NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

MALICIOUS THREAT DETECTION FOR THE NAVFAC-BASED SMART GRID NETWORK USING BAYESIAN CLASSIFICATION AND MACHINE LEARNING

by

Carolyn A. Schiesser

September 2020

Thesis Advisor:	Preetha Thulasiraman
Second Reader:	Murali Tummala

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

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

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2020	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
MALICIOUS THREAT DETECTION FOR THE NAVFAC-BASED SMART GRID NETWORK USING BAYESIAN CLASSIFICATION AND MACHINE LEARNING

5. FUNDING NUMBERS

6. AUTHOR(S) Carolyn A. Schiesser

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)
N/A

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)

With the Navy's focus on efficient energy consumption, Naval Facilities Engineering Command deployed its own Smart Grid in 2019, allowing shore commands to modernize and meet energy consumption mandates set in place by the Secretary of the Navy. With the addition of new "smart" technology comes additional risks in the form of cyber-attacks. This thesis implements a Bayesian classification and machine learning algorithm that explores how the data set, size of training data and number of features affect classification accuracy. Our experiment was performed using seven data sets, developed through the University of Montreal using a SCADA sandbox similar to that of the Navy Smart Grid. Three data sets contained nominal data, and four data sets contained malicious cyber-attacks. Our experiments, performed using MATLAB, showed that malicious packet distribution within the data set and size of the training data greatly affected classification accuracy. This thesis demonstrates machine learning operability for use in the Smart Grid environment and will provide data points to further research for Network Intrusion Detection Systems (NIDS).

14. SUBJECT TERMS Bayesian machine learning, Bayesian classification, Naval Facilities Engineering Command, NAVFAC, Supervisory Control and Data Acquisition, SCADA, cyber security	**15. NUMBER OF PAGES** 79
	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.

MALICIOUS THREAT DETECTION FOR THE NAVFAC-BASED SMART GRID NETWORK USING BAYESIAN CLASSIFICATION AND MACHINE LEARNING

Carolyn A. Schiesser
Lieutenant, United States Navy
BS, U.S. Naval Academy, 2013

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2020

Approved by: Preetha Thulasiraman
Advisor

Murali Tummala
Second Reader

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

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

With the Navy's focus on efficient energy consumption, Naval Facilities Engineering Command deployed its own Smart Grid in 2019, allowing shore commands to modernize and meet energy consumption mandates set in place by the Secretary of the Navy. With the addition of new "smart" technology comes additional risks in the form of cyber-attacks. This thesis implements a Bayesian classification and machine learning algorithm that explores how the data set, size of training data and number of features affect classification accuracy. Our experiment was performed using seven data sets, developed through the University of Montreal using a SCADA sandbox similar to that of the Navy Smart Grid. Three data sets contained nominal data, and four data sets contained malicious cyber-attacks. Our experiments, performed using MATLAB, showed that malicious packet distribution within the data set and size of the training data greatly affected classification accuracy. This thesis demonstrates machine learning operability for use in the Smart Grid environment and will provide data points to further research for Network Intrusion Detection Systems (NIDS).

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AOR	Area of Responsibility
CFS	Correlation Based Feature Selection
CON	Consistency Based Feature Selection
DOD	Department of Defense
ICS	Industrial Control System
MLP	Multi-layer Perception
MTU	Main Terminal Unit
NAVFAC	Naval Facilities Engineering Command
NIDS	Network Intrusion Detection System
PERA	Purdue Enterprise Reference Architecture
RTU	Remote Terminal Unit
SCADA	Supervisory Control and Data Acquisition

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I am continuously amazed at the opportunities the Navy has afforded me. To the intelligent, kind and encouraging friends and classmates who supported me on this journey, thank you. To Dr. Preetha Thulasiraman and Professor Murali Tummala, this thesis inspired passion about topics I thought were not my forte. Thank you for being excellent teachers. To my parents, your support is never-ending. I would not be where I am today without your guidance. To Kristen, from plebe year to finishing my master's degree, thank you for always taking the time to read my papers and helping me become a better writer. And last, to my husband, Steven, your love and encouragement gives me life so that I can accomplish anything.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

In October 2009, Secretary of the Navy Ray Mabus convened the first Naval Energy Forum and announced his five energy goals with corresponding mandates [1]. Mabus stated that "existing statutes and executive orders require reductions in facility energy intensity and greenhouse gas emissions while expanding the use of renewable and alternative sources of energy" [2]. To reduce facility energy usage, the integration of the smart grid was identified as a way to provide conservation and efficiency [2].

A. THE POWER GRID

The traditional power grid transmission infrastructure, shown in Figure 1, consists of the passive transfer of electricity from power plant to consumers in the home. This architecture requires technicians to manually record data from meters on-location, a time-consuming troubleshooting process that delays power restoration [3].

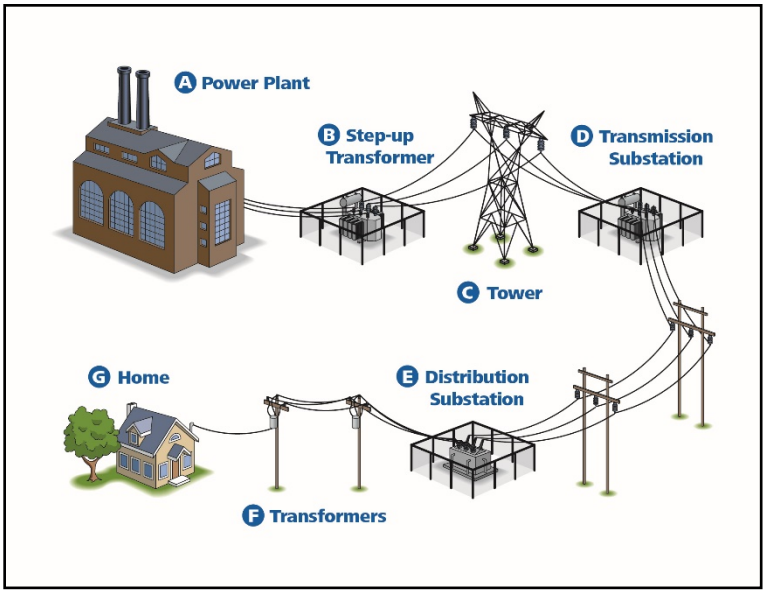

Figure 1. Example of Traditional Power Grid Architecture. Source: [3].

The Naval Facilities Engineering Command (NAVFAC) has provided power and all associated services to naval bases in their areas of responsibility (AOR) through the traditional power grid. With the fleet focus on energy consumption, changing from an expendable resource to a precious commodity, modernizations were necessary to meet energy mandates. In 2019, the NAVFAC deployed its own Smart Grid infrastructure to various Naval bases in the U.S. An example of a generic smart grid architecture is shown in Figure 2. The smart grid facilitates two-way power transmission and communication where information is passed from the energy management system to the consumer smart meter in the form of data packets. With real-time information about power flow, systems can be designed to automatically respond to outages and other internal/external problems. The new NAVFAC Smart Grid offers immediate assistance; with the ability to remotely monitor energy consumption and outages, it provides real-time solutions through smart meters and connectivity via a wide area network [4]. This results in a reduction in maintenance costs and a maximization of grid efficiency. Despite the benefits, the Smart Grid comes with cyber security challenges; every piece of infrastructure connected to the wide area network, both hardwired and wireless, greatly increases the attack surface for a cyber-attack, such as passively collecting and mining data [4].

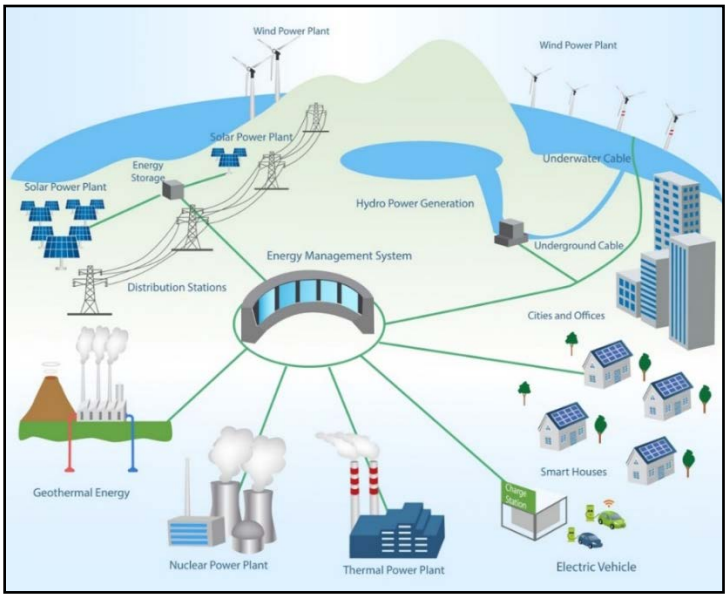

Figure 2. Example of Generic Smart Grid Architecture. Source: [5].

B. MOTIVATIONS

A malicious actor with access to a smart meter on a Naval base can passively collect data to better understand ship movements due to the correlation between naval base power consumption and the presence of ships in port. Also, they may actively execute commands in order to damage equipment and negatively affect operational readiness. Both scenarios present an unacceptable level of risk to both mission and personnel, requiring assessment of potential weaknesses. Cyber analytics can be used to continually define and mitigate evolving threat vectors after the smart grid is deployed. Machine learning is the combination of data analytics and computer science that allows computers to learn from experience. The human ability to recognize patterns within data is scaled to current technological demands. With millions of packets transmitted for system connectivity, machine learning is an effective cyber analytic approach where an immense number of packets can be analyzed in near real-time. This allows personnel to quickly isolate and re-secure infected systems should a packet be classified malicious.

C. RESEARCH CONTRIBUTION

Of the numerous machine learning algorithms available, we chose the Bayesian classification and machine learning model due to its ability to incorporate prior knowledge and current events to determine a final posterior probability. Packets travel substantially faster than our ability to verify purpose and intent. The Bayes theorem quickly incorporates large amounts of newly observed data incrementally, which results in an improvement of the final posterior probability [5]. This thereby increases the ability of the system to accurately predict certain threat vectors. The data set used in this thesis is based on the work of Lemay and Fernandez [6]. They developed a network security data set for the purpose of intrusion detection.

The thesis used the data set in [6] to train a Bayesian classification model for the NAVFAC smart grid. The contributions of this thesis are:

- Design of a network intrusion detection mechanism based on Bayesian classification and machine learning. We used data sets generated from a

Supervisory Control and Data Acquisition (SCADA) sandbox that closely aligned with the NAVFAC smart grid.

- Evaluation of developed Bayesian classification algorithm on 30 different data configurations run on 7 data sets to show high rate of prediction accuracy.

- Processing of data set based on test data and incrementally incorporating newly processed data to show effectiveness of machine learning approach.

D. THESIS ORGANIZATION

The remainder of this thesis is organized as follows: In Chapter II, we provide a thorough literature review and document the network security requirements for the Naval Smart Grid. We also discuss repercussions of a successful cyber-attack on the NAVFAC smart grid. Chapter III presents the decision to base the MATLAB program using Bayes theorem and the process by which Bayesian classification and machine learning on chosen data sets are implemented. Chapter IV describes and analyzes the results of the MATLAB program. In Chapter V, we provide the conclusion with recommendations for future research opportunities.

II. RELATED WORK

A successful network intrusion detection system (NIDS) relies on the following choices: data set, machine learning algorithm, training data size and features. This chapter addresses related work that concentrates on how manipulating these factors affects network security.

In [7], the author used the KDD Cup 1999 data set, produced through the third annual Knowledge Discovery and Data Mining Competition. The author implemented various supervised machine learning algorithms to demonstrate the ability of the algorithm to identify cyber intrusions and attacks. The attacks were categorized into four general groups: denial of service, unauthorized access from a remote machine, unauthorized access to local root privileges, and probing [7]. Forty-one features of the data set were identified and held constant throughout the algorithm assessment [8]. The supervised algorithms evaluated were j48, Random Forest, Random Tree, Decision Table, Multi-Layer Perception (MLP) Naïve Bayes Network, and Bayes Network. Average accuracy, false positive rates, and false negative rates were used to evaluate effectiveness. False negatives are detrimental to the network and consist of malicious data being classified as nominal, while false positives are nominal data classified as malicious and are undesirable as they alert an already taxed system. Table 1 compares various machine learning classifiers and their corresponding accuracy rates.

Table 1. Average Accuracy Rates. Source: [9].

Machine Learning Classifiers	Correctly classified Instances	Incorrectly classified Instances	Accuracy Rate
j48	55865	4135	93.10%
Random Forest	56265	3735	93.77%
Random tree	55345	5655	90.57%
Decision table	55464	4536	92.44%
MLP	55141	4859	91.90%
Naive Bayes	54741	5259	91.23%
Bayes Network	54439	5561	90.73%

While the Random Forest machine learning algorithm achieved the highest accuracy rate with regards to the KDD Cup 1999 data set, variable training data size and features can result in additional accuracy improvements when tailoring a NIDS to a specific system.

Similarly, a research group out of the University of Technology, Sydney proposed an innovative machine learning framework where different attacks were simulated to evaluate different machine learning classifiers [8]. The authors also used the KDD Cup 1999 data set. Theoretically, the training data size would need to be infinite in order to achieve optimum performance. This is not realistic and therefore optimization consists of minimizing required training data and false alarms while maintaining an effective detection accuracy. The authors utilized different strategies to categorize various anomalies within the data to produce the desired features. The features were then fused with different voting techniques. Voting techniques refer to the process of counting individual characteristics to produce a class which is then ranked against the others. While others have looked at algorithms individually, the authors used ensemble learning. This allows a set of machine learning algorithms to process the same data, then through the voting process decide which algorithm is best suited to the data. As every machine learning algorithm has strengths and weakness, this novel approach goes one step further and provides the machine with the necessary knowledge to make the decision.

In [9], authors utilize two feature reduction methods, Correlation Based Feature Selection (CFS) and Consistency Based Feature Selection (CON), to evaluate the effectiveness of different combinations of features. The CFS evaluates the usefulness of independent features along with their interdependencies. The CON determines the optimum features based on their ability to identify data consistently. Bayesian Network, C4.5 Decision Tree, Naïve Bayes, and Naïve Bayes Tree were used to compare the effectiveness of each machine learning algorithm. Figure 3 illustrates the effectiveness of each machine learning algorithm based on the feature combinations.

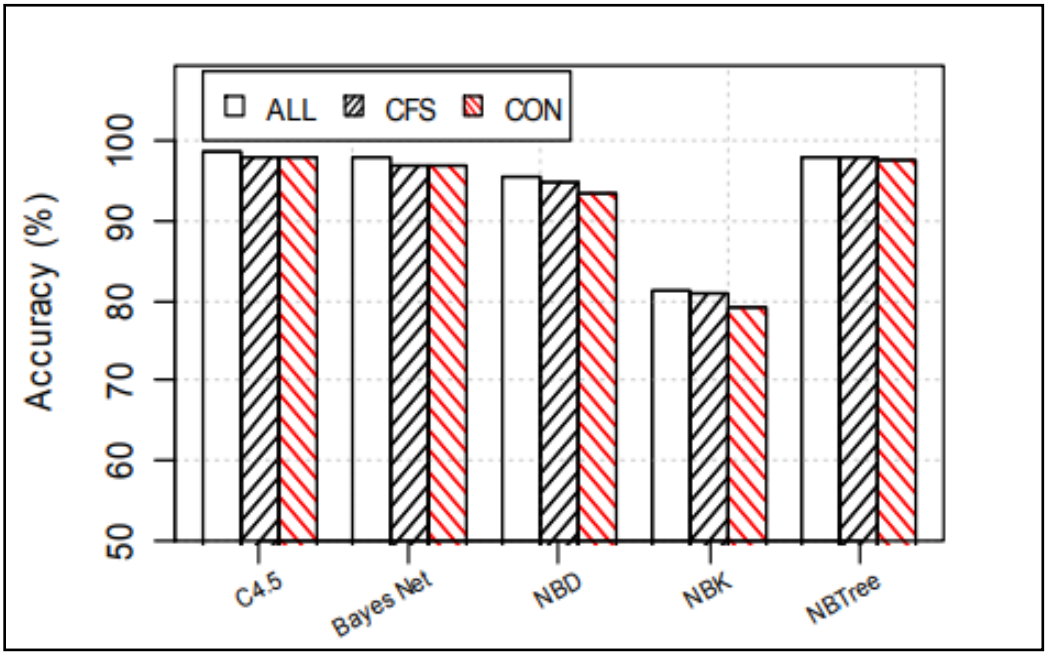

Figure 3. Accuracy of Algorithm Using CFS Subset, CON Subset and All Features. Source: [11].

Each algorithm has its own strengths and weakness and therefore must be carefully considered based on thorough analysis of the data set. In this thesis, we use a data set developed for a control system similar to the smart grid and train the Bayesian classification and machine learning algorithm using this data set. We maintain constant features and vary the training data size.

THIS PAGE INTENTIONALLY LEFT BLANK

III. DATA AND METHODOLOGY

A. DATA DESCRIPTION

The ideal data required for this thesis would be actual network traffic from one or more NAVFAC Smart Grids. Although NAVFAC provided a significant amount of documentation regarding its smart grid architecture and topology, they were unable to provide actual grid data within the time limits of this thesis. Therefore, finding a data set with similar architecture and packet protocol was essential to providing results that could be inferred onto the system of study. The NAVFAC Smart Grid is associated with a Supervisory Control and Data Acquisition (SCADA) system, an Industrial Control System (ICS) that operates to control power distribution to U.S. Naval installations. However, there are very few data sets derived from ICS and SCADA networks for the purpose of intrusion detection and network security research. This is due to the nature of proprietary and sensitive infrastructure, where releasing documentation and corresponding data sets would increase the risk of future cyber-attacks. We obtained data sets from Lemay and Fernandez at the University of Montreal that "were generated in a SCADA sandbox, where electrical network simulators were used to introduce realism in the physical component" [6]. This data set was developed for research and in an academic environment and, therefore, no proprietary limits exist.

The similarities between a SCADA Sandbox and NAVFAC network architectures are due to their origin in the Purdue Enterprise Reference Architecture (PERA) model shown in Figure 4.

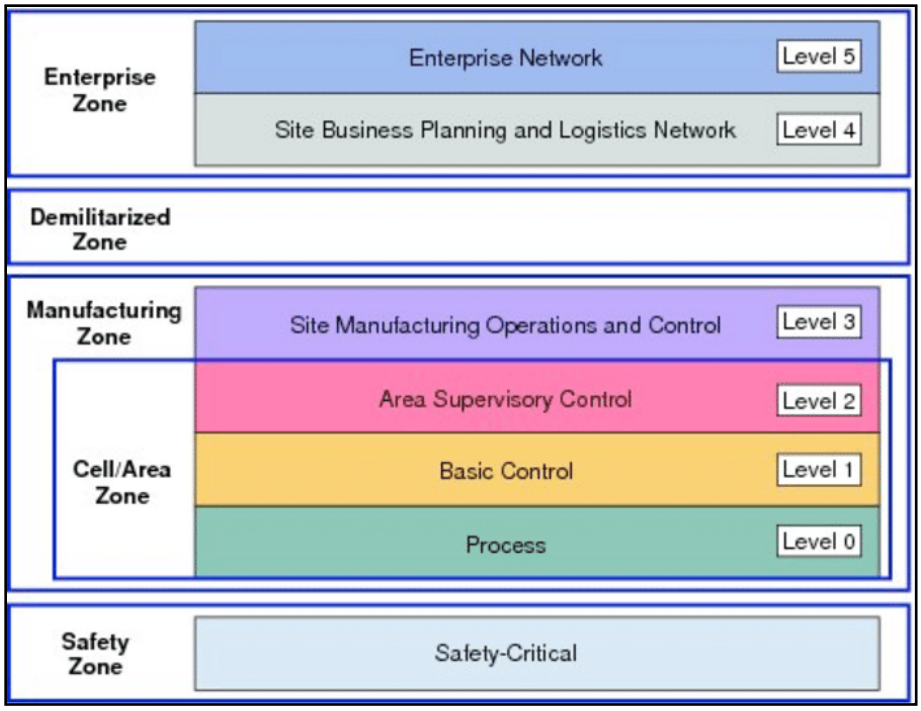

Figure 4. PERA Model. Source: [3].

PERA delineates processes by levels, thereby connecting the business management to the physical process. For the purpose of this thesis, focus is placed exclusively on packet transmission, which occurs between Level 1 and Level 2. In both architectures, Level 1 is comprised of smart meters, intelligent field devices and controllers that are constantly communicating with the SCADA software system, which is encompassed by Level 2. The SCADA Sandbox system architecture, which use the Modbus communications protocol, is shown in Figure 5 where Level 1 is delineated and connected via the Plant LAN, and the SCADA system is clustered in the Control Center LAN. NAVFAC provided the system level documentation and architecture that we use in this thesis. The NAVFAC system architecture is shown in Figure 6, where the communication between Level 1 and Level 2 is shown by the blue-dashed line.

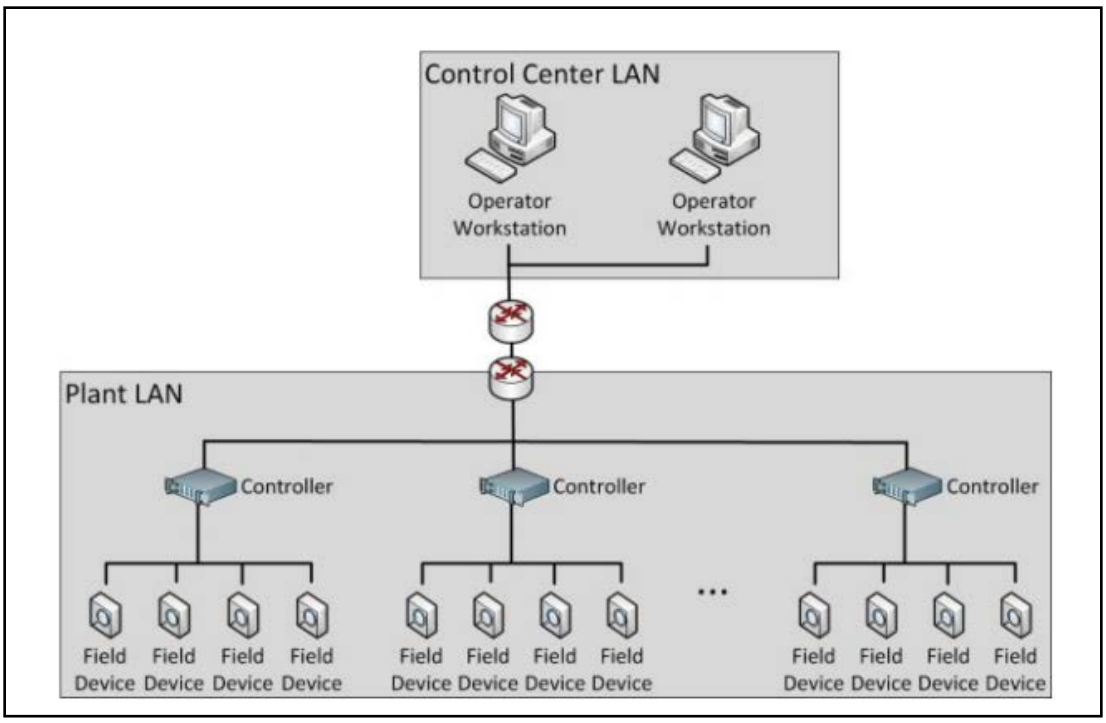

Figure 5. Architecture of a SCADA Sandbox that Uses Modbus Communication. Source: [6].

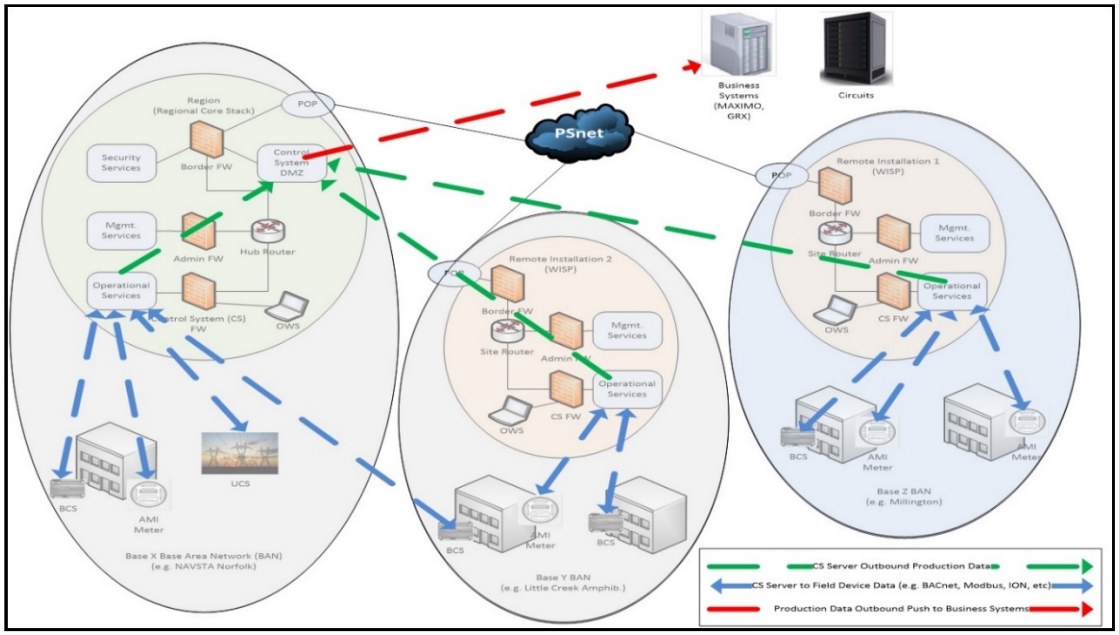

Figure 6. Information Flow of NAVFAC Smart Grid. Source:[3].

The SCADA Sandbox main component is the Master Terminal Unit (MTU) which is responsible for maintaining information about the state of the physical process. A secondary component is the Remote Terminal Unit (RTU), which controls communication with the MTU. In a Modbus communication scheme, this is accomplished by the continuous polling of each smart meter [6]. Modbus protocol is similarly used as one of the NAVFAC Smart Grid implemented protocols; therefore, enough comparison can be drawn to relate analysis of the SCADA Sandbox data sets to theoretical NAVFAC Smart Grid network traffic.

Lemay and Fernandez's research produced eleven SCADA network data sets, six of which contain nominal data and five of which contain malicious attacks. The descriptions of the nominal and malicious data sets are provided in Tables 2 and 3, respectively. Of these data sets, three nominal data sets and four data sets with malicious data were used and are highlighted in yellow in Table 2 and Table 3, respectively.

Table 2. Nominal Data Sets. Source: [6].

Name	Description	Malicious Activity?	Number 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
Run11	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
Run1_12RTU	1 hour of regular Modbus traffic including polling and manual operation - 2 MTU, 12 RTU and 10 seconds polling interval	No	238360
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

Highlighted text indicates nominal data sets were used.

Table 3. Malicious Data Sets. Source: [6].

Name	Description	Malicious Activity?	Number 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 exploit (ms08_netapi) from a compromised RTU to compromise another RTU using Metasploit	Yes	1856

Highlighted text indicates malicious data were used.

The three nominal data sets used as a benchmark were Run8, Run11 and Modbus_polling_only_6RTU. Run8 and Run11, shown in Table 2, each consisted of 1 hour of regular Modbus traffic with a network architecture comprised of 2 MTU, 3 RTU and a 10-second polling interval. Run8 generated 72,186 packets, and Run11 generated 72,489 packets. Modbus_polling_only_6RTU, shown in the last row of Table 2, was

similarly 1 hour in duration but comprised of a network architecture of 1 MTU, 6RTU and a 10-second polling interval and generated 58,325 packets. According to Lemay and Fernandez, the architecture was varied to demonstrate the impact of alternate network traffic configurations [6].

Lemay and Fernandez designed network traffic within each data set to come from recognized machines. This adds complexity to the intrusion detection process and illustrates an attacker launching a cyber-attack from a compromised machine [6]. All malicious data sets consist of regular Modbus traffic and are designed with the same architecture -1 MTU, 6RTU, and a 10-second polling interval. Moving_two_files_Modbus_6RTU, shown in the first row of Table 3, consists of 3 minutes of network traffic containing 3,319 packets. The purpose of malicious data within this data set is for a compromised machine to launch a remote exploit to compromise a second controller [6]. Send_a_fake_command_Modbus_6RTU_with_operate consists of 11 minutes of network traffic containing 11,166 packets. CnC_uploading_exe_modbus_6RTU_with_operate shown in Table 3, consists of 1 minute of network traffic that generated 1,426 packets and contained 121 malicious packets. The purpose of the malicious code within these two data sets is to move files through a Metasploit Meterpreter channel by adding new tools and installing new malware [6]. Metasploit is a penetration testing framework and is the most common framework for hackers to listen and take control of affected machines [10].

6RTU_with_operate consists of 5 minutes of network traffic that generated 1856 packets and contained 1199 malicious packets. Its architecture is designed with 1 MTU, 6RTU, and a 10-second polling interval. The purpose of the malicious code within this data set is to use an exploit from a compromised RTU to comprise another RTU using Metasploit [6]. This final attack type is implemented by sending an unauthorized command to a controller.

B. BAYESIAN MACHINE LEARNING

Supervised machine learning uses labeled training data to determine the function that maps input variables into output variables. This allows us to accurately predict outputs with the implementation of new data. Classification, a type of supervised machine learning,

is adept at accurately predicting the category or class from which the data originated. Of the various classification algorithms used for the purpose of prediction, Naïve Bayes was chosen due to its ease of implementation, its capability to provide real time predictions, scalability and ability to quickly process high dimensional data [11].

Bayes theorem is effective because it determines the probability of a future event based on the knowledge of prior conditional and marginal probabilities. Bayes theorem is given by:

$$\Pr[H|E] = \frac{\Pr[E|H]\cdot\Pr[H]}{\Pr[E]} \tag{3.1}$$

$$\Pr[E] = \Pr[E|H]\cdot\Pr[H] + \Pr[E|\neg H]\cdot\Pr[\neg H] \tag{3.2}$$

where "H" signifies the hypothesis and "E" represent the evidence. In equation 3.1, Pr [H|E] is the desired posterior probability (i.e., probability of nominal data given packet characteristics), and Pr [E] and Pr [H] are the probabilities of the events occurring independently and are known as the marginal probabilities (i.e., probability of the packet characteristics and probability of nominal data, respectively). Conditional probability Pr [E|H] is the overall likelihood or in other words, the conditional probability of the evidence to occur given the hypothesis (i.e., probability of the packet characteristics given nominal data). To scale this equation and implement the selected features shown in Table 4, we use:

$$\Pr[H|E_1, E_2, E_3, E_4] = \frac{\Pr[E_1|H]\cdot\Pr[E_2|H]\cdot\Pr[E_3|H]\cdot\Pr[E_4|H]\cdot\Pr[H]}{\Pr[E_1]\cdot\Pr[E_2]\cdot\Pr[E_3]\cdot\Pr[E_4]} \tag{3.3}$$

where E_1, E_2, E_3, and E_4 represents each feature, respectively.

Table 4. Features

Feature 1	Source Address
Feature 2	Destination Address
Feature 3	Packet Protocol
Feature 4	Packet Length

C. PACKET PROCESSING

Each data set was first viewed in Wireshark and then formatted into a workable MATLAB file. All figures and tables provided in this section were derived from and utilized Lemay and Fernandez's data set titled 6RTU_with_operate. We use this data set as an example to illustrate our methodology. Figure 7 illustrates the first packet sent and analyzed within this data set.

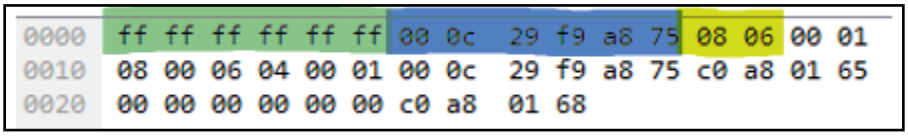

Figure 7. Packet Example. Source: [8].

Each packet was separated by the four features shown in Table 4: Packet Protocol, Source Address, Destination Address and Packet Length. As can be seen in Figure 7, the Source Address is highlighted in green and the Destination Address is highlighted in blue. Packet Protocol is delineated in bytes 13 through 14 and are highlighted in yellow. The packet shown in Figure 7 is 42 bytes long. To separate the varied packet lengths into manageable categories, they were split into increments of 400 bytes as shown in Table 5.

Table 5. Feature Separation by Length

Bin	Lower Limit	Upper Limit
1	0	400
2	401	800
3	801	1200
4	1201	1600
5	1601	2000

Using the example provided in Figure 7, the processed packet would be analyzed based on the following features and are shown in Figure 8.

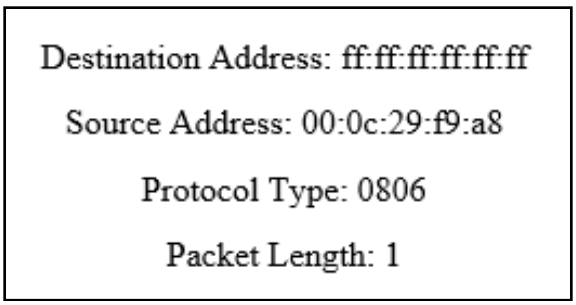

Figure 8. Packet Features Example

A limitation to Bayesian supervised machine learning is that training data must represent the test data, meaning that each feature must occur in training data. Each feature was comprised of numerous unique parameters; the unique parameters of the 6RTU_with_operate data set are shown in Table 6.

Table 6. 6RTU_with_operate Feature Parameters

Destination Address	Source Address	Protocol Type	Packet Length
000C293C113F	000C293C113F	0800	1
000C2958972A	000C2958972A	0806	2
000C296EBA5E	000C296EBA5E		3
000C2970AAB7	000C2970AAB7		4
000C29DC42E5	000C29DC42E5		5
000C29EEB784	000C29EEB784		
000C29F9A875	000C29F9A875		
FFFFFFFFFFFF			

The number of packets within each data set varied; and, the number of packets in the training data was a percentage of the number of packets in the entire data set. Each data set came with a corresponding Excel document that provided whether the packet was malicious or nominal. Malicious packets were labeled with a "1," and nominal packets were labeled with a "0." Both the data set and labeled outcome were separated into test data and trial data based on the training data percentage. Regarding only the training data,

Table 7 represents the probability of a packet to either be nominal or malicious. This is represented in Bayes theorem as Pr [H].

Table 7. 6RTU_with_operate Training Data Probability

	Nominal Packets (%) Pr [H]	Malicious Packets (%) Pr [¬H]
6RTU_with_operate	39.93	60.07

Referencing the known outcomes for the training data, shown in Table 7, the conditional probability of each feature occurring in either a nominal or malicious packet was calculated. This represents the overall likelihood and is represented in Bayes theorem as Pr [E|H]. Furthermore, this is illustrated in Tables 8–11 with regards to Source Address, Destination Address, packet protocol and packet length respectively.

Table 8. 6RTU_with_operate Source Address Probability

Source Address	Nominal Probability (%) Pr [E_1|H]	Malicious Probability (%) Pr [E_1|¬H]
000C293C113F	4.32	0
000C2958972A	5.04	0
000C296EBA5E	4.32	0
000C2970AAB7	4.68	18.35
000C29DC42E5	4.32	0
000C29EEB784	32.37	0
000C29F9A875	5.40	21.58

Table 9. 6RTU_with_operate Destination Address Probability

Destination Address	Nominal Probability (%) Pr [E_2|H]	Malicious Probability (%) Pr [E_2|¬H]
000C293C113F	5.40	0
000C2958972A	5.40	0
000C296EBA5E	5.40	0
000C2970AAB7	6.12	21.58
000C29DC42E5	5.40	0
000C29EEB784	25.90	0
000C29F9A875	5.40	18.35
FFFFFFFFFFFF	1.08	0

Table 10. 6RTU_with_operate Protocol Probability

Protocol	Nominal Probability (%) Pr [E_3|H]	Malicious Probability (%) Pr [E_3|¬H]
0800	58.63	39.93
0806	1.44	0

Table 11. 6RTU_with_operate Length Probability

Length	Nominal Probability (%) Pr [E_4|H]	Malicious Probability (%) Pr [E_4|¬H]
$0 \leqq L \angle 400$	60.07	35.25
$400 \leqq L \angle 800$	0	2.88
$800 \leqq L \angle 1200$	0	0
$1200 \leqq L \angle 1600$	0	1.80
$1600 \leqq L \angle 2000$	0	0

For each packet, the posterior probability was calculated for both nominal and malicious assumptions using:

$$\Pr[H|E_1, E_2, E_3, E_4] = \frac{\Pr[E_1|H]\cdot\Pr[E_2|H]\cdot\Pr[E_3|H]\cdot\Pr[E_4|H]\cdot\Pr[H]}{\Pr[E_1]\cdot\Pr[E_2]\cdot\Pr[E_3]\cdot\Pr[E_4]} \quad (3.4)$$

$$\Pr[\neg H|E_1, E_2, E_3, E_4] = \frac{\Pr[E_1|\neg H]\cdot\Pr[E_2|\neg H]\cdot\Pr[E_3|\neg H]\cdot\Pr[E_4|\neg H]\cdot\Pr[\neg H]}{\Pr[E_1]\cdot\Pr[E_2]\cdot\Pr[E_3]\cdot\Pr[E_4]} \quad (3.5)$$

The training data was then used to calculate posterior probabilities for both nominal and malicious packets. The larger probability was chosen for classification. Our results are shown and analyzed in the following chapter.

IV. RESULTS

A Bayesian classification and machine learning program was designed to process all data provided by Lemay and Fernandez, highlighted in Tables 1 and 2 of Chapter III. This chapter discusses the results of the data sets processed based on the methods given in Chapter III.

A. DATA ANALYSIS

Based on Bayes theorem as discussed in Chapter III, the posterior probability is calculated using the probabilities of prior events. The training set was processed, and the conditional probability of each feature was calculated and used to predict the posterior probability for the first packet of the trial data. The program classification was recorded and then compared to the actual classification. Once recorded, the individual feature probabilities that comprised the packet were incorporated into the training data, allowing the algorithm to further learn as more data was processed

For the nominal data sets, this means that the designed program should be 100% effective, as it cannot classify a packet as malicious if there are no prior malicious packets. For the data sets that contain malicious packets, the designed program will classify malicious packets based on the probability of occurrence. Since the packets were simulated to come from within the trusted network, the effectiveness of the program will be based off the location of the malicious packets and the ratio of malicious packets to the size of the data set.

B. SIMULATION RESULTS

The nominal data sets were used to demonstrate that the designed program was working as intended. The program was then used to process the data sets that contained malicious packets. The following definitions represent each column of data that is represented in the tables of results shown in this chapter.

- Training Data

    1. Percent (%): Percent of the overall data set that was included in the training data. For nominal data sets, training data percentage was increased by increments of 5% from 5% to 30% due to the expectation of 100% prediction accuracy (the program would not be able to classify malicious packets if none existed, thereby 100% prediction accuracy), For data sets with malicious packets, training data percentage was increased by increments of 1% from 1% to 30% in order to show that by adding one packet to the training data, classification could change.
    2. Number: The number of packets within the training data based on the training packet percent.
    3. Malicious Packets: The number of malicious packets within the training data.

- Trial Data

    4. Number: The remaining packets within the data set after the training data was processed.
    5. Malicious Packets: The number of malicious packets within the trail data.
    6. Errors: The number of packets that were classified incorrectly.
    7. Percent Error: The ratio of trial data errors to the number of packets within the trial data.

The results presented in the following sections are broken down as follows: The Run8, Run11 and Modbus_polling_only_6RTU are the nominal data sets that are the highlighted rows shown in Table 2 in Chapter III. The Moving_two_files_Modbus, the Send_a_fake_command_Modbus_6RTU_with_operate, the

CnC_uploading_exe_Modbus_6RTU_with_operate and the 6RTU_with_operate are the malicious data sets and are the highlighted rows shown in Table 3 in Chapter III.

1. Run8

Run8 consisted of 72,186 packets, none of which were malicious. The architecture of this data set was comprised of 2 MTU, 3RTU and a 10-second polling interval. The Bayesian Classification and Machine Learning Program predictions were 100% accurate as shown in Table 1.

Table 12. Run8

Training Data		Trial Data		
Percent	**Number**	**Number**	**Errors**	**Percent Error**
5%	3609	68577	0	0.00%
10%	7219	64967	0	0.00%
15%	10828	61358	0	0.00%
20%	14437	57749	0	0.00%
25%	18047	54139	0	0.00%
30%	21656	50530	0	0.00%

2. Run11

Run11 consisted of 72,498 packets, none of which were malicious. The architecture of this data set was comprised of 2 MTU, 3RTU and a 10-second polling interval. The Bayesian Classification and Machine Learning Program predictions were 100% accurate as shown in Table 2.

Table 13. Run11

Training Data		Trial Data		
Percent	**Number**	**Number**	**Errors**	**Percent Error**
5%	3625	68873	0	0.00%
10%	7250	65248	0	0.00%
15%	10875	61623	0	0.00%
20%	14500	57998	0	0.00%
25%	18125	54373	0	0.00%
30%	21749	50749	0	0.00%

3. Modbus_polling_only_6RTU

Modbus_polling_only_6RTU consisted of 58,325 packets, none of which were malicious. The architecture of this data set was comprised of 1 MTU, 3RTU and a 10-second polling interval. The Bayesian Classification and Machine Learning Program predictions were 100% accurate as shown in Table 3.

Table 14. Modbus_polling_only_6RTU

Training Data		Trial Data		
Percent	**Number**	**Number**	**Errors**	**Percent Error**
5%	2916	55409	0	0.00%
10%	5833	52492	0	0.00%
15%	8749	49576	0	0.00%
20%	11665	46660	0	0.00%
25%	14581	43744	0	0.00%
30%	17498	40827	0	0.00%

4. Moving_two_files_Modbus_6RTU

Moving_two_files_Modbus_6RTU contained 3,319 packets, 75 of which were malicious packets and 3,244 which were nominal packets. Malicious packets made up 2.2% of the data set with the attack starting at packet 325 as shown in Figure 9. Figure 9 through Figure 13 illustrate the malicious packet distribution within the data set, where “0”

represents a nominal packet and a “1” represents a malicious packet. This data set was used in order to determine if a small number of malicious packets at the beginning of the data set would allow for successful predictions.

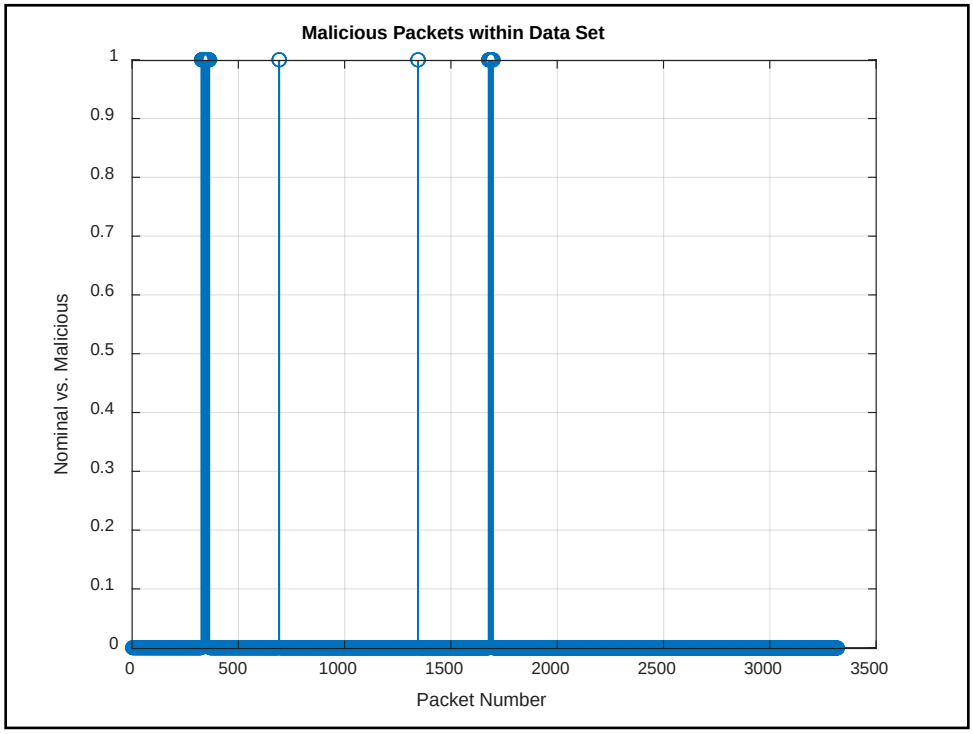

Figure 9. Malicious Packets Distribution within Moving_two_files_Modbus_6RTU

This proved unsuccessful; malicious packets incorporated into the training data were still classified as nominal. Due to the fact that the malicious packets were generated from a trusted source, by introducing a relatively small number of malicious packets, the program still identified these malicious packets as nominal because it is more probable.

5. Send_a_fake_command_Modbus_6RTU_with_operate

Send_a_fake_command_Modbus_6RTU_with_operate contained 11,166 packets of which 10 packets were malicious and 11,156 were nominal packets. The architecture of this data set was comprised of 1 MTU, 6RTU and a 10-second polling interval. Malicious packets made up 0.09% of the data set with the attack starting at packet 4,782 as shown in

Figure 10. This data set was used to determine if correct classification was possible for a brief singular attack.

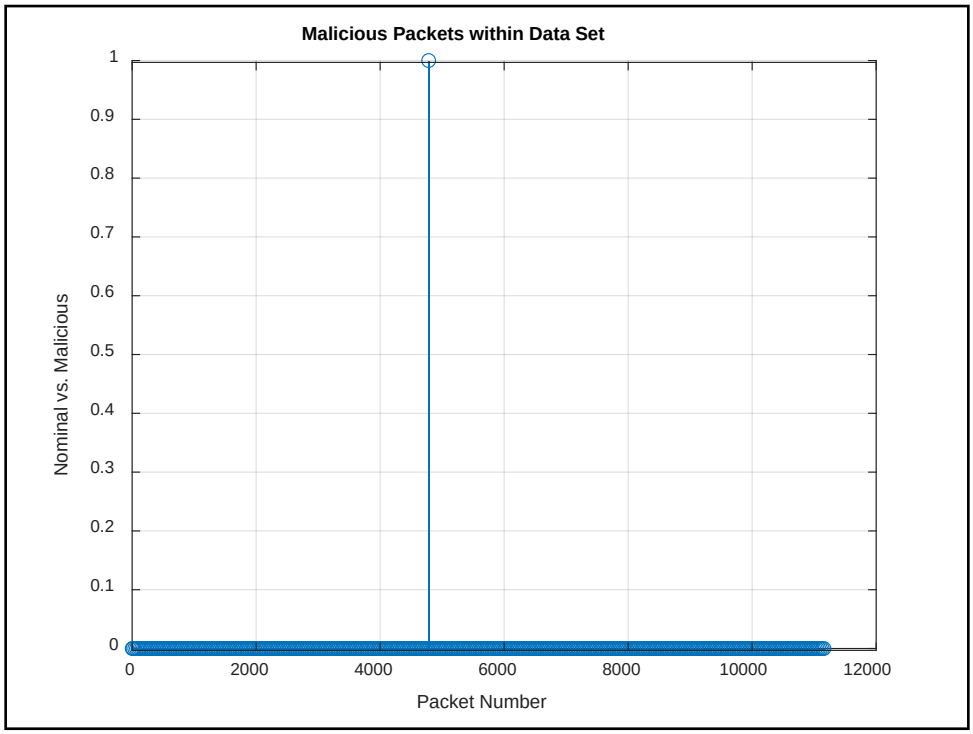

Figure 10. Send_a_fake_command_Modbus_6RTU_with_operate Malicious Packets Distribution within Data Set

Again, this proved unsuccessful, because the malicious packets were generated from a trusted source. By introducing a relatively small number of malicious packets, the program still identified these malicious packets as nominal because it is more probable.

6. CnC_uploading_exe_modbus_6RTU_with_operate

CnC_uploading_exe_modbus_6RTU_with_operate contained 1,426 packets of which 121 were malicious and 1,305 were nominal packets. The architecture of this data set was comprised of 1 MTU, 6RTU and a 10-second polling interval. Malicious packets made up 8.4% of the data set with the attack starting at packet 811 as shown in Figure 11. This data set was analyzed to determine if correct classification was possible if there were no instances of malicious packets within the training data.

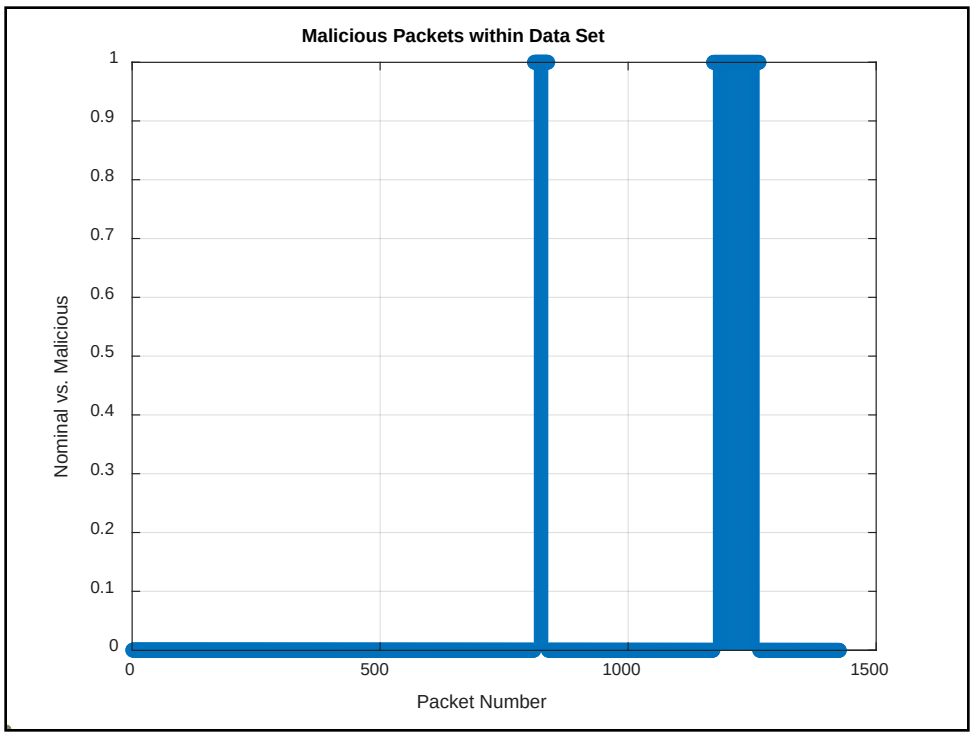

Figure 11. CnC_uploading_exe_modbus_6RTU_with_operate Malicious Packets Distribution within Data Set

This proved unsuccessful for training data percentage less than 30% as the program classified all malicious data as nominal. The training data percentage needed to be increased to 57% of the data set for any success of correctly classifying malicious packets. This is shown in Table 15.

Table 15. CnC_uploading_exe_modbus_6RTU_with_operate Trial Data Classification Error vs. Training Data Size

Training Data			Trial Data			
Percent	**Number**	**Malicious Packets**	**Number**	**Malicious Packets**	**Errors**	**Percent Error**
55%	784	0	642	121	121	18.85%
56%	799	0	627	121	121	19.30%
57%	813	3	613	119	69	11.26%
58%	827	17	599	105	55	9.18%
59%	841	28	585	94	43	7.35%
60%	856	28	570	93	43	7.54%

7. 6RTU_with_operate

6RTU_with_operate contained 1,856 packets of which 1,199 were malicious and 657 were nominal packets. The architecture of this data set was comprised of 1 MTU, 6RTU and a 10-second polling interval. The malicious packets comprised of 64% of the data set, and they were more evenly distributed as shown in Figure 12.

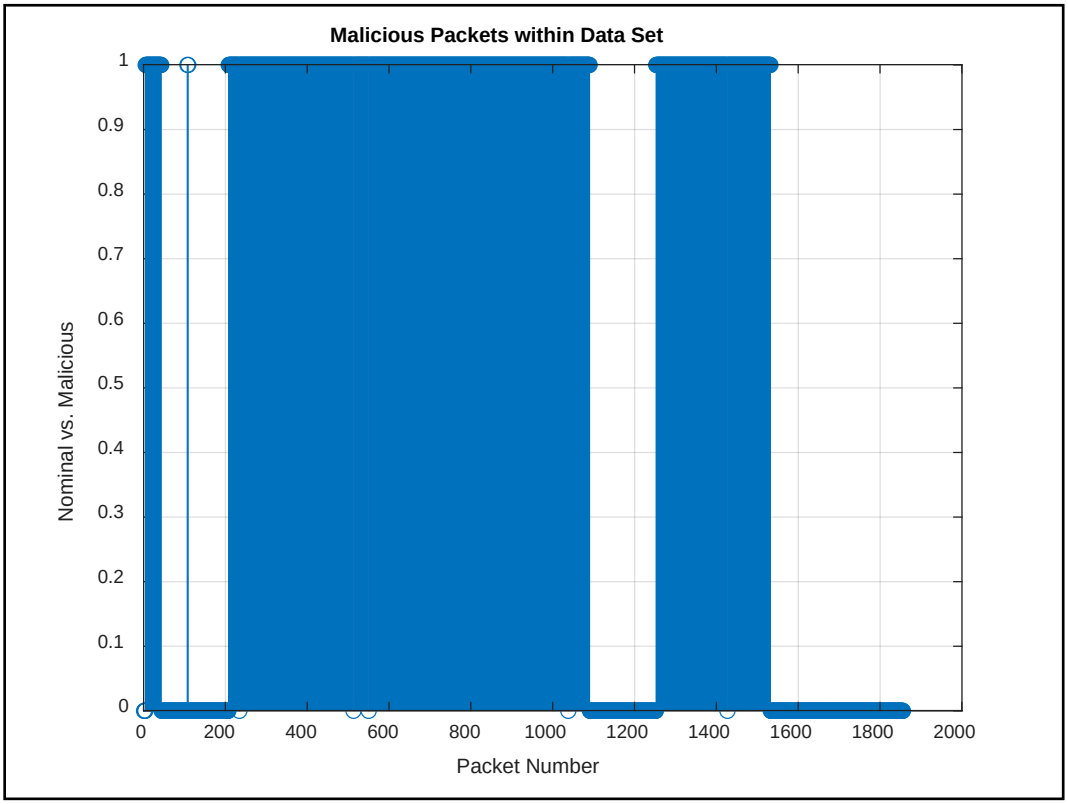

Figure 12. 6RTU_with_operate Malicious Packets Distribution within Data Set

The even distribution facilitated more accurate predictions. Since packets at the start of the data set were malicious, the initial prediction error rate for the remaining data was low. As nominal packets were introduced to the training data, the error rate increased. As malicious packets were again incorporated into the training set, the prediction error rate decreased to less than 1% as shown in Table 16.

Table 16. 6RTU_with_operate Trial Data Classification Error vs. Training Data Size

Training Data			Trial Data			
Percent	**Number**	**Malicious Packets**	**Number**	**Malicious Packets**	**Errors**	**Percent Error**
1%	19	15	1837	1185	11	0.60%
2%	37	33	1819	1167	2	0.11%
3%	56	39	1800	1161	18	1.00%
4%	74	39	1782	1161	29	1.63%
5%	93	39	1763	1161	37	2.10%
6%	111	41	1745	1159	41	2.35%
7%	130	41	1726	1159	44	2.55%
8%	148	41	1708	1159	48	2.81%
9%	167	41	1689	1159	50	2.96%
10%	186	41	1670	1159	53	3.17%
11%	204	41	1652	1159	56	3.39%
12%	223	57	1633	1143	35	2.14%
13%	241	74	1615	1126	13	0.80%
14%	260	93	1596	1107	2	0.13%
15%	278	111	1578	1089	1	0.06%
16%	297	130	1559	1070	1	0.06%
17%	316	149	1540	1051	1	0.06%
18%	334	167	1522	1033	1	0.07%
19%	353	186	1503	1014	1	0.07%
20%	371	204	1485	996	1	0.07%
21%	390	223	1466	977	1	0.07%
22%	401	241	1455	959	1	0.07%
23%	427	260	1429	940	1	0.07%
24%	445	278	1411	922	1	0.07%
25%	464	297	1392	903	1	0.07%
26%	483	316	1373	884	1	0.07%
27%	501	334	1355	866	1	0.07%
28%	520	352	1336	848	1	0.07%
29%	538	370	1318	830	1	0.08%
30%	557	388	1299	812	1	0.08%

C. SUMMARY OF ANALYSIS

The nominal data sets were processed and classified as expected. With 100% accuracy, they were used to determine the effectiveness of the program. The Moving_two_files_Modbus_6RTU data set introduced malicious data in small quantities towards the beginning of the data set, and because the packets were sent from within a trusted network, they were incorrectly classified as nominal. This is because the distinguishing features have a higher probability of being classified nominal than malicious. Send_a_fake_command_Modbus_6RTU_with_operate incorporated a small number of malicious packets in a singular event approximately halfway through the data set. Again, effective classification of malicious packets was unsuccessful due to the distinguishing features have a higher probability of being classified nominal than malicious. Characterization_uploading_exeModbus_6RTU_with_operate introduced malicious packets at the end of the set. Because these malicious packets could not be included into a reasonable training data size, the classification success of the program was deemed partially effective.

6RTU_with_operate performed the best out of the data sets that contained malicious packets. With a reasonable training data percentage of 13%, the classification error drops to below 1%. Figure 13 shows that as more nominal and malicious data are incorporated, the more effective classification becomes.

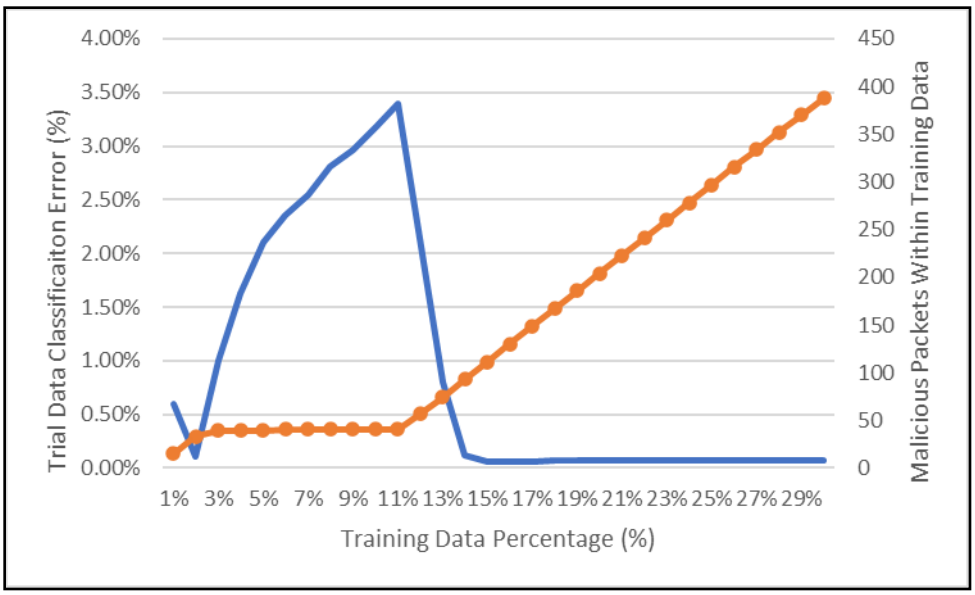

Figure 13. Trial Data Classification Error vs. Malicious Packets within Training Data

Based on the results obtained, we can state that NAVFAC can employ Bayesian classification and machine learning as a tool for intrusion detection. Using the approach developed in this thesis, cyber-attacks sending high volumes of malicious packets can be identified with high accuracy and in real-time to immediately address the intrusion. This will allow NAVFAC to continue running an efficient power grid while minimizing the risks associated with the cyber domain.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSION

This thesis is intended to provide a reference point for future designers to build successful machine learning programs for the purpose of intrusion detection. Understanding the threat of one malicious packet going undetected, as well as the difficulty of accurately classifying that malicious packet within an endless stream of nominal network traffic, highlights the necessity to utilize machine learning.

A. SUMMARY

The Navy has expanded its attack surface by meeting the Secretary of the Navy's mandates and introducing NAVFAC smart grids across the fleet. To minimize risk as power efficiency is maximized, machine learning must be utilized to protect secure networks and sensitive infrastructure. The data sets used in this thesis represent probable scenarios that an intrusion detection system needs to be able to identify in order to maintain a secure network. This thesis has demonstrated the effectiveness of Bayesian classification and machine learning using MATLAB for the purpose of intrusion detection.

Through the designed program, the success of Bayesian classification and machine learning was evaluated through 30 different data configurations run on 7 data sets. The classification outcome was more dependent on the quality of data set than originally thought. However, machine learning is very successful with respect to data sets with high malicious traffic as shown in the results for data set 6TRU_with_operate. The results will help future researchers to choose data sets and features as well as design a more efficient program that will generate faster and more desirable results. As the Navy, and NAVFAC specifically, expands smart grids to every base and begins to integrate smart technology while updating additional infrastructure, continued research and study on machine learning for network intrusion detection is required.

B. FUTURE WORK

(1) NAVFAC Data Set

Because we were processing a similar SCADA data set and not network traffic provided from the NAVFAC smart grid, our findings can only be inferred. The use of machine learning algorithms to process the data of study would generate the best results.

(2) Increased Features

Packet processing consisted of evaluating each packet based on Source Address, Destination Address, Packet Protocol and Packet length. By increasing the amount of features the probability of detecting a malicious packet will increase and would provide more accurate classification.

(3) Alternate Supervised Machine Learning Algorithms

While we based our machine learning algorithm off Bayes theorem, there are several other machine learning algorithms that may be successful in classification. Using the same data sets and comparing the effectiveness of alternate algorithms may allow for the most advantageous choice of algorithm for intrusion detection.

(4) supervised Machine Learning

Unsupervised machine learning operates with minimum personnel interaction and identifies patterns within a data set before labeling the patterns as features or classification groups. This would allow the machine learning algorithm to identify a greater number of features, as well as classifying new features as network traffic shifts and increases.

APPENDIX A. MAIN SCRIPT

The following script encompasses all functions that process the data set from processing to prediction.

```
fileID = fopen('moring_two_files_modbus_6RTU_txt.txt');
data=fread(fileID,'*char').';
[Number, Package] = Format_Data(data);

n = length(Number);
burnin = round(0.60*n)

fileID = fopen('moving_two_files_modbus_6RTU_labeled.csv');
label=fread(fileID,'*char').';
[ET,ER,Error] = labelerror(label,burnin,n);

[Source, Destination, Protocol, Length, SQN, ACK] =
Data_Characteristics(Package);
NT = Number(1:burnin,:); NR = Number(burnin:n,:);
ST = Source(1:burnin,:); SR = Source(burnin:n,:);
DT = Destination(1:burnin,:); DR = Destination(burnin:n,:);
LT = Length(1:burnin,:); LR = Length(burnin:n,:);
PT = Protocol(1:burnin,:); PR = Protocol(burnin:n,:);

t=table(Number,Package,Error);
Test = table(NT,ST,DT,LT,PT);

%TRAINING DATA
[O,OP,UL,LT,lnorm,lmal,UD,dnorm,dmal,US,snorm,smal,UP,pnorm,pmal] =
probability(ET, LT, DT, ST, PT);
[proboferrorsT,numoferrorsT] =
outcome(ST,US,snorm,smal,DT,UD,dnorm,dmal,UP,PT,pnorm,pmal,LT,lnorm,lmal,O
P,ET,NT)

%TEST THE REMAINING OF THE DATA
[N,M,outcome,comparer] =
rundataML(SR,US,snorm,smal,DR,UD,dnorm,dmal,PR,UP,pnorm,pmal,LR,lnorm,lmal
,OP,ER,burnin);

stem(comparer)
hold on
plot(ER,'o')
legend('Comparer','ER')

testerrors = sum(comparer(:) == 1)
testerrorpercent = testerrors/(n-burnin)
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. PROCESS SCRIPT

The following script processes the package information taken from Wireshark and formats each package for use in MATLAB.

```
function [Number, Package] = Format_Data(data)

data=string(data);
data=splitlines(data)';

l=round(length(data)/4);
time=string(zeros(l,1));
package=string(zeros(l,1));

for ix=1:l
 time(ix)=data((ix*4)-2);
 package(ix)=data((ix*4)-1);
end

%TIME-------------------------------
time=erase(time,'ETHER');
time=erase(time,' ');
time=erase(time,',');
splittime=split(time,':');
TIME=str2double(splittime);

Number=1:l;
Number=Number.';

%PACKAGE----------------------------
package=erase(package,'|0   |');
m=max(strlength(package));

C=zeros(l,m);
for i=1:l
 a=split(package(i,1),'|').';
 ll=length(a);
 B=char(a(:));
 B=hex2dec(B).';
 lll=length(B);
 for ii=1:lll
     C(i,ii)=B(1,ii);
 end
end

a=char(zeros(l,m));
d=dec2hex(C);
```

```
ld=length(d);

for x=1:(m/2)
 a(1:l,x*2-1:x*2)=d((1+(l*(x-1)):((l*(x-1))+l)),:);
end
Package=a;

end
```

APPENDIX C. LABEL ERROR SCRIPT

The following script formats the Xcel spreadsheet, provided by the data set developers, that correlates package number to whether the package is nominal or malicious.

```
function [errortest,errorrun,Error] = labelerror(label,burnin,n)

l=erase(label,';');

l=str2num(l);
N=length(l);

error=zeros(N,1);

for ii=1:N
 error(ii,1) = l(ii,1)-ii*10;

end
Error = error;
stem(error)
title('Malicious Packets within Data Set')
xlabel('Packet Number')
ylabel('Nominal vs. Malicious')
grid on

totalerrors = sum(error(:) == 1)

errortest = error(1:burnin,:);
trainingerrors = sum(errortest(:) == 1)
errorrun = error(burnin:n,:);
runerrors = sum(errorrun(:) == 1)
end
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. DATA CHARACTERISTICS SCRIPT

The following script separates each package into the following useable features: source address, destination address, packet protocol and packet length.

```
function [Source_Address,Destination_Address,Protocol_Type,ll,SQN,ACK] =
Data_Characteristics(P)

%Physical Address
Destination_Address = P(:,1:12);
Source_Address = P(:,13:24);
Protocol_Type = P(:,(25:28));

l=size(P);
l=l(1,1);

h=hex2dec(P(:,33:36));
s=hex2dec(P(:,77:84));
a=hex2dec(P(:,85:92));

Total_Length = zeros(l,1);
SQN = zeros(l,1);
ACK = zeros(l,1);

for ii=1:l
 if P(ii,(25:28)) == '0800'
     Total_Length(ii,:) = h(ii,:);
     SQN(ii,:) = s(ii,:);
     ACK(ii,:) = a(ii,:);
 elseif P(ii,(25:28)) == '0806' %ARP
     Total_Length(ii,:) = 42;
     SQN(ii,:) = 0;
     ACK(ii,:) = 0;
 else
     Total_Length(ii,:) = 0; %NOT ARP OR TCP
     SQN(ii,:) = 0;
     ACK(ii,:) = 0;
 end
end

ll=Total_Length;
tt=table(Protocol_Type , Destination_Address , Source_Address );
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E. PROBABILITY SCRIPT

The following script determines the probability of each feature in the training data.

```
function [O,OP,UL,l,lnorm,lmal,UD,dnorm,dmal,US,snorm,smal,UP,pnorm,pmal]
= probability(Error, Length, Destination, Source, Protocol)

n = length(Length);

%Probability of Outcome
e = sum(Error(:) == 1); %count amount of malicous packets
pmal = e/n; %probabiliy of malicious packet P(theta)
pnorm = 1-pmal;
O = [“malicious”; “normal”];
OP = [pmal ; pnorm];
T1 = table(O,OP);
%Probability of Length----------------------------------------
L=Length;
l = zeros(n,1);
for i = 1:n
 if L(i,:) >= 0 && L(i,:) < 400
     if Error(i,:) == 0
         l(i,:) = 1;
     else
         l(i,:) = 2;
     end
 elseif L(i,:) >= 400 && L(i,:) < 800
     if Error(i,:) == 0
         l(i,:) = 3;
     else
         l(i,:) = 4;
     end
 elseif L(i,:) >= 800 && L(i,:) < 1200
     if Error(i,:) == 0
         l(i,:) = 5;
     else
         l(i,:) = 6;
     end
 elseif L(i,:) >= 1200 && L(i,:) < 1600
     if Error(i,:) == 0
         l(i,:) = 7;
     else
         l(i,:) = 8;
     end
 elseif L(i,:) >= 1600 && L(i,:) < 2000
     if Error(i,:) == 0
         l(i,:) = 9;
     else
```

```
        l(i,:) = 10;
    end
 else
    l(i,:)=0;
 end
end

LP0= sum(l(:) == 0);
LP10 = sum(l(:) == 1)/n;
LP11 = sum(l(:) == 2)/n;
LP20 = sum(l(:) == 3)/n;
LP21 = sum(l(:) == 4)/n;
LP30 = sum(l(:) == 5)/n;
LP31 = sum(l(:) == 6)/n;
LP40 = sum(l(:) == 7)/n;
LP41 = sum(l(:) == 8)/n;
LP50 = sum(l(:) == 9)/n;
LP51 = sum(l(:) == 10)/n;

lnorm = [LP10;LP20;LP30;LP40;LP50];
lmal = [LP11;LP21;LP31;LP41;LP51;];
UL = ["0<l<400";"400<l<800";"800<l<1200";"1200<l<1600";"1600<l<2000"];
T2 = table(UL,lnorm,lmal);

%Probability of Destination-----------------------------------
D = string(Destination);
UDu = unique(D);
x = size(UDu);
x = x(1,1);

UD = string(zeros(16,1));
UD(1:x,:) = UDu;

dp = zeros(n,1);
for i=1:n
 if D(i,:) == UD(1,:)
    if Error(i,1) == 0
        dp(i,1) = 1;
    else
        dp(i,1) = 2;
    end
 elseif D(i,:) == UD(2,:)
    if Error(i,1) == 0
        dp(i,1) = 3;
    else
        dp(i,1) = 4;
    end
 elseif D(i,:) == UD(3,:)
    if Error(i,1) == 0
```

```
        dp(i,1) = 5;
    else
        dp(i,1) = 6;
    end
elseif D(i,:) == UD(4,:)
    if Error(i,1) == 0
        dp(i,1) = 7;
    else
        dp(i,1) = 8;
    end
elseif D(i,:) == UD(5,:)
    if Error(i,1) == 0
        dp(i,1) = 9;
    else
        dp(i,1) = 10;
    end
elseif D(i,:) == UD(6,:)
    if Error(i,1) == 0
        dp(i,1) = 11;
    else
        dp(i,1) = 12;
    end
elseif D(i,:) == UD(7,:)
    if Error(i,1) == 0
        dp(i,1) = 13;
    else
        dp(i,1) = 14;
    end
elseif D(i,:) == UD(8,:)
    if Error(i,1) == 0
        dp(i,1) = 15;
    else
        dp(i,1) = 16;
    end
elseif D(i,:) == UD(9,:)
    if Error(i,1) == 0
        dp(i,1) = 17;
    else
        dp(i,1) = 18;
    end
elseif D(i,:) == UD(10,:)
    if Error(i,1) == 0
        dp(i,1) = 19;
    else
        dp(i,1) = 20;
    end
elseif D(i,:) == UD(11,:)
    if Error(i,1) == 0
        dp(i,1) = 21;
```

```
    else
        dp(i,1) = 22;
    end
elseif D(i,:) == UD(12,:)
    if Error(i,1) == 0
        dp(i,1) = 23;
    else
        dp(i,1) = 24;
    end
elseif D(i,:) == UD(13,:)
    if Error(i,1) == 0
        dp(i,1) = 25;
    else
        dp(i,1) = 26;
    end
elseif D(i,:) == UD(14,:)
    if Error(i,1) == 0
        dp(i,1) = 27;
    else
        dp(i,1) = 28;
    end
elseif D(i,:) == UD(15,:)
    if Error(i,1) == 0
        dp(i,1) = 29;
    else
        dp(i,1) =30;
    end
elseif D(i,:) == UD(16,:)
    if Error(i,1) == 0
        dp(i,1) = 31;
    else
        dp(i,1) = 32;
    end
else
    dp(i,1) = 0;
end
end

DP0= sum(dp(:) == 0);
DP10 = sum(dp(:) == 1)/n;
DP11 = sum(dp(:) == 2)/n;
DP20 = sum(dp(:) == 3)/n;
DP21 = sum(dp(:) == 4)/n;
DP30 = sum(dp(:) == 5)/n;
DP31 = sum(dp(:) == 6)/n;
DP40 = sum(dp(:) == 7)/n;
DP41 = sum(dp(:) == 8)/n;
DP50 = sum(dp(:) == 9)/n;
DP51 = sum(dp(:) == 10)/n;
```

```
DP60 = sum(dp(:) == 11)/n;
DP61 = sum(dp(:) == 12)/n;
DP70 = sum(dp(:) == 13)/n;
DP71 = sum(dp(:) == 14)/n;
DP80 = sum(dp(:) == 15)/n;
DP81 = sum(dp(:) == 16)/n;
DP90 = sum(dp(:) == 17)/n;
DP91 = sum(dp(:) == 18)/n;
DP100 = sum(dp(:) == 19)/n;
DP101 = sum(dp(:) == 20)/n;
DP110 = sum(dp(:) == 21)/n;
DP111 = sum(dp(:) == 22)/n;
DP120 = sum(dp(:) == 23)/n;
DP121 = sum(dp(:) == 24)/n;
DP130 = sum(dp(:) == 25)/n;
DP131 = sum(dp(:) == 26)/n;
DP140 = sum(dp(:) == 27)/n;
DP141 = sum(dp(:) == 28)/n;
DP150 = sum(dp(:) == 29)/n;
DP151 = sum(dp(:) == 30)/n;
DP160 = sum(dp(:) == 31)/n;
DP161 = sum(dp(:) == 32)/n;

dnorm                                                                    =
[DP10;DP20;DP30;DP40;DP50;DP60;DP70;DP80;DP90;DP100;DP110;DP120;DP130;DP14
0;DP150;DP160];
dmal                                                                     =
[DP11;DP21;DP31;DP41;DP51;DP61;DP71;DP81;DP91;DP101;DP111;DP121;DP131;DP14
1;DP151;DP161];
T3 = table(UD,dnorm,dmal);

%Probagility of Source-----------------------------------
S = string(Source);
USu = unique(S);
x = size(USu);
x = x(1,1);

U.S. = string(zeros(10,1));
US(1:x,:) = USu;

sp = zeros(n,1);
for i=1:n
 if S(i,:) == US(1,:)
     if Error(i,1) == 0
         sp(i,1) = 1;
     else
         sp(i,1) = 2;
     end
 elseif S(i,:) == US(2,:)
```

```
    if Error(i,1) == 0
        sp(i,1) = 3;
    else
        sp(i,1) = 4;
    end
elseif S(i,:) == US(3,:)
    if Error(i,1) == 0
        sp(i,1) = 5;
    else
        sp(i,1) = 6;
    end
elseif S(i,:) == US(4,:)
    if Error(i,1) == 0
        sp(i,1) = 7;
    else
        sp(i,1) = 8;
    end
elseif S(i,:) == US(5,:)
    if Error(i,1) == 0
        sp(i,1) = 9;
    else
        sp(i,1) = 10;
    end
elseif S(i,:) == US(6,:)
    if Error(i,1) == 0
        sp(i,1) = 11;
    else
        sp(i,1) = 12;
    end
elseif S(i,:) == US(7,:)
    if Error(i,1) == 0
        sp(i,1) = 13;
    else
        sp(i,1) = 14;
    end
elseif S(i,:) == US(8,:)
    if Error(i,1) == 0
        sp(i,1) = 15;
    else
        sp(i,1) = 16;
    end
elseif S(i,:) == US(9,:)
    if Error(i,1) == 0
        sp(i,1) = 17;
    else
        sp(i,1) = 18;
    end
elseif S(i,:) == US(10,:)
    if Error(i,1) == 0
```

```
        sp(i,1) = 19;
    else
        sp(i,1) = 20;
    end
 else
    sp(i) = 0;
 end
end

SP0= sum(sp(:) == 0);
SP10 = sum(sp(:) == 1)/n;
SP11 = sum(sp(:) == 2)/n;
SP20 = sum(sp(:) == 3)/n;
SP21 = sum(sp(:) == 4)/n;
SP30 = sum(sp(:) == 5)/n;
SP31 = sum(sp(:) == 6)/n;
SP40 = sum(sp(:) == 7)/n;
SP41 = sum(sp(:) == 8)/n;
SP50 = sum(sp(:) == 9)/n;
SP51 = sum(sp(:) == 10)/n;
SP60 = sum(sp(:) == 11)/n;
SP61 = sum(sp(:) == 12)/n;
SP70 = sum(sp(:) == 13)/n;
SP71 = sum(sp(:) == 14)/n;
SP80 = sum(sp(:) == 15)/n;
SP81 = sum(sp(:) == 16)/n;
SP90 = sum(sp(:) == 17)/n;
SP91 = sum(sp(:) == 18)/n;
SP100 = sum(sp(:) == 19)/n;
SP101 = sum(sp(:) == 20)/n;


snorm = [SP10;SP20;SP30;SP40;SP50;SP60;SP70;SP80;SP90;SP100];
smal = [SP11;SP21;SP31;SP41;SP51;SP61;SP71;SP81;SP91;SP101];

T4 = table(US,snorm,smal);

%Probability of Protocol-----------------------------------
P = string(Protocol);
UPu = unique(P);
x = size(UPu);
x = x(1,1);

UP = string(zeros(3,1));
UP(1:x,:) = UPu;

pp = zeros(n,1);
for i = 1:n
 if P(i,:) == UP(1,:)
```

```
     if Error(i,1) == 0
         pp(i,1) = 1;
     else
         pp(i,1) = 2;
     end
 end
 if P(i,:) == UP(2,:)
     if Error(i,1) == 0
         pp(i,1) = 3;
     else
         pp(i,1) = 4;
     end
 end
 if P(i,:) == UP(3,:)
    if Error(i,1) == 0
         pp(i,1) = 5;
    else
         pp(i,1) = 6;
    end
 end
end

PP10 = sum(pp(:) == 1)/n;
PP11 = sum(pp(:) == 2)/n;
PP20 = sum(pp(:) == 3)/n;
PP21 = sum(pp(:) == 4)/n;
PP30 = sum(pp(:) == 5)/n;
PP31 = sum(pp(:) == 6)/n;
PPna = sum(pp(:) == 0);
pnorm = [PP10;PP20;PP30];
pmal = [PP11;PP21;PP31];

T5 = table(UP,pnorm,pmal);

end
```

APPENDIX F. TRAIL DATA SCRIPT

The following script utilizes the feature's individual probabilities to calculate the posterior probability for the trial data. This is then compared to the Xcel document, processed previously, to determine if the classification was correct.

```
function [N,M,outcome, comparer] =
rundataML(SR,US,snorm,smal,DR,UD,dnorm,dmal,PR,UP,pnorm,pmal,LR,lnorm,lmal
,OP,ER,burnin)

snormin = snorm*burnin;
smalin = smal*burnin;
dnormin = dnorm*burnin;
dmalin = dmal*burnin;
pnormin = pnorm*burnin;
pmalin = pmal*burnin;
lnormin = lnorm*burnin;
lmalin = lmal*burnin;

n = length(SR);
% n = 1;
Npostr = zeros(n,6);
Mpostr = zeros(n,6);

outcome = zeros(n,1);
comparer = zeros(n,1);

for i = 1:n
%SOURCE-------------------------------------------
SR = string(SR);

 if SR(i,1) == US(1,:)
     Npostr(i,1) = snorm(1,:);
     Mpostr(i,1) = smal(1,:);
 elseif SR(i,1) == US(2,:)
     Npostr(i,1) = snorm(2,:);
     Mpostr(i,1) = smal(2,:);
 elseif SR(i,1) == US(3,:)
     Npostr(i,1) = snorm(3,:);
     Mpostr(i,1) = smal(3,:);
 elseif SR(i,1) == US(4,:)
     Npostr(i,1) = snorm(4,:);
     Mpostr(i,1) = smal(4,:);
 elseif SR(i,1) == US(5,:)
     Npostr(i,1) = snorm(5,:);
     Mpostr(i,1) = smal(5,:);
 elseif SR(i,1) == US(6,:)
```

```
    Npostr(i,1) = snorm(6,:);
    Mpostr(i,1) = smal(6,:);
 elseif SR(i,1) == US(7,:)
    Npostr(i,1) = snorm(7,:);
    Mpostr(i,1) = smal(7,:);
 end

%DESTINATION---------------------------------------------
DR = string(DR);

 if DR(i,1) == UD(1,:)
    Npostr(i,2) = dnorm(1,:);
    Mpostr(i,2) = dmal(1,:);
 elseif DR(i,1) == UD(2,:)
    Npostr(i,2) = dnorm(2,:);
    Mpostr(i,2) = dmal(2,:);
 elseif DR(i,1) == UD(3,:)
    Npostr(i,2) = dnorm(3,:);
    Mpostr(i,2) = dmal(3,:);
 elseif DR(i,1) == UD(4,:)
    Npostr(i,2) = dnorm(4,:);
    Mpostr(i,2) = dmal(4,:);
 elseif DR(i,1) == UD(5,:)
    Npostr(i,2) = dnorm(5,:);
    Mpostr(i,2) = dmal(5,:);
 elseif DR(i,1) == UD(6,:)
    Npostr(i,2) = dnorm(6,:);
    Mpostr(i,2) = dmal(6,:);
 elseif DR(i,1) == UD(7,:)
    Npostr(i,2) = dnorm(7,:);
    Mpostr(i,2) = dmal(7,:);
 elseif DR(i,1) == UD(8,:)
    Npostr(i,2) = dnorm(8,:);
    Mpostr(i,2) = dmal(8,:);
 end

%PROTOCOL------------------------------------------------

PR = string(PR);

 if PR(i,1) == UP(1,:)
    Npostr(i,3) = pnorm(1,:);
    Mpostr(i,3) = pmal(1,:);
 elseif PR(i,1) == UP(2,:)
    Npostr(i,2) = pnorm(2,:);
    Mpostr(i,2) = pmal(2,:);
 end

%LENGTH--------------------------------------------
```

```
 if LR(i,:) >= 0 && LR(i,:) < 400
     Npostr(i,4) = lnorm(1,:);
     Mpostr(i,4) = lmal(1,:);
 elseif LR(i,:) >= 400 && LR(i,:) < 800
     Npostr(i,4) = lnorm(2,:);
     Mpostr(i,4) = lmal(2,:);
 elseif LR(i,:) >= 800 && LR(i,:) < 1200
     Npostr(i,4) = lnorm(3,:);
     Mpostr(i,4) = lmal(3,:);
 elseif LR(i,:) >= 1200 && LR(i,:) < 1600
     Npostr(i,4) = lnorm(4,:);
     Mpostr(i,4) = lmal(4,:);
 elseif LR(i,:) >= 1600 && LR(i,:) < 2000
     Npostr(i,4) = lnorm(5,:);
     Mpostr(i,4) = lmal(5,:);
 end

%Calculate
Npostr(i,5) = Npostr(i,1)*Npostr(i,2)*Npostr(i,3)*Npostr(i,4);
Mpostr(i,5) = Mpostr(i,1)*Mpostr(i,2)*Mpostr(i,3)*Mpostr(i,4);

Npostr(i,6) = Npostr(i,5)*OP(2,1)/(Npostr(i,5)+Mpostr(i,5));
Mpostr(i,6) = Mpostr(i,5)*OP(1,1)/(Npostr(i,5)+Mpostr(i,5));

%COMPARE-------------------------------------------------

 if Npostr (i,6) > Mpostr (i,6)
     outcome(i,1) = 0;
 elseif Npostr (i,6) < Mpostr (i,6)
     outcome(i,1) = 1;
 else
     outcome(i,1) = 0;
 end

 if outcome(i,:) == ER(i,:)
     comparer(i,:) = 0;
 else
     comparer(i,:) = 1;
 end

%Reimplement-----------------------------------------------------------------
----

 if SR(i,1) == US(1,:)
     if ER(i,1) == 0
     snormin(1,1) = snormin(1,1)+1;
     snorm(:,1) = (snormin(:,1))/(burnin+i);
     smal(:,1) = (smalin(:,1))/(burnin+i);
```

```
    else
    smalin(1,1) = smalin(1,1)+1;
    snorm(:,1) = (snormin(:,1))/(burnin+i);
    smal(:,1) = (smalin(:,1))/(burnin+i);
    end
elseif SR(i,1) == US(2,:)
    if ER(i,1) == 0
    snormin(2,1) = snormin(2,1)+1;
    snorm(:,1) = (snormin(:,1))/(burnin+i);
    smal(:,1) = (smalin(:,1))/(burnin+i);
    else
    smalin(2,1) = smalin(2,1)+1;
    snorm(:,1) = (snormin(:,1))/(burnin+i);
    smal(:,1) = (smalin(:,1))/(burnin+i);
    end
elseif SR(i,1) == US(3,:)
    if ER(i,1) == 0
    snormin(3,1) = snormin(3,1)+1;
    snorm(:,1) = (snormin(:,1))/(burnin+i);
    smal(:,1) = (smalin(:,1))/(burnin+i);
    else
    smalin(3,1) = smalin(3,1)+1;
    snorm(:,1) = (snormin(:,1))/(burnin+i);
    smal(:,1) = (smalin(:,1))/(burnin+i);
    end
elseif SR(i,1) == US(4,:)
    if ER(i,1) == 0
    snormin(4,1) = snormin(4,1)+1;
    snorm(:,1) = (snormin(:,1))/(burnin+i);
    smal(:,1) = (smalin(:,1))/(burnin+i);
    else
    smalin(4,1) = smalin(4,1)+1;
    snorm(:,1) = (snormin(:,1))/(burnin+i);
    smal(:,1) = (smalin(:,1))/(burnin+i);
    end
elseif SR(i,1) == US(5,:)
    if ER(i,1) == 0
    snormin(5,1) = snormin(5,1)+1;
    snorm(:,1) = (snormin(:,1))/(burnin+i);
    smal(:,1) = (smalin(:,1))/(burnin+i);
    else
    smalin(5,1) = smalin(5,1)+1;
    snorm(:,1) = (snormin(:,1))/(burnin+i);
    smal(:,1) = (smalin(:,1))/(burnin+i);
    end
elseif SR(i,1) == US(6,:)
    if ER(i,1) == 0
    snormin(6,1) = snormin(6,1)+1;
    snorm(:,1) = (snormin(:,1))/(burnin+i);
```

```
     smal(:,1) = (smalin(:,1))/(burnin+i);
     else
     smalin(6,1) = smalin(6,1)+1;
     snorm(:,1) = (snormin(:,1))/(burnin+i);
     smal(:,1) = (smalin(:,1))/(burnin+i);
     end
 elseif SR(i,1) == US(7,:)
     if ER(i,1) == 0
     snormin(7,1) = snormin(7,1)+1;
     snorm(:,1) = (snormin(:,1))/(burnin+i);
     smal(:,1) = (smalin(:,1))/(burnin+i);
     else
     smalin(7,1) = smalin(7,1)+1;
     snorm(:,1) = (snormin(:,1))/(burnin+i);
     smal(:,1) = (smalin(:,1))/(burnin+i);
     end
 end


if DR(i,1) == UD(1,:)
     if ER(i,1) == 0
     dnormin(1,1) = dnormin(1,1)+1;
     dnorm(:,1) = (dnormin(:,1))/(burnin+i);
     dmal(:,1) = (dmalin(:,1))/(burnin+i);
     else
     dmalin(1,1) = dmalin(1,1)+1;
     dnorm(:,1) = (dnormin(:,1))/(burnin+i);
     dmal(:,1) = (dmalin(:,1))/(burnin+i);
     end
 elseif DR(i,1) == UD(2,:)
     if ER(i,1) == 0
     dnormin(2,1) = dnormin(2,1)+1;
     dnorm(:,1) = (dnormin(:,1))/(burnin+i);
     dmal(:,1) = (dmalin(:,1))/(burnin+i);
     else
     dmalin(2,1) = dmalin(2,1)+1;
     dnorm(:,1) = (dnormin(:,1))/(burnin+i);
     dmal(:,1) = (dmalin(:,1))/(burnin+i);
     end
 elseif DR(i,1) == UD(3,:)
     if ER(i,1) == 0
     dnormin(3,1) = dnormin(3,1)+1;
     dnorm(:,1) = (dnormin(:,1))/(burnin+i);
     dmal(:,1) = (dmalin(:,1))/(burnin+i);
     else
     dmalin(3,1) = dmalin(3,1)+1;
     dnorm(:,1) = (dnormin(:,1))/(burnin+i);
     dmal(:,1) = (dmalin(:,1))/(burnin+i);
     end
```

```
elseif DR(i,1) == UD(4,:)
    if ER(i,1) == 0
    dnormin(4,1) = dnormin(4,1)+1;
    dnorm(:,1) = (dnormin(:,1))/(burnin+i);
    dmal(:,1) = (dmalin(:,1))/(burnin+i);
    else
    dmalin(4,1) = dmalin(4,1)+1;
    dnorm(:,1) = (dnormin(:,1))/(burnin+i);
    dmal(:,1) = (dmalin(:,1))/(burnin+i);
    end
elseif DR(i,1) == UD(5,:)
    if ER(i,1) == 0
    dnormin(5,1) = dnormin(5,1)+1;
    dnorm(:,1) = (dnormin(:,1))/(burnin+i);
    dmal(:,1) = (dmalin(:,1))/(burnin+i);
    else
    dmalin(5,1) = dmalin(5,1)+1;
    dnorm(:,1) = (dnormin(:,1))/(burnin+i);
    dmal(:,1) = (dmalin(:,1))/(burnin+i);
    end
elseif DR(i,1) == UD(6,:)
    if ER(i,1) == 0
    dnormin(6,1) = dnormin(6,1)+1;
    dnorm(:,1) = (dnormin(:,1))/(burnin+i);
    dmal(:,1) = (dmalin(:,1))/(burnin+i);
    else
    dmalin(6,1) = dmalin(6,1)+1;
    dnorm(:,1) = (dnormin(:,1))/(burnin+i);
    dmal(:,1) = (dmalin(:,1))/(burnin+i);
    end
elseif DR(i,1) == UD(7,:)
    if ER(i,1) == 0
    dnormin(7,1) = dnormin(7,1)+1;
    dnorm(:,1) = (dnormin(:,1))/(burnin+i);
    dmal(:,1) = (dmalin(:,1))/(burnin+i);
    else
    dmalin(7,1) = dmalin(7,1)+1;
    dnorm(:,1) = (dnormin(:,1))/(burnin+i);
    dmal(:,1) = (dmalin(:,1))/(burnin+i);
    end
elseif DR(i,1) == UD(8,:)
    if ER(i,1) == 0
    dnormin(8,1) = dnormin(8,1)+1;
    dnorm(:,1) = (dnormin(:,1))/(burnin+i);
    dmal(:,1) = (dmalin(:,1))/(burnin+i);
    else
    dmalin(8,1) = dmalin(8,1)+1;
    dnorm(:,1) = (dnormin(:,1))/(burnin+i);
    dmal(:,1) = (dmalin(:,1))/(burnin+i);
```

```
    end
end


 if PR(i,1) == UP(1,:)
    if ER(i,1) == 0
    pnormin(1,1) = pnormin(1,1)+1;
    pnorm(:,1) = (pnormin(:,1))/(burnin+i);
    pmal(:,1) = (pmalin(:,1))/(burnin+i);
    else
    pmalin(1,1) = pmalin(1,1)+1;
    pnorm(:,1) = (pnormin(:,1))/(burnin+i);
    pmal(:,1) = (pmalin(:,1))/(burnin+i);
    end
 elseif PR(i,1) == UP(2,:)
    if ER(i,1) == 0
    pnormin(2,1) = pnormin(2,1)+1;
    pnorm(:,1) = (pnormin(:,1))/(burnin+i);
    pmal(:,1) = (pmalin(:,1))/(burnin+i);
    else
    pmalin(2,1) = pmalin(2,1)+1;
    pnorm(:,1) = (pnormin(:,1))/(burnin+i);
    pmal(:,1) = (pmalin(:,1))/(burnin+i);
    end
 end


 if LR(i,:) >= 0 && LR(i,:) < 400
    if ER(i,1) == 0
    lnormin(1,1) = lnormin(1,1)+1;
    lnorm(:,1) = (lnormin(:,1))/(burnin+i);
    lmal(:,1) = (lmalin(:,1))/(burnin+i);
    else
    lmalin(1,1) = lmalin(1,1)+1;
    lnorm(:,1) = (lnormin(:,1))/(burnin+i);
    lmal(:,1) = (lmalin(:,1))/(burnin+i);
    end
 elseif LR(i,:) >= 400 && LR(i,:) < 800
    if ER(i,1) == 0
    lnormin(2,1) = lnormin(2,1)+1;
    lnorm(:,1) = (lnormin(:,1))/(burnin+i);
    lmal(:,1) = (lmalin(:,1))/(burnin+i);
    else
    lmalin(2,1) = lmalin(2,1)+1;
    lnorm(:,1) = (lnormin(:,1))/(burnin+i);
    lmal(:,1) = (lmalin(:,1))/(burnin+i);
    end
 elseif LR(i,:) >= 800 && LR(i,:) < 1200
    if ER(i,1) == 0
```

```
    lnorm(3,1) = (lnormin(3,1)+1)/(burnin+i);
    lnormin(3,1) = lnormin(3,1)+1;
    else
    lmalin(3,1) = lmalin(3,1)+1;
    lnorm(:,1) = (lnormin(:,1))/(burnin+i);
    lmal(:,1) = (lmalin(:,1))/(burnin+i);
    end
 elseif LR(i,:) >= 1200 && LR(i,:) < 1600
    if ER(i,1) == 0
    lnormin(4,1) = lnormin(4,1)+1;
    lnorm(:,1) = (lnormin(:,1))/(burnin+i);
    lmal(:,1) = (lmalin(:,1))/(burnin+i);
    else
    lmalin(4,1) = lmalin(4,1)+1;
    lnorm(:,1) = (lnormin(:,1))/(burnin+i);
    lmal(:,1) = (lmalin(:,1))/(burnin+i);
    end
 elseif LR(i,:) >= 1600 && LR(i,:) < 2000
    if ER(i,1) == 0
    lnormin(5,1) = lnormin(5,1)+1;
    lnorm(:,1) = (lnormin(:,1))/(burnin+i);
    lmal(:,1) = (lmalin(:,1))/(burnin+i);
    else
    lmalin(5,1) = lmalin(5,1)+1;
    lnorm(:,1) = (lnormin(:,1))/(burnin+i);
    lmal(:,1) = (lmalin(:,1))/(burnin+i);
    end
 end



N=Npostr(:,6);
M=Mpostr(:,6);
end

end
```

LIST OF REFERENCES

[1] P. Paige, “SECNAV outlines five ‘ambitious’ energy goals,” U.S. Navy, Oct. 16, 2009. [Online]. Available: https://www.navy.mil/submit/display.asp?story_id=49044.

[2] Department of the Navy, “Department of the Navy’s energy program for security and independence,” Washington, DC, USA, Oct. 2010. [Online]. Available: https://www.secnav.navy.mil/eie/ASN%20EIE%20Policy/Naval_Energy_Strategic_Roadmap.pdf.

[3] Energy Central, “NAVFAC Reaches Key Milestone in Deploying New Smart Energy Monitoring and Control Solution,” Energy Central, Apr. 2019. [Online]. Available: https://www.navfac.navy.mil/news/2019_press_releases/April2019-press-releases/NAVFAC-Reaches-Key-Milestone-in-Deploying-New-Smart-Energy-Monitoring-and-Control-Solution.html.

[4] K. Kimani, V. Oduol, and K. Langat, “Cyber security challenges for IoT-based smart grid networks,” *Int. J. Crit. Infrastruct. Prot.*, vol. 25, pp. 36–49, Jun. 2019, doi: 10.1016/j.ijcip.2019.01.001.

[5] “Bayesian Learning for Machine Learning: Part II - Linear Regression.” https://wso2.com/blog/research/part-two-linear-regression (accessed Jul. 29, 2020).

[6] A. Lemay and J. M. Fernandez, “Providing SCADA network data sets for intrusion detection research,” p. 8.

[7] C. Technologies, “Evaluation of Machine Learning Algorithms for Intrusion Detection System,” *Medium*, May 13, 2019. https://medium.com/cuelogic-technologies/evaluation-of-machine-learning-algorithms-for-intrusion-detection-system-6854645f9211 (accessed Aug. 12, 2020).

[8] T. Phuoc, P. Tsai, T. Jan, and X. Kong, “Network Intrusion Detection using Machine Learning and Voting techniques,” in *Machine Learning*, Y. Zhang, Ed. InTech, 2010.

[9] N. Williams, S. Zander, and G. Armitage, “A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 5, pp. 5–16, Oct. 2006, doi: 10.1145/1163593.1163596.

[10] J. M. Porup, “What is Metasploit? And how to use this popular hacking tool,” *CSO Online*, Mar. 25, 2019. https://www.csoonline.com/article/3379117/what-is-metasploit-and-how-to-use-this-popular-hacking-tool.html (accessed Aug. 12, 2020).

[11] S. Gupta, “Pros and cons of various Classification ML algorithms,” *Medium*, Jun. 23, 2020. https://towardsdatascience.com/pros-and-cons-of-various-classification-ml-algorithms-3b5bfb3c87d6 (accessed Aug. 02, 2020).

[12] A. Zanutto, B. Shreeve, K. Follis, J. Busby, and A. Rashid, “The Shadow Warriors: In the no man’s land between industrial control systems and enterprise IT systems,” presented at the Symposium of Usable Privacy and Security (SOUPS), Santa Clara, CA, USA, Jul. 12, 2017, [Online]. Available: Online. Available: https://www.usenix.org/system/files/conference/soups2017/wsiw2017-zanutto.pdf.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California