National College of Ireland

Configuration Manual

MSc Research Project

Gopi Kuchi
Student ID: x22159959

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Gopi Kuchi
Student ID:	X22159959
Programme:	MSc Cyber Security **Year:** 2023-24
Module:	MSc Research Project
Lecturer:	Vikas Sahni
Submission Due Date:	25-042024
Project Title:	Novel ways of detecting threats in IIOT networks

Word Count: 606 **Page Count: 09**

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: Gopi Kuchi

Date: 25-04-2024

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

Gopi Kuchi
Student ID: x22159959

1 Introduction

This study investigates how machine learning can improve cyber security in Industrial Internet of Things networks. The Industrial Internet of Things (IIoT) revolutionizes industrial automation by enabling seamless connectivity and data exchange among devices and systems. However, IIoT networks face cybersecurity challenges such as data breaches, intrusions, and unauthorized access. This manual explores the integration of machine learning to improve cybersecurity in IIoT networks.

2 Requirements

Before proceeding, ensure your system meets the following minimum requirements:
Hardware: Desktop with at least: Dual-core processor
4GB of RAM
10 GB of available storage space
Operating System: Windows 10
Internet Connection: Required for installing libraries and accessing online resources.
Python: Version 3.x installed
Required Libraries: Install the following libraries using the provided command:
!pip install pandas seaborn matplotlib scikit-learn

3 Usage

Jupyter Navigator:

If you're using a Jupyter Notebook, follow these steps to navigate through the notebook:
Open Jupyter Notebook in your preferred web browser.
Navigate to the directory where your notebook is located.
Click on the notebook file (with the .ipynb extension) to open it.
Use the navigation toolbar and cells to interact with the notebook content.

Running the File:

To run the Cryptographic Performance Analysis tool:
Ensure that Python 3.x is installed on your system.
Install the required libraries by running the following commands in your terminal:
For Pandas: pip install pandas
For Seaborn: pip install seaborn

For Matplotlib: pip install matplotlib
For Scikit-learn: pip install scikit-learn
Download the dataset and save it in the appropriate file format and location.
Open the tool file (e.g., .py or .ipynb) using your preferred Python editor or Jupyter Notebook.
Run the .py file to execute the analysis.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import MinMaxScaler
from sklearn.svm import SVC
```

```
# Importing the dataset into a pandas DataFrame
data = pd.read_csv(r"Train_Test_IoT_Modbus.csv")

# Displaying the first few rows of the dataset to verify import
print(Train_Test_IoT_Modbus.head())
```

```
        date        time  FC1_Read_Input_Register  FC2_Read_Discrete_Value  \
0  25-Apr-19   09:14:00                    49389                    52921
1  25-Apr-19   09:14:00                    49389                    52921
2  25-Apr-19   09:14:01                    49389                    52921
3  25-Apr-19   09:14:02                    49389                    52921
4  25-Apr-19   09:14:04                    40665                    44748

   FC3_Read_Holding_Register  FC4_Read_Coil  label       type
0                      25770          13625      1  injection
1                      25770          13625      1  injection
2                      25770          13625      1  injection
3                      25770          13625      1  injection
4                      21098          35371      1  injection
```

Figure 1 Installing the modules and data

```
data.head()
```

	date	time	FC1_Read_Input_Register	FC2_Read_Discrete_Value	FC3_Read_Holding_Register	FC4_Read_Coil	label	type
0	25-Apr-19	09:14:00	49389	52921	25770	13625	1	injection
1	25-Apr-19	09:14:00	49389	52921	25770	13625	1	injection
2	25-Apr-19	09:14:01	49389	52921	25770	13625	1	injection
3	25-Apr-19	09:14:02	49389	52921	25770	13625	1	injection
4	25-Apr-19	09:14:04	40665	44748	21098	35371	1	injection

```
pwd
```

```
'C:\\Users\\Prasad\\Documents'
```

```
missing_values = data.isnull().sum()
print("Missing Values:\n", missing_values)
```

```
Missing Values:
 date                        0
time                        0
FC1_Read_Input_Register     0
FC2_Read_Discrete_Value     0
FC3_Read_Holding_Register   0
FC4_Read_Coil               0
label                       0
type                        0
dtype: int64
```

Figure 2 Checking null values

```
data['datetime'] = pd.to_datetime(data['date'] + ' ' + data['time'])
data.drop(['date', 'time'], axis=1, inplace=True)
```

```
C:\Windows\Temp\ipykernel_17936\1068676715.py:1: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`.
To ensure parsing is consistent and as-expected, please specify a format.
  data['datetime'] = pd.to_datetime(data['date'] + ' ' + data['time'])
```

```
data = pd.get_dummies(data, columns=['type'])
```

```
X = data.drop(['datetime', 'label', 'type_normal', 'type_backdoor', 'type_password', 'type_injection', 'type_xss', 'type_scanning'], axis=1)
y = data['label']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
data[['FC1_Read_Input_Register', 'FC2_Read_Discrete_Value', 'FC3_Read_Holding_Register', 'FC4_Read_Coil']] = scaler.fit_transform(data[['FC1_Read_Input_Registe
```

```
model = RandomForestClassifier()
model.fit(X_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

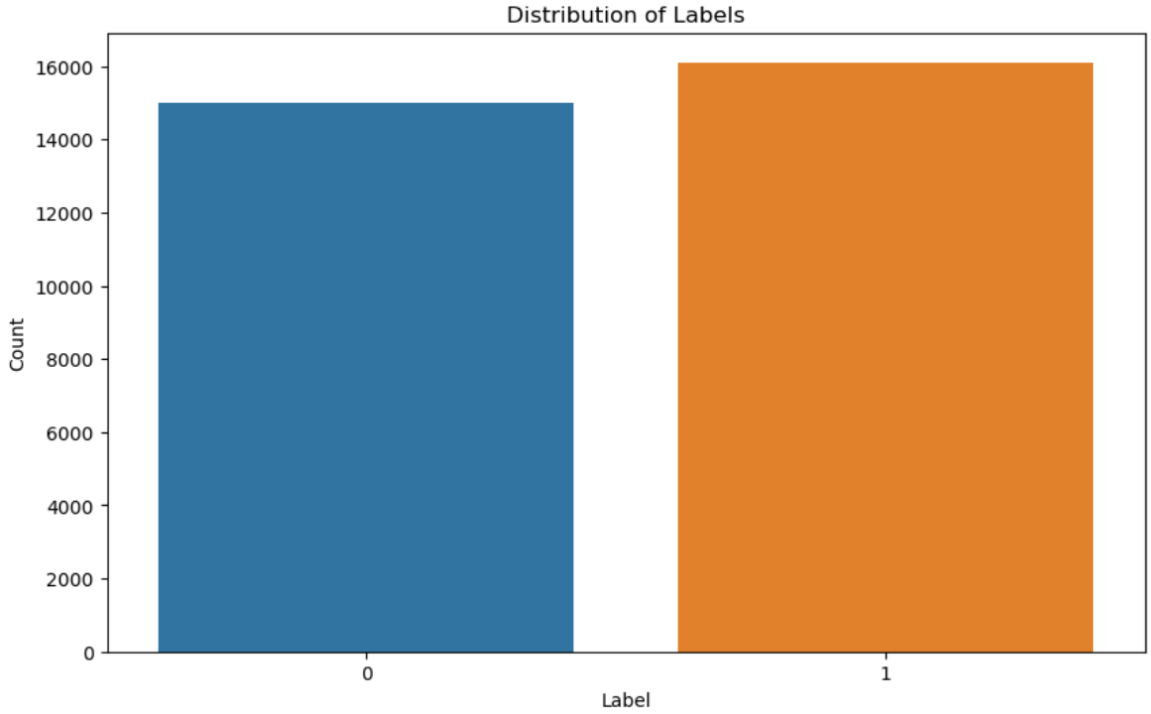

Figure 3 preprocessing the data, splitting and label Visualization

```
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.9898746383799422
Classification Report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      3010
           1       1.00      0.98      0.99      3212

    accuracy                           0.99      6222
   macro avg       0.99      0.99      0.99      6222
weighted avg       0.99      0.99      0.99      6222
```

```
plt.figure(figsize=(10, 6))
sns.countplot(x='label', data=data)
plt.title('Distribution of Labels')
plt.xlabel('Label')
plt.ylabel('Count')
plt.show()
```

Figure 4 Random Forest classifier results

```
# Feature importance
feature_importance = model.feature_importances_

# Create a DataFrame to store feature importance
feature_importance_df = pd.DataFrame({'Feature': X.columns, 'Importance': feature_importance})

# Sort the DataFrame by importance in descending order
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)

# Print feature importance
print("Feature Importance:\n", feature_importance_df)

# Visualize feature importance
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importance_df)
plt.title('Feature Importance')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.show()
```

```
Feature Importance:
                     Feature  Importance
3              FC4_Read_Coil    0.253358
2  FC3_Read_Holding_Register    0.252780
1    FC2_Read_Discrete_Value    0.248035
0    FC1_Read_Input_Register    0.245826
```

```
In [92]: feature_importance_rf = model.feature_importances_
```

```
In [93]: feature_importance_df_rf = pd.DataFrame({'Feature': X.columns, 'Importance': feature_importance_rf})
```

```
In [94]: feature_importance_df_rf = feature_importance_df_rf.sort_values(by='Importance', ascending=False)
```

```
In [95]: print("Random Forest Classifier Feature Importance:\n", feature_importance_df_rf)
```

```
Random Forest Classifier Feature Importance:
                     Feature  Importance
3              FC4_Read_Coil    0.253358
2  FC3_Read_Holding_Register    0.252780
1    FC2_Read_Discrete_Value    0.248035
0    FC1_Read_Input_Register    0.245826
```

```
In [96]: plt.figure(figsize=(10, 6))
         sns.barplot(x='Importance', y='Feature', data=feature_importance_df_rf)
         plt.title('Random Forest Classifier Feature Importance')
         plt.xlabel('Importance')
         plt.ylabel('Feature')
         plt.show()
```

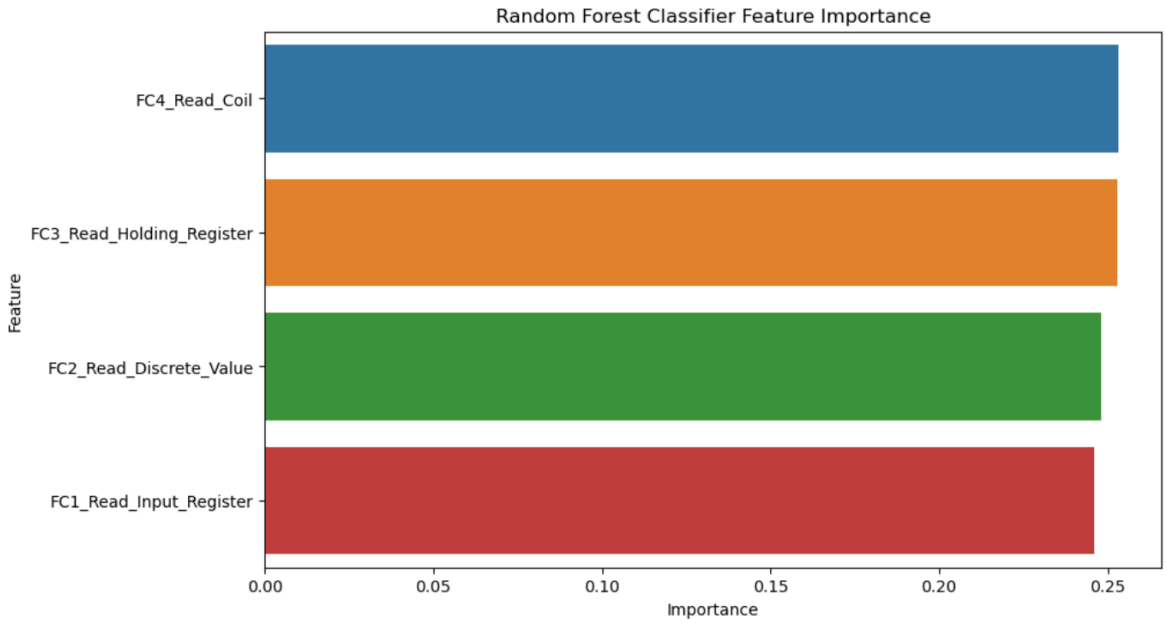

Figure 5 Feature Importance of Random Forest Classifier

```
In [86]: scaler = MinMaxScaler()
         X_train_scaled = scaler.fit_transform(X_train)
         X_test_scaled = scaler.transform(X_test)

In [87]: svm_model = SVC()
         svm_model.fit(X_train_scaled, y_train)

Out[87]: ▾ SVC
         SVC()

In [88]: y_pred_svm = svm_model.predict(X_test_scaled)
         accuracy_svm = accuracy_score(y_test, y_pred_svm)
         print("SVM Classifier Accuracy:", accuracy_svm)
         print("SVM Classifier Classification Report:\n", classification_report(y_test, y_pred_svm))

         # Visualizing distribution of labels

         SVM Classifier Accuracy: 0.5332690453230472
         SVM Classifier Classification Report:
                       precision    recall  f1-score   support

                    0       0.53      0.32      0.40      3010
                    1       0.54      0.73      0.62      3212

             accuracy                           0.53      6222
            macro avg       0.53      0.53      0.51      6222
         weighted avg       0.53      0.53      0.51      6222

In [89]: plt.figure(figsize=(10, 6))
         sns.countplot(x='label', data=data)
         plt.title('Distribution of Labels')
         plt.xlabel('Label')
         plt.ylabel('Count')
         plt.show()
```

Figure 6 Data scaling and SVM results

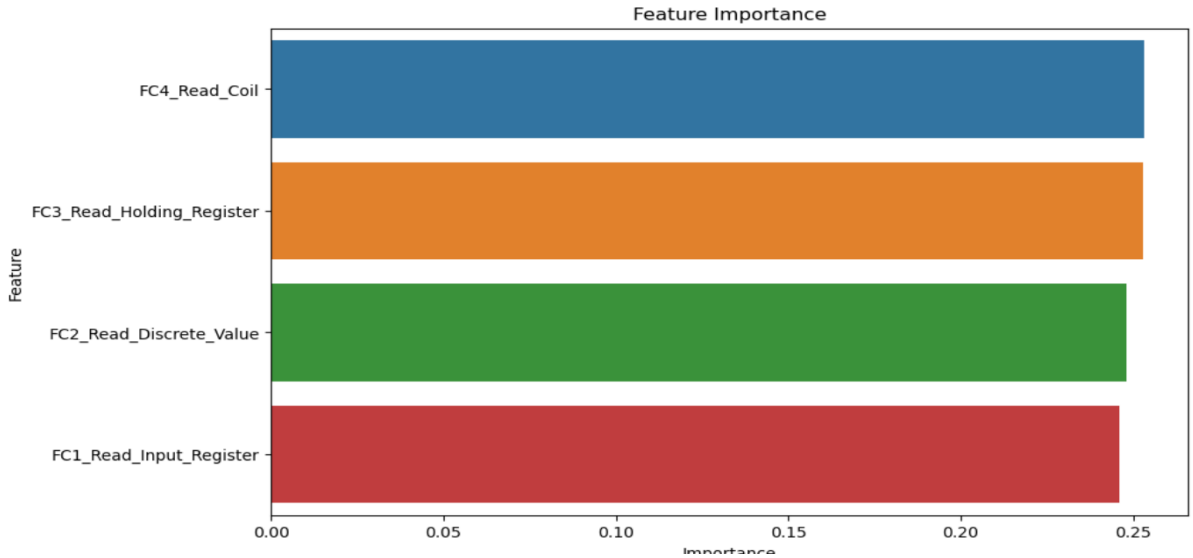

Figure 7 SVM feature Importance

```
In [97]: correlation_matrix = data.corr()

In [98]: plt.figure(figsize=(12, 8))
         sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
         plt.title('Correlation Matrix')
         plt.show()

In [101…: # Calculating correlation of all features with the 'label' column
          label_correlation = data.corr()['label'].sort_values(ascending=False)

          # Displaying correlation of features with the 'label'
          print("Correlation of features with label:")
          print(label_correlation)

Correlation of features with label:
label                        1.000000
datetime                     0.997391
type_password                0.422344
type_injection               0.422344
type_backdoor                0.422344
type_xss                     0.132673
type_scanning                0.126935
FC4_Read_Coil               -0.000025
FC3_Read_Holding_Register   -0.009267
FC1_Read_Input_Register     -0.009848
FC2_Read_Discrete_Value     -0.017784
type_normal                 -1.000000
Name: label, dtype: float64

In [ ]:
```

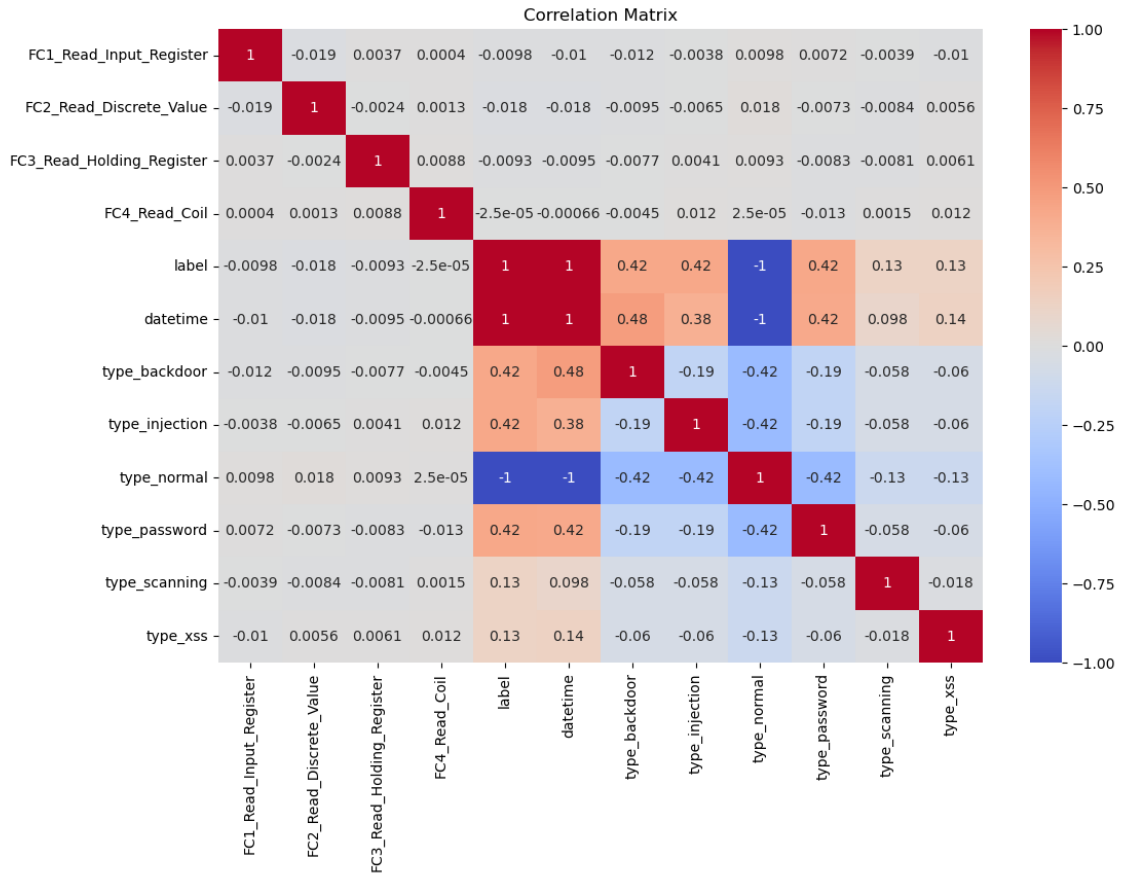

Figure 8 Correlation of features

```
import time

# Start time for RandomForestClassifier
start_time_rf = time.time()

# Train RandomForestClassifier model
model = RandomForestClassifier()
model.fit(X_train, y_train)

# End time for RandomForestClassifier
end_time_rf = time.time()

# Calculate runtime for RandomForestClassifier
runtime_rf = end_time_rf - start_time_rf
print("Runtime for RandomForestClassifier:", runtime_rf, "seconds")

# Start time for SVM Classifier
start_time_svm = time.time()

# Train SVM Classifier model
svm_model = SVC()
svm_model.fit(X_train_scaled, y_train)

# End time for SVM Classifier
end_time_svm = time.time()

# Calculate runtime for SVM Classifier
runtime_svm = end_time_svm - start_time_svm
print("Runtime for SVM Classifier:", runtime_svm, "seconds")
```

```
Runtime for RandomForestClassifier: 18.725443363189697 seconds
Runtime for SVM Classifier: 38.22719597816467 seconds
```

Figure 9 Run Time of the models