Build a ML Workflow For Scones Unlimited On Amazon SageMaker

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Hello Udacian,

Congratulations!
You've reached the end of the project. In this project, you created an event-driven ML workflow that can be incorporated into the Scones Unlimited production architecture. You used the SageMaker Estimator API to deploy your SageMaker Model and Endpoint, and you used AWS Lambda and Step Functions to orchestrate your ML workflow. Using SageMaker Model Monitor, you instrumented and observed your Endpoint, and at the end of the project, you built a visualization to help stakeholders understand the performance of the Endpoint over time. If you're up for it, you can even go further with these stretch goals:

- Extend your workflow to incorporate more classes: the CIFAR dataset includes other vehicles that Scones Unlimited can identify with this model.
- Modify your event-driven workflow: can you rewrite your Lambda functions so that the workflow can process multiple image inputs in parallel? Can the Step Function "fan out" to accomodate this new workflow?
- Consider the test data generator we provided for you. Can we use it to create a "dummy data" generator, to simulate a continuous stream of input data? Or a big paralell load of data?
- What if we want to get notified every time our step function errors out? Can we use the Step Functions visual editor in conjunction with a service like SNS to accomplish this? Try it out!

Extra Materials for your knowledge

Checkout the following material in your free time.

- Building, automating, managing, and scaling ML workflows using Amazon SageMaker Pipelines
- BUILDING AN ML WORKFLOW
- AWS re:Invent 2020: How to create fully automated ML workflows with Amazon SageMaker Pipelines

Train and Deploy a Machine Learning Model

✓

Setup a SageMaker studio and a kernel to run this project

- THIS SECTION MEETS SPECIFICATIONS

Brilliant work!

All SageMaker function calls are executed successfully. Well done. The starter code API calls to Sagemaker execute successfully, showing that you have configured Sagemaker with the appropriate permissions.👌

💡 Tips for setting up a SageMaker Studio and a kernel:

- Use a consistent naming convention for your notebooks and code. This will make it easier to find and manage your work.
- Use version control to track your changes. This will help you keep track of your progress and make it easier to revert to previous versions of your work.
  Document your work. This will help you remember what you did and why you did it. It will also be helpful for others who need to understand your work.

Extra Materials for your knowledge

Here is a nice workshop from the AWS official. I think it is quite helpful.

- Amazon SageMaker Workshop
- Best Practices for Improving Your Machine Learning and Deep Learning Models

Students have completed the ETL (extract, transform, load) section of the starter code.

- THIS SECTION MEETS SPECIFICATIONS

You correctly downloaded your training images, identified the objects with the correct label, and uploaded them to an S3 bucket. Well done!

- Training images are downloaded from the specified URL.✓
- Correctly identified the objects with correct label numbers.✓
- Images are successfully saved and uploaded to s3.✓

Extra Materials for your knowledge

- Define a Pipeline
  To orchestrate your workflows with Amazon SageMaker Model Building Pipelines, you need to generate a directed acyclic graph (DAG) in the form of a JSON pipeline definition.
- Get started with data ingestion

Students have successfully completed the Model training section up to "Getting ready to deploy", showing they trained an image classification model

- THIS SECTION MEETS SPECIFICATIONS

You did an excellent job! You created a metadata file, built an estimator, and then trained an image classification model appropriately.🥕

- Successfully construct parameters for the "image-classification" Estimator
- Created an estimator that uses one instance of `ml.p2.xlarge`
- Trained an image classification model without errors
- Did not use any special characters in names (S3 bucket/folders, model names etc.,)
- `image_shape` hyperparameter is `32x32x3` in that order
- `num_classes` hyperparameter is `2`

Extra Materials for your knowledge

- Recommendations for Reporting Machine Learning Analyses
  Research questions must first be clearly articulated to frame all subsequent choices regarding data preparation, method selection, results, and performance evaluation, as well as interpretation. When performing ML analyses, 2 assumptions are made the desired outputs of the data can be generated given the input data and the available data contain the necessary information to learn the desired output. It is important to keep these assumptions in mind when considering the input data, ML method, and overall analysis architecture that will be used to address the research question.

- **Students have successfully completed the "Getting ready to deploy" section, showing they have deployed a trained ML model**
- **Students have a unique model endpoint name printed in their notebook for use later in the project**
- **Successfully made predictions using a sample image**

- THIS SECTION MEETS SPECIFICATIONS

Nice work configuring the Model Monitor with DataCaptureConfig🤝

- Model endpoint name
- The model deployment was completed without any errors. 👏
- You deployed the model on a single `ml.m5.xlarge` instance with the data capture config.
- Predictions with a sample image have been equally made.

Extra Materials for your knowledge

- Top Machine Learning Research Papers Released
  Rebooting ACGAN: Auxiliary Classifier GANs with Stable Training
  The authors of this work examined why ACGAN training becomes unstable as the number of classes in the dataset grows. The researchers revealed that the unstable training occurs due to a gradient explosion problem caused by the unboundedness of the input feature vectors and the classifier's poor classification capabilities during the early training stage.

Build a full machine learning workflow

Students have authored three lambda functions.

- 1st lambda is responsible for return an object to step function as `image_data` in an event
- 2nd lambda is responsible for image classification
- 3rd lambada is responsible for filtering low-confidence inferences

Students have saved their code for each lambda function in a python script.

- THIS SECTION MEETS SPECIFICATIONS

This section meets specifications. You have successfully authored three lambda functions.👌

Extra Materials for your knowledge

- AWS Lambda is a Function-as-a-service (FaaS) computing platform provided by Amazon Web Services (AWS). As a FaaS, it provides a computing platform to execute code in the cloud. As in any serverless system, it abstracts away the complexities of provisioning and managing a cloud infrastructure. You can read more about this from this Introduction to AWS Lambda
- You may as well check out these Best practices for working with AWS Lambda functions.

Some best practices for using AWS Model Monitor:

- Choose the right monitoring metrics. Model Monitor provides a variety of metrics to monitor, such as accuracy, precision, recall, and F1 score. Choose the metrics that are most important for your application.
- Configure alerts. Model Monitor can send alerts when your model's performance degrades. Configure alerts so that you are notified as soon as possible of any problems.
- Use explainability features. Model Monitor can provide explanations for your model's predictions. This can help you to understand why your model is making the predictions that it is.
- Monitor your model in production. Once your model is in production, continue to monitor it. This will help you to ensure that your model is performing as expected.
- Build reusable, serverless inference functions for your Amazon SageMaker models using AWS Lambda layers and containers

✓

- Compose Lambdas together in a Step Function.
- Students will have a JSON export that defines the Step Function
- Students have a screenshot of the working Step Function.

- THIS SECTION MEETS SPECIFICATIONS

Perfect, you have composed the Step function and successfully executed it. I verified the JSON export that defines your Step Function.🙌

Step Function JSON export has the following:

- The step function file is valid JSON ✓
- Plugging the step function file content into the AWS editor produces a valid step function ✓
- The Step function sequences the Lambdas in the correct order ✓

Extra Materials for your knowledge

Please checkout this The Ultimate Guide to AWS Step Functions

- Best practices for Step Functions
- Best Practices for AWS Step Functions

Best practices for composing Lambdas together in a Step Function:

- Use a consistent naming convention for your Lambda functions and Step Functions. This will make it easier to keep track of your code and troubleshoot any problems.
- Use descriptive names for your states. This will make it easier to understand the flow of your Step Function.
- Use the waitForTaskToken property when invoking a Lambda function from a Step Function. This will ensure that the Step Function waits for the Lambda function to finish executing before continuing.
- Use error handling to gracefully handle errors in your Step Function. This will help to prevent your Step Function from failing.
- Use the retry and catch clauses to handle errors in your Lambda functions. This will help to ensure that your Lambda functions are resilient to errors.
- Use the timeout property to set a time limit for your Step Functions and Lambda functions. This will help to prevent your Step Functions and Lambda functions from running for too long.

Monitor the model for errors

✓

Students load the data from Model Monitor into their notebook

- THIS SECTION MEETS SPECIFICATIONS

Good work, you have provided steps to reterive model monitor data into the notebook. This data is important to track the model performance and report them using visualization techniques

You've imported the data from the data capture as part of the model monitor into the notebook correctly. Well done!💪

Extra Materials for your knowledge

- Monitor models for data and model quality, bias, and explainability
  Amazon SageMaker Model Monitor monitors the quality of Amazon SageMaker machine learning models in production. You can set up continuous monitoring with a real-time endpoint (or a batch transform job that runs regularly), or on-schedule monitoring for asynchronous batch transform jobs. With Model Monitor, you can set alerts that notify you when there are deviations in the model quality. Early and proactive detection of these deviations enables you to take corrective actions, such as retraining models, auditing upstream systems, or fixing quality issues without having to monitor models manually or build additional tooling.

✓

Students create their own visualization of the Model Monitor data outputs

- THIS SECTION MEETS SPECIFICATIONS

Good job with the custom visualization.👍

Extra Materials for your knowledge

📜 Best practices in documenting model refinement in a machine learning project:

Document the initial model. This includes the model architecture, the hyperparameters, and the training data.

- Document the steps taken to refine the model. This includes the changes made to the model architecture, the hyperparameters, and the training data.
- Document the results of the model refinement. This includes the accuracy, the precision, the recall, and the F1 score.
- Document the challenges faced during model refinement. This includes any problems with the data, the model, or the training process.
- Document the lessons learned from model refinement. This includes what worked well, what didn't work well, and what could be improved in the future.

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START