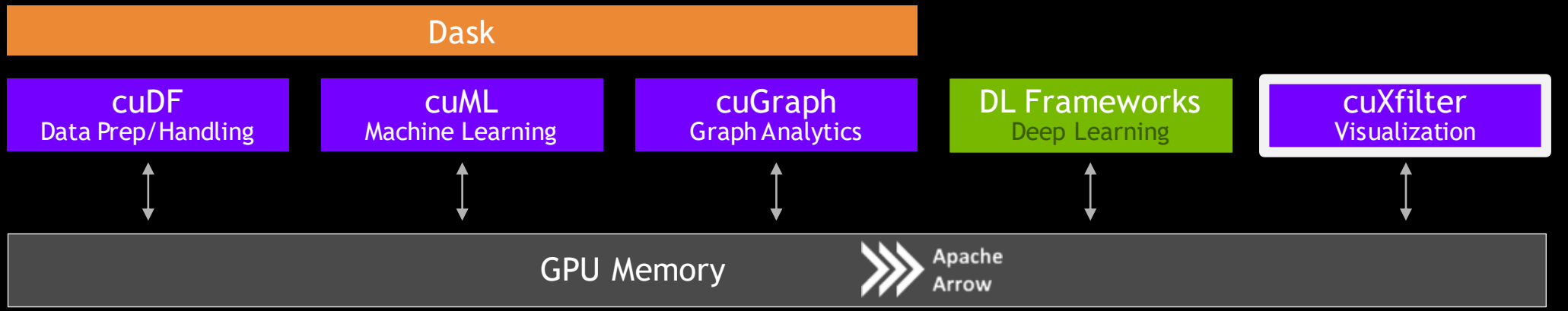
SECTION 2
01 - 07

RAPIDS PLATFORM



CUXFILTER

(coo-cross-filter)

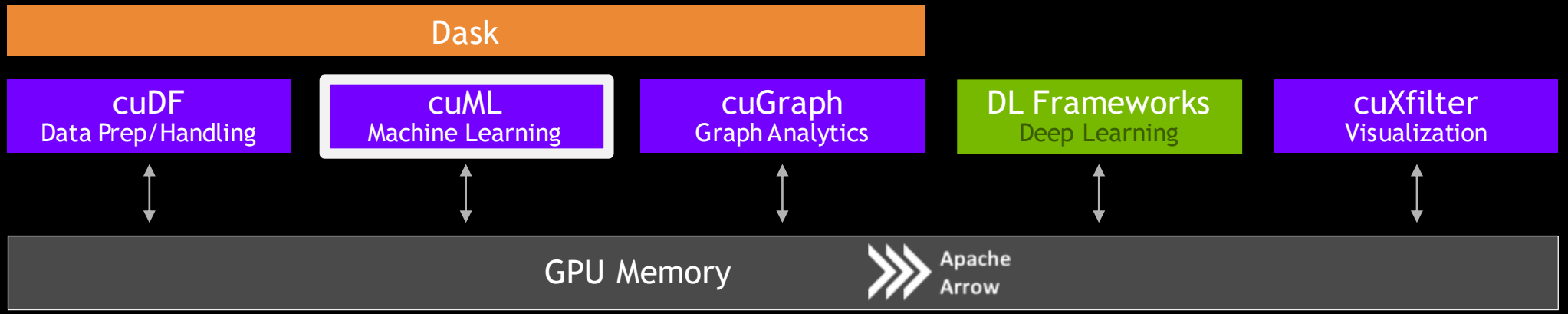
Efficient GPU-backed in-browser visualization engine

Built on a host of pyViz ecosystem tools

Bokeh and Datashader charts

Multiple charts with cross-filtering widgets

RAPIDS PLATFORM



CUML

Follows scikit-learn convention of model objects with `.fit` and `.predict/.transform`

Rapidly-growing subset of algorithms, driven by use cases

In today's workshop

- K-means (single- and multi-GPU)
- DBSCAN
- Logistic regression
- K-nearest neighbors
- XGBoost

K-MEANS

- Partitions data by iteratively moving cluster centers and reassigning datapoints based on which center is closest to the point
- Requires a known (or well-estimated) number of clusters
- Fast, simple, easy to understand
- Has an efficient multi-node, multi-GPU implementation

DBSCAN

- Uses spatial density to cluster—finds cluster points within at most *epsilon* distance of another point in the cluster
- Can identify outliers—not every point is “reachable” from a cluster core
- Does not require prior knowledge of the number of clusters, but does require an estimate for epsilon
- Can identify clusters of unusual shapes

K-NEAREST NEIGHBORS

- Enables rapid discovery of nearby, existing points to a new observation
- Algorithm fits a data structure for future use
- Foundation of other algorithms that require knowing which points are nearby a given point

LOGISTIC REGRESSION

- Regression with binary dependent variables
- Uses the logistic function to map (-inf, +inf) domain scores to (0, 1) range
- No simple coefficient formulas, as with ordinary least squares (OLS) linear regression
- Assumes independence of features

TRY NOTEBOOKS 01 - 07 NOW

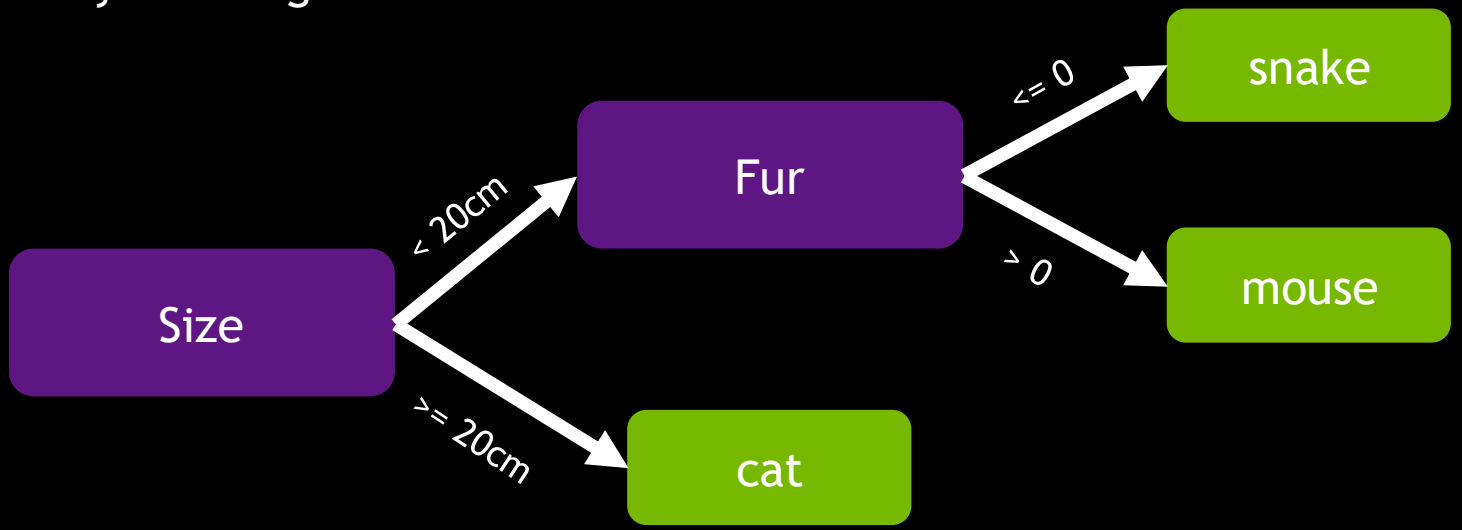
docs.rapids.ai/api

SECTION 2
08

XGBOOST

Popular and powerful general-purpose classification and regression algorithm for structured data—can identify and use complex patterns

Gradient-boosted decision trees—optimizing against an objective function while controlling tree complexity with regularization



TRY NOTEBOOK 08 NOW

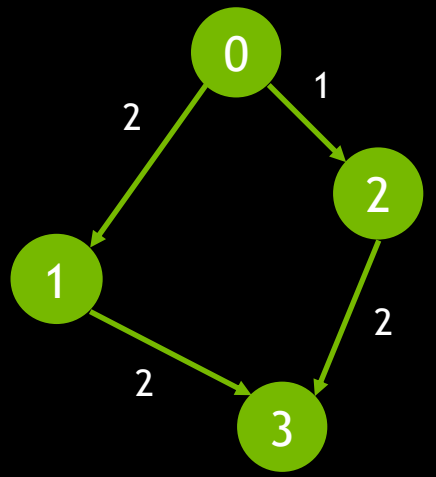
docs.rapids.ai/api

SECTION 2
09

SINGLE-SOURCE SHORTEST PATH

Input: Graph (w/o negative-weight cycles), node_id

Output: dataframe with Vertex, Distance, Predecessor columns



cg.sssp(G, 0)

Vertex	Distance	Predecessor
0	0	-1
1	2	0
2	1	0
3	3	2

Note: using mapped integer IDs in exercise for memory and speed

TRY NOTEBOOK 09 NOW

docs.rapids.ai/api