Configuration Manual

MSc Research Project
MSc. Cybersecurity

Laith Abu Saad
Student ID: X21148520

School of Computing
National College of Ireland

Supervisor: Mr. Jawad Salahuddin

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Laith Abu Saad		
Student ID:	X21148520		
Programme:	MSc. Cybersecurity	**Year:**	2022
Module:	Research Project		
Lecturer:	Mr. Jawad Salahuddin		
Submission Due Date:	December 15, 2022		
Project Title:	Prediction of Engagement Levels from Students' Facial Expression.		
Word Count:	……1100	**Page Count:**	……12……..

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: Laith Abu Saad…….

Date: December 15, 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

Laith Abu Saad

X21148520

1 Introduction

This documentation includes information on the hardware and software configurations, steps involved in data collection and pre-process data, and the whole project implementation. The project was aimed at assessing how does Randomforest compared to Logistic Regression in detecting ransomware attacks. Below are the technical requirements and steps which led to the results produced by the project.

2 System Configuration

Figure 1: System Configuration

The system environment used for the execution of this project is a 2GHz Intel Core i7 processor with 16GB of RAM and 500GB of SSD running in Windows 10.

3 Environment Setup

The project's software setup requirements are as follows:

1. Python
2. Jupyter Notebook
3. Anaconda IDE

For this project, Python was chosen as the programming language. Using Jupyter Notebook within Anaconda, all phases of data pre-processing, model training, testing, and evaluation were written in Python.

3.1 Python

The first step is to access the official website[1], download, and install the Python programming language.

Python version	Maintenance status	First released	End of support	Release schedule
3.11	bugfix	2022-10-24	2027-10	PEP 664
3.10	bugfix	2021-10-04	2026-10	PEP 619
3.9	security	2020-10-05	2025-10	PEP 596
3.8	security	2019-10-14	2024-10	PEP 569
3.7	security	2018-06-27	2023-06-27	PEP 537

Figure 2: Python download

3.2 Anaconda Installation

The next step is to download Anaconda from their official website2 because the Jupyter Notebook is already pre-installed within it. You can find information on the minimal system requirements and how to download and install Anaconda here[3].

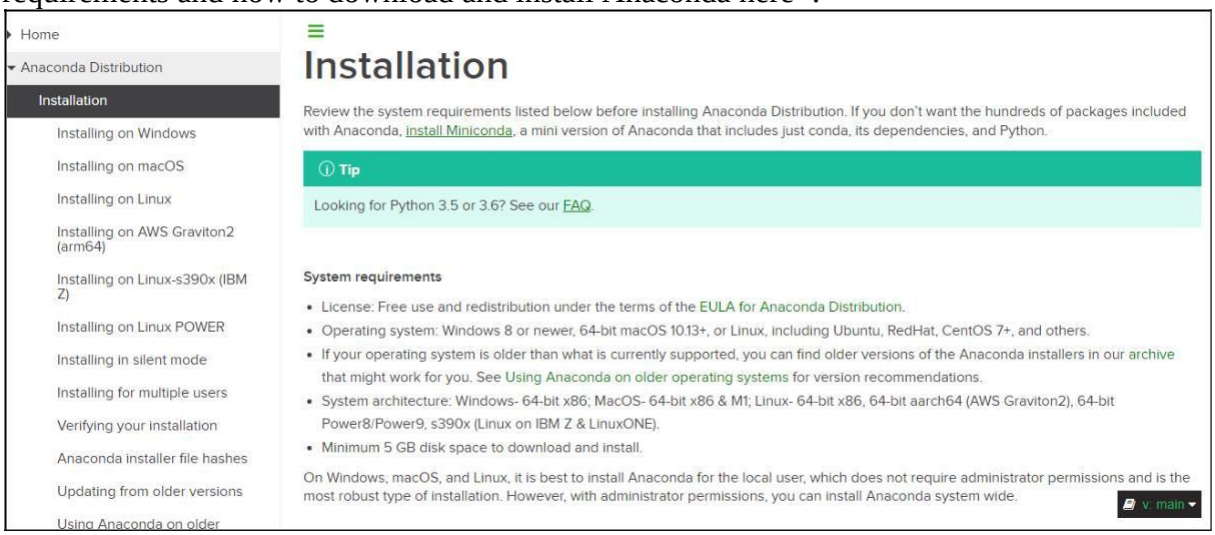

Figure 3: Anaconda Documentation

The Jupyter Notebook may be started from inside this environment once both installs are complete by clicking the Jupyter Notebook icon and the Anaconda Navigator symbol, respectively. The method is demonstrated below.

[1] https://www.python.org/downloads/

[2] https://docs.anaconda.com/anaconda/install/

[3] https://docs.anaconda.com/anaconda/navigator/install/

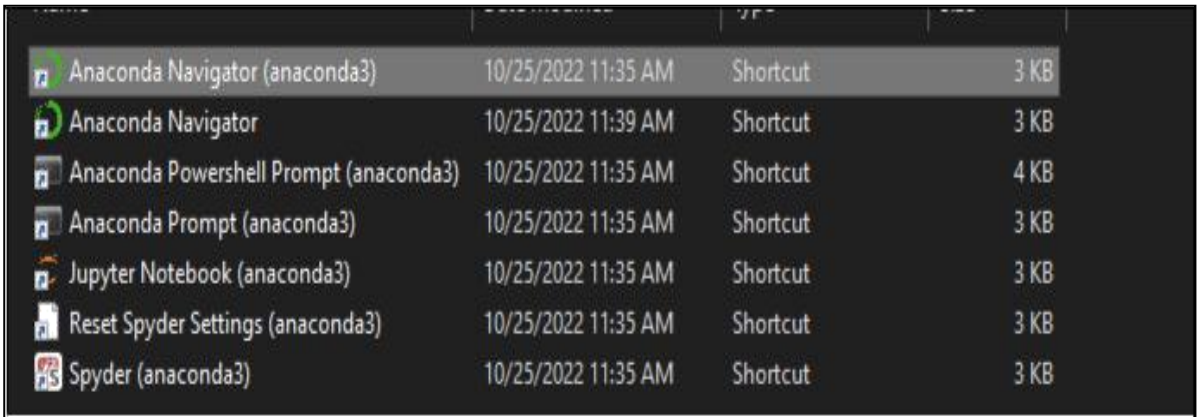

Figure 4: Anaconda Directory

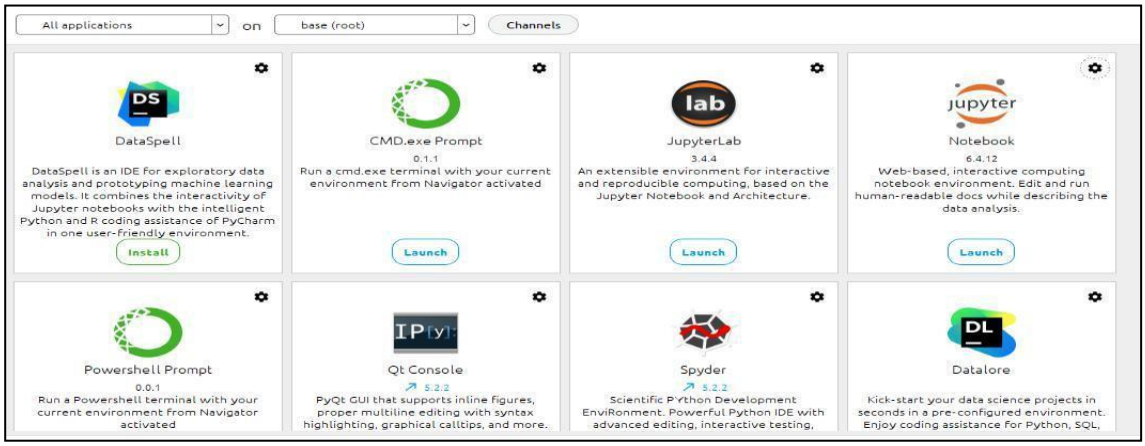

Figure 5: Anaconda Navigator

3.3 Libraries:

- brothon==0.2.5
  - This Library is used to read network files.
- matplotlib==3.6.2
  - This library is used to plot / visualize the matrices and correlations.
- numpy==1.23.5
  - this library is used for for data processing along with pandas.
- pandas==1.5.2
  - this library is used for for data processing along with numpy.
- scikit_learn==1.2.0
  - this library is used to load, train and test models.

- seaborn==0.12.1
    - this library is used for visualization for heatmaps.

4 Data Collection

The dataset for this project, was obtained from Information security and object technology (ISOT) research lab. Dataset was provided by researchers at university of Victoria from their website[4].

5 Pre-processing

It is essential to pre-process the data after downloading it to get it suitable for modeling. Three primary pre-processing procedures were completed. The same Jupyter notebook file, "Ransomware Detection.ipynb," was used for each of these procedures. Importing the required packages to enable code execution is the initial step, as illustrated in Figure[6]. As shown in the example below, any packages that have not yet been installed on the Anaconda environment can be added using the command "!pip install module name" from within Jupyter Notebook:

```
In [1]: #!pip install brothon

In [2]: # Libraries to access directory on local machine
        import shutil
        import os

        # Library to read network files
        from brothon import bro_log_reader as blr

        # Libraries for data processing
        import pandas as pd
        import numpy as np

        # Libraries for visualisation
        import matplotlib.pyplot as plt
        import seaborn as sns

        # Libraries for model training and testing
        from sklearn.linear_model import LogisticRegression
        from sklearn.ensemble import RandomForestClassifier

        # Libraries for model evaluation
        from sklearn.metrics import classification_report
        from sklearn.metrics import confusion_matrix
        from mlxtend.plotting import plot_confusion_matrix
```

Figure 6: Import Modules/ Dependencies

[4] https://www.uvic.ca/ecs/ece/isot/datasets/botnet-ransomware/index.php

Figure 7: library installations

6 Dataset Exploration

As we can see in the figure 8 below there is more than 10 million Rows in the dataset and 22 Columns.

```
#There are 10.4 millions rows in the data set and 22 columns
df.shape
```

```
(10447787, 22)
```

Figure 8: Shape of the Dataset

In Figure 9 we listed the columns names that are in the dataset and there is 22 column.

```
#These are all the names of the columns
df.columns
```

```
Index(['Unnamed: 0', 'ts', 'uid', 'id.orig_h', 'id.orig_p', 'id.resp_h',
       'id.resp_p', 'proto', 'service', 'duration', 'orig_bytes', 'resp_bytes',
       'conn_state', 'local_orig', 'local_resp', 'missed_bytes', 'history',
       'orig_pkts', 'orig_ip_bytes', 'resp_pkts', 'resp_ip_bytes',
       'tunnel_parents   label   detailed-label'],
      dtype='object')
```

Figure 9: Columns in the Dataset

In Figure 10 we can see the number of benign traffic and the number of malicious traffic and all these traffic fall under the column named “tunnel_parents label detailed-label”.

```
#Here we see the number of benign traffic and the number of malicious traffic
df['tunnel_parents   label   detailed-label'].value_counts()
```

```
-   Benign   -                    8262389
-   Malicious   DDoS               2185302
-   Malicious   C&C                    81
-   Malicious   C&C-FileDownload       12
-   Malicious   Attack                  3
Name: tunnel_parents   label   detailed-label, dtype: int64
```

Figure 10: Distribution of the Dataset

In Figure 11 the pie chart presents the percentage of each malicious file and the benign. It shows for benign there is more than 79% and for the Malicious distributed denial of service DDOS more than 20%.

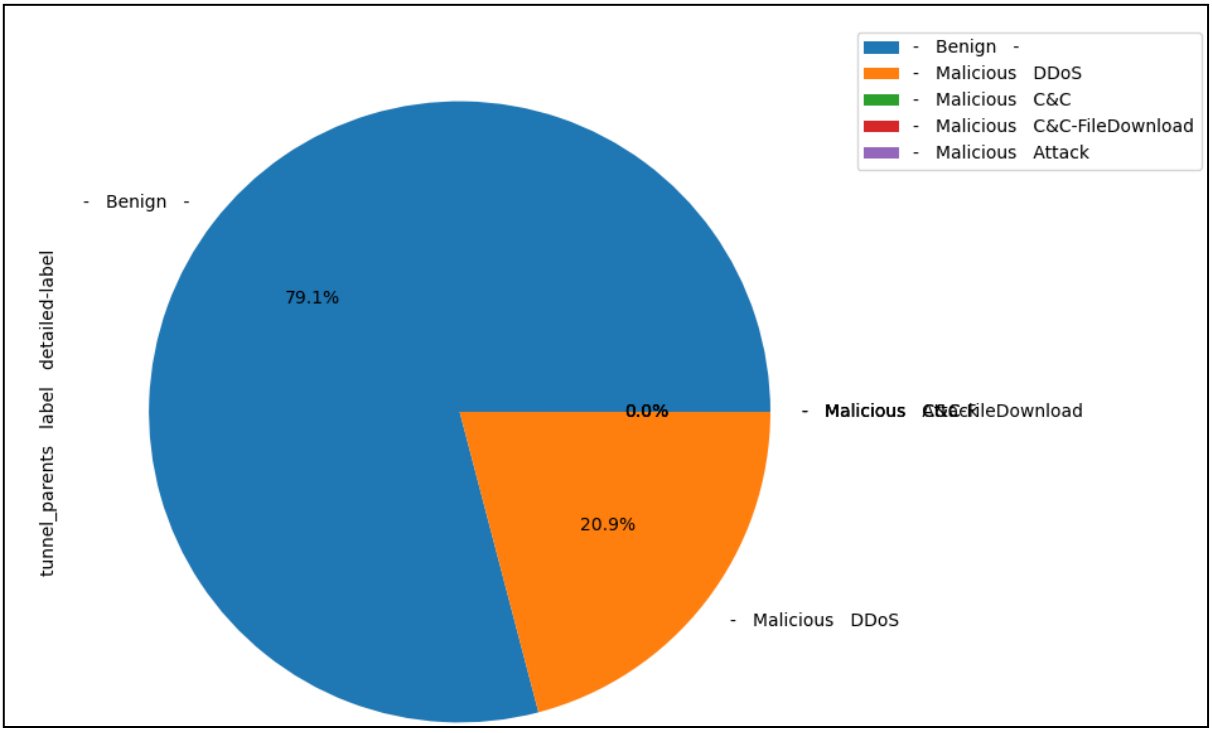

Figure 11: Visualise Distribution of Dataset

```
df.rename(columns={'tunnel_parents   label   detailed-label': 'traffic type'},
          inplace=True,
          errors='raise')
```

Figure 12: Rename the tunnel parents column.

In Figure 12 we renamed the column tunnel parents to traffic type.

```
# Rename traffic type to normalware or ransomeware respectively
df.replace("-   Benign   -", "normalware", inplace=True)
df.replace([
    "-   Malicious   DDoS", '-   Malicious   C&C',
    '-   Malicious   C&C-FileDownload', '-   Malicious   Attack'
],
           'ransomware',
           inplace=True)
```

Figure 13: Rename Traffic Type

In Figure we renamed the traffic type to Normalware and Ransomware instead of Malicious DDos, Malicious C&C and benign.

```
normalware = df[df['traffic type'] == 'normalware'].sample(
    n=1000000,
    random_state=111,
).reset_index(drop=True)
```

```
ransomware = df[df['traffic type'] == 'ransomware'].sample(
    n=1000000,
    random_state=111,
).reset_index(drop=True)
```

```
# Merge the normalware and ransomware dataframe into a single dataframe
final_df = pd.concat([normalware, ransomware])
```

Figure 14: Code for balancing the dataset

In Figure 14 we took a sample size of 1 million of each normalware and ransomware and merged them into the new final_df variable, this will help in achieving higher accuracy rate.

7 Feature Selection

```
def get_low_var_cols(df, thres):
    low_var_cols = []

    for col in df.columns:
        percent_count = df[col].value_counts() / len(df) * 100
        if percent_count.max() > thres:
            low_var_cols.append(col)

    return low_var_cols


variance_thres = 70
low_var_cols = get_low_var_cols(final_df, variance_thres)
print('{} are columns with same data across 70% of the rows'.format(
    str(low_var_cols)[1:-1]))
```

Figure 15: Identifying Coloumns that have similar data

In Figure 15 we set a threshold of 70% which means that we only take data of each column that have 70% or more.

```
from sklearn import preprocessing
```

```
# All the columns are encoded so they can be machine readable
cat_cols = list(data_clean.columns)

enc = preprocessing.LabelEncoder()

for col in cat_cols:
    data_clean[col] = data_clean[col].astype('str')
    data_clean[col] = enc.fit_transform(data_clean[col])
```

Figure 16: Encoding the columns

In Figure 16 we have to encode the data so that it can be machine readable.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.30, random_state = 100)
```

Figure 17: Splitting the Dataset

In Figure 17 we split the dataset 70% for training and 30% for testing.

8 Model Training & Testing

Model Training

```
LR = LogisticRegression(random_state = 100)
LR = LR.fit(X_train, y_train)
LR
```

```
LogisticRegression(random_state=100)
```

Model Test

```
lr_pred = LR.predict(X_test)
lr = LR.score(X_test, y_test)
print('Accuracy score= {:.4f}'.format(lr))
```

```
Accuracy score= 0.7446
```

Figure 18: LR Model training & Testing

In Figure 18 we trained the logistic regression model with 70% of the dataset and for the testing 30% dataset. The model predicted an accuracy score of 74%

Model Training

```
#Define the random forest model
rf = RandomForestClassifier(random_state = 100)

#Train the random forest model using training data
rf = rf.fit(X_train, y_train)
rf
```

```
RandomForestClassifier(random_state=100)
```

Modelling Testing

```
#Test the random forest model using the testing data
rf_pred = rf.predict(X_test)
rf_acc = rf.score(X_test, y_test)
print('Accuracy = {:.6f}%'.format(rf_acc))
```

```
Accuracy = 0.999995%
```

Figure 19: RF Model training & Testing

In Figure 19 we trained the Random Forest model with 70% of the dataset and for the testing 30% dataset. The model predicted an accuracy score of 99%