Steps for Deploying a prediction model on AWS Sagemaker and using Lambda function along with API Gateway to invoke and test it.

Create an Notebook instance from AWS SageMaker

A screenshot of a computer

Description automatically generated

Open Jupyter

You can upload all the files that you need like normal Jupyter notebook

A screenshot of a computer

Description automatically generated

Do data preparation, exploration, splitting etc.

(Check the notebook for exact steps for creating a sagemaker session and deploying the XGBoost model that was sourced from AWS ECR)

When you train the model, there is a training job created:

A screenshot of a computer

Description automatically generated

Additionally, if you try to get the importance features and other configuration from the model then a processing job is also created:

A screenshot of a computer

Description automatically generated

These are like logs and cannot be deleted.

Once the model is deployed you get an Model, Endpoint, Endpoint Configurations under Inference:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

To make this model accessible to outside via REST APIs:

Create a Lambda Function “PredictFromModel”:

A screenshot of a computer

Description automatically generated

Code for which is below:

*import json*

*import os*

*import boto3*

*import io*

*import csv*

*import numpy as np*

*#get environment variables*

*endpoint\_name = os.environ['ENDPOINT\_NAME']*

*runtime = boto3.client('runtime.sagemaker')*

*def lambda\_handler(event, context):*

*# TODO implement*

*print("Received event :" + json.dumps(event, indent = 2))*

*data = json.loads(json.dumps(event))*

*payload = data['data']*

*print(payload)*

*response = runtime.invoke\_endpoint(EndpointName = endpoint\_name,*

*ContentType = 'text/csv',*

*Body = payload)*

*print(response)*

*result = json.loads(response['Body'].read().decode())*

*print(result)*

*#result = result.decode("utf-8")*

*#result = result.split(',')*

*pred = round(np.expm1(result))*

*#pred = int(result['predictions'][0]['prediction\_label'])*

*return pred*

A screenshot of a computer

Description automatically generated

NOTE that we have used Numpy in this lambda function. However, we cannot directly use this here unless we add a layer to our function:

Scroll way down on the Lambda Function you created:

A screenshot of a computer

Description automatically generated

Layers> Add a Layer> Choose AWSSDKPandas..

This will automatically add pandas, numpy to be used in the lambda function

PS We were using numpy here because we log transformed our target variable before training and to get the correct predictions we had to convert them back to their original for using *np.expm1(result).* Therefore, we needed numpy in lambda function.

A screenshot of a computer

Description automatically generated

Also, we have created an environmental variable for the ENDPOINT\_NAME this was to configured in the Lambda Function:

A screenshot of a computer

Description automatically generated

Now we want this Lambda function to invoke endpoint (which is written in the code), however, we cannot pass the inputs to the model via only the Lambda function. So we will have to trigger this Lambda function using an API Gateway from where we can pass the inputs to the model.

API Gateway is like a bridge between Lambda Function and Sagemaker.

IMPORTANT: For Lambda Function to access both API Gateway and Sagemaker, the role that you created while creating the Lambda function should also have policies for API Gateway and Sagemaker.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Now go to API Gateway and create an REST API:

A screenshot of a computer

Description automatically generated

Give a name and description:

Also While creating the API, it will ask you if you want to connect it to a Lambda function( create the lambda function first otherwise this option wont pop up)

In that field, Copy the Lambda fun ARN

A screenshot of a computer

Description automatically generated

And paste it there.

A screenshot of a computer

Description automatically generated

Once the API is created > Actions> Create Method> POST

A screenshot of a computer

Description automatically generated

Click on Test>

A screenshot of a computer

Description automatically generated

Pass the payload (inputs for the model) and click on Test

A screenshot of a computer

Description automatically generated

This will trigger the lambda function > invoke\_endpoint > get the response

All the logs along with the response will be printed on the right side in the API gateway page:

A screenshot of a computer

Description automatically generated

Our prediction is highlighted in red.

In case of error, we can check the Monito Tab, of the Lambda function for CloudWatch logs, this is were the print statements get printed from our lambda function

