National College of Ireland

# Configuration Manual

MSc Research Project
Cybersecurity

Idehen Jimmy Irivbogbe
19144814

School of Computing
National College of Ireland

Supervisor; Michael Pantridge

**National College of Ireland**

**MSc Project Submission Sheet**

**School of Computing**

**Student Name:** …….Irivbogbe Idehen Jimmy ……………………………………………………………………………………………

**Student ID:** ………19144814………………………………………………………………………………………

**Programme:**………Cybersecurity…………………………………………**Year:** ……2021…………………..

**Module:** …Msc Internship……………………………………………………………………………….

**Lecturer:** ………Michael…Pantridge…………………………………………………………………………

**Submission Due Date:** ………………16/082021……………………………………………………………………………

**Project Title:** Securing Internet of things (IoT) using SDN- enabled Deep learning Architecture…………………………………………………………………………………….…

**Word Count:** ………631……………………… **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:** ……………………………………………………………………………………………………

**Date:** ……………………16/08/2021………………………………………………………………………

**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

Idehen Iirivbogbe
19144814

## 1 Introduction

The configuration manual plays an important role in the reproduction of the proposed work using the algorithms of deep learning for intrusion detection in internet of things. This manual comprises the overall setup and related tools for the implementation of this work.

## 2 Specifications of the System

The system specifications are as follow:

- Processor: Intel Core i7, $7^{th}$ generation @2.24 GHz
- GPU: Nvidia Geforce 6GB
- RAM: 16 GB
- SSD: 1 TB
- Operating System : Windows 10 professional

## 3 Software's/Tools

The software and tools that are used in the implementation of this work are as follow:

- Anaconda Navigator
- Python
- Spyder
- Origin

### 3.1 Software Installation

The step by step installation of the software are given below:
Anaconda navigator can be downloaded from the given link
(https://www.anaconda.com/products/individual)
As this work has used the windows version of the anaconda. So Anaconda for windows has been downloaded.

Figure 1: Downloading Anaconda

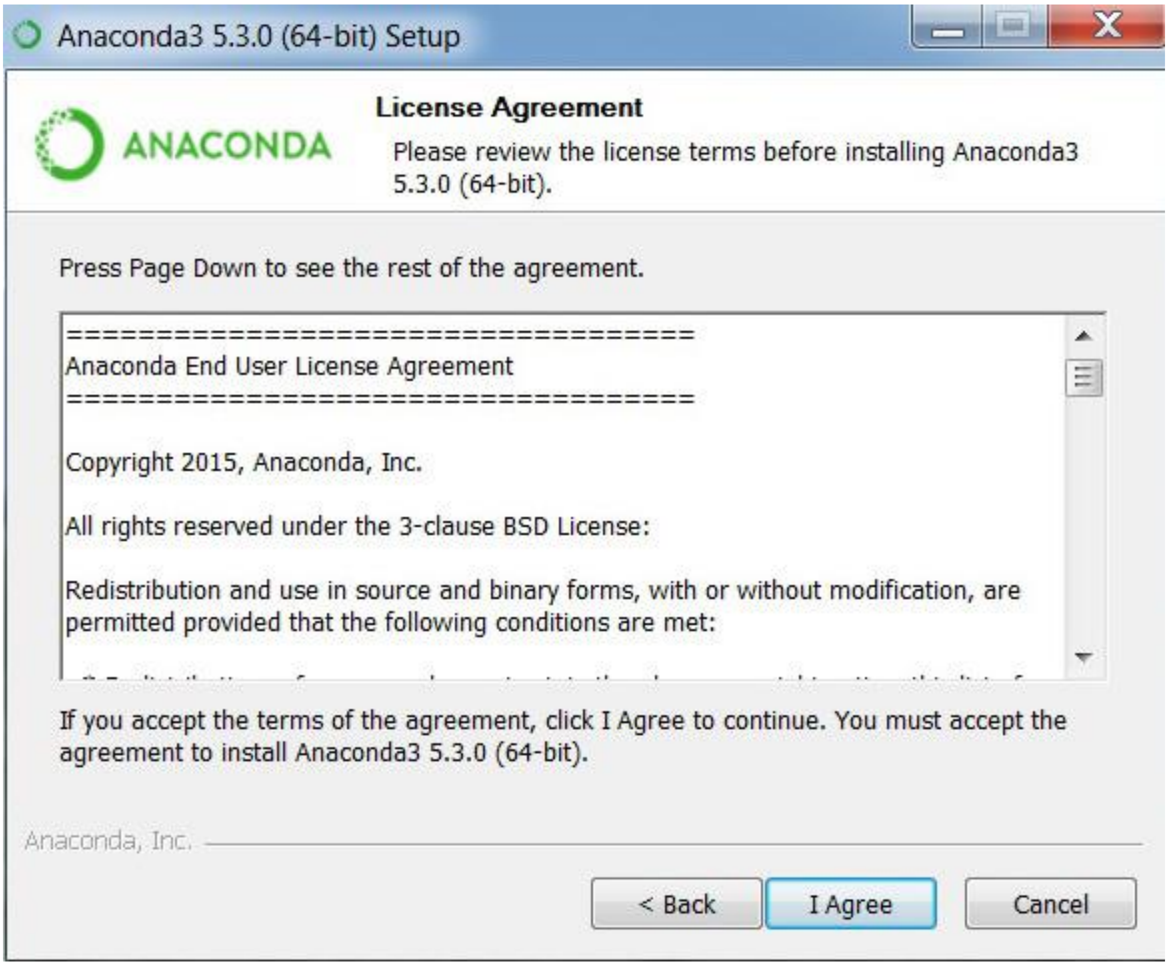

Figure 2: Anaconda Installation

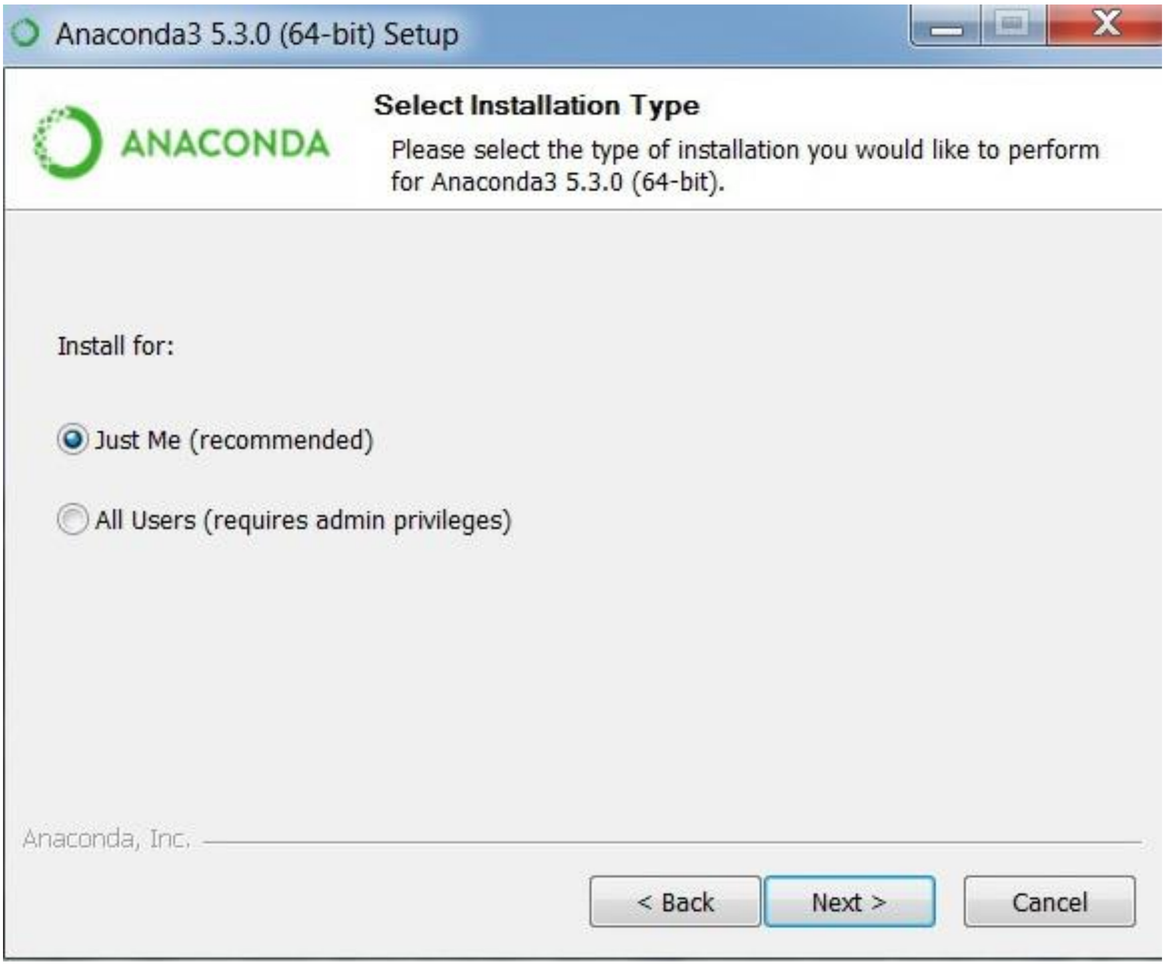


Figure 3: Anaconda Installation

After selection of the installation type(Just Me) click on the next button.

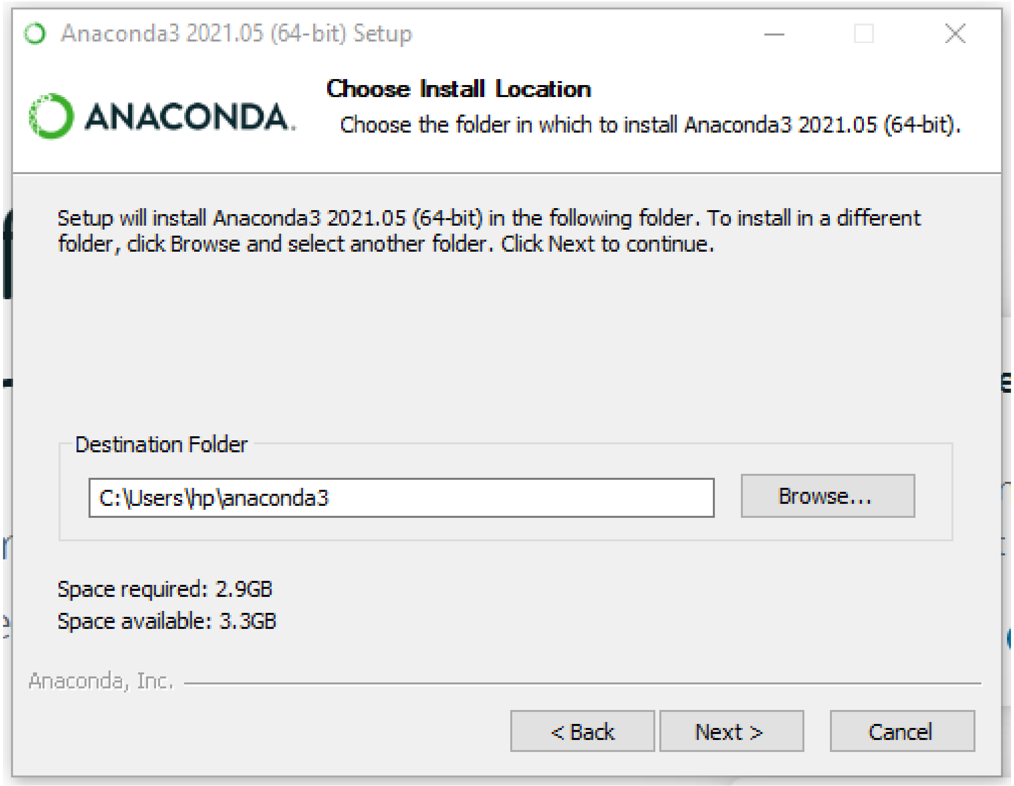


Figure 4; Path Selection

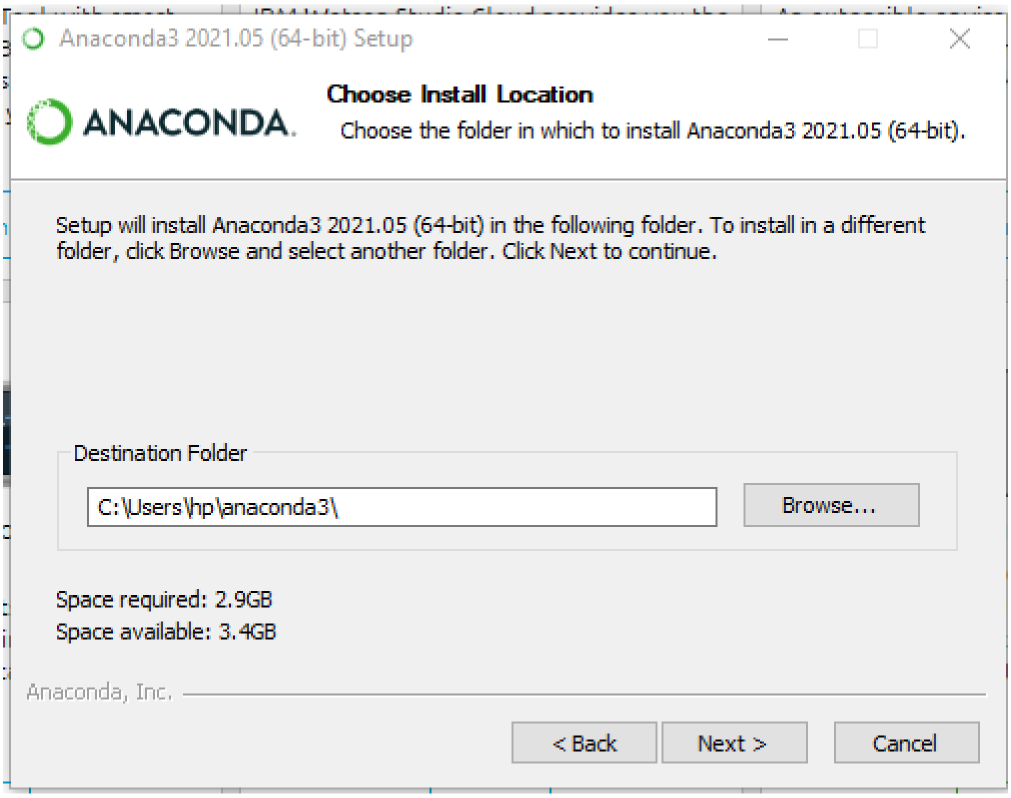


Figure 5: Setting default python

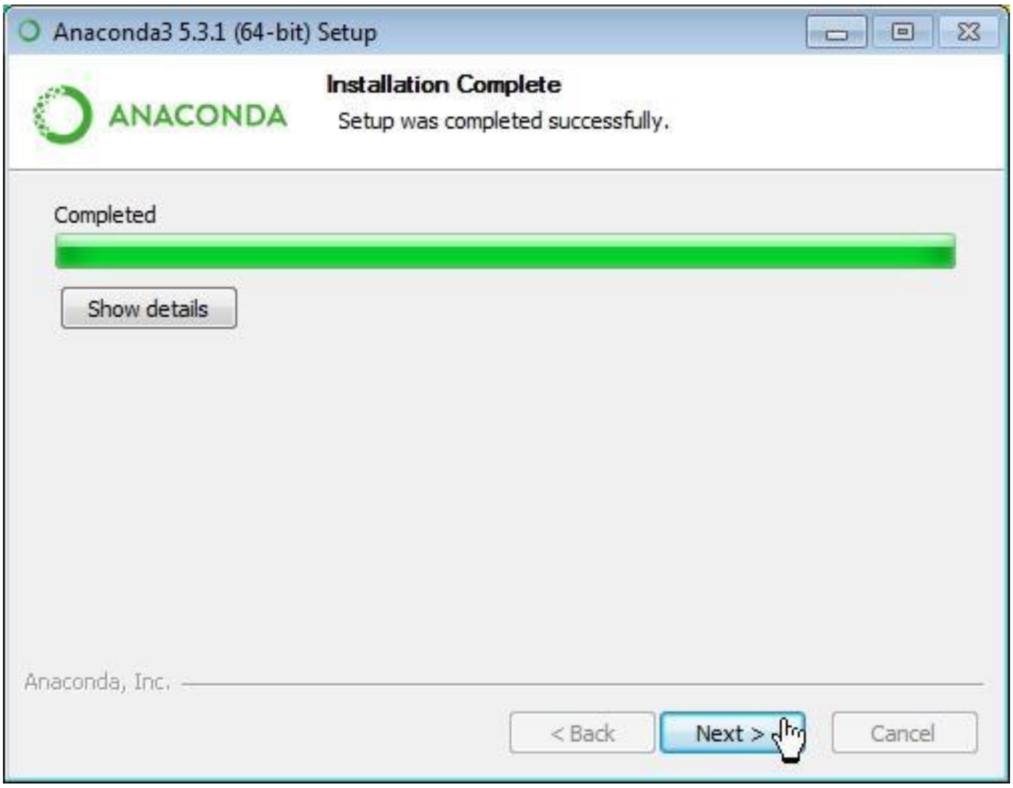


Figure 6: Setup completed

Further, you can download the python 3.9.0 from the below link (https://www.python.org/downloads/)

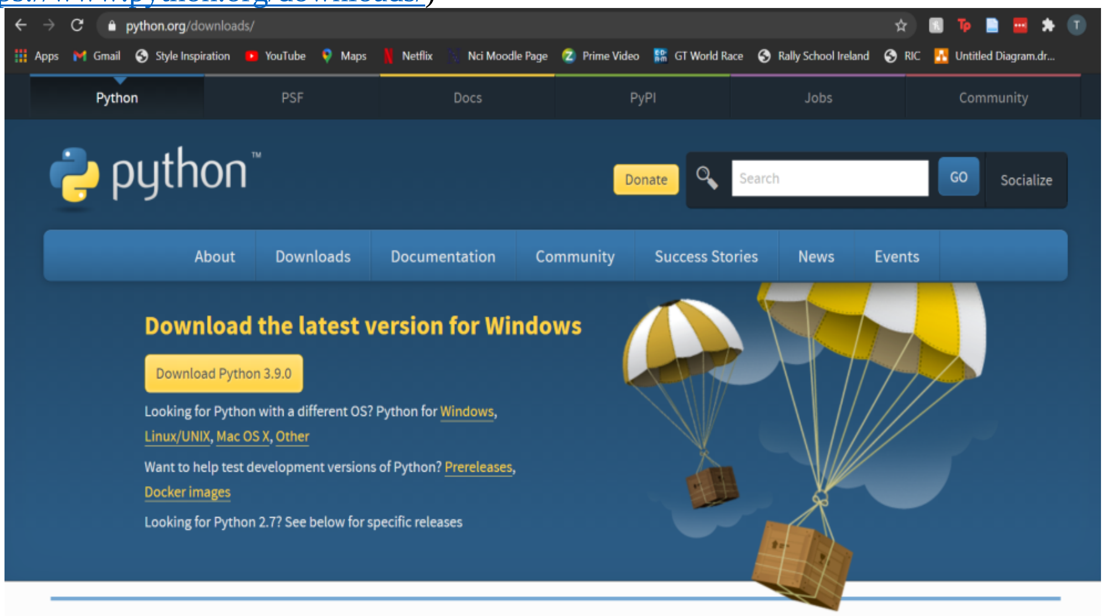


Figure 7: Downloading Python 3.9.0


Figure 8: Python 3.9.0 Installation

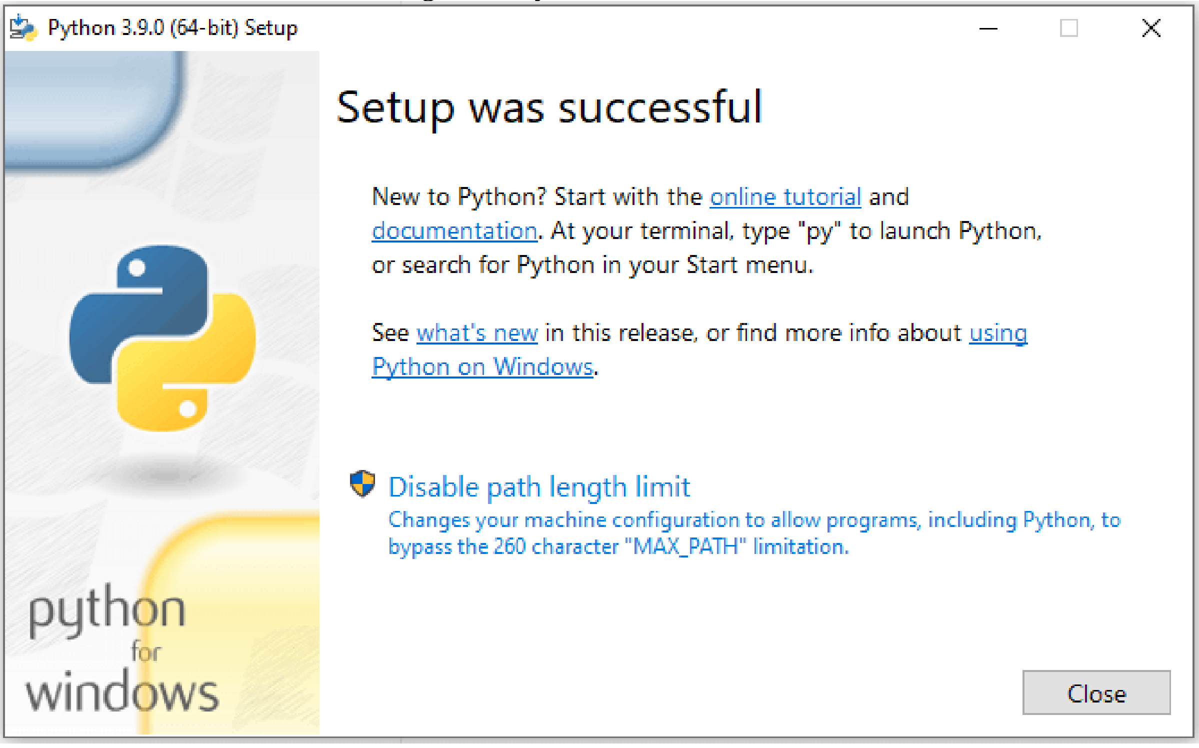


Figure 9: Python 3.9.0 Installation Competed

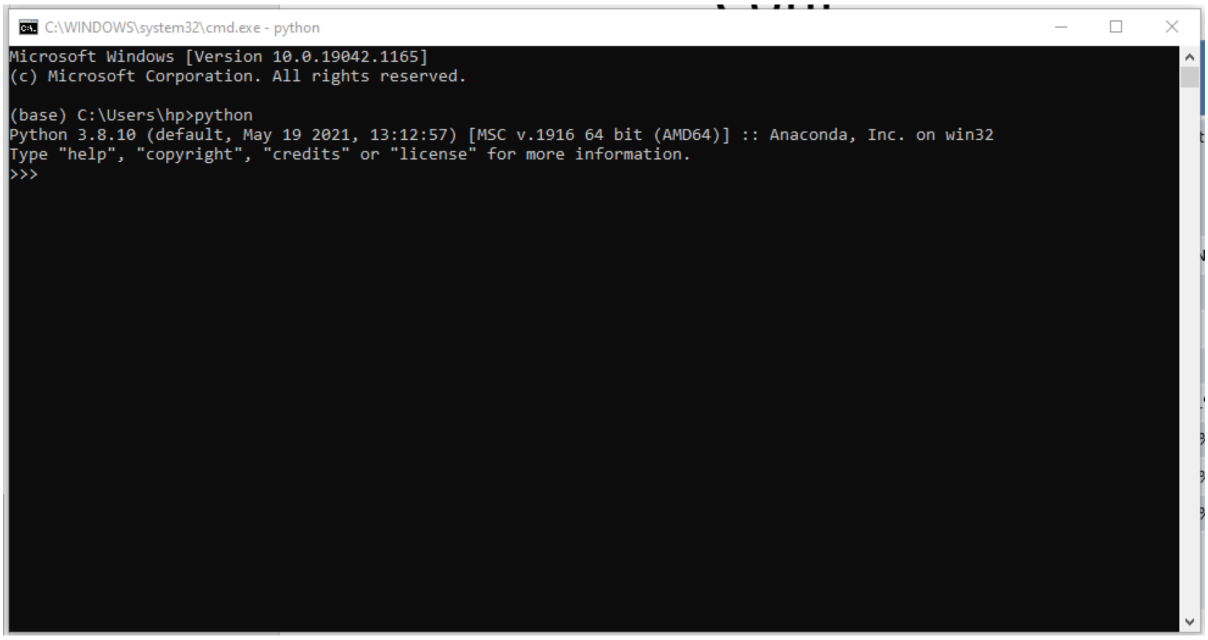

Figure 10: Python 3.9.0 Installation Confirmation

# 4 Implementation

The libraries used for the purpose of implementation are as follow:

- Pandas
- Numpy
- Sklearn
- Keras

For a proper experimentation. All of the libraries has been imported using spyder IDE.

```
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical

from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense , Dropout
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from keras.layers import  CuDNNLSTM, Activation
```

Figure 11: Importing Libraries and packages

## 1. Dataset Loading

```
from keras.layers import  CuDNNLSTM, Activation

#importing dataset
mydata=pd.read_csv("dataset.csv")

mydata = mydata.sample(frac = 1).reset_index(drop = True)

#mydata.fillna(0, inplace=True)
#feature extraction
```

Figure 12: Importing Dataset

## 2. Preprocessing

```
#mydata.fillna(0, inplace=True)
#feature extraction
X=mydata.iloc[:, 0:79]
Y=mydata.iloc[:,79:80]
#
sc=StandardScaler()
X = sc.fit_transform(X)
#
#min_max_scaler = preprocessing.MinMaxScaler()
#X = min_max_scaler.fit_transform(X)

x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.20)

#min_max_scaler = preprocessing.MinMaxScaler()
#X = min_max_scaler.fit_transform(X)

#min_max_scaler = preprocessing.MinMaxScaler()
#x_train = min_max_scaler.fit_transform(x_train)

x_train = np.array(x_train)
x_train = np.reshape (x_train,(x_train.shape[0], 1, x_train.shape[1]))
```

Figure 13: Dataset Preprocessing

## 3. One hot Encoding

```
temp.py*


#ONE HOT ENCODING
Num_classes = len(np.unique(Y))
y_train_ohe= to_categorical(y_train,Num_classes)
y_train_ohe = pd.DataFrame(y_train_ohe)
```

Figure 14: One hot Encoding

## 4. Building Model

```python
model = Sequential()

model.add(CuDNNLSTM(300,   input_shape= (x_train.shape[1],x_train.shape[2]), return_sequences=True))
#model.add(CuDNNLSTM(650,  return_sequences=True))
#model.add(CuDNNLSTM(600,  return_sequences=True))
model.add(CuDNNLSTM(200,  return_sequences=False))
model.add(Dropout(0.4))

#model.add(CuDNNLSTM(300,  return_sequences=False))

model.add(Activation('relu'))
#model.add(LSTM(200, activation= 'relu',  return_sequences=True))
#model.add(LSTM(150, activation= 'relu',  return_sequences=True))
#model.add(LSTM(100, activation= 'relu',  return_sequences=False))


model.add(Dense(200, activation='relu'))
#model.add(Dense(150, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(100, activation='relu'))
model.add(Dense(50, activation='relu'))

model.add(Dense(Num_classes, activation='softmax'))
```

Figure 15: Model Building

## 5. Optimizer Adding

```python
#from keras import optimizers
#import keras
#import keras.utils
#from keras import utils as np_utils
#keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)


model.compile(optimizer='adamax', loss='binary_crossentropy', metrics=['accuracy'])
#model.summary()
```

Figure 16: Adding optimizer

## 6. Model Training

```python
#model.summary()

import time

start = time.clock()

#train the model
model.fit(x_train, y_train_ohe, epochs=5, batch_size=64)

end = time.clock()
print ("%.2gs" % (end-start))
#predections
```

Figure 17: Training the model

## 7. Accuracy and Confusion Matrix

```
print("Accuracy:" + str(accuracy_score(y_test, prediction)* 100))
print(classification_report(y_test, prediction))
cm = confusion_matrix(y_test, prediction)
print (cm)
```

Figure 18: printing accuracy and confusion matrix

## 8. Calculating TP,FP,FN,TN

```
#Pred = np.round(Pred)
FP = cm.sum(axis=0) - np.diag(cm)
FN = cm.sum(axis=1) - np.diag(cm)
TP = np.diag(cm)
TN = cm.sum() - (FP + FN + TP)


FP = FP.astype(float)
FN = FN.astype(float)
TP = TP.astype(float)
TN = TN.astype(float)


# Sensitivity, hit rate, recall, or true positive rate
TPR = TP/(TP+FN)
# Specificity or true negative rate
TNR = TN/(TN+FP)
```

Figure 19: Calculating FP,FN,TP,TN

**9. Calculating overall accuracy, Precision**

```
# Sensitivity, hit rate, recall, or true positive rate
TPR = TP/(TP+FN)
# Specificity or true negative rate
TNR = TN/(TN+FP)
# Precision or positive predictive value
PPV = TP/(TP+FP)
# Negative predictive value
NPV = TN/(TN+FN)
# Fall out or false positive rate
FPR = FP/(FP+TN)
# False negative rate
FNR = FN/(TP+FN)
# False discovery rate
FDR = FP/(TP+FP)

# Overall accuracy
ACC = (TP+TN)/(TP+FP+FN+TN)
# Precision
Precision = TP / TP + FP
```

Figure 20: Calculating Accuracy, Precision

# 5 Results

Table 1. Results and Evaluation

| **Algorithm** | **Accuracy** | **Precision** | **Tpr** | **Fpr** | **F1-score** | **Recall** |
|---|---|---|---|---|---|---|
| DNNLSTM | 99.92% | 99.30% | 99.87% | 0.0070% | 99.50% | 99.50% |
| **GRU** | 99.79% | 99.33% | 99.77% | 0.0090% | 99.50% | 99.83% |
| **LSTMGRU** | 99.11% | 99.16% | 99.77% | 0.012% | 90.10% | 88.19% |