NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

VALIDATION OF MACHINE LEARNING ALGORITHM ON THE INTRUSION DETECTION SYSTEM (IDS) OF NAVY SMART GRID

by

Wee San Ng

September 2021

Thesis Advisor:	Preetha Thulasiraman
Second Reader:	Monique P. Fargues

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2021	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE VALIDATION OF MACHINE LEARNING ALGORITHM ON THE INTRUSION DETECTION SYSTEM (IDS) OF NAVY SMART GRID	**5. FUNDING NUMBERS** RMQ80
6. AUTHOR(S) Wee San Ng	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) ONR-ESTEP, San Diego, CA	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

The U.S. Navy currently deploys a centralized smart grid that consists of several control systems to analyze energy consumption and utilize data to drive more efficient operations. Being interconnected as a system-of-systems through the internet interface, the smart grid faces threats from the cyber realm aimed at disruption, intelligence gathering and destruction. In this thesis, an intrusion detection system (IDS) is developed for the smart grid as the first line of defense to guard against cyber attacks. We study the architecture of that Navy Smart Grid that utilizes Supervisory Control and Data Acquisition (SCADA) for control and monitor operations. We build the IDS using a random forest machine learning algorithm that can classify and identify malicious traffic. We then compare our approach to two machine learning benchmarks, namely the k-nearest neighbor and Bayesian learning models. We train our algorithm on an open-source SCADA data set that closely aligns with the type of traffic the smart grid would transmit. Simulations run on MATLAB show the efficacy of each of the algorithms and its ability to accurately classify different network threats.

14. SUBJECT TERMS smart grids, intrusion detection system, machine learning algorithm, NAVFAC dataset	**15. NUMBER OF PAGES** 75
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

VALIDATION OF MACHINE LEARNING ALGORITHM ON THE INTRUSION DETECTION SYSTEM (IDS) OF NAVY SMART GRID

Wee San Ng
Military Expert 5, Republic of Singapore Air Force
BEE, Nanyang Technological University, 2010

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2021

Approved by: Preetha Thulasiraman
Advisor

Monique P. Fargues
Second Reader

Douglas J. Fouts
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The U.S. Navy currently deploys a centralized smart grid that consists of several control systems to analyze energy consumption and utilize data to drive more efficient operations. Being interconnected as a system-of-systems through the internet interface, the smart grid faces threats from the cyber realm aimed at disruption, intelligence gathering and destruction. In this thesis, an intrusion detection system (IDS) is developed for the smart grid as the first line of defense to guard against cyber attacks. We study the architecture of that Navy Smart Grid that utilizes Supervisory Control and Data Acquisition (SCADA) for control and monitor operations. We build the IDS using a random forest machine learning algorithm that can classify and identify malicious traffic. We then compare our approach to two machine learning benchmarks, namely the k-nearest neighbor and Bayesian learning models. We train our algorithm on an open-source SCADA data set that closely aligns with the type of traffic the smart grid would transmit. Simulations run on MATLAB show the efficacy of each of the algorithms and its ability to accurately classify different network threats.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AMI	Advanced Metering Infrastructure
C2	Command and Control
DCS	Distributed Control Systems
DAG	Directed Acyclic Graph
DoE	Department of Energy
DoS	Denial-of-Service
FNR	False Negative Rate
FOC	Full Operational Capability
HMI	Human-Man-Interface
ICS	Industrial Control System
IDS	Intrusion Detection System
IT	Information Technology
KNN	K-Nearest Neighbor
MCC	Matthew Correlation Coefficient
MTU	Master Terminal Unit
MOSAICS	More Situational Awareness for Industrial Control Systems
NAVFAC	U.S. Naval Facilities Engineering Systems Command
NIST	National Institute of Standards and Technology
NextSTEP	Next Systems Technology Evaluation Program
PCA	Principal Component Analysis
PLC	Programmable Logic Controller
RTU	Remote Terminal Unit
SCADA	Supervisory Control and Data Acquisition
TPR	True Positive Rate
UDP	User Datagram Protocol
WAN	Wide Area Network

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First, I would like to thank my wife for taking care of our daughters in Singapore while allowing me to focus on my master's degree at Naval Postgraduate School. Next, I would like to thank my superiors, past and present, in Republic of Singapore Air Force for giving me the opportunity to further my studies. I would also like to thank my advisor, Professor Preetha Thulasiraman, for her unwavering support and guidance along the way.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. WHAT IS A SMART GRID?

A commercial smart grid uses an intelligent communication network to control the distribution of electricity on the traditional power grid. The integration of Information Technology (IT) devices with the power grid enhances the reliability and efficiency of the grid through two-way communication. The smart grid is considered the future of energy distribution.

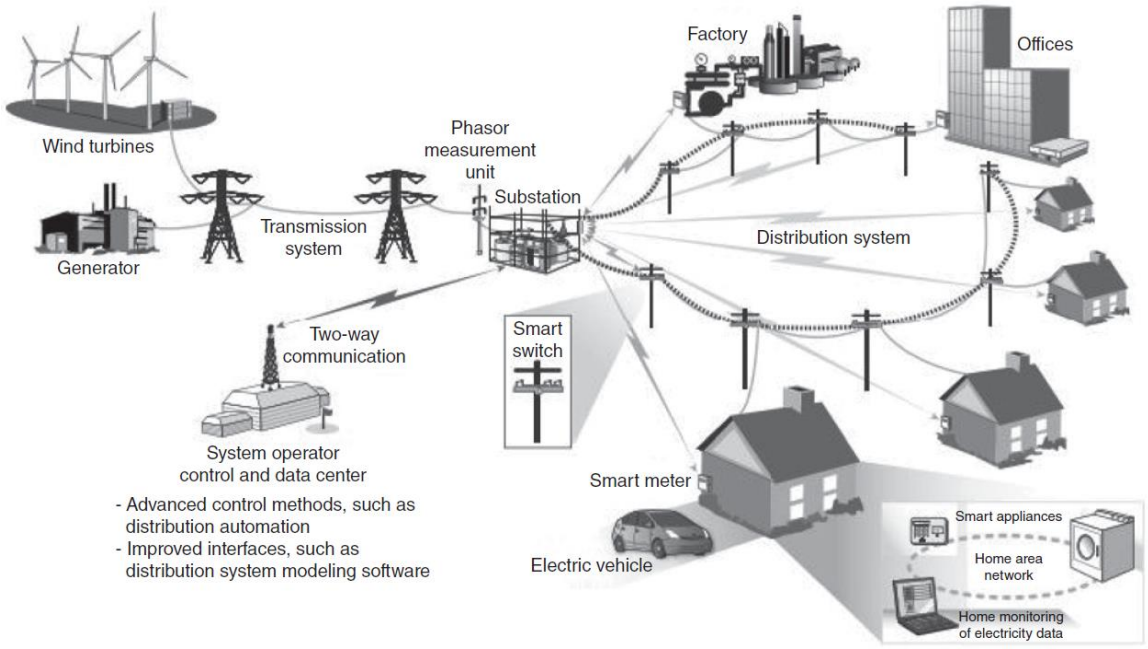

Figure 1. General Architecture of a Smart Grid Design. Source: [1].

The traditional power grid was built decades ago to serve a smaller community with a lower power requirement. Current power consumption in a typical household includes technology that requires large amounts of power such as charging of electrical vehicles. In Figure 1, the general architecture of the commercial smart grid is shown.

The advantages of the smart grid as summarized by the U.S. Department of Energy are as follow:

- More efficient transmission of electricity.
- Quicker restoration of electricity after power disturbances.
- Reduced operations and management costs for utilities, and ultimately lower power costs for consumers.
- Reduced peak demand, which will also help lower electricity rates.
- Increased integration of large-scale renewable energy systems.
- Better integration of customer-owner power generation systems, including renewable energy systems.
- Improved security. [2]

At the heart of the smart grid is the Advanced Metering Infrastructure (AMI). The AMI is a two-way transmission system that combines smart meters and communication networks. It enables communication between utility companies and consumers. A smart meter is an electronic device that collects data on energy consumption, including voltage levels, current and other power factors.

B. NAVY SMART GRID

In 2013, the U.S. Naval Facilities Engineering Systems Command (NAVFAC) developed the operational requirements to implement its own smart grid [3]. In 2019, NAVFAC deployed the Navy Smart Grid enterprise energy management solution. It was highlighted in the press release that "with Smart Grid, an operator in a central command location can monitor energy data in near-real time, deploying technicians to efficiently manage emergencies, outages, and repairs" [4]. NAVFAC has achieved Full Operational Capability (FOC) of the smart grid at several locations in the mid-Atlantic states.

Like the commercial smart grid, the Navy smart grid relies on the AMI and its smart meters to monitor and control energy consumption remotely. Information flows from smart meters to the Command and Control (C2) center are transmitted via a Wide Area Network (WAN). This results in a reduction of operational and maintenance costs.

C. SMART GRID VULNERABILITIES

Although the smart grid has benefitted both the suppliers and consumers, the introduction of network communication has created weaknesses in the system. The addition of network communication, wireless or wired, exposes the smart grid to a series of cyberattacks with several entry points [5].

One such prominent cyberattack is Stuxnet. Stuxnet [6] is a large, complex piece of malware that was written to target an industrial control system (ICS) or similar systems such as the smart grid. In this era of post-Stuxnet, systems such as the smart grid are not able to be air-gapped and secured fully. Furthermore, Raj Samani from Cloud Security Alliance [7] stated that a group of academics discovered various code vulnerabilities on smart meter platforms, allowing them to take entire control of the smart meter system. This could result in a power spike or a partial or complete smart grid outage.

D. MOTIVATIONS AND CONTRIBUTIONS

For the Navy to ensure resiliency of the smart grid network, effective countermeasures against security threats must be addressed using techniques that adapt to the massive data that is collected by smart sensors and meters. The goal of our research is to develop cost-effective, cyber threat detection techniques to secure energy infrastructure that is critical to the Navy. Our research is focused on the use of cyber analytics to continually define and mitigate evolving threat vectors for the Navy Smart Grid. Traditionally, cyber security is managed with reactive solutions that respond to incidents once they have occurred. Cyber analytics is a proactive method to cyber protection that anticipates future attack strategies and incorporates information into the real-time response management of ongoing attacks. Machine learning and statistical data analysis are key to developing cyber analytic tools that will allow the accurate classification/identification of adversarial and malicious network activity.

In recent years, NAVFAC has invested in threat detection methodologies, including in the Department of Energy (DoE) More Situational Awareness for Industrial Control Systems (MOSAICS) project. While the objective of MOSAICS is attack mitigation on general critical infrastructure control systems within DOD, the research presented in this thesis is focused on security of the network communications of the Navy smart grid using machine learning processes. The research in this thesis contributes to a project funded by the Office of Naval Research called Next Systems Technology Evaluation Program (NextSTEP) which is studying the security of the Navy smart grid.

The work presented in this thesis is built upon two previous research efforts by former NPS MSEE students: First, Vincent Chan proposed a K-Nearest Neighbor (KNN) supervised machine learning algorithm to develop a simple intrusion detection system (IDS) for the Navy smart grid. He trained his algorithm with the CICIDS2017 dataset [8]. Second, Carolyn Schiesser proposed a Bayesian classification algorithm for network threat detection. She trained her algorithm with Lemay and Fernandez's dataset from the University of Montreal [9].

In this thesis, we improve upon the research by these authors by increasing smart grid threat classification accuracy via the use of Random Forest machine learning algorithm. We use Lemay and Fernandez's dataset [10] to train our algorithm and compare against the KNN and Bayesian approaches.

The contributions of this thesis are:

1. Design of an IDS architecture for the Navy smart grid. The IDS is comprised of the following components: Network Data Collection, Flow Generation, Feature Construction, Classification Training, and Detection through Machine Learning.
2. Evaluation and comparison of KNN, Bayesian and Random Forest algorithm on Lemay and Fernandez's datasets.
3. Validation via MATLAB that the Random Forest algorithm is a more efficient approach based on performances shown on true positive rates, Matthew Correlation Coefficient (MCC) values and overall accuracy.

E. THESIS ORGANIZATION

The remainder of this thesis is organized as follows. In Chapter II, we provide a thorough literature review on the dataset and machine learning algorithm to be implemented on the IDS. In Chapter III, we cover the theory behind the various machine learning algorithms used in this thesis. In Chapter IV, we propose the IDS architecture and discuss in detail the chosen dataset and Random Forest algorithm. In Chapter V, we describe and analyze our results. Lastly, Chapter VI provides the conclusion and recommendations for future work.

II. RELATED WORK

A. SCADA AND ITS RELATION TO THE NAVY SMART GRID

The definition of an ICS, as described by the National Institute of Standards and Technology (NIST), is an overall term that incorporates different control systems, such as Supervisory Control and Data Acquisition (SCADA) systems, distributed control systems (DCS), and other control system configurations [5].

To understand the different layers of automation that can exist in an ICS, the automation pyramid was developed to guide companies in managing their system-of-systems. An automation pyramid classifies the different IT layers of automated productions plants. An example of an automation pyramid is shown in Figure 2.

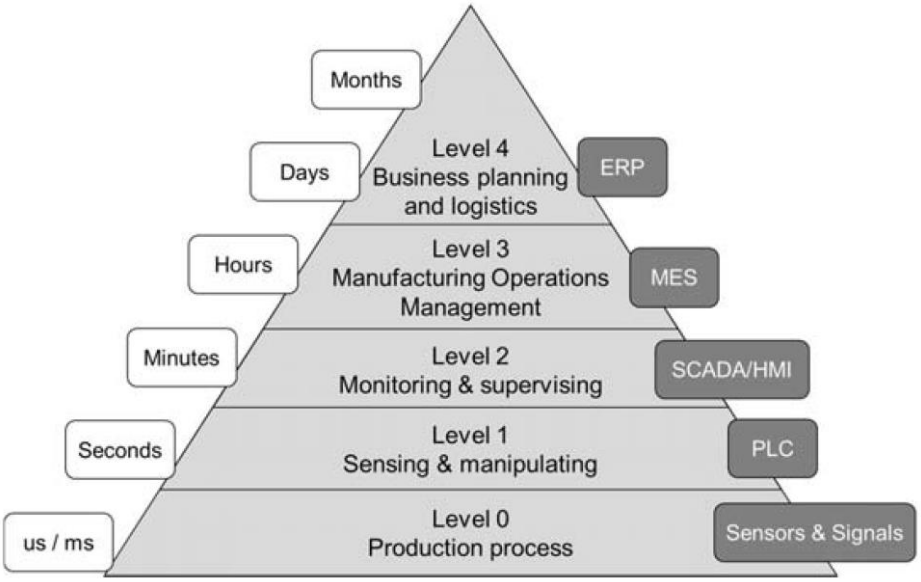

Figure 2. Automation Pyramid. Source: [11].

The automation pyramid categorizes each level in accordance with their individual purpose or function. SCADA systems are commonly located at level two, also known as the middle layer. SCADA allows electronic automation components to communicate even

if geographically dispersed. A basic SCADA system consists of a control center and a few field sites that are connected via a Wide Area Network (WAN).

Communications between the devices usually occurs through constant polling and by the use of network protocols such as ModBus. The control server records information gathered from the field sites and thereafter sends information to the Human-Man-Interface (HMI) display for the operator to analyze. The control center carries out integrated alarming, trending evaluation and reporting. The field site, adopted from level zero and level one of the Automation Pyramid, plays the role of controlling actuators and monitoring sensors. Figure 3 depicts a general SCADA architecture.

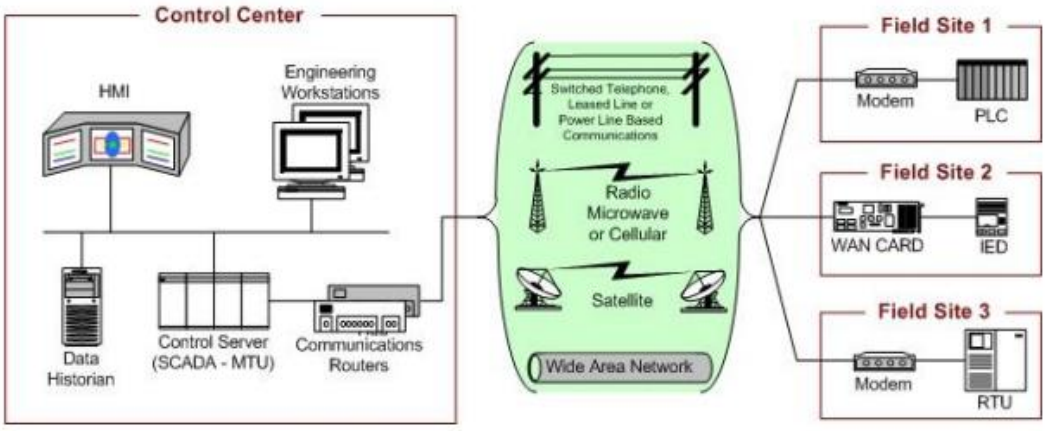

Figure 3. General SCADA Architecture. Source: [12].

SCADA is a central component of the Navy smart grid. An overview of the Navy smart grid is shown in Figure 4. Comparing Figure 3 and Figure 4, we can draw similarity in terms of the system architecture. Both architectures are composed of smart meters and controllers that are constantly communicating with the control server. The blue lines shown in Figure 4 reflect the communication path between the smart meters and controllers with the control server over the WAN.

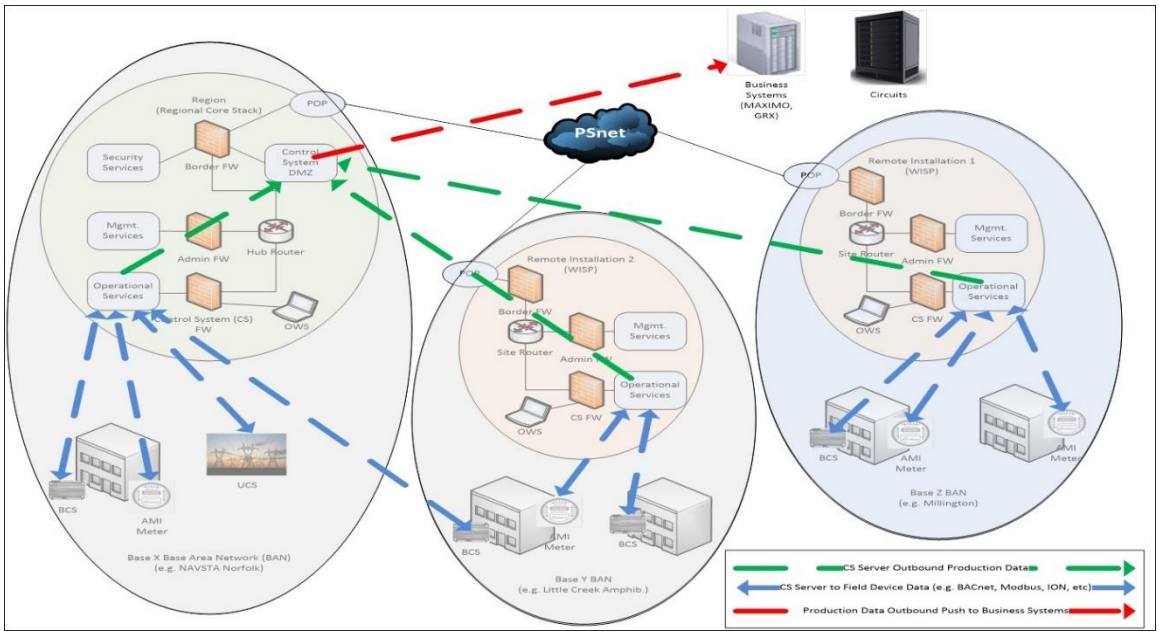

Figure 4. U.S. Navy Smart Grid Architecture. Source: [9].

B. INTRODUCTION TO MODBUS

As mentioned in the previous section, the U.S. Navy smart grid uses Modbus to communicate between servers and Programmable Logic Controllers (PLC). Previously known as Modicon and created in 1979 by Schneider-Electric, Modbus is a serial communication protocol between Remote Terminal Units (RTU) and PLCs [13]. Since then, many industrial companies have used this communication protocol as the default [14]. Modbus has many different versions. The commonly used ones are listed in Table 1.

Table 1. Different Versions of Modbus. Source: [15].

Version	Description
Modbus RTU	Serial communication via *RS-232* connector to connect PLCs with RTUs
Modbus ASCII	Same connector as above, but instead of binary coding, ASCII-encoded characters are used
Modbus TCP/IP	Communication based on the TCP/IP protocol stack
Modbus over TCP/IP	Same as above, but including a checksum in the payload, in addition to error correction mechanisms provided by layers 1 to 4 of the OSI model

In Modbus/TCP, the data is encapsulated in a TCP/IP packet and transmitted within the Ethernet frame. A majority of the Modbus/TCP messages include commands on reading and writing registers and coils. In a traditional control system, coils are referred to as one-bit registers while multi-bit registers are named as registers. In a default Modbus serial communication, the master controls the Modbus data transactions with multiple slaves that respond to the requests of the master to read from or write data to the slaves [16]. The significant difference between a default Modbus communication to Modbus/TCP is that Modbus/TCP uses a client/server architecture through Ethernet. A typical Modbus TCP architecture that is comprised of serial Modbus communication and Modbus/TCP communication is shown in Figure 5.

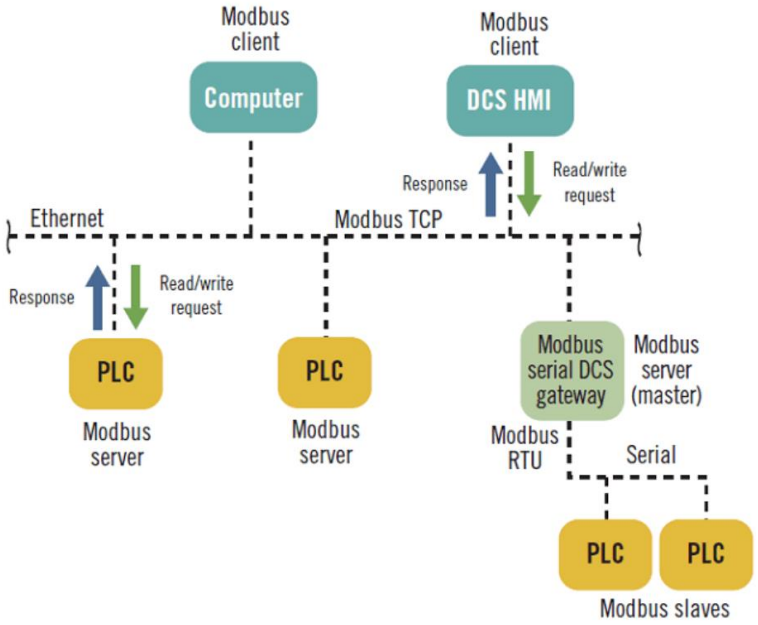

Figure 5. Typical Modbus/TCP Architecture. Source: [16].

C. RESEARCH ON DATASET

For optimum and accurate analysis, a dataset generated from the Navy smart grid would be of significant benefit. However, due to the classified nature of the Navy smart grid dataset, we were unable to utilize it for this work. Therefore, we researched suitable SCADA datasets that were available as open-source documents.

Teixeira et al. developed a SCADA system testbed that consists of a water storage tank control system for water treatment and distribution [17]. The dataset was collected by software such as Wireshark to capture the PCAP file. The authors also carried out five different type of reconnaissance attacks on their testbed. However, the datasets lack a variety of different type of attacks as only reconnaissance attacks are present in the datasets.

Rosa et al. developed a SCADA system testbed in a hybrid environment which consists of real network communication and SCADA assets that emulated a power grid [18]. By assessment, this dataset consists of the most comprehensive types of attack. Furthermore, it is based on the power grid testbed architecture, which is very similar to the Navy smart grid. However, this dataset is not shared in the open-source internet for analysis.

In this thesis, we use Lemay and Fernandez's data set developed at the University of Montreal [10]. Lemay et al. produced a dataset based on the construction of a traffic simulation environment where Modbus/TCP tools and a sandbox are used to introduce realism into the physical components. The similarities between the Navy smart grid and Lemay's sandbox are due to their origin in the automation pyramid shown in Figure 2 and the Modbus protocol used between the layer 2 and layer 1 components. A wide variety of attacks are carried out on the testbed. This data set is comprehensive and provides enough similarity to the Navy smart grid for use in this thesis. We provide further detail on the data set in Chapter IV.

D. RESEARCH ON MACHINE LEARNING ALGORITHMS

Teixeira et al. highlighted that there are a few malicious attacks which are harder to detect [17]. The authors added that some of the attacks can be used to map the network, making the attack traffic look like regular behavior. Therefore, the use of rule-based mechanisms such as anti-virus software will not be successful in detecting these attacks as the signature of the Modbus network remains unchanged. Therefore, the authors suggested the application of machine learning algorithms to identify attacks that can be easily hidden when using conventional approaches.

In this thesis, we chose to use the Random Forest machine learning algorithm. Angelo and Drummond highlighted a few advantages of Random Forest over other machine learning algorithms when it comes to implementing an IDS. They are

1. Low training time complexity, $O(nlog(n))$ and fast prediction.
2. Resilience to deal with imbalanced datasets.
3. Embedded feature selection method and intrinsic metrics to rank features by importance.
4. Able to deal natively with categorical and continuous features. [19]

Chauhan et al. [20] conducted experiments to find the best performing machine learning classification technique that is most suitable for IDS. They used the NSL-KDD dataset. A total of ten machine learning algorithms were tested and evaluated. The results showed that decision tree classifiers tend to perform better than others at classifying

network intrusions. Amongst the list of decision tree classifiers, Random Forest outperformed the rest with respect to accuracy, specificity and sensitivity.

Belavagi et al. [21] evaluated four supervised machine learning classifiers for an IDS. Similarly, the authors used the NSL-KDD dataset for their research. The four classifiers are namely: Support Vector Machine, Random Forest, Logistic Regression and Gaussian Naïve Bayes. Standard performance-based indicators such as precision, recall, F1-Score and accuracy were compared among the four algorithms. With a 99% accuracy, the authors concluded that the Random Forest classifier surpasses the other classifiers.

Ashraf et al. [22] experimented with three supervised machine learning algorithms and used the NSL-KDD dataset on their proposed IDS. Naïve Bayes, J48 and Random Forest machine learning algorithms were used. Outputs such as precision, recall and F-measure were recorded and compared amongst the three algorithms. In terms of both accuracy and detection rate, the authors concluded that Random Forest outperformed Naïve Bayes and J48.

THIS PAGE INTENTIONALLY LEFT BLANK

III. MACHINE LEARNING ALGORITHMS

There are two types of machine learning algorithms: supervised learning and unsupervised learning. We provide a short summary of both groups in this section. We go through the supervised learning algorithms KNN, Bayesian, and Random Forest in depth because they are the foundation for the research given in this thesis. Figure 6 shows categorization of supervised and unsupervised algorithms.

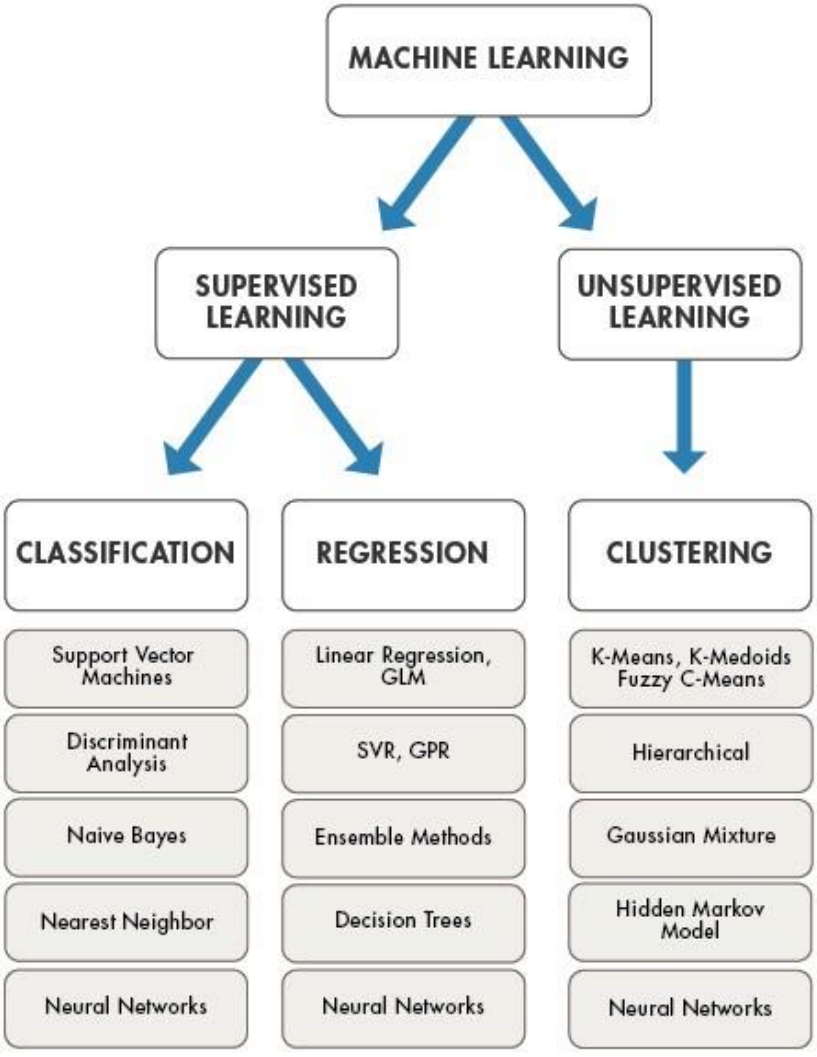

Figure 6. Supervised Learning versus Unsupervised Learning. Source: [23].

A. SUPERVISED LEARNING VS UNSUPERVISED LEARNING

1. Supervised Learning

In supervised learning, the algorithm requires a dataset that needs to be trained on labelled data that consists of normal traffic and anomalous traffic. This dataset usually requires human intervention to determine the labelling of the data. Depending on how the datasets are generated, the dataset can consist of all anomaly scenarios combined as a single or different class. For example, in IDS, the anomaly data can be either labeled as "malicious" or according to the respective type of attacks such as Denial-of-Service (DOS). Next, the training and testing datasets must be chosen for cross validation. In this method, the training dataset will be modeled and thereafter evaluation on the model is carried out on the testing dataset. In supervised learning, there are two techniques currently being modeled. They are:

- Classification: This technique predicts categorical responses. For example, whether the network data from a certain IP address is malicious or normal. Classification models categorize response data into categories. Algorithms such as KNN and Naïve Bayes are classification techniques.

- Regression. This technique predicts continuous responses. For example, whether the traffic data is transmitting in small packets or large packets over a period of time. In this way, the regression models can classify large packets of certain range to be malicious. On such example of regression technique is Ensemble Method.

2. Unsupervised Learning

In unsupervised learning, the algorithm does not require labelled data, and is therefore applicable to most applications. This method discovers hidden patterns or data groupings without the need for human intervention [24]. In the dataset required for unsupervised learning, normal data instances will be far more common than anomalous ones. Otherwise, the algorithm will face a high false alarm rate. Compared to supervised learning, there are three techniques used in unsupervised learning: clustering, association,

and dimensionality reduction. Of the three mentioned, clustering technique is the most commonly used by researchers.

B. KNN

As discussed in Chapter I, we use the KNN algorithm as one of our benchmarks to test against our Random Forest based IDS. The KNN algorithm, developed by Thomas Cover et al. in 1966, is one of the simplest and oldest algorithms for pattern categorization [25]. The rule categorizes every unlabeled data in the training set according to the majority label among k-nearest neighbors. The KNN performance usually hinge on the distance metric, "k" in identifying its nearest neighbors. Without any knowledge on the distribution, KNN uses Euclidean metric to quantify the differences between each sampled data. Euclidean distance is described as follows:

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} w_r(a_r(x_i) - a_r(x_j))^2} \tag{4.1}$$

where we define vector $x = (a_1, a_2, a_3, \dots, a_n)$, where n denotes the vector input's dimensionality, or the number of sample characteristics. a_r is the example's rth characteristics while w_r is the weight of the rth characteristics. r ranges from *1* to n [26]. This means that the smaller the Euclidean distance, $d(x_i, x_j)$ between any two examples, the similarity between them rises.

A test example's class label is determined by a majority poll of its k nearest neighbors as shown below in equation 4.2 [26].

$$y(d_i) = \arg\max_{k} \sum_{x_j \in kNN} y(x_j, c_k) \tag{4.2}$$

In the above formula, d_i is a test example while x_j refers to its k nearest neighbors based on a training dataset, and$y(x_j, c_k)$ signifies if $x_j$belongs to class c_k. Equation 4.2 estimates the class having many of the members in the k nearest neighbors. Figure 7

illustrates the KNN methodology. From Figure 7, we can see that if k = 3 (from the inner circumference), it will be assigned with class B, respectively, if k = 6 (from the outer circumference), it will be assigned with class A.

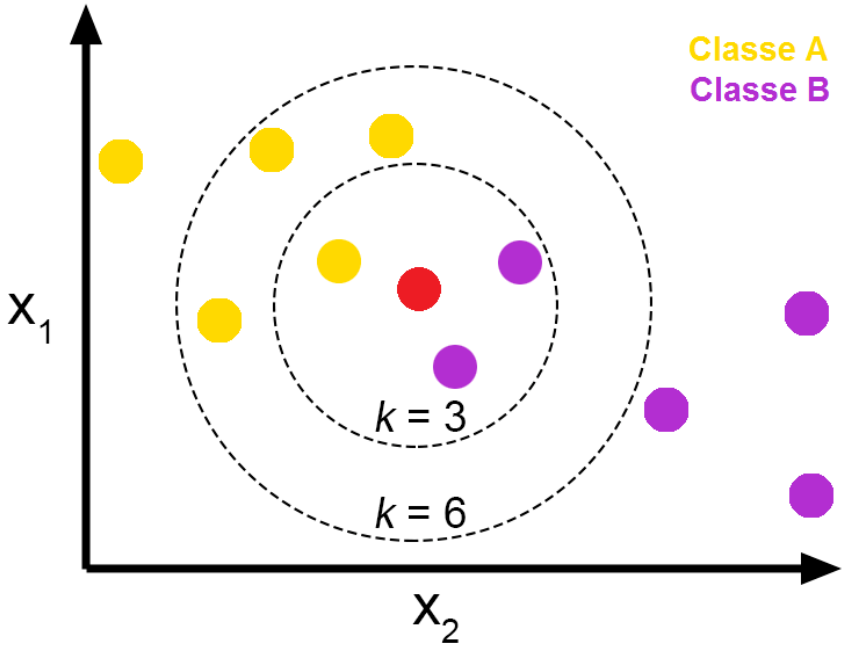

Figure 7. Illustration of the KNN Methodology. Source: [27].

While KNN is widely used due to its simplicity and its ease of application, the algorithm may sometimes fail to achieve good accuracy due to the uneven allocation of the examples among classes. Trial and error on getting the best k value are required to achieve the best result. This is not ideal as such a method is time consuming to train. More so, large datasets that are generated from a SCADA network will add significantly to the training time.

C. BAYESIAN

The second benchmark used in this thesis is the Bayesian learning method which was used as part of a former thesis [9]. The Bayesian algorithm is another classification technique that is widely used. The Bayesian algorithm is built upon two components: “A graphical component, also known as a directed acyclic graph (DAG), where the vertices

represent events, and the edges are relations between events, and a numerical component that quantifies different links in the DAG by a conditional probability distribution of each parent node" [28].

Naïve Bayes comprises DAGs of a parent node with several child nodes where the child nodes are independent of one another. The categorization is guaranteed by deliberating the parent node as a concealed parameter that specifies which class each item in the dataset should be allocated to, and the child nodes as various properties that define this object [28]. Naïve Bayes was chosen in [9] due to the same advantages it had as compared to KNN. Furthermore, Naïve Bayes is more effective than KNN because it can determine the likelihood of a future occurrence centered on the knowledge of prior conditional and marginal probabilities [9]. The Bayesian rule is expressed as such:

$$P(s_i \mid E) = \frac{P(E \mid s_i) \cdot P(s_i)}{P(E)} \tag{4.3}$$

where in the session class, E denotes the sum of evidence on attribute nodes and s_i represent a potential value. The sum of evidence E are dispersed into individual parts., say e_1, e_2, ..., e_n with relation to E_1, E_2, ..., E_n, respectively [30]. Since these attributes are independent of one another, the combined probability is achieved as such in Equation 4.4.

$$P(s_i \mid E) = \frac{P(e_1 \mid s_i) \cdot P(e_2 \mid s_i) \cdot \ldots \cdot P(e_n \mid s_i) \cdot P(s_i)}{P(E)} \tag{4.4}$$

D. RANDOM FOREST

Random Forest is an "ensemble of unpruned classification or regression trees usually trained with the 'bagging' method" [29]. The key features of Random Forest are mentioned in Chapter II. In particular, the main benefit of using this method as compared to the previous two algorithms is its versatility in utilizing both regression and classification techniques. Naïve Bayes and KNN can only be used for classification.

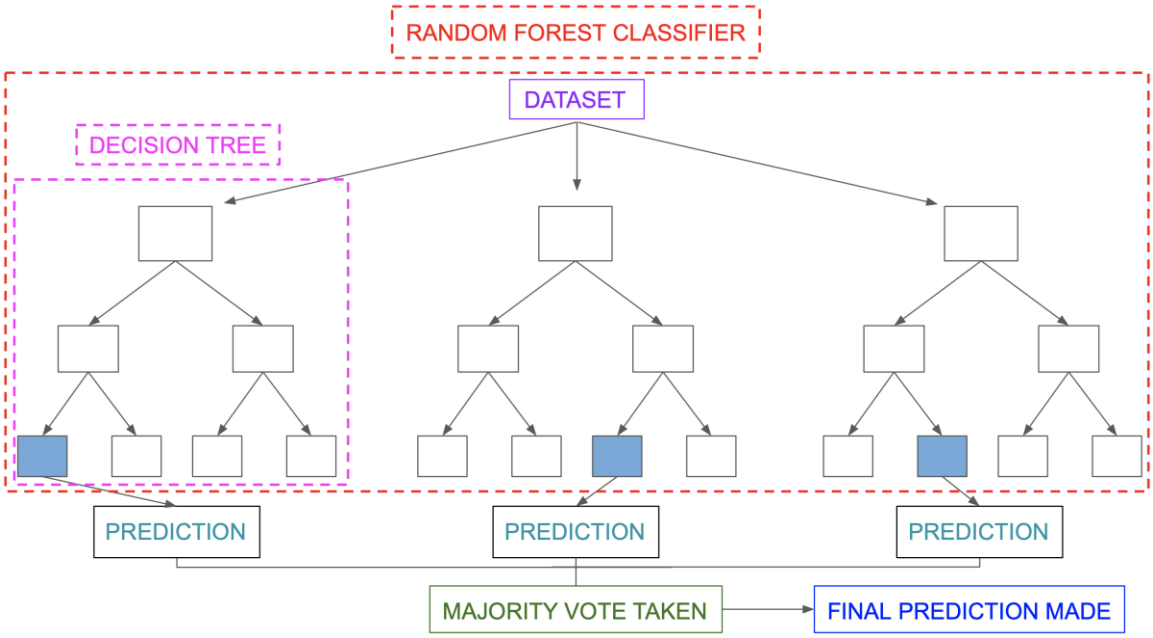

Figure 8. Diagram of a Random Forest Classifier based on Classification Technique. Source: [30].

The algorithm produces various classification trees constructed from decision tree algorithms. A decision tree consists of a root node at the top layer, decision nodes at the subsequent layers and leaf nodes at the last layer where the leaf node is the final output generated. Each tree is comprised of a data sample taken from a training set, called the bootstrap sample. From the data sample, usually thirty percent of it is set aside to test the algorithm. This is also known as the out-of-bag sample. Next, randomness is induced through feature bagging to reduce the correlation among all the classification trees. As mentioned, Random Forest utilizes both regression and classification techniques. In a classification technique, a majority vote based on the most frequent categorical variable of the classification trees will be the final prediction for the algorithm. Figure 8 depicts a simplified Random Forest classifier based on the classification technique. For regression, the individual decision tree is averaged to output the final prediction. Finally, the out-of-bag sample is then applied for cross-validation to conclude the prediction.

All machine learning algorithms require the tuning of several parameters to achieve its highest effectiveness. For the Random Forest, there are three factors to fine tune the

accuracy of the algorithm: 1) the number of classification trees, 2) the number of nodes, and 3) the number of features sampled.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. RESEARCH APPROACH

A. PROPOSED INTRUSION DETECTION SYSTEM (IDS)

In this chapter, we propose our approach to detect and sense network intrusions in a SCADA network using supervised machine learning. Our approach is built upon the assumption that malicious intent against a SCADA network, especially on Modbus/TCP, will trigger considerable variation from normal network data. As such, we use the Random Forest machine learning algorithm to analyze traffic patterns, detect network anomalies and alert the network user when such threats arise.

An overview of the proposed IDS architecture is shown in Figure 9. The proposed approach is segmented into five portions.

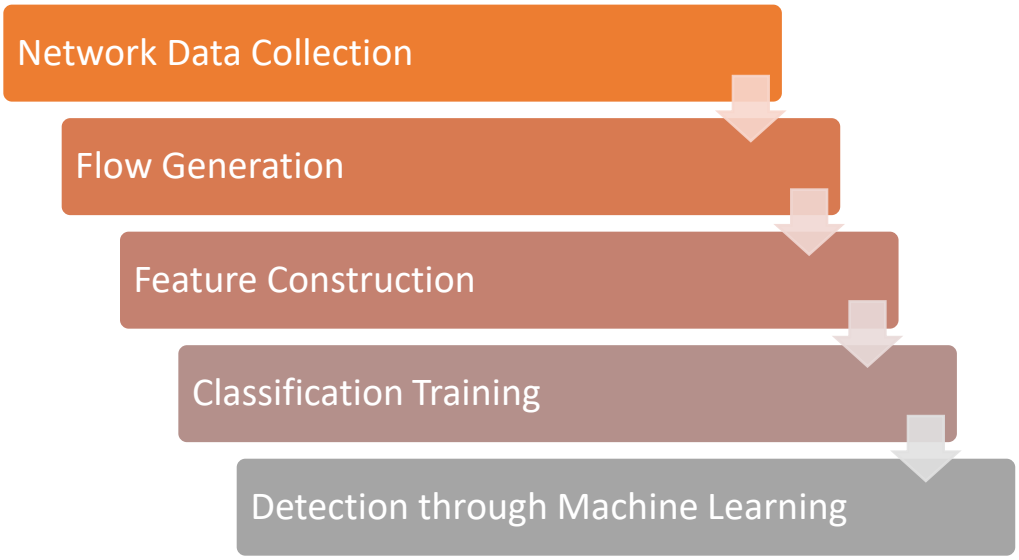

Figure 9. Overview of Proposed IDS Architecture

The first step of our IDS architecture is Network Data Collection. This process is responsible for collecting and capturing the Modbus/TCP data. Modbus/TCP is the connection-oriented version of traditional Modbus [31]. While there are other SCADA protocols such as Profibus, DNP3 and IEC-61850, our thesis is centered only on the Modbus/TCP data as this is what is used in the Navy smart grid

The Modbus/TCP data are translated into PCAP format. Basically, a PCAP file format comprises of (1) a number to identify the packet in sequence, (2) timestamp of each network packets, (3) the source IP address, (4) the source port, (5) the destination IP address, (6) the destination port, (7) the network protocol and (8) the length of the packet. Besides these eight fields, the last field, which is the information field, contains the polling and controlling information of the Modbus devices.

The second step of our architecture is denoted as Flow Generation. This is where the packet data extracted in the first step is used to cluster the packets into IP flows using an open-sourced software called CICFlowMeter. The CICFlowMeter is used on many datasets. The creator of this tool mentioned that "it can be used to generate bidirectional flows, where the first packet determines the forward (source to destination) and backward (destination to source) directions, hence more than 80 statistical network traffic features such as Duration, Number of packets, Number of bytes, Length of packets, etc., can be calculated separately in the forward and backward directions. Additional functionalities include, selecting features from the list of existing features, adding new features, and controlling the duration of flow timeout. The output of the application is the CSV format file that has six columns labeled for each flow (FlowID, SourceIP, DestinationIP, SourcePort, DestinationPort, and Protocol) with more than 80 network traffic analysis features" [32]. The advantage of having Flow Generation in the IDS is that it is able to provide an outline of the network performance by displaying the key indicators of the interfaces among the network nodes, which can disclose abnormalities from the routine behavior caused by malicious data.

The next step after Flow Generation is to construct relevant features, called Feature Construction. In this process, features are extracted from the IP flows and data instances are generated. One of the key features of the traffic in SCADA is that it is very well recognized, displaying noticeable patterns in the transmission among devices. This implies that the polling connections between the masters, Master Terminal Units (MTU), and slaves, Remote Terminal Units (RTU), adhere to a predictable schedule. Further, the network packets that communicate in SCADA typically have similar ranges. Centered on these characteristics, we deem that some of the key IP flow features such as average packet

size, source port numbers, destination port numbers and packets per second may suggest that the network packet is malicious. The key features used in this thesis are as shown in Table 2.

Table 2. Key Flow Based Features. Source: [33].

Feature	Description
source_port	Source Port, originally created on Flow Generation step, represents the source transport port used on the traffic between two devices.
dest_port	Destination Port, originally created on Flow Generation step, represents the destination transport used on the traffic between two devices.
prot	Transport Protocol, originally created on Flow Generation step, represents the transport protocol used on the traffic between two devices. It is expected to have only the ModBus/TCP inside.
num_pack	Number of Packets that compose the flow, created on the Feature Generation step, represents the count of number of packets on a flow.
size_pack	Size of Packets, created on the Feature Generation step, represents the sum of the size of all packets that compose the flow.
ppf	Packets per flow, created on the Feature Generation step, represents the count of all packets that compose the flow.
bpf	Bytes per Flow, created on the Feature Generation step, represents the ratio between size_pack and flow_duration.
avg_size_pack	Average of packets size, created on the Feature Generation step, represents the average size of the packets that compose the flow.
flow_duration	Flow Duration, created on the Feature Generation step, represents the duration of the flow.

In the Classification Training process, a set of data instances which consist of malicious or normal traffic generated from the previous Flow Generator is fed to the input of the machine learning algorithm to train and classify the data into malicious or normal traffic. This training process is an iterative method to ensure that the model is always up to date to detect new malicious data.

In the last phase, Detection through Machine Learning, the machine learning algorithm trained in the previous stage is used to detect the data instances as malicious and inform the administrator for the next course of action.

B. DATASET

1. Description of Dataset

Lemay et al. simulated a lab environment which comprised of several MTUs and RTUs connected within a local network [10]. The controllers manage a simulated physical system that is power up by a 12kV power source. The data sets include regular polling as well as physical operation to emulate the real environment of a SCADA system. In their dataset collection experiment, different types of data have been collected through the injection of different malicious attacks. Specifically, new penetration assessment tools such as Metasploit are injected into the system [15].

In terms of the weakness or rather drawbacks of this dataset, it does not incorporate a wide group of Modbus-based attacks. According to Morris and Gao, "there are several different groups of Modbus-based attacks" [15], [36]. All these attacks are not injected into the dataset that we are training. Instead, malicious data commonly tested for home and office-based networks are used in this dataset. What this means is that this dataset does not simulate a wider variety of attacks that could be used on SCADA network. On the other hand, the benefit that one can reap from this dataset is that it imitates the timing behavior and the rate of packets per duration, which could be an excellent feature for detecting malicious data.

In the dataset used, eleven sub-datasets are produced which contain six nominal data sets and five malicious data sets. Table 3 shows the description of the six nominal data sets, and Table 4 shows the description of the five malicious data sets.

Table 3. Nominal Datasets. Source: [10].

Name of Dataset	Description	Malicious Activity?	No. of entries
Run8	1 hour of regular Modbus traffic including polling and manual operation - 2 MTU, 3 RTU and 10 seconds polling interval	No	72186
Run 11	1 hour of regular Modbus traffic including polling and manual operation - 2 MTU, 3 RTU and 10 seconds polling interval	No	72498
Run1_6RTU	1 hour of regular Modbus traffic including polling and manual operation - 2 MTU, 6 RTU and 10 seconds polling interval	No	134690
Run_12RTU	1 hour of regular Modbus traffic including polling and manual operation - 2 MTU, 12 RTU and 10 seconds polling interval	No	238260
Run1_3RTU_2s	1 hour of regular Modbus traffic including polling and manual operation - 2 MTU, 3 RTU and 2 seconds polling interval	No	305932
Modbus_polling_only_6RTU	1 hour of regular Modbus traffic including polling only - 1 MTU, 3 RTU and 10 seconds polling interval	No	58325

Table 4. Malicious Datasets. Source: [10].

Name of Dataset	Description	Malicious Activity?	No. of entries
Moving_two_files_Modbus _6RTU	3 minutes of regular Modbus traffic including polling only - 1 MTU, 6 RTU and 10 seconds polling interval	Yes	3319
Send_a_fake_command_Modbus_6RTU_with_operate	11 minutes of regular Modbus traffic including polling and manual operation - 1 MTU, 6 RTU and 10 seconds polling interval. Also includes sending a Modbus write operation from a compromised RTU using Metasploit proxy functionality and the proxychains tool	Yes	11166
Characterization_Modbus_6RTU_ with_operate	5.5 minutes of regular Modbus traffic including polling and manual operation - 1 MTU, 6 RTU and 10 seconds polling interval Also includes sending a series of modbus read commands to characterize available registers from a compromised RTU	Yes	12296
CnC_uploading_exe_modbus_ 6RTU_with_operate	1 minute of regular Modbus traffic including polling and manual operation - 1 MTU, 6 RTU and 10 seconds polling interval. Also includes sending an EXE file from a compromised RTU to another compromised RTU through a Metasploit meterpreter channel	Yes	1426
6RTU_with_operate	5 minutes of regular Modbus traffic including polling and manual operation - 1 MTU, 6 RTU and 10 seconds polling interval. Also includes using an	Yes	1856

Name of Dataset	Description	Malicious Activity?	No. of entries
	exploit (ms08_netapi) from a compromised RTU to compromise another RTU using Metasploit		

2. Feature Extraction

In supervised learning, Anton et al. highlighted that one of the crucial steps before detecting an anomaly is the need to select relevant features or data required for training [15]. These characteristics are important as they represent data instances related to a certain objective in identifying occurrences that deviate significantly from the norm. Therefore, extracting such features are required to define the general behavior of the system before training them on the machine learning algorithm.

Many researchers studying datasets for IDS implementation have determined the critical features required to train their respective machine learning algorithm. Anton et al. proposed a list of 14 features in the Lemay and Fernandez dataset to train their machine learning algorithm. These features are obtained from the Ethernet header, TCP/IP header, User Datagram Protocol (UDP) headers and the capturing tool [15].

Relevant feature extraction in large data sets requires a massive amount of effort. While it is possible to code a program to evaluate and remove features that have little or negligible impact on the resultant model, it is time exhausting and therefore not economical. In our work, we are using the CICFlowMeter to extract at least 83 flow features from the Lemay and Fernandez dataset. See Appendix A for the description of the 83 flow features extracted from CICFlowMeter.

V. EXPERIMENTS AND EVALUATION

To implement our IDS, the MATLAB Classification Learner App is used to train and test KNN, Bayesian and Random Forest algorithm with the datasets provided by Lemay and Fernandez, highlighted in Tables 3 and 4 of Chapter IV. Prior to training and testing, the datasets were input to the CICFlowmeter to generate features. This chapter discusses the results that were obtained using these three algorithms.

A. PREPARATION OF DATASETS

In the malicious datasets, there were a total of four different types of attack. They are as follows:

1. Remote exploit by using a compromised machine to compromise a controller. Henceforth labelled as “Exploit” in the datasets.
2. Uploading of Malware. Henceforth labelled as “Malware” in the datasets.
3. Fingerprinting attack by sending of read packets to controller to gather information. Henceforth labelled as “Fingerprint” in the datasets.
4. Sending of unauthorized command to controller. Henceforth labelled as “Unauthorized” in the datasets.

While all the attacks mentioned in Table 4 of Chapter IV were part of the five malicious datasets, there was a mixture of normal traffic with malicious traffic. Table 5 shows a summary of the entries for each dataset in their PCAP file format that is provided by Lemay and Fernandez. There is a total of 27,432 normal entries and a total combined number of 2,631 malicious entries. The ratio of malicious to normal entries is about 1:10. These combined entries will serve its intended purpose to train and test the three machine learning algorithms.

Table 5. Summary of Entries for Dataset in PCAP File Format

Files Name	Number of Entries					
	Normal	Exploit	Malware	Fingerprint	Unauthorized	Total
Moving_two_files_Modbus_6RTU	3244				75	3319
Send_a_fake_command_Modbus_6RTU_with_operate	11156		10			11166
Characterization_Modbus_6RTU_ with_operate	11070			1226		12296
CnC_uploading_exe_modbus_ 6RTU_with_operate	1305		121			1426
6RTU_with_operate	657	1199				1856

From the PCAP files, there are very few features to train the algorithm. As such, there is a need to convert the PCAP file into a flow format CSV file where over eighty-three features can be captured for effective training. As mentioned in Chapter II, CICFlowmeter is used to convert the PCAP file into an IP flow format to display the key indicators of the interfaces among the network nodes, which can disclose abnormalities from the routine behavior caused by malicious data.

The Appendix B lays out how to set up and use the CICFlowmeter to convert the PCAP file into an IP flow format in CSV file. The final CSV file consists of a total of 3,186 entries. Out of this total, there are 3,162 normal entries and 24 malicious entries. The ratio of malicious to normal entries is about 1:132.

B. EXPERIMENTS

A confusion matrix shown in Figure 10 is normally referenced to visualize the results in a classification problem. This tool is used in the analysis of the experiments in

this section. A true positive means that the actual malicious packet is correctly predicted as malicious while a false positive means that the actual malicious packet is incorrectly predicted as a normal packet. A true negative means that the actual normal packet is correctly predicted as normal while a false negative means that the actual normal packet is incorrectly predicted as a malicious packet.

Actual Positive	True Positive	False Positive
Predicted Actual	False Negative	True Negative
	Predicted Positive	Predicted Negative

Figure 10. Confusion Matrix

1. Analysis Parameters

To determine and compare the effectiveness and efficiency of the three machine learning algorithms, we consider the following measures:

1. Overall Accuracy: The overall accuracy is the proportion of correct predictions made to the size of the dataset. However, the overall accuracy parameter is highly unreliable in classification applications for unbalanced datasets, as is the case in our experiments where the proportion of malicious to normal traffic is 1/132. It is included here for reference only, and we will complement this measure with better suited ones for unbalanced datasets scenarios.

2. True Positive Rate: The rate in which an occasion is correctly determined as an intrusion or normal packet. This parameter is indicated as TPR in the confusion matrix results.

3. False Negative Rate: The rate in which an occasion is incorrectly determine as an intrusion or normal packet. This parameter is indicated as FNR in the confusion matrix results.

4. Matthew Correlation Coefficient (MCC): It is defined as "the measurement of the quality of binary classifications that considers true and false positives and negatives" [35]. MCC is a correlation coefficient among the actual and predicted binary and will return a value between -1 to 1. The higher the value of the MCC, the better prediction is on the algorithm itself. Due to the unbalance datasets used in this experiment, we use MCC as it is a more balanced measurement compared to overall accuracy. The MCC is calculated in Equation 5.1.

$$MCC = (TP \times TN - FP \times FN)/\sqrt{((TP + FP)(TP + FN)(TN + FP)(TN + FN))} \quad (5.1)$$

where TP, TN, FP, and FN are defined as the number of true positives, true negatives, false positives and false negatives, respectively [38].

2. Simulation Set-up

Appendix C lays out how to set up and use the MATLAB Classification Learner application to carry out the tests. The general parameters for KNN, Bayesian and Random Forest are listed in Tables 6, 7 and 8, respectively. Five-fold cross-validation is used in our implementations. In such scenario the dataset is split into five subsets of equal size, and the classifier is successfully trained on four of the sets and tested on the fifth subset. The classification-testing process is repeated five times and overall classification performance obtained by averaging the performance over the five testing subsets. As mentioned in Chapter IV, there were only 24 malicious entries out of a total of 3186 entries. PCA was disabled as simulations showed no difference in results between PCA enabled and PCA disabled.

Table 6. MATLAB Parameters for KNN

Model Type	Weighted KNN
Validation	5 folds cross-validation
Number of neighbors	10
Distance Metric	Euclidean
Distance Weight	Inverse
Standardize Data	True
PCA	Disabled

Table 7. MATLAB Parameters for Bayesian

Model Type	Kernel Naïve Bayes
Validation	5 folds cross-validation
Kernel Type	Gaussian
Support	Unbounded
PCA	Disabled

Table 8. MATLAB Parameters for Random Forest

Model Type	Bagged Trees
Validation	5 folds cross-validation
Ensemble	Bagged
Learner Type	Decision Tree
Number of learners	30
PCA	Disabled

C. SIMULATION RUNS

There were two types of simulation runs carried out on the three machine learning algorithms. The first type of simulation requires the algorithm to learn based on two types of outputs labelled namely normal and malicious traffic. This is to ensure that all types of attack can be discerned from the normal traffic. The second type of simulation requires the

algorithm to learn based on the four different types of attack with normal traffic. This is to ensure that the algorithm can discern the specific type of attack from all traffic. A total of 5 runs were carried out on each algorithm to get the average result.

1. Normal – Malicious

Figure 11 shows that the attack traffic can be distinguished from normal traffic with a probability of 99.53% of overall accuracy based on the KNN model. All 5 runs showed 50% of the 24 malicious entries (12 out of 24 entries) classified incorrectly as normal traffic while 0.1% of the normal entries were incorrectly classified as malicious traffic (3 out of 3162 entries). The number of true positives, true negatives, false positives and false negatives are 12, 3159, 12 and 3 respectively. Substituting these values into equation 5.1, leads to a MCC value of 0.63. We include the MCC value as it provides a better representation of the capability of the algorithm to discriminate between normal and malicious data than the accuracy level does in the unbalanced dataset environments investigated in our study.

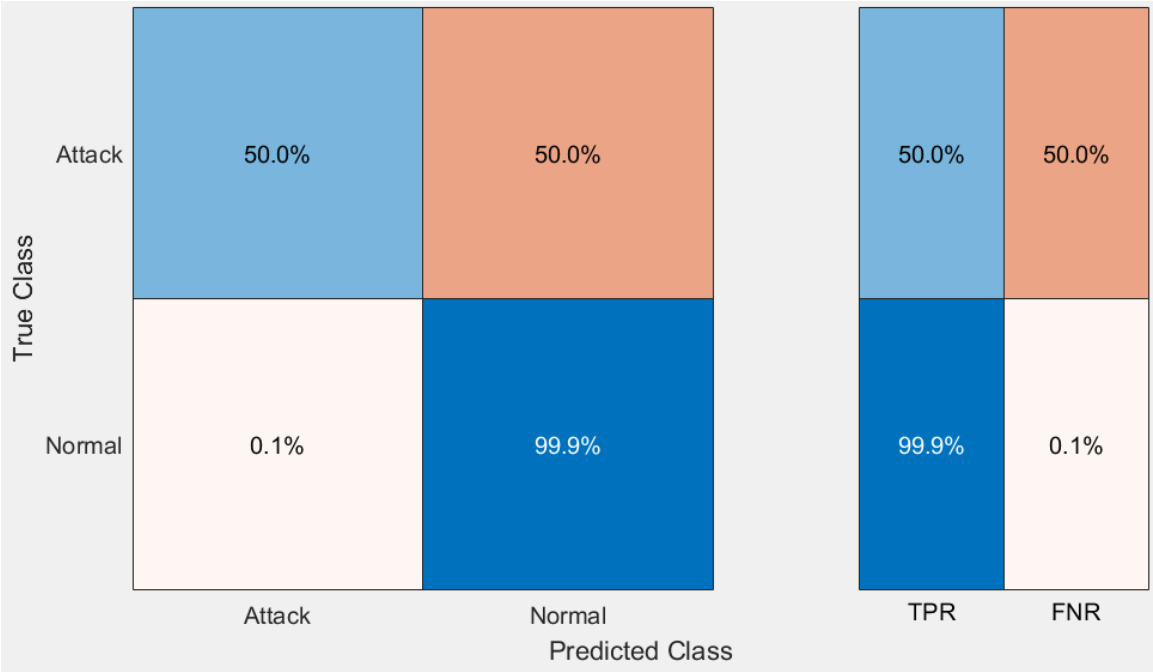

Figure 11. Normal-Malicious: Results from KNN Model

Figure 12 shows that the attack traffic can be distinguished from normal traffic with a probability of 99.37% of overall accuracy based on a Bayesian model. All 5 runs show that 66.7% of the 24 malicious entries (16 out of 24 entries) were classified incorrectly as normal traffic while 0.1% of the normal entries were incorrectly classified as malicious traffic (4 out of 3162 entries). The number of true positives, true negatives, false positives and false negatives are 8, 3158, 16 and 4, respectively. Substituting these values into equation 5.1, leads to a MCC value of 0.47.

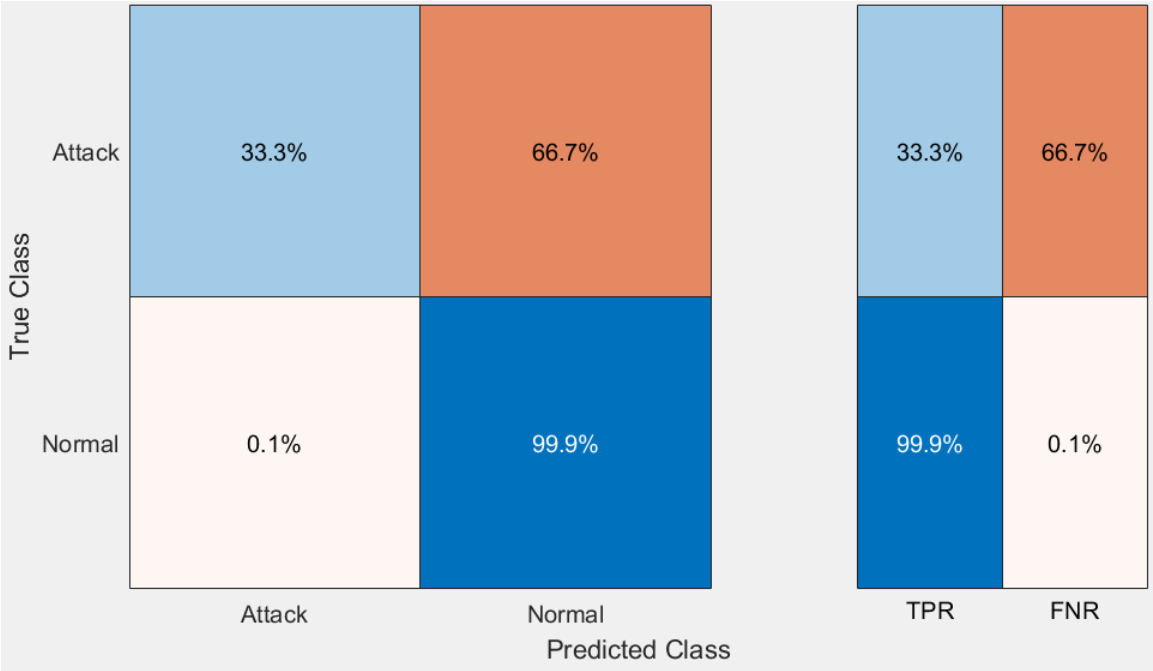

Figure 12. Normal-Malicious: Results from Bayesian Model

Figure 13 shows that the attack traffic can be distinguished from normal traffic with a probability of 99.69% of overall accuracy based on the Random Forest model. This model achieved the highest overall accuracy amongst the three models tested. All 5 runs showed 29.2% of the 24 malicious entries (7 out of 24 entries) classified incorrectly as normal traffic while 0.1% of the normal entries were incorrectly classified as malicious traffic (3 out of 3162 entries). The number of true positives, true negatives, false positives and false

negatives are 17, 3159, 7 and 3, respectively. Substituting these values into Equation 5.1, leads to a MCC value of 0.77.

Thus, based on MCC values obtained, results showed the random forest classifier is better able to discriminate between normal and malicious data than the other two classifiers investigated in the study.

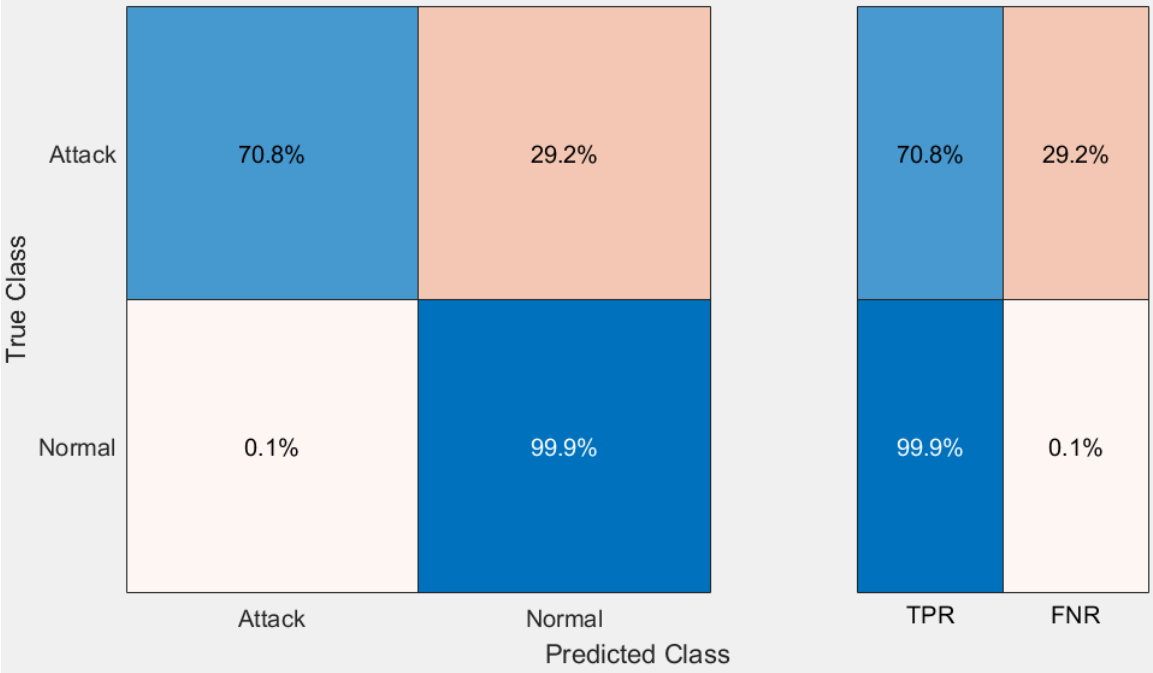

Figure 13. Normal-Malicious Results from Random Forest Model

2. Normal – Exploit – Fingerprint – Malware – Unauthorize

Figure 14 shows that the traffic can be distinguished correctly with a probability of 99.47% of overall accuracy based on the KNN model in all 5 runs. Figure 15 shows the number of observations for KNN Model. All the entries from “Malware” and “Unauthorize” entries were classified incorrectly.

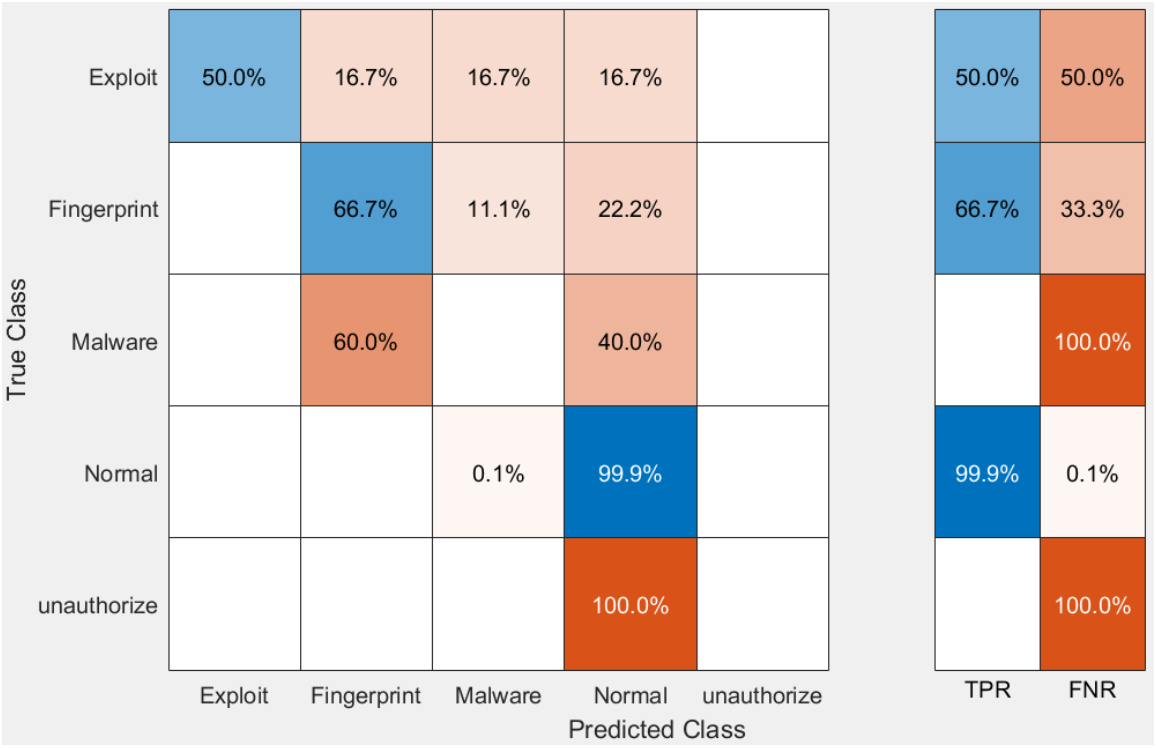

Figure 14. Results from the KNN Model for Individual Attacks

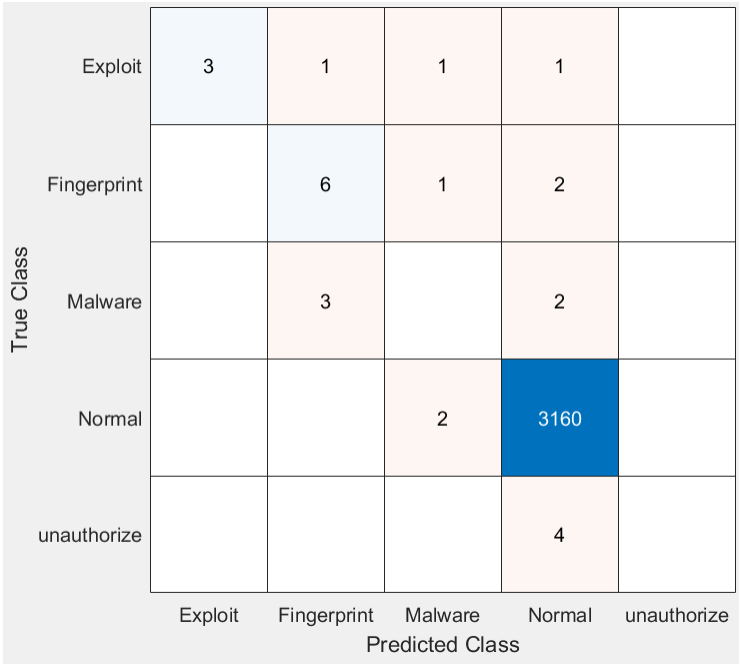

Figure 15. Number of Observations for KNN Model

Figure 16 shows that the traffic can be distinguished correctly with a probability of 95.95% of overall accuracy based on the Bayesian model. Figure 17 shows the number of observations for Bayesian Model.

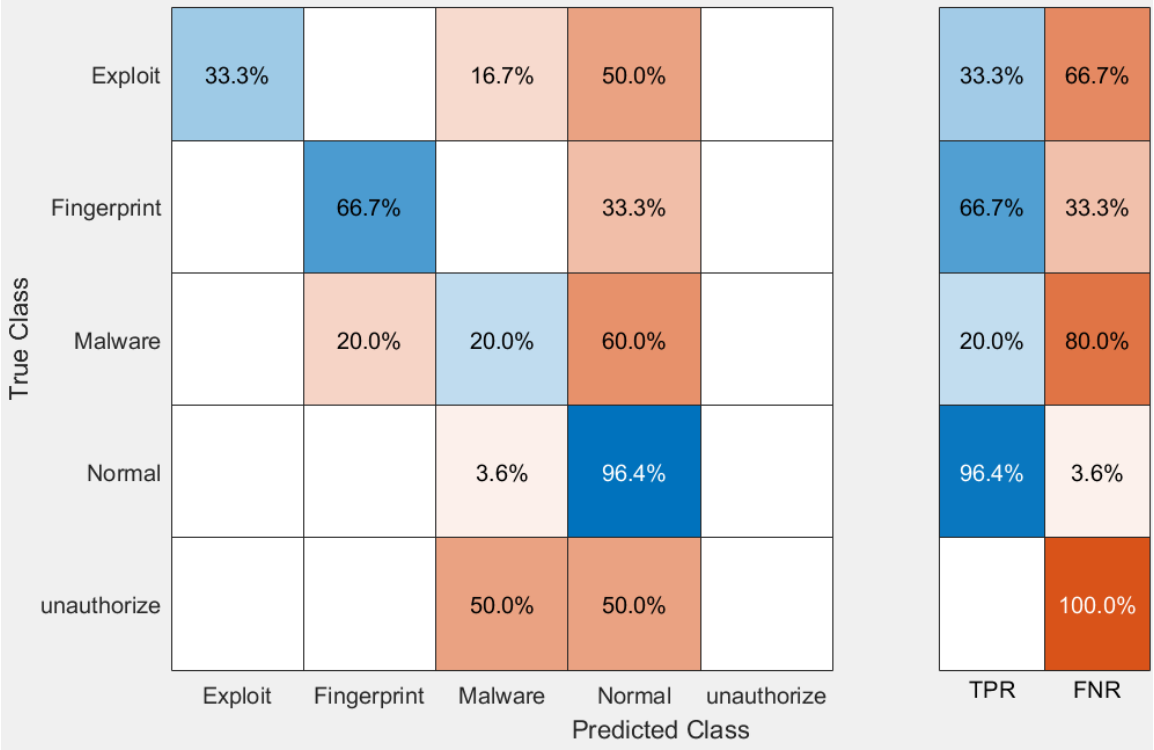

Figure 16. Results from Bayesian Model for Individual Attacks

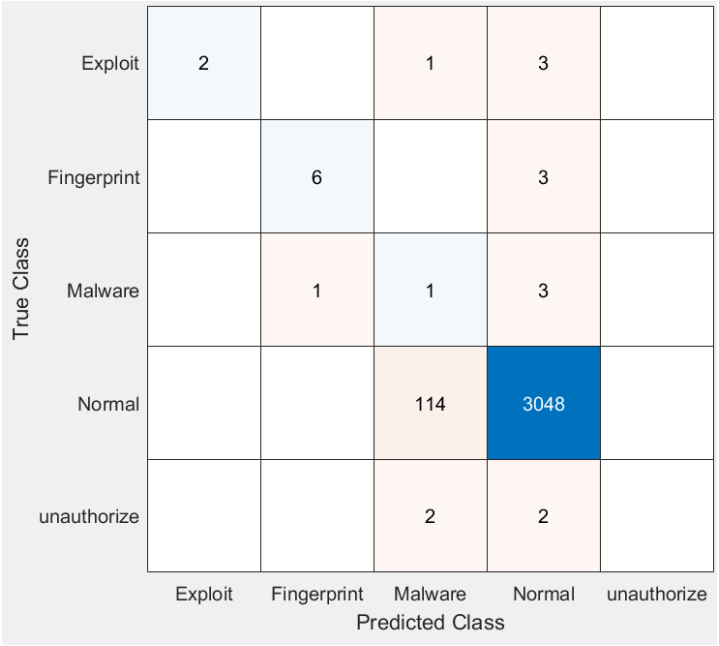

Figure 17. Number of Observations from Bayesian Model

Figure 18 shows that the attack traffic can be distinguished from normal traffic with a probability of 99.53% of overall accuracy based on Random Forest model. Figure 19 shows the number of observations for Random Forest Model.

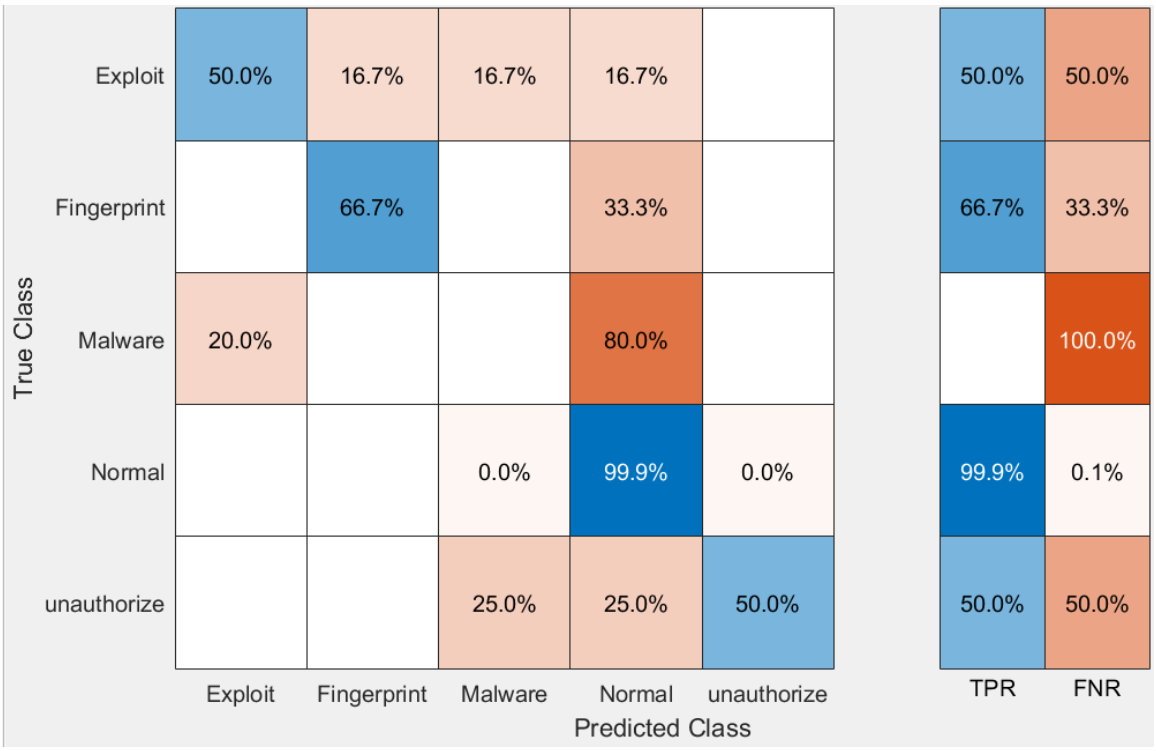

Figure 18. Results from Random Forest Model for Individual Attacks

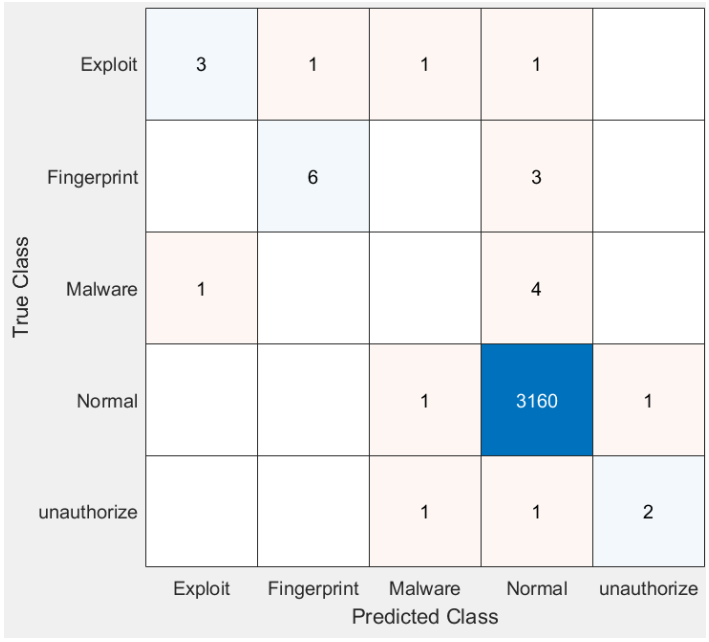

Figure 19. Number of Observations from Random Forest Model

D. SUMMARY OF ANALYSIS

In Table 9, we show the overall accuracy for the simulation runs for the three machine learning algorithms. The results show that Random Forest performs the best when it comes to detecting either malicious or normal traffic with an overall accuracy of 99.69%. When it comes to detecting the normal traffic with different types of malicious traffic, Random Forest also performs the best but slightly better than KNN with an overall accuracy of 99.53%.

Table 9. Overall Accuracy

Experiments	**Random Forest**	**KNN**	**Bayesian**
Normal - Malicious	99.69%	99.53%	99.37%
Normal – Exploit – Fingerprint – Malware – Unauthorize	99.53%	99.47%	95.95%

However, as explained in the earlier chapter, the ratio of malicious to normal entries is 1:132. As a result, the overall accuracy rate may not be a reliable parameter for such an unbalanced dataset as it does not accurately represent how malicious packets are classified. Thus, we also considered additional measures such as (1) the true positive rate defined as the ratio of malicious events detected as malicious over the total number of malicious events and (2) the Matthew correlation coefficient (MCC) to provide additional insight in the classifier performances. Simulations showed the KNN, Bayesian and Random Forest classifiers have true positive rates (also referred to as sensitivity or recall) of 50%, 33.3% and 70.8% respectively. Table 10 lists the MCC values obtained for the three models in the malicious-normal scenario. Results based on MCC values indicate that the random forest implementation performs better than the other two classifiers considered in our study.

Table 10. MCC Results in Malicious-Normal Scenario

	Random Forest	**KNN**	**Bayesian**
MCC	0.77	0.63	0.47

VI. CONCLUSION AND FUTURE WORKS

A. SUMMARY AND CONCLUSION

The Navy cybersecurity defenses must stay resilient while retaining its technical edge. By deploying its smart grid, the Navy aims to increase the dependability and efficiency of its energy supply. Once placed on a network, the Navy smart grid infrastructure can be exposed to cyber attacks by both state and non-state actors.

Our objective in this thesis was to build an IDS using the Random Forest machine learning algorithm. We then compared our approach to that of previous NPS research that used KNN and Bayesian learning models to detect and classify anomalous traffic and incidents. In this thesis, we have completed the following work:

1. Designed an IDS architecture for the Navy smart grid that is comprised of the following components: Network Data Collection, Flow Generation, Feature Construction, Classification Training, and Detection through Machine Learning.
2. Compared KNN, Bayesian and Random Forest algorithm on Lemay and Fernandez's datasets and average the result of each model by running 5 times.
3. Validated via MATLAB that the Random Forest algorithm is a more efficient approach based on performances shown on true positive rates, MCC values and overall accuracy.

B. FUTURE WORK

1. Generate Datasets from Navy Smart Grid

Due to the classification of dataset from the Navy smart grid, we were not able to use the classified dataset from the Navy smart grid. In this thesis, we used an alternative dataset that is publicly available to run our simulations. However, such publicly available dataset generated from the SCADA sandbox is not able to closely mimic the Navy smart

grid. An actual dataset generated from the Navy smart grid that can be used to train specific machine learning algorithms will be of practical benefit.

2. Using Unsupervised Machine Learning Algorithms

Unsupervised machine learning discovers patterns in a data set before classifying them as features or classification groups with the least amount of human intervention. This would allow the machine learning algorithm to recognize a larger number of characteristics while also categorizing new data as network traffic moves and grows.

APPENDIX A. DESCRIPTIONS OF FLOW FEATURES FROM CICFLOWMETER

The following table describes the flow features extracted from the CICFlowMeter.

Table 11. Descriptions of Flow Features from CICFlowMeter. Source: [36].

Feature Name	Description
Flow duration	Duration of the flow in Microsecond
total Fwd Packet	Total packets in the forward direction
total Bwd packets	Total packets in the backward direction
total Length of Fwd Packet	Total size of packet in forward direction
total Length of Bwd Packet	Total size of packet in backward direction
Fwd Packet Length Min	Minimum size of packet in forward direction
Fwd Packet Length Max	Maximum size of packet in forward direction
Fwd Packet Length Mean	Mean size of packet in forward direction
Fwd Packet Length Std	Standard deviation size of packet in forward direction
Bwd Packet Length Min	Minimum size of packet in backward direction
Bwd Packet Length Max	Maximum size of packet in backward direction
Bwd Packet Length Mean	Mean size of packet in backward direction
Bwd Packet Length Std	Standard deviation size of packet in backward direction
Flow Bytes/s	Number of flow bytes per second
Flow Packets/s	Number of flow packets per second
Flow IAT Mean	Mean time between two packets sent in the flow
Flow IAT Std	Standard deviation time between two packets sent in the flow
Flow IAT Max	Maximum time between two packets sent in the flow
Flow IAT Min	Minimum time between two packets sent in the flow
Fwd IAT Min	Minimum time between two packets sent in the forward direction
Fwd IAT Max	Maximum time between two packets sent in the forward direction

Feature Name	Description
Fwd IAT Mean	Mean time between two packets sent in the forward direction
Fwd IAT Std	Standard deviation time between two packets sent in the forward direction
Fwd IAT Total	Total time between two packets sent in the forward direction
Bwd IAT Min	Minimum time between two packets sent in the backward direction
Bwd IAT Max	Maximum time between two packets sent in the backward direction
Bwd IAT Mean	Mean time between two packets sent in the backward direction
Bwd IAT Std	Standard deviation time between two packets sent in the backward direction
Bwd IAT Total	Total time between two packets sent in the backward direction
Fwd PSH flags	Number of times the PSH flag was set in packets travelling in the forward direction
Bwd PSH Flags	Number of times the PSH flag was set in packets travelling in the backward direction
Fwd URG Flags	Number of times the URG flag was set in packets travelling in the forward direction
Bwd URG Flags	Number of times the URG flag was set in packets travelling in the backward direction
Fwd Header Length	Total bytes used for headers in the forward direction
Bwd Header Length	Total bytes used for headers in the backward direction
FWD Packets/s	Number of forward packets per second
Bwd Packets/s	Number of backward packets per second
Packet Length Min	Minimum length of a packet
Packet Length Max	Maximum length of a packet
Packet Length Mean	Mean length of a packet
Packet Length Std	Standard deviation length of a packet
Packet Length Variance	Variance length of a packet
FIN Flag Count	Number of packets with FIN
SYN Flag Count	Number of packets with SYN

Feature Name	Description
RST Flag Count	Number of packets with FRST
PSH Flag Count	Number of packets with PUSH
ACK Flag Count	Number of packets with ACK
URG Flag Count	Number of packets with URG
CWR Flag Count	Number of packets with CWR
ECE Flag Count	Number of packets with ECE
down/Up Ratio	Download and upload ratio
Average Packet Size	Average size of packet
Fwd Segment Size Avg	Average size observed in the forward direction
Bwd Segment Size Avg	Average size observed in the backward direction
Fwd Bytes/Bulk Avg	Average number of bytes bulk rate in the forward direction
Fwd Packet/Bulk Avg	Average number of packets bulk rate in the forward direction
Fwd Bulk Rate Avg	Average number of bulk rate in the forward direction
Bwd Bytes/Bulk Avg	Average number of bytes bulk rate in the backward direction
Bwd Packet/Bulk Avg	Average number of packets bulk rate in the backward direction
Bwd Bulk Rate Avg	Average number of bulk rate in the backward direction
Subflow Fwd Packets	The average number of packets in a sub flow in the forward direction
Subflow Fwd Bytes	The average number of bytes in a sub flow in the forward direction
Subflow Bwd Packets	The average number of packets in a sub flow in the backward direction
Subflow Bwd Bytes	The average number of bytes in a sub flow in the backward direction
Fwd Init Win bytes	The total number of bytes sent in initial window in the forward direction
Bwd Init Win bytes	The total number of bytes sent in initial window in the backward direction
Fwd Act Data Pkts	Count of packets with at least 1 byte of TCP data payload in the forward direction
Fwd Seg Size Min	Minimum segment size observed in the forward

Feature Name	Description
	direction
Active Min	Minimum time a flow was active before becoming idle
Active Mean	Mean time a flow was active before becoming idle
Active Max	Maximum time a flow was active before becoming idle
Active Std	Standard deviation time a flow was active before becoming idle
Idle Min	Minimum time a flow was idle before becoming active
Idle Mean	Mean time a flow was idle before becoming active
Idle Max	Maximum time a flow was idle before becoming active
Idle Std	Standard deviation time a flow was idle before becoming active

APPENDIX B. CONVERSION OF PCAP FILE TO FLOW BASED CSV FILE USING CICFLOWMETER

This appendix aims to provide a guide to use the CICFlowmeter.

Step 1. Download the CICFlowmeter from the official website by Canadian Institute of Cybersecurity. Unzip the file to "c" drive.

Step 2. Run the "CICFlowMeter.bat" found in "c:\CICFlowmeter\bin" folder. The following program will be loaded. You will need to have a Java Development Kit (JDK) installed to load the program. See Figure 20 below.

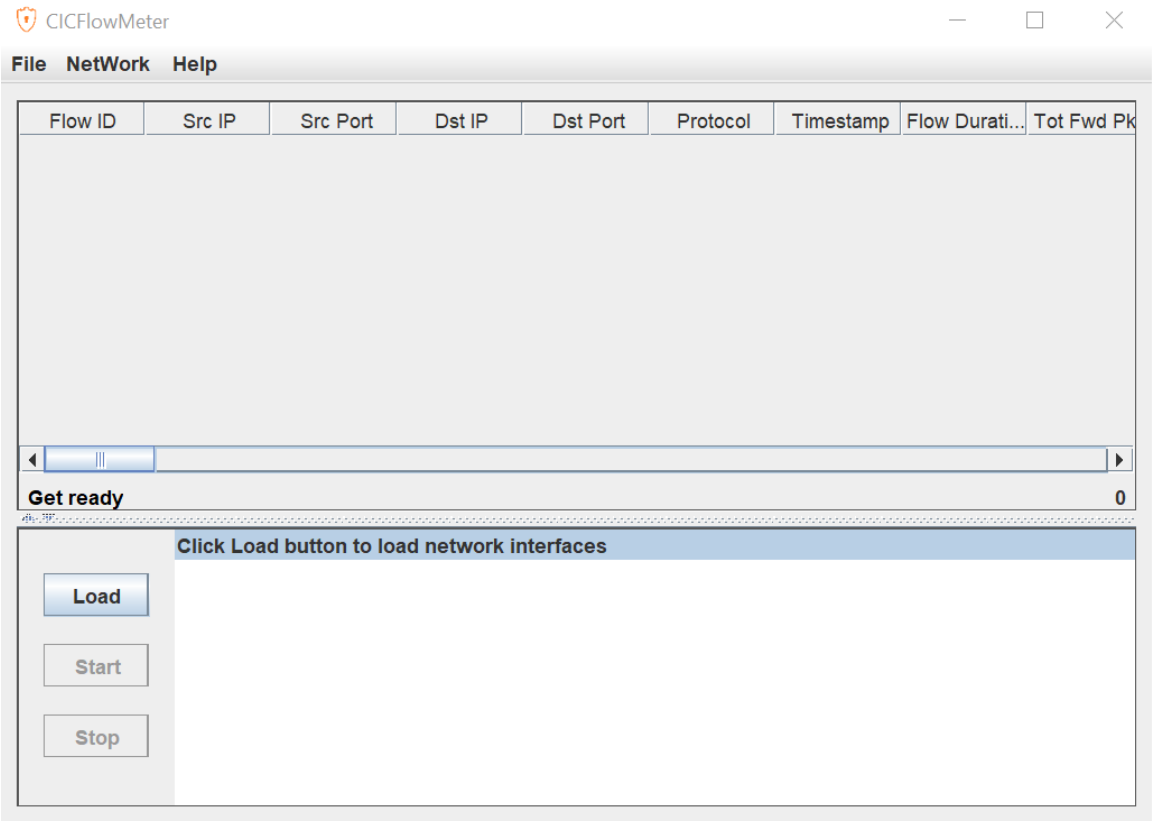

Figure 20. CICFlowmeter Program

Step 3. Select "Offline" from the "NetWork" dropdown list to convert PCAP file offline. See Figure 21 after clicking on "Offline."

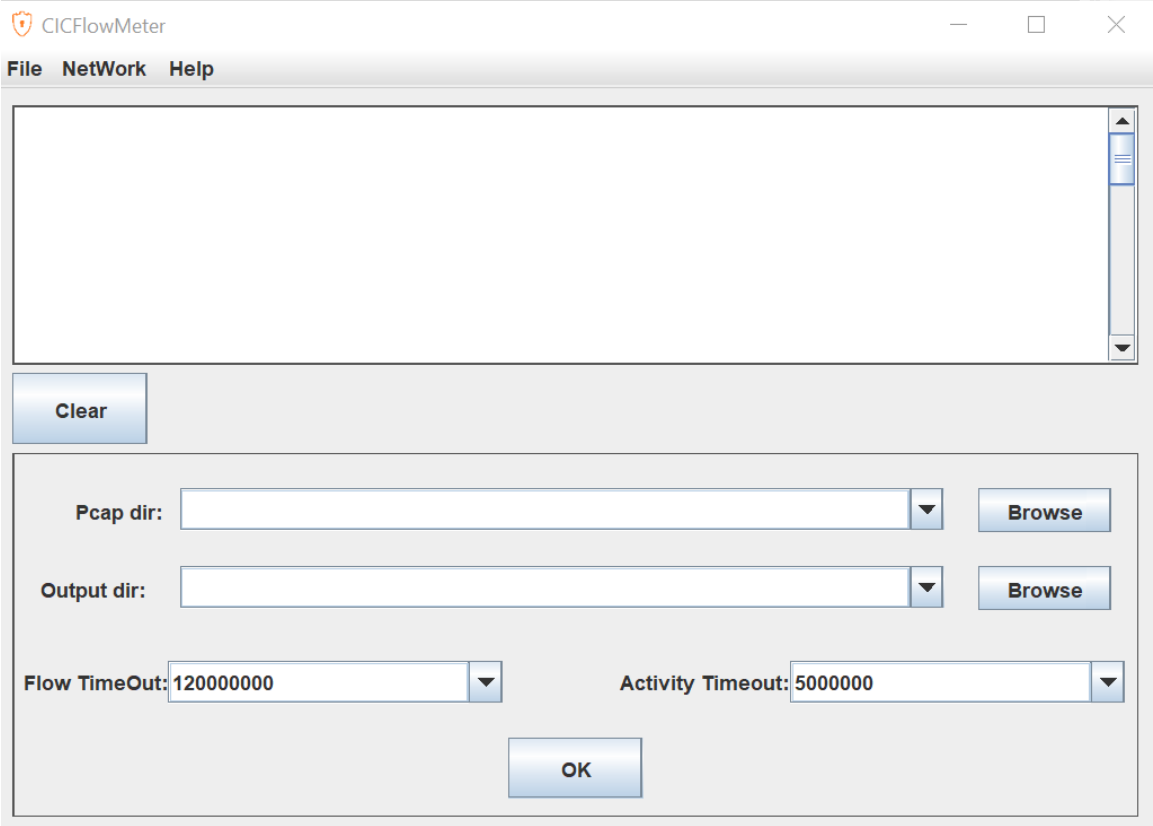

Figure 21. Offline Conversion

Step 4. In the “Pcap dir,” select the PCAP file to convert. In the “Output dir,” select the location of the file to save to after conversion. Click on “OK” to convert.

APPENDIX C. USING THE MATLAB CLASSIFICATION LEARNER APP

This appendix aims to provide a guide to use the MATLAB classification learner app. The figures are provided using version 2020b.

Step 1. Click on the “Classification Learner” from the “APPS” tab.

Figure 22. Locating Classification Learner App

Step 2. Next, click the “New Session” and select “From File.”

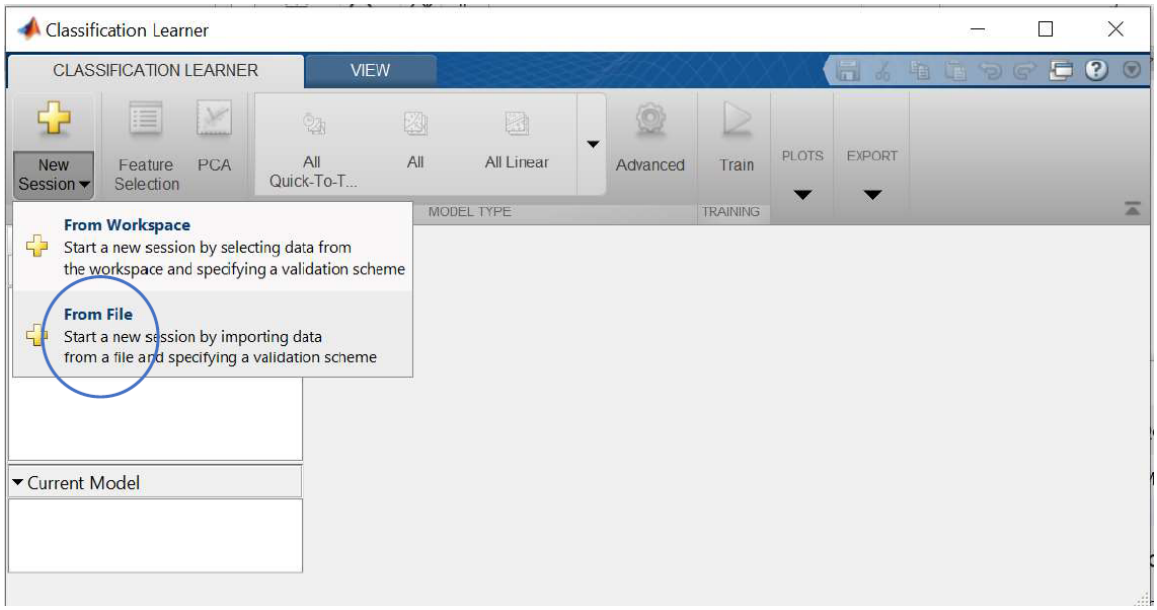

Figure 23. Starting a New Session

Step 3. Choose the CSV file that is generated from the CICFlowmeter and click “Import Selection.”

Step 4. Select “holdout validation” and set to 25%. Click on “Start Session.”

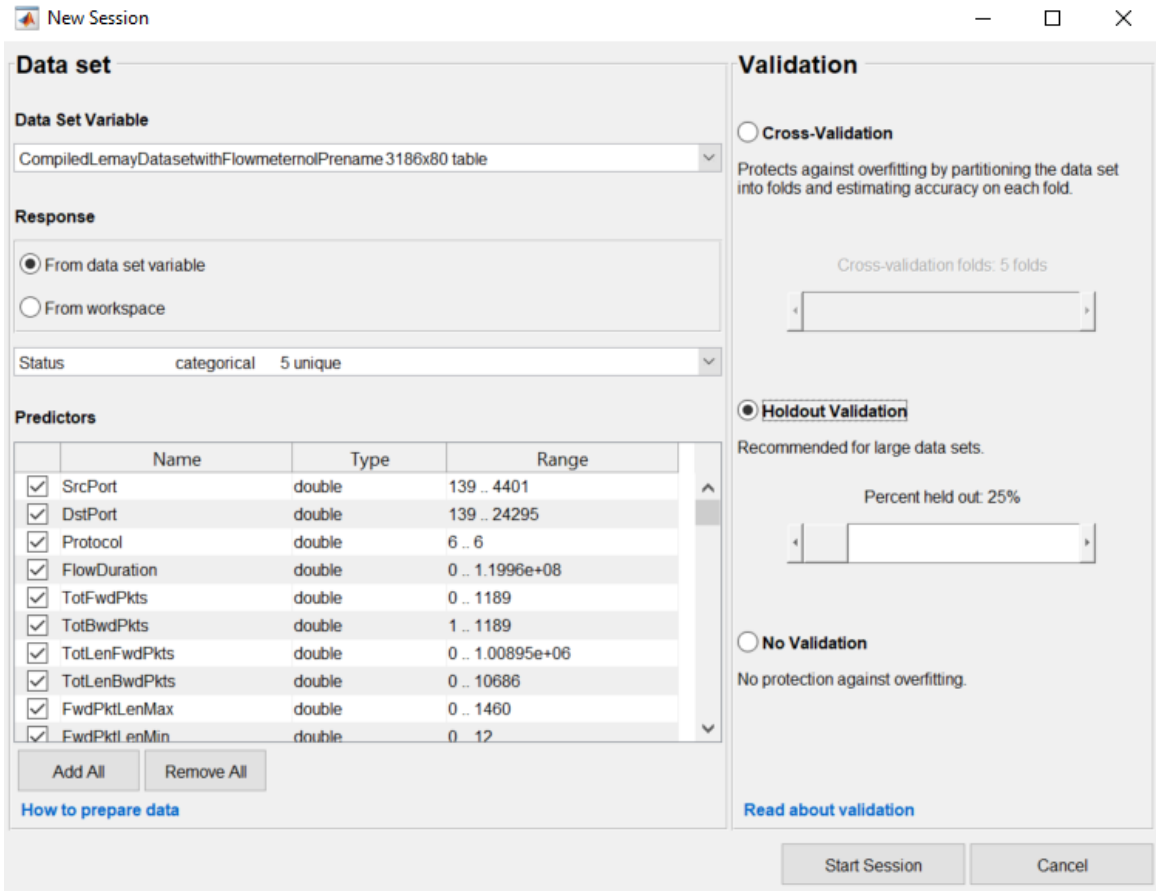

Figure 24. Holdout Validation at 25%

Step 5. From the algorithm selection drop down box, select “Weighted KNN,” “Kernel Bayes Naïve” and “Bagged Trees.” Click “Train.”

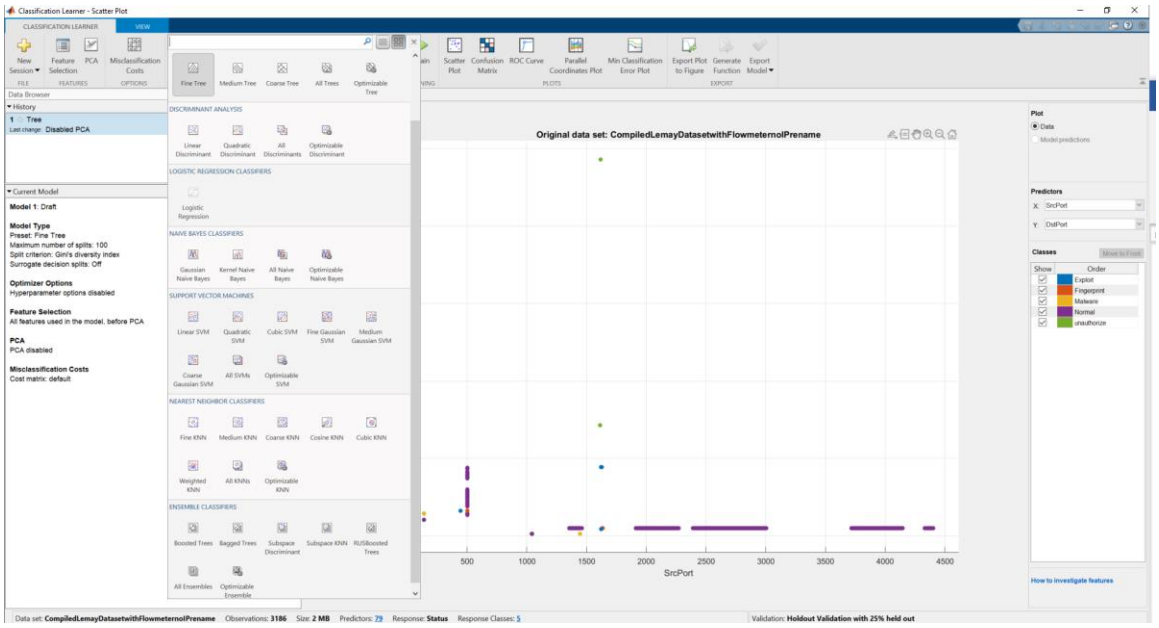

Figure 25. Selecting the Various Algorithms for Training

Step 6. Once training is completed, run the following code to check for the results.

```
%This script is to check the accuracy of the imported models

testdata=readtable("Compiled_Lemay_Dataset_with_Flowmeter_noIP.csv");
%% Bayesian
predictions1 = char(BayesianModel.predictFcn(CompiledLemayDatasetwithFlowmeternoIP));
%accuracy
iscorrect1=predictions1==cell2mat(string((CompiledLemayDatasetwithFlowmeternoIP.Status)));
iscorrect1=iscorrect1(:,2);
Bayesian_accuracy = sum(iscorrect1)*100/3186

%% KNN
predictions2 = char(KNNModel.predictFcn(CompiledLemayDatasetwithFlowmeternoIP));
%accuracy
iscorrect2=predictions2==cell2mat(string((CompiledLemayDatasetwithFlowmeternoIP.Status)));
iscorrect2=iscorrect2(:,2);
KNN_accuracy = sum(iscorrect2)*100/3186

%% Random Forest
predictions3 = char(RandomForestModel.predictFcn(CompiledLemayDatasetwithFlowmeternoIP));
%accuracy
iscorrect3=predictions3==cell2mat(string((CompiledLemayDatasetwithFlowmeternoIP.Status)));
iscorrect3=iscorrect3(:,2);
RandomForest_accuracy = sum(iscorrect3)*100/3186
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] J. A. Momoh, Energy Processing and Smart Grid, New Jersey: John Wiley & Sons, Inc., 2018.

[2] U.S. Department of Energy, "The Smart Grid," [Online]. Available: https://www.smartgrid.gov/the_smart_grid/smart_grid.html. [Accessed 01 Jul 2021].

[3] "Navy and Marine Corps Smart Grid CDD Industry Version," NAVFAC, Washington, DC, USA, 2014.

[4] U.S. Navy Public Affairs, "NAVFAC Reaches Key Milestone in Deploying New Smart Energy Monitoring and Control Solution," 30 Apr 2019. [Online]. Available: https://www.navy.mil/submit/display.asp?story_id=109430#. [Accessed 01 Jul 2021].

[5] National Institure of Standards and Technology, "Guide to Industrial Control Systems (ICS) Security," May 2015. [Online]. Available: http://dx.doi.org/10.6028/NIST.SP.800-82r2. [Accessed 01 Jul 2021].

[6] N. Falliere, L. O. Murchu and E. Chien, "W32.Stuxnet Dossier," Symantec Security Response, 2010.

[7] N. Stinchcombe, "Cloud Computing in the Spotlight," in *Infosecurity*, 2009, pp. 30–33.

[8] V. Chan, "Using a K-Nearest Neighbors Machine Learning Approach to Detect Cyberattacks on the Navy Smart Grid," Naval Postgraduate School, Monterey, California, 2020.

[9] C. A. Schiesser, "Malicious Threat Detection for NAVFAC Based Smart Grid Network Using Bayesian Classification and Machine Learning," Naval Postgraduate School, Monterey, California, 2020.

[10] A. Lemay and J. Fernandez, "Providing SCADA network data sets for intrusion detection research," 2016.

[11] M. ÅKERMAN, "Implementing Shop Floor IT for Industry 4.0," Chalmers University of Technology, Sweden, 2018.

[12] K. Stouffer, J. Falco and K. Kent, "Guide to Supervisory Control and Data Acquisition (SCADA) and Industrial Control Systems Security," National Institute of Standards and Technology, 2006.

[13] MODICON Inc., “Modicon Modbus Protocol Reference Guide,” Jun 1996. [Online]. Available: https://modbus.org/docs/PI_MBUS_300.pdf. [Accessed 01 Jul 2021].

[14] B. Drury, *Control Techniques Drives and Controls Handbook* (2nd edition), Institution of Engineering and Technology, 2009.

[15] S. D. Anton, S. Kanoor, D. Fraunholz and H. D. Schotten, “Evaluation of Machine Learning-based Anomaly Detection Algorithms on an Industrial Modbus/TCP Data Set,” German Research Center for Artificial Intelligence, Kaiserslautern, Germany, 2019.

[16] Control, “Introduction to Modbus,” 2021. [Online]. Available: https://www.controlglobal.com/articles/2019/introduction-to-modbus/. [Accessed 03 Jul 2021].

[17] M. A. Teixeira, T. Salman, M. Zolanvari, R. Jain, N. Meskin and M. Samaka, “SCADA System Testbed for Cybersecurity Research Using Machine Learning Approach,” Basel, Switzerland, 2018.

[18] L. Rosa, T. Cruz, P. Simoes, E. Monteiro and L. Lev, “Attacking SCADA systems: a practical perspective,” in *FIP/IEEE International Symposium on Integrated Network Management*, Lisbon, Portugal, 2017.

[19] P. Angelo and A. C. Drummond, “A Survey of Random Forest Based Methods for Intrusion Detection Systems,” *ACM Computing Surveys*, vol. 51, no. 3, pp. 1–36, 2018.

[20] H. Chauhan, V. Kumar, S. Pundir and E. Pilli, “A Comparative Study of Classification Techniques for Intrusion Detection,” in *International Symposium on Computational and Business Intelligence*, 2013.

[21] M. Belavagi and B. Muniyal, “Performance evaluation of supervised machine learning algorithms for Intrusion Detection,” in *Twelfth International Multi-Conference on Information Processing*, 2016.

[22] N. Ashraf, W. Ahmad and R. Ashraf, “A Comparative Study of Data Mining Algorithms for High Detection Rate in Intrusion Detection System,” *Annals of Emerging Technologies in Computing (AETiC),* vol. 2, no. 1, 2018.

[23] MathWorks, Inc., “Machine Learning in MATLAB,” [Online]. Available: https://www.mathworks.com/help/stats/machine-learning-in-matlab.html. [Accessed 04 Jul 2021].

[24] IBM, “Unsupervised Learning,” 21 Sep 2020. [Online]. Available: https://www.ibm.com/cloud/learn/unsupervised-learning. [Accessed 06 Jul 2021].

[25] R. Duda and P. Hart, “Nearest neighbor pater classification,” *IEEE Trans. Inf. Theory,* vol. 13, pp. 21–27, 1967.

[26] S. Sun and R. Huang, “An Adaptive k-Nearest Neighbor Algorithm,” in *2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery*, Shanghai, China, 2010.

[27] I. Jose, “KNN (K-Nearest Neighbors) #1,” 8 Nov 2018. [Online]. Available: https://towardsdatascience.com/knn-k-nearest-neighbors-1-a4707b24bd1d. [Accessed 05 Jul 2021].

[28] K. R. Chowdhary, “Statistical Learning Theory,” in *Fundamentals of Artificial Intelligence*, Springer, 2020, p. 428.

[29] N. B. Amor, S. Benferhat and Z. Elouedi, “Naive Bayes vs Decision Trees in Intrusion Detection Systems,” in *ACM Symposium on Applied Computing*, Nicosia, Cyprus, 2004.

[30] F. V. Jensen, “Introduction to Bayesian Networks,” UCL Press, 1996.

[31] L. Breiman, “Random Forests,” *Machine Learning,* vol. 45, no. 1, pp. 5–32, 2001.

[32] O. Mbaabu, “Introduction to Random Forest in Machine Learning,” 11 Dec 2020. [Online]. Available: https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/. [Accessed 01 Jul 2021].

[33] A. Swales, “Open Modbus/TCP Specification,” 29 Mar 1999. [Online]. Available: https://wingpath.co.uk/docs/modbus_tcp_specification.pdf. [Accessed 02 Jul 2021].

[34] Canadian Institute for Cybersecurity, “CICFlowMeter,” 2017. [Online]. Available: https://www.unb.ca/cic/research/applications.html#CICFlowMeter. [Accessed 03 Jul 2021].

[35] G. Vasquez, R. Miani and B. Zarpelao, “Flow-Based Intrusion Detection for SCADA networks using Supervised Learning,” 2017.

[36] T. Morris and W. Gao, “Industrial Control System Traffic Data Sets for Intrusion Detection Research,” in *Critical Infrastructure Protection VIII*, Berlin, IFIP Advances in Information and Communication Technology, 2014, pp. 65–78.

[37] B. W. Matthews, “Comparison of the predicted and observed secondary structure of T4 phage lysozyme,” *Biochimica et Biophysica Acta (BBA) - Protein Structure*, vol. 405, no. 2, pp. 442–451, 20 Oct 1975.

[38] B. Shmueli, “Matthews Correlation Coefficient is The Best Classification Metric You’ve Never Heard Of,” 22 Nov 2019. [Online]. Available: https://towardsdatascience.com/the-best-classification-metric-youve-never-heard-of-the-matthews-correlation-coefficient-3bf50a2f3e9a. [Accessed 01 Aug 2021].

[39] A. H. Lashkari, “CICFlowMeter/ReadMe.txt,” 07 Jun 2021. [Online]. Available: https://github.com/ahlashkari/CICFlowMeter/blob/master/ReadMe.txt. [Accessed 10 Jul 2021].

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California