Operationalizing ML on AWS Sagemaker

Sagemaker notebook kernel choice

The ml.t3.medium kernel is an excellent choice because it balances cost and performance, which is essential. It has cheap but sufficient computing resources to run the algorithm effectively. It is also part of the latest generation of EC2 instances, which means it has the newest hardware and software features.

EC2 Instance choice

Since not much computing power is needed, I chose a t2.micro instance with AWS Linux OS for my EC2 instance.

EC2 Code

Training on an EC2 instance is similar to training in Sagemaker except that there is no hyperparameter tuning. Whereas Sagemaker trains models with the best hyperparameters from a training job, EC2 uses hard-coded values

---

Lambda function

The lambda_function.py takes an input event and context as parameters. The function then decodes the input event using base64 and sends the decoded data as a request to an Amazon SageMaker endpoint. The response from the endpoint is then parsed and returned as a JSON object, along with an HTTP status code and headers. The function also logs debug messages using the Python logging module.

Security

Generally, my account is safe because I am the only one that has access to it. Furthermore, execution roles are used with only minimum-privilege policies and are deleted after use.

Concurrency and Autoscaling

I configured my endpoint's autoscaling to kick in when invocations reach 100/s. With a scale-in of 3s and a scale-out time of 300s. I did this to minimize costs while also dealing with great traffic.
I also reserved three instances for my lambda function's concurrency. This was done with the intention of minimizing costs.