Name of Event

# Intro to Intel Extensions of Scikit-learn to Accelerate Machine Learning Frameworks

Bob Chesebrough
AI Software Solutions Engineer

intel.

# Learning Objectives

- At the end of the webinar you will be able to:
  - Describe the value of the Intel AI Analytics Toolkit
  - Describe the value of one component of the library called Intel Extensions for Scikit-learn*
  - Where to get the toolkit
  - Identify classification, regression, clustering, and dimensionality reduction algorithms powered by AI Analytics Toolkit
  - Describe the application of a few lines of code to enable these optimizations
  - Stretch Goal: Describe Compute Follows Data methodology for application of Scikit-learn to Intel GPUs

# oneAPI

One Programming Model for Multiple Architectures and Vendors

HPC wire 2021 Readers' Choice Awards
Best HPC Programming Tool or Technology

### Freedom to Make Your Best Choice

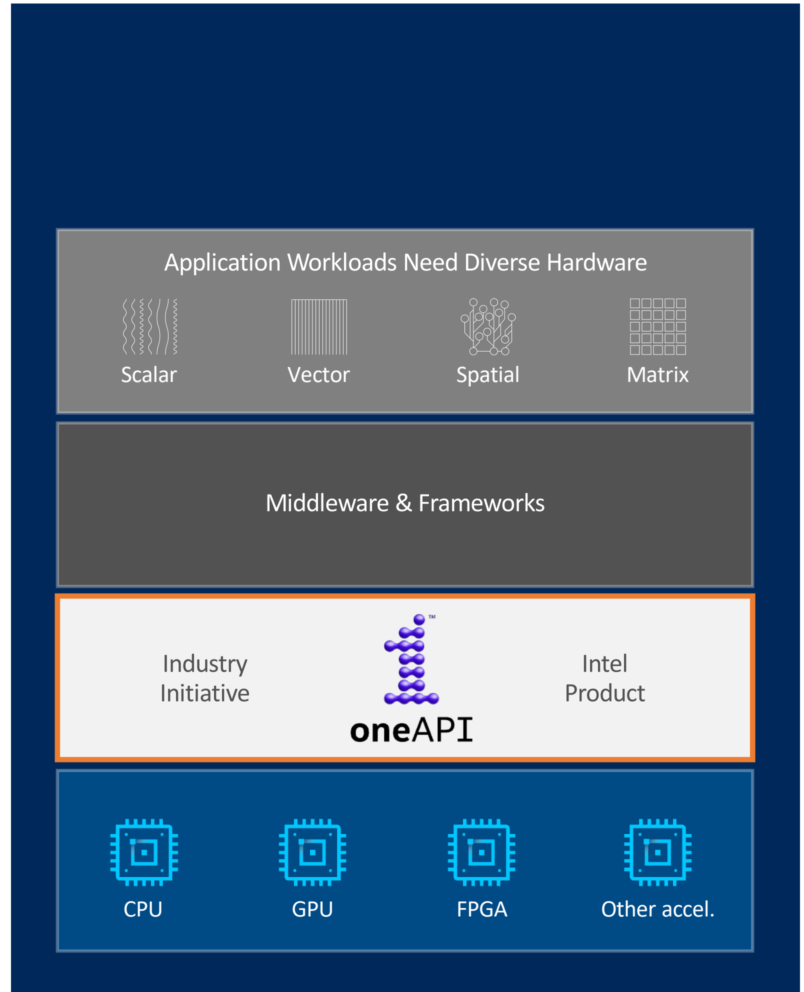
- Choose the best accelerated technology the software doesn't decide for you

### Realize all the Hardware Value

- Performance across CPU, GPUs, FPGAs, and other accelerators

### Develop & Deploy Software with Peace of Mind

- Open industry standards provide a safe, clear path to the future
- Compatible with existing languages and programming models including C, C++, Python, SYCL, OpenMP, Fortran, and MPI



# Intel® oneAPI Toolkits

A complete set of proven developer tools expanded from CPU to Accelerators

oneAPI

## Intel® oneAPI Base Toolkit

A core set of high-performance libraries and tools for building C++, SYCL and Python applications

## Add-on Domain-specific Toolkits

**Intel® oneAPI Tools for HPC**

Deliver fast Fortran, OpenMP & MPI applications that scale

**Intel® oneAPI Tools for IoT**

Build efficient, reliable solutions that run at network's edge

**Intel® AI Analytics Toolkit**

Accelerate machine learning & data science pipelines end-to-end with optimized DL frameworks & high-performing Python libraries

**Intel® oneAPI Rendering Toolkit**

Create performant, high-fidelity visualization applications

## Toolkit powered by oneAPI

OpenVINO

**Intel® Distribution of OpenVINO™ Toolkit**

Deploy high performance inference & applications from edge to cloud

Latest version available 2022.1

# Intel's oneAPI Ecosystem

## Built on Intel's Rich Heritage of CPU Tools Expanded to XPUs

**oneAPI**

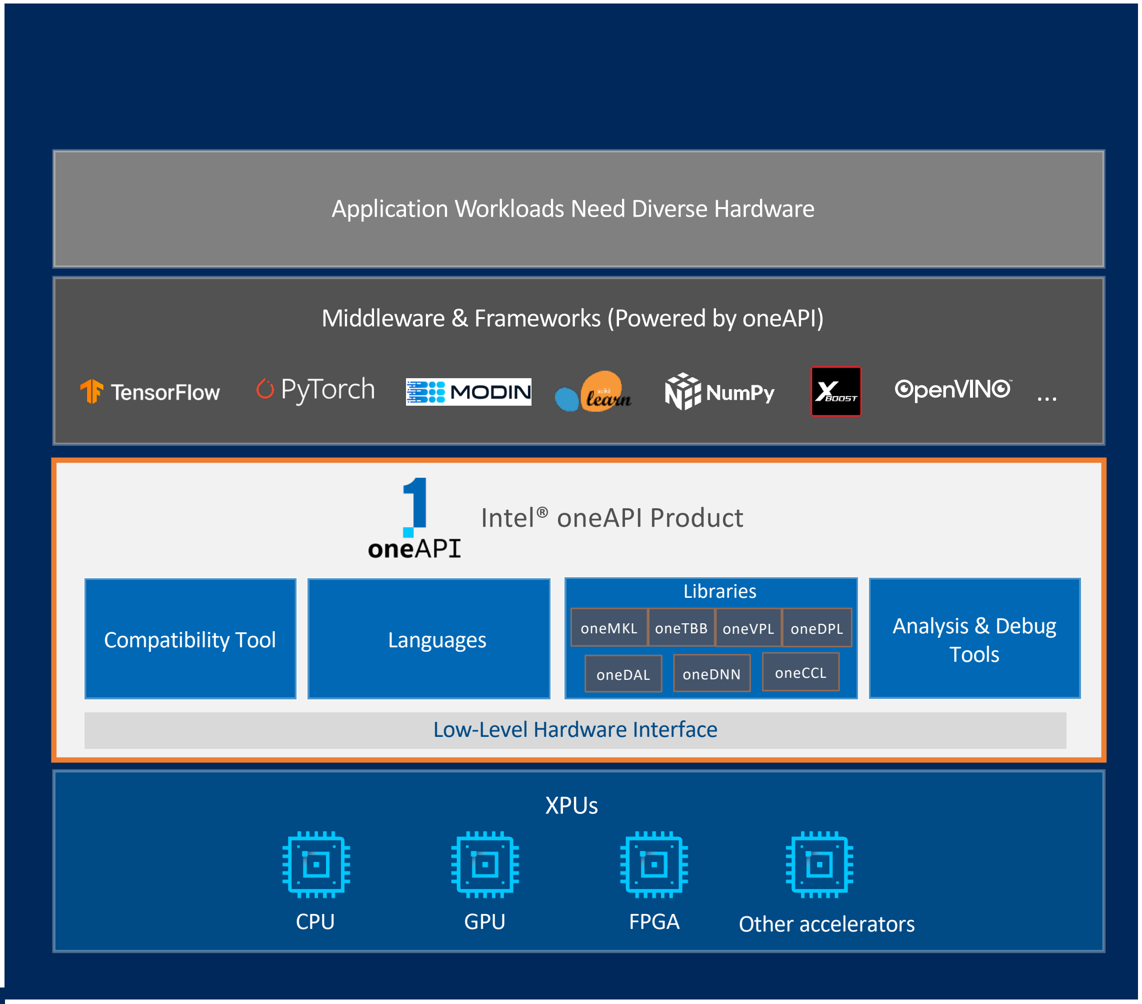
A cross-architecture language based on SYCL standards

Powerful libraries designed for acceleration of domain-specific functions

A complete set of advanced compilers, libraries, and porting, analysis and debugger tools

**Powered by oneAPI**

Frameworks and middleware that are built using one or more of the oneAPI industry specification elements, the SYCL language, and libraries listed on oneapi.com.




[Available Now]

Visit software.intel.com/oneapi for more details
Some capabilities may differ per architecture and custom-tuning will still be required. Other accelerators to be supported in the future.

# Intel® oneAPI AI Analytics Toolkit

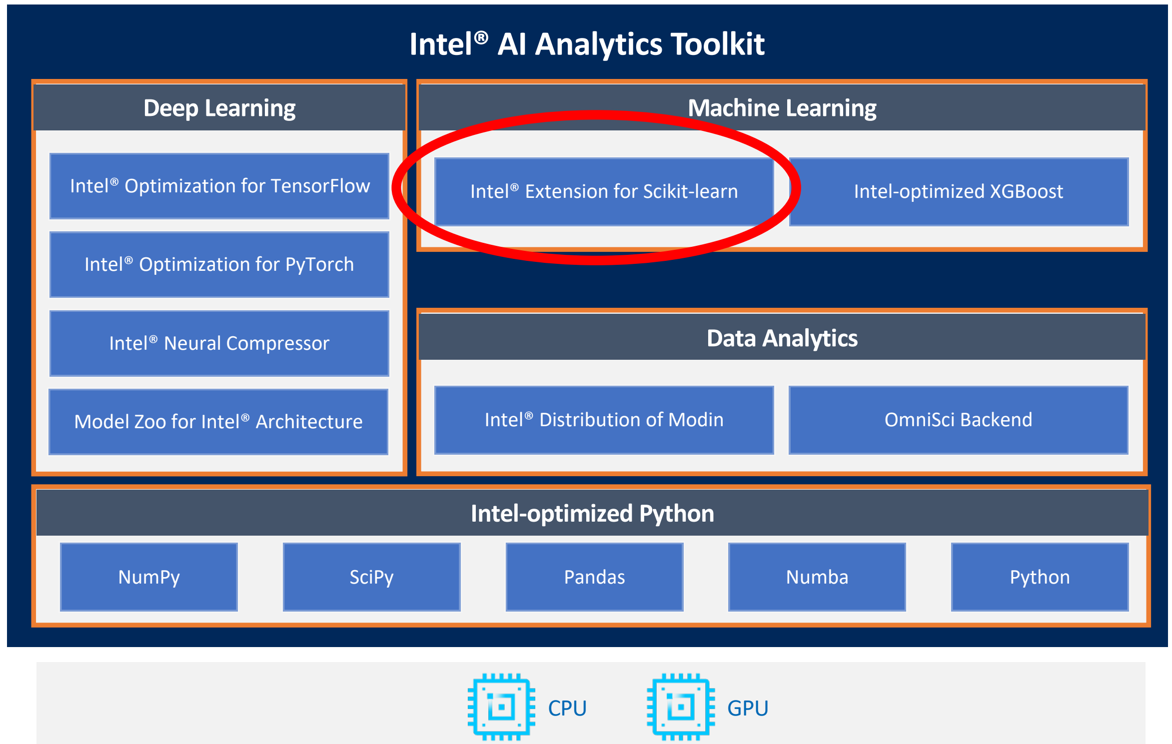
Accelerate end-to-end AI and data analytics pipelines with libraries optimized for Intel® architectures

## Who Uses It?

Data scientists, AI researchers, ML and DL developers, AI application developers

## Top Features/Benefits

- Deep learning performance for training and inference with Intel optimized DL frameworks and tools
- Drop-in acceleration for data analytics and machine learning workflows with compute-intensive Python packages




Hardware support varies by individual tool. Architecture support will be expanded over time.

Get the Toolkit HERE or via these locations

Intel Installer | Docker | Apt, Yum | Conda | Intel® DevCloud

Learn More: software.intel.com/oneapi/ai-kit

Back to Domain-specific Toolkits for Specialized Workloads

# Acquiring oneAPI AI Analytics Toolkit
# or Intel Extensions for scikit-learn specifically

If you just want the Intel Extensions for scikit-learn for your system (without the other library components):

There are multiple ways: visit link below

- https://www.intel.com/content/www/us/en/developer/articles/guide/intel-extension-for-scikit-learn-getting-started.html

Summary – installation by: Conda, pip, docker container, …

Comprehensive Library:
https://www.intel.com/content/www/us/en/developer/tools/oneapi/ai-analytics-toolkit-download.html

# Motivation

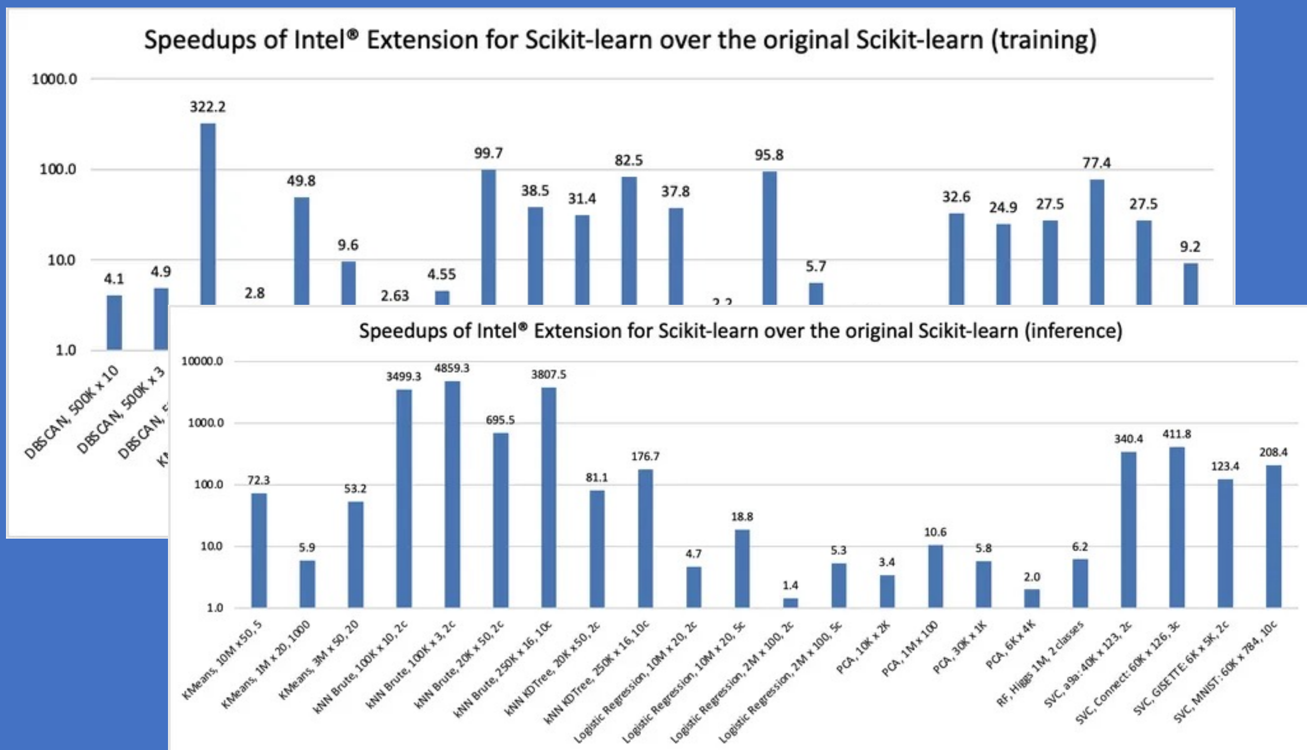
- Achieve acceleration on Intel CPU & current and future GPU
- Be ready for future innovations from Intel
- Can be achieved with a few lines of code

# Procedure for Ignition!

- Intel CPU: for Intel current and future CPUs
- - Apply import patch for  Intel Optimized function
- Intel GPU: for Intel current and future GPUs
- - Apply import patch for  Intel Optimized function
- - Apply Compute Follows Data method to cast NumPy arrays to tensor

# What’s in this for the a Developer?

- Seamless way to speed up your Scikit-learn application.




- 23 commonly used machine learning algorithms have been accelerated
- We will explore patching to enable these optimized algorithms
- In Hands on labs you will see, and experience dramatic speedups using several of these

# Gallery of Algorithms

# Gallery of Algorithms Optimized for Intel CPU

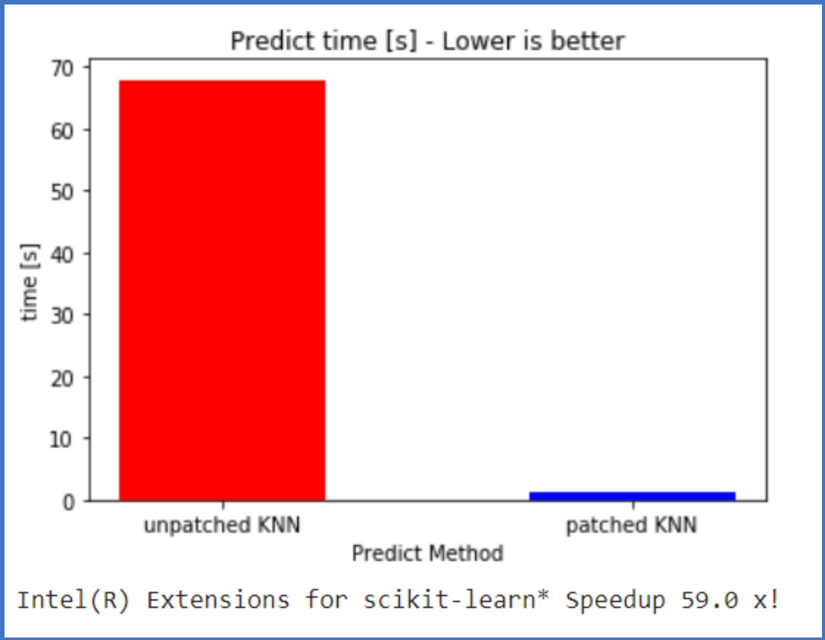
| Name | | | |
|---|---|---|---|
| DBSCAN | Linear Regression | K Nearest Neighbor Classifier | K Nearest Neighbor Regressor |
| KMeans | Elasticnet Regression (L1 & L2) | Random Forest Classifier | Assert All Finite |
| Nearest Neighbor (Unsupervised) | Lasso Regression (L1) | Random Forest Regressor | ROC AUC score |
| Principal Component Analysis (PCA) | Ridge Regression (L2) | Support Vector Classifier | Train Test Split |
| tSNE | Logistic Regression | Support Vector Regressor | Pair Wise Distance |

# Gallery of Algorithms Optimized for Intel GPU

| Name | | |
|---|---|---|
| Linear Regression | Logistic Regression | DBSCAN |
| KNeighborsRegressor | Support Vector Classifier | KMeans |
| Random Forest Regressor | K Nearest Neighbor Classifier | Principal Component Analysis (PCA) |
| | Random Forest Classifier | |

# Code & Results

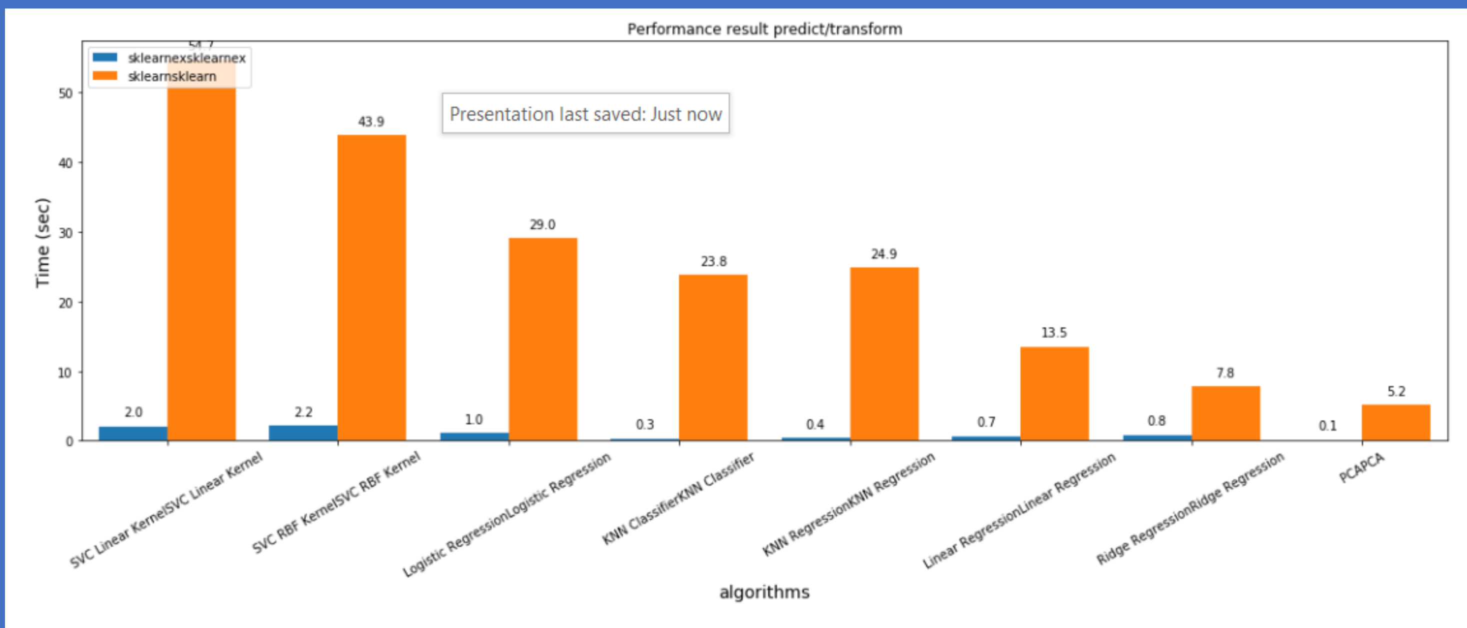
# Some results: recent run




KNN acceleration results we expect to see in next months workshop on the Intel DevCloud.

This is a comparison of stock library to Intel Extensions for Scikit-learn*

# A few more results you will generate in next months workshop!



# Patching

# Introduction to patching

- Intel® Extension for Scikit-learn* provides a way to accelerate existing scikit-learn code.
- In code, we will **import sklearnex** – this is the python library name for Intel Extensions for Scikit-learn*
- Via patching: replacing the stock scikit-learn algorithms with their optimized versions provided by the extension.
- You may enable patching in different ways:
- Without editing the code: using a command line flag
- Within code: using an import and a function call
- Un-patching: using methods to follow

# Patching Alternatives

- Command line:

```
python -m sklearnex my_application.py
```

- Inside script or Jupyter Notebook:

```
from sklearnex import patch_sklearn
patch_sklearn()
```

# Patching Alternatives

- Unpatching is similar:

```
from sklearnex import unpatch_sklearn
unpatch_sklearn()
```

# Patching Alternatives

- Patching or patching or unpatch specified functions surgically:

```
from sklearnex import patch_sklearn
patch_sklearn(“SVC”)
patch_sklearn([“SVC”, “PCA”])
```

```
from sklearnex import unpatch_sklearn
unpatch_sklearn(“SVC”)
```

- Getting the list of optimized functions:

```
from sklearnex import get_patch_names
get_patch_names()
```

For up to date info
[Supported Functions and Parameters]

intel

# Patching and imports: The Order

Import sklearnex

Apply “monkey patch” BEFORE

Import desired sklearn algorithms AFTER the patch

```
from sklearnex import patch_sklearn
patch_sklearn() # apply BEFORE import of targets
```

```
from sklearn.model_selection import train_test_split
```

- The import order is very important!
- Patch BEFORE you import the targeted scikit-learn* library!

# Train and Test Split: The Syntax

Import sklearnex

Apply "monkey patch"

Import desired sklearn algorithms AFTER the patch

```
from sklearnex import patch_sklearn
patch_sklearn() # apply BEFORE import of targets
```

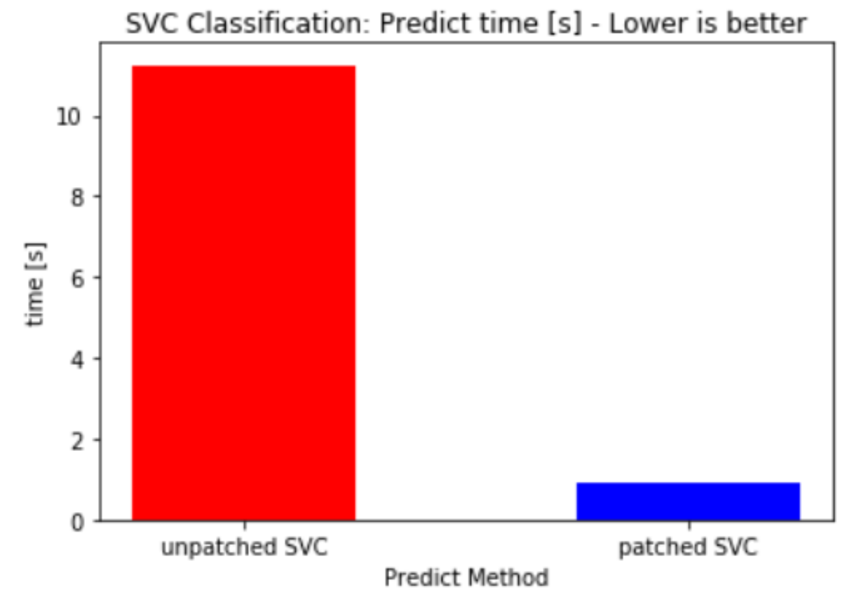
```
from sklearn.model_selection import train_test_split
```

```
# Split the data and put 30% into the test set
X_train, X_test, y_train, y_test = train_test_split(
... X, y, test_size=0.3)
```

# Example Code for SVC on CPU

```
'SVC Linear Kernel': {
    "algorithm": {
        'estimator': sklearn.svm.SVC,
        'params': {
            'C': 1.0,
            'kernel': 'linear',
        }
    },
    "data": {
        'generator': sklearn.datasets.make_classification,
        'params':
        {
            'n_samples': 20000,
            'n_features': 30,
            'n_classes': 3,
            'n_informative': 3,
            'random_state': 43,
        }
    }
```

```
patch_sklearn()
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3)
start_time = time.time()
knn = SVC(**Aparams).fit(x_train, y_train)
predicted = knn.predict(x_test)
patched_time = time.time() - start_time
```




Intel(R) Extensions for scikit-learn* Speedup 12.3 x!

Optimization Notice

# Target Intel GPU

- The intent is to demonstrate “HOW” to target Intel GPU using Intel Extensions for Scikit-learn*
- The intention today is NOT to demonstrate GPU performance

# Compute Follows Data

- Using a support library, we cast our data to device tensor, then send it to the device
- Any enabled sklearn methods involving that tensor are computed on the device, typically during call to fit()
- Results are returned to the host
- This is Compute Follows Data

# dpctl library

- dpctl is the library used to control access to the compute devices
- Import dpctl
- Getting a list of available devices :

```python
import dpctl
for d in dpctl.get_devices():
    d.print_device_info()
```

```
Name              Intel(R) Xeon(R) E-2176G CPU @ 3.70GHz
Driver version    2021.13.11.0.23_160000
Vendor            Intel(R) Corporation
Profile           FULL_PROFILE
Filter string     opencl:cpu:0

Name              Intel(R) UHD Graphics P630 [0x3e96]
Driver version    1.2.21786
Vendor            Intel(R) Corporation
Profile           FULL_PROFILE
Filter string     level_zero:gpu:0
```

# Device Queue

- Get queue of all devices available
- Pick the one you wish to target: in this case gpu_device

```
for d in dpctl.get_devices():
    gpu_available = False
    for d in dpctl.get_devices():
        if d.is_gpu:
            gpu_device = dpctl.select_gpu_device()
            gpu_available = True
        else:
            cpu_device = dpctl.select_cpu_device()
```

# Apply patch to sklearn

```
from sklearnex import patch_sklearn
patch_sklearn()

from sklearn.cluster import DBSCAN
```

# Prepare data

- Cast Numpy arrays to dpctl tensor
- Supply tensor to sklearn fit ("Compute Follow Data")

```
x_device = dpctl.tensor.asarray(X,
             usm_type = 'device', device = "gpu")

kmeans = KMeans(n_clusters=2, init='random', random_state=0).fit(x_device)
```

# Casting

- Cast NumPy array FROM_NUMPY or asarray to dpctl. Tensor

```
x_train_device = dpctl.tensor.asarray(x_train, usm_type = 'device', device = "gpu")
y_train_device = dpctl.tensor.asarray(y_train, usm_type = 'device', device = "gpu")
```

- If tensor is returned from sklearn, then cast it TO_NUMPY

```
clf = RandomForestClassifier(random_state=0).fit(x_train_device, y_train_device)
predictedGPU = clf.predict(x_test_device) #Predict on GPU
predictedGPUNumpy = dpctl.tensor.to_numpy(predictedGPU)
```

# When to cast data Returned from the device?

Pay attention to **return** types from:

- **fit** - many cases in scikit-learn, fit returns selfobject do NOT cast
- **fit_predict** - returns **ndarray** requires casting after the call on host (to_numpy)
- **predict** - returns **ndarray** requires casting after the call on host (to_numpy)
- **fit_transform** - returns returns **ndarray** requires casting after the call on host (to_numpy)
- **tranform** - typically returns **ndarray** requires casting after the call on host (to_numpy)

```
clf = RandomForestClassifier(random_state=0).fit(x_train_device, y_train_device)
predictedGPU = clf.predict(x_test_device) #Predict on GPU
#predictedCPU = clf.predict(x_test) #Predict on CPU
predictedGPUNumpy = dpctl.tensor.to_numpy(predictedGPU)
```

# DevCloud Access

- To run on the Intel DevCloud:
- Setting up an account is quick and easy. Enroll here: https://devcloud.intel.com/oneapi/get_started/
- For use on Intel® DevCloud, there are environment already configured for you:
  Simply launch a Jupyter* Notebook using the Python* 3.7 (Intel® oneAPI) Icon


- For play outside of DevCloud, instructions are provided detailing how to acquire the library for your own system

# Access to Notebooks

## Repo:

- From DevCloud Terminal:
  git clone https://github.com/IntelSoftware/Machine-Learning-using-oneAPI.git

# Continue the series!

## Aurora Learning Paths: Intel Extensions of Scikit-learn to Accelerate Machine Learning Frameworks


https://www.alcf.anl.gov/aurora-learning-paths-intel-extensions-scikit-learn-accelerate-machine-learning-frameworks

## Give us feedback…

- Tell us what you thought of this webinar.
- Give us feedback on what topics you'd like to see in future webinars.

Webinar Survey

intel®

# System Info

Architecture: x86_64 CPU op-mode(s): 32-bit, 64-bit Byte Order: Little Endian Address sizes: 46 bits physical, 48 bits virtual CPU(s): 24 On-line CPU(s) list: 0-23 Thread(s) per core: 2 Core(s) per socket: 6 Socket(s): 2 NUMA node(s): 2 Vendor ID: GenuineIntel CPU family: 6 Model: 85 Model name: Intel(R) Xeon(R) Gold 6128 CPU @ 3.40GHz Stepping: 4 CPU MHz: 1200.254 CPU max MHz: 3700.0000 CPU min MHz: 1200.0000 BogoMIPS: 6800.00 Virtualization: VT-x L1d cache: 384 KiB L1i cache: 384 KiB L2 cache: 12 MiB L3 cache: 38.5 MiB NUMA node0 CPU(s): 0-5,12-17 NUMA node1 CPU(s): 6-11,18-23 Vulnerability Itlb multihit: KVM: Vulnerable Vulnerability L1tf: Mitigation; PTE Inversion Vulnerability Mds: Mitigation; Clear CPU buffers; SMT vulnerable Vulnerability Meltdown: Mitigation; PTI Vulnerability Spec store bypass: Mitigation; Speculative Store Bypass disabled v ia prctl and seccomp Vulnerability Spectre v1: Mitigation; usercopy/swapgs barriers and __user pointer sanitization Vulnerability Spectre v2: Mitigation; Full generic retpoline, IBPB condit ional, IBRS_FW, STIBP conditional, RSB filling Vulnerability Srbds: Not affected Vulnerability Tsx async abort: Mitigation; Clear CPU buffers; SMT vulnerable Flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtr r pge mca cmov pat pse36 clflush dts acpi mmx f xsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rd tscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperf mperf pni pclmulqdq dtes64 monitor ds_cpl vmx s mx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid d ca sse4_1 sse4_2 x2apic movbe popcnt tsc_deadli ne_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb cat_l3 cdp_l3 inv pcid_single pti ssbd mba ibrs ibpb stibp tpr_sh adow vnmi flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpci d rtm cqm mpx rdt_a avx512f avx512dq rdseed adx smap clflushopt clwb intel_pt avx512cd avx512b w avx512vl xsaveopt xsavec xgetbv1 xsaves cqm_l lc cqm_occup_llc cqm_mbm_total cqm_mbm_local dt herm ida arat pln pts hwp hwp_act_window hwp_ep p hwp_pkg_req pku ospke md_clear flush_l1d