View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Eastern Washington University: EWU Digital Commons

Eastern Washington University

EWU Digital Commons

EWU Masters Thesis Collection

Student Research and Creative Works

Winter 2021

Intrusion detection for industrial control systems

Kurt Lamon

Follow this and additional works at: https://dc.ewu.edu/theses

 Part of the Digital Communications and Networking Commons, and the Other Computer Engineering Commons

Intrusion Detection for Industrial Control Systems

A Thesis

Presented To

Eastern Washington University

Cheney WA

In Partial Fulfillment of the Requirements

for the Degree

Master of Science in Computer Science

---

by

Kurt Lamon

Winter 2021

Authorization to Submit Thesis

This Thesis of **Kurt Lamon**, submitted for the degree of Master of Science with a major in Computer Science and entitled **"Intrusion Detection for Industrial Control Systems,"** has been reviewed in final form. Permission, as indicated by the signatures and dates given below, is now granted to submit final copies to the College of Graduate Studies for approval.

Major Professor: _____ Date: _____
Dr. Stuart Steiner

Committee Member: _____ Date: _____
Dr. Dan Li

Committee Member: _____ Date: _____
Dr. Martin Weiser

Abstract

Industrial Control Systems (ICS) are rapidly shifting from closed local networks, to remotely accessible networks. This shift has created a need for strong cybersecurity anomaly and intrusion detection for these systems; however, due to the complexity and diversity of ICSs, well defined and reliable anomaly and intrusion detection systems are still being developed.

Machine learning approaches for anomaly and intrusion detection on the network level may provide general protection that can be applied to any ICS. This paper explores two machine learning applications for classifying the attack label of the UNSW-NB15 dataset. The UNSW-NB15 is a benchmark dataset that was created off general network communications and includes labels for normal behavior and attack vectors. A baseline was created using K-Nearest Neighbors (kNN) due to its mathematical simplicity.

Once the baseline was created a feed forward artificial neural network known as a Multi-Layer Perceptron (MLP), was implemented for comparison due to its ease of reuse for running in a production environment. The experimental results show that both kNN and MLPs are effective approaches for identifying malicious network traffic; although, both still need to be further refined and improved before implementation on a real-world production scale.

Acknowledgments

I would like to thank my advisor Dr. Stuart Steiner for his continued guidance, support, and constant stream of answers to my constant stream of questions as I navigated research, thesis ideas, and my degree in general.

In addition, I would like to thank my committee members, Dr. Dan Li and Dr. Martin Weiser for their time, expertise, and feedback throughout the process of completing this thesis.

Finally, I would like to thank the professors of both Eastern Washington University and Gonzaga University, my undergraduate alma mater, for their work in providing classes and instruction that would fascinate and inspire me to learn all I can about computer science.

Table of Contents

List of Figures

List of Tables

List of Equations

List of Acronyms

AIDS Anomaly-based Intrusion Detection System

AUC Area Under Curve

CSV Comma Separated Value

DoS Denial of Service

EDA Exploratory Data Analysis

FAR False Alarm Rate

FN False Negative

FNR False Negative Rate

FP False Positive

FPR False Positive Rate

FTP File Transfer Protocol

HTTP HyperText Transfer Protocol

ICS Industrial Control System

IDS Intrusion Detection System

IDPS Intrusion Detection and Prevention System

IoT Internet of Things

kNN k-Nearest Neighbors

MLP Multi-Layer Perceptron

NIDS Network Intrusion Detection System

RELU Rectified Linear Unit

RTU Remote Terminal Unit

ROC Receiver Operating Characteristic

SCADA Supervisory Control And Data Acquisition

SIDS Signature-based Intrusion Detection System

SGD Stochastic Gradient Descent

TCP/IP Transport Communication Protocol/Internet Protocol

TN True Negative

TNR True Negative Rate

TP True Positive

TPR True Positive Rate

Glossary Of Terms

Anomaly Detection - A branch of machine learning where models search for outliers within a dataset. Intrusion detection is a practical application of anomaly detection for cybersecurity.

Anomaly Based Intrusion Detection - A form of intrusion detection that seeks to classify intrusions by determining which communications are out of the ordinary from the normal behavior.

Industrial Control Systems - A collective term for the variety of electronic systems, networks, and hardware that monitor and control industrial processes.

k-Nearest Neighbors - A simple supervised machine learning model that can be used for both classification and regression.

Malware - Software that is designed to damage, intrude on, or compromise a system.

Multi-Layer Perceptron - A type of feedforward artificial neural network.

Precision - A performance metric that is the rate of correct positive predictions out of all positive predictions.

Recall - A performance metric that is the proportion of negatives correctly predicted.

ROC AUC - The Receiver Operating Characteristic Area Under Curve is a performance measure that combines the true positive and false positive rates.

Signature Based Intrusion Detection - A form of intrusion detection that seeks to classify intrusions by exploring their similarity to a list of known intrusions. This method has lower overhead to test data but cannot identify new attacks.

Supervisory Control and Data Acquisition Systems - A class of industrial control systems that focus on receiving data from sensors, reporting that data, and making system updates.

UNSW-NB15 - A benchmark dataset designed for testing intrusion detection systems.

Chapter 1 Introduction

Industrial Control Systems (ICS) and Supervisory Control And Data Acquisition (SCADA) systems are critical components of modern industrial infrastructure. In the past, these components of critical infrastructure would run on closed, local networks or no network at all, requiring a technician to be present on site to make changes or supervise the hardware. By connecting these systems to a remotely accessible network, technicians enabled themselves to control more complicated systems remotely as well as increasing and improving capabilities of the infrastructure components, however doing so also enables the possibility of intrusion by unauthorized parties. Bhamere et al. describe the security risks associated with opening these systems to remote terminals [1].

Due to the diversity of industrial control systems and the infrastructure that they manage, no unified rule-based security approach could be made. Stouffer et al. describes how due the critical nature of industrial infrastructure, any sort of intrusion can result in catastrophic consequences [2].

The diversity of malware and its constant evolution makes rule-based intrusion detection ineffective at detecting novel attacks and requiring of regular updates. Buczak and Guven describe this growth and the corresponding need for adaptable solutions [3]. Machine learning approaches train an effective model to recognize complex patterns in existing data. This enables the testing of new data to a high degree of accuracy, and can be updated with new labeled data to improve performance in an ever changing environment. According to Liu and Lang, machine learning algorithms for anomaly detection tend to have a lower missed rate but a higher false alarm rate when compared to signature or misuse based detection methods [4].

Intrusion detection is a critical component of network security. It is not yet possible to build a system that is perfectly closed to any outside agents while being open to correct ones, therefore it is necessary to implement solutions that recognize when attacks are being launched and notify the system of potential intrusions [2]. Intrusion detection is directly related to the machine learning task of anomaly detection, identifying which behaviors are out of the ordinary for a normal set of values.

1.1 Literature Review

As cybersecurity, industrial control systems, and machine learning all make continuous advancements, each of these topics as well as their interdependence has been a widely explored topic. Previous works researched applications and techniques for intrusion detection as well as the challenges and options available for SCADA specific benchmark datasets.

Network intrusions can be determined by a variety of methods. Mudzingwa and Agrawal group these IDS methods into three major approaches, Anomaly based Intrusion Detection Systems (AIDS), Signature based Intrusion Detection Systems (SIDS), and hybrid approaches which combine aspects of the other two [5]. SIDS work by comparing observed communication signatures to a local database or file of known malicious signatures. This approach has exceptionally low overhead for both testing and implementing the system since no training or learning of the environment is required. The drawback of SIDS is its inability to recognize new attacks or ones that are not included in file. AIDS mitigates the weaknesses of SIDS by using machine learning or statistical-based models to identify abnormal communications in the network traffic. The benefit to AIDS is the ability to catch new attacks; however these systems do so at the expense of training and learning time.

AIDS can also be further broken down into subcategories: Statistics based models, knowledge based models, and machine learning models. Statistics based models define a profile for normal behavior and flag any points that are then determined to be low probability. Knowledge based models similarly create a profile for normal behavior and then flag any low probability occurrences; what separates these models from the statistical category is knowledge based models are created using human knowledge about the system rather than automated data collected from the system. Machine learning based models work by extracting knowledge from large amounts of data and define complex transfer functions that flag individual points as anomalies or intrusions. Khraisat et al. describe the approaches and respective challenges to creating successful intrusion detection systems [6].

There are many different approaches within the realm of machine learning models for anomaly detection. Khraisat et al. outline some of the more popular algorithms and their proprietary effectiveness, including Decision Trees, Naive Bayes, Genetic Algorithms, Artificial

Neural Networks, Fuzzy logic, Support Vector Machines, Hidden Markov Models, k-Nearest Neighbors (kNN), and k-Means [6]. With a fairly basic Remote Terminal Unit (RTU) serial communication dataset, the approaches of Naive Bayes, Random Forests, Non-Nested Generalized Exemplars, Support Vector Machines, and some Decision Tree/rule-based approaches were shown to be promising at detecting malicious communications [7].

Generalizing the dataset to include broader communications provides broader insights and any models created could then potentially be utilized in multiple applications. Benchmark datasets for network intrusion detection were created to analyze this problem. The UNSW-NB15 dataset, created in 2015, improves upon the KDD-99 dataset which at that point was becoming dated. Moustafa and Slay analyzed and described the UNSW-NB15 dataset in detail, doing several statistical explorations to prove the quality of the dataset. The UNSW-NB15 dataset was validated to accurately mimic both normal and attack behavior and is shown to be complex enough that any existing or novel Network Intrusion Detection System (NIDS) could be reliably tested using the dataset [8].

Chapter 2 Dataset Selection

The UNSW-NB15 dataset was selected because it is one of several key Intrusion Detection benchmark datasets. It is designed to represent normal and attack behaviors in modern network traffic. Each data point represents one network packet or communication sent or received. The dataset contains both normal behavior and nine different modern attack types [8].

2.1 Intrusion Detection Benchmark Datasets

The world of network connected computing systems is one experiencing extremely high growth, change, and diversification. With so many devices, from every day IoT devices to large infrastructure systems, becoming increasingly dependent on network connectivity, the threat of malware in these systems is also increasing, thus creating an express need for comprehensive network cyber security. Hamid et al. describe that due to the wide diversity and growth rate of these systems, it is ineffective to try to create measures of cyber defense on a per-system basis [9].

Intrusion Detection benchmark datasets help mitigate the inefficiencies of per-system cyber defense measures. Ring et al. survey the available IDS benchmark datasets to validate their design and describe their intentions. These benchmark systems are specifically designed to be more global in application and research, with the intent of developing solutions that can be applied to any facet of network connected applications [10].

Many benchmark datasets have been created since the first notable dataset, DARPA, was developed in 1998 [10]. Table 2.1 shows the progression of benchmark datasets created for intrusion detection experiments.

Generally, each dataset incrementally improved upon the previous. The early datasets were criticized for being generated with artificially created network traffic, both for attacks and normal behavior, thereby lacking representative data of the problem space. As new datasets were created to iterate, replace, or strengthen the previous, each had its own unique challenges in data generation or sourcing. As network usage matures and evolves these datasets lose their relevancy because the data they hold becomes less similar to modern architectures and attacks.

Dataset	Year Created	Created By
DARPA	1998-1999	Lincoln Laboratory
KDD99	1998-1999	UC Irvine
DEFCON	2000	The Shmoo Group
CAIDA	2002/2016	Center of Applied Internet Data Analysis
LBNL	2004/2005	Lawrence Berkeley National Laboratory
NSL-KDD	2009	University of New Brunswick
CDX	2009	United States Military Academy
Kyoto	2009	Kyoto University
Twente	2009	University of Twente
UMASS	2011	University of Massachusettes
ISCX	2012	University of New Brusnwick
ADFA	2013	University of New South Wales
UNSW-NB15	2015	Australian Centre for Cyber Security

Table 2.1: Development of intrusion detection benchmark datasets.

2.2 The UNSW-NB15 Dataset

The UNSW-NB15 dataset was created by researchers in the Cyber Range Lab of the Australian Centre for Cyber Security in an effort to improve upon and update existing network cybersecurity benchmark datasets. Previous benchmarks, the KDD99 and NSLKDD datasets, the former explored by Özgür and Erdam and the latter explored by Dhanabal and Shantharaj, were outdated because attacks have become more advanced and low profile and normal network behavior has also changed [11]. In the analysis of Moustafa and Slay, the UNSW-NB15 dataset is shown to have updated both the attacks and normal behavior to represent more modern network traffic [8].

The UNSW-NB15 dataset was created by simulating normal network traffic and attacks using the IXIA PerfectStorm and tcpdump tools to capture network data. Then the Argus and Zeek-IDS tools were used to extract feature information into CSV-formatted datasets. The UNSW-NB15 dataset consists of approximately 2.5 million rows and is split into four CSV files [8].

While this dataset is seemingly very comprehensive, it is also succeptible to becoming dated due to the growth and diversification of both normal behavior and malware. Although this dataset is still relevant, it will eventually need to be updated and replaced as the needs of network communications and the ways hackers exploit them change.

Attributes and Structure

The UNSW-NB15 dataset has 49 features that each describe some attribute of a communication or its context within an arriving set of communications. Most features are continuous but some are categorical and two are binary. The dataset also contains label and metadata for the type of attack, attack vs normal, and dataset indexing. Excluding the labels and metadata, Moustafa and Slay divide the informational features into five categories [8]:

1. **Flow Features:** Features that identify the hosts/recipients of a communication and the communication protocol.

2. **Basic Features:** Features that contain meta-information about the particular instance of connection and communication, such as the duration or size of the communication, time-to-live, etc.

3. **Content Features:** Features describing attributes of TCP/IP and HTTP services.

4. **Time Features:** Features describing the timing information of the particular communication.

5. **Additional Generated Features:** General purpose generated flags and statistical information about the content and delivery of the communication, or statistical information about the particular communications relationship to surrounding communications.

See Appendix A.1 for information on individual features within each of these groups.

Attacks and Intrusions

Where the previous benchmark datasets lacked present day attack vectors, the UNSW-NB15 dataset supplements this with attacks classified into nine groups:

1. **Fuzzers:** Attacks characterized by feeding a system large amounts of data in attempt to make it crash.

2. **Analysis:** Intrusions that target web applications through ports, email, or scripts in order to gain entrance.

3. **Backdoor:** Connections that attempt to circumvent normal security measures of a webapp, attempting to gain access for an external user.

4. **Denial of Service (DoS):** An attack designed to occupy the resources of a system, causing it to be too busy to respond to legitimate users.

5. **Exploit:** An attack that takes advantage of a particular unintended vulnerability of a system.

6. **Generic:** An attack that seeks to identify collision with every block cipher of a system.

7. **Reconnaissance:** A probe designed to gain information about a system without necessarily intruding or causing damage.

8. **Shellcode:** An attack where shell scripts are used to gain entrance to the code of a web application.

9. **Worms:** An attack where the malware seeks to replicate itself across a network once it has gained entrance.

The dataset is approximately 87% non-malicious values and 13% attack values. Table 2.2 illustrates the distribution of attack categories in the dataset.

Attack Category	Label	Count	Percentage
normal	0	2,218,764	87.3513%
generic	1	215,481	8.4833%
exploits	1	44,525	1.7529%
fuzzers	1	24,246	0.9545%
dos	1	16,353	0.6438%
reconnaissance	1	13,987	0.5507%
analysis	1	2,677	0.1054%
backdoor	1	2,329	0.0917%
shellcode	1	1,511	0.0595%
worms	1	174	0.0069%

Table 2.2: UNSW-NB15 attack type distribution

2.3 Comparison: UNSW-NB15 and NSL-KDD

The DARPA dataset was one of the original datasets created for IDS testing. It was designed by running network traffic in a simulated air force base for two weeks, followed by five weeks of attack traffic. The KDD99 set was then created by processing the DARPA logs. Because of the preprocessed format of the KDD99 dataset, it became more popular for machine learning experiments. Later, the NSL-KDD dataset was created by removing redundancies and duplicates from KDD99. The NSL-KDD dataset effectively reduced the size of KDD99 without taking away from the knowledge that it represents, as described by Özgür and Erdam [12].

The NSL-KDD dataset, while an improvement on KDD99, retains all of the shortcomings of KDD99 as well. The attacks featured in KDD99 and NSL-KDD are limited and lack many of the attacks that modern networks face. Each of the attacks in the NSL-KDD and KDD99 datasets can be divided into four categories:

1. Denial of Service (DoS) - attempting to overwhelm a system in order to make it unavailable to legitimate users.
2. R2L - Unauthorized access or password guessing, remote to local.
3. U2R - Unauthorized access to a root account, user to root.
4. Probing - Observation of a system in order to gain information about that system.

The UNSW-NB15 dataset contains similar versions of each of these attack categories, as well as more modern ones. The UNSW-NB15 dataset also contains more up to date normal behavior as well as a more appropriate balance of normal and attack behaviors. Where NSL-KDD is imbalanced towards attacks, the UNSW-NB15 emphasizes on normal behaviors.

2.4 Comparison: UNSW-NB15 and ICS Attack Datasets

T. Morris provides a collection and survey of several ICS specific datasets. These datasets are created from simulated critical infrastructure systems, including a gas pipeline, a water storage tank, and a power supply system [13]. Each of these datasets includes both generic network

data as well as data specific to each piece of infrastructure, including setpoints for pipeline pressure and PID controller values.

These datasets could be very useful in the future when determining potential models for specific systems, however, as current research is exploring more general solutions, the UNSW-NB15 dataset would likely apply more broadly to systems outside these ones listed.

Chapter 3 Architectural Overview

This thesis expands on the existing work related to creating and evaluating intrusion detection systems for the UNSW-NB15 dataset. In the experiments of S. Maji, logistic regression, gradient boosted decision trees, ensemble classifiers, and other machine learning models were used. The ensemble method performed best as it was able to combine the strengths of multiple types of classifiers [14].

This thesis is structured in the following general machine learning pipeline:

1. **Data Preprocessing:** Null values, alternate spellings, and out of range values must be adjusted to expected ranges and data types before applying any machine learning model.

2. **Exploratory Data Analysis:** General data exploration yields insights into the dataset providing assistance for feature engineering. This can include correlation analysis and exploring dataset metrics.

3. **Feature Engineering:** Certain features can be dropped, scaled, transformed, or created to add new insights to the dataset or to remove unnecessary complexity before applying machine learning models.

4. **Baseline Model Implementation:** An easy-to-implement model was used to create a baseline for expectations of the more advanced model. This application of k-Nearest Neighbors (kNN) is used as the baseline model.

5. **Advanced Model Implementation:** A more advanced model is used to explore the dataset. For this application, a Multi-Layer Perceptron neural network was used as the more advanced model.

6. **Performance Metrics and Analysis:** Once predictions are made for the baseline and advanced models' test sets, the effectiveness of these models can be explored. Due to dataset balancing constraints, both F1 score and accuracy were analyzed.

One key component of exploration is assessing the success of the advanced model. According to Zuech et al. a successful application will be highly accurate, minimize false alarms, and be

able to recognize many different attacks [15].

This thesis develops a k-Nearest Neighbors classifier for identifying attacks to use as the baseline comparison for a subsequent MLP approach.

This application is theoretical in nature and should be viewed in the context of a preliminary step in identifying the most effective system that could be implemented in a production network environment. This work focused on quantitative analysis of the results, F1 score, measures of accuracy, etc; Stouffer et al. describe that it is important to consider the qualitative measures associated with each potential model [2]. Qualitative measures are minimally explored but it is useful to consider the training time, testing time, and ease of updates to the model as measures of what might make an applicable industry implementation. Additionally, the repercussions of a successful cyber attack on an industrial control system can be immense, so any approach that is short of perfect must still be improved or combined with additional efforts before being deemed effective enough to be put into production.

3.1 Data Preprocessing

While the UNSW-NB15 dataset is of very high quality and is relatively comprehensive, there are still a few steps required to prepare data and clean it before it is ready for model application. The data is stored in four separate CSV files that must be combined into a unified data structure. In this application a Pandas dataframe in Python was used, though any data structure can be used.

Only a few columns required preprocessing for null values or mismatched data types. The columns "ct_ftp_cmd" and "is_ftp_login", containing the count of ftp commands in the sessions and a boolean flag if the ftp session was accessed by user and password, needed to be recast from objects to integers. The aforementioned columns as well as the columns "attack_cat" and "ct_flw_http_mthd", containing the attack type label and the count of HTTP flows in the session, needed to have null values filled. Because each of these columns had meaningful null values, indicating that the specific column did not apply to that particular data point, the nulls were replaced with a meaningful categorical point indicating the same effect. If there were any continuous variables with null values, those should be replaced with the mean of the training

data for that feature. Two columns, "attack_cat" and "ct_ftp_cmd", also included categorical data points that were either misspelled or out of range. These were fixed with replacement lambda functions.

3.2 Exploratory Data Analysis

The UNSW-NB15 dataset contains over 2.5 million rows and 49 features. In effort to save a significant amount of computing requirements, it is crucial to identify redundancies in data, namely any pieces of data that may be eliminated without cost to the effectiveness of the model, as is shown in the S. Maji's exploration [14]. This was explored through various correlation analysis.

Finding the correlation between variables helped identify possible redundancies. Maintaining a set of redundant variables does not serve to add any additional insight to a machine learning model. When compared to keeping only one of the variables, the model should perform nearly as well but with improved training and testing time. Figure 3.1 shows the correlation between features of the UNSW-NB15 dataset in a heatmap.

Finding the correlation between features of the dataset and the attack label serves as a helpful preliminary step at identifying which features might be particularly useful for a classifier. Appendix A.2 gives the correlations between each feature and the attack category for the UNSW-NB15 dataset.

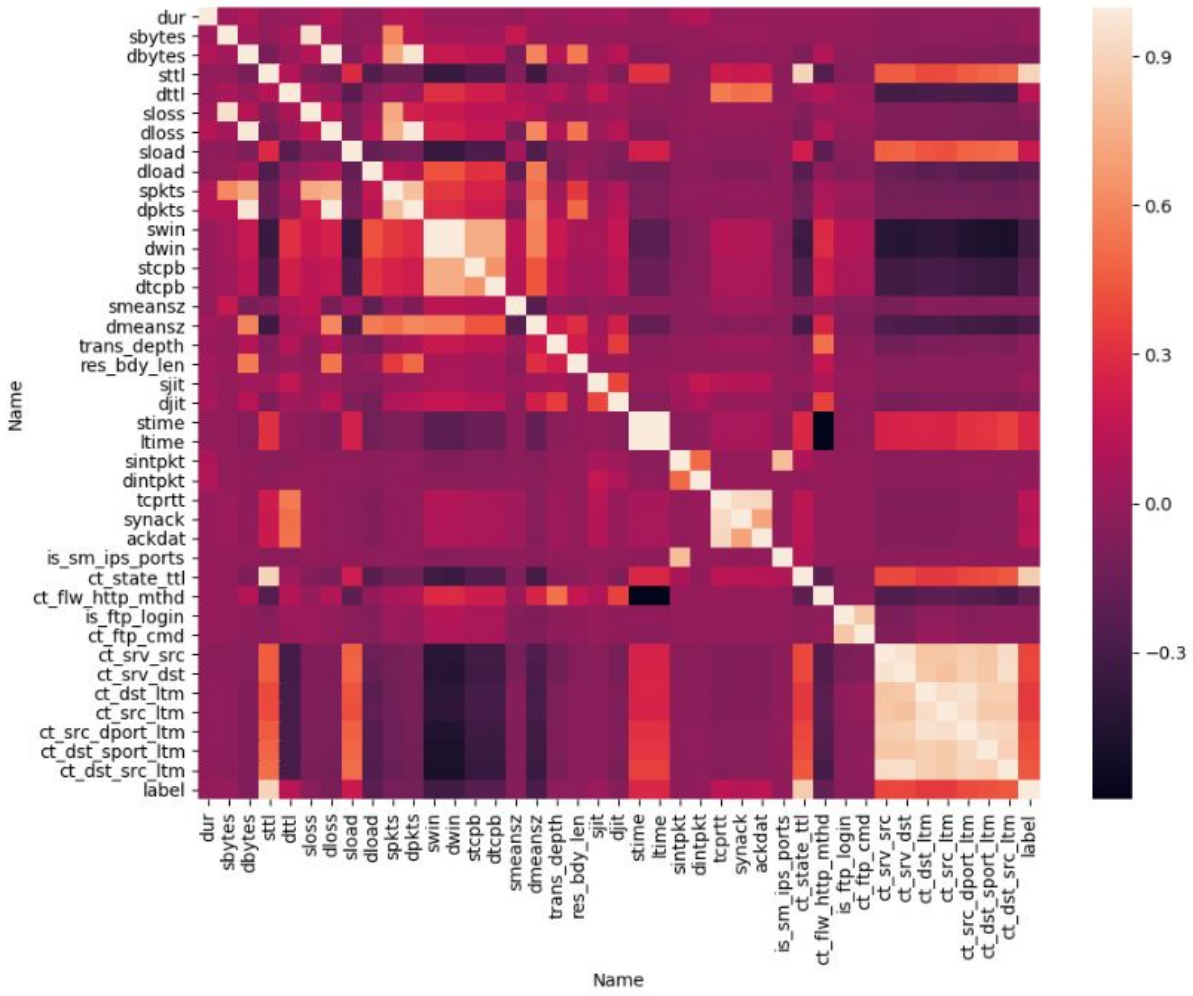

Figure 3.1: A heatmap of the correlation between variables in the UNSW-NB15 dataset.

3.3 Feature Engineering

The UNSW-NB15 dataset contains two variables, sbytes and dbytes, that represent the source bytes and destination bytes. Summing these two variables results in a new vector for the total number of bytes exchanged in the communication. While this information can be easily determined from the data, listing it explicitly may provide additional insight for the system.

It was also necessary to drop columns that are unhelpful to the machine learning models. For both kNN and MLP, any features that were specific to the generation of the dataset (IP addresses, timestamps, etc) were dropped. This included *srcip*, *sport*, *dstip*, *dsport*, *stime*, and *ltime*.

For the MLP model, one out of any two features that were highly correlated, having a correlation value of greater than 0.95, with each other were dropped. The features dropped due to high correlation were *sloss*, *dloss*, *dpkts*, *dwin*, *ltime*, *ct_srv_dst*, and *ct_src_dport_ltm*. The correlation values for these features with their highly correlated pairs can be found in table 3.1. A heatmap of all features correlations can be found in figure 3.1.

Kept Feature	Dropped Feature	Correlation
sbytes	*sloss*	0.9533
dbytes	*dloss*	0.9913
dbytes	*dpkts*	0.9708
swin	*dwin*	0.9972
stime	*ltime*	0.9999

Table 3.1: Pairs of features with correlations higher than 0.95 along with whether or not they were kept in the feature set for the machine learning approaches used.

For kNN, any features that were highly correlated with the attack label served as a helpful starting point at selecting a feature set. These correlations can be found in Appendix A.2.

Any remaining columns needed to be feature scaled or encoded to avoid larger features overshadowing smaller ones. Each continuous feature was normalized by z-score to a mean of 0 and standard deviation of 1. Equation 3.1 is the z-score normalization equation used.

$$x' = \frac{x - \bar{x}}{\sigma} \tag{3.1}$$

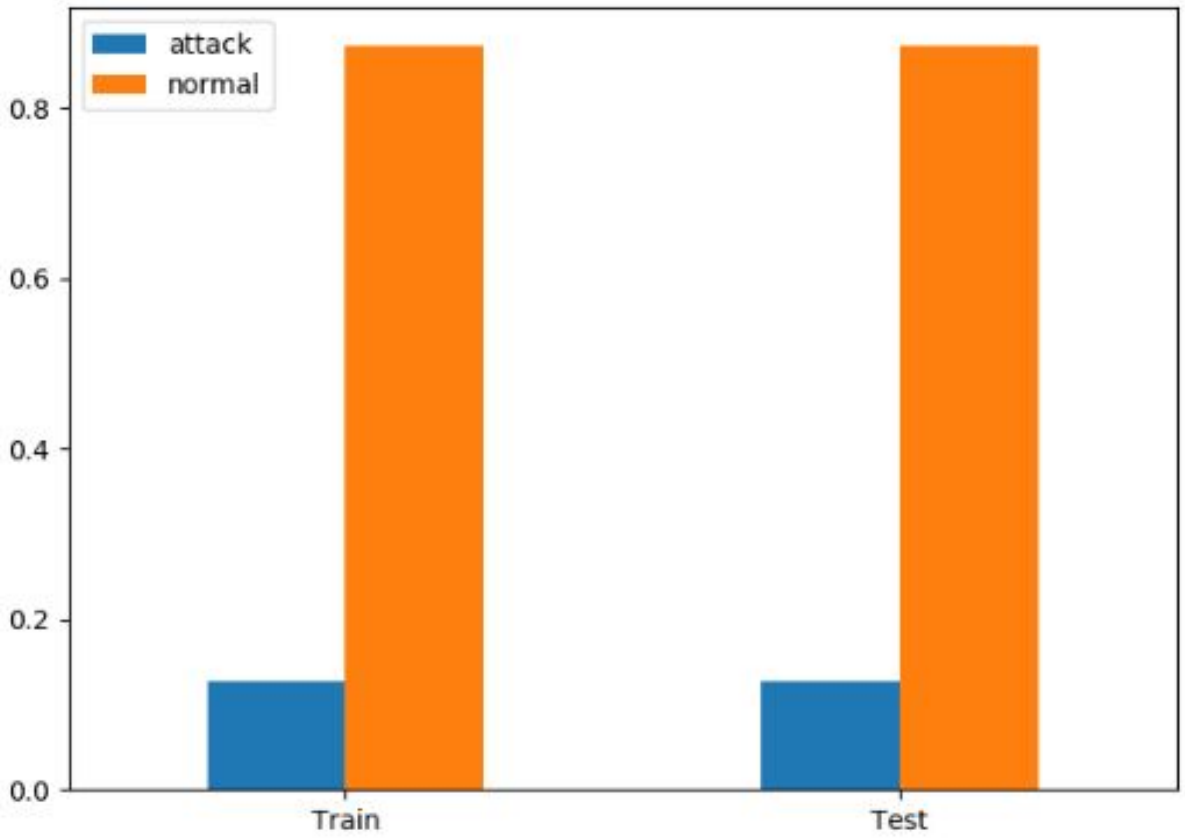

Figure 3.2: Label distribution on the train/test split.

Each categorical, non-binary feature was one-hot encoded into a group of binary features. Each feature in the group would represent one possible value of the original variable and within that group, one feature would be flagged as true to indicate that original category.

Before the data can be used in machine learning applications, it needs to be split into a training set and a testing set. For both subsequent models, an 80/20 train/test split was used. Figure 3.2 shows the proportions of attack labels between the training and testing sets, verifying that they both have an appropriately similar distribution.

3.4 Performance Metrics

Performance metrics for this application can be divided into two general categories: how effective the model is and how practical the model is. Measures regarding how effective the model

is were created using information from the test set predictions and actual values.

	Predicted Attack	Predicted Normal
Actual Attack	True Positive	False Negative
Actual Normal	False Positive	True Negative

Table 3.2: Predicted values vs. actual values. Combinations of these fields are used in a variety of performance metrics.

F1 Score

The UNSW-NB15 dataset is heavily skewed towards normal data points. This makes looking at individual statistics and the general accuracy ineffective. S. Raschka describes that the F-Measure is an effective measure for such datasets [16].

The F1 Score is the F-Measure of a binary classifier. It combines both the precision and recall, equations 3.3 and 3.4, to evaluate both how precise and how robust the model is.

$$F1 = \frac{2}{recall^{-1} + precision^{-1}} \tag{3.2}$$

$$Precision = \frac{tp}{tp + fp} \tag{3.3}$$

$$Recall = \frac{tn}{tn + fp} \tag{3.4}$$

Equation 3.2 is the equation for the F1 score of a system's performance. A perfect F1 score is 1, indicating both perfect precision and recall. The worst possible F1 score is 0, indicating either 0 precision or 0 recall.

The F1 measure is a good measure for this data because of the high skew towards normal values. The F1 measure equally weights the precision and recall, so a successful model can only be recognized as such if it is both able to properly classify correct values and avoid misclassifying

values [16].

ROC AUC Score

The Receiver Operating Characteristic (ROC) curve plots the True Positive Rate (TPR) seen in equation 3.5 against the False Positive Rate (FPR) seen in equation 3.6. The area under curve (AUC) of the ROC curve is a measure of how effective a model is at making correct predictions [17]. A perfect ROC AUC score is 1, indicating no mis-classifications, and a score above 0.5 indicates that a model is more likely than not to make correct predictions as described by S. Narkhede [18].

$$TPR = \frac{TP}{TP + FN} \tag{3.5}$$

$$FPR = \frac{FP}{FP + TN} \tag{3.6}$$

The two components of the ROC AUC score represent two critical classification rate in intrusion detection. The TPR represents the ability that the model has to correctly identify attacks. In intrusion detection, this is a particularly critical score, as the repercussions for a false negative are greater than that of a true positive. The FPR, or False Alarm Rate (FAR), identifies the tendency of the model to misclassify normal behaviors as malicious [15]. While a high FAR is still not good, the damage caused in an ICS by flagging normal communications as possibly malicious is generally significantly less than allowing attacks in.

3.5 Baseline Analysis: kNN

The k-Nearest Neighbors (kNN) classifier works by comparing an unknown data point to known, labeled data points and taking a vote of the k nearest points, by some established distance measure. Lin et al. support kNN as generally a good baseline for machine learning approaches because it is easy to implement and will correctly classify a majority of data points, so long

as they are relatively clustered with each other. In intrusion detection, this inherent clustering is often the case because if a vulnerability is found, exploits will generally focus on that vulnerability and end up looking similar to each other [19].

kNN generally serves as an excellent benchmark for intrusion detection applications. With many attacks exploiting a particular vulnerability or pattern, clusters naturally appear around these types of communications [19]. This makes kNN a good benchmark because it is easy to implement and will catch a bulk of the attacks. Additionally, the extremely large size and depth of attacks that the UNSW-NB15 dataset holds is likely to make clusters more definite as attacks are likely to be repeated with minor changes. However, new attacks, attacks specific to an individual system, and outliers are harder to catch with kNN.

Distance Measures

To perform kNN, the data needs to be in a form that enables a distance measure to be taken [20]. Most commonly, this is the Euclidean distance measure. Categorical data requires some manipulation before a distance measure can be applied. The one-hot encoding process from the feature engineering step enabled the Euclidean algorithm to be used. One downfall of kNN is its inability to use multiple distance measures. In a situation with only binary categorical features, the Jaccard distance could be used more effectively than the Euclidean distance. Given the mixed nature of the dataset as well as the heavy favor towards continuous features, Euclidean distance worked effectively.

As an algorithm, kNN identifies clusters of data points to assert label membership. By nature, distance measures between points are inherently linear. Should the data have more complex, non-linear relationships between variables and the label, kNN may miss these irregular or non-linear clusters.

Feature Selection

Since the UNSW-NB15 dataset has a relatively small feature set, it is possible to run kNN across all features. However, looking into the spread of correlations between individual features and the label vector makes it clear that certain variables would be more beneficial than others. Additionally, lowering the number of features included would also substantially improve the

runtime of the algorithm. It is important to choose the correct subset of variables to use. Because kNN must compare all variables for each data point in order to classify, the runtime requirements can be excessively high for testing each data point. Additionally, certain variables can confound the clusters and adversely contribute to proper classification.

Determining the perfect subset of features to use for kNN, the one that finds the global maximum accuracy score, required a comprehensive search of all feature permutations. Since the dataset has 49 features, this would be an excessively time consuming feat. Instead, a variety of methods were used to find a satisfactory local maximum accuracy score.

To streamline the development process, a randomly selected one tenth subset of the data was selected each run for intermediate testing. The full set was only used to determine the final baseline score and to occasionally validate results along the way.

Determining an optimal feature set was done as follows:

1. A test was run on the full feature set of the data.

2. Subsequent tests were run on the feature sets that the original UNSW-NB15 paper outlined; basic features, flow features, content features, time features, and additional generated features.

3. Each of these tests were run several times to establish which value of k, the number of neighbors used for classification, would be most effective. This was determined to be $k = 5$ and was used for the remaining tests.

4. It was noted that the basic features feature set performed the best, followed by the full feature set.

5. Feature correlations were calculated between the informational (non-meta) features of the dataset and the attack label.

6. Test were run on different combinations of the most strongly correlated feature, including positive, negative, and mixed correlations.

7. Beyond the first few features, correlation was not a helpful indicator of kNN accuracy, so individual features were tested instead.

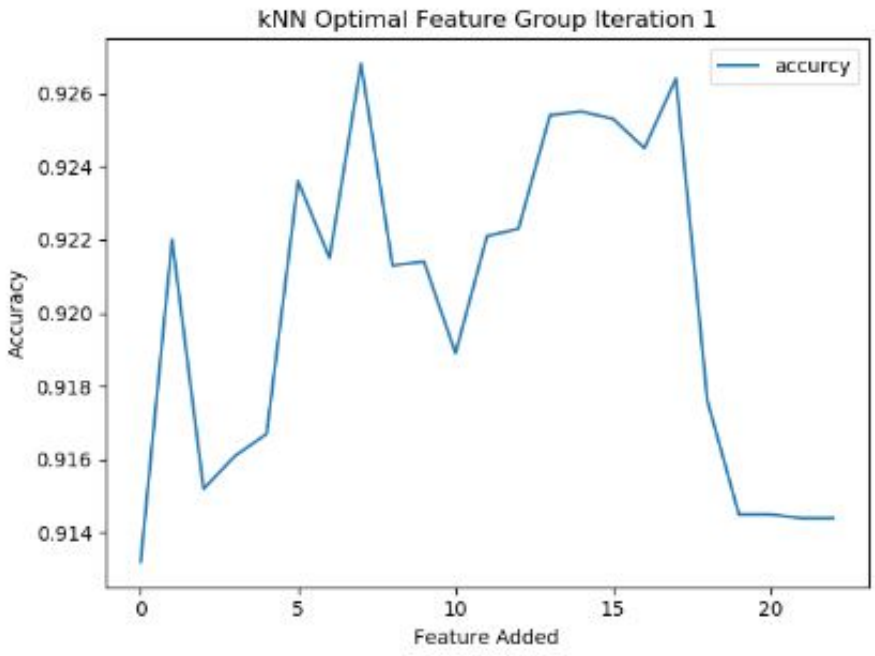

Figure 3.3: The resulting kNN accuracy by adding features in order of best individual kNN performance.

8. kNN was run on individual features and they were then ordered by performance.

9. Starting with the best performers, *sbytes* and *sload*, the next features were added on and their group performance was noted along with whether or not it was more effective than without the new feature.

10. Each feature that resulted in a positive delta performance was then grouped together and the process was repeated until a satisfactory result, a local maximum accuracy, was obtained.

The first iteration of testing features resulted in a maximum accuracy score of 0.92675. Appendix A.3 shows the strength of each feature in a single-feature kNN. This is the order that features were then tested in the group. Figure 3.3 shows the group performance as each new feature was added.

The second iteration of testing features began with the set created by taking only the features that had a positive effect on the group performance from the previous test. Adding these features incrementally produced figure 3.4. Notably, the best performance of this resulted

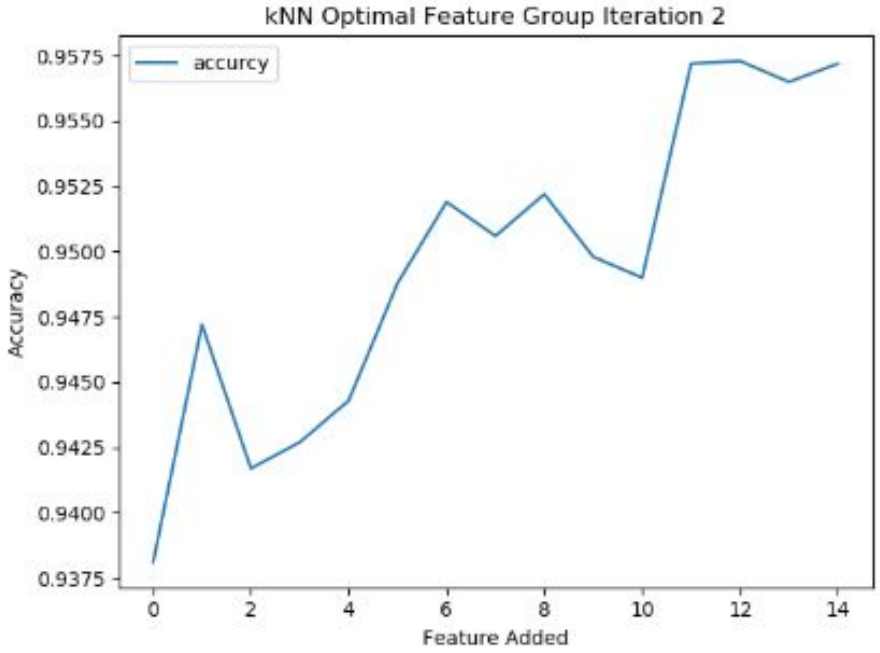

Figure 3.4: The resulting kNN accuracy created by adding features that resulted in a positive accuracy delta during the previous test (iteration 2).

in an improved accuracy of 0.9573.

This process was repeated again, taking only the features that resulted in a positive delta from iteration two and adding them incrementally. The resulting best kNN accuracy of 0.9562 was never able to surpass the previous best, so it was determined that a local maximum had been reached, and the optimal feature set would be the one that resulted in the best score from iteration two. Figure 3.5 shows the relative performance of this test.

Ultimately, the final feature set was comprised of 16 features: *sbytes*, *sload*, *smeansz*, *dload*, *dbytes*, *dmeansz*, *ct_state_ttl*, *sinpkt*, *sttl*, *spkts*, *dloss*, *ct_srv_dst*, *ct_dst_src_ltm*, *ct_dst_ltm*, *trans_depth*, and *ct_ftp_cmd*. Some of these features listed were one-hot encoded; for simplicity of listing values, each group of encoded columns has been grouped back into their original column name. The method used for parameter selection is not guaranteed to produce the optimal result. Because of the complexity of the dataset features, not all possible combinations could be tested and not all possible methods of feature engineering could be applied in a reasonable amount of both development and run time. However, the feature selection done should yield a value at or close to a local maximum with results satisfactory for a baseline assessment.

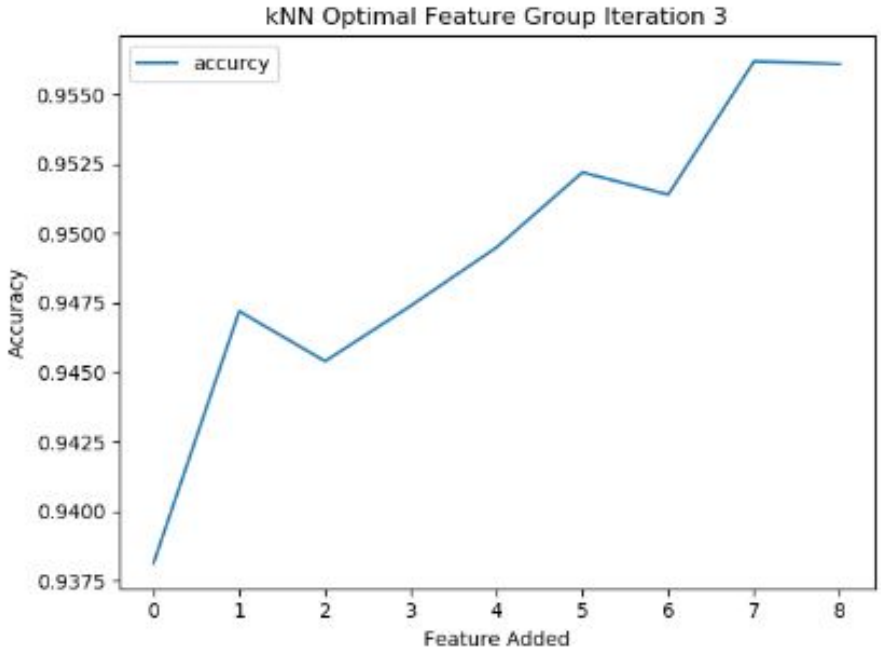

Figure 3.5: The resulting kNN accuracy created be adding features that resulted in a positive accuracy delta during the previous test (iteration 3).

In exploring the effectiveness of individual features on kNN results, certain features, when added or removed from the active feature set, could either increase or decrease the accuracy of the algorithm, depending on the remaining feature set used. Due to the linear nature of kNN and this fact that certain features can be more effective when paired leads to the hypothesis that there are more complex patterns in the data that, if properly identified, may yield better results than the kNN baseline.

3.6 Advanced Method: Multi-Layer Perceptron

A Multi-Layer Perceptron supervised learning algorithm and is a form of neural network that includes one or more hidden layers [21]. It can be used as both a classifier or a regressor but in this application we will be using its classifier functionality. The MLP classifier seeks to learn a non-linear function.

$$f(\cdot) : R^m -> R^o \tag{3.7}$$

Equation 3.7 describes a multi-layer perceptron where m is the dimensionality of the feature set and o is the possible options for classification. The model is trained using the backpropogation algorithm over the training set. Tests are then evaluated by passing the m features of the test point into the input of the model and the resulting o options are then output with a likelihood weight. The highest weight is then assigned as the classification.

In this instance, after one-hot encoding the input data has 197 features: $m = 197$. Because the data is binary classified as attack or not, the output vector is size 2: $o = 2$.

Multi-Layer Perceptrons are able to represent complex non-linear functions. Each neuron in the neural network implements a weighted linear sum of the values output by the neurons of the previous layer. By combining multiple layers, a neural network is able to identify and represent complex non-linear functions. This is particularly useful in intrusion detection because while many attack data points fall into linear clusters, capturing new or more sophisticated attacks requires non-linear capabilities.

Multi-Layer Perceptrons are susceptible to local minimums dependent on the random weight initialization of the untrained model. A given set of tuned parameters does not guarantee the best possible outcome of the model or an identical outcome from retraining a similar model.

Once an MLP classifier is built and trained, testing datapoints is very quick and the testing and classification of data can be used to continue to train the model in production. In a network environment where network communications are changing and adapting, this can make a model longer standing.

One of the benefits and challenges of an MLP are the variety of hyperparameters available to tune. These options enable MLP classifiers to be effective in a multitude of applications but they also then need to be tuned to determine an effective configuration, which can be time consuming.

Hyperparameter Tuning

The most significant hyperparameters available to the MLP Classifier are the activation function and the solver which are used for the initialization of weights and for how they are updated in training. The particular choice of activation function and solver also unlocks additional

parameters that can be tuned.

Hyper-parameter tuning also requires performance metrics to assess and validate performance. For this, the F1 score and ROC AUC scores were chosen, prioritizing ROC AUC and then F1, though the performance of these two measures will generally scale together as can also be seen in the experiments of S. Maji [14].

Since the UNSW-NB15 dataset contains such a large number of datapoints, a stochastic gradient descent based optimization function for the solver is likely the best selection. The Sci-Kit Learn libraries give two such options: SGD and Adam. Beginning with each of these, most of the remaining hyperparameters do not have any intuitive best answer and will need to be tuned individually. The parameters tuned and options for each are listed in table 3.3.

Test #1: Solver=sgd

Parameter	Options tested
Solver	sgd
Activation Function	identity, logistic, tanh, relu
Alpha	1e-5, 0.0001, 0.001
Learning Rate	constant, invscaling, adaptive

Test #2: Solver=adam

Parameter	Options tested
Solver	adam
Activation Function	identity, logistic, tanh, relu
Alpha	1e-5, 0.0001, 0.001

Table 3.3: Multi-Layer Perceptron hyper-parameter configurations tested

Parameters are tuned and options are explored using a twice validated gridsearch to ensure the results are reliable. Ultimately, 72 tests were run for the SGD solver, 36 different parameter combinations twice validated, and 24 tests were run for the Adam solver, 12 different parameter combinations twice validated. In an ideal scenario, these tests would be five times validated, but due to runtime constraints twice validated provided sufficient results.

Chapter 4 Results

4.1 Baseline: kNN

The kNN algorithm achieved varying results depending on the selection of variables used for analysis.

The locally optimal feature set selected contained the following 16 features: *sbytes*, *sload*, *smeansz*, *dload*, *dbytes*, *dmeansz*, *ct_state_ttl*, *sinpkt*, *sttl*, *spkts*, *dloss*, *ct_srv_dst*, *ct_dst_src_ltm*, *ct_dst_ltm*, *trans_depth*, and *ct_ftp_cmd*. It was also determined that the most effective value of k is 5. Running kNN with these configurations produced an F1 score of 0.9720 and a ROC AUC score of 0.9834. Table 4.1 shows the confusion matrix for the best kNN predictions.

Predicted	Attack	Normal
Actual		
Attack	62404	1884
Normal	1716	442006

Table 4.1: The confusion matrix for the best kNN predictions

As a baseline exploration, these results are highly promising. For the optimized feature set, 0.97 is a high F1 score, especially for a straightforward algorithm like kNN with little to no feature engineering. Additionally, the fact that a smaller subset of features produced not only a more time efficient run but better results.

4.2 Multi-Layer Perceptron

The Multi-Layer Perceptron Artificial Neural Network has many more parameters to fine tune than the kNN baseline. Some of the parameters can be selected with confidence by looking at their design and documentation but other required experimentation and testing to find an optimized set.

Many of the MLP hyper parameters are dependent on the particular solver function. Each test begins with a solver function and uses gridsearch to tune the corresponding hyper-parameters.

Table 4.2 summarizes each experiment and its results.

Test #1: Solver=sgd

Parameter	Options tested	Best Selected
Solver	sgd	sgd
Activation Function	identity, logistic, tanh, relu	relu
Alpha	1e-5, 0.0001, 0.001	1e-5
Learning Rate	constant, invscaling, adaptive	adaptive
		F1 Score: 0.9633
		ROC AUC: 0.9825

Test #2: Solver=adam

Parameter	Options tested	Best Selected
Solver	adam	adam
Activation Function	identity, logistic, tanh, relu	tanh
Alpha	1e-5, 0.0001, 0.001	1e-5
		F1 Score: 0.9697
		ROC AUC: 0.9828

Table 4.2: Multi-Layer Perceptron tests and hyper-parameter configurations

Test #1: Stochastic Gradient Descent Solver

The first test used the Stochastic Gradient Descent solver (SGD), and tested the hyper-parameters activation, alpha, and learning rate. The most effective combination, an activation function of Rectified Linear Unit (RELU) function, an alpha value of 0.00001, and an adaptive learning rate, resulted in an F1 score of 0.9633 and a ROC AUC score of 0.9825. The most significant hyper-parameters in this test were the activation function and learning rate. An adaptive learning rate runs similar to a constant learning rate, but decreases each time consecutive epochs fail. Where a constant rate would quit early, the adaptive rate will narrowly approach the local maximum more before quitting. This explains the similar, yet marginally better results of the adaptive rate, as well as it's additional runtime.

Test #2: Adam Solver

The second test used the solver Adam, a modified Stochastic Gradient Descent function, and tuned the hyper-parameters activation and alpha. The most effective configuration, an activation function of hyperbolic tangent (tanh) and an alpha of 1e-5, resulted in an F1 score of

0.9697 and a ROC AUC score of 0.9828.

Analysis

While there was some variation in effectiveness, the parameters that were the most significant between tests were activation functions of either tanh or relu, outperforming the other options across both tests. Additionally, smaller alpha values also gave better results at the cost of higher runtime.

The best tuned MLP classifier from Test #2 produced results with an F1 score of 0.9679 and a ROC AUC score of 0.9828. Table 4.3 shows the confusion matrix for this model's predictions.

Predicted	Attack	Normal
Actual		
Attack	62358	1930
Normal	1966	441756

Table 4.3: The confusion matrix for the best MLP predictions

Comparing with the baseline results of kNN, this is a reasonably close score. A visual comparison of the performances of kNN and the two MLP tests is presented in figure 4.1. With the shorter test runtime of a trained neural network, this approach is arguably more practical than the kNN baseline; however, this model would still need to be improved or paired with other systems before it is sufficient for a production system. Other experiments have also produced better F1 scores than this one achieved by the MLP system [14].

Presumably, more exploratory data analysis and feature engineering could yield even better results by enabling the discovery of new patterns that may not be visible or as pronounced in this model. Additionally, the abundance of parameters to tune led to discovering a local maximum in performance and not the global maximum. With so many parameters to tune and the runtime required for training the neural network, it was not feasible to guarantee best results by a particular configuration.

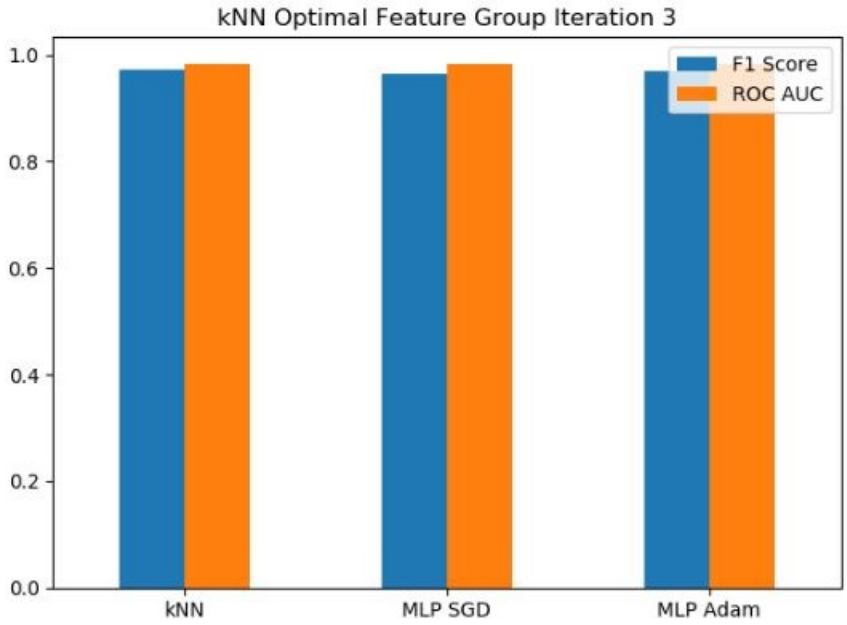

Figure 4.1: The performances of kNN, MLP stochastic gradient descent solver, and MLP Adam solver.

4.3 kNN and MLP Performance by Attack Category

Attack Type	No. Correct	No. Incorrect	Accuracy
No Attack	441756	1966	0.9956
Generic	43051	20	0.9995
Exploits	8771	155	0.9826
Analysis	458	84	0.8450
Fuzzers	3266	1554	0.6776
DOS	3247	51	0.9845
Reconnaissance	2771	47	0.9833
Backdoor	474	1	0.9979
Shellcode	295	18	0.9425
Worms	25	0	1

Table 4.4: MLP predictions by attack category

Table 4.4 shows the prediction success dependent on the type of attack for the MLP classifier while table 4.5 shows the prediction success dependent on the type of attack for the kNN classifier. Table 4.6 shows the comparative performance between the kNN and MLP classifiers.

Compared to the kNN classifier, the MLP model excelled at backdoor, shellcode, and worm attacks. The MLP model comparatively performed the worst at fuzzer attacks but was very

Attack Type	No. Correct	No. Incorrect	Accuracy
No Attack	442006	1716	0.9961
Generic	43040	31	0.9993
Exploits	8713	213	0.9761
Analysis	458	84	0.8450
Fuzzers	3380	1440	0.7012
DOS	3251	47	0.9857
Reconnaissance	2791	27	0.9904
Backdoor	468	7	0.9853
Shellcode	280	33	0.8946
Worms	23	2	0.92

Table 4.5: kNN predictions by attack category

close for remaining attacks/normal values. Looking at the sizes of each subset, The kNN classifier performed better on attack categories with larger data points whereas the MLP classifier performed better on categories with fewer data points.

Attack	kNN Performance	MLP Performance	Improvement
No Attack	0.9961	0.9956	-0.0005
Generic	0.9993	0.9995	+0.0002
Exploits	0.9761	0.9826	+0.0065
Analysis	0.8450	0.8450	±0
Fuzzers	0.7012	0.6776	-0.0236
DOS	0.9857	0.9845	-0.0012
Reconnaissance	0.9904	0.9833	-0.0071
Backdoor	0.9853	0.9979	+0.0126
Shellcode	0.8946	0.9425	+0.0479
Worms	0.92	1	+0.08

Table 4.6: Comparison of the kNN and MLP performance by attack category

Chapter 5 Conclusion and Future Work

Multi-Layer Perceptrons exemplified a potentially effective measure of intrusion detection for network cybersecurity. Achieving an F1 score of 0.9679 and a ROC AUC score of 0.9828, as compared to the kNN F1 score of 0.9720 and ROC AUC score of 0.9834. MLP was also able to improve upon the accuracy of kNN for lesser recorded attacks. MLPs illustrated that they were able to begin to recognize some of the complex patterns of identifying attack vectors in network communications while retaining general performance. Additionally, once trained, MLPs make a practical application of machine learning for network intrusion detection due to their shorter testing time.

Machine learning models targeting more complex patterns, such as MLP, are generally benefited by the comprehensive nature of the dataset. The fact that kNN performed best on a smaller subset of the dataset's features than the MLP approach also indicates that the inclusion of detailed data when analyzing network communications benefits machine learning methods targeting more complex patterns.

The UNSW-NB15 dataset represents more generic network traffic than could typically be found within an ICS setting. The inclusion of this more specific data would enable a more effective model however limiting the scope of the system to that in which the data was recorded.

Based on the data in this thesis, the results from kNN and the MLP classifier were satisfactory for intrusion detection, however they are not yet sufficient for a system that can be trusted for critical infrastructure systems in production.

Both systems in this thesis were binary classifiers trained on a dataset of both known anomalies and normal traffic. This would make both systems unlikely to catch zero-day attacks unless they particularly resembled an existing attack. The MLP approach can be restructured into a regressor that predicts the likelihood that a particular data point is an anomaly rather than a binary prediction. This could possibly produce better results at catching zero-day and previously unseen attacks.

Future Work

Further development of intrusion detection for Industrial Control Systems, would benefit from a few genres to be expanded:

ICS Specific Datasets: The UNSW-NB15 dataset contains general information that can be applied to nearly any network-connected device. To build strong defenses for a given industrial control system, it would be helpful to also explore ICS specific datasets, particularly those that represent systems similar to the one to be protected.

Improving the Dataset: The UNSW-NB15 dataset contains exceptional information on general network traffic. Expanding into ICS-specific traffic would give more insights into the benefits and challenges of setting up an Intrusion Detection And Prevention System (IDPS).

Exploratory Data Analysis: There are multiple ways to scale and represent feature values within the dataset. In these experiments each feature was represented on its original scale; it is possible that some features provide more insight into the data if scaled in other ways. For example, logarithmic representations of some of the data or boxing it into categories may improve the performance of a model. This concept would require additional exploratory data analysis to assess these correlations with the attack labels before they can be tested on existing machine learning models.

Exploring other methodologies: Neural networks and multi-layer perceptrons are only one of many machine learning models that yield successful results in such an experiment. Adding to the MLP approach by exploring other tuning factors, models, and combinations of models could yet give results that surpass those of this experiment. Additionally, a replicator neural network or an autoencoder could be used in place of the multi-layer perceptron. These models specialize at anomaly detection. The success of the MLP approach would merit an exploration into those models for an ICS intrusion detection system.

Model Assessment: F1 score and ROC AUC are two of the most effective means of measuring performance of an intrusion detection system. There are other methods for assessing the effectiveness of a system which should be explored.

Bibliography

[1] D. Bhamare, M. Zolanvari, A. Erbad, R. Jain, K. Khan, and N. Meskin, "Cybersecurity for industrial control systems: A survey," *Computers & Security*, vol. 89, p. 101677, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404819302172

[2] K. Stouffer, V. Pillitteri, M. Abrams, and A. Hanh, "Guide to industrial control systems (ics) security," *NIST Special Publication 800-82*, 2015.

[3] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys Tutorials*, vol. 18, pp. 1153–1176, 2016.

[4] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *Applied Sciences*, vol. 9, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/20/4396

[5] D. Mudzingwa and R. Agrawal, "A study of methodologies used in intrusion detection and prevention systems (idps)," in *2012 Proceedings of IEEE Southeastcon*, 2012, pp. 1–6.

[6] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets, and challenges," *Cybersecurity*, vol. 2, 2019.

[7] J. M. Beaver, R. C. Borges-Hink, and M. A. Buckner, "An evaluation of machine learning methods to detect malicious scada communications," in *2013 12th International Conference on Machine Learning and Applications*, vol. 2, 2013, pp. 54–59.

[8] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set," *Information Security Journal: A Global Perspective*, vol. 25, pp. 18–31, 2016. [Online]. Available: https://doi.org/10.1080/19393555.2015.1125974

[9] Y. Hamid, V. R. Balasaraswathi, L. Journaux, and M. Sugumaran, "Benchmark datasets for network intrusion detection: A review," *International Journal of Network Security*, vol. 20, 2018.

[10] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hortho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016740481930118X

[11] L. Dhanabal and S. P. Shantharajah, "A study on nsl-kdd dataset for intrusion detection system based on classification algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, 2015.

[12] A. Özgür and H. Erdem, "A review of kdd99 dataset usage in intrusion detection and machine learning between 2010 and 2015," *PeerJ Preprints*, vol. 4, 2016. [Online]. Available: https://doi.org/10.7287/peerj.preprints.1954v1

[13] T. Morris, "Industrial control system (ics) cyber attack datasets," WebPage. [Online]. Available: https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets

[14] S. Maji, "Building an intrusion detection system on unsw-nb15 dataset based on machine learning algorithm," WebPage, September 2020.

[15] R. Zeuch, T. M. Khoshgoftaar, N. Seliya, M. M. Najafabadi, and C. Kemp, "A new intrusion detection benchmarking system," in *Proceedings of the Twenty-Eighth International Florida Artificial Intelligence Research Society Conference*, 2015.

[16] S. Raschka, "About feature scaling and normalization – and the effect of standardization for machine learning algorithms," WebPage, July 2014. [Online]. Available: https://sebastianraschka.com/Articles/2014_about_feature_scaling.html

[17] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 233–240. [Online]. Available: https://doi.org/10.1145/1143844.1143874

[18] S. Narkhede, "Understanding auc-roc curve," WebPage, June 2018. [Online]. Available: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

[19] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "Cann: An intrusion detection system based on combining cluster centers and nearest neighbors," *Knowledge-Based Systems*, vol. 78, pp. 13–21, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705115000167

[20] H. A. Abu Alfeilat, A. B. Hassanat, O. Lasassmeh, A. S. Tarawneh, M. B. Alhasanat, H. S. Eyal Salman, and V. S. Prasath, "Effects of distance measure choice on k-nearest neighbor classifier performance: A review," *Big Data*, vol. 7, p. 221–248, Dec 2019. [Online]. Available: http://dx.doi.org/10.1089/big.2018.0175

[21] J. Esmaily, R. Moradinezhad, and J. Ghasemi, "Intrusion detection system based on multi-layer perceptron neural networks and decision tree," in *2015 7th Conference on Information and Knowledge Technology (IKT)*, 05 2015, pp. 1–5.

Appendix

A.1 UNSW-NB15 Features and Feature Groups

Features and Feature Groups from Nour Moustafa and Jill Slay [8]

Flow Features

No.	Feature Name	Description
1	srcip	Source IP address
2	sport	Source port number
3	dstip	Destination IP address
4	dsport	Destination Port number
5	proto	Protocol type

Basic Features

No.	Feature Name	Description
6	state	Indicates that state and its dependent protocol
7	dur	Record total duration
8	sbytes	Source to destination bytes
9	dbytes	Destination to source bytes
10	sttl	Source to destination time to live
11	dttl	Destination to source time to live
12	sloss	Source packets re-transmitted or lost
13	dloss	Destination packets re-transmitted or lost
14	service	Such as http, ftp, smtp, ssh, dns, and ftp-data
15	sload	Source bits per second
16	dload	Destination bits per second
17	spkts	Source to destination packet count
18	dpkts	Destination to source packet count

Content Features

No.	Feature Name	Description
19	swin	Source TCP window advertisement value
20	dwin	Destination TCP window advertisement value
21	stcpb	Source TCP base sequence number
22	dtcpb	Destination TCP base sequence number
23	smeansz	Mean of the flow packet size transmitted by the source
24	dmeansz	Mean of the flow packet size transmitted by the destination
25	trans_depth	Represents the pipelined depth into the connection of http request/response transaction
26	res_bdy_len	Actual uncompressed content size of the data transferred from the server's http response

Time Features

No.	Feature Name	Description
27	sjit	Source jitter in milliseconds
28	djit	Destination jitter in milliseconds
29	stime	Record start time
30	ltime	Record last time
31	sinpkt	Source inter-packet arrival time in milliseconds
32	dinpkt	Destination inter-packet arrival time in milliseconds
33	tcprtt	TCP connection setup round-trip time; sum of 'synack' and 'ackdat'
34	synack	TCP connection setup time; the time between the SYN and SYN_ACK packets
35	ackdat	TCP connection setup time; the time between the SYN_ACK and ACK packets

Additional Generated Features

No.	Feature Name	Description
36	is_sm_ips_ports	If both the source and destination IP addresses and ports are the same then this is flagged with a 1, else 0
37	ct_state_ttl	No. for each state (6) according to specific ranges of values for sttl (10) and dttl (11)
38	ct_flw_http_mthd	No. of flows that contain HTTP methods within an HTTP service
39	is_ftp_login	If the ftp session is accessed by user and password then 1, else 0
40	ct_ftp_cmd	No. of flows that has a command in ftp session
41	ct_srv_src	No. of records that contain the same service (14) and srcip (1) in the last 100 records, according to ltime (26)
42	ct_srv_dst	No. of records that contain the same service (14) and dstip (3) in the last 100 records, according to ltime (26)
43	ct_dst_ltm	No. of records of the same dstip (3) in the last 100 records according to ltime (26)
44	ct_src_ltm	No. of records of the same srcip (3) in the last 100 records according to ltime (26)
45	ct_src_dport_ltm	No. of records of the same srcip (1) and the dsport (4) in the last 100 records according to ltime (26)
46	ct_dst_sport_ltm	No. of records of the same dstip (3) and the sport (2) in the last 100 records according to ltime (26)
47	ct_dst_src_ltm	No. of records of the same srcip (1) and the dstip (3) in the last 100 records according to ltime (26)

A.2 UNSW-NB15 Feature and Attack Label Correlation

Feature	Attack Label Correlation
sttl	0.5042
ct_dst_sport_ltm	0.3937
ct_src_dport_ltm	0.3415
rate	0.3286
ct_state_ttl	0.3185
ct_srv_dst	0.2929
ct_srv_src	0.2902
ct_dst_src_ltm	0.2800
ct_src_ltm	0.2765
ct_dst_ltm	0.2580
sload	0.1245
sbytes	0.0206
state	0.0142
sloss	0.0064
dur	-0.0011
is_ftp_login	-0.01621
response_body_len	-0.0164
ct_ftp_cmd	-0.0171
trans_depth	-0.0258
djit	-0.0271
sjit	-0.0274
spkts	-0.0277
dbytes	-0.0326
dinpkt	-0.0376
proto	-0.0384
dloss	-0.0444
smean	-0.0611
dpkts	-0.0615
ct_flw_http_mthd	-0.0750
dttl	-0.0986
is_sm_ips_ports	-0.1174
ackdat	-0.1205
sinpkt	-0.1208
tcprtt	-0.1488
synack	-0.1499
dmean	-0.2115
dload	-0.2805
stcpb	-0.2814
dtcpb	-0.2829
dwin	-0.3693
swin	-0.4145

A.3 UNSW-NB15 Feature and Individual kNN Accuracy

Feature	Single-Feature kNN Accuracy
sbytes	0.8857
sload	0.8857
smeansz	0.8429
dinpkt	0.8368
dload	0.8206
dbytes	0.7990
dur	0.7929
dmeansz	0.7872
synack	0.7855
tcprtt	0.7844
ackdat	0.7743
ct_state_ttl	0.7737
sinpkt	0.7687
sttl	0.7685
sjit	0.7673
dttl	0.7543
djit	0.7540
dpkts	0.7505
dtcpb	0.72462
stcpb	0.7238
spkts	0.7161
sloss	0.7146
swin	0.7044
dloss	0.6946
dwin	0.6840
ct_srv_dst	0.6304
ct_srv_src	0.6272
ct_dst_src_ltm	0.5672
ct_dst_ltm	0.5595
ct_src_dport_ltm	0.5582
trans_depth	0.5539
ct_flw_http_mthd	0.5530
ct_ftp_cmd	0.5508
ct_src_ltm	0.5437
ct_dst_sport_ltm	0.5288

Kurt Lamon

(425) 440-1584 | klamon955@gmail.com | www.kurtlamon.com

Education

EASTERN WASHINGTON UNIVERSITY **SEPTEMBER 2019-SPRING 2021 (EXPECTED)**

- Master of Science in Computer Science (4.0 GPA)

GONZAGA UNIVERSITY **SEPTEMBER 2015-MAY 2019**

- B.S. in Computer Science, Minor in Mathematics (3.36 GPA)

UNIVERSITY OF AUCKLAND **JUNE-NOVEMBER 2016**

- Gonzaga-partnered study abroad program in Auckland, New Zealand (3.54 GPA)

Technical Skills and Courses

Languages: Java, Kotlin, Python, C/C++, JavaScript, SQL, HTML/CSS, XML, MatLab, R, Swift, x86_64 Assembly, Verilog
Frameworks/Platforms: Git/GitHub, Jira, Android Studio, Node.js, Jupyter, AngularJS, RX, Mokito, Amazon AWS
General Skills: Mobile/Web Apps, Machine Learning, Data Science, Agile Management, Object Oriented Programming
Course Highlights: Natural Language Processing, Data Mining and Big Data, Parallel and Cloud Computing, Database Systems, Machine Learning and Artificial Intelligence, Computer Networks and Security, Web and Mobile development, Embedded Systems, Research Methods, Computer Architecture, Object Oriented Programming

Work Experience and Projects

STRAVA ANDROID ENGINEER INTERNSHIP **SUMMERS 2019 & 2020**

- Wrote high quality, sustainable, and testable code to implement new app features
- Developed internal tools that help streamline workflow and debugging
- Collaborated with designers and data analysts to determine the impact features have on users

Anti-Racism and Hate Speech Reporting Features, Developer June-September 2020

- Worked with researchers to identify unique needs of POC athletes
- Collaborated with other developers and product designers to create new safety features

Challenge Celebration Dialog, Project Lead and Primary Developer June-September 2019

- Led a project creating features that encourage users to participate in challenges and redeem rewards
- Designed and implemented a new local database schema optimized for local storage
- Utilized RX to handle threading for API and database calls

ARTIFICIAL INTELLIGENCE

Othello AI, Co-developer April 2019-May 2019

- Utilized breadth-first search, alpha-beta pruning, and a four part heuristic to create a successful AI
- Placed 2nd in a single elimination tournament of 19 student Othello AIs

WEB DEVELOPMENT AND DATABASE MANAGEMENT

The Food Guide, Creator, Database Developer, and Full Stack Web Developer April 2018-May 2019

- Designed a relational database for tracking users, restaurants, menus, items, and ingredients
- Developed user experience elements and control flow to create an intuitive application

Achievements

- First Annual Spokane Mayor's Cup: Cybersecurity Hackathon — First Place Team February 2019
- Second Annual Spokane Mayor's Cup: Cybersecurity Hackathon — Coach/Volunteer February 2020
- Computer Science/Computer Engineering Tutor/TA/GSA 2016-Present
  - Tutored, graded, and gave occasional fill-in lectures for foundational computer science courses at EWU
  - Mentored and tutored freshmen, sophomores, and juniors in fundamental computer science courses
- Boy Scouts of America, Eagle Scout, Senior Patrol Leader, Instructor 2010-2015

Hobbies and Interests

- Cycling, running, rock climbing, environmentalism, hiking and travel, cooking, piano, painting and art, robotics