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 ………………………………………………….………

Lecturer: …………………Ahmed Makki……………………………………………………………….………

Submission Due Date: ………………12th December 2024……………………………………………….………

Project Title: Enhancing the Efficiency of Heart Disease Prediction Using Cloud Machine Learning Techniques………..

Word Count: ……………720………………… **Page Count:** …………………6………..…….………

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: ……………………11th December 2024……………………………………………

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

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)

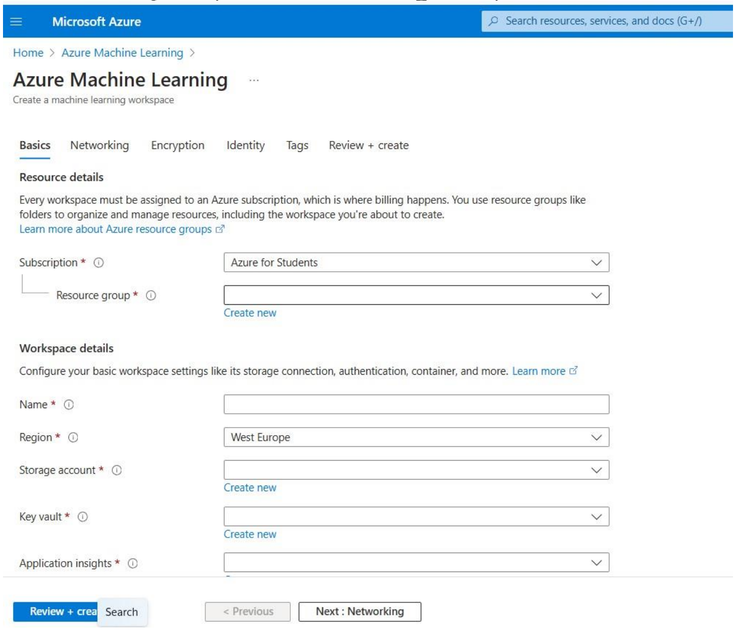

Figure 1 - Workplace Creation in Azure ML Studio

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)

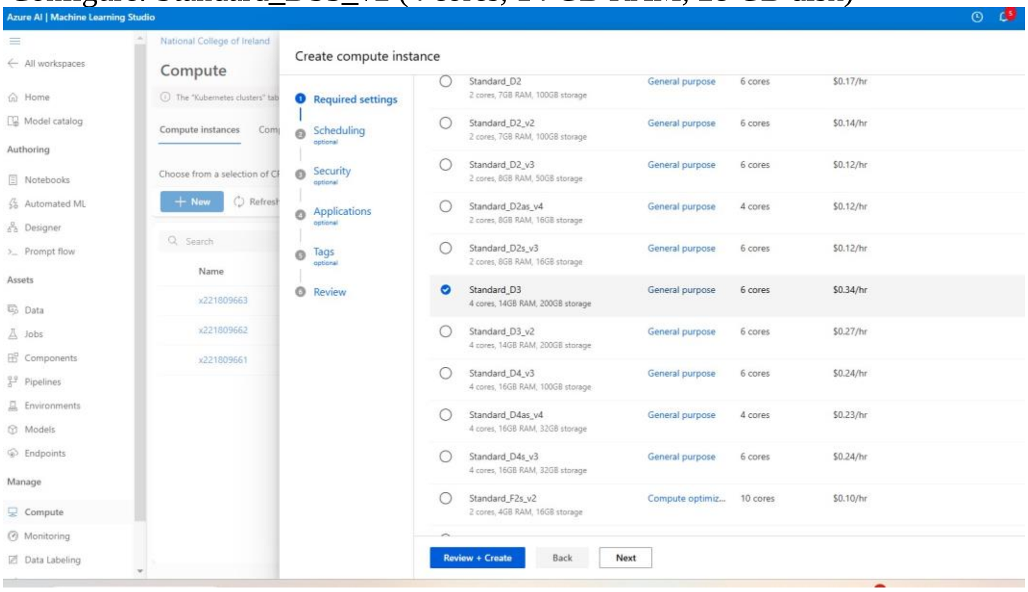

Figure 2 - Configure a 4 Cores Compute Instance on Azure

- 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.

```python
from 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

# connect to my workplace
ml_client = MLClient.from_config(credential=DefaultAzureCredential())

# create the environment
env = ml_client.environments.get(name="AzureML-ACPT-pytorch-1.13-py38-cuda11.7-gpu", version="1")

# define the data asset
data_asset = Data(
    name="heart-disease-data",
    path="../data/train_200000.csv",
    type=AssetTypes.URI_FILE
)

data_asset = ml_client.data.create_or_update(data_asset)

# create the training job
job = command(
    code="./scripts",
    command="python 1203-1node_1process.py --input_data ${{inputs.data}} --output_model outputs/heart_disease_model --model_type all",
    inputs={
        "data": Input(type="uri_file", path=data_asset.path)
    },
    environment=env,
    # specify the azure compute instance
    compute="x221809662",
    display_name="heart-disease-model-training"
)

# uploade the job
returned_job = ml_client.jobs.create_or_update(job)

# check the job result
ml_client.jobs.stream(returned_job.name)
```

Figure 3 - Default Configuration for the Cloud-based Implementation

```python
from azure.ai.ml import MLClient, command, Input
from azure.ai.ml.entities import Data, Environment
from azure.identity import DefaultAzureCredential
from azure.ai.ml.constants import AssetTypes


# connect to my workplace
ml_client = MLClient.from_config(credential=DefaultAzureCredential())

# create the environment
env = ml_client.environments.get(name="AzureML-ACPT-pytorch-1.13-py38-cuda11.7-gpu", version="1")

# define the data asset
data_asset = Data(
    name="heart-disease-data",
    path="../data/train_200000.csv",
    type=AssetTypes.URI_FILE
)
data_asset = ml_client.data.create_or_update(data_asset)
# create the distributed training job
job = command(
    code="./scripts",
    command="python 1203_azure_2node_2process.py "
            "--input_data ${{inputs.data}} "
            "--output_model outputs/heart_disease_model "
            "--model_type all "
            "--distributed",
    inputs={
        "data": Input(type="uri_file", path=data_asset.path)
    },
    environment=env,
    compute="x221809662",
    display_name="distributed-heart-disease-model-training",

    # allocate the number of the process and the node
    distribution={
        "type": "PyTorch",
        "process_count_per_instance": 2,
        "node_count": 2
    }
)

# uploade the job
returned_job = ml_client.jobs.create_or_update(job)
# check the job result
ml_client.jobs.stream(returned_job.name)
```

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 real-time 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

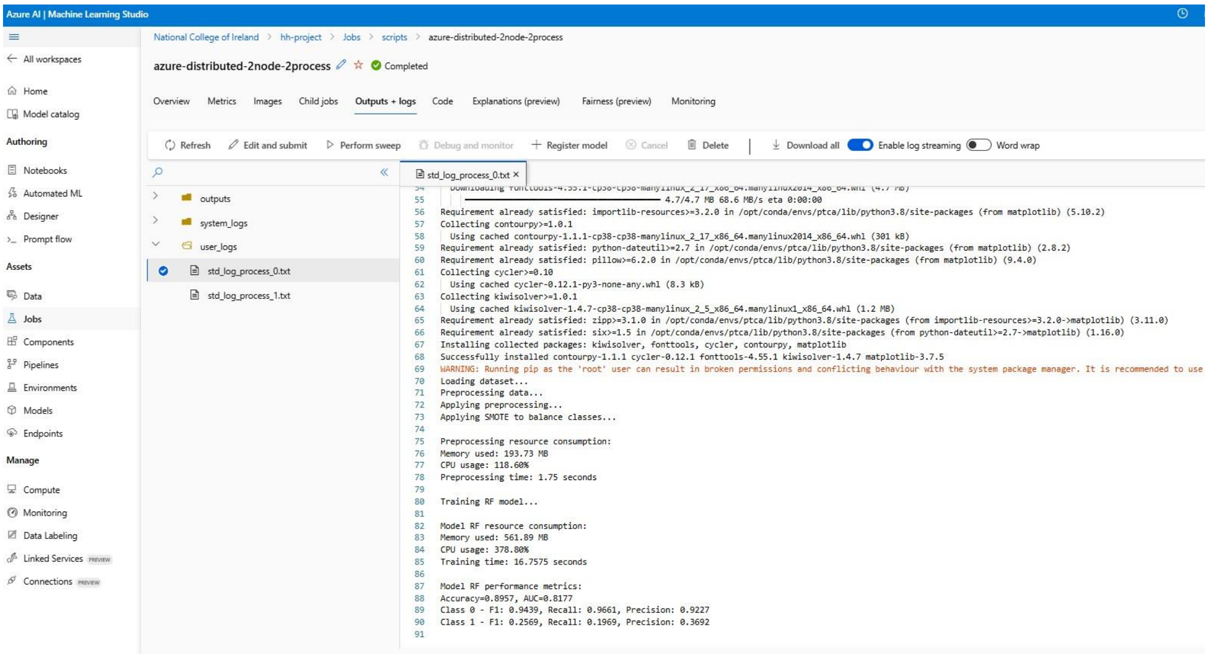

Figure 5 - Experimental Results Review

References

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