National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Dharma Teja Venkatesh Nagothi
Student ID: X22173897

School of Computing
National College of Ireland

Supervisor: Professor Vikas Sahni

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Dharma Teja Venkatesh Nagothi
Student ID:	X22173897
Programme:	Cloud Computing
Year:	2024
Module:	Msc Research Project
Supervisor:	Vikas Sahni
Submission Due Date:	25/04/24
Project Title:	Configuration Manual
Word Count:	1100
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:	Venkatesh
Date:	25/04/24

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:	Venkatesh
Date:	25/04/24
Penalty Applied (if applicable):	

Configuration Manual

Dharma Teja Venkatesh Nagothi
X22173897

1 Introduction

For local and Azure ML cloud reproduction of the study and its findings, the Configuration Manual provides all the procedures. This manual details the system specifications needed to run the program on Azure ML Studio, the dataset source, the Python machine learning packages used, the Azure ML cloud environment, and the project's Azure pipeline execution model.

2 System specifications

2.1 Hardware Configuration for the local run:

- Processor: Intel 11th Gen Core i7-1155G7
- RAM: 32 GB DDR4 RAM 3200MHz
- Storage (SSD): 512GB
- Operating System: Windows 11, 64-bit

2.2 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 following machine learning packages were put on the local system for the purpose of early code development prior to shifting to the cloud. To ease 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, utilize 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

```
env_name = "vit_env"
conda_deps = {
    "channels": ["defaults"],
    "dependencies": [
        "python=3.8",
        "numpy",
        "pandas",
        "matplotlib",
        "seaborn",
        "scikit-learn",
        "opencv-python",
        "tensorflow",
        "tensorflow-addons",
        {"pip": ["azureml-core", "azureml-dataset-runtime"]}
    ],
    "name": env_name
}
```

Figure 1: Pneumonia Detection using MLHOps.

5 Dataset

The data collecting process includes gathering of dataset for pneumonia detection from Kaggle as provided below. This data collection contains 1,583 (27.0) images with pneumonia class and 4,280 (73.0) normal images.Dincer (n.d.)

6 Azure ML Configuration

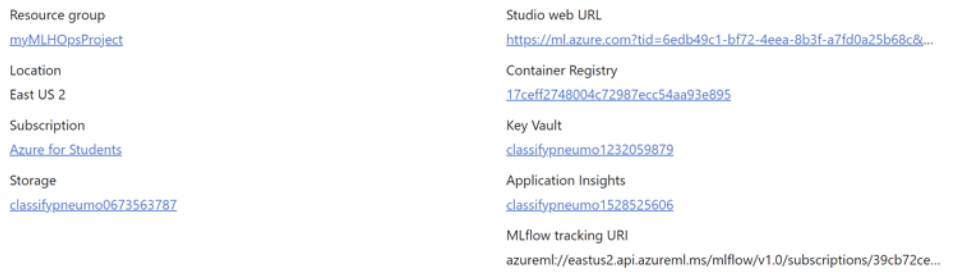

Figure 2: AZURE account details.

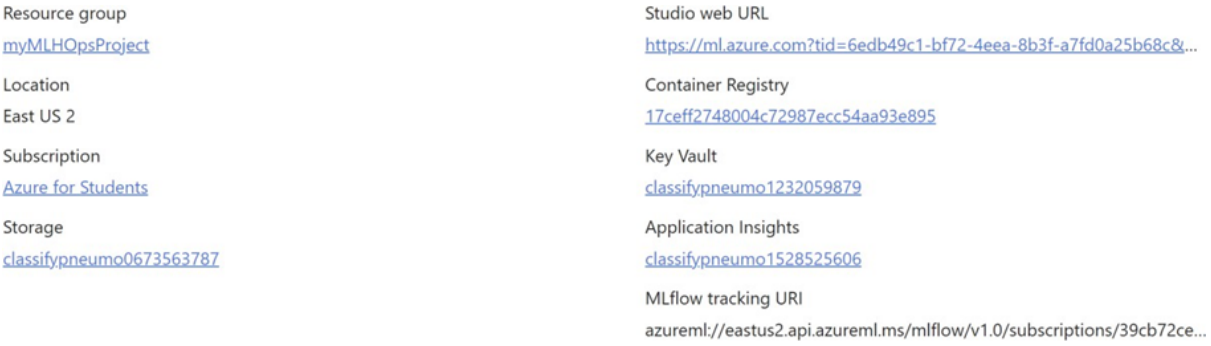

Figure 3: AZURE ML group resource group

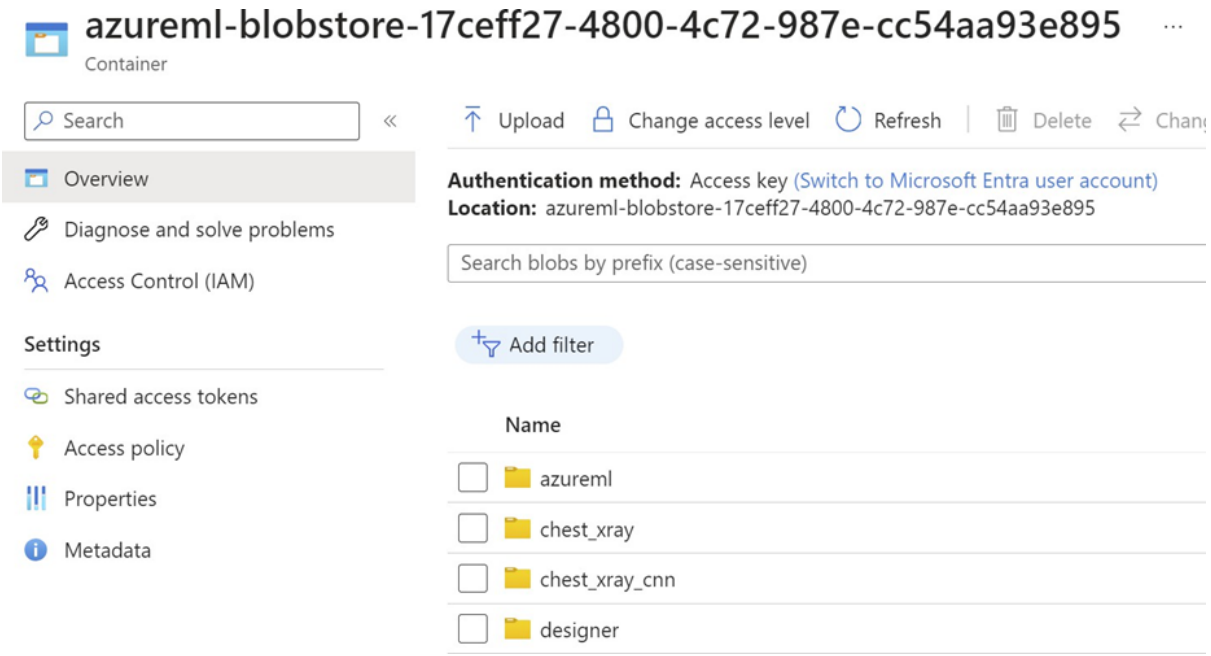

Figure 4: AZURE Blob Storage

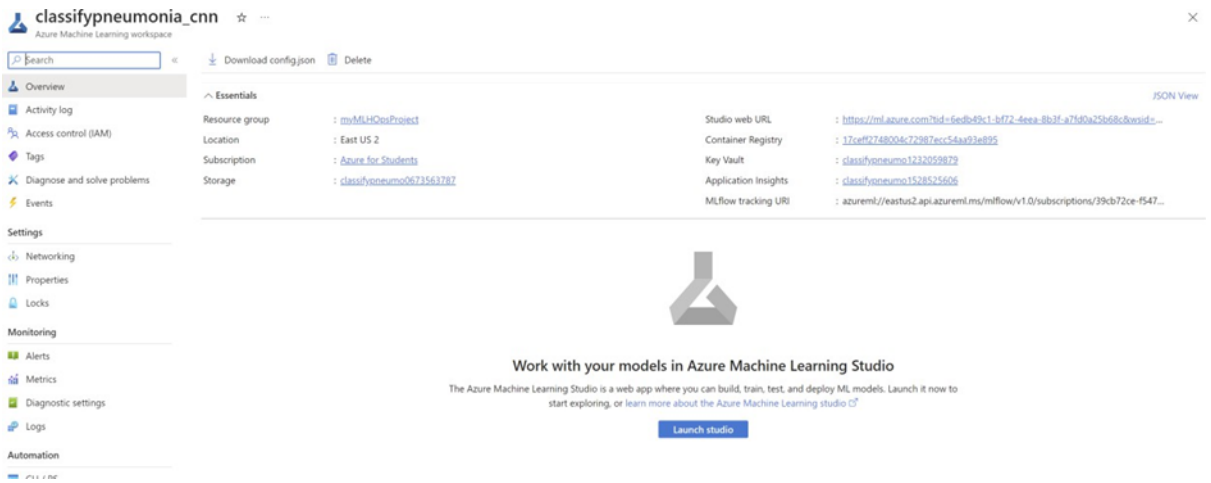

Figure 5: Launch Azure Machine Learning Studio

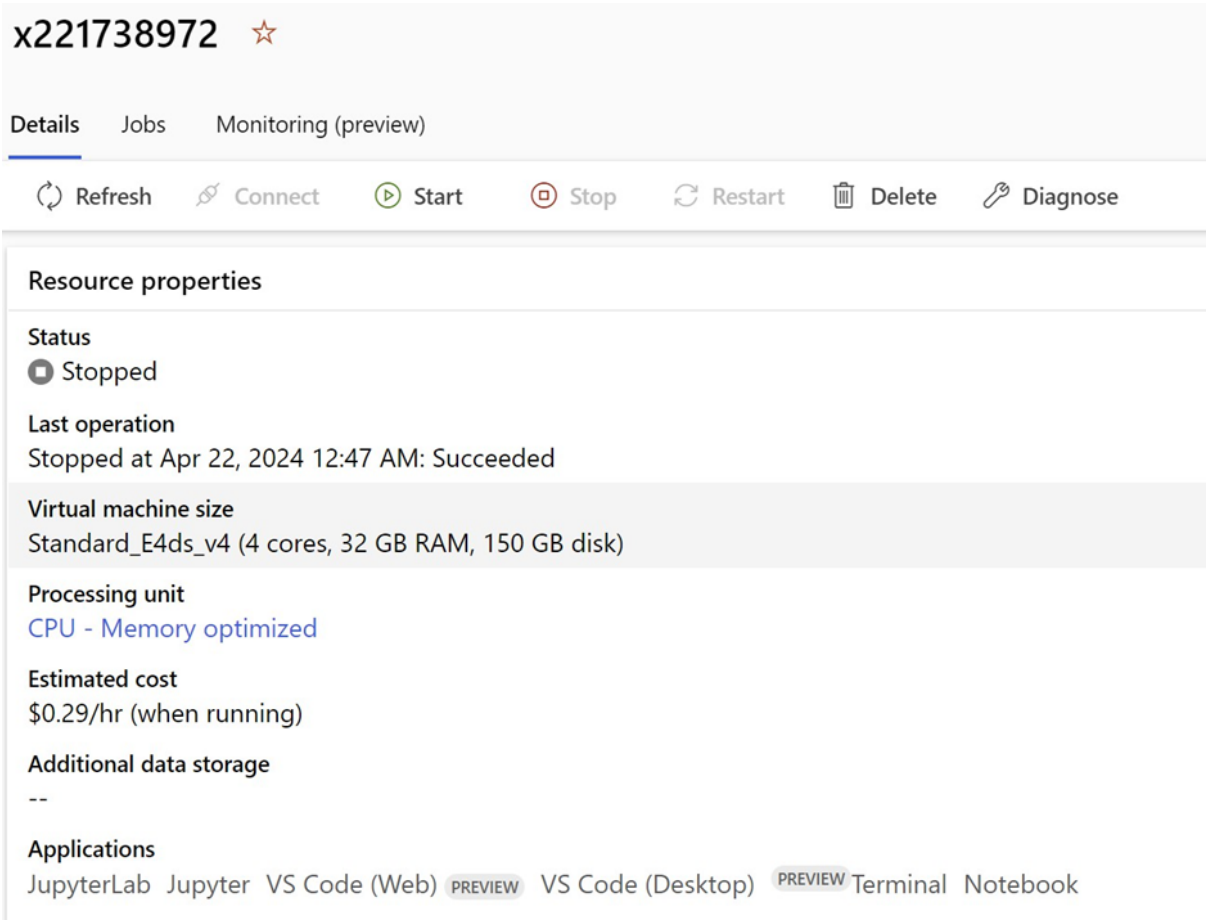

Figure 6: Compute Instance for the Jupyter notebook

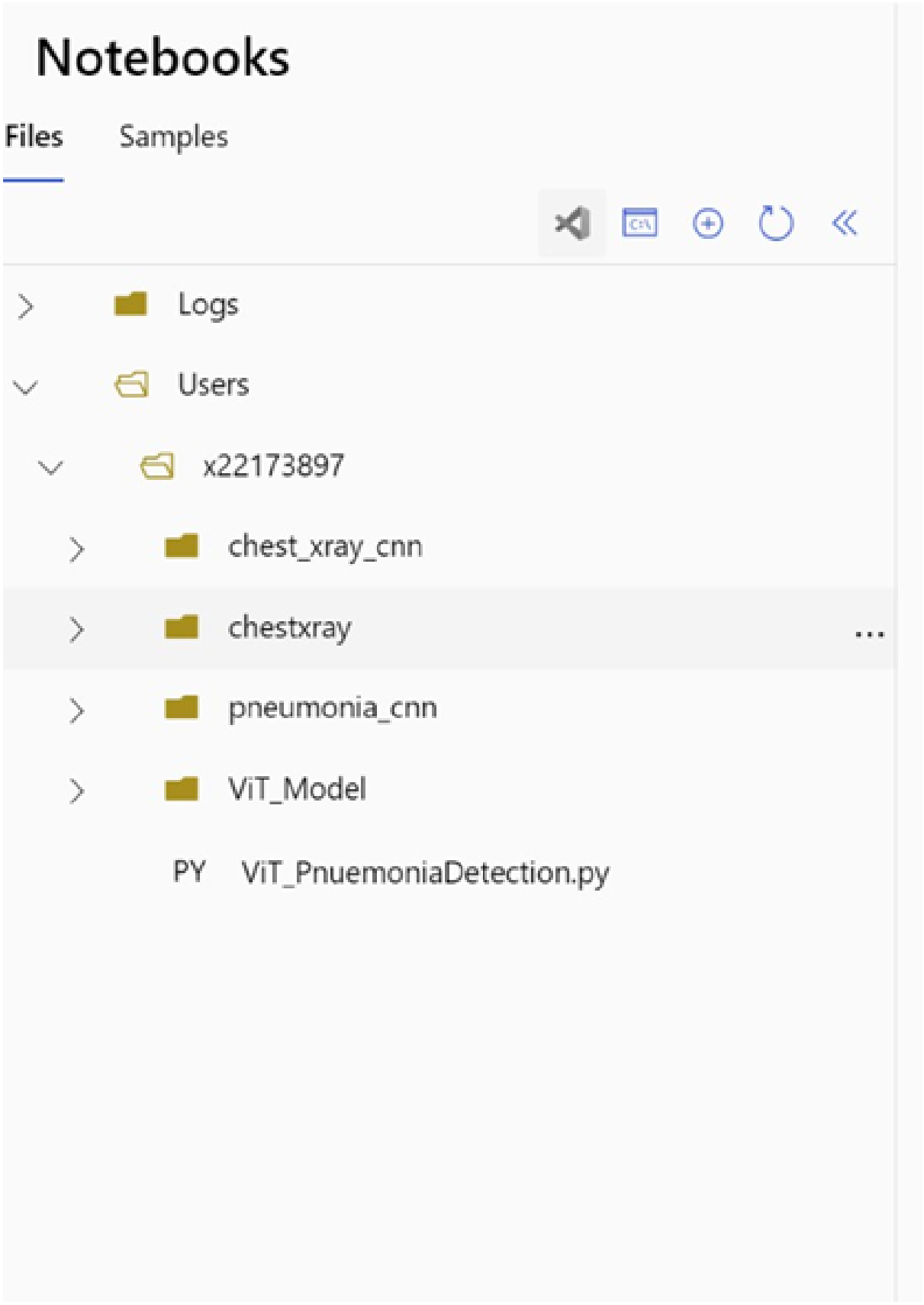

Figure 7: Jupyter notebook directory

```
ws = Workspace.from_config()

# Choose a name for your CPU cluster
cpu_cluster_name = "cpu-cluster"

# Verify that the cluster does not exist already
try:
    cpu_cluster = ComputeTarget(workspace=ws, name=cpu_cluster_name)
    print('Found existing cluster, use it.')
except ComputeTargetException:
    compute_config = AmlCompute.provisioning_configuration(vm_size='STANDARD_D2_V2',
                                                           max_nodes=4)
    cpu_cluster = ComputeTarget.create(ws, cpu_cluster_name, compute_config)

cpu_cluster.wait_for_completion(show_output=True)
```

Figure 8: Azure Compute Target for Pipeline

```
from azureml.core import Workspace, Datastore, Dataset, Experiment, ScriptRunConfig, Environment
from azureml.core.conda_dependencies import CondaDependencies
# Create the environment directly
env_name = "vit_env"
conda_deps = {
    "channels": ["defaults"],
    "dependencies": [
        "python=3.8",
        "numpy",
        "pandas",
        "matplotlib",
        "seaborn",
        "scikit-learn",
        "opencv-python",
        "tensorflow",
        "tensorflow-addons",
        {"pip": ["azureml-core", "azureml-dataset-runtime"]}
    ],
    "name": env_name
}

#env = Environment.from_conda_specification(name=env_name, conda_dependencies=conda_deps)
env = Environment.from_conda_specification(name=env_name, file_path='')
```

Figure 9: Assign Pipeline Execution Environment

```
# Create an Azure Blob Datastore using shared key credential
blob_datastore_name = 'pneumoniablobstore'
account_name = 'classifypneumo0673563787'
account_key = 'uKN2XaRdccI6pB4IVqg1ymvoT+Ngf0DLati9pCapa8YDGNIqxtznoWIWen5aw6flFjGmiLiDMW4W+AStPnxXDg=='

try:
    blob_datastore = Datastore.register_azure_blob_container(
        workspace=ws,
        datastore_name=blob_datastore_name,
        account_name=account_name,
        container_name='azureml-blobstore-17ceff27-4800-4c72-987e-cc54aa93e895',
        account_key=account_key
    )
except Exception as e:
    if 'Another datastore with the same name already exists' in str(e):
        existing_datastore = Datastore.get(ws, datastore_name=blob_datastore_name)
        existing_datastore.unregister()
        blob_datastore = Datastore.register_azure_blob_container(
            workspace=ws,
            datastore_name=blob_datastore_name,
            account_name=account_name,
            container_name='azureml-blobstore-17ceff27-4800-4c72-987e-cc54aa93e895',
            account_key=account_key
        )
```

Figure 10: Azure Blob Datastore Access

```
# Define pipeline steps
data_prep_step = PythonScriptStep(
    name="Data Preparation",
    script_name="data_preparation.py",
    arguments=["--dataset", dataset.as_named_input("raw_data"),
               "--prepared_data", prepared_data],
    outputs=[prepared_data],
    compute_target=cpu_cluster,
    source_directory="Scripts/",
    runconfig=run_config,
    allow_reuse=True
)
```

Figure 11: Pipeline Step - 1 – Data Preparation

```
model_training_step = PythonScriptStep(
    name="Model Training",
    script_name="vit_model_training.py",
    arguments=["--prepared_data", prepared_data,
               "--model_data", model_data],
    inputs=[prepared_data],
    outputs=[model_data],
    compute_target=cpu_cluster,
    source_directory="Scripts/",
    runconfig=run_config,
    allow_reuse=True
)
```

Figure 12: Pipeline Step - 2 – ViT Model Training

```
model_evaluation_step = PythonScriptStep(
    name="Model Evaluation",
    script_name="model_evaluation.py",
    arguments=["--prepared_data", prepared_data,
               "--model_data", model_data],
    inputs=[prepared_data, model_data],
    compute_target=cpu_cluster,
    source_directory="Scripts/",
    runconfig=run_config,
    allow_reuse=True
)
```

Figure 13: Pipeline Step - 3 – Model Evaluation

```
# Create the pipeline
pipeline_steps = [data_prep_step, model_training_step, model_evaluation_step]
pipeline = Pipeline(workspace=ws, steps=pipeline_steps)
```

Figure 14: Build Pipeline

```
# Submit the pipeline run
experiment = Experiment(ws, "vit_pneumonia_detection")
run = experiment.submit(pipeline)
status = run.get_status()
print("Pipeline run status:", status)
```

Figure 15: Submit Pipeline and Click on the link to access the pipeline.

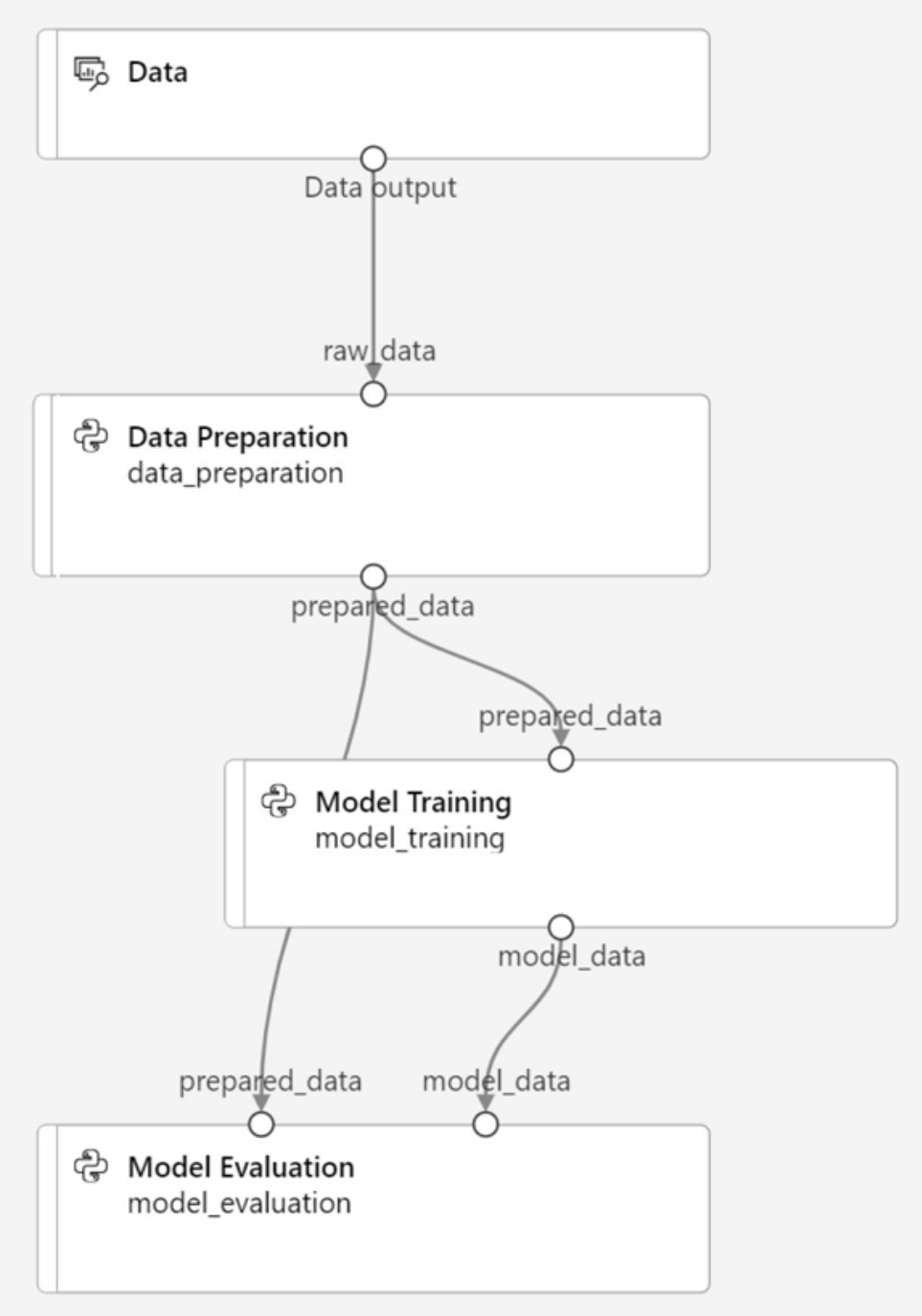

Figure 16: Pipeline Execution.

References

Dincer, T. (n.d.). Labeled chest x-ray images, https://www.kaggle.com/datasets/tolgadincer/labeled-chest-xray-images.