# INTEL® ARTIFICIAL INTELLIGENCE OPTIMIZED SOFTWARE

Dr. Fabio Baruffa

Sr. Technical Consulting Engineer, Intel IAGS

Visit:
www.intel.ai/technology

# SPEED UP DEVELOPMENT

## using open AI software

MACHINE LEARNING ⟷ DEEP LEARNING

### TOOLKITS
App developers

**ANALYTICS ZOO**
Open source platform for building E2E Analytics & AI applications on Apache Spark* with distributed TensorFlow*, Keras*, BigDL

**OpenVINO™**
Deep learning inference deployment on CPU/GPU/FPGA/VPU for Caffe*, TensorFlow*, MXNet*, ONNX*, Kaldi*

**NAUTA**
Open source, scalable, and extensible distributed deep learning platform built on Kubernetes (BETA)

### LIBRARIES
Data scientists

**Python**
- Scikit-learn
- Pandas
- NumPy

**R**
- Cart
- Random Forest
- e1071

**Distributed**
- MlLib (on Spark)
- Mahout

**Intel-optimized Frameworks**
TensorFlow*, Caffe2*, ONNX*, mxnet*, PyTorch*, BigDL

And more framework optimizations underway including PaddlePaddle*, Chainer*, CNTK* & others

### KERNELS
Library developers

**Intel® Distribution for Python***
*Intel distribution optimized for machine learning*

**Intel® Data Analytics Acceleration Library (Intel® DAAL)**
*High performance machine learning & data analytics library*

**Intel® Math Kernel Library for Deep Neural Networks (Intel® MKL-DNN)**
*Open source DNN functions for CPU / integrated graphics*

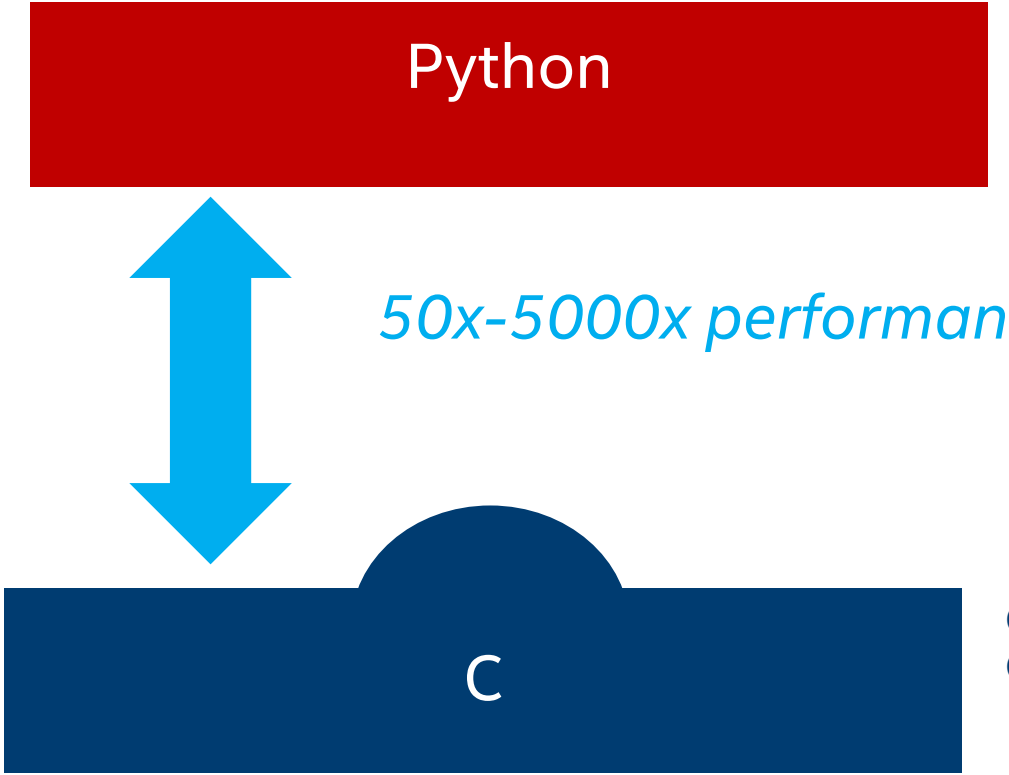
**nGraph**
*Open source compiler for deep learning model computations optimized for multiple devices (CPU, GPU, NNP) from multiple frameworks (TF, MXNet, ONNX)*

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Software

# INTEL® DISTRIBUTION FOR PYTHON 2019

# PERFORMANCE OF PYTHON



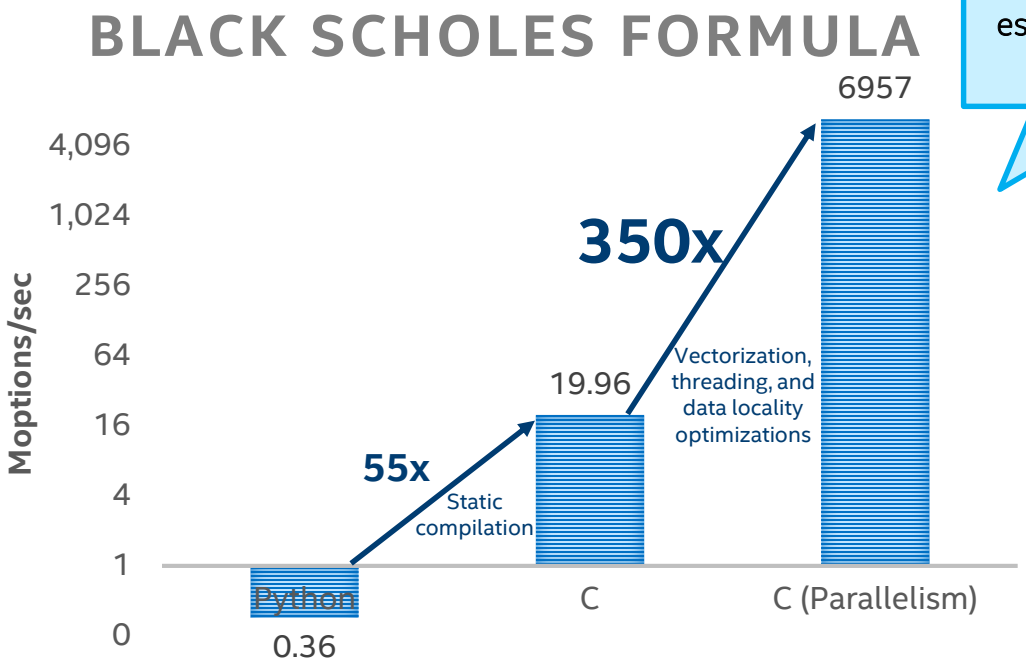

Interpreter
GIL (constraining parallelism)

*50x-5000x performance gap (or even more)*

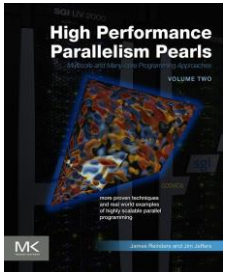
Optimizing compiler
OpenMP*/TBB/pthreads

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

# PARALLELISM MATTERS MOST




Unlocking parallelism is essential to make Python useful in production




Chapter 19: Performance Optimization of **Black—Scholes** Pricing

$$V_{call} = S_0 \cdot \mathrm{CDF}(d_1) - e^{-rT} \cdot X \cdot \mathrm{CDF}(d_2)$$

$$V_{put} = e^{-rT} \cdot X \cdot \mathrm{CDF}(-d_2) - S_0 \cdot \mathrm{CDF}(-d_1)$$

$$d_1 = \frac{\ln\left(S_0/X\right) + \left(r + \sigma^2/2\right)T}{\sigma\sqrt{T}}$$

$$d_2 = \frac{\ln\left(S_0/X\right) + \left(r - \sigma^2/2\right)T}{\sigma\sqrt{T}}$$

Configuration info: - Versions: Intel® Distribution for Python 2.7.10 Technical Preview 1 (Aug 03, 2015), icc 15.0; Hardware: Intel® Xeon® CPU E5-2698 v3 @ 2.30GHz (2 sockets, 16 cores each, HT=OFF), 64 GB of RAM, 8 DIMMS of 8GB@2133MHz; Operating System: Ubuntu 14.04 LTS.

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

# TWO INGREDIENTS TO GET CLOSE-TO-NATIVE PERFORMANCE

**Pure Python**

Serial
Interpreted

**Python + Libraries**

Partially Ninja-level
Partially Interpreted

**Libraries + JITC**

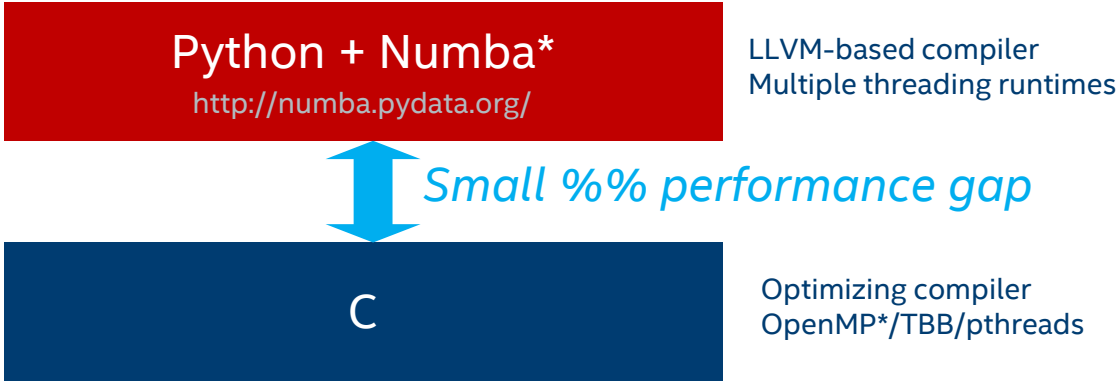
Largely Ninja-level
100% native

**C++**

100% Ninja-level
100% native

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

# PERFORMANCE OF PYTHON




```
@numba.jit(nopython=True, parallel=True)
def logistic_regression(Y, X, w0, step, iterations):
    """SGD solver for binary logistic regression."""
    w = w0.copy()
    for i in range(iterations):
        w += step * np.dot((1.0/(1.0 + np.exp(Y * np.dot(X, w)))) * Y, X)
    return w
```

https://www.anaconda.com/blog/developer-blog/parallel-python-with-numba-and-parallelaccelerator/

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

# HIGH PERFORMANCE PYTHON

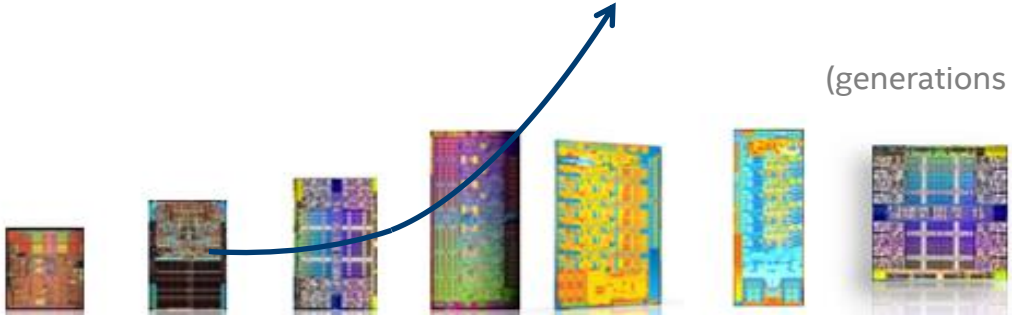
**Python Libraries**

Thin layer in Python or Cython

**Intel® Performance Libraries**

Native highly optimized libraries
(Intel® MKL, Intel® DAAL, Intel® IPP)

(generations of processors)

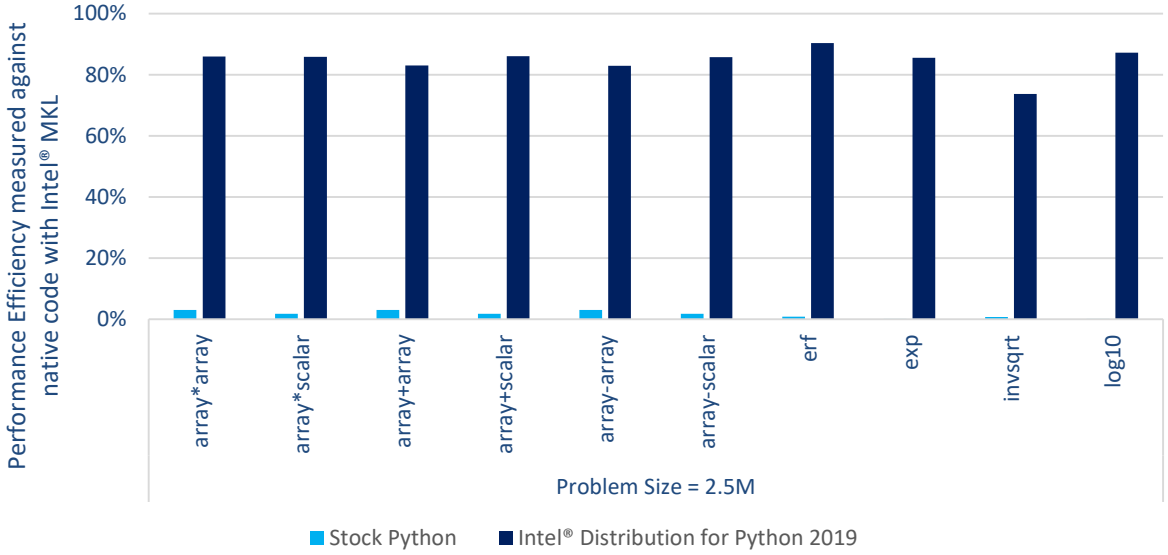


*more nodes,*
*more cores,*
*more threads,*
*wider vectors, …*

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

# CLOSE TO NATIVE CODE UMATH PERFORMANCE WITH INTEL PYTHON 2019

## COMPARED TO STOCK PYTHON PACKAGES ON INTEL® XEON PROCESSORS




***87%***

*native efficiency on a full **Black-Scholes** code with Intel **numpy + numba**.*

**Configuration: Stock Python: python 3.6.6 hc3d631a_0 installed from conda, numpy 1.15, numba 0.39.0, llvmlite 0.24.0, scipy 1.1.0, scikit-learn 0.19.2 installed from pip;Intel Python: Intel Distribution for Python 2019 Gold: python 3.6.5 intel_11, numpy 1.14.3 intel_py36_5, mkl 2019.0 intel_101, mkl_fft 1.0.2 intel_np114py36_6,mkl_random 1.0.1 intel_np114py36_6, numba 0.39.0 intel_np114py36_0, llvmlite 0.24.0 intel_py36_0, scipy 1.1.0 intel_np114py36_6, scikit-learn 0.19.1 intel_np114py36_35; OS: CentOS Linux 7.3.1611, kernel 3.10.0-514.el7.x86_64; Hardware: Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz (2 sockets, 18 cores/socket, HT:off), 256 GB of DDR4 RAM, 16 DIMMs of 16 GB@2666MHz**

**Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks. Source: Intel Corporation - performance measured in Intel labs by Intel employees. Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804.**

Optimization Notice

Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

# Intel® Distribution for Python*

https://software.intel.com/en-us/distribution-for-python

conda create –c intel intelpython3_full
docker pull intelpython/intelpython3_full
pip install intel-numpy intel-scipy intel-scikit-learn

**Accelerated NumPy, SciPy**
- Intel® MKL
- Intel® C and Fortran compilers
  - Linear algebra, universal functions, FFT

**Accelerated Scikit-Learn**
- Intel® MKL
- Intel® C and Fortran compilers
- Intel® Data Analytics Acceleration Library (DAAL)

via NumPy/Scipy

**Solutions for efficient parallelism**
- TBB4py
- github.com/IntelPython/smp
- Intel® MPI library

**Python APIs for Intel® MKL functions**
- github.com/IntelPython/mkl_fft
- github.com/IntelPython/mkl_random
- github.com/IntelPython/mkl-service [*]

**Python APIs for Intel® DAAL**
- github.com/IntelPython/daal4py

**Numba** with upstreamed Intel contributions
- Parallel Accelerator
- support for SVML
- support for TBB/OpenMP threading runtimes

https://software.intel.com/en-us/distribution-for-python/benchmarks

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Software

# HANDS-ON PREPARATION

# Hands-On Sessions are for You!

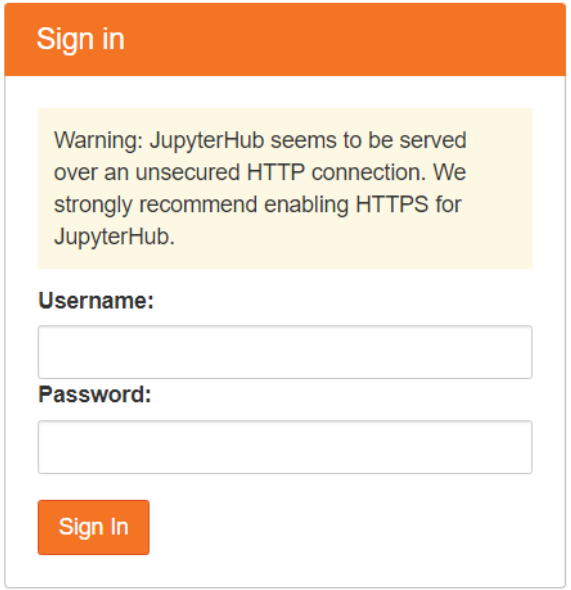
Take your time to understand the Python code samples – don't just execute Jupyter cells 1by1

Also… there are solution files available, while it is in your own interest trying to find a solution yourself …

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

# ACCESS THE JUPYTER SERVER

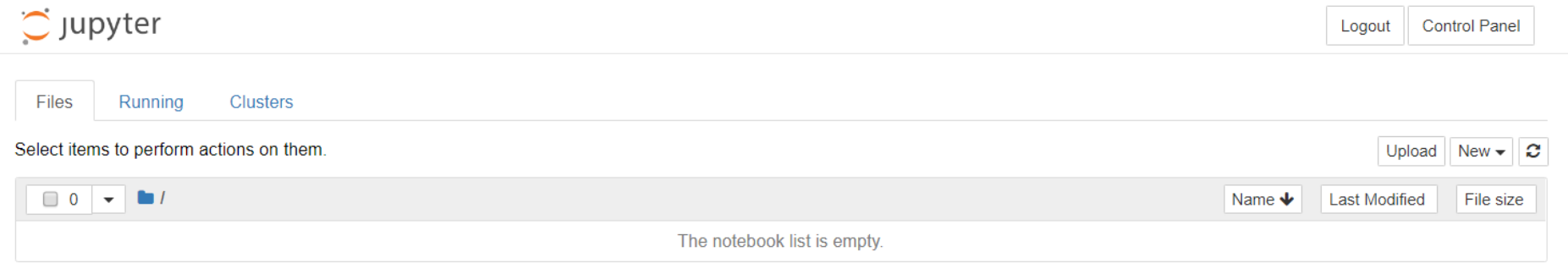
- Please take one **IP_ADDRESS** from the list we have provided to you
- Open your browser and go to the following url: **http://IP_ADDRESS**




- ➢ Choose any Username and Password for your access
- ➢ Then Sign in

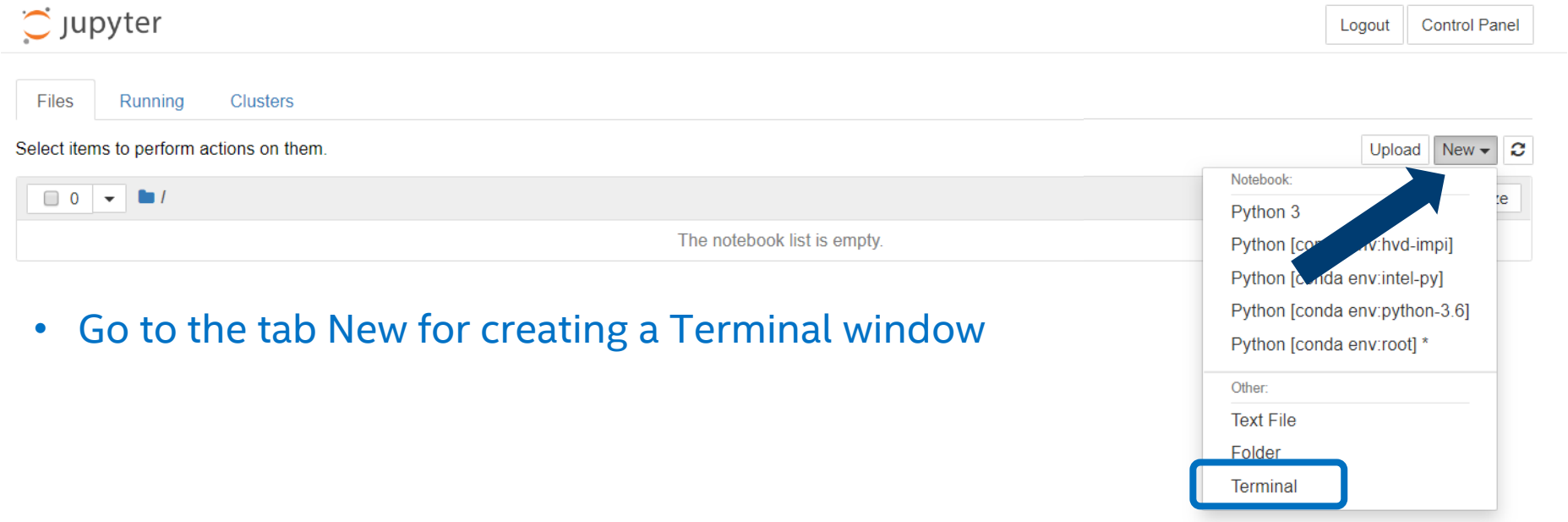
Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

# ACCESS THE JUPYTER SERVER




Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

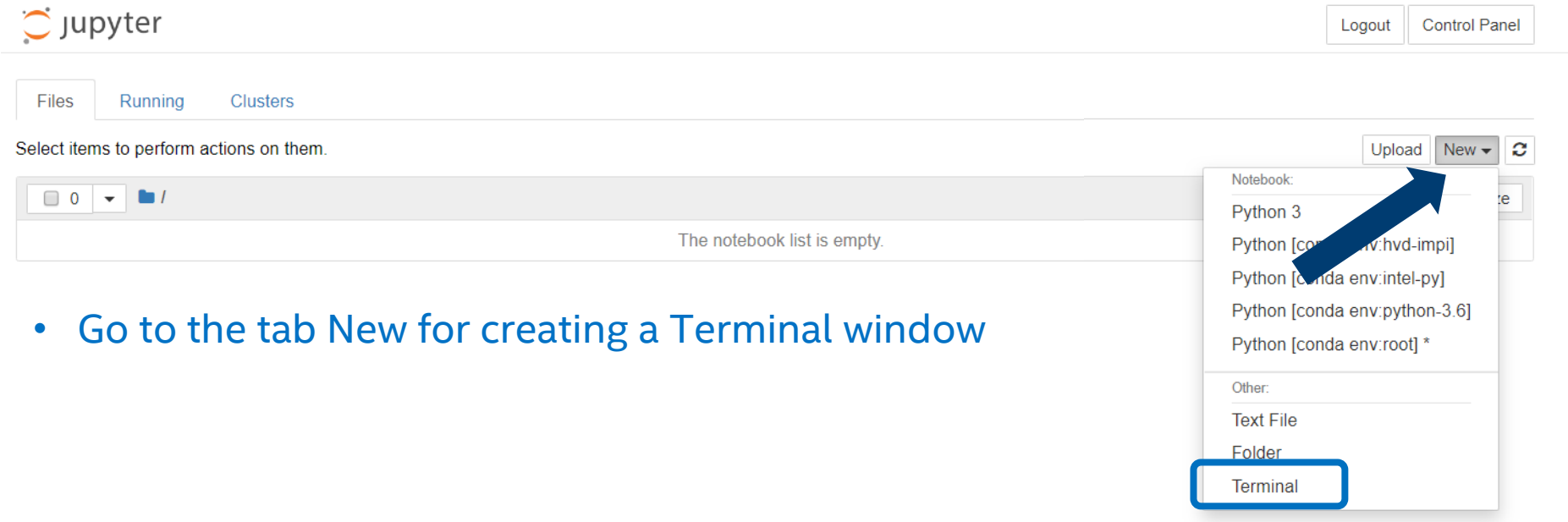
# ACCESS THE JUPYTER SERVER




- Go to the tab New for creating a Terminal window

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

# ACCESS THE JUPYTER SERVER



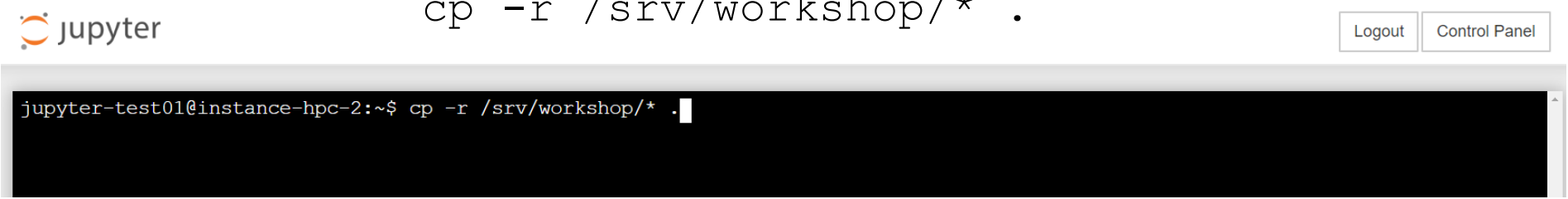

- Go to the tab New for creating a Terminal window

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

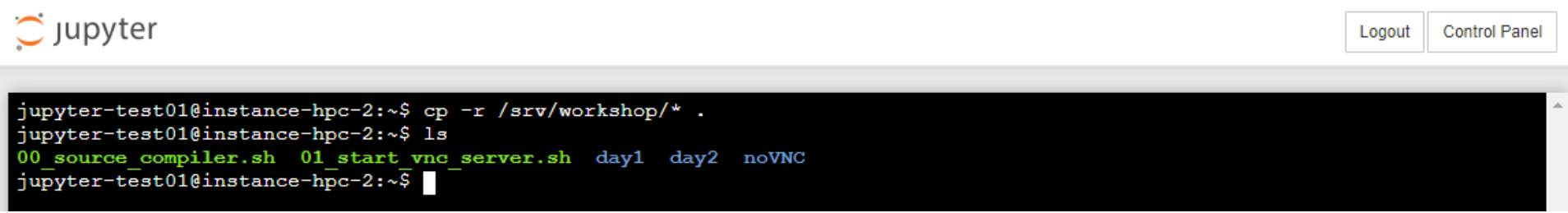
# ACCESS THE JUPYTER SERVER

- Type the following in the Terminal Window to copy the workshop materials:

```
cp -r /srv/workshop/* .
```




- When you type `ls`, you should see something like:




Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

# NUMPY-NUMBA:

## black-scholes.ipynb

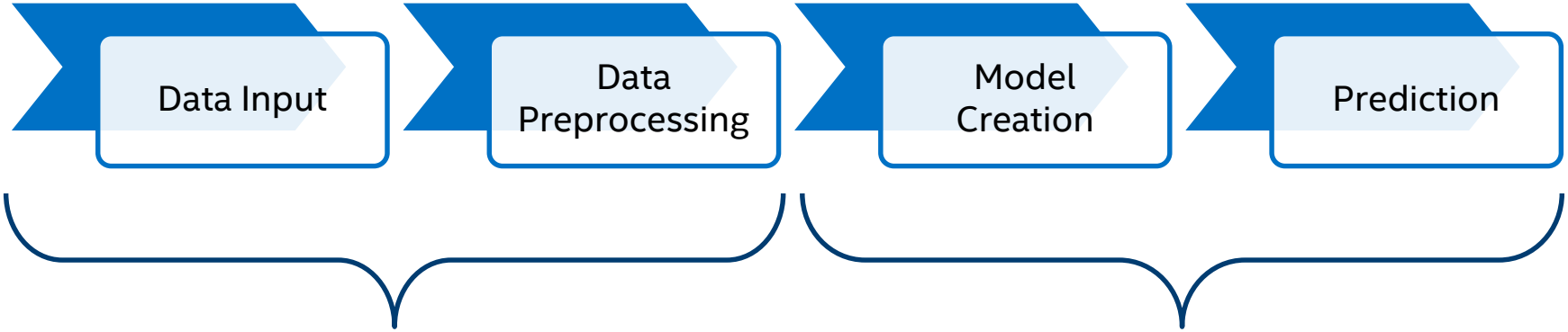
1) Why is the performance using the NumPy functions is lower than expected?
2) Implement the black_scholes function in a NumPy like fashion
3) Measure the speedup and explain where exactly it is coming from
4) Implement the black_scholes function using Numba

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Software

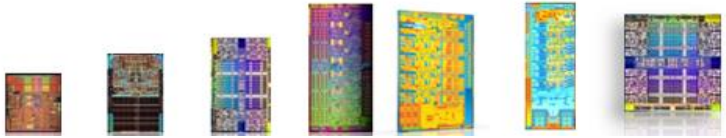
# INTEL® DATA ANALYTICS ACCELERATION LIBRARY

# DATA ANALYSIS AND MACHINE LEARNING




Pandas
Spark
HPAT

SciKit-Learn
Spark
DL-frameworks
daal4py



*more nodes, more cores, more threads, wider vectors, …*

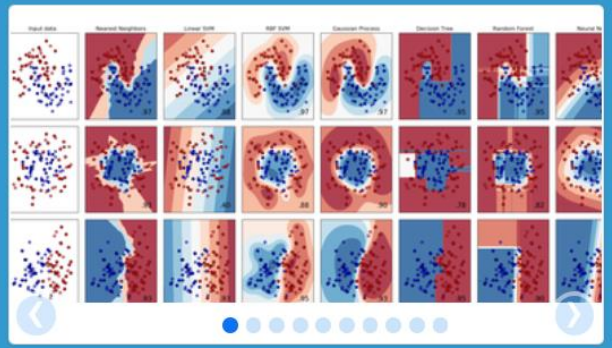
Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

# THE MOST POPULAR ML PACKAGE FOR PYTHON*

scikit learn

Home | Installation | Documentation | Examples

Google Custom Search | Search



## scikit-learn

*Machine Learning in Python*

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

### Classification

Identifying to which category an object belongs to.

**Applications**: Spam detection, Image recognition.
**Algorithms**: SVM, nearest neighbors, random forest, … — Examples

### Regression

Predicting a continuous-valued attribute associated with an object.

**Applications**: Drug response, Stock prices.
**Algorithms**: SVR, ridge regression, Lasso, … — Examples

### Clustering

Automatic grouping of similar objects into sets.

**Applications**: Customer segmentation, Grouping experiment outcomes
**Algorithms**: k-Means, spectral clustering, mean-shift, … — Examples

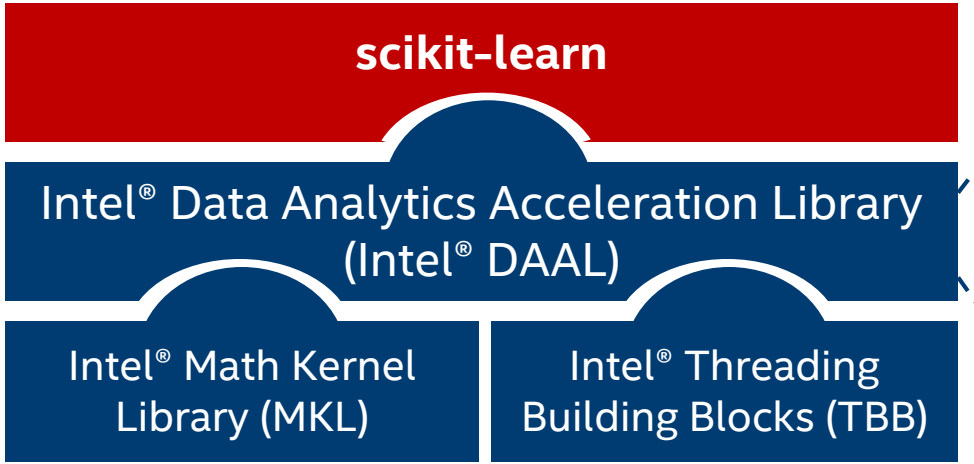
Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

# DAAL4PY: ACCELERATED ANALYTICS TOOLS FOR DATA SCIENTISTS

- Package created to address the needs of **Data Scientists and Framework Designers** to harness the **Intel® Data Analytics Acceleration Library (DAAL)** with a **Pythonic API**
- Pandas compatible, **one-liner API** for accessing many hardware accelerated **Machine Learning and Analytics functions**
- Powers our ***Scikit-Learn* accelerations*** in our shipped version of the package
- **Extends capabilities** past Scikit-learn by providing **scaling and distributed modes**

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

# ACCELERATING MACHINE LEARNING

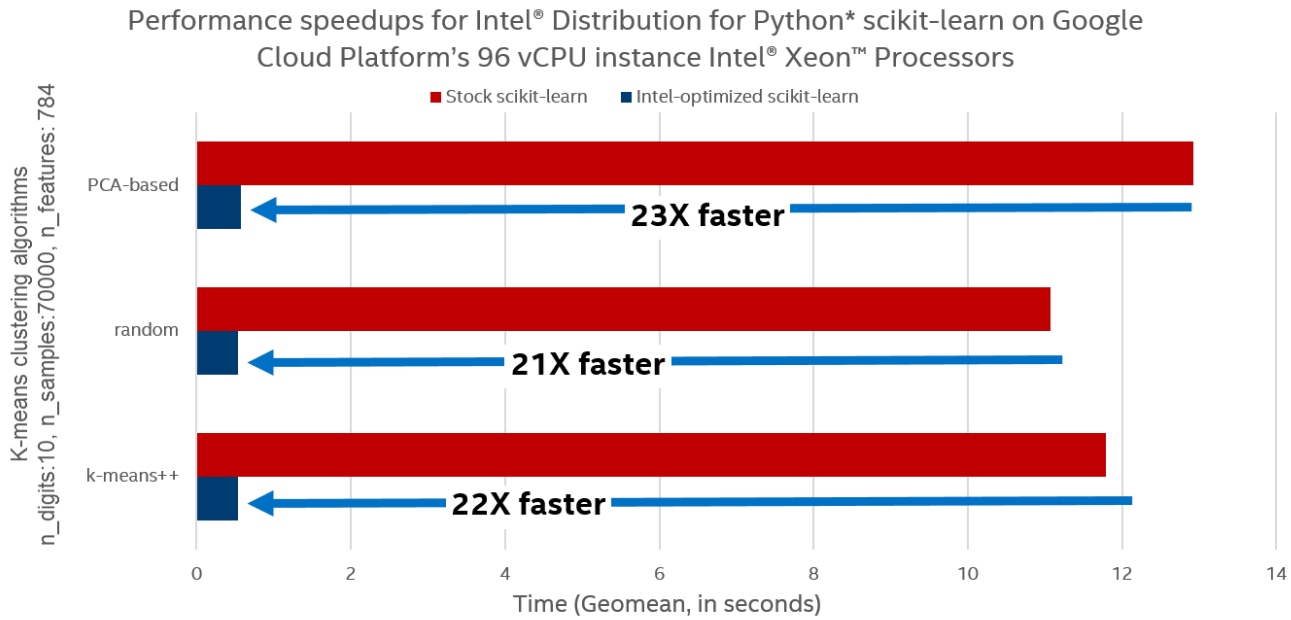



- **Efficient memory layout** via `Numeric Tables`
- **Blocking** for optimal cache performance
- **Computation mapped to most efficient matrix operations** (in MKL)
- **Parallelization** via TBB
- **Vectorization**

Try it out! `conda install -c intel scikit-learn`

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

# ACCELERATING K-MEANS




**System Configuration:** GCP VM, zone us-central1-c; 96 vCPU, Intel Skylake; 360 GB memory. Ubuntu 16.04.3 LTS; Linux instance-1 4.10.0-38-generic #42~16.04.1-Ubuntu SMP Tue Oct 10 16:32:20 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux; Intel® Distribution for Python* from Docker image intelpython/intelpython3_full:latest (created 2017-09-12T20:10:42.862965559Z); Stock Python*: pip install scikit-learn

https://cloudplatform.googleblog.com/2017/11/Intel-performance-libraries-and-python-distribution-enhance-performance-and-scaling-of-Intel-Xeon-Scalable-processors-on-GCP.html

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

# DAAL4PY

| | |
|---|---|
| Fast & Scalable | • Close to native performance through Intel® DAAL<br>• Efficient MPI scale-out<br>• Streaming |
| Easy to use | • Known usage model<br>• Picklable |
| Flexible | • Object model separating concerns<br>• Plugs into scikit-learn<br>• Plugs into HPAT |
| Open | • Open source: https://github.com/IntelPython/daal4py |

https://intelpython.github.io/daal4py/

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

# ACCELERATING SCIKIT-LEARN THROUGH DAAL4PY

Scikit-Learn **Equivalents**

Scikit-Learn **API** Compatible

daal4py

Intel® DAAL

```
> python -m daal4py <your-scikit-learn-script>
```

Monkey-patch any scikit-learn on the command-line

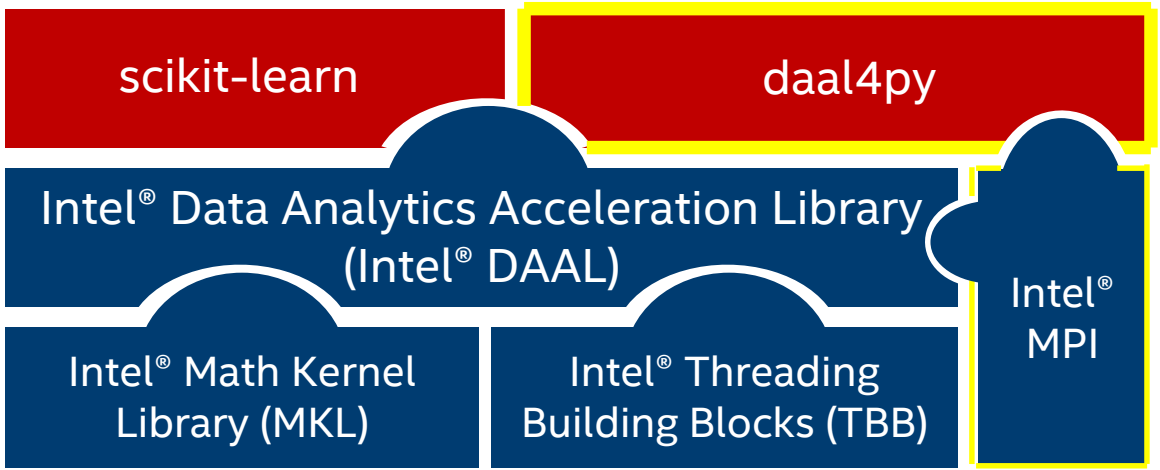
```
import daal4py.sklearn
daal4py.sklearn.patch_sklearn()
```

Monkey-patch any scikit-learn programmatically

https://intelpython.github.io/daal4py/

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

# SCALING MACHINE LEARNING BEYOND A SINGLE NODE




Simple Python API
Powers scikit-learn

Powered by DAAL

Scalable to multiple nodes

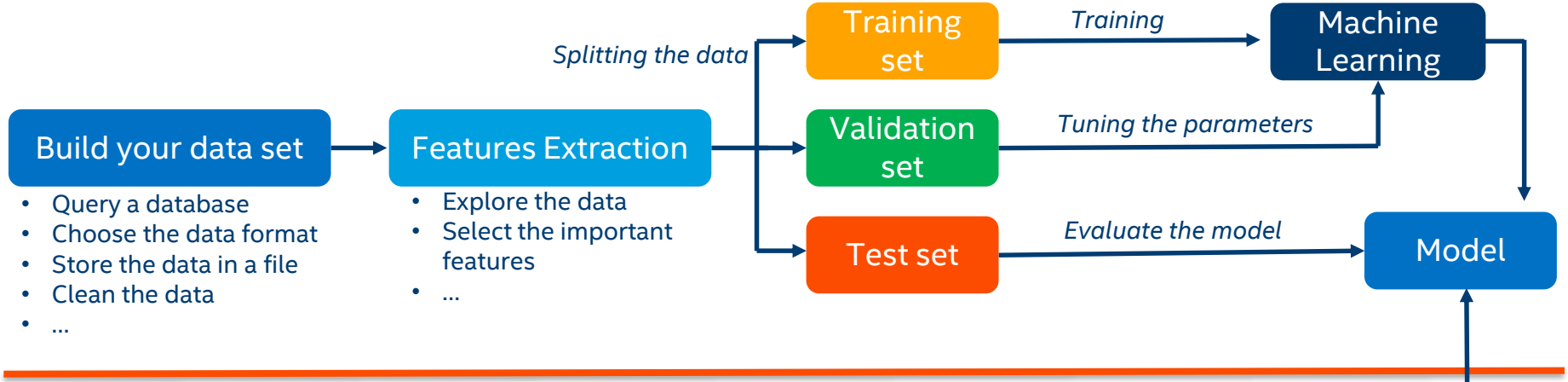
Try it out! `conda install -c intel daal4py`

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Software

# MACHINE LEARNING ALGORITHM: K-MEANS

# MACHINE LEARNING WORKFLOW

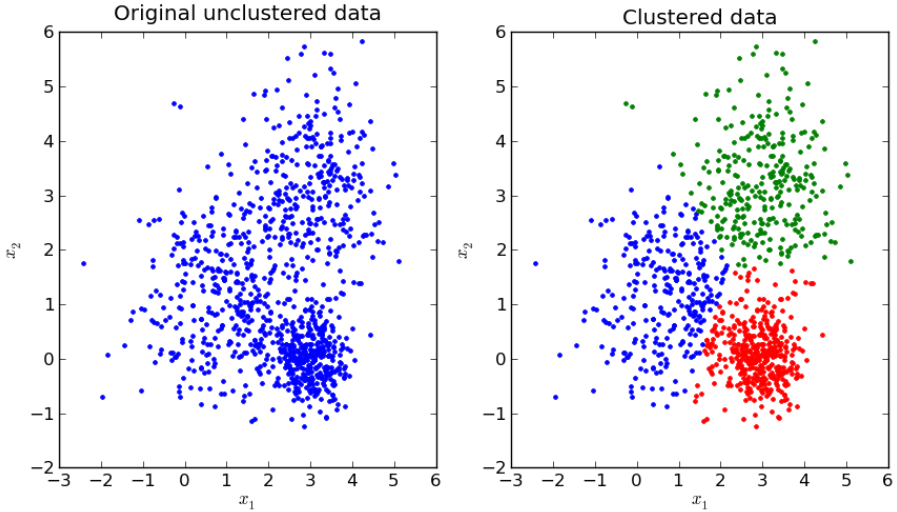



## INFERENCE AND DEPLOYMENT

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

# K-MEANS ALGORITHM

- K-means is a clustering method
- It is used to group objects in a (high-dimensional) sample
- K-means is based on centroids




Interesting use cases for k-means

- Document classification
  - Similarity identification
- Customer segmentation
  - Identifying patters of interest areas
- Insurance fraud detection
  - Isolate new claims based on proximity to past fraud

source: https://dzone.com/articles/10-interesting-use-cases-for-the-k-means-algorithm

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

# K-MEANS ALGORITHM DESCRIPTION

1. k (number of clusters in the dataset) is an external free parameter
2. k random objects are associated with initial centroids
3. each object of the dataset is assigned to form a cluster with its closest centroid
4. new centroids are computed by taking the average position of the objects in a given cluster
5. Evaluate convergence condition
6. Output: centroids location and association of the objects

Pros and cons:

- Simple and relatively robust

- Finding the best k is not trivial
- Results might depend on initial centroids
- Sensitive to outliers and to features with different dynamical range → data preparation

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

# K-MEANS EXAMPLE: COLOR QUANTIZATION

- An interesting application of K-means is to reduce the number of colors in a figure, while preserving the overall quality
- Initial data: every pixel has three color channels (RGB) expressed with 8 bit (0…255)
- For comparison, a clustering based on randomly picked colors will be shown

Original image (96,615 colors)

The image of the Summer Palace (China) is among the sample data contained on SciKit-Learn

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

# HANDS-ON

# K-MEANS EXAMPLE: COLOR QUANTIZATION

`kmeans.ipynb`

The hands-on is based on six main steps:

1) Data preparation
2) Computing the cluster centers
3) Labelling the data
4) From scikit-learn to daal4py: command line
5) From scikit-learn to daal4py: monkey-patch in the script
6) **K-means with daal4py: kmeans-daal4py.py**

Optimization Notice
Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

# Legal Disclaimer & Optimization Notice

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Copyright © 2019, Intel Corporation. All rights reserved. Intel, the Intel logo, Pentium, Xeon, Core, VTune, OpenVINO, Cilk, are trademarks of Intel Corporation or its subsidiaries in the U.S. and other countries.

**Optimization Notice**

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804


Copyright © 2019, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

# BACKUP

intel®

Software