National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Naaga Barani Govindan Venkatraj
Student ID: x22104038

School of Computing
National College of Ireland

Supervisor: Ahmed makki

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Naaga Barani Govindan Venkatraj
Student ID:	x22104038
Programme:	Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Ahmed makki
Submission Due Date:	14/12/2023
Project Title:	Configuration Manual
Word Count:	430
Page Count:	10

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Naaga Barani Govindan Venkatraj
Date:	14th December 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Naaga Barani Govindan Venkatraj
Student ID: x22104038

1 Introduction

The Configuration Manual contains all the instructions for reproducing the study and its findings on a local environment and the Azure ML cloud platform. This manual provides thorough information about the system specs required for running the program locally, the source of the dataset, the Python machine learning packages used, the process of creating the cloud environment using Azure SDK v2, and the execution model of the Azure pipeline for the project.

2 System Specifications

Hardware Configuration for the local run:

- Processor: Intel 9th Gen Core i7-9750h @3.6 GHz
- RAM: 16 GB DDR4 RAM 3200MHz
- Storage (SSD): 512GB
- Operating System: Windows 10, 64-bit

Software Packages for the local run:

- Python 3.8
- Anaconda Navigator 2.3.2
- PyCharm IDE Community Edition 2021.3
- Jupyter Notebook

3 ML Packages

The subsequent machine learning packages were installed on the local system for early code development before migration to the cloud. To simplify the process of setting up the environment and installing packages, a requirements.txt file was provided for the project.

To execute the command on a local computer, utilise the following instruction in the Windows terminal:

Create a conda environment using the configuration file "environment.yml" by using the command "conda env create -f config/environment.yml".

4 Environment Setup – Package Versions

```
blis==0.4.1
certifi==2021.10.8
charset-normalizer==2.0.7
click==8.0.3
cycler==0.11.0
cymem==2.0.6
fonttools==4.28.1
ftfy==6.0.3
idna==3.3
joblib==1.1.0
kiwisolver==1.3.2
matplotlib==3.5.0
murmurhash==1.0.6
nltk==3.6.5
numpy~=1.21.6
packaging==21.2
pandas==1.3.4
Pillow==8.4.0
plac==0.9.6
preshed==3.0.6
pyparsing==2.4.7
python-dateutil==2.8.2
pytz==2021.3
regex==2021.11.10
requests==2.26.0
scikit-learn==1.0.1
scipy==1.7.2
seaborn==0.11.2
setuptools-scm==6.3.2
six==1.16.0
spacy==2.2.0
srsly==1.0.5
thinc==7.1.1
threadpoolctl==3.0.0
tomli==1.2.2
tqdm==4.62.3
urllib3==1.26.7
wasabi==0.8.2
wcwidth==0.2.5
wordcloud==1.8.1
Flask==1.1.2
Flask-Bootstrap==3.3.7.1
```

5 Dataset

The data collection process involves gathering the tweets from the Twitter API and Tweepy libraries in addition to the Kaggle dataset.

Link: https://www.kaggle.com/datasets/ywang311/twitter-sentiment/data

6 Azure ML Configuration

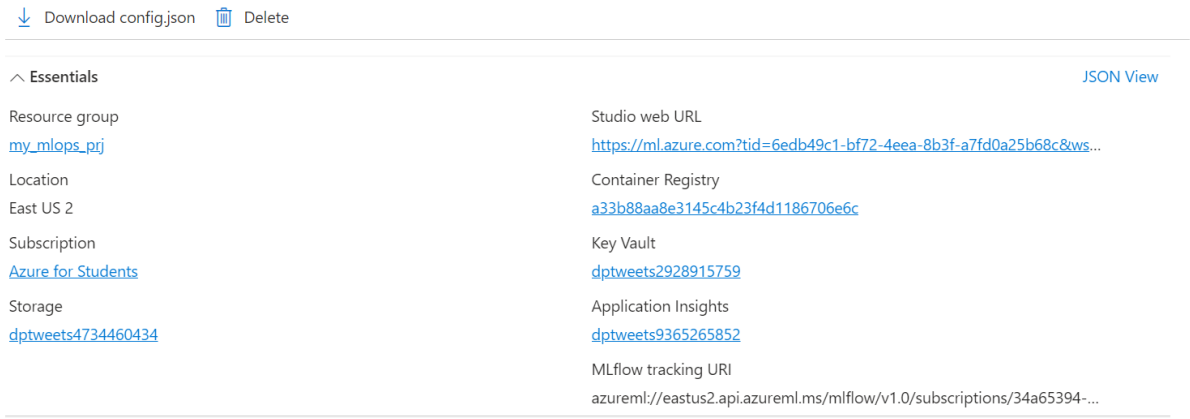

Figure 1: Azure Account Details

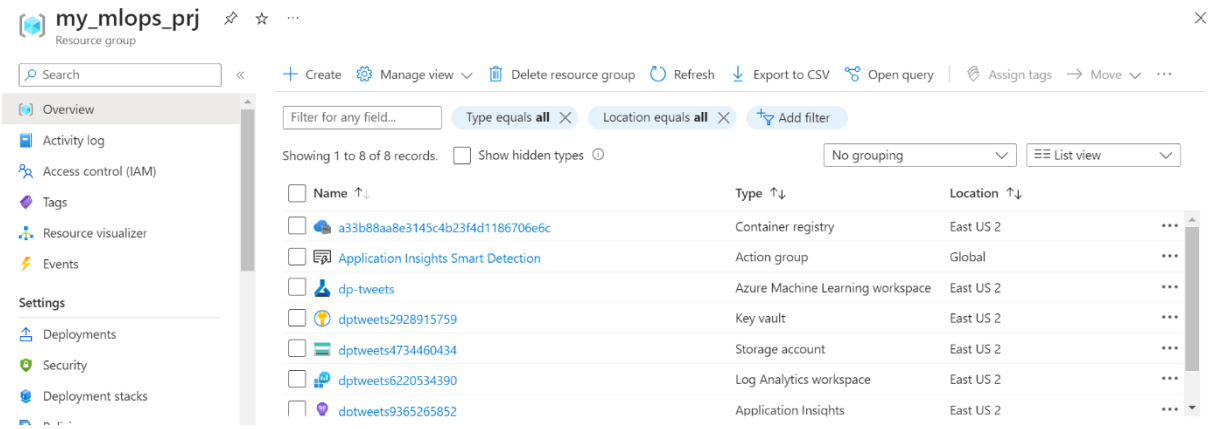

Figure 2: Azure ML Resource Group

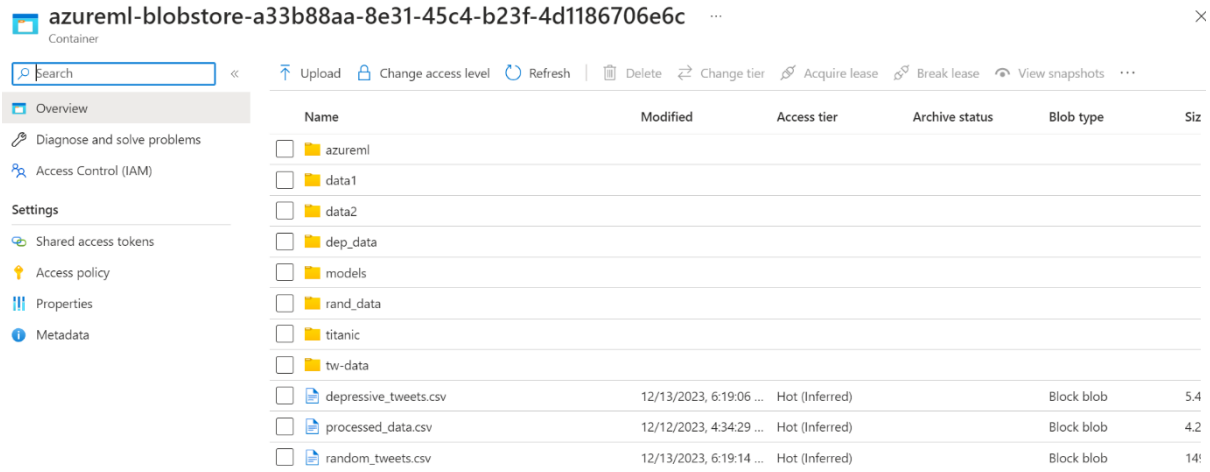

Figure 3: Azure Blob Storage

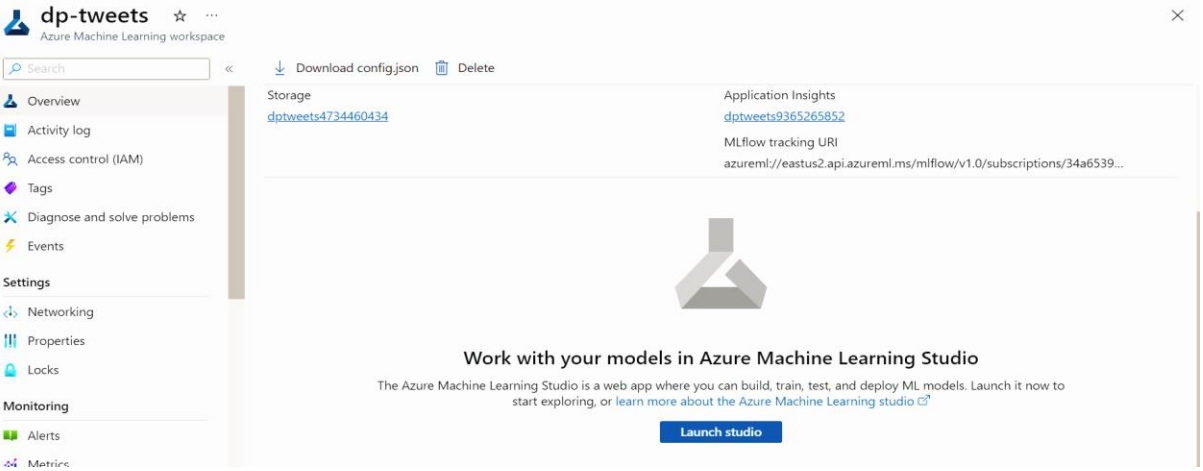

Figure 4: Launch Azure Machine Learning Studio from dp-tweets workspace.

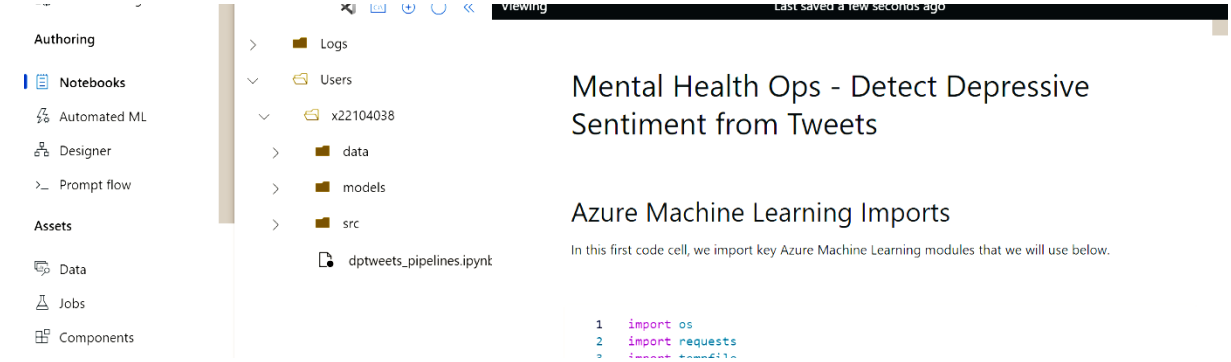

Figure 4: Open Notebook Instance.

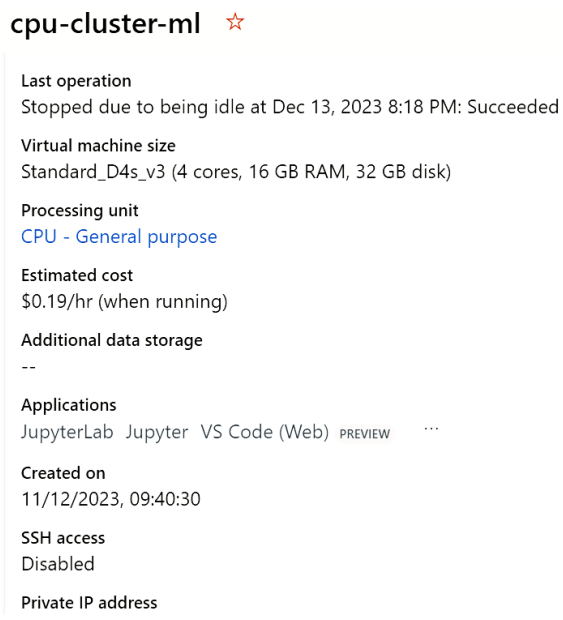

Figure 5. Compute Instance for the notebook

```python
import os
import requests
import tempfile
import azureml.core
from azureml.core import Workspace, Experiment, Datastore
from azureml.widgets import RunDetails

# Check core SDK version number
print("SDK version:", azureml.core.VERSION)
```

```
SDK version: 1.51.0
```

Figure 6. Import Azure SDK

```python
from azureml.pipeline.core import Pipeline
from azureml.pipeline.steps import PythonScriptStep

print("Pipeline SDK-specific imports completed")
```

```
Pipeline SDK-specific imports completed
```

Figure 7. Import Azure Pipeline-specific SDK

```python
ws = Workspace.from_config()
print(ws.name, ws.resource_group, ws.location, ws.subscription_id, sep = '\n')

# Default datastore
def_blob_store = ws.get_default_datastore()
# The following call GETS the Azure Blob Store associated with the workspace.
def_blob_store = Datastore(ws, "workspaceblobstore")
print("Blobstore's name: {}".format(def_blob_store.name))
```

```
dp-tweets
my_mlops_prj
eastus2
34a65394-13fc-43ed-8e03-a0f81df6e347
Blobstore's name: workspaceblobstore
```

Figure 8. Azure user workspace configuration

```
from azureml.core import Dataset
from azureml.data.data_reference import DataReference


tweet1 = "./data/depressive_tweets.csv"
tweet2 = "./data/random_tweets.csv"
modelsvm = "./models/model_svm1.pkl"

#Uploading Depressive Tweets
with open(tweet1, "r") as f:
   # get_default_datastore() gets the default Azure Blob Store associated with the workspace.
   # Here we are reusing the def_blob_store object we obtained earlier
    def_blob_store.upload_files([tweet1], overwrite=True)
print("Depressive Tweets: Upload call completed")

#Uploading Random Tweets
with open(tweet2, "r") as f:
   # get_default_datastore() gets the default Azure Blob Store associated with the workspace.
   # Here we are reusing the def_blob_store object we obtained earlier
    def_blob_store.upload_files([tweet2], overwrite=True)
print("Random Tweets: Upload call completed")
```

Figure 9. Dataset upload to Azure Blob Storage

```
cts = ws.compute_targets
for ct in cts:
    print(ct)
```

✓

```
x221040381
cpu-cluster
cpu-cluster-ml
```

Figure 10: Check Available compute targets

```
from azureml.core.compute import ComputeTarget, AmlCompute
from azureml.core.compute_target import ComputeTargetException

# Attempts to retrieve an existing compute target with the specified name
aml_compute_target = "cpu-cluster"
try:
    aml_compute = AmlCompute(ws, aml_compute_target)
    print("found existing compute target.")
except ComputeTargetException:
    print("creating new compute target")
#If the compute target doesn't exist, the code proceeds to create a new compute target named
    provisioning_config = AmlCompute.provisioning_configuration(vm_size = "STANDARD_D2_V2",
                                                                 min_nodes = 1,
                                                                 max_nodes = 4)
    aml_compute = ComputeTarget.create(ws, aml_compute_target, provisioning_config)
    aml_compute.wait_for_completion(show_output=True, min_node_count=None, timeout_in_minutes=20)

print("Azure Machine Learning Compute attached")

```

✓

```
found existing compute target.
Azure Machine Learning Compute attached
```

Figure 11: AML Compute Provisioning for the pipeline

```
from azureml.core.runconfig import RunConfiguration
from azureml.core.conda_dependencies import CondaDependencies
from azureml.core.runconfig import DEFAULT_CPU_IMAGE

# create a new runconfig object
run_config = RunConfiguration()

# enable Docker
run_config.environment.docker.enabled = True

# set Docker base image to the default CPU-based image
run_config.environment.docker.base_image = DEFAULT_CPU_IMAGE

# use conda_dependencies.yml to create a conda environment in the Docker image for execution
run_config.environment.python.user_managed_dependencies = False

# specify CondaDependencies obj
#run_config.environment.python.conda_dependencies = CondaDependencies.create(conda_packages=['scikit-learn'])
run_config.environment.python.conda_dependencies = CondaDependencies.create(conda_packages=["pip", "python=3.8", "pandas==1.3.4", "numpy", "matplotlib"
pip_packages=['blis==0.4.1',
'certifi==2021.10.8',
'charset-normalizer==2.0.7',
'click==8.0.3',
'cycler==0.11.0',
'cymem==2.0.6',
'fonttools==4.28.1',
'ftfy==6.0.3',
'idna==3.3',
'joblib==1.1.0'
```

Figure 12: Run Configuration for conda environment creation in the docker image for execution

```
from azureml.pipeline.core import PipelineData
from azureml.pipeline.steps import PythonScriptStep
from azureml.data.data_reference import DataReference

# Uses default values for PythonScriptStep construct.

source_directory = './src'
print('Source directory for the step is {}.'.format(os.path.realpath(source_directory)))
# Referencing Depression tweets
dep_data = DataReference(
        datastore=def_blob_store,
        data_reference_name="dep_data",
        path_on_datastore=os.path.join("dep_data","depressive_tweets.csv"),
    )
print("DataReference object1 created")
#Referencing Random tweets
rand_data = DataReference(
        datastore=def_blob_store,
        data_reference_name="rand_data",
        path_on_datastore=os.path.join("rand_data","random_tweets.csv"),
    )
print("DataReference object2 created")
```

Figure 13: Pass DataReference object to the pipeline to access datasets across pipeline stages

```
26
27  step1 = PythonScriptStep(name="preprocess_step",
28                           script_name="preprocess.py",
29                           arguments=["--dep_data", dep_data, "--rand_data", rand_data, "--preprocessed_data",
30                           preprocessed_data],
31                           inputs = [dep_data, rand_data],
32                           outputs= [preprocessed_data],
33                           compute_target=aml_compute,
34                           source_directory=source_directory,
35                           runconfig=run_config,
36                           allow_reuse=True)
37    print("Step1 created")
```

```
Source directory for the step is /mnt/batch/tasks/shared/LS_root/mounts/clusters/cpu-cluster-ml/code/Users/x22104038/src.
DataReference object1 created
DataReference object2 created
Step1 created
```

Figure 14: Pipeline Step1 creation – preprocess_step

```
# For this step, we use a different source_directory
source_directory = './src'
print('Source directory for the step is {}.'.format(os.path.realpath(source_directory)))
# Creating DataReference for Preprocessed Data
preprocessed_data = DataReference(
        datastore=def_blob_store,
        data_reference_name="preprocessed_data",
        path_on_datastore=os.path.join("preprocessed_data.csv"),
    )
print("DataReference object4 created")

#This step is intended for training a machine learning model using the preprocessed data
# All steps use the same Azure Machine Learning compute target as well
step2 = PythonScriptStep(name="training_step",
                         script_name="modeltrain.py",
                         arguments=["--preprocessed_data", preprocessed_data],
                         inputs = [preprocessed_data],
                         compute_target=aml_compute,
                         source_directory=source_directory,
                         runconfig=run_config)
step2.run_after(step1)
#The second step depends on the output of the first step, ensuring a sequential execution order in the pipeline

```

Figure 15: Pipeline Step2 creation – training_step

```
# For this step, we use yet another source_directory
# Predictions using a trained machine learning model
source_directory = './src'
print('Source directory for the step is {}.'.format(os.path.realpath(source_directory

step3 = PythonScriptStep(name="prediction",
                         script_name="prediction.py",
                         compute_target=aml_compute,
                         source_directory=source_directory,
                         runconfig=run_config)
step3.run_after(step2)
#This step is designed for making predictions using a trained machine learning model
# Prediction after Training
# list of steps to run
steps = [step1, step2, step3]

print("Step lists created")
```

Figure 16: Pipeline Step3 creation – prediction

```
# Syntax
# Pipeline(workspace,
#          steps,
#          description=None,
#          default_datastore_name=None,
#          default_source_directory=None,
#          resolve_closure=True,
#          _workflow_provider=None,
#          _service_endpoint=None)

pipeline1 = Pipeline(workspace=ws, steps=steps)
print ("Pipeline is built")
```

✓

```
Pipeline is built
```

Figure 17: Build Pipeline

```
pipeline1.validate()
print("Pipeline validation complete")
```

```
Pipeline validation complete
```

Figure 18: Validate Pipeline

```
# Submit syntax
# submit(experiment_name,
#        pipeline_parameters=None,
#        continue_on_step_failure=False,
#        regenerate_outputs=False)

pipeline_run1 = Experiment(ws, 'Depression_Tweets').submit(pipeline1, regenerate_outputs=False)
print("Pipeline is submitted for execution")
```

```
Created step preprocess_step [e043d517][9eb0fc25-c483-4c48-b466-784f00805a13], (This step is eligible to reuse a previous run's output)
Created step training_step [64bdee9d][c71a5ad9-84ca-4eef-a9f9-542fb94fb05e], (This step is eligible to reuse a previous run's output)
Created step prediction [3ecbcc88][9449445c-cfee-44c7-85e8-009bc5b59883], (This step is eligible to reuse a previous run's output)
Using data reference dep_data for StepId [a57e9120][5ec54ca7-e6ac-406f-802e-74fe8ac91e59], (Consumers of this data are eligible to reuse prior runs.)
Using data reference rand_data for StepId [5bdcd7d8][afe62eae-3fa3-4e15-9519-57ba4b7bcc08], (Consumers of this data are eligible to reuse prior runs.)
Using data reference preprocessed_data for StepId [9dba6346][20022763-cfd2-4e69-b7cc-1a12d4492b2b], (Consumers of this data are eligible to reuse prior runs.)
Submitted PipelineRun 71e05035-08d3-4e9f-904e-366e69c47c3d
Link to Azure Machine Learning Portal: https://ml.azure.com/runs/71e05035-08d3-4e9f-904e-366e69c47c3d?wsid=/subscriptions/34a65394-13fc-43ed-8e03-a0f81df6e347/resourcegroups/my_mlops_prj/workspaces/dp-tweets&tid=6edb49c1-bf72-4eea-8b3f-a7fd0a25b68c
Pipeline is submitted for execution
```

Figure 19: Submit the Pipeline and Click on the link to access the pipeline.

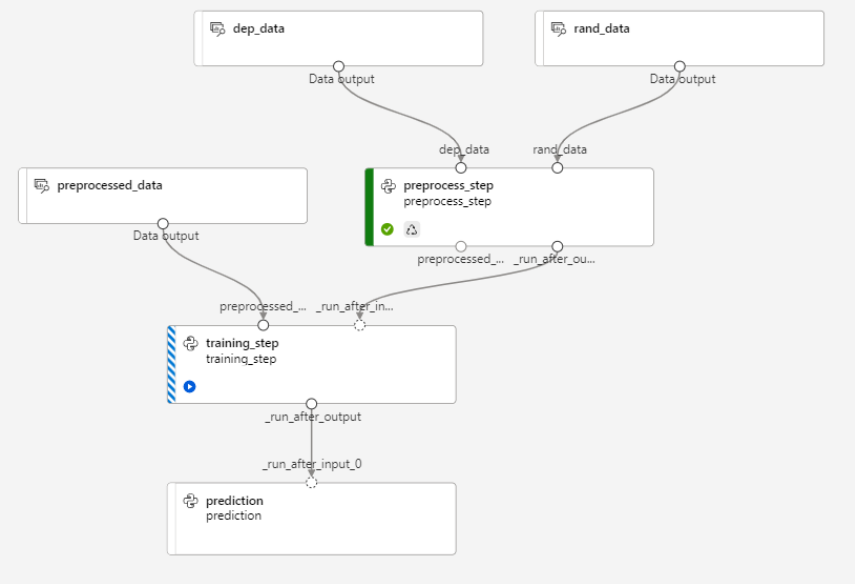

Figure 20: Running Pipeline