1-2021

Towards Enhancement of Machine Learning Techniques Using CSE-CIC-IDS2018 Cybersecurity Dataset

Dharshini Ravikumar
dr3086@rit.edu

Towards Enhancement of Machine Learning Techniques Using CSE-CIC-IDS2018 Cybersecurity Dataset

by

Dharshini Ravikumar

A Thesis Submitted in Partial Fulfilment of the Requirements for the

Degree of Master of Science in Networking and System Administration

Department of Computing Sciences

Rochester Institute of Technology

RIT Dubai

January 2021

Committee Approval

Dr Ali Raza — Date
Professor of Computing Sciences
Thesis Advisor

Dr Muhieddin Amer — Date
Professor and Chair
Electrical Engineering and Computing Sciences

Abstract

In machine learning, balanced datasets play a crucial role in the bias observed towards classification and prediction. The CSE-CIC IDS datasets published in 2017 and 2018 have both attracted considerable scholarly attention towards research in intrusion detection systems. Recent work published using this dataset indicates little attention paid to the imbalance of the dataset. The study presented in this paper sets out to explore the degree to which imbalance has been treated and provide a taxonomy of the machine learning approaches developed using these datasets. A survey of published works related to these datasets was done to deliver a combined qualitative and quantitative methodological approach for our analysis towards deriving a taxonomy.

The research presented here confirms that the impact of bias due to the imbalance datasets is rarely addressed. This data supports further research and development of supervised machine learning techniques that reduce bias in classification or prediction due to these imbalance datasets. This study's experiment is to train the model using the train, and test split function from sci-kit learn library on the CSE-CIC-IDS2018. The system needs to be trained by a learning algorithm to accomplish this. There are many machine learning algorithms available and presented by the literature. Among which there are three types of classification based Supervised ML techniques which are used in our study: 1) KNN, 2) Random Forest (RF) and 3) Logistic Regression (LR). This experiment also determines how each of the dataset's 67 preprocessed features affects the ML model's performance. Feature drop selection is performed in two ways, independent and group drop. Experimental results generate the threshold values for each classifier and performance metric values such as accuracy, precision, recall, and F1-score. Also, results are generated from the comparison of manual feature drop methods. A good amount of drop is noticed in the group for most of the classifiers.

Table of Contents

List of Tables

List of Figures

Thesis Content

This paper is divided into 5 chapters, where each chapter contains various subsections. The first chapter focuses on the Introduction and Background Concepts. Chapter two deals with Literation Review of the CSE-CIC-IDS2018 dataset. It highlights multiple topics such as paper related to the dataset and its framework. In the third chapter, literature analysis of Balancing Techniques for CIC dataset is focused. Chapter four describes the implementation setup along with dataset preprocessing steps and the machine learning classifiers. Chapter five is solely allocated for experimentation and analysis of the CIC dataset.

Chapter 1

1 Introduction

Recent growth in data volumes has been explosive due to the proliferation of applications that generate the data, which has to be collected, stored, and processed. Social media applications such as YouTube, Facebook, etc. and devices connected to the internet such as smart meter and autonomous vehicles, have added volume growth and the feature richness. Each event for a data transaction consists of a large volume of information about that specific event. With a high velocity of feature-rich data, systems would eventually struggle to cope with the computational and storage demand [1].

Security becomes a pivotal challenge to address in systems required to manage such high volumes of data. Information overload leads to stress among Business Owners and IT professionals. There are various types of data being recorded and stored online, which are of great value and importance [2]. Malicious third parties target vulnerabilities in systems and attempt to exploit them. In an exploit, a user can attempt to destroy, alter, and snitch illegal files. The sensitivity of the data requires the use of intelligent and real-time approaches.

We have systems developed and implemented to safeguard critical information, known as Intrusion Detection Systems (IDS) [3]. IDS is a software application that monitors malicious threats or attacks in a network in a timely-automatic manner. Network traffic is monitored to identify anomalous behaviours and patterns. With attackers' ability to use sophisticated tools that prevent such detection, more advanced approaches have been the focus of research, such as Artificial Intelligence (AI) techniques.

Machine Learning (ML) is a subset of approaches available in the domain of AI [4]. Neural networks are a branch of supervised ML and are driven by correctly labelled datasets. Once a

network is trained on a dataset, it can classify and predict unknown traffic into malicious and benign. Training of a neural network produces a model that can accurately predict or classify traffic.

In ML, the learning process is through feedback intending to make computers learn automatically without any assistance or human intervention. ML techniques deal with powerful and sophisticated attacks leading to natural computation and communication costs, high detection rates, and low false alarm rates [5].

Supervised machine learning (ML) techniques require labelled datasets to train a classification or prediction model. In a binary classification problem, the true or false labels need to be sufficient in quantity when each data sample has many features used as an input to train the model. An ML model's performance can be strongly affected by the dataset, which is used for training. An imbalanced dataset can lead to a biased classification or prediction [6].

Intrusion detection systems (IDS) can collect network traffic samples using fiber optical taps at various points in a network [7]. The samples can be used as inputs to a trained ML model to classify a benign or malicious data sample in the simplest case of binary classification [8]. Predictions could also be made on potential attacks using such models [9]. In a real-world enterprise environment, the traffic collected from these taps is expected to have a high benign ratio to malicious samples.

This paper focuses on the CSE-CIC IDS datasets, which was published in 2018. The dataset is both rich in terms of the features each data sample has and the different types of cyber-attacks. The goal of this study is to analyze the contributions which use these datasets for ML.

Research on the development of trained models which can accurately classify cyber intrusion events has received growing attention [10, 11]. The CIC-IDS dataset [12] has been recently used in several studies towards ML for IDS [1, 13]. In [14] and [15], the authors have reported the accuracy of proposed ML techniques that account for the dataset's imbalance. However, little research to date has focused on the imbalance of the dataset.

The results of this analytical study suggest that the aims are well defined. This is due to a high count of published work in which a treatment of these dataset's imbalance is missing. However, it is also encouraging to report that the imbalance issue is not entirely ignored in the literature, and some work has provided insights to this.

1.1 Background Concepts

1.1.1 Machine Learning Types

At a high level, the study of teaching or training a computer program is known as ML. The machine is trained to progressively improve at each level upon a set of tasks that it is provided. Furthermore, it is practically the study of how to mould an application to incorporate iterative progress. Several ways can be framed to propose this idea, but there are three types of recognized ML categories: supervised, unsupervised and reinforcement learning [16].

1) Supervised Learning

ML is the most popular and straightforward algorithm for implementation and is the easiest to understand and capture. Flashcards are one example which can be related to the concept of supervised learning. It is commonly identified as task-oriented. Since this type of learning is observed in many applications, it is encountered frequently [16]. On labelled data, the model is trained, which helps to learn the relationship between data points. After several training, the system is capable enough to make predictions. It can make a comparison between the predicted and expected output for the detection of errors. Then by using the ML model, these errors are rectified accordingly [17].

2) Unsupervised Learning

The opposite of supervised learning is unsupervised learning. It does not involve the concept of labelled data. In this learning, a tremendous amount of data is fed, and the given tools are allowed to understand the properties of data. It gains knowledge to organize, group, cluster, and make it sensible for humans to manage the data newly. This learning is more advantageous than

supervised learning since it can process large datasets without spending time on data preprocessing [16]. An abstract sequence pattern is noticed when it comes to data point relationships. No manual inputs are required from humans, making this learning versatile and open to dynamic changes [18].

3) Reinforcement Learning

Reinforcement learning is different from both supervised and unsupervised learning. There was a clear line of difference between supervised and unsupervised learning, but there is not much of a comparison in the case of reinforcement learning. Reinforcement learning is a type of ML that learns from mistakes. This is behaviour-driven learning which can be placed into any environment. This learning trains machine learning models to generate sequence decisions. It achieves a goal in an uncertain and potentially complex environment [16].

1.1.2 IDS, Deep Learning, and AI

IDS plays a significant role in detecting or identifying attacks, anomalies, and malware. It is an early warning device which acts as a defence system for network administrators against malicious traffic. An autonomous process of detecting anomalous behavior that aims to breach security policies in the network is a typical intrusion detection system. At present, attacks like brute force, SQL injection, bot, infiltration, denial of service, etc. have been established by cybercriminals in cyber warfare. These cyber-attacks dominate the internet by stealing sensitive data that lowers the global economy. IDs are categorized by extensive research as the most familiar network infrastructure to encounter cyber-attacks. ML methods are also used to detect intrusions and predict accuracy, performed through their self-learning capacity [19].

ML is one of the swiftly growing approaches [20] in the field of technology. This lie at the intersection of statistics and computer science, and the core of data science and AI. The growth in ML is due to the development of new learning algorithms, theory, availability of online data and low computation cost. Data-intensive ML approaches have been adopted by technology, commerce, and science, including healthcare, education, marketing, policing, and manufacturing.

Simulation of human intelligence in machines are programmed to think like humans and imitate their actions is known as AI. This term can be applied to any device that mimics the human nature or mind in problem-solving and learning. Execution or imitation of human tasks by AI can vary from the simplest to complex tasks. The main aim of AI includes the process of learning, reasoning, and perception. Algorithms play a significant role in the concept of AI. Complex algorithms frame strong intelligence, whereas simple ones are made using in simple applications. AI is used in several different industries, some of which includes healthcare and finance [21].

As a subdomain of ML, Deep Learning is a breakthrough in the cybersecurity domain [19]. It is an AI function that reciprocates the work of the human brain in the data processing. Human Brain activities like recognizing, speech, translating languages, detecting objects and decision making. This learning [22] can learn without human supervision that draws data from both unstructured and unlabeled. Deep learning is a form of ML through which money laundering or fraud can be detected. Hence, Deep learning is an AI function which is a subset of ML.

1.1.3 Basic Steps of ML Techniques

The strength of ML is to comprehend the difference using models than the judgment of humans. There are 7 steps that will lead and teach the working of ML, which are mentioned below [23]

1) **Gathering Data:** Once the requirements and equipments are analyzed, the data gathering step comes into action. This step is essential as it impacts the predictive model directly. The quality and quantity of data will determine the efficiency of the predictive model. This collected data is known as the training data which tabulated.
2) **Data Preparation**: In this step, the data is stored in a convenient place and then processed for use in ML. First, the data is combined, and its order is randomized since the data order should not have adverse effects on what is learned. During data preparation, data visualizations can be done to discover the various relationships present between the variables. Data imbalances can also be discovered. Here the data is split into two parts training and testing parts and also steps like normalization and error correction takes place in this step.

3) **Choosing a model:** Among the several models created by the researchers and scientists, an apt and right model needs to be selected to get reliable results
4) **Training:** This step is utilized to improve the ML model's ability to predict iteratively. The actual results generated during the training process are compared with the expected results to match the predictions.
5) **Evaluation:** Once the dataset's training is completed, the dataset's evaluation is measured by utilizing the data that was kept aside for the testing process to see if the results are good enough.
6) **Parameter Tuning:** To further improve the model, this step is performed. Various parameters are tuned or adjusted for improvement, which makes the model good.
7) **Prediction:** This is the last step where the value of ML is recognized. In this step, the values are used to predict the results finally.

1.1.4 Preprocessing Steps

Real-world data is many a time unpolished, uncertain, inaccurate, and lacks few characteristics or trends. The process of converting raw data into a recognizable form is called Data Preprocessing. It prepares raw data for further processing and is used in database type applications like neural networks. This technique is critical in ML mechanisms as it is difficult to encode the dataset in a structure that is understood and parsed by the algorithm [24].

There are six steps involved in the process of preprocessing, which are depicted in the figure below:

1) **Data Cleaning:** Data cleaning consists of steps such as deleting rows with omitted data or entering the missing values, smoothing the noisy data or clearing the data disparities. Data consistency is critical in machine learning, as machines cannot deal with data that cannot be interpreted. Inconsistencies can take place due to human blunders, such as storing the information in the wrong field. Deduplication can be performed to eradicate duplicate values to avert bias. Data that cannot be interpreted is of no use to machines; hence smoothing noisy data is essential for ML datasets. The data cleaning process includes dividing the data into equal sizes, then smoothed by regression or clustering.

2) **Data Integration:** Different representations of data are combined to resolve the contentions that arise within the data.
3) **Data Transformation:** In this step, the data undergoes a process of normalization & generalization. These processes ensure that there is no duplication of data stored in one place and has logical dependencies.
4) **Data Reduction**: Data reduction aims to reduce data representation in a data warehouse since large chunks of data can reduce performance, slower databases, expensive access, and storage challenges. There are several steps to minimise data; for example, when a subset of significant attributes is picked, everything is eradicated. To reduce the size of data, encoding mechanisms can be applied. After reducing data, if the original data can be recovered, the mission is stated as 'lossless' else, known as a 'lossy' reduction.
5) **Data Discretization:** This step focuses on reducing an extended attribute's values by breaking down the range of intervals associated with the attributes.
6) **Data Sampling:** When a dataset is too large, it becomes difficult to work with it, leading to storage & memory limitations. In such cases, sampling methods can provide a subset of data that can be performed with it. This subset of data would relatively hold the same properties as the original one.

1.1.5 Attacks of CSE-CIC-IDS2018

This cybersecurity dataset has been generated systematically using the notion of profiles that contain an accurate description of intrusions, applications with abstract distribution models, and protocols. Human operators or agents can make use of these profiles for the generation of network events. These profiles can be applied to a disparate range of network protocols with several topologies due to its abstract nature. A dataset for specific needs can be generated by using the profiles together. In [25], two different classes of profiles are built B-Profile & M-Profile. B-Profiles has been created to remove the abstract behavior of human users in a group. The network events generated by human users are encapsulated using ML and techniques like statistical analysis. M-Profiles try and attempt to define scenarios for each attack explicitly. Human users can easily understand these profiles and finally carry them out. There are several attacks in the CSE-CIC-IDS2018 which have been described below:

Infiltration: This attack is performed by sending a malicious file through email to the victim to exploit an application's vulnerability. After the attack has been used, a backdoor is performed on the victim's machine, then the victim's internal network is scanned to exploit vulnerable regions [25].

Brute Force Attacks: These attacks are widespread in systems with weak username and password and can easily break such systems. A brute force attack can be run against the primary server to acquire an SSH and MySQL accounts [25]. The attacks guess the passwords or encryption keys by trying different combinations using a word or pattern list to limit guesses [26].

SQL Injection: An attack allows the attacker to hinder the queries made to a database by an application. It is a web security vulnerability that allows the attacker to retrieve the information they can view or access. This information could either be a part of a user's data or applications data. This data can be modified or deleted by an attacker which changes the content of the application frequently [27].

DDoS: These attacks are flooding attacks which are the most significant concern prevailing over security professionals. Explicit attempts are made to disrupt authorized users access to their services or systems. To arrange a large attack army or Botnets, attackers exploit the vulnerability of several computer systems. Once these armies are formed, an attacker can provoke a large-scale organized attack on different targets [28].

DoS: The regular use of computer systems is prevented using the DoS attacks. This attack aims at overloading a network by continuously sending packets to debase the performance or by preventing users from accessing the target host system [26].

Botnet: These attacks make use of Bots known as Zombies which are computer systems that are already affected and infected by malware. These infected systems are activated to attack other systems by sending spam emails or DoS [26].

Heartbleed (Heartleech): it leads to exploitation based on vulnerability, which grants random access to a protected memory space of a remote system [29].

1.1.6 Motivation

In ML, data bias is a form of error that arises when specific components of a dataset are heavily weighted than the others. A model's use case is not accurately addressed by biased dataset since it results in skewed outcomes, analytical errors, and low accuracy levels. Training data for machine learning projects have to involve real-world scenarios in general. The machines can then get familiar with the real-world and learn to do their job accordingly [30].

In recent years the risk of bias has significantly impacted ML systems than people due to how the computer systems generate logical patterns. As ML systems grow, the data that is stored in it results in being profitable. Due to this hype, data has become the new trend since it is an excellent source for several products and produce extensive results article [31]

A growing body of literature recognizes the importance of balanced datasets to reduce the bias in ML. Inequality of classes is formed when malicious samples are lower than that of benign samples. As presented in [32, 33], imbalanced data causes low detection rates of minority classes where the dataset consisted of more instances belonging to the ethical behaviour class than the attack class. A drawback of this is presented in [33], which discusses the performance issues and proposed ML techniques as a mitigation approach. It significantly decreases performance, which results in data loss or overfitting. It intends the trainees to be biased towards the majority class, and all such classes will be identified correctly. Hence, to solve classifiers' low accuracy and reliability, balancing datasets is of primary importance for identifying sparse classes [34].

Datasets used for classification need to be well balanced so that a neural network trained using the dataset is not biased. A biased neural network model used to classify new, previously unseen data points will result in the wrong output. Many features in a dataset do not serve real-time security applications. A large and growing body of literature has investigated ML methods applied to cybersecurity datasets. This study found that 75 articles were published between 2017

and 2020 on ML, using the CCSE-CIC IDS dataset. In this work, the approaches used to ensure a balanced dataset will be investigated, explicitly considering what has been provided in the CIC repository.

1.1.7 Contribution

The goals of the study are as follows:

1. Provide a survey that:

    - Study and compare the research work of different authors on balancing datasets.
    - Analyze and investigate ML techniques that are used across different papers.
    - Investigate the approaches used to ensure a balanced dataset.
    - Identifies gaps in existing research and provides directions and recommendations for future research.

2. Find a labeled dataset related to IoT botnet attacks with opportunities to explore in the domain of Supervised Machine Learning.
3. Implement the algorithms required to evaluate the techniques proposed on the labeled dataset.
4. Compare and contrast the results of the new improvements for the techniques.

Chapter 2

2 LITERATURE REVIEW

In this chapter, we examine and discuss the related works appropriate to our thesis from which our research work has branched out.

2.1 Papers related to CSE-CIC-IDS2018

The significant features of the CSE-CIC-IDS2018 dataset have been presented below from different research papers.

Datasets have a prior role in the field of machine learning, which is a crucial component to train ML-based solutions. IDS are one of the most sophisticated and essential defense tools against ever-growing attacks. IDS using anomaly-based approaches can fail to detect attacks due to a lack of adequate datasets leading to inaccurate deployment, evaluation, and analysis [12]. Some datasets lack essential features such as traffic diversity, volumes, feature set, metadata, and reflection of current trends. Hence, a comprehensive and reliable dataset must be chosen depending on the approaches used.
CSE-CIC-IDS2018 is a popular dataset with a large number of published work exploring in the field of IDS. CSE-CIC-IDS2018 is developed using the AWS platform (Amazon Web Services). This dataset provides numerous attack profiles that can be used in intelligent security and applied to network topologies and protocols in a generic approach [35].

Datasets such as the CSE-CIC-IDS2018 dataset has been created for training the predictive models on network-based intrusion detection. These datasets are significantly used to endorse anomaly-based detection researches rather than signature-based detection systems through several ML approaches [36].
The dataset presents a stimulation of an attacking period of 10 days that uses 50 attack and 420 victim machines along with 30 servers [37]. Over these 10 days of network traffic,

16,233,002 instances were gathered. This dataset is the most recent and includes a variety of attack types which is available publicly. This dataset contains an imbalanced class distribution, with approximately 17% of instances of anomalous traffic [36].

This dataset is divided and distributed over 10 CSV files which can be downloaded from the cloud. Among these ten files, nine files have 79 independent features, and the rest contains 83 independent features. The cybersecurity datasets by CIC are worldwide known. CIC-IDS2018 contains most of the modern and current attacks such as Botnet, DoS (Denial of Service), Brute-force attacks and password cracking, Heartbleed, DDoS (Distributed Denial of Service), Web attacks (vulnerable web app attacks), and infiltration [35].

CSE-CIC-IDS2018 dataset is the most recent dataset in 2018/2019. Additionally, the standards used in CIC-IDS2017 was adopted to enhance the CSE-CIC-IDS2018 was created by incorporating and enhancing the standards used in 2017. There are various advantages and fundamental norms. The redundant data is meagre, the absence of uncertain data and the dataset can be used without processing as it is present in a CSV format. Furthermore, the creation of different scenarios and the collection of data were done daily. The dataset was edited, and raw data were noted every day. Data creation involves statistical properties of approximately 80, such as the number of bytes, packet length, number of packets, etc. These properties are separately estimated in the forward and reverse direction, giving a piece of information if an attack is added. Finally, the eventual dataset with approximately 5 million data in both PCAP and CSV is released to the researchers over the internet. If a new feature has to be extracted PCAP (unprocessed) form of data must be used, and if AI is to be used the CSV form of the dataset should be used [32].

This dataset is a project that resulted due to the collaboration of Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC). This dataset mainly includes seven types of attacks: Heartbleed, Botnet, DoS, DDoS, Web attacks, and Infiltration. Its infrastructure consists of 50 machines and 420 PCs and 30 servers a part of 5 departments in the victim organization. The dataset also comprises of the network traffic and log files from the

victim side machines. Besides, 80 traffic features are extracted using CICFlowMeter-V3 from captured traffic. [25]

CICFlowMeter-V3 is an analyzer and flow generator for network traffic. Bidirectional flows can be generated using CICFlowMeter-V3. This flow determines the forward (source to destination) and backward (destination to source) directions for the first packet and 80 statistical network traffic features can be estimated in the forward and backwards directions separately. Additionally, certain functionalities include adding new features, the selection of features from the existing list of features, and the time limit of the flow timeout can be controlled. [25]

2.2 Papers related to CSE-CIC-IDS2018 framework

Several papers have opted the CSE-CIC-IDS2018 for their research of IDS based on ML techniques.

In this research [38] decision trees, random forest and support vector machine algorithms have been used as they are quick enough and are among the popular classification methods. These three popular and standard machine learning classifiers are trained and applied to the CSE-CIC-IDS2018 dataset. The testing takes place on datasets which are removed from the .pcap logs. It was found that the records can have different values of features when created with similar attack programs but with separate parameters or with some differences in time. Hence, train data results tend to be much better than the results on the test data. To improvise this situation, we normalize the attributes of test dataset values, which improved 10-20 per cent. Decision tree showed the maximum performance, but in this case, it looks like trained models are not applicable in real life. This paper proposes a betterment in the preprocess and normalizing stage of the dataset, testing new algorithms, improving developed models, and dynamically testing on several types of intrusions as future discussions.

In this paper [37] the detection accuracy is assessed. The detection accuracy of determining all states (AS), the prediction of the next state (NS), and the current state (CS) of a sequence is observed using the Baum Welch (BW) and Viterbi Training (VT). These two are the conventional Hidden Markov Model (HMM) algorithms used for training. Uniform, count-based, and random

parameters are used to initialize both BW and VT. Experimental results show that both BW and VT count based initialization methods offer a better performance rate than uniform and random initialization when AS and CS is detected. In comparison, uniform and random initialization methods perform much better than BW and VT count-based procedures in the case of NS prediction.

An alert aggregation method is proposed in this paper [39] to fix the issue of redundant and repetitive alert data in network security devices. Conditional rough entropy and knowledge granularity is used to generate the weights of the attributes. This approach determines different significant weights for different attacks. Similarity value of two alerts can be weighted by using the results of attribute weighting. Finally, the scheme that has been proposed in this paper is applied to two datasets the CSE-CIC-IDS2018 and the DARPA 98 dataset. The experimental results show that the redundant alerts can be significantly reduced, which improves the data efficiency of data processing. It provides precise and correct information for analysis of the next stage.

This study [40] analyses different established datasets such as NSL-KD, ISCXIDS2012, CICIDS2017, and CICIDS2018. Supervised ML is used for analyzing these datasets. The results of this analysis are used to compare the generated results of various datasets. Conclusively, comprehensive results are obtained based on the concept of IDS through supervised machine learning. Main contributions are the classification outcomes which are outstanding on these modern datasets for intrusion detection. Remarkable resilience is observed in the performance of classification even when the data is limited aggressively.

2.3 Data Pre-processing Techniques

The most recent and publicly available cybersecurity dataset is used in this study, which is the CSE-CIC-IDS2018. This dataset provides CSV, PCAP and logs files to transform this raw data into a recognizable form for further processing, known as data preprocessing.
Several papers have performed various preprocessing techniques on the CSE-CIC-IDS2018 dataset, mentioned in this section.
In Paper [32] several operations are enforced during the preprocessing of the dataset, including the conversion of missing values. These missing values are also known as Not a Number (NaN) values converted to 0 to ensure that value errors are avoided with ML systems. The next step is

to set two columns (Flow 'Bytess' and 'Flow Pktss') which contain infinity values to the highest value present in the column + 1. Then the 'Timestamp' column is divided into two new columns as 'Date' and 'Time' to ensure no text values present in the dataset. Two columns Init Fwd Byts and Init Bwd Win Bytes contains -1 as the value in some of the samples. Two more new columns Init 'Fwd Win Byts Neg' and 'Init Bwd Win Byts Neg' is created with values 1 in the same place where -1 values are present in the columns with identical names and 0 is placed at non-negative values. Column' Label' contains attack names identified in the dataset that are converted to numeric values. Finally, the last step increases the number of features from 80 to 83 as the combined dataset is shuffled for randomness.

Several steps are performed in the data processing stage of the dataset CSE-CIC-IDS2018 in [41] such as rescaling, data transformation, removal of invalid value, normalization and feature engineering. Since zero frequencies and massive correlations introduce noise, they are removed, and this process is known as feature engineering.

The preprocessing steps in [42] involve an impute module from the scikit-learn package. This package is used to change the NULL/NaN values to mean or median values present in the dataset based on the nature of values in the same column. The next is the feature selection step where the best features that contribute to an enhanced prediction are determined. In this paper, 10 features were picked from the dataset, and the columns are then separated into class/labels and features. Furthermore, label encoding is used, which makes categorical features and classes more apt for analysis.

An alarming incident which takes place during an import of a dataset is the presence of an infinity value. Runtime crashes occur when string-type data is mixed with numeric data. This is amended by changing string values to NaN values. The label column is binarized in [43] instead of having separate attacks to classify the difference between benign and malicious for an outer defence layer.

The preprocessing phase in [44] consists of a few fundamental operations to generate a ready to use datasets for the training state. The outliers and undesirable traffic are eliminated, and instead, we add additional features to enhance the feature set to ensure the classifiers acquire exceptional detection performance.

Before training the model, preprocessing steps are performed on the dataset in [45]. First, the dataset is prepared by eradicating noise and missing values. The pandas library is then replaced with a data frame. Features that do not contribute to the performance of the model are removed. NaN and infinity values are replaced with mean values. Data is formatted into a standard datatype. Information is chosen randomly to start the experiment. Two methods, up and down sampling, are used to balance and unbalance data.

The data is preprocessed in [35] by removing noisy, missing, and inappropriate data.

Data preprocessing in [46] involves removing source port numbers and source and destination IP addresses. The feature spaces of all the attributes between 0 & 1 were normalized using feature scaling, and the labels with string values were encoded. Removal of the number of rows and columns weren't mentioned.

In [47] for preprocessing the missing, NaN, or infinity values are dropped. The timestamps are transformed into Unix epoch numeric values. After the data cleanup process, approximately 20,000 samples were dropped. Categorical data was taken as protocol features and destination port, whereas the remaining features were treated as numerical data.

2.4 Impact of Dataset Balancing

Though these approaches have made noticeable progress, the imbalance of datasets still prevails and has been addressed as a challenging problem that prohibits intrusion detection systems performance factors. Inequality of classes is formed when malicious samples are lower than that of benign samples. As presented in [32, 33], imbalanced data causes low detection rates of minority classes where the dataset consisted of more instances belonging to the ethical behavior class than the attack class. A drawback of this is presented in [33] discussing performance issues and ML techniques as a mitigation approach. It significantly decreases performance, which results in data loss or overfitting. It intends the trainees to be biased towards the majority class, and all such

classes will be identified correctly. Hence, to solve the issues of low accuracy and reliability of classifiers, balancing datasets is of primary importance, enhancing the identification of sparse classes [34].

In several domain applications, class imbalance distribution is an issue that transpires regularly. A dataset contains imbalanced class distribution when one class which is often the one of interest or minority class is not represented adequately. This can cause issues in determining the algorithm's accuracy as an imbalanced dataset can give incorrect and ambiguous performance results. Another problem is that standard classifiers often assume datasets are balanced when, in reality, the majority of them have imbalance class distribution. Inaccurate results produced using an imbalanced dataset will affect the overall performance of the machine learning algorithms in classifications and is not practically useful [48].

When datasets contain large volumes of data, the high-class imbalance is often observed, making the identification of minority class complicated as imbalance class distribution can introduce a bias towards the majority class. This kind of biased learning can produce a high false accuracy rate and makes it difficult for the learner to effectively differentiate between the majority and minority class in the case of extreme class imbalance [49].

When a dataset is generated, the dataset's objective is to capture normal events and abnormal events. Abnormal events are caused either due to the software's poor performance or due to a malicious attack. We are interested in malicious activities that can be captured in our dataset to find objects that are different from the most common objects present. Normal and abnormal objects are viewed as two distinct classes. A substantial amount of detection systems discover oddity as a dual data partitioning issue where the samples have to be categorized into normal and abnormal.

Since anomalies are comparatively low in ratio to that of the normal traffic, the issue of imbalance is considered to be inbuilt and persists in all the anomaly detection systems [50]. Hence, the significant problem addressed by an anomaly detection system is the issue of imbalance where the majority of the traffic samples point to benign and only a minor point to attack.

2.5 Taxonomy of Balancing Level Techniques

There are various approaches proposed to handle the issue of class imbalance which are present in many datasets. The approaches are categorized as 1) Data Level Approach 2) Algorithm Level Approach 3) Hybrid Level

1) Data Level

Many forms of resampling solutions exist in data-level such as random oversampling the small class, random undersampling the prevalent class, informatively oversampling the small class, informatively undersampling the prevalent, by generating new synthetic data oversampling can be performed on the small class and combinations of all the above techniques. Resampling is the method to solve the class imbalance; the main topic focuses on deciding the optimal class distribution. Random sampling is simple and not sufficient in many cases. In a biclass application, the prevalent class was referred to the majority class, and the small class referred to the minority class. On some parts, random sampling may over duplicate if the problem of class imbalance is shown within class concepts. Informatively undersampling and oversampling indicate that the choice of samples to eradicate is targeted and no new samples are created [50]. The data-level approach is also known as external techniques that rebalances the class distribution by employing a pre-processing step. This is done using either under-sampling or over-sampling to lessen the ratio of imbalance in training data. Under-sampling is the removal of a smaller number of examples from the majority class to decrease the differences between the two classes. Over-sampling replicates examples from minority class [48] [50].

Data-level can be further divided into two subcategories, data-sampling methods and feature selection methods.

Sampling: Data sampling is done randomly or using an algorithmic approach on the given dataset. In over-sampling via replication, the instances from the minority classes are added to the dataset. Replication is done randomly or by applying an intelligent algorithm. Whereas in the under-sampling process, instances from the majority class are removed from the given dataset and are done randomly. Under-sampling and oversampling are more straightforward methods than a complicated process like Synthetic Minority Over Sampling Technique (SMOTE) [49]. SMOTE

is an essential technique that gained popularity in treating class imbalance. A probability distribution is computed by the SMOTE method by adding new samples to the minority class to form a smaller class and extend the boundary to capture the minority class samples [48].

Feature Selection: This method is similar to sampling methods, a pre-processing step that is gaining popularity in class imbalance classification to address the issue of imbalanced class distribution. Feature selection algorithms, in general, can be applied by adopting a filter or wrapper method. Filter method is used to measure the goodness of the feature subset, taking into account the intrinsic features. Wrapping of the feature selection process around the induction algorithm is called the wrapper method. Wrapper methods are expensive than the filter methods since they do not depend on the induction algorithm [48]. Apart from improving the classification performance, feature selection selects the most significant attributes to produce distinct knowledge for inter-class discrimination. It lessens the negative effects of class imbalance on classification performance [49].

2) Algorithm Level

Algorithm level methods directly learn from the imbalanced class distribution in the datasets. This level has dedicated methods such as recognition-based one class learning classifications, ensemble, and cost-sensitive learning methods [48]. The knowledge of the corresponding classifier learning algorithm and the application domain is needed to develop an algorithmic solution. Also, one needs to know the reason for the failure of learning algorithm when the class distribution of the data is uneven [50]. Various subcategories of algorithm level are mentioned below:

Improved Algorithm: It is a leading approach that has the capability to manage the issue of imbalance in the classification of datasets by directly learning from the imbalance class distribution. Before extracting significant data, these algorithms learn about the imbalance distribution of the classes. This is done to establish a model based upon the target objective [48].

One class learning: This learning is also known as recognition based methods that model the classifier on depicting the minority class. This method uses a neural network which tries to recognize different patterns by only learning from the examples of minority class rather than the examples of majority and minority class. The threshold is the key point in this approach since a

strict threshold separates the minority class, and a lenient one will cover the majority class [48]. In this learning, the system is formed only using the examples of the target class with the absence of offset classes. Attempts to create boundaries that encompass the target concept [50].

Cost-sensitive learning: Different misclassification types have different varying costs which are considered by the cost-sensitive algorithm. Cost matrix is taken into consideration during the creation of a low-cost model [50]. The cost-sensitive algorithm's idea is that the high cost is enforced on a classifier when misclassification occurs. That is the occurrence of misclassification is when a high cost is designated by the classifier to a false negative than a false positive, which emphasizes the misclassification of a positive class [48].

Ensemble Learning: Another option for class imbalance problem is Ensemble Learning that trains various classifiers on training data. Their evaluations are consolidated to generate the final decision of the classification. In general ensemble models can be called as boosting or bagging. The main idea for combining classifiers in repeating ensembles is to enhance the ability of generalization [50] [48].

3) Hybrid Approach

Apart from cost-sensitive learning, one-class learning, and ensemble learning, a new breed is called the hybrid approach. This approach has been created to handle the issue of imbalance in the given datasets. It implies more than a single ML algorithm to improve the quality of classification through hybridization with various learning algorithms to attain better results. It uses multiple algorithms or two or more individual methods to address the issue of imbalance [48] [49].

2.6 Review of Contentions in Existing Taxonomies

Perhaps the most comprehensive account of existing techniques and an indication of a taxonomy is provided in [36] [50] and [48]. The authors in [49] review the approaches which span over the last 8 years. In contrast, [50] and [48] do not specify the year span of their reviews.

Table 1 derives the taxonomy based on approaches in [48-50]. Two essential classifications emerge from the studies in [48-50]: techniques classed as data level; techniques classed as algorithm level. These studies collectively converge on the definition of data-level methods to include data sampling and feature selection approaches, while algorithm level methods include cost-sensitive and hybrid/ensemble approaches.

In [48] and [49], the authors define data-level methods to include data sampling and feature selection approaches, while algorithm level methods are defined to include cost-sensitive, and hybrid/ensemble approaches. Across all three surveys shown in Table 1, several divergent accounts of algorithm level classifications have been proposed, creating numerous discrepancies. In [50], the significant deviation is in the algorithm-level definition. In contrast to [48] and [49], subcategories of algorithm-level are not defined in [50]. The subcategory of one class, however, is mentioned in [50] under the discussion of algorithm-level but not distinctly classified as in [49] and [48].

Table 1: Contentions in Existing Taxonomies

Reference	Title of Paper	Publication Date	Citations	Data Level			Algorithm Level						Cost Sensitive Learning	Boosting Approaches
				Data Level	Feature Selection	Data Sampling	Algorithm Level	Cost Sensitive Learning	Ensemble	Hybrid	One Class	Improved		
[49]	A survey on addressing high-class imbalance in big data	2018	55	☑	☑	☑	☑	☑	☑	☑	☐	☐	☐	☐
[50]	Classification of Imbalanced Data: A Review	2009	788	☑	☐	☑	☑	☐	☐	☐	☑	☐	☑	☑
[48]	Classification with class imbalance problem: A review	2015	157	☑	☑	☑	☑	☑	☑	☑	☑	☑	☐	☐

The derived taxonomies in [48] and [49], which are more recent, show that feature selection is a subcategory of data level, whereas [50] does not. We cautiously suggest that this may be because the feature selection approach gained popularity in the study of dataset bias after the publication on [50]. Both [48] and [49] discuss techniques in feature selection, such as principal component analysis (PCA) and the likes since the publication of [50]. The specifics of the feature reduction techniques and its development over the years are beyond this paper's scope. In contrast to [48]

which addresses the concept of Improved learning, [49] and [50] do not discuss this as a subcategory or part of the taxonomies provided.

In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone [51]. In contrast to [48] which defines ensemble as boosting (an iterative technique which adjusts the weight of an observation based on the last classification), [49] defines ensemble as both bagging (a way to decrease the variance in the prediction by generating additional data for training from the dataset using combinations with repetitions to produce multi-sets of the original data) and boosting. The definition of ensemble provided in [49] aligns better with the definition provided in [51].

Another significant deviation observed in [49] and [48] from [50] is the inclusion of the ensemble under algorithm level in [49] and [48]. In [50] ensemble, cost-sensitive and other boosting are included as subcategories of boosting. This is not shown in Table 1 due to the lack of space. However, the definition of ensemble provided in [50] agrees with the description provided in [51].

2.7 Literature of Balancing CSE-CIC-IDS2018

A large and growing body of literature has investigated ML methods applied to cybersecurity datasets. In this study, we found that 75 articles were published between 2017 and 2020 on ML, using the CSE-CIC IDS dataset. In [12], the authors detail the features and attacks which constitute their CSE-CIC IDS 2017 dataset. The main goal of this paper is to generate dynamic, diverse, & extensive benchmark datasets. These datasets are generated to inspect, test, & assess intrusion detection systems. The CSE-CIC IDS 2018 dataset depicts a summary of actions and behaviours observed on the network. Diverse sets of datasets are generated by combining the profiles with different sets of features in which a portion of evaluation metrics is covered. The CSE-CIC IDS 2018 dataset is organized for each day where the network traffic and the logs of the events form a raw data per machine. The raw data then undergoes a feature extraction process, and a CICFlowMeter-V3 is used to extract 80 traffic features and more which are saved per device as CSV files.

In [52], an IDS is proposed, implemented in the computer networks to detect malicious events. The CICIDS2017 dataset is used in which the most important and influential features are identified. The aim is to know the effects of these features present in the dataset. Through which the threats can be avoided that are created by big data. The authors apply (DMLP) structure which uses recursive feature elimination performed using random forest (RFE-RF) and reports on the accuracy metric. In this work, the authors do not indicate any treatment of the imbalance of the dataset. This paper aims to reduce the dataset and increase the speed without decreasing the accuracy to achieve the detection rate.

An approach towards addressing dataset imbalance was provided in [53] where a network IDS, which is highly important and based on deep learning method is proposed. This study aims to solve the issue of imbalance and provide an IDS of high performance using unsupervised learning models, autoencoder (AE) and generative adversarial networks (GAN) during deep learning. Based on the GAN model, the rare classes are oversampled to resolve the issue of performance deterioration that is provoked by data imbalance. After which the characteristics of data are processed to a lower level by using the autoencoder model. Hence, a data level technique is applied in which the minority malicious traffic is oversampled using Conditional GAN (CGAN), which is a better approach than the original GAN. Since the distribution of data and feature to learn are considered as the labels of each class.

A similar technique in [54] also applies a data level technique in which downsampling of the majority benign traffic is used to mitigate the class imbalance. In [55], an algorithm level approach is proposed as a solution to mitigating the imbalance. In this work the metrics reported include the accuracy, precision, F-Measure, Detection Rate (DR), Attack Detection Rate (ADR), False Alarm Rate (FAR), Model Building Time (MBT). The CSE-CIC IDS 2018 dataset is thoroughly covered in [32] similar to [12] in which the authors detail the improvement over the 2017 dataset. Later in 2018, [40] is one of the many published works that apply supervised ML classifiers and reports on the well-known metrics related to the ML model's performance. In 2019, [32] is one of the few papers found that applied the data level technique to the CSE-CIC IDS 2018 dataset. In the same year, in [29], an algorithm level technique was used.

A combination of data level techniques and algorithm level techniques were reported in [56] and [57]. Such a variety of techniques was first reported in [22] as a hybrid technique. However, we note that the techniques in [56] and [57] are applied to the CSE-CIC IDS 2017 dataset.

Chapter 3

3 Literature Analysis of Balancing Techniques CSE-CIC IDS 2018

3.1 Overview of CSE-CIC IDS 2018 Dataset

CSE-CIC IDS 2018 is a popular dataset with many published works explored in the field of IDS. CSE-CIC IDS 2018 is developed using the AWS platform (Amazon Web Services). This dataset provides numerous attack profiles that can be used in the field of intelligent security and applied to network topologies and protocols in a generic approach [40]. This dataset was enhanced in consideration with the standards of CSE-CIC IDS2017. CSE-CIC IDS2018 is a public dataset currently used that has 2 profiles classified and consists of 5 different attack methods. Various data scenarios were collected, and the raw was edited daily. 80 statistical properties such as packet length, number of packets, number of bytes etc. were calculated in forward and reverse direction separately while creating data. Finally, the dataset was published over the internet to all the researchers out there. The dataset is published in two formats CSV and PCAP with approximately 5 million records. CSV format is mainly used in the field of AI, and PCAP format is used for extracting new features [9].

3.2 Imbalanced Class Distribution

Figure 1 highlights the extent of the imbalance observed in the CSE-CIC IDS 2018 dataset in [48]. There are different types of malicious events such as BOT, DoS, Brute Force, Infiltration, SQL injection. Heartbleed is not included in Figure 1 as the number of samples is minimal. The most typical trend observed when evaluating the distribution of samples per class in datasets is exponential. This trend indicates that the percentage of benign samples is higher than malicious samples. A dataset best serves in machine learning algorithms when the distribution is normal or close to normal.

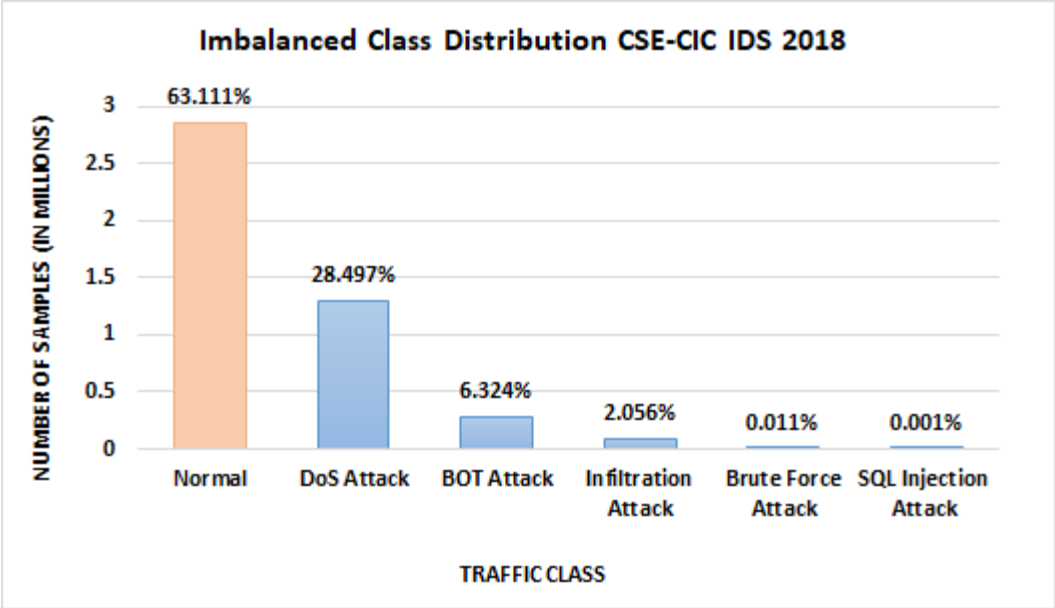

Figure 1: Imbalanced Class Distribution CSE-CIC IDS 2018

The issue of class imbalance distribution is dominant across all domains [58]. In this paper, two metrics - precision and recall - are studied and referenced for issues that relate to class imbalance. A dataset contains imbalanced class distribution when one class - which is often the one of interest - is a minority class or not represented adequately. A number of publications have been reviewed to determine the impact of dataset imbalance. By far the most widely accepted account can be found in [58] which states that issues arising from this are poor accuracy in classification and a general bias in the results obtained. Classifier algorithms used in machine learning as mentioned in [25] require a balanced dataset. In spite of these findings, there are recent developments in algorithms that account for the imbalance in a dataset. Such approaches are generalized as algorithm level approaches towards the mitigation of bias [48].

In this paper, we determine the CSE-CIC IDS 2018 dataset of having a benign to malicious imbalance ratio of 1.71 to 1. Table 2 represents the imbalance taking into account all the attack types included in the CSE-CIC IDS 2018 dataset in [48]. The benign traffic in this dataset is found to be 1.71 times more than the total malicious traffic.

Table 2: Class Distribution for CSE-CIC IDS 2018

Class Label	Count
Benign	2,856,035
Bot	286,191
Brute Force	513
DoS	1,289,544
Infiltration	93,063
SQL Injection	53
Total	**4,525,399**

Table 3: Imbalance Ratio Distribution for CSE-CIC IDS 2018

Normal Traffic Count	AttackTraffic Type	Count	Normal to Attack Ratio
2856035 **(63.11%)**	BoT	286191 **(6.324%)**	10 ***to*** 1
	Brute Force	513 **(0.011%)**	5567 ***to*** 1
	DoS	1289544 **(28.49%)**	2 ***to*** 1
	Infiltration	93063 **(2.056%)**	31 ***to*** 1
	SQL Injection	53 **(0.001%)**	53887 ***to*** 1

Table 3 provides an approximation of the ratios of benign to malicious traffic in the various minority classes provided in Table 2 from [9]. For the most populous minority class, DoS, a ratio of more than 2 to 1 is observed. The next minority class, BOT has a ratio of approximately ten to one. Infiltration has a ratio of 31 to 1 followed by Brute Force with a ratio of 5567 to 1. SQL Injection has a severely imbalanced ratio of 53887 to 1. Hence, a ML algorithm trained on this dataset will tend to bias towards classifying a SQL Injection event as a benign event, a false negative. The general classification of malicious will have a ratio of 1.71 to 1 which is an improvement but still a distance from a much required balanced dataset. Cross validation techniques may be used to mitigate the effect but will not be very accurate in the case of minority

classes as discussed in [12]. Hence, from the above table it can be inferred that the attack traffic will yield a high rate of False Negatives and a low rate of False Positives.

3.3 Criteria for CSE-CIC IDS 2018 Paper Selection

Our approach to deriving a new taxonomy is based on the study of work published in the cybersecurity domain anchored on the CSE-CIC IDS dataset. A criteria for selecting published work for this dataset, related to AI based intrusion detection systems was developed. The criteria used for selecting papers related to IDS and dataset balancing is specified in this section as follows:

Criteria for selecting the papers related to IDS were as follows:

- Publications were only included if they were relevant to the BoT-IoT dataset
- Publications were only included if certain keywords and phrases related to dataset balancing were found. These included but not limited to: imbalance; minority class(es); majority class(es), sampling, downsampling, upsampling, balance(ing).
- Publications were only included if they were published between 2019 and 2020 to align with the first public announcement of the dataset.
- The number of citations and venue of publication was considered for each work assessed.
- Publications were only included if they approached imbalance and cited the other non-IDS related work on dataset balancing.

To come up with this methodology, previous approaches published in [59] and [60] were reviewed and analyzed. The two papers were compared, in [59] more advanced techniques were proposed due to the recent advancements provided by google scholar platform reported in this paper.

From [59], the google scholar as a platform provides access to key information about the citation of a paper. The name of the primary dataset paper is first entered in the google scholar search engine. The number of times the paper has been cited is displayed and clicking on it leads to the total list of cited papers. The papers are filtered down by clicking the checkbox **"Search within cited articles"**, the keyword search and custom range process used for the datasets are explained below:

The keyword **"imbalance"** was used for the CSE-CIC IDS dataset and a total of 75 papers were generated. A custom range option is also available in google scholar to select the key papers. The range applied for our research is 2018-2020.

3.4 Analysis of Published Work

A systematic review of the literature in the cohort of published works from Section III allowed us to divide the work into groups according to the level of contribution each work makes towards balancing of a dataset. This grouping has been done in a hierarchical order as shown in Figure 4 with the first level determining whether the paper has a contribution or not. The second level identifies the extent of the contribution in terms of a proposed method or the application of an existing method. In the case of non-contributing work, the second level identifies whether imbalance has been recognized and mentioned or not.

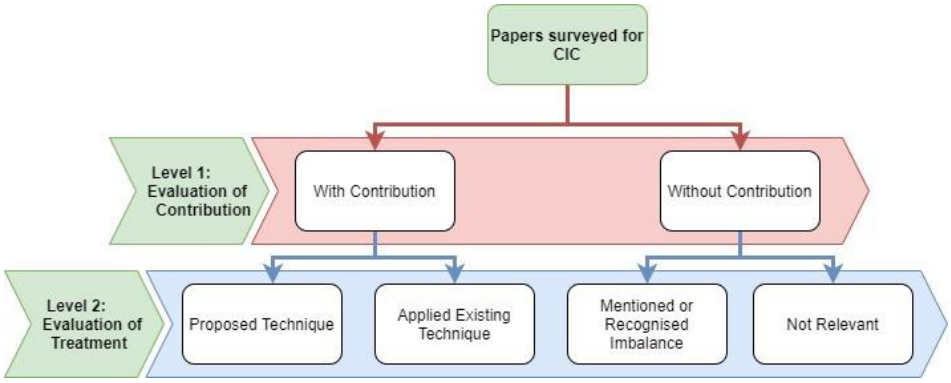

Figure 2: Hierarchical grouping of published work

A more thorough definition of the grouping has been provided below:

- **Proposed:** The authors have proposed a technique to solve imbalance.
- **Applied Existing:** The authors have applied existing techniques in solving imbalance.
- **With Contribution:** This is a cumulative of papers in which the authors have either proposed or applied existing techniques.
- **Mentioned:** These are the papers in which the authors have mentioned an imbalance technique with respect to our analysis but have not treated it.
- **Not Relevant:** The authors have mentioned imbalance in general and not with respect to our analysis of dataset imbalance.
- **Without Contribution:** The authors of these papers have either mentioned imbalance or did not have any discussion relevant to the imbalance of the datasets.

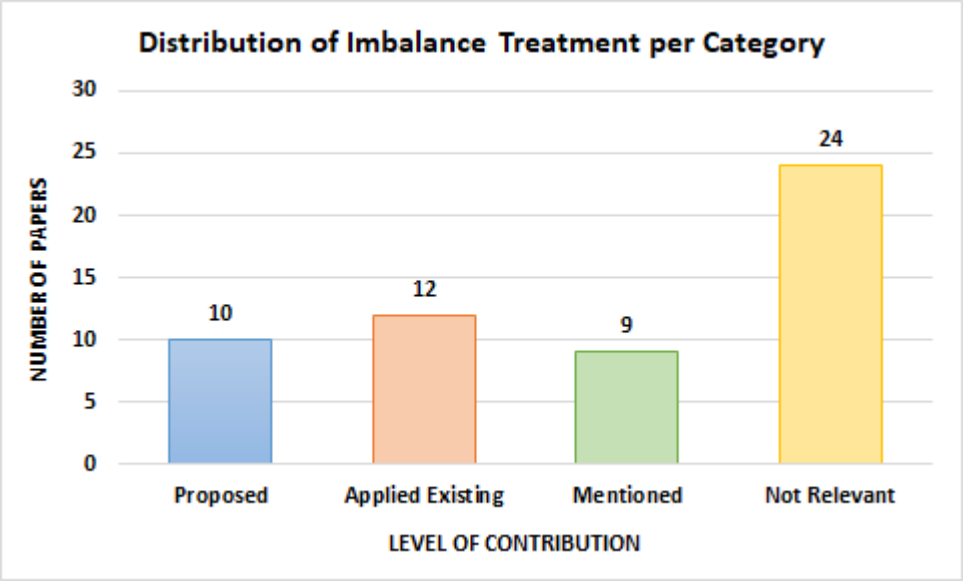

Figure 3: Comparison of the imbalance treatment categories across all three datasets

The categories: with contribution and without contribution have been added for the ease of presenting the analysis. Papers that have proposed a new technique or applied an existing technique to deal with the imbalance are labelled as With Contribution. Furthermore, papers that have no relevance to dataset imbalance or have reservedly mentioned the word imbalance are collectively labelled as Without Contribution.

Figure 3 shows the distribution of published work across the four groups defined for CSE-CIC IDS dataset. Figure 4 presents a cumulative percentage distribution across the four groups irrespective of the datasets used. It has been observed that only 40% of the papers have either proposed or applied techniques to treat dataset imbalance and the remaining 60% of the papers have not contributed to this study. These results further support the idea that there is a lack of attention to imbalance because 44% of the papers are not relevant which takes precedence over other groups.

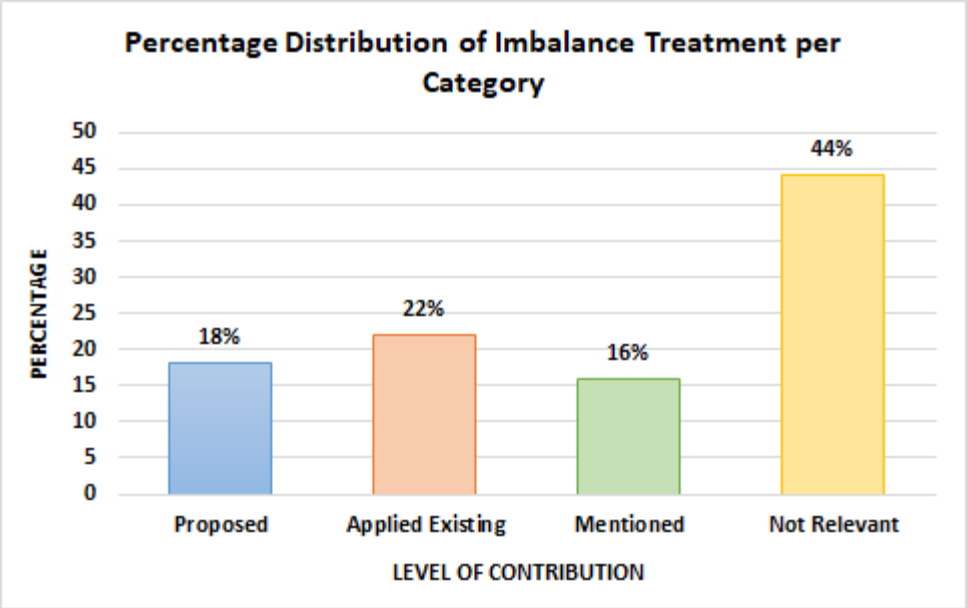

Figure 4: Categorization of Imbalance Treatment- A comparison of all papers surveyed, across all categories of imbalance treatment.

The "without contribution" category shown in Figure 3 takes precedence with the highest count. In the "with contribution" category the applied existing takes precedence as shown in Figure 5.

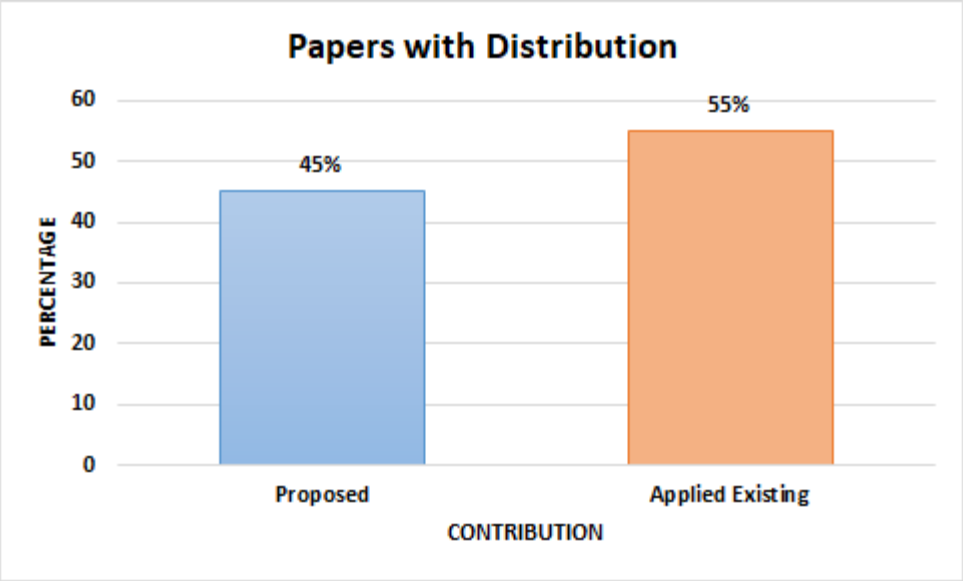

Figure 5: Comparison of papers in which proposed techniques are compared with existing techniques that have been applied.

3.5 Level Distribution of CSE-CIC IDS

A study of the proposed and applied existing papers was undertaken to determine the technique which has been used. In contrast to the findings in the literature review, the outcome of this particular study demonstrates that there are three distinct classifications of techniques for balancing of datasets as shown in Figure 6. The grouping or classification of these techniques are defined as levels to be consistent with published literature presented in Section II.

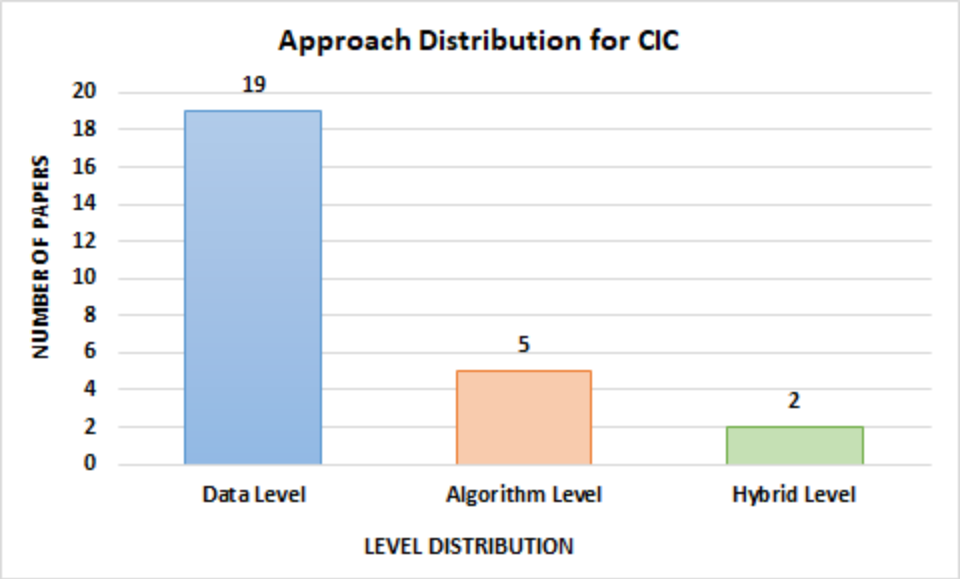

Figure 6: Comparing the number of papers that propose, apply or mention approaches across all three datasets

The statistical representation in Figure 6 spans the 26 papers published in the CSE-CIC IDS. This dataset shows that the majority of papers use data level techniques to solve the issue of imbalance.

3.6 Ranking of undertaken approaches

The percentage distribution for ***proposed*** and ***applied existing*** from the total amount of papers ***with contribution*** is depicted in Figure 5. Papers with contribution had 45% of new methods proposed and 55% of existing methods applied. This result may be explained by the fact that a majority of the papers have not focused on proposing a new technique to overcome bias. This discrepancy can be a dominant focus area for future research directions.

Chapter 4

4 Implementation

4.1 Setup

Jupyter Notebook was used to experiment on a Lenovo Laptop with a Windows 10 64 bit operating system. The processor was an Intel Core i5 with processing 1.60 GHz-1.80 GHz unit and RAM of 8.00 GB. The server used for implementation is mentioned below:

- HP ProLiant DL380p Gen8
- CPU: Dual Intel(R) Xeon(R) CPU E5-2660 v2 @ 2.20GHz (40 cores total)
- Memory: 256 GB ECC (1866 MT/s)
- Storage: 9TB
- Operating System: Linux (Fedora release 32)

Pandas and NumPy library was used to load the dataset, manipulate data, and perform preprocessing. Scikit learn was used for training and evaluating the model, data analysis, and evaluation metrics. Matplotlib and Seaborn library was used for data visualization.
There are several stages in the implementation phase. Pre-processing of data for supervised classification is the first stage where the input of the dataset description takes place along with data cleaning. Data cleaning includes removing duplicate records, resolving the class imbalance problem, undersampling, and data standardisation. Sample preparation is done in the stage of data pre-processing for unsupervised anomaly detection. Classifier algorithms are used as a part of supervised learning which is used for the training process. Then the results obtained through this training method is validated. Manual feature drop functions are used, which is of two types, and the results are compared [64].

4.2 Preprocessing Steps of CSE-CIC IDS 2018

The following are the main changes done on CSE-CIC IDS 2018 during preprocessing.

1) **Removal of Attributes:** Some of the CSV files contain the attributes FLOW ID, Src IP, Src Port, Dst IP. These attributes are not available in the other files CSV files and hence have been removed. The FLOW ID, Src IP, and Dst IP are non-numeric data types. Therefore, they can't be used directly for machine learning, but the loss of src Port may or may not affect the model.
2) **Removal of Rows Containing NaN values:** Some of the values are filled with NaN or infinity values. Since the number of occurrences of these is low, we have removed the rows containing these values.
3) **Removal of Rows Containing Attribute Names:** Some files contain rows that are filled with attributes names instead of values. These rows have also been removed in the preprocessed files.
4) **Adding New Label:** Since the model, we are working on is based on binary classification, we added a new label to the preprocessed files and the existing label. The new label contains the binary classification of the packets as "BENIGN" or "ATTACK".
5) **Dropping Unnecessary Column:** The column label is dropped as the newly created label 'new_label' will be used.
6) **Dropping Columns with 0 Values:** Columns containing zero as a value are dropped since they do not impact the dataset.
7) **Transforming Column To Boolean Values:** The 'new_label' column is converted into 0 & 1 where the attack samples are assigned as 1, and benign samples are assigned as 0.

The preprocessing steps used on the CSE-CIC IDS 2018 dataset reduces the features from 80 to 67 features, and this preprocessed data is used to create the training and testing subsets.

4.3 Machine Learning Classifiers

Firstly, the expected behaviour of the network should be determined for the anomaly-based IDS. The system needs to be trained by a learning algorithm to accomplish this. There are many machine learning algorithms available and presented by the literature. Among which there are three types of classification based Supervised ML techniques which are used in our study: 1) KNN, 2) Random Forest (RF) and 3) Logistic Regression (LR)

4.3.1 K-Nearest Neighbor

Supervised learning algorithm comprises K Nearest Neighbor (KNN) Algorithm. This particular algorithm does not involve a training stage, unlike other supervised learning algorithms. Data from an original sample class is used for the implementation of KNN. A data nearest to the new data is chosen, which is the k data and is decided to which sample class it is added. This algorithm data is categorized based on a similar distance measure, and this is a simple and easy to implement ML algorithm. A Manhattan or Euclidean type is the distance in this case. The k value is an integer value, and the same class is assigned to the nearest distance data point. For simple and noisy data, this, the method is highly resistant. This algorithm also has a disadvantage of acquiring a high memory as all the cases are stored in distance calculations [32].

4.3.2 Random Forest

Random Forest (RF) is also a part of supervised machine learning architecture. This algorithm is mainly used for regression and classification problems. This algorithm's usage is effortless, and a decision forest is created by using decision trees, and problem-solving is performed this way. For which this particular algorithm makes a random collection of trees. More than a single decision tree is trained during the process to generate the classification accurately. RF produces good results most of the times, even without the use of a hyperparameter. This is the highly preferred algorithm due to its accurate and speedy results for noisy, incomplete, and even mixed data [32].

4.3.3 Logistic Regression

Logistic regression (LR) is different from linear regression in terms of values where LR predictions are of district values, and linear regression values are continuous values. For binary classifications, where dataset values are assigned to 0 or 1, LR is best suited. 1 denotes the default class, and in terms of whether an event will occur or not, there are only two capabilities. It would be an occurrence of an event which is denoted as 1 nor 0, which indicates that the event does not occur. The output in LR takes the probabilities of mostly the default class. Since being a probability, the result is generated in the range of 0-1 [61].

4.4 Validation Metrics of Machine Learning

4.4.1 Confusion matrix

Confusion matrix provides a one-stop solution to monitor model performance on the imbalanced dataset. It provides a guide to calculate real positive (TP), true negative (TN), false positive (FP) and false-negative (FN) for each class. Confusion Matrix is a performance measurement for machine learning classification. The confusion matrix in Table 4, comprises of four items for binary classifiers:

- **True Positives (TP):** when the classifier identifies the true positive label as positive
- **True Negatives (TN):** when the classifier identifies the true negative label as negative
- **False Positives (FP):** when the classifier identifies the true negative label as positive
- **False Negatives (FN):** when the classifier identifies the true positive label as negative

In the context of cybersecurity research, a well-known understanding is that a positive event is defined as a malicious event, and the correct classification of such an event is deemed as a true positive outcome. A negative event is a benign event, and the correct classification is deemed as true negative. Inaccurate classification can mean that a benign event is classified as a malicious event. This misclassification is deemed as a false positive. Likewise, a malicious event to be classified as a benign event is deemed a false negative [62].

Table 4. Confusion Matrix

Confusion Matrix		**Classification**	
		Positive	**Negative**
Target	**Positive**	TP	FN
	Negative	FP	TN

4.4.2 Performance Evaluation

In this section, we evaluate the performance of different IDS datasets [63]. The evaluation metrics presented below are defined on the True Positive (TP), True Negative (TN), False Positive (FN), False Negative (FN) values as shown in Table 4. The metrics used for this purpose are as follows:

Accuracy: It is a metric that generates the number of correct predictions over the total number of predictions made by the model. Accuracy is best suited for a balanced dataset and biased for an imbalanced dataset.

$$\textbf{Accuracy} = \frac{\textbf{TP+TN}}{\textbf{TP+TN+FP+FN}}$$

Precision: It is a metric that generates correct optimistic predictions which are divided by the total number of positive predicted values. A low precision value indicates a high number of False Positives.

$$\textbf{Precision} = \frac{\textbf{T P}}{\textbf{TP + F P}}$$

Recall/Sensitivity/TPR: This metric is generated by the True Positives' ratio to the sum of True Positive and False Negative predictions. In this case, a low recall value indicates a high number of False Negatives.

$$\textbf{Recall} = \frac{\textbf{T P}}{\textbf{TP + F N}}$$

F1 score or F-Measure: It is the harmonic mean value generated between recall and precision measures. It is best suited for an unevenly distributed dataset

$$\textbf{F1 − Score} = \frac{\textbf{2 × (Recall × Precision)}}{\textbf{(Recall + Precision)}}$$

Chapter 5

5 Experiment and Analysis

Three experiments are performed on the dataset

5.1 Experiment 1: Full Dataset Modeling Using Train and Test Split

This study's first experiment is to train the model using the train, and test split function from sci-kit learn library on the CSE-CIC-IDS2018. For this firstly the entire dataset is loaded on the jupyter notebook platform. After loading the whole dataset, which approximately contains around 15 million records. The classifier accuracy results for the entire dataset is provided in Table 5 below:

Table 5: Classifier Accuracy using Scikit Learn to Train and Test Split

Attack Type	Classifier Type	Classifier Score
CSE-CIC-IDS2018	KNN	0.987
	RF	0.987
	LR	0.986

As noticed in Table 5, the results produced show the model attaining around 98% accuracy for all the classifiers, namely, KNN, RF, and LR even if the dataset is highly imbalanced. This experiment's output gives an idea about the subsequent investigation to find the threshold value (breakpoint). The model attains the highest performance by using small sets of samples.

5.2 Experiment 2: Threshold Value Analysis for Botnet and Infiltration Attack

As shown in the above experiment, the model attains the highest classifier indicating high performance using a large number of the dataset. This experiment is performed on a small subset of the attack samples from the whole dataset. Two attacks Botnet and Infiltration attacks are extracted alone from the entire dataset. This experiment aims to find the threshold value or limit at which the model attains the highest performance.

This experiment is divided into two parts. In the first part, the benign samples are kept constant, and attack samples are varied in a series for ten iterations across three classifiers (KNN, RF, and LR) and two attacks, Botnet and Infiltration.

Various iterations were tried out using the trial and error method to observe the first breakpoint value or the first highest classifier score known as the Threshold Value. The trial and error method involved several iterations by increasing and decreasing the benign and attack samples through which the breakpoints were obtained.

5.2.3 Part 1: Fixing Benign Samples

Botnet Attack

KNN Classifier

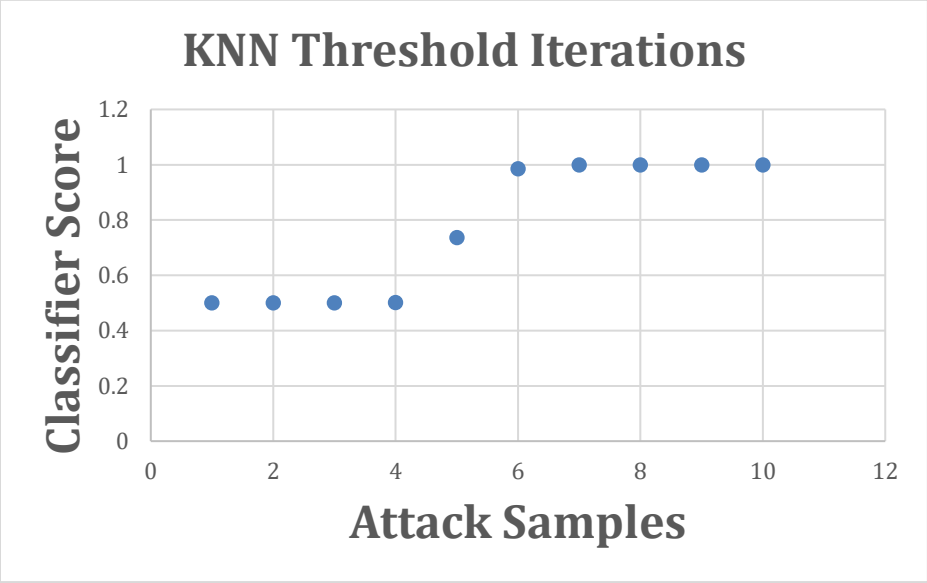

Figure 7: Threshold value analysis for KNN

The threshold value analysis of KNN classifier for Botnet attack is shown in Figure 7. Botnet attack attains a breakpoint value of 100% at 8000 benign samples and 7 attack samples. In this classifier; the classifier scores increase as the attack samples increase, which is a positive behaviour. Nominal growth is indicated by the jump in values from 50% to 100% in 10 iterations.

RF Classifier

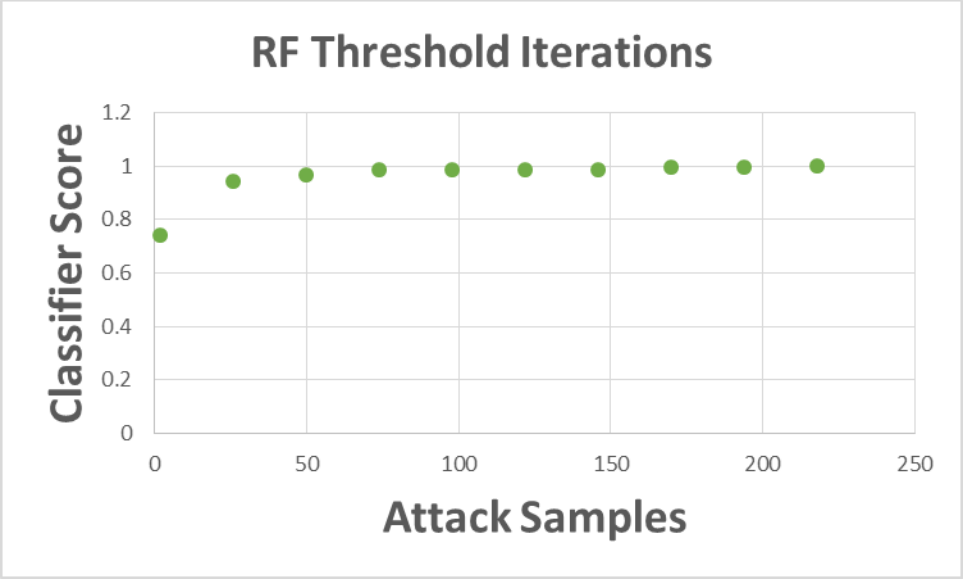

Figure 8: Threshold value analysis for KNN

Figure 8 provides an insight into the RF classifier score. In this case, the breakpoint is not observed at the same point as in KNN. Hence, the attack sample iteration values are changed using the trial and error method to identify the first threshold value. Here the iterations are taken with a difference of 24 and a breakpoint of 100% is attained at the last iteration with 8000 benign samples and 218 attack samples. The sudden growth of 20% is recognized from the first iteration to the second iteration in this case. Compared to KNN in RF, the attack samples need to be increased to achieve a breakpoint.

LR Classifier

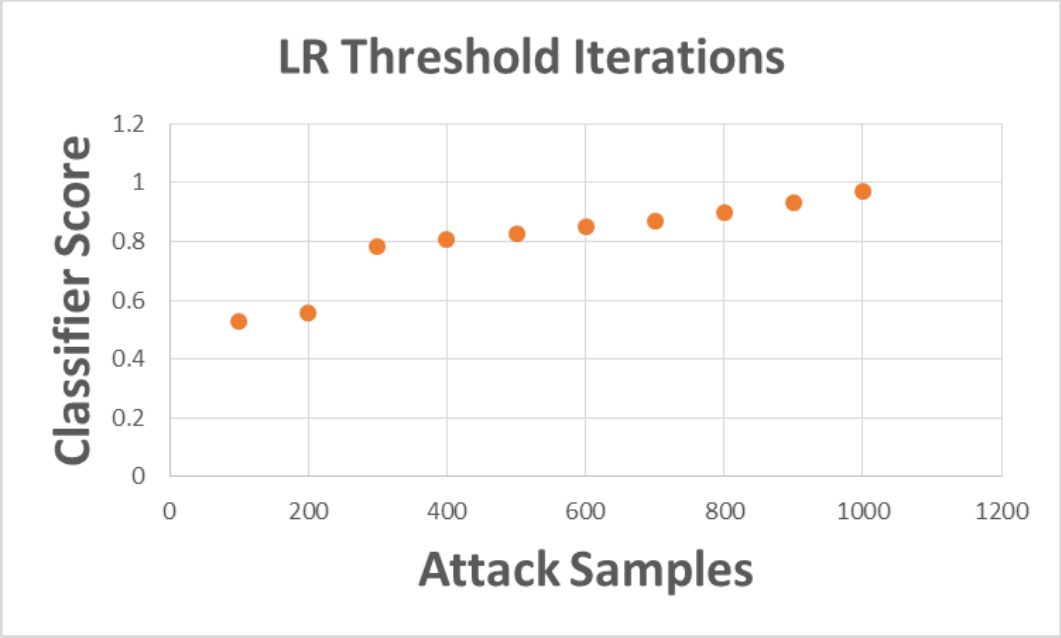

Figure 9: Threshold value analysis for KNN

The breakpoint analysis for LR is depicted in Figure 9, where the iteration values are taken with a difference of 100. In this case, a breakpoint of 97% is attained using 1000 attack samples. A gradual increase is observed in the classifier scores, from 52% to 97% for ten iterations. Compared to KNN and RF, in LR, the breakpoint values are achieved with a large number of attack samples.

Hence, the breakpoints attained by KNN and RF are 100% whereas, in the case of LR, it's a lower percentage value of 97%. In all the three cases, the benign samples are kept constant, and the attack sample has been changed accordingly to attain a threshold value.

Infiltration Attack

KNN, RF, and LR

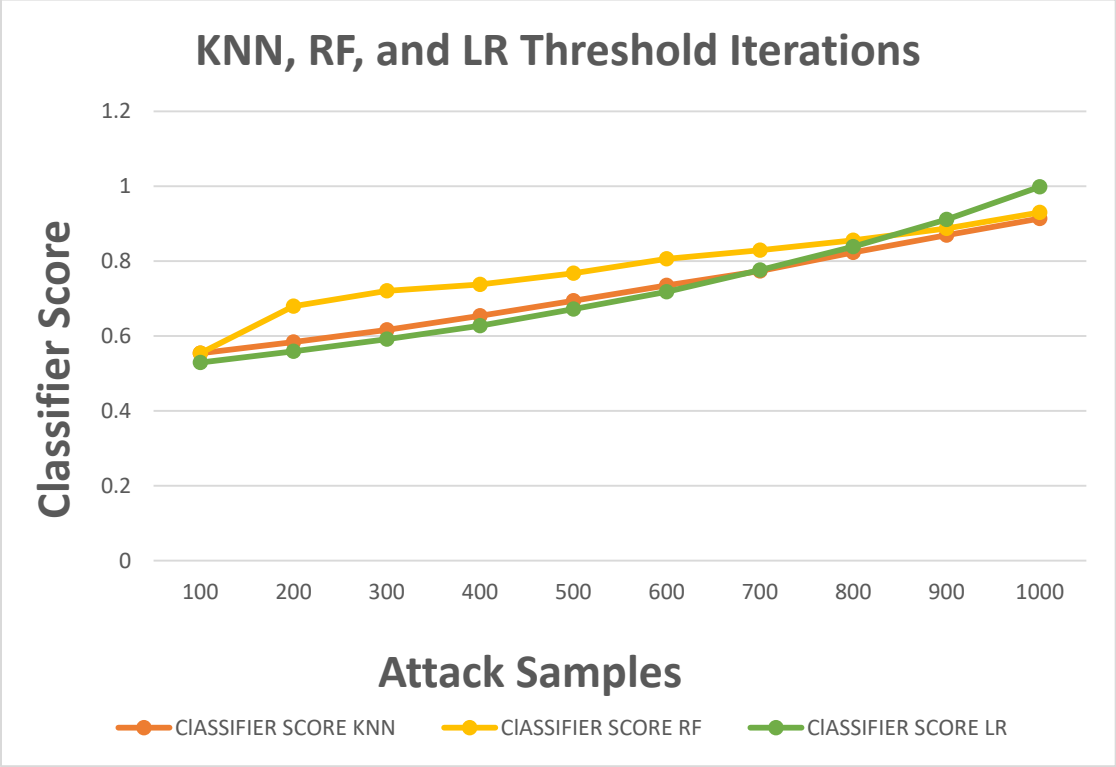

Figure 10: Threshold value analysis for KNN, RF, and LR

The above Figure 10 represents the classifier scores for all the three classifiers with the same benign and attack samples. In contrast to the Botnet attack, this particular attack attains a threshold value using the same iteration values (Attack Samples) for all the three classifiers. Benign samples remain fixed, as mentioned earlier, which is 9000 in this case. Also, Infiltration attack achieves threshold values at 1000 attack samples which is the last iteration value for all the three classifiers. In all the three classifiers there is a gradual increase from approximately 50% to 90%. The breakpoint values for each classifier are KNN – 91%, RF- 92%, and LR – 99

To conclude with the first experiment, it is noted that Botnet and Infiltration attacks attain breakpoints with a diverse range of iteration values or attack samples. There is a discrepancy in the range of iterations (attacks values) within the Botnet attack, but the same attack values are maintained in Infiltration. Hence, when both the attacks are compared, there is a different range of benign and attacks samples. In both the attacks approximately all the classifiers had a gradual increase in the classifier score, which is a positive behaviour.

5.2.4 Part 2: Fixing Attack Samples

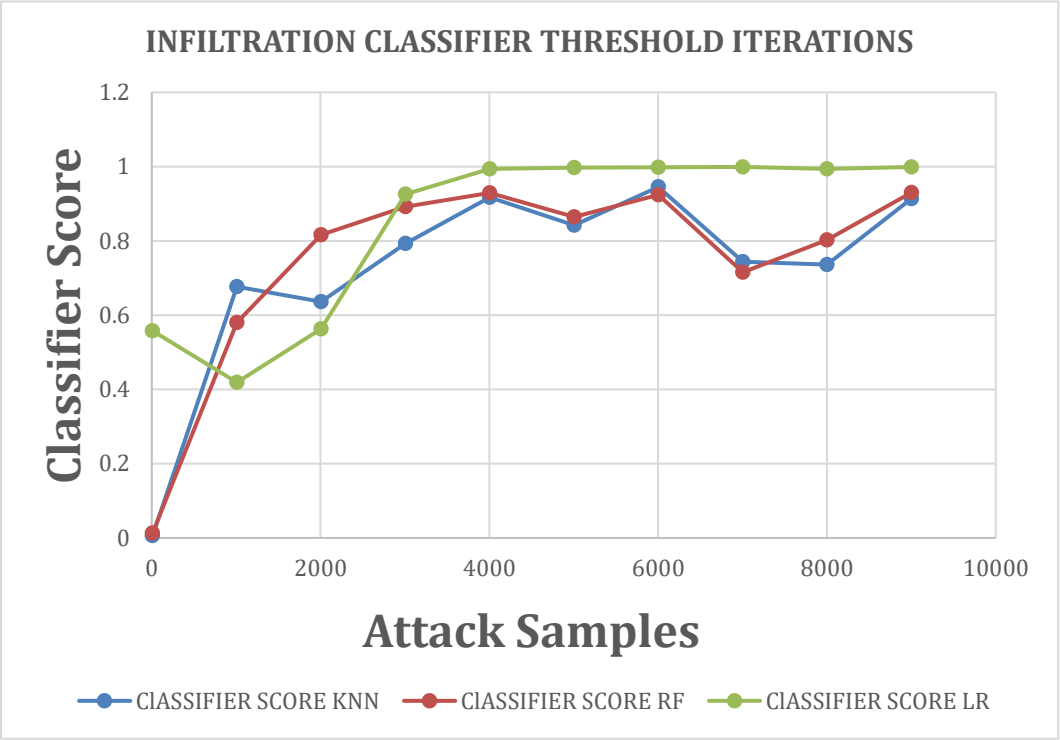

Figure 11: Infiltration Classifier Threshold Iterations

Figure 11 represents the second half of the experiment, where the first part's threshold value is used as the first iteration. The number of attack samples is fixed, and the benign samples are reduced in a series of ten iterations for KNN, RF, and LR. This experiment is performed to determine the effectiveness of the breakpoint sample analysis. It is observed that the classifier score decreases as the attacks samples are reduced, which is a positive behaviour for all the three classifiers.

5.3 Experiment 3: Performance Metric Analysis for Iteration Cycles

In the previous experiment using the classifier accuracy, the threshold value is obtained. In this experiment, the performance metric values of experiment 2 are analyzed, and the threshold values are crosschecked. The performance metrics which are analyzed in this experiment are accuracy, recall, precision, and F1-score. The metrics values for Botnet and infiltration attack are provided in Table 6 and Table 7 for part 1 iteration values of the previous experiment.

5.3.1 Part 1: Performance Values for fixed Benign Samples

Botnet Attack

Table 6: Performance Metric Values for Botnet attack

Iterations fore Botnet Attack	KNN				RF				LR			
	Accuracy	Precision	Recall	F1 score	Accuracy	Precision	Recall	F1 score	Accuracy	Precision	Recall	F1 score
Iteration 1	0.5	0.25	0.5	0.33	0.74	0.83	0.74	0.72	0.53	0.26	0.5	0.34
Iteration 2	0.5	0.25	0.5	0.37	0.95	0.95	0.94	0.94	0.56	0.44	0.5	0.35
Iteration 3	0.5	0.25	0.5	0.37	0.97	0.97	0.96	0.97	0.78	0.86	0.74	0.74
Iteration 4	0.5	0.25	0.5	0.37	0.99	0.99	0.98	0.99	0.81	0.87	0.74	0.76
Iteration 5	0.74	0.83	0.74	0.72	0.99	0.99	0.95	0.99	0.83	0.88	0.74	0.76
Iteration 6	0.99	0.98	0.98	0.99	0.99	0.99	0.98	0.99	0.85	0.89	0.74	0.77
Iteration 7	1	1	1	1	0.99	0.99	0.99	0.99	0.87	0.88	0.73	0.78
Iteration 8	1	1	1	1	1	1	0.99	1	0.9	0.86	0.73	0.77
Iteration 9	1	1	1	1	1	1	1	1	0.93	0.81	0.73	0.76
Iteration 10	1	1	1	1	1	1	1	1	0.97	0.5	0.48	0.49

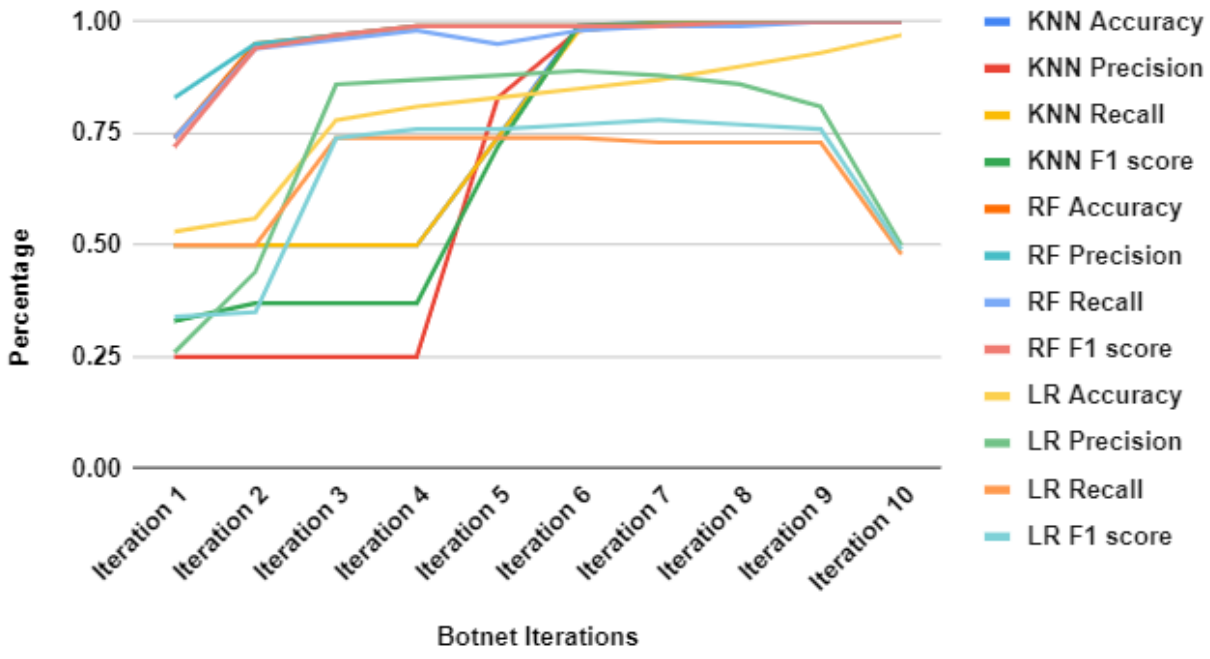

Figure 12: Comparison of performance metrics for KNN, RF and LR - Botnet

In this section, the performance metric values such as accuracy, precision, recall, and F1 score for each Botnet attack classifier have been presented. The trends can be observed from the graph. As seen in the above table and figure, it is noticed that the breakpoints attained for each classifier is same as that observed in experiment 1.

Infiltration Attack

Table 7: Performance Metric Values for Infiltration attack

Iterations for Infiltration Attack	KNN				RF				LR			
	Accuracy	Precision	Recall	F1 score	Accuracy	Precision	Recall	F1 score	Accuracy	Precision	Recall	F1 score
Iteration 1	0.55	0.68	0.53	0.42	0.59	0.77	0.56	0.48	0.53	0.76	0.5	0.35
Iteration 2	0.58	0.77	0.54	0.44	0.68	0.81	0.64	0.61	0.56	0.72	0.5	0.37
Iteration 3	0.62	0.67	0.54	0.47	0.72	0.82	0.66	0.65	0.59	0.73	0.5	0.38
Iteration 4	0.65	0.68	0.55	0.5	0.74	0.83	0.65	0.65	0.63	0.73	0.5	0.39
Iteration 5	0.69	0.69	0.55	0.53	0.77	0.82	0.66	0.67	0.67	0.78	0.51	0.42
Iteration 6	0.74	0.68	0.57	0.56	0.81	0.83	0.68	0.7	0.72	0.79	0.51	0.43
Iteration 7	0.77	0.66	0.57	0.58	0.83	0.8	0.67	0.7	0.78	0.84	0.51	0.46
Iteration 8	0.82	0.65	0.6	0.61	0.86	0.74	0.67	0.7	0.84	0.86	0.52	0.49
Iteration 9	0.87	0.6	0.6	0.6	0.89	0.65	0.64	0.65	0.91	0.83	0.51	0.5
Iteration 10	0.91	0.5	0.45	0.48	0.93	0.5	0,46	0.48	1	0.5	0.5	0.5

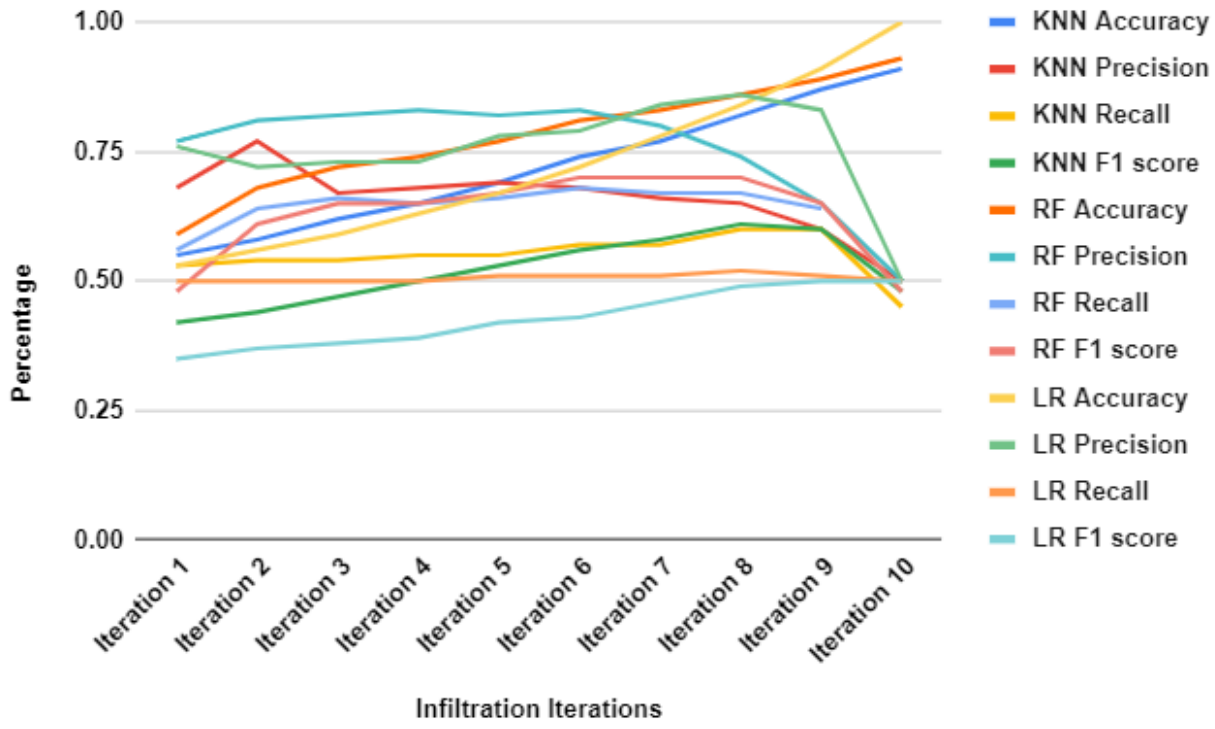

Figure 13: Comparison of performance metrics for KNN, RF and LR - Infiltration

In this section, the performance metric values such as accuracy, precision, recall, and F1 score for each Infiltration attack classifier have been presented. The trends can be observed from the graph. As seen in the above table and figure, it is noticed that the breakpoints attained for each classifier is same as that observed in experiment 1.

5.4 Experiment 4: Manual Feature Drop Selection

This experiment determined how each of the dataset's 67 preprocessed features affects the ML model's performance. Feature drop selection is performed in two ways, independent and group drop. Reference point used for both the feature drop selections are obtained by using the threshold values from experiment 2.

5.4.1 Independent Feature Drop

Botnet

KNN, RF, and LR

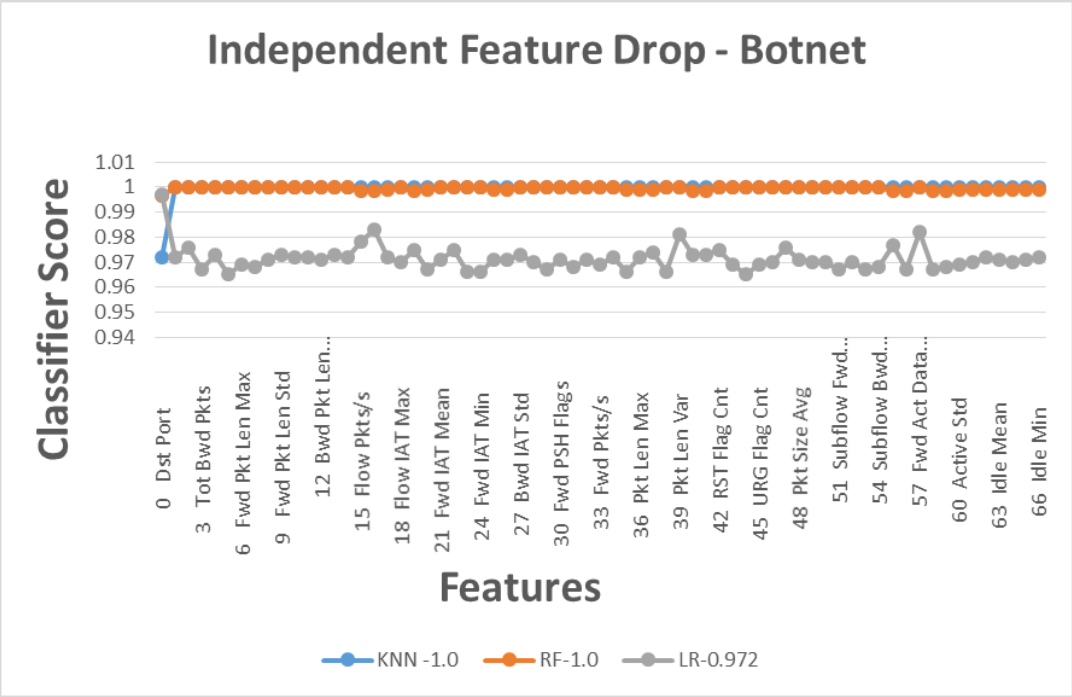

Figure 14: Independent feature drop - Botnet

Figure 14 presents the independent feature drop trend for all the three classifiers, namely KNN, RF, and LR. In this experiment, we take the breakpoint value obtained from the first experiment and compare the decrease in value with the feature drop values. After preprocessing the dataset, there are 67 features, and there is only a 3% drop from the threshold value for the first feature Dst Port. The remaining features attain the same value as the breakpoint value, which is 100% for KNN. In RF, there is a drop in value of about 1% for 24 different features. The remaining 43 features have the same threshold value of 100%. In LR, all the features generate different values which either increase or decrease by 1% or 2%.

Infiltration

KNN, RF, and LR

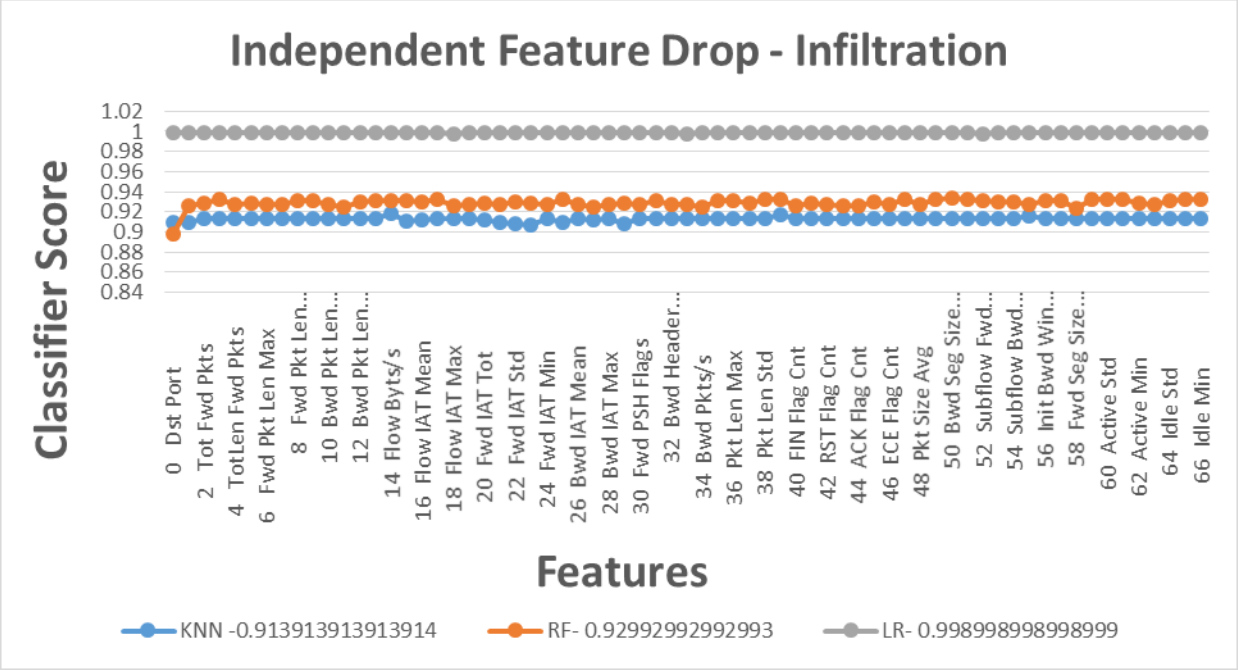

Figure 15: Independent feature drop - Infiltration

As seen in Figure 15, the independent feature drop representation of Infiltration is presented for all the three classifiers. In the case of KNN, there is a maximum drop of 1% from the threshold value for 7 features, and the remaining 60 features have the same value as the breakpoint value. Whereas in RF, a significant reduction was noticed for the first feature, from 92% to 89%, which is about 3%. The remaining 66 features experience a very slight reduction or 1% increase in values. LR attains the same threshold value for all the independent feature drops.

5.4.2 Group Feature Drop

Botnet

KNN, RF, and LR

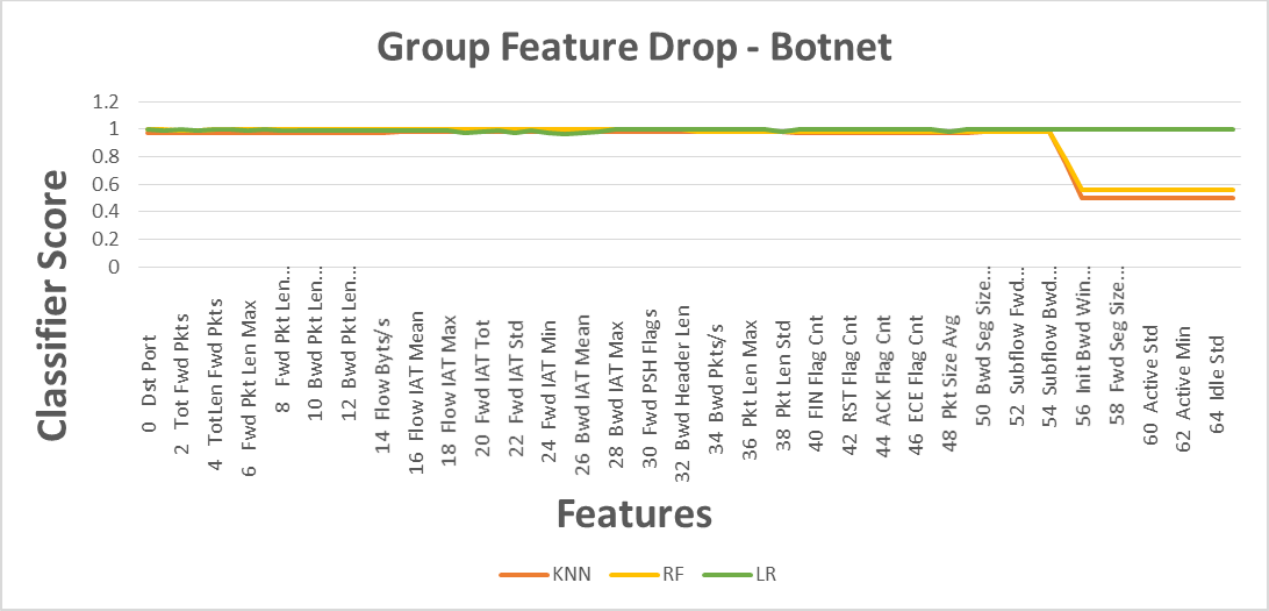

Figure 16: Group feature drop - Botnet

Feature drop values represented in Figure 16 is for all the three classifiers. In this case, a gradual drop of 3% is noticed from the threshold value. A significant drop from 100% to 75% is observed at the 55th feature. **'Init Fwd Win Byts'** shows the first drop which is about 25% and the following 10 features 'Init **Bwd Win Byts', 'Fwd Act Data Pkts', 'Fwd Seg Size Min', 'Active Mean', 'Active Std', 'Active Max', 'Active Min', 'Idle Mean', 'Idle Std', and 'Idle Max'** have a decrease of 50% from the threshold value.

Similarly, in RF, a gradual drop of 1%-2% is observed until the 55th feature. A considerable reduction of 28% is analyzed at the 55th feature **'Init Fwd Win Byts'**. The remaining features **'Init Bwd Win Byts', 'Fwd Act Data Pkts', 'Fwd Seg Size Min', 'Active Mean', 'Active Std', 'Active Max', 'Active Min', I'dle Mean', I'dle Std', and 'Idle Max'** show a standard drop of 44%.

In LR, there is a small increase of 1%-2% from the threshold value for 37 features, and 4 features remain similar to the threshold value. At feature 26 there is a decrease of 1% from the threshold value of 97% to 96%. The remaining 24 features give 100% results, which is an increase of 3% from the threshold value.

Infiltration

KNN, RF, and LR

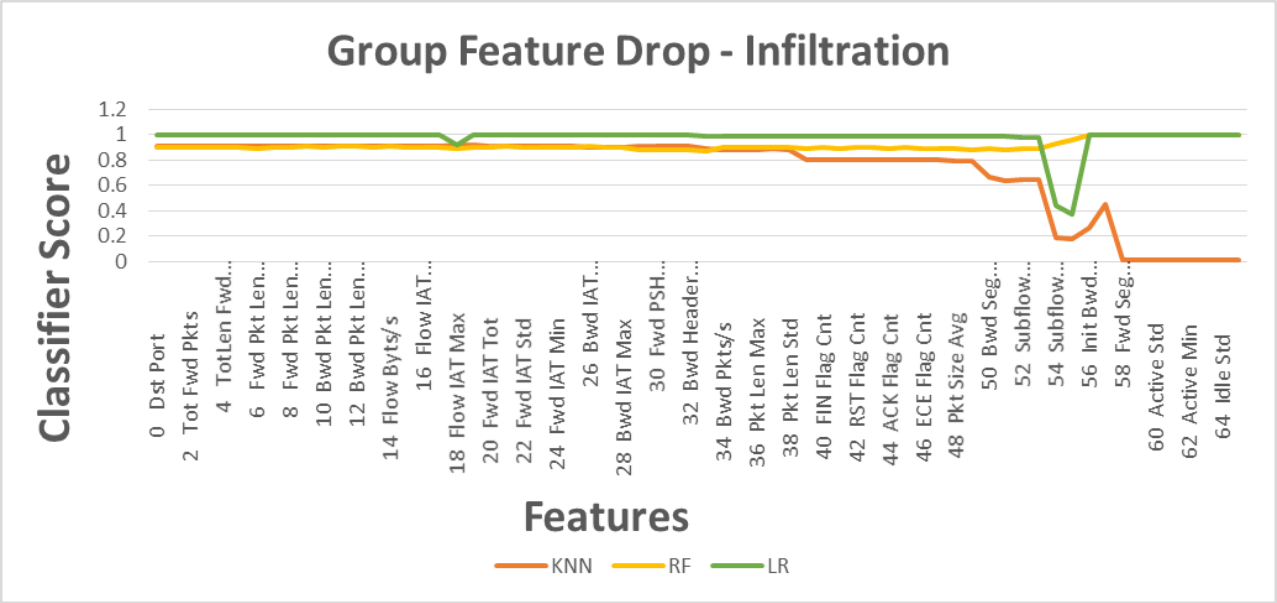

Figure 17: Group feature drop - Infiltration

In the above Figure 17 for KNN, there is a 1% drop from the threshold value at the 1st feature until the 14th feature. Again, this drop is noticed at the 18th feature and from feature 26 to 29 and also at feature 33. Values similar to the threshold value (91%) are noticed for 12 features. A gradual drop of 2% from the threshold value is noticed at feature 34 **'Fwd Pkts/s'**. From feature 35 to 54 there is a gradual decrease up to 27% from the threshold value. At feature 55, a sudden jump is noticed from 64% to 18% and reduces up to 1% from 91% (threshold value). A huge decrease of 90% is experienced in group feature drop for KNN.

In RF, there is a fluctuating decrease of about 5% until the 55th feature. At feature 55, the threshold value is seen, and from the 96th feature, there is an increase in the value which goes up to 100%.

In the case of LR, 32 features have gained the threshold value (99%). An unexpected drop of 8% is noticed at the feature at the 19th feature. A gradual decrease of 1%-2% is seen from feature 34 to 54. Massive reduction of 56% and 62% is noticed at the 55th feature **'Subflow Bwd Byts'** and 56th feature **'Init Fwd Win Byts'** from the threshold value. After which an increase of 1% from the threshold value is seen, which gives a 100% result.

5.4.3 Comparison between Independent and Group Feature Drop

Botnet

KNN Combined Graphs

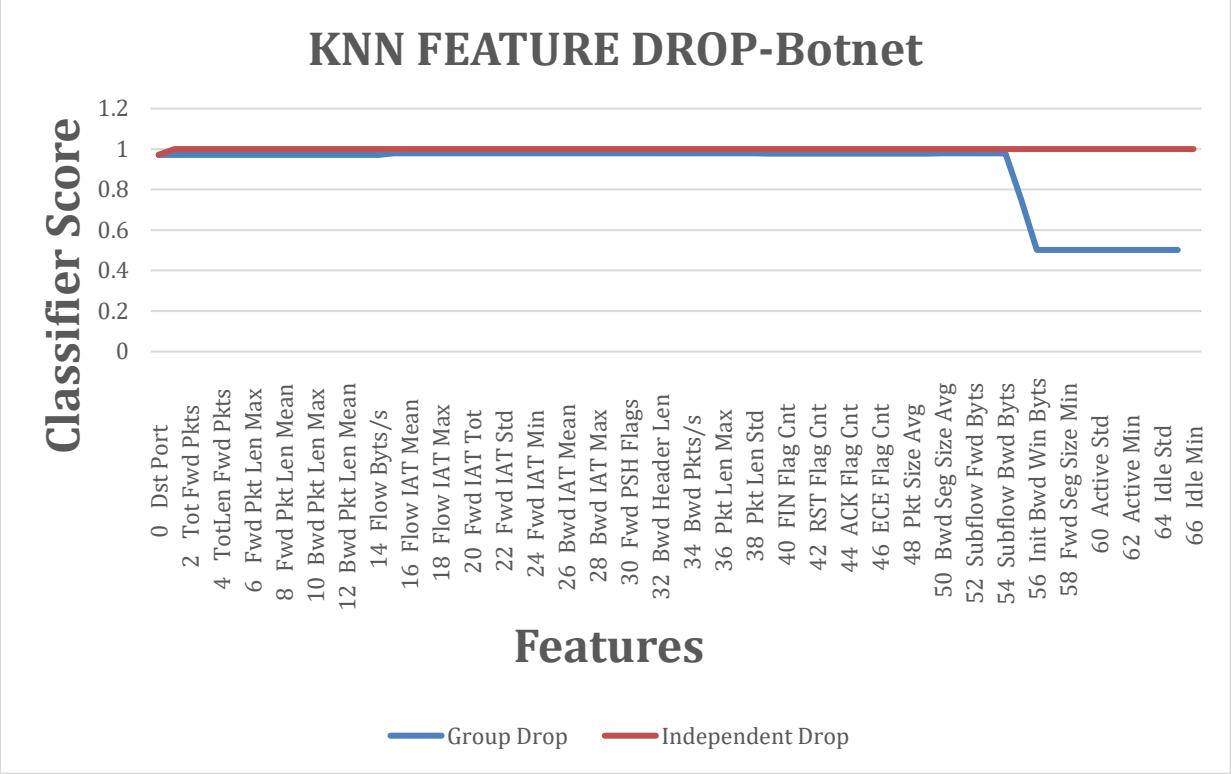

Figure 18: Manual Feature Drop Comparison for KNN-Botnet

The above figure for independent drop shows no decrease in values from the threshold value (100%) except at the first feature, which experiences a decline of 3%. This 3% drop at the first feature experiences in both the independent and feature drop. Whereas in group drop there is a small, steady drop of 3% from the threshold value. This drop then increases at the 56th feature and falls up to 50% until the 66th feature.

RF Combined Graphs

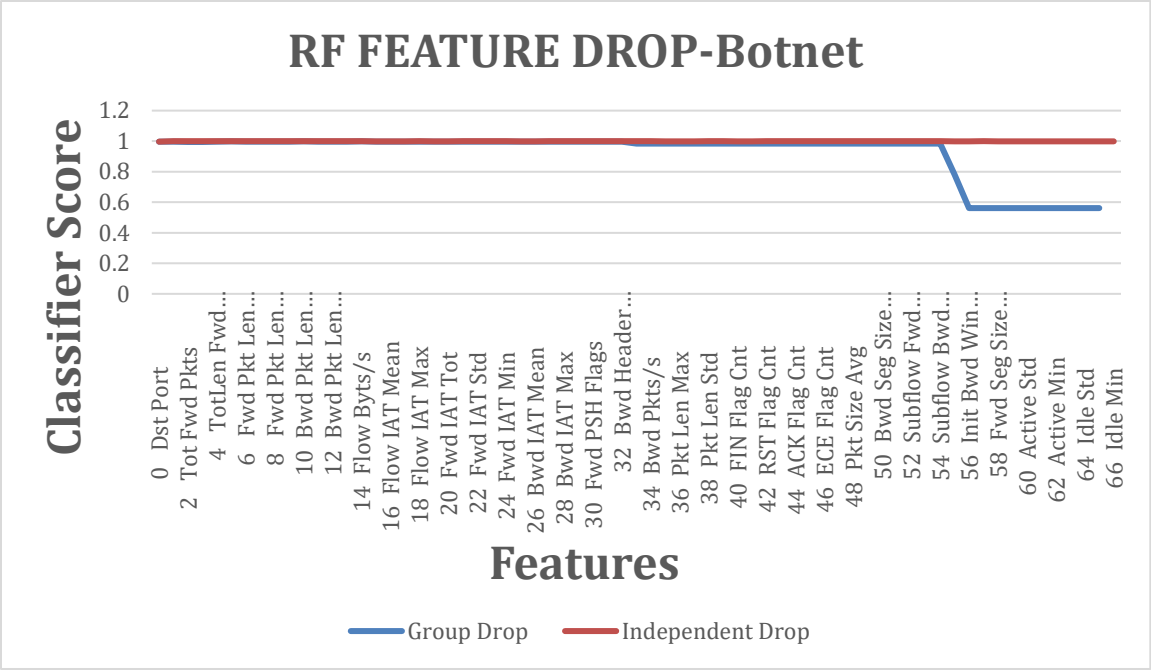

Figure 19: Manual feature drop comparison for RF – Botnet

Figure 19 shows the manual feature drop comparison for RF classifier. The first feature shows a similar reduction of 1% in both the drops from the threshold value (100%). In the case of an independent drop, there is a fall of 1%, and the rest of the features maintain the same threshold value. Whereas in group drop there is a fluctuation in the reduction between 2% - 3%. A sudden significant decrease of 22% is observed at the 56th feature, which extends up to 44% drop from the threshold value.

LR Combined Graphs

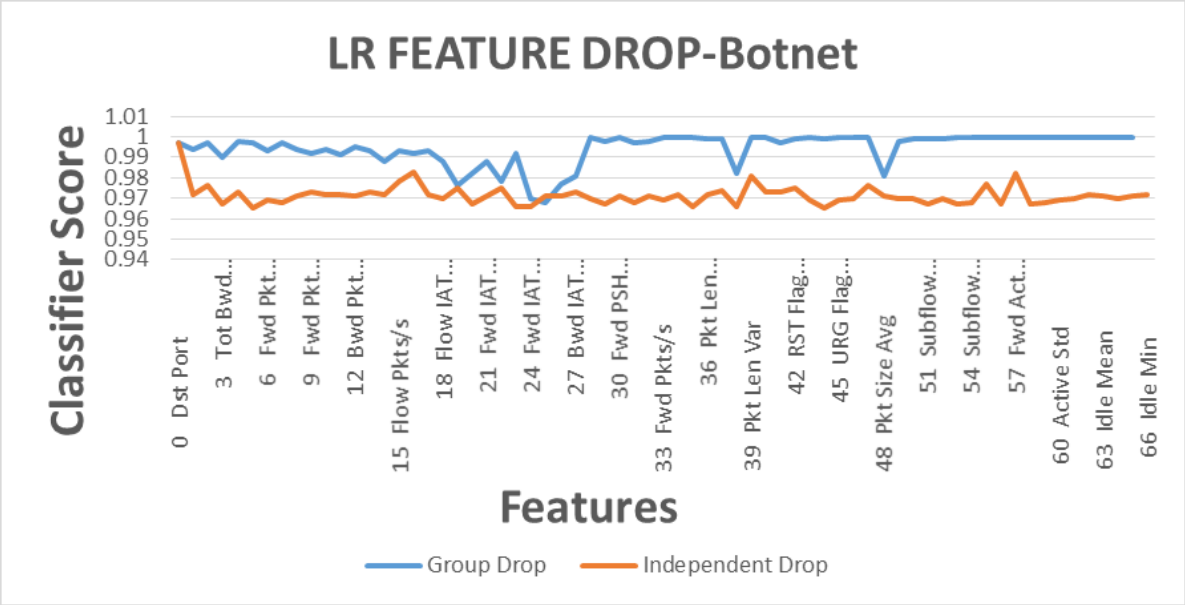

Figure 20: Manual feature drop comparison for LR – Botnet

Like the other two classifiers, LR also shows a similarity in value, but the value increases by 2% for the first feature drop from the threshold value (97%). For the first feature, no fall is noticed, and only an increase in value is observed. Some features attain threshold value in the independent drop, and some features experience an increase in the value of about 1% - 2% from the threshold value. Whereas in group drop, some features experience an increase in value up to 3% (up to 100%), and some features attain the same value as the threshold value. There is also a decrease in the value of 1% for 1 feature in group drop, but approximately 22 features attain a 1% drop in the independent drop,

INFILTRATION

KNN Combined Graphs

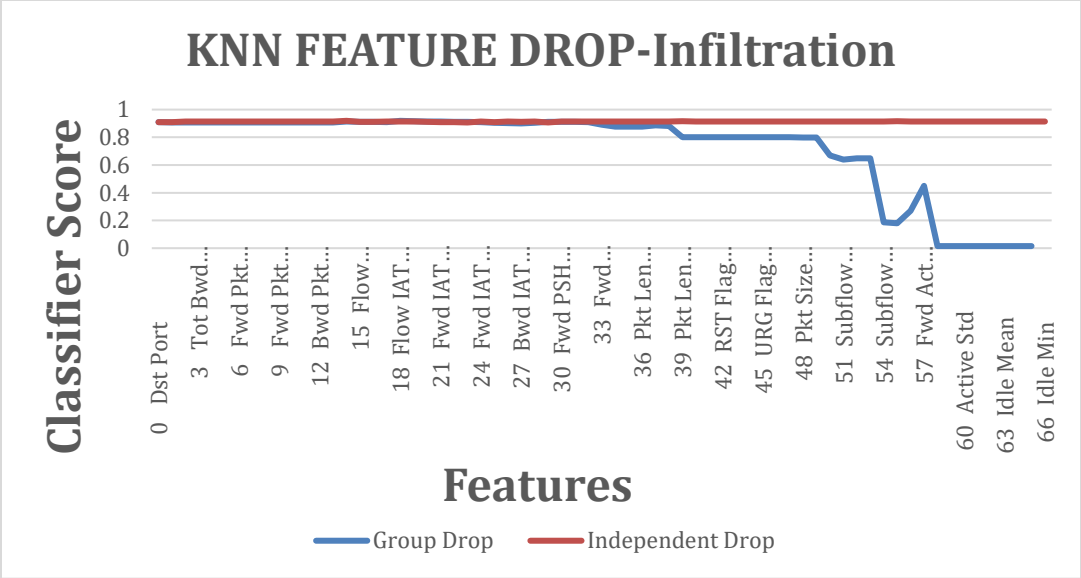

Figure 21: Manual feature drop comparison for KNN – Infiltration

In this case, it is observed that the first feature in both independent and group drop has a drop in value of about 1%from the threshold value (91%). The feature drop value for all the features fluctuates between a 1% drop and threshold value in independent drop. Whereas in the group a gradual decrease of up to 27% is noticed until the 54th feature. Then a sudden massive reduction of 73% is observed at 55th feature which extends up to 90% drop from the threshold value.

RF Combined Graphs

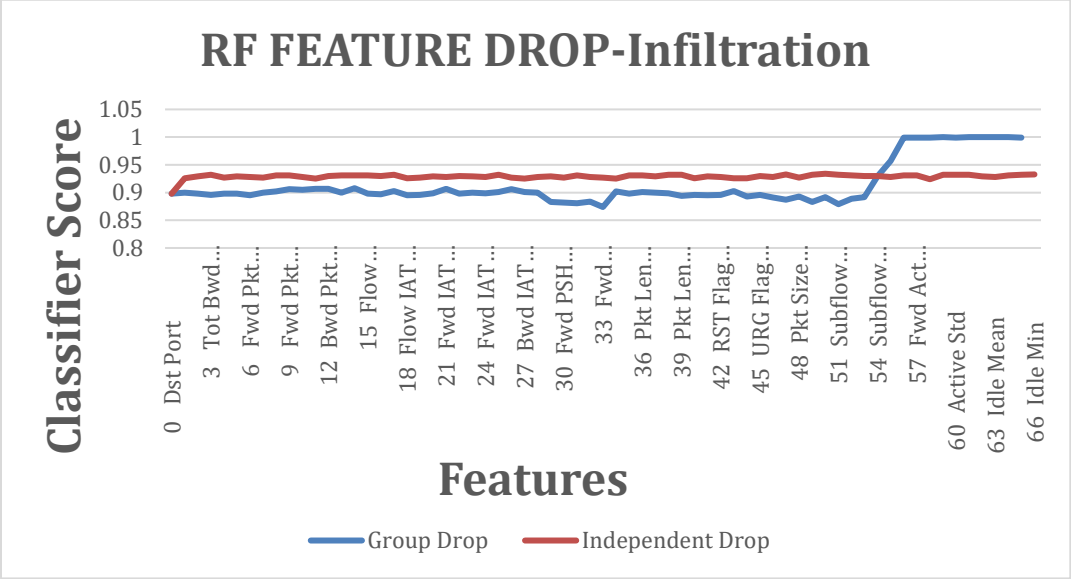

Figure 22: Manual feature drop comparison for RF – Infiltration

There is a drop of 2% from the threshold value (92%) in RF classifier at the first feature in both independent and group drop. In independent drop, most features experience a 1% increase from the threshold value or maint values similar to threshold value. Whereas in group drop, there is a drop upto 5% for some of the features and few maintain the threshold value. Also, some feature shows an increase up to 8% from the threshold value.

LR Combined Graphs

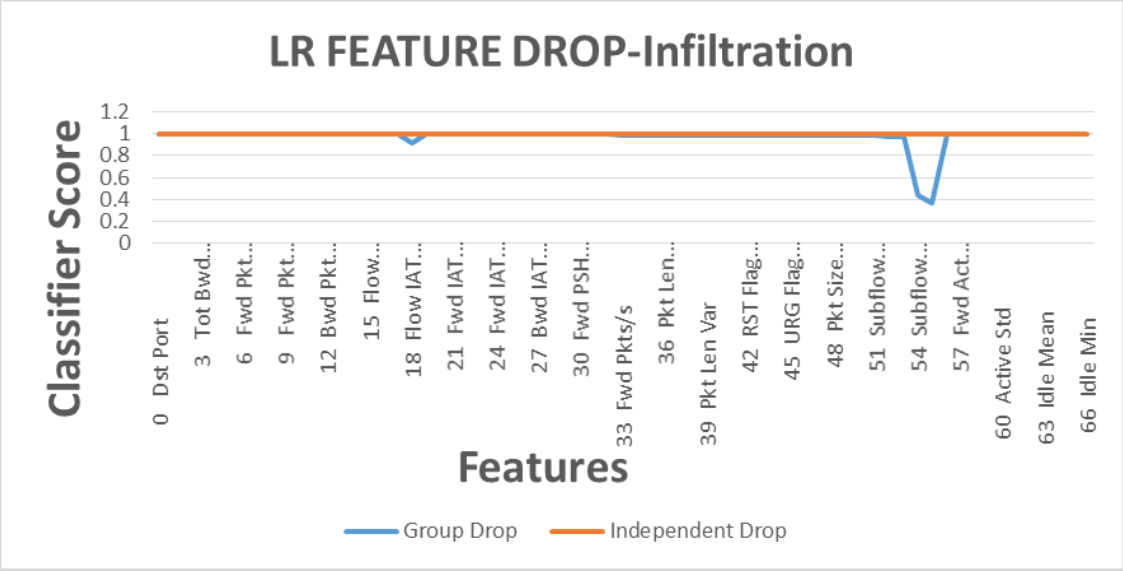

Figure 22: Manual feature drop comparison for LR - Infiltration

In LR, as observed in Figure 22, there is no decrease from the threshold value (99%) at the first feature in both the cases. In the independent feature, all the features maintain the same value as the threshold value and do not experience a drop or increase. Whereas in the case of group drop, some of the features maintain the same thrshold value and some of the features experience a reduction of up to 62%. Also, an increase of 1% is observed in some features.

CONCLUSION

The literature analysis of balancing techniques for the CSE-CIC-IDS2018 clearly suggests an opportunity for research work which proposes novel techniques to handle the dataset imbalance. This result may be explained by the fact that a majority of the papers have not focused on proposing a new technique to overcome bias. This discrepancy can be a dominant focus area for future research directions. In the case of experimentation performed on the CSE-CIC-IDS2018, several experiments were performed like modeling on the full dataset using the train, and test split function from sci-kit learn library on the CSE-CIC-IDS2018. The accuracy results attained for all the classifiers KNN, RF, and LR was about 98% even if the dataset is highly imbalanced.

This experiment then lead to a subsequent investigation to find the threshold value at which the model attains the highest performance by using small sets of samples. In the 2nd part of the experiment, a small subset of the dataset is used to extract the botnet and infiltration attacks. This is done to notice the threshold trend for ten iterations in both the cases by fixing the benign sample. The classifier scores obtained in an increasing trend which is appositive behaviour. Whereas in the next part, the attack samples are fixed, and the classifier scores for 10 iterations generate a gradual decrease in the threshold values due to the decrease in benign samples. Performance metric scores are also analyzed for each attack, and the threshold values are compared with the 2nd experiment for verification. The results compared are similar, which again shows a positive behaviour.

Next experimentation deals with manual feature dropping, which is of two types, Independent and Group Drop. Independent drop shows a very minute drop for all the classifiers whereas group drop shows a huge drop of up to 90% in some classifiers. Hence, in future techniques like Principal Component Analysis (PCA), dimensionality reduction can be used due to a large number of features present in the CIC dataset. Finally, the manual feature drop and PCA results can be compared to see, which serves better.

REFERENCES

[1] C. F. Tsai, Y. F. Hsu, C. Y. Lin, and W. Y. Lin, “Intrusion detection by machine learning: A review,” *Expert Syst. Appl.*, vol. 36, no. 10, pp. 11994–12000, 2009, doi: 10.1016/j.eswa.2009.05.029.

[2] FSabahi and AMovaghar, “Intrusion detection: A survey,” *Proc. - 3rd Int. Conf. Syst. Networks Commun. ICSNC 2008 - Incl. I-CENTRIC 2008 Int. Conf. Adv. Human-Oriented Pers. Mech. Technol. Serv.*, pp. 23–26, 2008, doi: 10.1109/ICSNC.2008.44.

[3] G. Kumar, K. Kumar, and M. Sachdeva, “The use of artificial intelligence based techniques for intrusion detection: A review,” *Artif. Intell. Rev.*, vol. 34, no. 4, pp. 369–387, 2010, doi: 10.1007/s10462-010-9179-5.

[4] D. P. Vinchurkar and A. Reshamwala, “A Review of Intrusion Detection System Using Neural Network and Machine Learning Technique,” *Int. J. Eng. Sci. Innov. Technol.*, vol. 1, no. 2, pp. 54–63, 2012.

[5] S. Das and M. J. Nene, “A survey on types of machine learning techniques in intrusion prevention systems,” *Proc. 2017 Int. Conf. Wirel. Commun. Signal Process. Networking, WiSPNET 2017*, vol. 2018-Janua, pp. 2296–2299, 2018, doi: 10.1109/WiSPNET.2017.8300169.

[6] R. Malhotra and S. Kamal, “An empirical study to investigate oversampling methods for improving software defect prediction using imbalanced data,” *Neurocomputing*, vol. 343, pp. 120–140, 2019, doi: 10.1016/j.neucom.2018.04.090.

[7] “Introduction to Network-Based Intrusion Detection Systems | Network-Based Intrusion Detection | InformIT.” https://www.informit.com/articles/article.aspx?p=782118 (accessed Dec. 26, 2020).

[8] C. Yin, Y. Zhu, J. Fei, and X. He, “A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks,” *IEEE Access*, vol. 5, pp. 21954–21961, 2017, doi: 10.1109/ACCESS.2017.2762418.

[9] V. Jaganathan, P. Cherurveettil, and P. Muthu Sivashanmugam, “Using a prediction model to manage cyber security threats,” *Sci. World J.*, vol. 2015, 2015, doi: 10.1155/2015/703713.

[10] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, “Survey of intrusion detection systems: techniques, datasets and challenges,” *Cybersecurity*, vol. 2, no. 1, 2019, doi: 10.1186/s42400-019-0038-7.

[11] R. Mitchell and I. R. Chen, “A survey of intrusion detection techniques for cyber-physical systems,” *ACM Comput. Surv.*, vol. 46, no. 4, 2014, doi: 10.1145/2542049.

[12] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” *ICISSP 2018 - Proc. 4th Int. Conf. Inf. Syst. Secur. Priv.*, vol. 2018-Janua, no. Cic, pp. 108–116, 2018, doi: 10.5220/0006639801080116.

[13] R. Sommer and V. Paxson, “Outside the closed world: On using machine learning for network intrusion detection,” *Proc. - IEEE Symp. Secur. Priv.*, pp. 305–316,

2010, doi: 10.1109/SP.2010.25.
[14] D. Chicco and G. Jurman, “The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation,” *BMC Genomics*, vol. 21, no. 1, pp. 1–14, 2020, doi: 10.1186/s12864-019-6413-7.
[15] M. Buda, A. Maki, and M. A. Mazurowski, “A systematic study of the class imbalance problem in convolutional neural networks,” *Neural Networks*, vol. 106, pp. 249–259, 2018, doi: 10.1016/j.neunet.2018.07.011.
[16] “What are the types of machine learning? | by Hunter Heidenreich | Towards Data Science.” https://towardsdatascience.com/what-are-the-types-of-machine-learning-e2b9e5d1756f (accessed Dec. 26, 2020).
[17] “What is Machine Learning? A definition - Expert System | Expert.ai.” https://www.expert.ai/blog/machine-learning-definition/ (accessed Dec. 26, 2020).
[18] “What Is Machine Learning: Definition, Types, Applications and Examples | Potentia Analytics Inc.” https://www.potentiaco.com/what-is-machine-learning-definition-types-applications-and-examples/ (accessed Dec. 26, 2020).
[19] G. Kaur, A. Habibi Lashkari, and A. Rahali, “Intrusion Traffic Detection and Characterization using Deep Image Learning,” pp. 55–62, 2020, doi: 10.1109/dasc-picom-cbdcom-cyberscitech49142.2020.00025.
[20] M. I. Jordan and T. M. Mitchell, “Machine learning: Trends, perspectives, and prospects,” *Science*, vol. 349, no. 6245. American Association for the Advancement of Science, pp. 255–260, 17-Jul-2015, doi: 10.1126/science.aaa8415.
[21] “Artificial Intelligence (AI) Definition.” https://www.investopedia.com/terms/a/artificial-intelligence-ai.asp (accessed Dec. 26, 2020).
[22] “Deep Learning Definition.” https://www.investopedia.com/terms/d/deep-learning.asp (accessed Dec. 26, 2020).
[23] “The Seven Steps Of Machine Learning.” https://www.techleer.com/articles/379-the-seven-steps-of-machine-learning/ (accessed Dec. 29, 2020).
[24] “What is Data Preprocessing? - Definition from Techopedia.” https://www.techopedia.com/definition/14650/data-preprocessing (accessed Dec. 26, 2020).
[25] “Applications | Research | Canadian Institute for Cybersecurity | UNB.” https://www.unb.ca/cic/research/applications.html#CICFlowMeter (accessed Dec. 26, 2020).
[26] V. ENGEN, “Machine Learning For Network Based Intrusion Detection,” 2010, [Online]. Available: http://eprints.bournemouth.ac.uk/15899/1/Engen2010-PhD_single_sided.pdf.
[27] “What is SQL Injection? Tutorial & Examples | Web Security Academy.” https://portswigger.net/web-security/sql-injection (accessed Dec. 26, 2020).
[28] S. T. Zargar, J. Joshi, and D. Tipper, “A survey of defense mechanisms against distributed denial of service (DDOS) flooding attacks,” *IEEE Commun. Surv. Tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013, doi: 10.1109/SURV.2013.031413.00127.
[29] V. A. Chastikova and V. V. Sotnikov, “Method of analyzing computer traffic based on recurrent neural networks,” *J. Phys. Conf. Ser.*, vol. 1353, no. 1, 2019, doi:

10.1088/1742-6596/1353/1/012133.
[30] “7 Types of Data Bias in Machine Learning | Lionbridge AI.” https://lionbridge.ai/articles/7-types-of-data-bias-in-machine-learning/ (accessed Dec. 26, 2020).
[31] “Data Bias: Causes and Effects | Mailchimp.” https://mailchimp.com/resources/data-bias-causes-effects/ (accessed Dec. 26, 2020).
[32] G. Karatas, O. Demir, and O. K. Sahingoz, “Increasing the Performance of Machine Learning-Based IDSs on an Imbalanced and Up-to-Date Dataset,” *IEEE Access*, vol. 8, pp. 32150–32162, 2020, doi: 10.1109/ACCESS.2020.2973219.
[33] B. Subba, S. Biswas, and S. Karmakar, “A Neural Network based system for Intrusion Detection and attack classification,” *2016 22nd Natl. Conf. Commun. NCC 2016*, 2016, doi: 10.1109/NCC.2016.7561088.
[34] H. Al Najada, I. Mahgoub, and I. Mohammed, “Cyber Intrusion Prediction and Taxonomy System Using Deep Learning and Distributed Big Data Processing,” *Proc. 2018 IEEE Symp. Ser. Comput. Intell. SSCI 2018*, pp. 631–638, 2019, doi: 10.1109/SSCI.2018.8628685.
[35] V. Kanimozhi and D. T. P. Jacob, “Calibration of Various Optimized Machine Learning Classifiers in Network Intrusion Detection System on the Realistic Cyber Dataset Cse-Cic-Ids2018 Using Cloud Computing,” *Int. J. Eng. Appl. Sci. Technol.*, vol. 04, no. 06, pp. 209–213, 2019, doi: 10.33564/ijeast.2019.v04i06.036.
[36] J. L. Leevy and T. M. Khoshgoftaar, “A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data,” *J. Big Data*, vol. 7, no. 1, 2020, doi: 10.1186/s40537-020-00382-x.
[37] T. Chadza, K. G. Kyriakopoulos, and S. Lambotharan, “Contemporary Sequential Network Attacks Prediction using Hidden Markov Model,” *2019 17th Int. Conf. Privacy, Secur. Trust. PST 2019 - Proc.*, pp. 2019–2021, 2019, doi: 10.1109/PST47121.2019.8949035.
[38] B. Lypa, O. Iver, and V. Kifer, “Application of Machine Learning Methods for Network Intrusion Detection System,” 2018.
[39] J. Sun, L. Gu, and K. Chen, “An efficient alert aggregation method based on conditional rough entropy and knowledge granularity,” *Entropy*, vol. 22, no. 3, pp. 1–23, 2020, doi: 10.3390/e22030324.
[40] L. D’Hooge, T. Wauters, B. Volckaert, and F. De Turck, “Classification hardness for supervised learners on 20 years of intrusion detection data,” *IEEE Access*, vol. 7, pp. 167455–167469, 2019, doi: 10.1109/ACCESS.2019.2953451.
[41] Q. R. S. Fitni and K. Ramli, “Implementation of ensemble learning and feature selection for performance improvements in anomaly-based intrusion detection systems,” *Proc. - 2020 IEEE Int. Conf. Ind. 4.0, Artif. Intell. Commun. Technol. IAICT 2020*, pp. 118–124, 2020, doi: 10.1109/IAICT50021.2020.9172014.
[42] S. Dwibedi, “A Comparative Study on Contemporary Intrusion Detection Datasets for Machine Learning Research,” 2018.
[43] L. D’hooge, T. Wauters, B. Volckaert, and F. De Turck, “Inter-dataset generalization strength of supervised machine learning methods for intrusion detection,” *J. Inf. Secur. Appl.*, vol. 54, 2020, doi: 10.1016/j.jisa.2020.102564.

[44] A. Venturi, G. Apruzzese, M. Andreolini, M. Colajanni, and M. Marchetti, “DReLAB - Deep REinforcement Learning Adversarial Botnet: A benchmark dataset for adversarial attacks against botnet Intrusion Detection Systems,” *Data Br.*, vol. 34, p. 106631, 2021, doi: 10.1016/j.dib.2020.106631.

[45] F. Alghayadh and D. Debnath, “A Hybrid Intrusion Detection System for Smart Home Security,” *IEEE Int. Conf. Electro Inf. Technol.*, vol. 2020-July, pp. 319–323, 2020, doi: 10.1109/EIT48999.2020.9208296.

[46] R. Atefinia and M. Ahmadi, “Network intrusion detection using multi-architectural modular deep neural network,” *J. Supercomput.*, no. 0123456789, 2020, doi: 10.1007/s11227-020-03410-y.

[47] R. B. Basnet, R. Shash, C. Johnson, L. Walgren, and T. Doleck, “Towards detecting and classifying network intrusion traffic using deep learning frameworks,” *J. Internet Serv. Inf. Secur.*, vol. 9, no. 4, pp. 1–17, 2019, doi: 10.22667/JISIS.2019.11.30.001.

[48] A. Ali, S. M. Shamsuddin, and A. L. Ralescu, “Classification with class imbalance problem: A review,” *Int. J. Adv. Soft Comput. its Appl.*, vol. 7, no. 3, pp. 176–204, 2015.

[49] J. L. Leevy, T. M. Khoshgoftaar, R. A. Bauder, and N. Seliya, “A survey on addressing high-class imbalance in big data,” *J. Big Data*, vol. 5, no. 1, 2018, doi: 10.1186/s40537-018-0151-6.

[50] Y. Sun, A. K. C. Wong, and M. S. Kamel, “Classification of imbalanced data: A review,” *Int. J. Pattern Recognit. Artif. Intell.*, vol. 23, no. 4, pp. 687–719, 2009, doi: 10.1142/S0218001409007326.

[51] “A Primer to Ensemble Learning – Bagging and Boosting .” https://analyticsindiamag.com/primer-ensemble-learning-bagging-boosting/ (accessed Dec. 27, 2020).

[52] S. Ustebay, Z. Turgut, and M. A. Aydin, “Intrusion Detection System with Recursive Feature Elimination by Using Random Forest and Deep Learning Classifier,” *Int. Congr. Big Data, Deep Learn. Fight. Cyber Terror. IBIGDELFT 2018 - Proc.*, pp. 71–76, 2019, doi: 10.1109/IBIGDELFT.2018.8625318.

[53] J. H. Lee and K. H. Park, “AE-CGAN model based high performance network intrusion detection system,” *Appl. Sci.*, vol. 9, no. 20, 2019, doi: 10.3390/app9204221.

[54] Y. Zhang, X. Chen, L. Jin, X. Wang, and D. Guo, “Network Intrusion Detection: Based on Deep Hierarchical Network and Original Flow Data,” *IEEE Access*, vol. 7, pp. 37004–37016, 2019, doi: 10.1109/ACCESS.2019.2905041.

[55] Y. Zhou, G. Cheng, S. Jiang, and M. Dai, “Building an efficient intrusion detection system based on feature selection and ensemble classifier,” *Comput. Networks*, vol. 174, no. March, 2020, doi: 10.1016/j.comnet.2020.107247.

[56] I. A. Jimoh, I. Ismaila, and M. Olalere, “Enhanced Decision Tree-J48 with SMOTE Machine Learning Algorithm for Effective Botnet Detection in Imbalance Dataset,” *2019 15th Int. Conf. Electron. Comput. Comput. ICECCO 2019*, no. Icecco, 2019, doi: 10.1109/ICECCO48375.2019.9043233.

[57] R. Panigrahi and S. Borah, “Dual-stage intrusion detection for class imbalance scenarios,” *Comput. Fraud Secur.*, vol. 2019, no. 12, pp. 12–19, 2019, doi: 10.1016/S1361-3723(19)30128-9.

[58] R. Min, A. Bonner, and Z. Zhang, “Modifying Kernels using label information improves SVM classification performance,” *Proc. - 6th Int. Conf. Mach. Learn. Appl. ICMLA 2007*, pp. 13–18, 2007, doi: 10.1109/ICMLA.2007.76.
[59] J. Beel, B. Gipp, and E. Wilde, “Academic search engine optimization (ASEO/span),” *J. Sch. Publ.*, vol. 41, no. 2, pp. 176–190, 2010, doi: 10.3138/jsp.41.2.176.
[60] M. H. Abdulraheem and N. B. Ibraheem, “A detailed analysis of new intrusion detection dataset,” *J. Theor. Appl. Inf. Technol.*, vol. 97, no. 17, pp. 4519–4537, 2019.
[61] “The Top 10 Machine Learning Algorithms for ML Beginners.” https://www.dataquest.io/blog/top-10-machine-learning-algorithms-for-beginners/ (accessed Dec. 27, 2020).
[62] R. Abdulhammed, “Intrusion Detection: Embedded Software Machine Learning and Hardware Rules Based Co-Designs,” p. 176, 2019.
[63] “Performance Metrics for Classification problems in Machine Learning | by Mohammed Sunasra | Medium.” https://medium.com/@MohammedS/performance-metrics-for-classification-problems-in-machine-learning-part-i-b085d432082b (accessed Dec. 30, 2020).
[64] V. Tisler and A. Guerra, “Feature Selection for Machine Learning Based Iot Botnet Attack,” 2019.