

# Intro to AutoML *with Python*

---

Lauren DiPerna // Data Scientist @ H2O.ai

# Agenda

- Intro to H2O-3 software
- How the Python API works
- What is AutoML
- Demo

# What is H2O?

*A Machine Learning Platform*



# H2O.ai The Company

---

Founded in 2012  
**Advised by** Stanford Professors:  
Hastie, Tibshirani & Boyd

Headquarters: Mountain View, California, USA

# H2O-3 The Platform



**Open Source Software**  
R, Python, Scala and Web Interfaces  
Distributed Machine Learning Algorithms

# H2O-3: The Machine Learning Platform

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



**Parallel:** uses many CPUs



**Distributed:** uses many machines

# H2O-3: The Machine Learning Platform

**Core Algorithms are written in high-performance Java**

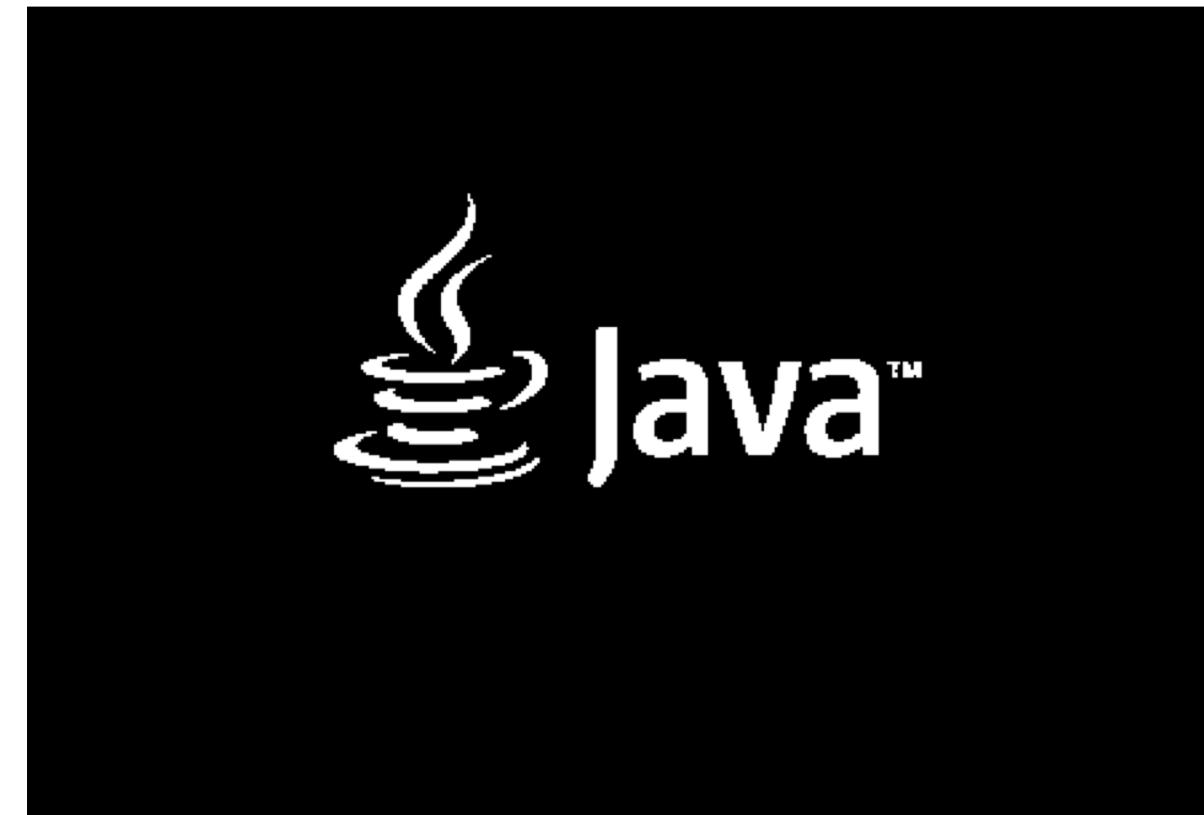
**14+ ALGORITHMS**



- |                      |                          |
|----------------------|--------------------------|
| Statistical Analysis | Dimensionality Reduction |
| Ensembles            | Word Embedding           |
| Deep Neural Networks | Time Series              |
| Clustering           | ML Tuning                |

# H2O-3: The Machine Learning Platform

**Core Algorithms are written in high-performance Java**



**Supervised & Unsupervised**

# H2O-3: The Machine Learning Platform

**Multiple APIs**



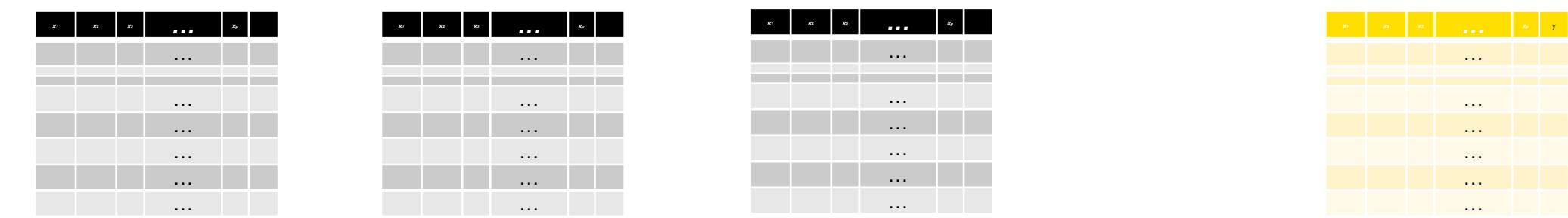
**H<sub>2</sub>O Flow**

 **Scala**

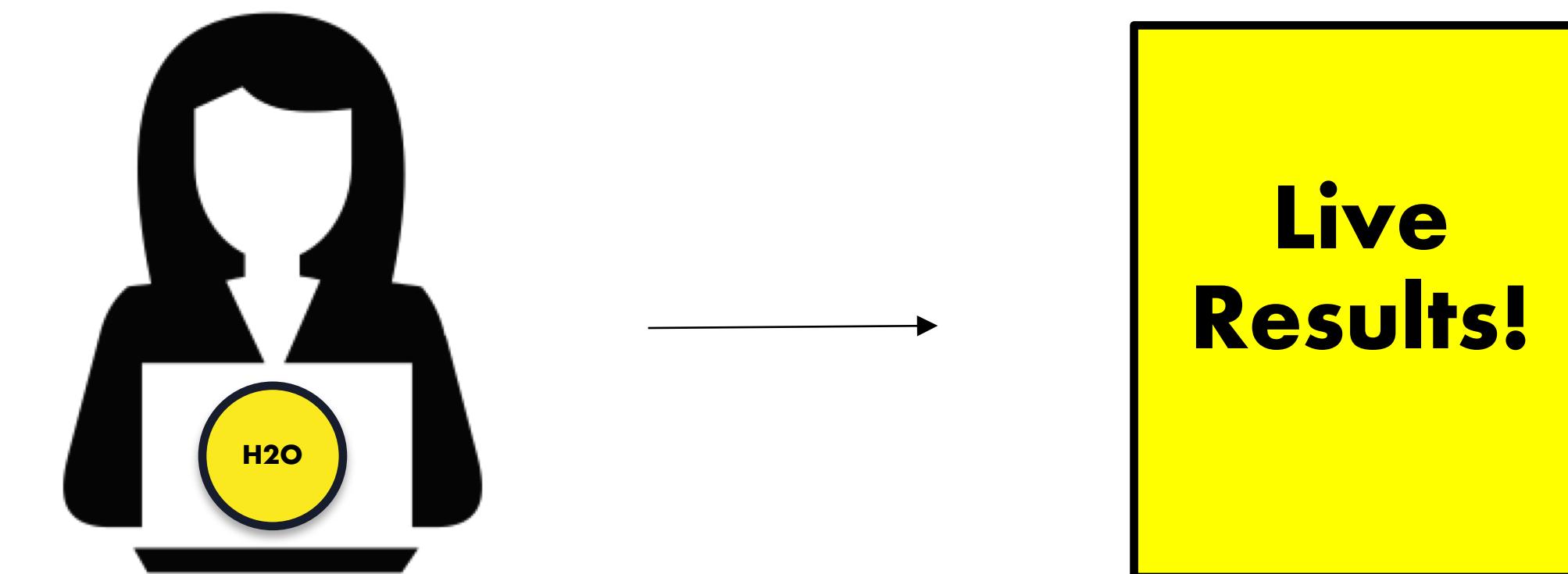
The Scala logo icon consists of three thick, dark gray vertical bars of decreasing height from left to right, resembling a stylized "S".

# Benefits of H2O-3

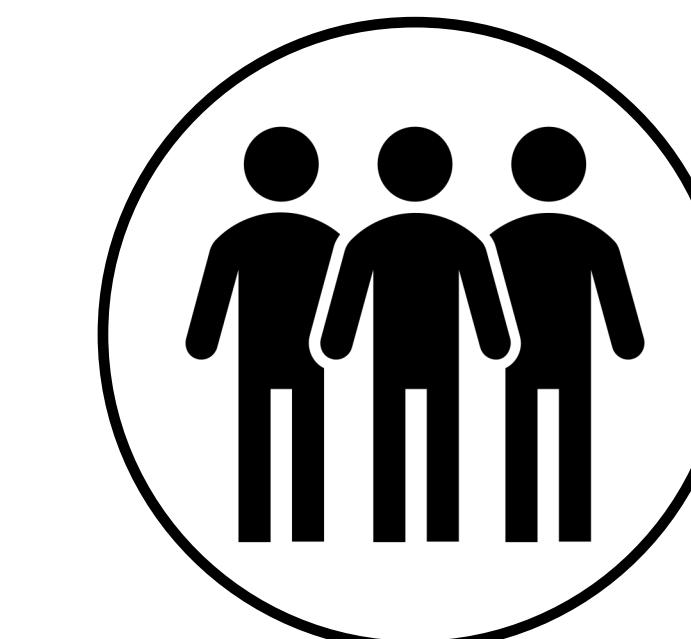
No Data Size Limit



Easy Deployment



Easy Learning Curve



♥ R + ♥ Py. + ♥ UI

# What is H2O-3?



# What is H2O-3?

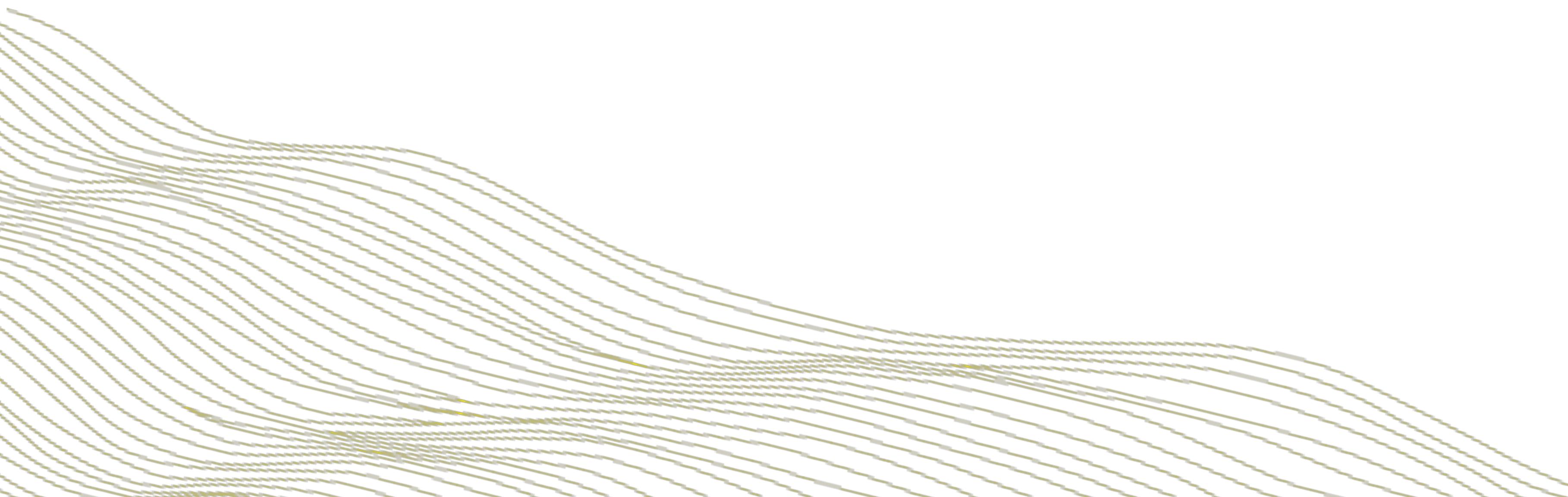


`pip install h2o`

`conda install -c h2oai h2o`

# H2O Cluster & H2O Frame

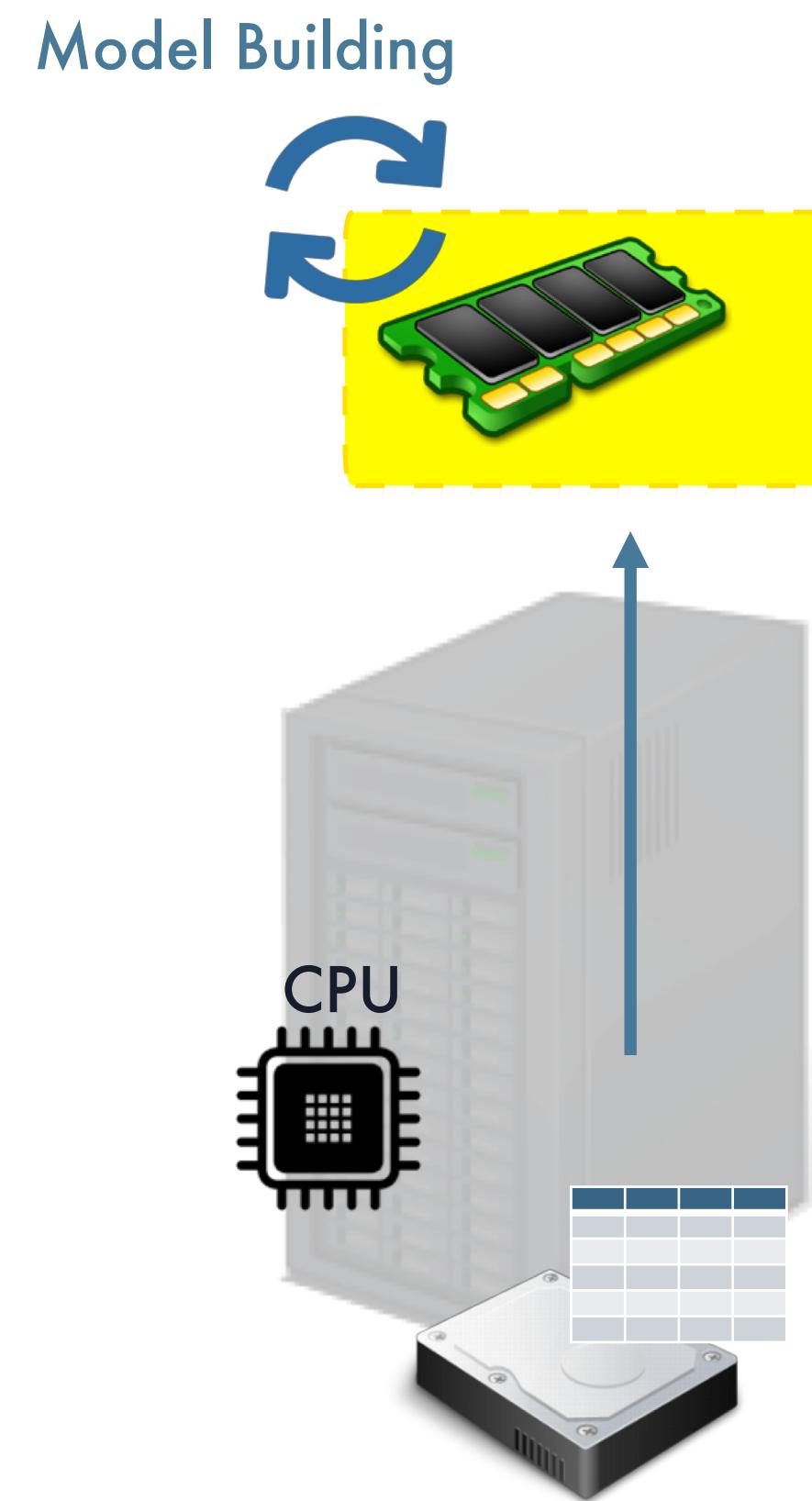
**2** key concepts



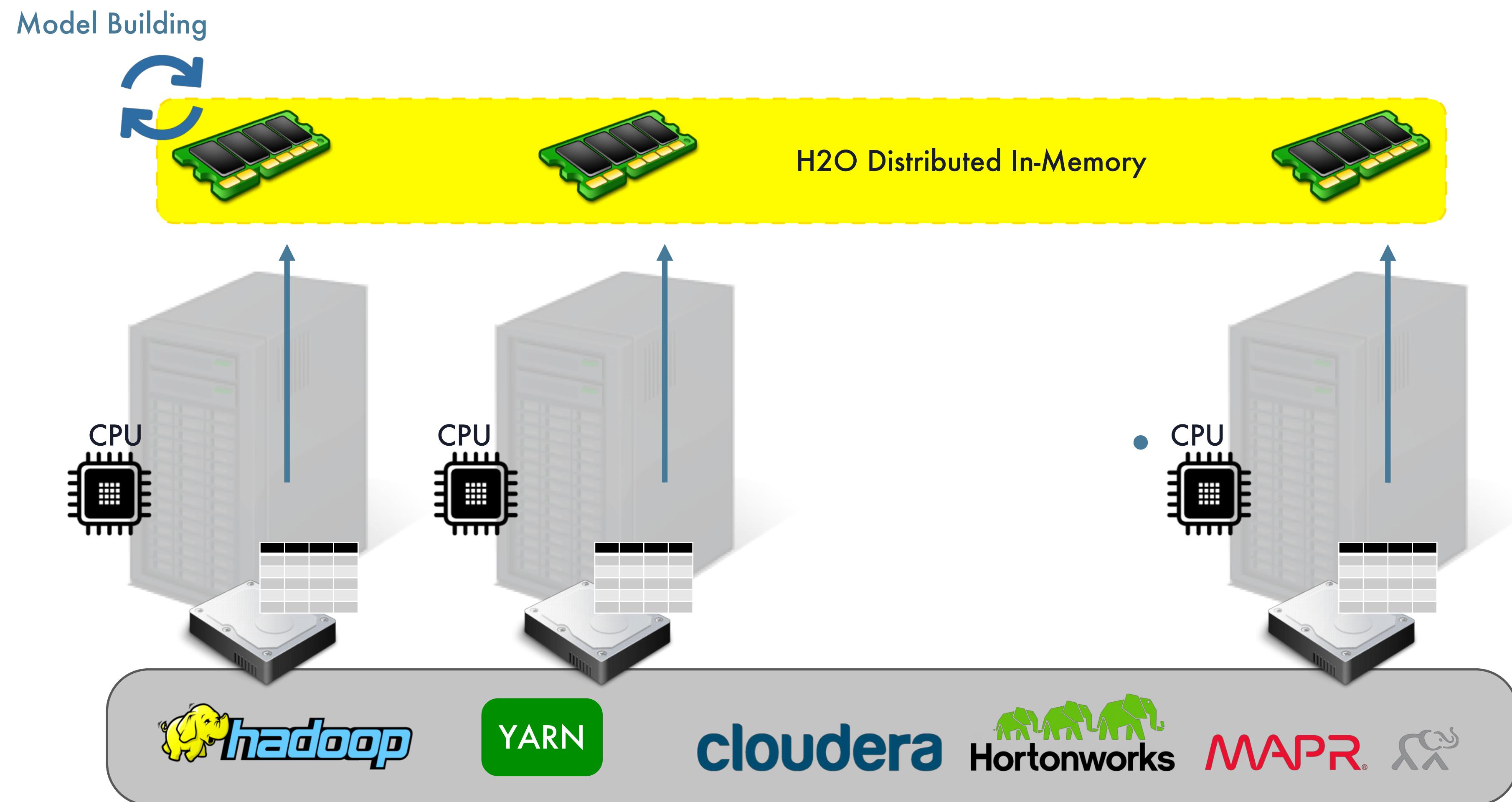
# The H2O Cluster



# The H2O Cluster



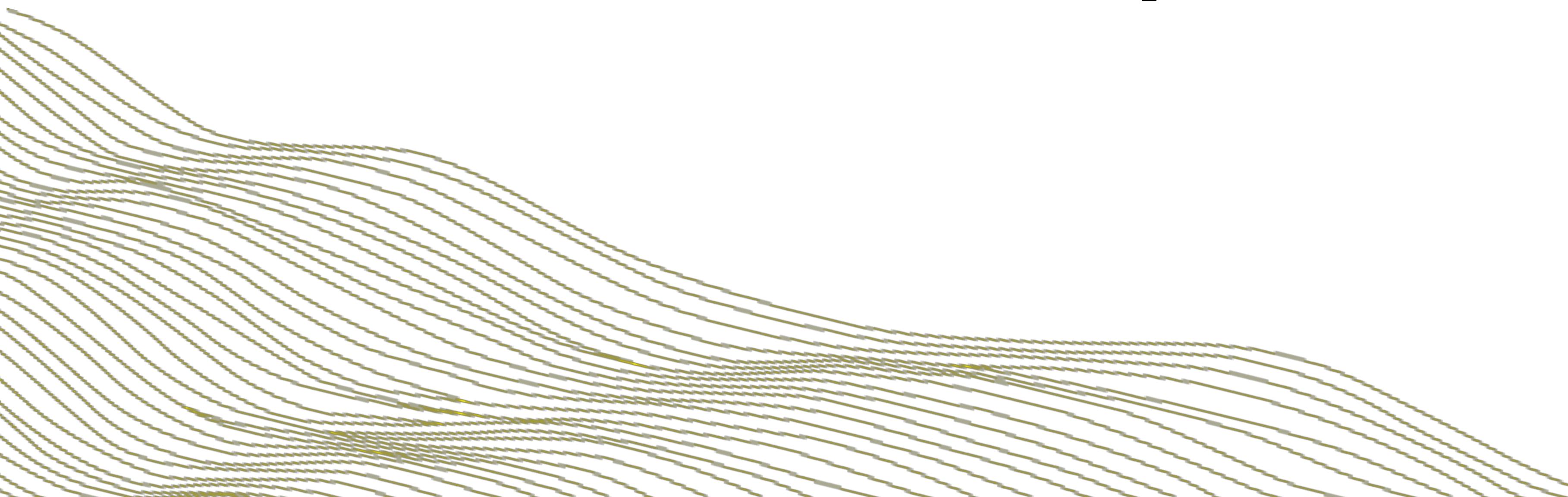
# The H2O Cluster



# The Distributed H2O Frame



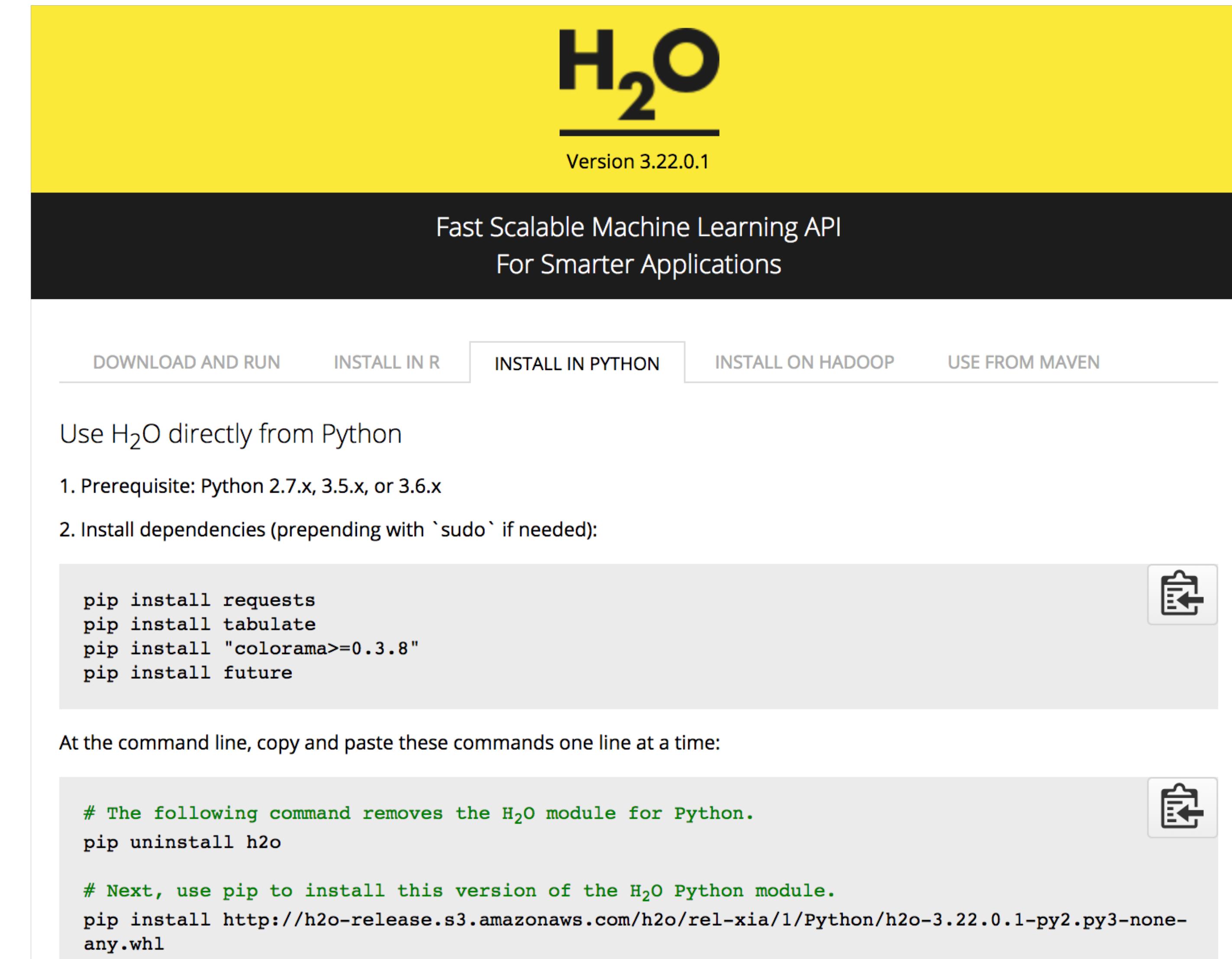
# The H2O Python API



# Importing the Python Package

If you want to use  
the H2O Python API ...

You have to import the  
**H2O Python Module**



The image shows a screenshot of the H2O Python API documentation page. At the top, the H2O logo and version 3.22.0.1 are displayed. Below the logo, the text "Fast Scalable Machine Learning API" and "For Smarter Applications" is shown. A navigation bar at the bottom includes links for "DOWNLOAD AND RUN", "INSTALL IN R", "INSTALL IN PYTHON" (which is highlighted), "INSTALL ON HADOOP", and "USE FROM MAVEN". The main content area starts with the heading "Use H2O directly from Python" followed by two numbered steps: "1. Prerequisite: Python 2.7.x, 3.5.x, or 3.6.x" and "2. Install dependencies (prepend with `sudo` if needed):". Below these steps is a code block containing the following pip commands:

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

At the bottom of the content area, there is a note: "At the command line, copy and paste these commands one line at a time:". Below this note is another code block:

```
# The following command removes the H2O module for Python.
pip uninstall h2o

# Next, use pip to install this version of the H2O Python module.
pip install http://h2o-release.s3.amazonaws.com/h2o/rel-xia/1/Python/h2o-3.22.0.1-py2.py3-none-any.whl
```

Two small clipboard icons are located in the top right corner of the content area.

# Communication



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

```
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/BostonHousing.csv")

# 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
# crosses river)
boston['chas'] = boston['chas'].asfactor()

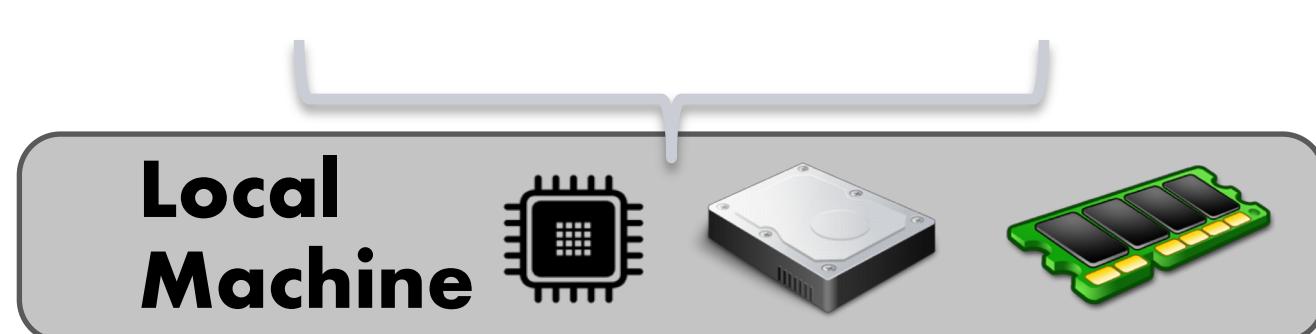
# split into train and validation sets
train, valid = boston.split_frame(ratios = [.8])

# 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

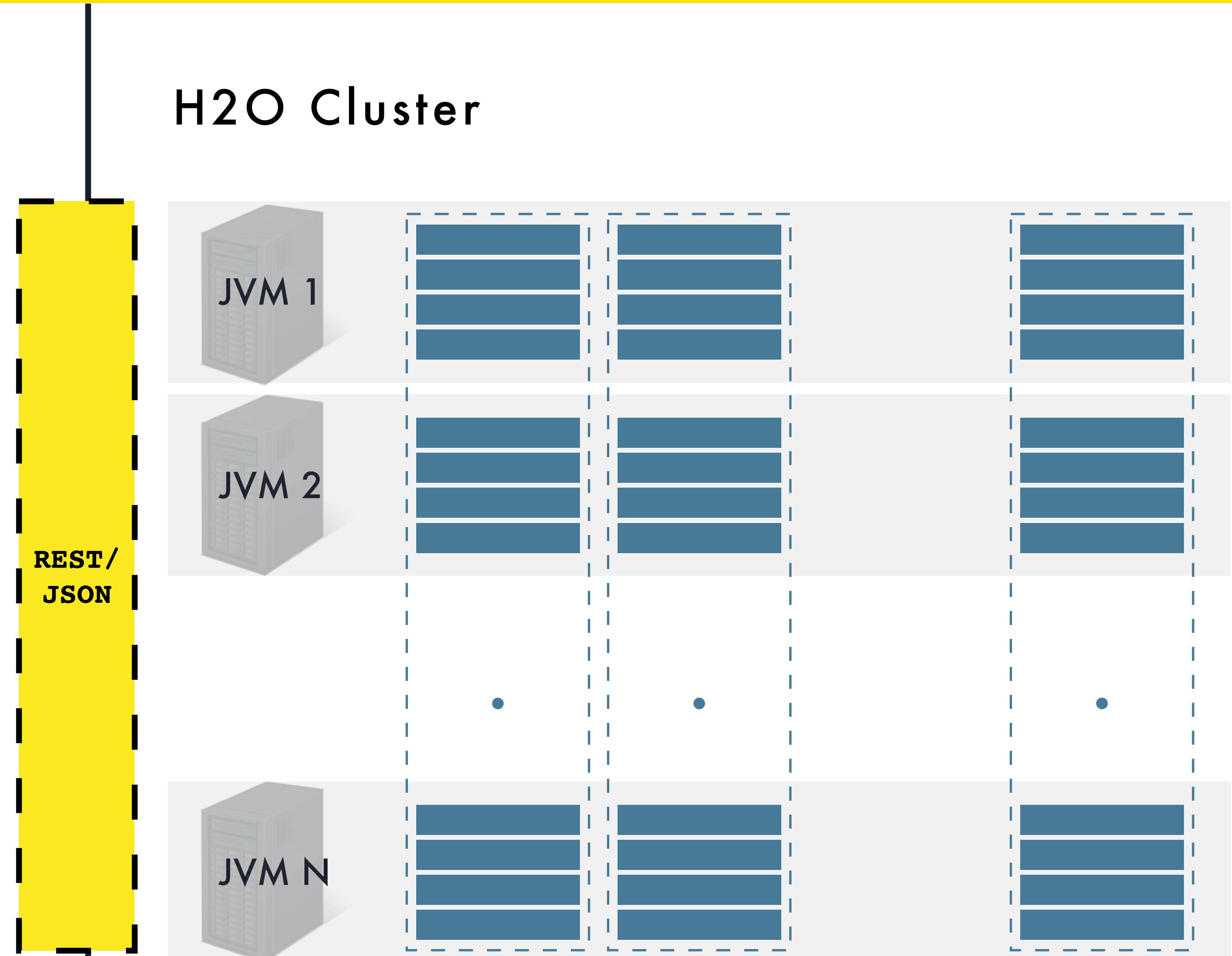
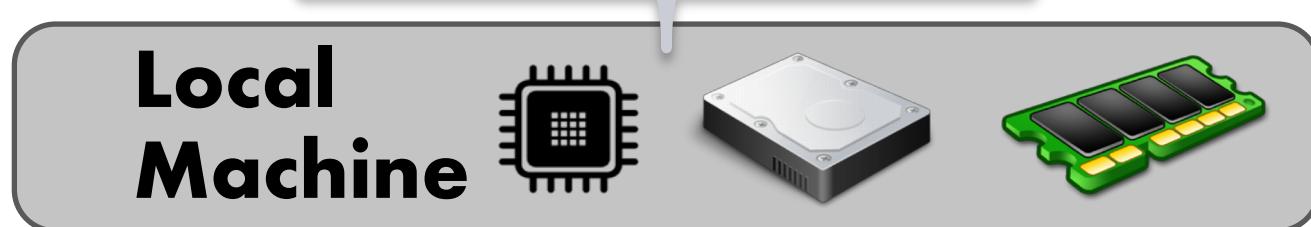
H2O Cluster



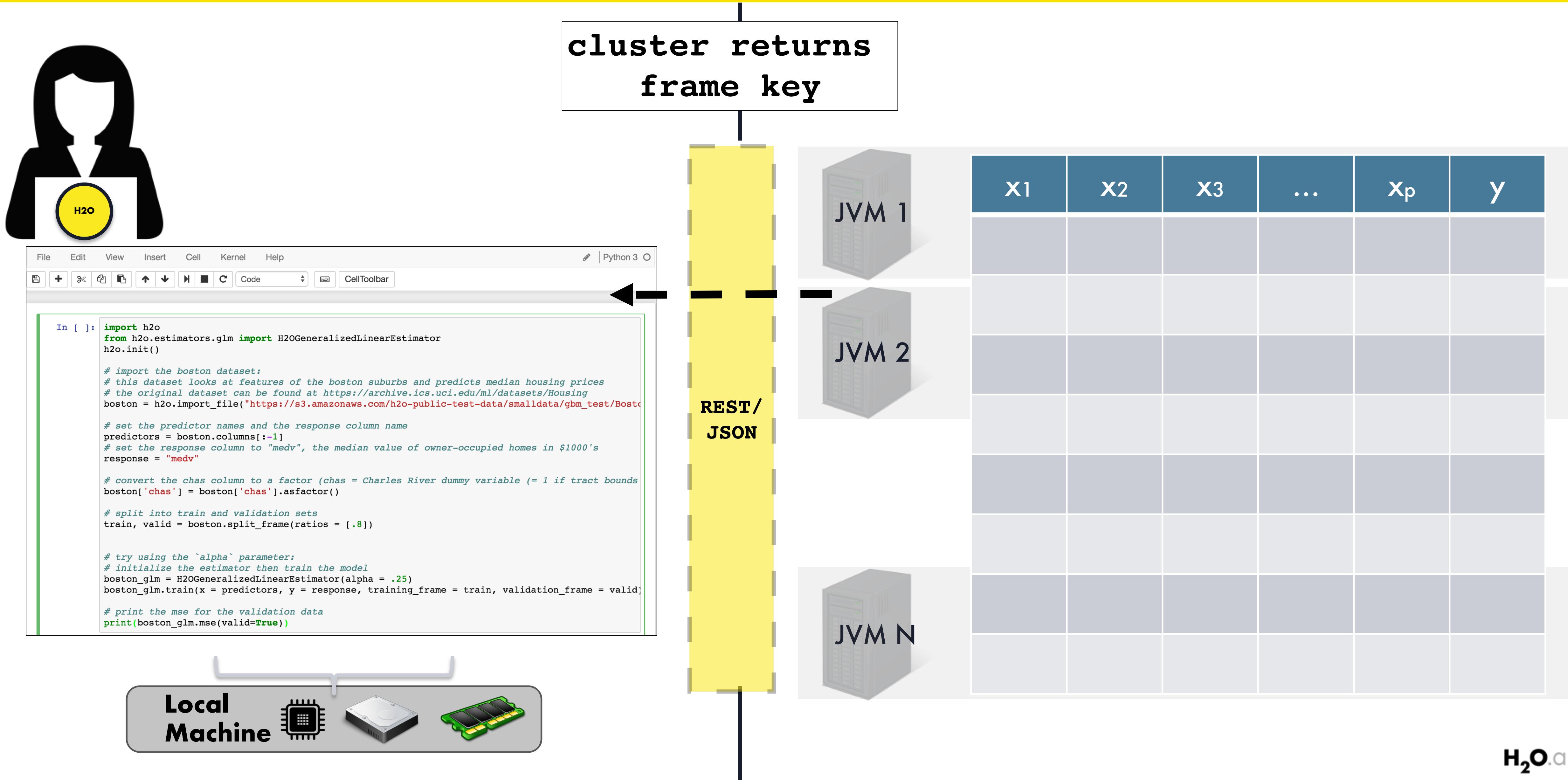
# Cluster Does Heavy Lifting



```
File Edit View Insert Cell Kernel Help Python 3  
Cell Toolbar  
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/Bos...  
  
# 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))
```



# H2O Client

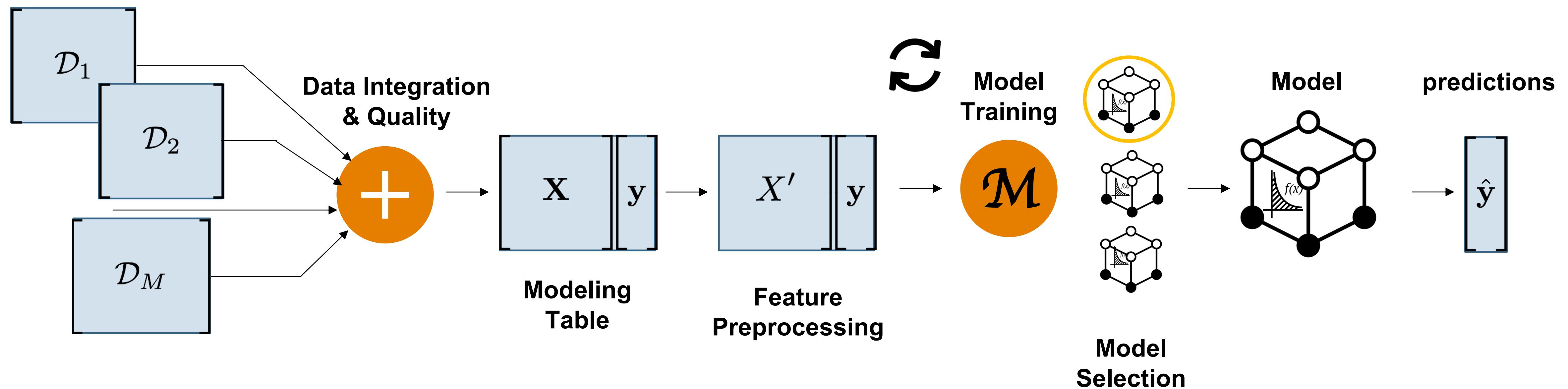


# Auto-What?

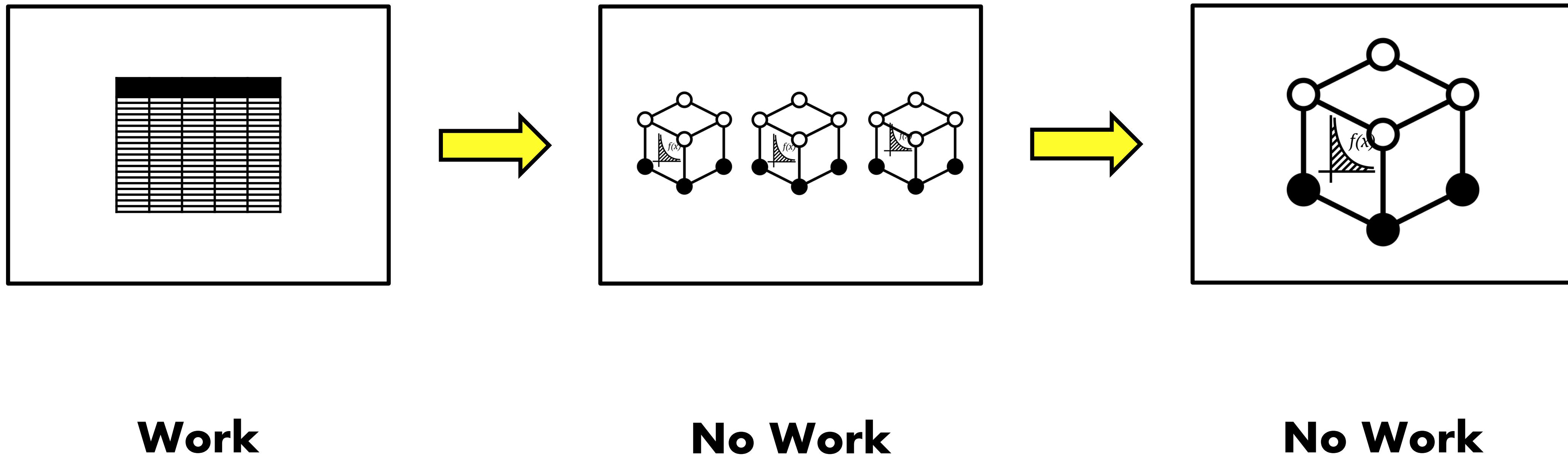
*What is AutoML?*



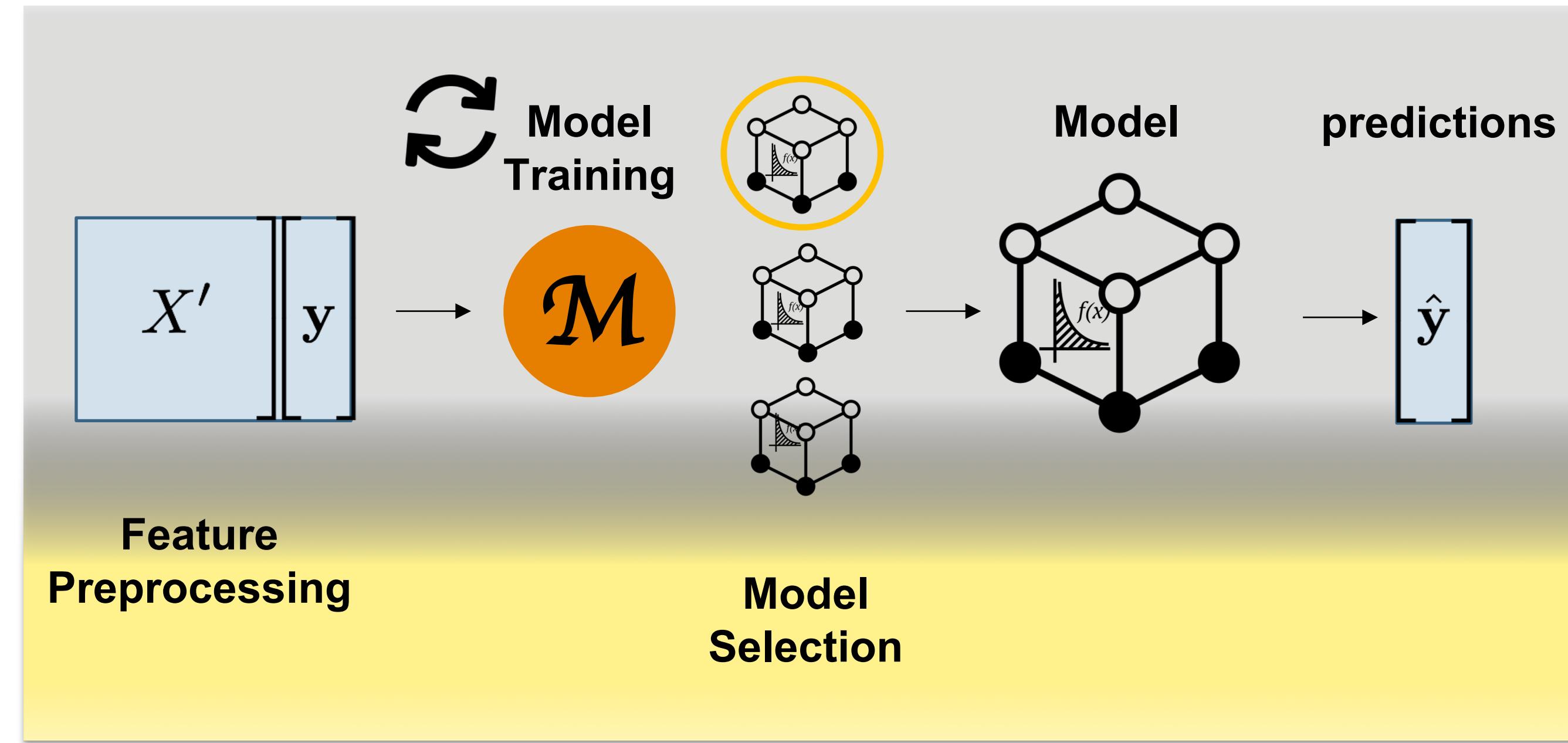
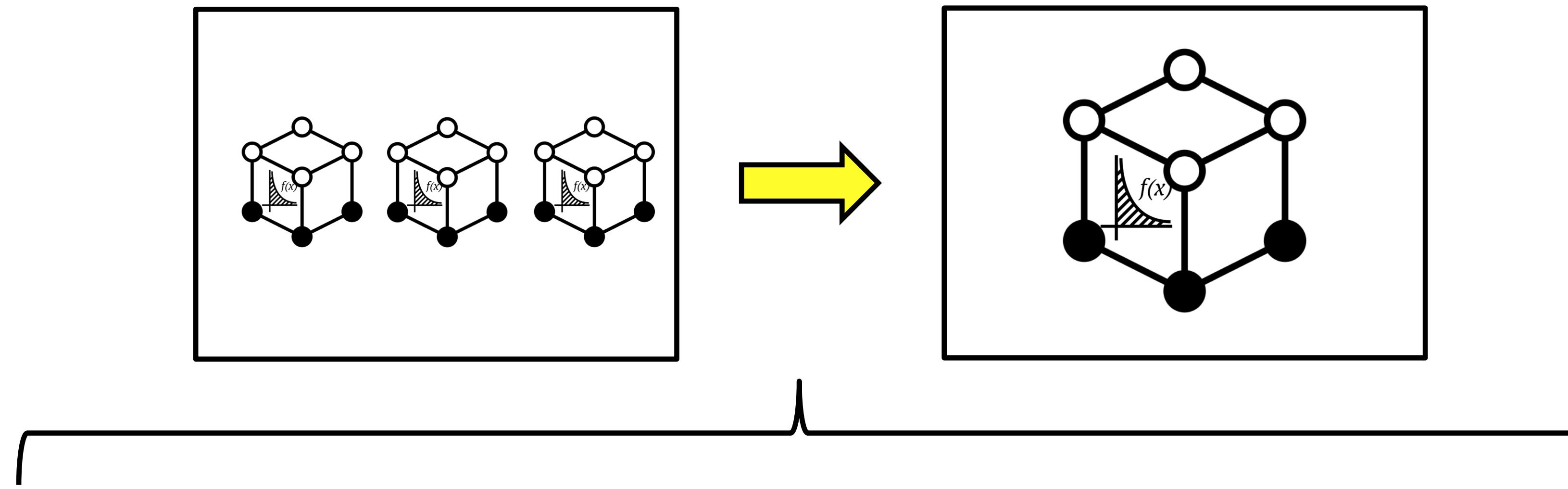
# The Machine Learning Pipeline



# What is Automatic Machine Learning?



# What is Automatic Machine Learning?

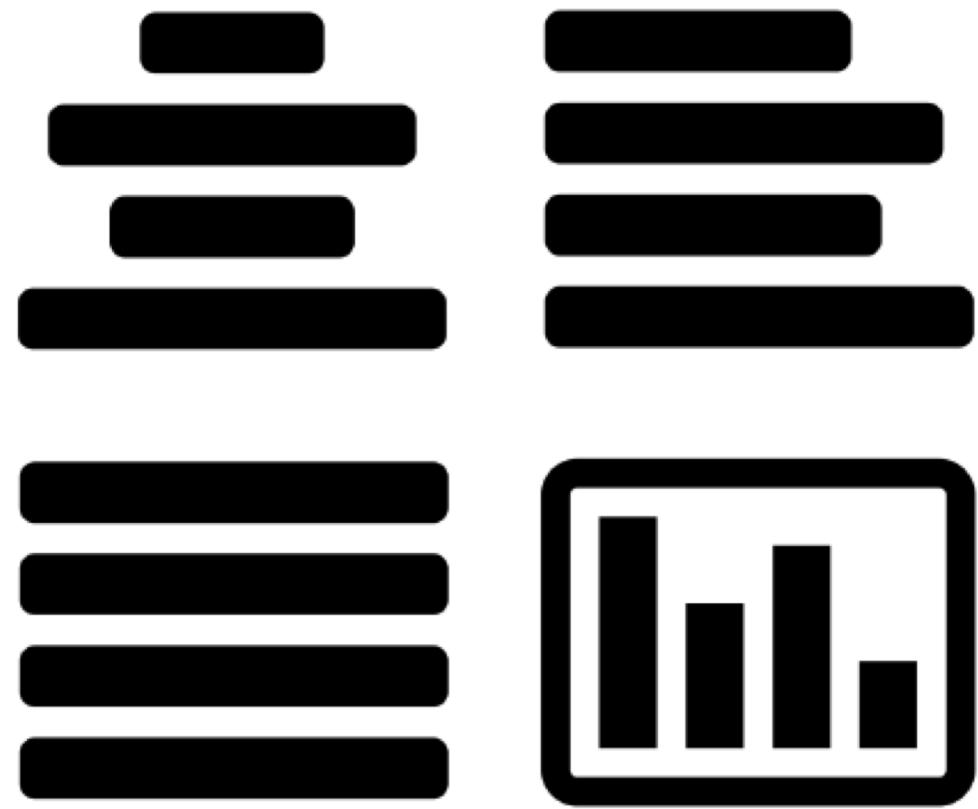


# Automatic Machine Learning

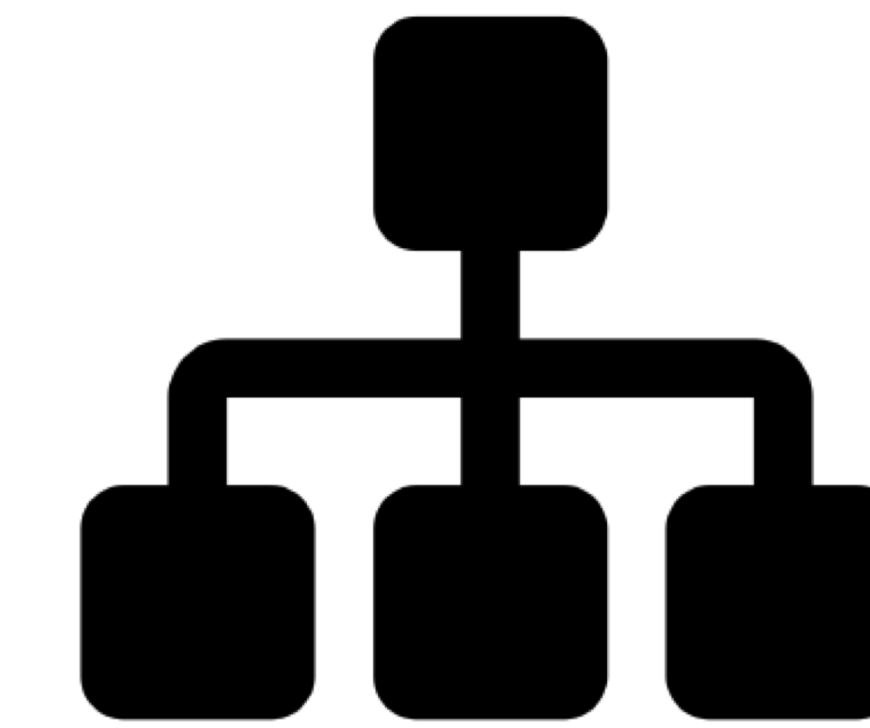
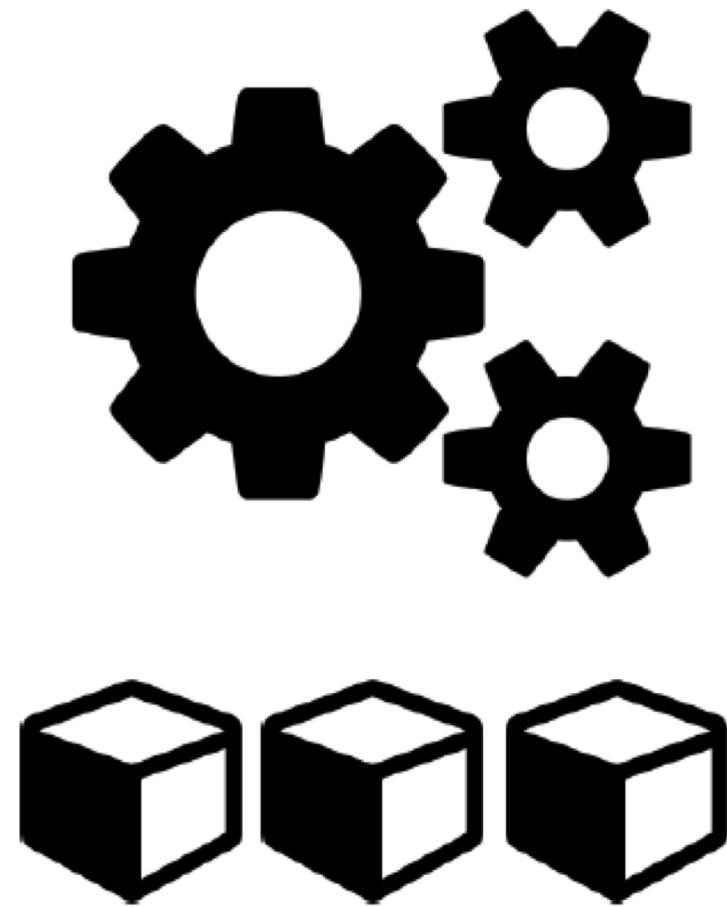
*The Aspects of AutoML*



# Model Generation



Data Prep

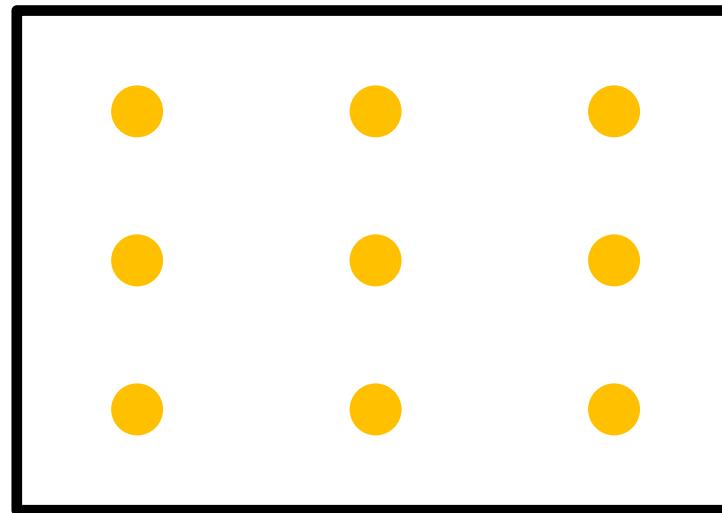


Ensembles

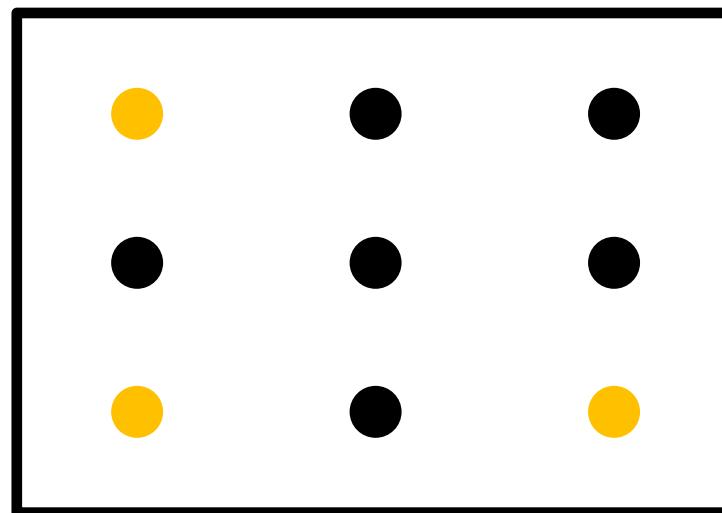
# Data Prep

- **Format Data:**
  - Imputation, one-hot encoding, standardization
- **Choose Columns:**
  - Feature selection, feature extraction (e.g. PCA)
- **Advanced Encoding:**
  - Count, Label, Target encoding of categorical features

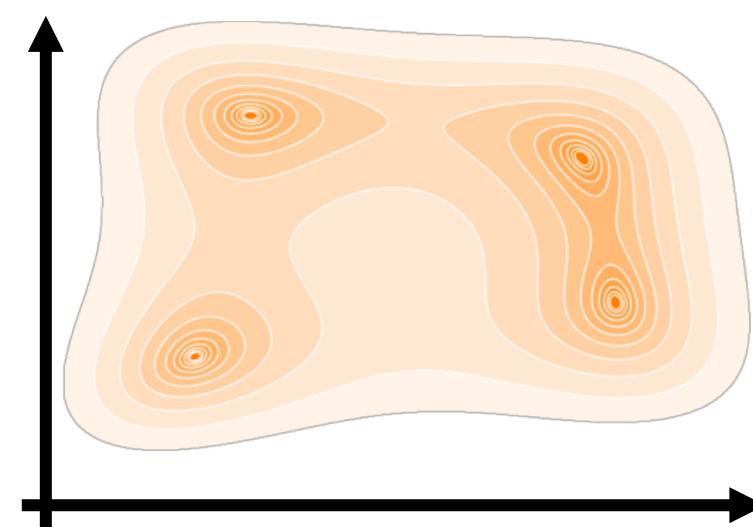
# Model Generation



**Cartesian grid search**

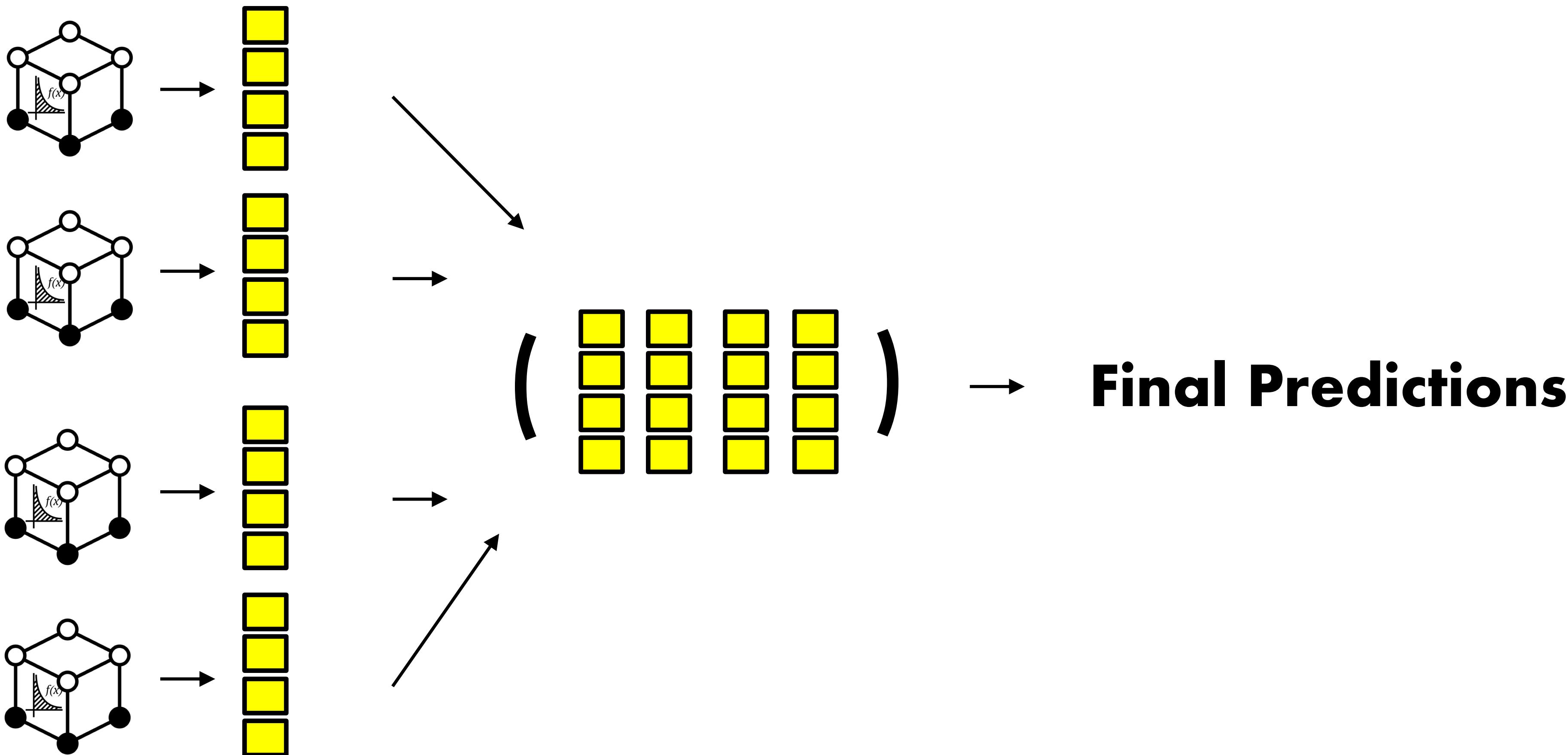


**Random grid search**



**Bayesian Hyperparameter Optimization**

# Ensembles



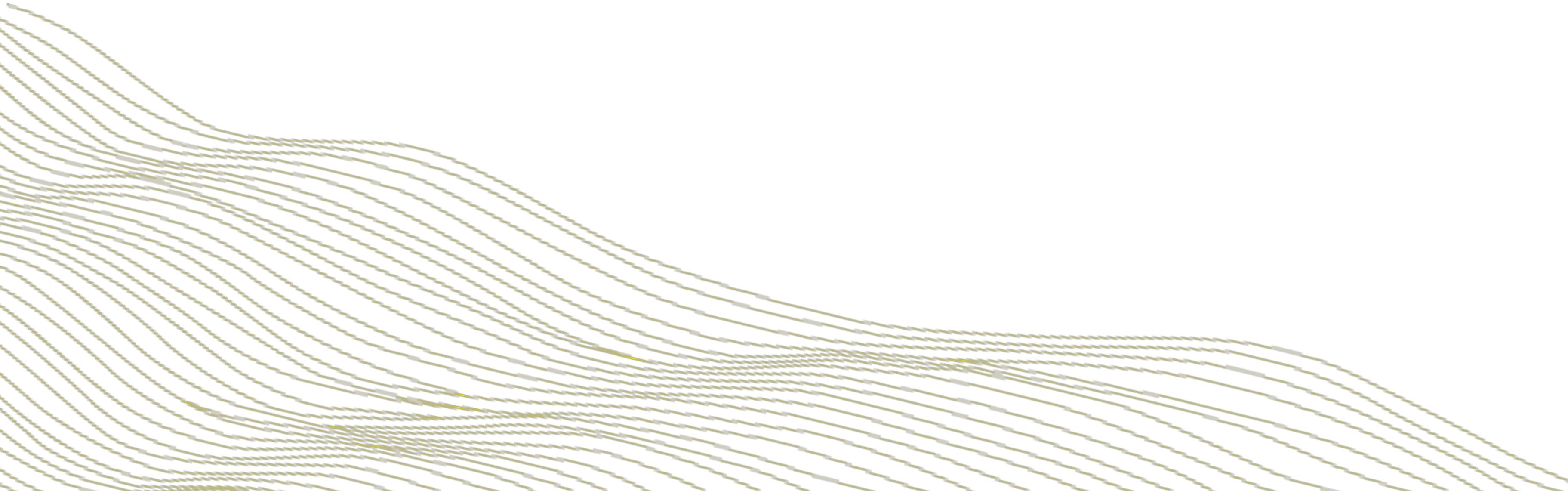
# Ensembles

**Ensembles often out-perform individual models**

- Bagging
- Boosting
- Stacking / Super Learning (Wolpert, Breiman)
- Ensemble Selection (Caruana)

# H2O's AutoML

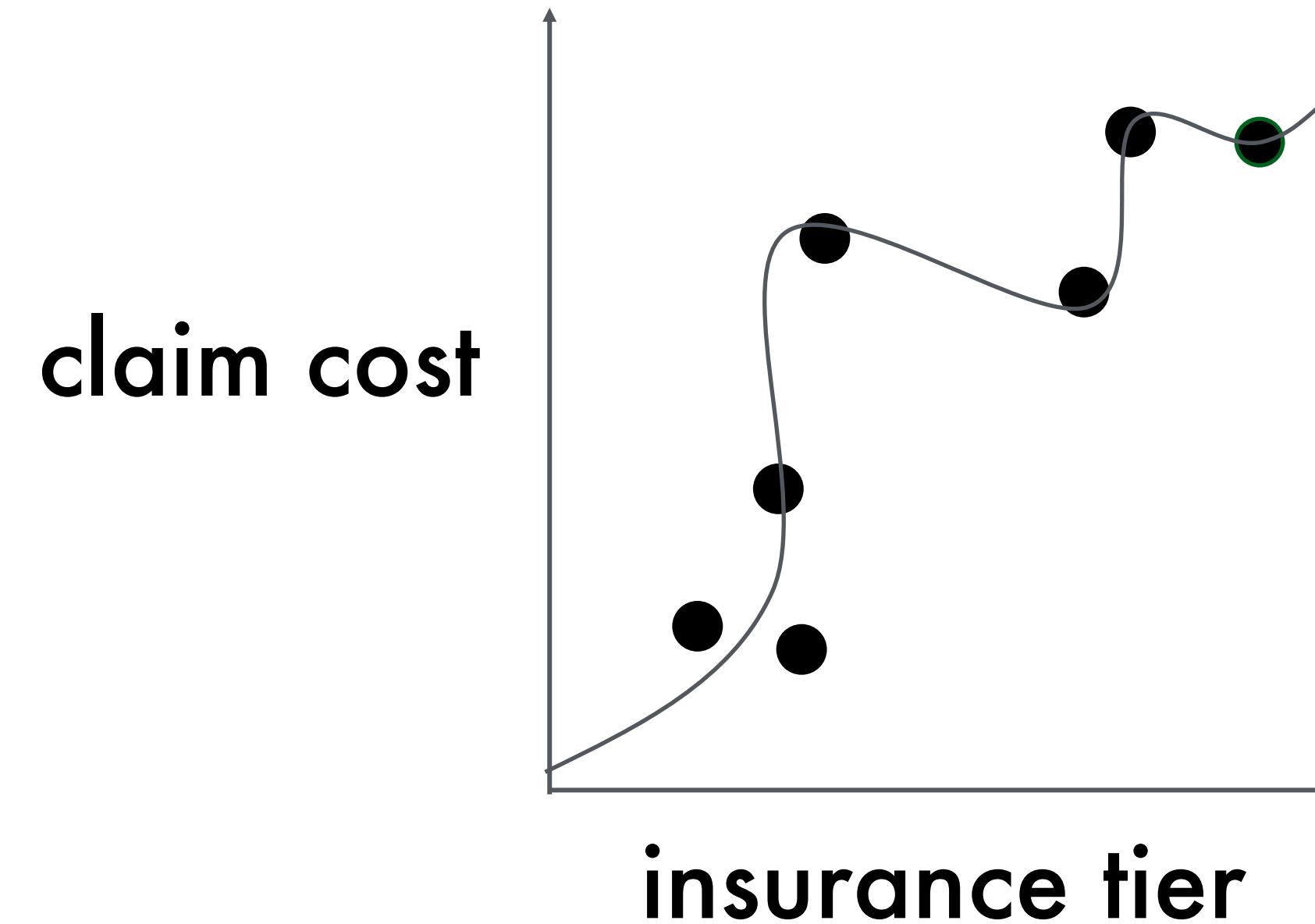
*Scalable Automatic Machine Learning*



# AutoML: Supervised Learning

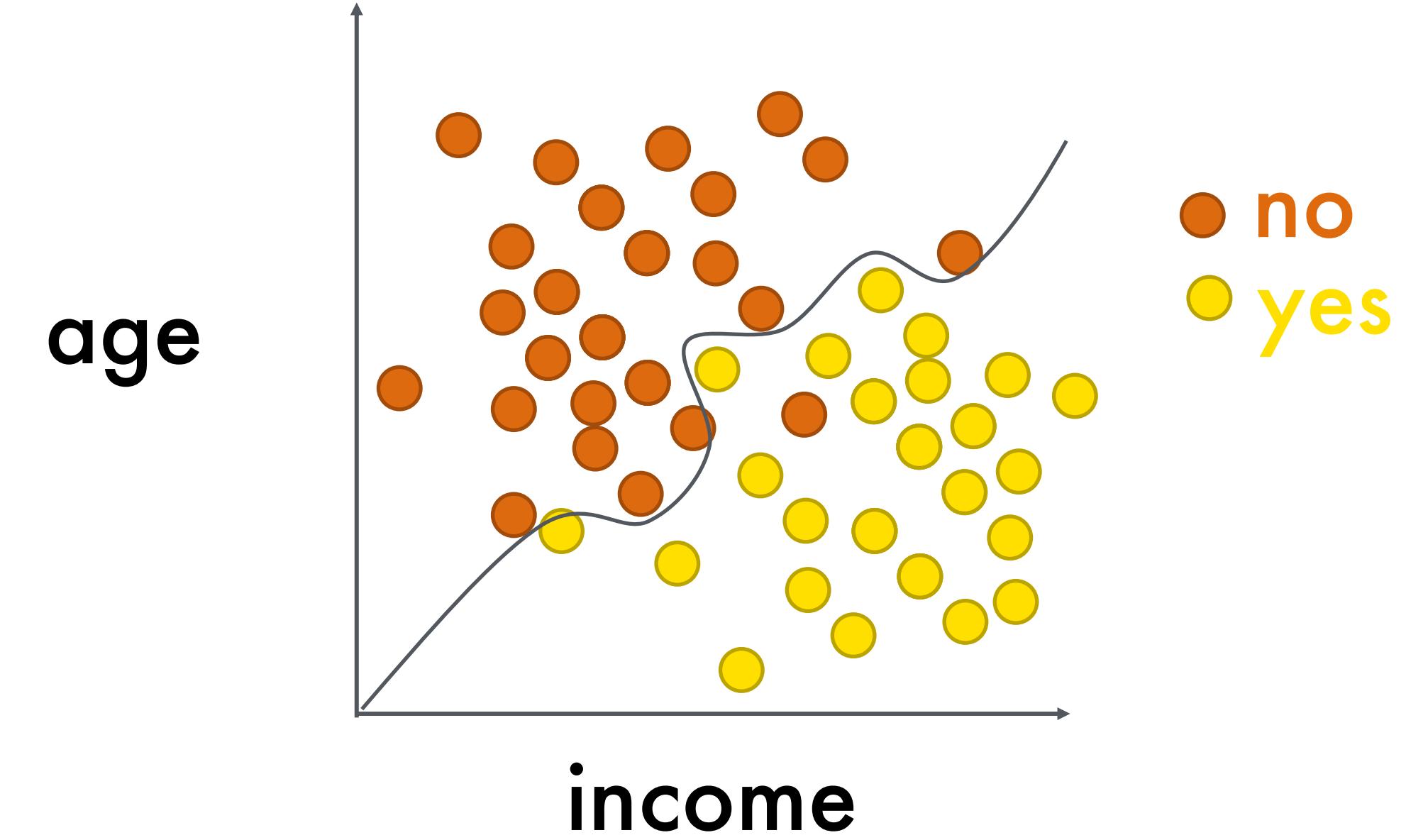
Regression:

*How much will a claim cost?*



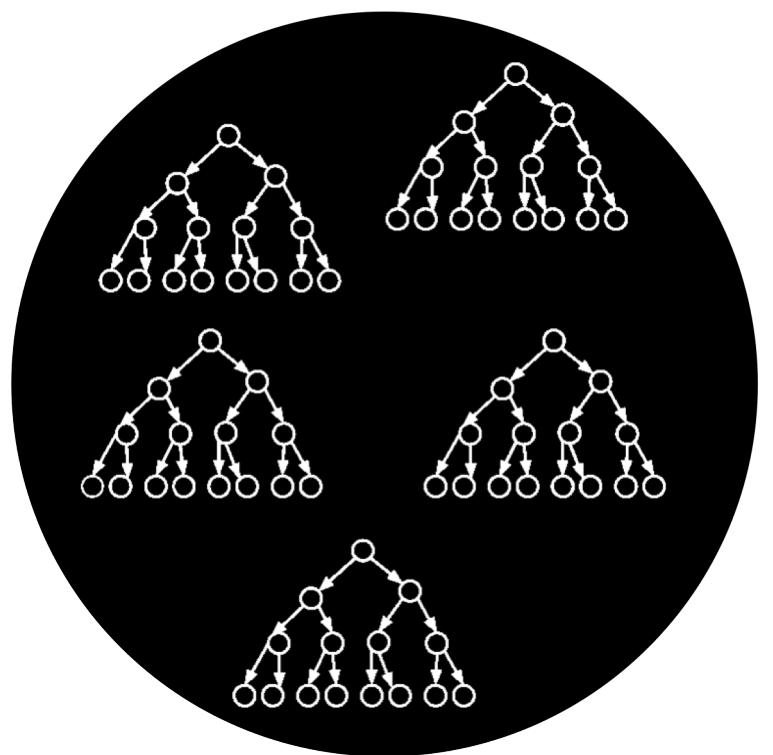
Classification:

*Will a borrower pay off their loan?*

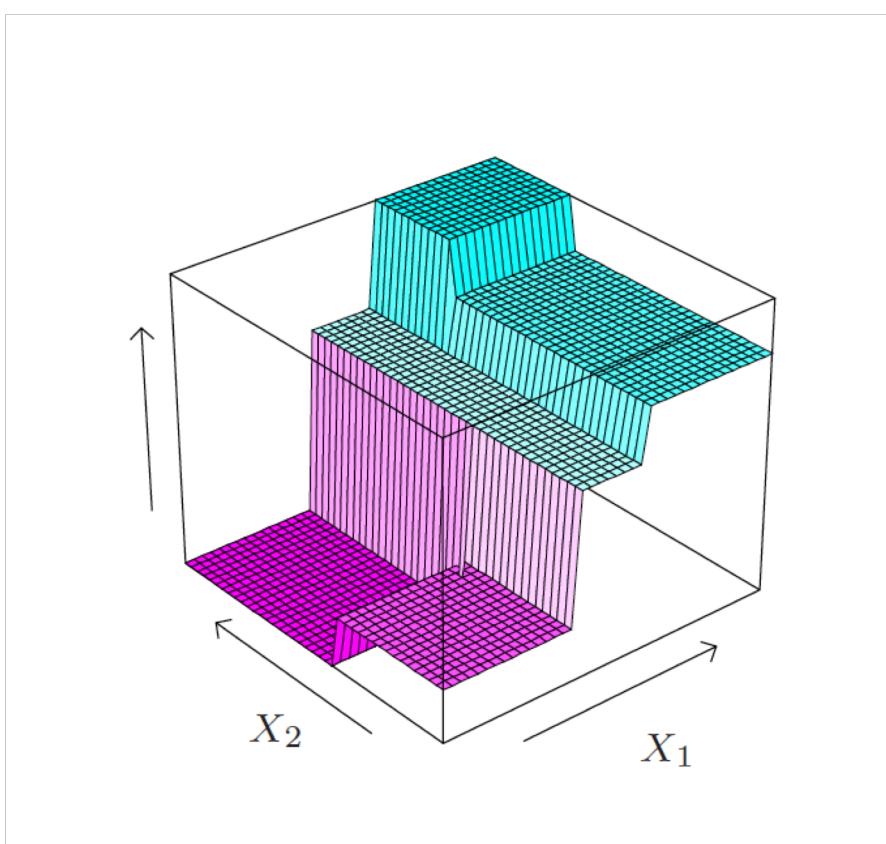


# H2O AutoML: Available Algos

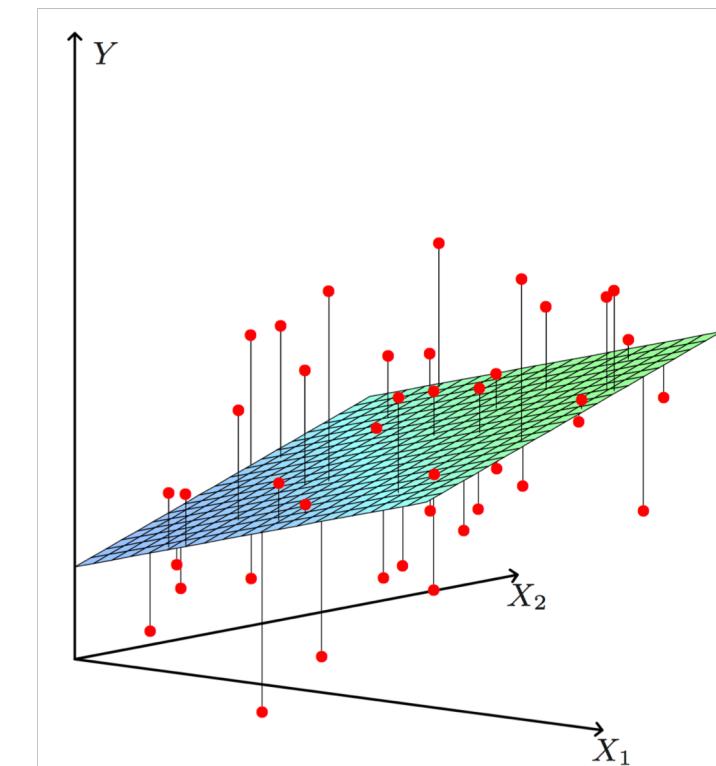
**DRF**  
**XRT**



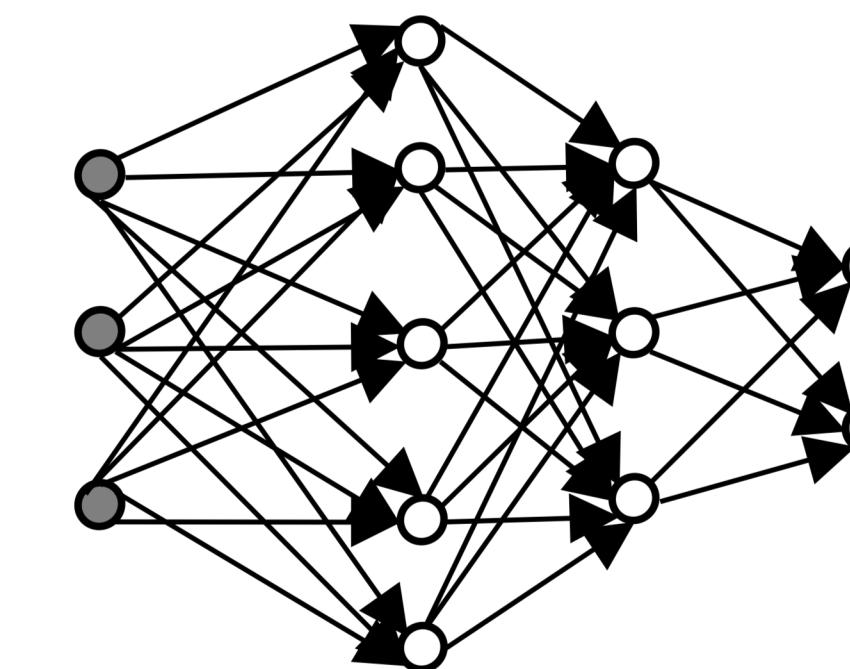
**GBM**  
**XGBoost**



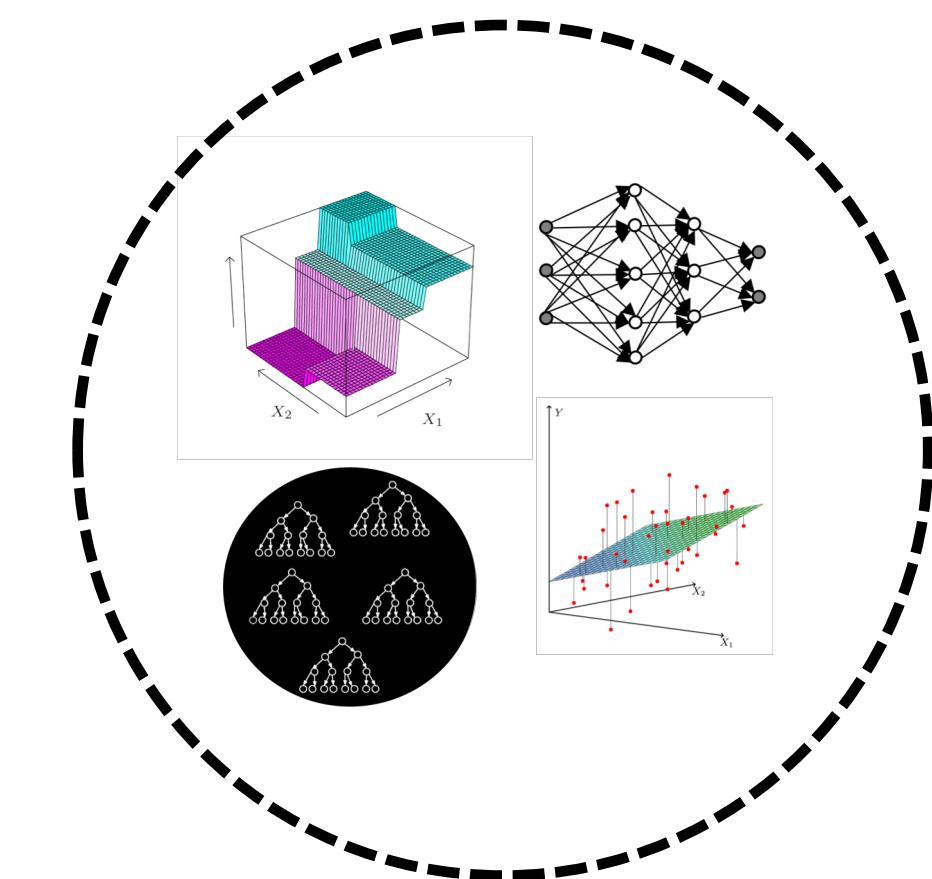
**GLM**



**DNN**



**Stacked  
Ensemble**



# AutoML: Data Preprocessing

Inherits from **H2O Algorithm**

Imputation

NA
10
50

→

30
10
50

Standardization

30
10
50

→

0
-1
1

One-hot Encoding

dog
cat
dog

→

1
0
1

0
1
0

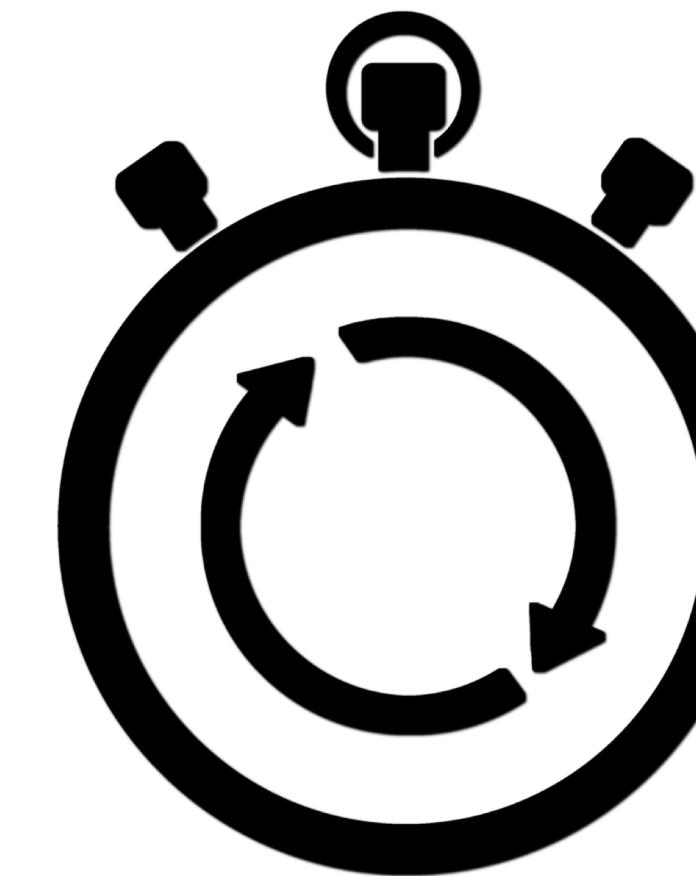
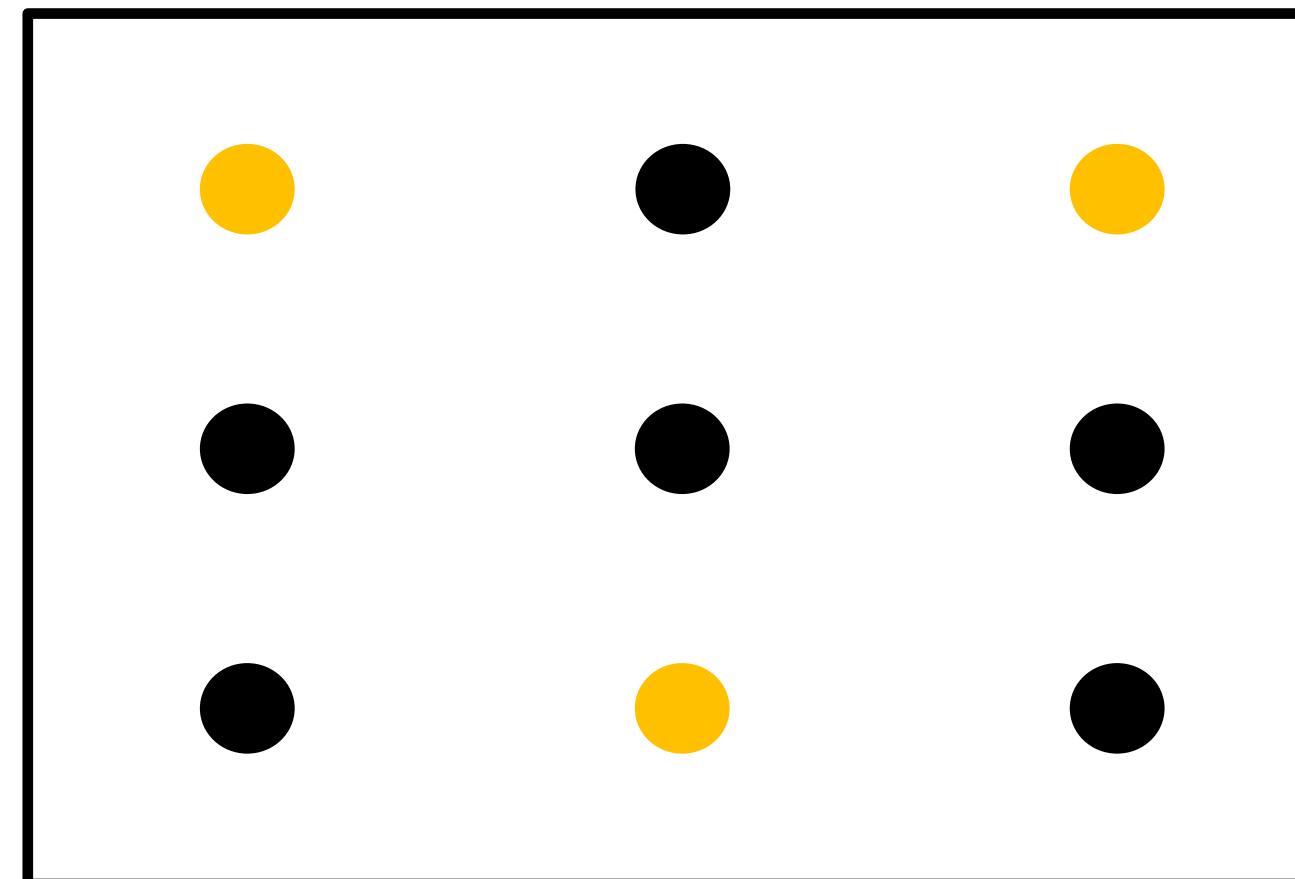
Label Encoder

dog
cat
dog

→

2
1
2

# AutoML: Model Generation

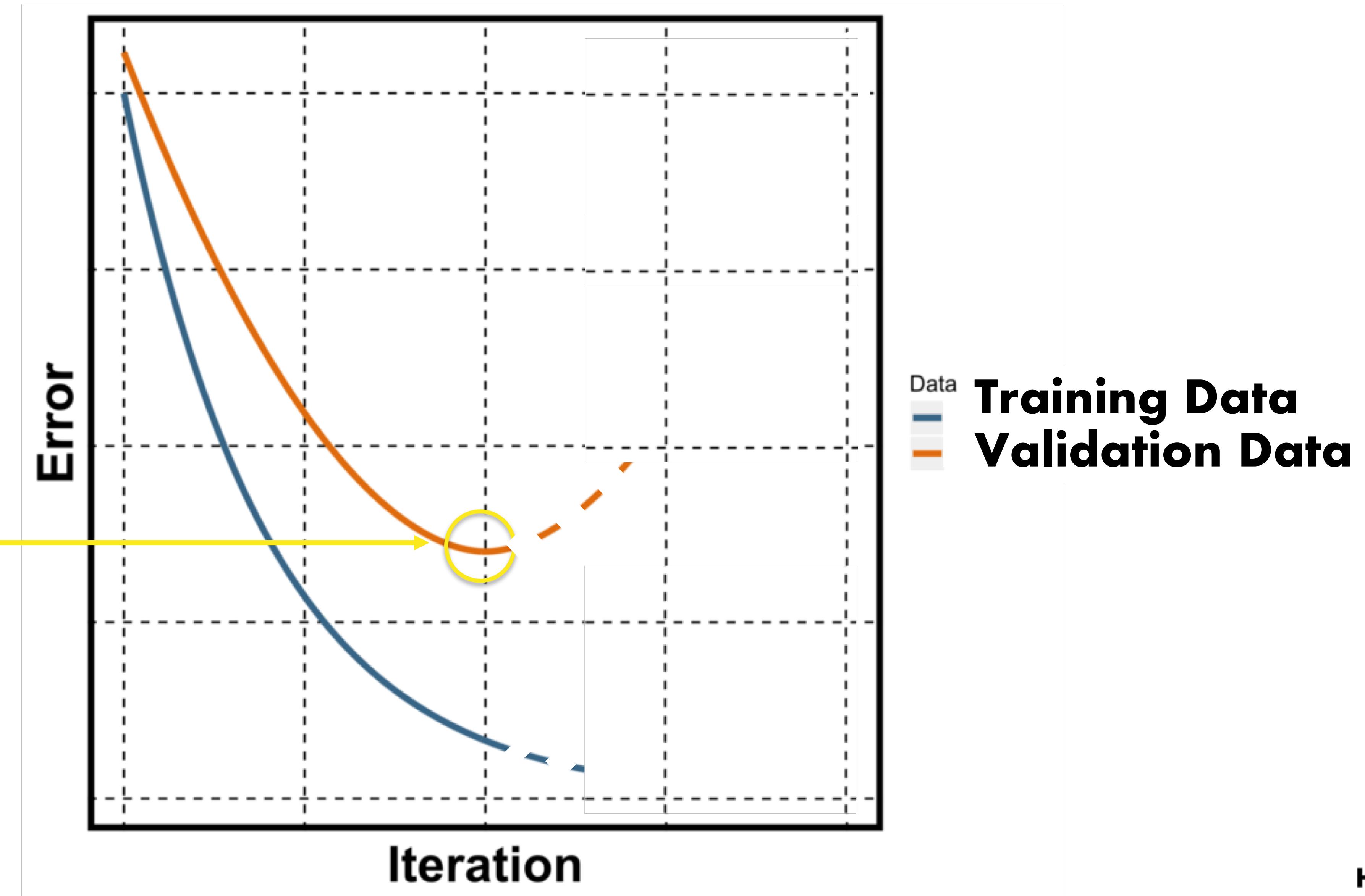


```
1037 void defaultSearchGBM(Key<Grid> gridKey) {
1038     Algo algo = Algo.GBM;
1039     WorkAllocations.Work work = workAllocations.getAllocation(algo, JobType.HyperparamSearch);
1040     if (work == null) return;
1041
1042     GBMParameters gbmParameters = new GBMParameters();
1043     setCommonModelBuilderParams(gbmParameters);
1044     gbmParameters._score_tree_interval = 5;
1045     gbmParameters._histogram_type = SharedTreeParameters.HistogramType.AUTO;
1046
1047     Map<String, Object[]> searchParams = new HashMap<>();
1048     searchParams.put("_ntrees", new Integer[]{10000});
1049     searchParams.put("_max_depth", new Integer[]{3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17});
1050     searchParams.put("_min_rows", new Integer[]{1, 5, 10, 15, 30, 100});
1051     searchParams.put("_learn_rate", new Double[]{0.001, 0.005, 0.008, 0.01, 0.05, 0.08, 0.1, 0.5, 0.8});
1052     searchParams.put("_sample_rate", new Double[]{0.50, 0.60, 0.70, 0.80, 0.90, 1.00});
1053     searchParams.put("_col_sample_rate", new Double[]{0.4, 0.7, 1.0});
1054     searchParams.put("_col_sample_rate_per_tree", new Double[]{0.4, 0.7, 1.0});
1055     searchParams.put("_min_split_improvement", new Double[]{1e-4, 1e-5});
1056
1057     Job<Grid> gbmJob = hyperparameterSearch(gridKey, work, gbmParameters, searchParams);
1058     pollAndUpdateProgress(Stage.ModelTraining, "GBM hyperparameter search", work, this.job(), gbmJob);
1059 }
```

## Random Grid Search

# AutoML: Early Stopping

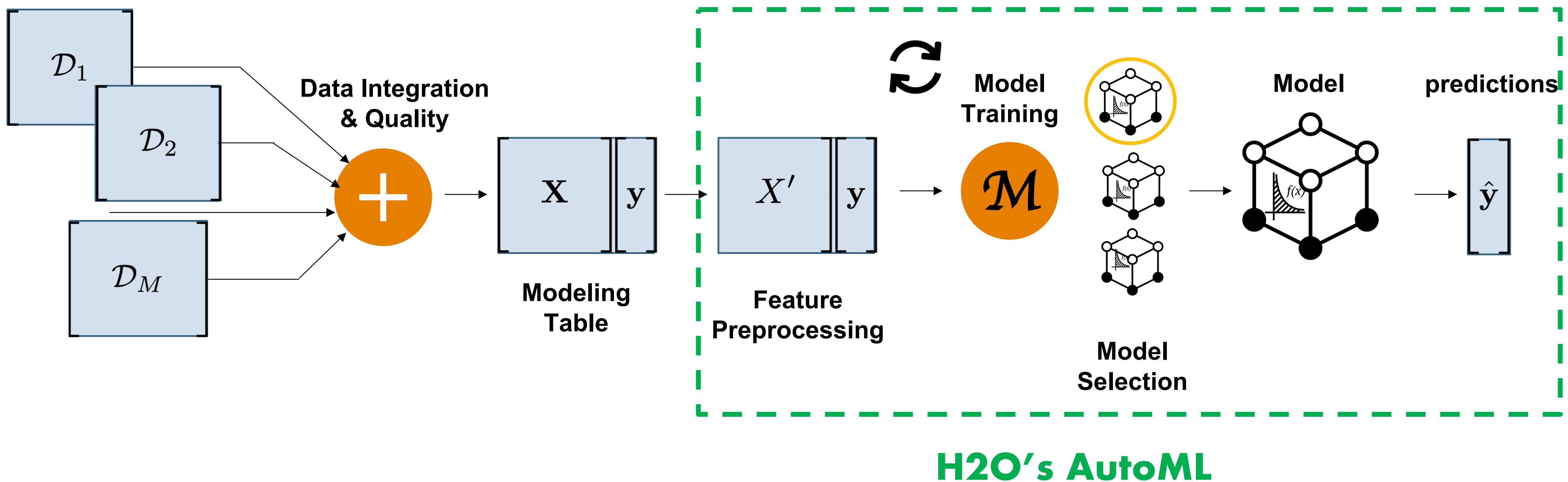
**The  
Sweet  
Spot**



# 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 Machine Learning Pipeline



# The Interface: Web UI

## 2 Required parameters training frame & response

The screenshot shows the H2O FLOW web interface with the title "AutoML in Flow". The main section is titled "Run AutoML" and contains the following configuration fields:

- Project Name: [Input field]
- Training Frame: [Select dropdown] (Select)
- Balance classes: [checkbox]
- Exclude these algorithms:
  - [checkbox] GLM
  - [checkbox] DRF
  - [checkbox] GBM
  - [checkbox] XGBoost
  - [checkbox] DeepLearning
  - [checkbox] StackedEnsemble
- Max models to build: [Input field]
- Max Run Time (sec): 3600
- Early stopping metric: AUTO
- Leaderboard sort metric: AUTO
- Early stopping rounds: 3
- Early stopping tolerance: [Input field]
- nfold: 5
- Keep cross-validation predictions:
- Keep cross-validation models:
- Keep cross-validation fold assignment:
- Seed: -1
- Checkpoints path: [Input field]

# 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

# Thanks!

