



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

AI model training.....	3
AI model deployment.....	8
AI model integration.....	12
Ecommerce web application.....	13
CICS-COBOL application.....	18



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

1. AI model training.

Access rapid AI on IBM Z development environment

Provide data

Model training

Access trained AI model

2. AI model deployment

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
```

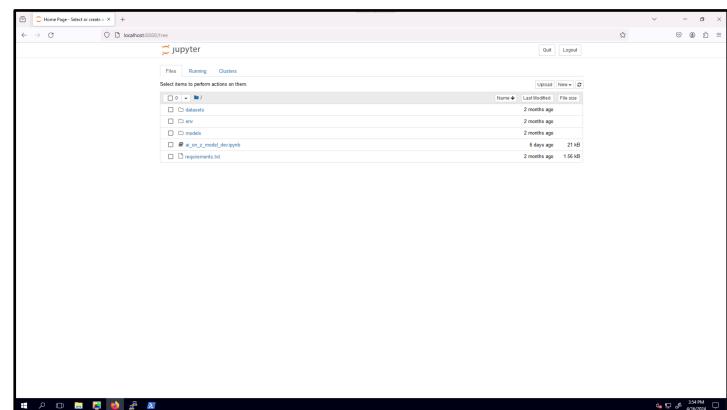
4. Run Jupyter

```
jupyter notebook
```

5. View Jupyter interface

- a. Go to localhost:8888 in a web browser

4. AI model integration



1. AI model training.

Access rapid AI on IBM Z development environment

[Provide data](#)

[Model training](#)

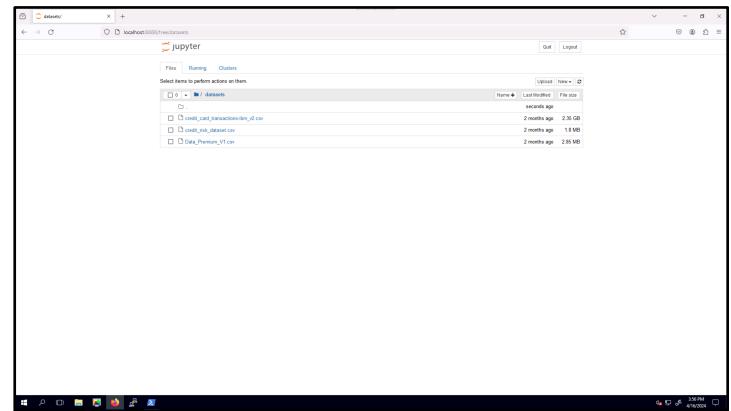
Access trained AI model

2. AI model deployment

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

4. AI model integration



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

```
In [1]: import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, RandomForestRegressor
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
from sklearn.preprocessing import OrdinalEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
In [2]: # Import required python packages
# Data processing
# Model training
# Model evaluation
# Model deployment
```

Import required python packages

```
In [1]: import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, RandomForestRegressor
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
from sklearn.preprocessing import OrdinalEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
```

Input dataset and label

```
In [2]: # If user hasn't provide filename to dataset and label_name
# Associate dataset and label_name
# credit_card_transactions-lens_2d.csv | X ReadData
# user_id_dataset.csv | Y ReadLabel
# credit_risk_assessment.csv | CreditStatus
# user_profile.csv | UserProfile
# DATASET_FILENAME = 'credit_card_transactions-lens_2d.csv'
# DATASET_LABEL_NAME = 'User_Status'
```

Split features and labels from dataset

```
In [3]: # Split features and labels from dataset
# DATASET_FILENAME = 'credit_card_transactions-lens_2d.csv'
# DATASET_LABEL_NAME = 'User_Status'
```

1. AI model training.

Access rapid AI on IBM Z development environment

Provide data

Model training

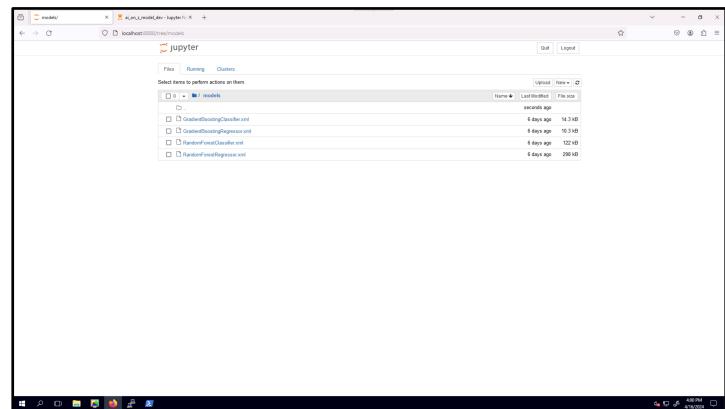
[Access trained AI model](#)

2. AI model deployment

Access trained AI model

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

4. AI model integration

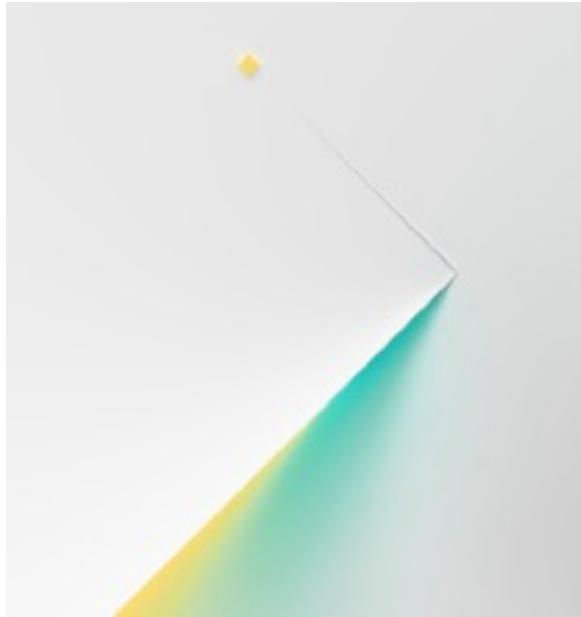


1. AI model training.

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.

1. AI model training.

2. AI model deployment

4. AI model integration

[Go to MLz UI](#)

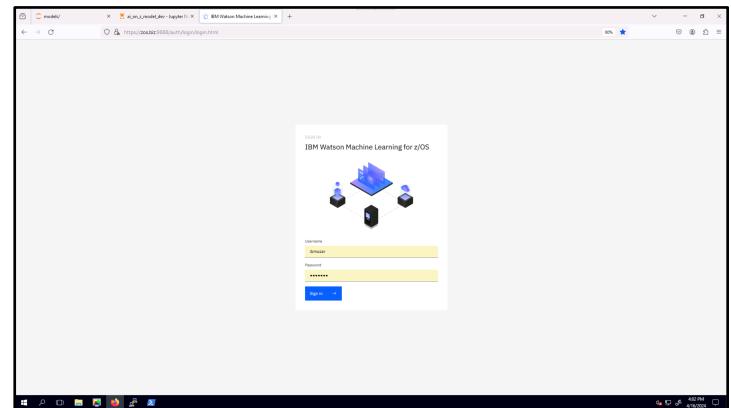
[Import AI model](#)

[Deploy AI model](#)

[View deployed AI model](#)

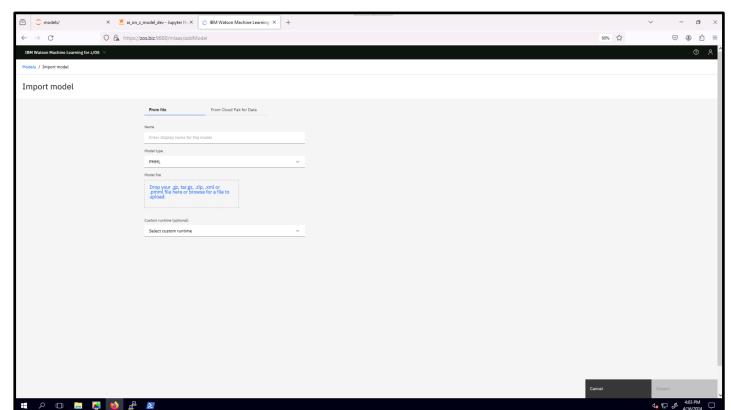
Go to MLz UI

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



1. AI model training.

2. AI model deployment

4. AI model integration

Go to MLZ UI

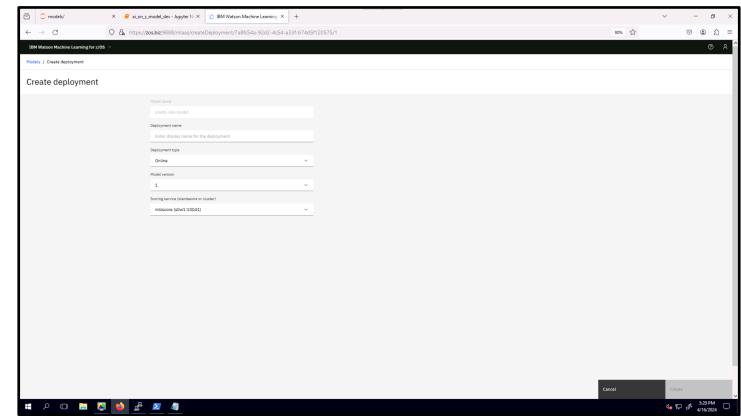
Import AI model

Deploy AI model

View deployed AI model

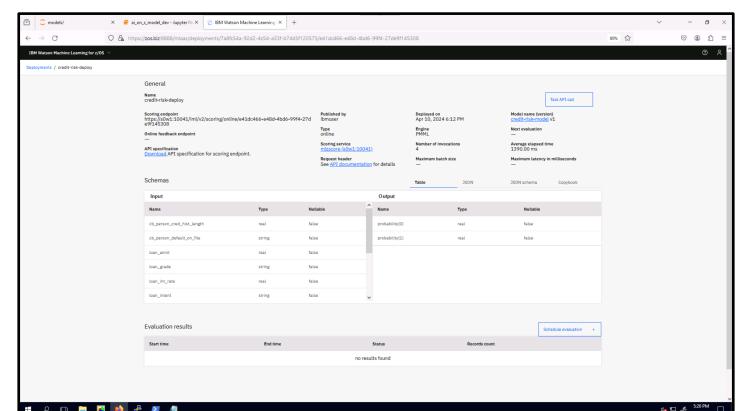
Deploy AI model

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

1. AI model training.

[Get model details for inferencing](#)

[Configure sample application](#)

Build sample application

Deploy sample application

Access sample application

2. AI model deployment

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

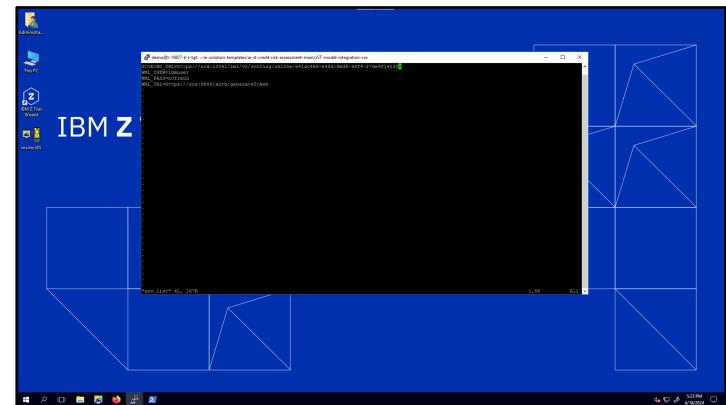
4. AI model integration

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)



1. AI model training.

2. AI model deployment

4. AI model integration

Get model details for
inferencing

Configure sample
application

[Build sample application](#)

[Deploy sample application](#)

Access sample application

Build sample application

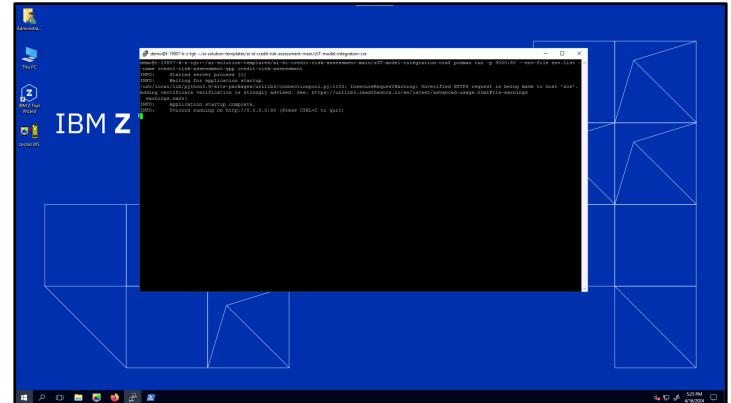
1. Run command in terminal

```
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
```



1. AI model training.

Get model details for inferencing

Configure sample application

Build sample application

Deploy sample application

Access sample application

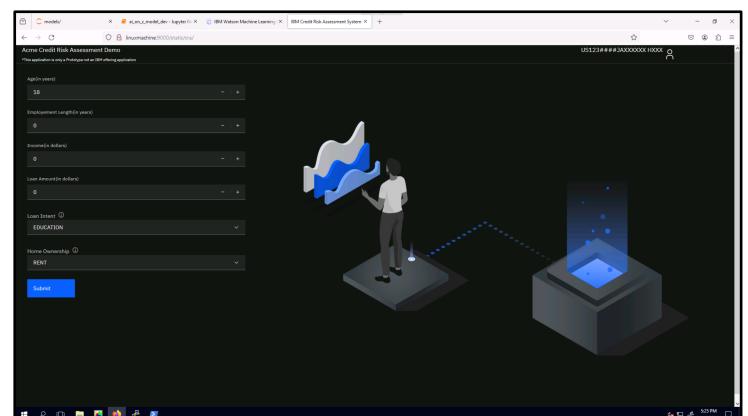
2. AI model deployment

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

4. AI model integration



1. AI model training.

2. AI model deployment

4. AI model integration

AI model integration complete (web)

CICS-COBOL application

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

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

[Integrate into CICS application](#)

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

```
MOVE 'Web open'          to ws-step.
EXEC CICS WEB OPEN
      http
      host(ws-host)
      portnumber(ws-portnumber)
      SESSTOKEN(ws-sesstoken)
      RESP (ws-resp)
      RESP2(ws-resp2)
END-EXEC.
```

```
MOVE 'ASSIGN APPLID'    to ws-step.
EXEC CICS ASSIGN
      APPLID(ws-applid)
END-EXEC.
```

- ## 1. AI model training.

- ## 2. AI model deployment

- ## 4. AI model integration

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

```
ATONZ.POC.COBOL (CRAURL) - 01.99 Columns 80  
=> *Supply all the input values  
      MOVE '25'           TO WS-INPUT-AGE  
      MOVE '6960'          TO WS-INPUT-INCOME  
      MOVE 'MORTGAGE'      TO WS-INPUT-OWNERSHIP  
      MOVE '1'              TO WS-INPUT-LENGTH  
      MOVE '55000'          TO WS-INPUT-AMT  
      MOVE '/cra/predictwml' TO WS-PATH  
      MOVE LENGTH OF WS-PATH TO WS-PATH-LEN
```

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

```
RIONZ.POC.COBOL (CRURL) - 01.99          COLUMNS 00001
d ----> STRING ("age":") DELIMITED BY SPACES
           us->input-age      DELIMITED BY SPACES
           ("annual_income":") DELIMITED BY SPACES
           us->input-income    DELIMITED BY SPACES
           ("emp_length":")   DELIMITED BY SPACES
           us->input-length    DELIMITED BY SPACES
           ("home_ownership":") DELIMITED BY SPACES
           us->input-ownership  DELIMITED BY SPACES
           ("loan_amount":")   DELIMITED BY SPACES
           us->input-amt       DELIMITED BY SPACES
           ("")
INFO RS-FROM.
```

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

```
RIONZ.POC.COBOL(CRAURL) - 01.99 Cc  
=> EXEC CICS WEB CONVERSE  
      SESSTOKEN     (WS-SESSTOKEN)  
      POST  
      MEDIATYPE    (WS-MEDIATYPE)  
      PATH          (WS-PATH)  
      PATHLENGTH   (WS-PATH-LEN)  
      FROM          (ws-FROM)  
      STATUSCODE   (WS-status)  
      STATUSTEXT   (WS-statusdata)  
      STATUSLEN    (Ws-statuslen)  
      INTO          (Ws-recdata)  
      TOLENGTH    (Ws-reclen)  
      CLOSE  
      RESP         (WS-RESP)  
      RESP2        (WS-RESP2)  
END-EXEC
```

1. AI model training.

2. AI model deployment

4. AI model integration

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-sess-token)
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
```

1. AI model training.

2. AI model deployment

4. AI model integration

[Integrate into CICS application](#)

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

AIONZ.WKSHP.COMPILE.jcl

```
//COMPI EXEC PGM=IGYCRCTL,REGION=0M,COND=(4,LT)
//PRRM=('NODYNAM,LIB,MAP,XREF,ADATA,CICS(''COBOL3'')')
//STEPLIB DD DISP=SHR,DSN=IGY.SIGYCOMP
//          DD DISP=SHR,DSN=CICSTS61.CICS.SDFHLOAD
//SYSIN  DD DISP=SHR,DSN=AIONZ.POC.COBOL(CRAURL)
//SYSPRINT DD SYSOUT=*
//SYSLIN  DD DSN=&LUDRSET,DISP=(NEW,KEEP),
//          UNIT=SYSDA,SPACE=(60,(250,10))
//SYSMDECK DD SYSOUT=*
//SYSUT1  DD UNIT=SYSDA,SPACE=(460,(350,100))
//SYSUT2  DD UNIT=SYSDA,SPACE=(460,(350,100))
//SYSUT3  DD UNIT=SYSDA,SPACE=(460,(350,100))
//SYSUT4  DD UNIT=SYSDA,SPACE=(460,(350,100))
//SYSUT5  DD UNIT=SYSDA,SPACE=(460,(350,100))
//SYSUT6  DD UNIT=SYSDA,SPACE=(460,(350,100))
//SYSUT7  DD UNIT=SYSDA,SPACE=(460,(350,100))
```

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>)
```

1. AI model training.

2. AI model deployment

4. AI model integration

[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

SP	STATUS	DISPLAY	ALL CLASSES	LINE 1-16 (208)						
NP	JOBNAME	JobID	Owner	Prtg	Queue	C	Pos	SATr	ASys	Status
	RIZOS001	TSU07664	RIZOS001	15	EXECUTION		ZLP1	ZLP1		
	HIBCP11	STC07169	STCSYS	15	EXECUTION		ZLP1	ZLP1		
	SYSLOG	STC07170	*MASTER*	15	EXECUTION		ZLP1	ZLP1		
	TCSF	STC07172	STCUSR	15	EXECUTION		ZLP1	ZLP1		
	TCP/IP	STC07173	STCSYS	15	EXECUTION		ZLP1	ZLP1		
	SDSF	STC07176	STCUSR	15	EXECUTION		ZLP1	ZLP1		
	VTERM150	STC07178	STCUSR	15	EXECUTION		ZLP1	ZLP1		
	TNS270	STC07179	STCSYS	15	EXECUTION		ZLP1	ZLP1		
	RMF	STC07180	STCUSR	15	EXECUTION		ZLP1	ZLP1		
	INIT	STC07184	STCUSR	15	EXECUTION		ZLP1	ZLP1		
	INIT	STC07185	STCUSR	15	EXECUTION		ZLP1	ZLP1		
	INIT	STC07186	STCUSR	15	EXECUTION		ZLP1	ZLP1		
	INIT	STC07187	STCUSR	15	EXECUTION		ZLP1	ZLP1		
	INIT	STC07188	STCUSR	15	EXECUTION		ZLP1	ZLP1		
	INIT	STC07189	STCUSR	15	EXECUTION		ZLP1	ZLP1		
	INIT	STC07190	STCUSR	15	EXECUTION		ZLP1	ZLP1		

1. AI model training.

2. AI model deployment

4. AI model integration

Integrate into CICS application

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

SDSF STATUS DISPLAY ALL CLASSES							LINE 1-10 (026)		SCROLL ==> CSR	
COMMAND	INPUT	JobID	Owner	Prtg	Queue	C	Pos	SRTF	ASyS	Status
NP	JOBNAME	TSU07837	VSMBT29	15	EXECUTION	R	101	A101		
	SYSLG	STC07163	+MASTER+	15	EXECUTION	R	101	A101		
	RMF	STC07164	SYSPROG	15	EXECUTION	R	101	A101		
	SDSF	STC07167	SYSPROG	15	EXECUTION	R	101	A101	ARHELEM	
	HZSPROC	STC07168	SYSPROG	15	EXECUTION	R	101	A101		
	DINIT	STC07192	SYSPROG	15	EXECUTION	R	101	A101		
	DINIT	STC07193	SYSPROG	15	EXECUTION	R	101	A101		
	DINIT	STC07194	SYSPROG	15	EXECUTION	R	101	A101		
	DINIT	STC07195	SYSPROG	15	EXECUTION	R	101	A101		
	DINIT	STC07196	SYSPROG	15	EXECUTION	R	101	A101		
	DINIT	STC07197	SYSPROG	15	EXECUTION	R	101	A101		
	SUSPMUX	STC07198	SYSPROG	15	EXECUTION	R	101	A101		
	AKR04	STC07199	SYSPROG	15	EXECUTION	R	101	A101		
	BPXRS	STC07200	SYSPROG	15	EXECUTION	R	101	A101		
	BPXRS	STC07201	SYSPROG	15	EXECUTION	R	101	A101		
	BPXRS	STC07202	SYSPROG	15	EXECUTION	R	101	A101		

- Put ? to see the details of the Spool job

SDSF STATUS DISPLAY ALL CLASSES							LINE 1-1 (1)		SCROLL ==> CSR	
COMMAND	INPUT	JobID	Owner	Prtg	Queue	C	Pos	SRTF	ASyS	Status
NP	JOBNAME	STC07358	SYSPROG	15	EXECUTION	R	101	A101	ARHELEM	

- Check the CEEMSG dataset name

SDSF JOB DATA SET DISPLAY - JOB CICSSW01 (STC07350)							NO DATA IN DATA SETS		SCROLL ==> CSR	
COMMAND	INPUT	JobID	StepName	ProcStep	DSID	Owner	C	Dest	Rec-Cnt	Page
NP	DSNNAME	STC07350	JESMSGLG	2	SYSPROG	S			340	
			JESJCL	3	SYSPROG	S			243	
			JESYSMLG	3	SYSPROG	S			462	
			SYSPRINT	101	SYSPROG	S			33	
			SYSPRINT	102	SYSPROG	S			33	
			DFHGXRF	103	SYSPROG	S			0	
			MSGUSR	105	SYSPROG	S			564	
			CEEMSG	106	SYSPROG	S			490	
			CEEOUT	107	SYSPROG	S			0	
			SYSPRINT	109	SYSPROG	S			2	
			COVIT	110	SYSPROG	S			0	
			CRPG	120	SYSPROG	S			0	

1. AI model training.

2. AI model deployment

4. AI model integration

[Integrate into CICS application](#)

d. Check the display from the module

```
TC90CURL 20240202052607 SUCCESS FOR WEB CLOSE
TC90CURL 20240202053140 SUCCESS FOR web open
TC90CURL 20240202053140 SUCCESS FOR ASSIGN APPLID
TC90CURL 20240202053140 SUCCESS FOR WEB CONVERSE
TC90CURL 20240202053140 Age: 25
TC90CURL 20240202053140 Income: 6960
TC90CURL 20240202053140 Home_Ownership: MORTGAGE
TC90CURL 20240202053140 Employment_Length: 1
TC90CURL 20240202053140 Loan_amt: 55000
TC90CURL 20240202053140 ****
TC90CURL 20240202053140 *** HIGH RISK ***
TC90CURL 20240202053140 ****
TC90CURL 20240202053140 SUCCESS FOR WEB CLOSE
```

1. AI model training.

2. AI model deployment

4. AI model integration

AI model integration complete (CICS)