12-2021

Network Traffic Analysis Using Local Outlier Factor

Khalifa Almheiri
kaa7155@rit.edu

RIT

Network Traffic Analysis Using Local Outlier Factor

by

Khalifa Almheiri

A Capstone Submitted in Partial Fulfilment of the Requirements for the Degree of Master of Science in Professional Studies:

Data Analytics

Department of Graduate Programs & Research

Rochester Institute of Technology

RIT Dubai

December 2021

RIT

Master of Science in Professional Studies:

Data Analytics

Graduate Capstone Approval

Student Name**: Khalifa Almheiri**

Graduate Capstone Title**: Network Traffic Analysis Using Local Outlier Factor**

Graduate Capstone Committee:

Name: Dr. Sanjay Modak **Date:**

Chair of committee

Name: Dr. Ehsan Warriach **Date:**

Member of committee

Acknowledgments

I would like to express my outmost gratitude and appreciation towards those who helped reach this point. I would like to thank my colleagues, my mentors, and my work. They have all supported me to pursue this journey and successfully reach to where I am today. I would like to send special thanks to Dr. Ehsan, and Dr. Sanjay both to which whom I owe a great debt in supporting me in my academics.

Abstract

The issue that this study addresses is the high rate of false positives, high maintenance, and lack of stability and precision that the existing network intrusion detection algorithm faces. To address this problem, we proposed a Local Outlier Factor (LOF) Algorithm that locates outliers and anomalies by comparing the deviation of one data point with respect to its neighbors. To gather data, we will use DARPA's KDDCup99 as well as questions towards analysts. This data will help determine whether the LOF algorithm is more effective than existing solutions that are presented in the network intrusion detection space.

Keywords:

Machine learning, prediction modeling, intrusion detection system, local outlier factor, data preprocessing, data classification, data prediction, anomaly detection, outlier detection.

Table of Contents

List of Figures

List of Tables

Chapter 1

1.1 Background

As cyber ecosystems evolve to become more mature and complex, cyber security continues to be one of the major concerns of our current decade (Holt & Bossler, 2013). The existence of anomalies in network traffic made it quite clear that the typical implementation of firewall and antivirus appliances are not sufficient enough to sustain a healthy cyber secure environment. It has been evident that many organizations have been suspectable to a wide variety of cyberattacks such as Twitter, Marriot, MGM resorts, Zoom, Magellan Health, Finastra, and SolarWinds (Downs & Brewer, 2020). Most IT based operations depends on having a robust environment than can tackle the forever evolving and adapting cyber threats. Such cyberattacks are attributed with major disturbances in the workflow of IT based operations. Such disturbances will exponentially increase the cybercrime cost to $10.5 trillion annually USD by 2025. Morgan states the jump from $3 trillion cybercrime costs to $10.5 trillion will represent "the greatest transfer of economic wealth in history" (Morgan, 2020) . 4 One of the most common method to tackle cyber terrorism is the integration of Intrusion Prevention Systems which prevents an attack from occurring based on a profile of a known threat. Signature based systems need to be periodically updated to be introduced to newer known threats, this have led multiple parties to invest into intrusion detection systems which is based on data mining techniques. These data mining approaches are favoured over manual and ad-hoc approaches due to the automation of detection model generation (Lee & Stolfo, 1998). It is dire to proactively handle arising issues by monitoring network traffic, analysing it and

properly weeding out outliers and anomalies that do not fit within a defined normal behaviour. We can tackle this issue by implementing a multi-layered approach by introducing an outlier detection algorithm (J & Dr.B.Muthukumar, 2015). Most existing outlier detection approaches face issues in three major areas which are: (a) Stability, (b) precision, and (c) rate of false positives (Lazarevic, Ertoz, Ozgur, Kumar, & Srivastava, 2003; J & Dr.B.Muthukumar, 2015). This paper will revolve around a proposed solution based on unsupervised outlier detection called Local Outlier Factor algorithm.

1.2 Statement of problem

Digital transformation has been the focus of many countries. The race to gain technological influence has propelled technological advancements to heights that were thought to be nearly impossible. As more countries became aware of the benefits of digitalizing their infrastructure, more dangers appeared. Such dangers are known as cyber threats. One way of combating cyber threats is to adopt an intrusion detection system in addition to the firewall. An intrusion detection system uses many tools and methods to combat and repel attacks. One of the methods used, is the adoption of machine learning techniques to detect anomalies. Many models have been introduced and debated through various scholarly papers, but intrusion detection systems still face several challenges in many sectors such as precision and response time. Anomaly based Intrusion detection systems has been associated with having a high number of false positives, this has increased the load on operation teams. A team would have to investigate, maintain, enhance, and respond to every single alert which requires time, and more manpower thus siphoning precious resources from whatever organization it is hosted on. There is a need of a stable, precise, and fast

model to be adopted to combat the associated challenges. There are new types of attacks discovered daily and an anomaly-based intrusion detection system is one of the best ways to combat that trend.

1.3 Project goals

Research Questions:

Research Question 1: Clockworks behind the Local Outlier Factor algorithm?

Justification: It is detrimental to the research to show the reader how does the proposed solution works within a typical IT infrastructure. It will highlight the solution's strengths and create a robust and cohesive argument that would direct a reader's attention to why this solution is the most suitable.

Research Question 2: Comparison between Local Outlier Factor algorithm and existing implemented techniques in various areas?

Justification: To justify our proposed solution for outlier detection over network traffic a comparison must be made to see how will our proposed solution compares and contrasts to other proposed solutions mentioned in the existing literatures.

Research Question 3: Impact of implementing Local Outlier Factor algorithm on the rate of false positive alarms?

Justification: One of the major concerns of outlier analysis algorithm is that they suffer from a high rate of false positives. False positive hinders the workflow of operations so it is important to verify whether our proposed solution tackles this issue or not. The objective of our research is to

use the Local Outlier Algorithms as a proposed solution to tackle network threats by observing how does it handle anomaly detection of multiple network datasets based on a specific metric. It will be our objective to highlight how precise and stable LOF is, by testing the occurrence of false positive alarms on both 5 static, and stream environments. We will justify why our proposed solution has an edge over all other existing solutions, and why does it need to be implemented on existing intrusion detection systems by making a comparison between our proposed solution and existing approaches, and by making an analysis of the factors which caused the issues to be present within the existing approaches.

The goals of this project are to:

1. Increase anomaly/intrusion detection systems precision and reduce the number of false positives.
2. Decrease response time, and enhance detection time.
3. Reduce strain and workload on technical teams that are maintaining the intrusion detection system.
4. Reduce costs for organizations that invest into adopting or developing their own anomaly/hybrid intrusion detection systems.
5. Enhance operations in regards to the validation of IDS triggered alarm classification.

1.4 Methodology

Overview:

This project will follow a mixed method approach, more specifically, an explanatory sequential mixed method approach. In this approach, a quantitative analysis will be performed initially, and then the results will be explained in greater detail using qualitative results. In our project, we will follow a deductive procedure for the research. Starting with our theory which is LOF algorithm can perform better than the other intrusion detection algorithms, we will then obtain our hypothesis. The hypothesis would be tested by comparing the LOF algorithms with the other intrusion detection systems, and if it succeeds then the hypothesis is confirmed. To better understand the difference between different intrusion detection systems, we will also include qualitative data to gain insight and better understand the quantitative analysis. Network Intrusion has wide range of datasets that could be analyzed. In this project, KDDCup99 dataset will be utilized. Quantitative analysis will be performed on the datasets to compare the performance of our proposed solution, Local Outlier Factor (LOF), with the existing solutions. The existing solutions suffer from a high rate of false positives, frail stability, and low precision. Therefore, the research will focus on analyzing those factors quantitatively and determine the LOF algorithm's performance.

Roadmap:

The project is going to go through several phases mentioned as shown below:

Figure 1 Project Roadmap

Tools and add-ons:

During this study, I have used various tools, and add-ons to generate the evidences present in this study, and the corresponding visualizations. I have utilized a combination of Anaconda's Spyder platform to make use of Python commands, R studio for R language, and Tableau. Python has been used mainly to test various algorithms against the proposed solution and capture the statistical results for analysis. Utilizing python, R, and Tableau, I was able to visualize the plots and graphs that are present within this study.

Python Libraries Used:

Table 1 Python Libraries

Library	Function	Purpose
Sklearn.model_selection	Train_test_split	To use to split the dataset into test and training datasets.
Sklearn.neighbors	LocalOutlierFactor	To use the LOF algorithm with ease while tuning the parameters.
Sklearn.preprocessing	LabelEncoder	To encode categorical variables where needed.
Sklearn.metrics	Mean_absolute_error	To calculate the mae.
Sklearn.metrics	Mean_squared_error	To calculate the mse.
Sklearn.metrics	Accuracy_score	To calculate the accuracy.
Numpy	-	
Pandas	-	
Datetime	Datetime	To get the start and end time of the fitting process.
Sklearn.covariance	EllipticEnvelope	To use the EE algorithm with ease while tuning the parameters.

Sklearn.vsm	OneClassSVM	To use the SVM algorithm with ease while tuning the parameters.
Sklearn.ensemble	IsolationForest	To use the IF algorithm with ease while tuning the parameters.
Matplotlib	Pyplot	To plot certain plots and graphs.
Itertools	-	-
Sklearn.metrics	Confusion_matrix	To generate a confusion matrix.
Sklearn.ensemble	RandomForestClassifier	To use the RF algorithm with ease while tuning the parameters.
Sklearn.ensemble	IsolationForest	To use the IF algorithm with ease while tuning the parameters.
Sklearn.metrics	F1_score	To retrieve the F1 score.
Sklearn.metrics	Precision_score	To retrieve the precision score.
Sklearn.metrics	Recall_score	To retrieve the recall score.
Sklearn.metrics	Classification_report	To generate the classification report which

		include f1, precision, and recall scores.
Seaborn	-	

R Packages Used:

Table 2 R libraries

Libraries
Tidyverse
Dplyr
Data.table
Pander
Ggplot2
Plotrix
Ggthemes
Xtable
Lubricate
knitr

1.5 Limitations of the Study

The study had one major limitation which was insufficient computational resources to run multiple iterations in parallel. One iteration takes a lot of time and uses a huge number of computational resources. Not having a sufficient platform to conduct tests and analysis elongated the entire process which limit how thorough I could conduct the research.

Chapter 2 – Literature Review

Introduction:

As the world hurls towards digital transformation or rather a digital era, security concerns in regards to confidentiality, integrity and availability are becoming more common and apparent (Aljanabi, Ismail, & Ali1, 2020). To detect malicious attempts there is a need to observe and analyse different records in a network. This is one of the most significant ways to deal with the rising network security concerns. The increase in network usage and proliferation, and data transfer rates resulted in a high anomaly occurrence rate (J & Dr.B.Muthukumar, 2015). To tackle cyber security concerns various outlier detectionbased intrusion detection systems were developed. An intrusion detection system is an effective enabler to (a) effective network traffic monitoring, (b) continuous servers and network monitoring in regards to misuse actions or abuse policy, and (c) network attack alerting, response, and reporting; which is all possible by utilizing three different approaches which can be either: (a) anomaly-based, (b) mis-use based, or (c) a hybrid approach (Beigh, Bashir, & Chachoo, 2013). This literature review is going to go through various concepts to explain and elaborate on the nature of anomaly and outlier detection, how outlier analysis fits in intrusion detection systems, some of the major concerns of the existing approaches to outlier detection, an overview of existing literature about the LOF proposed solution, and schema comparison between various methods and algorithms. This literature review hopes to produce an evaluation of existing concepts of outlier analysis in network traffic. Many publications exist that covers outlier detection approaches, but it is required to review the research in order to identify

under-examined aspects of various existing solutions and to build a foundation which would justify our proposed solution.

Anomaly/outlier detection:

Anomaly detection refers to the process of identifying abnormal network behaviour that is considered to be peculiar when fitted against data patterns. Anomaly detection is valued due to its' ability to translate outputs or rather results into actionable information which can be quite significant in certain domains (Bhuyan, BhattacharyyaJ, & Kalita, 2011). Whilst Anomalies encompass important actionable information, they can also be just noise. The detection of anomalies is purely based on the method used. The method can drastically impact the classification of an event or object. Outlier detection utilizes various methods that can be either distance-based, density-based, and machine-learning based (Gogoi, Bhattacharyya, & and, 2011). Omar (2020) categorizes outlier detection algorithms as either supervised, unsupervised, and semi-supervised. Supervised, which works under the assumption that both test, and labelled datasets are included. Unsupervised, in which labelled dataset is not required, and also eliminates discrimination between testing, and training datasets. Whilst in semi-supervised, labels are not needed for outlier class, and works under the assumption that tables were included in the data in encompassed in the training data is only for the normal class.

Outlier Analysis in Intrusion Detection Systems:

Intrusion detection monitors events that are on a system or a network, and analyses those events to check for intrusions such as peculiar activities and unauthorized access. The process of intrusion detection can be divided into three separate steps: (a) Monitoring and analysis, (b)

identifying anomalies and peculiarities, and (c) assessing severity and sending alarms (Gupta, Gupta, & Bhattacharjee, 2019).

IDS has two models to discover and analyse attacks:

1. Misuse model: Detects attacks by using known vulnerabilities and signatures as metrics.
2. Anomaly detection model: it detects attacks by searching for abnormalities within network traffic.

Companies such as FortiGate, F5, and Cisco has their own IDS. The IDS contains both misuse models that are updated for known signatures, and anomaly models that detect attacks without specifying attack models. The main issue with anomaly models is that due to its nature it suspectable to high rate of false positives (J & Dr.B.Muthukumar, 2015).

In intrusion detection systems, intrusion techniques are generally characterized as either (a) signature based, or (b) anomaly based. Signature based detection compares an intrusion event to a known pattern, while anomaly-based detection is based on peculiarities of established profiles of predefined and specific activities (Bhuyan, BhattacharyyaJ, & Kalita, 2011).

Alert Classification:

Both Intrusion prevention systems and intrusion detection systems alike generate multiple types of alarms. An alarm is raised when an attack or a violation is detected. While an alarm can classify an event as an attack, it doesn't always classify it correctly. (Cisco Press, 2009) explains the different types of alerts in the following manner:

1. False positive: This alarm is triggered when an IDS or IPS classify an event in a normal traffic as an attack. For example: An alarm will be raised because an IDS cannot differentiate between incorrect password entry by a network administrator and a rogue user.
2. False Negative: This occurs when an IDS classifies an attack as normal behaviour. This means that the IDS failed to detect an attack.
3. True Positive: Is when an attack is correctly classified and detect, and an alarm is raised.
4. True Negative: Is when an event is classified as normal behaviour thus not ignoring this event and not generating an alarm.

Major Concerns of existing approaches:

Some of the major concerns that had been highlighted by the existing literatures are the issues that are attributed to the existing outlier analysis approaches and methods. (J & Dr.B.Muthukumar, 2015) stated that there are certain drawbacks to some of the existing techniques such as (a) weak detection stability and (b) low detection precision. He also highlighted that there are many issues in the existing literature such as accurate identification and classification of attacks.

Omar (2020) described the weaknesses of existing approaches by dividing them based on taxonomy nearest Neighbor-Based Techniques and clustering-based techniques:

Neighbor-based techniques have the advantage of not needing an assumption to be applied and can be applied on different data types; its' weakness lies in the fact that it needs an appropriate distance to do its' calculations. It also takes a long computation time when applied to big data.

Clustering-based techniques is quite simple but comes with a lot of weaknesses. It has a dependency on the efficiency of the clustering algorithm, it is not designed well to detect outliers, it has also the issue of viewing some abnormalities as normal data due having to process each data points to be distributed in multiple clusters, and the measurement for clustering algorithm is quite complicated.

Anomaly based detection suffers for a high rate of false positives. Monowar (2011) highlighted that the main challenge is to minimize the rate of false positives.

Established LOF(Local Outlier Algorithm) Concepts:

LOF is a density-based algorithm, it locates outliers and anomalies by comparing the deviation of one data point in respect to its neighbours. It assumes that the group with lower density is an outlier or an anomaly. It uses the Euclidian distance to estimate the local density. It hold an advantage with spherical data (Breunig, Kriegel, Ng, & Sander, 2000).

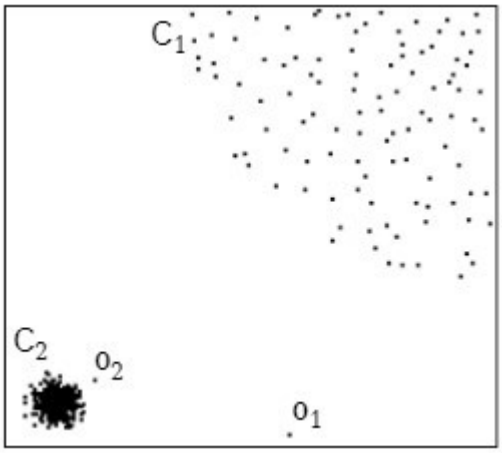


Figure 2 LOF Sample (Breunig, Kriegel, Ng, & Sander, 2000)

Figure 1, is a sample dataset that shows a dataset with 502 objects. Based on Hawkins' definition both O2 and O1 are considered as outliers. As observed O2 and O1 are quite distant from the cluster of C2, and C1 (Breunig, Kriegel, Ng, & Sander, 2000).

Schema comparison of some of the existing algorithms:

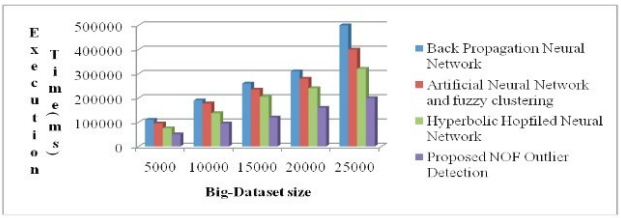


Figure 3 Detection Rate (J & Dr.B.MUTHUKUMAR, 2015)

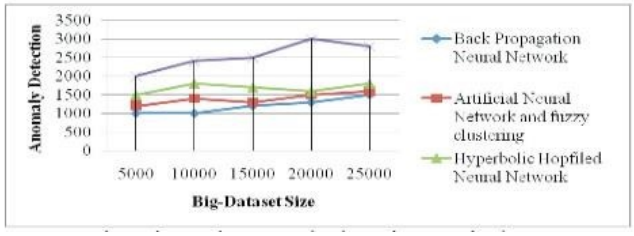


Figure 4 Dataset vs Execution Time (Dr.B.Muthukumar, 2015)

It has been observed that a variety of neural network and clustering algorithms had a higher execution time when compared to some density-based approaches such as local outlier factor algorithm. Such deficiencies have been observed using the same algorithms in terms of anomaly detection rate and CPU utilization. (Bhuyan, BhattacharyyaJ, & Kalita, 2011) also highlighted

the false positive alarm issues that existed within the existing approaches and made a claim that a NADO approach that he suggested can solve this major concern.

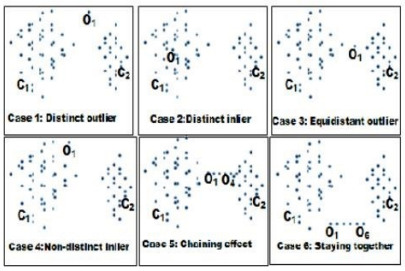


Figure 5 6 different cases (Gogoi, Bhattacharyya, & and, 2011)

Table 3 Outlier detection approaches comparison (Gogoi, Bhattacharyya, & and, 2011)

Approaches	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
Distance-based	Yes	Yes	Yes	Yes	Yes	No
Density-based	Yes	Yes	No	Yes	Partially	Partially
Soft com-puting	Yes	Yes	Yes	Yes	Partially	No

Fig 3. Shows 6 different cases using synthetic data which were used to evaluate the effectiveness of the outlier detection methods as follows: (a) case 1: Distinct Outlier, (b) Case 2: Distinct inlier, (c) Case 3: Equidistant Outlier, (d) Case 4: Non-Distinct Outlier, (e) Case 5: Changing Effect, and (d) Case 5: Staying effect (Gogoi, Bhattacharyya, & and, 2011). Table 1 shows how effective the outlier detection approaches against those cases using three different approaches: (a) Distance-based, (b) Density-Based, (c) Machine learning or soft computing based (Gogoi, Bhattacharyya, & and, 2011).

Conclusion:

After going through various papers, it was apparent that the cyber community agree that outlier detection is a vital task to ensure the security of any system or network environment. Outlier based techniques aim to find abnormal events that are peculiar when compared to normal events. There are a lot of existing literature that examined almost every algorithm and methods, Jabez

(2015), Gupta (2019), and Breunig (2000) extensively went through securing the foundation for our proposed solution, but they have yet to examine some of the issues that have been observed in other studies. The research is purposed around the evaluation of the LOF proposed solution and how does it compare to its counterparts. The research will also focus on the infrastructure as well while highlighting the benefits and efficiency of the proposed solution. We hope to address the prosed solution's precision, stability in outlier detection and the rate of which false positive occur.

Chapter 3- Project Description

3.1 Overview:

The project will go through several phases in which it will prepare the selected dataset to be utilized to test and model the proposed solution. The project will start by the pre-processing steps of data cleaning, and normalization which consist of dealing with redundant, and missing data and making sure that the data the is present within the dataset adheres to integrity. The dataset will be later explored by visualizing the data using numerous appliances to generate a diverse selection of visuals to gain insight from the different features and their relationships. Testing and modeling will focus on validifying our proposed solution by various means such as model comparison and several estimates regarding several dimensions such as precision, computational expense, and time.

3.2 Data Collection and Information:

The dataset that is used in this project is a known dataset used by many to evaluate anomaly detection methods. It was originally used in the 1999 KDD intrusion detection contest. The objective of the contest was to evaluate research made in regard to anomaly and intrusion detection. The information present in the dataset might look irregular and that is due to it being simulated to match military network environment. The Dataset is made of 41 features and consist of 4.9 million records. The records are also labeled as either normal or an attack.

Chapter 4- Project Analysis

4.1 Dataset Overview

The KDDCup99 dataset is a widely known dataset mainly used to test for anomaly and outlier detection. The dataset is made up of 41 features which are a mix of both numerical and categorical variables. The features present in the dataset are as follows:

Table 4 Feature Description

Feature Name	Description	Type
Duration	Duration of connection in seconds	Continuous
Protocol_type	Type of protocol such as udp, tcp, icmp, etc.	Discrete
Service	Destination network service such as http, https, telnet, etc.	Discrete
Src_bytes	Source to destination bytes count	Continuous
Dst_bytes	Destination to source bytes count	Continuous
Flag	Connection status	Continuous
Land	1: connection is from or to the same host or port	Discrete

	0: anything else	
Wrong_fragment	Count of wrong fragments	Continuous
Urgent	Number of urgent packets	Continuous
Hot	Hot indicators count	Continuous
Num_failed_logins	Failed logins count	Continuous
Logged_In	1: logged in successfully 0: anything else	Discrete
Num_compromised	Compromised condition count	Continuous
Root_shell	1: shell obtained 0: Anything else	Discrete
Su_attempted	1: su command attempted 0: Anything else	Discrete
Num_root	Root access count	Continuous
Num_file_creations	File creation count	Continuous
Num_shells	Shell prompts count	Continuous
Num_access_files	Control files operations counts	Continuous
Num_outbound_cmd	Outbound commands count in an ftp session	Continuous
Is_hot_login	1: login belongs to hot list 0: Anything else	Discrete
Is_Guest_login	1: login is guest login 0: Anything else	Discrete

Count	Connection count to same host as current connection in the previous 2 seconds.	Continuous
Serror_rate	Percentage of connections with SYN errors	Continuous
Rerror_rate	Percentage of connections with REJ errors	Continuous
Same_srv_rate	Percentage of connections to similar services	Continuous
Diff_srv_rate	Percentage of connection to different services	Continuous

Table 5 Additional Features

Additional Server and Host Features		
Srv_diff_host_rate	Dst_host_count	Dst_host_srv_count
Dst_host_same_srv_rate	Dst_host_diff_srv_rate	Dst_host_same_src_port_rate
Dst_host_srv_diff_host_rate	Dst_host_serror_rate	Dst_host_srv_serror_rate
Dst_host_rerror_rate	Dst_host_srv_rerror_rate	outcome

The KDD'99 dataset has approximately 4 million records which are label as either normal/good connections or a malicious/bad connection. The attacks generally fall under four different categories which are denial of service attacks, user to root attacks, remote to local attacks, and probing attacks. The attacks in the dataset are labeled as: back, buffer overflow, ftp write, guess

password, imap, ipsweep, land, load module, multihop, Neptune, nmap, perl, phf, pod, pordsweep, rootkit, satan, smurf, spy, teardrop, warezclient, and warezmaster.

4.2 Data Cleaning

Prior to testing, it is a must to make sure that the dataset does not contain any inadequacies that may hinder the testing process, or rather much more importantly result in inaccurate analysis. Such inadequacies or issues can be either missing variables, redundant records, curse of dimensionality or all the issues that were mentioned prior. It is highly likely that the dataset will suffer from the curse of dimensionality, and it will need to be handled properly since there are several algorithms that suffer greatly for its presence.

Initial Overview:

Loading in the KDD'99 dataset it is observed that the features do not contain the correct feature names that reflect the natures of the records of each column as shown below:

V1 <int>	V2 <chr>	V3 <chr>	V4 <chr>	V5 <int>	V6 <int>	V7 <int>	V8 <int>	V9 <int>	V10 <int>
0	tcp	http	SF	215	45076	0	0	0	0
0	tcp	http	SF	162	4528	0	0	0	0
0	tcp	http	SF	236	1228	0	0	0	0
0	tcp	http	SF	233	2032	0	0	0	0
0	tcp	http	SF	239	486	0	0	0	0
0	tcp	http	SF	238	1282	0	0	0	0

6 rows | 1-10 of 42 columns

Figure 6 Sample Dataset View

Using the list provided from the source of the dataset, it was possible to match the columns with their corresponding names. Below is a sample of how the dataset looks like after fixing the feature names:

duration <int>	protocol_type <chr>	service <chr>	flag <chr>	src_bytes <int>	dst_bytes <int>	land <int>	wrong_fragment <int>	urgent <int>	hot <int>
0	tcp	http	SF	215	45076	0	0	0	0
0	tcp	http	SF	162	4528	0	0	0	0
0	tcp	http	SF	236	1228	0	0	0	0
0	tcp	http	SF	233	2032	0	0	0	0
0	tcp	http	SF	239	486	0	0	0	0
0	tcp	http	SF	238	1282	0	0	0	0

6 rows | 1-10 of 42 columns

Figure 7 Labeled Dataset

Dealing with missing values:

Dealing with missing values is an integral part of the data cleaning process, it avoids complication that may arise during later operations. Dealing with missing values can come in many forms. We can either remove the records altogether or impute them by mean, median or regression.

```
### Check for NA

```{r missingvalues}
#Using the combination of the is.na function, to get the NA values, and the sum function to calculate the number of NA values.
sum(is.na(KcupDataDistinct))
```
```

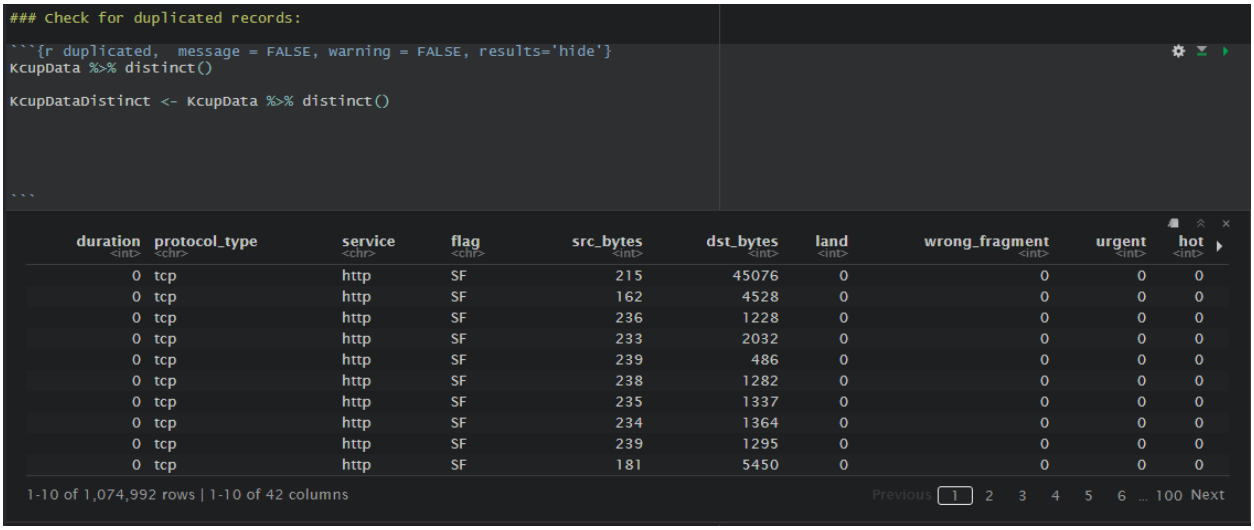
```
[1] 0
```

Figure 8 Missing Values

Calculating the sum of NA values returns an output of 0. This concludes that the dataset contains no missing data. There will be no need to impute nor remove any records.

Dealing with redundant records:

Duplicated records can lead to many complications down the line. Not only it can skew data in an unfavorable way, but it can also provide in accurate data.



```
### Check for duplicated records:

```{r duplicated,  message = FALSE, warning = FALSE, results='hide'}
KcupData %>% distinct()

KcupDataDistinct <- KcupData %>% distinct()

```
```

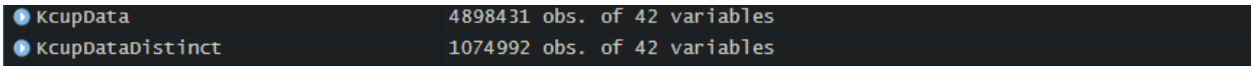
duration <int>	protocol_type <chr>	service <chr>	flag <chr>	src_bytes <int>	dst_bytes <int>	land <int>	wrong_fragment <int>	urgent <int>	hot <int>
0	tcp	http	SF	215	45076	0	0	0	0
0	tcp	http	SF	162	4528	0	0	0	0
0	tcp	http	SF	236	1228	0	0	0	0
0	tcp	http	SF	233	2032	0	0	0	0
0	tcp	http	SF	239	486	0	0	0	0
0	tcp	http	SF	238	1282	0	0	0	0
0	tcp	http	SF	235	1337	0	0	0	0
0	tcp	http	SF	234	1364	0	0	0	0
0	tcp	http	SF	239	1295	0	0	0	0
0	tcp	http	SF	181	5450	0	0	0	0

1-10 of 1,074,992 rows | 1-10 of 42 columns

Previous 1 2 3 4 5 6 ... 100 Next

Figure 9 Dealing with Duplicate Records

Using R studio and utilizing R language in addition the ‘dyplr’ package, it was possible to remove the duplicated records from the dataset. As observed on the image above, the number of rows has decreased significantly with the removal of the redundant records. It has decreased the number of records from 4,898,431 to 1,074,992.



```
KcupData                4898431 obs. of 42 variables
KcupDataDistinct        1074992 obs. of 42 variables
```

Figure 10 Record Count

4.4 Exploratory Data Analysis

The dataset has a lot of insight to offer visualizing the dataset gives us an overview of the nature of the dataset and what is present within it.

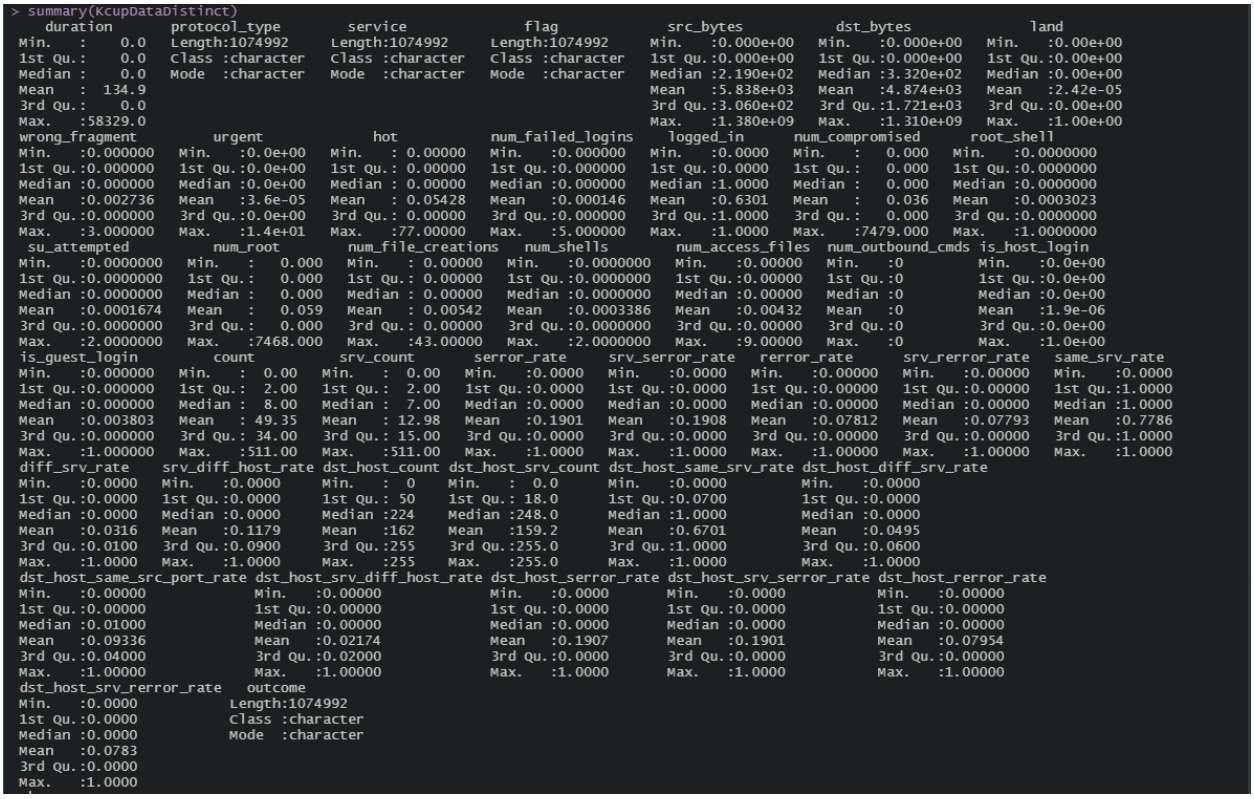
Statistical Summary:

The KddCup99 dataset is made up of 41 features, out of the 41 feature there are 38 numerical features and 4 categorical features. Post-processing, it seems that there are no null values, and all of the features are observed to contain the total amount of records present in the dataset which is 1,074,992.

```
Data columns (total 43 columns):
 #   Column                       Non-Null Count    Dtype
---  ------                       --------------    -----
 0   Unnamed: 0                   1074992 non-null  int64
 1   duration                     1074992 non-null  int64
 2   protocol_type                1074992 non-null  object
 3   service                      1074992 non-null  object
 4   flag                         1074992 non-null  object
 5   src_bytes                    1074992 non-null  int64
 6   dst_bytes                    1074992 non-null  int64
 7   land                         1074992 non-null  int64
 8   wrong_fragment               1074992 non-null  int64
 9   urgent                       1074992 non-null  int64
 10  hot                          1074992 non-null  int64
 11  num_failed_logins            1074992 non-null  int64
 12  logged_in                    1074992 non-null  int64
 13  num_compromised              1074992 non-null  int64
 14  root_shell                   1074992 non-null  int64
 15  su_attempted                 1074992 non-null  int64
 16  num_root                     1074992 non-null  int64
 17  num_file_creations           1074992 non-null  int64
 18  num_shells                   1074992 non-null  int64
 19  num_access_files             1074992 non-null  int64
 20  num_outbound_cmds            1074992 non-null  int64
 21  is_host_login                1074992 non-null  int64
 22  is_guest_login               1074992 non-null  int64
 23  count                        1074992 non-null  int64
 24  srv_count                    1074992 non-null  int64
 25  serror_rate                  1074992 non-null  float64
 26  srv_serror_rate              1074992 non-null  float64
 27  rerror_rate                  1074992 non-null  float64
 28  srv_rerror_rate              1074992 non-null  float64
 29  same_srv_rate                1074992 non-null  float64
 30  diff_srv_rate                1074992 non-null  float64
 31  srv_diff_host_rate           1074992 non-null  float64
 32  dst_host_count               1074992 non-null  int64
 33  dst_host_srv_count           1074992 non-null  int64
 34  dst_host_same_srv_rate       1074992 non-null  float64
 35  dst_host_diff_srv_rate       1074992 non-null  float64
 36  dst_host_same_src_port_rate  1074992 non-null  float64
 37  dst_host_srv_diff_host_rate  1074992 non-null  float64
 38  dst_host_serror_rate         1074992 non-null  float64
 39  dst_host_srv_serror_rate     1074992 non-null  float64
 40  dst_host_rerror_rate         1074992 non-null  float64
 41  dst_host_srv_rerror_rate     1074992 non-null  float64
 42  outcome                      1074992 non-null  object
dtypes: float64(15), int64(24), object(4)
memory usage: 352.7+ MB
```

Figure 11 Statistical Summary

Overall Feature Statistics:



```
> summary(KcupDataDistinct)
    duration       protocol_type        service             flag          src_bytes           dst_bytes             land
 Min.   :    0.0   Length:1074992     Length:1074992     Length:1074992     Min.   :0.000e+00   Min.   :0.000e+00   Min.   :0.00e+00
 1st Qu.:    0.0   Class :character   Class :character   Class :character   1st Qu.:0.000e+00   1st Qu.:0.000e+00   1st Qu.:0.00e+00
 Median :    0.0   Mode  :character   Mode  :character   Mode  :character   Median :2.190e+02   Median :3.320e+02   Median :0.00e+00
 Mean   :  134.9                                                           Mean   :5.838e+03   Mean   :4.874e+03   Mean   :2.42e-05
 3rd Qu.:    0.0                                                           3rd Qu.:3.060e+02   3rd Qu.:1.721e+03   3rd Qu.:0.00e+00
 Max.   :58329.0                                                           Max.   :1.380e+09   Max.   :1.310e+09   Max.   :1.00e+00
 wrong_fragment        urgent              hot          num_failed_logins    logged_in      num_compromised      root_shell
 Min.   :0.000000   Min.   :0.0e+00   Min.   : 0.00000   Min.   :0.000000   Min.   :0.0000   Min.   :    0.000   Min.   :0.0000000
 1st Qu.:0.000000   1st Qu.:0.0e+00   1st Qu.: 0.00000   1st Qu.:0.000000   1st Qu.:0.0000   1st Qu.:    0.000   1st Qu.:0.0000000
 Median :0.000000   Median :0.0e+00   Median : 0.00000   Median :0.000000   Median :1.0000   Median :    0.000   Median :0.0000000
 Mean   :0.002736   Mean   :3.6e-05   Mean   : 0.05428   Mean   :0.000146   Mean   :0.6301   Mean   :    0.036   Mean   :0.0003023
 3rd Qu.:0.000000   3rd Qu.:0.0e+00   3rd Qu.: 0.00000   3rd Qu.:0.000000   3rd Qu.:1.0000   3rd Qu.:    0.000   3rd Qu.:0.0000000
 Max.   :3.000000   Max.   :1.4e+01   Max.   :77.00000   Max.   :5.000000   Max.   :1.0000   Max.   :7479.000   Max.   :1.0000000
  su_attempted         num_root        num_file_creations   num_shells        num_access_files  num_outbound_cmds is_host_login
 Min.   :0.0000000   Min.   :    0.000   Min.   : 0.00000   Min.   :0.0000000   Min.   :0.00000   Min.   :0         Min.   :0.0e+00
 1st Qu.:0.0000000   1st Qu.:    0.000   1st Qu.: 0.00000   1st Qu.:0.0000000   1st Qu.:0.00000   1st Qu.:0         1st Qu.:0.0e+00
 Median :0.0000000   Median :    0.000   Median : 0.00000   Median :0.0000000   Median :0.00000   Median :0         Median :0.0e+00
 Mean   :0.0001674   Mean   :    0.059   Mean   : 0.00542   Mean   :0.0003386   Mean   :0.00432   Mean   :0         Mean   :1.9e-06
 3rd Qu.:0.0000000   3rd Qu.:    0.000   3rd Qu.: 0.00000   3rd Qu.:0.0000000   3rd Qu.:0.00000   3rd Qu.:0         3rd Qu.:0.0e+00
 Max.   :2.0000000   Max.   :7468.000   Max.   :43.00000   Max.   :2.0000000   Max.   :9.00000   Max.   :0         Max.   :1.0e+00
 is_guest_login        count           srv_count       serror_rate     srv_serror_rate   rerror_rate      srv_rerror_rate   same_srv_rate
 Min.   :0.000000   Min.   :  0.00   Min.   :  0.00   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000   Min.   :0.00000   Min.   :0.0000
 1st Qu.:0.000000   1st Qu.:  2.00   1st Qu.:  2.00   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:1.0000
 Median :0.000000   Median :  8.00   Median :  7.00   Median :0.0000   Median :0.0000   Median :0.00000   Median :0.00000   Median :1.0000
 Mean   :0.003803   Mean   : 49.35   Mean   : 12.98   Mean   :0.1901   Mean   :0.1908   Mean   :0.07812   Mean   :0.07793   Mean   :0.7786
 3rd Qu.:0.000000   3rd Qu.: 34.00   3rd Qu.: 15.00   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:1.0000
 Max.   :1.000000   Max.   :511.00   Max.   :511.00   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000   Max.   :1.00000   Max.   :1.0000
 diff_srv_rate    srv_diff_host_rate dst_host_count dst_host_srv_count dst_host_same_srv_rate dst_host_diff_srv_rate
 Min.   :0.0000   Min.   :0.0000     Min.   :  0    Min.   :  0.0      Min.   :0.0000          Min.   :0.0000
 1st Qu.:0.0000   1st Qu.:0.0000     1st Qu.: 50    1st Qu.: 18.0      1st Qu.:0.0700          1st Qu.:0.0000
 Median :0.0000   Median :0.0000     Median :224    Median :248.0      Median :1.0000          Median :0.0000
 Mean   :0.0316   Mean   :0.1179     Mean   :162    Mean   :159.2      Mean   :0.6701          Mean   :0.0495
 3rd Qu.:0.0100   3rd Qu.:0.0900     3rd Qu.:255    3rd Qu.:255.0      3rd Qu.:1.0000          3rd Qu.:0.0600
 Max.   :1.0000   Max.   :1.0000     Max.   :255    Max.   :255.0      Max.   :1.0000          Max.   :1.0000
 dst_host_same_src_port_rate dst_host_srv_diff_host_rate dst_host_serror_rate dst_host_srv_serror_rate dst_host_rerror_rate
 Min.   :0.00000             Min.   :0.00000            Min.   :0.0000       Min.   :0.0000           Min.   :0.00000
 1st Qu.:0.00000             1st Qu.:0.00000            1st Qu.:0.0000       1st Qu.:0.0000           1st Qu.:0.00000
 Median :0.01000             Median :0.00000            Median :0.0000       Median :0.0000           Median :0.00000
 Mean   :0.09336             Mean   :0.02174            Mean   :0.1907       Mean   :0.1901           Mean   :0.07954
 3rd Qu.:0.04000             3rd Qu.:0.02000            3rd Qu.:0.0000       3rd Qu.:0.0000           3rd Qu.:0.00000
 Max.   :1.00000             Max.   :1.00000            Max.   :1.0000       Max.   :1.0000           Max.   :1.00000
 dst_host_srv_rerror_rate   outcome
 Min.   :0.0000           Length:1074992
 1st Qu.:0.0000           Class :character
 Median :0.0000           Mode  :character
 Mean   :0.0783
 3rd Qu.:0.0000
 Max.   :1.0000
```

Figure 12 Overall Features Statistics Summary

Figure 12 shows that most features have the values of either 1 or 0 excluding features such as count, features that pictures the rate or duration of the traffic whether on the destination or source. It is also observed that there are few categorical features compared to numerical features.

Feature Visualizations:

Figure 13 illustrates the distribution of protocol types within the KDDcup99 cleaned dataset. It can be observed that most connections follow the TCP protocol type, followed by UDP, and ICMP.

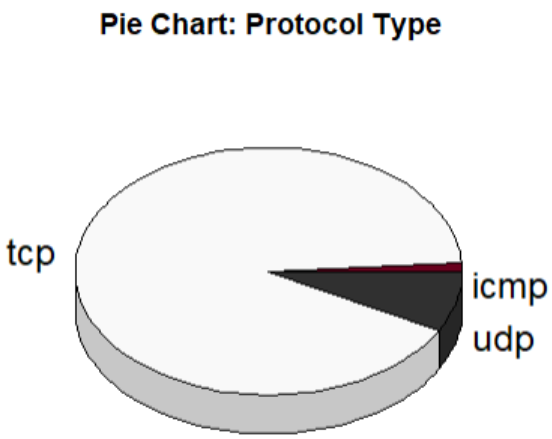


Figure 13 Protocol Types Pie Chart

There are many types of services that reside within the KDD'99 dataset. Some which are majorly present such as http, followed by private, smtp, domain_u, ftp_data, and others.

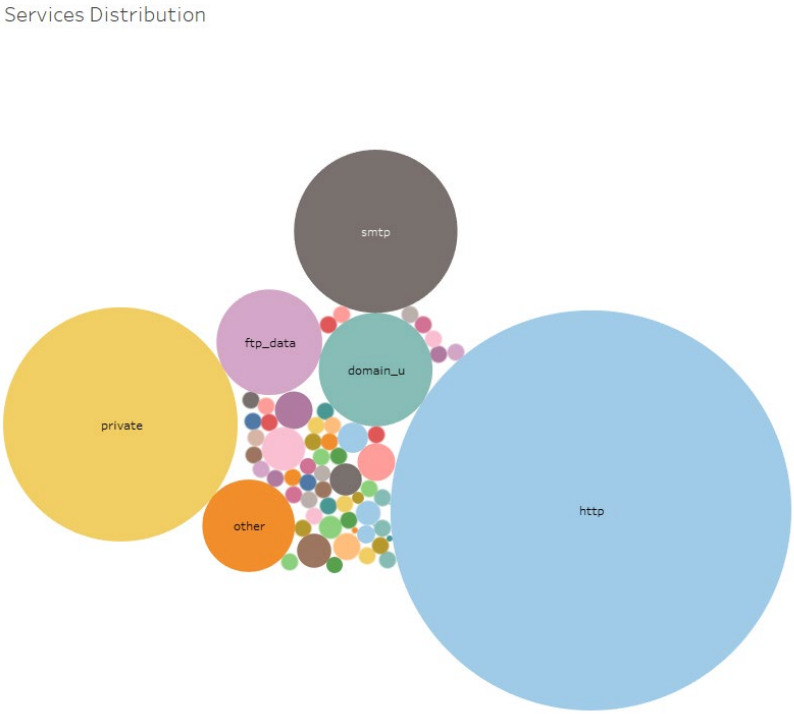


Figure 14 Services Bubble Graph

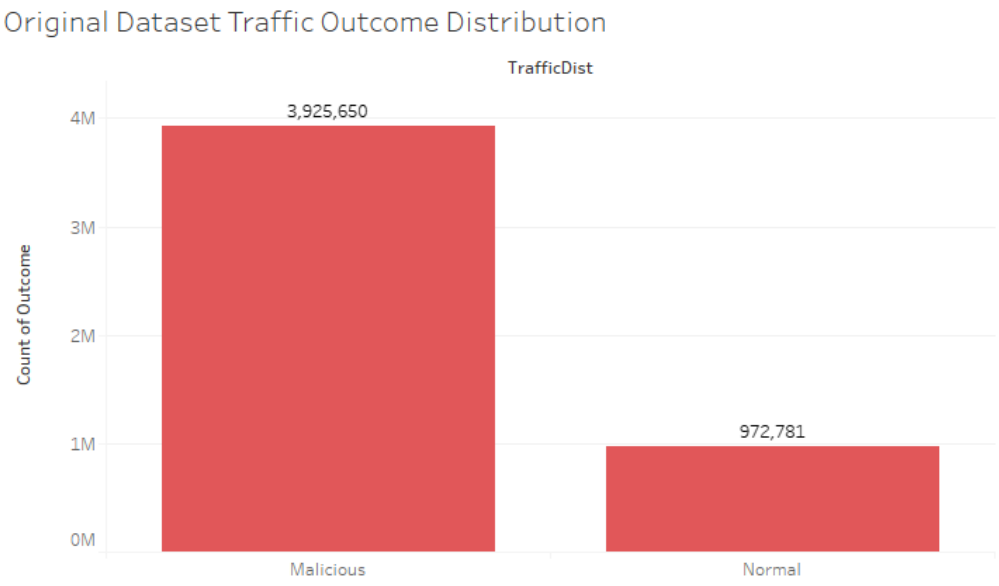


Figure 16 Original Traffic Distribution

We can observe that the original uncleaned dataset contains more malicious traffic compared to normal traffic. The distribution shows us that the majority of the dataset contains malicious traffic which is unusual if we are to compare the dataset with an exert of any real-life traffic. While it is not impossible to have such irregular distribution, it is highly unlikely to expect this in a normal environment.

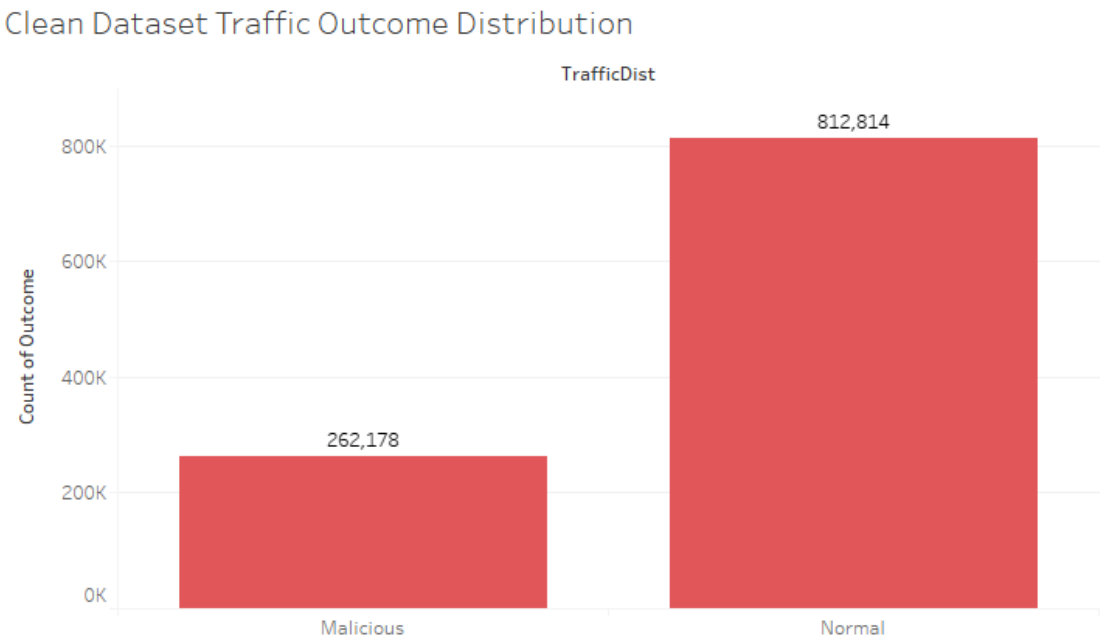


Figure 15 Clean Traffic Distribution

The cleaned the dataset gives us the insight that the dataset contained a huge percentage of duplicate records. We can infer from the new distribution that most of the redundant records had a malicious label. After cleaning the dataset, the ratio between malicious and normal traffic differed. The number of normal traffic records now far exceeds the number of malicious traffic. While it was not intentional, it can be said that the cleaned dataset now more resembles how the traffic distribution would look in a normal day to day traffic log where it would be expected that the number of good connections exceed the number of bad connections.

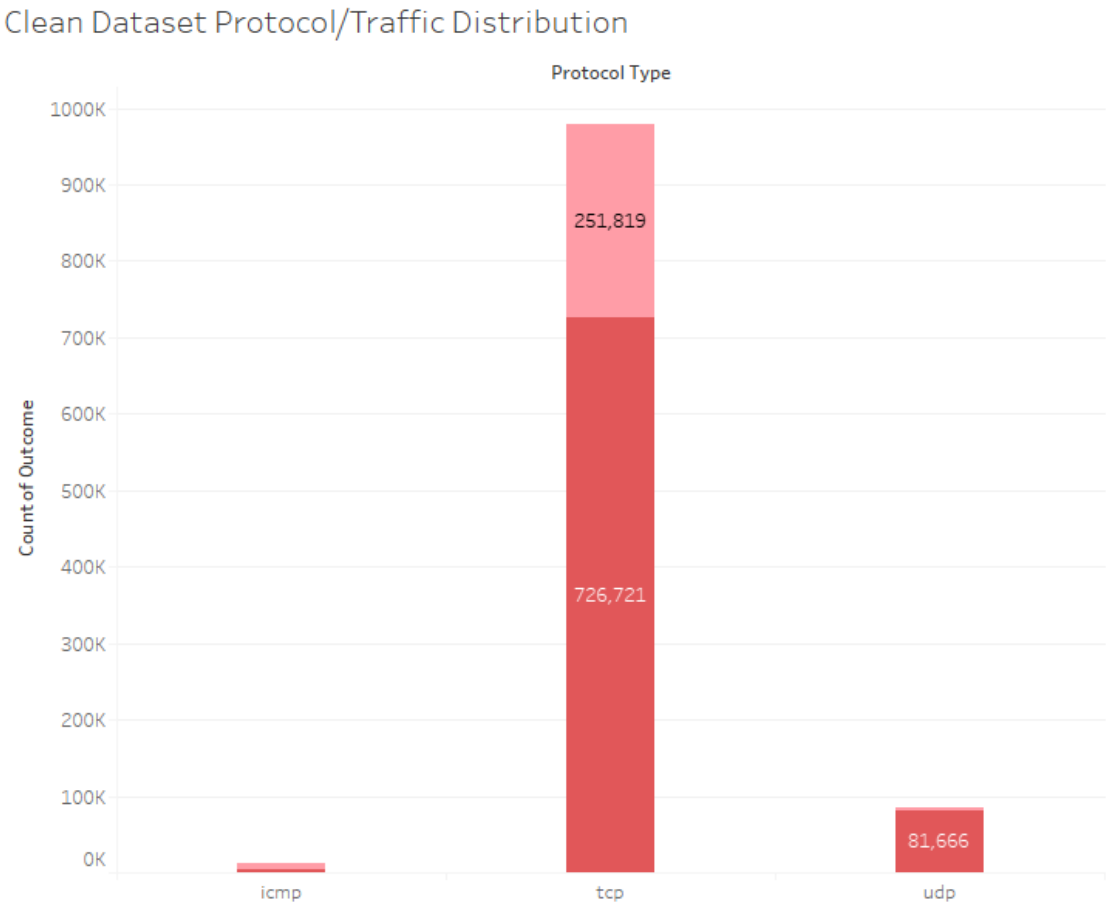


Figure 17 Clean Traffic Distribution via Protocol

Grouped by the protocol types, we can see that most of the normal traffic follows TCP followed by UDP, and ICMP. Similarly, we can observe that the number of malicious traffic follows TCP, followed by ICMP, and UDP.

By exploring the distribution of both the old and the new cleaned dataset, it was obvious that the dataset comes under what would be considered as an imbalanced dataset. While it resembles normal traffic, it still stands that the distribution is highly skewed to one label compared to the other. This might create several issues later on when it comes to testing and modeling.

It can also be observed on Table 6 considering the different times of labels, that normal traffic far exceeds the presence of bad/malicious connections in the dataset.

Table 6 Traffic Ratio Distribution

Traffic Ratio Distribution

TrafficDist	Outcome	
Malicious	back.	0.09%
	buffer_overflow.	0.00%
	ftp_write.	0.00%
	guess_passwd.	0.00%
	imap.	0.00%
	ipsweep.	0.35%
	land.	0.00%
	loadmodule.	0.00%
	multihop.	0.00%
	neptune.	22.53%
	nmap.	0.14%
	perl.	0.00%
	phf.	0.00%
	pod.	0.02%
	portsweep.	0.33%
	rootkit.	0.00%
	satan.	0.47%
	smurf.	0.28%
	spy.	0.00%
	teardrop.	0.09%
	warezclient.	0.08%
	warezmaster.	0.00%
Normal	normal.	75.61%

Table 7 gives us a detailed insight into the distribution of all connection types and their labels grouped by protocol types. Normal traffic is observed to be in abundance in connections that uses UDP, and TCP while other malicious connections can be observed to have exceeded the number of normal connections in connection that uses ICMP. We can also observe that certain probing actions or attacks only occur to specific protocol types such as Neptune which can only be observed in traffic that uses ICMP.

Table 7 Protocol/Outcome Count

Clean Dataset Protocol/Outcome Count

Protocol Type	Outcome	
icmp	ipsweep.	3,223
	nmap.	1,007
	normal.	4,427
	pod.	206
	portsweep.	5
	satan.	32
	smurf.	3,007
tcp	back.	968
	buffer_overflow.	30
	ftp_write.	8
	guess_passwd.	53
	imap.	12
	ipsweep.	500
	land.	19
	loadmodule.	9
	multihop.	7
	neptune.	242,149
	nmap.	297
	normal.	726,721
	perl.	3
	phf.	4
	portsweep.	3,559
	rootkit.	7
	satan.	3,279
	spy.	2
	warezclient.	893
	warezmaster.	20
udp	nmap.	250
	normal.	81,666
	rootkit.	3
	satan.	1,708
	teardrop.	918

The distribution of both normal, and bad connections is visually illustrated below:

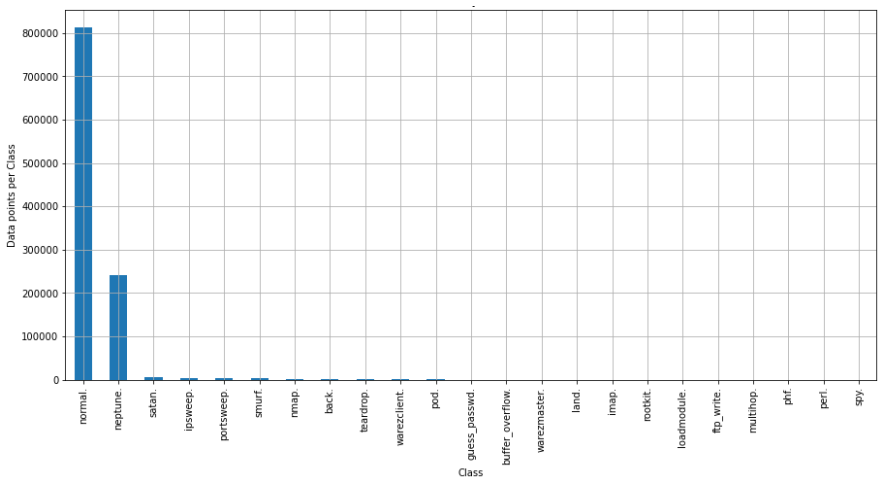


Figure 18 Connection Distribution

4.5 Modeling and Testing

To validify our proposed solution we shall be testing it against other solutions.

1. Isolation Forest: is an unsupervised machine learning method that is based on random forest and decision trees. It identifies outliers rather than inliers.
2. Local Outlier Factor: is a semi-supervised machine learning method that compares the density of one point to the density of another to highlight how likely a point is an outlier or not.

Feature Importance

Prior to testing, the aim was to reduce the computational expense. To do achieve a lower computational expense, feature importance calculations were conducted.

The KDD'99 dataset contains 41 features which means that our dataset suffers from the curse of dimensionality. To handle this issue, DecisionTreeRegressor package was utilized to retrieve the feature_importance_ property. Decision trees calculates the importance scores based on several criterions such as Gini or Entropy.

Table 8 Feature Importance Calculations

Feature	Score
0	0.08085
1	0.00577
2	0.01319
3	0.01474
4	0.00204
5	0.08090
6	0.04450
7	0.00005
8	0.04475
9	0.00011
10	0.00908
11	0.00101
12	0.00015
13	0.04000
14	0.00009
15	0.00004
16	0.00004
17	0.00040
18	0.00002
19	0.00000
20	0.00000

Feature	Score
21	0.00000
22	0.00149
23	0.10853
24	0.00036
25	0.00000
26	0.00012
27	0.00147
28	0.00004
29	0.06019
30	0.00293
31	0.00006
32	0.00314
33	0.00445
34	0.00595
35	0.00727
36	0.00499
37	0.05151
38	0.39352
39	0.00033
40	0.01446
41	0.00149

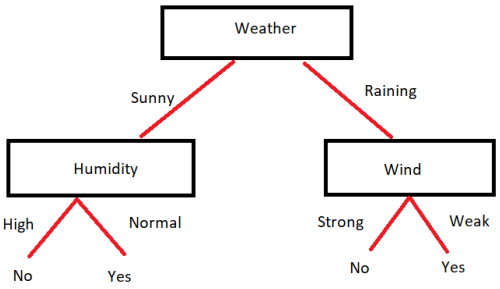


Figure 19 Decision Tree Example

A decision tree is a supervised machine learning technique that is used to make predictions. It is made up of a root node, branches, and contain leaves. It is easy to read or analyze since it closely mimic human thinking. A feature is usually visualized by a node, a branch which demonstrates a rule and leaves that depicts and outcome.

Feature	Score
0	0.08085
2	0.01319
5	0.08090
6	0.04450
8	0.04475
13	0.04000
23	0.10853
37	0.39352
40	0.01446

Figure 21 Top Features

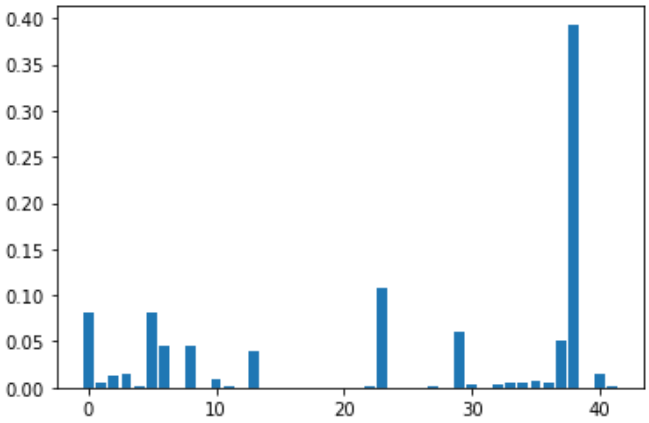


Figure 20 Feature Importance Visual

After fitting the model, the coefficient value for each feature can be retrieved. Out of the 41 features, only 9 with the highest coefficient value will be chosen prior to testing. This will not only enhance the results but will also reduce the computational expense of conducting analysis, fitting, and modeling later.

Classifier Score

After fixing the imbalanced dataset, we utilized the test-train-split package in python and their machine learning packages to calculate classifier scores for various models to compare against our proposed local outlier factor solution:

Table 9 Classification Score Table

Label	Model	Classifier Score
Normal	Isolation Forest	0.5127820807843785
	Local Outlier Factor	0.702128654280069
Outlier/Anomaly	Isolation Forest	0.48723947553244434
	Local Outlier Factor	0.8631349693719534

Classification Report

The classification report contains several metrics that are important to evaluate any model. The metrics that are included are accuracy, precision, recall, and F1.

- Accuracy: it is the ratio of correct predictions against total observations.
- Precision: is the ratio of correct positive predictions against total positive observations.
- Recall: which is also known as sensitivity, is the ratio of correct predictions of positive observations against all observation in a class (actual).
- F1: is the average of the recall and precision scores.

The sckitlearn packages comes within default parameters. The default parameters have resulted in undesired results. After conducting some hyperparameter tuning by trying different combinations

of parameters using GridSearchCV, and RandomSearchCV we were able to get the following results:

Table 10 Classification Bad Connections Report Table

Metric	Isolation Forest	LOF
Accuracy	0.74	0.74
Precision	0.77	0.75
Recall	0.94	0.96
F1	0.84	0.84

Table 10 shows different scores but as we are aiming for network anomaly prediction the most important feature that the research will use as validation is how well the model correctly predict positive observations against all observations is in an actual class. The proposed solution resulted in a score of 0.96. While the proposed solution is decent in correctly predicting normal traffic, it is highly effective in predicting anomalies and outliers.

Inferences and Points of Interest:

To achieve the above results, we had to compute the threshold at which any point above it would be considered as an outlier. Since the dataset labeled the traffic, we were able to calculate the threshold to fine tune our algorithm. To implement this on a live environment we need to consider that new data points will be added which means the densities and the Local outlier score could range quite differently thus forcing the threshold to be recomputed.

Chapter 5 Conclusion

5.1 Conclusion

Local outlier factor analysis results into a high prediction accuracy regarding anomaly detection compared to normal detection. The proposed solution has fared well against other models, but it seems that it is highly affected by the curse of dimensionality. Further tuning is needed, and a better dataset is required to produce better and accurate results. The cleaned dataset was imbalanced at the start as well. The LOF required a lot of tuning in regard to trying different hyperparameters to produce the required results, but it should fare much better when tested against other datasets. We were able to determine that the local outlier factor model was able to properly detect anomalies, and outliers even when compared to other models. The machine utilized was quite weak in computational resources, so it was not possible to calculate the duration of detections due to having forced the models to use all available processors, which is not the case in a real-life case.

5.2 Recommendations

Further hyperparameter tuning will increase detection rate and reduce possible false positives. Contamination value was calculated during this research, but this needs to be considered if new datasets are introduced. Testing random combinations or a specified set of parameters is recommended to test if the proposed solution will fair well against other dataset that might imitate other environments that are not necessarily to the environment produced after the data cleaning.

To reduce further computational expense, it is recommended to reduce the number of features either exported from network logs or simulated as per results of the paper's feature importance operation or any similar operation using any other appropriate model. The research used Decision Trees to retrieve the feature importance coefficients, but it is also possible to derive such values from other models such as Random Forest.

5.3 Future Work

Further research should aim for trying advanced or newly emerged LOF techniques such as genetic -based incremental local outlier factors and density summarizations incremental local outlier factor with different subsets of the same dataset. It is also recommended to use other intrusion dataset since it was discovered that the KDD dataset is an imbalanced dataset. Further assessment should go into calculating feature importance across different dataset which may or may not have similar features which may help to narrow down into a much more universal approach in regard to feature selection and focus. It was unfortunate that the research could not provide accurate durations of the model fitting and prediction process, but this is all due to computational limitations. For future research this should be considered, and an appropriate machine should be used to avoid forcing the model to use all available processors as we did during this research. It will provide a more accurate results into the duration of detection.

Bibliography

Alghushairy, O., Alsini, R., Soule, T., & Ma, X. (2020). *A Review of Local Outlier Factor Algorithms for Outlier.* MDPI.

Aljanabi, M., Ismail, M. A., & Ali1, A. H. (2020). Intrusion Detection Systems, Issues, Challenges, and Needs. *International Journal of Computational Intelligence Systems*, 560-571.

Beigh, B. M., Bashir, U., & Chachoo, M. (2013). Intrusion Detection and Prevention System: Issues. *International Journal of Computer Applications*, 26-30.

Bhuyan, M. H., BhattacharyyaJ, D. K., & Kalita, K. (2011). NADO : Network Anomaly Detection Using Outlier. *ICCCS* (pp. 531--536). ACM.

Cisco Press. (2009). Network Security Using Cisco IOS IPS. *Cisco Press*.

Downs, F., & Brewer, D. (2020). Top Cyberattacks of 2020 and How to Build Cyberresiliency. *ISACA*.

Gogoi, P., Bhattacharyya, D. K., & and, B. B. (2011). A Survey of Outlier Detection Methods in Network Anomaly. *Comput. J.*, 570--588.

Gupta, U., Gupta, S., & Bhattacharjee, V. (2019). Intrusion Detection System using Outlier Analysis. *HBRP*, 10.

Holt, T. J., & Bossler, A. M. (2013). An Assessment of the Current State of Cybercrime Scholarship. *Deviant Behavior*, 20-40.

J, J., & Dr.B.MUTHUKUMAR. (2015). Intrusion Detection System (IDS): Anomaly Detection using Outlier. *Procedia Computer Science*, 338-346.

Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity 2*.

Lazarevic, A. C., Ertoz, L., Ozgur, A., Kumar, V., & Srivastava, J. (2003). A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection. *Third SIAM International Conference on Data Mining* (pp. 25-36). San Francisco: SIAM.

Lee, W., & Stolfo, S. J. (1998). Data Mining Approaches for Intrusion Detecti. *7th USENIX Security Symposium* (pp. 1-16). San Antonio: Columbia University.

Morgan, S. (2020). *Cybercrime To Cost The World $10.5 Trillion Annually By 2025.* Cybercrime Magazine.

W.Creswell, J. (2018). In J. W. Creswell, & J. D. Creswell, *Research: Design: Qualitative, Quantitative, and Mixed Methods Approaches.* SAGE Publications, Inc.