Intel® oneAPI Data Analytics Library for Python* Examples

By Ying Hu , Published: 03/04/2020, Last Updated: 03/04/2020

Introduction

Intel® oneAPI Data Analytics Library (oneDAL) is a highly optimized data analytics and machine learning (ML) library. It provides C++, Java*, and Python* interfaces, as well as newly introduced Data Parallel C++ (DPC++) interface.

The library with Python interface is also called DAAL4py. It is available along with Intel® Distribution for Python* (IDP), which is part of Intel® oneAPI Base Toolkit and Intel® AI Analytics Toolkit, as well as available from open-source repository: DAAL4py GitHub* repository.

Intel® oneDAL DAAL4py Public Samples

In order to help users to understand the Intel® AI Analytics Toolkit product, we have officially published oneAPI DAAL4py samples at AIkit code-sample. For example:

- Intel oneAPI Data Analytics Library Hello World (daal4py)
- Intel oneAPI Data Analytics Library for Distributed Linear Regression Training and Prediction (daal4py)
- Intel oneAPI Data Analytics Library for Distributed K-Means Training and Prediction (daal4py)

View All Code Samples

Intel® DAAL4py Self-Owned Examples

In order to help users to understand the Intel DAAL4py product, we also provide a suit of examples to demonstrate the product functionality. For example:

- Decision Forest for Classification and Regression
- Gradient Boosted Trees (GBT) for Classification and Regression
- K-Means Clustering
- K-Nearest Neighbor (KNN)
- Logistic Regression
- Multinomial Naïve Bayes Classifier
- Support Vector Machines (SVM) with Linear and Radial Basis Function (RBF) Kernels
- Multiclass Classification Using a One-Against-One Strategy
- Principal Components Analysis (PCA)

- And more

These samples are available from both the Intel oneAPI Base Toolkit and Intel® AI Analytics Toolkit, as well as the DAAL4py GitHub* repository.

Intel DAAL4py-related Samples and Products

Thus, we have three DAAL4py-related samples and products at hand. See the table:

Product	Product Samples	Resources /Loca
Intel® AI Analytics Toolkit(beta)	AIKit Code Samples	https://github.com/intel/AiKit-code-sa
Intel oneAPI Base Toolkit (beta)	DAAL4py Self-owned Examples	<oneAPI INSTALLATION PATH> /Intelpyton/latest/pkgs/daal4py-xxx/ir
DAAL4py GitHub* Repository	DAAL4py Self-owned Examples in GitHub*	https://github.com/IntelPython/daal4p

In order to avoid confusion, we encourage you to use the matching product samples for a product.

Troubleshoot

If you try the samples in a different product other than the intended one, errors may arise like below,

Unexpected Error: TypeError: __cinit__() got an unexpected keyword argument 'votingMethod'

To run the DAAL4py Self-owned Example in GitHub* with Intel® oneAPI Base Toolkit product

```
#Set oneAPI environment:
source /opt/intel/inteloneapi/setvars.sh
#But using the DAAL4py Example from the DAAL4py public GitHub*
repository:
$git clone https://github.com/IntelPython/daal4py.git
$cd daal4py/examples
$ python decision_forest_classification_batch.py
Error Message
Traceback (most recent call last):
  File "decision_forest_classification_batch.py", line 76, in
<module>
    (train_result, predict_result, plabels) = main()
  File "decision_forest_classification_batch.py", line 62, in main

resultsToEvaluate="computeClassLabels|computeClassProbabilities",
votingMethod="unweighted")
  File "build/daal4py_cy.pyx", line 12663, in
_daal4py.decision_forest_classification_prediction.__cinit__
TypeError: __cinit__() got an unexpected keyword argument
```

Root Cause

We keep developing new algorithms and committing the latest changes to DAAL4py GitHub* repository. Not all of the latest changed are applicable to other products. Hence, it is wrong to check the DAAL4py GitHub* examples with the oneAPI product.

Resolution

Use matched sample with its intended product. For example, to use the self-owned DAAL4py examples with the Intel oneAPI Base Toolkit (beta):

```
$source /opt/intel/inteloneapi/setvars.sh
$ cd /opt/intel/inteloneapi/intelpython/latest/pkgs/daal4py-2021.1b4-py37ha68da19_4/info/test/examples/
$ python decision_forest_classification_batch.py
Variable importance results:
 [[78.0208408  79.09469066  0.27575422]]
......
OOB error:
 [[0.03299533]]
PASSED
```

Product and Performance Information

[1] 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