

Generating Firewall Rules Through a User Interface and Machine Learning

A Major Qualify Project Report

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE



in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by:

Matthew Iandoli

Jerry Perez

Vinit Kothari

Jiaqi Zhang

6 May 2021

Sponsored by
Arco IT GmbH

Prof. Randy Paffenroth, Advisor

Prof. Oren Mangoubi, Advisor

Prof. Andrew Trapp, Advisor

This report represents the work of one or more WPI undergraduate students submitted to the faculty as evidence of completion of a degree requirement. WPI routinely publishes these reports on the web without editorial or peer review. For more information about the projects program at WPI, see <http://www.wpi.edu/Academics/Projects>.

Abstract

Cyber attacks cost the world economy \$6 trillion per year and are increasing by 15 percent per year. By 2025, the world economy will have to pay a heavy price of \$10 trillion for such attacks. A major factor contributing to these attacks is inadequate attention paid to computer network architectures. Accordingly, automation of technologies, like firewalls, that keep check of network architectures is desired. An automated firewall will be able to point out the rules missing or changes to rules when required. Along with automation it is also important to keep a human in the loop to overrule incorrect suggestions from the technology. In order to generate firewall rules that are effective, there are several thousands of network flows that need to be analyzed, which is simply beyond a reasonable execution time for a human. Overall, targeted approach is to use a machine learning algorithm in order to derive more complex rules that can simultaneously consider many more inputs.

Acknowledgements

We would like to express the highest level of gratitude to each member of our inner-most team that have supported this project. From coping with the repercussions of COVID-19 to navigating time zone restrictions, our team could not have been more grateful to those who aided us in ensuring this project was a success.

First, and foremost, we would like to thank our sponsors from Arco IT GmbH Security Services: Bertram Duskus, Amy Vaillancourt, and Zaheer Uddin Qureshi. Their support and guidance allowed us to complete this project to its fullest extent, providing them with tools and resources that aid in the security of the clients of Arco IT GmbH.

In addition, we would like to thank our advisors from Worcester Polytechnic Institute: Randy Paffenroth, Oren Mangoubi, Andrew Trapp. These individuals, despite troubling times believed that our work as a team would conquer and instilled the motivation within us to continue pressing forward. The assistance, feedback, and continuous push for learning more allowed us to bring this project to its utmost potential.

Executive Summary

There is a hacker attack on average every 39 seconds in the United States, affecting nearly one in every three Americans [1]. Whether this be through Smurf attacks, PoD attacks, or even botnet attacks, the average cost globally for a data breach at a publicly-traded company is 116 million dollars [1]. This doesn't even begin to touch the surface on the crippling side effects of a business losing possession over sensitive client information. The protecting of highly valuable information has come to be known as cybersecurity. A term that Cisco defines as, "Cybersecurity is the practice of protecting systems, networks, and programs from digital attacks. These cyberattacks are usually aimed at accessing, changing, or destroying sensitive information; extorting money from users; or interrupting normal business processes." [2].

Cybersecurity has become extraordinarily prevalent in the modern day due to several rising business infrastructures that need to store various bits of information regarding their clientele. With that being said, there has also been a dramatic spike in unethical hacking in order to manipulate, leverage, or steal personal information.

One way to protect several machines within an organization is through a firewall, which monitors incoming and outgoing data to ensure that the data that is transferred and the machines that are communicating are safe. However, there are several ways that attackers can go undetected. When designing firewall rules in order to ensure that the only traffic that is acceptable are those that are safe, there are several things to keep in mind. For one, there needs to be some fundamental information regarding the communication methods of existing traffic. Whether this be through IP addresses, port numbers, or packet analysis, there needs to be a tell-tale sign that the traffic is malicious.

Arco IT GmbH is a cybersecurity company located in Zurich Switzerland specializing in cybersecurity analysis and advice that can be implemented into business strategies [3]. Through a user interface that generates firewall rules, the Arco IT GmbH team hopes to increase data privacy in the workplace for all users.

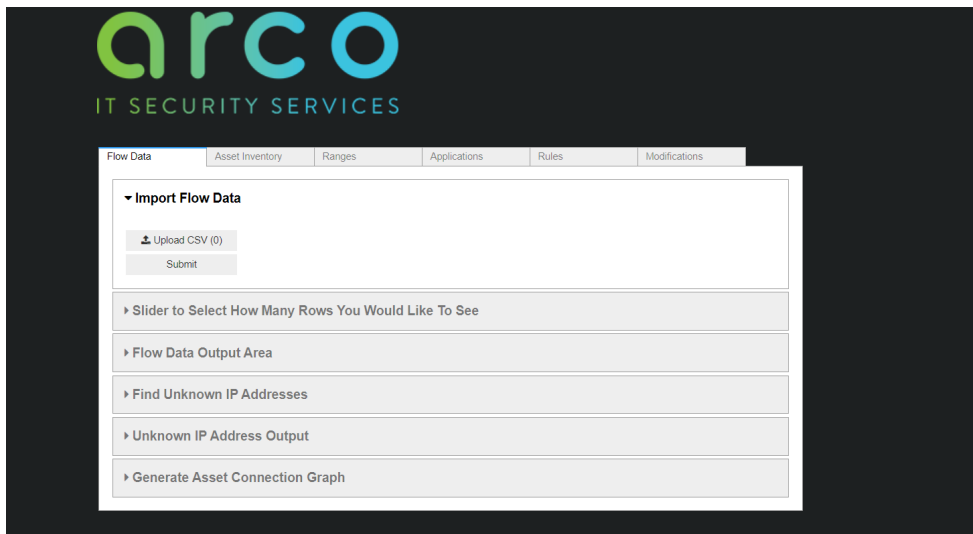


Figure 1: Displays the user interface layout which has several layers of functionality. Namely, the generation of semi-autonomous firewall rules that utilize the other tabs of data manipulation.

When designing firewall rules, Arco IT GmbH needed a way to manipulate flow data, asset inventory, IP ranges, and applications alongside autonomously generated firewall rules. This was to be done through a user interface that gives the user the ability to modify several forms of data in order to train a generative adversarial network that would then generate firewall rules that can be implemented into the system.

Generative adversarial networks function on the premise of two conflicting networks, a generator, and a discriminator. Typically this method is most effective for generating images where the generator consistently creates fake data that the discriminator needs to differentiate between real and fake. However, our team was able to bend the functionality of this generative adversarial network to function with flow data that concerns network traffic, in order to train a model that would eventually generate firewall rules.

Contents

1	Introduction	8
2	Background	9
2.1	Software Engineering	9
2.1.1	Use Cases	9
2.1.2	StoryBoards	9
2.2	Software Requirements	10
2.2.1	Collector Agents	11
2.2.2	Transfer and Data Store	11
2.2.3	Inventory System	11
2.2.4	Front-End	11
2.2.5	Analyzer and Firewall Connectors	11
2.3	Software Delivery	11
2.3.1	Docker	12
2.4	Computer Networks	13
2.4.1	IP Addresses	14
2.4.2	Ports	14
2.4.3	Firewalls	15
2.5	Feature Engineering	15
2.6	Machine Learning	16
2.6.1	Defining the Machine Learning Problem	17
2.6.2	A One Class Problem	17
2.6.3	Deep learning	17
2.6.4	Related Machine Learning Work	19
2.7	Industrial Engineering Application	19

2.7.1	Queuing Theory	19
2.7.2	Regression Analysis	21
2.7.3	Correlation Matrices	22
3	Methodology	22
3.1	Fixing the One Class Problem	22
3.2	Generative Adversarial Networks	23
3.3	Featuring Engineering Network Flow Traffic	24
3.4	Firewall Rule Syntax	24
3.5	Selecting a Machine Learning Model	25
3.6	Evaluating the Machine Learning Model	25
3.7	Developing the User Interface	26
3.8	Jupyter Notebooks, IPywidgets, and JupyterLab	29
3.9	Dashboard and Callbacks	31
4	Results	32
4.1	Model Performance	32
4.1.1	Second Class Generation	32
4.1.2	Model Accuracy	35
5	Industrial Engineering Application	36
5.1	Methods	36
5.1.1	Determining the performance of the current network	36
5.1.2	Determining the predictability of one parameter by another parameter	37
5.1.3	Determining the Correlation Between Shifted Data Set	38
5.2	Results	39
5.2.1	Determining the performance of the current system	40
5.2.2	Determining the predictability of one parameter by another parameter	41

5.2.3 Determining the Correlation Between Shifted Data Set 43

5.3 Conclusion 46

6 Conclusion 46

6.1 Key Points 46

6.2 Future Work 47

A Software Delivery 52

B Sample Network Flow Data 53

C IP Inventory 54

D Model Output 55

1 Introduction

Computer networks are responsible for a massive amount of communications in today's societies. Whether it may be used for machines in a factory to transfer data from one system to another or if it used for keeping connected with family members, computer networks are everywhere. Since computer networks play such a vital role in our lives, network attacks are a huge risk and have costed companies not just money but data losses, while also compromising their security systems. In fact, from 2006 onward network attacks have consistently cost companies around \$8.64 million dollars on average. Therefore, in order to minimize these and prevent attacks as much as possible firewalls are used. The purpose of a firewall is to detect any anomaly behavior in the network and prevent risks by putting a halt to it before it has an opportunity to impose a threat. However, optimizing such firewalls is difficult because of various factors. Firstly because, there is always a constant change in networks. Hence, protection in the form of the firewall also has to be updated with this and with this more rules need to be added, while at the same time ensuring that another important rule needed by another application or program is not deleted. Constant updating of rule sets also causes rule sets to grow chaotic and therefore introduce unnecessary risks. Secondly an existing firewall may stop offering protecting if threats change. For example if new malware is introduced, it may not be detected by a firewall. Therefore, causing a new type of threat, risking the security of the company possibly to a greater extent. Moreover another problem could result from misconfigured firewalls. For example, if a company does not configure the firewall accordingly for their network by leaving the settings on default, they will put themselves as bad a risk as without a firewall. Accordingly, we want to focus on the automated creation of firewall rules. Such a system will constantly learn from patterns and constantly create rules for a network based on the net-flow data.

One of our key contributions is training an appropriate machine learning algorithm. What makes this hard is that we only have normal network traffic, meaning we have a one-class problem. One-class problems are difficult since it limits how you can train your model without a second class of data [4]. Examples of one-class problems can be seen with anomaly detection. Since the anomaly that could occur in the future is unknown, training a model to look for these anomalies is very difficult. Different approaches in solving the one-class problem exist such as strictly looking for outliers or creating an artificial class of data to supplement the single class. The latter is what we will try to accomplish. By including a GUI that a human can supplement information about the network that will aid the automated firewall rule creation, we can improve on current existing technologies and help to protect important networks from irrelevant traffic and malicious attacks.

Another key contribution was adding a human in the loop that would be able to maintain the firewall rules through a GUI. Leaving a tool to be completely automated may seem great but it can possess certain risks if they algorithm behind it creates a rule that can allow risks, it can be harmful for a company. In order to prevent this we decided to make a GUI where certain people from the company can keep a check on the rules. In addition to this they will also have the option of accepting the rules that they seem fit for their network.

2 Background

2.1 Software Engineering

2.1.1 Use Cases

Almost, if not all, software engineering approaches hold the client to the highest standard, ensuring that their needs are met throughout the process of developing a user interface. Ensuring the client is satisfied can be done in several ways ranging from creating use-cases, developing storyboards and generating mock-up scenarios that allow for further communication between the two parties. However, a successful developer must begin with well thought-out use cases that describe what each function of the user interface will mean to the client. In other words, use cases generally define the interactions between an actor and a system in order to derive an end result. Some questions that need to be answered to effectively generate use cases are: What action yields the desired result? and Who are the actors of each action? Through effectively generating these use-cases, the developer can find several features that might not have strong importance in their final product, and by that same token, find features that are of the utmost importance [5].

2.1.2 StoryBoards

Story-boards, on the other hand, are typically generated after the use-cases, and give the developer a very rough idea as to what their deliverable will visually look like. For user interfaces, this would mean sketching out what buttons, drop-downs, text-boxes, etc. would be put in which location and deriving decisions for the logical flow of these interactive elements. This allows the developer to progress with a purpose, using the story-board as a check-list for their first draft. Eventually, through client interaction, comes improvements on that user interface.

Now, a storyboard is typically hand-drawn and very roughly puts together the functioning elements of the user interface that gives the programmer a road-map for getting started. In our case, Arco IT GmbH generated figures on exactly what was desired, and that was our starting point. As we progressed through the term, certain elements of the planned user interface would be re-evaluated, determining whether or not the practicality of these elements were what was envisioned. Despite constant changes in order to satisfy the vision of the client, story-boarding is crucial in developing a user interface since it gives a sense of direction and allows the programmer a vital starting point.

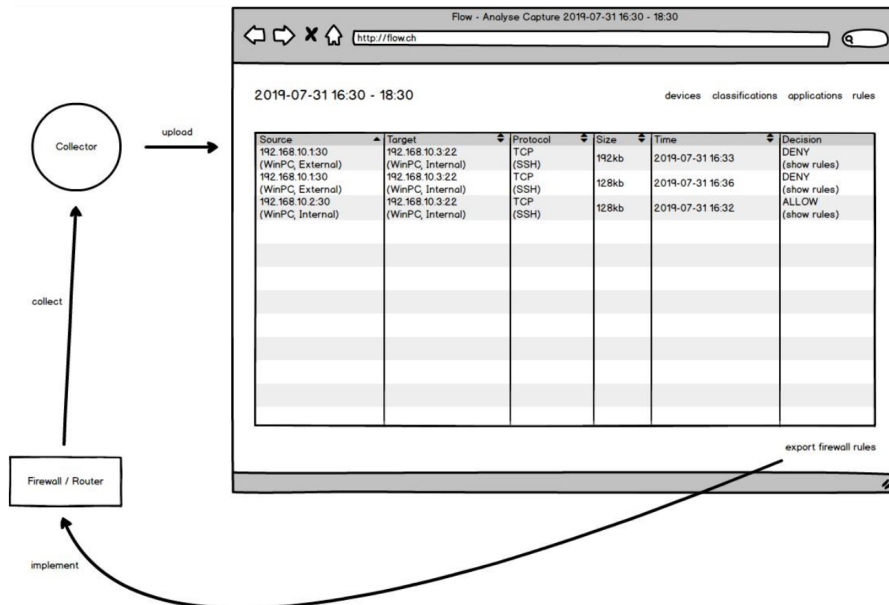


Figure 2: Shows the initial storyboard generated by Arco IT GmbH on the official project description. PriARily, it shows a tab system made up of “Devices”, “Classifications”, “Applications”, and “Rules”. Secondly, it shows the body of the user interface with a table that consists of “Source”, “Target”, “Protocol”, “Size”, “Time”, and “Decision”. Following the logical blueprint of beginning at the top and progressing downward, there is an export firewall rules button at the bottom that the user can click on to save the rules that have been generated. This storyboard tells us that there ought to be some form of collector agent, which collects information about data flows in a network, proceeds to upload it to the user interface which will generate machine learning rules, export those rules, and have those rules implemented within a firewall or router.

It is important to mention that finding a starting point can be extremely difficult when there are many features to implement in a short period of time. However, generating use-cases allowed us to determine what was needed, and what was just adding complexity to the simplistic state of the project. Creating story-boards is what gave us a foundation to start with, and a solid direction as to where to improve our user interface. Lastly, daily meetings is what allowed us to continue progressing, making subtle improvements to the user interface, machine learning, and business side of the project until it was exactly what was pictured by our sponsors.

2.2 Software Requirements

The most important portion of the software engineering stage of any project is understanding the requirements. Moreover, the developer should put their best foot forward in adhering to these requirements closely to produce a successful project. In the case of this project, we were given a project description that described in very fine detail what was desired and how it could be obtained. There were several elements that we needed to grasp a firm understanding of in order to make the final deliverable as successful as possible.

2.2.1 Collector Agents

The first project component was the implementation of collector agents. These collector agents are located within the network that is being analyzed and collect various forms of information regarding data flows on that network. This can be in the form of PCAP files, Netflow imports, or live captures. The collector agents will be used to upload the information that they capture to the next stage of processing.

2.2.2 Transfer and Data Store

The second project component was a transfer and data store aspect which takes the information collected by the collectors and converts it to a more general format. This general format is one that can be understood by the user for further processing.

2.2.3 Inventory System

The third project component was an inventory system that would be used to store static information regarding the devices on the network such as: name, device type, manufacturer, host-name, etc. The static information that is stored in this system would then be used to assign different groups for devices. In addition, the inventory system would store information regarding IP ranges, sub-nets, and traffic type.

2.2.4 Front-End

The fourth project component is the user interface that incorporates all of the project components listed above. The user interface should have the ability for users to manipulate several aspects of non-static data such as inventory host-name configuration, the addition and deletion of tags, ranges, applications and so on. More importantly, the user should have ability to generate firewall rules through this user interface using the data that they have uploaded and manipulated.

2.2.5 Analyzer and Firewall Connectors

The final project component was grouped into two: an analyzer and firewall connectors. The analyzer, which took advantage of machine learning, evaluates the observed traffic flows and recommends a set of firewall rules. The firewall connectors transform the internal creation of these rules into vendor specific format that could then be implemented.

2.3 Software Delivery

There are several approaches one could take in generating a user interface of this scale. The first decision that needs to be made is which programming language is going to be used, which also needs full consideration for which tools are going to be needed as a byproduct. Our team decided that the programming language of best choice would be Python since it is the simplest language to use and there are several tools for data science, machine learning, and graphical user interfaces. However, Python has several requirements in order to use these tools, and while it might be convenient on a machine that already satisfies the requirements, this would not be the situation that we face. The overarching goal of this project was to

deliver a user interface that would require very little contribution from the Arco IT GmbH team. For this reason alone, our team decided to use Docker for the software delivery.

2.3.1 Docker

Docker uses operating system level virtualization to deliver software in packages, otherwise known as containers. Containers bundle their own software, libraries and configuration files are being used by various companies currently. In fact, several companies such as Red Hat, Canonical, Oracle, and Microsoft have embraced Docker [6]. There are several reasons why massive companies such as the ones listed are beginning to incorporate docker more freely, consisting of being more efficient than hypervisors, continuous integration, portability, and ease of deployment.

A hypervisor is commonly referred to as a virtual machine monitor (VMM), which not only creates, but runs virtual machines. What makes a hypervisor special is that it allows the sharing of resources between multiple guests and a host. Hypervisors give the user the ability to use processing power, memory, and even storage that aren't available on their own machine. There are two types of hypervisors: type 1 ("bare metal") or type 2 ("hosted"). Type 1 hypervisors act as an operating system, running directly on a host's machine, whereas a type 2 hypervisor runs on a layer of the host's installed operating system [7].

Docker, on the other hand, runs on the host's Linux kernel, which means that it has full access to the underlying hardware, and essentially does whatever it pleases. Docker aims at isolation as opposed to virtualization meaning the startup time for a Docker container is in the range of milliseconds [8]. Since virtualization typically requires loading the entire operating system in order to function, Docker containers are significantly more performance driven compared to its resource intensive counterpart. Despite this, both of these entities only have a few instances where they can be compared and debated. In our case, there was no question that Docker was the correct choice. It satisfies all of Python's requirements, and it has full access to the hardware which in turn allows for a smooth, rapid delivery.

Since Docker automatically contains all of the packages that the creator desires to use for the software being delivered, Docker was able to provide the most seamless experience. Our idea was to utilize Jupyter [9] Notebooks while also utilizing a GitHub folder that hosted all of the Python completed scripts leading up to this point. GitHub is a hosting platform that allows several users to collaborate on projects from any location as long as they have access to the internet [10]. Utilizing this platform allowed all members of our team to collaborate on the coding portion of this project seamlessly. Overall, utilizing Docker would create an environment that had all of the resources Arco IT GmbH needed in order to run our user interface.

Docker has a feature that gives the user the ability to mount a volume onto the container, which shows and updates a local directory from the configured Docker container. Any GitHub pulls that were done take part on the machine used to configure the Docker container, and since it would be updated within the directory, it would also take effect on the Docker container. This also meant that any files from the local directory that needed to be incorporated into the user interface could simply be imported as a library and run whenever an action on the UI took place.

Overall, Docker gave us the environment to build our user interface in a way that utilized very little hardware, while still providing a smooth workflow for our team. By utilizing containers, any and all packages that were needed for the UI would be installed on the container, meaning when it was shared to the sponsors, they wouldn't have to deal with installing anything on their own machine.

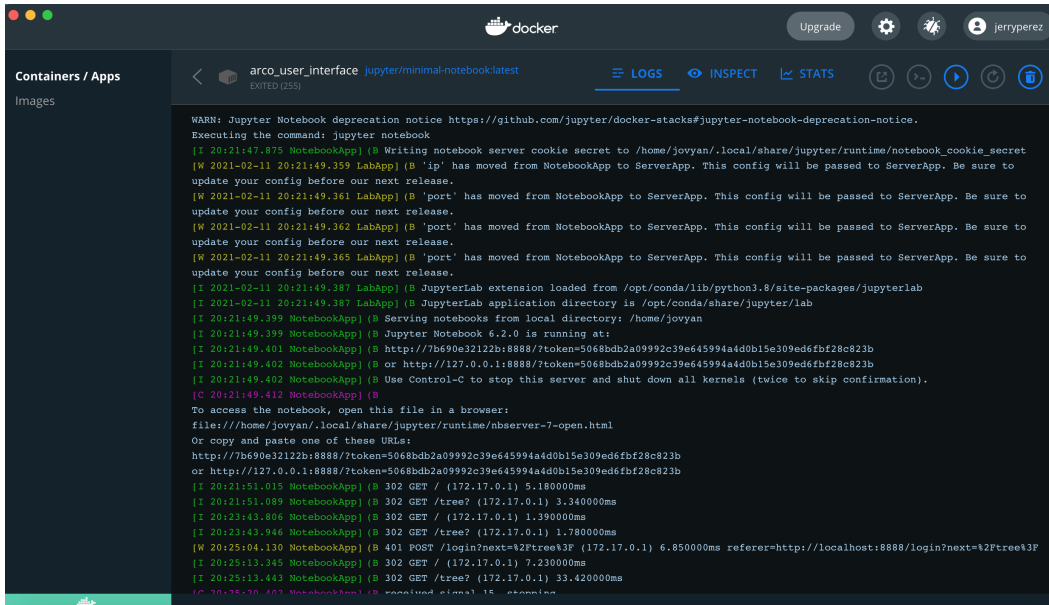


Figure 3: Displays the application version of Docker alongside the user interface container configuration. The two sections on the left hand side allow the user to view all created images, as well as all created containers. Viewing images and containers can also be done through command line, however, the application provides a much more graphically-oriented approach. There are also options to launch the command line that the container is utilizing, which is what was used to install all of the python requirements that were needed for the user interface.

2.4 Computer Networks

Data communication is a new way of communication which is produced by the combination of communication technology and computer science. In order to transmit information between the two stations, there must be a transmission channel. According to the different transmission media, there are wired data communication and wireless data communication. However, they all connect the data terminal with the computer through the transmission channel, so that the data terminals in different places can share the software, hardware and information resources [11].

A computer network or data network is a telecommunication network that allows computers to exchange data. In computer networks, networked computing devices exchange data with each other through data links. The connection between nodes is established by using wired media or wireless media [12].

Computer network is the mixed interconnection of hardware and software, which is the

common feature of computer networks. In today's globalized world, it has become a major tool for data communication. The computer network is designed to handle a certain amount of traffic to ensure an acceptable level of network performance. If the network traffic exceeds the network capacity, the performance will be degraded [13].

Network traffic analysis is the process of monitoring network availability and activity patterns to identify potential threats, including security and operational issues [14]. Actualizing a solution that monitors network traffic will provide you with the knowledge you need to optimize network performance, minimize attack surface, enhance security, and improve resource management. However, it is not enough to know how to monitor network traffic. You also need to consider the data sources of the network monitoring tool. The two most common are flow data and packet data [15].

Network traffic analysis is critical. The system can detect network attacks at early stage and ensure the security of the whole network [16]. The performance of modern networks is affected by many factors, a minor change in the network will give unpredictable results [17].

2.4.1 IP Addresses

Every device on a network will have an IP address, a unique ID that allows network traffic to be sent and received from. Although a unique ID, an IP address is completely random. The standard IPv4 address contains 4 octets, each octet represents a piece of information from left being granular, moving right becoming more fine. Often times a network will have the same first 2 octets (Class B) and the third octet giving the subnet and the fourth octet determining the specific device on the subnet [18].

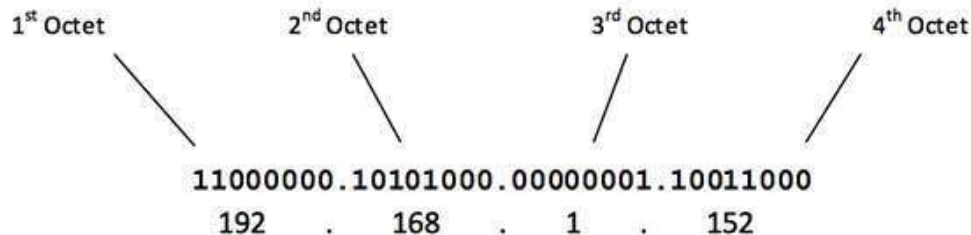


Figure 4: Diagram the structure of an IP address, in this case, a Class C IP Address. Each octet, 8 bits, being a value from 0 to 255. [19]

2.4.2 Ports

When an application communicates on a network, the application chooses a port to transfer data on. An applications port may be chosen randomly selected based off a range and availability but which port it sends data to is not random but pre-determined, these are called well-known ports. Ports are in the range from 0 to 65535 (unsigned 16-bit integer) where the first 1024 contain the well-known ports. Table 1 shows the most common of these well-known ports for TCP and UDP [20].

TCP		UDP	
FTP	20, 21	DNS	53
SSH	22	BooTPS/DHCP	67
Telnet	23	TFTP	69
SMTP	25	NTP	123
DNS	53	SNMP	161
HTTP	80		
POP3	110		
IMAP4	143		
HTTPS	443		

Table 1: Some of the most common of the well-known ports used for TCP and UDP connections. TCP is used more often than UDP, so there are more commonly used ports from the 0 to 1024 range used in TCP.[20]

As noted above, an application can communicate through network ports on either a TCP or a UDP connection. TCP is the more commonly used of the two types of connections with the biggest difference of having error checking on the data being transferred. Since TCP has an additional task of confirming the data being transmitted back and forth hasn't been corrupted, TCP has more overhead than UDP does. TCP accomplishes this error checking by using acknowledgement segments, sending a small bit of data that relays back to the sender if the data was received. While TCP uses acknowledgements to confirm data integrity, UDP leaves these checks up to the application [21].

2.4.3 Firewalls

Firewalls are a great tool in adding security to the network. A firewall works like a filter, allowing some traffic in and blocking other traffic that it deems as unsafe or just traffic that does not belong on the network. A firewall can be compared to an email spam filter. A spam filter will determine, based on a set of rules, if the email trying to be sent to the user is spam or not spam. If the spam filter deems the email as spam, it will move the email into a spam folder. In the case of firewalls, if it deems the network traffic as unsafe or unfit, it will deny that traffic on the network. Examples of traffic that a firewall may block could be an external device is trying to connect to an internal device on the network which may be trying to send malicious data to gain further access to the network, or could be an employee using his personal cellphone on their company's network taking up bandwidth that is slowing down other processes trying to take place on the network.

2.5 Feature Engineering

Feature engineering is the task of creating a set of features from the data that is better suited to be inputted into a machine learning model. For example, a machine learning model such as an artificial neural network (ANN) with one feature, color, cannot take "red", "yellow", or "blue" as an input since it expects numerical values. Instead, we can feature engineer the data before inputting it into the model. For categorical data, we can use the most common technique of using a one-hot vector. A one-hot vector is taking a one dimensional categorical set and blowing it up into N dimensions where N is the number of different categories. For

example:

Categorical	One-hot Vector
Red	$\langle 1, 0, 0 \rangle$
Yellow	$\langle 0, 1, 0 \rangle$
Blue	$\langle 0, 0, 1 \rangle$

Table 2: Example of feature engineering where a categorical data set containing color is turned into a one-hot vector. The first element in the vector represents if the color is red, the second element represents if the color is yellow and the final element for blue.

Since the data is represented as a vector and still not a numerical value, we then would split the one categorical feature into N features, so that each feature will be a 0 or 1. This can be shown for our example from Table 2:

Categorical	Red	Yellow	Blue
Red	1	0	0
Yellow	0	1	0
Blue	0	0	1

Table 3: Showing how one hot vectors are represented as individual features with each dimension of the vector is its own feature.

Feature engineering is not only about creating usable features but also creating good features. What makes a good feature? A good feature would be a feature that can create the most variance in the data. This is important because the machine learning model will be able to more accurately split the data into classes, if classifying, or creating a better regression. For example, if we were trying to determine fitness of adults based off height and weight, we could have height in units of feet and weight in units of pounds. Since height could contain values from 5-7 feet and weight could contain values from 100-250 pounds, weight appears to have more variance. This could be misleading since weight is just in a smaller unit. If we normalize these values onto a mean of 0, we would be able to better see which dataset has more variance.

2.6 Machine Learning

Machine learning says when dealing with a problem, that there is some function f that computes input data $X \subset \mathbb{R}^{M \times N}$ to approximate some value of interest y . The goal of machine learning focuses on getting as close to the true f by using a family of functions $y = f(X, \beta)$.

Finding β is an optimization problem, as the goal is to find a set of parameters for a given function that will produce the most accurate output. Parameters can be of two types, model parameters and hyperparameters. Model parameters will be “learned” during training, supervised or unsupervised. Hyperparameters, on the other hand, are set by the creator and gives instructions on the structure of the model and how it should calculate its model parameters [22]. While the user does configure these values for the model, there are many different

ways to find the best set of hyperparameters, this is called hyperparameter optimization. An example of a simple approach is grid search, which trains models on all different values for hyperparameters by a given accuracy [23]. Others include gradient-based optimization [24], bayesian optimization [25], and random search [26].

Machine learning problems come in many forms, the most common distinction can be explained by if a problem is supervised or unsupervised learning. Supervised learning is when a model is trained on a dataset when y is known and unsupervised learning is when y is unknown. There is an obvious downside of unsupervised learning since y is unknown, the model cannot determine if it is correct or not [21].

2.6.1 Defining the Machine Learning Problem

If given a dataset, X , containing a list of flow data (source IP, source port, destination IP, destination port, and connection type), classify each data point as either normal traffic or non-normal traffic, y . Normal traffic would be as the name implies, normal, so we would expect for example: common IP addresses, similar connections, etc. Non-normal traffic could be malicious or non-malicious but either would be different from, doesn't follow the same distribution as, normal traffic.

2.6.2 A One Class Problem

A class in machine learning is a subset of data that all share the same characteristic. An example would be classifying cookies as either chocolate-chip or oatmeal raisin. In this case, we have two classes: chocolate-chip and oatmeal raisin, but we could have up to however many classes given/needed.

In our dataset, we were only given a single class: normal traffic, y is always 1. However, we have two classes defined: normal traffic and non-normal traffic. Since we were only given normal traffic and not non-normal traffic, this turns our machine learning problem into a one class problem. There are many difficulties when working with only one class, the largest of these difficulties is training a model. Typically, when training a model that classifies, also known as a classifier, there are two or more classes defined because if there is only one class defined the classification would be trivial, always the first class.

This is problem is similar to unsupervised learning, since we don't have a second class, it effectively means we cannot supervise the model as it trains. However, since we are assuming there is no non-normal traffic in our sample data, we will try to create a second class to use supervised learning rather than using any unsupervised learning strategies.

2.6.3 Deep learning

In machine learning, there are many different models for many different purposes. A common model is a neural network, with the most basic neural network being an artificial neural network (ANN) [27]. An ANN consists of an input layer, hidden layers, and activation functions. The input layer is the same dimension of the input data while the hidden layers can grow larger and contain weights that can be updated during training. With these weights, the neural network will calculate a value that goes through an activation function. A common

activation function for hidden layers in an ANN is ReLU, linear for non-negative values and 0 for any negative values [28]. At the final layer, usually a different activation function is used to get the data into its final form. A common activation function for the last layer is softmax, a logistical function for higher dimensions (see equation below) [29].

$$\text{ReLU: } f(x) = \max(0, x)$$

$$\text{softmax: } \sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

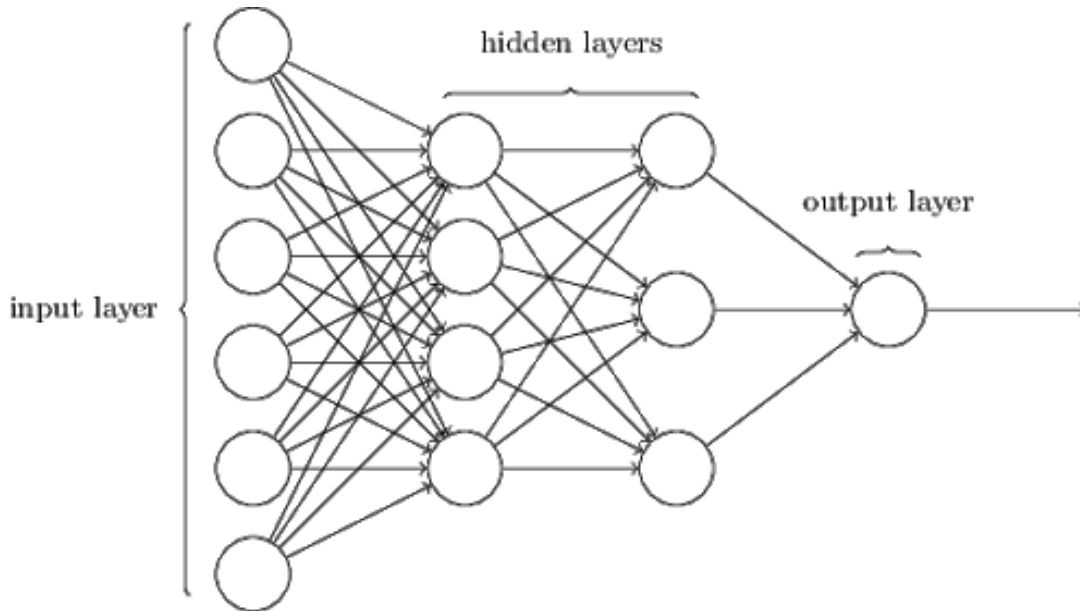


Figure 5: An example of a neural network with an input layer of six, two hidden layers with a size of four and three respectively and finally an output layer of one. Each line in the network will contain a weight that will be updated during training. At each node in the hidden layer contains an activation function which normalizes the value calculated from the weights. [27]

Deep learning is pushing the boundaries of neural networks by using many layers, often times using large layers. By increasing the size of the neural network, this adds a lot parameters into the model. By having lots of parameters to work with, the model can train on large amounts of data and fine tune all the weights to produce a very accurate result. While naively thinking increasing the size of the neural network can produce more accurate results, by having too large of a neural network can lead to problems [30]. If the size of the neural network grows to a dimension greater than the true dimension of f would certainly lead to overfitting. Overfitting is when the model can fit itself to the input data as it would be able to map each input to its correct output. An overfit model will be really accurate for the training data but will most likely fail when tried on new data it has not seen before [31]. Increasing the size will also lead to more work for the creator since the increasing number of hyperparameters that will have to be set and if the hyperparameters are not ideal or near ideal, it can lead to poor results.

2.6.4 Related Machine Learning Work

While we are taking a unique approach by pairing a machine learning model with user inputted information to create firewall rules, our work overlaps previously done work in the field. Since one of our goals in the project is to take a one-class problem and supplement it with a second-class of data, we use generative adversarial networks to create partial flow data. This has been attempted and done in different scopes such as simulating flow-based network traffic [32]. Since the scope of this project is not trying to create realistic network traffic but rather to create network traffic on a specific network, we had to take on our own approach in creating a network that outputs the data required for our network. Since networks will output a set of continuous values but instead we want a set of categorical values that represent plant area, port, connection type, etc. we need a generator that will output multi-categorical values. Previous work in this area has been done with using generative adversarial networks to generate multi-categorical samples [33], we can use similar network structures to produce similar results. While we will be using different strategies of creating a second class of data, there has been previous success in anomaly detection with training a model only on one-class [34]. A machine learning model was trained on an extremely large dataset consisting of only one class and was able to correctly identify outliers from the rest of the data as anomalies, correctly guessing the second class of data.

2.7 Industrial Engineering Application

In Industrial Engineering, we are concerned more about the operations of whole systems. Industrial engineers often concern themselves with finding the effective ways to utilize resources [35]. In the context of this project, we are focused on how various industrial components of the current system communicate within the computer network. In particular, we analyzed the network data using queuing theory, regression analysis, and correlation matrix. It can provide important insight into the performance of real world systems. For example, the relationships between parameters such as bytes used, bit rate, number of devices, and duration can be used as proxies to determine which manufacturing factors depend on each other. It would be interesting to demonstrate if network parameters such as bytes used, bit rate, and number of devices can be used to predict which components of the manufacturing system depend on each other. Furthermore, there are clear patterns from the network data. For example, there is a subset whose network type is miscellaneous, and repeats itself around every 20s. To make our analysis precise, we chose to focus finding correlations between different manufacturing elements at *different time shifts*. For example, if machine Acid Plant always activates 30 seconds before machine Blast Furnace then one might be able to conclude that machine Blast Furnace depends on machine Acid Plant to finish before it can start its processing.

2.7.1 Queuing Theory

Queuing is a phenomenon often encountered in daily life, examples of queuing include customers buying goods in grocery stores, patients waiting in a hospital for service, and queuing for toilets. Studying how queues fill and empty can lead to a deeper understanding of how packets are transmitted among network nodes within the current system.

In our particular case, packets are queued into the memory buffers of network devices such

as routers and switches. The packets in the queue are usually arranged in first-in, first-out order, like a normal queue [36]. We do not have data that could directly show how these packets queue. However, we can use the network data we do have to evaluate how well the system performs based on an analysis parameters such as bytes used, bit rate, number of devices, and duration.

Queuing theory is a branch of mathematical operations research which studies the randomness of queuing phenomena in service systems. In computer networks, queuing theory are widely used in manufacturing, transportation, inventory maintenance, and other resources sharing service systems [37]. Using queuing theory, one can often determine the stability or traffic intensity of the current queue by looking at arrival rate and service rate.

In our case, the data we received from our sponsor is extracted from network packets captured by Wireshark [38]. The data describes the flow of information among different machines. Such information includes IP Addresses, port numbers, bits rate, duration, packets, network type. In our analysis, we mainly focused on packets, bytes used, number of IP addresses and time related parameters. However, this analysis is limited because the physical conditions of the network devices are unknown.

After analyzing the data we got from our sponsor, we distinguished that there is one center hub that all information passes through. Therefore the network is best described by the M/M/1 queue model in this context. M/M/1 queue model is a queue model where the first M stands for arrivals following the Poisson process, the second M stands for service rate following exponential distribution, and there is only one server [39]. Arrival rate is the number of arrivals per unit of time, and service rate is the number of services provided per unit of time. We denote the arrival rate as λ , and the service rate is μ . We can also define the stability or traffic intensity as ρ and this can be used to describe the performance of the current system

$$\rho = \frac{\lambda}{\mu}$$

We require that $\rho < 1$, $\rho < 1$ denote the system reaches a steady state. If ρ greater than 1, there would be an exploded queue length, which means the system is unstable [17]. We would need to find out the bottlenecks of the current queue.

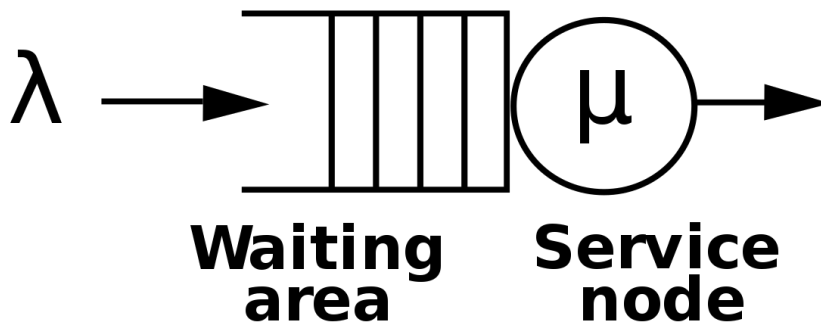


Figure 6: An Example of How M/M/1 Model Works with Arrival Rate λ and Service Rate μ .

In the queuing theory, when the service rate μ is less than the arrival rate λ , then ρ is high, which leads to system instability. ρ greater than 1 indicates that the arrival rate exceeds the service rate. Therefore, for a stable system, ρ is less than 1. When ρ is high, performance degrades, which can lead to high latency and packet loss [17].

2.7.2 Regression Analysis

Network traffic is an important indicator of network operation, which reflects the running state of the network. Network traffic characteristics is the premise of network design planning and network analysis. The research of network traffic prediction model has far-reaching significance for better understanding the performance of network service, planning network design, deciding network congestion control, applying to network security, detection of abnormal network activities, and improving service quality [40]. Other than finding the potential bottleneck of the system, we are also interested in research on if one parameter can be predicted by another parameter. Multiple Linear Regression is a useful tool. It is a mathematical technique which uses several independent variables to explain dependent variables. Dependent variables are the parameters we want to predict, and independent variables are the factors that might have an impact on the dependent variables [41].

The formula of Multiple Linear Regression is:

$$Y = a_0 + a_1X_1 + a_2X_2 + \dots + a_nX_n + e$$

where Y is the dependent variable, $X_1, X_2, X_3, \dots, X_n$ are independent variables, and $a_0, a_1, a_2, \dots, a_n$ are constants which needed to be determined, and e is the residual error.

In our analysis, for example, we would let:

$$Y = \text{Bytes Used}$$

$$X_1 = \text{Number of Devices}$$

$$X_2 = \text{Duration}$$

In this example, we want to know if the number of devices and duration would have an impact on the bytes being used, and the analysis was done based on captured flow data.

Additionally, multiple regression analysis involves correlation coefficient, coefficient of determination, and p-value.

Correlation coefficient, also called Pearson's R (r), is used to measure how strong a relationship is between two variables [42].

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

, where x_i is the values of x variables in the sample, \bar{x} is the mean of the values of x variables in the sample, y_i is the values of y variables in the sample, and \bar{y} is the mean of the values of y variables in the sample.

The coefficient of determination, R^2 , is used to analyze how differences in one variable can be explained by differences in a second variable.

The coefficient of determination can be thought of as a percentage. It gives an idea of how many data points fall into the result of the line formed by the regression equation. The higher the coefficient, the higher the percentage of points falls into the line, the better the fitting of the given values [43]. P-value is the probability of obtaining a result that is at least as extreme as the observation of hypothesis testing, assuming the null hypothesis is true [44].

In our analysis, we will use correlation coefficient and coefficient of determination, and p value to determine how the estimated regression model fits. Pearson's R can show us how strong the relationship among number of users, duration, and bytes used. Also, R^2 can indicate what percentage of bytes used can be explained by number of users and duration. P value can be used to determine if number of users and duration are significant parameters of the bytes being used.

2.7.3 Correlation Matrices

A correlation matrix is a table showing the correlation coefficients between sets of parameters. This table allows us to see which pairs have the highest relevance [45]. The matrix describes the correlations among all possible pairs. It is a powerful tool for summarizing data sets and visualizing potential patterns from the given data [36].

In our analysis, we focus on finding correlations between different manufacturing elements at *different time shifts*. From the correlation matrices, we might find potential patterns including, repeating the data itself and special ordering of the process.

3 Methodology

3.1 Fixing the One Class Problem

Although only given one class in the dataset, we still have two classes the data could be classified as: normal traffic and non-normal traffic. Specifically, we were not given the non-normal traffic in our sample data. This means the data still can be a second class, and we can train a classifier to determine based on a traffic flow, if it is normal or non-normal traffic. However, we will need to train the model on both classes and since we only have one class in our sample data, we will need to create the second class, non-normal traffic.

Creating realistic non-normal traffic is highly complex so instead of trying to create realistic non-normal traffic, we can create unrealistic non-normal traffic which will still be usable to train the machine learning model. The goal for this second class isn't to model real-world malicious traffic but to supply a "good-enough" second class where the model can still learn the distribution of the normal traffic.

The most naive approach for a second class is to create completely random network traffic. There are five values to randomly create: source IP, source port, destination IP, destination port and connection type. For source IP and destination IP, we can take a random IP address from the set of IP addresses in the sample data. For source port and destination port, we can randomly select a port number using a Python library, Faker [46]. And for the final value, connection type, we can randomly select from TCP or UDP, the two connection types available. Another simple method for creating a second class is taking a sample from the real traffic and modify one of the values to a different value. This way we have an almost real sample that is only slightly different. This can simulate malicious tactics like IP spoofing, where the user modifies their source IP address to an IP address that appears to be normal, e.g. an internal computer on the network. While this method does create random network flow data, it does not create believable network traffic, so we need to take a more advanced approach to create more believable network traffic.

3.2 Generative Adversarial Networks

Currently, we have mentioned one type of machine learning model, classifiers. Classifiers are an example of a discriminator, a machine learning model that is used for classification or regression. Another type of model in machine learning is a generator, which tries to generate new samples from the same distribution. A generative adversarial network (GAN) is a type of generator that utilizes two neural networks, one that is a generator and the other is a discriminator. The generative network will take in random noise and generate new samples. The new samples are then evenly mixed with samples from the training set and are inputted into the discriminator. The discriminator will try to classify the data as either real data points or generated data points. The generator and discriminator will play a zero-sum game, as the generator tries to produce better samples to “trick” the discriminator, and the discriminator will try to “out-sArt” the generator by getting better at determining real versus generated data. Below is the loss function for the GAN [47]:

$$E_x[\log(D(x))] + E_z[\log(1 - D(G(z)))]$$

In the equation above, E_x represents the expected value over the real data and E_z represented the expected value over the generated data, where $D(x)$ is the discriminator’s probability the data is real and $G(x)$ is the generated data based off random noise from the generator. That means $D(G(x))$ is the discriminator’s probability the generated data is real. Since we want the probability that the generated data is not real, we take the inverse of this probability. We then can use these values into the log-loss function to create our final loss function [47].

GANs as mentioned above, utilizes neural networks for the discriminator and the generator. Typically, GANs are used with images, so a specific type of neural network is used, a convolutional neural networks (CNN). A CNN is similar to an ANN but includes convolutional layers, hidden layers that filter and abstract the input, and pooling layers, layers that follow convolutional layers that reduce the input to a small dimension. However, since the discriminator is trying to compute the probability the data is real, the network uses convolutions but since the generator is trying to create fake data, the network uses de-convolutions, the inverse of convolutions. In the case of our data, we do not need to use CNNs and can use ANNs instead since we are processing network traffic which only lies along a single dimension unlike images which have a strong connection between their two dimensions.

Our more advanced approach to create more believable network traffic will use a GAN that will be trained on the sample data and use the generative model from the GAN to produce new data points. These new data points will be similar to the sample data but still not be in the same distribution. This is important since if the data was so similar that it would be more similar to normal traffic, that we could not use it for non-normal traffic. We can say, if the discriminator has a small loss, that the new data points are not similar to the sample data, and if there is a large loss, that the new data points are similar to the sample data.

3.3 Featuring Engineering Network Flow Traffic

Network flow traffic contains five categorical values: two IP address, two port numbers, and a connection type. While IP address and port numbers are numbers or a list of numbers, these are still categorical values. For example, five star ratings are also categorical numbers since the distance between a two star and three star rating is not necessarily the same from a four star and five star rating. We will need to take our categorical data and feature engineer them into numerical values with meaningful distances.

Luckily, we are given an IP inventory of the given sample dataset. This means we can map IP addresses to more information such as plant area, device number, etc. If we map the IP addresses to their plant area and then one-hot encode the values, we now have IP addresses that are numerical values with meaningful distances. However, we may have usable features for IP addresses, we have lost some information in the process. By converting the IP address to a plant area, we lost the information stored in each octet. Using our resources from the company, we know for the most part that IP addresses in the same plant area will have similar IP addresses. So for our purposes, only a small amount of information is lost and if needed we can create an addition feature that could be a one-hot vector of the IP addresses subnet.

Port numbers while already numerical values, do not have a meaningful distance. For example, port 443 is used for Hypertext Transfer Protocol Secure (HTTPS) and port 444 is used for Simple Network Paging Protocol (SNPP) [48]. While port 443 and 444 are only a distance of 1 away, these protocols are not similar. Port 80 is used for HTTP which is very similar to HTTPS but is a distance of 363 away. As we are not given a list of commonly used ports specific to the sample data, we can determine which are commonly used in the dataset and look the ports up with a reference. For example, we could map both port 80 and port 443 to “HTTP” and then one-hot encode all of the mappings.

3.4 Firewall Rule Syntax

It is important to define what syntax we will be using for firewall rules. While there are many different syntaxes for firewall rules, they all follow similar patterns. The syntax we chose to follow was Triton. We can define a new rule as being:

“FROM <source> TO <destination> <ALLOW/DENY> <tcp/udp> PORT <port>”

We can define to source and destination as a list of tags using conditional operators: *or* and *and*. A tag can be either plant area, subnet, or a user defined value. We can also give a list of ports, for example, if we wanted to allow on HTTP/HTTPS, we could say “FROM any

TO any ALLOW tcp PORT 80 and PORT 443”.

3.5 Selecting a Machine Learning Model

There are many machine learning models that classify data but our machine learning problems has specific restrictions. We need a model that will be able to be converted into a list of firewall rules. One model that jumps out with similarities are decision trees. Decision trees are a tree where each node is a decision made on a feature that splits the data into two groups. Looking back at our example determining fitness based off height and weight, one node in the tree can have the decision if height is greater than 170cm move left else move right. The decision tree could then look at the people less than 170cm and make another decision if weight is greater than 70 kilos this person is unfit else this person is fit. This is an example of a leaf node, a node where the decision tree classifies the group after the final split. We can apply this to our dataset, a possible decision the tree can make is if the traffic tried to connect to port 80 or not. We can then use this decision in our rule, interpreting it as allowing network traffic if port 80 is being used. By starting at the root node and going down every path to a leaf node, we can create discrete firewall rules directly from the machine learning model.

While a decision tree can be converted into firewall rules, is this the best model to use? Well, we can take a look at one of the short comings of using a decision tree to create firewalls. If we go down each path in the decision tree and create a firewall rule, the root node will be present in every rule. Is this a bad thing? Well, we know that the decision tree will try to create the best tree it can based on the training and testing set, the root node should be a feature with one of the highest variances but we may not want all of our rules to be based off that feature. To avoid our rules being over-fit on one decision, we can create multiple decision trees and examine the different set of rules produced. Creating multiple decision trees is a form of ensemble learning and is called a random forest. However, since our ensemble step is after converting the trees into firewall rules, we cannot use the same ensemble methods for a typical random forest. We will have to do our own method of taking multiple sets of firewall rules and selecting which rules we want to use and which rules we will throw out.

3.6 Evaluating the Machine Learning Model

After we have our machine learning model selected, we have to evaluate how well the model does. While there are many ways to determine how successful a machine learning model is, we will focus on cross-validation. Cross-validation is when a portion of the training dataset is left out to be used to evaluate the accuracy of the model after it has been trained on the remaining training dataset. This process can either be done once or “K” number of times, where “K” is the number of groups the training data can be split into. For example, if the training dataset has 1000 samples, we can split that data into 10 groups of 100 samples. Each group is used as the test set once while the 9 other groups are used for training. This is called K-fold cross-validation, in our example we used 10-fold, each group of data being one fold [49].

Another important aspect to evaluating the model is not only what percentage the model is right but when it’s wrong, whether or not it’s a false-negative or a false-negative. In our

case, a false-positive represents when the firewall blocks normal traffic and a false-negative is when the firewall allows non-normal traffic. While blocking traffic that should be let through is not ideal, it is not as dangerous as allowing traffic that should not be allowed. We should also use an additional metric to evaluate our model then, rate of false-negatives. A better model will then have a lower rate of false-negatives, allowing less potentially malicious traffic on the network.

3.7 Developing the User Interface

After getting a very rough version of the machine learning algorithm functioning, it was time to shift the attention over to the user interface. The user interface would provide Arco IT GmbH with a place to import and modify flow data, the asset inventory, IP ranges, applications, and rules. The asset inventory, IP ranges, and applications would be necessary in order to activate the machine learning generation, whereas the flow data and asset inventory would be more-so of a visual cue for the user.

Developing this roadmap of features was extremely useful, but it was vital that the appropriate environment was configured with an end goal in mind. While some Python tools can develop an eye-catching user interface, it might not be particularly easy for Arco IT GmbH to install on their end. A more functional user interface is far more viable than one based off of aesthetics since it ensures usage.

[143]:

	Address A	Port A	Address B	Port B	Packets	Bytes	Packets A -> B	Bytes A -> B	Packets B -> A	Bytes B -> A	Rel Start	Duration	Bits/s A -> B
0	192.168.100.2	60371	157.55.56.173	80	3	194	3	194	0	0	877.579045	8.999548	172.453105
1	192.168.100.2	60372	157.56.52.35	80	3	194	3	194	0	0	877.579046	8.999548	172.453105
2	192.168.100.2	60373	111.221.77.170	80	3	194	3	194	0	0	877.580154	8.998441	172.474321
3	192.168.100.2	60374	91.190.218.52	443	3	194	3	194	0	0	877.750186	8.999809	172.448104
4	192.168.100.2	60375	91.190.218.52	80	3	194	3	194	0	0	879.360044	8.999510	172.453834
5	192.168.100.2	60377	65.55.223.17	443	3	194	3	194	0	0	886.119864	9.000660	172.431799
6	192.168.100.2	60378	65.55.223.17	80	3	194	3	194	0	0	887.730104	9.000373	172.437298
7	192.168.100.2	60381	157.56.52.41	443	2	132	2	132	0	0	898.261471	2.970304	355.519166
8	192.168.100.2	60383	157.55.130.155	443	2	132	2	132	0	0	899.291877	2.970637	355.479313
9	192.168.100.2	60387	157.56.52.41	80	2	132	2	132	0	0	899.851871	3.000662	351.922343
10	192.168.100.2	60388	111.221.77.162	443	1	66	1	66	0	0	900.302073	0.000000	0.000000
11	192.168.100.2	60389	157.55.130.155	80	1	66	1	66	0	0	900.881977	0.000000	0.000000
12	192.168.100.2	60390	157.55.235.153	443	1	66	1	66	0	0	901.342021	0.000000	0.000000

Figure 7: Shows the first stage of the User Interface. This stage only had a functioning import button that would require running a Jupyter cell in order to show the table pictured. The table produced was just the contents of the import file without data transformation.

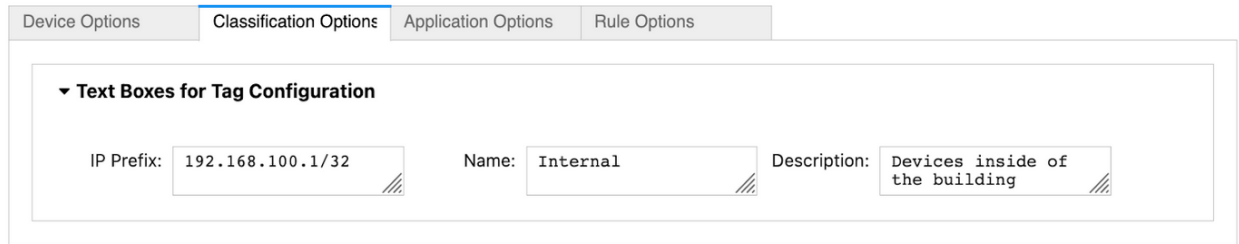


Figure 8: Shows the second stage of the User Interface. At this point, there was a functioning tab system consisting of “Device Options”, “Classification Options”, “Application Options”, and “Rule Options”. The device tab would allow the user the ability to import a CSV file in the form of flow data and would prompt the user to select how many rows they would like to see. There was another tab system that was generated after running a few cells which would have the user click a “Generate” button that would then output the contents of the flow data. This stage still didn’t have any data transformation. The classification, application, and rule tab only had functionality for text boxes, which would have the user create their own table of the corresponding information. For these tabs, much like the device tab, the user would have to manually run cells in order to generate another tab system that would contain a button labeled “add” that would allow them to add their entries into a new table of a CSV file created in the mounted file volume.

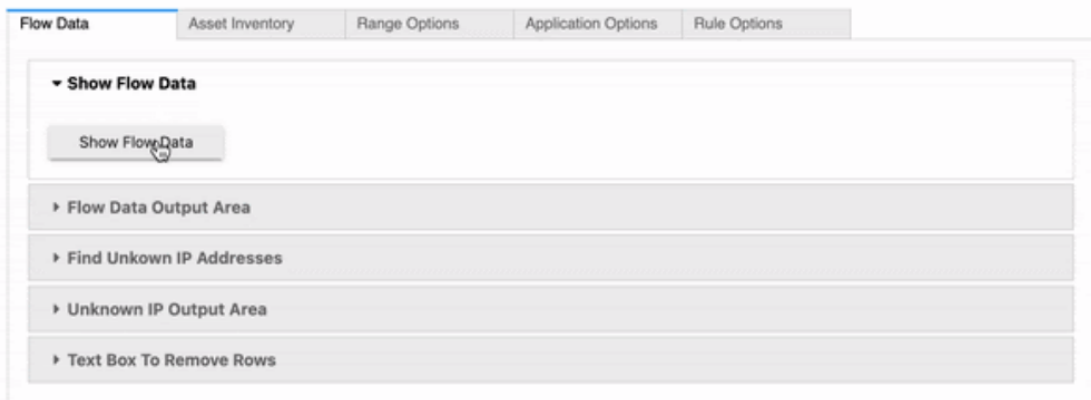


Figure 9: Shows the third stage of the User Interface. This stage fixed several issues that were present in stage two. For one, the user now had the ability to view all functionality within two tab system as opposed to several. In addition, the tab structure was revamped to target the functionality that was desired from Arco IT GmbH. The tabs were renamed to “Flow Data”, “Asset Inventory”, “Range Options”, “Application Options”, and “Rule Options”. The flow data tab now allowed the user to “Find Unknown IP Addresses”, and view the generated data from button presses in corresponding output areas.

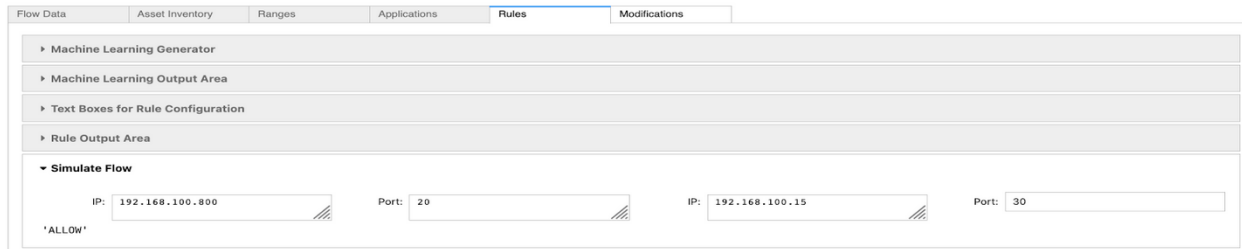


Figure 10: Shows the fourth stage of the User Interface. This was the final stage of the user interface presented to Arco IT GmbH. The user interface now made full use of observe statements and callbacks in order to present all functionality to the user after running a singular cell. In addition, there was another tab restructuring which added a modification tab. This tab would be used for anything that is related to either adding, removing, or editing data, whether that be for the flow data, the asset inventory, ranges, or applications. Finally, the final stage contained the machine learning portion of the project that generated firewall rules semi-autonomously using the data that was uploaded, entered, and modified from the previous tabs. Moreover, there was a flow simulator added to the rule tab which had the user enter in a source IP address and port and a destination IP address and port in order to use the machine learning generated rules to determine whether or not that flow would be allowed or denied.

While the very first stage of the user interface had practically no user interaction, it provided the foundation for the interaction that was to be had. It gave our team the starting point that we needed in order to arrive at the later stages. The second stage shows a significantly improved interface that now give the user space to enter data, click buttons, and change tabs. The third stage has a design change that was recommended by Arco IT GmbH after seeing the second stage of the user interface. As mentioned above, all storyboards are apt to change, and despite the fact that Arco IT GmbH designed the storyboard that we used for a structure, design implementations are constantly changing. In this stage, instead of the user having a device, classification, application, and rule tab... there is a flow data, asset inventory, range, application, and rule tab.

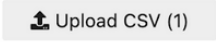
This change was made after a consensus that the user needed to have the ability to modify the asset inventory in addition to the flow data. Moreover, the manipulation of this data must be labeled accurately and visible for the user to navigate with confidence. The classification tab that is seen in stage two was converted into the range tab that is seen in stage three, with the application and rule tab remaining the same. The fourth stage kept the same tab system with accordions throughout that allowed the user to expand based on the type of functionality they were searching for, with the same tab categories as that in stage three. In addition to those changes, there was also the addition of the modifications tab which grouped together all of the accordion-style drop-downs within the tab system that allowed the user to modify sections of data. This change was reversed as it logistically made more sense to keep the corresponding modification functions within the respective tabs. For example, if the user needed to remove rows from the flow data, they wouldn't have to navigate to the modifications tab, and instead they would have the option to do so within the flow data tab.

3.8 Jupyter Notebooks, IPywidgets, and JupyterLab

The user interface that was created for Arco IT GmbH was in the form of a Jupyter Notebook that was primarily run through JupyterLab. Jupyter Notebooks is an open-source application that can be used to share live sections of code, equations, visuals, or even text on a cell-by-cell basis. Figure 11 displays the layout of a Jupyter Notebook. In our case, the notebook was configured with Python code that incorporated the use of IPywidgets, a python library dedicated to giving Python code a user interface feel.

```
[133]: import ipywidgets as widgets
      from IPython.display import display
      import pandas as pd

      #This is where the file upload button will be created
      upload = widgets.FileUpload(
          description = 'Upload CSV',
          multiple = False,
          accept = '.csv'
      )
      #display the button
      display(upload)

[136]: #this takes the resulting dictionary and assigns it to a variable
      uploaded_file_val = upload.value
      if(len(uploaded_file_val) == 0):
          raise Exception("You must input a file")
      #show that variable
      #uploaded_file_val
```

Figure 11: Demonstrates the cell-by-cell coding interface using IPywidgets to create a File Upload Button.

By running the cell on the top portion of the figure, we get a physical file upload button that will prompt the user to select a file. The attributes of the file upload will be stored inside of a python dictionary which we could then invoke specific lines of code to retrieve the contents. Instead of this code living in a traditional .py file, it lives inside of a .ipynb file that represents the Jupyter Notebook.

IPywidgets is an extensive library that enables the programmer to functionally generate interactive elements, custom styling, and a lengthy amount of layouts. With a series of built in function calls used specifically for specific actions, our team was able to tailor a user interface to the needs of our sponsor. The first function of importance is that of “observe”, which we discovered late in the project. Early on, our configuration utilized IPywidgets in a way that was useful for a demonstration version of a user interface on Jupyter Notebook form. For example, we would generate a file upload button, run some cells in between, then have them select how many rows they would like to see from that file, run more cells after that, and finally get an output demonstrating the results. While that might have been useful for giving a sequential walkthrough of code, the sponsor didn’t need that element. The “observe” call was extremely useful in observing each interactive element and performing what we needed it to do when that element was used.

To further put this into perspective, when a user would use an IPywidget slider to select how many rows they would like to view, on a file that they previously uploaded, they would automatically have the result placed in a certain location that would be predefined by us. This is when output areas proved mightily helpful. Another function of IPywidgets is the ability to configure output areas. This is pictured in Figure 12 in addition to the tab system and accordion layout.

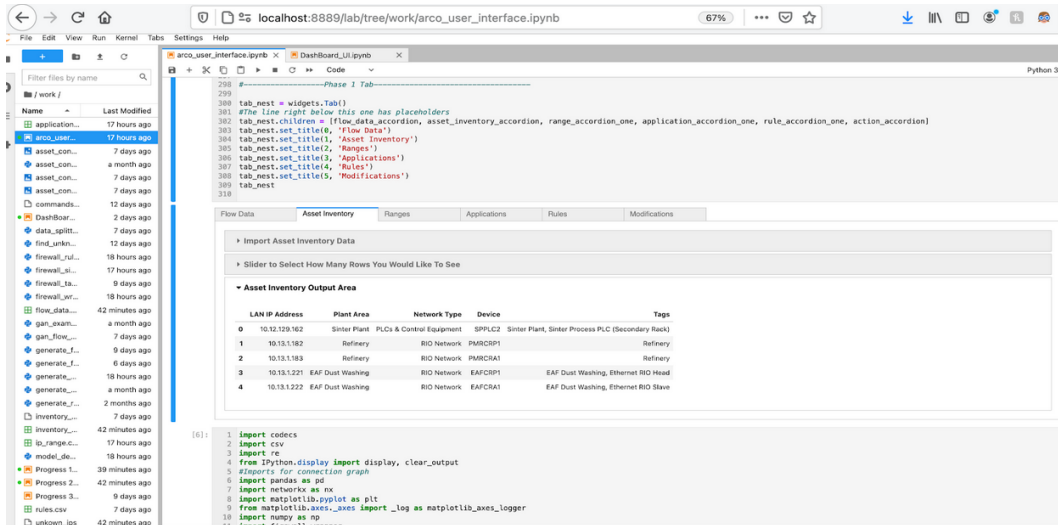


Figure 12: Demonstrates a tab system layout, with an accordion system that contains an output area to highlight the results of the asset inventory.

The tab system gives the user the ability to click between separate sections, each of which will have differing accordions. The “tabs” shown in Figure 12 represent the key sections of functionality that the user might need. The first tab contains functionality related to Flow Data, the second has Asset Inventory, third being IP Ranges, and so on. Within the second tab, we have an accordion system that can be dropped down depending on the desired action. In this case, we are interested in the output area. When the user first clicks on the tab, the first portion of the accordion system will be dropped down, however, for the sake of demonstration, the output area is open. When the user moves the slider to select how many rows they would like to see, the observe statement automatically waits for the final value, then uses the output area to output the results. This combination of tabs, accordions, observe statements, and output areas is what truly created the user interface originally in mind.

On the other hand, JupyterLab grants the programmer a user interface for managing all of the files that might be needed in order to create their user interface. In the previous section, we described how we used Docker in order to mount a volume to the container in order to, very fluidly, work on the project. JupyterLab gives a very neat space to manage these files, open a command line, and manipulate cells. This just scrapes the surface on the functionality of JupyterLab as there are numerous sections of functionality that are helpful in developing a smooth, and functional user interface using Jupyter Notebooks. One vital piece of JupyterLab is the ability to directly manipulate a terminal in order to install all of the Python packages necessary onto our Docker container. The terminal functionality also

allows the ability to view the file structure that is currently being used, and grants practically all other functionality that a normal terminal would allow for.

3.9 Dashboard and Callbacks

Despite JupyterLab being relatively easy to work with in order to create the UI from a developer's point of view, an interface in the form of a Jupyter Notebook, executed in Jupyter Lab looked very intimidating. The UI had many cells, which needed to be executed firstly in order to get the Interface up and running. In order to avoid this and make life a little easy on the user, we decided to employ Dashboards in order to display the UI as an interactive application. Dashboards are open source frameworks that are very useful for building analytical web applications. The biggest pro is that, a dashboard executes all cells in a notebook upon launch and displays them together in a format that is more interactive and application like. There are various dashboard tools such as Dash, Voila, Jupyter Dashboards etc. We experimented and researched each of these different dashboards, in order to determine which of them was best suited for our needs. Despite initially finding failure with Dash and Voila Dashboards, success came in the form of Jupyter Dashboards. In order to make this work we had to install the Jupyter dashboard dependencies and select the Dashboard option inside of the view tab of Jupyter notebooks. However using dashboards meant that some information would not be updated when the user made desired changes. To ensure that these updated changes were registered, the use of callbacks was implemented. A callback is a feature that helps a certain display or piece of information update correctly whenever there is an update to the information provided by the user. The way this works is by assigning a keyID to the area where the information is input by the user. This keyID is then fed into the functions that utilize the input. However, with IPython widgets, callbacks apart from observe could not directly be integrated into the input. Therefore we resorted to the use button that would complete the callback and update the information when clicked. The idea is that whenever the user has an updated file or information they input they click the callback 'Submit' button, updating information just like a regular application. We also worked on condensing the code by adding more callback events such as *'on_click'* and *observe* to the sliders, buttons and entry boxes. These ensured a smoother interface and also helped further cut down the entirety of the code to two cells.

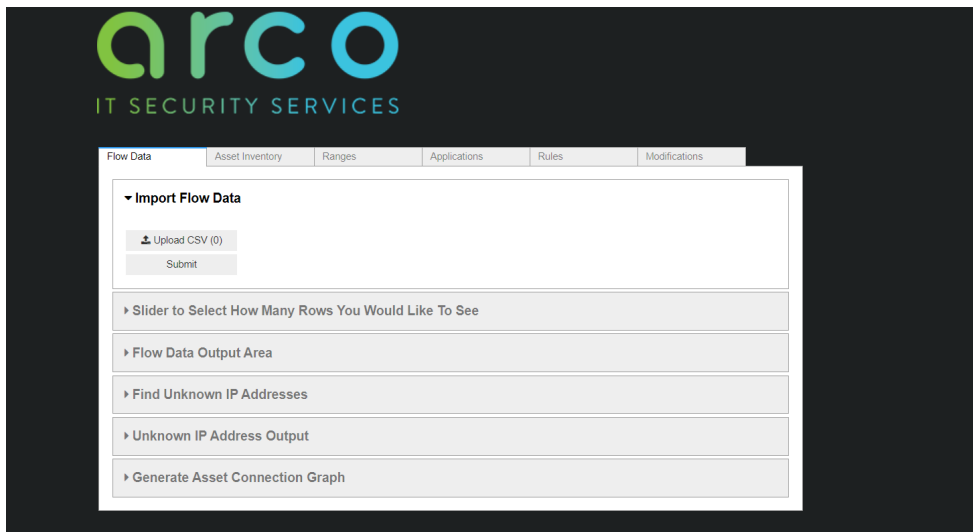


Figure 13: The dashboard layout.

4 Results

4.1 Model Performance

The model’s performance is partly determined by what data is fed into the model during training and testing. This is because our underlying machine learning problem is a one-class problem. Since we are creating the second class of data, if the second class is so distinct from the first class, the model’s accuracy would be near if not 100%. This can be counter-intuitive then, we want a model with a high accuracy but we also want a second class that is really close to the first class so that we have a robust model. We can look at our pipeline in two parts, the second class generation and the model training. For determining the best second class generation, we can select some values for the model’s hyperparameters that will be close to optimal but just a starting point so we can test the accuracies of each different generation strategy for the second class. We want to find the strategy that will produce accuracies that aren’t too close to perfect but also not so low as say, 50%. After we test and select the best second class generation strategy, we can then perform some hyperparameter optimization to find the model with the best hyperparameters.

4.1.1 Second Class Generation

Our most naive approach was to create purely random flow data partially based off the sample flow data. We expect this not to give a decision tree a difficult time determining normal traffic from the random traffic but it was a good starting point.

Trial	Accuracy
1	98.3%
2	98.9%
3	98.8%
4	98.8%
5	98.5%
Avg.	98.7%

Table 4: Five trials from training and testing a decision tree with a max of 50 leaf nodes on a 80% training 20% testing dataset. The first class is 1874 samples of normal traffic and the second class is 1874 samples of randomly generated flow data partially based off normal traffic.

All of the accuracies for our first approach are with 98% and 99%. Since we are not looking at which approach generates the highest or lowest accuracies, we need to carefully examine the accuracies. This approach yields a high and consistent accuracy, we can interpret this as the model easily determining the first class from the second class with a few errors. With our goal in mind, we want to choose a second class that will create an accurate and robust model. From these results, we only have accuracy but lack robustness. While this approach did not create a very good second class, we can take some inspiration from it and create a better second class using a similar strategy. Instead of each piece of data in the flow being purely random, we can take a random sample from the existing flow data and modify one of the columns to a different value. This combines the randomness from the naive approach to create new and different data but fixes the problem of the naive approach by now trying to more closely model the existing sample data. We could see this second class being more accurate to the real world non-normal traffic a firewall may encounter. A hacker may spoof their IP address but try to talk to a different device on the network that usually would never communicate data to each other, this new way will help create a more robust model by reinforcing these similar samples that are only slightly different.

Trial	Accuracy
1	65.5%
2	56.1%
3	55.6%
4	51.7%
5	48.4%
Avg.	55.5%

Table 5: Five trials from training and testing a decision tree with a max of 50 leaf nodes on a 80% training 20% testing dataset. The first class is 1874 samples of normal traffic and the second class is 1874 samples of randomly altered normal traffic.

The accuracy of the decision trees trained with our new approach does produce bad accuracies, these accuracies show the robustness the new second class forces the model to have. The model now has to look at a lot more features since we have randomly altered them. While adding a lot of robustness, it creates a model that is looking at every different value

for each feature which can lead to overfitting. While this approach is better than the first approach, we believe we can create an even better system to produce the second class of data. This is where GAN’s come to the advantage, if we can train a GAN on a large amount of existing network traffic, it can learn this data and create new samples that model the existing samples. Since we do not have a very large dataset, this approach will be difficult but believe it will greatly aid training robust decision trees.

Trial	Accuracy
1	100.0%
2	100.0%
3	100.0%
4	100.0%
5	100.0%
Avg.	100.0%

Table 6: Five trials from training and testing a decision tree with a max of 50 leaf nodes on a 80% training 20% testing dataset. The first class is 1874 samples of normal traffic and the second class is 1874 samples of generated flow data from a GAN trained on 1874 samples of normal traffic.

As we can see, using a GAN to create the second class can lead to a non-robust model in our case. However, looking into our sample dataset, it is much too small to properly train a GAN. We can keep our GAN as part of our second class but supplement them with a portion of the randomly altered data to reap the benefits of a robust and accurate model but also in the future use more sample data to better train a GAN to improve our robustness to an even greater extent. By having a combination of datasets in our second class, we can more accurately model real life circumstances since there is a large variety of non-normal traffic that can take place on a network.

Trial	Accuracy
1	93.3%
2	92.5%
3	94.3%
4	92.4%
5	93.9%
Avg.	93.3%

Table 7: Five trials from training and testing a decision tree with a max of 50 leaf nodes on a 80% training 20% testing dataset. The first class is 1874 samples of normal traffic and the second class is 187 samples of randomly altered normal traffic and 1687 samples of generated flow data from a GAN trained on 1874 samples of normal traffic.

Now having a solid second class, we can take these over and fine tune our machine learning model’s hyperparameters by looking more closely into our accuracies.

4.1.2 Model Accuracy

Now that we have our strategy for creating our second class, we can start to optimize the hyperparameters of the model. The two hyperparameters to optimize are the maximum number of leaf nodes, and the biases of each class. By limiting the number of leaf nodes, we can reduce the number of firewall rules created. We want to find the right balance between lowering the number of firewall rules while maintaining a high accuracy. Since we aren't only looking at the accuracy of the model but also the false-negative rate, we can adjust the bias of the second class so that there is a larger penalty when getting a false-negative during training.

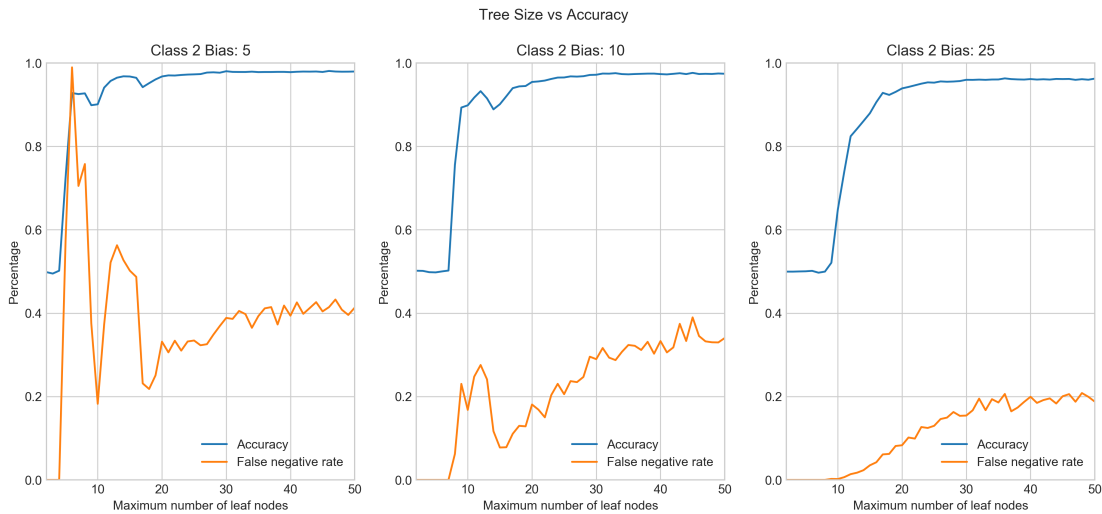


Figure 14: Accuracies of decision trees with a maximum number of leaf nodes from 2 to 50 averaged over 50 trials. Accuracies are further broken down by the bias for class 2 to lower the false-negative rate, allowing non-normal traffic through. False-negative rate decreases as the bias increases and accuracy increases as leaf nodes.

In figure 14, the accuracy increases as the number of leaf nodes increases with all three different bias values for the second class. While the accuracy slightly lowers with a higher bias value, in all cases the accuracies flatten out around 25 leaf nodes. This would be a good value to max the number of leaf nodes at to limit the number of rules the model will be converted to. The false-negative rate decreases as the bias for the second class increases. While we could test even greater biases for the second class, the accuracy decreases so we should limit how high the bias is for class 2, for our case, 25 produces the best results. One side-effect of increasing the number of leaf nodes is that it also increases the number of false-negatives. This is most likely due to the increase number of leaf nodes that resolve to the first class, an increase in first class leaf nodes can open up to more false-negatives. However, the false-negative rate seems to flatten out as the number of leaf nodes increase and does not increase any further which we can conclude that there is some stability within balancing between our two metrics, accuracy and false-negative rate.

5 Industrial Engineering Application

5.1 Methods

The goal of this portion is analyzing the performance of the current computer network that the various industrial components use to communicate. The performance of the current network can be impacted by many other factors which can result in numerous changes that make the network traffic unpredictable [17]. Analyzing network traffic is necessary to identify faults, errors, and may help to resolve future problems. In the context of this project, we are focused on how various industrial components of the current system communicate within the computer network. The objectives we identified to reach our goal were determining the performance of the current system, determining the predictability of one parameter by another parameter, and determining the correlation between shifted data sets.

5.1.1 Determining the performance of the current network

To determine the performance of the current network, we use queuing theory to analyze the flow data we are given. A flow is defined as a sequence of packets carrying information between two hosts [50]. We introduce three parameters that play important roles in the model: Arrival Rate (AR), Service Rate (SR) and Traffic Intensity (ρ).

Arrival Rate Arrival rate is the rate that the packets arrive at the center hub from a node within the current network. The arrival duration is the period of time over which we observe packets arriving across distinct flows. The unit of arrival rate of packets captured is the number of packets per second. In the following equation, n represents the number of flows, and a_i indicates the number of packets captured over the corresponding flow i .

$$AR = \frac{\text{Total number of packets captured}}{\text{Arrival duration}} = \frac{\sum_{i=1}^n a_i}{\text{Arrival duration}}$$

Service Rate Service rate can be obtained after the service time is determined. Service time is the time that the center hub takes to let packets pass through. Packet size is the number of bytes of the packet, and bit rate is the number of bytes that are transmitted per second. Service time is calculated by the division of packet size and bit rate for each flow. The service rate can be determined by the inverse of mean service time. In the following equation, n represents the number of flows, and b_i indicates the service time for its corresponding flow i .

$$\text{Service Time} = b_i = \frac{\text{Packet Size (bytes)}}{\text{Bytes Rate (bytes/s)}}$$

$$SR = \frac{1}{\text{Average service time}} = \frac{1}{\sum_{i=1}^n b_i}$$

Traffic Intensity The stability or traffic intensity is denoted as ρ , which can be used to describe the performance of the current system. The number of arrivals per time period is λ , which is the AR in our case. The number of customers being served per time period is μ , which is the SR we defined.

$$\rho = \frac{\lambda}{\mu} = \frac{AR}{SR}$$

By comparing the results of the parameters mentioned above, we would have a general sense of the performance of the current network. If $AR < SR$, then $\rho < 1$, denote the system reaches a steady state. If $AR \geq SR$, then $\rho \geq 1$, denoting the system is unstable [17].

5.1.2 Determining the predictability of one parameter by another parameter

Network traffic is an important indicator of network operation, which reflects the running state of the network. To determine whether one parameter is dependent on another parameters, the concept of regression analysis can help us better understand the process. Regression analysis is a way to determine the importance of one or more independent variables on another dependent variable. There are two kind of variables. Dependent variables are the parameters we want to predict, and independent variables are the parameters that might have impact on the dependent variables [41].

The formula of Multiple Linear Regression is:

$$Y = a_0 + a_1X_1 + a_2X_2 + \dots + a_nX_n + e,$$

where Y is the dependent variable, $X_1, X_2, X_3, \dots, X_n$ are independent variables, and $a_0, a_1, a_2, \dots, a_n$ are some coefficients that need to be determined, and e is the residual error.

Initially, we need to determine what parameters are our dependent and independent variables from the given data. To conduct a regression analysis, we used the built-in Excel tool called Regression [51]. Thus, we can generate regression statistics including correlation coefficient, coefficient of determination, and p-value to find out how well the estimated model fits the data.

With the rapid development of network communication technology, the network begins to carry more and more application services, which puts forward high requirements for network quality of service, flow control and network management. Traffic analysis and prediction is the basis of network management and performance analysis [40]. We want to know if the number of devices connected and duration would have an impact on the bytes being used, because this prediction can monitor networks to ensure good quality of service. In our analysis, we calculated out sum of bytes used, sum of duration time and number of devices connected every 100 seconds. The number of devices connected can be found counting the unique IP addresses per time interval. Therefore, the regression statistics can be generated for estimation.

5.1.3 Determining the Correlation Between Shifted Data Set

Traffic analysis not only can predict the trend of traffic behavior, but also can be applied to congestion control, quality of service and other fields [40]. To quantify the relevance degree between random variables, correlation is often used. For example, if we are interested in the relevance of the number of bytes used among the blast furnace, acid plant, and miscellaneous, we can denote them as X_1 , X_2 , and X_3 respectively, and look for correlations among them.

For computer networks, the correlation analysis is complicated by the fact that different devices are activated at different times. Thus, we can denote the number of bytes used by device i at time t as $k_{(i,t)}$. Our analysis is complicated by the fact that device i and device j are not activated at the same time. Considering time intervals is a useful perspective. Accordingly, we can denote the sum of bytes used by device i between time t_b and t_e as $totalbytes(i, t_b, t_e)$.

$$totalbytes(i, t_b, t_e) = \sum_{t_b < t < t_e} k_{(i,t)}$$

For example, device X_1 will use $k(X_1, t)$ bytes at time instant t and device X_2 will use $k(X_2, t_2)$ bytes at time instant t_2 . Thus, we can think about the problem in terms of time intervals. We can denote the sum of bytes used by device i within a time interval c as $Y_{i,c}$. Also, b is the beginning of the time interval c , and e is the end of the time interval c .

$$Y_{i,c} = \sum_{b < t < e} k_{(i,t)}$$

In particular, if device X_1 is active at times t_1, t_2, t_3, \dots then we should collect all activity times that lay in the given time interval c_1 . Also, b_1 is the beginning of the time interval c_1 , and e_1 is the end of the time interval c_1 .

$$Y_{X_1, c_1} = \sum_{b_1 < t < e_1} k_{(X_1, t)}$$

$Y_{X_1, (0,5)} = \sum_{0 < t < 5} k_{(X_1, t)}$ represents the sum of bytes used by device X_1 from 0 to 5 seconds. X_1 uses $k_{i,t}$ bytes at time instant t . More specifically, focusing on the Blast Furnace, at time instants t_1, t_2, t_3, \dots the Blast Furnace will use $k_{X_1, t_1}, k_{X_1, t_2}, k_{X_1, t_3}$ to represent the bytes used at corresponding time.

We are also interested in finding if any trends or patterns exists. Thus, we can shift the data by time interval with duration d . We can denote the sum of bytes used by device i within a time interval c , and shifted by time interval with duration d as $P_{i,c,d}$. Also, b is the beginning of the time interval c , and e is the end of the time interval c .

$$P_{i,c,d} = \sum_{b+d < t < e+d} k_{(i,t)}$$

In our case, $P_{X_1,(5,10),5}$ represents the sum of bytes used by device X_1 from 5 to 10 seconds, then shift the data set by a time interval which is 5 seconds.

$$P_{X_1,(5,10),5} = \sum_{10 < t < 15} k_{(X_1,t)}$$

The equation above expresses the sum of bytes used from 10 to 15 seconds by device X_1 .

Suppose for a given time interval of 5 seconds, we can generate a data set and denote it as BF :

$$BF = \{Y_{X_1,(0,5)}, Y_{X_1,(5,10)}, Y_{X_1,(10,15)}, \dots, Y_{X_1,(895,900)}\}$$

Furthermore, with a given time interval of 5 seconds, and shifted time interval of 5 seconds, we can also generate a shifted data set and denote it as $BF5$:

$$BF5 = \{P_{X_1,(0,5),5}, P_{X_1,(5,10),5}, P_{X_1,(10,15),5}, \dots, P_{X_1,(895,900),5}\}$$

Therefore, we can generate series of data using different combinations of given time interval c and shifted time interval with duration d . For each device, we will generate 4 series of data sets. Thus we have 12 data sets in total. For example, we can generate a series of data including, BF , $BF5$, $BF10$, $BF15$, AP , $AP5$, $AP10$, $AP15$, MS , $MS5$, $MS10$, and $MS15$. The built-in Excel tool called Correlation [51] can be used to generate a correlation matrix. A correlation matrix which would help us discover if there exists any patterns between different data sets. Furthermore, the matrix can show us if one particular data set can be predicted by another data set. Also, adding color to the correlation matrix would help us visualize the data. The absolute value between 0.1 and 0.2 is weak relationship, and the absolute value between 0.2 and 0.4 is moderate relationship. Furthermore, the absolute value is 0.4 or higher would be considered as strong relationship. Yellow cells means the correlation between this pair of the data set is weak, green cells means the correlation is moderate, and blue cells means the correlation is strong.

5.2 Results

The goal of this portion is analyzing the performance of the current computer network that the various industrial components use to communicate. The performance of the current network can be impacted by many other factors that make the network traffic unpredictable [17]. Analyzing network traffic is necessary to identify faults, errors, and may help to resolve future problems. In the context of this project, we are focused on how various industrial components of the current system communicate within the computer network. The objectives we identified to reach our goal were determining the performance of the current system, determining the predictability of one parameter by another parameter, and determining the correlation between shifted data sets.

In the following sections we describe the resulting data that we collected to achieve these objectives and our goal.

5.2.1 Determining the performance of the current system

To determine the performance of the current system, use queuing theory to analyze the data traffic we are given. We can determine the performance of current network by calculating Arrival Rate (AR), Service Rate (SR), and Traffic Intensity (ρ).

Arrival Rate Arrival rate is the rate that the packets arrive at the center hub from a node within the current network. The arrival duration is the period of time over which we observe packets arriving across distinct flows. The unit of arrival rate of packets captured was set to number of packets per second. To make our calculations concrete, we include a specific example to show how these values are calculated. In this case, we will use the particular example which all packets pass through the center hub. Thus, for this case, these values are:

$$\text{Total number of packets captured} = \sum_{i=1}^n a_i = \sum_{i=1}^{2169} a_i = 170,030,$$

where n represents the number of flows, and a_i indicates the packets captured in the corresponding flow i . Also, the number of flows can be found from the given data.

The total arrival time used can be found directly from the given data:

$$\text{Total arrival time used} = 902s$$

$$AR = \frac{\text{Total number of packets captured}}{\text{Total arrival time used}} = \frac{170,030}{902} = 158.61$$

Service Rate Service rate can be obtained after the service time is determined. In this case, service time is the time that the hub takes to let the packets pass through and can be computed using the packet size, which is the number of bytes in the packet, and bit rate, which is the number of bytes that are transmitted per second. Service time is calculated by the division of packet size and bit rate for each flow. The service rate can be determined by the inverse of mean service time.

$$\text{Service Time} = b_i = \frac{\text{Packet Size (bytes)}}{\text{Bytes Rate (bytes/s)}}$$

$$SR = \frac{1}{\text{Average service time}} = \frac{1}{\sum_{i=1}^n b_i} = 198.62,$$

where n represents the number of flows, and b_i indicates the corresponding service time for flow i .

Traffic Intensity We define the stability or traffic intensity as ρ and this can be used to describe the performance of the current system. λ is the number of arrivals per time period, which is AR in our case, and μ is the number of customer being served per time period, which is the SR we defined above. So, we can write

$$\rho = \frac{\lambda}{\mu} = \frac{AR}{SR} = \frac{158.61}{198.62} = 0.80.$$

From the calculation above, we know that ρ is 0.80, and SR is greater than AR . Therefore the system is stable, and reached a steady state [17] in this particular case.

5.2.2 Determining the predictability of one parameter by another parameter

The network traffic prediction is based on the past traffic data, and the future traffic state is predicted by establishing the appropriate mathematical model. Therefore, it is very important to understand the characteristics of network traffic to improve the accuracy of prediction and to analyze the essence of prediction.

The performance of the current network can be impacted by many other factors which can result in numerous changes that make the network traffic unpredictable [17]. The network prediction model proposed by Oluwadare et al. [17] can monitor the trend of network traffic so as to reduce network congestion and paralysis. We want to know if the number of devices and duration would have an impact on the bytes being used, whether this prediction can assist on monitoring and planning network usage to ensure good quality of service.

For example, we may want to know how the number of devices connected and the duration impact the number of bytes used.

Towards this end, we will run a linear regression with the number of bytes used Y as dependent variable and the duration X_1 , and the number of devices connected X_2 , as independent variables.

Before running our linear regression, every 100 seconds we calculated out the sum of bytes used (Y), the sum of duration (X_1) and the number of devices connected (X_2). The number of devices connected (X_2) can be found by counting the unique IP addresses.

Time Interval	Sum of Bytes Used	Sum of Duration	Number of Devices
0-100	35918478	18203.80314	64
101-200	845115	5129.15	55
201-300	62252	4667.15	45
301-400	26984	3755.24	35
401-500	642296	5079.89	40
501-600	71691	3867.54	59
601-700	20715608	4037.04	40
701-800	11528699	3761.36	42
801-900	50522	1934.01	45

Table 8: Summary of sum of bytes used, sum of duration time and number of users every 100s.

Now we can conduct a regression analysis using the data from Table 8. The built-in Excel tool called Regression [51] can be used to generate regression statistics including correlation coefficient, coefficient of determination, and p-value. The summary of the multiple regression model statistics are presented in Table 9 and Table 10. X_1 is the sum of duration and X_2 is the number of devices connected.

Model	Coefficients	Standard Error	t Stat	P-value
Intercept	7195431.24	16401958.97	0.44	0.68
X_1	2517.14	807.34	3.12	0.02
X_2	-286705.08	396274.83	-0.72	0.50

Table 9: Coefficients Table.

The estimated Y can be written as

$$Y = 7195431.235 + 2517.14114X_1 - 286705.0812X_2$$

The p-values can tell us whether X_1 and X_2 are statistical determinants of Y . At critical value of 0.05, only X_1 is an significant determinant of Y .

Model Summary	Statistics
Pearson's R	0.82
R Square	0.68
Adjusted R Square	0.57

Table 10: Model summary of regression analysis

Table 10 presents the summary of regression analysis. Pearson's R is used to measure how strong a relationship is. The value of R we obtained is 0.82, which indicates a positive weak relationship. R^2 is used to analyze how differences in dependent variables can be explained by differences in independent variables. The results shows that the duration and the number of devices connected can explain 67.72% of the difference in the sum of bytes used.

5.2.3 Determining the Correlation Between Shifted Data Set

Network traffic is an important indicator of network operation, which reflects the running state of the network. The research of network traffic prediction model has far-reaching significance for better understanding the performance of network service, planning network design, deciding network congestion control, applying to network security, detection of abnormal network activities, and improving service quality [40]. In our case, we are trying to understand if there are any patterns, or predictions exists so as to reduce network congestion and paralysis.

For computer networks, the correlation analysis is complicated by the fact that different devices are activated at different times. Therefore, we need to think about this problem in terms of time intervals. Furthermore, a correlation matrix which would help us discover if there exists any patterns between different data sets. Furthermore, the matrix can show us if one particular data set can be predicted by another data set.

Due to the inconsistency of data size of all groups, we only consider three groups, which are Acid Plant, Blast Furnace, and Miscellaneous (see Figure 15). The following groups were not included in our analysis because lack of data points, including Copper plant, EAF Dust washing, Fume control, Other Pronc Device, Powerplant, Refinery, Sinter Plant, Slag Fumer. The only determinant we consider is the sum of bytes used.

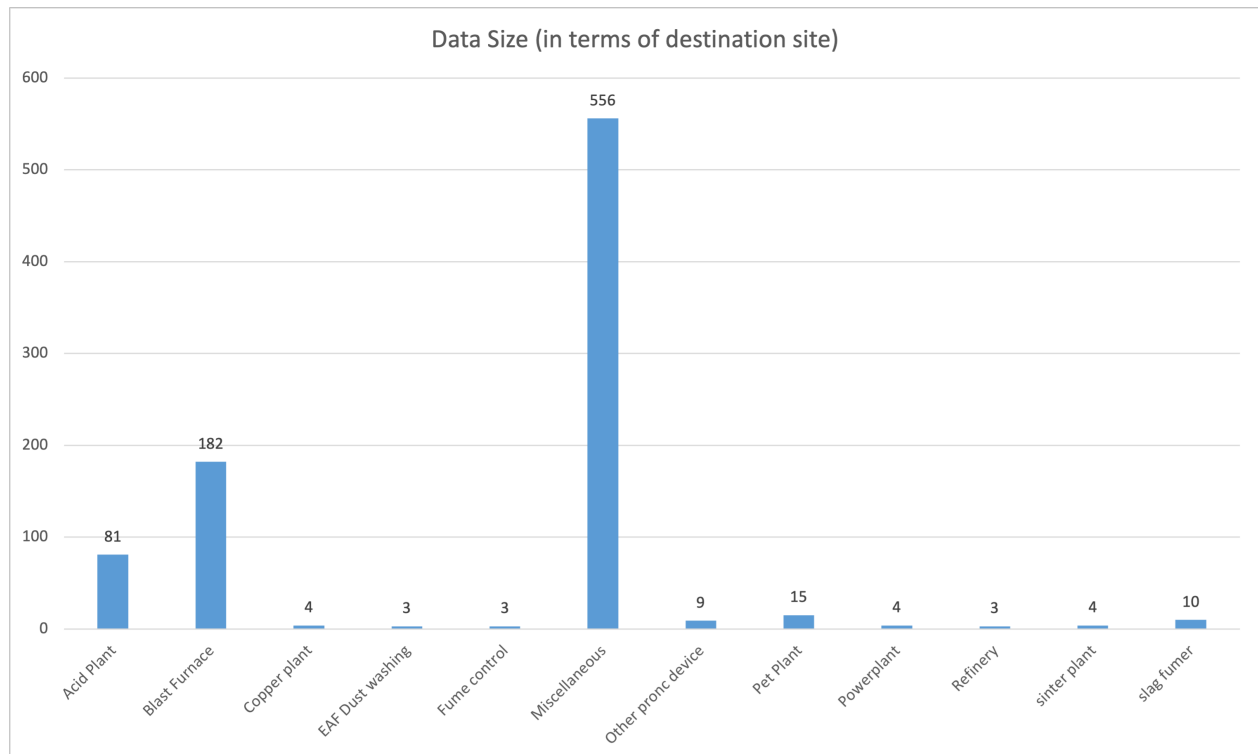


Figure 15: A bar graph showing the spread of data size in terms of their destination site. There are 556 data points in Miscellaneous, 182 data points in Blast Furnace, and 81 data points in Acid Plant.

We are interested in the capacity of the current network because we want to ensure a good level of service. The sum of bytes used is a indicator showing the network capacity. We can denote the number of bytes used by device i at time t as $k_{(i,t)}$, and the sum of bytes used by device i within a given time interval c as $Y_{i,c}$. Also, b is the beginning of the time interval c , and e is the end of the time interval c .

$$Y_{i,c} = \sum_{b < t < e} k_{(i,t)}$$

We can denote the sum of bytes used by device i within a given time interval c , and shifted by time interval with duration d as $P_{i,c,d}$. Also, b is the beginning of the time interval c , and e is the end of the time interval c .

$$P_{i,c,d} = \sum_{b+d < t < e+d} k_{(i,t)}$$

We can generate series of data using different combinations of given time interval c and shifted time interval with duration d using the equations above. For example, with a given time interval of 2 seconds and a shifted time interval of 2s, we can generate data series for device BF as:

$$BF = \{Y_{X_1,(0,2)}, Y_{X_1,(2,4)}, Y_{X_1,(4,6)}, \dots, Y_{X_1,(898,900)}\}$$

$$BF2 = \{P_{X_1,(0,2),2}, P_{X_1,(2,4),2}, P_{X_1,(4,6),2}, \dots, P_{X_1,(898,900),2}\}$$

After generating 4 series of data for each device with a given time interval of 2 seconds and a shifted time interval of 2 seconds, now we can generate the correlation matrix (Figure 16).

	AP	AP2	AP4	AP6	BF	BF2	BF4	BF6	MS	MS2	MS4	MS6
AP	1											
AP2	0.17931255	1										
AP4	0.13026638	0.17931255	1									
AP6	-0.0236222	0.13026638	0.17931255	1								
BF	-0.0078708	0.00297793	-0.0088147	0.01061208	1							
BF2	-0.0156764	-0.0078708	0.00297793	-0.0088147	-0.0066389	1						
BF4	-0.0155343	-0.0156764	-0.0078708	0.00297793	-0.002971	-0.0066389	1					
BF6	-0.0158402	-0.0155343	-0.0156764	-0.0078708	0.03574837	-0.002971	-0.0066389	1				
MS	-0.0336252	0.00080436	0.03753765	0.03856855	-0.008738	-0.0326523	-0.0383651	-0.0299801	1			
MS2	-0.0474716	-0.0336252	0.00080436	0.03753765	-0.0143658	-0.008738	-0.0326523	-0.0383651	0.68769672	1		
MS4	-0.0771258	-0.0474716	-0.0336252	0.00080436	0.07746612	-0.0143658	-0.008738	-0.0326523	0.34278125	0.68769672	1	
MS6	-0.023593	-0.0771258	-0.0474716	-0.0336252	0.10713702	0.07746612	-0.0143658	-0.008738	0.00477514	0.34278125	0.68769672	1

Figure 16: 2s shift correlation matrix for acid plant, blast furnace, and miscellaneous data.

In this figure, the absolute value between 0.1 and 0.2 is weak relationship, and the absolute value between 0.2 and 0.4 is moderate relationship. Furthermore, the absolute value is 0.4 or higher would be considered as strong relationship. Yellow cells means the correlation between

this pair of the data set is weak, green cells means the correlation is moderate, and blue cells means the correlation is strong.

The previous correlation matrix, which was shifted by 2 seconds has very weak correlations. Thus, we need to look at the correlation matrix which was shifted by 15 seconds. Using the same equations mentioned above, we can generate 4 series of data for each device with a given time interval of 15 seconds and a shifted time interval of 15 seconds, now we can generate another correlation matrix.

	AP	AP15	AP30	AP45	BF	BF15	BF30	BF45	MS	MS15	MS30	MS45
AP	1											
AP15	0.02232517	1										
AP30	-0.1330806	0.02232517	1									
AP45	-0.1534926	-0.1330806	0.02232517	1								
BF	-0.0831446	-0.0227585	-0.0520887	-0.0830947	1							
BF15	-0.0590711	-0.0831446	-0.0227585	-0.0520887	-0.0548544	1						
BF30	-0.0044907	-0.0590711	-0.0831446	-0.0227585	0.08715042	-0.0548544	1					
BF45	-0.0321023	-0.0044907	-0.0590711	-0.0831446	-0.0404571	0.08715042	-0.0548544	1				
MS	0.25466847	0.2329483	0.17124828	-0.1244401	-0.1378989	0.10565021	0.12213838	-0.0994133	1			
MS15	-0.2434214	0.25466847	-0.2329483	0.17124828	0.14489206	-0.1378989	0.10565021	0.12213838	-0.9027038	1		
MS30	0.17369741	-0.2434214	0.25466847	-0.2329483	-0.1350439	0.14489206	-0.1378989	0.10565021	0.76405101	-0.9027038	1	
MS45	-0.1802309	0.17369741	-0.2434214	0.25466847	0.1773948	-0.1350439	0.14489206	-0.1378989	-0.5457326	0.76405101	-0.9027038	1

Figure 17: 15s shift correlation matrix for acid plant, blast furnace, and miscellaneous data.

Comparing Figure 16 and Figure 17, we notice visually by the coloring that there are more significant correlation pairs in the 15s correlation matrix. Furthermore, the reason why there is a strong negative correlation between MS pairs is there is a break between each iteration of data, which happened to be around 15s.

We are interested in the combination of a short given time interval and a long shifted time interval. When the given time interval is short, less data points would be included when we calculate the sum of bytes used. Using the same equations mentioned above, we can generate 4 series of data for each device with a given time interval of 2 seconds and a shifted time interval of 30 seconds, which gives us another correlation matrix.

	BF	BF30	BF60	BF90	AP	AP30	AP60	AP90	MS	MS30	MS60	MS90
BF	1											
BF30	-0.0941176	1										
BF60	-0.0941176	-0.0941176	1									
BF90	0.45294118	-0.0941176	-0.0941176	1								
AP	-0.1004151	-0.1004151	-0.1004151	0.15582427	1							
AP30	0.42156283	-0.1004151	-0.1004151	-0.1004151	-0.107134	1						
AP60	-0.1004151	0.42156283	-0.1004151	-0.1004151	-0.107134	-0.107134	1					
AP90	-0.1004151	-0.1004151	0.42156283	-0.1004151	-0.107134	-0.107134	-0.107134	1				
MS	0.00471161	-0.0370198	0.10904011	0.19250292	-0.1953297	-0.1356025	0.10330643	0.43071926	1			
MS30	-0.0370198	0.00471161	-0.0370198	0.10904011	-0.1953297	-0.1953297	-0.1356025	0.10330643	0.50816389	1		
MS60	-0.0787512	-0.0370198	0.00471161	-0.0370198	-0.1953297	-0.1953297	-0.1953297	-0.1356025	-0.0720117	0.50816389	1	
MS90	0.00471161	-0.0787512	-0.0370198	0.00471161	-0.1953297	-0.1953297	-0.1953297	-0.1953297	-0.3227049	-0.0720117	0.50816389	1

Figure 18: 30s Shift - 2s given time interval correlation matrix.

By comparing Figure 16, Figure 17 and Figure 18, we can notice visually by coloring that there

are more significant correlation pairs if we make the shifted time longer, and the correlations between MS pairs decreases as shorter the given time interval, and longer the shifted time.

From the given statistics, we can notice that the strong correlations between MS pairs are strong. We can conclude that Miscellaneous data can be predicted by its own shifted data. To ensure the accuracy of the statement, we can calculate out the p-value to check if the correlations are significant.

Pairs	P Value
MS-MS30	0.0035
MS30-MS60	0.0035
MS60-MS90	0.0035

Table 11: p-Value for the MS correlations pairs.

From Table 11, p-value for all the Miscellaneous data pair, including MS-MS30, MS30-MS60, and MS60-MS90 are 0.0035, which is significant at 0.05 level. They have the same p-value, because their correlations are the same. Thus, we can conclude that Miscellaneous data can be predicted by its own shifted data.

We are interested in the correlations because we are interested in the dependency among the data sets. For example, if the device blast furnace is highly correlated with the device acid plant, the chance that when the device blast furnace breaks, the device acid plant breaks together is high.

5.3 Conclusion

Computer networks are usually designed to handle a certain amount of traffic. Once the fixed capacity is exceeded, it may result in a lower service quality. In our analysis, queuing model, regression analysis, and correlation matrix are used to analyze the network traffic. The queuing model is used to analyze the performance of the network. In addition, a multiple regression model was developed to determine the important determinants of the number of bytes used. The results show that the duration is the significant determinant of current network traffic. The model can be used for monitoring and forecasting purposes. Also, we identified Miscellaneous data can be predicted by its own shifted data using correlation Matrix. The results can assist in planning the use and monitoring of computer networks to ensure a good quality of service.

6 Conclusion

6.1 Key Points

Since the given data was non-diverse and the second class is generated, the machine learning model will almost always try to learn the second class of data rather than learning the first class of data. This is an important distinction because we want our model to learn the first class of data since network traffic will primarily be the first class, normal traffic. The model instead learns the artificial second class of data which will not be beneficial since our second

class, non-normal traffic, does not represent the whole, all different types of non-normal traffic. If the model learns this second class, it could confuse other non-normal traffic for the first class, normal traffic, giving a false-positive. This is especially bad in terms of firewalls, a firewall would rather give a false-negative, rejecting a network flow that is normal, rather than a false-positive, allowing a network flow that is non-normal, and potentially malicious.

The rules generated from the decision tree appear to create rules that may typically not be created by a human user but currently have a few vulnerabilities, by allowing a source or destination from “any” or allowing all ports is too open for a firewall and could let in traffic that could be malicious or just traffic that shouldn’t be on the network. While this is difficult to determine whether this is a direct flaw in the machine learning model or just a lack of a large and diverse dataset, the user should supplement the rules generated by the firewall or use these rules as a suggestion to create new rules, i.e. the user could see a rule but it may allow on all ports, the user can use the rest of the rule and modify to only allow on a specific set of ports.

6.2 Future Work

If repeated, larger and more diverse network traffic flows would allow more analysis of hyper-parameters in the machine learning model and could produce a more robust model that represents the first class of data, normal traffic. Recommended datasets would include: a large number of samples that contain a short capture of network traffic from different times of the day, week, and month or a few samples that contain a large capture of network traffic last a day to a few days. These larger datasets will capture a lot more samples and contain a lot more diversity in the network traffic. One could imagine that a device on a network only communicates to another device when an error occurs, which could only occur once a week. If the capture does not contain any of these network flows when these two devices communicate when an error occurs, the machine learning model could determine that this is non-normal traffic and reject the network flows from these devices preventing the device to communicate that there has been an error.

A further step that can be taken after the firewall rules are generated from the machine learning model could be to check their strength. These checks could be as simple as: do not allow any rules through that contain a “blank” section (“FROM any”, “TO any”, or “PORT all”) or adding an extra accuracy check by checking it against a set of non-normal traffic samples with a maximum error threshold. This would add an extra protection since the firewall rules are purely computer generated and can create weird rules especially if over-fitting occurs or if the user entered a tag incorrectly.

References

- [1] *15 Alarming Cyber Security Facts and Stats*. URL: <https://www.cybintsolutions.com/cyber-security-facts-stats/>.
- [2] *What Is Cybersecurity?* URL: <https://www.cisco.com/c/en/us/products/security/what-is-cybersecurity.html>.
- [3] *Arco IT*. URL: <https://arco-it.ch/>.
- [4] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, et al. “Deep one-class classification”. In: *International conference on machine learning*. PMLR. 2018, pp. 4393–4402.
- [5] *Designing Use Cases for a Project*. URL: <https://www.geeksforgeeks.org/designing-use-cases-for-a-project/>.
- [6] *What is Docker and why is it so darn popular?* URL: <https://www.zdnet.com/article/what-is-docker-and-why-is-it-so-darn-popular/> (visited on 03/21/2018).
- [7] *What is a hypervisor?* URL: <https://www.vmware.com/topics/glossary/content/hypervisor>.
- [8] *The Difference: Docker vs. Virtualization*. URL: <https://munzandmore.com/2015/cc/docker-container-vs-virtualization> (visited on 09/21/2015).
- [9] *Project Jupyter: Home*. URL: <https://jupyter.org/>.
- [10] *Hello World*. URL: <https://guides.github.com/activities/hello-world/>.
- [11] *Data Communications (DC)*. URL: <https://www.techopedia.com/definition/6765/data-communications-dc> (visited on 04/13/2021).
- [12] *Information, People, and Technology*. URL: <https://psu.pb.unizin.org/ist110/chapter/2-2-computer-networks/> (visited on 04/08/2021).
- [13] Ken Chen. *Introduction to Simulation*. John Wiley & Sons, Ltd, 2015. Chap. 2, pp. 11–19. ISBN: 9781119006190. DOI: <https://doi.org/10.1002/9781119006190.ch2>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119006190.ch2>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119006190.ch2>.
- [14] *Network Traffic Analysis*. URL: <https://awakesecurity.com/glossary/network-traffic-analysis/> (visited on 04/13/2021).
- [15] *What is Network Traffic Analysis (NTA)?* URL: <https://www.rapid7.com/fundamentals/network-traffic-analysis/> (visited on 04/13/2021).
- [16] *Network traffic analysis: what is it, and why do we need NTA systems?* URL: <https://www.ptsecurity.com/ww-en/analytics/knowledge-base/network-traffic-analysis-what-is-it-and-why-do-we-need-nta-systems/> (visited on 04/13/2021).
- [17] Samuel Oluwadare, Oluwatoyin Agbonifo, and Ayomikun Babatunde. “Network Traffic Analysis Using Queuing Model and Regression Technique”. In: *Journal of Information* 5 (Jan. 2019), pp. 16–26. DOI: 10.18488/journal.104.2019.51.16.26.
- [18] Kaspersky. *What is an IP Address – Definition and Explanation*. Jan. 2021. URL: <https://www.kaspersky.com/resource-center/definitions/what-is-an-ip-address>.

- [19] 2021. URL: https://www.tutorialspoint.com/ipv4/ipv4_address_classes.htm.
- [20] Jan. 2016. URL: <https://study-ccna.com/ports-explained/>.
- [21] Krishna Rungta. *TCP vs UDP: What's the Difference?* Jan. 2020. URL: <https://www.guru99.com/tcp-vs-udp-understanding-the-difference.html>.
- [22] Pier Paolo Ippolito. *Hyperparameters Optimization - Towards Data Science*. Sept. 2019. URL: <https://towardsdatascience.com/hyperparameters-optimization-526348bb8e2d>.
- [23] Rohan Joseph. *Grid Search for model tuning - Towards Data Science*. Dec. 2018. URL: <https://towardsdatascience.com/grid-search-for-model-tuning-3319b259367e>.
- [24] Dougal Maclaurin, David Duvenaud, and Ryan Adams. “Gradient-based hyperparameter optimization through reversible learning”. In: *International conference on machine learning*. PMLR. 2015, pp. 2113–2122.
- [25] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. “Practical bayesian optimization of machine learning algorithms”. In: *arXiv preprint arXiv:1206.2944* (2012).
- [26] James Bergstra and Yoshua Bengio. “Random search for hyper-parameter optimization.” In: *Journal of machine learning research* 13.2 (2012).
- [27] Michael A Nielsen. *Neural Networks and Deep Learning*. 2019. URL: <http://neuralnetworksanddeeplearning.com/chap1.html>.
- [28] <https://www.facebook.com/MachineLearningMastery>. *A Gentle Introduction to the Rectified Linear Unit (ReLU)*. Jan. 2019. URL: <https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/>.
- [29] Thomas Wood. *Softmax Function*. May 2019. URL: <https://deeppai.org/machine-learning-glossary-and-terms/softmax-layer>.
- [30] Ben Dickson. *The limits and challenges of deep learning*. Feb. 2018. URL: <https://bdtechtalks.com/2018/02/27/limits-challenges-deep-learning-gary-marcus/>.
- [31] Anas Al-Masri. *What Are Overfitting and Underfitting in Machine Learning?* June 2019. URL: <https://towardsdatascience.com/what-are-overfitting-and-underfitting-in-machine-learning-a96b30864690>.
- [32] Markus Ring, Daniel Schlör, Dieter Landes, et al. “Flow-based Network Traffic Generation using Generative Adversarial Networks”. In: (2018). DOI: 10.1016/j.cose.2018.12.012. eprint: [arXiv:1810.07795](https://arxiv.org/abs/1810.07795).
- [33] Ramiro Camino, Christian Hammerschmidt, and Radu State. “Generating Multi-Categorical Samples with Generative Adversarial Networks”. In: (2018). eprint: [arXiv:1807.01202](https://arxiv.org/abs/1807.01202).
- [34] Konstantinos Demertzis, Lazaros Iliadis, and Stefanos Spartalis. “A spiking one-class anomaly detection framework for cyber-security on industrial control systems”. In: *International Conference on Engineering Applications of Neural Networks*. Springer. 2017, pp. 122–134.

- [35] Philip Kosky, Robert Balmer, William Keat, et al. “Chapter 11 - Industrial Engineering”. In: *Exploring Engineering (Fifth Edition)*. Ed. by Philip Kosky, Robert Balmer, William Keat, et al. Fifth Edition. Academic Press, 2021, pp. 229–257. ISBN: 978-0-12-815073-3. DOI: <https://doi.org/10.1016/B978-0-12-815073-3.00011-9>. URL: <https://www.sciencedirect.com/science/article/pii/B9780128150733000119>.
- [36] *What is a Correlation Matrix?* URL: <https://corporatefinanceinstitute.com/resources/excel/study/correlation-matrix/> (visited on 04/13/2021).
- [37] *Queuing theory: Definition, history & real-life applications*. URL: <https://queue-it.com/blog/queuing-theory/> (visited on 04/13/2021).
- [38] Wireshark Foundation. *Wireshark*. Version 16.48. Apr. 19, 2020. URL: <https://www.wireshark.org/>.
- [39] *M/M/1 QUEUING MODEL*. URL: <https://www.oreilly.com/library/view/quantitative-techniques-theory/9789332512085/xhtml/ch9sec14.xhtml> (visited on 04/13/2021).
- [40] Saher Manaseer, Oroba Al-Nahar, and Abdallah Hyassat. “Network Traffic Modeling, Case Study: The University of Jordan”. In: *International Journal of Recent Technology and Engineering* 7 (Jan. 2019).
- [41] *A Refresher on Regression Analysis*. URL: <https://hbr.org/2015/11/a-refresher-on-regression-analysis> (visited on 04/13/2021).
- [42] *Correlation Coefficient: Simple Definition, Formula, Easy Steps*. URL: <https://www.statisticshowto.com/probability-and-statistics/correlation-coefficient-formula/> (visited on 04/13/2021).
- [43] *Coefficient of Determination (R Squared): Definition, Calculation*. URL: <https://www.statisticshowto.com/probability-and-statistics/coefficient-of-determination-r-squared/> (visited on 04/13/2021).
- [44] *P-Value*. URL: <https://www.investopedia.com/terms/p/p-value.asp> (visited on 04/13/2021).
- [45] *What is a Correlation Matrix?* URL: <https://www.statisticshowto.com/correlation-matrix/> (visited on 04/13/2021).
- [46] 2021. URL: <https://faker.readthedocs.io/en/master/>.
- [47] *Loss Functions — Generative Adversarial Networks*. URL: <https://developers.google.com/machine-learning/gan/loss> (visited on 04/08/2021).
- [48] *List of TCP and UDP port numbers*. URL: https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers (visited on 04/04/2021).
- [49] 2013. URL: https://scikit-learn.org/stable/modules/cross_validation.html.
- [50] *What is a Network Traffic Flow?* URL: <https://mattjhayes.com/2018/09/26/what-is-a-network-traffic-flow/> (visited on 04/13/2021).
- [51] Microsoft. *Excel*. Version 3.5.0. Apr. 19, 2020. URL: <https://www.microsoft.com/en-us/microsoft-365/excel>.
- [52] *Queuing*. URL: <https://www.linktionary.com/q/queuing.html> (visited on 04/13/2021).

- [53] *Docker vs. Virtual Machines: Differences You Should Know*. URL: <https://cloudacademy.com/blog/docker-vs-virtual-machines-differences-you-should-know/> (visited on 10/29/2019).
- [54] *Jupyter Notebook: An Introduction*. URL: <https://realpython.com/jupyter-notebook-introduction/>.
- [55] *JupyterLab Overview*. URL: https://jupyterlab.readthedocs.io/en/stable/getting_started/overview.html.

A Software Delivery

Docker holds all of its containers inside of images, which can be saved into tar files. However, our team needed to extract the container since that is what holds all of the packages necessary in order to run the user interface. Docker doesn't directly provide functionality for creating a runnable container without converting it into an image first, which at that point can be turned into a .tar file and shared. When the .tar file is loaded, it will create a Docker image on Arco IT GmbH's machine, based off of the container that was originally created. Since saving a Docker image doesn't include the volume that has all of the necessary files, that volume needed to be shared as well. Our team needed to convert the desired volume into a zipped folder that can be shared with relatively little size.

Arco IT GmbH uses teams primarily for their file sharing, so all components were uploaded to their tenant for execution. The contents of the upload are as follows: the .tar file in the form of a Docker image that holds all dependencies, a .zip file that will act as the volume holding python scripts and Jupyter Notebooks, a Microsoft Word document with instructions for configuring, and a Microsoft Word document for using the user interface. The official documentation for downloading Docker, loading the Docker image, and running it with the correct volume is shown below in Figure 19.

Arco IT GmbH User Interface Configuration

1. Download the Arco_User_Interface_Environment.tar and ArcoMQP2021.zip files on Arco's Team Tenant
 - a. Unzip ArcoMQP2021.zip once it has been downloaded
2. Download Docker at: <https://www.docker.com/products/docker-desktop>
3. Once Docker is installed, navigate to CMD for Windows PC's or Terminal for Mac Machines
 - We will run the following commands in sequential order:
 - docker load -i <Arco_User_Interface_Environment.tar path name>
 - docker run --rm -p 8889:8888 -e JUPYTER_ENABLE_LAB=yes -v <arcomqp2021.zip path name>:/home/jovyan/work arco.user.interface.environment
4. The final command will generate two URL's which will be needed for the next step
5. Navigate to the browser of your choice and enter the following in the navigation pane:
 - a. Localhost:8889
 - b. At this point, you will be prompted with a password text box
 - c. Navigate back to CMD or Terminal and locate the portion of text immediately after "token="
 - d. Copy this text from start to finish and paste it into the password portion of the browser.
 - e. Strike "login"

Figure 19: Demonstrates the directions to successfully run the user interface using Docker.

Further breaking down the first Docker statement under step three: The "load" command creates the image so that it can be run in the next step. The second Docker statement under step three consists of: The "run" command which runs the image, "rm" says to remove the container every time we exit, "p" indicates the port that we would like to utilize, "-e" is used to run Jupyter Lab, "-v" specifies the volume that we would like to mount to the container, which in our case was our GitHub configured repository, the ":" that follows is the path inside of the container that we would like to mount that volume to, and the final portion of the statement specifies the name of the created image.

B Sample Network Flow Data

Address A	Port A	Address B	Port B	Packets	Bytes	Rel Start	Duration
10.11.1.31	60892	10.12.0.74	445	416	264930	479.924519	25.213583
10.11.1.31	60893	10.12.0.72	445	447	149971	479.929053	13.189887
10.11.1.31	60894	10.12.0.73	445	462	189861	479.937726	13.209826
10.11.1.31	60895	10.12.0.74	445	1	66	481.014597	0.0
10.11.1.31	60896	10.12.0.74	139	1	66	481.015527	0.0
10.11.1.79	2804	10.12.1.24	502	10	614	54.211439	0.042355
10.11.1.79	2805	10.12.1.24	502	10	613	54.235811	0.068309
10.11.1.79	2806	10.12.1.24	502	10	614	54.327464	0.039092
10.11.1.79	2807	10.12.1.24	502	10	613	54.35474	0.058421
10.11.1.79	2808	10.12.1.24	502	10	614	54.41895	0.047702
10.11.1.79	2809	10.12.1.24	502	10	614	59.227064	0.045256
10.11.1.79	2810	10.12.1.24	502	10	613	59.254572	0.087284
...
192.168.100.2	60375	91.190.218.52	80	3	194	879.360044	8.99951
192.168.100.2	60377	65.55.223.17	443	3	194	886.119864	9.00066
192.168.100.2	60378	65.55.223.17	80	3	194	887.730104	9.000373
192.168.100.2	60381	157.56.52.41	443	2	132	898.261471	2.970304
192.168.100.2	60383	157.55.130.155	443	2	132	899.291877	2.970637
192.168.100.2	60387	157.56.52.41	80	2	132	899.851871	3.000662
192.168.100.2	60388	111.221.77.162	443	1	66	900.302073	0.0
192.168.100.2	60389	157.55.130.155	80	1	66	900.881977	0.0
192.168.100.2	60390	157.55.235.153	443	1	66	901.342021	0.0
192.168.100.2	60391	157.55.56.161	443	1	66	901.342023	0.0
192.168.100.2	60392	157.55.130.146	443	1	66	901.342025	0.0
192.168.100.2	60393	111.221.77.162	80	1	66	901.913229	0.0

C IP Inventory

LAN IP Address	Plant Area	Network Type	Device
10.11.9.2	Zinc Plant	iFIX Nodes \$ Ethernet Switches	PPPCZP6
10.11.9.14	Refinery	iFIX Nodes \$ Ethernet Switches	PPPCR4
10.11.9.15	Refinery	PLCs \$ Control Equipment	RFMBPE1
10.11.9.16	Refinery	iFIX Nodes \$ Ethernet Switches	PPPCR4
10.11.9.28	Miscellaneous	iFIX Nodes \$ Ethernet Switches	EGX100
10.11.9.35	Slag Fumer	iFIX Nodes \$ Ethernet Switches	PPPCSF4
10.11.9.36	Slag Fumer	iFIX Nodes \$ Ethernet Switches	PPPCSF12
10.11.9.37	Slag Fumer	iFIX Nodes \$ Ethernet Switches	PPPCSF13
10.11.9.41	Refinery	iFIX Nodes \$ Ethernet Switches	PP_KBA3
10.11.9.43	Zinc Plant	PLCs \$ Control Equipment	ZPMBPE1
10.11.9.45	Miscellaneous	iFIX Nodes \$ Ethernet Switches	nan
10.11.9.47	Miscellaneous	iFIX Nodes \$ Ethernet Switches	PP_SMS
...
10.12.128.180	Sinter Plant	iFIX Nodes \$ Ethernet Switches	SPSW1A/B
10.12.128.254	Other ProcN Devices	iFIX Nodes \$ Ethernet Switches	PPPCRT1
10.12.129.21	Blast Furnace	PLCs \$ Control Equipment	BFPLC1
10.12.129.22	Blast Furnace	PLCs \$ Control Equipment	BFPLC2
10.12.129.41	Acid Plant	PLCs \$ Control Equipment	APPLC1
10.12.129.42	Acid Plant	PLCs \$ Control Equipment	APPLC2
10.12.129.161	Sinter Plant	PLCs \$ Control Equipment	SPPLC1
10.12.129.162	Sinter Plant	PLCs \$ Control Equipment	SPPLC2
10.13.1.182	Refinery	RIO Network	PMRCRP1
10.13.1.183	Refinery	RIO Network	PMRCRA1
10.13.1.221	EAF Dust Washing	RIO Network	EAFCRP1
10.13.1.222	EAF Dust Washing	RIO Network	EAFORA1

D Model Output

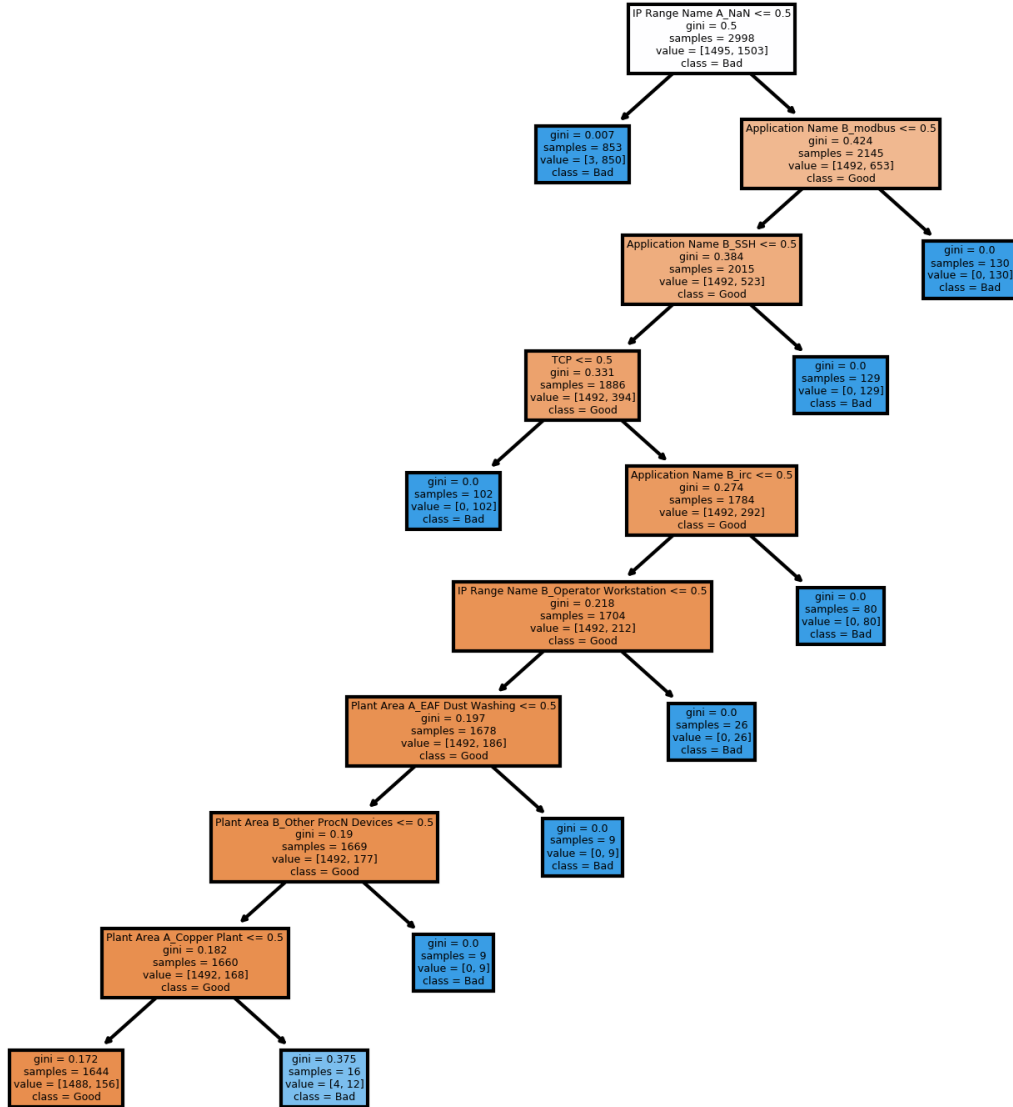


Figure 20: This is a visualization of one of the trees generated with a max leaf node count of 10, this was done since a tree with a higher count while will most likely have a higher accuracy and better rules it is too difficult to visualize. This tree was trained on a 50/50 split of real traffic and artificially created traffic (90% GAN generated, 10% slightly modified real traffic). Each node is labelled with the decision made, the Gini impurity value, the number of samples, the value or the number of samples of each class, and the class the model classified the node as (only important for leaf nodes). Since all the features being used are one-hot encoded, and the decision tree is selecting a threshold, all of our thresholds will be 0.5 since the value could either be a 0 or a 1. For example, the root node is labelled “IP Range Name A_NaN ≤ 0.5 ” which can translate to if the IP range name is NaN (or is undefined) go right, else go left. Since there is a lot of user specified information like tags for sets of IP addresses and tags for different port numbers, this data is using a small set of simulated values so the model can have a difficult time splitting the “good” and “bad” traffic apart. The accuracy at determining whether traffic is normal or non-normal is 93.73%.

The decision tree from Figure 20 will be converted into these firewall rules:

```
FROM ((tag Plant Area = Zinc Plant OR tag Plant Area = Refinery OR tag Plant Area = Miscellaneous OR tag Plant Area = Slag Fumer OR tag Plant Area = Kilns/KDR OR tag Plant Area = Power Plant OR tag Plant Area = Other ProcN Devices OR tag Plant Area = Acid Plant OR tag Plant Area = Fume Control OR tag Plant Area = Blast Furnace OR tag Plant Area = PETs Plant OR tag Plant Area = Sinter Plant OR tag Plant Area = Spare Plant Area) AND tag IP Range Name = NaN) TO ((tag Plant Area = Zinc Plant OR tag Plant Area = Refinery OR tag Plant Area = Miscellaneous OR tag Plant Area = Slag Fumer OR tag Plant Area = Kilns/KDR OR tag Plant Area = Copper Plant OR tag Plant Area = Power Plant OR tag Plant Area = Acid Plant OR tag Plant Area = Fume Control OR tag Plant Area = Blast Furnace OR tag Plant Area = PETs Plant OR tag Plant Area = Sinter Plant OR tag Plant Area = EAF Dust Washing OR tag Plant Area = Spare Plant Area) AND tag IP Range Name = Ethernet Bridge) ALLOW tcp (PORT 80 AND PORT 443)
```

Notice that only one rule is generated, this is because only one leaf node in the tree resolved to the first class, normal traffic. Since the above example is a smaller decision tree, see below for a different example from a decision tree with a max leaf node count of 50 instead of 10 that resulted in an accuracy of 97.2%:

```

FROM (tag Plant Area = Fume Control AND tag IP Range Name = Operator Work-
station) TO tag Plant Area = NaN ALLOW tcp PORT all
FROM ((tag Plant Area = Slag Fumer OR tag Plant Area = Kilns/KDR OR tag Plant
Area = Other ProcN Devices OR tag Plant Area = Fume Control OR tag Plant Area
= PETs Plant) AND tag IP Range Name = NaN) TO (tag Plant Area = Miscellaneous
OR tag Plant Area = PETs Plant) ALLOW tcp PORT all
FROM (tag Plant Area = NaN AND tag IP Range Name = NaN) TO tag Plant Area =
Blast Furnace ALLOW tcp PORT all
FROM (tag Plant Area = Refinery AND tag IP Range Name = NaN) TO tag Plant
Area = NaN ALLOW tcp PORT all
FROM (tag Plant Area = Blast Furnace AND tag IP Range Name = NaN) TO (tag
Plant Area = Miscellaneous OR tag Plant Area = Blast Furnace OR tag Plant Area =
PETs Plant) ALLOW tcp PORT all
FROM (tag Plant Area = Other ProcN Devices AND tag IP Range Name = NaN) TO
tag Plant Area = Refinery ALLOW tcp PORT all
FROM (tag Plant Area = Other ProcN Devices AND tag IP Range Name = NaN) TO
tag Plant Area = Fume Control ALLOW tcp PORT all
FROM (tag Plant Area = NaN AND tag IP Range Name = NaN) TO tag Plant Area =
Copper Plant ALLOW tcp PORT all
FROM (tag Plant Area = Other ProcN Devices AND tag IP Range Name = NaN) TO
tag Plant Area = Slag Fumer ALLOW tcp PORT all
FROM (tag Plant Area = Other ProcN Devices AND tag IP Range Name = NaN) TO
tag Plant Area = Acid Plant ALLOW tcp PORT all
FROM ((tag Plant Area = Refinery OR tag Plant Area = Slag Fumer OR tag Plant
Area = Kilns/KDR OR tag Plant Area = Power Plant OR tag Plant Area = Other
ProcN Devices OR tag Plant Area = Blast Furnace) AND tag IP Range Name = NaN)
TO tag Plant Area = EAF Dust Washing ALLOW tcp PORT all
FROM (tag Plant Area = Sinter Plant AND tag IP Range Name = NaN) TO tag Plant
Area = NaN ALLOW tcp PORT all
FROM (tag Plant Area = Power Plant AND tag IP Range Name = NaN) TO tag Plant
Area = Power Plant ALLOW tcp PORT all
FROM (tag Plant Area = Acid Plant AND tag IP Range Name = NaN) TO ((tag Plant
Area = Zinc Plant OR tag Plant Area = Refinery OR tag Plant Area = Miscellaneous
OR tag Plant Area = Slag Fumer OR tag Plant Area = Kilns/KDR OR tag Plant Area
= Copper Plant OR tag Plant Area = Power Plant OR tag Plant Area = Acid Plant
OR tag Plant Area = Fume Control OR tag Plant Area = Blast Furnace OR tag Plant
Area = PETs Plant OR tag Plant Area = EAF Dust Washing) AND tag IP Range
Name = Ethernet Bridge) ALLOW tcp PORT all
FROM (tag Plant Area = Other ProcN Devices AND tag IP Range Name = NaN) TO
(tag Plant Area = Sinter Plant AND tag IP Range Name = Ethernet Bridge) ALLOW
tcp PORT all
FROM (tag Plant Area = Copper Plant AND tag IP Range Name = NaN) TO (tag
Plant Area = NaN AND tag IP Range Name = Ethernet Bridge) ALLOW tcp (PORT
80 AND PORT 443)

```

One short coming from using decision trees, is it can create very long rules. The reason behind this, is the binary nature. Since at each node, it will split the samples into feature X or not feature X, because the firewall syntax we are using does not have a not-equals operator, we have to compare it to all the other tags. If the tag being used has many options, the length of a rule can grow extremely long.