

Configuration Manual

MSc Research Project MSc in Cloud Computing

Hui Huang Student ID: 22180966

School of Computing National College of Ireland

Supervisor:

Ahmed Makki

National College of Ireland



MSc Project Submission Sheet

School of Computing

Student Name:	Hui Huang
Student ID:	x22180966
Programme:	MSc in Cloud Computing Year:2024-2025
Module:	Research Project
	Ahmed Makki
Submission Due Date:	12 th December 2024
Project Title:	Enhancing the Efficiency of Heart Disease Prediction Using Cloud Machine Learning Techniques

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:	Hui Huang
Date:	

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	On	ly

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

Configuration Manual

Hui Huang Student ID: x22180966

1 Introduction

This configuration manual outlines the implementation of heart disease prediction models using Azure Machine Learning Studio. Local environment control group implementations were executed on the Jupyter Notebooks on the Azure ML studio using traditional implementing scripting with Python. It provides comprehensive guidance for setting up cloud-based machine learning environments that range from basic single-node setups to advanced distributed computing systems for reproducing the cloud-based implementation. It aims to ensure standardized implementation practices for fair performance comparisons across various computational scenarios.

2 Dependencies Installation

The required dependencies are categorized as follows:

- (1)General Purpose
 - os, subprocess, time, psutil
- (2)Data Manipulation and Analysis
 - pandas, numpy
- (3)Visualization

matplotlib.pyplot, matplotlib.ticker

- (4) Machine Learning
 - 1) Preprocessing: StandardScaler, OneHotEncoder, LabelEncoder, SimpleImputer
 - 2) Data Splitting: train_test_split
 - 3) Pipeline: ColumnTransformer, Pipeline
 - 4) Algorithms: LogisticRegression, RandomForestClassifier
 - 5) accuracy_score, roc_auc_score, f1_score, recall_score, precision_score, auc
- (5)Imbalanced Data Handling

SMOTE

(6)Advanced ML and Optimization

Xgboost, XGBClassifier, optuna

- (7)Distributed Computing and Parallel Processing joblib, torch
- (8)Cloud and Azure Integration azure.ai.ml.MLClient, azure.ai.ml.Input, azure.ai.ml.command, azure.ai.ml.entities.Environment

3 Data Collection

The models were developed using the Heart Disease Health Indicators Dataset that from Kaggle (Teboul, 2019). The datasets was chosen for its widely range of related indicators and

features with moderate size that can be handled by the created compute instance for required distributing computing approaches.

- 1. Dataset Characteristics:
 - Source: Kaggle (CDC BRFSS Dataset)
 - Size: 200,000 records
 - Format: CSV file
 - Target Variable: Heart disease attack (binary)
 - Features include:
 - Demographic information
 - Health metrics (BMI, blood pressure)
 - Lifestyle factors
 - Medical history
- 2. Data Preparation:
 - Dataset downloaded from Kaggle platform
 - Data exploring and split the data into different sizes
 - Stored as 'train_200000.csv'
 - Located in the "../data/"directory
 - Automatically distributed across nodes in cloud implementation

4 Cloud Environment Configuration

- 1. Azure Portal Access
 - Login to Azure Portal
 - Navigate to Machine Learning services
- 2. Workspace Creation
 - Create new ML workspace(As shown in Figure 1)

Microsoft Azure		
Iome > Azure Machine Learning	g >	
Azure Machine Lea reate a machine learning workspace	irning	
Basics Networking Encry	ption Identity Tags Review + create	
Resource details		
	d to an Azure subscription, which is where billing hap esources, including the workspace you're about to cre groups \mathbb{C}^3	
Subscription * ①	Azure for Students	\checkmark
Resource group * ①		~
	attings like its storage connection, authentication, co	strings and more Learn more G
	ettings like its storage connection, authentication, co	ntainer, and more. Learn more \vec{c}
Configure your basic workspace se	ettings like its storage connection, authentication, con	ntainer, and more. Learn more of
Configure your basic workspace se	ettings like its storage connection, authentication, con	ntainer, and more. Learn more of
Configure your basic workspace se Name * ① Region * ①		ntainer, and more. Learn more of
Configure your basic workspace se Name * ① Region * ①		ntainer, and more. Learn more of
Workspace details Configure your basic workspace se Name * © Region * © Storage account * © Key vault * ©	West Europe Create new	ntainer, and more. Learn more of
Configure your basic workspace se Name * ③ Region * ④ Storage account * ④	West Europe	× ×

Figure 1 - Workplace Creation in Azure ML Studio

- 3. Compute Instance Setup(As shown in Figure 2)
 - Select "Compute" in workspace

- Create compute instance
- Configure: Standard_DS3_v2 (4 cores, 14 GB RAM, 28 GB disk)

Azure AI Machine Learnin								0 4
-	 National College of Ireland 	Create compute insta	ince					
 All workspaces 	Compute		0	Standard_D2	General purpose	6 cores	\$0.17/hr	
ි Home	① The "Kubernetes clusters" tab	Required settings	0	2 cores, 7GB RAM, 100GB storage				
G Model catalog	Compute Instances Com	Scheduling	0	Standard_D2_v2 2 cores, 7G8 RAM, 100G8 storage	General purpose	6 cores	\$0.14/hr	
luthoring			0	Standard_D2_v3	General purpose	6 cores	\$0.12/hr	
Notebooks	Choose from a selection of Cl	Security setional	0	2 cores, BGB RAM, 50GB storage				
Automated ML	+ New 🗘 Refrest	Applications	0	Standard_D2as_v4 2 cores, 8GB RAM, 16GB storage	General purpose	4 cores	\$0.12/hr	
Designer	Q. Search	optionel	0					
 Prompt flow 		Tags optional	0	Standard_D2s_v3 2 cores, 8GB RAM, 16GB storage	General purpose	6 cores	\$0.12/hr	
ssets	Name	Review	0	Standard_D3	General purpose	6 cores	\$0.34/hr	
🖟 Data	x221809663			4 cores, 14GB RAM, 200GB storage				
adol. 🖉	x221809662		0	Standard_D3_v2 4 cores, 14G8 RAM, 200G8 storage	General purpose	6 cores	\$0.27/hr	
6 Components	x221809661		0					
Pipelines			0	Standard_D4_v3 4 cores, 16GB RAM, 100GB storage	General purpose	6 cores	\$0.24/hr	
Environments			0	Standard_D4as_v4	General purpose	4 cores	\$0.23/hr	
Models				4 cores, 16GB RAM, 32GB storage				
Endpoints			0	Standard_D4s_v3 4 cores, 16GB RAM, 32GB storage	General purpose	6 cores	\$0.24/hr	
Aanage			0	TROUGH CANADACTAC A SAFEY - STORAG	-		10.000	
Compute			0	Standard_F2s_v2 2 cores, 4GB RAM, 16GB storage	Compute optimiz	10 cores	\$0.10/hr	
Monitoring			~					
Z Data Labeling			Rev	iew + Create Back Back	Next			

Figure 2 - Configure a 4 Cores Compute Instance on Azure

- Enable Jupyter Notebooks service
- 4. Environment Configuration
- (1) Workspace Connection
 - Using MLClient.from_config with DefaultAzureCredential provides secure and simplified authentication
 - Enables automatic credential management without explicit credential input
- (2) Environment Setup
 - Selected pre-configured PyTorch environment to ensure compatibility with distributed training
 - Version 1.13 with Python 3.8 offers stable support for our machine learning requirements
- (3) Data Asset Configuration
 - Chose Data Asset approach for efficient data handling in distributed environments
 - Enables version control and reduces data transfer overhead between nodes
 - Provides consistent data access across distributed training processes
- (4) Distributed Job Configuration
 - Implemented PyTorch distributed framework for efficient multi-node processing
 - Configured 2 nodes with 2 processes each for optimal resource utilization
 - Command structure allows flexible parameter passing and output management
- (5) Job Submission and Monitoring
 - Used create_or_update for job submission
 - Stream function enables real-time monitoring of training progress

Figures 3 and 4 present the implementation code for single-node and distributed computing configurations in Azure ML Studio's Jupyter Notebooks. Figure 3 demonstrates the default configuration using 1 node and 1 process, serving as our baseline implementation, while Figure 4 shows the distributed setup using 2 nodes with 2 processes per node to enable parallel processing capabilities. Similar procedures were referred for other distributing computing configurations for 1 node and 2 processes per node in the distributing experiment, while using 4 nodes and 1 process per node for improving the optimizing performance.

```
from azure.ai.ml import MLClient
1
                      rrom azure.ai.ml import MLClient
from azure.identity import DefaultAzureCredential
from azure.identity import DefaultAzureCredential
from azure.ai.ml import command, Input
from azure.ai.ml.entities import Environment
from azure.ai.ml.entities import Data
from azure.ai.ml.constants import AssetTypes
                                                                                                  orkplace
                      ml client = MLClient.from config(credential=DefaultAzureCredential())
 10
  11
  12
                     # create the environment
                         env = ml_client.environments.get(name="AzureML-ACPT-pytorch-1.13-py38-cuda11.7-gpu", version="1")
  13
  14
15
16
17
                     # define the data asset
data_asset = Data(
    name="heart-disease-data",
    path="./data/train_200000.csv",
    type=AssetTypes.URI_FILE
  18
  19
                    )
  20
21
22
23
24
25
26
                      data_asset = ml_client.data.create_or_update(data_asset)
                              create the training job
                      job = command(
    code="./scripts",
    command="python 1203-Inode_lprocess.py --input_data ${{inputs.data}} --output_model outputs/heart_disease_model --model_type all",
  27
                                    inputs={
    "data": Input(type="uri_file", path=data_asset.path)
  28
  29
30
                                  sector interpretation in the sector interpretation in the sector interpretation in the sector interpretation interpretation in the sector interpretation int
  31
32
  33
  34
                    )
  35
  36
  37
38
39
                     # uploade the job
returned_job = ml_client.jobs.create_or_update(job)
                      # check the job result
ml_client.job.stream(returned_job.name)
  40
41
```

Figure 3 - Default Configuration for the Cloud-based Implementation



Figure 4 - Configuration Script for Distributed Training

5 Experimental Result Reproduce

All experiments were conducted under controlled conditions using identical compute resources to ensure the fairness and consistency of all comparisons. The implementation details are as follows:

- 1. Hardware Configuration All experiments utilized Azure ML Studio's Standard_DS3_v2 compute instance (4 cores, 14 GB RAM, 28 GB disk) to maintain consistent hardware specifications across the local implementation to distributed computing implementations.
- 2. Results Collection
 - (1) Local Environment Results
 - All results from 1-node implementation are directly displayed within Jupyter Notebook interface
 - Performance metrics, resource utilization, and training times are captured in realtime during execution
 - Results are stored in variables and displayed through notebook cells
 - (2) Cloud-Based Results
 - Distributed training results are accessed through Azure ML Studio's experiment logs
 - Track the results through the jobs portal (shown in Figure 5)
 - Training logs cotains:
 - Training time
 - Resource utilization
 - Model performance metrics

Azure Al Machine Learning St	hudio O
=	National College of Ireland > Mh-project > Jobs > scripts > azure-distributed-2node-2process
$\leftarrow \text{ All workspaces}$	azure-distributed-2node-2process 🖉 🔅 🕲 Completed
 G Home G Model catalog 	Overview Metrics Images Child jobs Outputs + logs Code Explanations (preview) Fairness (preview) Monitoring
Authoring	🗘 Refresh 🖉 Edit and submit 🕨 Perform sweep 💧 Debug and monitor 🕂 Register model 🛞 Cancel 📓 Delete 🚽 Download all 🌑 Enable log streaming 💽 Word wrap
Notebooks Automated ML Control Designer Prompt flow	P (* B ad log_process_01x x > outputs - <td< th=""></td<>
Assets	Repaireent already satisfici pillon-6.2.8 in /opt/conde/envs/ptca/lib/python3.8/site-packages (from matplotlib) (9.4.0)
© Data ▲ Jobs ☆ Components ☆ Environments ● Endpoints ▲ Environments ● Models ● Compute ② Monitoring ② Data Labeling Ø Linked Services menses Ø Connections messes	Using cached sycle*-0.12093-home-amy.wh (0.5 MG) Using cached sycle*-0.13093-home-amy.wh (0.5 MG) Collecting kindslove*-1.0.1 do in <i>opt/condermary.com</i> (10.13/51k*-packages (from importile-resources)-3.2.0-metplotile) (3.11.0) Responses in the system of a set of the system statistic in the system set of the sy

Figure 5 - Experimental Results Review

References

Teboul, A. (2019) *Heart Disease Health Indicators Dataset*. [Dataset] Kaggle. Available at: <u>https://www.kaggle.com/datasets/alexteboul/heart-disease-health-indicators-dataset</u> [Accessed 9 December 2024].