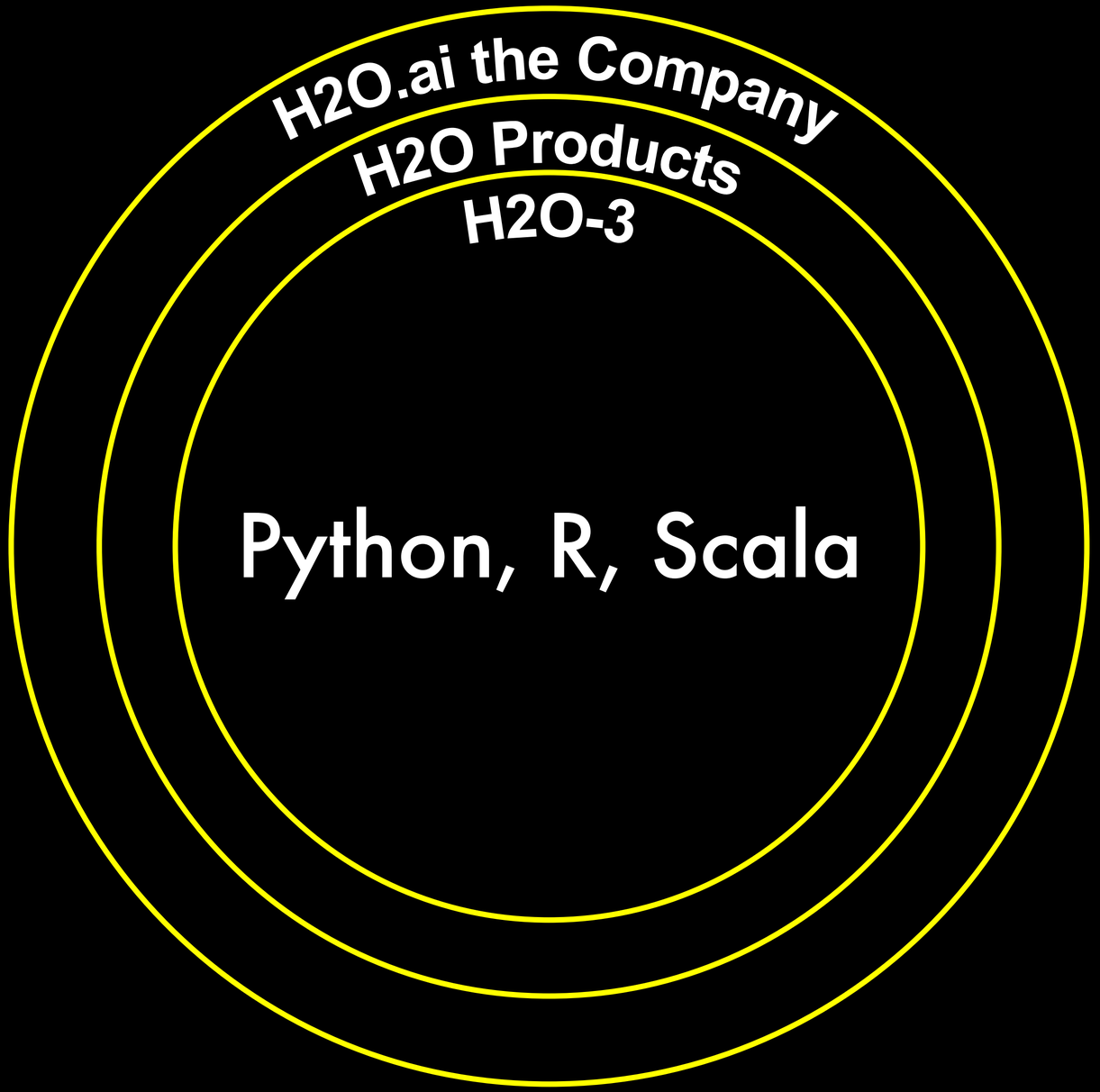

H2O.ai the Company
H2O Products

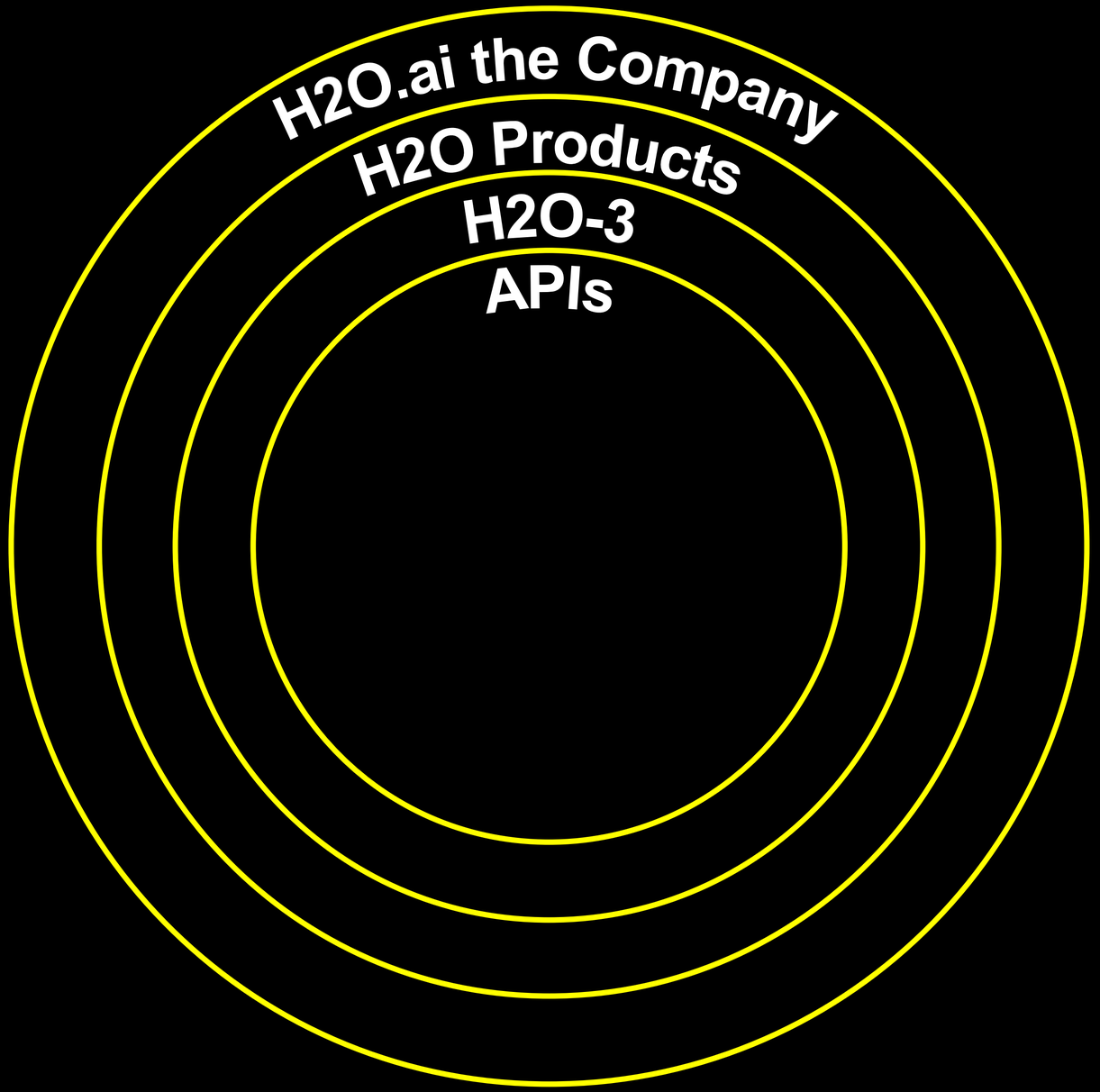

H2O.ai the Company
H2O Products
• H2O-3
• Enterprise Steam
• Sparkling Water
• Driverless AI
• H2O4GPU

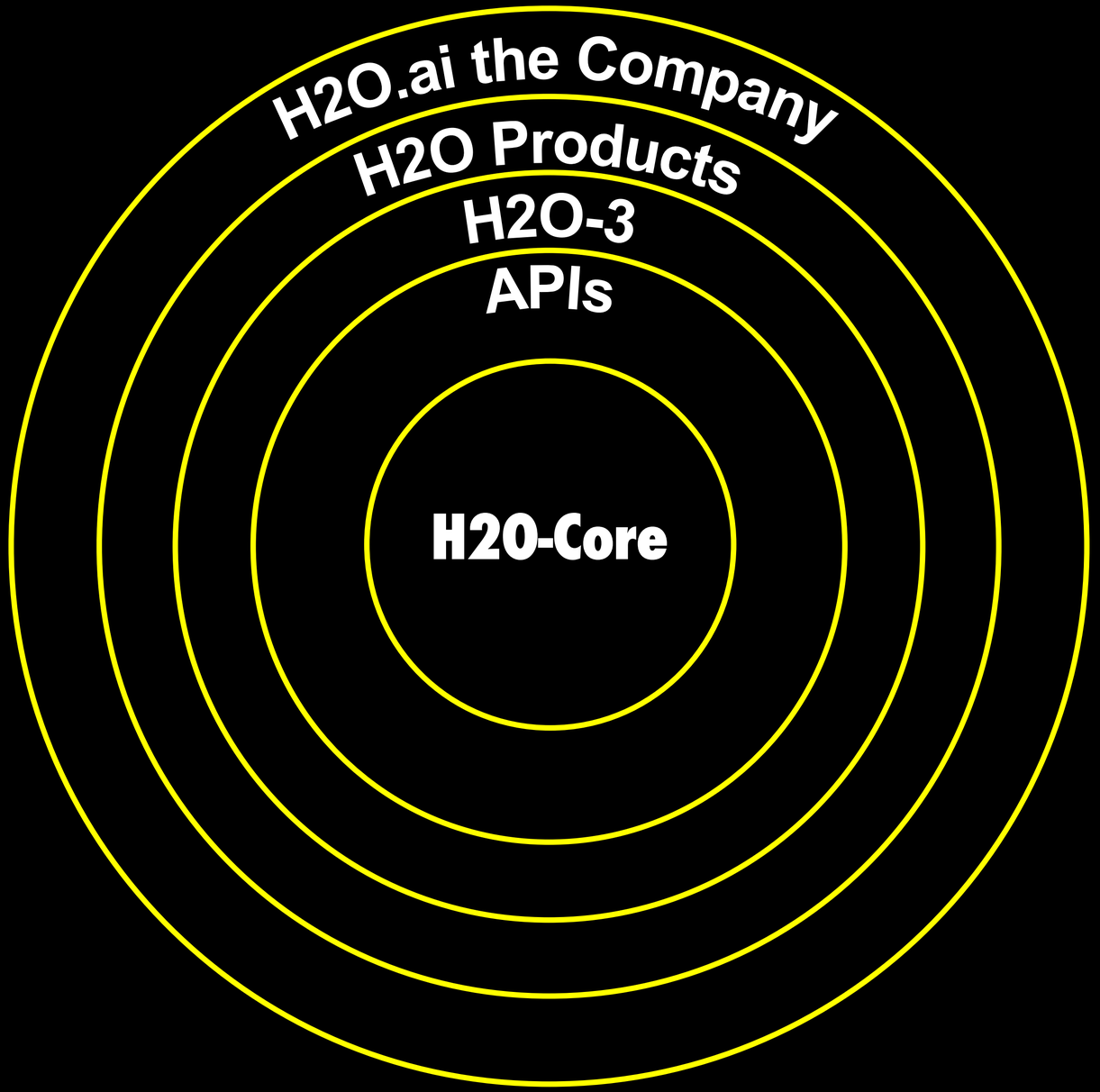

H2O.ai the Company
H2O Products
H2O-3
Enterprise Steam
Sparkling Water
Driverless AI
H2O4GPU

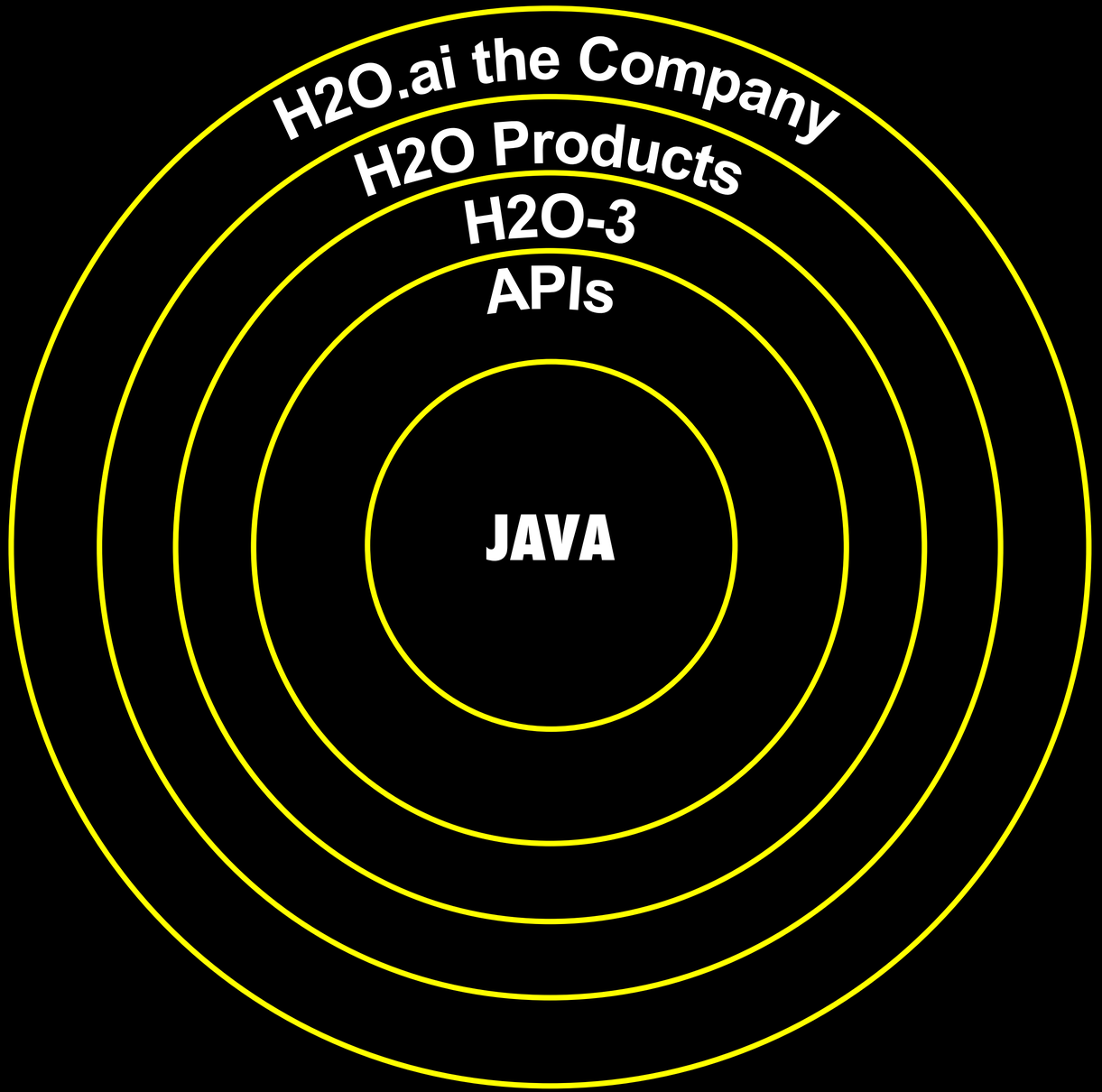

H2O.ai the Company
H2O Products
H2O-3


H2O.ai the Company
H2O Products
H2O-3
The APIs

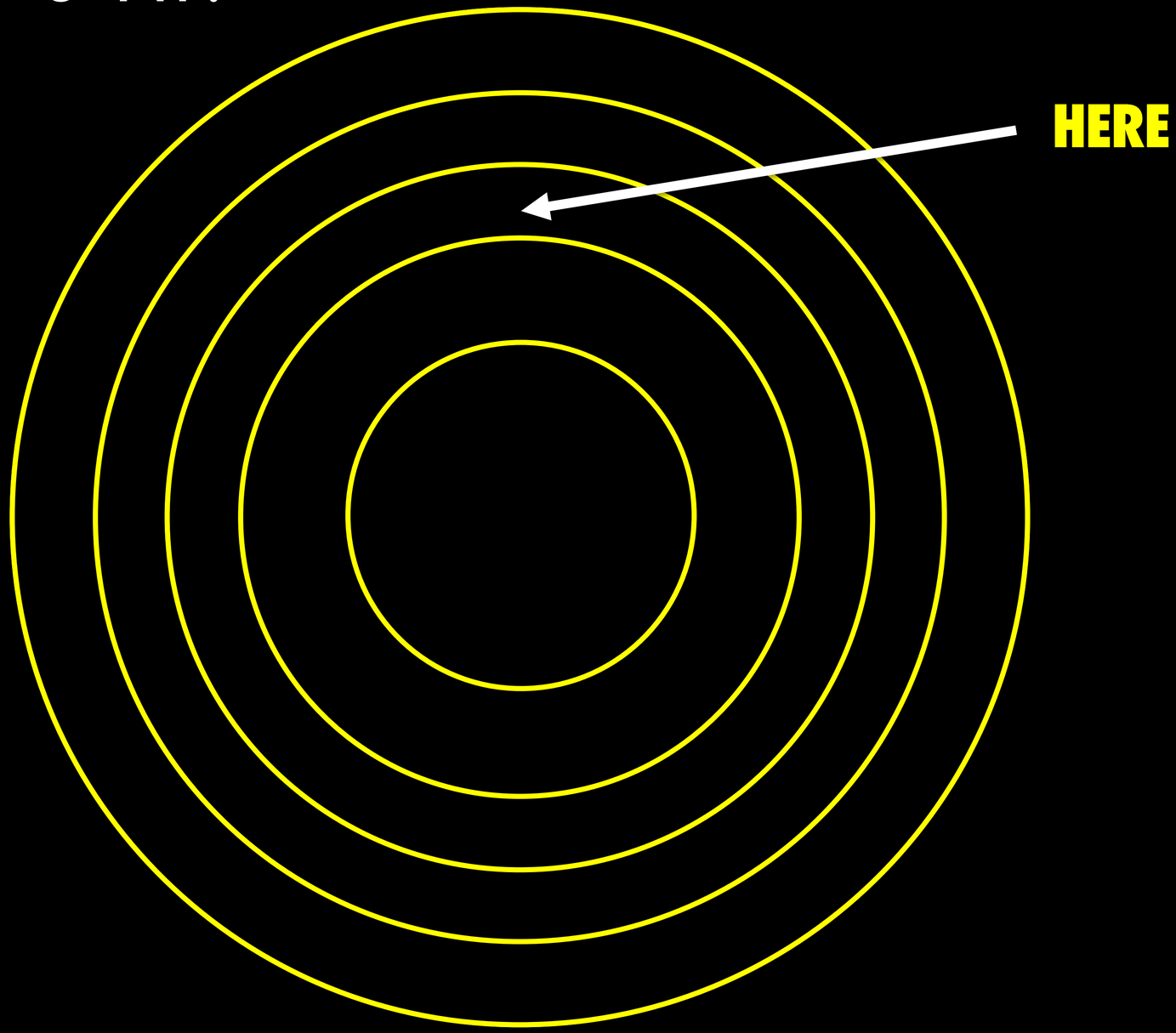

H2O.ai the Company
H2O Products
H2O-3
Python, R, Scala

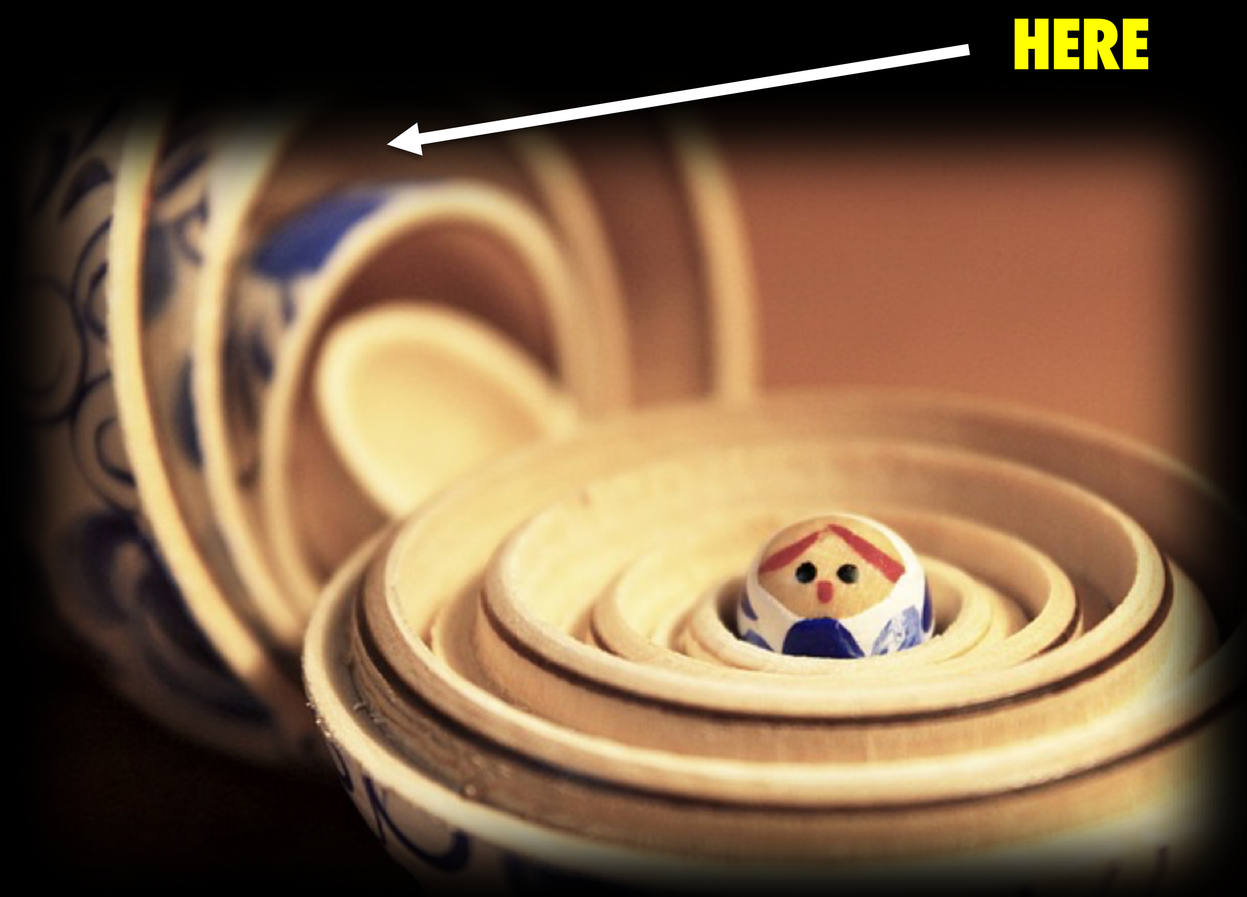

H2O.ai the Company
H2O Products
H2O-3
APIs


H2O.ai the Company
H2O Products
H2O-3
APIs
H2O-Core

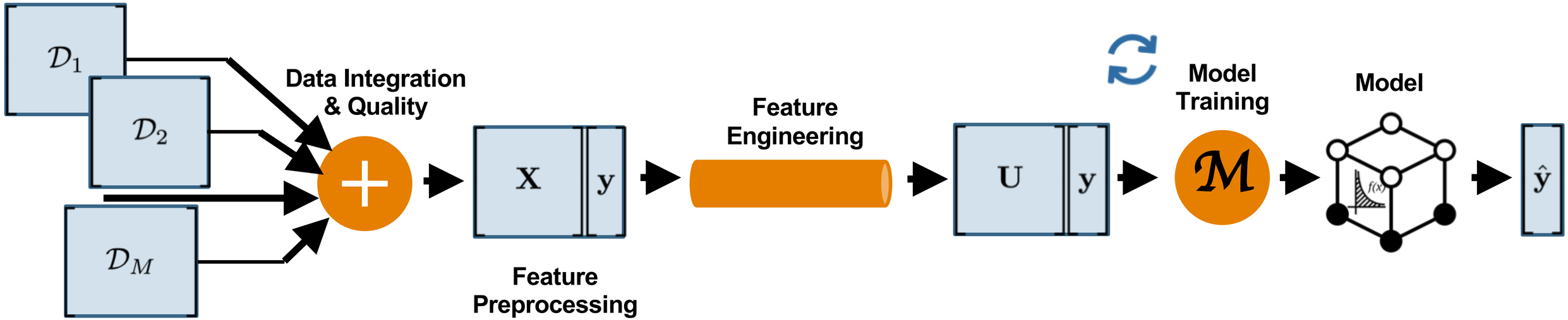

H2O.ai the Company
H2O Products
H2O-3
APIs
JAVA

Where does H2O-3 Fit?

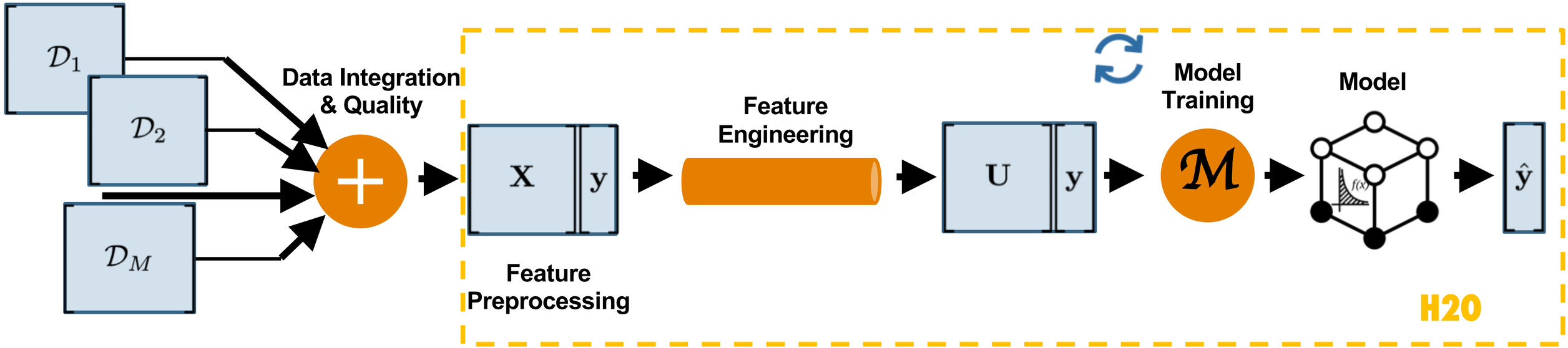



HERE

The Machine Learning Pipeline



This is Where H2O-3 Helps



Everything in One Place

Everything in One Place

Current Algorithm Overview

Statistical Analysis

- Linear Models (GLM)
- Naïve Bayes

Ensembles

- Random Forest
- Distributed Trees
- Gradient Boosting Machine
- Stacking / Super Learner

Deep Neural Networks

- MLP
- Autoencoder
  - Anomaly Detection
  - Deep Features

Clustering

- K-Means (Auto-*K*)

Dimension Reduction

- Principal Component Analysis
- Generalized Low Rank Models

Word Embedding

- Word2Vec

Time Series

- iSAX

Machine Learning Tuning

- Hyperparameter Search
- Early Stopping

Scientific Advisory Council

Dr. Trevor Hastie

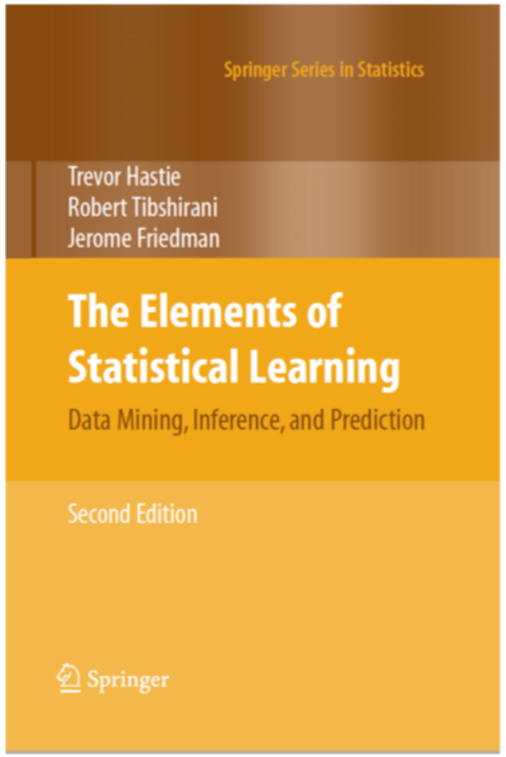
- PhD in Statistics, Stanford University
- John A. Overdeck Professor of Mathematics, Stanford University
- Co-author, *The Elements of Statistical Learning: Prediction, Inference and Data Mining*
- Co-author, *Generalized Additive Models*
- 108,404 citations (via Google Scholar)

Dr. Robert Tibshirani

- PhD in Statistics, Stanford University
- Professor of Statistics and Health Research and Policy, Stanford University
- COPPS Presidents' Award recipient
- Co-author, *The Elements of Statistical Learning: Prediction, Inference and Data Mining*
- Author, *Regression Shrinkage and Selection via the Lasso*
- Co-author, *An Introduction to the Bootstrap*

Dr. Steven Boyd

- PhD in Electrical Engineering and Computer Science, UC Berkeley
- Professor of Electrical Engineering and Computer Science, Stanford University
- Co-author, Convex Optimization
- Co-author, Linear Matrix Inequalities in System and Control Theory
- Co-author, *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*



H$_2$O.ai

H2O Core

What Makes Machine learning at Scale Possible

How H2O Core Works

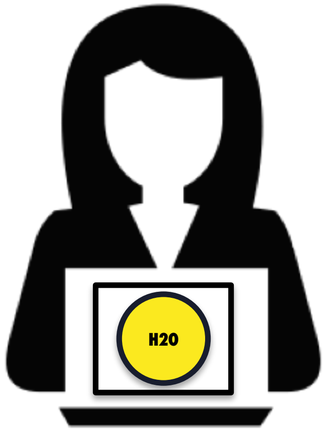
(just a java application)

How H2O Core Works

How H2O Core Works

on a laptop

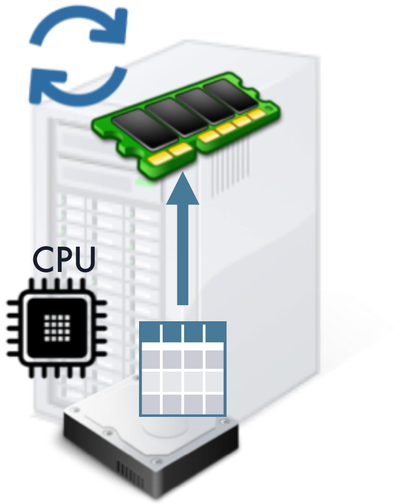
on a virtual machine



in a container

H2O Core

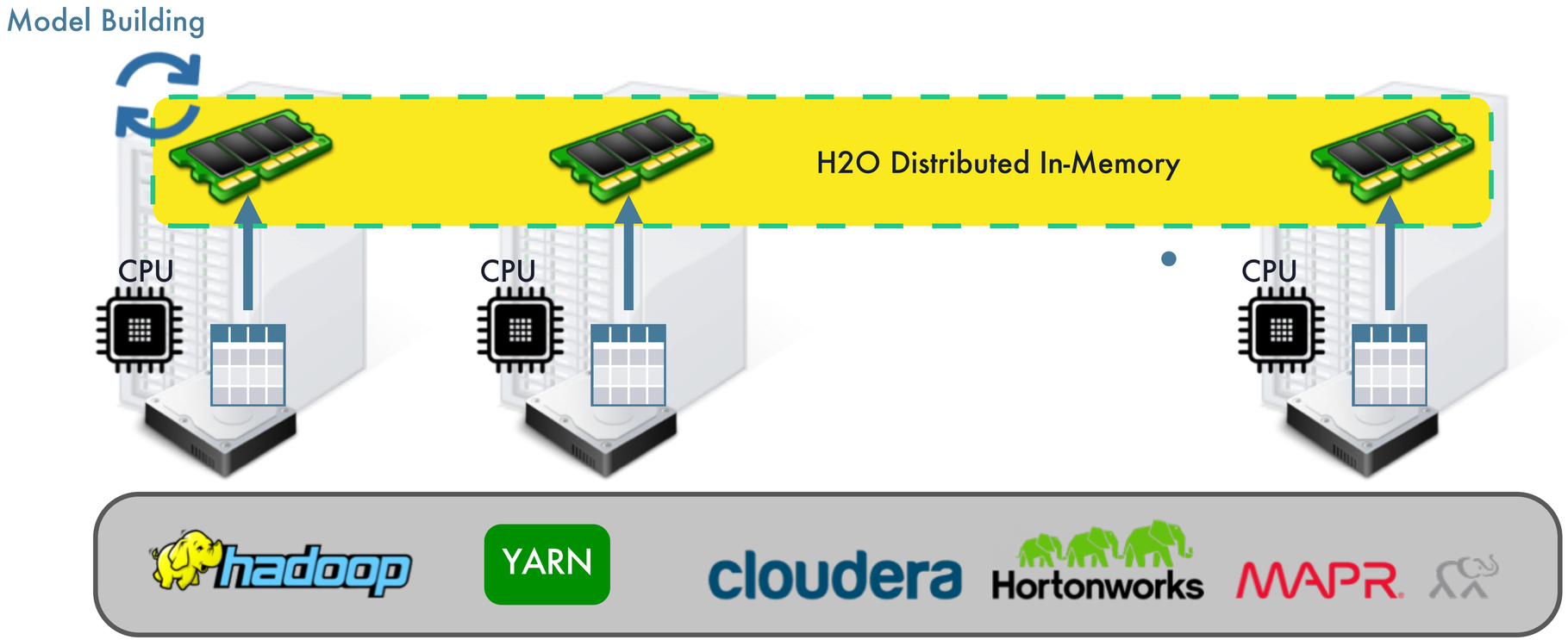
Model Building



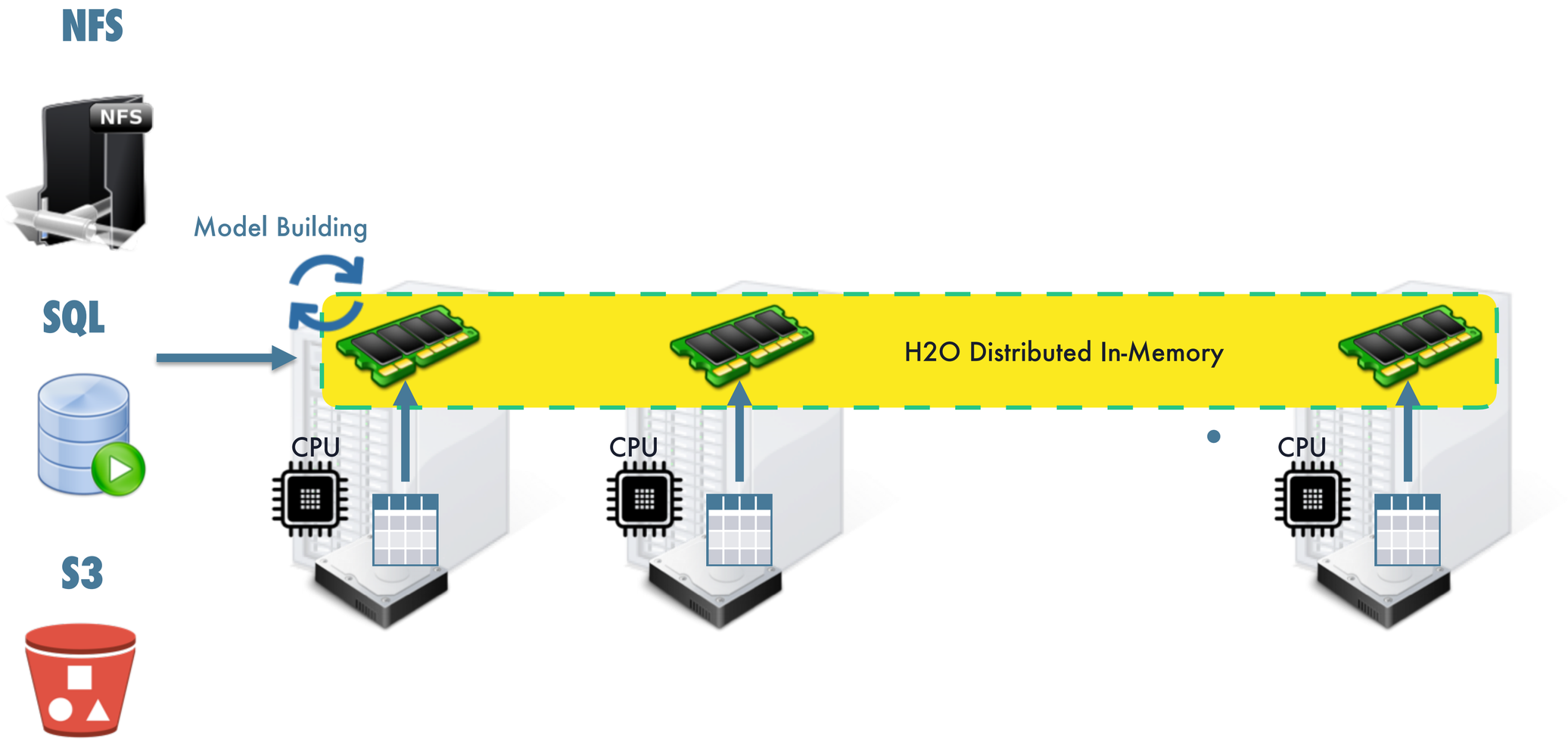
H2O Core

H2O Core

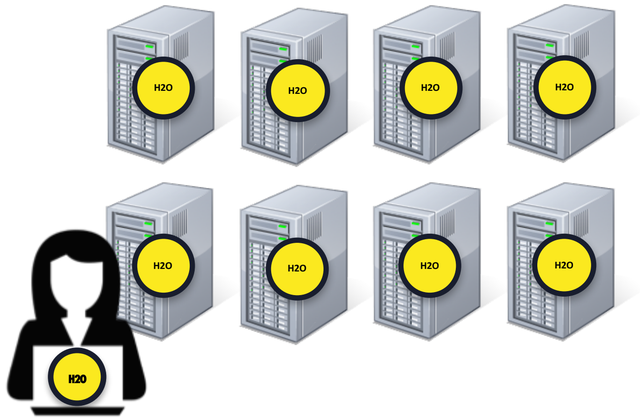
H2O Core with Other Data Sources



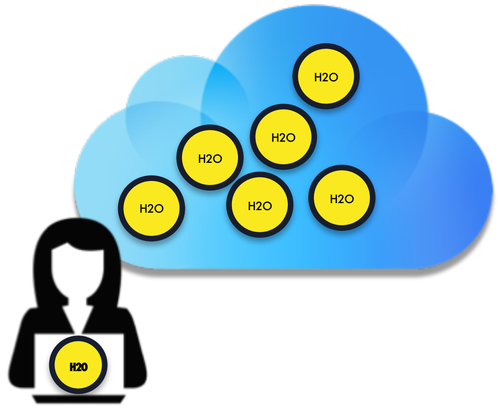
H2O Core with Other Data Sources



H2O Distributed Environments



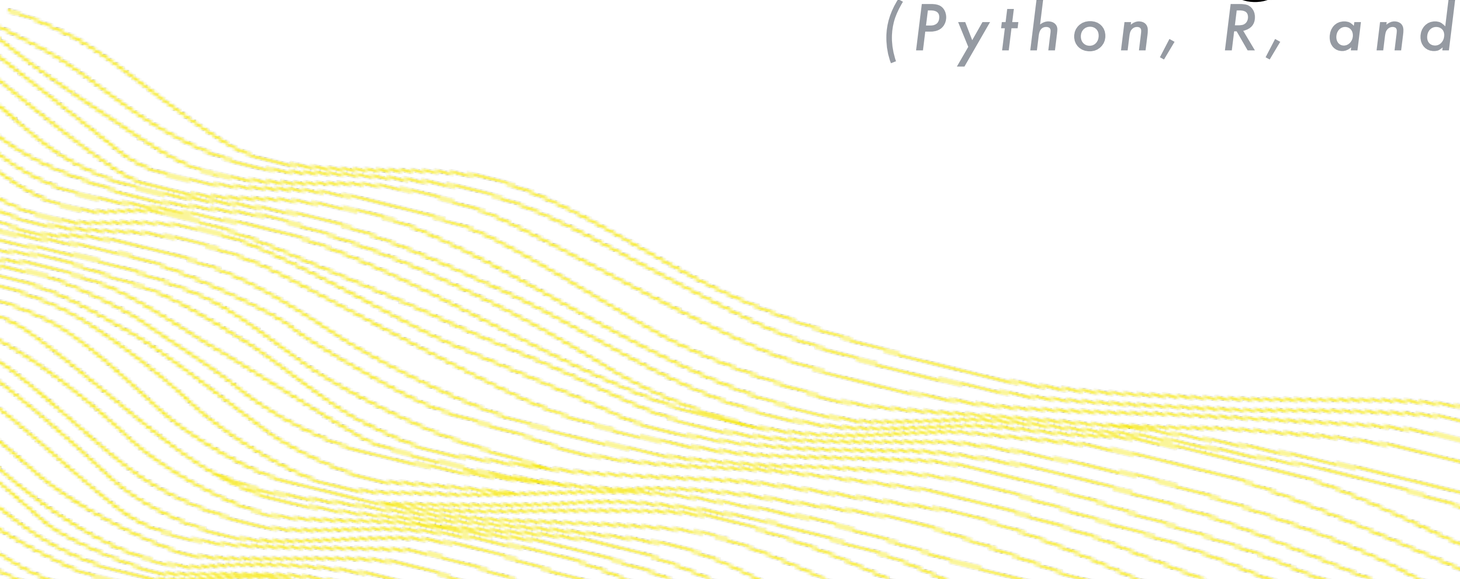
distributed across multiple machines



distributed on the cloud

Interfacing with H2O

(Python, R, and Flow)



Python API

File Edit View Insert Cell Kernel Help Python 3

Code CellToolbar

In [ ]:

```
import h2o
from h2o.estimators.glm import H2OGeneralizedLinearEstimator
h2o.init()

# import the boston dataset:
# this dataset looks at features of the boston suburbs and predicts median housing prices
# the original dataset can be found at https://archive.ics.uci.edu/ml/datasets/Housing
boston = h2o.import_file("https://s3.amazonaws.com/h2o-public-test-data/smalldata/gbm_test/Bosto

# set the predictor names and the response column name
predictors = boston.columns[:-1]
# set the response column to "medv", the median value of owner-occupied homes in $1000's
response = "medv"

# convert the chas column to a factor (chas = Charles River dummy variable (= 1 if tract bounds
boston['chas'] = boston['chas'].asfactor()

# split into train and validation sets
train, valid = boston.split_frame(ratios = [.8])


# try using the `alpha` parameter:
# initialize the estimator then train the model
boston_glm = H2OGeneralizedLinearEstimator(alpha = .25)
boston_glm.train(x = predictors, y = response, training_frame = train, validation_frame = valid)

# print the mse for the validation data
print(boston_glm.mse(valid=True))
```

Python API

Import the python package

```
import h2o
from h2o.estimators.glm import H2OGeneralizedLinearEstimator
h2o.init()

# import the boston dataset:
# this dataset looks at features of the boston suburbs and predi
# the original dataset can be found at https://archive.ics.uci.e
boston = h2o.import_file("https://s3.amazonaws.com/h2o-public-te
```

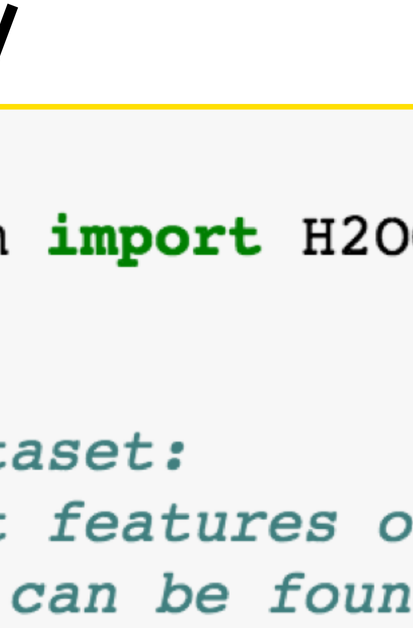
Python API

Launch an H2O cluster

```
import h2o
from h2o.estimators.glm import H2OGeneralizedLinearEstimator
h2o.init()

# import the boston dataset:
# this dataset looks at features of the boston suburbs and predi
# the original dataset can be found at https://archive.ics.uci.e
boston = h2o.import_file("https://s3.amazonaws.com/h2o-public-te
```

Python API

Import a Dataset



```
import h2o
from h2o.estimators.glm import H2OGeneralizedLinearEstimator
h2o.init()

# import the boston dataset:
# this dataset looks at features of the boston suburbs and predi
# the original dataset can be found at https://archive.ics.uci.e
boston = h2o.import_file("https://s3.amazonaws.com/h2o-public-te
```

Python API

Build a Model

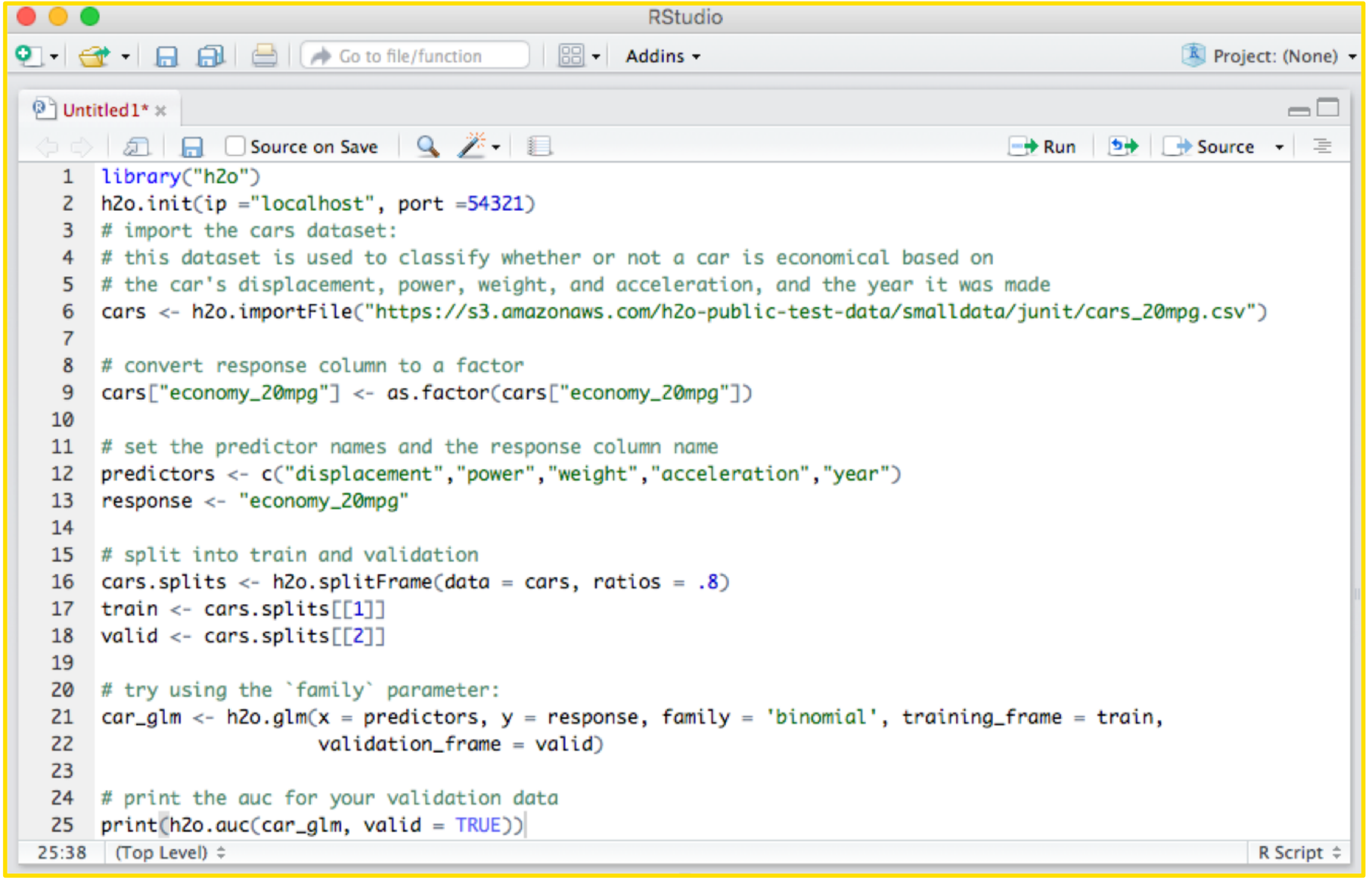

```
# initialize the estimator then train the model
boston_glm = H2OGeneralizedLinearEstimator(alpha = .25)
boston_glm.train(x = predictors, y = response, training_frame = tra

# print the mse for the validation data
print(boston_glm.mse(valid=True))
```

Python API

Get Model Performance

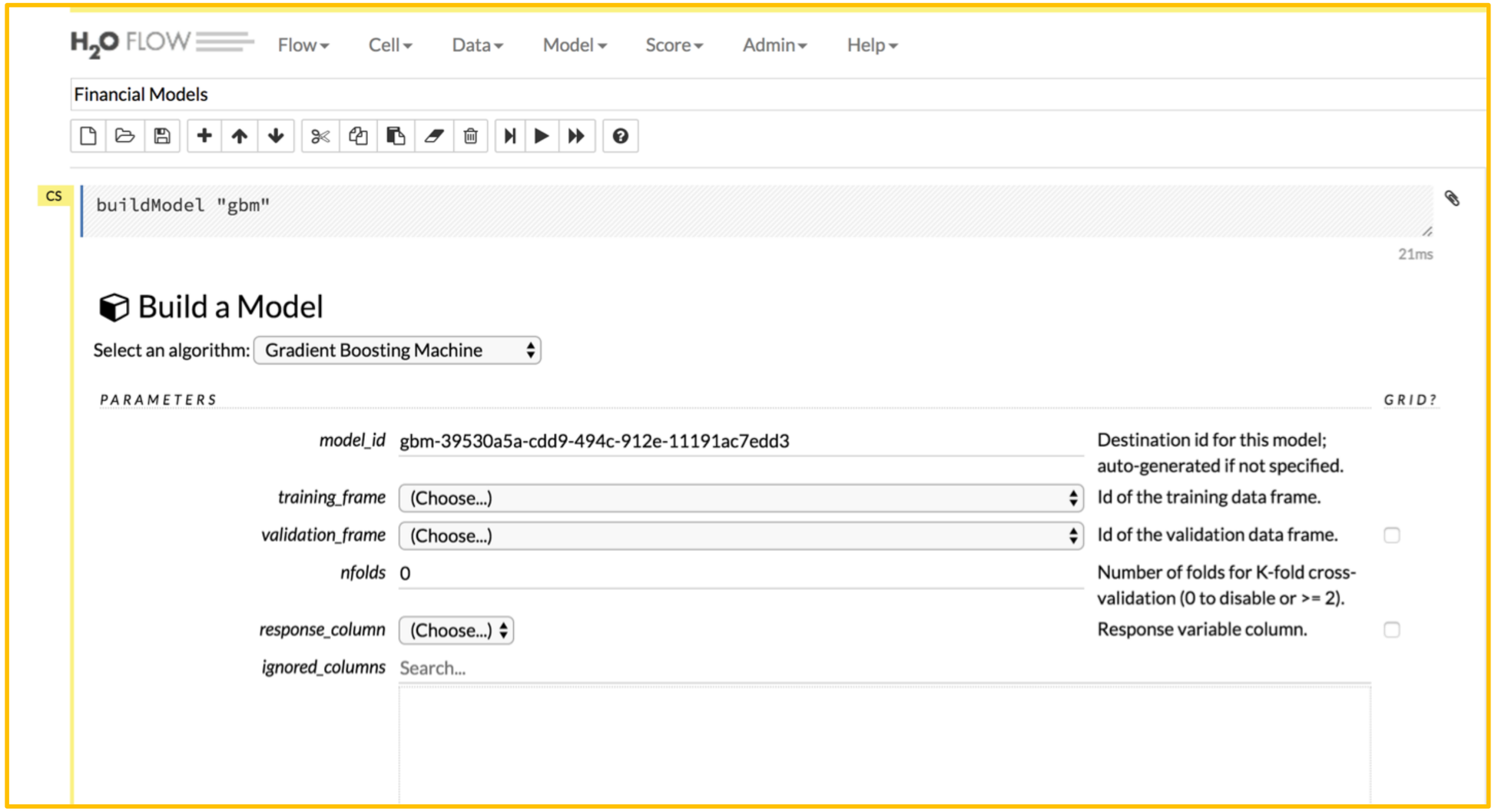

```
# initialize the estimator then train the model
boston_glm = H2OGeneralizedLinearEstimator(alpha = .25)
boston_glm.train(x = predictors, y = response, training_frame = tra

# print the mse for the validation data
print(boston_glm.mse(valid=True))
```

R API


RStudio
Go to file/function
Addins
Project: (None)
Untitled1*
Source on Save
Run
Source
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

H2O.ai

Flow: Custom Web UI



The H2O Python API*

(how the client & server communicate)

*Same applies for R

Client & Cluster Communication

H2O Client

REST/
JSON

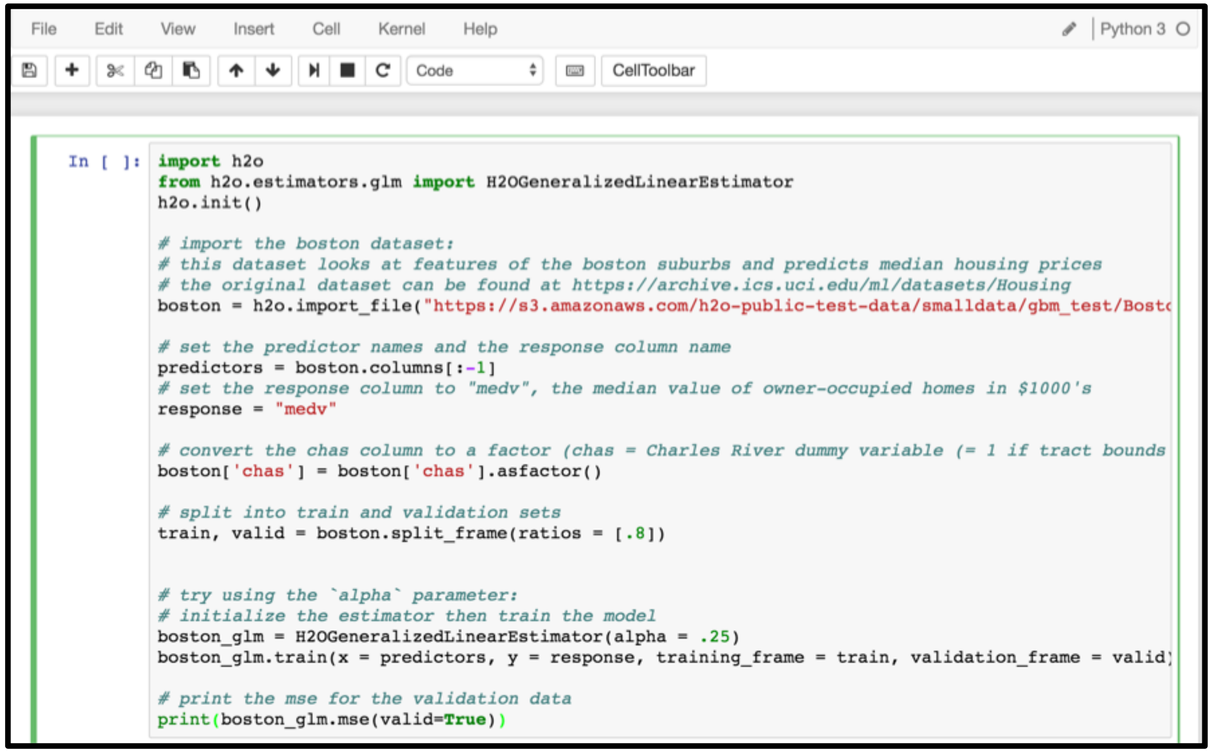
H2O Cluster

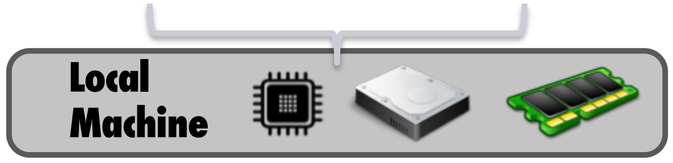
Communication Layers: A Command

Python Commands

the Client

Importing Big Data with Python Code

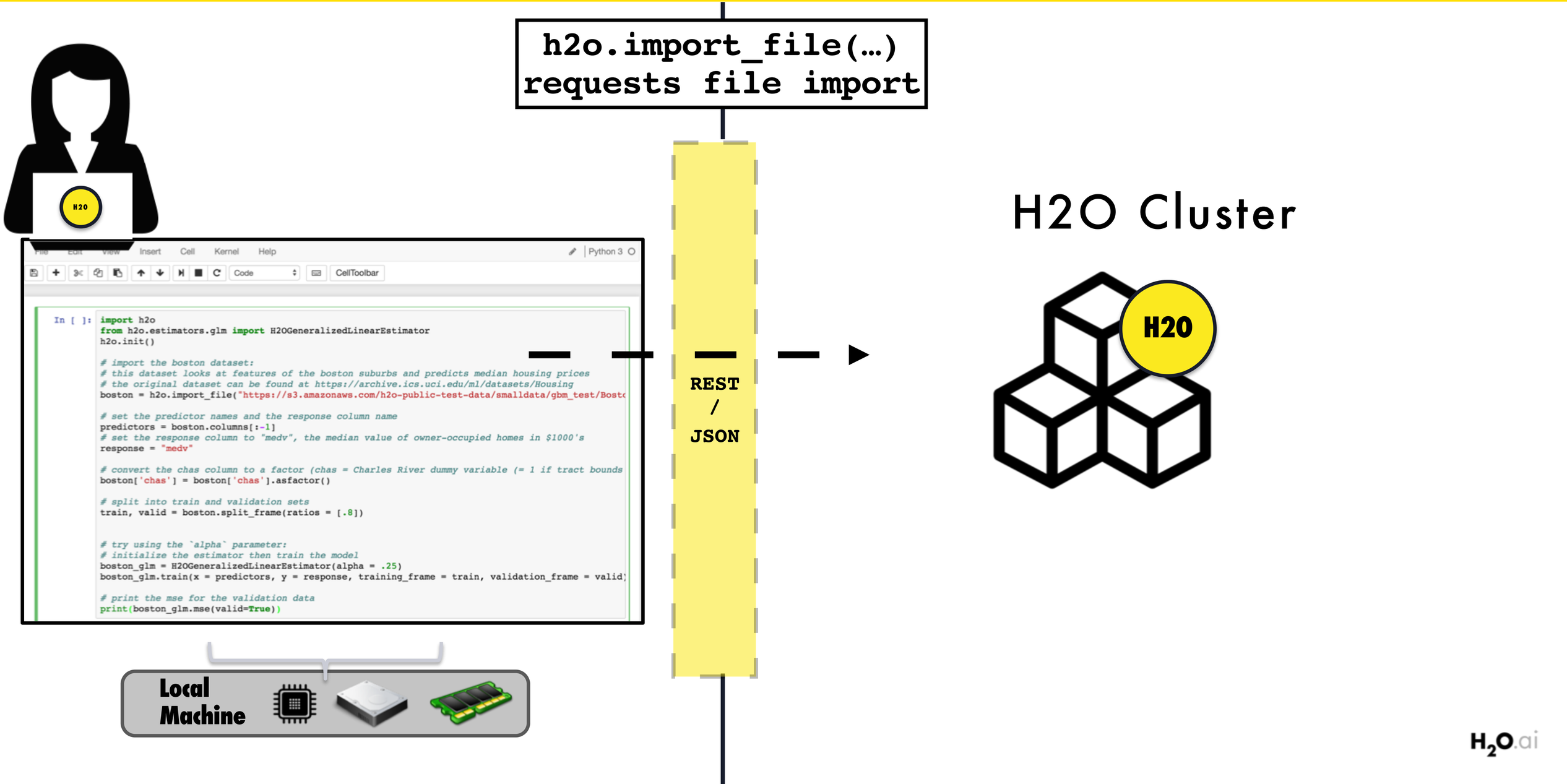




Path to Your Dataset

`h2o.import_file(…)`

Communication

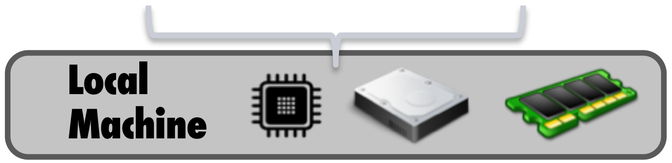


Cluster Does Heavy Lifting

File Edit View Insert Cell Kernel Help Python 3

Code CellToolbar

```
In [ ]: import h2o
        from h2o.estimators.glm import H2OGeneralizedLinearEstimator
        h2o.init()

        # import the boston dataset:
        # this dataset looks at features of the boston suburbs and predicts median housing prices
        # the original dataset can be found at https://archive.ics.uci.edu/ml/datasets/Housing
        boston = h2o.import_file("https://s3.amazonaws.com/h2o-public-test-data/smalldata/gbm_test/Bost

        # set the predictor names and the response column name
        predictors = boston.columns[:-1]
        # set the response column to "medv", the median value of owner-occupied homes in $1000's
        response = "medv"

        # convert the chas column to a factor (chas = Charles River dummy variable (= 1 if tract bounds
        boston['chas'] = boston['chas'].asfactor()

        # split into train and validation sets
        train, valid = boston.split_frame(ratios = [.8])


        # try using the `alpha` parameter:
        # initialize the estimator then train the model
        boston_glm = H2OGeneralizedLinearEstimator(alpha = .25)
        boston_glm.train(x = predictors, y = response, training_frame = train, validation_frame = valid)

        # print the mse for the validation data
        print(boston_glm.mse(valid=True))
```



REST/
JSON

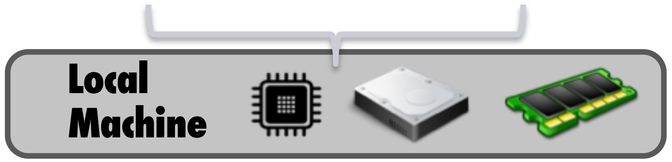
H2O Cluster

Cluster Does Heavy Lifting

File Edit View Insert Cell Kernel Help Python 3

Code CellToolbar

```
In [ ]: import h2o
        from h2o.estimators.glm import H2OGeneralizedLinearEstimator
        h2o.init()

        # import the boston dataset:
        # this dataset looks at features of the boston suburbs and predicts median housing prices
        # the original dataset can be found at https://archive.ics.uci.edu/ml/datasets/Housing
        boston = h2o.import_file("https://s3.amazonaws.com/h2o-public-test-data/smalldata/gbm_test/Bost

        # set the predictor names and the response column name
        predictors = boston.columns[:-1]
        # set the response column to "medv", the median value of owner-occupied homes in $1000's
        response = "medv"

        # convert the chas column to a factor (chas = Charles River dummy variable (= 1 if tract bounds
        boston['chas'] = boston['chas'].asfactor()

        # split into train and validation sets
        train, valid = boston.split_frame(ratios = [.8])


        # try using the `alpha` parameter:
        # initialize the estimator then train the model
        boston_glm = H2OGeneralizedLinearEstimator(alpha = .25)
        boston_glm.train(x = predictors, y = response, training_frame = train, validation_frame = valid

        # print the mse for the validation data
        print(boston_glm.mse(valid=True))
```



REST/
JSON

H2O Cluster

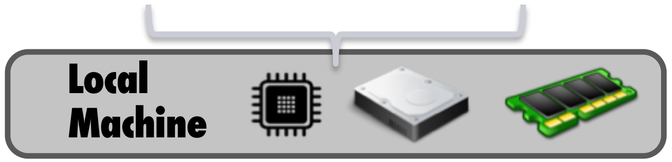
H2O.ai

Cluster Does Heavy Lifting

File Edit View Insert Cell Kernel Help Python 3

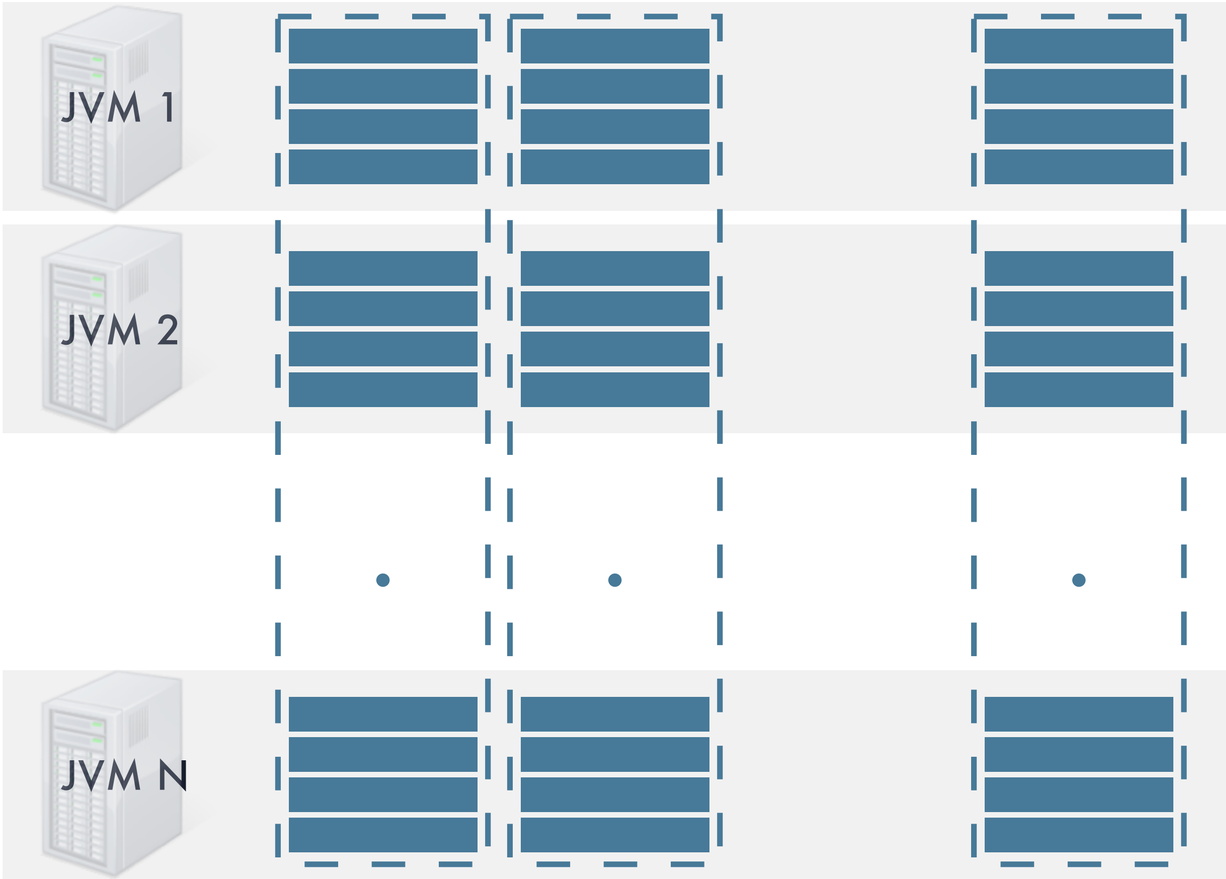
Code CellToolbar

```
In [ ]: import h2o
        from h2o.estimators.glm import H2OGeneralizedLinearEstimator
        h2o.init()

        # import the boston dataset:
        # this dataset looks at features of the boston suburbs and predicts median housing prices
        # the original dataset can be found at https://archive.ics.uci.edu/ml/datasets/Housing
        boston = h2o.import_file("https://s3.amazonaws.com/h2o-public-test-data/smalldata/gbm_test/Bost

        # set the predictor names and the response column name
        predictors = boston.columns[:-1]
        # set the response column to "medv", the median value of owner-occupied homes in $1000's
        response = "medv"

        # convert the chas column to a factor (chas = Charles River dummy variable (= 1 if tract bounds
        boston['chas'] = boston['chas'].asfactor()

        # split into train and validation sets
        train, valid = boston.split_frame(ratios = [.8])


        # try using the `alpha` parameter:
        # initialize the estimator then train the model
        boston_glm = H2OGeneralizedLinearEstimator(alpha = .25)
        boston_glm.train(x = predictors, y = response, training_frame = train, validation_frame = valid)

        # print the mse for the validation data
        print(boston_glm.mse(valid=True))
```



REST/
JSON

H2O Cluster



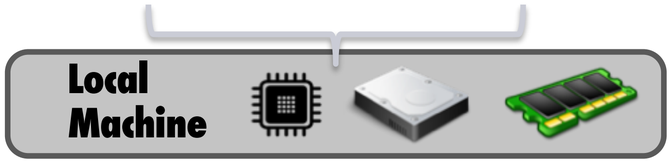
H2O.ai

Cluster Does Heavy Lifting

File Edit View Insert Cell Kernel Help Python 3

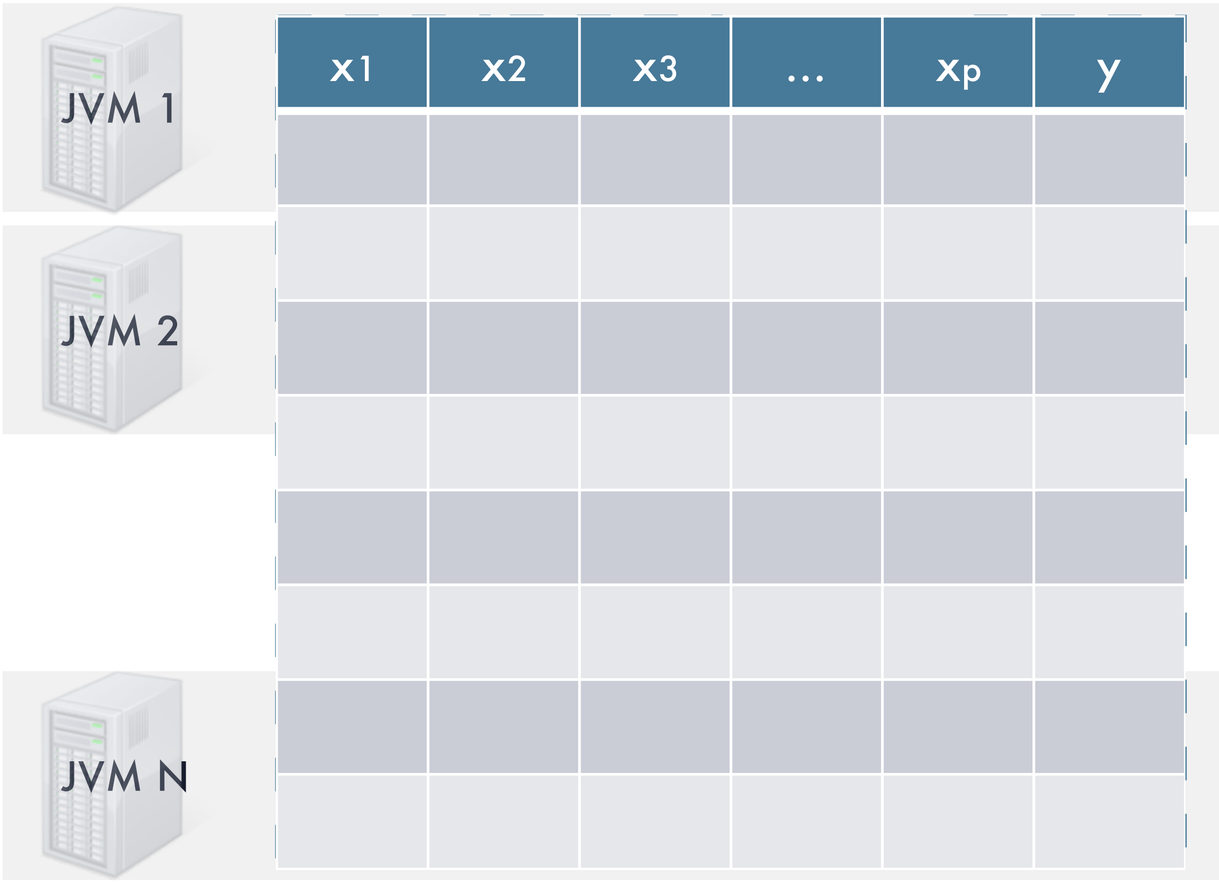
Code CellToolbar

```
In [ ]: import h2o
        from h2o.estimators.glm import H2OGeneralizedLinearEstimator
        h2o.init()

        # import the boston dataset:
        # this dataset looks at features of the boston suburbs and predicts median housing prices
        # the original dataset can be found at https://archive.ics.uci.edu/ml/datasets/Housing
        boston = h2o.import_file("https://s3.amazonaws.com/h2o-public-test-data/smalldata/gbm_test/Bost

        # set the predictor names and the response column name
        predictors = boston.columns[:-1]
        # set the response column to "medv", the median value of owner-occupied homes in $1000's
        response = "medv"

        # convert the chas column to a factor (chas = Charles River dummy variable (= 1 if tract bounds
        boston['chas'] = boston['chas'].asfactor()

        # split into train and validation sets
        train, valid = boston.split_frame(ratios = [.8])


        # try using the `alpha` parameter:
        # initialize the estimator then train the model
        boston_glm = H2OGeneralizedLinearEstimator(alpha = .25)
        boston_glm.train(x = predictors, y = response, training_frame = train, validation_frame = valid

        # print the mse for the validation data
        print(boston_glm.mse(valid=True))
```



REST/
JSON

H2O Cluster



H2O.ai

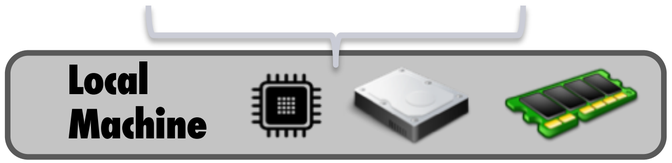
H2O Client

cluster returns frame key

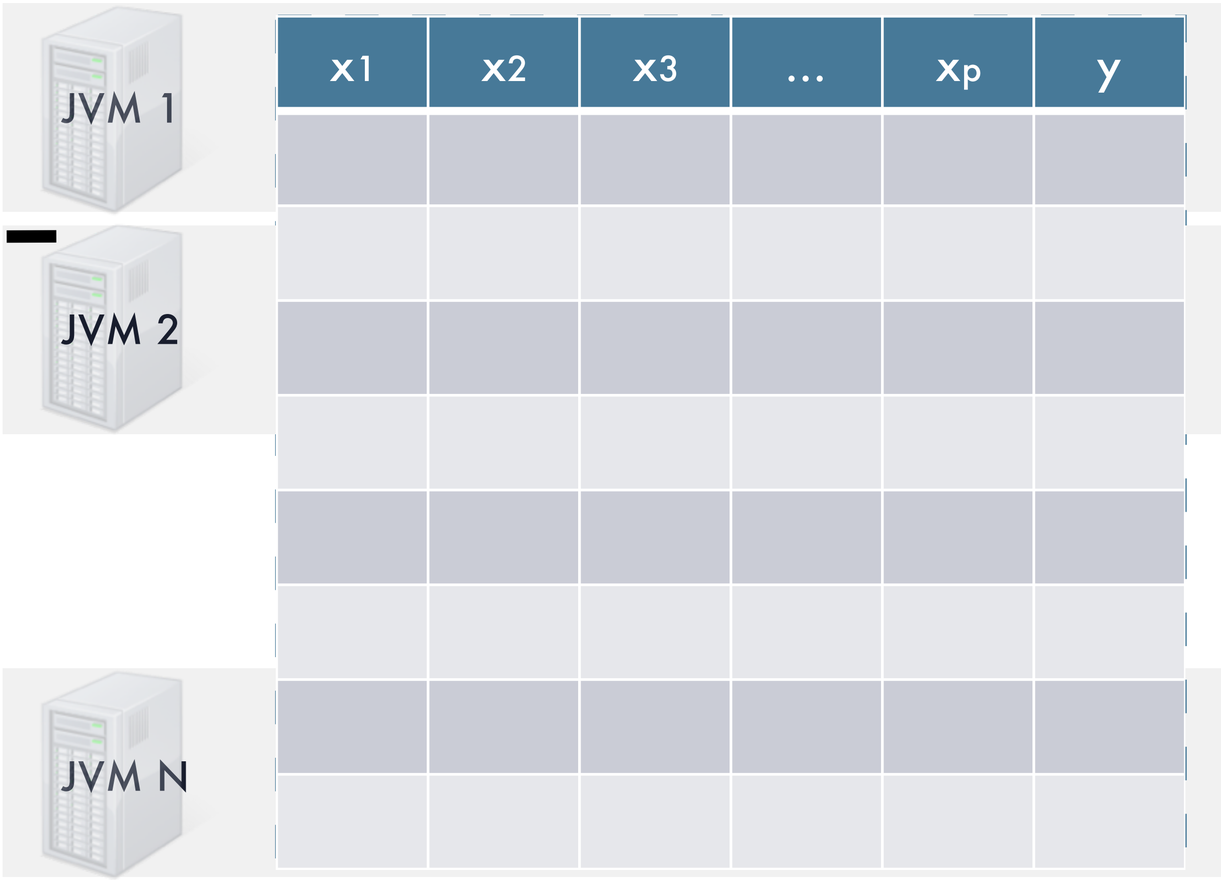
File Edit View Insert Cell Kernel Help Python 3

Code CellToolbar

```
In [ ]: import h2o
        from h2o.estimators.glm import H2OGeneralizedLinearEstimator
        h2o.init()

        # import the boston dataset:
        # this dataset looks at features of the boston suburbs and predicts median housing prices
        # the original dataset can be found at https://archive.ics.uci.edu/ml/datasets/Housing
        boston = h2o.import_file("https://s3.amazonaws.com/h2o-public-test-data/smalldata/gbm_test/Bost

        # set the predictor names and the response column name
        predictors = boston.columns[:-1]
        # set the response column to "medv", the median value of owner-occupied homes in $1000's
        response = "medv"

        # convert the chas column to a factor (chas = Charles River dummy variable (= 1 if tract bounds
        boston['chas'] = boston['chas'].asfactor()

        # split into train and validation sets
        train, valid = boston.split_frame(ratios = [.8])


        # try using the `alpha` parameter:
        # initialize the estimator then train the model
        boston_glm = H2OGeneralizedLinearEstimator(alpha = .25)
        boston_glm.train(x = predictors, y = response, training_frame = train, validation_frame = valid

        # print the mse for the validation data
        print(boston_glm.mse(valid=True))
```



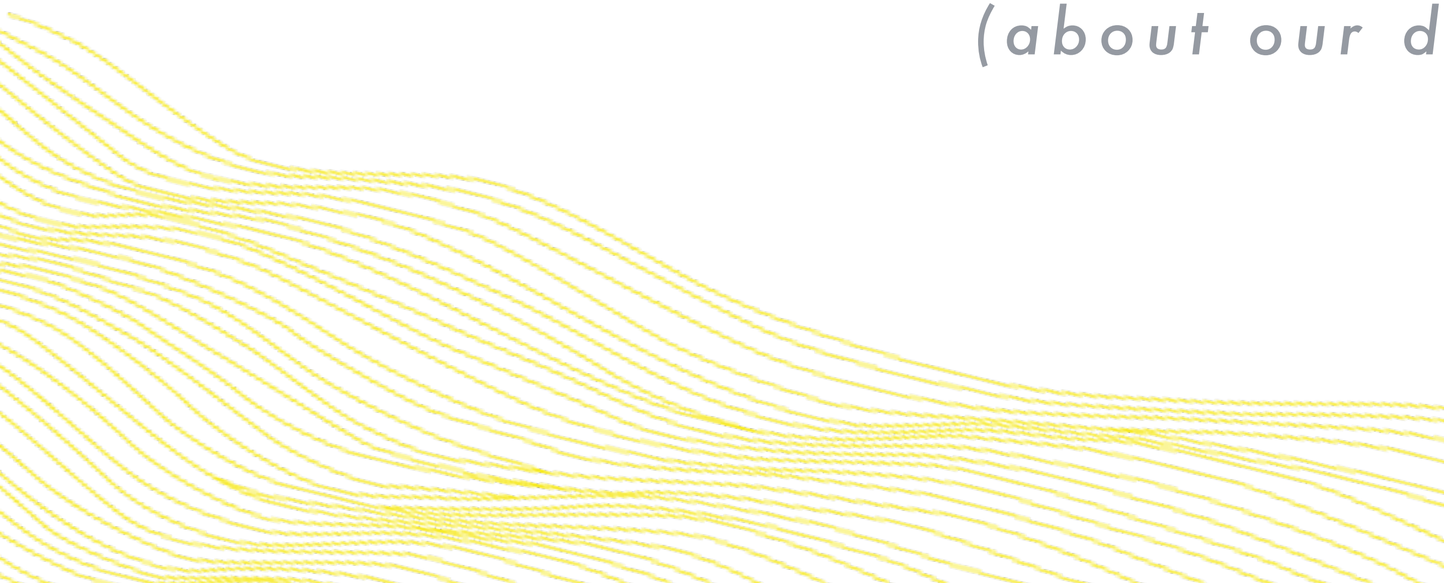
REST/ JSON



H2O.ai

The Use Case

(about our data)



Backorders Dataset

The Backorders Data

	sku	national_inv	lead_time	in_transit_qty	forecast_3_month	forecast_6_month
1	1113121	0	8	1	6	6
2	1113268	0	8	0	2	3
3	1113874	20	2	0	45	99
4	1114222	0	8	0	9	14
5	1114823	0	12	0	31	31
6	1115453	55	8	0	216	360

The Use Case: **End Goal**

Predict if a product will go on backorder

The Use Case: **End Goal**

Predict if a product will go on backorder
(**based on our historic backorders data**)

The Use Case: **End Goal**

Build a model to predict

our **target**:

Yes | No

The Use Case: **End Goal**

Build a model to predict
our target (**aka Ŷ**):
Yes | No

The Use Case: **End Goal**

Build a model to predict
our target (**aka response column**):
Yes | No

The Use Case: **End Goal**

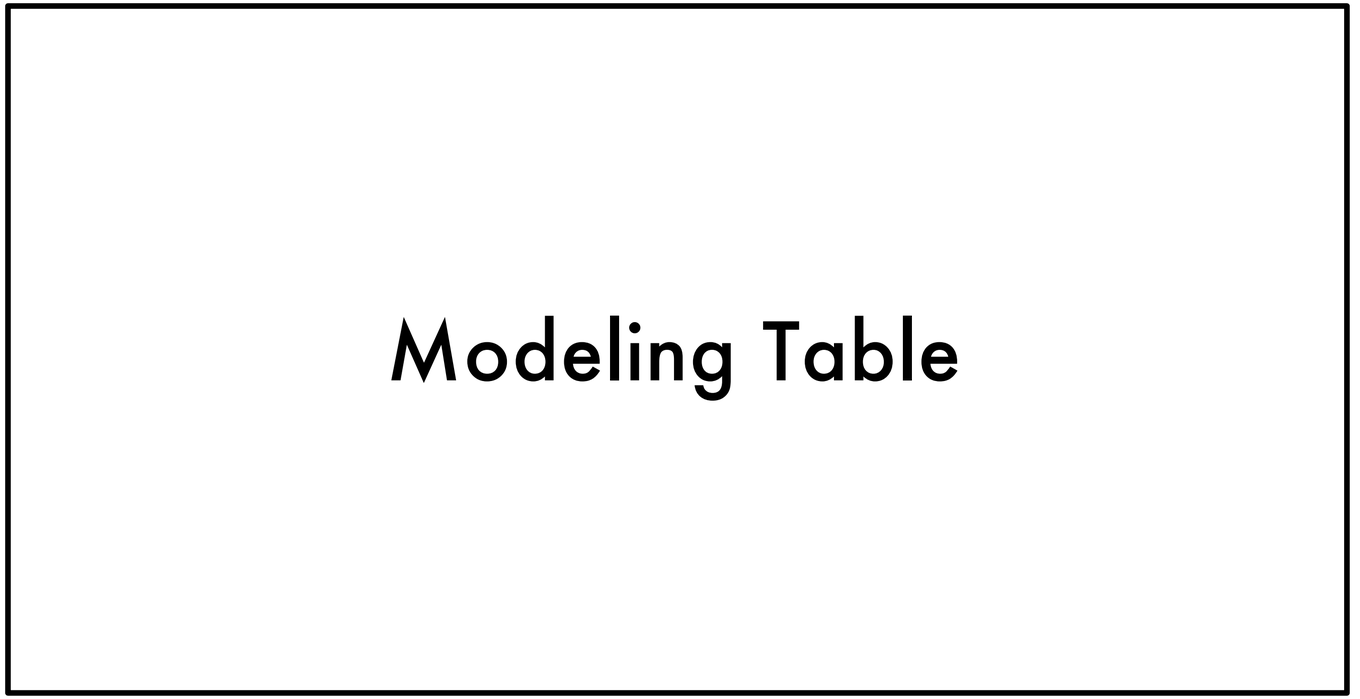
Build a model to predict our target:

Yes | No

Given Data: How to Learn?

How does an algorithm **learn from data** ?

Learning from Data



Learning from Data

sku	national_inv	lead_time	...	went_on_backorder
1113121	0	8	...	YES
1113268	0	8	...	NO
1113874	20	2	...	YES
1114222	0	8	...	YES
1114823	0	12	...	NO
1115453	55	8	...	NO
1115620	-34	8		

Learning from Data

observations

sku	national_inv	lead_time	...	went_on_backorder
1113121	0	8	...	YES
1113268	0	8	...	NO
1113874	20	2	...	YES
1114222	0	8	...	YES
1114823	0	12	...	NO
1115453	55	8	...	NO
1115620	-34	8		

Learning from Data

training ex.

sku	national_inv	lead_time	...	went_on_backorder
1113121	0	8	...	YES
1113268	0	8	...	NO
1113874	20	2	...	YES
1114222	0	8	...	YES
1114823	0	12	...	NO
1115453	55	8	...	NO
1115620	-34	8		

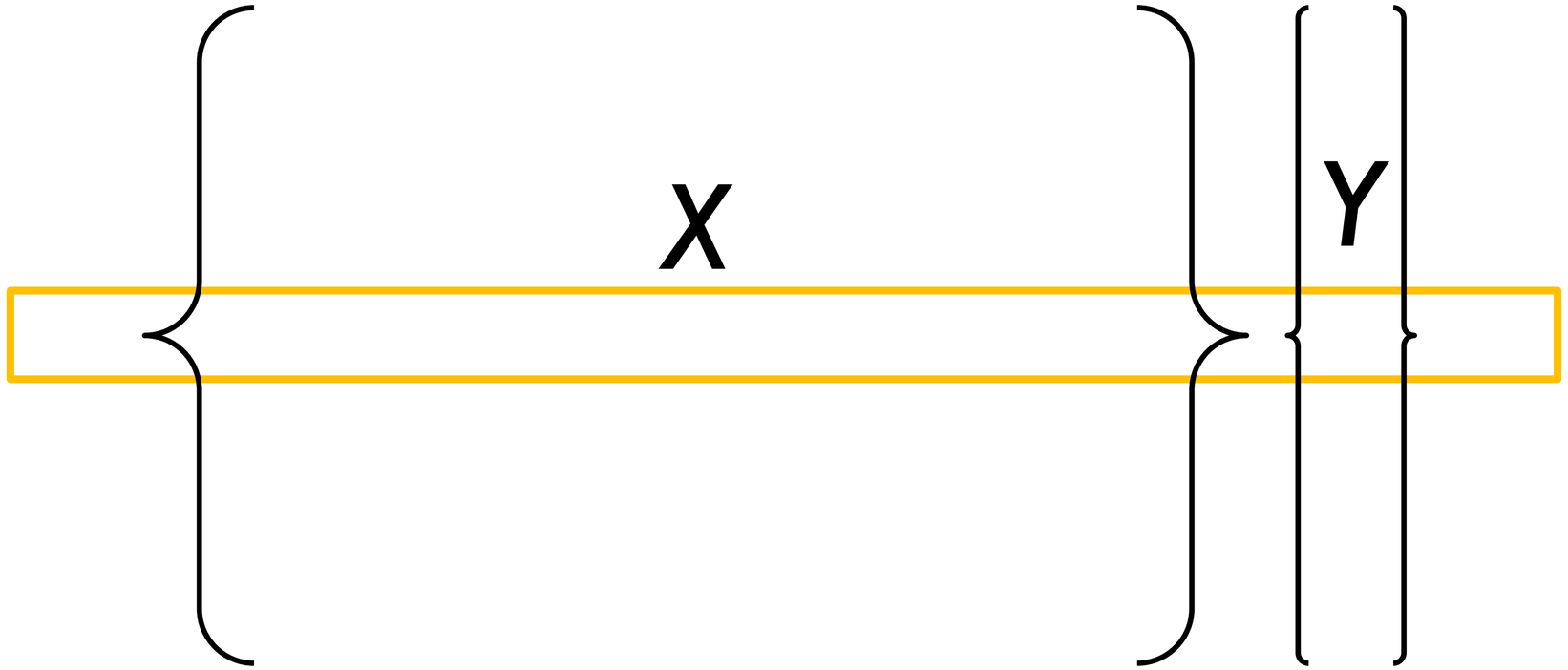
Learning from Data

(x, y)

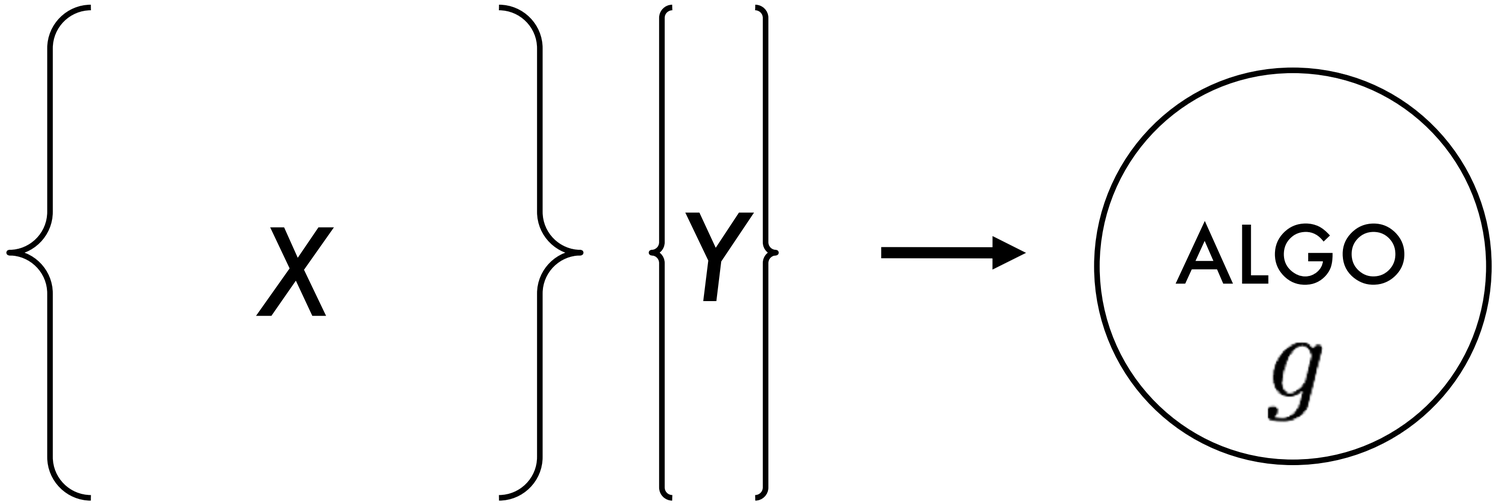
sku	national_inv	lead_time	...	went_on_backorder
1113121	0	8	...	YES
1113268	0	8	...	NO
1113874	20	2	...	YES
1114222	0	8	...	YES
1114823	0	12	...	NO
1115453	55	8	...	NO
1115620	-34	8		

Learning from Data

(x, *y*)



Learning from Data

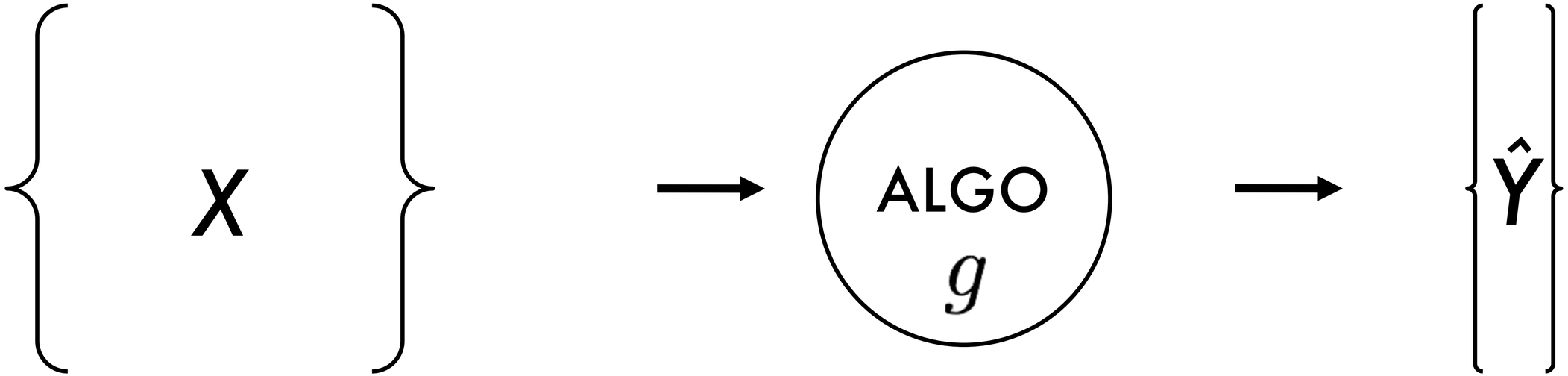


$$g : x \rightarrow y$$

Learn the Pattern

Learning from Data

New Input Data



Supervised Learning

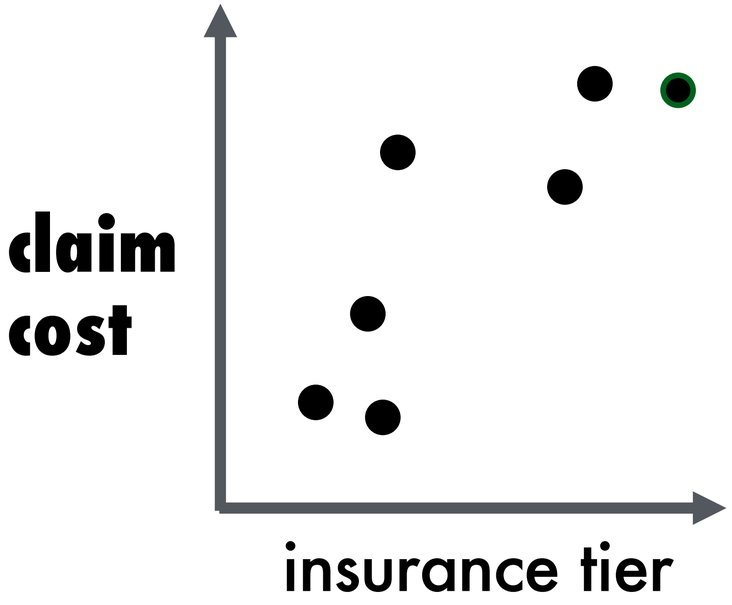
REGRESSION

How much will a claim cost?

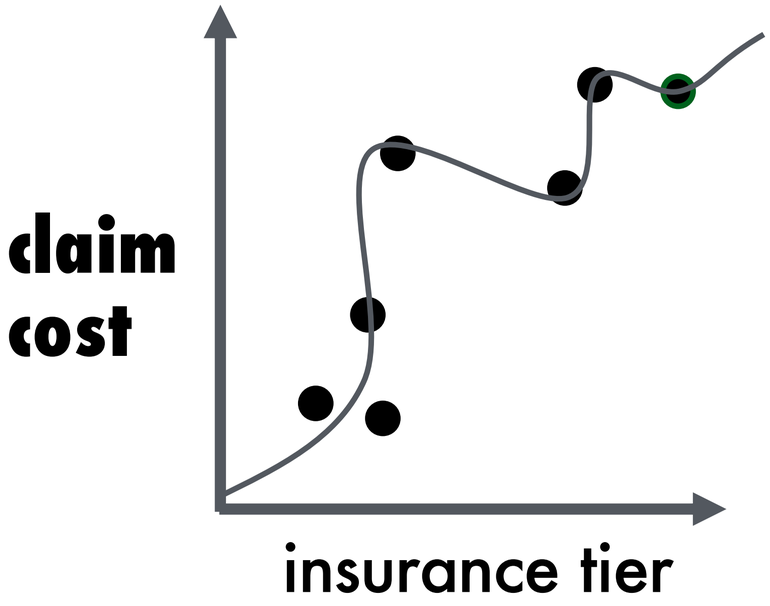
CLASSIFICATION

Will a borrower default on loan?

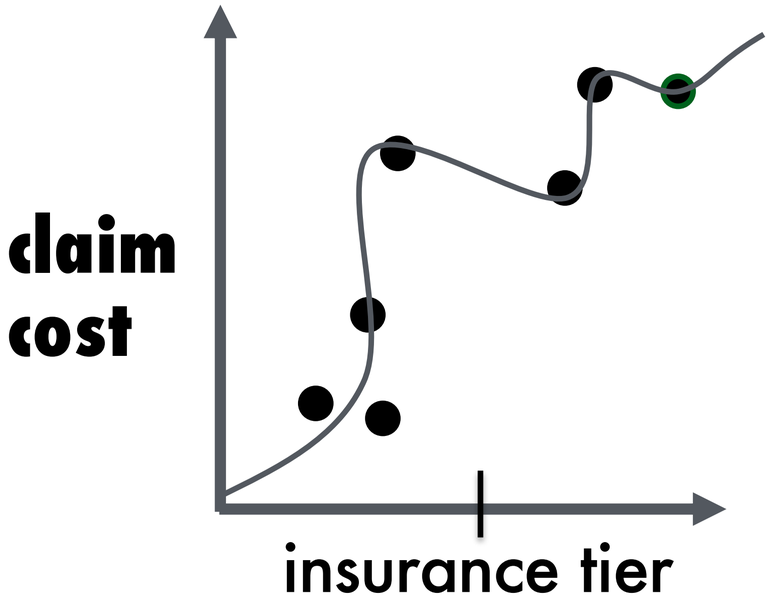
Supervised Learning



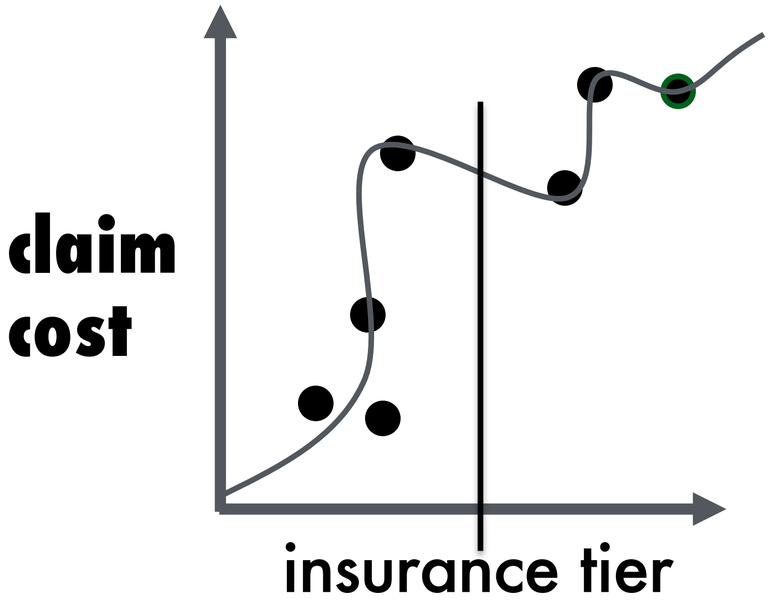
Supervised Learning



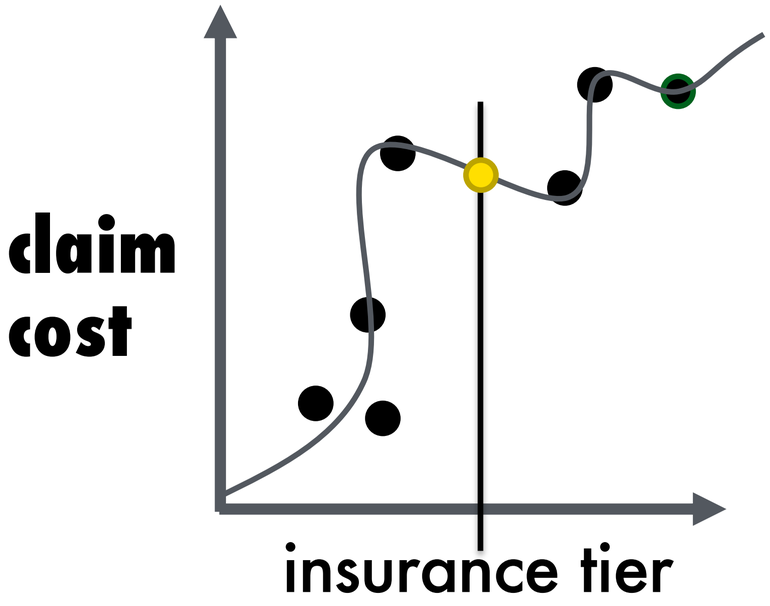
Supervised Learning



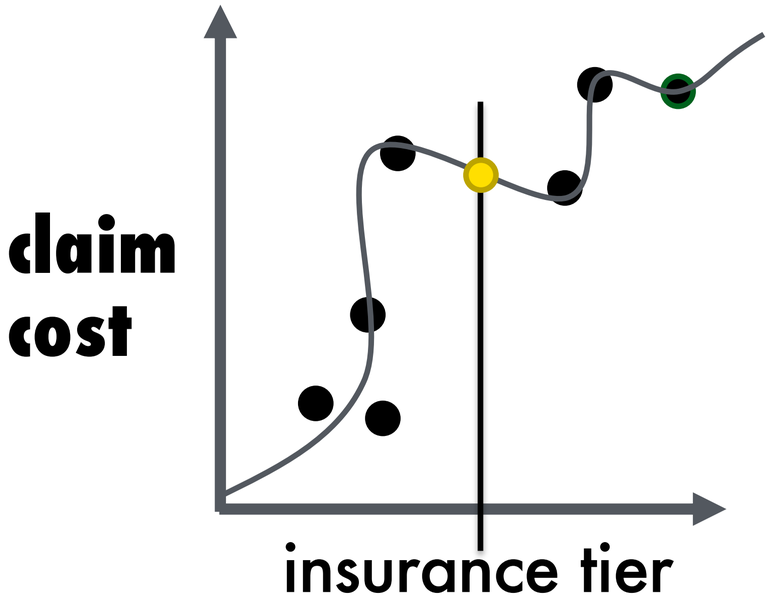
Supervised Learning

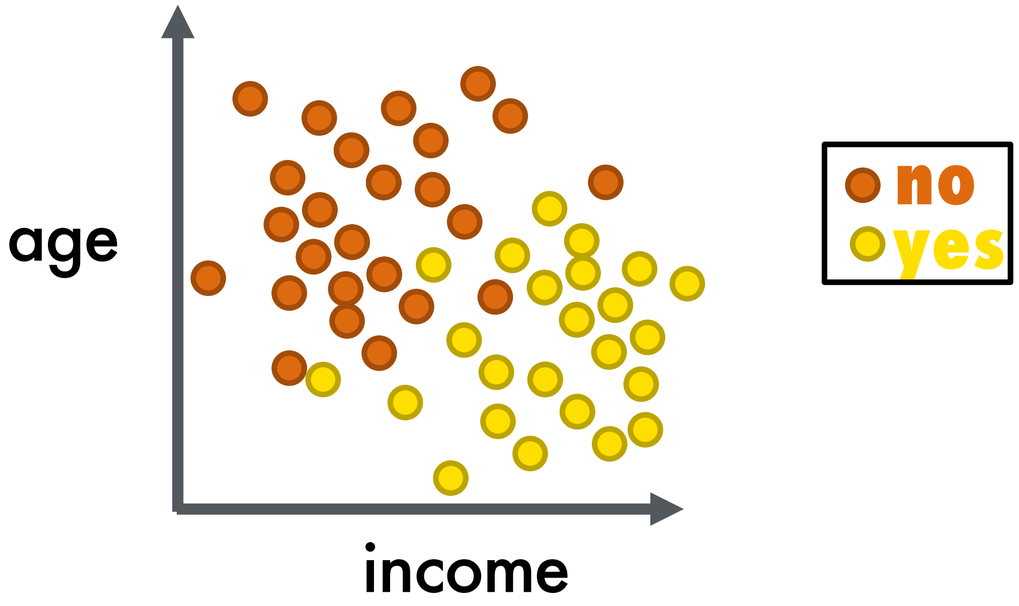


Supervised Learning

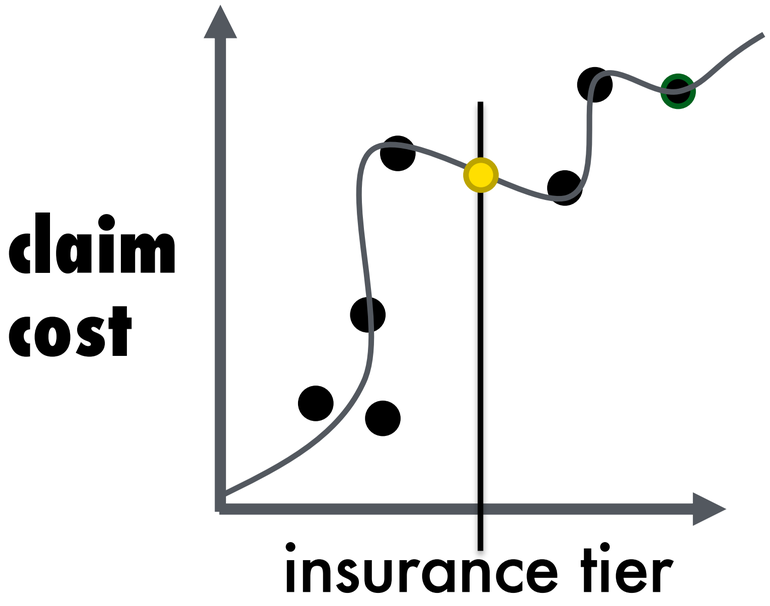


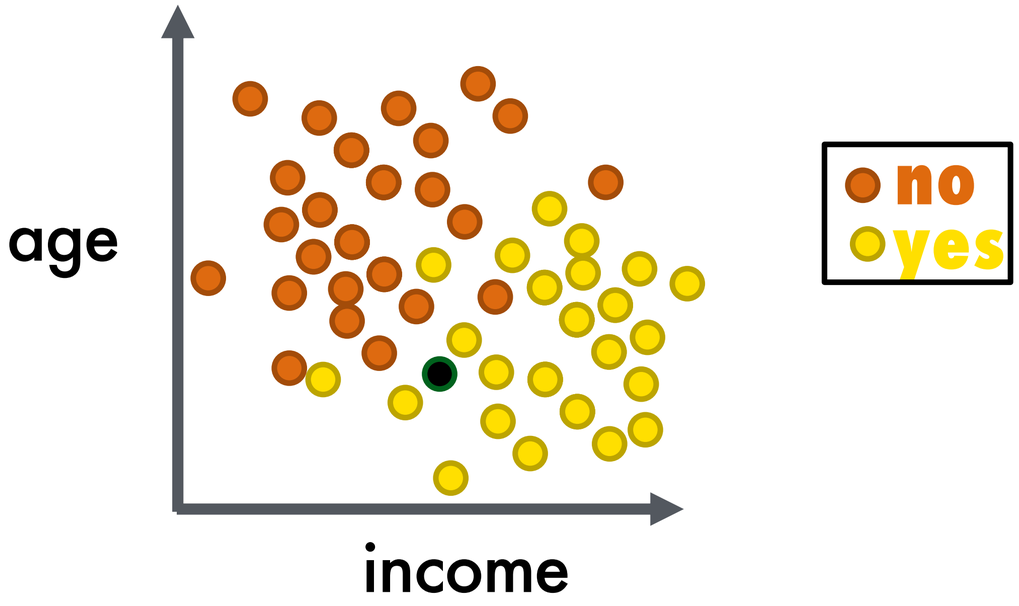
Supervised Learning



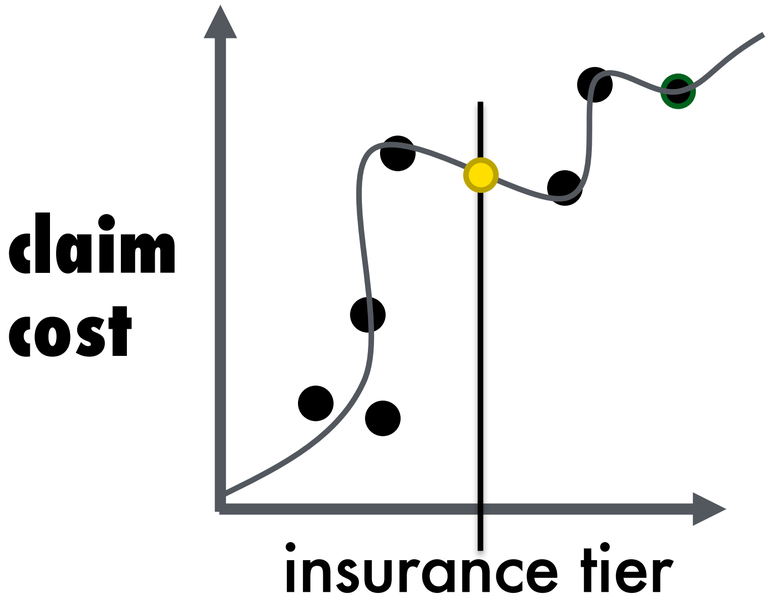


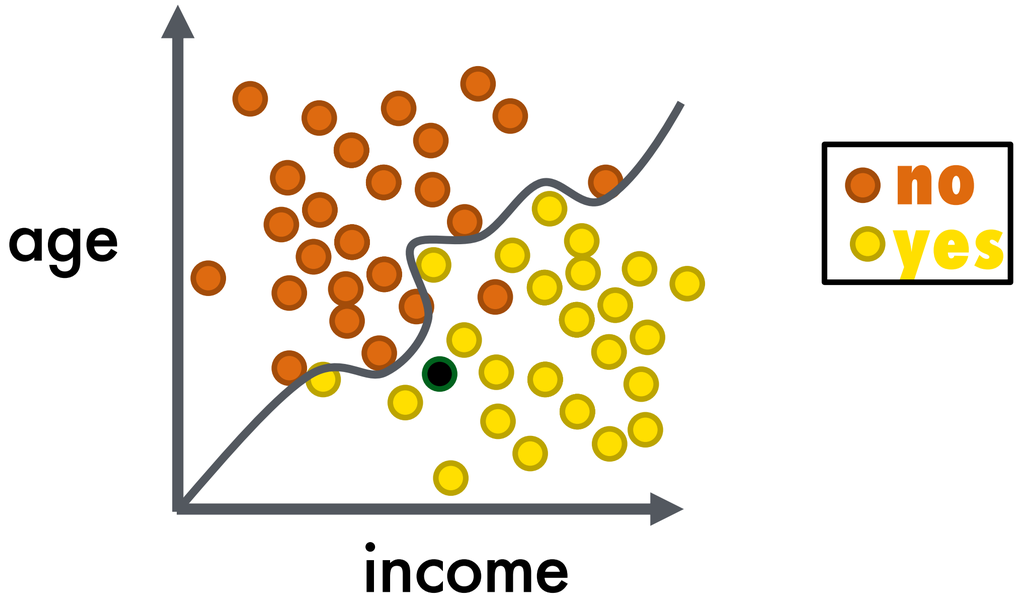
Supervised Learning



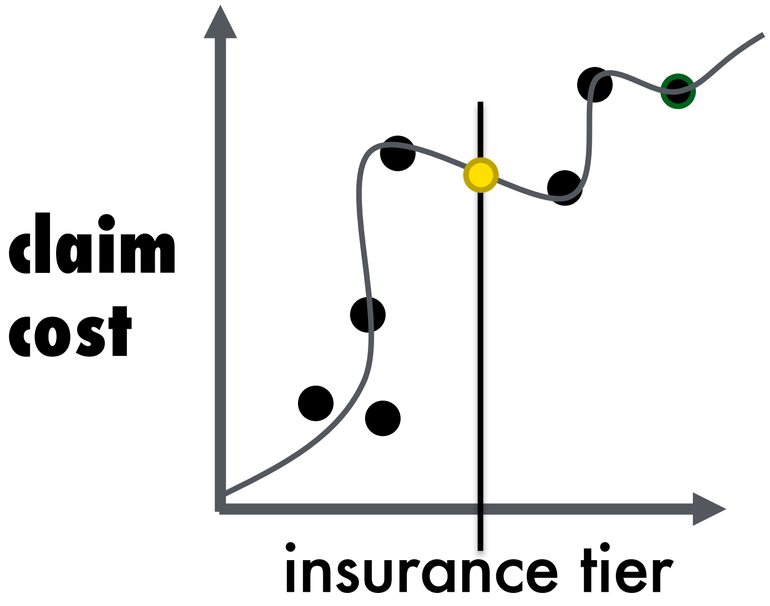


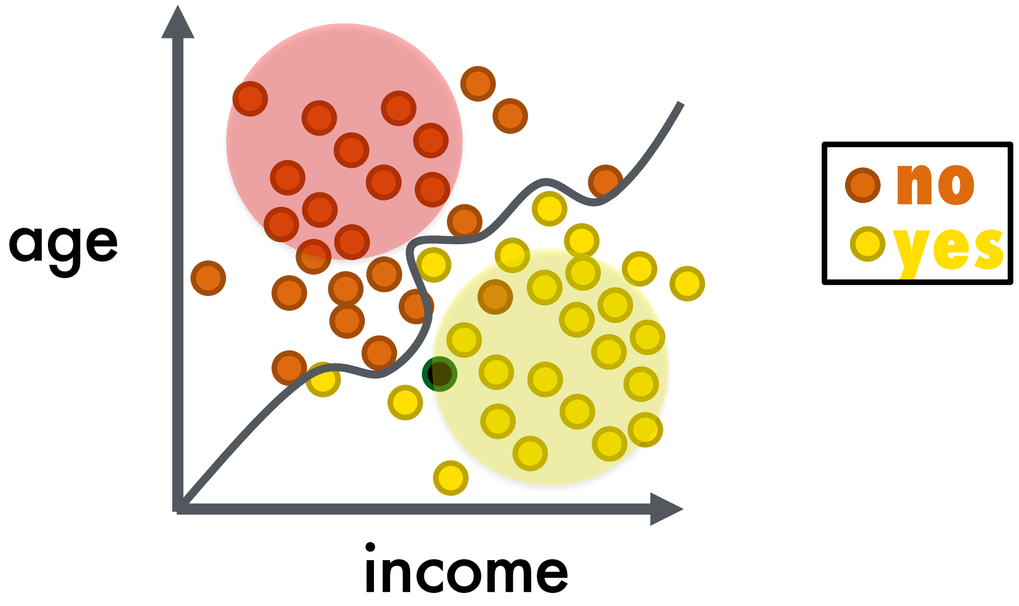
Supervised Learning





Supervised Learning





Current Algorithm Overview

Supervised Algorithms

Statistical Analysis

- Linear Models (GLM)
- Naïve Bayes

Ensembles

- Random Forest
- Distributed Trees
- Gradient Boosting Machine
- Stacking / Super Learner

Deep Neural Networks

- MLP
- Autoencoder
  - Anomaly Detection
  - Deep Features

Clustering

- K-Means (Auto-K)

Dimension Reduction

- Principal Component Analysis
- Generalized Low Rank Models

Word Embedding

- Word2Vec

Time Series

- iSAX

Machine Learning Tuning

- Hyperparameter Search
- Early Stopping

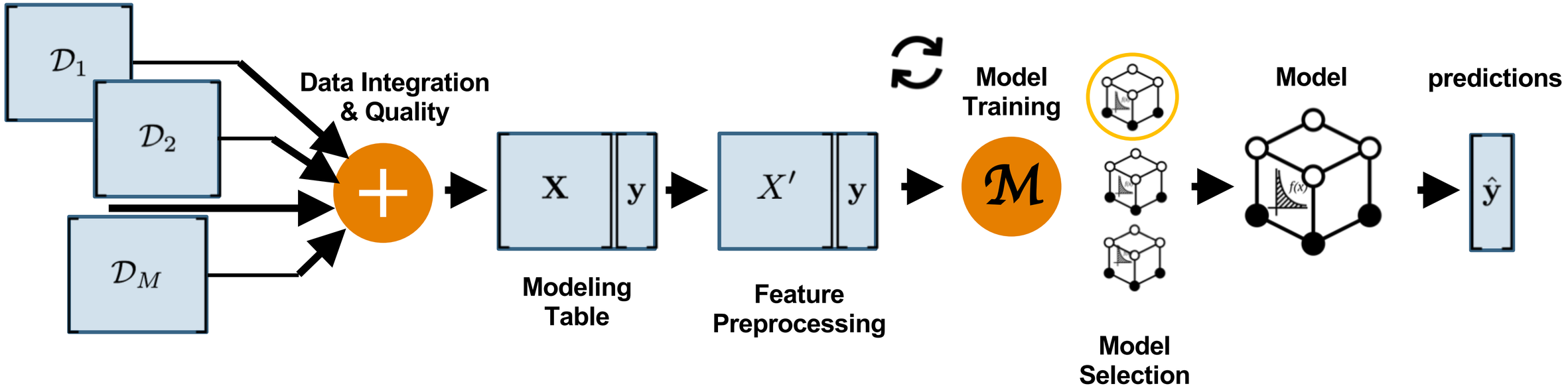
Intro to H2O's AutoML

Scalable Automatic Machine Learning

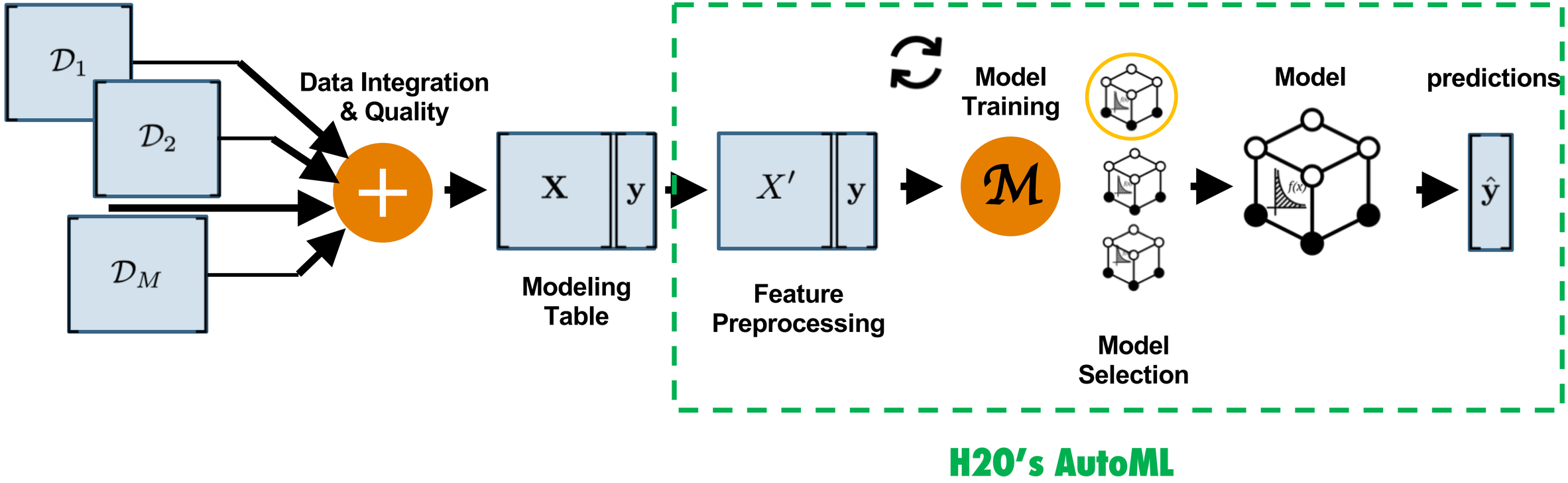
AutoML Agenda

- The Machine Learning Pipeline
- What is Automatic Machine Learning?
- How to Use H2O's AutoML

Updated Machine Learning Pipeline



Updated Machine Learning Pipeline

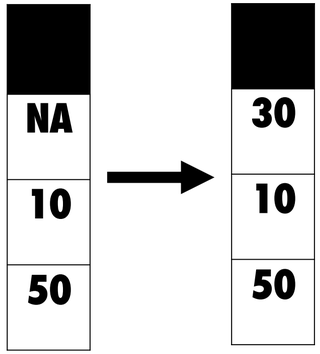


Automatic ML Overview

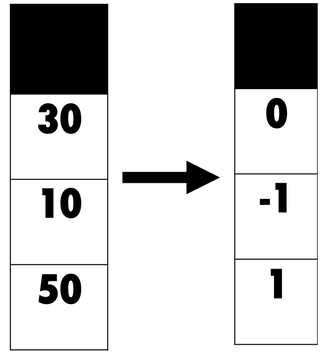
- **Feature Preprocessing**
- Hyperparameter Search
- Ensembles

Feature Preprocessing

Missing Values

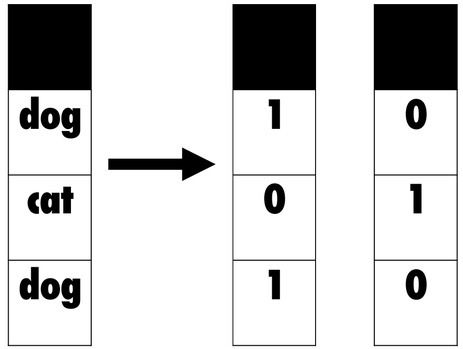


Standardization

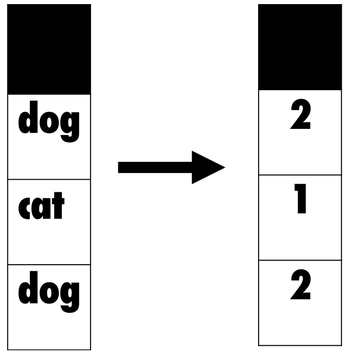


Feature Preprocessing

Handling Categorical Columns



One-hot Encoding



Label Encoder

Feature Preprocessing ≠ Feature Engineering

Domain Expertise

Timestamp
jan/02/2017
march/03/2019
dec/11/2018

→

Month	Day	Year
01	02	2017
03	03	2019
12	11	2018

Date Expansion

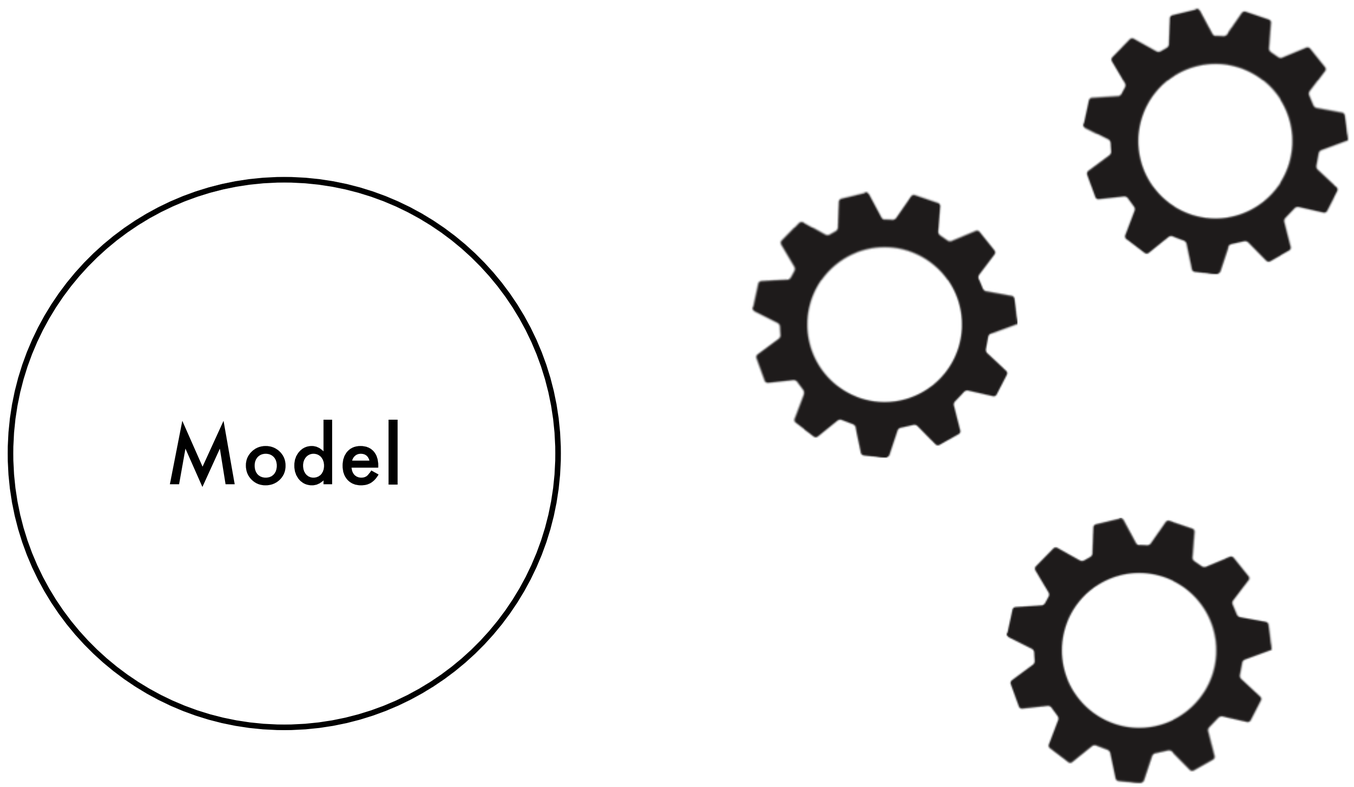
Automatic ML Overview

- ~~Feature Preprocessing~~
- **Hyperparameter Search**
- Ensembles

What are Hyperparameters?

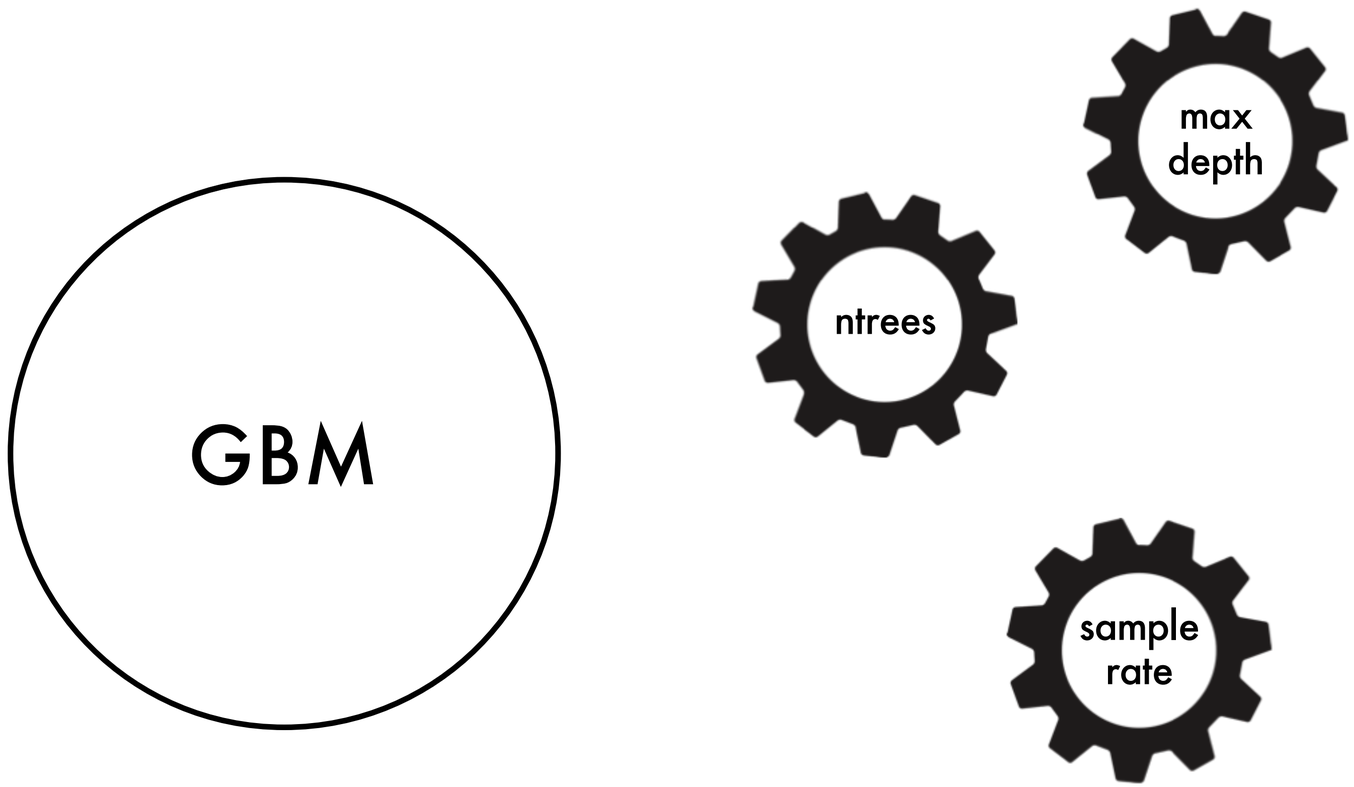
(a quick review)

What are Model Hyperparameters?



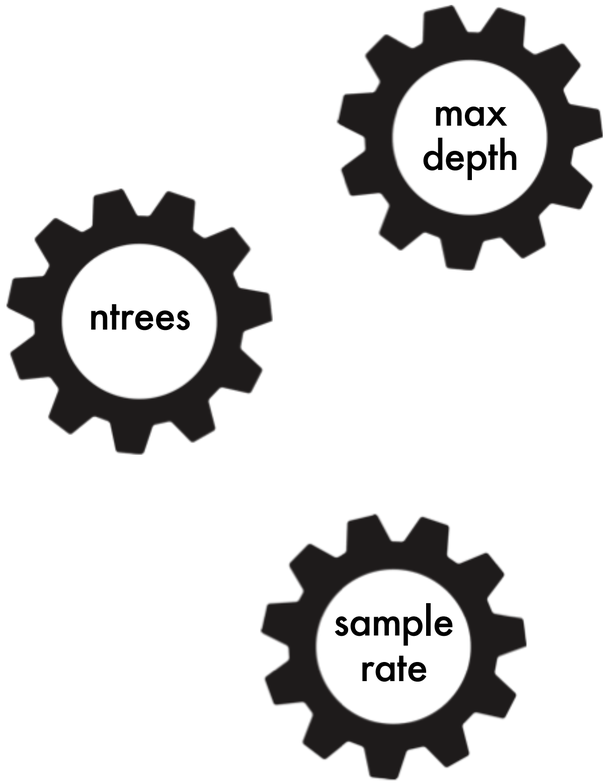
Every model comes with knobs

What are Model Hyperparameters?



Every model comes with knobs

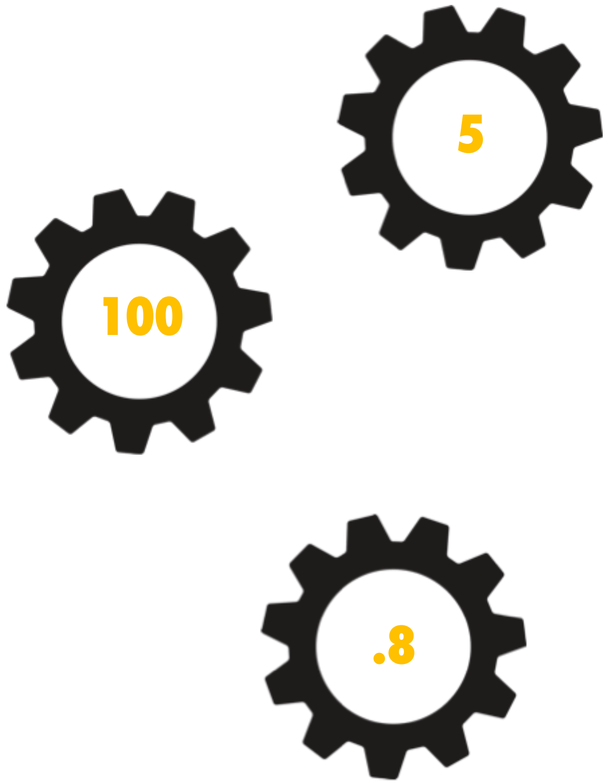
What are Hyperparameter?



Unlike regular model parameters, hyperparameters are not learned from the data

The user has to set the hyperparameters values

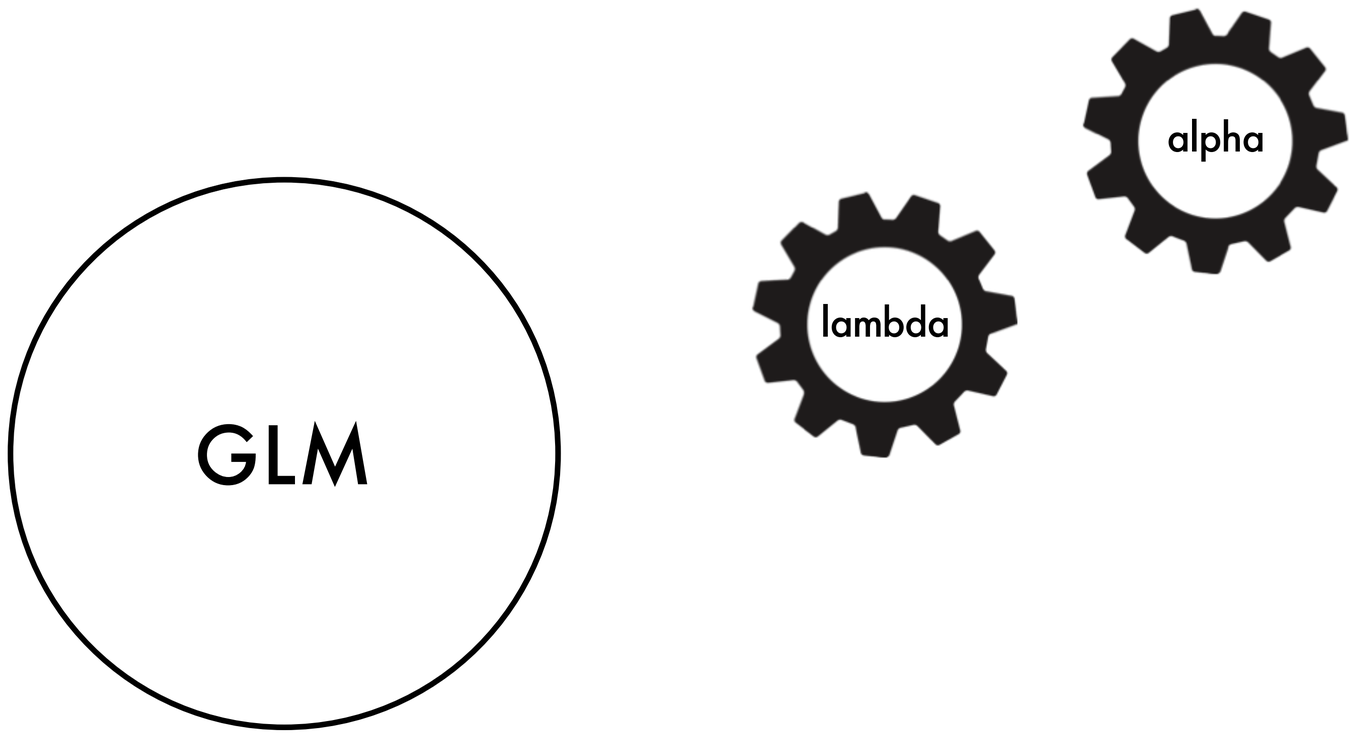
What are Hyperparameter?



Unlike regular model parameters, hyperparameters are not learned from the data

The user has to set the hyperparameters values

What are GLM's Hyperparameters?



GLM Model Parameters

$$\max_{\beta,\beta_0} \sum_{i=1}^{N} \log f\left(y_i;\beta,\beta_0\right) - \lambda\left(\alpha\|\beta\|_1 + \frac{1}{2}(1-\alpha)\|\beta\|_2^2\right)$$

GLM Model Parameters

$$\max_{\beta,\beta_0} \sum_{i=1}^{N} \log f\left(y_i; \beta, \beta_0\right) - \lambda \left(\alpha \|\beta\|_1 + \frac{1}{2}(1-\alpha)\|\beta\|_2^2 \right)$$

GLM Hyperparameter

$$\max_{\beta,\beta_0} \sum_{i=1}^{N} \log f\left(y_i;\beta,\beta_0\right) - \lambda\left(\alpha\|\beta\|_1 + \frac{1}{2}(1-\alpha)\|\beta\|_2^2\right)$$

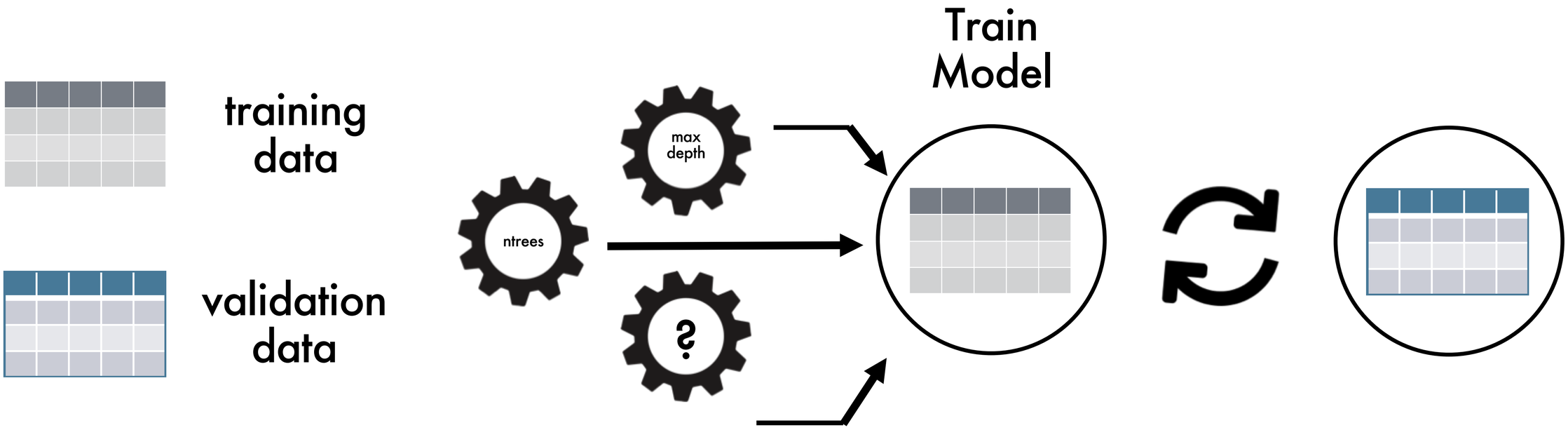
Model Building: Hyperparameter Search

3 Main Ways:

- Manual Model Tuning
- Cartesian Grid Search
- Random Grid Search*

*What AutoML Uses

Manual Model Tuning



Split Dataset

Specify Hyperparameter Values

Validate Performance

Manual Model Tuning

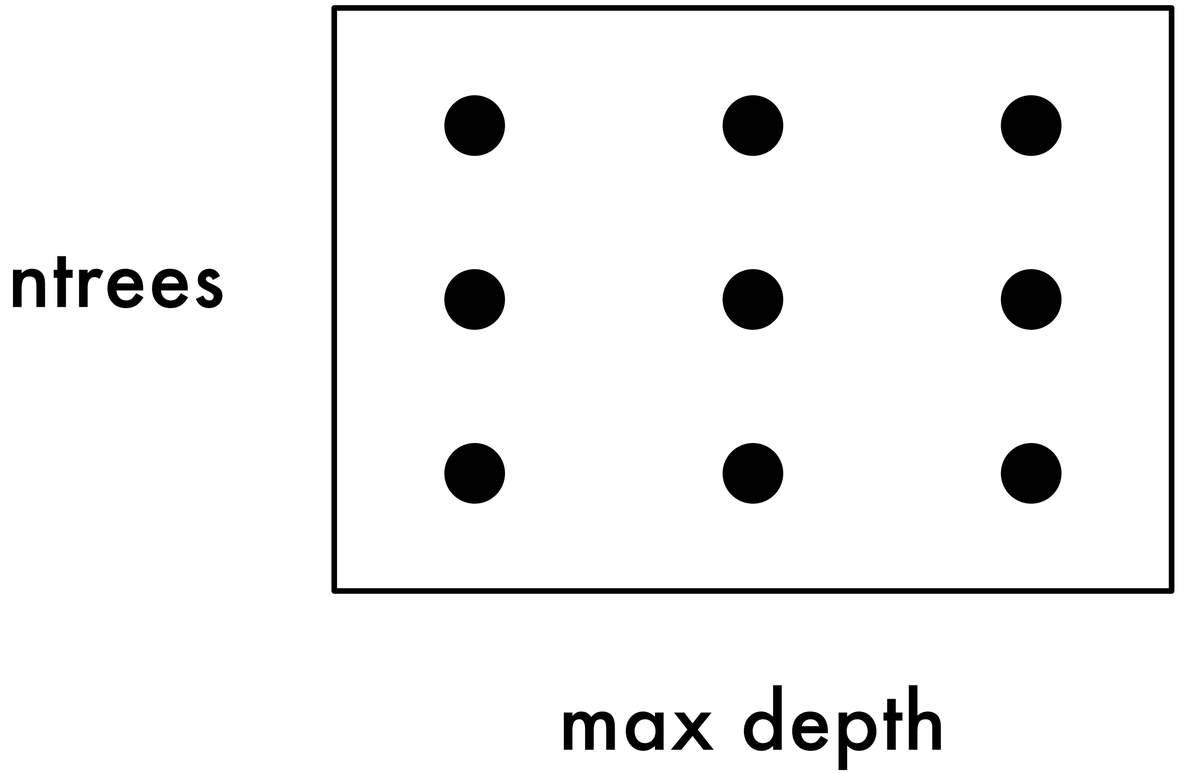
How did you do?

- Not so good: **Repeat** Process
- GOOD! : Stop

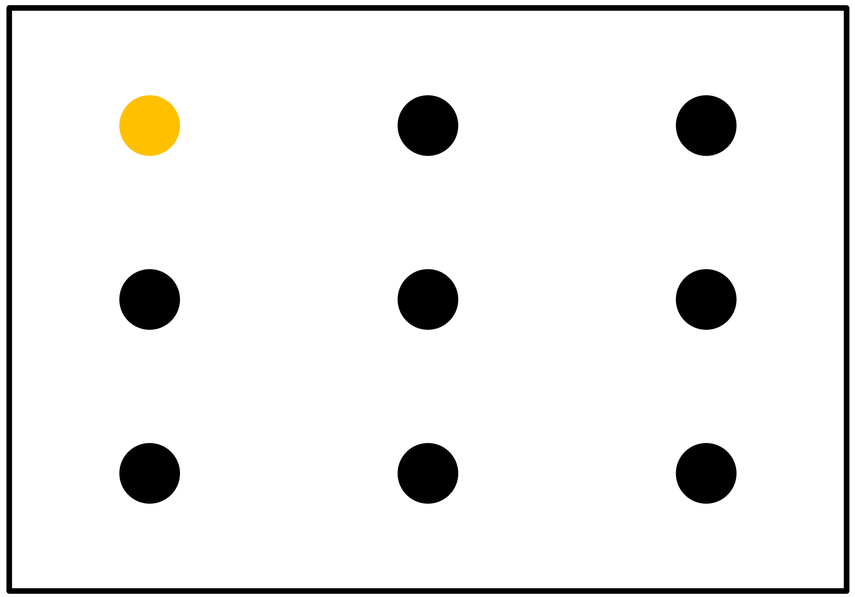
Cartesian Grid Search

AKA Exhaustive Search: Tries **Every Combination**

Cartesian Grid Search



Cartesian Grid Search

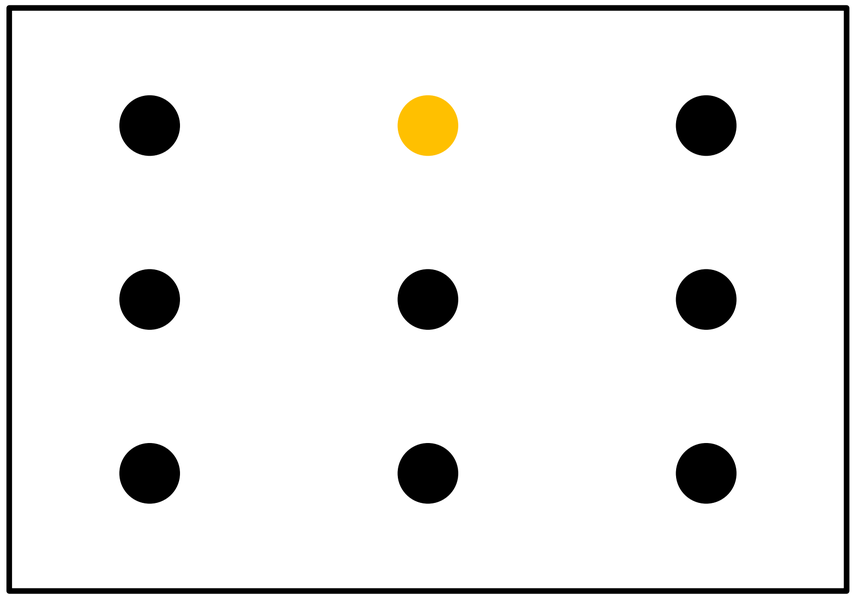


single trial

ntrees = 1000

max depth = 5

Cartesian Search

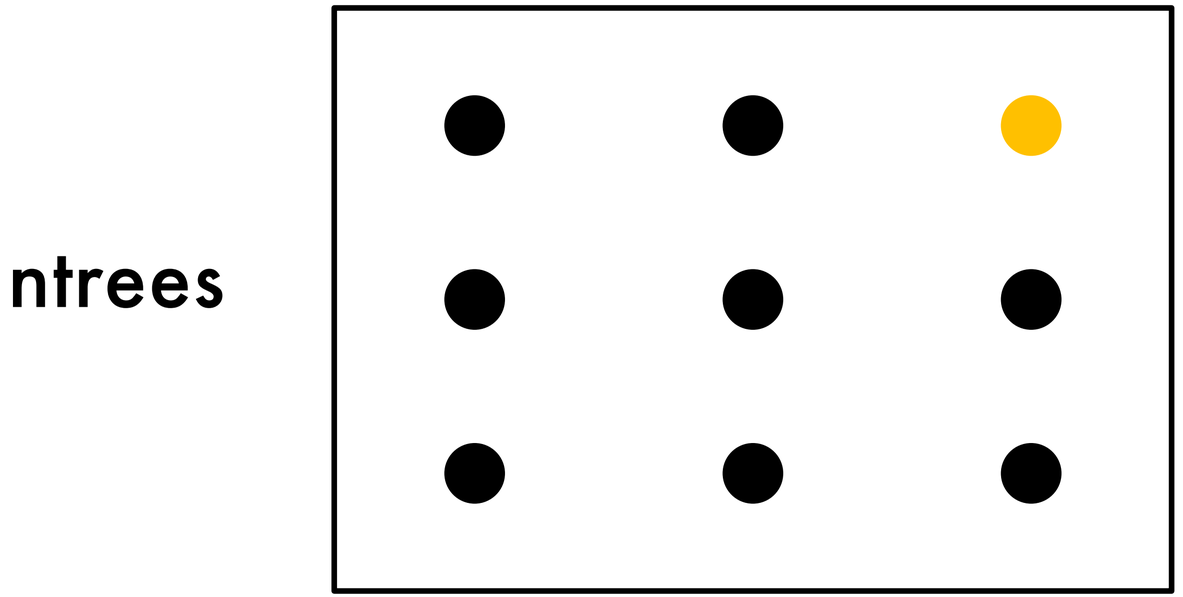


single trial

ntrees = 1000

max depth = 10

Cartesian Search

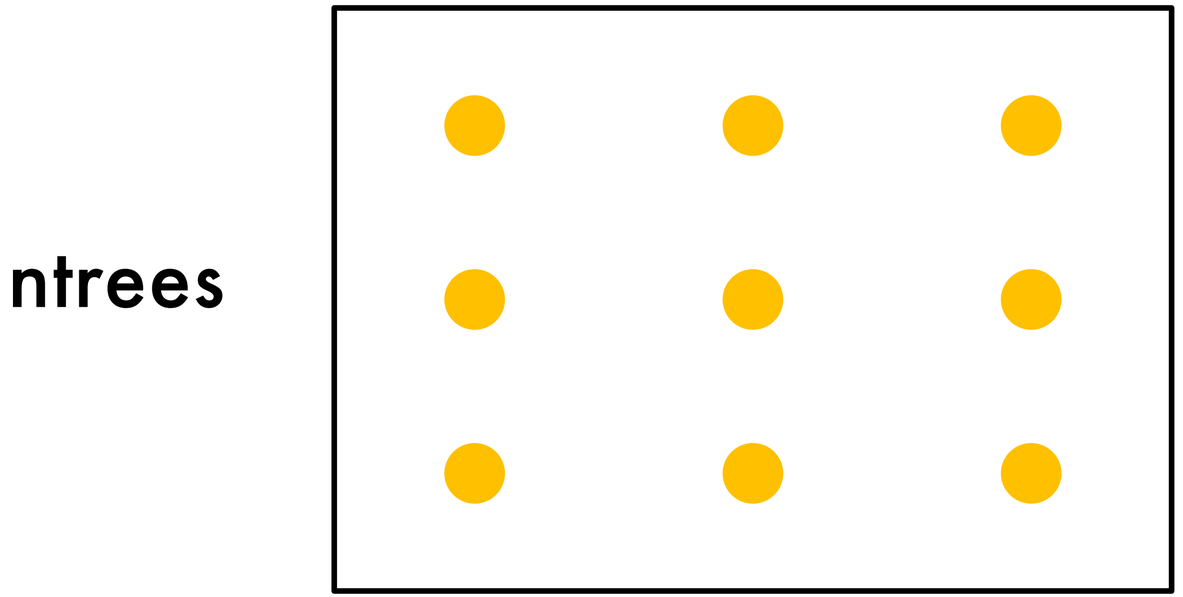


single trial

ntrees = 1000

max depth = 15

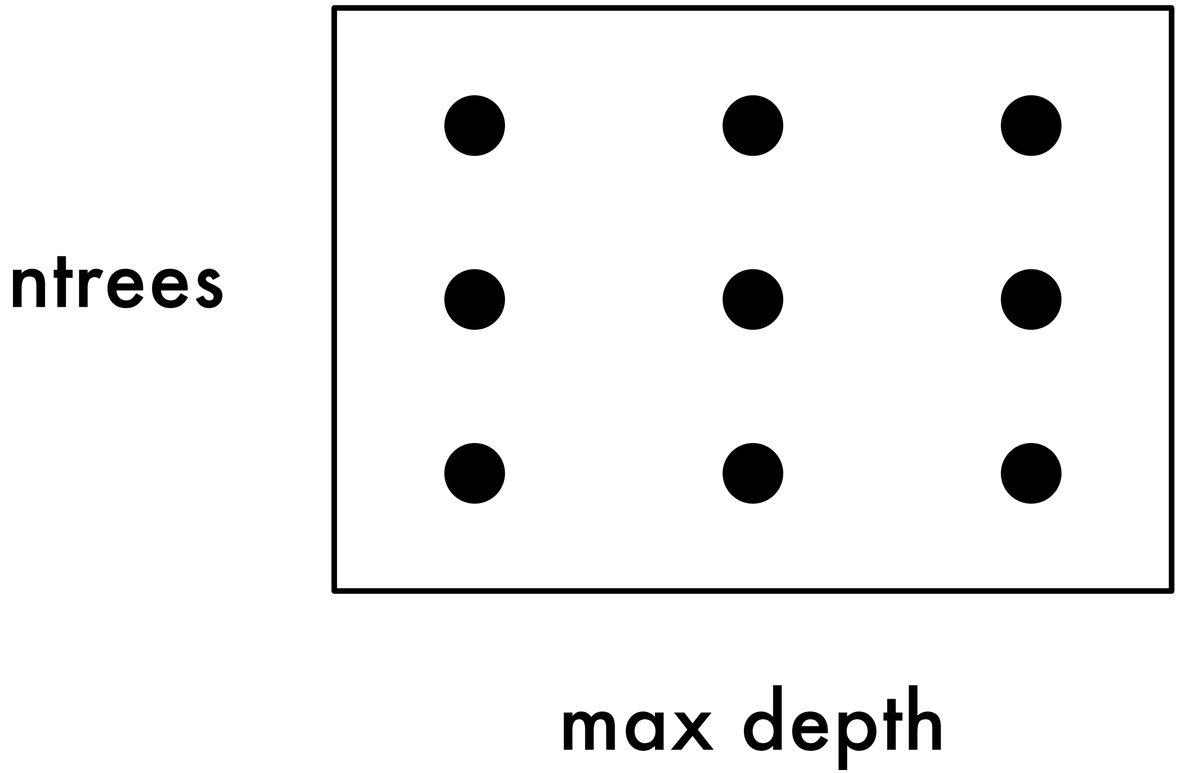
Cartesian Grid Search



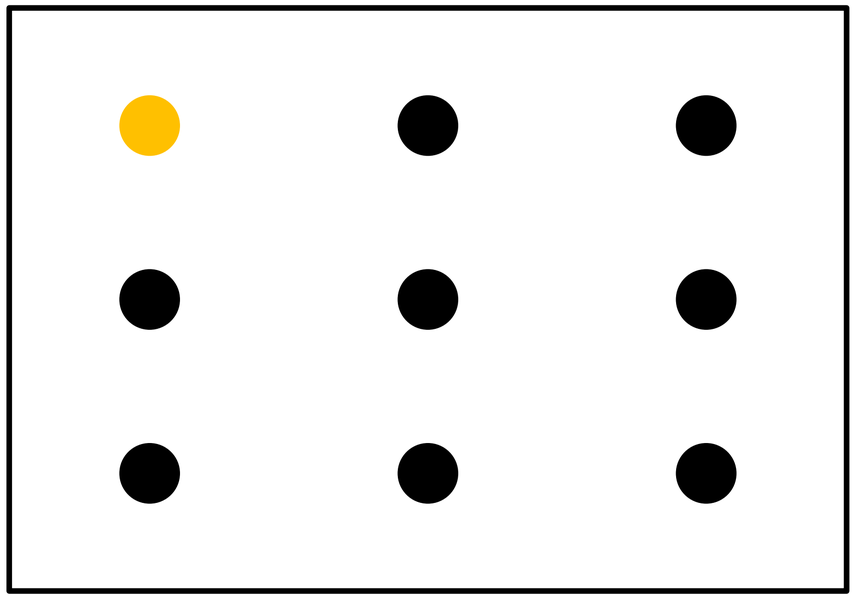
Cartesian Grid Search

Complete Exploration

Random Grid Search



Random Grid Search

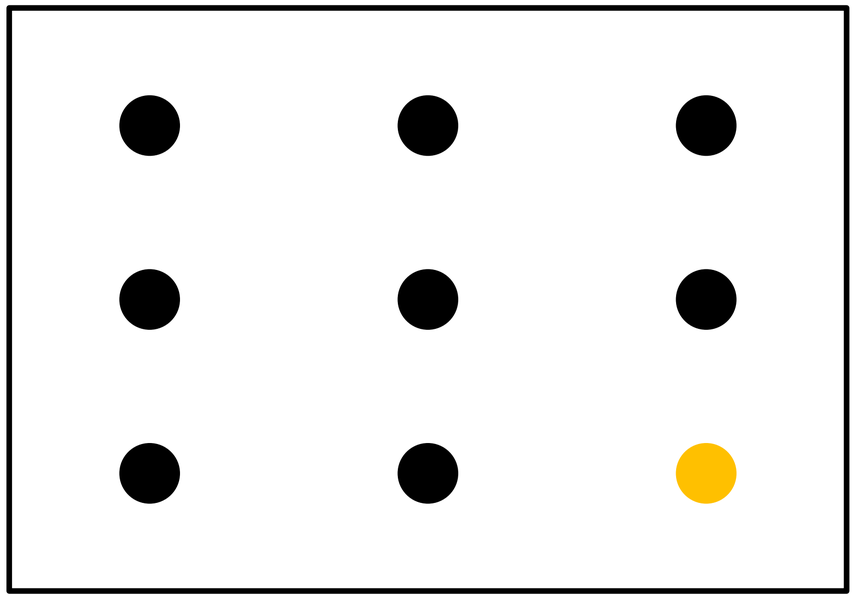


single trial

ntrees = 1000

max depth = 5

Random Grid Search

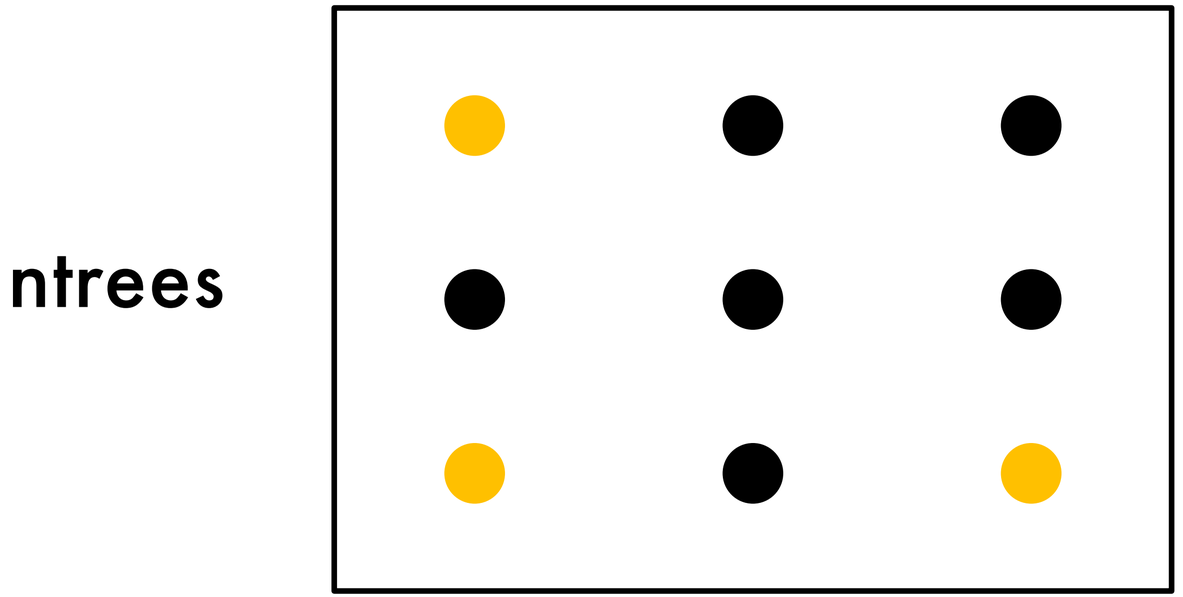


single trial

ntrees = 3000

max depth = 15

Random Grid Search



Random Grid Search

explores more in less time

Automatic ML Overview

- ~~Feature Preprocessing~~
- ~~Feature Engineering~~
- ~~Hyperparameter Search~~
- Ensembles

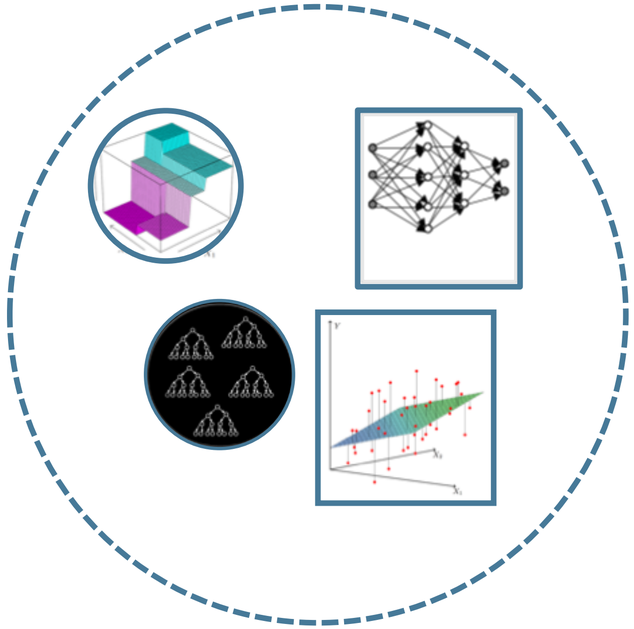
Ensembles

Vanilla Ensembles

Gradient Boosting Machine

Random Forest

Super Learning Ensembles



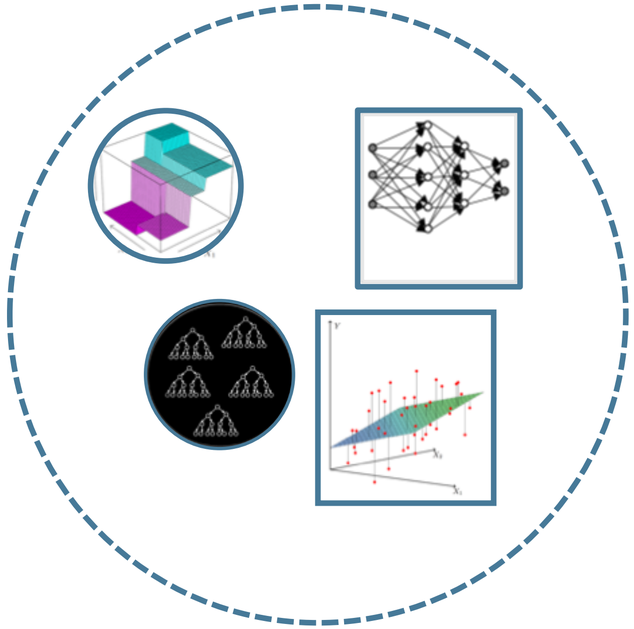
Ensembles

Vanilla Ensembles

Gradient Boosting Machine

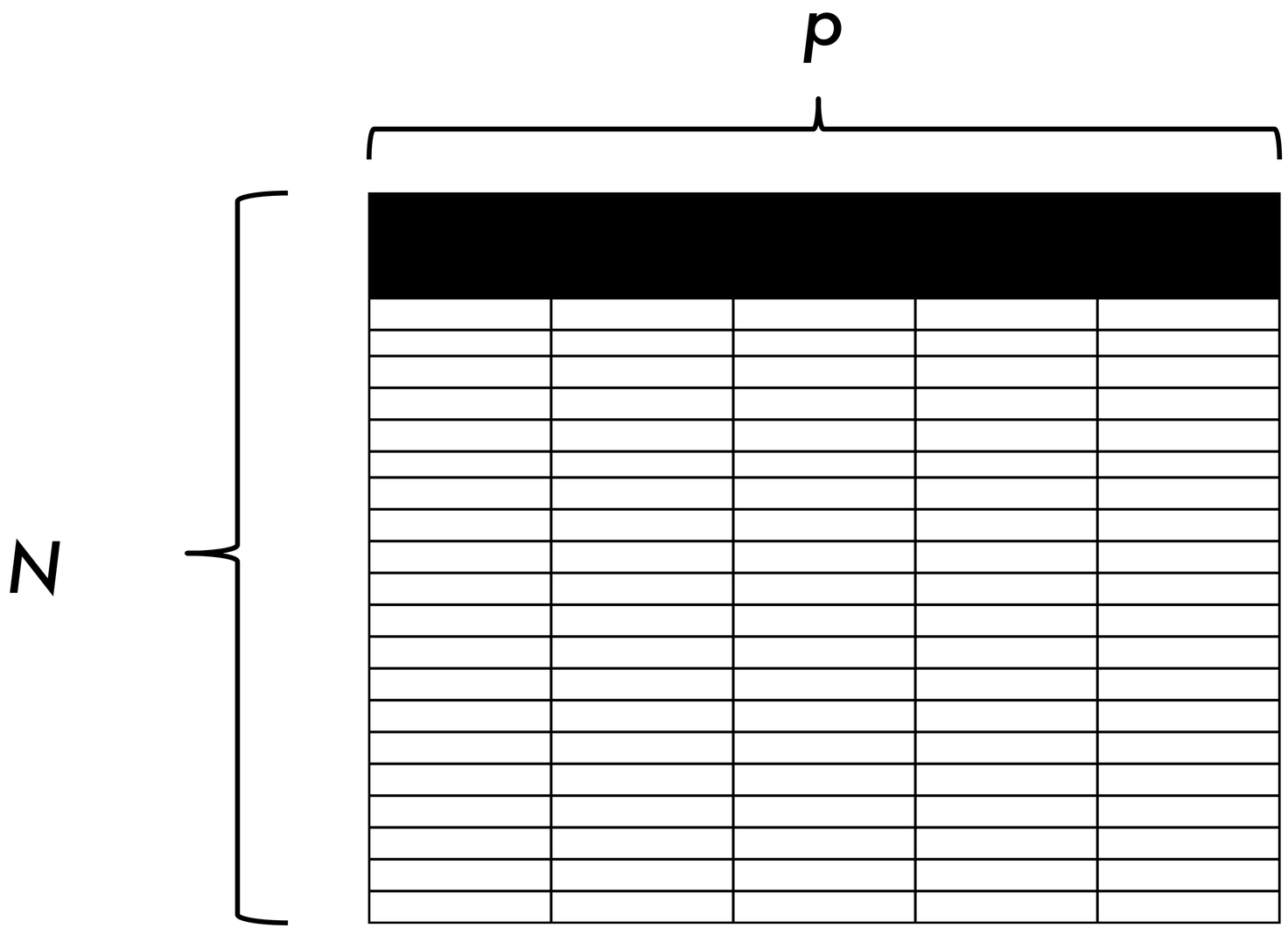
Random Forest

(AKA Stacking)



Stacked Ensemble Steps

1. **Specify original dataset for training**
2. Specify algorithms with which to train
3. Stack each algo's cross-validated predictions
4. Train final model on cross-validated results



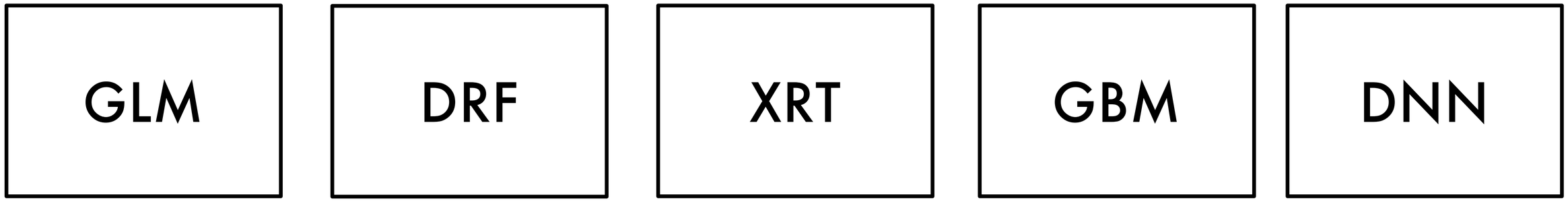
your original training data

"Level-zero" data

Stacked Ensemble Steps

1. Specify original dataset for training
2. **Specify algorithms with which to train**
3. Stack each algo's cross-validated predictions
4. Train final model on cross-validated results

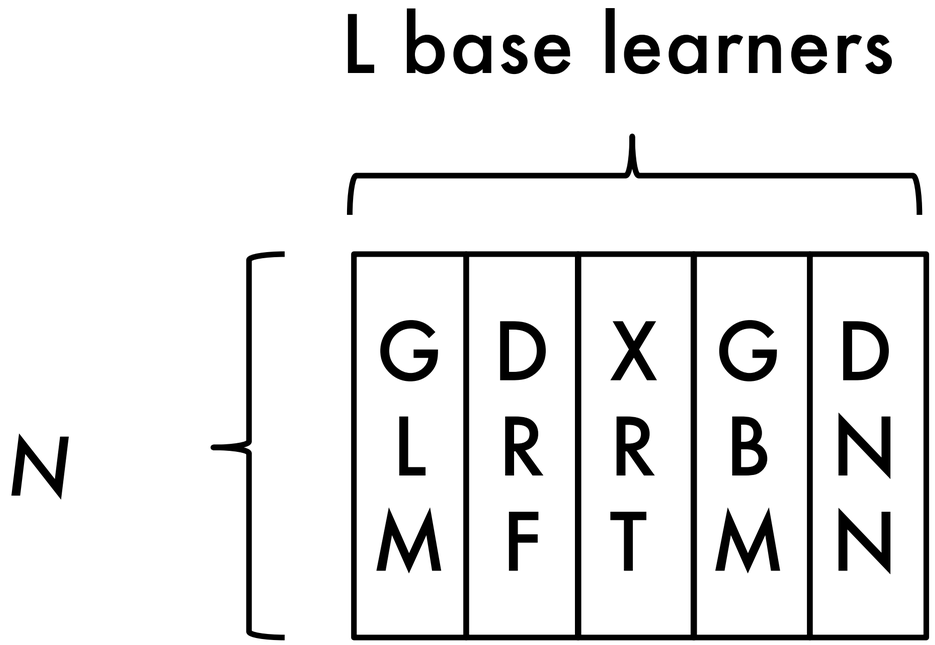
supervised algos used to train



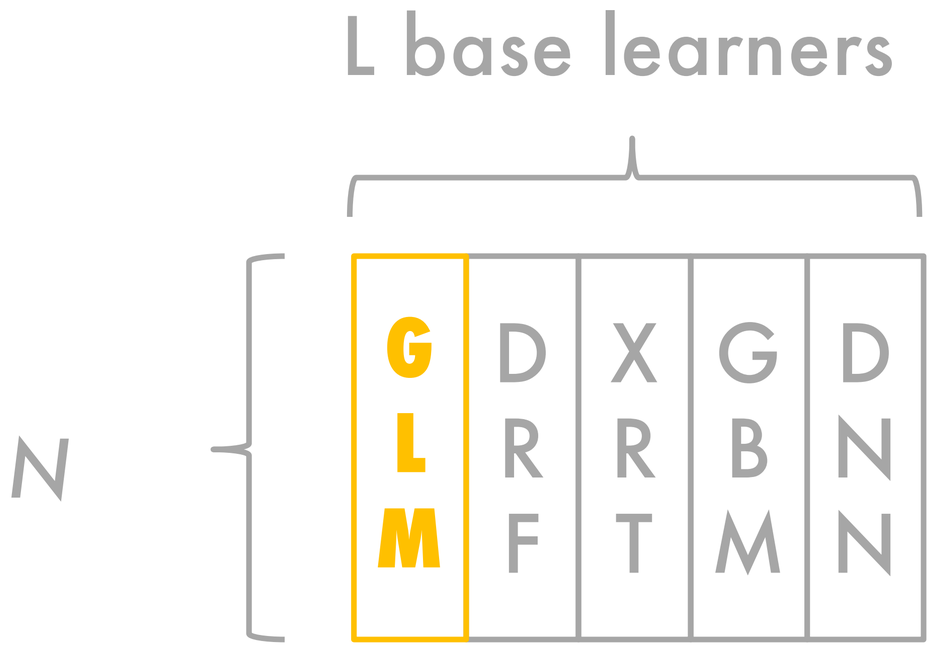
base learners

Stacked Ensemble Steps

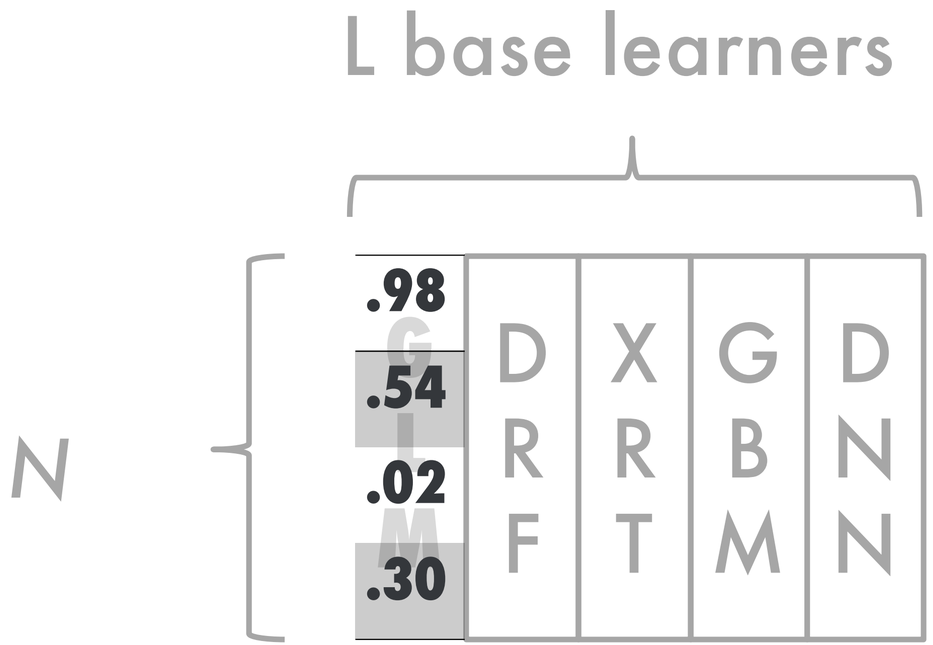
1. Specify original dataset for training
2. Specify algorithms with which to train
3. **Stack each algo's cross-validated predictions**
4. Train final model on cross-validated results



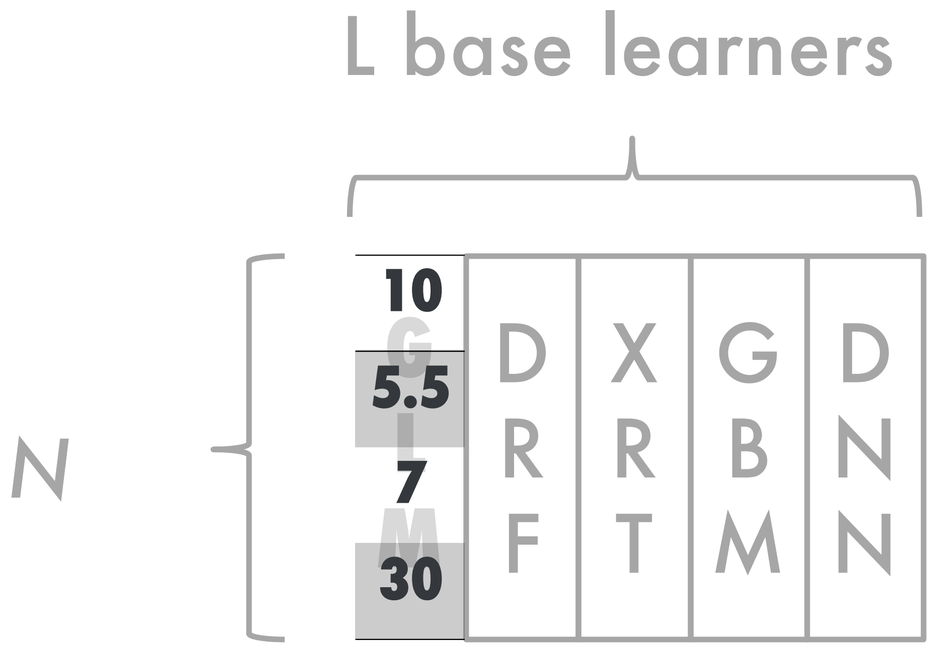
stacking your base learners



meaning:
each column is an algorithm's
cross-validated predictions



meaning:
each column is an algorithm's
cross-validated predictions



meaning:
each column is an algorithm's
cross-validated predictions

How do we get the cross-validated predictions?

How do we get the cross-validated predictions?

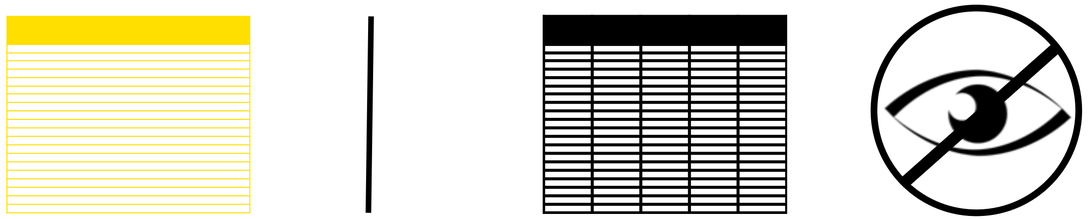
```
keep_cross_validation_predictions =  True
```

what is cross-validation?

quick review

Without K-Fold Cross Validation:

A)

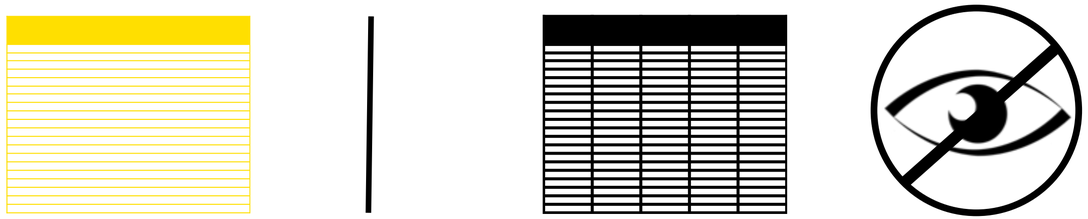


Train

Test

Without K-Fold Cross Validation:

A)

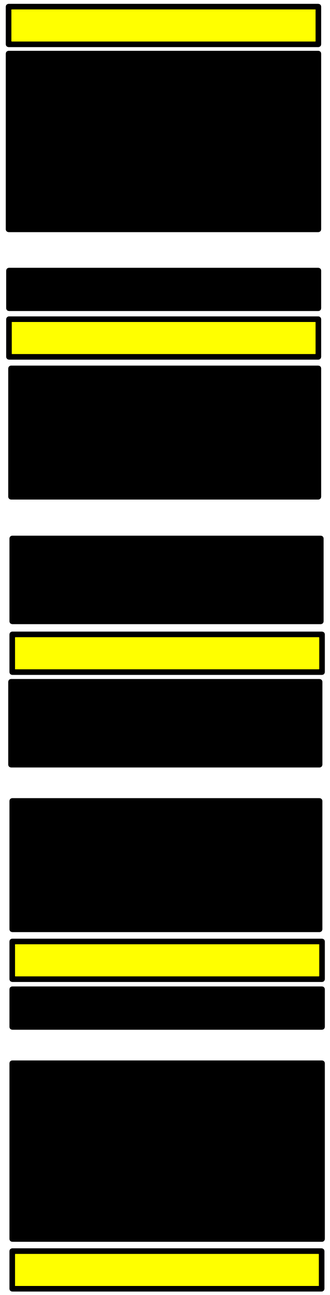


Train | Test

B)

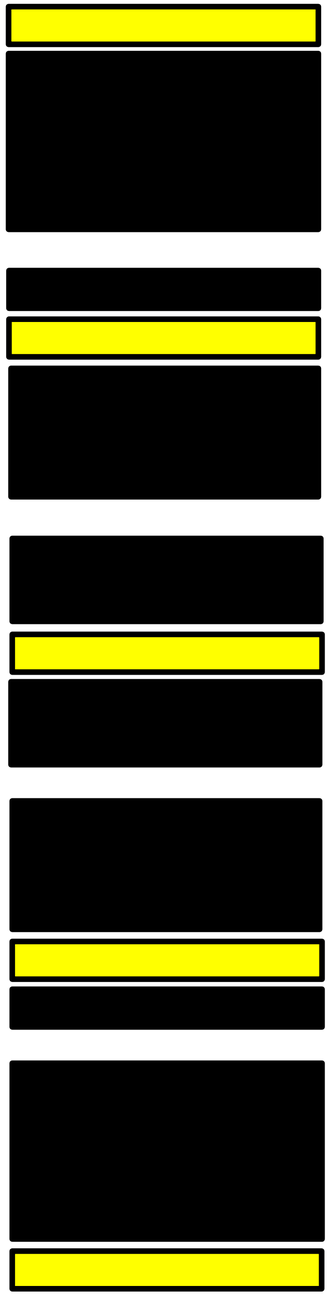
Train | Validate | Test

Model Selection

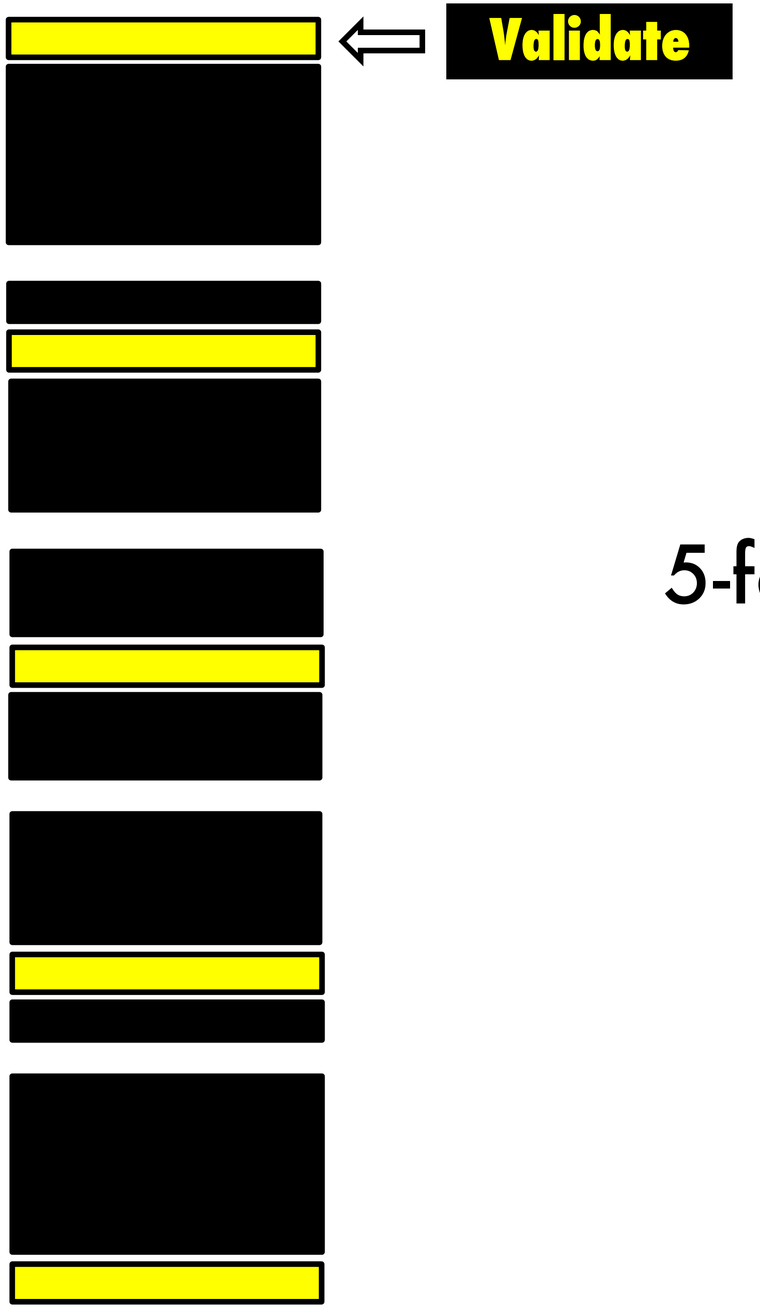


With K-Fold Cross Validation

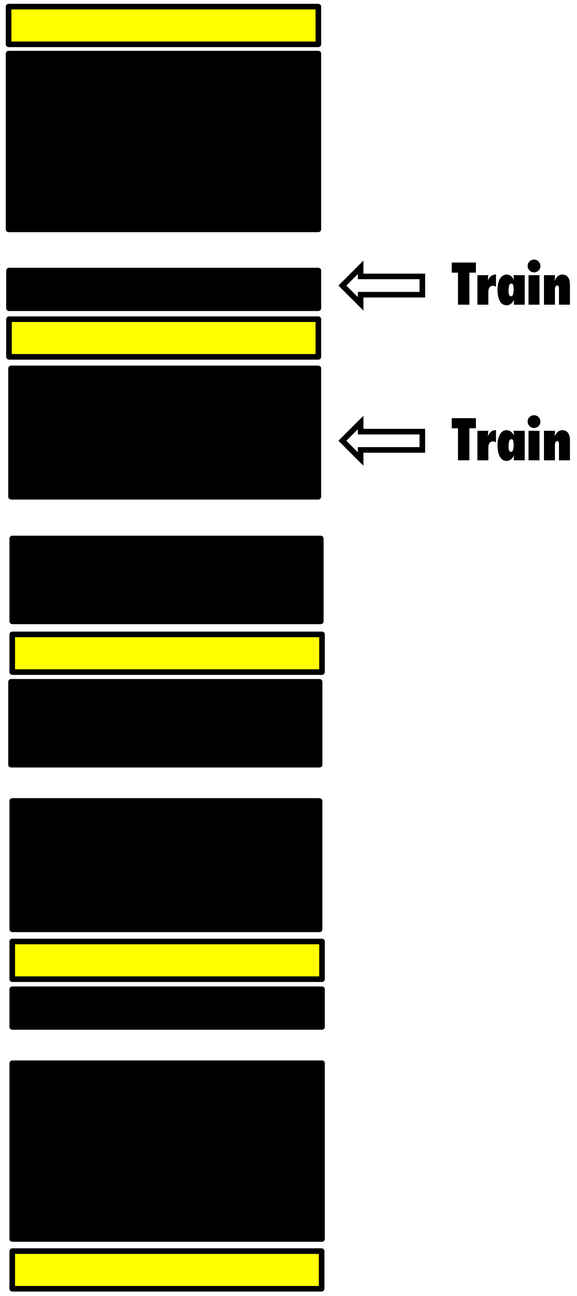
5-fold cross validation
for each algo



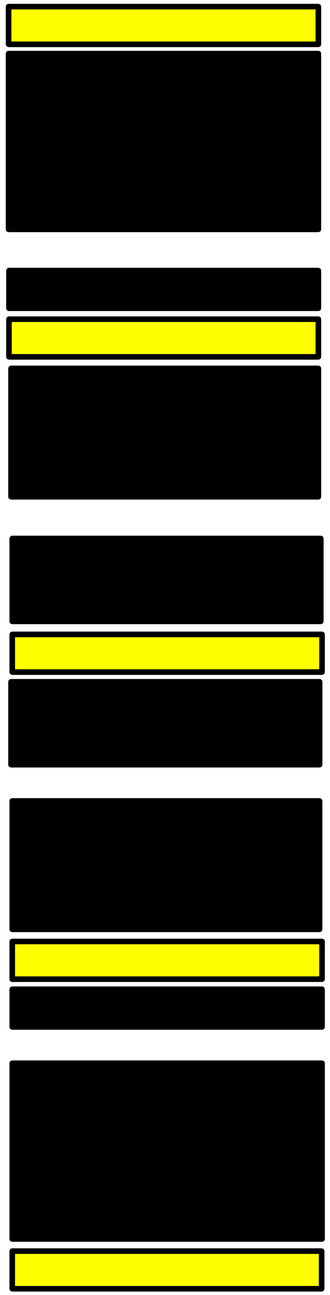
5-fold cross validation for each algo



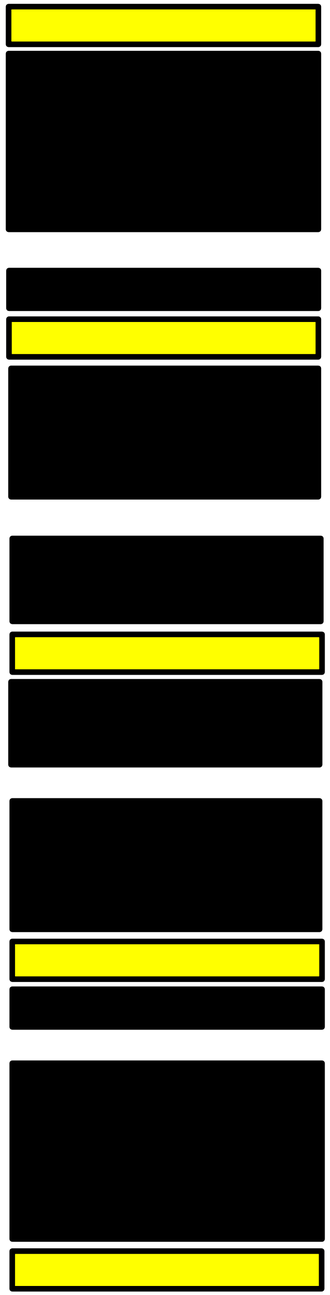
5-fold cross validation
for each algo



5-fold cross validation
for each algo



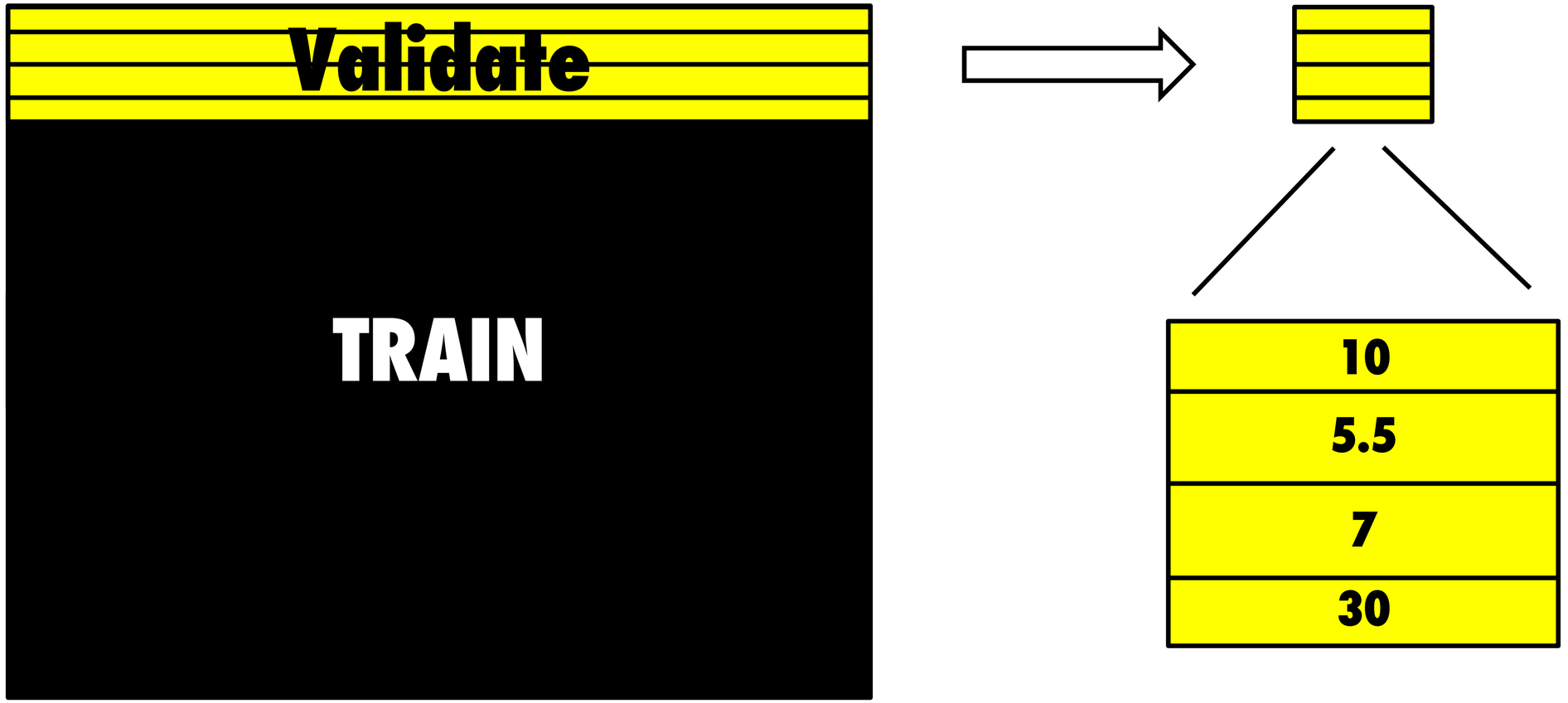
5-fold cross validation
for each algo

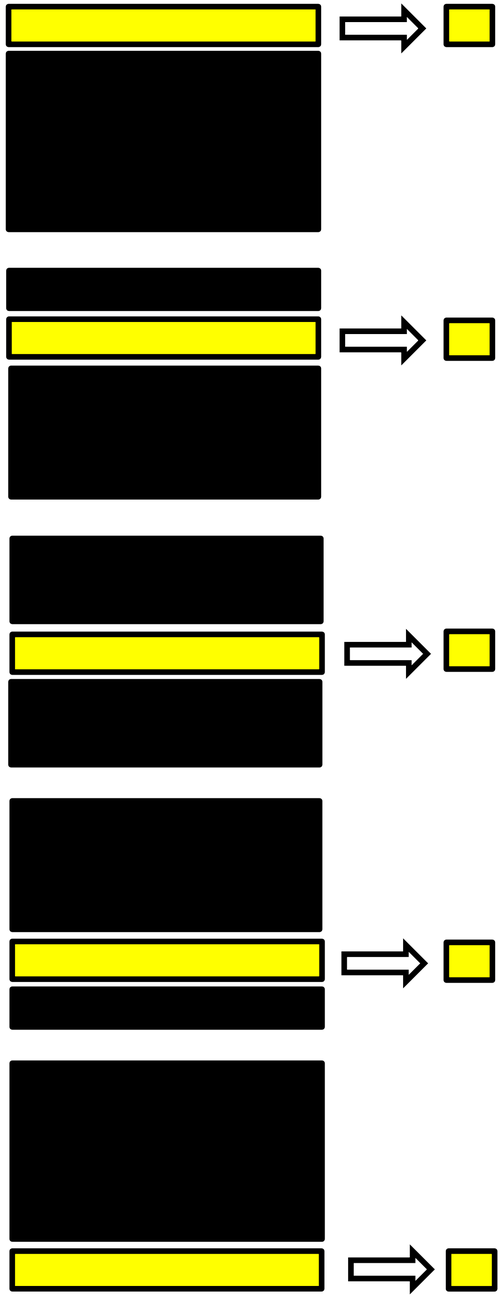


Continue Process

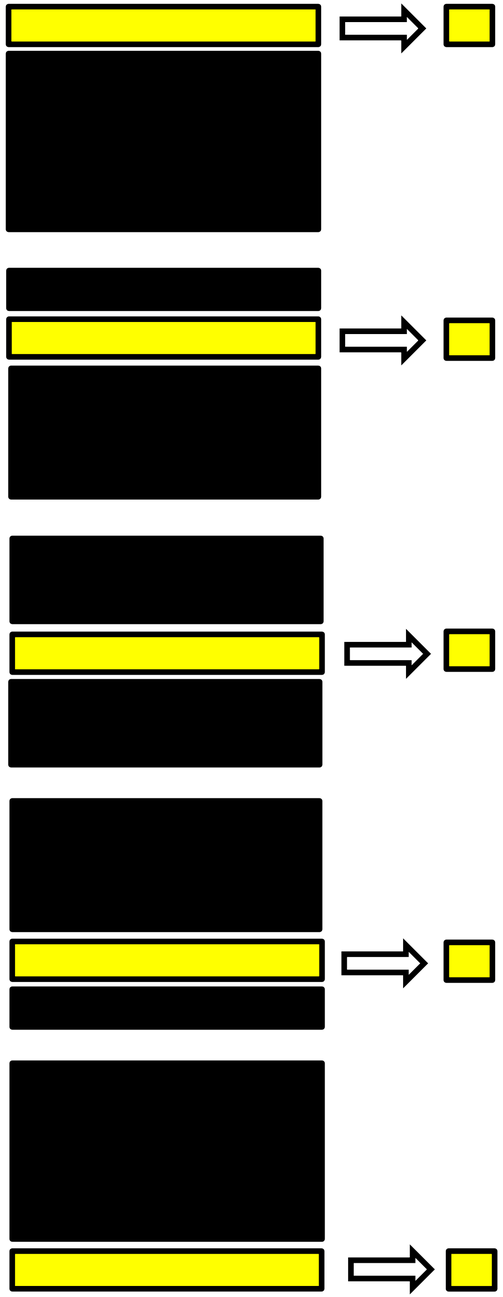
5-fold cross validation for each algo

Cross-Validated Prediction Values

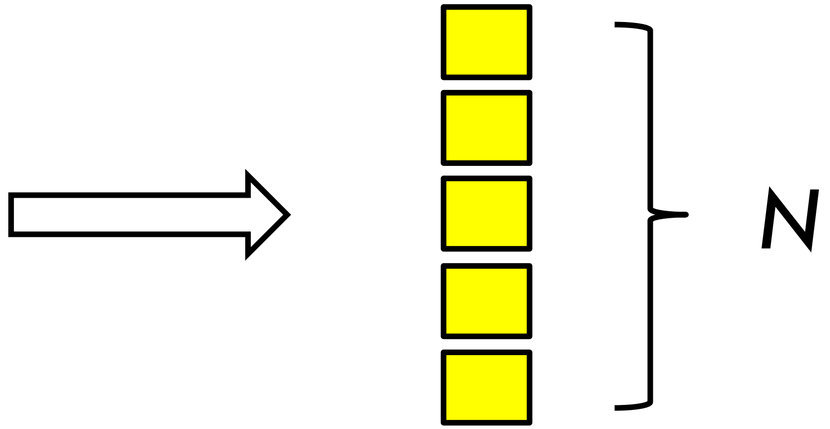




collect the
cross-validated
predictions



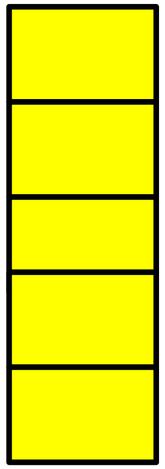
collect the cross-validated predictions



results for each algorithm

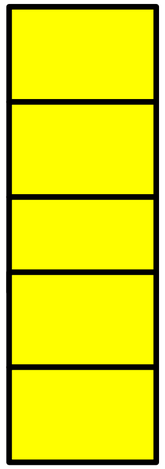
results for each algorithm

XRT

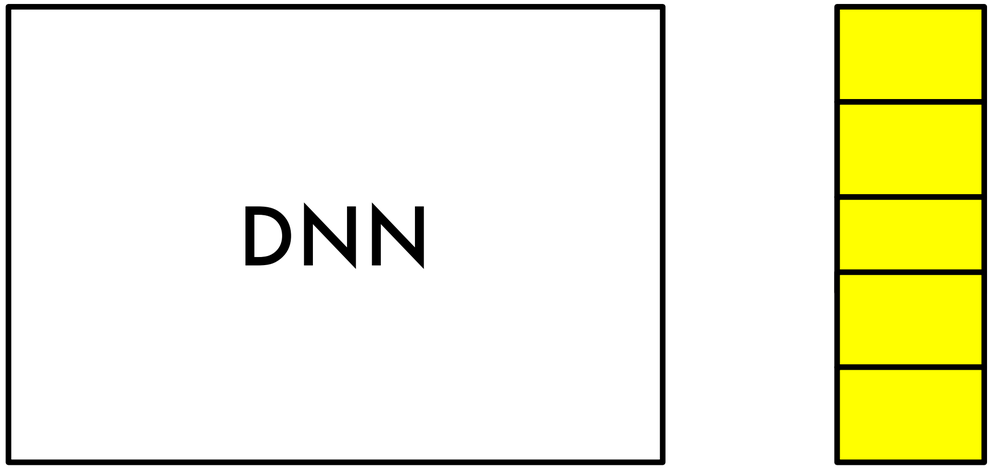


results for each algorithm

GBM



results for each algorithm



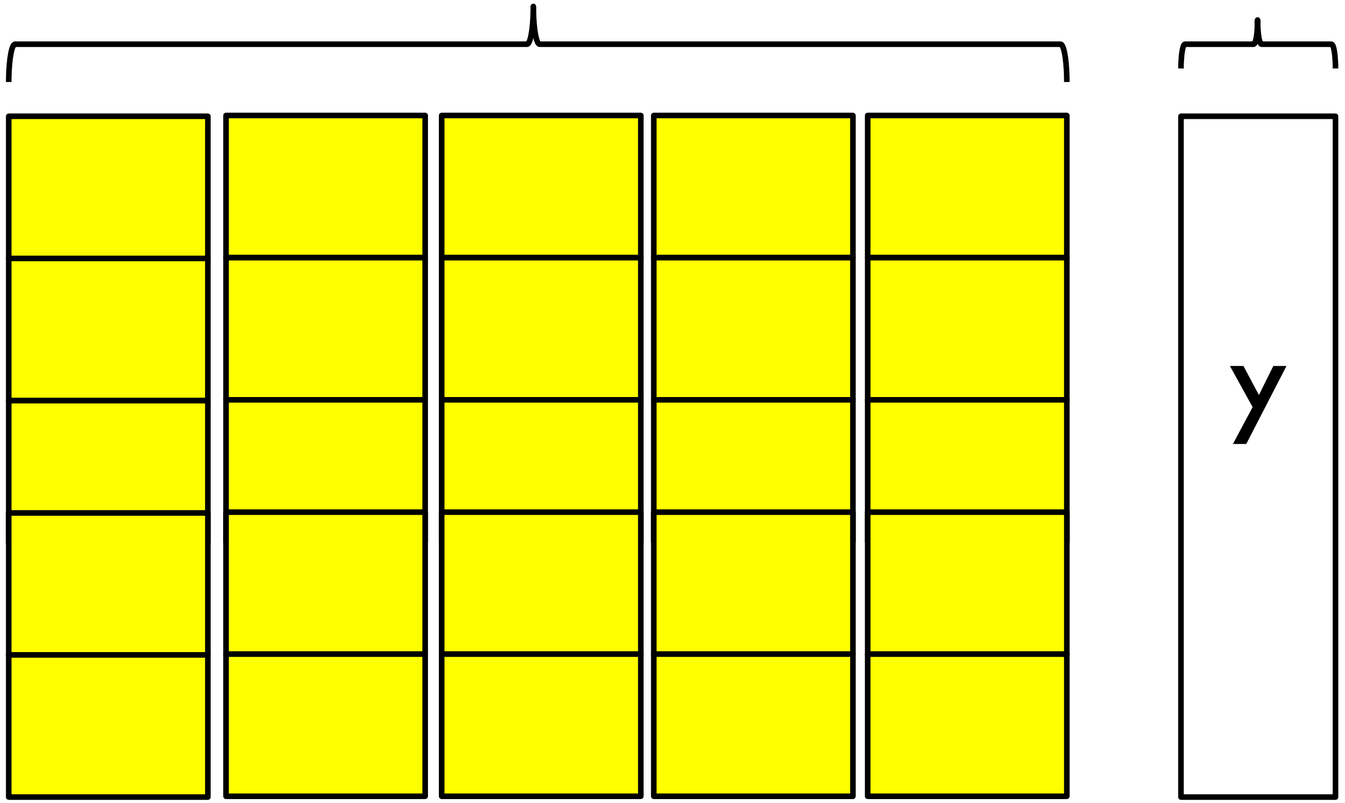
results for each algorithm

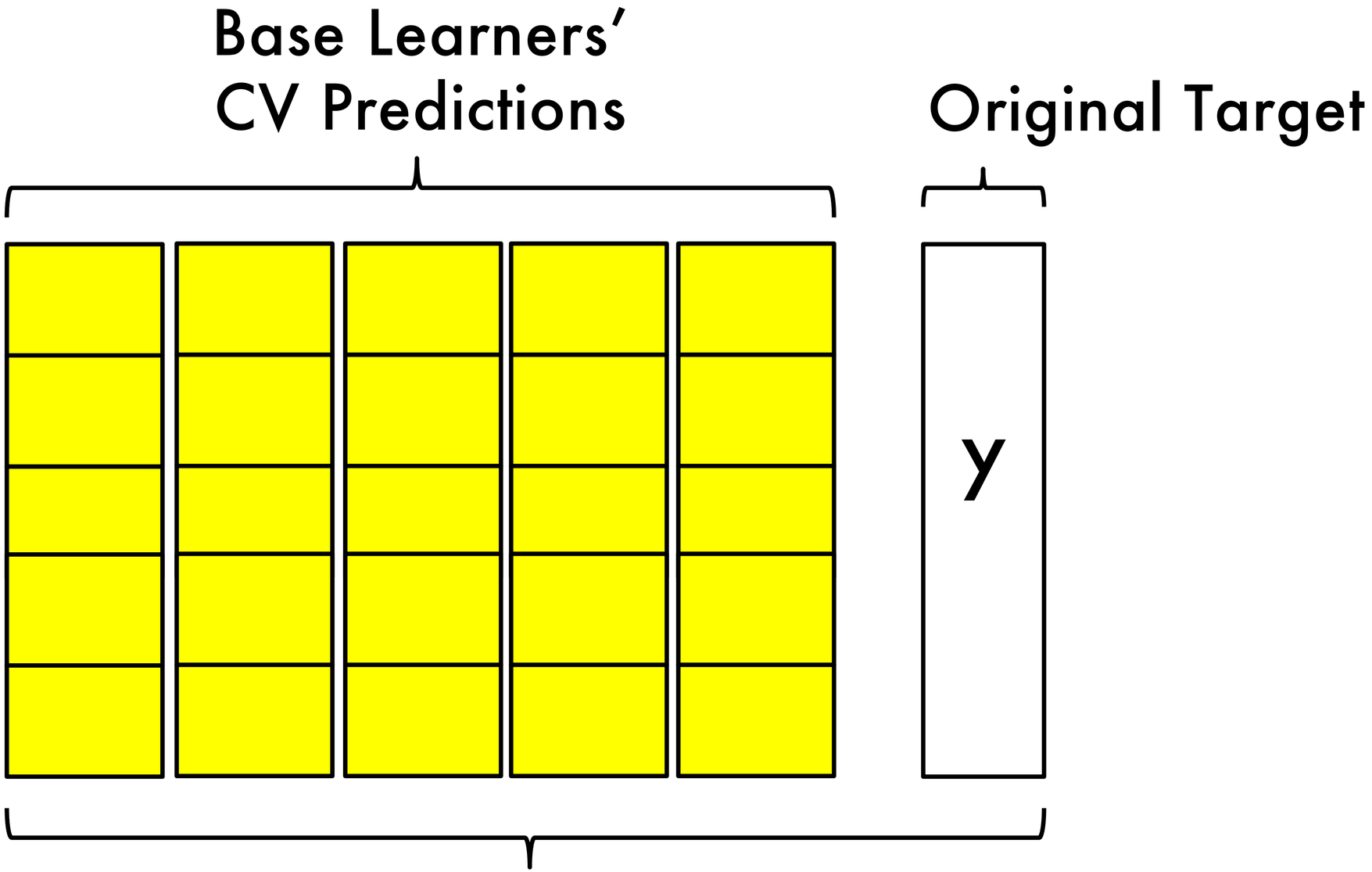
Stacked Ensemble Steps

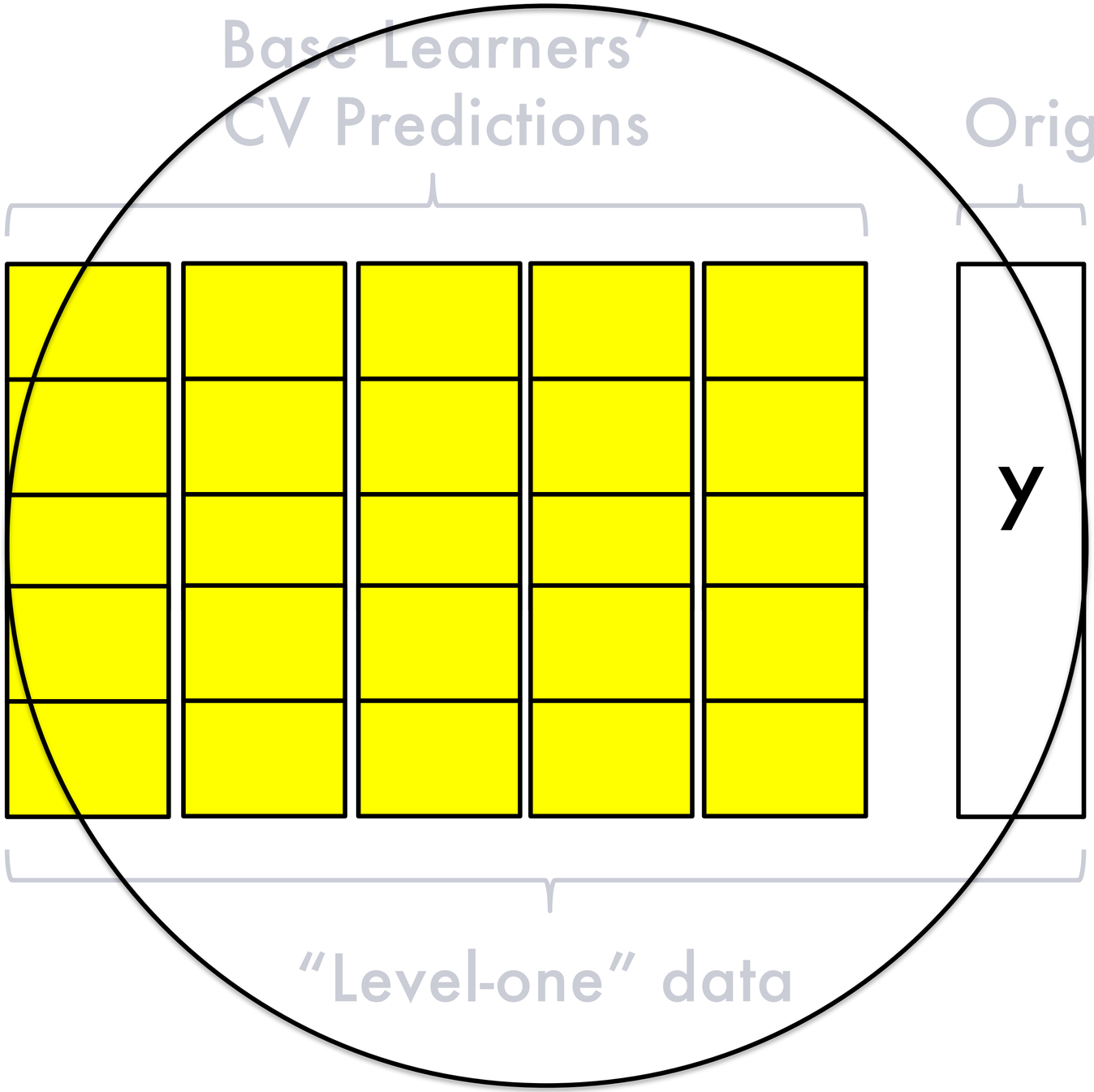
1. Specify original dataset for training
2. Specify algorithms with which to train
3. Stack each algorithm's cross-validated predictions
4. **Train final model on cross-validated results**

Base Learners'
CV Predictions

Original Target




Base Learners'
CV Predictions
Original Target
y
"Level-one" data



train metalearner on level-one data

train **metalearner**
on level-one data

Summary: H2O's AutoML

1. Imputation, Categorical Encoding & Standardization
2. Random Grid Search
3. Early Stopping for Individual Models
4. 2 Stacked Ensembles
5. Auto-selects Winning Model

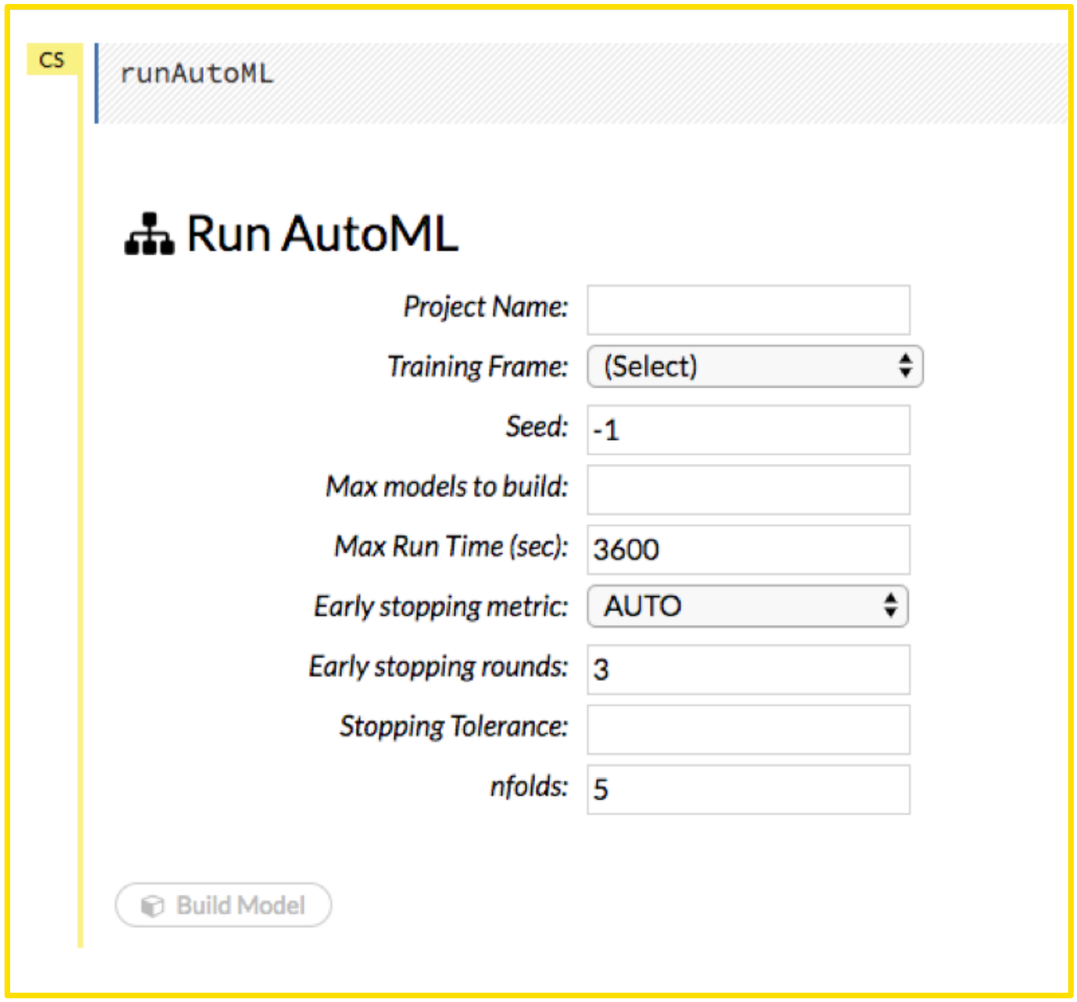
The AutoML Experience

The Interface: Flow

Simplify Machine Learning

2 Required parameters

training frame & response



The Interface: Python

```
from h2o.automl import H2OAutoML

automl = H2OAutoML(max_runtime_secs = 60, seed = 12345)

automl.train(x = predictor_columns, y = target, training_frame = loan_stats)
```

```
automl.leaderboard
```

model_id	auc	logloss
StackedEnsemble_AllModels_0_AutoML_20180113_105834	0.720685	0.385048
StackedEnsemble_BestOfFamily_0_AutoML_20180113_105834	0.720319	0.385191
GLM_grid_0_AutoML_20180113_105834_model_0	0.71506	0.384576
GBM_grid_0_AutoML_20180113_105834_model_0	0.709894	0.387613
GBM_grid_0_AutoML_20180113_105834_model_1	0.703851	0.388938
GBM_grid_0_AutoML_20180113_105834_model_2	0.699872	0.391217
DRF_0_AutoML_20180113_105834	0.690366	0.406982
XRT_0_AutoML_20180113_105834	0.685729	0.39767

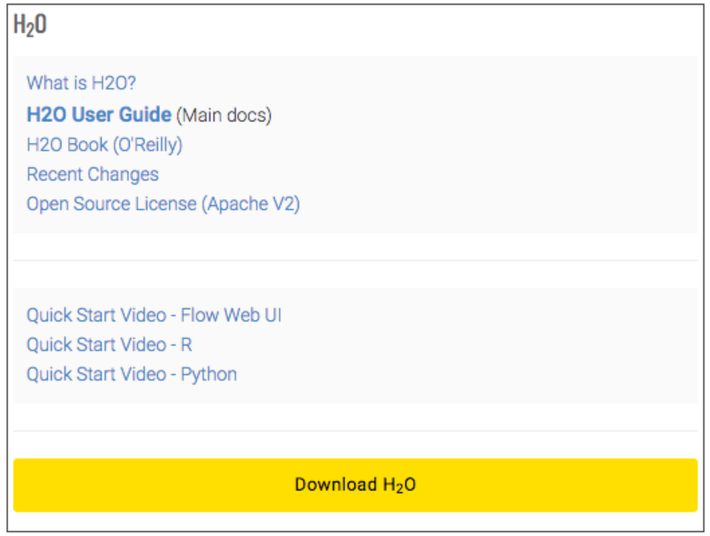
Additional Resources

H2O Open Source Software Documentation | H2O Commercial Software Documentation

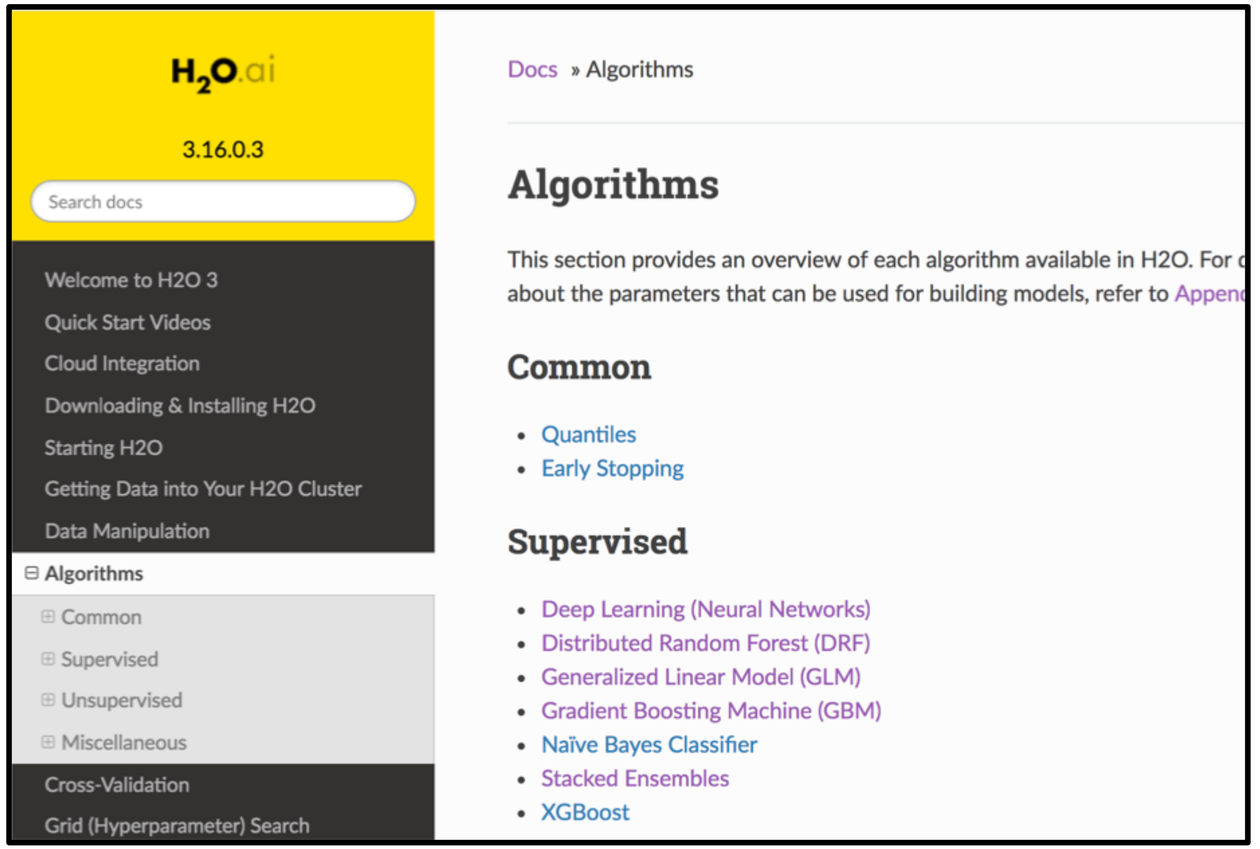
Open Source Software Documentation

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

Getting Started & User Guides



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



post questions to: https://stackoverflow.com/questions/tagged/h2o

Send questions to me: laurend@h2o.ai

H2O.ai

Thanks!