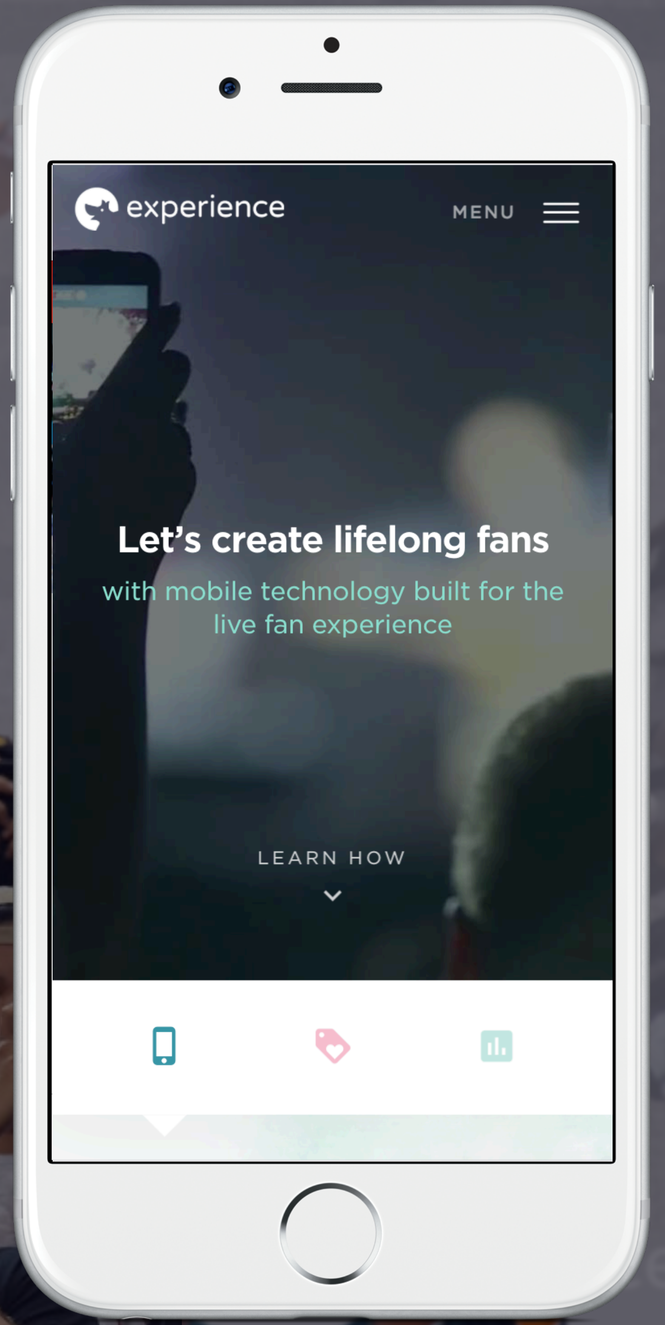
Model Fitting with h2o.ai

Matt Mills



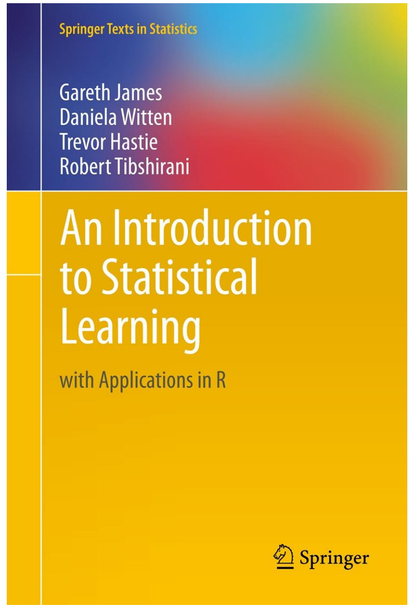
experience

Agenda

- Introduction
- h2o Infrastructure
- h2o R Package
- Algorithms Introduction
- Parameter Tuning
- Extending h2o

Before We Start

- Some basic knowledge of
  - Machine Learning Models (regression vs trees)
  - Model Fitting (test/train, cross validation, parameters)
- If you'd like an introduction download a free copy of Introduction to Statistical Learning
- Will include all code used during the live demos if you need a reference later



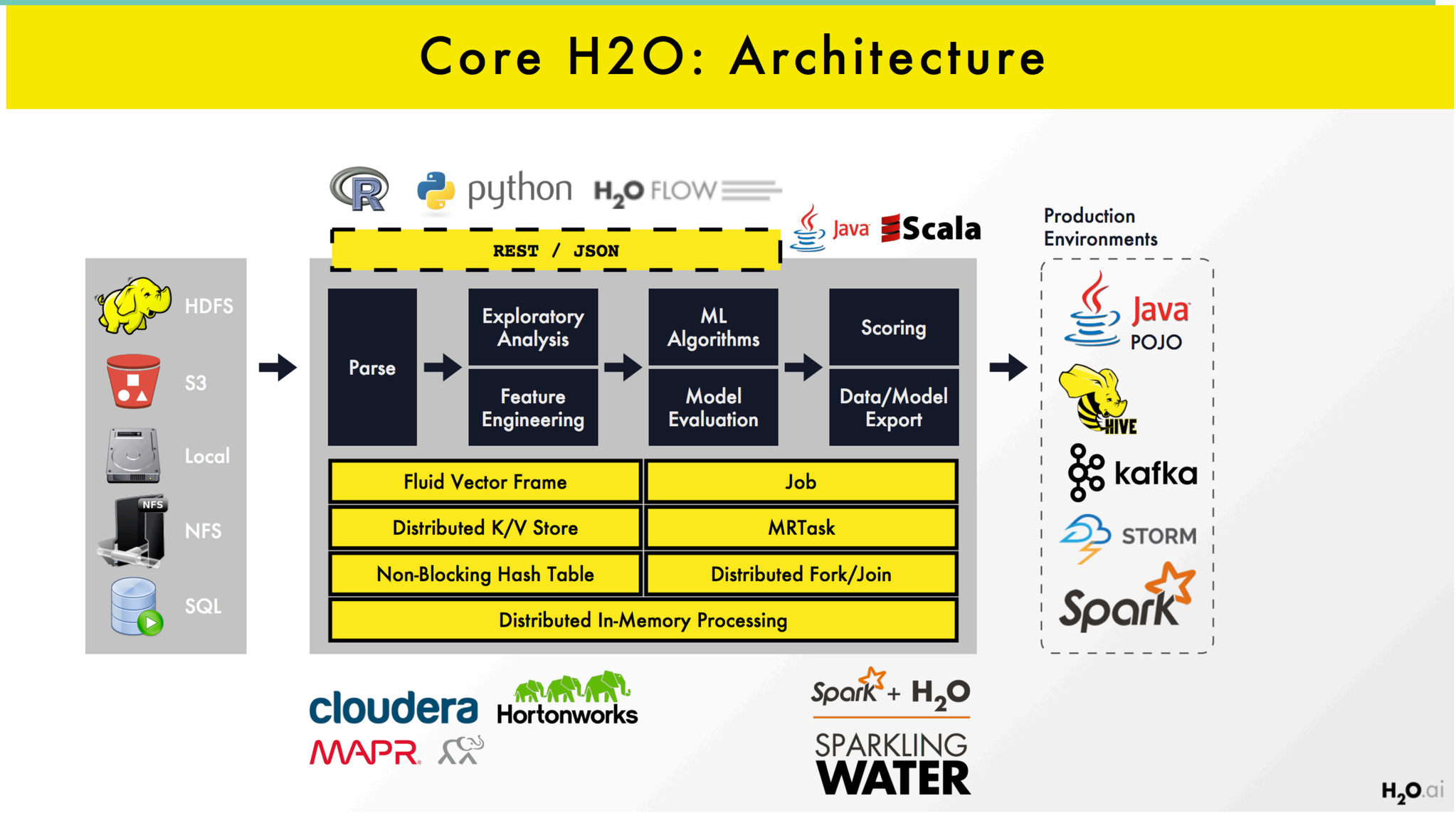
What is h2o.ai?

- Silicon Valley startup founded in 2011
- Machine learning algorithms designed to be distributed and parallel out of the box
- Science advisors include Stephen Boyd, Robert Tibshirani, and Trevor Hastie
- Provide software for free and charges clients for support, deployment, and custom features
  - Comcast, Progressive, Capital One are all clients

Agenda

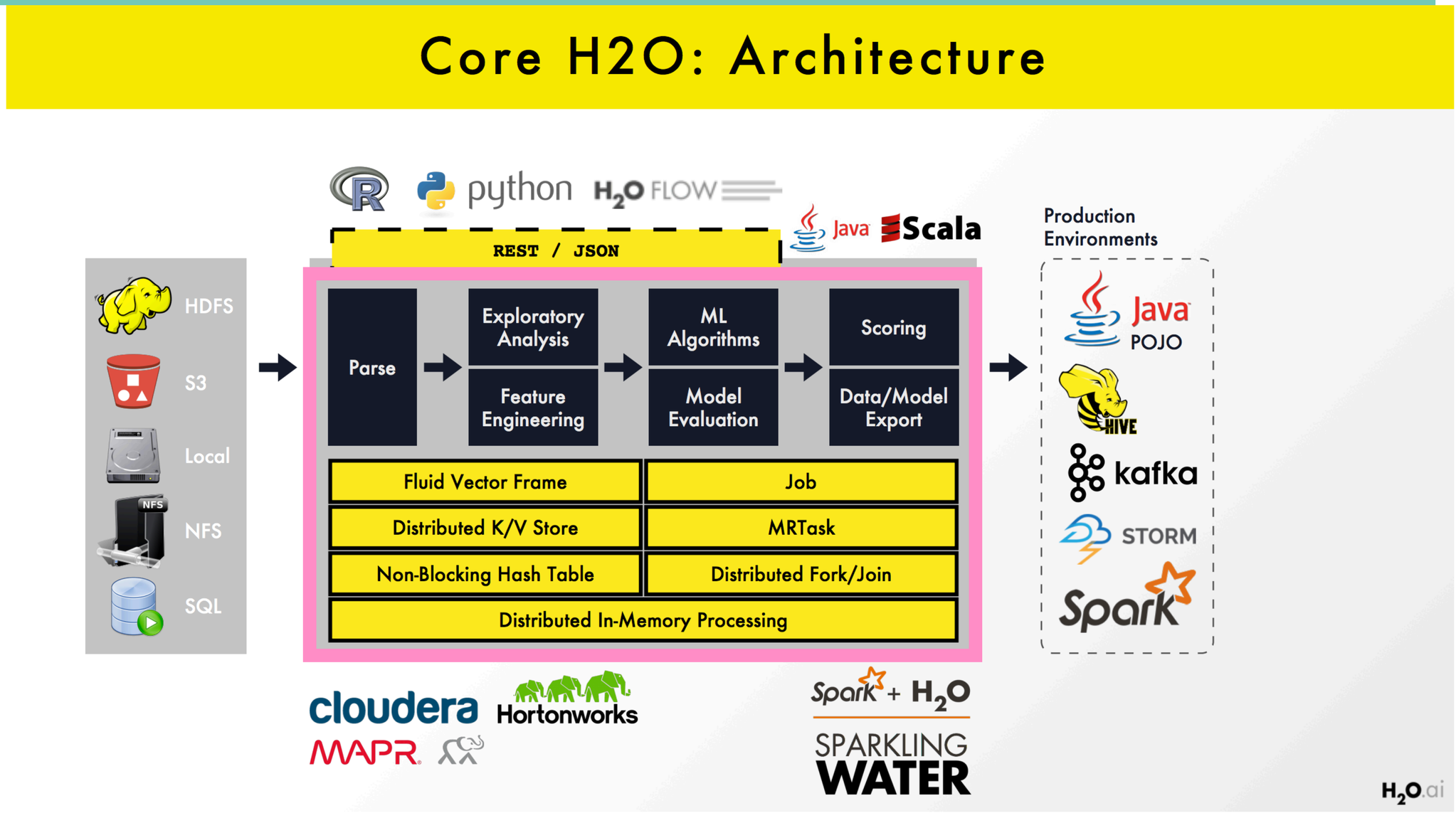
- Introduction
- **h2o Infrastructure**
- h2o R Package
- Algorithms Introduction
- Parameter Tuning
- Extending h2o

h2o Architecture



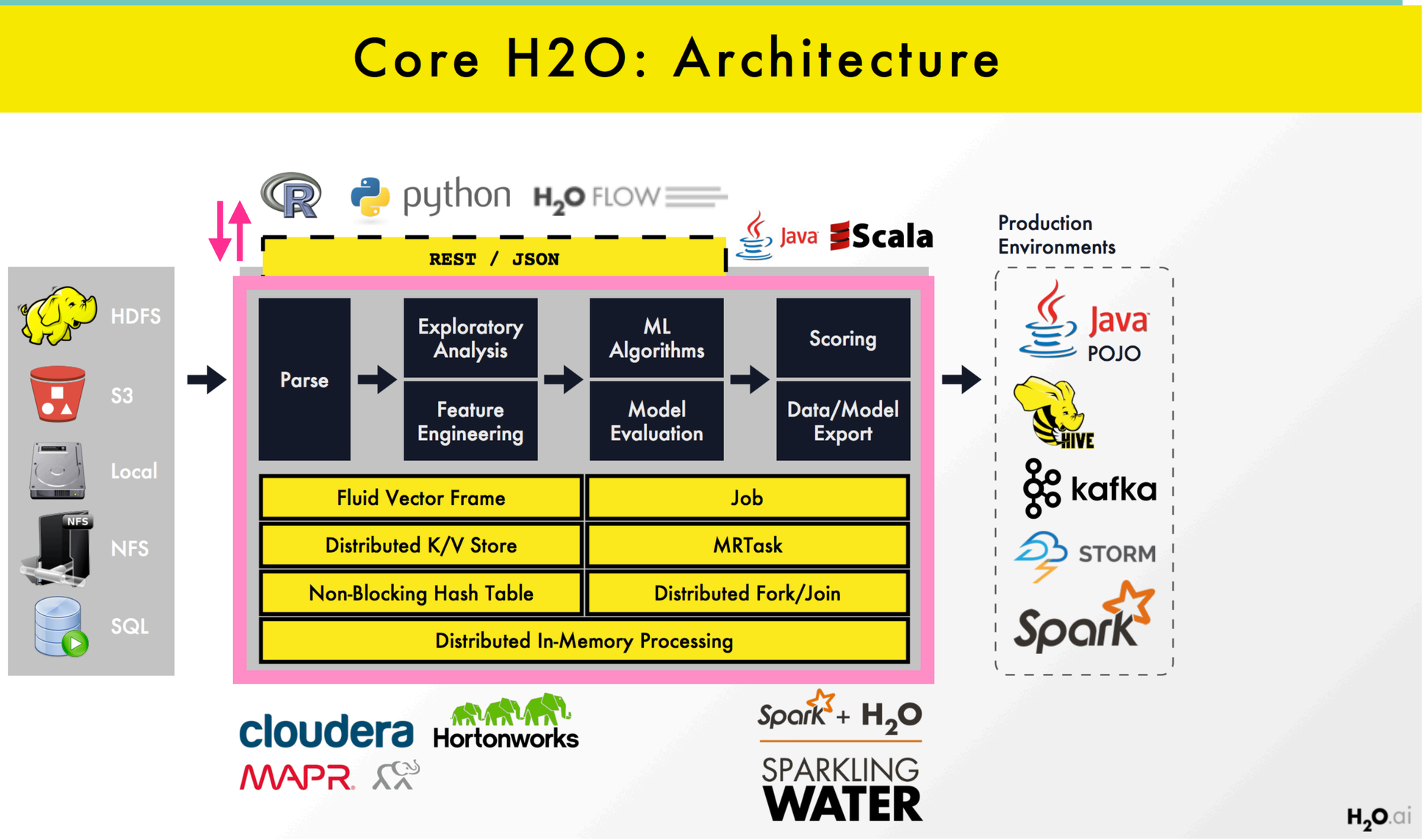
Source: https://github.com/h2oai/h2o-meetups/tree/master/2017_02_22_Seattle_STC_Meetup

h2o Architecture



Source: https://github.com/h2oai/h2o-meetups/tree/master/2017_02_22_Seattle_STC_Meetup

h2o Architecture



Source: https://github.com/h2oai/h2o-meetups/tree/master/2017_02_22_Seattle_STC_Meetup

Documentation

- **Extensive documentation at docs.h2o.ai**
- **User and Developer Guides for each language (R, Python, Scala, Java)**
- **Algorithm references and tutorials**
- **Use cases/examples, past presentations, videos**
- **All code is on GitHub: https://github.com/h2oai**

Agenda

- Introduction
- h2o Infrastructure
- **h2o R Package**
- Algorithms Introduction
- Parameter Tuning
- Extending h2o

Using the h2o R Package

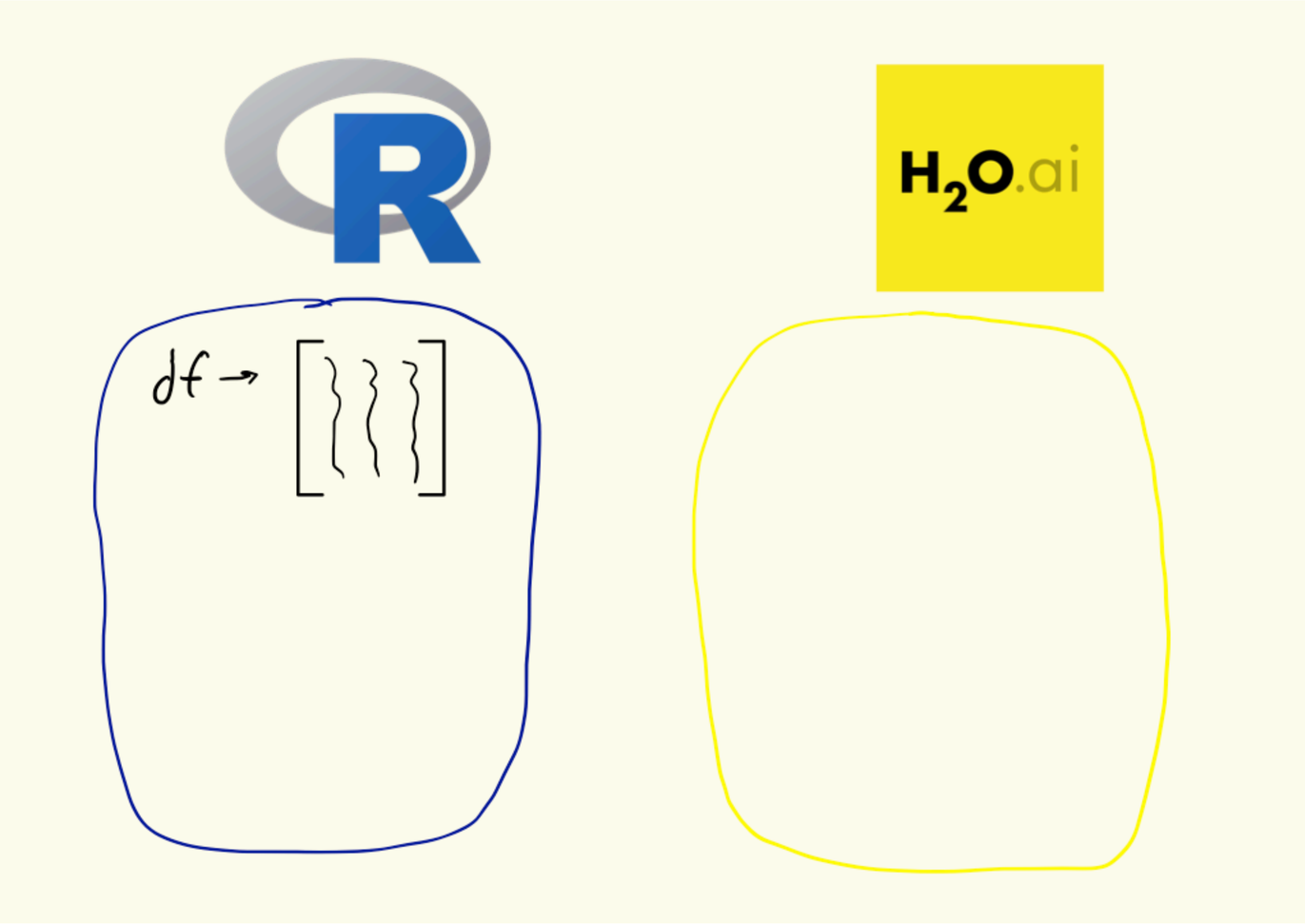
- 2 “h2o”s to run
  - h2o R package
  - h2o server (can be local)
- Use R to tell the server what to do

```
library(h2o)
h2o.init(nthreads = 4, max_mem_size = "4g", min_mem_size = "1g")
```

Process Name	Memory	Compressed M...	Threads
kernel_task	1.43 GB	0 bytes	178
Google Chrome Helper	786.1 MB	110.7 MB	19
Google Chrome Helper	643.7 MB	64.3 MB	21
WindowServer	493.9 MB	161.7 MB	10
Spotify Helper	364.8 MB	0 bytes	17
rsession	348.5 MB	69.4 MB	7
RStudio	329.2 MB	79.8 MB	12
java	310.1 MB	0 bytes	36

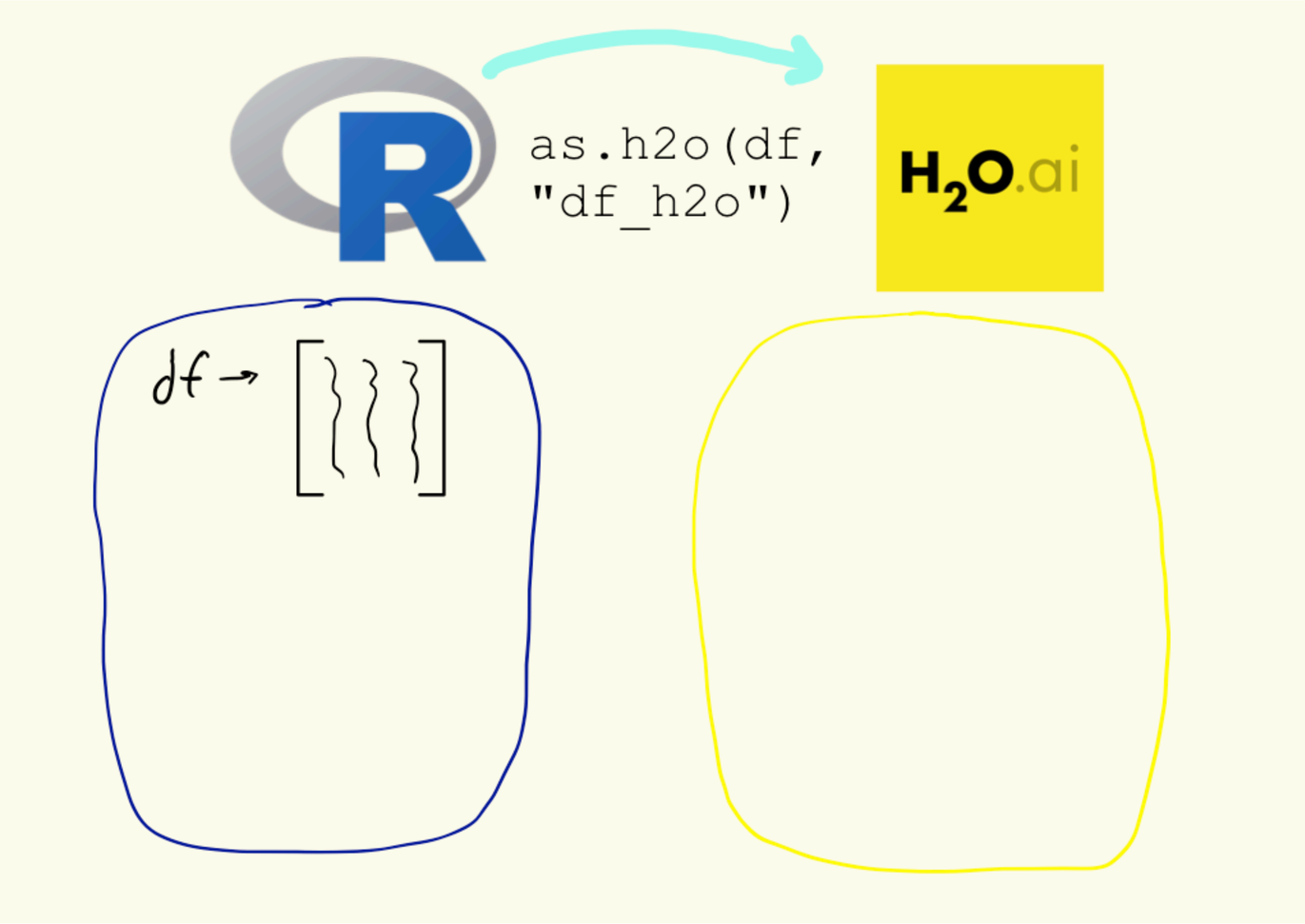
Using the h2o R Package

- **R uses APIs to communicate with h2o**



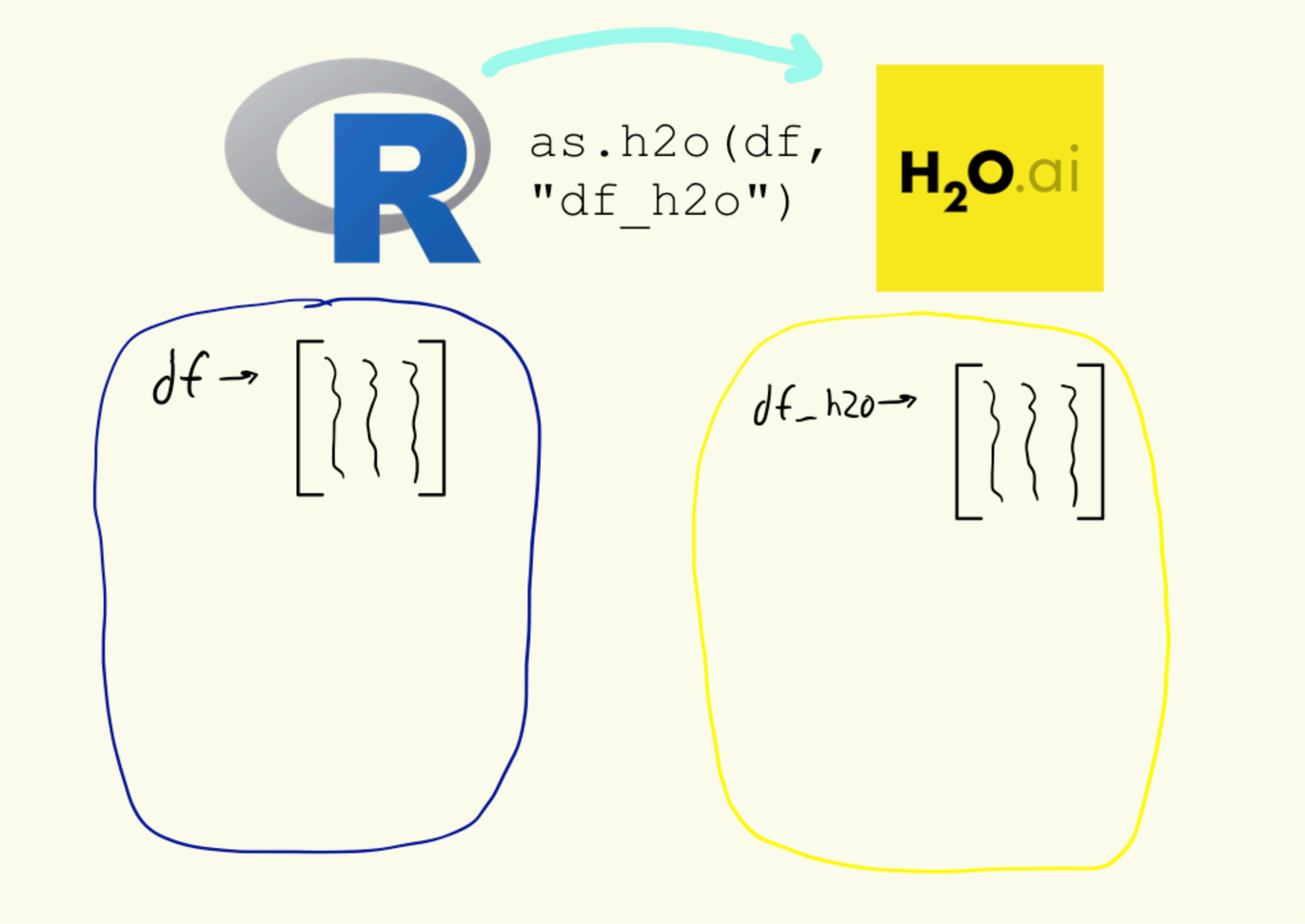
Using the h2o R Package

- **R uses APIs to communicate with h2o**



Using the h2o R Package

- **R uses APIs to communicate with h2o**



Using the h2o R Package

- **R uses APIs to communicate with h2o**

```
> df <- iris
> df_h2o <- as.h2o(df, "df_h2o")
  |======================================================================| 100%
> class(df_h2o)
[1] "H2OFrame"
> str(df_h2o)
Class 'H2OFrame' <environment: 0x128d8be60>
 - attr(*, "op")= chr "Parse"
 - attr(*, "id")= chr "df_h2o"
 - attr(*, "eval")= logi FALSE
 - attr(*, "nrow")= int 150
 - attr(*, "ncol")= int 5
 - attr(*, "types")=List of 5
  ..$ : chr "real"
  ..$ : chr "real"
  ..$ : chr "real"
  ..$ : chr "real"
  ..$ : chr "enum"
 - attr(*, "data")='data.frame':        10 obs. of  5 variables:
  ..$ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9
  ..$ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1
  ..$ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5
  ..$ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1
  ..$ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1
>
```

Some Housekeeping Before we Start

- **If your h2o cluster version doesn't align with your R and h2o package version you may see some weird behavior**
- **All character variables *must* be converted to factors**
- **Can use `h2o.importFile` instead of loading data from R**

Agenda

- Introduction
- h2o Infrastructure
- h2o R Package
- **Algorithms Introduction**
- Parameter Tuning
- Extending h2o

h2o Algorithm List

- Generalized Linear Models
- Gradient Boosting Machine
- Random Forest
- Naive Bayes
- Ensembles
- XGBoost
- Deep Learning
- K-Means
- PCA
- SVD
- Generalized Low Rank Models (GLM + PCA)

Generalized Linear Models: `h2o.glm`

- **Replacement for `glm` in base R**
  - **Doesn't use formula (`y ~ x`) interface**
- **Gaussian, Binomial, Multinomial, Quasibinomial, Poisson, Gamma, Tweedie, and Ordinal distributions**
- ***Defaults to Ridge Regression***
- **Can do sampling, balancing, weights, offsets**

Generalized Linear Models Continued

- **Returns an `H2OModel S4` object with basically everything you'd want**
  - **Fit Parameters**
  - **Coefficient Values**
  - **Training and Validation Metrics**
  - **Scoring Iterations**
- **Warning: `summary` prints *a lot***
- **Easiest way to get metrics is to write helper functions to return tidy results**

Generalized Linear Models Reference

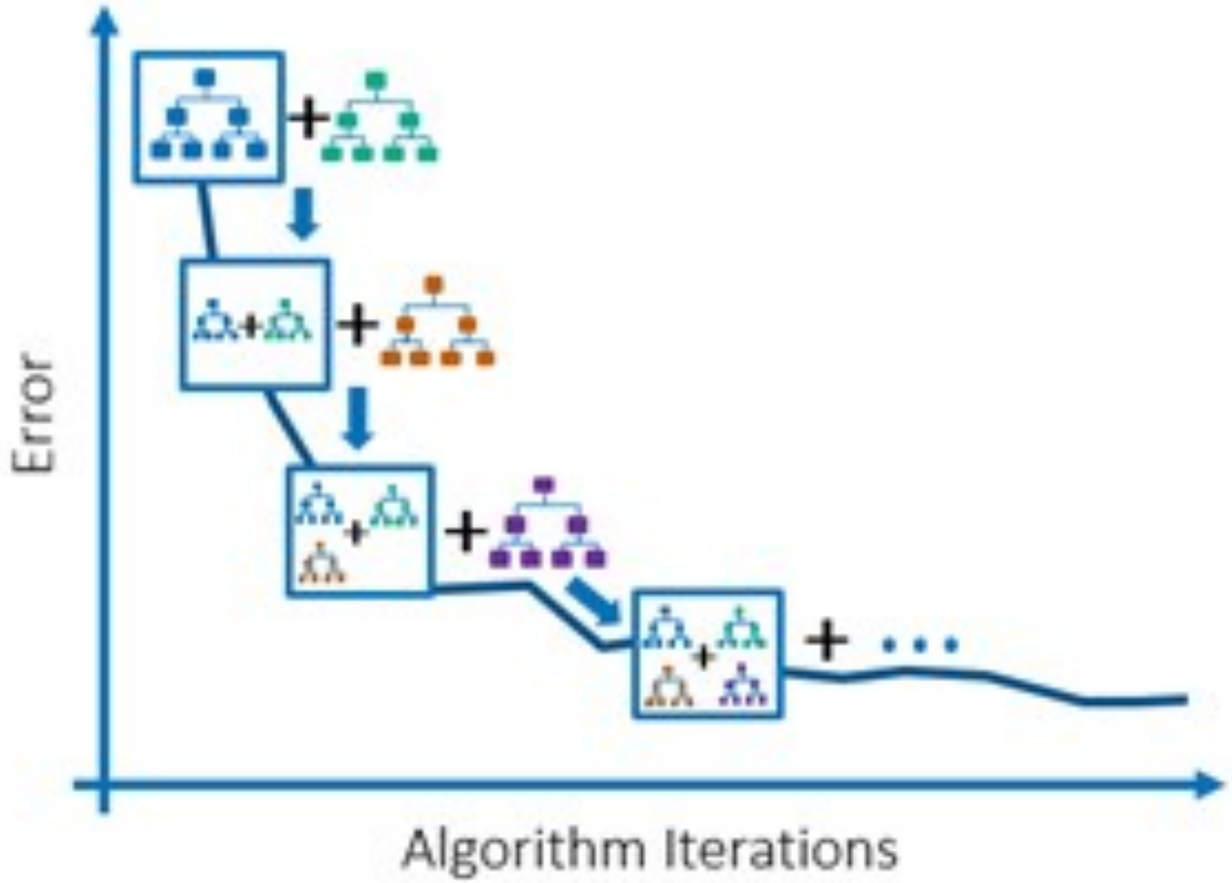
```
loan_h2o <- as.h2o(clean_loan_data, destination_frame = "loan_h2o")

default_glm_plain <- h2o.glm(x = 1:46,
                             y = "bad_loan",
                             training_frame = "loan_h2o",
                             family = "binomial",
                             lambda = 0) # must do this to avoid regularized regression

class(default_glm_plain)
str(default_glm_plain@parameters)
default_glm_plain@model$training_metrics@metrics[c("MSE", "RMSE", "AUC", "logloss", "Gini")]

default_glm_lasso <- h2o.glm(x = 1:46,
                             y = "bad_loan",
                             training_frame = "loan_h2o",
                             model_id = "default_glm_lasso",
                             nfolds = 4,
                             family = "binomial",
                             lambda_search = T,
                             nlambdas = 50,
                             alpha = 1) # 1 is lasso, 0 is ridge
```

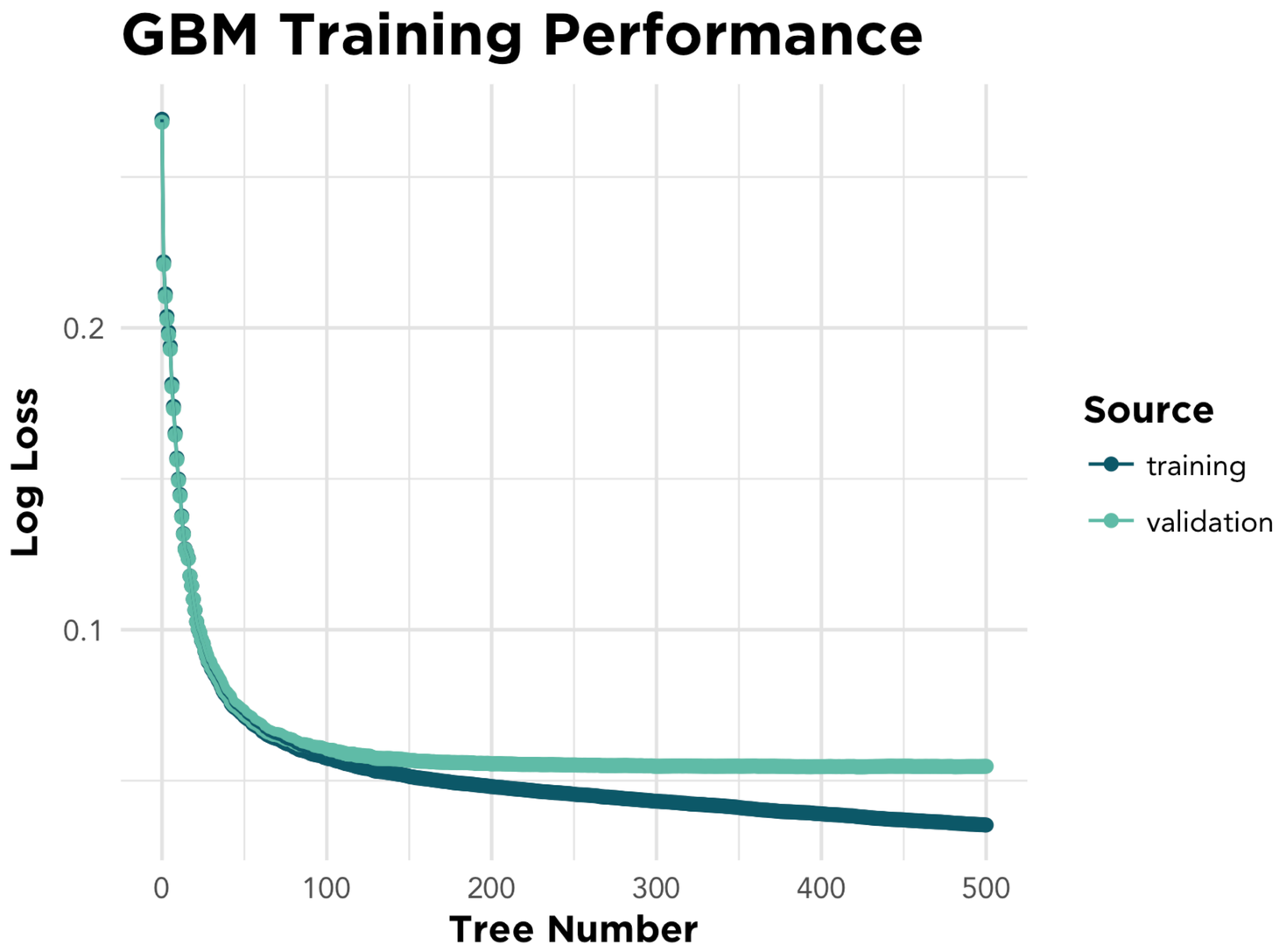
Gradient Boosting Machine Intro



Gradient Boosting Machine: `h2o.gbm`

- **Same Modeling API as `h2o.glm`**
- **Extra distributions like Laplace, Quantile, and Huber**
- **Tons of parameters (`?h2o.gbm` for full list)**
  - `ntrees`
  - `max_depth` **(max number of splits)**
  - `min_rows` **(min obs in a terminal node)**
  - `learn_rate`
  - `sample_rate` **(% of obs to sample for each tree)**
  - `col_sample_rate_per_tree` **(% of variables to sample *per tree*)**
  - `col_sample_rate` **(% of variables to sample *per split*)**
  - `categorical_encoding`

GBM Reference



GBM Reference

```
train <- runif(n = nrow(loan_h2o)) < .75
train_rows <- seq_len(nrow(loan_h2o))[train]
test_rows <- seq_len(nrow(loan_h2o))[!train]
loan_h2o_train <- loan_h2o[train_rows, ]
loan_h2o_test <- loan_h2o[test_rows, ]

h2o.getId(loan_h2o_test)

default_gbm <- h2o.gbm(x = 1:46,
                       y = "bad_loan",
                       training_frame = loan_h2o_train, # can use R reference
                       validation_frame = "RTMP_sid_9cfc_4", # or h2o ID
                       score_each_iteration = T,
                       distribution = "bernoulli",
                       ntrees = 500,
                       max_depth = 3,
                       learn_rate = .1,
                       sample_rate = 2/3,
                       col_sample_rate_per_tree = 2/3,
                       categorical_encoding = "Enum")

training_accuracy <- default_gbm@model$scoring_history
training_accuracy <- h2o.scoreHistory(default_gbm)
```

Agenda

- Introduction
- h2o Infrastructure
- h2o R Package
- Algorithms Introduction
- **Parameter Tuning**
- Extending h2o

Data Splits

- **Instead of splitting manually can also split using** `h2o.splitFrame`
- **Returns a list of** `H2OFrames`

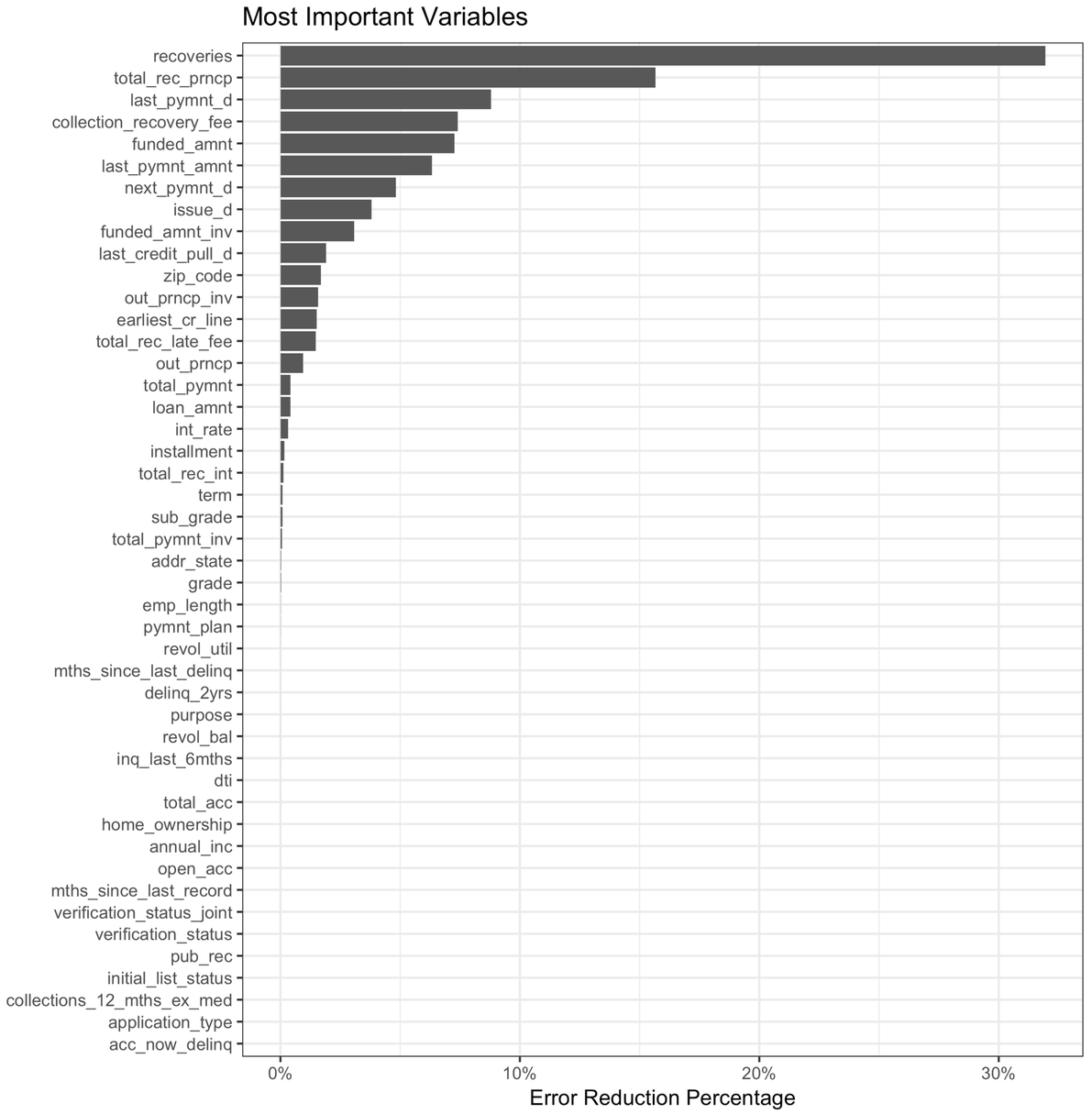
Grid Search for Hyper-Parameter Tuning

- `nfolds` **argument will preform cross validation**
- **call using** `h2o.grid` **to pass in a list of hyper-parameters to test**
- **returns an** `H2OGrid` **object with a list of table of individual model_ids and performance information**

Variable Importance

- **Variable Importance measures the total error reduction when each variable is used in a split**
- **Use `h2o.varimp` or can access it directly from model**

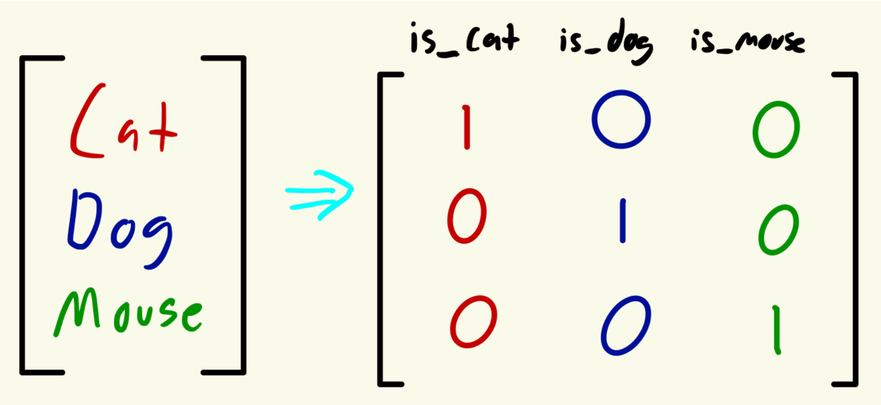
Variable Importance / Partial Plots



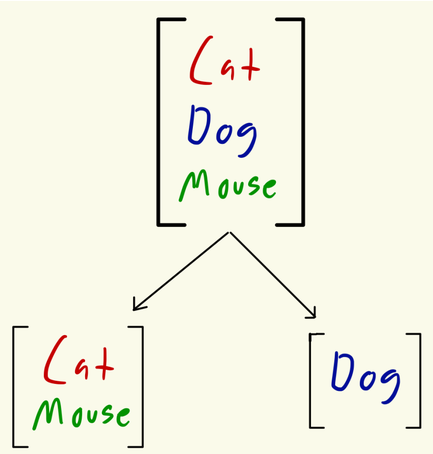
experience

h2o Categorical Variable Handling

- **ML Standard is to use one-hot encoding**



- **h2o uses *bitset encoding* which allows for perfect splits**



Good Sources: https://groups.google.com/forum/#!topic/h2ostream/VChVUAJPjEc
https://aichamp.wordpress.com/2017/03/09/treatment-of-categorical-variables-in-h2os-drf-algorithm/

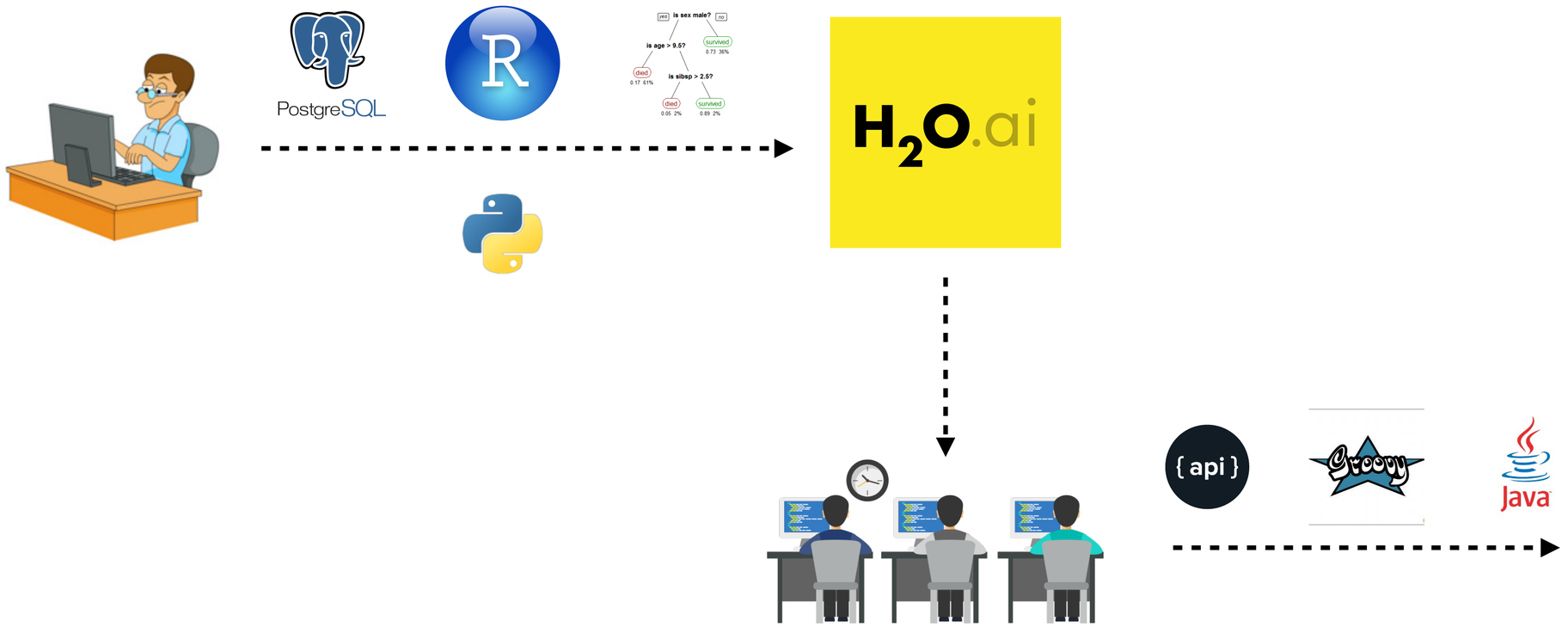
Agenda

- Introduction
- h2o Infrastructure
- h2o R Package
- Algorithms Introduction
- Parameter Tuning
- **Extending h2o**

Saving h2o Models

- Each h2o model can be saved as a Plain Old Java Object (POJO)
  - Every model parameter is included in this file
- Can be embedded into any application running on the JVM

Using h2o in Production



Using h2o in Production Continued

1. App Sends Data

4. App Gets Prediction

input()

{JSON}

2. Data Cleaning in Python

3. Predictions done in h2o

sys.stdout.flush()

Full Pres: http://bit.ly/jan18pydata

experience

Conclusion

- **h2o can speed up modeling and scale if necessary**
- **Full suite of configurable algorithms**
- **Plenty of information to perform model diagnostics**

www.statmills.com

http://docs.h2o.ai/

https://github.com/mattmills49/AtlantaRUsers

Title Text

Text