

# Configuration Manual

MSc Research Project  
MSc in Cloud Computing

Ruksar Shaikh  
Student ID: X22174711

School of Computing  
National College of Ireland

Supervisor: Shaguna Gupta

National College of Ireland  
Project Submission Sheet  
School of Computing



<b>Student Name:</b>	Ruksar Shaikh
<b>Student ID:</b>	X22174711
<b>Programme:</b>	MSc in Cloud Computing
<b>Year:</b>	2023-24
<b>Module:</b>	MSc Research Project
<b>Supervisor:</b>	Shaguna Gupta
<b>Submission Due Date:</b>	31/01/2024
<b>Project Title:</b>	Configuration Manual
<b>Word Count:</b>	XXX
<b>Page Count:</b>	11

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.

<b>Signature:</b>	
<b>Date:</b>	30th January 2024

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:**

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

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

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# Configuration Manual

Ruksar Shaikh  
X22174711

## 1 Introduction

By using machine learning and deep learning learning models, this research project aims to predict the resource utilisation in cloud computing system. This configuration manual will help to understand the implementation process of the project. It also includes the system setup used for installing the project's required tools.

## 2 System Configuration Setup

### 2.1 Software Requirement

- **Google Colab:** free, cloud-based platform by Google that provides a Jupyter Notebook environment allowing users to write, execute, and share Python code interactively. Python version is 3.10.12.
- **Email:** Gmail account is required to access the drive.
- **Browser:** Google Chrome, Firefox and Safari.

### 2.2 Hardware Requirement

The following hardware are used for the implementation:

- **Operating System:** Windows 10 64-bit operating system.
- **RAM/Processor:** 12gb 5th Gen Intel Core i5

## 3 Project Implementation

### 3.1 Environmental Setup

Gmail account is a required prerequisite to use Google Colaboratory. Once an account is created, follow the below steps:

- **Step 1:** Go to drive folder where ipynb python notebook files and dataset files are available.

- **Step 2:** Click on ipynb python notebook file to open refer figure 1.

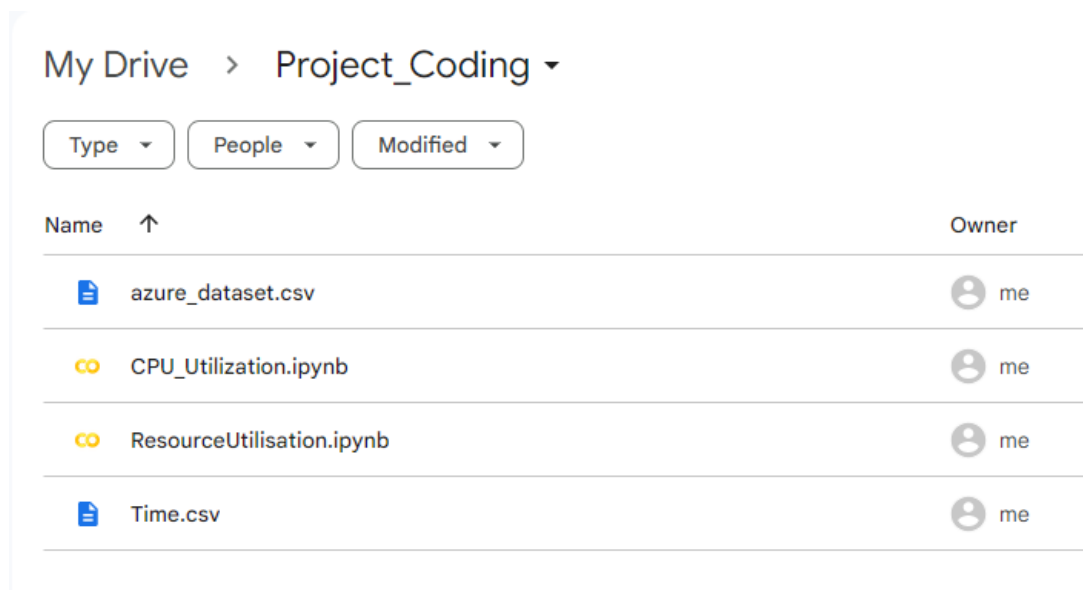


Figure 1: ipynb and dataset files in google drive

- **Step 3:** To connect the file to Google Colab, click on **Run all** in runtime. Mount the drive using the code mentioned in the given figure 2 and figure 3.

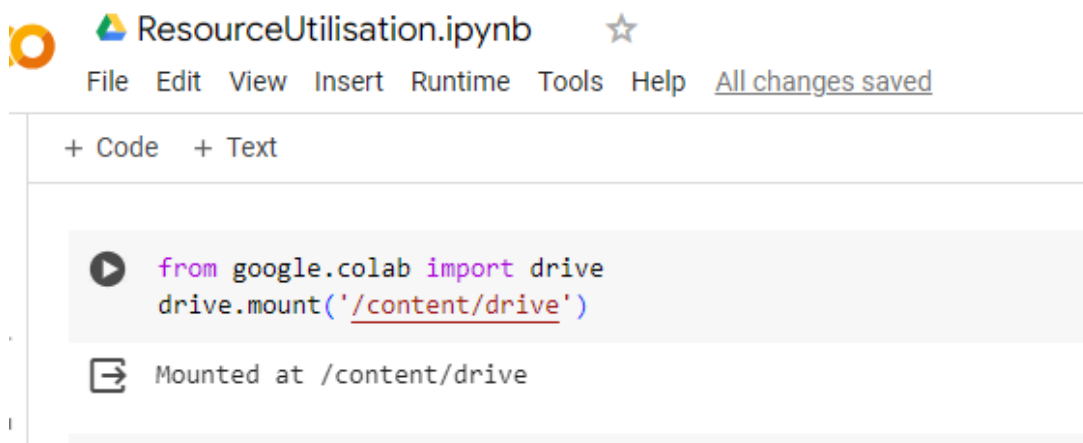


Figure 2: Google Drive Mounted in Google Colab for ResourceUtilization ipynb file

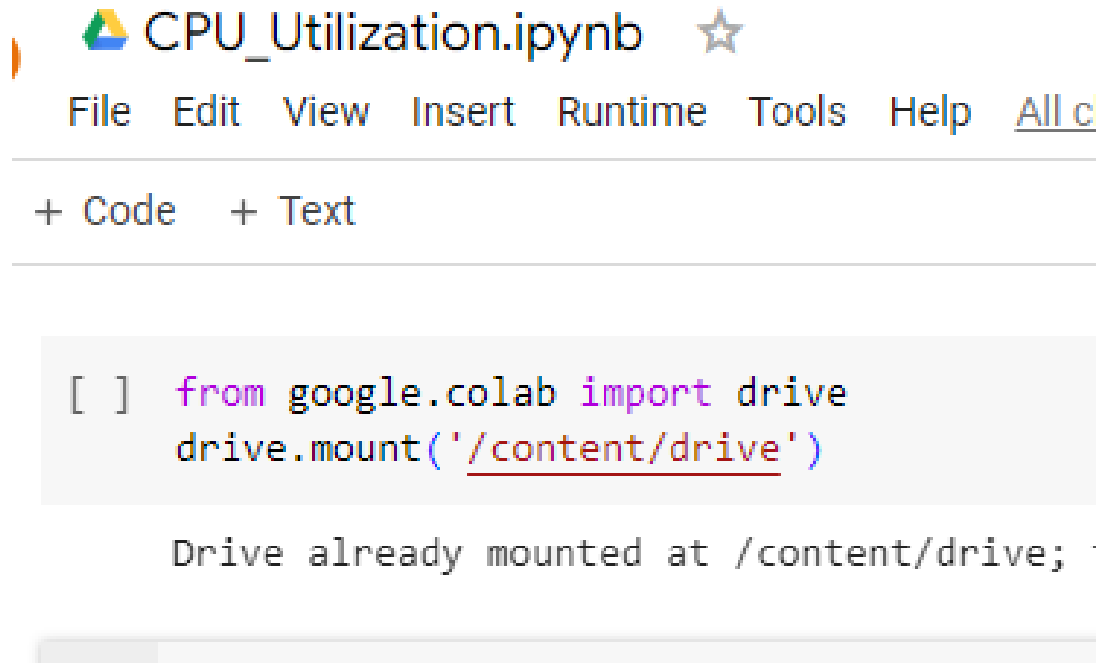


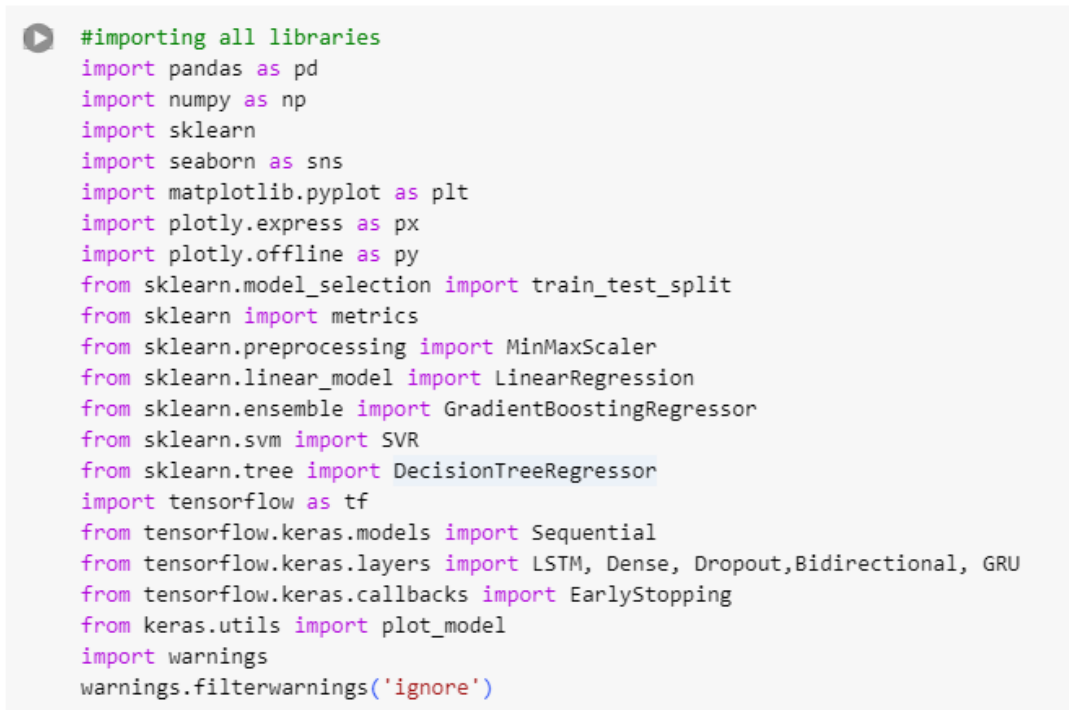
Figure 3: Google Drive Mounted in Google Colab for CPU\_Utilization ipynb file

### 3.2 Packages/Libraries Used

After successfully mounting the drive in Google Colab, all the libraries are imported before starting with the code implementation. The following are the libraries used for execution of models refer figure 4.

- **NumPy:** Fundamental package for numerical operations and array manipulation in Python.
- **Pandas:** Data manipulation and analysis tool for handling structured data with DataFrame structures.
- **Scikit-learn (Sklearn):** Machine learning library offering various tools for classical ML algorithms.
- **Seaborn:** Statistical data visualization library based on Matplotlib, simplifying complex data visualizations.
- **Plotly:** Interactive visualization library for creating web-based interactive plots.
- **Matplotlib:** Comprehensive plotting library for static, interactive, and animated visualizations.
- **TensorFlow:** Open-source deep learning framework for building and training neural network models.
- **Keras:** High-level neural networks API for quick neural network prototyping and development.

- **Adam:** Optimization algorithm for training deep learning models by adapting learning rates.
- **Sequential, Dense, Dropout, LSTM, Bi-directional:** Components and layers in Keras for building neural network architectures.
- **Min-max Scaler:** Sklearn tool for feature scaling to a specified range for machine learning models.
- **EarlyStopping:** Keras callback for stopping training when a monitored metric stops improving.
- **LinearRegression, GradientBoostingRegressor, SVR, DecisionTreeRegressor:** Sklearn regression algorithms for predicting continuous values based on input features.



```
#importing all libraries
import pandas as pd
import numpy as np
import sklearn
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.offline as py
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, Bidirectional, GRU
from tensorflow.keras.callbacks import EarlyStopping
from keras.utils import plot_model
import warnings
warnings.filterwarnings('ignore')
```

Figure 4: Required Libraries/Packages

## 4 Phases

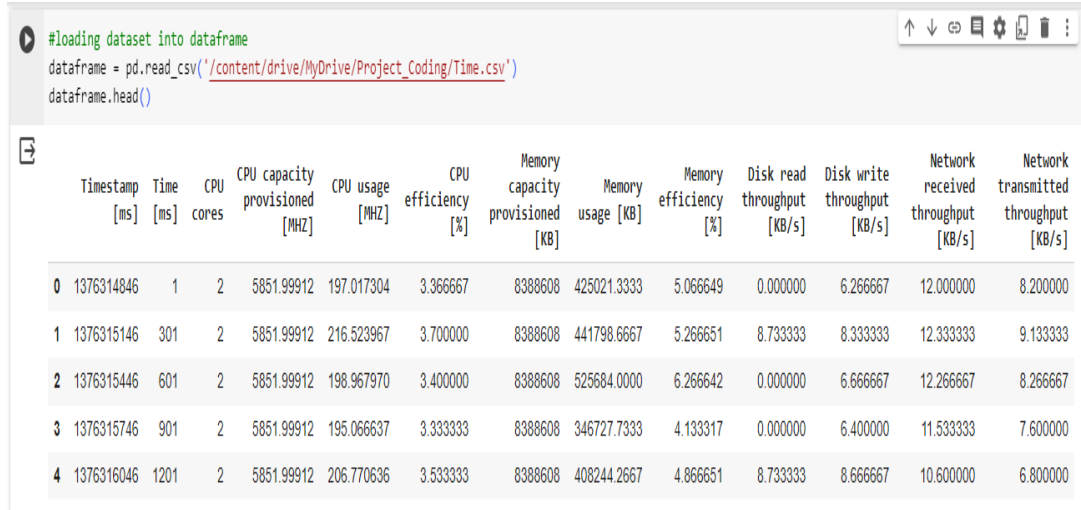
Implementation methodology for predicting resource utilization using ML and DL models is as follows:

### 4.1 Data Collection:

In this research project, there are two datasets utilised. Bitbrains Delft University of Technology (2015) dataset for CPU utilization and Network transmission throughput and Microsoft azure Azure (2017) dataset for CPU utilization.

## 4.2 Loading Dataset:

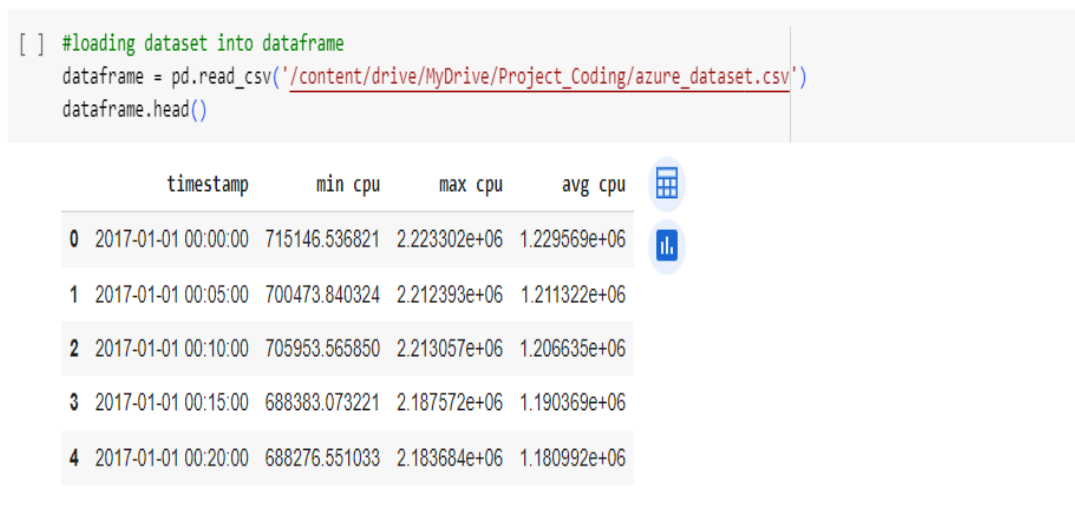
Figure 5 and figure 6, is presenting how Time.csv and azure\_dataset.csv dataset is loaded using pandas dataframe library named 'dataframe' in ResourceUtilization and CPU\_Utilization ipynb files respectively.



```
#loading dataset into dataframe
dataframe = pd.read_csv('/content/drive/MyDrive/Project_Coding/Time.csv')
dataframe.head()
```

	Timestamp [ms]	Time [ms]	CPU cores	CPU capacity provisioned [MHZ]	CPU usage [MHZ]	CPU efficiency [%]	Memory capacity provisioned [KB]	Memory usage [KB]	Memory efficiency [%]	Disk read throughput [KB/s]	Disk write throughput [KB/s]	Network received throughput [KB/s]	Network transmitted throughput [KB/s]
0	1376314846	1	2	5851.99912	197.017304	3.366667	8388608	425021.3333	5.066649	0.000000	6.266667	12.000000	8.200000
1	1376315146	301	2	5851.99912	216.523967	3.700000	8388608	441798.6667	5.266651	8.733333	8.333333	12.333333	9.133333
2	1376315446	601	2	5851.99912	198.967970	3.400000	8388608	525684.0000	6.266642	0.000000	6.666667	12.266667	8.266667
3	1376315746	901	2	5851.99912	195.066637	3.333333	8388608	346727.7333	4.133317	0.000000	6.400000	11.533333	7.600000
4	1376316046	1201	2	5851.99912	206.770636	3.533333	8388608	408244.2667	4.866651	8.733333	8.666667	10.600000	6.800000

Figure 5: Dataset loaded for ResourceUtilization file



```
[ ] #loading dataset into dataframe
dataframe = pd.read_csv('/content/drive/MyDrive/Project_Coding/azure_dataset.csv')
dataframe.head()
```

	timestamp	min cpu	max cpu	avg cpu
0	2017-01-01 00:00:00	715146.536821	2.223302e+06	1.229569e+06
1	2017-01-01 00:05:00	700473.840324	2.212393e+06	1.211322e+06
2	2017-01-01 00:10:00	705953.565850	2.213057e+06	1.206635e+06
3	2017-01-01 00:15:00	688383.073221	2.187572e+06	1.190369e+06
4	2017-01-01 00:20:00	688276.551033	2.183684e+06	1.180992e+06

Figure 6: Dataset loaded for CPU\_Utilization file

## 4.3 Dataset Info

Figure 7 and figure 8, is presenting the structure of both the dataset.

```
#again checking datatypes after converting to datetime
dataframe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8634 entries, 0 to 8633
Data columns (total 13 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   Timestamp [ms]                             8634 non-null   int64
1   Time [ms]                                   8634 non-null   int64
2   CPU cores                                   8634 non-null   int64
3   CPU capacity provisioned [MHZ]             8634 non-null   float64
4   CPU usage [MHZ]                             8634 non-null   float64
5   CPU efficiency [%]                         8634 non-null   float64
6   Memory capacity provisioned [KB]           8634 non-null   int64
7   Memory usage [KB]                         8634 non-null   float64
8   Memory efficiency [%]                     8634 non-null   float64
9   Disk read throughput [KB/s]                8634 non-null   float64
10  Disk write throughput [KB/s]               8634 non-null   float64
11  Network received throughput [KB/s]          8634 non-null   float64
12  Network transmitted throughput [KB/s]      8634 non-null   float64
dtypes: float64(9), int64(4)
memory usage: 877.0 KB
```

Figure 7: Defined Dataset Column for ResourceUtilization file

```
#again checking datatypes after converting to datetime
dataframe.info()
```

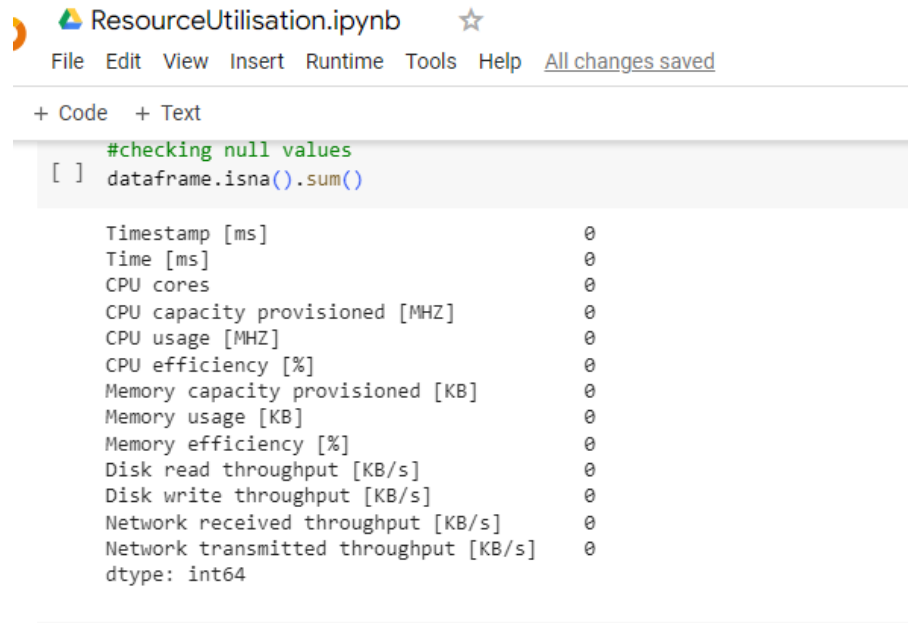
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8640 entries, 0 to 8639
Data columns (total 7 columns):
#   Column      Non-Null Count  Dtype
---  -
0   timestamp   8640 non-null   datetime64[ns]
1   min cpu     8640 non-null   float64
2   max cpu     8640 non-null   float64
3   avg cpu     8640 non-null   float64
4   Year        8640 non-null   int64
5   Month       8640 non-null   int64
6   Day         8640 non-null   int64
dtypes: datetime64[ns](1), float64(3), int64(3)
memory usage: 472.6 KB
```

Figure 8: Defined Dataset Column for CPU\_Utilization file

## 4.4 Dataset Cleaning

Figure 9 and figure 10, checking for if any missing values we have in the dataframe .



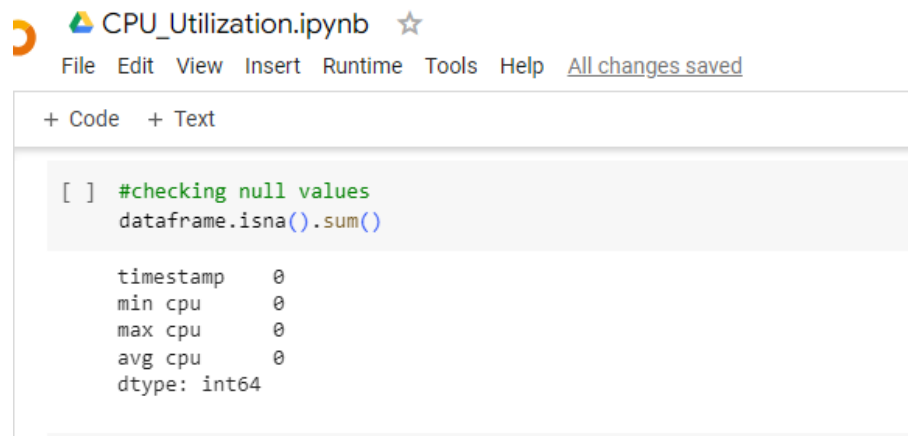


The image shows a Jupyter Notebook titled "ResourceUtilisation.ipynb". The interface includes a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", "Help", and a status bar indicating "All changes saved". Below the menu bar, there are tabs for "+ Code" and "+ Text". The code cell contains the following Python code:

```
#checking null values
[ ] dataframe.isna().sum()
```

The output of the code is a series of 14 rows, each representing a different resource utilization metric. Each row has a label on the left and a value of 0 on the right, indicating that there are no missing values for any of these metrics. The metrics are: Timestamp [ms], Time [ms], CPU cores, CPU capacity provisioned [MHZ], CPU usage [MHZ], CPU efficiency [%], Memory capacity provisioned [KB], Memory usage [KB], Memory efficiency [%], Disk read throughput [KB/s], Disk write throughput [KB/s], Network received throughput [KB/s], and Network transmitted throughput [KB/s]. The output ends with "dtype: int64".

Figure 9: Checking for missing values in ResourceUtilization file



The image shows a Jupyter Notebook titled "CPU\_Utilization.ipynb". The interface includes a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", "Help", and a status bar indicating "All changes saved". Below the menu bar, there are tabs for "+ Code" and "+ Text". The code cell contains the following Python code:

```
#checking null values
[ ] dataframe.isna().sum()
```

The output of the code is a series of 4 rows, each representing a different CPU utilization metric. Each row has a label on the left and a value of 0 on the right, indicating that there are no missing values for any of these metrics. The metrics are: timestamp, min cpu, max cpu, and avg cpu. The output ends with "dtype: int64".

Figure 10: Checking for missing values in CPU\_Utilization file

## 4.5 Dataset Visualization

Many ways in which data can be presented using matplotlib library. Figure 11 presents CPU utilization graph at x-axis and timestamp at y-axis , 12 illustrates the network-transmitted throughput at a certain period of time. 14 presents a comprehensive depiction of CPU Utilization trends over a specified duration.

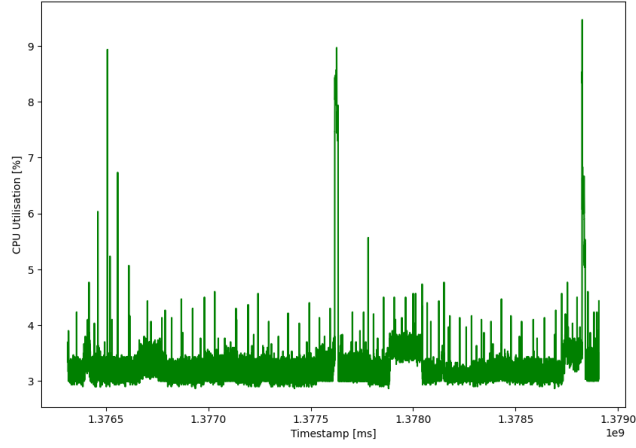


Figure 11: Visual representation of CPU utilization using Bitbrains Dataset

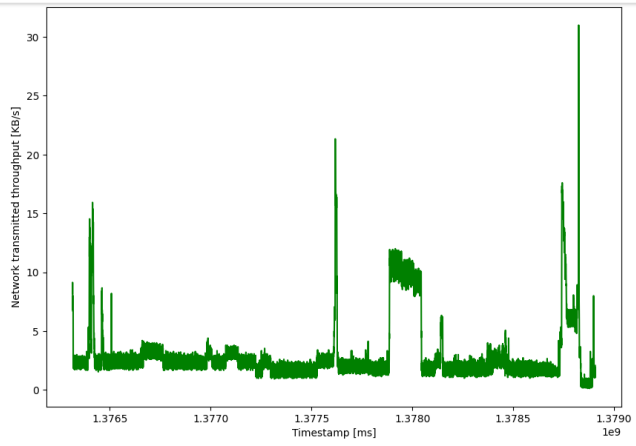


Figure 12: Visual representation of Network Transmission Throughput using Bitbrains Dataset

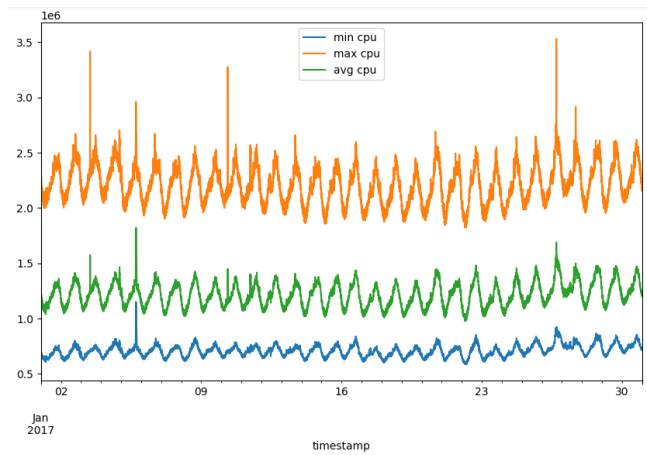


Figure 13: Visual representation of CPU utilization using Microsoft Azure Dataset

## 4.6 Feature Scaling

Feature scaling such as Min-Max scaling, is a technique used in data preprocessing to bring numerical features to a similar scale. Min-Max scaling specifically transforms each feature's values to a range between 0 and 1 in figure 14. It does this by subtracting the minimum value of the feature and dividing by the difference between the maximum and minimum values. This process ensures that all features have the same scale, preventing certain features from dominating due to their larger numerical ranges and helping algorithms converge faster during training.

```
[ ] scaler=MinMaxScaler(feature_range=(0,1))
    data3=scaler.fit_transform(np.array(data3).reshape(-1,1))
```

Figure 14: Feature scaling: Min-max scaler

## 4.7 Data Splitting

Dataset splitting into 90% for training and 10% for testing involves allocating 90% of the data to train a machine learning model to recognize patterns, while reserving the remaining 10% to assess its performance on unseen data, ensuring the model generalizes well to new instances and doesn't overfit to the training data refer figure 15.

```
] training_size=int(len(data3)*0.90)
   test_size=len(data3)-training_size
   train_data,test_data=data3[0:training_size,:],data3[training_size:len(data3),:1]

def create_dataset(dataset, time_step=1):
    dataX, dataY = [], []
    for i in range(len(dataset)-time_step-1):
        a = dataset[i:(i+time_step), 0]
        dataX.append(a)
        dataY.append(dataset[i + time_step, 0])
    return np.array(dataX), np.array(dataY)

] time_step = 3
   X_train, y_train = create_dataset(train_data, time_step)
   X_test, y_test = create_dataset(test_data, time_step)
```

Figure 15: Feature scaling: Min-max scaler

## 4.8 Training of Models

In this section, four machine learning models including Linear Regression, Decision Tree Regression, Gradient Boosting Regression, Support Vector Regression and two deep learning models including Long Short Term Memory and Bi-directional Long Short Term Memory models are trained to predict the resource utilisation in cloud computing environments.

## 4.9 Performance Evaluation

The MSE, MAE, RMSE and R2 metrics of the ML and DL algorithms compared in this paper as shown in Table 1, Table 2 and Table 3 .

Table 1: For BitBrains dataset - CPU Utilization prediction

Model	MSE	MAE	RMSE	R2 score
LiR	0.0027	0.0215	0.0521	0.7794
DTR	0.0099	0.0402	0.0995	0.1951
GBR	0.0055	0.0294	0.0747	0.5465
SVR	0.0030	0.0389	0.0556	0.7488
LSTM	0.0026	0.0233	0.0515	0.7843
BiLSTM	0.0024	0.0224	0.0490	0.8042

Table 2: For Microsoft Azure dataset - CPU Utilization prediction

Model	MSE	MAE	RMSE	R2 score
LiR	0.0002	0.0131	0.0169	0.9833
DTR	0.0023	0.0390	0.0480	0.8661
GBR	0.0010	0.0255	0.0321	0.9399
SVR	0.0015	0.0337	0.0388	0.9127
LSTM	0.0009	0.0239	0.0304	0.9462
BiLSTM	0.0004	0.0169	0.0214	0.9732

Table 3: For BitBrains dataset - Network Transmission Throughput prediction

Model	MSE	MAE	RMSE	R2 score
LiR	0.0012	0.0114	0.0359	0.9256
DTR	0.0043	0.0473	0.0656	0.752
GBR	0.00183	0.0259	0.0428	0.894
SVR	0.0037	0.0495	0.0610	0.786
LSTM	0.0026	0.0232	0.0513	0.848
BiLSTM	0.00172	0.0203	0.0415	0.901

## 4.10 Conclusion

This section concludes that BiLSTM model gives lower error RMSE and MAE values hence, gives better prediction results as compared to other predictive models followed by Linear Regression and LSTM.

## References

Azure (2017). Azurepublicdataset repository on github, <https://github.com/Azure/AzurePublicDataset>. Accessed 15th Nov 2023.

Delft University of Technology (2015). Gwa-t-12: Bitbrains dataset. Dataset retrieved from Delft University of Technology. Accessed 20th Nov 2023.

**URL:** *<http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains>*