Solution Brief

Cross-Industry Architecture
Intel® Cloud Optimization Modules

intel software

Intel® Cloud Optimization Modules for AWS*: XGBoost* on SageMaker*

Author: Eduardo Alvarez

Date: August 9, 2023

Build an accelerated model development SageMaker pipeline and Lambda Inference Endpoint with Intel Optimizations for XGBoost.

The **Intel Cloud Optimization Modules for AWS: XGBoost on SageMaker** is designed to facilitate training and deploying with AWS SageMaker, Lambda, and the Intel® AI Analytics Toolkit. This reference solution focuses on end-to-end customer churn. We use the daal4py library (Python* API) as part of the Intel® oneAPI Data Analytics Library to accelerate inference with an XGBoost* classifier. This module focuses on a step-by-step process of building a custom model training and inference image for SageMaker and integrating these assets into an end-to-end pipeline that leverages AWS Lambda and API Gateway to expose the trained model as an inference service.

Use it as a reference solution for:

- Data Processing with Scikit-Learn* Extension
- Building custom SageMaker Training and Inference Images with Intel® oneAPI Data Analytics Library
- Setup SageMaker Pipelines with Custom Images
- Accelerated XGBoost Inference Service with Lambda and API Gateway

Who needs it?

- Developers looking to customize SageMaker training and inference images with Intel AI Analytics Toolkit components to unlock the added benefit of leveraging Intel hardware-accelerated libraries.
- Developers building AI applications on Intel® Xeon® CPU platforms in the AWS Cloud with M5 instances.

What it does

The solution enables users to customize Amazon SageMaker pipelines by integrating Intel AI Analytics Toolkit's hardware-accelerated libraries, enhancing machine learning performance on Intel hardware in the AWS cloud.

XGBoost-Daal4py SageMaker Pipeline and Inference

- Provides a complete set of instructions, configurations, and scripts for building custom images and pipelines:
  - **XGBoost oneDAL container** is the main motivation behind this optimization module because it provides the components required for building custom training and inference images for SageMaker Pipelines.
  - **Lambda container** contains all components necessary for building custom AWS Lambda functions from docker images with Intel oneDAL optimizations and XGBoost.
  - **Pipeline code** these components follow the standard SageMaker template for training and deploying AI models. We include instructions for specifying compute instances that enable the software-level optimizations in the SageMaker and Lambda images.
- **Implements a complete, end-to-end machine learning pipeline to predict credit risk using an XGBoost model,** from data preprocessing to model inference with **Intel oneDAL** optimizations.
- **Leverages oneDAL** which uses the **Intel® Advanced Vector Extensions 512 (Intel® AVX-512) instruction** set to maximize gradient boosting performance on **Intel® Xeon® processors**.

Cloud Solution Architecture

The architecture involves building and registering images to Elastic Container Register (ECR) for the xgboost-daal4py app and the lambda inference handler. The xgboost-daal4py ECR URI is provided to the sagemaker pipeline script. When the pipeline is executed, it processes data, trains/validates a model, and deploys a valid model to the inference endpoint. The lambda image is used to build a lambda function, which handles raw data pre-processing in real-time and passes processed data to the SageMaker inference endpoint. Access to the inference endpoint is managed by the API Gateway service. The training and inference compute is on a 2nd Generation Intel Xeon Instance.

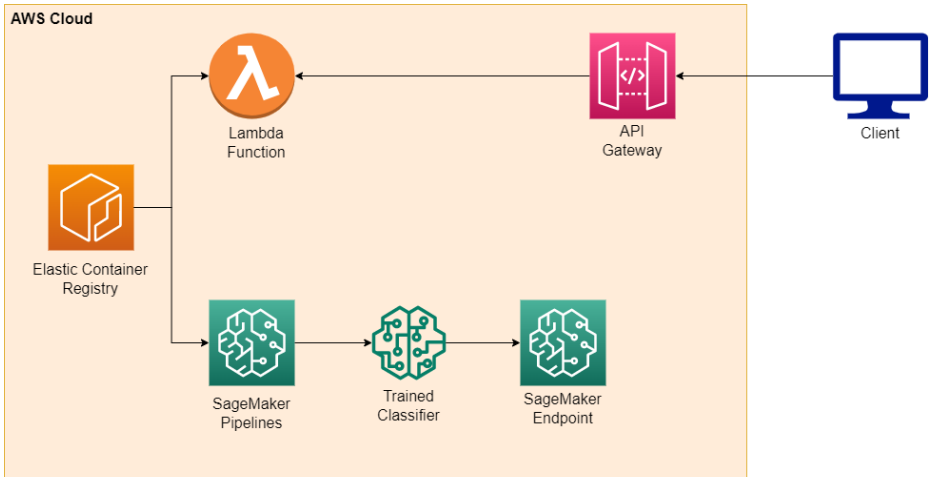

Figure 1 AWS Cloud Optimization Module Solution Architecture. Image by Author.

Solution Component Overview

This solution is based on the Customer Churn SageMaker Template. The SageMaker Pipeline is composed of 5 components:

- **CustomerChurnProcess:** preprocesses the data using scikit-learn, applying one-hot encoding and standard scaling to categorical and numerical features, respectively.
- **CustomerChurnTrain:** trains an XGBoost classifier and converts the model to daal4py format for optimized inference.
- **CustomerChurnEval:** evaluates the train model using a hold-out test dataset, producing an Area Under the Curve (AUC) metric.
- **CustomerChurnAccuracyCond:** evaluate model performance metric against AUC threshold. Models that do not pass accuracy condition are rejected by the pipeline.
- **CustomerChurnRegisterModel:** registers model in SageMaker model registry. Model is available for deployment once user approves the model for production or staging.

To build the custom images that support the pipeline and inference endpoint, we include scripts, configurations, and instructions. For a deep dive into these components please review the documentation on the GitHub code repository.

Code Highlights

Enable the Intel optimizations for XGBoost

XGBoost optimizations for training and inference on CPUs are upstreamed. Ensure you are using the latest version for the most Intel optimizations. The following code is implemented in the **predictor.py** python module. This module is packaged into the docker image that supports both our SageMaker model development pipeline and the flask model inference server. The AWS Lambda function, calls on the flask endpoints created by **predictor.py** when a POST request is received through the API Gateway.

Convert the trained XGBoost model to daal4py

Daal4py is the Python API of the oneAPI Data Analytics Library, oneDAL. Daal4py helps to further optimize model prediction, or inference, on CPUs. The following code is implemented in the **predictor.py** python module

```
# Scoring service class which handles model loading and conditional conversion to daal4py format.
class ScoringService(object):
    model = None  # Where we keep the model when it's loaded

    @classmethod
    def get_model(cls):
        """Get the model object for this instance, loading it if it's not already loaded."""
        if cls.model == None:
            with open(os.path.join(model_path, "xgboost-model"), "rb") as inp:
                cls.model = pickle.load(inp)
        return cls.model

    @classmethod
    def predict(cls, input, daal_opt=False):
        """Receives an input and conditionally optimizes xgboost model using daal4py conversion.
        Args:
            input (a pandas dataframe): The data on which to do the predictions. There will be
                one prediction per row in the dataframe"""

        clf = cls.get_model()

        #  Conditional conversion of XGBoost classifier to daal4py format.
        if daal_opt:
            daal_model = d4p.get_gbt_model_from_xgboost(clf.get_booster())
            return d4p.gbt_classification_prediction(nClasses=2, resultsToEvaluate='computeClassProbabilities',
fptype='float').compute(input, daal_model).probabilities[:,1]

        return clf.predict(input)
```

Using the SageMaker Python SDK to Specify Instances

The accelerations from oneDAL require that an appropriate Intel Xeon CPU instance is specified in the **pipeline.py** script. The instance that we specify is "ml.m5.xlarge" on-demand general purpose instance. This is a 4 vCPU 16GiB 2nd Generation Xeon Processor.

```
# Specifying instance type for estimator data preprocessing pipeline component.
 sklearn_processor = SKLearnProcessor(
         framework_version="0.23-1",
         instance_type="ml.m5.xlarge",
         instance_count=processing_instance_count,
         base_job_name=f"{base_job_prefix}/sklearn-CustomerChurn-preprocess",  # choose any name
         sagemaker_session=sagemaker_session,
         role=role)
```

```
# Specifying instance type for estimator training pipeline component.
 xgb_train = Estimator(
         image_uri=image_uri,
         instance_type= "ml.m5.xlarge",
         instance_count=1,
         output_path=model_path,
         base_job_name=f"{base_job_prefix}/CustomerChurn-train",
         sagemaker_session=sagemaker_session,
         role=role,
         use_spot_instances=True,
         max_run=300,
         max_wait=600
    )
```

```
# Specifying instance type for the SageMaker inference endpoint.
 step_register = RegisterModel(
        name="CustomerChurnRegisterModel",
        estimator=xgb_train,
        model_data=step_train.properties.ModelArtifacts.S3ModelArtifacts,
        content_types=["text/csv"],
        response_types=["text/csv"],
        inference_instances=["ml.m5.xlarge"],
        transform_instances=["ml.m5.xlarge"],
        model_package_group_name=model_package_group_name,
        approval_status=model_approval_status,
        model_metrics=model_metrics,
    )
```

Next Steps

Download the module from GitHub ›

Register for office hours to get help on your implementation ›

Check out the full suite of Intel Cloud Optimization Modules ›

Come chat with us on our DevHub Discord server to keep interacting with other developers ›