Shabna Nazar

FEB 2023

Section 2: ML Model Deployment in AZURE

Table of Contents

[1. Use Case Details 2](#_Toc127812106)

[2. Pre-requisites 2](#_Toc127812107)

[3. Steps to deploy the model in AZ ML Studio 2](#_Toc127812108)

[4. Get the REST API URL from the Real time endpoint 4](#_Toc127812109)

[5. Steps to use the deployed model to predict an API request 5](#_Toc127812110)

[6. How we can operationalize the model at scale? 6](#_Toc127812111)

# Use Case Details

* Train and deploy a model for MNIST dataset in Azure Machine Learning using Azure MNIST Open Dataset and create a real time endpoint which can be consumed by an API Interface.
* The MNIST database of handwritten digits has a training set of 60,000 examples and a test set of 10,000 examples. The digits have been size-normalized and centered in a fixed-size image.

# Pre-requisites

* An Azure account
* Create an Azure Machine Learning workspace in the Azure account
* Get the Machine Learning Studio web URL

# Steps to deploy the model in AZ ML Studio

3.1 Open the Azure Machine Learning Studio web URL

3.2 Click Notebooks under Authoring pan on the left hand side

3.3 Upload the folder [**MNIST\_AZURE**](https://github.com/ShabnaNazar/MNIST_AZURE) under the username in Azure Notebooks

Github link: <https://github.com/ShabnaNazar/MNIST_AZURE>

The folder MNIST\_AZURE will have the below file

1. image\_classifier\_mnist.ipynb 🡪 Code to deploy a classification model for Azure Open Dataset MNIST
2. image\_classifier\_mnist.yml 🡪 Environment requirements to execute the code
3. score.py 🡪 script required to initialize the model and do predictions
4. utils.py 🡪 script contains load\_data function which parses the compressed files into numpy arrays
5. Sample\_test\_data.txt 🡪 a sample test data for testing purpose

Graphical user interface, text, application, Word

Description automatically generated

3.4 Run the notebook file [image\_classifier\_mnist.ipynb](https://github.com/ShabnaNazar/MNIST_AZURE/blob/main/image_classifier_mnist.ipynb) present in the MNIST\_AZURE folder

3.5 Once the notebook is executed successfully, a training job is created, the model is registered in the model registry and a real time endpoint is created

1. Training job with MLFlow tracking

Jobs > **mnist-sklearn-classification**

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface

Description automatically generated

1. Model registry with Model version

Models > sklearn\_mnist\_model

Graphical user interface, text, application, email

Description automatically generated

1. Real time Endpoint

Graphical user interface, text, application, email

Description automatically generated

# Get the REST API URL from the Real time endpoint

Click the endpoint name starting with ‘mnist-svc” and get the RESRT endpoint url from the Details tab

Graphical user interface, application

Description automatically generated

Eg: (**SAMPLE ONLY**)

URL :

<http://f8771be4-4e9f-43ea-8d89-4a1aada1fd65.eastus2.azurecontainer.io/score>

# Steps to use the deployed model to predict an API request

For Testing the rest endpoint, there are multiple ways we can test the online endpoint. Below are some of the options

Option 1: Send an API POST request using POSTMAN API Interface

1. Create a POST request with scoring url
2. Add a key value pair for the Headers as ”Content-Type" as KEY "application/json" as the VALUE
3. Get a sample test data from the file Sample\_test\_data.txt and copy it in the Body as raw
4. Click Send button
5. You will get the predicted result as the API response

Graphical user interface, text, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Option 2: Test the Endpoint directly in Azure Machine Learning Studio

Navigate to the Endpoint created after the code run. Click Test tab present next to the Details tab. Copy the contents of the file present in Sample\_test\_data.txt and click the Test button

# How we can operationalize this model at scale?

Currently the model has been deployed in Azure ML workspace for Dev subscription. To operationalize the model, we may need to deploy the model to a Test/QA account, do QA/Model Testing. Based on the testing results, deploy the model to a Prod Container Instance or a Kubernetes cluster.

Here are the pre-requisites for multi account deployment using MLOps process

* Create a Azure DevOps project
* Data Scientists will upload the source code, yaml files, score.py, testing scripts, environment details in the Code Repo
* ML Engineers to create the template files for ML pipelines and confifurations required for training jobs like compute type, no of nodes etc and upload to Code Repo
* DevOps engineers will create the required configuration files and scripts to run the ML pipelines and IaC pipelines and upload to Code Repo

We are multiple design patterns for operationalizing a model at scale. I’m listing one of such method here.

Deployment to QA

1. Create an Infra as Code (IaC) pipeline in the Azure DevOps project: Create the azure resources like resource group, ML workspace, compute instance, storage accounts etc using Terraform scripts in QA environment
2. Create a Continuous Integration (CI) Pipeline in Azure DevOps project and add the scripts for the below tasks for automated run in QA environment a. Register the dataset to ML workspace, if required b. Run Code Testing scripts c. Train and register the model d. Baseline the date quality f. Baseline the model quality
3. Create a Continuous Deployment (CD) Pipeline in Azure DevOps project and add the scripts for the below tasks for an automated run in QA environment a. Deploy the model to an online endpoint b. Smoke testing Based on the succesful testing in QA environment, we can start the automated deployment to PROD

Deployment to PROD

1. Create an IaC pipeline in the Azure DevOps project: Create the azure resources like resource group, ML workspace, compute instance, storage accounts etc using Terraform scripts in PROD environment
2. Create a CD Pipeline in Azure DevOps project and add the scripts for the below tasks for automated run in QA environment a. Download the required version of model from model registry in QA b. Deploy the model to an online endpoint. e. Smoke testing
3. Create a Monitoring pipeline to monitoring to review the data drift or model drift (Based on ground truth availabilty) and system performance matrices to alert users in case of any issues. Trigger the CI pipeline incase retraining is required with the new dataset or with the new code