Google Discussions

EXAM PROFESSIONAL MACHINE LEARNING ENGINEER TOPIC 1 QUESTION 179 DISCUSSI...

Actual exam question from Google's Professional Machine Learning Engineer

Question #: 179

Topic #: 1

[All Professional Machine Learning Engineer Questions]

You recently used XGBoost to train a model in Python that will be used for online serving. Your model prediction service will be called by a backend service implemented in Golang running on a Google Kubernetes Engine (GKE) cluster. Your model requires pre and postprocessing steps. You need to implement the processing steps so that they run at serving time. You want to minimize code changes and infrastructure maintenance, and deploy your model into production as quickly as possible. What should you do?

A. Use FastAPI to implement an HTTP server. Create a Docker image that runs your HTTP server, and deploy it on your organization’s GKE cluster.

B. Use FastAPI to implement an HTTP server. Create a Docker image that runs your HTTP server, Upload the image to Vertex AI Model Registry and deploy it to a Vertex AI endpoint.

C. Use the Predictor interface to implement a custom prediction routine. Build the custom container, upload the container to Vertex AI Model Registry and deploy it to a Vertex AI endpoint.

D. Use the XGBoost prebuilt serving container when importing the trained model into Vertex AI. Deploy the model to a Vertex AI endpoint. Work with the backend engineers to implement the pre- and postprocessing steps in the Golang backend service.

Show Suggested Answer

by vale_76_na_xxx at *Jan. 8, 2024, 8:06 p.m.*

Comments

Comments

Type your comment...

Submit

ddogg Highly Voted 8 months, 3 weeks ago

Selected Answer: C

Use the Predictor interface to implement a custom prediction routine. This allows you to include the preprocessing and postprocessing steps in the same deployment package as your model.

Build the custom container, which packages your model and the associated preprocessing and postprocessing code together, simplifying deployment.

Upload the container to Vertex AI Model Registry. This makes your model available for deployment on Vertex AI.

Deploy it to a Vertex AI endpoint. This allows your model to be used for online serving.

https://blog.thecloudside.com/custom-predict-routines-in-vertex-ai-46a7473c95db

upvoted 6 times

Prakzz Most Recent 3 months, 3 weeks ago

Selected Answer: B

This approach minimizes code changes and infrastructure maintenance by leveraging Vertex AI's managed services for deployment. Implementing the preprocessing and postprocessing steps in a FastAPI server within a Docker container allows you to handle these steps at serving time efficiently. Deploying this Docker image to a Vertex AI endpoint simplifies the deployment process and reduces the burden of managing the infrastructure.

upvoted 1 times

AzureDP900 4 months ago

Option C is a good choice if
You have specific requirements for preprocessing or postprocessing that can't be met by the prebuilt XGBoost serving container.
You need more control over the deployment process or want to integrate with other services.
You're comfortable building and managing custom containers.
However, if you just want a simple, straightforward way to deploy your model as a RESTful API, Option D (using the XGBoost prebuilt serving container) might be a better fit!

upvoted 1 times

livewalk 4 months, 4 weeks ago

Selected Answer: B

FastAPI allows to create a lightweight HTTP server with minimal code.

upvoted 1 times

Yan_X 7 months, 1 week ago

Selected Answer: D

▯

Pre-built XGBoost container already includes pre- and postprocessing steps.

upvoted 1 times

guilhermebutzke 8 months, 2 weeks ago

Selected Answer: C

My answer C:

Considering pre- and postprocessing implementation, The option C directly deals with implementing the processing steps in a custom container, offering full control over their placement and execution.

This documentation says: “Custom prediction routines (CPR) lets you build [custom containers](https://cloud.google.com/vertex-ai/docs/predictions/use-custom-container) with pre/post processing code easily, without dealing with the details of setting up an HTTP server or building a container from scratch.”
https://cloud.google.com/vertex-ai/docs/predictions/custom-prediction-routines

So, it is better to use C instead of A or B. D is better because it offers the option of pre and post-processing, which is not available in D due to its use of prebuilt serving.

upvoted 2 times

36bdc1e 9 months, 1 week ago

C

. Build the custom
container, upload the container to Vertex AI Model Registry, and deploy it to a Vertex AI endpoint.
This option allows you to leverage the power and simplicity of Vertex AI to serve your XGBoost model
with minimal effort and customization. Vertex AI is a unified platform for building and deploying
machine learning solutions on Google Cloud. Vertex AI can deploy a trained XGBoost model to an online prediction endpoint,
which can provide low-latency predictions for individual instances. A custom prediction routine (CPR) is a Python script that
defines the logic for preprocessing the input data, running the prediction, and postprocessing the output data.

upvoted 4 times

pikachu007 9 months, 2 weeks ago

Selected Answer: D

Considering the goal of minimizing code changes, infrastructure maintenance, and quickly deploying the model into production, option D seems to be a pragmatic approach. It leverages the prebuilt XGBoost serving container in Vertex AI, providing a managed environment for serving. The pre- and postprocessing steps can be implemented in the Golang backend service, maintaining consistency with the existing Golang implementation and reducing the need for significant code changes.

upvoted 1 times

vale_76_na_xxx 9 months, 2 weeks ago

I would say D

upvoted 1 times

ExamTopics doesn't offer Real Microsoft Exam Questions. ExamTopics doesn't offer Real Amazon Exam Questions. ExamTopics Materials do not contain actual questions and answers from Cisco's Certification Exams.

CFA Institute does not endorse, promote or warrant the accuracy or quality of ExamTopics. CFA® and Chartered Financial Analyst® are registered trademarks owned by CFA Institute.