Lecture 14
Advanced applications of ML & Python ML Ecosystem

Olexandr Isayev

Department of Chemistry, CMU

olexandr@cmu.edu

Thanks for taking 38/09-615 course

Class project presentation

Thursday, December 7

- Plan for ~10 min
- Peer grading + teacher!

Class project presentation

- Background
- Scientific Problem
- Dataset
- Data curation and processing (if any)
- ML experiments
- Lessons Learned

- Go into Jupyter and show important moment if you like

Grading Form

Final Project Evaluation Form 38-615/09-615: Comp. Modeling, Statistical Analysis and Machine Learning

Student name: _____

Grading System: **A** (Excellent) – **B** –**C** –**D** – **F** (Unacceptable)

Project	Team	Quality of project	Quality of solution	Presentation	Comments
Project B	Name 1 Name 2				
Project B	Name 3 Name 4				

Course Overview

615 (Fall) :

- Data
- Exploratory data analysis and visualization
- Unsupervised learning, clustering, dimensionality reduction
- Supervised learning, model training and evaluation
- Linear and nonlinear models
- Classification, SVM
- Decision trees and RF

616 (Spring):

- Neural networks & deep learning
- CNN
- RNN
- Graph neural networks
- Autoencoders
- Embeddings
- Generative models & GANs
- Reinforcement Learning
- Diffusion

616 Course Outline

- Basic concepts: Model accuracy, prediction accuracy, interpretability, supervised and unsupervised training, regularization.
- Deep Learning tools: PyTorch, PyTorch Geometric, Hugging face
- Artificial neural networks, feed-forward, activation functions, loss functions.
- Non-linear optimization, gradient descent, back-propagation
- Autoencoders, dense embedding, dimensionality reduction
- Convolutional networks, Graph-CNNs, transfer learning, applications in image processing and sciences
- Recurrent networks, LSTM, GRU, applications in NLP
- Generative Models: Autoregressive, GANs, Reinforcement Learning, Diffusion
- Other: Multitask Learning, advanced applications of deep learning in STEM sciences.

Course Evaluations

38-615

To ensure high FCE response rates, you can ask your class to scan this QR code into their cell phones and complete their MCS 38615 A taught by: ISAYEV Lect Graduate 2023 Fall evaluations.

09-615

To ensure high FCE response rates, you can ask your class to scan this QR code into their cell phones and complete their CMY 09615 A taught by: ISAYEV Lect Graduate 2023 Fall evaluations.

Please Give Feedback!

Fresh brand-new program & course

Please gave your feedback. olexandr@cmu.edu

Examples

Any topics missing? Not very useful topics..

Do you want to see exams? Do you want to see quizzes?

Were home assignments useful?

Due to interdisciplinary nature of the program, we did not do any formal derivations. Course takes pragmatic user-centered approach

Please Give Feedback!

Please gave your feedback. olexandr@cmu.edu

Examples

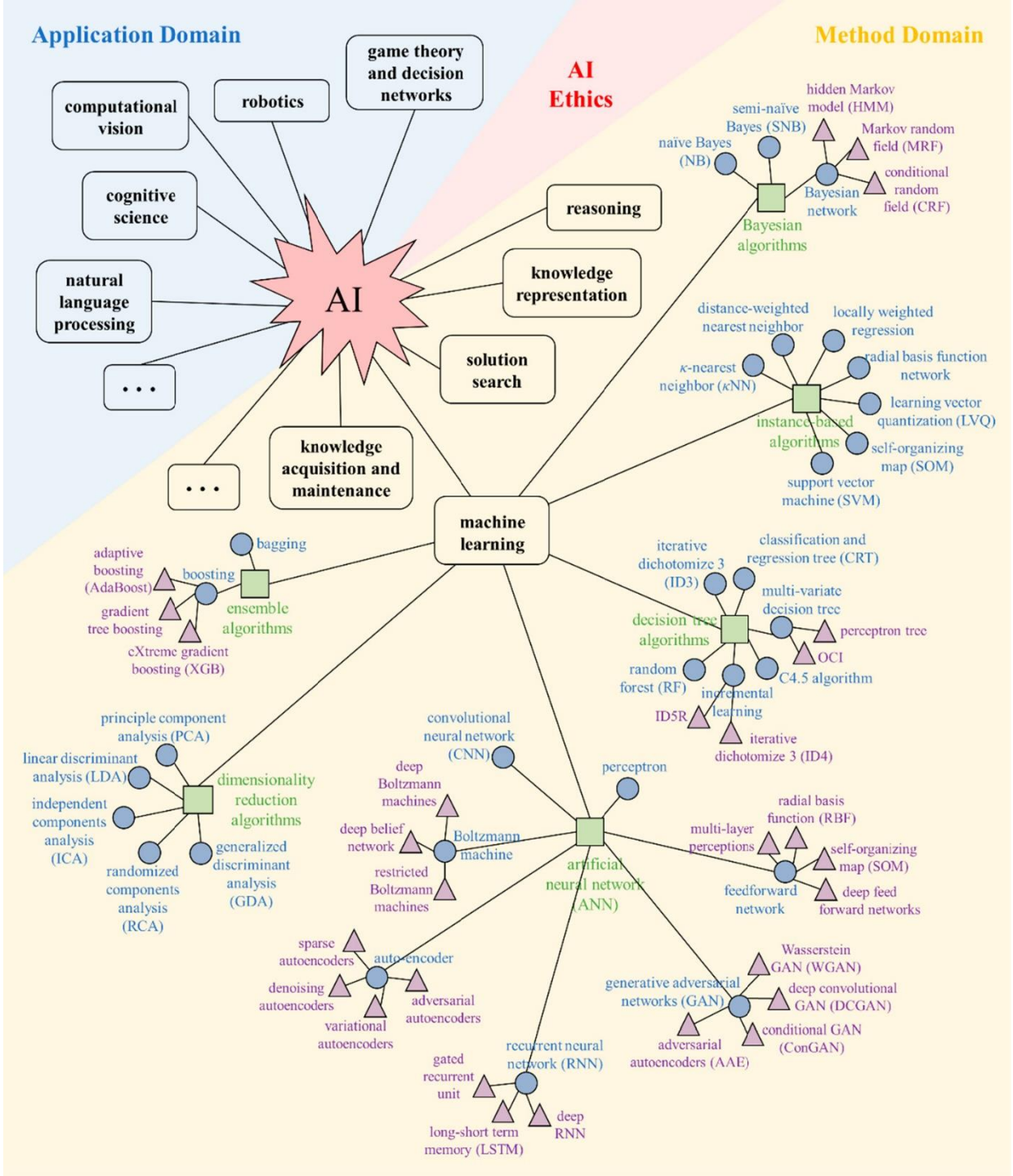
Any topics missing? Not very useful topics..

Do you want to see exams? Do you want to see quizzes?

Were home assignments useful?

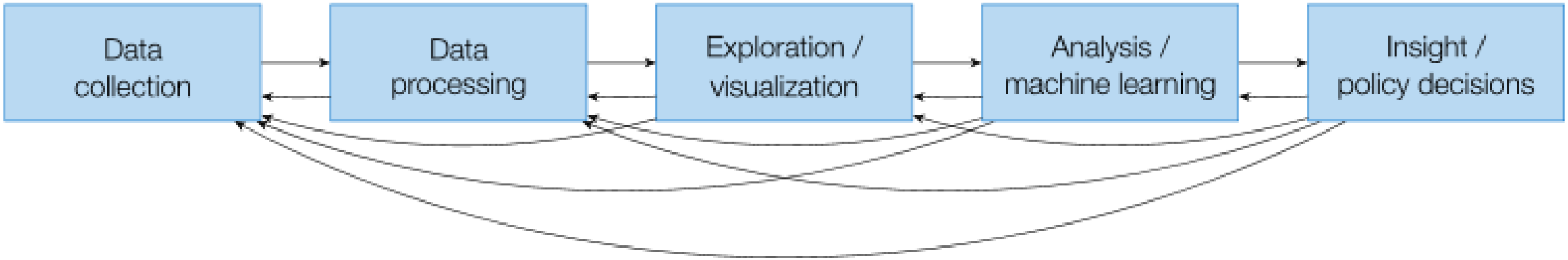
Due to interdisciplinary nature of the program, we did not do any formal derivations. Course takes pragmatic user-centered approach

Zoo of AI methods

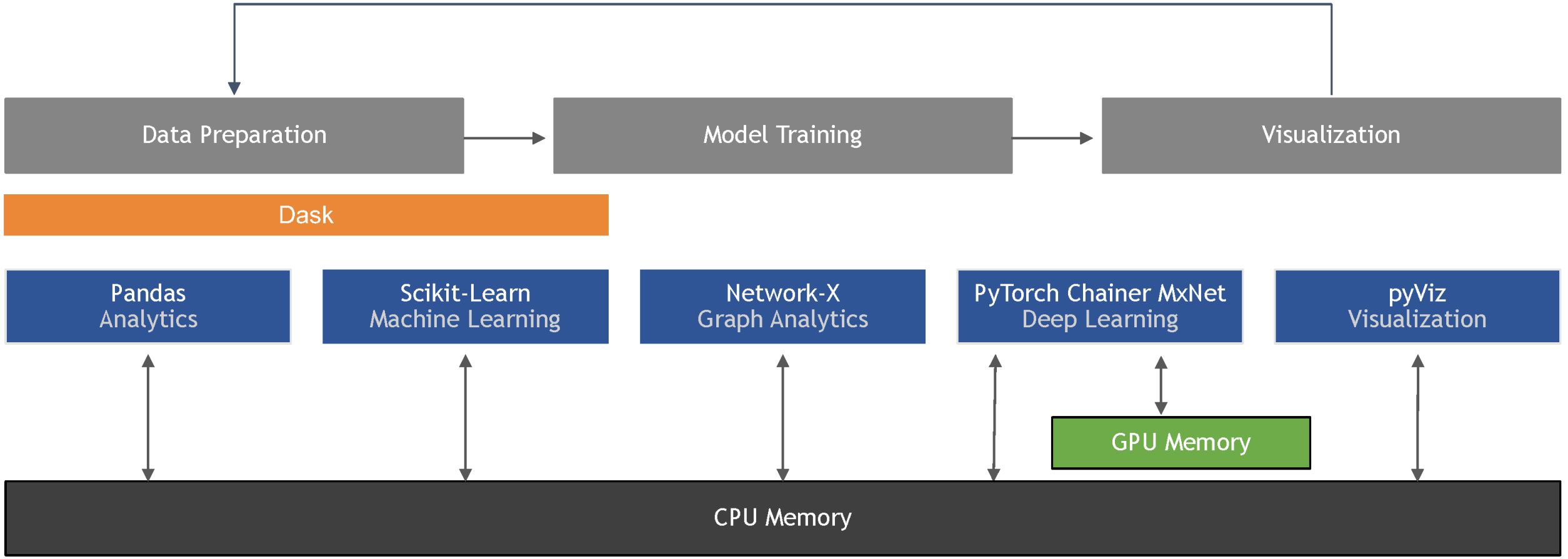


https://pubs.acs.org/doi/10.1021/acs.chemrev.8b00728

Course point of view: Data Pipeline



Python Ecosystem



The standard Python ecosystem for machine learning, data science, and scientific computing

Performance Optimization!

Optimizing Python's Performance for Numerical Computing

Take advantage of **Binary Linear Algebra Subroutines (BLAS)** and **Linear Algebra Pack (LAPACK)** libraries, for efficient matrix and vector operations

OpenBLAS

Intel Math Kernel Library (Intel MKL)

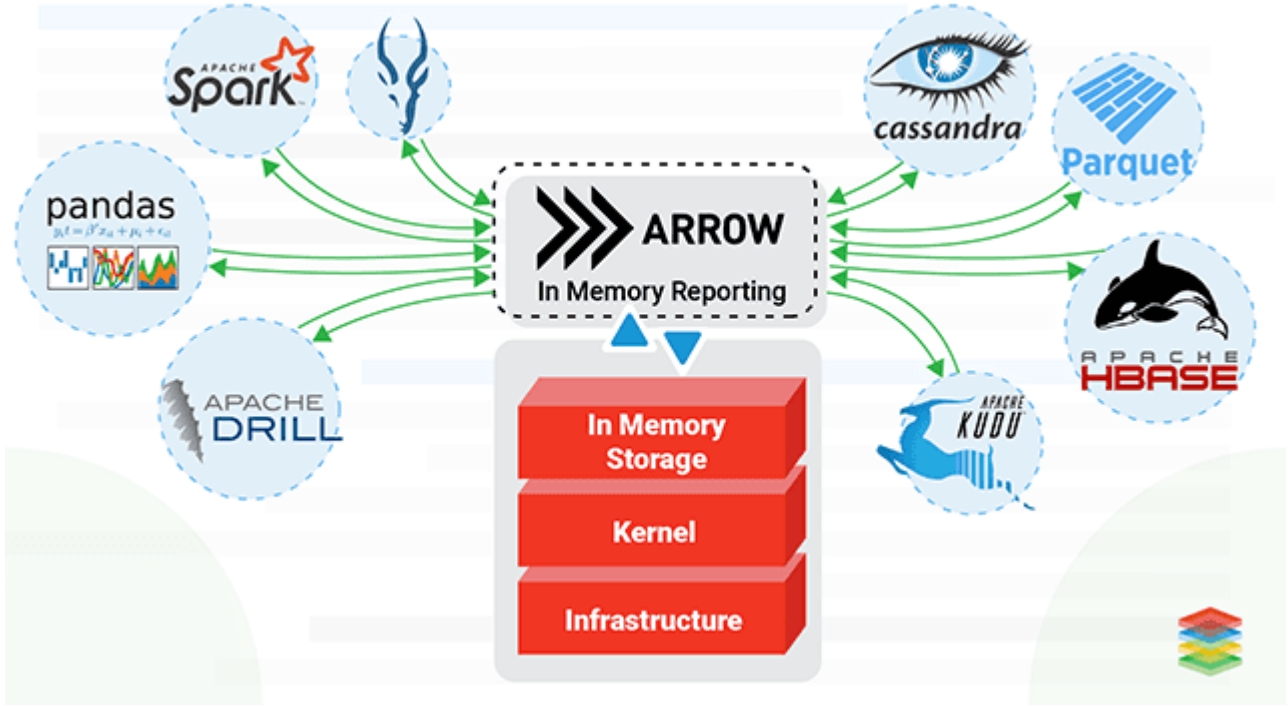
```
The following NEW packages will be INSTALLED:

  blas               pkgs/main/win-64::blas-1.0-mkl
  ca-certificates    pkgs/main/win-64::ca-certificates-2021.4.13-haa95532_1
  certifi            pkgs/main/win-64::certifi-2020.12.5-py39haa95532_0
  icc_rt             pkgs/main/win-64::icc_rt-2019.0.0-h0cc432a_1
  intel-openmp       pkgs/main/win-64::intel-openmp-2021.2.0-haa95532_616
  mkl                pkgs/main/win-64::mkl-2021.2.0-haa95532_296
  mkl-service        pkgs/main/win-64::mkl-service-2.3.0-py39h2bbff1b_1
  mkl_fft            pkgs/main/win-64::mkl_fft-1.3.0-py39h277e83a_2
  mkl_random         pkgs/main/win-64::mkl_random-1.2.1-py39hf11a4ad_2
  numpy              pkgs/main/win-64::numpy-1.20.2-py39ha4e8547_0
  numpy-base         pkgs/main/win-64::numpy-base-1.20.2-py39hc2deb75_0
  openssl            pkgs/main/win-64::openssl-1.1.1k-h2bbff1b_0
  pip                pkgs/main/win-64::pip-21.1.1-py39haa95532_0
  python             pkgs/main/win-64::python-3.9.5-h6244533_3
  scipy              pkgs/main/win-64::scipy-1.6.2-py39h66253e8_1
  setuptools         pkgs/main/win-64::setuptools-52.0.0-py39haa95532_0
  six                pkgs/main/win-64::six-1.15.0-py39haa95532_0
  sqlite             pkgs/main/win-64::sqlite-3.35.4-h2bbff1b_0
  tzdata             pkgs/main/noarch::tzdata-2020f-h52ac0ba_0
  vc                 pkgs/main/win-64::vc-14.2-h21ff451_1
  vs2015_runtime     pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e58377_2
  wheel              pkgs/main/noarch::wheel-0.36.2-pyhd3eb1b0_0
  wincertstore       pkgs/main/win-64::wincertstore-0.2-py39h2bbff1b_0


Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

```
# Name                    Version                   Build  Channel
blas                      1.0                         mkl
ca-certificates           2021.4.13            haa95532_1
certifi                   2020.12.5         py39haa95532_0
icc_rt                    2019.0.0             h0cc432a_1
intel-openmp              2021.2.0           haa95532_616
mkl                       2021.2.0           haa95532_296
mkl-service               2.3.0             py39h2bbff1b_1
mkl_fft                   1.3.0             py39h277e83a_2
mkl_random                1.2.1             py39hf11a4ad_2
numpy                     1.20.2            py39ha4e8547_0
numpy-base                1.20.2            py39hc2deb75_0
openssl                   1.1.1k               h2bbff1b_0
pandas                    1.2.1             py39hf11a4ad_0
pip                       21.1.1            py39haa95532_0
python                    3.9.5                h6244533_3
python-dateutil           2.8.1               pyhd3eb1b0_0
pytz                      2021.1              pyhd3eb1b0_0
scipy                     1.6.2             py39h66253e8_1
setuptools                52.0.0            py39haa95532_0
six                       1.15.0            py39haa95532_0
sqlite                    3.35.4               h2bbff1b_0
tzdata                    2020f                h52ac0ba_0
vc                        14.2                 h21ff451_1
vs2015_runtime            14.27.29016          h5e58377_2
wheel                     0.36.2              pyhd3eb1b0_0
wincertstore              0.2               py39h2bbff1b_0
```



The Apache Arrow cross-language development platform for in-memory data standardizes the columnar format so that data can be shared across different libraries without the costs associated with having to copy and reformat the data.

Another library that takes advantage of the columnar format is Apache Parquet. Whereas libraries such as Pandas and Apache Arrow are designed with in-memory use in mind, Parquet is primarily designed for data serialization and storage on disk.

Both Arrow and Parquet are compatible with each other, and modern and efficient workflows involve Parquet for loading data files from disk into Arrow's columnar data structures for in-memory computing.


pandas
DASK
Spark

Spark

Pros

- Established and mature technology (original release in May 2014).
- Plenty of companies providing commercial support / services.
- Ideal for data engineering / ETL type of tasks against large datasets.
- Provides higher-level SQL abstractions (Spark SQL).

Cons

- A steep learning curve involving a new execution model and API.
- Debugging can be challenging.
- Complex architecture, which is difficult to maintain by IT alone as proper maintenance requires understanding of the computation paradigms and inner workings of Spark (e.g. memory allocation).
- Lack of a rich data visualization ecosystem.
- No built-in GPU acceleration. Needs **RAPIDS Accelerator** for accessing GPU resources.

Dask

Pros

- Pure Python framework - very easy to ramp up.
- Out-of-the-box support for Pandas DataFrames and NumPy arrays.
- Easy exploratory data analysis against billions of rows via **Datashader**.
- Provides ***Dask Bags*** - a Pythonic version of the PySpark RDD, with functions like *map*, *filter*, *groupby*, etc.
- Dask can lead to impressive performance improvements. Runs on CPUs & GPUs

Cons

- Not much commercial support is available (but several companies are starting to work in this space, for example, Coiled and QuanSight).
- No built-in GPU support. **Relies on RAPIDS** for GPU acceleration.

a

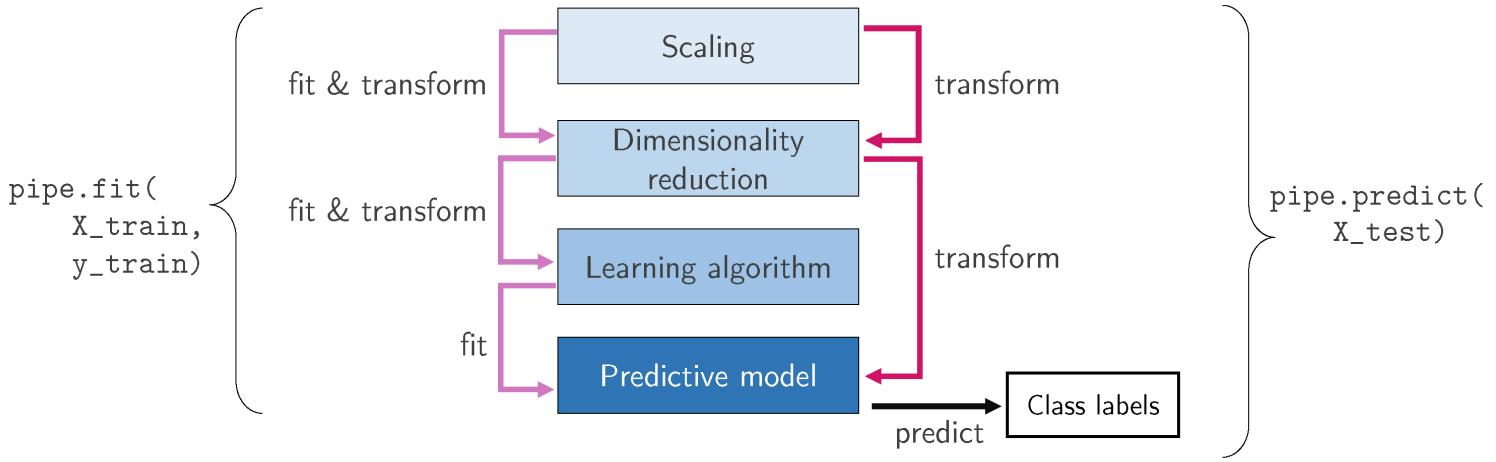
```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn.pipeline import make_pipeline
from sklearn import datasets
from sklearn.model_selection import train_test_split

iris = datasets.load_iris()
X, y = iris.data, iris.target
X_train, X_test, y_train, y_test =\
    train_test_split(X, y, test_size=0.3,
                     random_state=42, stratify=y)

pipe = make_pipeline(StandardScaler(),
                     PCA(n_components=2),
                     SVC(kernel='linear'))

pipe.fit(X_train, y_train)
y_pred = pipe.predict(X_test)
print('Test Accuracy: %.3f' % pipe.score(X_test, y_test))
```

b



Addressing Class Imbalance

Imbalanced-learn - https://imbalanced-learn.org/stable/

Imbalanced-learn is a Scikit-contrib library written to address the above problem with four different techniques for balancing the classes in a skewed dataset.

The first two techniques resample the data by either reducing the number of instances of the data samples that contribute to the over-represented class (under-sampling) or generating new data samples of the under-represented classes (over-sampling).

Since over-sampling tends to train models that overfit the data, the third technique combines over-sampling with a “cleaning” under-sampling technique that removes extreme outliers in the majority class.

The final technique that Imbalanced-learn provides for balancing classes combines bagging with AdaBoost whereby a model ensemble is built from different under-sampled sets of the majority class, and the entire set of data from the minority class is used to train each learner.

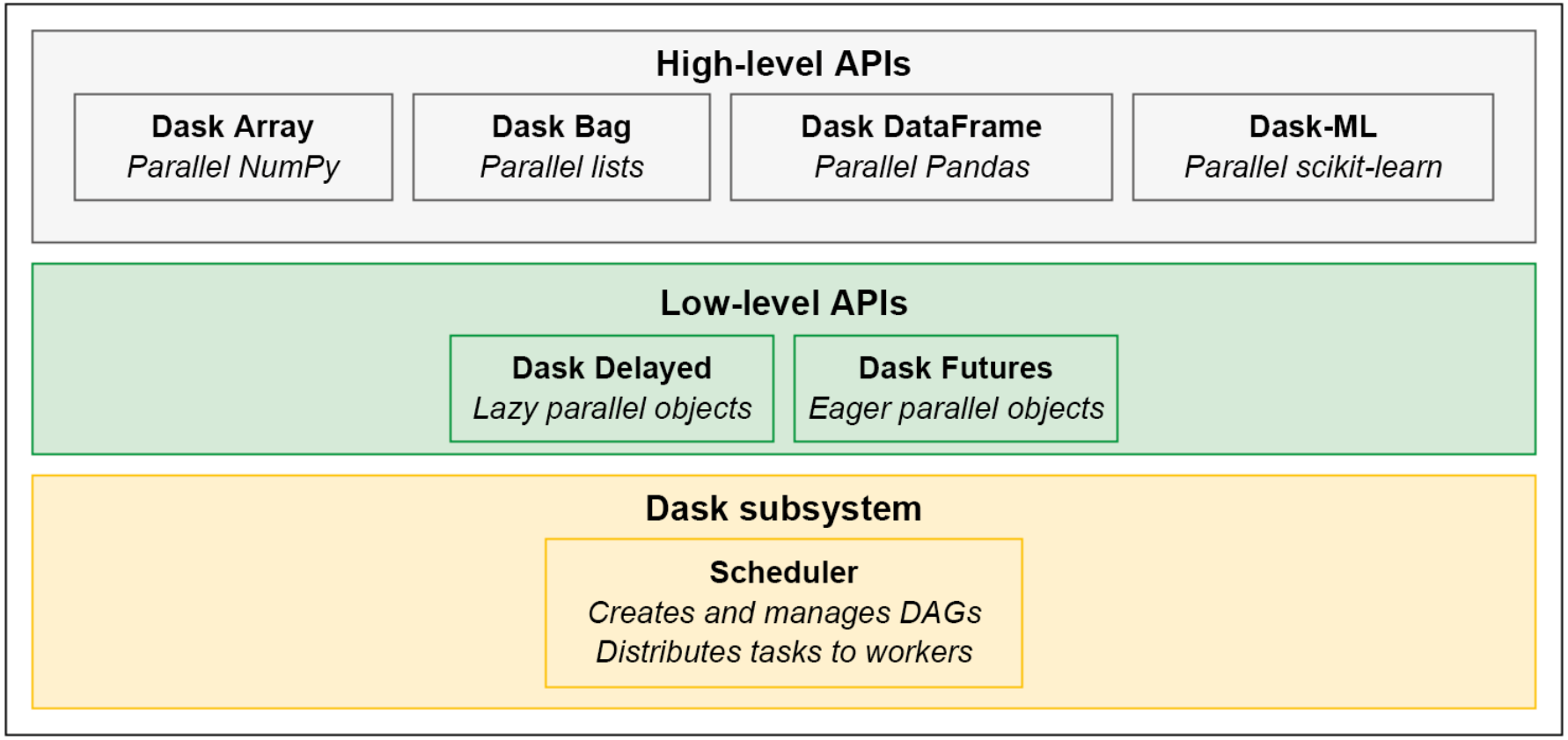
Scalable Distributed Machine Learning

While Scikit-learn is targeted for small to medium-sized datasets, modern problems often require libraries that can scale to larger data sizes.

Using the Joblib (**https://github.com/joblib/joblib**) API, a handful of algorithms in Scikit-learn are able to be parallelized through Python's multiprocessing.

Unfortunately, the potential scale of these algorithms is bounded by the amount of memory and physical processing cores on a single machine.

Dask-ML

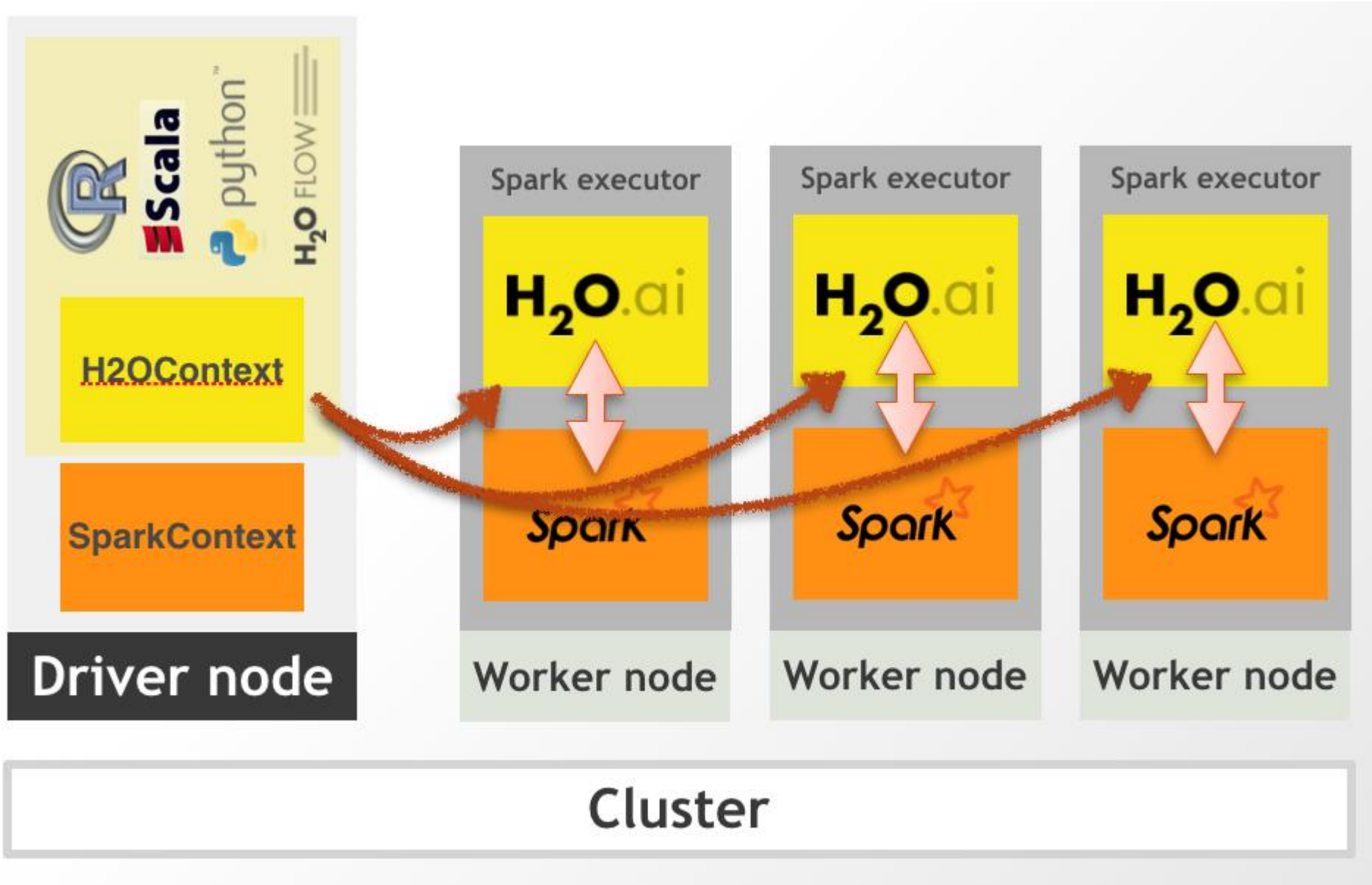


Dask-ML provides distributed versions of a subset of Scikit-learn's classical ML algorithms with a Scikit-learn compatible API. These include supervised learning algorithms like linear models, unsupervised learning algorithms like k-means, and dimensionality reduction algorithms like principal component analysis and truncated singular vector decomposition. Dask-ML uses multiprocessing with the additional benefit that computations for the algorithms can be distributed over multiple nodes in a compute cluster.

Hyperparameter tuning

- Hyperparameter tuning is a very important use-case in machine learning, requiring the training and testing of a model over many different configurations to find the model with the best predictive performance.
- The ability to train multiple smaller models in parallel, especially in a distributed environment, becomes important when multiple models are being combined.
- Dask-ML also provides a hyperparameter optimization (HPO) library that supports any Scikit-learn compatible API.
- Dask-ML's HPO distributes the model training for different parameter configurations over a cluster of Dask workers to speed up the model selection process.

H2O machine learning library


Scala
python
H2O FLOW
H2OContext
SparkContext
Driver node
Spark executor
H2O.ai
Spark
Worker node
Spark executor
H2O.ai
Spark
Worker node
Spark executor
H2O.ai
Spark
Worker node
Cluster

Automatic Machine Learning (AutoML)

a

b

AutoML stages for generating and tuning models

Several major AutoML libraries have become quite popular since the initial introduction of Auto-Weka in 2013. Currently:

- Auto-sklearn (https://www.automl.org/automl/auto-sklearn/)
- TPOT (http://epistasislab.github.io/tpot/)
- H2O-AutoML (https://h2o.ai/platform/h2o-automl/)
- Optuna (https://optuna.org/)
- Microsoft’s NNI (https://github.com/microsoft/nni),

are the most popular ones among practitioners

Hyperparameter Optimization and Model Evaluation

Hyperparameter optimization (HPO) algorithms form the core of AutoML.

The most naïve approach to finding the best performing model would exhaustively select and evaluate all possible configurations to ultimately select the best performing model.

The goal of HPO is to improve upon this exhaustive approach by optimizing the search for hyperparameter configurations or the evaluation of the resulting models, where the evaluation involves cross-validation with the trained model to estimate the model's generalization performance.

- *Grid search* is a brute-force-based search method that explores all configurations within a user-specified parameter range.

- Related to grid search, *random search* is a brute-force approach. However, instead of evaluating all configurations in a user-specified parameter range exhaustively, it chooses configurations at random, usually from a bounded area of the total search space.

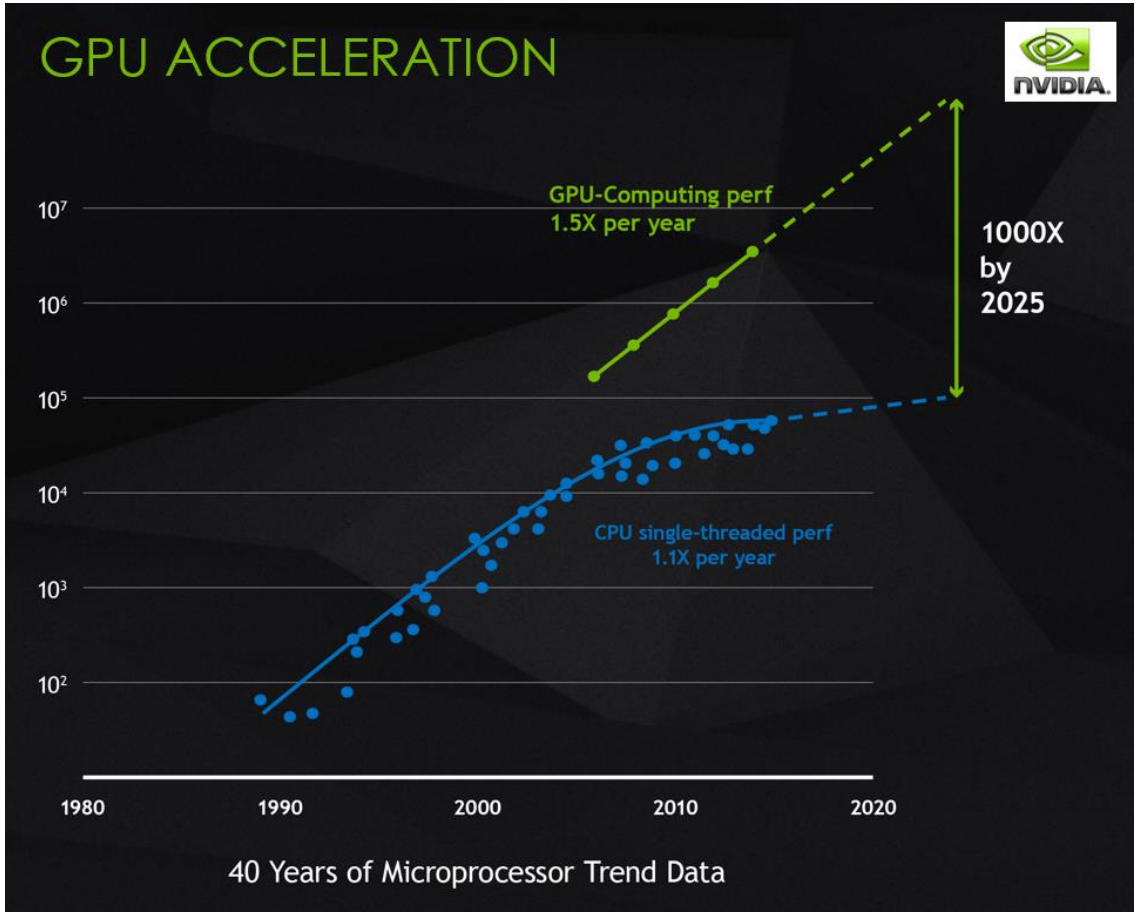
Hyperband

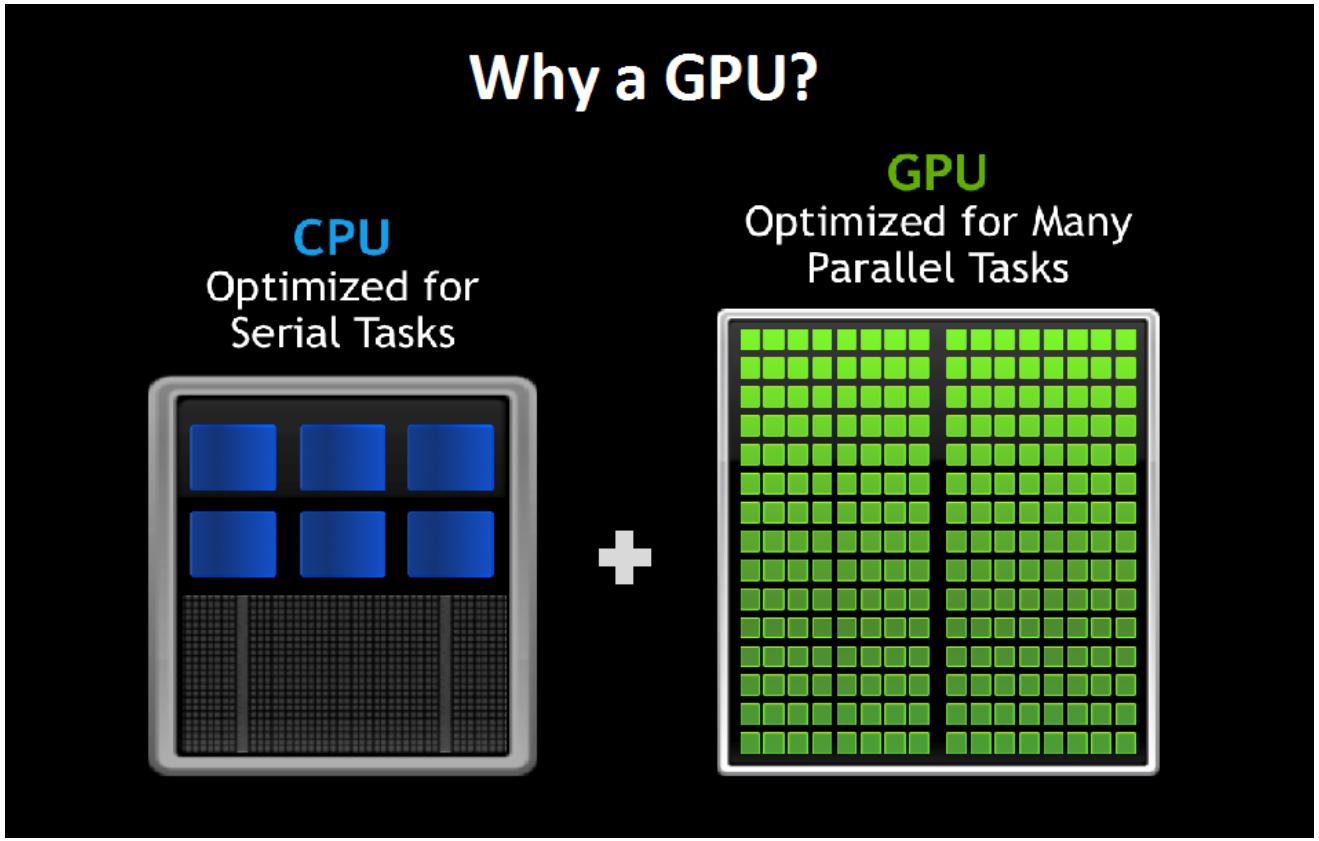
- Some algorithms, such as the *Hyperband* algorithm used in Dask-ML, Auto-sklearn, and H2O-AutoML, resort to random search and focus on optimizing the model evaluation stage to achieve good results.
- Hyperband uses an evaluation strategy known as early stopping, where multiple rounds of cross-validation for several configurations are started in parallel

Bayesian optimization (BO)

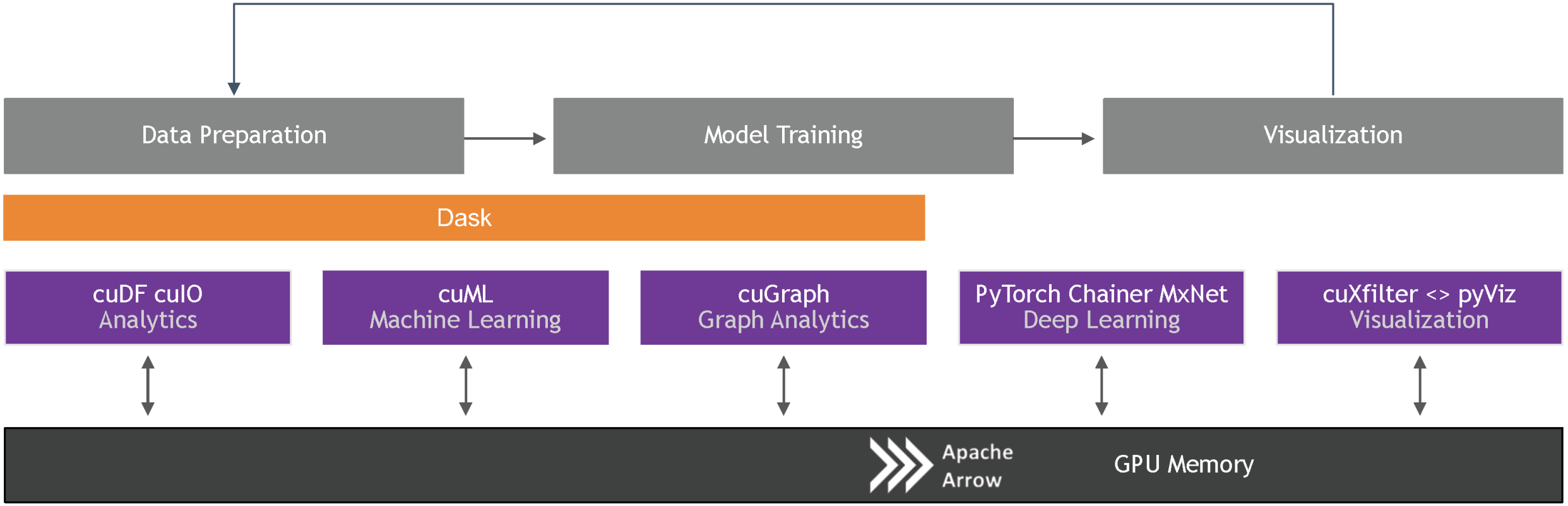
- Bayesian optimization (BO) focus on selecting better configurations using probabilistic models.
- Several libraries use a formalism of BO, known as sequential model-based optimization (SMBO), to build a probabilistic model through trial and error.
- The Hyperopt library brings SMBO to Spark ML, using an algorithm known as tree of Parzen estimators.
- The Bayesian optimized hyperband (BOHB) library combines BO and Hyperband, while providing its own built-in distributed optimization capability.
- Auto-sklearn uses an SMBO approach called sequential model algorithm configuration (SMAC).
- Similar to early stopping, SMAC uses a technique called adaptive racing to evaluate a model only as long as necessary to compare against other competitive models (https://github.com/automl/SMAC3).

GPU Acceleration





RAPIDS (https://rapids.ai)



RAPIDS is an open source effort to support and grow the ecosystem of GPU-accelerated Python tools for data science, machine learning, and scientific computing. RAPIDS supports existing libraries, fills gaps by providing open source libraries with crucial components that are missing from the Python community, and promotes cohesion across the ecosystem by supporting interoperability across the libraries.

HW5: Regression

Build any regression model, focus on model engineering

Find BEST model

Use your model to score Ytest on Leaderboard

Problem: predict band gap in solid state