National College of Ireland

Configuration Manual

MSc Research Project
Msc. In Cyber Security

Parth Bhardwaj
Student ID: x21169578

School of Computing
National College of Ireland

Supervisor: Dr. Arghir Nicolae Moldovan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Parth Bhardwaj		
Student ID:	x21169578		
Programme:	Msc. In Cyber Security	**Year:**	2022-2023
Module:	Msc. Research Project		
Lecturer:	Dr. Arghir Nicolae Moldovan		
Submission Due Date:	15/12/2022		
Project Title:	Finding IoT privacy issues through malware Detection using XgBoost machine learning technique		
Word Count:	654	**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: Parth Bhardwaj

Date: 15/12/2022

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

Parth Bhardwaj
Student ID: x21169578

1 Introduction

This document offers information on the proposed model's configuration and needs, such as technical details and required software. This setup configuration also covers how to execute the algorithms required to construct the suggested model with the different datasets.

2 Hardware

- 11th Gen Intel(R) Core (TM) i7-1165G7 @ 2.80GHz 2.80 GHz
- 16.0 GB Ram, DDR4
- Windows 11, 64-bit operating system
- 1 TB Solid State Drive
- Graphics 2GB

3 Software Specifications

The Integrated Development Environment utilized for the Research Project is anaconda prompt, and the programming language used for the same is Python. Various other packages and libraries are also utilized in this for better results and systematical approaches.

- Anaconda Prompt
- Python 3.9.12
- Anaconda Navigator
- Sublime Text
- NumPy
- Matplotlib
- Pandas
- Scikit-learn
- Tkinter
- SelectKbest
- XgBoost Library
- Pickle

4 Configuration Steps

1. Download and Install Anaconda3
2. Ml_env.rar extraction to the anaconda’s environment folder

3. Extract Detection Malware (IoT)

4. Open the Anaconda Navigator

5. Start the ml_env and then open the anaconda3 prompt

6. Navigate the folder Detection Malware (IoT).

7. Use command cd /d to change the directory to the Detection Malware (IoT).

8. Now python final1.py and python final2.py for GUI result of different dataset

9. Final1.py is for CICIDS-2017 dataset & Final2.py is for IoT-23 dataset.

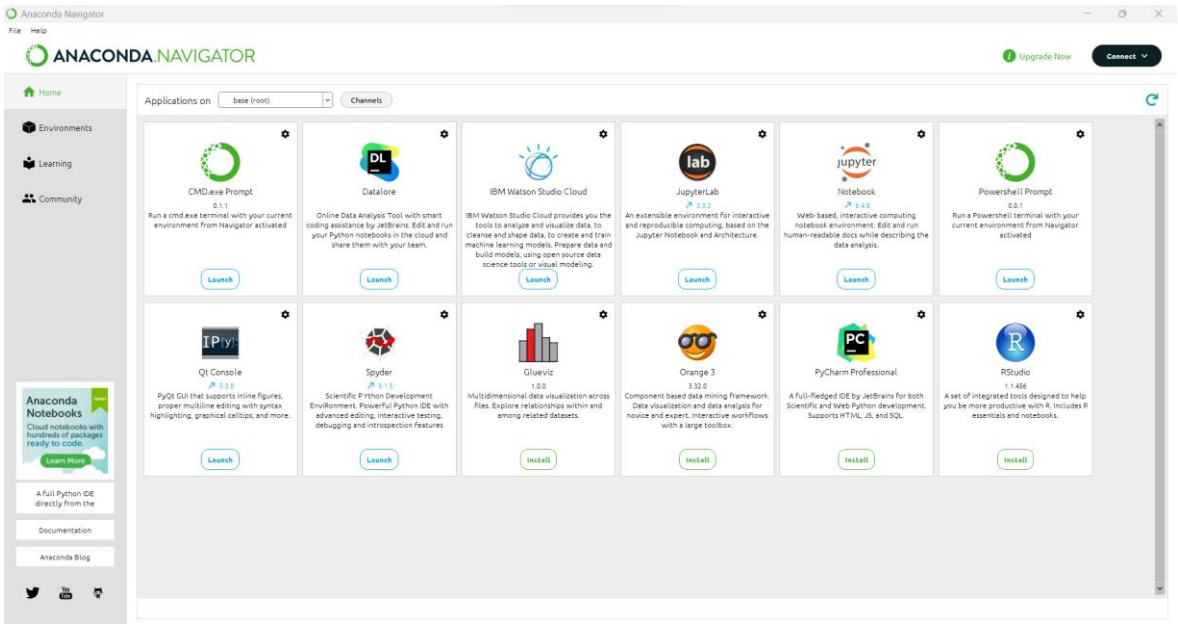

Figure 1 : Anaconda Navigator

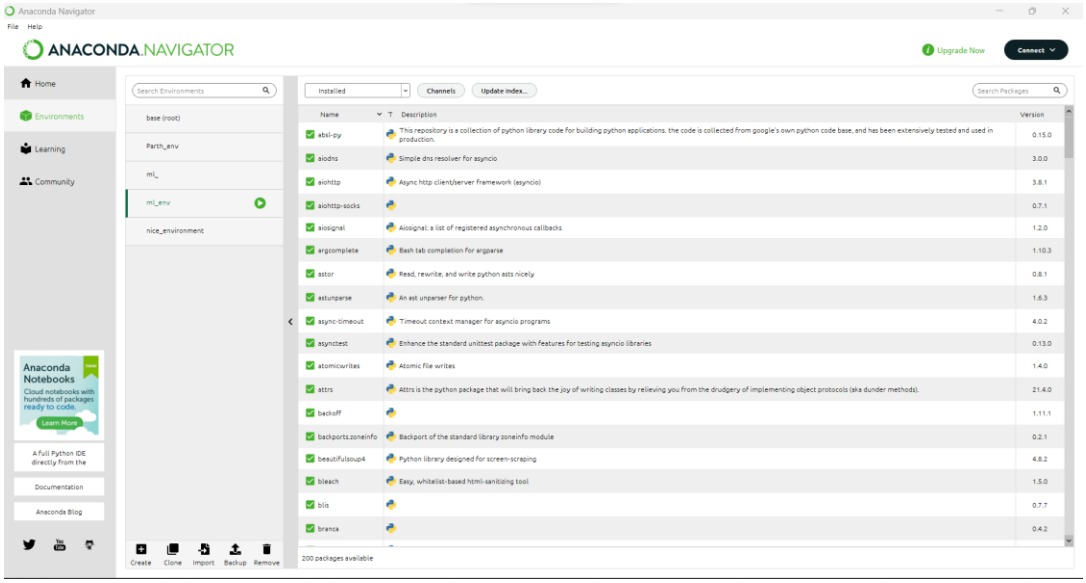

Figure 2 : ml_env

5 Procedure

5.1 Pre-processing of the data

- Load the data from the dataset and merge in case of CICIDS-2017 dataset
- Pre-process the IoT-23

```
#perform merging
_dframe = pd.concat([d_1,d_2])
_dframe = pd.concat([_dframe,d_3])
_dframe = pd.concat([_dframe,d_4])


#Converting to numeric value
for iteration in _dframe.columns:
    _dframe = _dframe[_dframe[iteration] != "Infinity"]
    _dframe = _dframe[_dframe[iteration] != np.nan]
    _dframe = _dframe[_dframe[iteration] != ",,"]
_dframe[['Flow Bytes/s', ' Flow Packets/s']] = _dframe[['Flow Bytes/s', ' Flow Packets/s']].apply(pd.to_numeric)
#print(_dframe.shape)

_dframe.drop([' Bwd PSH Flags'], axis=1, inplace=True)
_dframe.drop([' Bwd URG Flags'], axis=1, inplace=True)
_dframe.drop(['Fwd Avg Bytes/Bulk'], axis=1, inplace=True)
_dframe.drop([' Fwd Avg Packets/Bulk'], axis=1, inplace=True)
_dframe.drop([' Fwd Avg Bulk Rate'], axis=1, inplace=True)
_dframe.drop([' Bwd Avg Bytes/Bulk'], axis=1, inplace=True)
_dframe.drop([' Bwd Avg Packets/Bulk'], axis=1, inplace=True)
_dframe.drop(['Bwd Avg Bulk Rate'], axis=1, inplace=True)


#replacing
_dframe.loc[_dframe[' Label'] != 'BENIGN', ' Label']=1
_dframe.loc[_dframe[' Label'] == 'BENIGN', ' Label']=0


#replace infinte values with nan & nan with Zeroes
_dframe.replace([np.inf, -np.inf], np.nan, inplace=True)
_dframe.fillna(0, inplace=True)

#separating independent and dependent variable
data_y=_dframe[' Label']
data_x=_dframe.drop([' Label'],axis=1)
```

Figure 3 : Pre-Processing of CICIDS-2017

```
#load dataframe
def data_loading(filepath):
    df = pd.read_table(filepath, names=columns, skiprows=8)
    df.drop(df.shape[0]-1, inplace=True)
    return df



list_all_dfs = []

#append loaded dataframe to a list
for subdir, dirs, files in os.walk(path):
    for file in files:
        filepath = os.path.join(subdir, file)
        if filepath.endswith('conn.log.labeled'):
            curr_df = data_loading(filepath)
            list_all_dfs.append(curr_df)

for df in list_all_dfs:
    print(df.shape)

concat_data = pd.concat(list_all_dfs, ignore_index=True)
print(concat_data.shape)
print(concat_data.head())

# print(concat_data['label'].value_counts())

#replace column name (preprocess)
concat_data.loc[(concat_data.label == '-   Benign   -'), 'label'] = 'Benign'
concat_data.loc[(concat_data.label == '(empty)   Benign   -'), 'label'] = 'Benign'
concat_data.loc[(concat_data.label == '-   benign   -'), 'label'] = 'Benign'

#replacing
concat_data.loc[concat_data['label'] != 'Benign', 'label']=1
concat_data.loc[concat_data['label'] == 'Benign', 'label']=0

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

#concat_data.to_csv('check.csv')
#remove unnecessary columns #column 'service' have many empty values so removing it
non_informative_columns = ['ts', 'uid', 'id.orig_h', 'id.orig_p', 'id.resp_h', 'id.resp_p', 'local_orig', 'local_resp', 'history','service']
concat_data.drop(non_informative_columns, axis=1, inplace=True)

#checking null values
print(concat_data.isna().sum())

concat_data.replace({"-":0}, inplace=True)

print(concat_data['proto'].value_counts())
print(concat_data['conn_state'].value_counts())

#convert categorical values to numerical values
concat_data.loc[concat_data['proto'] == 'tcp', 'proto']=0
```

Figure 4 : Pre-Processing of IoT-23

5.2 Feature Selection of the datasets

```
#feature selection
get_features = SelectKBest(score_func=f_classif, k="all")

# learn relationship from training data
get_features.fit(data_x,data_y)


list_col=[]
#append columns to a list

for k in data_x.columns:
    list_col.append(k)


feat=[]
#append feature scores to a list
for i in range(len(get_features.scores_)):
  # print('Feature %d: %f' % (i, get_features.scores_[i]))
  feat.append(get_features.scores_[i])

#convert to dict and sorting
convert_dict = dict(zip(list_col, feat))
p_clean_dict = {k: convert_dict[k] for k in convert_dict if not pd.isna(convert_dict[k])}
sorted_dict = dict( sorted(p_clean_dict.items(), key=operator.itemgetter(1),reverse=True))

#print(sorted_dict)
#print(len(sorted_dict))


#final dataframe wrt feature score
final_dframe=_dframe[[' PSH Flag Count',' Bwd Packet Length Std',' Min Packet Length','Bwd Packet Length Max',' Bwd Packet Length Mean',
' Avg Bwd Segment Size',' Packet Length Variance',' Packet Length Std',' Bwd Packet Length Min',' Max Packet Length',' Average Packet Size',' Packet Length Mean',' URG Flag Count','
' Destination Port',' Label']]

# ##final_dframe.to_csv("Dataset/final_dataset_cicids.csv",index=False)
#

final_dframe = pd.read_csv("Dataset/final_dataset_cicids.csv")

#data division
model_out = final_dframe[' Label']
model_in = final_dframe.drop([' Label'], axis=1)
```

Figure 5 : Feature selection CICIDS-2017

5.3 Testing and Training

```
# TRAIN - TEST SPLITTING
x_train, x_test, y_train, y_test = train_test_split(data_x, data_y, test_size=0.2)
print("\nTraining set")
print(x_train.shape)
print(y_train.shape)
print("\nTesting set")
print(x_test.shape)
print(y_test.shape)


#perform Data balancing
counter = Counter(y_train)
print("before balancing : ", counter)

nr = NearMiss()

x_train_balanced, y_train_balanced = nr.fit_resample(x_train, y_train)

counter = Counter(y_train_balanced)
print("after balancing : ", counter)

print("x_train_balanced shape:", x_train_balanced.shape)
print("y_train_balanced shape:", y_train_balanced.shape)


#standardization
scaler = StandardScaler()
x_train_balanced = scaler.fit_transform(x_train_balanced)
x_test = scaler.transform(x_test)
pickle.dump(scaler,open('Models/scaler_iot23.pkl','wb'))



#initialize xgboost classifier
model = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss')

#training
model.fit(x_train_balanced, y_train_balanced)

#prediction on test data
y_pred = model.predict(x_test)
```

Figure 6 : Testing and Training of IoT-23

```
# TRAIN - TEST SPLITTING
x_train, x_test, y_train, y_test = train_test_split(model_in, model_out, test_size=0.2)
print("\nTraining set")
print(x_train.shape)
print(y_train.shape)
print("\nTesting set")
print(x_test.shape)
print(y_test.shape)


#perform Data balancing
counter = Counter(y_train)
print("before balancing : ", counter)

nr = NearMiss()

x_train_balanced, y_train_balanced = nr.fit_resample(x_train, y_train)

counter = Counter(y_train_balanced)
print("after balancing : ", counter)

print("x_train_balanced shape:", x_train_balanced.shape)
print("y_train_balanced shape:", y_train_balanced.shape)


#standardization
scaler = StandardScaler()
x_train_balanced = scaler.fit_transform(x_train_balanced)
x_test = scaler.transform(x_test)
pickle.dump(scaler,open('Models/scaler_cicids.pkl','wb'))



#initialize xgboost classifier
model = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss')

#training
model.fit(x_train_balanced, y_train_balanced)

#prediction on test data
y_pred = model.predict(x_test)
```

Figure 7 : Testing and Training of CICIDS-2017

6 Result and Prediction

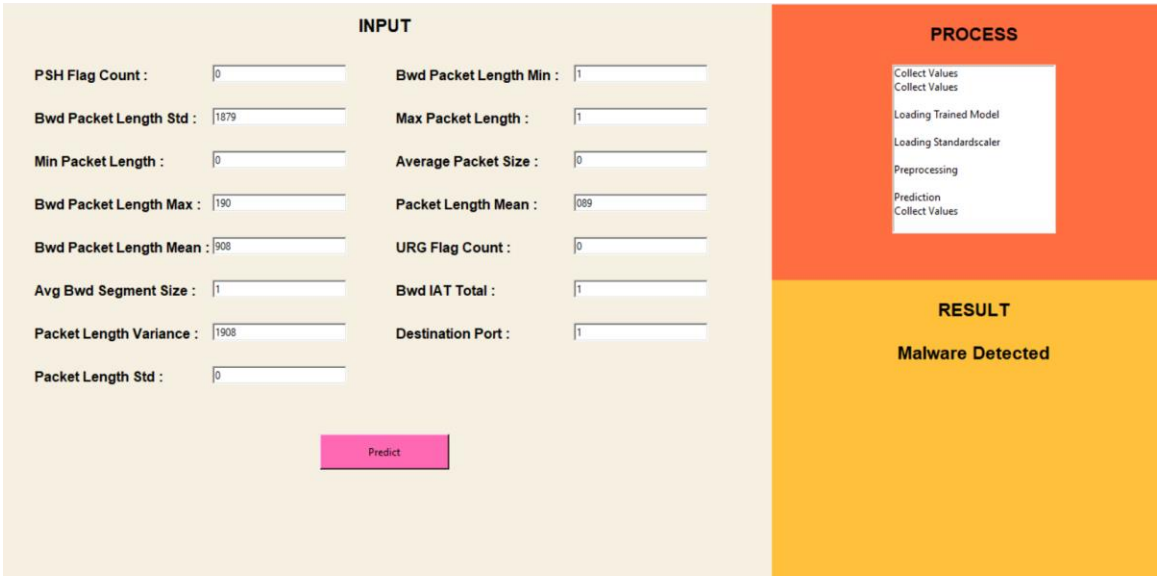

Figure 8 : Predicted Malware in CICIDS-2017

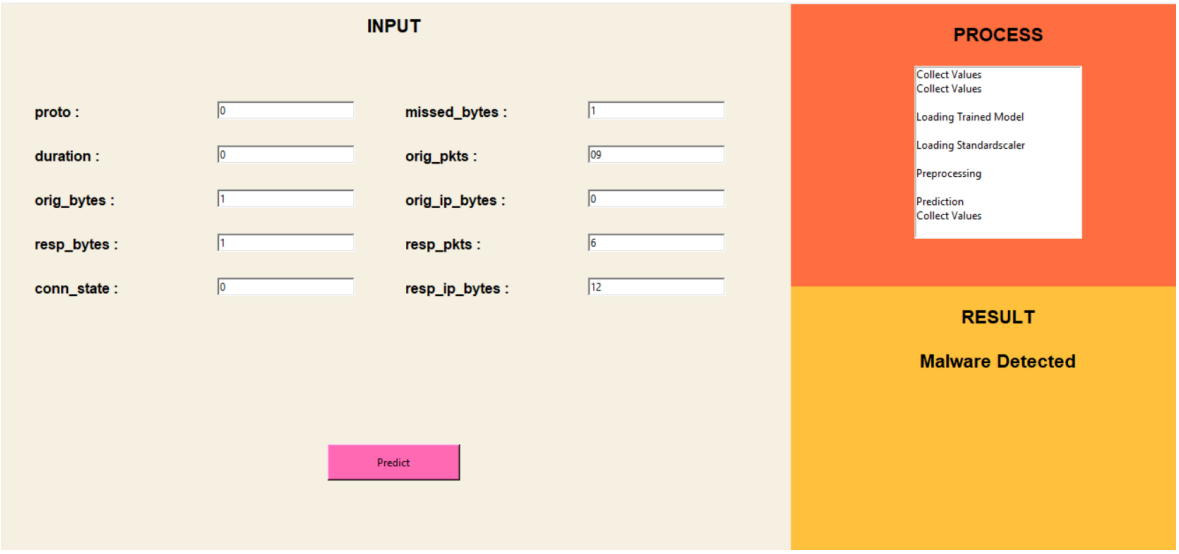

Figure 9 : Predicted Malware in IoT-23

References

The Python Standard Library (no date) *Python documentation*. Available at: https://docs.python.org/3/library/index.html.

Dwivedi, R. (2021) *Complete tutorial on Tkinter to deploy machine learning model*, *Analytics India Magazine*. Available at: https://analyticsindiamag.com/complete-tutorial-on-tkinter-to-deploy-machine-learning-model/.

Erenkervan (2020) *XGBoost classification*, *Kaggle*. Kaggle. Available at: https://www.kaggle.com/code/erenkervan/xgboost-classification.

guest_blog (2020) *XGBoost algorithm: XGBoost in machine learning*, *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/#h2_8.

Madhukar, B. (2021) *Using near-miss algorithm for imbalanced datasets*, *Analytics India Magazine*. Available at: https://analyticsindiamag.com/using-near-miss-algorithm-for-imbalanced-datasets/.