AI on IBM Z

Credit risk assessment solution template

This solution template provides an example on how to deploy AI using an IBM Z environment, while making use of open source frameworks, Machine Learning for IBM z/OS (MLz), and more.

Within this solution template, there are various phases of the AI lifecycle included. Work through each of the following steps to deploy your own credit risk assessment solution on IBM Z.

Table of contents

Step 1

AI model training

We will build a credit risk assessment AI model by training with the provided Rapid AI on IBM Z Development Jupyter notebook. Simply point the Jupyter notebook to your dataset and run it to generate your AI model. This trained AI model can then be deployed with MLz.

All sample code for this section is within

`aionz-st-credit-risk-assessment/zST-model-training-jupyter`

Prerequisites

- Must have Python (3.9 or 3.10) installed

Dataset guidance

Sample credit risk assessment dataset can be found on Kaggle –

https://www.kaggle.com/datasets/laotse/credit-risk-dataset

Required features

- person_age
- person_income
- person_home_ownership
- person_emp_length
- loan_intent
- loan_grade
- loan_amnt
- loan_int_rate
- loan_status
- loan_percent_income
- cb_person_default_on_file
- cb_person_cred_hist_length

Access rapid AI on IBM Z development environment

1. Access sample code

   ```
   cd zST-model-training-jupyter
   ```

2. Create and activate Python virtual environment

   ```
   python -m venv env
   source env/bin/activate
   ```

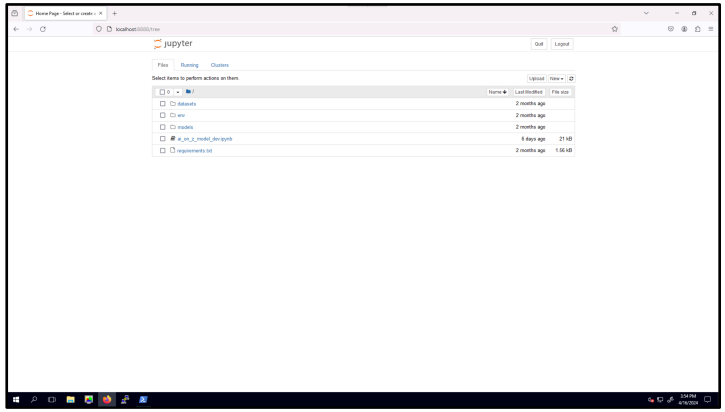
3. Install required Python packages

   ```
   pip install -r requirements.txt
   ```

4. Run Jupyter

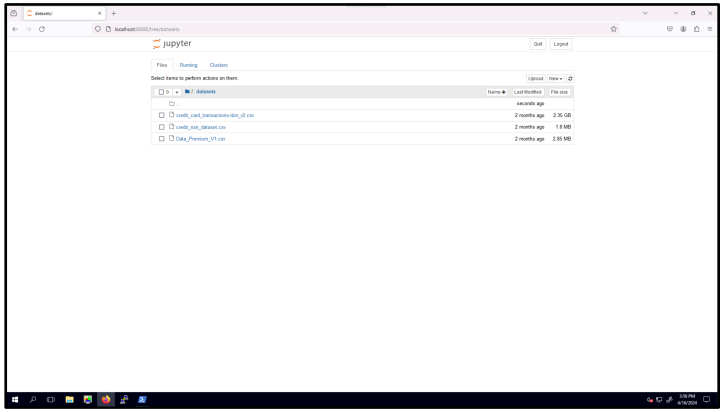
   ```
   jupyter notebook
   ```

5. View Jupyter interface
   a. Go to localhost:8888 in a web browser



Provide data

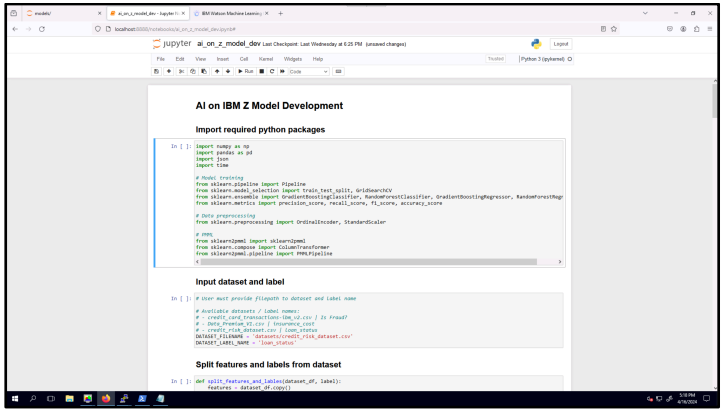
1. Your input dataset (csv) in `datasets/` directory
2. Add input data to Jupyter notebook (ai_on_z_model_dev.ipynb)
    a. Set `DATASET_FILENAME` to the path to your dataset
    b. Set `DATASET_LABEL_NAME` to the name of the column you're predicting from the dataset



Model training

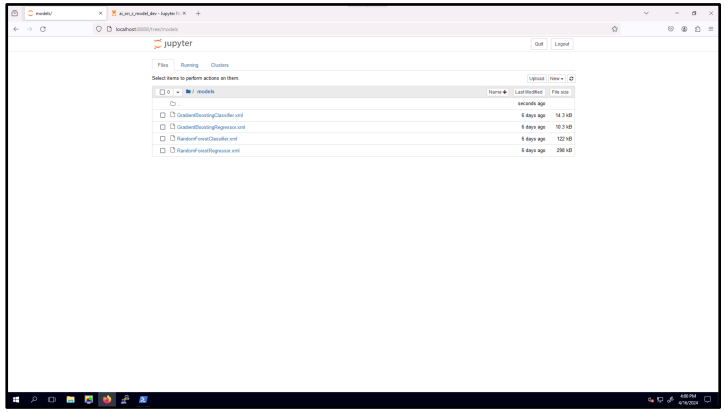
1. Step through and run all cells within Jupyter notebook (ai_on_z_model_dev.ipynb) within web browser

    Note: This may take several minutes



Access trained AI model

1. Once training is complete, you can find your AI models within the models/ directory (choose one for the following AI model deployment step)



1. AI model training.

2. AI model deployment

4. AI model integration

AI model training complete

Prerequisites

- Must have MLz installed

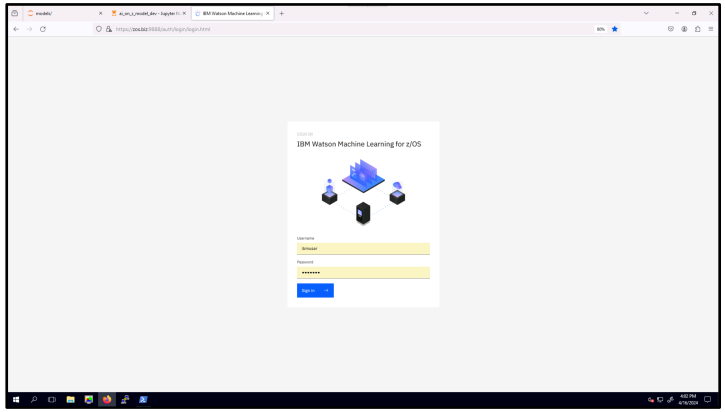
Step 2

AI model deployment

We will deploy our credit risk assessment AI model using MLz. We can utilize the model import functionality on the MLz UI. This deployed AI model can then be integrated into applications within the IBM Z environment.

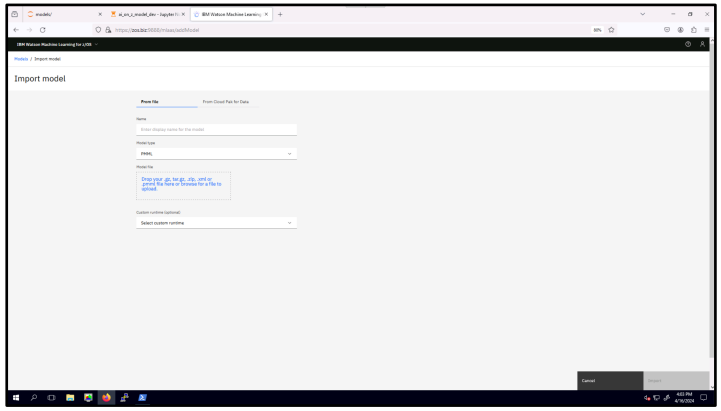
Go to MLz UI

1. Sign in with username/password

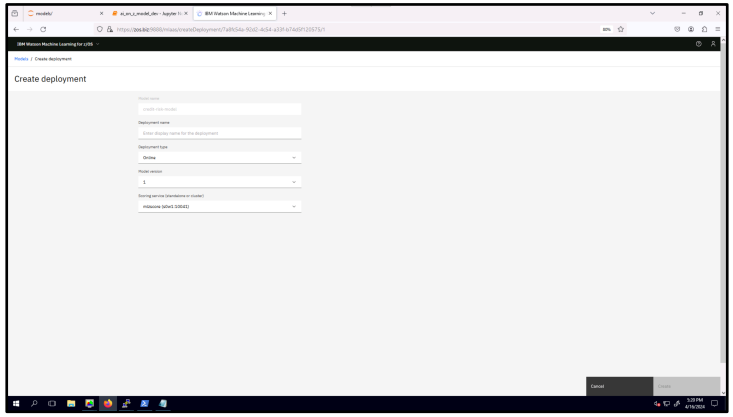


Import AI model

1. Go to models tab
2. Click import model
3. Enter model name
4. Choose model type
    a. Choose PMML if using your previously trained model
5. Drag and drop model file
   Use your previously trained model
6. Click import

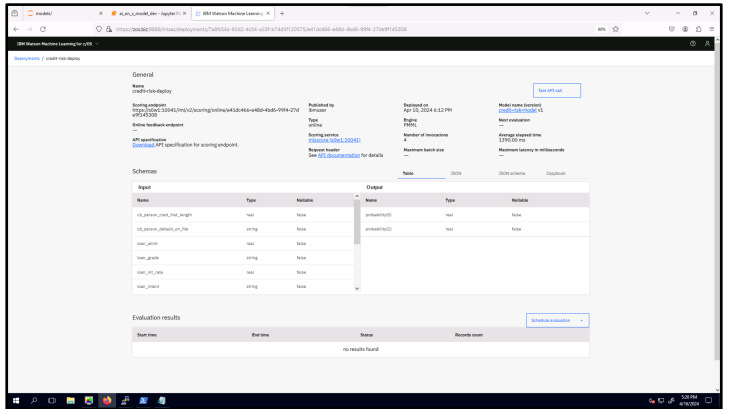


Deploy AI model



1. Go to models tab
2. Click action button for your model (on right side)
3. Click deploy
4. Enter deployment name
5. Choose deployment type
6. Choose model version
7. Choose scoring service
   Note: you should choose the correct scoring service based on your application (e.g. CICS or REST)
8. Click create

View deployed AI model



1. Go to deployments tab
2. Click on action button for your deployed model (on right side)
3. Click view details

1. AI model training.

2. AI model deployment

4. AI model integration

AI model deployment complete

Step 3

AI model integration

Choose One:

Web application
CICS-COBOL application

We can use our deployed MLz credit risk assessment AI model and integrate it into different types of applications. The AI model can be analyzed and/or provide inferencing APIs using the sample AI on IBM Z Credit Risk Assessment Dashboard.

Web application

We can use our deployed MLz credit risk assessment AI model and integrate it into different types of applications. Guidance on integrating the AI model into a sample credit risk assessment application is below.

All sample code for this section is within

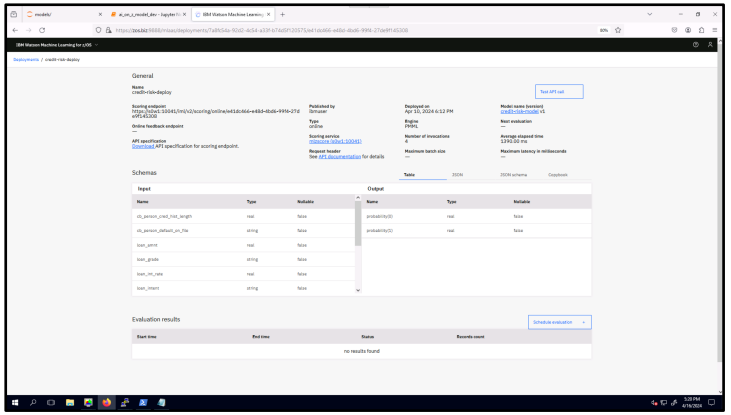
`aionz-st-credit-risk-assessment/zST-model-integration-cra`

Prerequisites

- Must have node.js v16 or newer installed
- Must have Docker or Podman installed
- Must have Git installed

Get model details for inferencing

1. Go to MLz UI
2. Go to deployments tab
3. Click on action button for your deployed model (on right side)
4. Click view details
5. Copy scoring endpoint

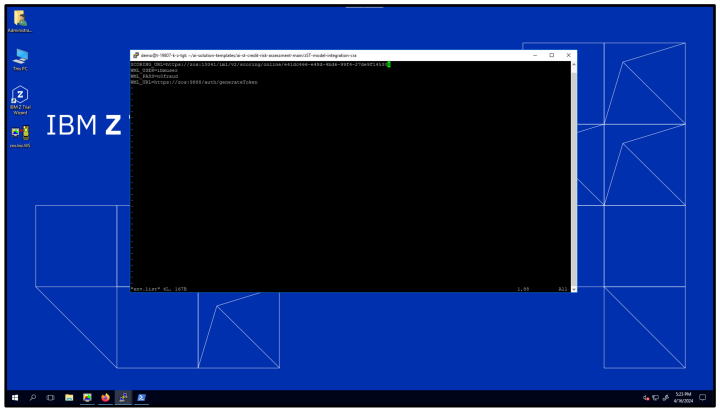


Configure sample application

1. Set the environment variables within

   `aionz-solution-templates/ai-st-credit-risk-main/zST-model-integration-cra/env.list`

   WML_IP_W_PORT (IP address of MLz with port)
   WML_USER (MLz username)
   WML_PASS (MLz password)
   WMLZ_ENDPOINT (scoring endpoint for deployed AI model)



Build sample application

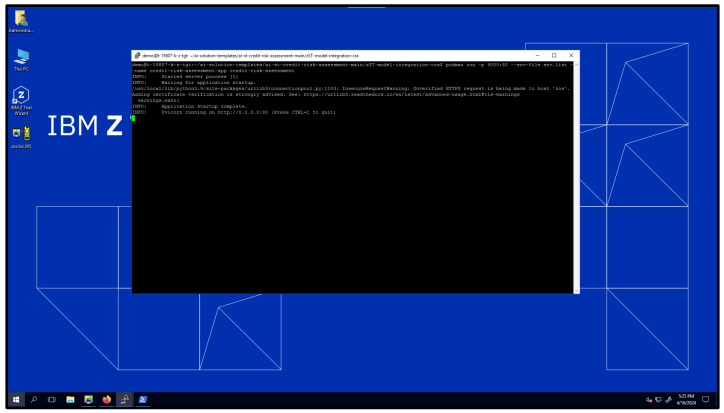
1. Run command in terminal

```
podman build -t credit-risk-assessment .
```

Deploy sample application

1. Run command in terminal (e.g. port 9000)

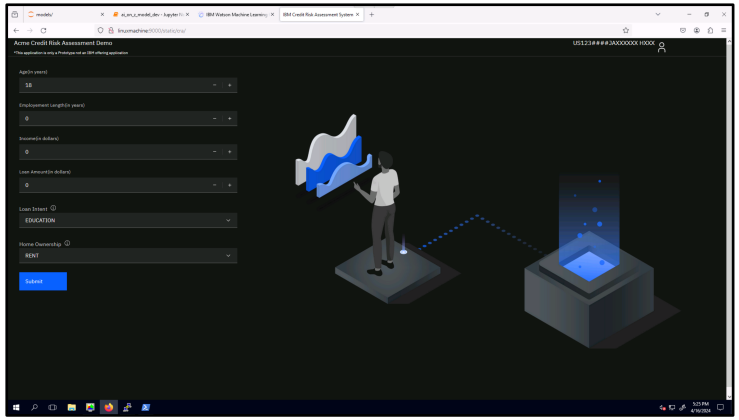
```
podman run -p 9000:80 --env-file env.list --
name credit-risk-assessment-app credit-risk-
assessment
```



Access sample application

1. View the following URL in a web browser
   a. Credit risk assessment
      http://linuxMachine:{port}/static/cra/
   b. Dashboard
      http://linuxMachine:{port}/static/dashboard/

Note: use same port as used within podman run



1. AI model training.

2. AI model deployment

4. AI model integration

☑ AI model integration complete (web)

CICS-COBOL application

In this type of CICS-COBOL program, we will be inferencing the Credit Risk Assessment model deployed into the MLz using REST API calls to a hosted UI. The UI will be making call to the MLz for scoring and the result will be sent back to the CICS-COBOL program.

All sample code for this section is within

`aionz-st-credit-risk-assessment/zST-model-integration-CICS`

Prerequisites

- Must have access to z/OS CICS Environment
- Must have MLz installed
- Must have model deployed with the CICS scoring server as scoring service

Integrate into CICS application

A sample COBOL file for the below integration can be found here:

`aionz-st-credit-risk-assessment/zST-modelintegration-CICS/CRAURL.cbl`

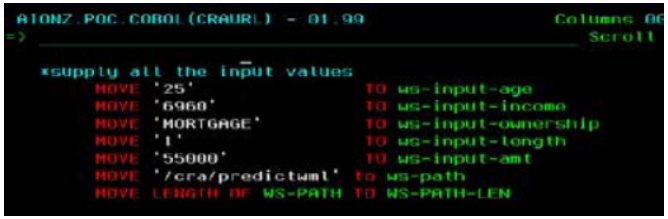
1. Update @HOSTNAME and @PORTNUM within the sample CRAURL.cbl provided

2. Create connection with the hosted UI, using the Web Open command. Mention the host number & port number where the UI is hosted

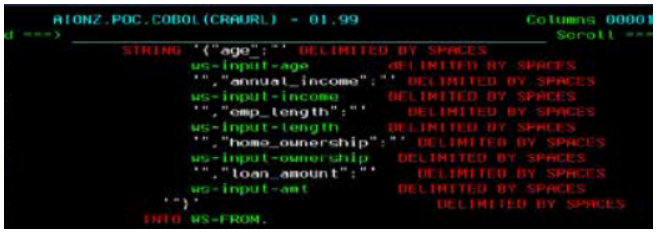
3. Using CICS ASSIGN get the application id of the UI

Integrate into CICS application

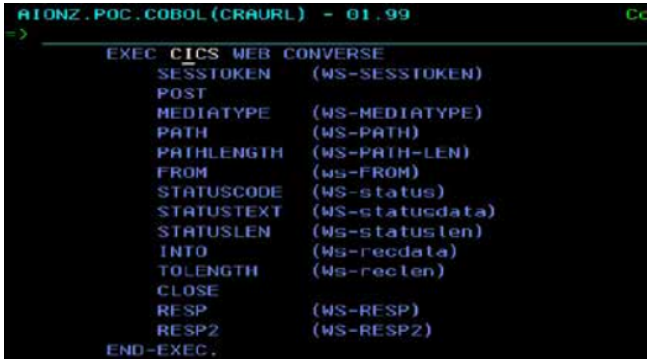
4. Supply all the inputs required by the UI service along with the API path of the inference Click view details



5. Prepare the JSON data for the REST API call using COBOL's String statement



6. Use the web converse command in CICS to pass the data to the UI backend service and get the response



Integrate into CICS application

7. Close the web connection to the server

```
* Close the Session to the Remote Server
     EXEC CICS WEB CLOSE SESSTOKEN(ws-sessstoken)
     END-EXEC.
     PERFORM 0700-CHK-RESP.
```

8. Process the response received from API call

```
    UNSTRING ws-recdata delimited by '"loan_status":'
        into ws-str4
             ws-loan-status
    END-UNSTRING.
```

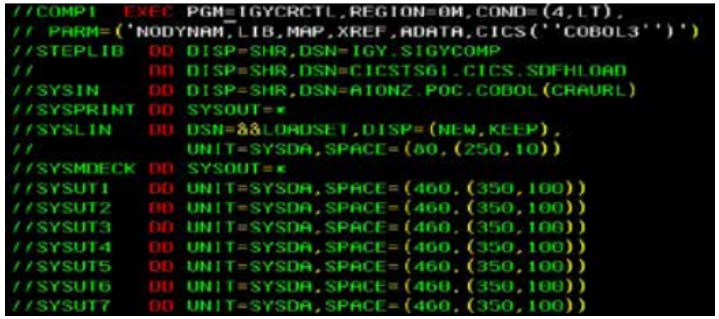
9. Handle the error codes as needed

```
    MOVE SPACES      TO WS-MESSAGE
    IF WS-RESP NOT EQUAL ZERO
       MOVE WS-RESP      TO err-resp
       MOVE WS-RESP2     TO ERR-RESP2
       STRING WS-STEP DELIMITED BY SPACES
              'failed with RESP = '
              ERR-resp delimited by spaces
              'RESP2 = '
              ERR-resp2 delimited by spaces
            into ws-message
       END-STRING
       display 'failure for ' ws-step
       display ws-message
       EXEC CICS RETURN
       END-EXEC
```

Integrate into CICS application

10. Compile the COBOL program. Sample compile JCL provided here:

```
AIONZ.WKSHP.COMPILE.jcl
```



11. Define the transaction

```
CEDA DEFINE TRANS(<transaction name>)
GROUP(<group name>)
```

```
PROGRAM(<program name>)
DESCRIPTION(<transaction description>)
```

12. Define the program

```
CEDA DEFINE PROGRAM(<program name>)
GROUP(<group name>)
```

```
LANGUAGE(COBOL) DESCRIPTION(<program
description>)
```

Integrate into CICS application

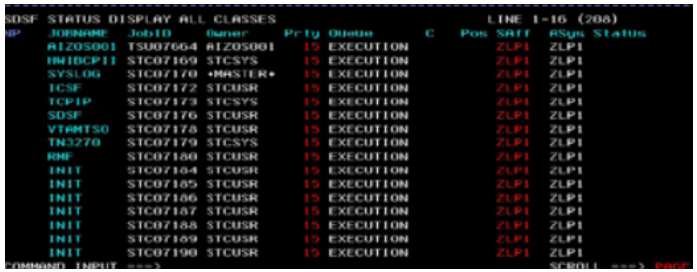
13. Install the transaction and program in the CICS region. Execute the below mentioned command to define the transaction replacing transaction name with the transaction name, group name with the name of the group & program name with name of the COBOL program, transaction description with appropriate description for the transaction, finally, program description with appropriate description for the program

```
CEDA INS TRANS(<transaction name>)
GROUP(<group name>)

CEDA INS PROGRAM(<program name>)
GROUP(<group name>)
```

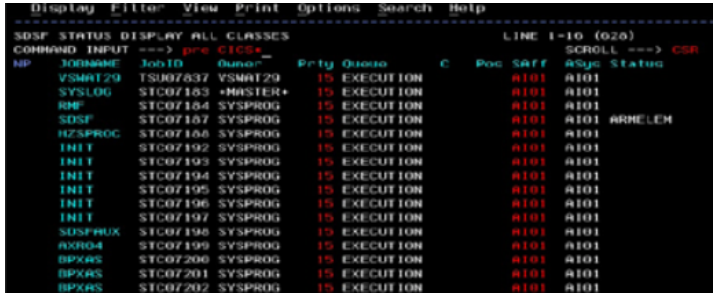
14. To invoke the transaction Type the transaction name and hit Enter

15. Verify the result
16. Lets go back to the TSO screen. Navigate to the Spool



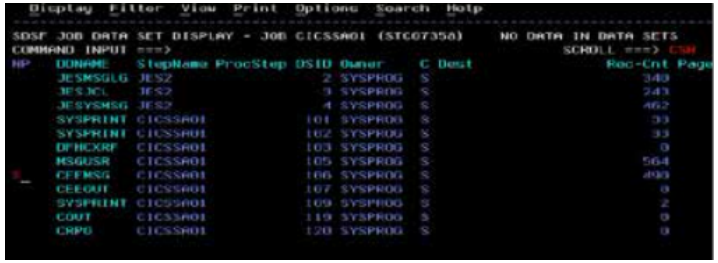
Integrate into CICS application

a. To check the started task, go to Spool pre CICS* (this will list the active CICS regions)



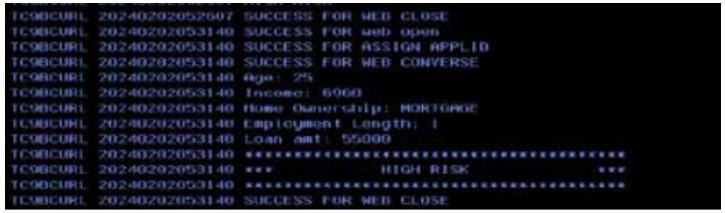
b. Put ? to see the details of the Spool job

c. Check the CEEMSG dataset name



Integrate into CICS application

d. Check the display from the module



1. AI model training.

2. AI model deployment

4. AI model integration

☑ AI model integration complete (CICS)