ANACONDACON

Scalable Machine Learning with Dask

Tom Augspurger
Data Scientist

Hi! I'm Tom

- I work at Anaconda on dask, pandas, & ML things

Machine Learning Workflow

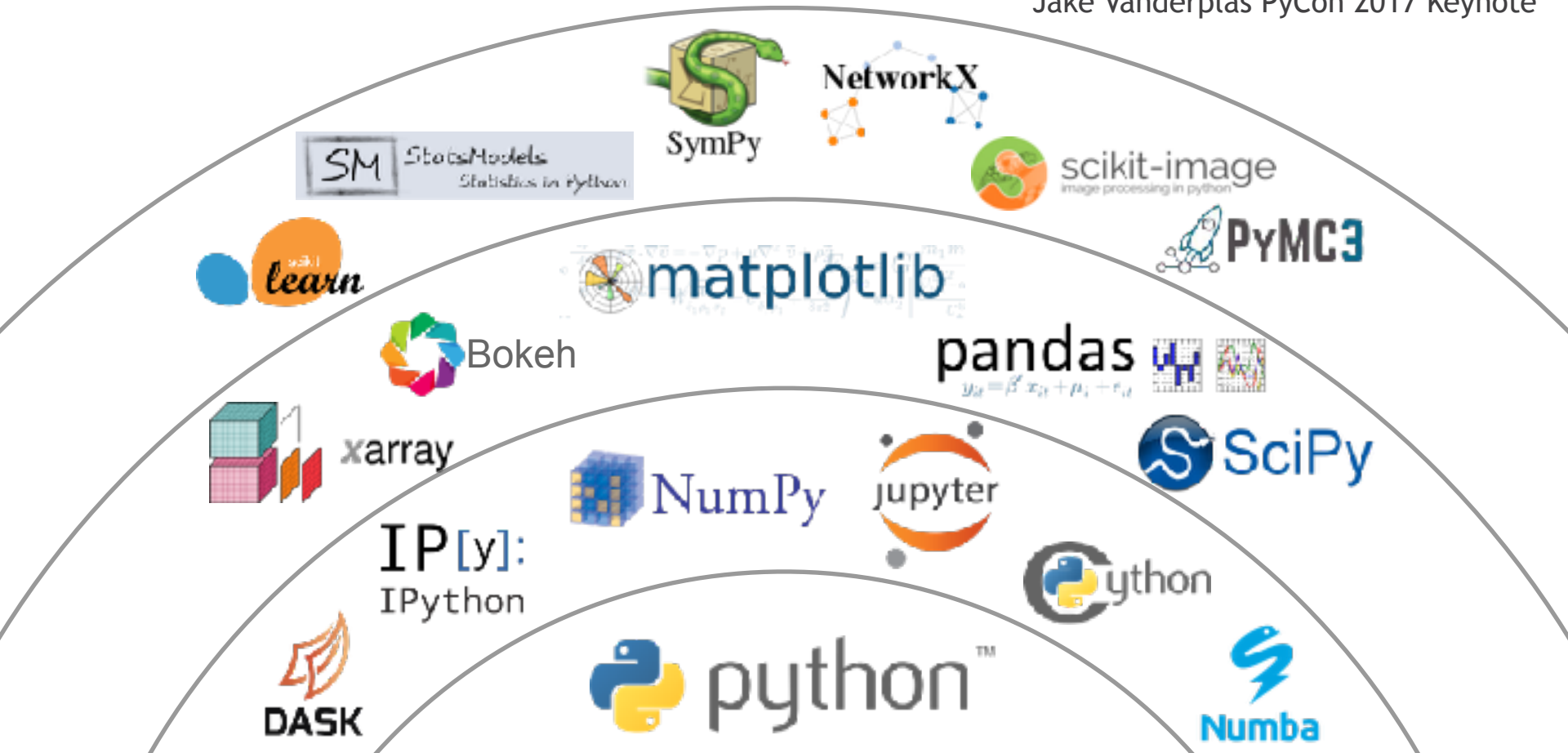
It's not just `.fit(X, y)`

Machine Learning Workflow

- Understanding the problem, objectives
- Reading from data sources
- Exploratory analysis
- Data cleaning
- Modeling
- Deployment and reporting

The Numeric Python Ecosystem

Jake Vanderplas PyCon 2017 Keynote



Machine Learning Workflow

A rich ecosystem of tools

But they don't scale well

Dask

Parallelizing the Numeric Python Ecosystem

Dask

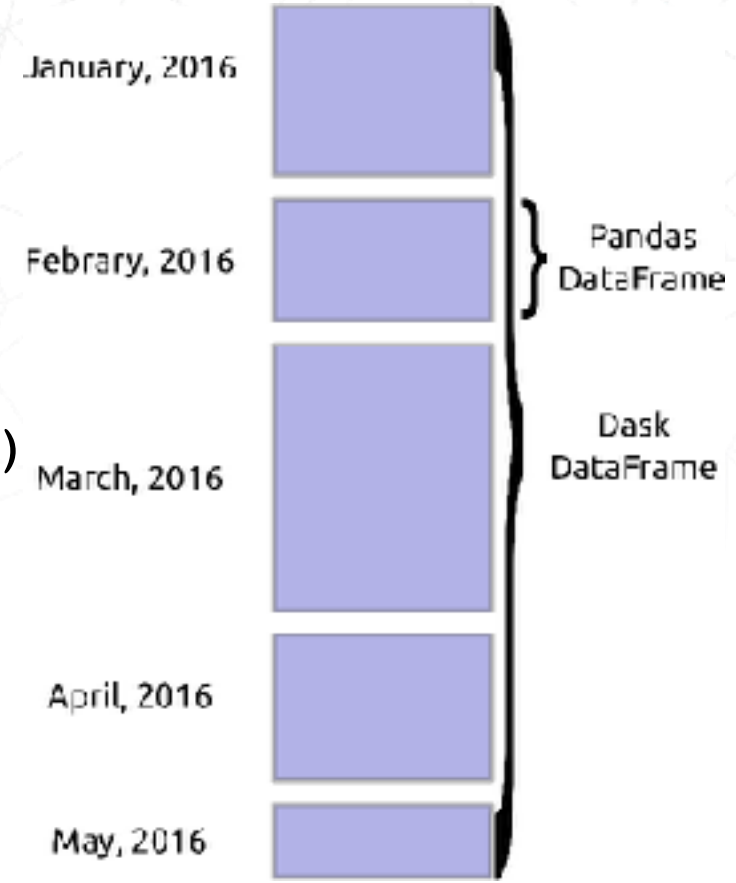
High Level: Parallel Pandas, NumPy, Scikit-Learn

Low Level: High performance task scheduling

Dask

High-Level: Scalable Pandas DataFrames

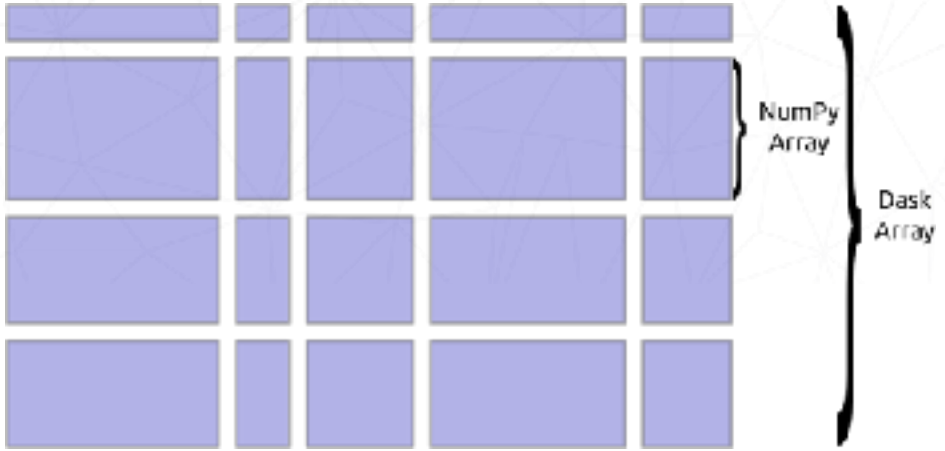
```
import dask.dataframe as dd

df = dd.read_csv('s3://abc/*.csv')
df.groupby(df.name).value.mean()
```



Dask

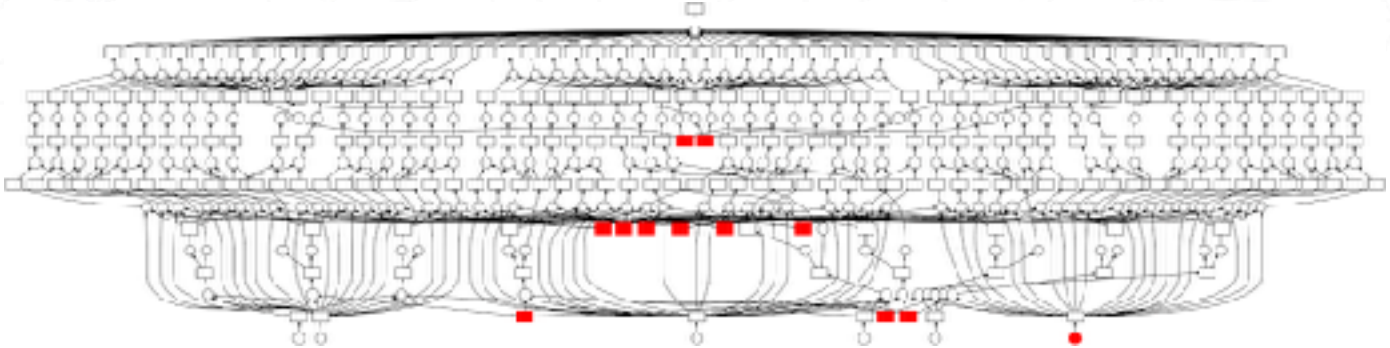
High-Level: Scalable NumPy Arrays

```
import dask.array as da

x = da.random.random(…)
y = x.dot(x.T) - x.mean(axis=0)
```



Dask

Low-Level: Scalable, Fine-Grained Task Scheduling



Demo

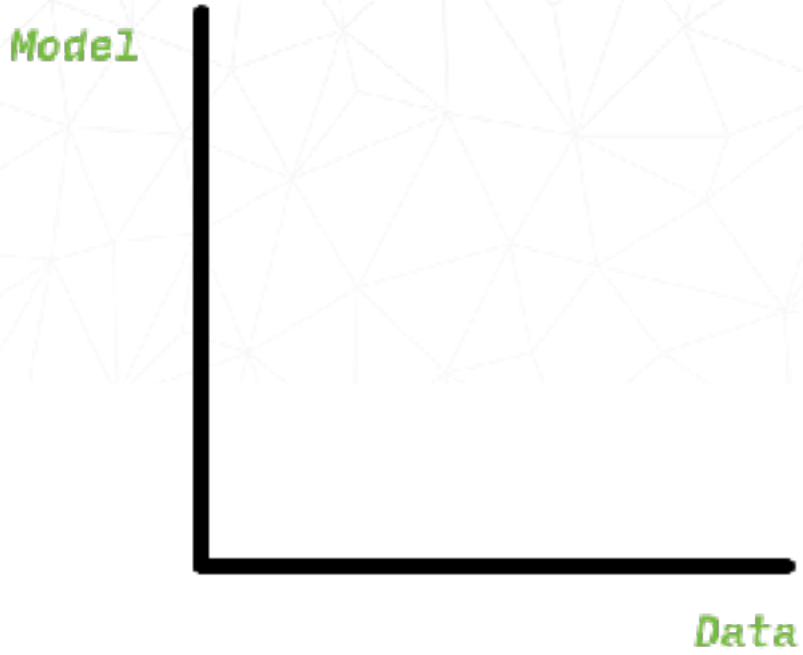
Dask

- **Parallelizes libraries** like NumPy, Pandas, and Scikit-Learn
- **Adapts** to custom algorithms with a flexible task scheduler
- **Scales** from a laptop to thousands of computers
- **Integrates easily,** Pure Python built from standard technology

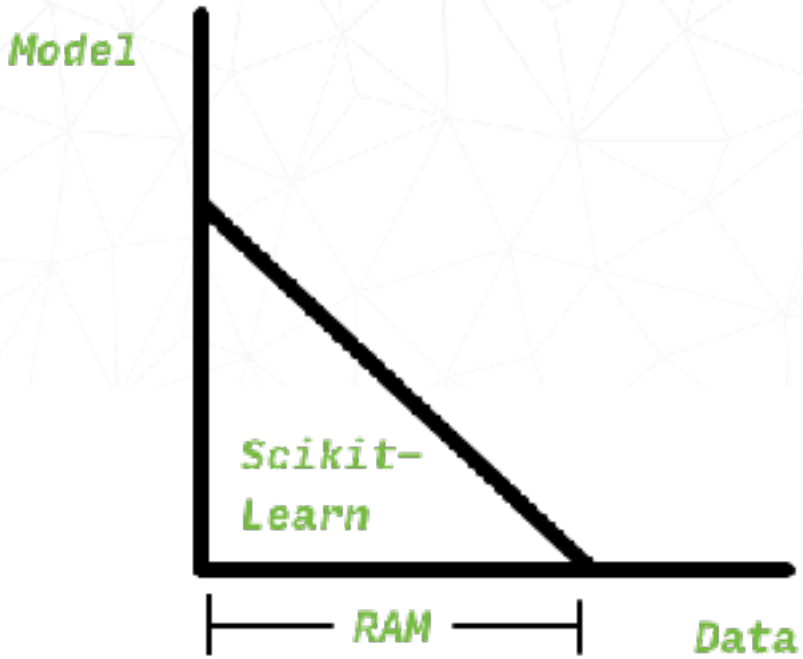
Scalable Machine Learning

Dask-ML

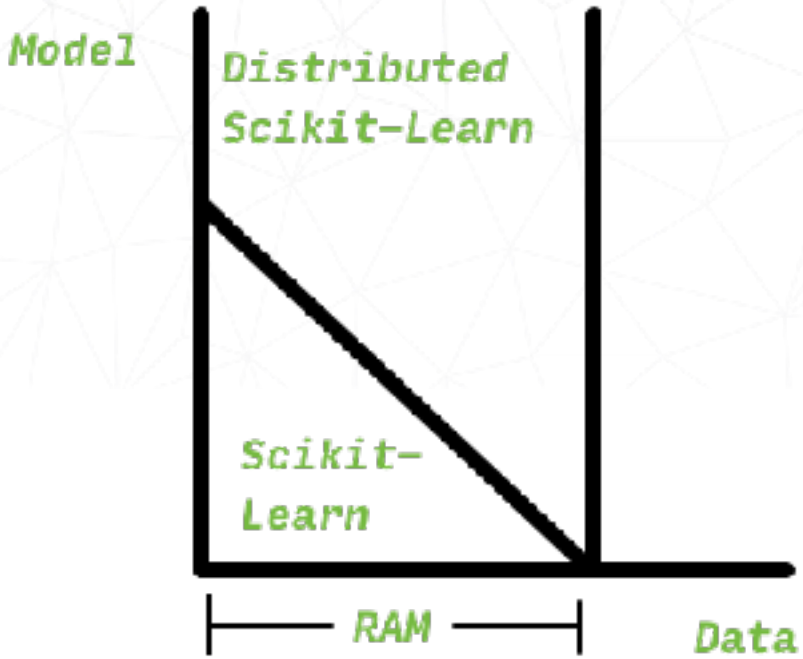
Scaling Pains



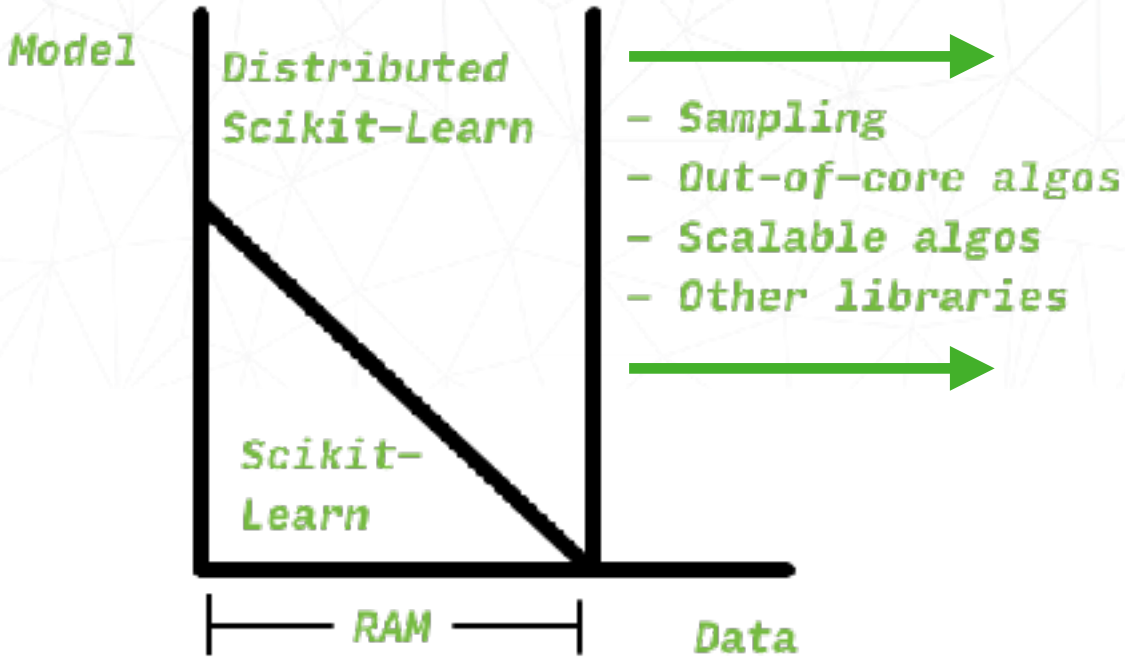
Scaling Pains



Scaling Pains



Scaling Pains

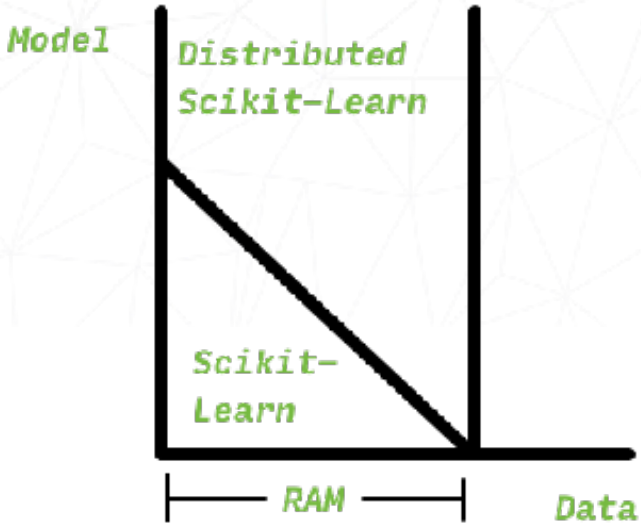


Distributed Scikit-Learn

Distributed Scikit-Learn

Large models, smaller datasets

- Use dask to distribute computation on a cluster



Distributed Scikit-Learn

Single-Machine Parallelism with Scikit-Learn


```
from sklearn.ensemble import RandomForestClassifier


clf = RandomForestClassifier(n_estimators=200, n_jobs=-1)


clf.fit(X, y)
```

Distributed Scikit-Learn

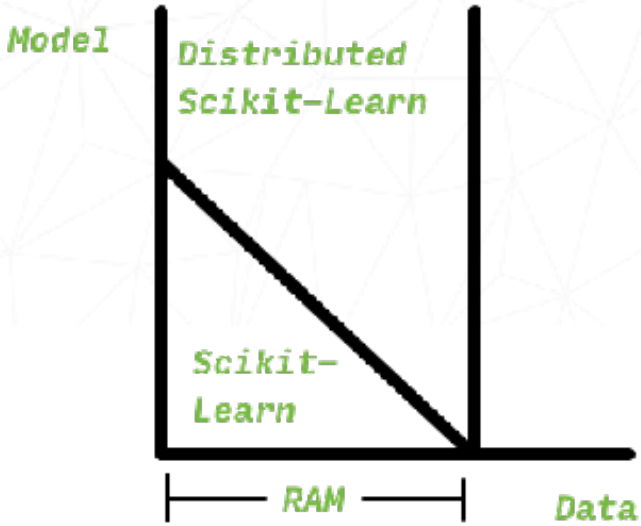
Multi-Machine Parallelism with Dask


```
from sklearn.ensemble import RandomForestClassifier
from sklearn.externals import joblib
import dask_ml.joblib

clf = RandomForestClassifier(n_estimators=200, n_jobs=-1)

with joblib.parallel_backend("dask", scatter=[X, y]):
    clf.fit(X, y)
```

Distributed Scikit-Learn

Caveats

- Data has to fit in RAM
- Data shipped to each worker
  - Each parallel task should be expensive
  - There should be many parallel tasks

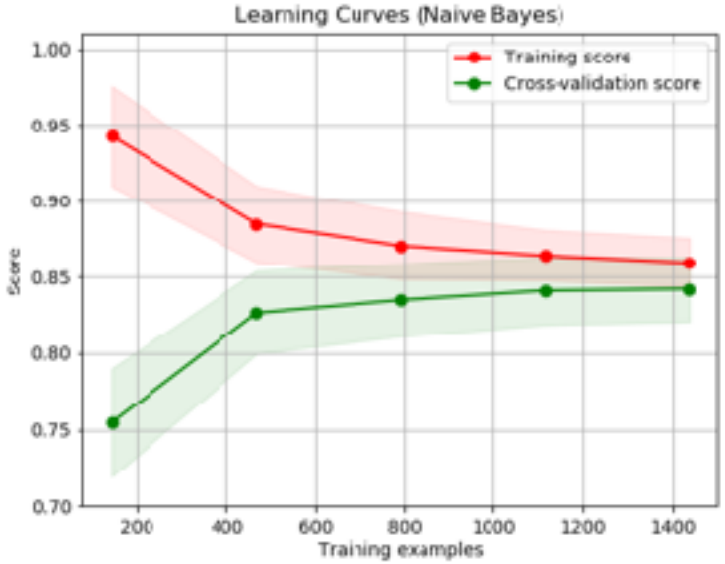


Scalable Algorithms

When your dataset is larger than RAM

First: Do you need all the data?

- Sampling may be OK
- *Plotting Learning Curves* from scikit-learn docs



Second: Parallel Meta-estimators

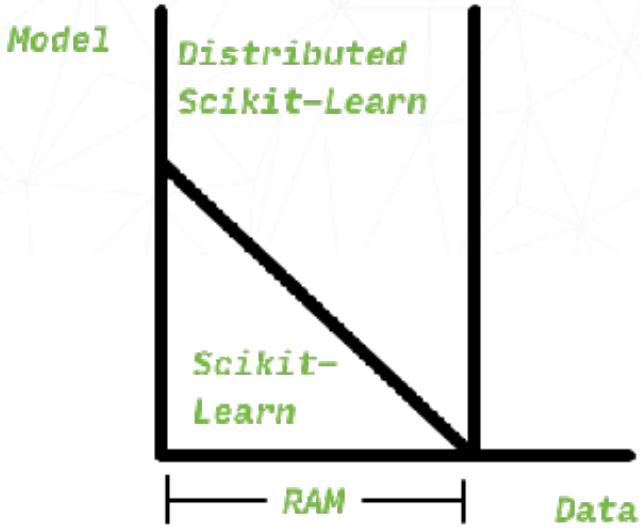
```
from dask_ml.wrappers import ParallelPostFit
import dask.dataframe as dd

clf = ParallelPostFit(SVC())
clf.fit(X_small, y_small)


X_large = dd.read_csv("s3://abc/*.parq")
y_large = clf.predict(X_large)
```

- Train on subset
- Predict for large dataset, in parallel

Scalable Estimators

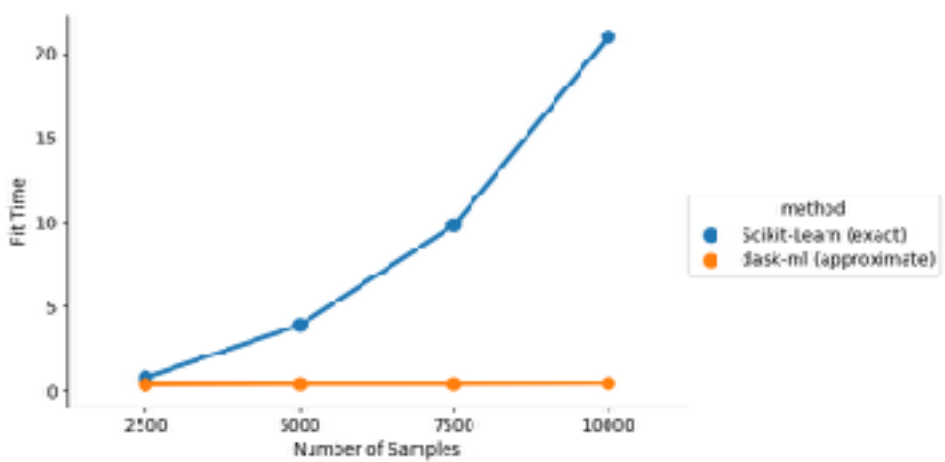
When the training dataset is larger than RAM

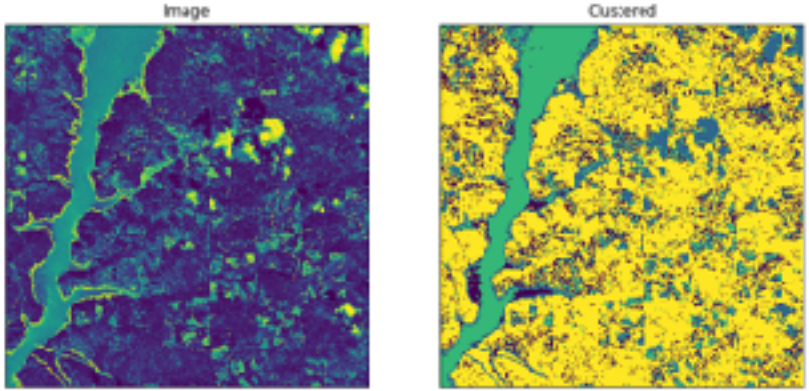
- Scikit-Learn wasn't designed for distributed datasets
- Dask-ML implements scalable variants of some estimators
- Works well with Dask DataFrames & Arrays



Scalable, Parallel Algorithms

Spectral Clustering Comparison





Scalable, Parallel Algorithms

Some Notable Estimators

- Distributed GLM
  - `LogisticRegression, LinearRegression, ...`
- Clustering
  - `KMeans(init='k-means||'), SpectraclClustering, ...`
- `Preprocessing`
  - `QuantileTransformer, RobustScalar, ...`
- `Dimensionality Reduction`
  - `PCA, TruncatedSVD`
- `...`

Familiar

Works well with existing libraries

Familiar

- Dask-ML estimators are Scikit-Learn estimators
- Dask-ML pipelines are Scikit-Learn Pipelines

Familiar

```
>>> from sklearn.pipeline import make_pipeline
>>> from sklearn.preprocessing import FunctionTransformer

>>> pipe = make_pipeline(
...     ColumnSelector(columns),
...     HourExtractor(['Trip_Pickup_DateTime']),
...     FunctionTransformer(payment_lowerer, validate=False),
...     Categorizer(categories),
...     DummyEncoder(),
...     StandardScaler(scale),
...     LogisticRegression(),
... )
```

Familiar

```
>>> from sklearn.pipeline import make_pipeline
>>> from sklearn.preprocessing import FunctionTransformer

>>> pipe = make_pipeline(
...     ColumnSelector(columns),
...     HourExtractor(['Trip_Pickup_DateTime']),
...     FunctionTransformer(payment_lowerer, validate=False),
...     Categorizer(categories),
...     DummyEncoder(),
...     StandardScaler(scale),
...     LogisticRegression(),
... )
```

Scikit-Learn objects

Custom transformers

Dask-ML estimators

Full Example: https://git.io/vAi7C

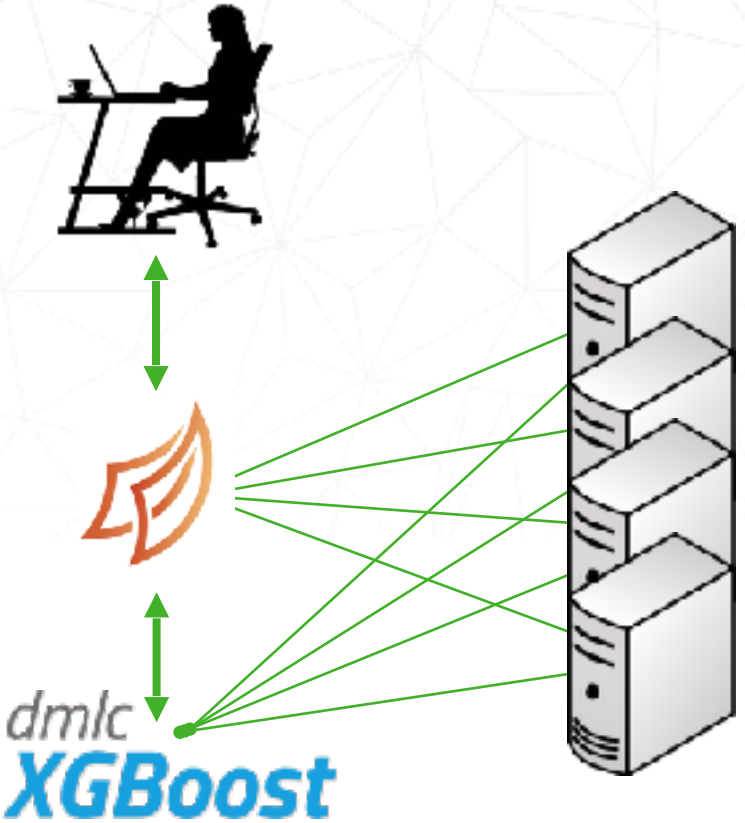
Distributed Systems

Integrate with XGBoost and Tensorflow

Distributed System

Peer with systems like XGBoost or Tensorflow

```
>>> import dask_ml.xgboost as xgb
>>> df = dd.read_csv("trips*.csv")
>>> y = df['Tip_Amt'] > 0
>>> X = df[columns]
>>> booster = xgb.train(
...     client, params, X, y
... )
```



Dask & Dask-ML

- **Parallelizes libraries** like NumPy, Pandas, and Scikit-Learn
- **Scales** from a laptop to thousands of computers
- **Familiar** API and in-memory computation
- https://dask.pydata.org

Questions?