Configuration Manual

MSc Research Project
Master of Science in Cyber Security

Albin Shaju
Student ID: 23215496

School of Computing
National College of Ireland

Supervisor: Prof. Jawad Salahuddin

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Albin Shaju
Student ID:	23215496
Programme:	Master of Science in Cyber Security **Year:** 2024
Module:	MSc Research Project
Lecturer:	Prof. Jawad Salahuddin
Submission Due Date:	Thursday, 12 December 2024
Project Title:	Exploring Machine Learning Approaches for Robust Anomaly Detection and Responsive Security in IoT Frameworks
Word Count:	680 **Page Count:** 8

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: Albin Shaju

Date: 12 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

Albin Shaju
Student ID: 23215496

1 Introduction

The purpose of this configuration manual is to give the complete description of setting up the experimental setup and implementing the proposed intrusion detection system (IDS) using machine learning and deep learning models. It then presents the technical details, software requirements, and step by step processes to reproduce its research environment in a flawless manner. It divides into sections on experimental setup, tools and technologies, and implementation steps. It has included detailed steps for importing the libraries, preprocessing data and training and testing of machine learning model (SVC, XGBoost, ConvLSTM). This manual also details the hardware and software specifications of the system used for this project. The main goal is to create a comprehensive guide that allows researchers and practitioners to easily reproduce the results and to continue building on the work.

2 Hardware Configuration

The configuration of the system used for this project is given below:

Processor: AMD Ryzen 7 5700U with Radeon Graphics @1.80 GHz
RAM: 24 GB
SSD: 1 TB
HDD: 500 GB
OS: Windows 11
System Type: 64-bit operating system, x64-based processor

3 Technologies and Software used

The software and technologies that were used in this project are:

- Anaconda Navigator 2.6.3
- Jupyter Notebook 7.2.2
- Python 3.9.20
- Pickle
- TensorFlow: 2.8.2
- Scikit-learn: 0.24.2
- XGBoost: 2.1.3
- Pandas: 1.3.5
- NumPy: 1.21.5

- Matplotlib: 3.5.3
- Seaborn: 0.11.2
- Yagmail: 0.15.293

4 Other Requirements

CICIoT2023 Dataset(*IoT Dataset 2023 | Datasets | Research | Canadian Institute for Cybersecurity | UNB*, n.d.)

Email Account for Notifications: A valid email id is needed to setup the system to send notifications when malicious traffic is detected.

5 Implementation

5.1 Initial steps

Step 1: Anaconda navigator was installed in the system and configured for the project.

Step 2: Created a new environment in anaconda as myproject and installed Jupyter Notebook in it.

Step 3: Installed python and other necessary libraries in the environment.

Step 4: The dataset was downloaded from its website.

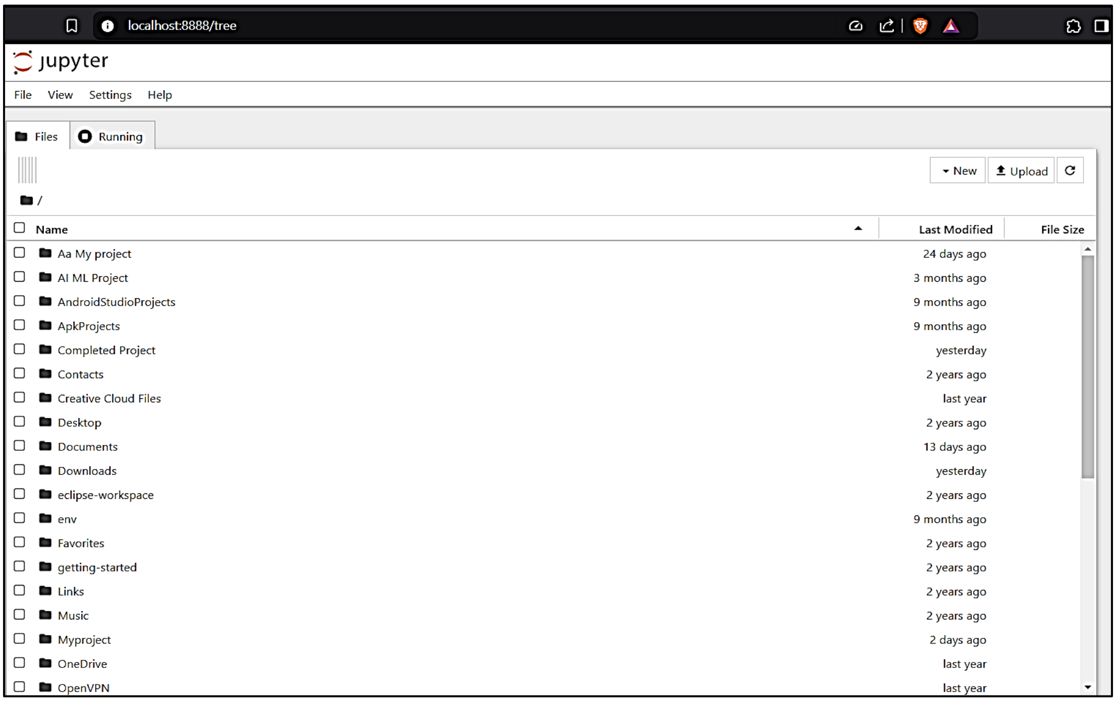

Figure 1: Jupyter Notebook Homepage

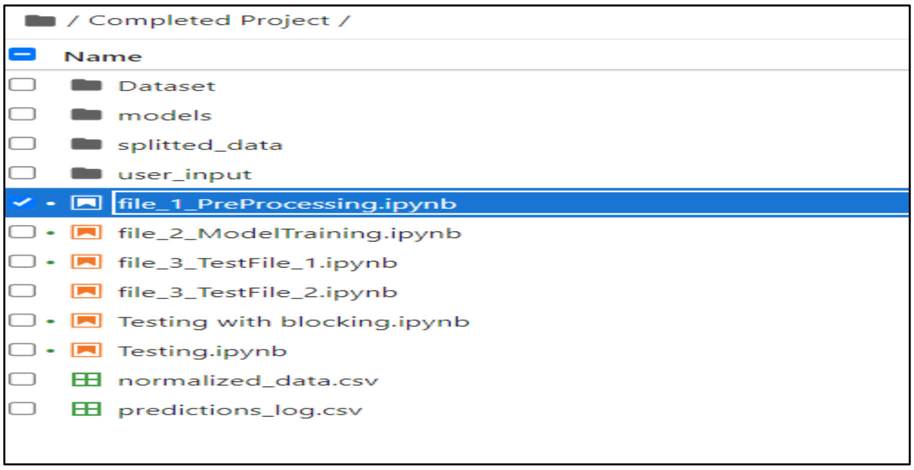

Figure 2: File Structure

5.2 Preprocessing

Step 5: A new notebook was created for preprocessing the dataset.

Step 6: The libraries required for the program were imported.

Step 7: Loaded 2 parts of the dataset as it was a very large dataset and required more computing capabilities and time. Both datasets were analysed and combined.

Step 8: Analyzed and visualized the dataset to identify class imbalances.(*CICIoT2023/01-Data_Exploration.Ipynb at Main · Plumpmonkey/CICIoT2023 · GitHub*, n.d.)

Step 9: Filtered and balanced the dataset by retaining specific labels and samples.(*Ensemble_learning/CIC_IOT_Dataset2023_dataset_preprocessing.Md at Main · Nickjeffrey/Ensemble_learning · GitHub*, n.d.)

Step 10: MinMaxScaler was used to normalize feature values.

Step 11: The dataset was spit into two as training (80%) and testing (20%) subsets.

```
[1]: import warnings
     warnings.filterwarnings("ignore")

     import pandas as pd
     pd.set_option("display.max_columns",None)
     pd.set_option("display.max_rows",None)
     import numpy as np
     import matplotlib.pyplot as plt
     %matplotlib inline
     plt.rcParams["font.size"]=15
     from sklearn.preprocessing import MinMaxScaler
     import pickle
     import os
     from sklearn.model_selection import train_test_split
```

Figure 3: Importing Libraries

```
[2]: #Loading dataset
     df1 = pd.read_csv("Dataset/part-00000-363d1ba3-8ab5-4f96-bc25-4d5862db7cb9-c000.csv")
     df2 = pd.read_csv("Dataset/part-00168-363d1ba3-8ab5-4f96-bc25-4d5862db7cb9-c000.csv")

[3]: df1.shape

[3]: (238687, 47)

[4]: df2.shape

[4]: (234745, 47)

[5]: df1.head()
```

Figure 4: Analysing the Dataset

```
[21]: #Data Extraction
      # label list
      label_list = ['BenignTraffic', 'DDoS-ICMP_Flood', 'DDoS-UDP_Flood', 'DDoS-TCP_Flood',
                    'DDoS-PSHACK_Flood', 'DDoS-SYN_Flood', 'DDoS-RSTFINFlood', 'DDoS-SynonymousIP_Flood']

      # Filtering the DataFrame to keep only labels in the label_list
      df = df[df['label'].isin(label_list)]

      # Ensure each label has at least 11081 records
      valid_labels = df['label'].value_counts()[df['label'].value_counts() >= 11081].index

      # Filter again to keep only valid labels
      df = df[df['label'].isin(valid_labels)]

      # For each label, keep only the first 11081 records
      df = df.groupby('label').head(11081)

      print(df['label'].value_counts())

      DDoS-RSTFINFlood           11081
      DDoS-ICMP_Flood            11081
      DDoS-SynonymousIP_Flood    11081
      DDoS-SYN_Flood             11081
      DDoS-PSHACK_Flood          11081
      DDoS-TCP_Flood             11081
      DDoS-UDP_Flood             11081
      BenignTraffic              11081
      Name: label, dtype: int64
```

Figure 5: Filtering and balancing the dataset

```
[40]: #Saving the preprocessing model
      with open(file="models/scaler.pkl", mode="wb") as file:
          pickle.dump(obj=scaler, file=file)

[41]: #saving the feature selection data
      normalized_df.to_csv("normalized_data.csv", index=False)

[42]: normalized_df.shape

[42]: (88648, 40)

[43]: #Data Splitting
      X = normalized_df.drop(labels='label', axis=1)
      y = normalized_df[['label']]

[44]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
      print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

      (70918, 39) (17730, 39) (70918, 1) (17730, 1)
```

Figure 6: Saving the preprocessing model and Splitting the Data

5.3 Model Training

Step 12: Created a new notebook for model training.

Step 13: Loaded the preprocessed training and testing datasets.

Step 14: Imported necessary libraries and trained the models.

Step 15: Generated classification reports and confusion matrices for each model.

Step 16: Saved all trained models for future use.

```
[1]: #importing necessary libraries
     import warnings
     warnings.filterwarnings("ignore")

     import os
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     %matplotlib inline
     import seaborn as sns
     import pickle
     from sklearn.metrics import accuracy_score,classification_report,confusion_matrix

     for dirname,_,filenames in os.walk('splitted_data'):
         for filename in filenames:
             print(os.path.join(dirname,filename))

     splitted_data\X_test.csv
     splitted_data\X_train.csv
     splitted_data\y_test.csv
     splitted_data\y_train.csv

[2]: #Loading the split datasets
     X_train = pd.read_csv('splitted_data/X_train.csv')
     X_test = pd.read_csv('splitted_data/X_test.csv')
     y_train = pd.read_csv('splitted_data/y_train.csv')
     y_test = pd.read_csv('splitted_data/y_test.csv')

     print(X_train.shape,X_test.shape,y_train.shape,y_test.shape)

     (70918, 39) (17730, 39) (70918, 1) (17730, 1)
```

Figure 7: Loading Libraries and Preprocessed Dataset

```
[7]: #SVC algorith
     from sklearn.svm import SVC
     svc_model = SVC(kernel='linear', C=0.01)
     svc_model = svc_model.fit(X_train.values, y_train.values.ravel())
```

Figure 8: Training SVC Model

```
[20]: #traing clstm model
      history = clstm_model.fit(
          x=x_train,
          y=y_train,
          batch_size=32,
          epochs=10,
          validation_data=(x_test,y_test),
          callbacks=ReduceLROnPlateau(monitor='val_accuracy', patience=2, min_lr=0)
      )
```

Figure 9: Training CLSTM Model

```
[32]: # Best XGBoost model
      xgb_model = xgb_random_search.best_estimator_
      print("Best Parameters for XGBoost:", xgb_random_search.best_params_)

      Best Parameters for XGBoost: {'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.1}

[33]: # Train and evaluate the XGBoost model
      xgb_model.fit(X_train, y_train)
      xgb_predictions = xgb_model.predict(X_test)
```

Figure 10: Training and Evaluating XGBoost Model

5.4 Response System Implementation

Step 17: Created a new notebook for the response system.

Step 18: Loaded the pre-trained XGBoost model, selected features, and scaler.

Step 19: Implemented an email notification function using yagmail.

Step 20: Developed blocklist verification to skip already-blocked IPs.

Step 21: Created a prediction function to classify incoming traffic and log results.

Step 22: Tested the inference system using network traffic data.

```
[6]: def send_email_notification(sender_email, app_password, recipient_email, ip_address, predicted_label, probability_score):
         try:
             yag = yagmail.SMTP(user=sender_email, password=app_password)
             subject = f"Alert: Suspicious Traffic Detected ({predicted_label})"
             contents = (
                 f"Suspicious traffic detected:\n\n"
                 f"IP Address: {ip_address}\n"
                 f"Predicted Label: {predicted_label}\n"
                 f"Probability Score: {probability_score}\n\n"
                 f"Please take appropriate actions."
             )
             yag.send(to=recipient_email, subject=subject, contents=contents)
             print(f"Email notification sent to {recipient_email}.")
         except Exception as e:
             print(f"Failed to send email: {e}")

[22]: def prediction(input_data, features, model, scaler, output_csv='predictions_log.csv'):
          # Drop 'ip_address' if present
          if 'ip_address' in input_data.columns:
              ip_address = input_data['ip_address'].iloc[0]
              input_data = input_data.drop('ip_address', axis=1)
          else:
              ip_address = 'Unknown'

          # Check if the predictions log exists and if the IP address is blocked
          if os.path.exists(output_csv):
              blocked_ips = pd.read_csv(output_csv)['ip_address'].unique()
              if ip_address in blocked_ips:
                  print(f"IP address '{ip_address}' is already in the blocklist.")
                  return f"IP address '{ip_address}' is blocked."

          # Extract and scale features
          input_data_multiclass = input_data[features]
          input_data_multiclass_scaled = scaler.transform(input_data_multiclass.values)
```

Figure 11: Email Notification Function

```
[22]: def prediction(input_data, features, model, scaler, output_csv='predictions_log.csv'):
          # Drop 'ip_address' if present
          if 'ip_address' in input_data.columns:
              ip_address = input_data['ip_address'].iloc[0]
              input_data = input_data.drop('ip_address', axis=1)
          else:
              ip_address = 'Unknown'

          # Check if the predictions log exists and if the IP address is blocked
          if os.path.exists(output_csv):
              blocked_ips = pd.read_csv(output_csv)['ip_address'].unique()
              if ip_address in blocked_ips:
                  print(f"IP address '{ip_address}' is already in the blocklist.")
                  return f"IP address '{ip_address}' is blocked."
```

Figure 12: Blocklist Verification

```
[30]: # Load the input file and perform the prediction
      user_input_filepath = "user_input/user_input_0/DDoS-SYN_Flood_test (3).csv"
      df = pd.read_csv(user_input_filepath)
      df.head()
```

[30]:

	ip_address	flow_duration	Header_Length	Protocol Type	Duration	Rate	Srat
0	8.8.8.8	0	54	6	64	2.366215	2.36621

```
[31]: # Call the prediction function
      prediction(df, selected_features, model, scaler)

      Class Index: 4
      Class Label: DDoS_SYN_Flood
      Probability Score: 100.00%
      Email notification sent to errorguy000@gmail.com.
```

Figure 13: Prediction and Alert Function Implementation

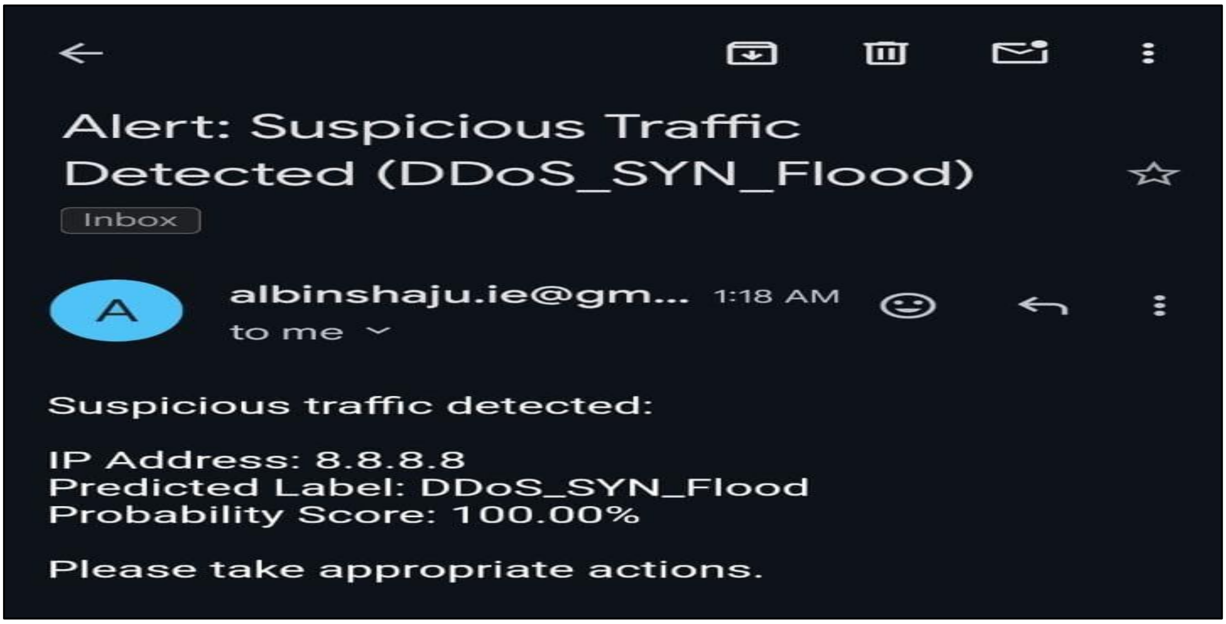

Figure 14: Email Alert

References

CICIoT2023/01-Data_Exploration.ipynb at main · plumpmonkey/CICIoT2023 · GitHub. (n.d.). Retrieved December 12, 2024, from https://github.com/plumpmonkey/CICIoT2023/blob/main/01-Data_Exploration.ipynb

ensemble_learning/CIC_IOT_Dataset2023_dataset_preprocessing.md at main · nickjeffrey/ensemble_learning · GitHub. (n.d.). Retrieved December 12, 2024, from https://github.com/nickjeffrey/ensemble_learning/blob/main/CIC_IOT_Dataset2023_dataset_preprocessing.md

IoT Dataset 2023 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. (n.d.). Retrieved December 12, 2024, from https://www.unb.ca/cic/datasets/iotdataset-2023.html