Configuration Manual

MSc Research Project

Hemanth Murukutla
Student ID: x21219214

School of Computing
National College of Ireland

Supervisor: Michael Prior

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name	Hemanth Murukutla
Student ID	X21219214
Programme	(MSCCYB1)
Year:	2022-2023
Module:	Thesis
Supervisor:	Michael Prior
Submission Due Date:	AUG 14th 2:00PM
Project Title:	Detecting IOT Attacks Using AI
Word Count:	
Page Count:	

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	Hemanth
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 Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

1 Introduction

The present Configuration Manual compiles all the necessary prerequisites required for the replication of the studies and their impacts within a particular context. The provided information includes an overview of the data importing process and exploratory data analysis (EDA), followed by data pre-processing techniques such as class balancing, label encoding, and feature selection. Additionally, the document includes details about the developed algorithms and their evaluations. Furthermore, the necessary hardware components and software applications are also mentioned. The report is structured in the following manner, wherein Section 2 presents comprehensive information pertaining to the configuration of the environment.

A comprehensive explanation regarding the process of data collection can be found in Section 3. Section 4 encompasses the process of data pre-processing, which involves various techniques such as exploratory data analysis (EDA) for information extraction. Section 5 of the document provides a description of the process known as class balancing. Section 6 provides an overview of two important concepts in machine learning, namely Label encoding and Feature Selection. Section 7 of the document offers comprehensive information regarding the models that were developed and subsequently subjected to rigorous testing. The methodology for calculating and presenting the results is outlined in Section 8.

2 System Requirements

This section provides a comprehensive delineation of the specific hardware and software requirements necessary for the practical implementation of the research findings.

2.1 Hardware Requirements

The essential hardware specifications are depicted in Figure 1 as presented below. The system configuration comprises a MacOs M1 Chip, running on the macOS 10.15.x (Catalina) operating system. It is equipped with 8GB of RAM and a storage capacity of 256GB. The display size measures 24 inches.

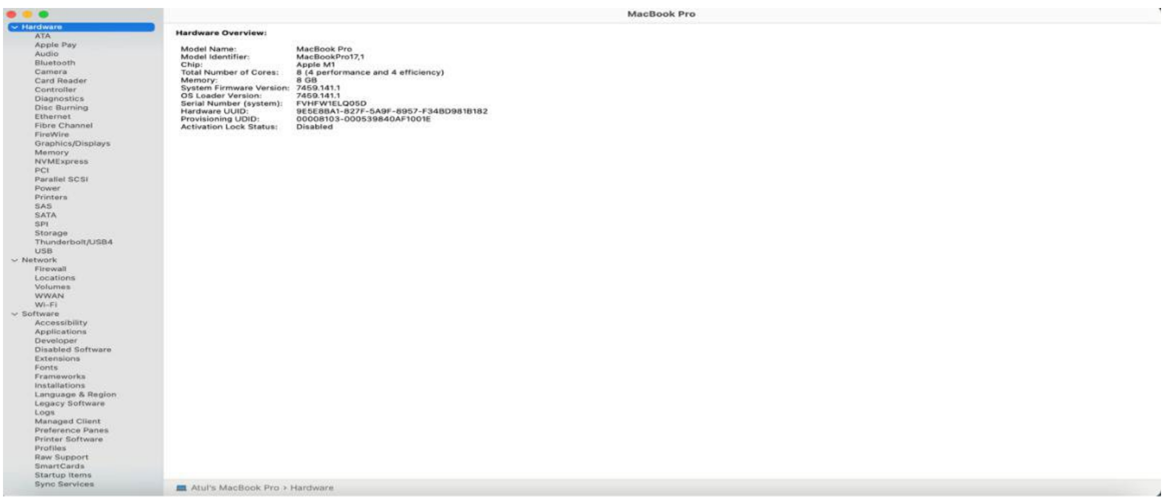


Figure 1: Hardware Requirements

2.2 Software Requirements

- Anaconda 3 (Version 4.8.0)
- Jupyter Notebook (Version 6.0.3)
- Python (Version 3.7.6)

2.3 Code Execution

The code has the capability to be executed within the Jupyter Notebook environment. The Jupyter Notebook is included in the Anaconda 3 distribution and can be launched upon system startup. Executing this command will launch Jupyter Notebook within a web browser. The web browser interface will display the hierarchical arrangement of directories within the operating system. Users are advised to navigate to the specific directory containing the desired code file. To access the code file, navigate to the designated folder and proceed to open it. To execute the code, locate the Kernel menu and follow the necessary steps. Execute all code cells.

3 Data Collection

The dataset utilized in this study was sourced from the public repository on Kaggle, accessible via the following link: https://www.kaggle.com/datasets/mohamedamineferrag/edgeiiotset-cyber-security-dataset-of-iot-iiot. The dataset contains a comprehensive collection of authentic cyber security data pertaining to Internet of Things (IoT) and Industrial Internet of Things (IIoT) applications.

4 Data Exploration

Figure 2 includes a list of every Python library necessary to complete the project.

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

from sklearn.model_selection import train_test_split

import warnings
from imblearn.under_sampling import NearMiss
from collections import Counter
warnings.filterwarnings("ignore")
import matplotlib.pyplot as plt
import seaborn as sns


import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

from sklearn.metrics import accuracy_score, f1_score, recall_score, precision_score, confusion_matrix, classification_report
```

Figure 2: Necessary Python libraries

The Figure 3 represents the block of code to check data information.

```
data1 = pd.read_csv("/content/drive/MyDrive/BI-Network Security/Edge-IIoTset/Selected_dataset_for_ML_and_DL/DNN-EdgeIIoT-datase
t.csv")
data2 = pd.read_csv("/content/drive/MyDrive/BI-Network Security/Edge-IIoTset/Selected_dataset_for_ML_and_DL/ML-EdgeIIoT-dataset.
csv")

data = [data1,data2]
data = pd.concat (data, axis=0, sort=False, ignore_index=True)
data
```

	frame.time	ip.src_host	ip.dst_host	arp.dst.proto_ipv4	arp.opcode	arp.hw.size	arp.src.proto_ipv4	icmp.checksum	icmp.seq_le	icmp.tra
0	2021 11:44:10.081753000	192.168.0.128	192.168.0.101	0	0.0	0.0	0	0.0	0.0	
1	2021 11:44:10.162218000	192.168.0.101	192.168.0.128	0	0.0	0.0	0	0.0	0.0	
2	2021 11:44:10.162271000	192.168.0.128	192.168.0.101	0	0.0	0.0	0	0.0	0.0	
3	2021 11:44:10.162641000	192.168.0.128	192.168.0.101	0	0.0	0.0	0	0.0	0.0	
4	2021 11:44:10.166132000	192.168.0.101	192.168.0.128	0	0.0	0.0	0	0.0	0.0	

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2377001 entries, 0 to 2377
Data columns (total 63 columns):
 #   Column                      Dtype
---  ------                      -----
 0   frame.time                  object
 1   ip.src_host                 object
 2   ip.dst_host                 object
 3   arp.dst.proto_ipv4          object
 4   arp.opcode                  float6
 5   arp.hw.size                 float6
 6   arp.src.proto_ipv4          object
 7   icmp.checksum               float6
 8   icmp.seq_le                 float6
 9   icmp.transmit_timestamp     float6
 10  icmp.unused                 float6
 11  http.file_data              object
 12  http.content_length         float6
 13  http.request.uri.query      object
 14  http.request.method         object
 15  http.referer                object
 16  http.request.full_uri       object
 17  http.request.version        object
 18  http.response               float6
 19  http.tls_port               float6
 20  tcp.ack                     float6
 21  tcp.ack_raw                 float6
 22  tcp.checksum                float6
 23  tcp.connection.fin          float6
```

Figure 3: EDA for Checking Data Information

In figure 4, the code to check for missing data.

```
: data.isnull().sum()

: frame.time              0
  ip.src_host             0
  ip.dst_host             0
  arp.dst.proto_ipv4      0
  arp.opcode              0
                         ..
  mbtcp.len               0
  mbtcp.trans_id          0
  mbtcp.unit_id           0
  Attack_label            0
  Attack_type             0
  Length: 63, dtype: int64

: data= data.dropna()
```

Figure 4: Missing data information

5 Class Balancing

The Figure 5, illustrate the code to check to value counts for each class and removing unknown class data.

```
data['Attack_type'].value_counts()

Normal                   1639944
DDoS_UDP                  136066
DDoS_ICMP                 130526
SQL_injection              61514
DDoS_HTTP                  60472
DDoS_TCP                   60309
Vulnerability_scanner      60186
Password                   60142
Uploading                  47903
Backdoor                   35057
Port_Scanning              32635
XSS                        25967
Ransomware                 21850
MITM                        2428
Fingerprinting              2002
Name: Attack_type, dtype: int64
```

Figure 5: Class counts

Figures 6 show the code used to merge similar category of attacks into one.

```
data['Attack_type'] = data['Attack_type'].replace('DDoS_UDP', 'DDoS')
data['Attack_type'] = data['Attack_type'].replace('DDoS_ICMP', 'DDoS')
data['Attack_type'] = data['Attack_type'].replace('DoS slowloris', 'DDoS')
data['Attack_type'] = data['Attack_type'].replace('SQL_injection', 'DDoS')
data['Attack_type'] = data['Attack_type'].replace('DDoS_HTTP', 'DDoS')
data['Attack_type'] = data['Attack_type'].replace('DDoS_TCP', 'DDoS')
data['Attack_type'] = data['Attack_type'].replace('Vulnerability_scanner', 'WebAttack'
data['Attack_type'] = data['Attack_type'].replace('Password', 'WebAttack')
data['Attack_type'] = data['Attack_type'].replace('Uploading', 'WebAttack')
data['Attack_type'] = data['Attack_type'].replace('Port_Scanning', 'WebAttack')
data['Attack_type'] = data['Attack_type'].replace('XSS', 'WebAttack')
data['Attack_type'] = data['Attack_type'].replace('Fingerprinting', 'WebAttack')
data['Attack_type'] = data['Attack_type'].replace('MITM', 'WebAttack')
data['Attack_type'] = data['Attack_type'].replace('Backdoor', 'WebAttack')
data['Attack_type'] = data['Attack_type'].replace('Ransomware', 'DDoS')
data['Attack_type'].value_counts()
```

```
Normal       1639944
DDoS          470737
WebAttack     266320
Name: Attack_type, dtype: int64
```

```
data['Attack_type'].value_counts().plot(kind='bar')
```

```
<Axes: >
```

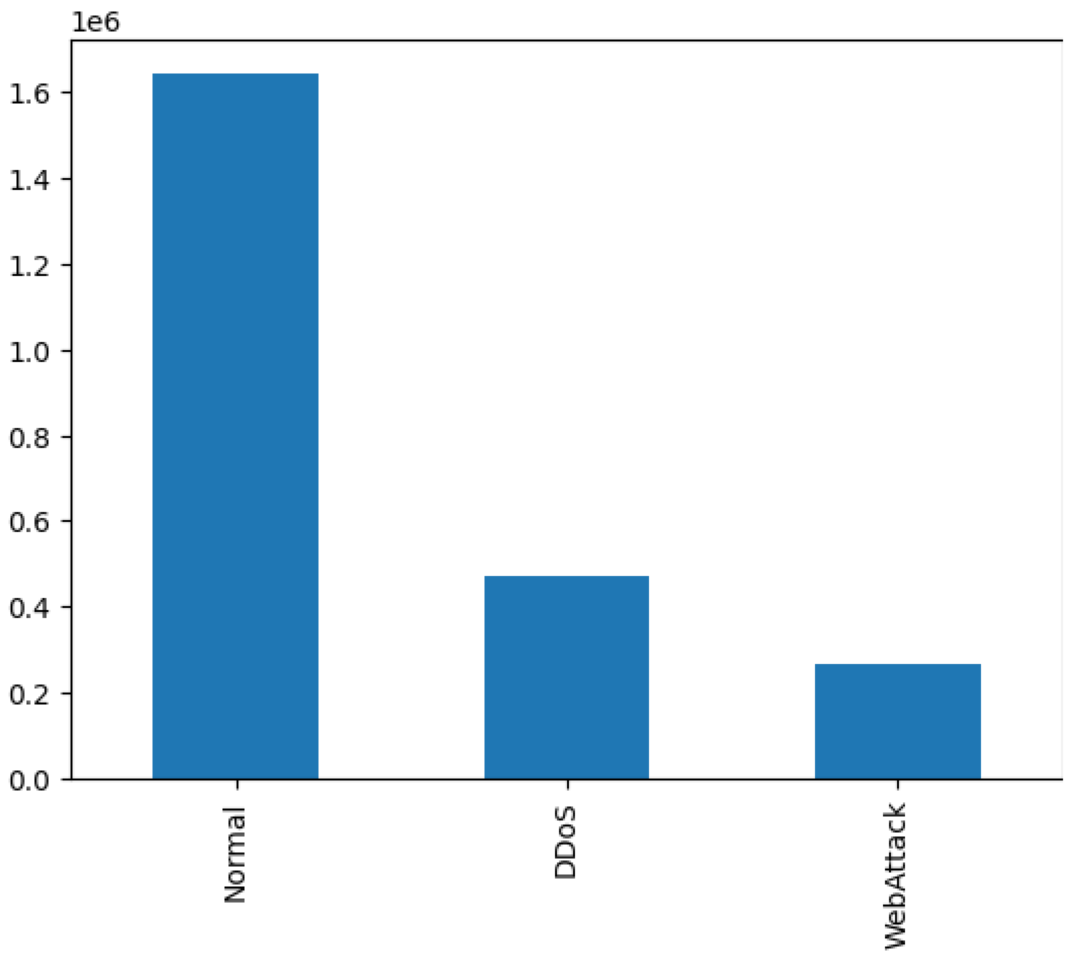


Figure 6: Class Balancing

The Figure 7, illustrate the code to balance the class by random sampling to minority number.

```
categories = data['Attack_type'].unique()
categories
```

```
array(['Normal', 'WebAttack', 'DDoS'], dtype=object)
```

```
n=len(categories)
data1 = pd.DataFrame(columns=data.columns)
for cat in categories:
    data1 = data1.append(data.loc[data['Attack_type']==cat].sample(20000))
```

```
data = data1
del data1
data['Attack_type'].value_counts().plot(kind='bar')
```

```
<Axes: >
```

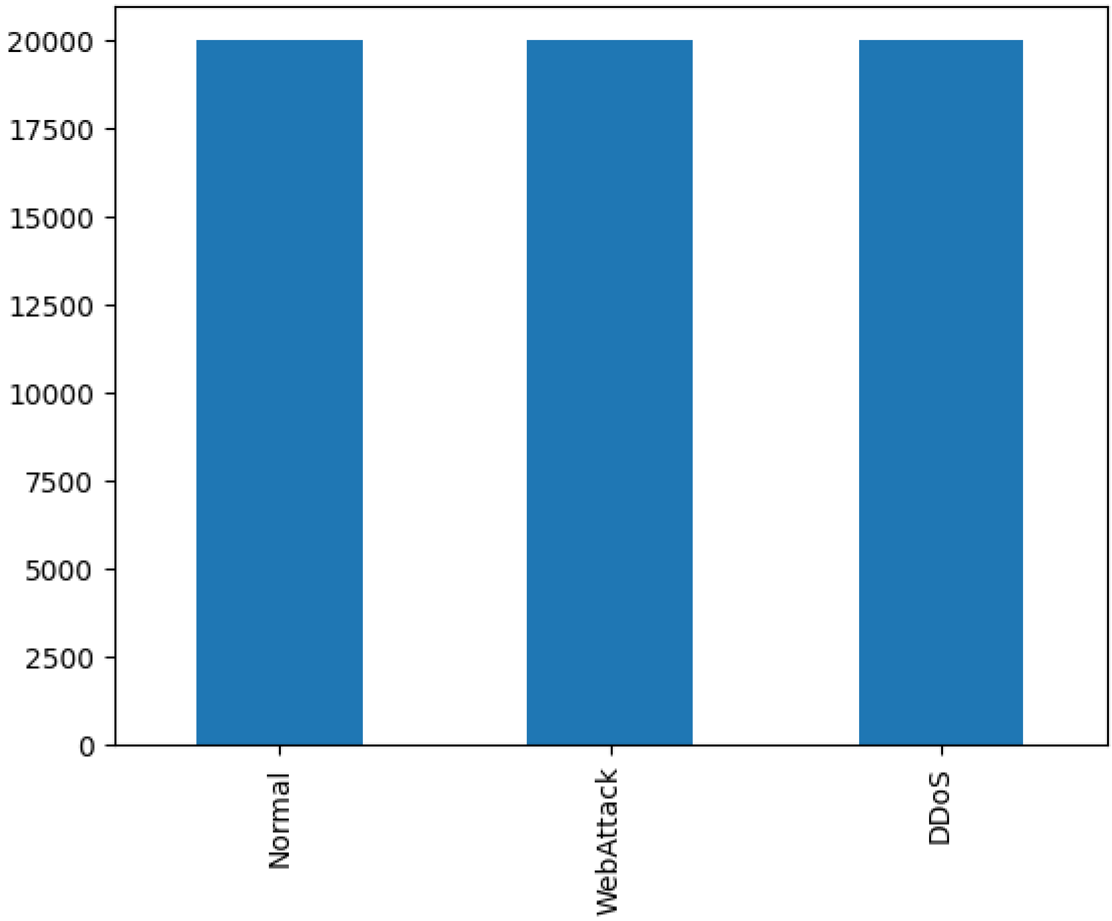


Figure 7: Class Balancing

5 Label Encoding and Feature Selection

The Figure 8, illustrate the code to encode all the object type columns and then separating the feature and target data.

```
le = LabelEncoder()
```

```
col_list = data.select_dtypes(include = "object").columns
for cols in col_list:
    data[cols] = le.fit_transform(data[cols].astype(str))
```

```
X= data.drop(['Attack_type'], axis=1)
y = data['Attack_type']
```

*

Figure 8: Label Encoding

Figures 9 show the code used for feature selection using chi-square and then splitting the data into training and test data .

```
X_new = SelectKBest(chi2, k=25).fit_transform(X, y)
X_new.shape
```

```
(60000, 25)
```

```
# split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X_new, y, test_size=0.05, random_state=1234)
```

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((57000, 25), (3000, 25), (57000,), (3000,))
```

```
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
```

```
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)
```

```
X_train.shape, X_test.shape
```

```
((57000, 25, 1), (3000, 25, 1))
```

Figure 9: Feature Selection

7 Deep Learning Models

7.1 Variational Autoencoder

```
encoder_inputs = keras.Input(shape=(X_test.shape[1],1))
encoder = layers.SimpleRNN(512, return_sequences=True)(encoder_inputs)
encoder = layers.SimpleRNN(128, return_sequences=True)(encoder)
encoder = layers.SimpleRNN(64)(encoder)
```

```
output = layers.Dense(units=64, activation='sigmoid')(encoder)
output = layers.Dropout(.2)(output)
decoder = layers.Dense(units=4, activation='sigmoid')(output)
```

```
var = keras.Model(encoder_inputs, decoder, name="VAR")
var.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
var.summary()
```

```
Model: "VAR"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 25, 1)]           0

 simple_rnn (SimpleRNN)      (None, 25, 512)           263168

 simple_rnn_1 (SimpleRNN)    (None, 25, 128)           82048

 simple_rnn_2 (SimpleRNN)    (None, 64)                12352

 dense (Dense)               (None, 64)                4160

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 4)                 260

=================================================================
Total params: 361,988
Trainable params: 361,988
Non-trainable params: 0
_________________________________________________________________
```

```
history = var.fit(X_train, y_train, epochs = 10, validation_split=0.02)
```

```
Epoch 1/10
```

Figure 10: Implementation of Variational Autoencoder

7.2 LSTM

```
: lstm = keras.Sequential()
  lstm.add(layers.LSTM(512, activation='relu', input_shape=(X_test.shape[1], 1)))
  lstm.add(layers.Dense(128))
  lstm.add(layers.Dense(32))
  lstm.add(layers.Dense(4))
  lstm.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
  lstm.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 512)               1052672

 dense_2 (Dense)             (None, 128)               65664

 dense_3 (Dense)             (None, 32)                4128

 dense_4 (Dense)             (None, 4)                 132

=================================================================
Total params: 1,122,596
Trainable params: 1,122,596
Non-trainable params: 0
_________________________________________________________________
```

```
: history = lstm.fit(X_train, y_train, epochs = 10, validation_split=0.02)
```

Figure 11: Implementation of LSTM

7.3 RNN

```
rnn = keras.Sequential([
    layers.SimpleRNN(32, input_shape=(X_test.shape[1], 1)),
    layers.Dense(10, activation='relu'),
    layers.Dense(4, activation='sigmoid')
])
rnn.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn_3 (SimpleRNN)    (None, 32)                1088

 dense_5 (Dense)             (None, 10)                330

 dense_6 (Dense)             (None, 4)                 44

=================================================================
Total params: 1,462
Trainable params: 1,462
Non-trainable params: 0
_________________________________________________________________
```

```
rnn.compile(loss="sparse_categorical_crossentropy", metrics=["accuracy"])
rnn.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn_3 (SimpleRNN)    (None, 32)                1088

 dense_5 (Dense)             (None, 10)                330

 dense_6 (Dense)             (None, 4)                 44

=================================================================
Total params: 1,462
Trainable params: 1,462
Non-trainable params: 0
_________________________________________________________________
```

```
history = rnn.fit(X_train, y_train, epochs = 10, validation_split=0.02)
```

Figure 12: Implementation of RNN

7.4 GRU

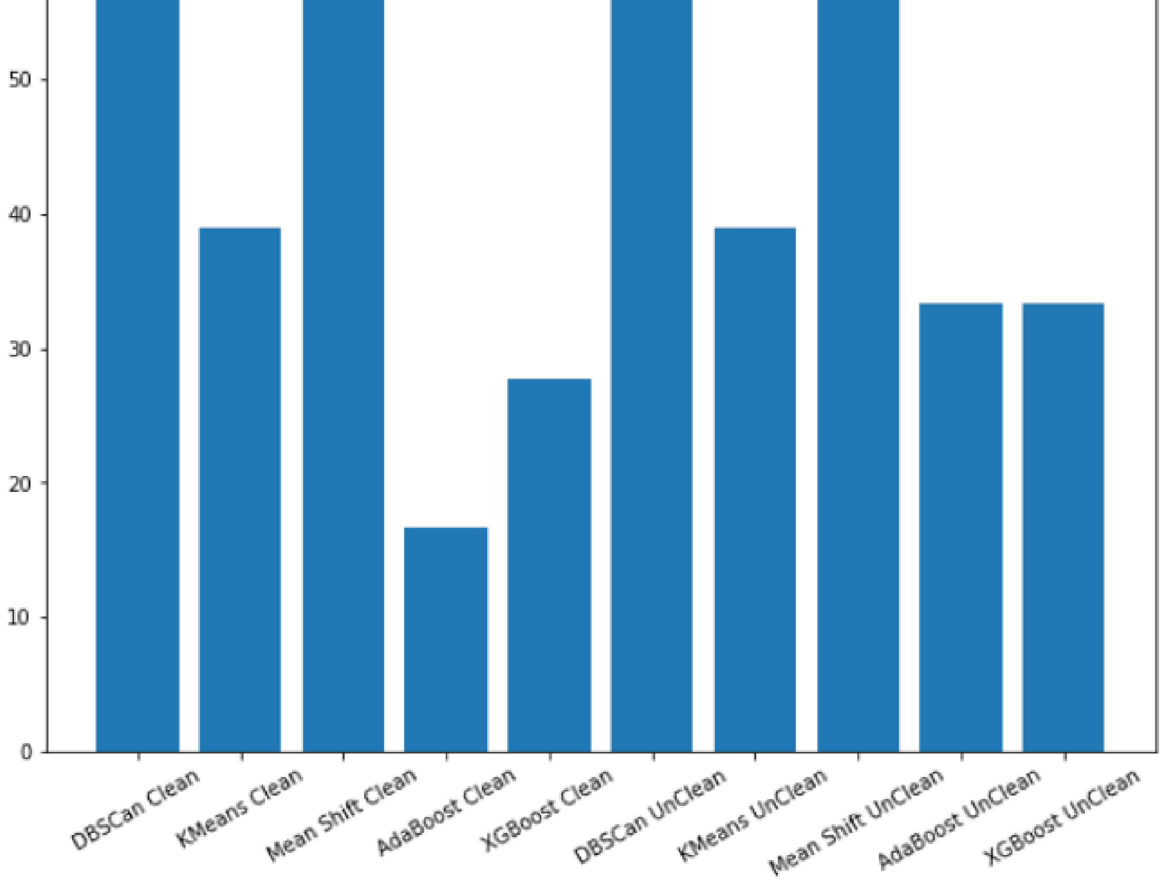
```
: gru = keras.Sequential([
      layers.GRU(32, input_shape=(X_test.shape[1], 1)),
      layers.Dense(10, activation='relu'),
      layers.Dense(4, activation='sigmoid')
  ])
```

```
: gru.compile(loss="sparse_categorical_crossentropy", metrics=["accuracy"])
  gru.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 32)                3360

 dense_7 (Dense)             (None, 10)                330

 dense_8 (Dense)             (None, 4)                 44

=================================================================
Total params: 3,734
Trainable params: 3,734
Non-trainable params: 0
_________________________________________________________________
```

```
: history = gru.fit(X_train, y_train, epochs = 10, validation_split=0.02)
```

Figure 13: Implementation of GRU



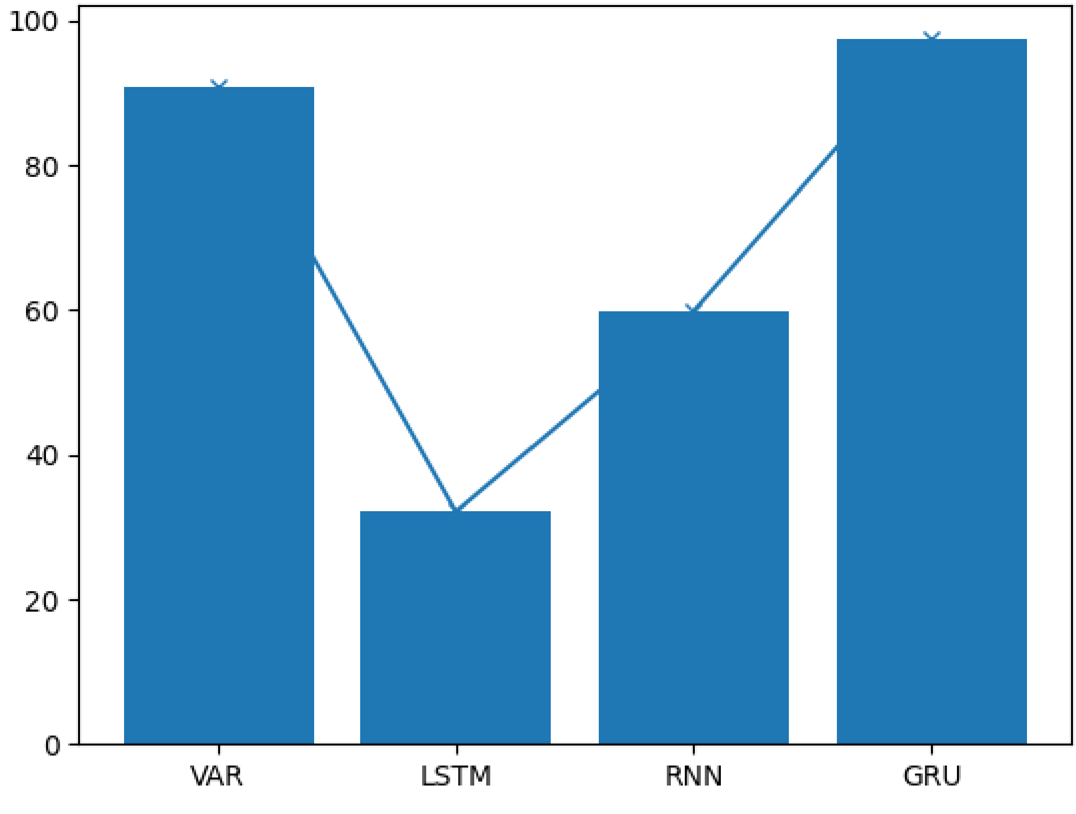
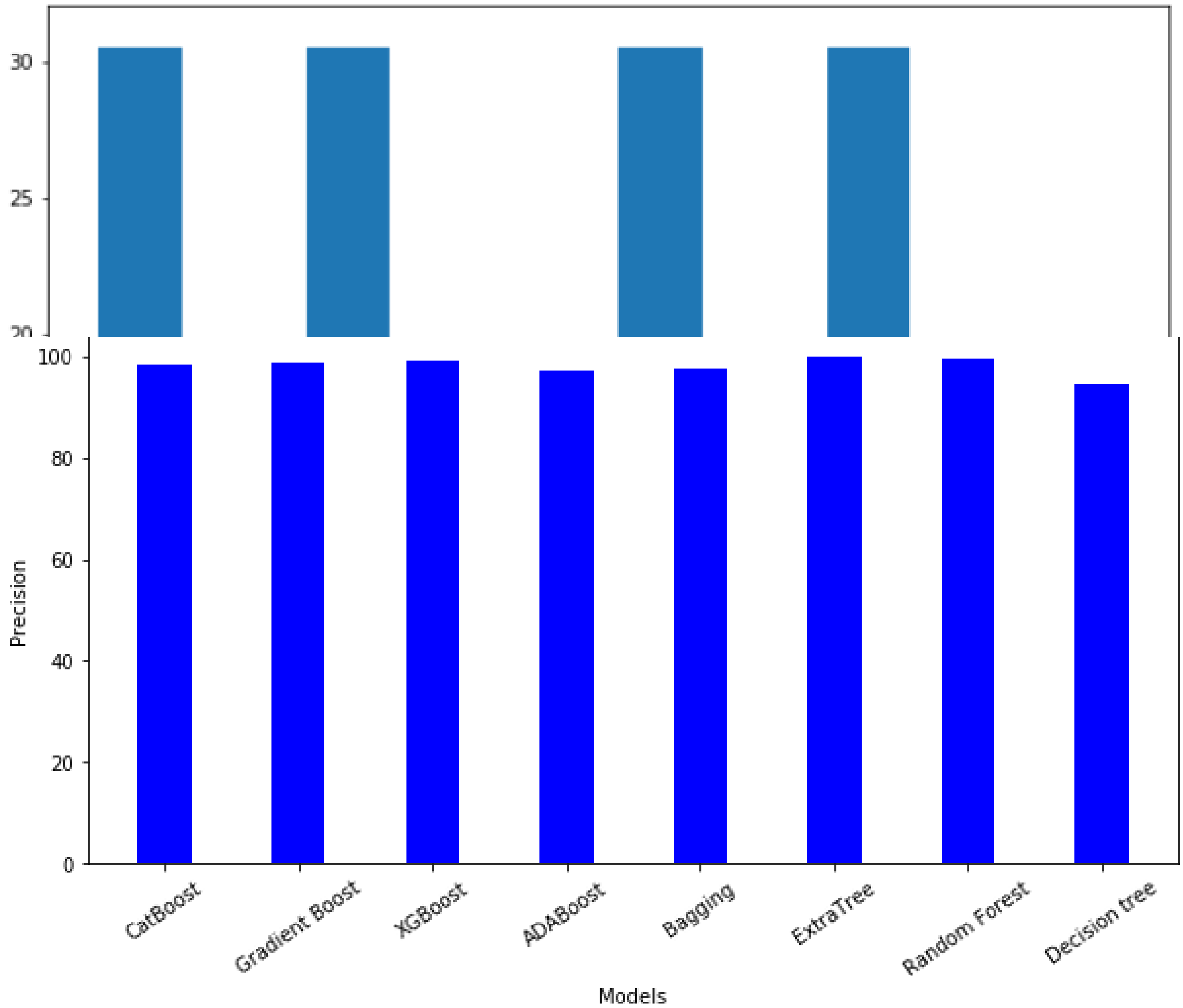


Figure 15: Accuracy

```
plt.figure(figsize=(10,8))
plt.bar(result['Model'],result['Precision'])
plt.xticks(rotation=30)
```

([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], <a list of 10 Text xticklabel objects>)



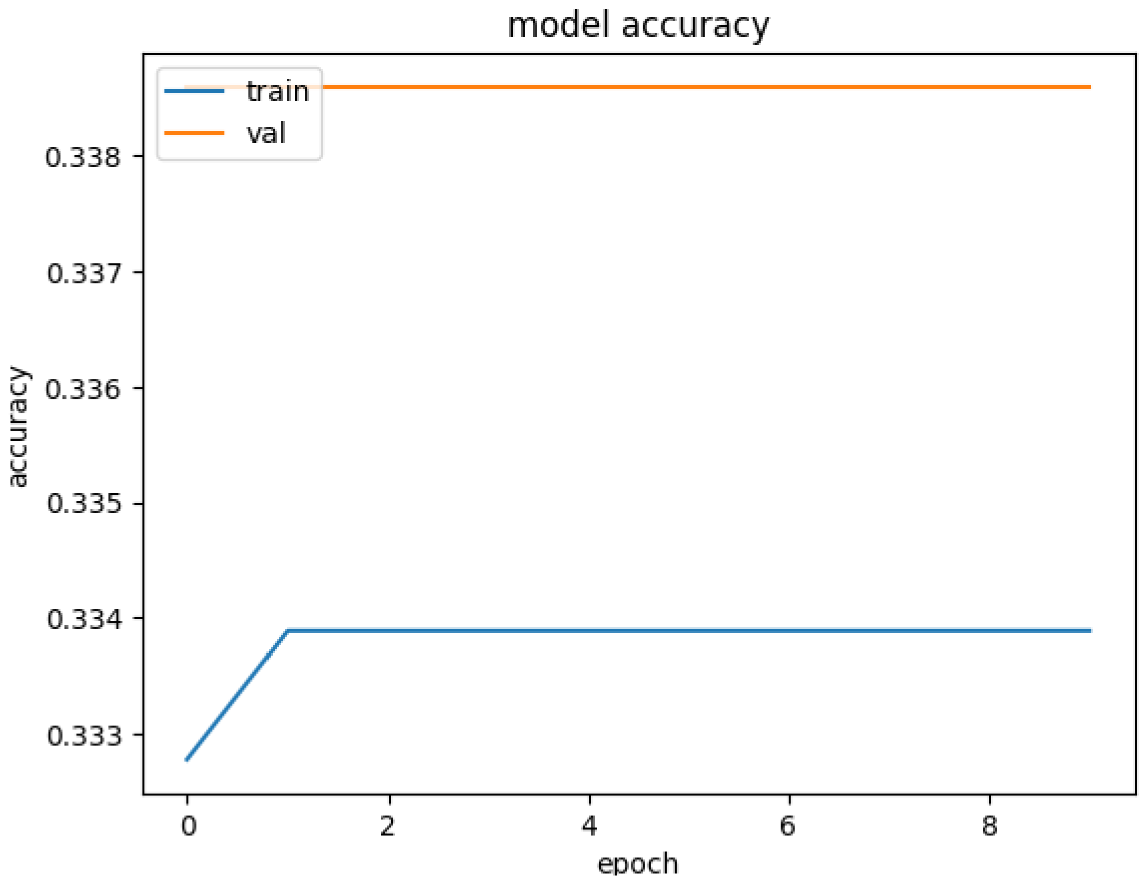


Figure 17: Training accuracy and Validation Accuracy of LSTM

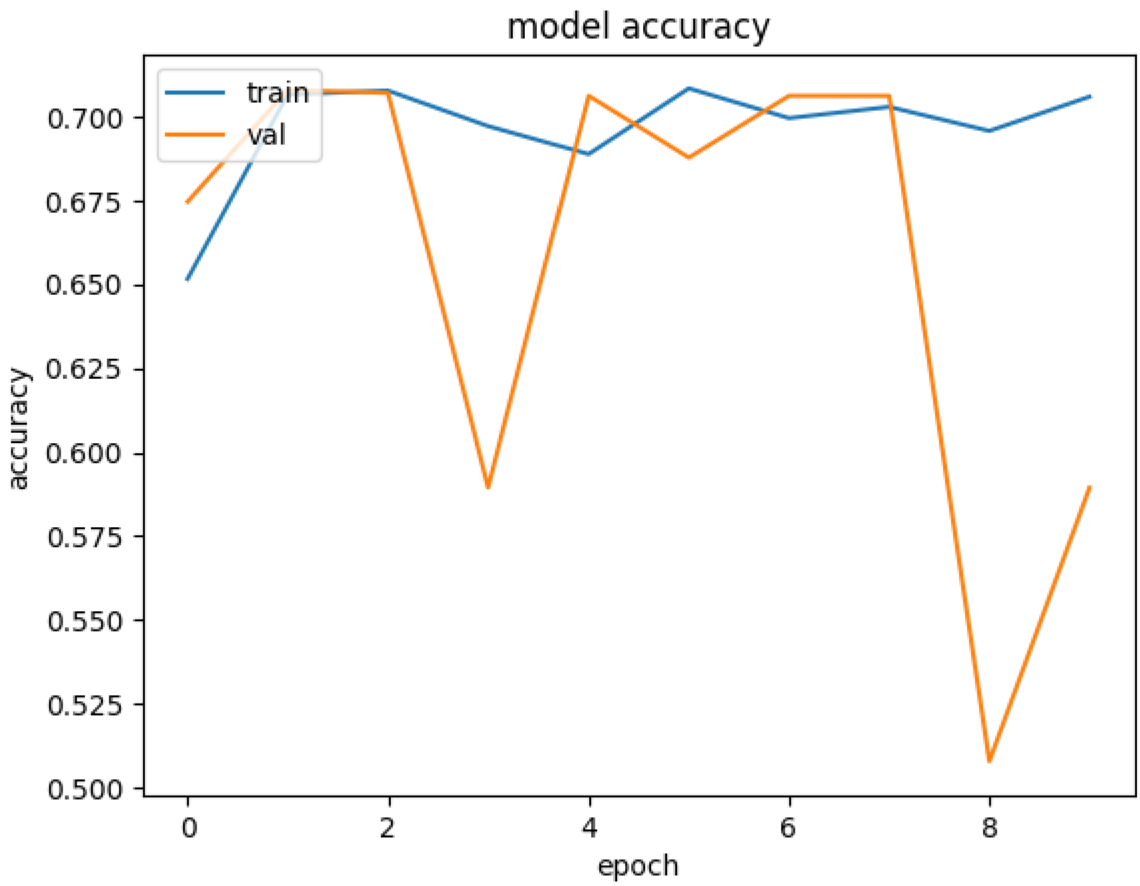


Figure 18: Training accuracy and Validation Accuracy of RNN

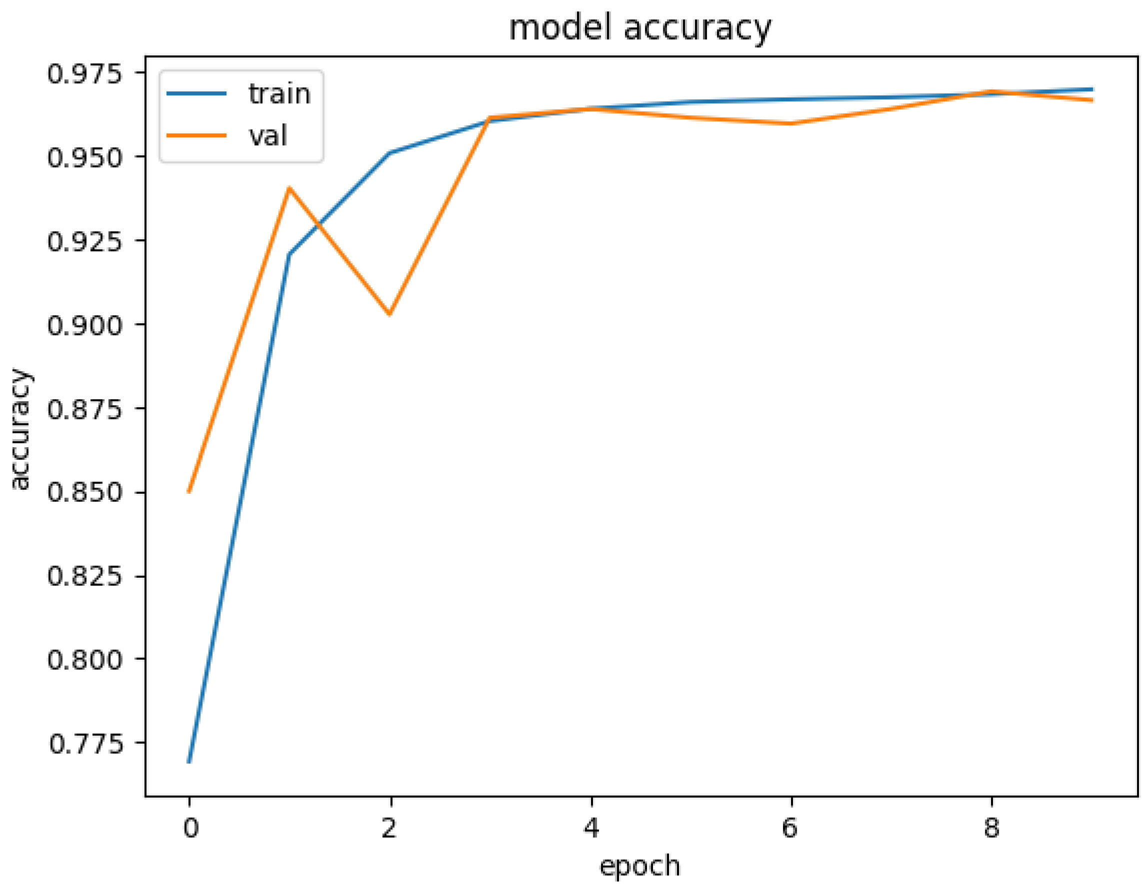


Figure 19: Training accuracy and Validation Accuracy of GRU

References

https://www.kaggle.com/datasets/mohamedamineferrag/edgeiiotset-cyber-security-dataset-of-iot-iiot

Understanding Variational Autoencoders (VAEs) | by Joseph Rocca | Towards Data Science

A Gentle Introduction to Long Short-Term Memory Networks by the Experts - MachineLearningMastery.com

Introduction to Recurrent Neural Network - GeeksforGeeks