Intel® Cloud Optimization Modules for Azure*:
XGBoost* Kubeflow* Pipeline

The Intel® Cloud Optimization Modules for Azure bring together an efficient, scalable, and unified cloud-native reference architecture ready to implement for enterprise AI workloads running on the Azure cloud.

This reference solution provides an optimized training and inference Kubeflow Pipeline using XGBoost to predict the probability of a loan default. The module enables the use of Intel® optimizations for XGBoost and Intel® daal4py in a full end-to-end reference architecture. It also leverages secure and confidential computing nodes using Intel® Software Guard Extensions on an Azure Kubernetes Service (AKS) cluster. This sheet outlines some of the key components of the XGBoost Daal4py Kubeflow Pipeline on Microsoft Azure.

Deploy the Azure Confidential Computing Cluster with Intel® Software Guard Extensions (Intel® SGX)

I. Sign in to your account with the Azure CLI:

```
az login
```

II. Create an Azure Resource Group:

```
export RG=intel-sgx-loan-default-app
export LOC=westus
az group create -n $RG -l $LOC
```

III. Create an Azure Kubernetes Services (AKS) Cluster system node pool with the confidential computing add-on enabled:

```
export AKS=aks-intel-sgx-kubeflow

az aks create --name $AKS \
--resource-group $RG \
--node-count 1 \
--node-vm-size Standard_D4_v5 \
--enable-addons confcom \
--enable-managed-identity \
--generate-ssh-keys -l $LOC \
--load-balancer-sku standard
```

IV. Add the Intel SGX node pool to the cluster using an instance from the Azure DCSv3 series:

```
az az aks nodepool add \
--resource-group $RG \
--name intelsgx \
--cluster-name $AKS \
--node-vm-size Standard_DC4s_v3 \
--node-count 2 \
--labels intelvm=sgx
```

V. Get the cluster access credentials and merge them into your local `.kube/config` file:

```
az aks get-credentials -n $AKS -g $RG
```

Next Steps:

All Cloud Modules | GitHub Repo | DevMesh Discord

Intel® Cloud Optimization Modules for Microsoft Azure*:
XGBoost* Kubeflow* Pipeline

Set the Intel® SGX Node Selector in the Kubeflow Pipeline

I. To install Kubeflow on the AKS cluster, follow the instructions here.

II. Using the Python API for Kubernetes, add a **`nodeSelector`** that will look for an agent node from the **`intelsgx`** node pool to schedule the pods. Call the **`add_node_selector()`** method for each Pipeline task that you want to be scheduled on an Intel SGX node using the corresponding node label key value pair.

```
from kfp import kubernetes
kubernetes.add_node_selector(task = train_xgboost_model_op,
    label_key = 'intelvm', label_value = 'sgx')
```

XGBoost* dmlc XGBoost v1.x+

Optimizations for training and prediction on CPU are **upstreamed.**

Install the latest XGBoost with PyPi or Anaconda – newer versions have the most optimizations.

```
pip install xgboost
```

```
conda install xgboost -c conda-forge
```

Put data in XGBoost DMatrix:

```
dtrain = xgb.DMatrix(
X_train.values, y_train.values)
```

Train XGBoost model:

```
model = xgb.train(params = params,
dtrain = dtrain,
num_boost_round = 500)
```

Cheat Sheet | Docs | Medium Example

daal4py*

The Intel Daal4py from the oneAPI Data Analytics Library (oneDAL) can be used to speed up inference of the XGBoost model. Install the latest daal4py:

```
pip install daal4py
```

```
conda install daal4py -c conda-forge
```

Convert a model to daal4py format from XGBoost:

```
d4p_model =
d4p.get_gbt_model_from_xgboost(model)
```

For optimized inference:

```
prediction = (d4p.
gbt_classification_prediction(
nClasses, resultsToEvaluate)
.compute(data, model)
.probabilities[:,1]
```

GitHub Repo | Docs | Medium Example

scikit-learn*

Works with scikit-learn v0.22.x+

Install the extension, choosing from:

```
pip install scikit-learn-intelex
```

```
conda install scikit-learn-intelex
```

```
docker pull intel/intel-optimized-ml:scikit-learn
```

Activate the patch in your Python code:

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

Or run it without changing code:

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

With Intel® Extension for Scikit-learn*, you can accelerate up to 10-100x,[3] while conforming to scikit-learn APIs.

All it takes is **two lines of code**!

Documentation | Get Started | Cheat Sheet | Examples

Next Steps:

All Cloud Modules | GitHub Repo | DevMesh Discord