Intel® Cloud Optimization Module for AWS*:
XGBoost* on Kubernetes*

Building and deploying high-performance AI applications can be a challenging task that requires a significant amount of computing resources and expertise. Fortunately, modern technologies such as Kubernetes*, Docker*, Intel® daal4py, and XGBoost* make it easier to develop and deploy AI applications optimized for performance and scalability.

By using cloud services like Amazon Web Services* (AWS), developers can further streamline the process and take advantage of the flexible and scalable infrastructure provided by the cloud.

Kubernetes* Setup

To set up your Kubernetes cluster, you can follow the detailed steps outlined on the GitHub* Repo. There is an important YAML file called *cluster.yaml* that specifies the number of nodes as well as the type of node in AWS.

```
managedNodeGroups:
- name: "eks-cluster-loanDefault-mng"
  desiredCapacity: 3
  instanceType: "m6i.large"
```

Our example deploys with 3 nodes of 3rd Generation Intel® Xeon Scalable processors (Amazon Elastic Compute Cloud* (EC2) m6i.large instance). You can adjust the desiredCapacity and instanceType based on your needs for your ML application deployment.

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:

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

Train an XGBoost model:

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

Cheat Sheet | Docs | Medium Example

daal4py*

Speed up inference of the XGBoost model using daal4py, which is based on the Intel® oneAPI Data Analytics Library (oneDAL). 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

Next Steps:

All Cloud Modules | GitHub Repo | DevMesh Discord