National College of Ireland

Predictive Modelling and Cold Start Mitigation in Function-as-a-Service Using BiLSTM with Multi-Head Mechanism

MSc Research Project
Cloud Computing

Mohan Sai Morla
Student ID: x23418613

School of Computing
National College of Ireland

Supervisor: Prof Aqeel Kazmi

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Mohan Sai Morla
Student ID:	x23418613
Programme:	Cloud Computing
Year:	2025-2026
Module:	MSc Research Project
Supervisor:	Prof Aqeel Kazmi
Submission Due Date:	11/12/2025
Project Title:	Predictive Modelling and Cold Start Mitigation in Function-as-a-Service Using BiLSTM with Multi-Head Mechanism
Word Count:	1074
Page Count:	13

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:	Mohan Sai Morla
Date:	11th December 2025

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

Mohan Sai Morla
x23418613

1 Introduction

This document provides an comprehensive overview of the software and tools used throughout the entire project. It provides a clear steps and guidelines for installing all required software dependencies for executing the project. Additionally it also provides an structured and systematic roadmap for successful project implementation by showcasing the mechanisms used to achieve the desired outcomes. Considering the extensive resource requirements and prerequisites associated with the Machine Learning algorithm and AWS Cloud with serverless functions.

2 System Configuration

This section outlines the Hardware and Software Requirements for the project.

2.1 Hardware & Platform specifications

Type	Tool & Platforms
Cloud VM	Amazon Web Service EC2 (Amazon Elastic Compute Cloud), Cloud9
Operating System	Linux Ubuntu 24
Container Technology	Docker version 28.3
Performance/Load Testing	Apache JMeter 5.6.3
FaaS	Openwhisk

2.2 Software specification

Software specifications	Details
Python	3.12 version
Google Colab & Cloud 9	For Machine Learning - Python 3.12
Visual Studio Code	Version: 1.97.0 IDE, OS: Windows 11

Note: This experiment was performed in Google Colab Notebook and Kaggle Notebook considering high resource utilisation. If anyone attempt to replicate the experiment on a local system, it is advisable to utilize Jupyter Notebooks.

GitHub Link for Project Code:

https://github.com/Mohannci/Research-Project-Artefacts

3 Implementation and Integration

3.1 Environment setup

I have used AWS for jmeter and openwhisk load testing, Kaggle Notebook for model training and testing and Google Colab notebook for data visualization.

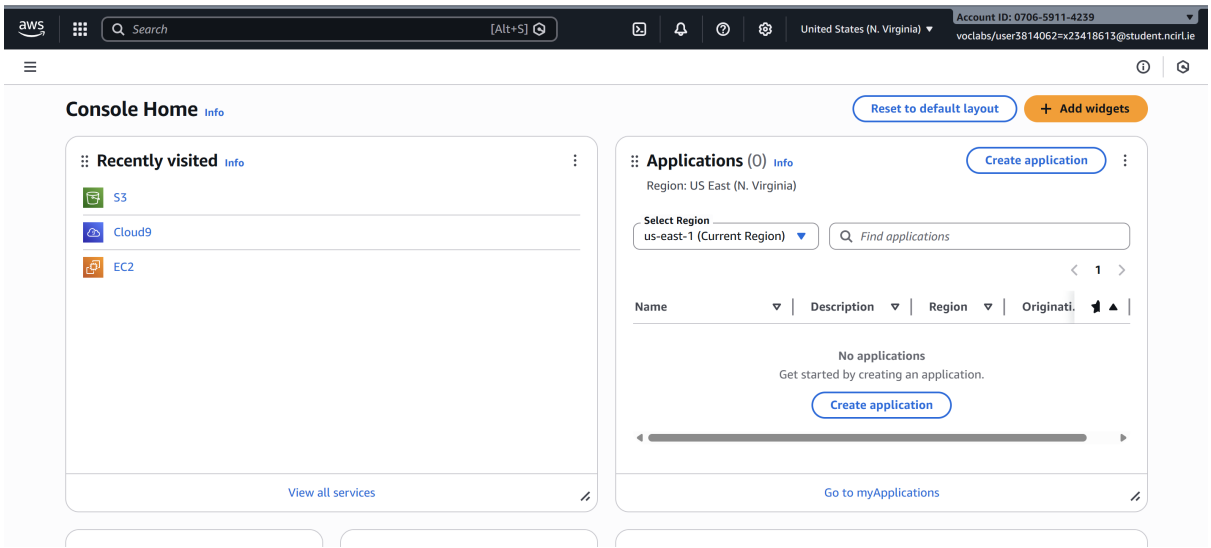


Figure 1: Aws Home Page

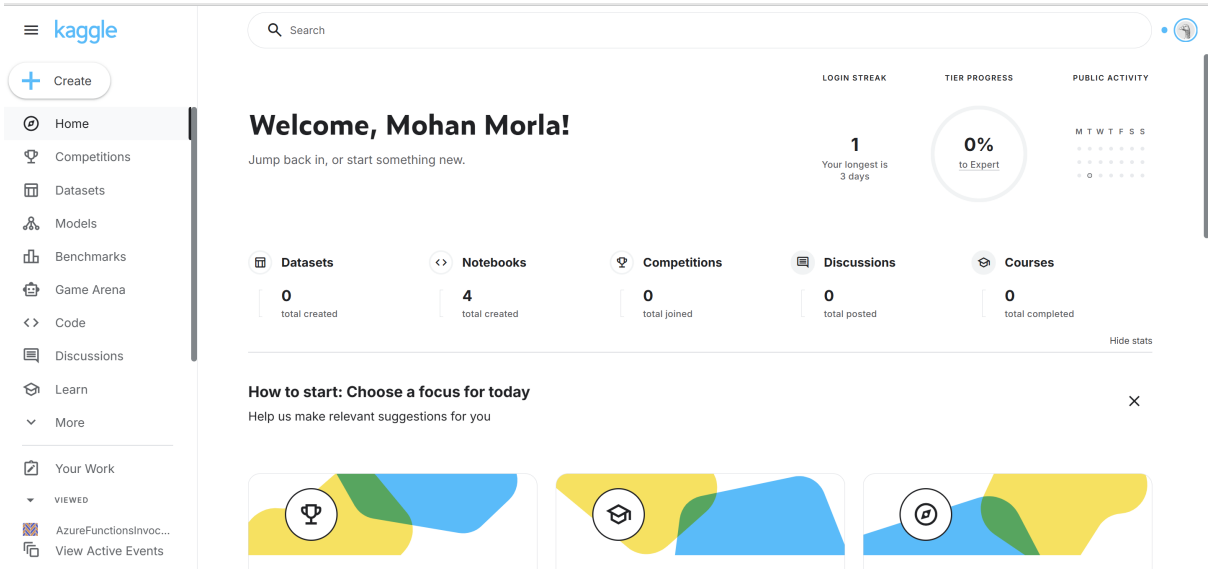


Figure 2: Kaggle Notebook home page

Figure 3: Colab notebook homepage

3.2 Software Packages and Libraries

The models were implemented using Google Colab Notebook, utilizing the Python programming language. The following packages and imports are essential for the proper functioning of the models:"

Python is used to programme the suggested model. The machine learning project is best suited for the Python programming language. Python is the best choice for machine learning applications due to its consistency, platform freedom, ease of use, accessibility to top-notch ML tools and frameworks, flexibility, huge community, and ease of use.

Pandas is an open source python library. It is a free and high-performance data analytics tool. With qualities like extreme user friendly and creating a table like format from given dataset which intern is very user friendly. It's a library or tool used in machine learning for a wide rage of fields like statistics, finance, economics etc.

Numpy:

Numpy(Numerical python) is a python library. It consist of multi-d array objects and a set/collection of routines for the existing array processing. Mathematical and logical operation in python are done with the help of numpy.

Sklearn:

Sklearn also known as Scikit learn is a python library used for pre-processing, visualization and cross-validation algorithm. It is a library built on Numpy, Scipy and Matplotlib. It is also used for data mining and analysis of data. Sklearn consist of various classification, regression and clustering algorithms.

Matplotlib:

Data Visualization in python can be done wth the help of Matplotlib. It is for creating two-dimensional graphs in ML from data set available in form of arrays. It makes use of GUI tool PyQt in python for creating user friendly graphs.

Keras & TensorFLow: Keras is a high-level, deep learning API developed by Google for implementing neural networks. It is written in Python and is used to make the implementation of neural networks easy. It also supports multiple backend neural network computation. TensorFlow is an open-source library for numerical computation, large-scale machine learning, deep learning, and other statistical and predictive analytics.

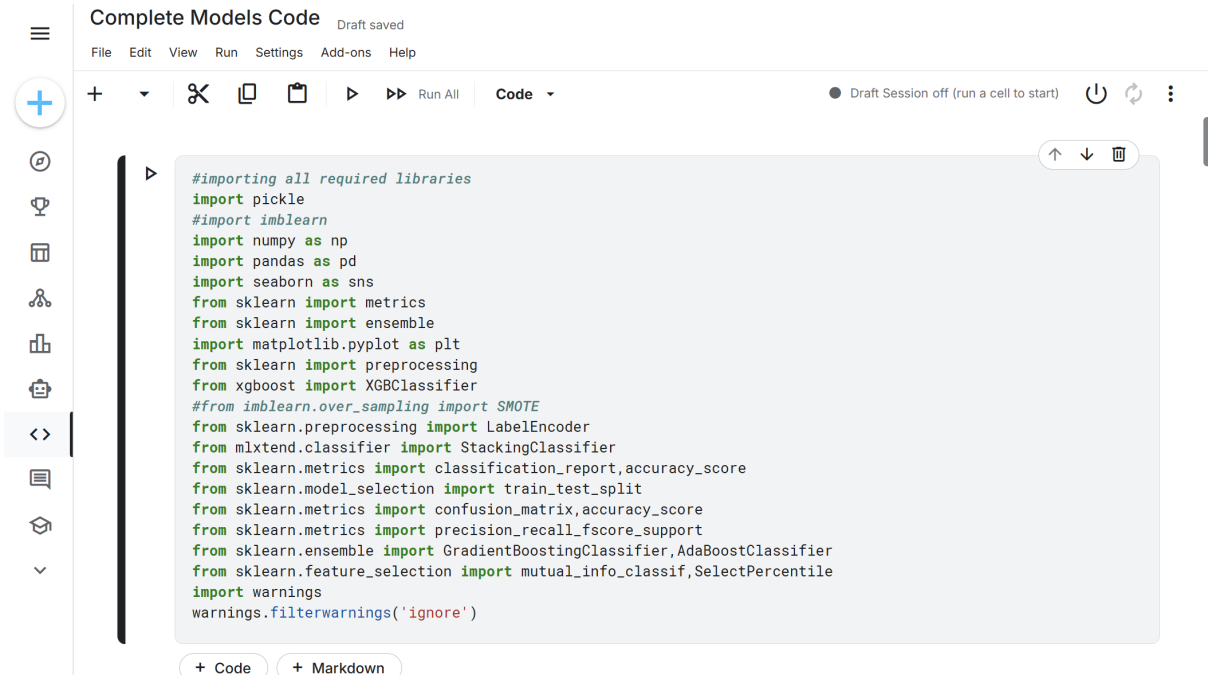


Figure 4: Importing packages in Kaggle notebook.

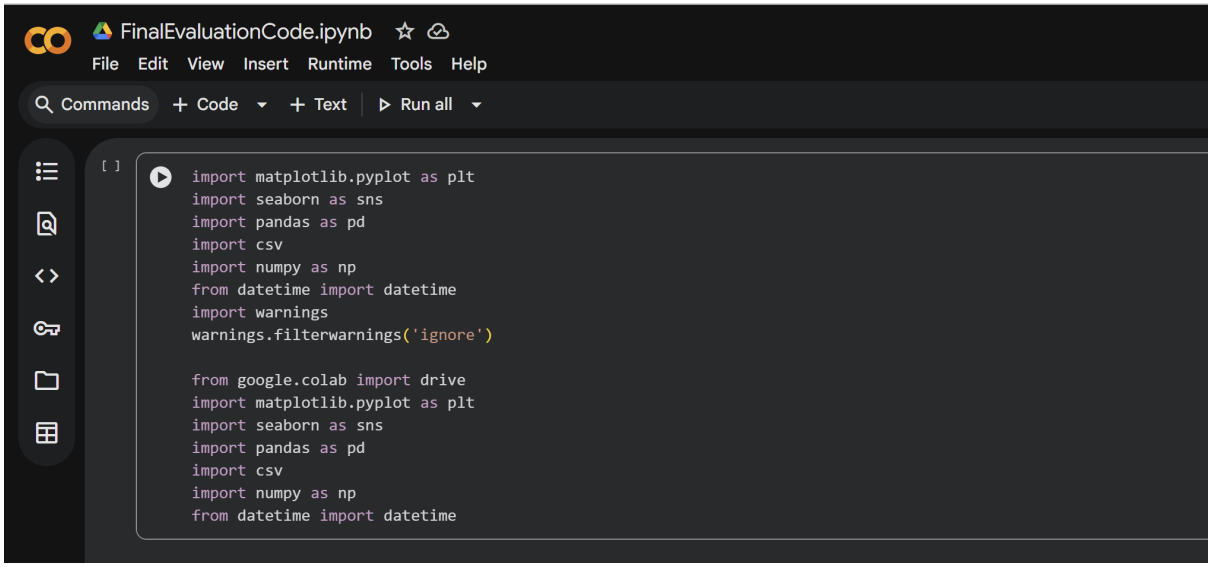


Figure 5: Importing packages in Colab notebook.

3.3 Project Execution Phases

This section has the details of my project Implementation and evaluation.

3.3.1 Data Cleaning

First I have read the data then I listed down columns it has and calculated percentage of inter-arrival times and at last checked for null values.

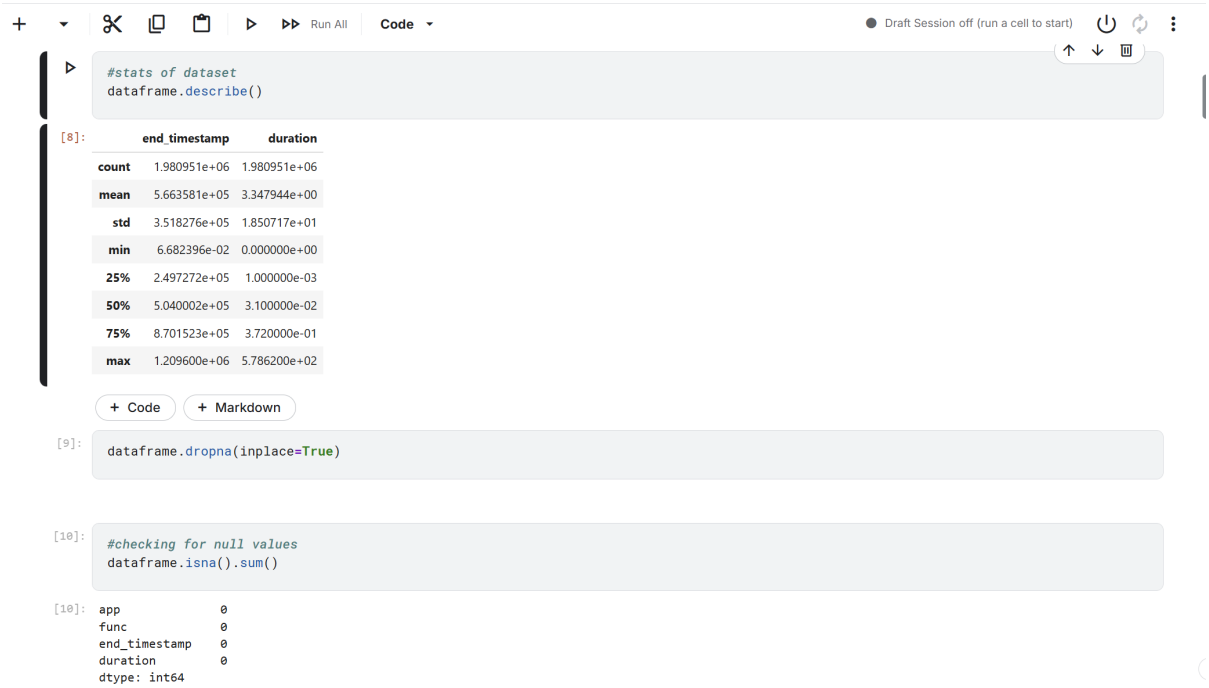


Figure 6: Data Cleaning

3.3.2 Data Preprocessing

In these phase I have added few columns to the dataset because dataset has only 4 columns which will lead models to predict values incorrectly so I have added additional columns like wait time, hours, minute, month, year so that models have enough attributes to predict inter-arrival time accurately.

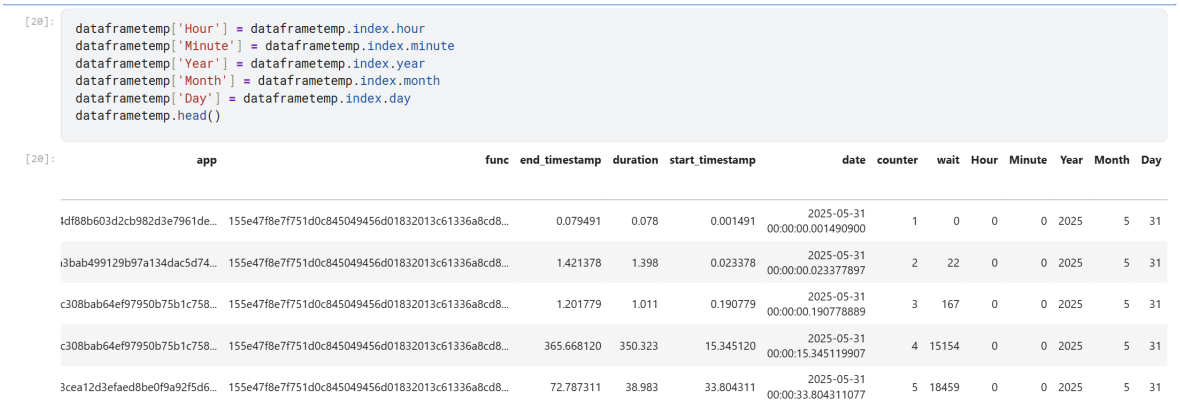


Figure 7: Data preprocessing.

3.3.3 Data Exploration

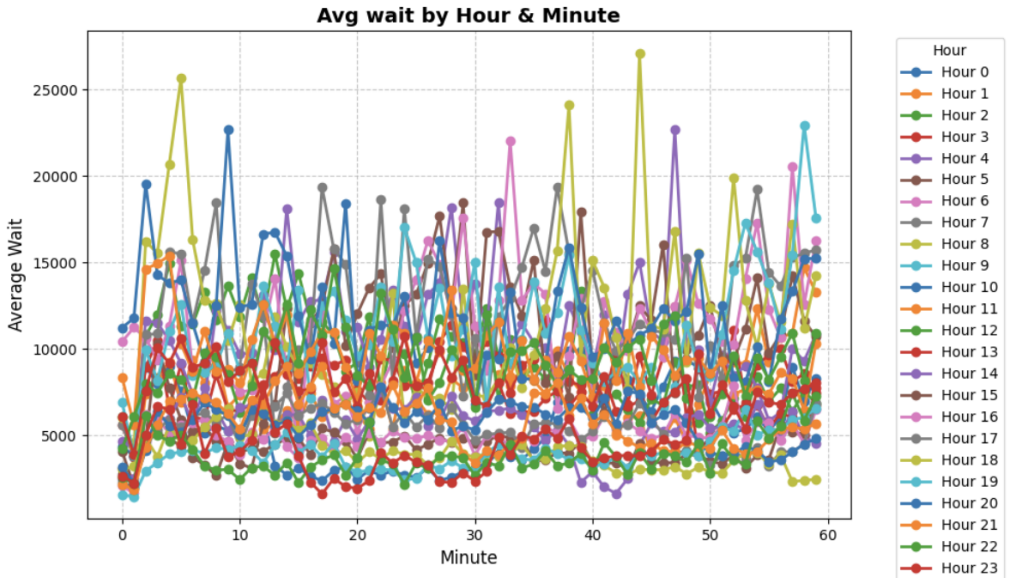
In these phase I have explored the data like what is the average inter-arrival time per the day. Reason behind this exploration is it helps models to learn inter-arrival time of functions Precisely.

```python
# Group by Hour and Minute, compute mean
edadf = dataframetemp[['Hour', 'Minute', 'wait']].groupby(['Hour', 'Minute']).mean().reset_index()

# Create a line plot for each Hour
plt.figure(figsize=(10, 6))
for hour in sorted(edadf['Hour'].unique()):
    subset = edadf[edadf['Hour'] == hour]
    plt.plot(subset['Minute'], subset['wait'], marker='o', linewidth=2, label=f'Hour {hour}')

# Add titles, labels, and legend
plt.title('Avg wait by Hour & Minute', fontsize=14, fontweight='bold')
plt.xlabel('Minute', fontsize=12)
plt.ylabel('Average Wait', fontsize=12)
plt.legend(title='Hour', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True, linestyle='--', alpha=0.7)

# Save the figure
file_name = "hour_minute_graph.png"
plt.tight_layout()
plt.savefig(file_name, dpi=300)
plt.show()
```



+ Code + Markdown

Figure 8: Data Exploration.

3.3.4 Training Deep Learning Models

I have used three deep learning models (LSTM, BiLSTM, Multi-head BiLSTM) out of these 3 Multi-head Lstm scored good mse and rmse scores and predicted 90.9% of warm starts.

Model: "functional_15"

Layer (type)	Output Shape	Param #	Connected to
input_layer_3 (InputLayer)	(None, 5, 1)	0	-
bidirectional_2 (Bidirectional)	(None, 5, 128)	33,792	input_layer_3[0]…
global_average_poo… (GlobalAveragePool…	(None, 128)	0	bidirectional_2[…
global_max_pooling… (GlobalMaxPooling1…	(None, 128)	0	bidirectional_2[…
reshape (Reshape)	(None, 1, 128)	0	global_average_p…
reshape_1 (Reshape)	(None, 1, 128)	0	global_max_pooli…
dense_6 (Dense)	(None, 1, 16)	2,064	reshape[0][0]
dense_8 (Dense)	(None, 1, 16)	2,064	reshape_1[0][0]
dense_7 (Dense)	(None, 1, 128)	2,176	dense_6[0][0]
dense_9 (Dense)	(None, 1, 128)	2,176	dense_8[0][0]
add (Add)	(None, 1, 128)	0	dense_7[0][0], dense_9[0][0]
activation (Activation)	(None, 1, 128)	0	add[0][0]
multiply (Multiply)	(None, 5, 128)	0	bidirectional_2[… activation[0][0]
global_average_poo… (GlobalAveragePool…	(None, 128)	0	multiply[0][0]
dense_10 (Dense)	(None, 8)	1,032	global_average_p…
dense_11 (Dense)	(None, 1)	9	dense_10[0][0]

Total params: 43,313 (169.19 KB)

Trainable params: 43,313 (169.19 KB)

Non-trainable params: 0 (0.00 B)

Figure 9: Multi-head BiLSTM model training

3.4 Testing and Evaluation

3.4.1 Phase 1: Deep Learning

Here we implement 3 experiments using 3 different algorithms namely : LSTM, BiLSTM, Multiview feature focused Bilstm. All these algorithms use the same Azure functions dataset.

These models will forecast 100 delay values which will be used by JMeter further to perform the load testing experiment.

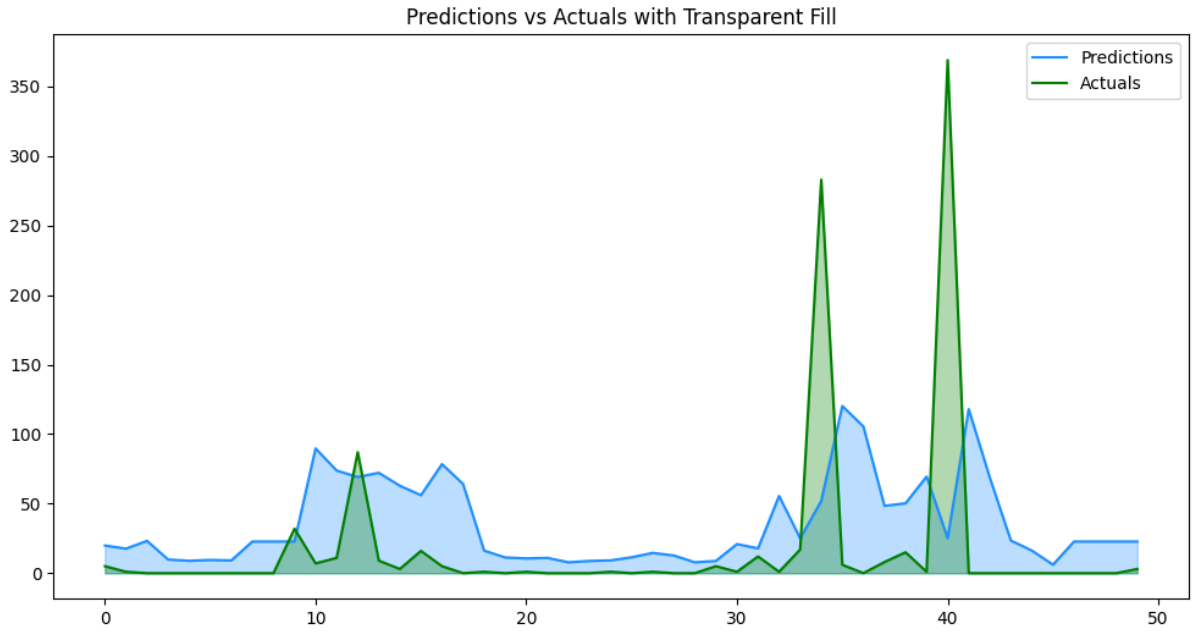


Figure 10: Training Data results

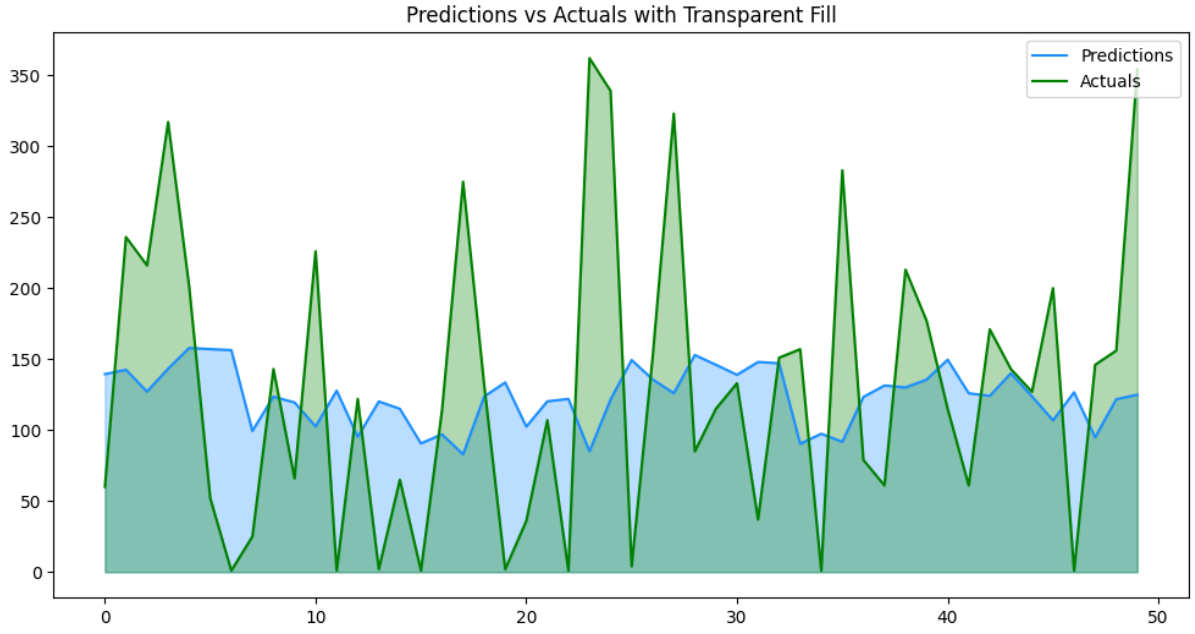


Figure 11: Testing Data Results

3.4.2 Phase 2: JMeter Load Testing

A total of 3 main experiments were conducted to validate the hypothesis put forward by this research. Phase 1 experiments simulated the machine learning models to forecast Wait times based on Azure Functions data. Phase 2 experiments were performed using this forecasted values as an input to custom controller function for getting the docker

warm. In the background a process consumes the machine learning predictions and heats function containers accordingly. Following are the steps involved in replicating the same one AWS.

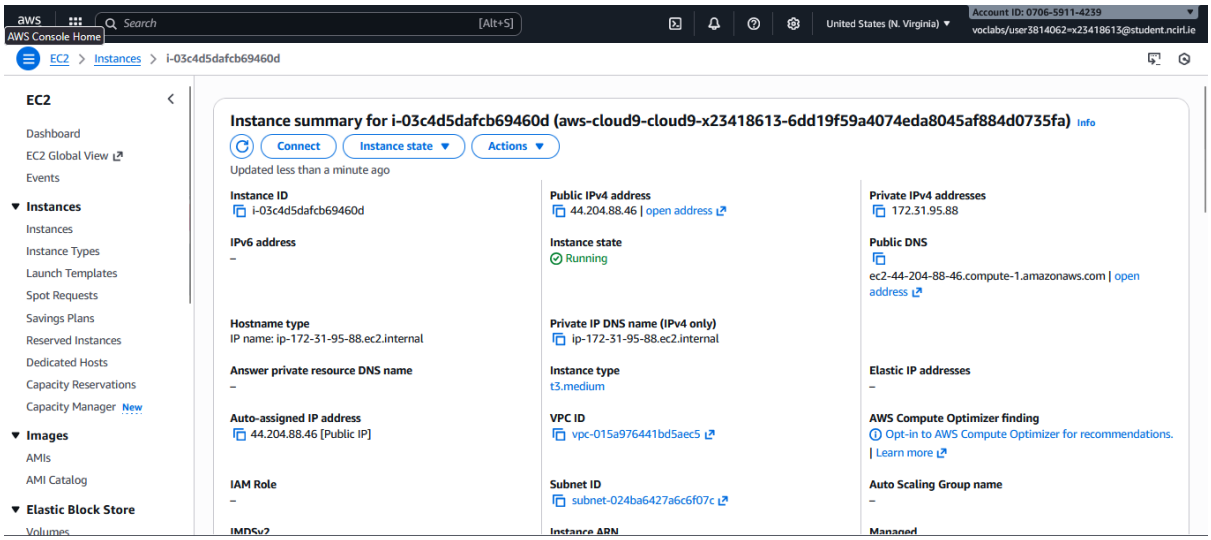


Figure 12: AWS EC2 Instance Created.

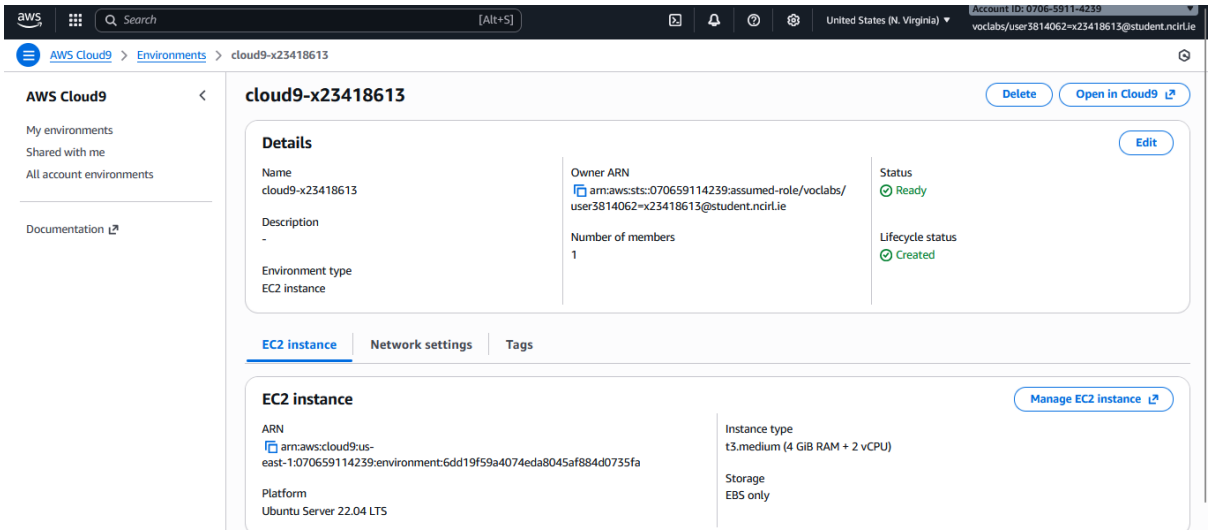


Figure 13: Cloud 9 Env Created.

3.4.3 Phase 3 : Running the JMeter Test

Step 1: Verify git is installed on the VM (which it should be) with the command git –version. Type sudo apt install git if it's not already installed.

Step 2: Now clone the projects codebase from the Github repository with the following command sudo git clone.

Step 3: Use the following commands to move the downloaded Jmeter files and executables to an appropriate location on the server

sudo unzip /apache-jmeter-5.6.3.zip

sudo mv /apache-jmeter-5.6.3 /jmeter
echo 'export PATH="$PATH: /jmeter/bin"' ¿¿ /.bashrc
source /.bashrc

Step 4: Verify Jmeter install on EC2 can be executed sudo /jmeter/bin/jmeter.sh –version

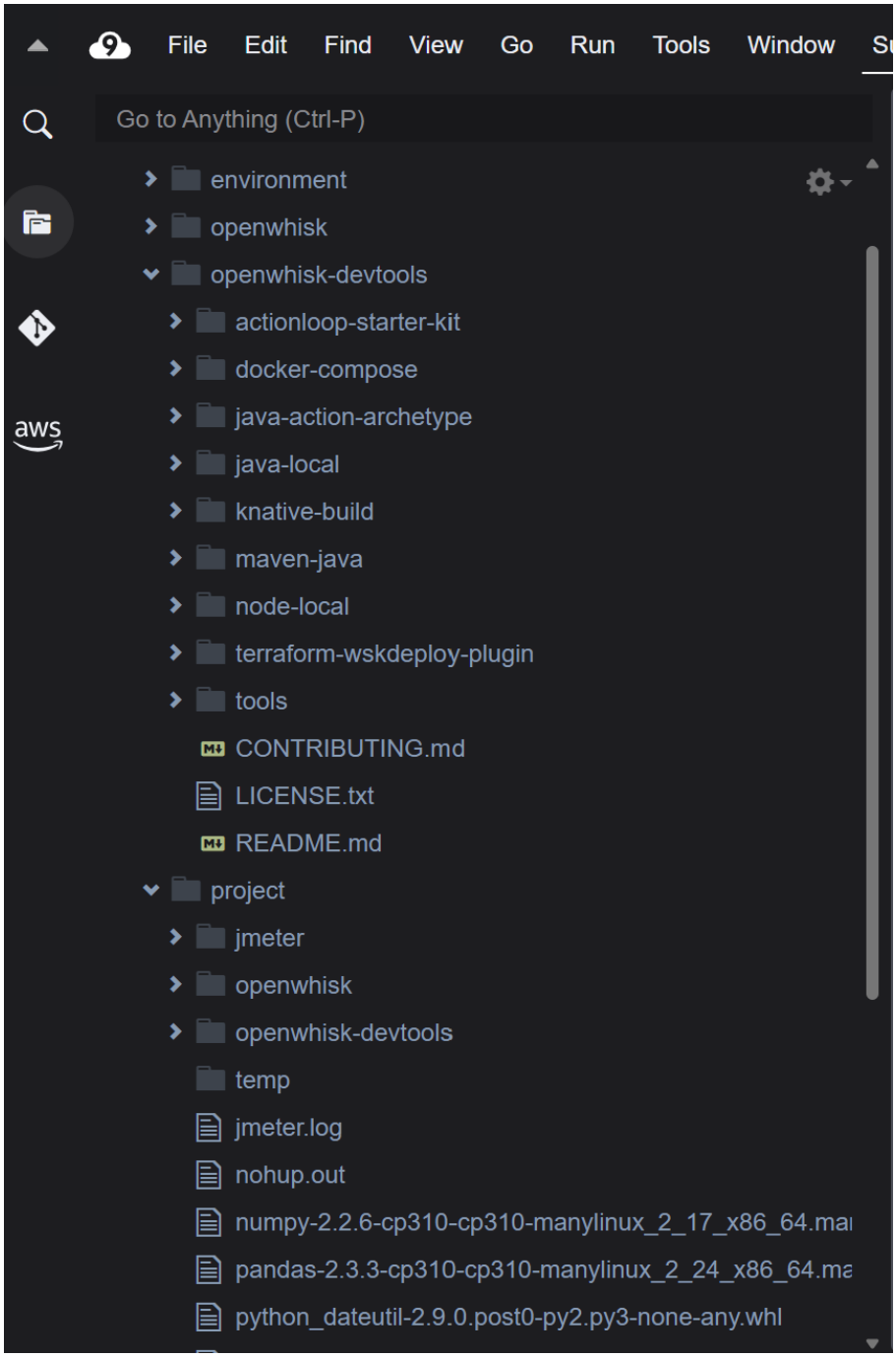


Figure 14: Jmeter Installation

3.4.4 Phase 4 : Running the Experiments

Make sure you're in the root directory cd /home/ubuntu/

Experiment 1:

Run the Jmeter test plan against the custom modules for “experiment_1_results.csv” generated by the LSTM model:

./run_experiment_1_with_controller.sh

Experiment 2:

Run the Jmeter test plan against the custom modules for “experiment_2_results.csv” generated by the BiLSTM model:

./run_experiment_2_with_controller.sh

Experiment 3:

Run the Jmeter test plan against the custom modules for “experiment_3_results.csv” generated by the Multiheaded BiLSTM model:

./run_experiment_3_with_controller.sh

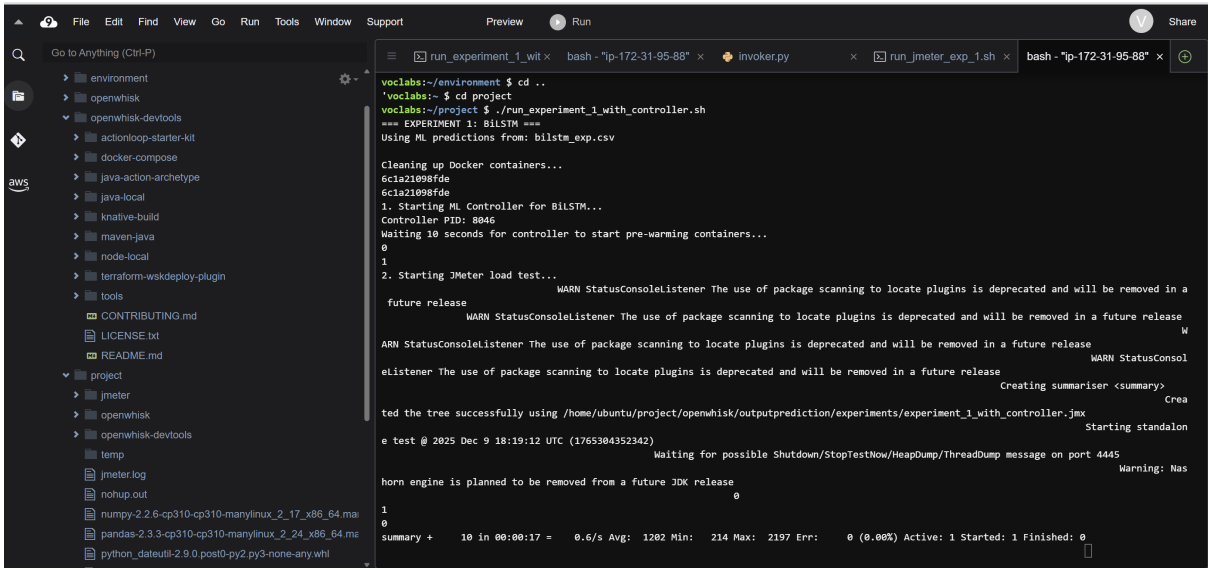


Figure 15: Experiment 1 Execution

3.4.5 Phase 5 : Evaluating the Results

Experiments will generate have an additional results CSV file that contains cold start information. These files were written directly to:

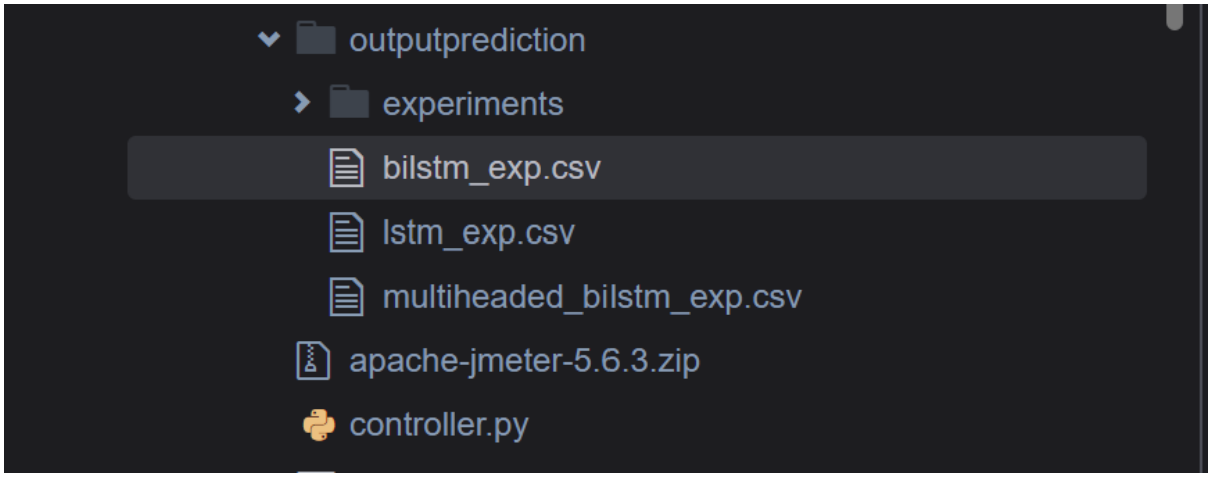


Figure 16: Experiments Results

These files will evaluated based on their cold / warm start values for the respective serverless custom function.

3.5 Data visualization

Using csv files generated by the jmeter I have visualized cold and warm starts percentage predicted by the deep learning models I have used google colab for data visualization.

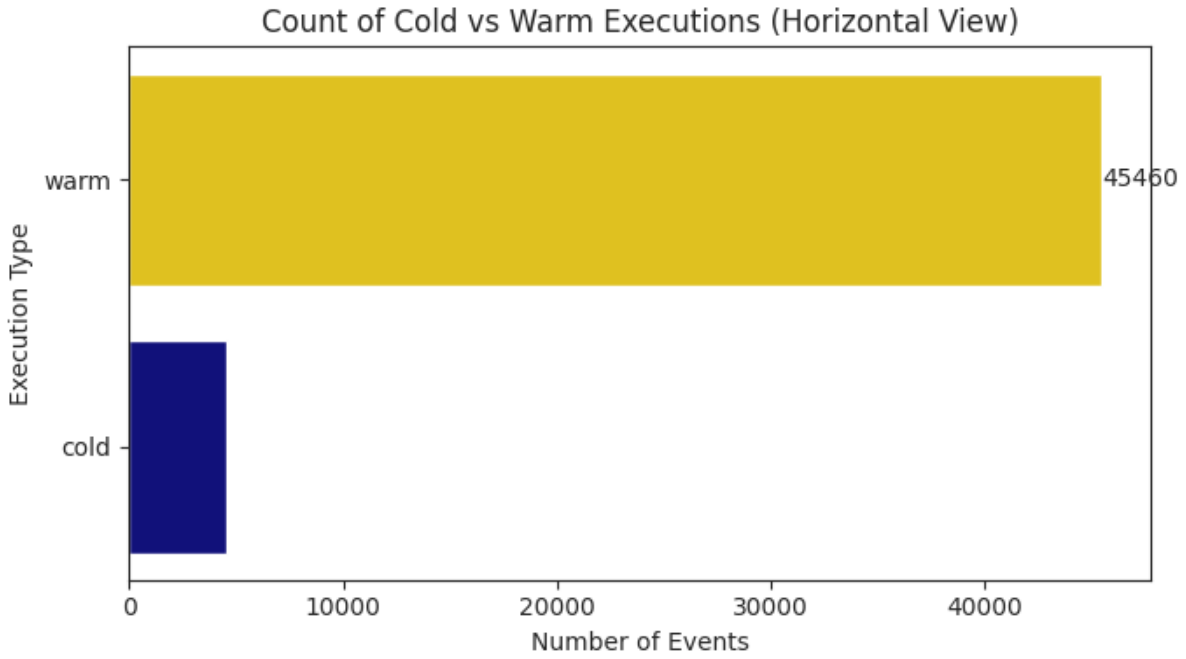


Figure 17: Multi-head BiLSTM Prediction

4 References

Amazon Web Services, Inc. (n.d.). AWS Lambda Documentation.
URL: https://docs.aws.amazon.com/lambda/

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. *Nature, 585*(7825), 357–362.
URL: https://numpy.org/

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering, 9*(3), 90–95.
URL: https://matplotlib.org/

Python, P. (n.d.). Installing packages using pip and virtual environments.
URL: https://packaging.python.org/guides/installing-using-pip-and-virtual-environmen