Exploring Cybersecurity Data Science

Dimensionality Reduction, Clustering, and Anomaly Detection

Muuzaani Nkhoma

May 2024

Abstract

Cybersecurity incidents have become the norm of the day as there is hardly a day without news of a breach or discovery of illegally obtained data on the dark web. This is despite the presence of large amounts of data collected by various devices that are used in the protection of data in information systems. With this high volume of network data that is available through several logging systems, human analysts become overwhelmed to manually analyze the data. Even though the data flagged by the monitoring devices is what is thought to be malicious, a great portion of this data is comprised of false positives. Most of the time these analysts are cybersecurity professionals with little data analytics/science knowledge. If analysts with data analytics/science knowledge are used, they will usually have little cybersecurity knowledge. To aid with finding insights as well as problematic issues from the data, there is an increasing use of data mining/data science, machine learning, deep learning, and artificial intelligence methods. The users of these methods should therefore be knowledgeable of both data analytics/science and cybersecurity concepts and methods. In this paper we will look at how using data analytics techniques can aid in the analysis of cybersecurity data by performing a comparison of dimensionality reduction techniques, and of clustering techniques on some cybersecurity datasets.

Contents

List of Tables

List of Figures

1 Introduction

The rapid advance of computer and communication technology has resulted in a vast amount of data being created and consumed at a rate that is unprecedented. This has created the phenomenon known as 'Big Data' commonly identified by the four V's namely volume, variety, velocity, and veracity. This in turn has resulted in vast amounts of traffic moving in our computer networks. Cybercriminals are also not left behind. They are embracing the new and smart technologies with the same or even more determination than the rest of society. This has been made possible and hard to fight due to the sophistication, coordination, and determination of Advanced Persistent Threats (APTs). APTs are difficult to detect, to prevent, and to remove.

There are now many devices and systems that help in the protection of computer and information systems, and these devices produce vast amounts of monitoring information through logging systems. Most of the flagged data consists of false positives. "The biggest problem in cybersecurity is not better endpoint detection but how to enable the analyst to keep pace with the sheer volume of alerts being generated" [1]. There is also an acute shortage of skilled cybersecurity professionals worldwide, according to the (ISC)² Blog post [2], the cybersecurity skills shortage is nearing 3 million. This shortage means that cybersecurity professionals are overwhelmed and cannot cope with the current workload. These are all sources of vulnerabilities. "The complexity and scale of digital infrastructure has rendered rules-based methods almost ineffective in their ability to detect attacks and hence presents a big challenge" [3]. "Machine learning outperforms human analysts when it comes to recognizing and predicting specific patterns due to the analysts reliance of manual investigation and decision-making" [4].

These are some of the reasons motivating researchers to elevate the need for incorporating new smart technologies like artificial intelligence. "Security researchers believe they can utilize attack pattern recognition or detection methods to provide protection against future" attacks [4]. "By developing systems that utilize artificial intelligence (AI) and machine learning, researchers expect to create very sophisticated defense models that use new technologies like big data, pattern mapping and matching, cognitive computing, and deep learning methods that simulate the way human mind works" [5].

Already many organizations are using AI to help them in their cybersecurity efforts where AI and ML can provide insights that would otherwise be impossible for humans to achieve alone. "Intelligent decision-making utilizing machine learning technology to achieve automation has become possible" [4].

But cybercriminals are not left behind as AI is already being used for nefarious ends by hackers and other cybercriminals [6]. Whether it is DDoS attacks, ransomware, social engineering or some other kind of malware, cyber criminals are using AI to spread the threat faster and target more vulnerable machines and individuals. "Generative AI tools

enable cybercriminals to automate and scale their attacks, taking a task that used to require five minutes and turning it into five seconds. Making matters worse, ChatGPT and other tools make it nearly impossible to differentiate between real emails and malicious ones, as it eliminates the telltale signs of an attack like grammar mistakes and spelling errors" [7]. This is another challenge with cybersecurity that requires the use of sophisticated methods to counter them. The challenge is that cybercriminals are human beings with intelligence who are aware of the advances in technology and the opportunities that advanced technology presents and are working hard to defeat our systems by using the same tools that are available to defenders to enhance their capabilities.

In this paper we will first provide an overview of cybersecurity data science based on literature review in section 2. Since usually cybersecurity data will be multidimensional and its visualization can be a challenge, a comparison of various dimensionality reduction techniques will be conducted and followed by a comparison of clustering techniques. Both these techniques are compared using two cybersecurity datasets in section 3. We finally end with the conclusion in section 4.

2 Cybersecurity

2.1 OSI, TCP/IP Protocols and the Protocol Data Unit (PDU)

In any communication system we have a communication model (Fig.1) that represents the exchange of data between two entities. Data can be exchanged between two devices (sender and receiver) via some form of transmission medium such as a wire cable.

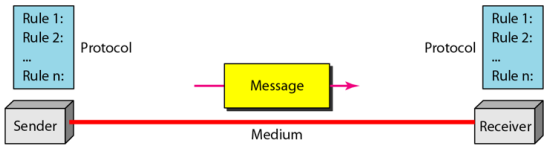
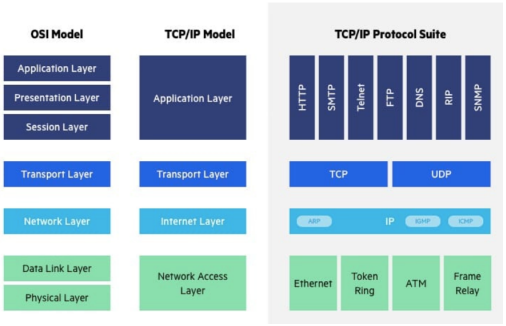


Figure 1: Communication Model

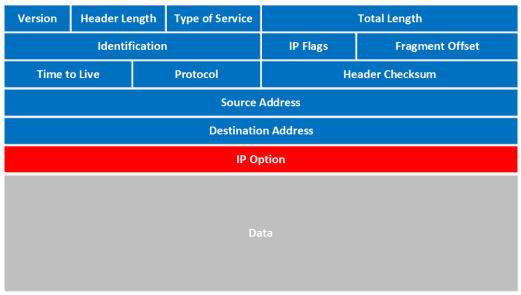
There are five components of data communication namely sender, receiver, message, transmission medium, and the protocol.The network protocol is defined as "an established set of rules that determine how data is transmitted between different devices in the same network" [8]. These protocols have a layered architecture. There are two network protocols that are mainly in use, and these are the TCP/IP model suite and the OSI reference model, Fig 2(a).

When data is moving between the layers, headers are added/removed to the data and a protocol data unit (PDU) is formed from the process. This process of adding/removing

headers/trailers from as the PDU travels between the layers is known as encapsulation/de-encapsulation. The information contained in the headers is very important as it represents what and how each PDU is behaving as it moves within the network. The packet is the network layer PDU, Fig. 2(b).



(a) OSI vs TCP/IP Model. . Source: [9]



(b) IPv4 Packet

Figure 2: Network protocols and IPv4 packet

2.2 Cyberattacks

Stallings presents NIST's definition of computer security as "the protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability, and confidentiality of information system resources (includes hardware, software, firmware, information/data, and telecommunications)" [10]. The three concepts of confidentiality, integrity, and availability in computer security are commonly referred to as the CIA triad. Confidentiality deals with authorization, integrity deals with maintaining unaltered information from send to receiver, and availability means that information and its resources can always be accessed when needed.

According to [11], information security is, broadly, the practice of securing your data, no matter its form, and cybersecurity is a subset of information security that deals with protecting an organization's internet-connected systems from potential cyberattacks; moreover, network security is a subset of cybersecurity that is focused on protecting an organization's IT infrastructure from online threats.

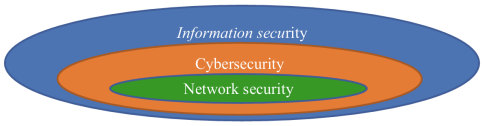


Figure 3: Information security vs. cybersecurity vs. network security. Source: [12]

Every information system will have vulnerabilities and threats. Vulnerabilities are weaknesses within the system and threats are activities that represent a possible danger to the system. Having a threat that matches a vulnerability presents a risk to the system. If the threat manages to exploit the weakness, then loss of confidentiality, integrity, and availability is experienced.

NIST defines a cyberattack as any kind of malicious activity that attempts to collect, disrupt, deny, degrade, or destroy information system resources or the information itself [13]. Thus, a cyberattack is any activity that compromises the confidentiality, integrity, and availability of information systems.

Social Engineering	Malware	Password Attacks	Denial of Service	Advanced Persistent Threats	Database/ browser/ Software	Mobile Ad hoc networks	Cyberphysical systems
Phishing	Virus	Brute force	Distributed denial of service	Hactivism	SQL injection	Byzantine	Covert
Baiting	Spyware	Dictionary	Botnet	Commodity threat	Logic bomb	Blackhole	Resilient control
Quid Pro Quo	Worms		Buffer overflow	Cyber espionage	Cross-site scripting	Flood rushing	Replay
IVR or Vishing or Phone Phishing	Adware		Teardrop	Indirect attack	Man in the Middle	Byzantine wormhole	
Eavesdropping	Rootkits		Smurf		Tampering	Byzantine overflow network overflow	
Spoofing	Key logger		Physical			Blue snarfing	
Direct Access	Backdoors		Exploits			Blue jacking	
Identity theft	Trojan horses		Privilege escalation				
Repudiation Attack	Ransomware						

Figure 4: Types of cyberattacks.

Janeja provided a consolidation of cyberattacks (Fig.5) based on the general characteristics of the attacks into the following eight major categories: *social engineering* – relies on understanding the social interactions of the individual and trying to gain access to user credentials; *malware* – includes a broad range of software threats that exploit various network, operating system, software, and physical security vulnerabilities to spread malicious payloads to computer systems. It is a class of attacks that install a malware, such as virus, spyware, Trojans, ransomware etc.; Many forms of malicious code take advantage of zero-day vulnerabilities, security flows discovered by *password attacks* – that focus on getting user credentials either through brute force or by checking the passwords against a dictionary of words; *denial of services (DoS)* – that occur when malicious users tend to block legitimate traffic by sending too many requests to a server; *advanced persistent threats (APTs)* – perpetrated through coordinated long-term attacks and mostly originate from organization or state actors with long-term interests in the assets on the hacked system; *database/browser software* – caused due to bugs in the software such as in the case of a SQL injection attack where the front-end user form can be used to initiate a query request to the back-end database; *mobile ad hoc networks* – where information flow is disrupted; and

cyberphysical systems – attacks such as replay attack , where valid data are sent maliciously but repeatedly with the intent to cause delay or block traffic [14].

A type of malware that has become very common nowadays is called *ransomware.* It is a type of malware that weaponizes cryptography where once a system has been infected with malware, critical data on the systems drives is encrypted and rendered inaccessible to users. The attackers will be the only ones having the encryption key and will demand payment, ransom, to be paid before access can be granted to the owners [15].

Zero-day attack is considered as the term that is used to describe the threat of an unknown security vulnerability for which either the patch has not been released or the application developers were unaware [16].

The main steps of attacks include Information Gathering, Attack and Penetrate, Local Information Gathering, Privilege Escalation, Pivoting, and followed by Cleanup. Information Gathering involves the gathering of relevant information about the target network including active IP addresses, operating systems, and available services. Using the information gathered, the attacker then launches remote exploits on the victim system. An exploit is software that takes advantage of a vulnerability. Once the attacker has successfully penetrated the system, they can then collect local information from within the compromised system. In the privilege step, the attacker tries to obtain high level privileges like administrator privileges. If successful, the high-level privileges can allow the attacker to run malicious code from within the system. The last step in an attack is to try and clean up so that there is as little trails as possible and thus make it difficult to be traced [17].

2.3 Sources of Cybersecurity Data

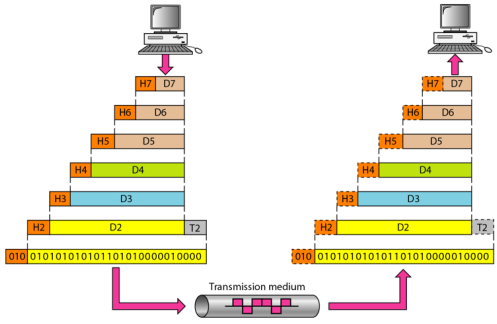


Figure 5: Data exchange using OSI model

From Figure 6, we see that a message exchanged between two devices will undergo various processes and will move through different layers regardless of whatever network architecture is used. The initial message that is sent through the Application Layer is broken into manageable segments in a process known as segmentation. These segments consisting of

raw data (payload) are combined with headers and/or trailer as they travel down the architecture and this process is known as encapsulation until they are sent as binary digits (bits) in the Physical Layer. These bits are transmitted through various transmission mediums until they reach their destination. Once they reach the next device, the bits are combined into a frame at the Data Link Layer and each header/trailer is removed and the PDU checked for further instructions. The process of removing the header/trailer as the PDU goes up the architecture is known as de-encapsulation.

All along its journey, the data passes through various devices that keep track of details in the PDU. This provides opportunities for end-to-end data collection. The information collected is very useful and enables the analysis of the system.

Verma and Marchette provided a list of cybersecurity datasets that are typically collected to be network traffic data, malware data, static and dynamic information, phishing data, obfuscated commands, authentication logs, audit logs, event logs, VPN logs, suspicious domain names, and CVE: Common Vulnerabilities and Exposures [18].The above data sources can be viewed as raw payload data, network topology data, access control data, and vulnerability data. Payload data consists of the actual message (data) being exchanged and together with header and/or trailer information (control information) make up the PDU. When permission is granted to access payload data, it can be used in multiple ways, such as to discover an individual user's behavior, the presence of malwares in the payloads, and other security threats that can be detected based on the actual content of the payload" [14]. Payload data can be accessed by packet sniffer tools like Wireshark and Snort. A network logical topology can be represented as graph with end point being represented by vertices. "Network traffic data dump can be used to generate the communication graph of all exchanges taking place over the network" [19]. We can analyze how certain vulnerabilities are affecting different systems over by using vulnerability data from the National Vulnerability Database provided by NIST [20].

The collected data can be analyzed individually though combining various data sources can provide some great insights using data mining methods.

3 Exploratory Data Analysis (EDA) through Dimensionality Reduction and Clustering Analysis in Cybersecurity

3.1 Overview

Since most of the datasets will have more than two variables (features) i.e. they are multivariate datasets, it is almost impossible to visualize them in more than three dimensions. Therefore, there is need to reduce the dimension of the data, i.e. transforming the data from a high-dimensional space to a low-dimensional space, and the procedures involved are known as dimensionality reduction techniques. Dimensionality reduction plays an important role

in data science, being a fundamental technique in both visualization and pre-processing for machine learning to avoid the curse of dimensionality [21].

Clustering can be described as finding instances whose feature values are most similar and grouping them together. The goal of cluster analysis is to group objects or individuals into homogeneous clusters such that objects or subjects in a given cluster are more similar to one another than objects or subjects in a different cluster [22].

In clustering, we try to group data points that are more similar to each while separating groups with data points that are very dissimilar. In anomaly detection, we try to find data points that struggle to belong to any of the groups. Anomaly detection refers to the problem of finding patterns in data that do not conform to expected behavior [23].

3.2 Datasets Used

The datasets that have been analyzed in this project are NSLKDD [24] and UNSW-NB15 [25]. Both are synthetic network traffic datasets generated through simulations.

NSL-KDD data set has forty two features including one target feature *label*. The forty one features are of different data types. The target feature has forty different attack types, levels. They can be grouped in two different ways. The first one categorizes them into representing whether the network packet was from normal traffic or it was an attack. The attack records are further classified into four subcategories according to the nature/type of the attacks namely Denial of Service (DoS), Probe, User to Root(U2R), and Remote to Local (R2L). Probe or surveillance is an attack that tries to get information from a network. U2R is an attack that starts off with a normal user account and tries to gain access to the system or network, as a super-user (root). The attacker attempts to exploit the vulnerabilities in a system to gain root privileges/access. R2L is an attack that tries to gain local access to a remote machine. An attacker does not have local access to the system/network and tries to "hack" their way into the network [26].

UNSW-NB15 data set has forty four features including two target features, One is the main feature that categorizes the network traffic into normal or malicious. A the second is the feature where the attack records are further classified into ten categories with nine of them being subcategories according to the nature/type of the attacks. These subcategories are *Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode* and Worms. Fuzzing is when malicious hackers try to find vulnerabilities. *Fuzzers* attempt to cause a program or network to be suspended by feeding it randomly generated data. *Analysis* contains different attacks of port scan, spam, and html files penetrations. *Backdoors*, a technique in which a system security mechanism is bypassed stealthily to access a computer or its data. *Exploits*, when the attacker knows of a security problem within an operating system or a piece of software and leverages that knowledge by exploiting the vulnerability. *Generic* - a technique that works against all block ciphers (with a given block and key size),

without consideration about the structure of the block-cipher. *Reconnaissance* - Contains all strikes that can simulate attacks that gather information. *Shellcode* - A small piece of code used as the payload in the exploitation of software vulnerability. *Worms* - Attacker replicates itself in order to spread to other computers. Often, it uses a computer network to spread itself, relying on security failures on the target computer to access it [25].

3.3 Data Preprocessing

Frequently, most data collected will not be in a format or form that can be analyzed right away. Some of the data quality issues might be missing values, outliers, duplicate data, features that have different scales and ranges etc. Therefore, data needs to be cleaned and transformed when necessary. Another issue is that most of the algorithms easily work with numerical data as their input data and not categorical data. Categorical data requires some transformation.

For cybersecurity data, most of the network data is captured from the monitoring systems and therefore the problem of missing data is uncommon. But the data mostly consists of mixed data type features. We have some features that are nominal, logical, and numerical. And the nominal data type are very important as they collect the type of protocol and service used in addition to the state of the flags in the PDU which can lead to the determination of cyberattack types. To enable the nominal features to be analyzed together with the numerical feature several approaches are available. We can transform the nominal features into numerical features by transforming the categories into factors that have numerical values, we can transform the nominal features through one-hot encoding with dummy variables, similarity measures like the Gower distance [27], and FAMD [28]. The Gower distance works by computing the different data types using different metrics with numerical using distance measures like the Manhattan distance and nominal using the Dice coefficient [29]. The numerical features were standardized to have mean 0 and variance 1.

3.4 Dimensionality Reduction

There are two main approaches to dimensionality reduction: projection and manifold learning. Manifold-learning methods attempt to find a sub-space in which the high-dimensional distances can be preserved [30]. In projection, every data point in a high-dimensional space is projected onto a lower-dimensional space in such a way that distances between points are approximately preserved. There are various techniques employed in dimensionality reduction and they can be categorized into two namely linear and nonlinear dimensionality reduction methods [31]. The various methods can be used according to the different use cases as needed but usually linear methods preserve global structures and nonlinear methods do better representing local structures. Some of the linear methods are principal components analysis (PCA), correspondence analysis and multiple correspondence analysis (CA/MCA), principal coordinate analysis (PCoA) which is also known as classical (metric) multidi-

mensional scaling (cMDS) while the nonlinear methods are nonmetric multidimensional scaling (nMDS) [32][42] , isometric feature mapping (ISOMAP) [33], neighbor embedding techniques like the t-distributed stochastic neighbor embedding (t-SNE) [34], and uniform manifold approximation and projection (UMAP) [21]. Some of the methods were originally designed for numerical data types (PCA) while others for nominal/qualitative/categorical data (CA/MCA). CA is a graphical way to represent associations in two-way contingency tables [35]. Therefore, transformation of categorical data through methods like one-hot encoding might be required when working with data that has nominal data. Computing distance/similarity matrices can also be utilized for methods that accept numerical input data only. Mixed data types can be used with methods like UMAP and Factor Analysis for Mixed Data (FAMD) [36]. FAMD involves applying PCA on numerical data and MCA on categorical data and combines the results by weighing variable groups [31].

PCA, FAMD, t-SNE, UMAP, MDS, and ISOMAP are the dimensionality reduction methods that were compared on the two cybersecurity datasets, UNSW-NB15 and NSL-KDD.

3.4.1 Comparison of Dimensionality Reduction Methods

PCA is a linear method that projects high-dimensional data onto a lower dimensional space. The new variables formed, known as principal components, are uncorrelated linear combinations of the original variables that maximize variance in the data or minimize the sum of squared errors. Essentially, the principal components are the eigenvectors computed from the covariance (correlation) matrix of the data. It uses singular value decomposition to get matrices of eigenvalues and eigenvectors. PCA can be used to identify patterns and to reduce the dimensions of the data.

$$J(\mathbf{W}) = \frac{1}{m}\sum_{i=1}^{m} \|\mathbf{x}^{(i)} - \mathbf{W}\mathbf{z}^{(i)}\|_2^2$$

FAMD uses PCA and MCA concepts in reducing the dimensions of data that has both numerical and categorical types. It works by trying to maximize the variance of the numerical variables and the projected inertia on the categorical variables. Let the data be The data include $\mathbf{K}$ quantitative variables $k = 1, \ldots, K$ and $\mathbf{Q}$ qualitative variables $q = 1, \ldots, Q$. z is a quantitative variable. Let $r(z, k)$ be the correlation coefficient between variables k and z: and $\eta^2(z, q)$ the squared correlation ratio between variables z and q. For PCA, we are trying to maximize the squared correlation coefficient

$$\sum_k r^2(z, k)$$

and in MCA we trying to maximize the squared correlation ratio

$$\sum_q \eta^2(z, q)$$

leading to maximizing

$$\sum_k r^2(z, k) + \sum_q \eta^2(z, q)$$

in FAMD [37] [28].

MDS is a technique whose purpose is to visualize similarity/dissimilarity in high-dimensional data. cMDS is a linear technique while nMDS is a nonlinear technique. Metric/classical MDS satisfies the triangle inequality while nMDS uses ranks and thus preserves the order of the distances. MDS techniques attempt to preserve the pairwise distances $d(i, j)$ of the input space in the output space to the greatest possible extent. It aims to find a projection of the data that minimizes the difference between the distances in the original space and distances in the lower-dimensional space. MDS uses gradient descent to minimize the stress function that best preserves the pairwise distances between the data points.

Metric/Classical MDS (cMDS) objective function:

$$\text{Stress} = \sqrt{\sum_{i<j}(d_{ij} - \hat{d}_{ij})^2}$$

$\hat{d}_{ij}$ are the fitted distances in the lower dimension that minimizes the stress function which is essentially a residual sum of squares and measures the goodness of fit.

Non-metrix MDS (MDS) objective function:

$$\text{Stress} = \sqrt{\frac{\sum_{i<j}(d_{ij} - \delta_{ij})^2}{\sum_{i<j} d_{ij}^2}}$$

δ_{ij} are the fitted distances that are in a strictly ascending order in the lower dimension that minimizes the stress function. That is in rank order.

ISOMAP, which can be viewed as an extension of MDS as it uses cMDS in its final step, seeks a lower-dimensional embedding that maintains geodesic distances between all points. It has three steps:i - construct a neighborhood graph; ii - compute shortest path by estimating geodesic distances between all points on the manifold; and iii - constructing a d-dimensional embedding. The final step applies classical MDS to the matrix of graph distances $D_G =$

$d_G(i,j)$, constructing an embedding of the data in a d-dimensional Euclidean space Y that best preserves the manifold's estimated intrinsic geometry. The coordinate vectors y_i for points in Y are chosen to minimize the cost function: [33]

$$\mathbf{E} = ||\tau(D_G) - \tau(D_Y)||_{L^2}$$

D_Y is the matrix of Euclidean distances in the lower dimensional space Y that minimizes the geodesic distances. The τ operator converts distances to inner products, and $||\mathbb{A}||_{L^2}$ is the L^2 norm or the largest absolute singular value of matrix $\mathbb{A}$.

t-SNE is a nonlinear method whose purpose is to visualize high-dimensional data in lower dimensions. It converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler (KL) divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. t-SNE has two main steps. The first step involves constructing a probability distribution over pairs of high-dimensional objects in a way where similar data points are assigned a higher probability and dissimilar data points get lower probabilities. Similarities are represented by conditional probabilities where for point x_i, the conditional probability for data point x_j, $p_{j|i}$, is higher if the points are closer and lower for widely separated data points. The joint probabilities in the higher dimension are normal distributions. The next step involves defining joint distributions in the lower-dimensional space where the distibutions are student t distributions. Gradient descent is then used to minimize the KL divergence between the probability distributions in the higher and lower dimensional space. [34]:

$$J(\mathbf{Y}) = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right)$$

Following the comparison in the next section, UMAP was the dimensionality reduction method that outperformed the other methods. We will then finally look at UMAP [21] and describe its properties and how it works.

UMAP is a manifold learning technique for dimension reduction whose design decisions were all grounded in a solid theoretic foundation and not derived through experimentation with any particular task focused objective function according to [21]. It is based on manifold learning techniques and ideas from topological data analysis. The algorithm is based on three assumptions:

1. The data is uniformly distributed on a Riemannian manifold;
2. The Riemannian metric is locally constant; and
3. The manifold is locally connected.

At a high level, UMAP works by constructing a high-dimensional graph representation of the data and then optimizes a low-dimensional graph representation to be as structurally similar as possible.

UMAP makes use of fuzzy topological representation through the topological structure of simplicial sets. Simplicial complexes are a means to construct topological spaces out of simple combinatorial components. Simplices are some simple building blocks. A simplicial complex is a set of simplices glued together along faces. Simplicial sets which are purely combinatorial, have a nice category theoretic presentation, and can generate a much broader class of topological spaces. A fuzzy topological representation of a dataset can be defined that provides a single fuzzy simplicial set as the global representation of the manifold formed by patching together the many local representations [16]. This is a brief theoretical view of UMAP.

From a computational view, a fuzzy simplicial complex is a representation of a weighted graph. This makes UMAP a k-neighbor-based graph algorithm. It has two major steps namely graph construction and graph layout (optimization). High-dimensional graph construction involves the building of a fuzzy simplicial complex which is a representation of weighted k-neighbor graph. In the second step, a low-dimensional weighted graph is constructed and through a force directed graph layout algorithm optimizes the edge-wise cross-entropy between the high-dimensional weighted graph and the low-dimensional weighted graph. This ensures that the low-dimensional graph is as similar as possible to the high-dimensional graph.

UMAP uses binary cross-entropy as a cost function and optimization of the embedding is achieved through minimization of the fuzzy set cross entropy. The final major component of UMAP.

$$
\begin{aligned}
C((A,\mu),(A,\nu)) &= \sum_{a\in A} \mu(a)\log\left(\frac{\mu(a)}{\nu}\right) + (1-\mu(a))\log\left(\frac{1-\mu(a)}{1-\nu}\right) \\
&= \sum_{a\in A} \mu(a)\log(\mu(a)) + (1-\mu(a))\log(1-\mu(a)) - \sum_{a\in A} \mu(a)\log(\nu(a)) \\
&+ (1-\mu(a))\log(1-\nu(a)) \\
&= -\sum_{a\in A} \mu(a)\log(\nu(a)) + (1-\mu(a))\log(1-\nu(a))
\end{aligned}
$$

Approximate stochastic gradient descent algorithm using probabilistic edge sampling and negative sampling minimizes the third line as all terms with μ which is constant as it represents the original high-dimensional representation are dropped. Nearest-Neighbor-Descent algorithm is used for efficient approximate k-nearest-neighbor computation and stochastic gradient descent (SGD) for efficient optimization.

UMAP has two main parameters namely n_neighbors and min_dist. n_neighbors controls the balance between local and global structure in the final projection. Low values concentrate on local structure and high values look at the larger neighborhood. The minimum distance between points in the low-dimensional space, min_dist, controls how tightly UMAP is allowed to pack points together. Low values produce much more tightly packed embeddings. High values pack points more loosely and focuses on broad structure.

3.4.2 Evaluation of Dimensionality Reduction Methods

Initially, we tried to use the entire data sets NSL KDD (148717) and USNW-NB15 (257673) with the dimensionality reduction methods. MDS and ISOMAP had runtime issues. NSL KDD data set sizes were gradually reduced and ISOMAP produced an output at n=120,000. Even at n=75,000 we were we unable to have output from MDS in a reasonable timeframe.

We then had a visual comparison of the five methods at n=75,000. UMAP produced a better output for the NSL KDD dataset and followed by ISOMAP while ISOMAP performed better for the UNSW-NB15 dataset and followed by ISOMAP see appendices A - D. ISOMAP had the longest runtime while PCA had the shortest runtime. Fig 9 shows the runtimes of the dimensionality reduction methods.

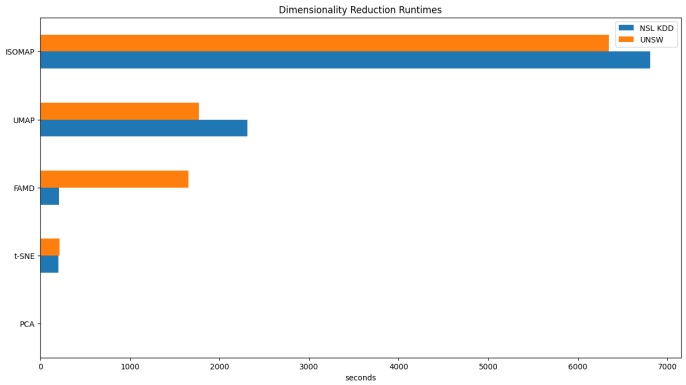


Figure 6: Dimensionality reduction methods runtimes.

Using the whole datasets, NSL KDD (148717, 41) and UNSW-NB15 (257000, 44), both ISOMAP and MDS were unable to produce outputs due to runtime issues. UMAP not only captured the most information as can be seen by how the different attacks are grouped together but it also performed better in terms of time and memory. MDS was the worst performing in terms of memory and time. Even after adding multiple processing python library *dask* and access to high-performance computing resources with 364GB RAM and 40 cores, the whole dataset could not be processed. MDS has time and space complexity of $O(n^3)$.

UMAP preserved both local and global structures, had acceptable runtime, no major memory issues, scalable to large data sets, and well-separated and fairly interpretable groupings (high visualization quality). Fig 10 shows the UMAP output for the fulle NSL KDD and UNSW-NB15 datasets. ISOMAP came second in capturing the most information but had scalability issues.

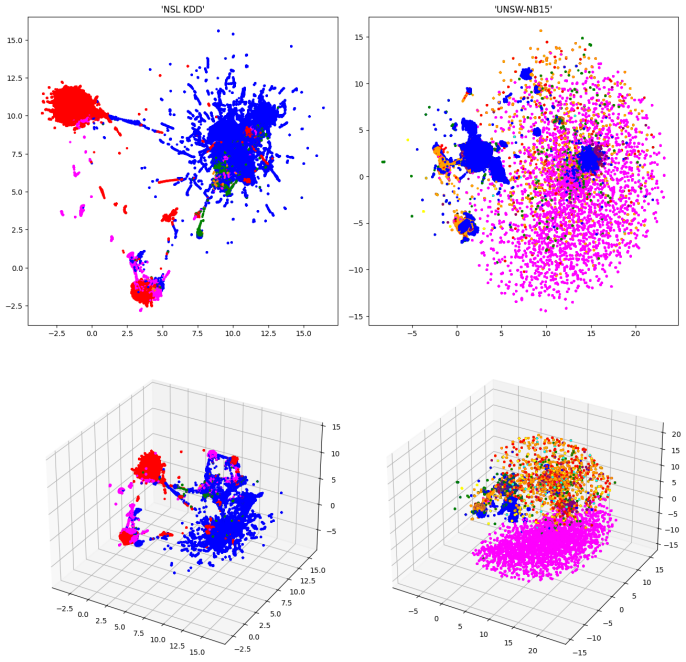


Figure 7: 2D and 3D UMAP outputs for the NSW KDD and UNSW-NB15 data sets.

3.5 Cluster Analysis

"In the area of cybersecurity, cyber-attacks like malware stays hidden in some way, including changing their behavior dynamically and autonomously to avoid detection. Clustering techniques can help to uncover the hidden patterns and structures from the datasets, to identify indicators of such sophisticated attacks" [16].

Once we have decided on the dimensionality reduction method that serves us well, there is need to extract the data points that belong to individual groups so that they can be investigated. As we have seen from the above visualizations that some attacks were nicely grouped, investigating data points by group might expedite in discovering attacks on our network. UMAP so far does not have a method that can directly export the data points by group. The deficiency of UMAP and ISOMAP dimensionality reduction methods is that you cannot directly extract the data points from the groupings in the lower dimensional

space. So, even though the techniques fairly grouped the data points into well separated groups, it will be impossible to analyze the network packets per group and determine if they are from malicious or normal traffic. One way to do that would be to use clustering algorithms and assign labels to the clusters. This is possible if the clusters formed are almost the same as the groups found with the dimensionality reduction techniques. But this is rarely the case.

We, therefore need to assign labels to our data. There are two ways of performing clustering analysis in this scenario. The first method is to use the original dataset or a pre-computed proximity matrix as the input to the clustering algorithm. The other way is to use the reduced, lower dimensional output of the dimensionality reduction technique as the input to the clustering algorithm.

We begin by distinguishing between partitioning and clustering. In partitioning, every data point is assigned a label while in clustering some data points might not be assigned a label. Methods like DBSCAN and HDBSCAN do not partition as points determined to be noise are not assigned a cluster.

The main types of clustering techniques are prototype-based techniques that cluster instances by some prototype like a proximity measure, density-based techniques that try to form clusters by separating regions of high density from regions of low density, and graph-based methods that use a proximity graph with each node being a data point and each edge being a weight that is a proximity measure between the two data points [38]. Examples of proximity (prototype) are k-means, k-prototypes, Gaussian Mixture Model (GMM). Examples of density-based methods include DBSCAN, HDBSCAN, OPTICS, SNN Density-based, DENCLUE. And graph-based methods include hierarchical measures (MIN, MAX, Average), CURE, CHAMELEON, BIRCH, and spectral clustering.

In choosing a clustering algorithm, the following factors must be taken into account as different algorithms perform differently: shape of the clusters (whether they are round, circular, spherical, or arbitrary), size (whether they are large, small, or varying), density (whether they are sparse, dense, or varying), whether we have nested/embedded clusters, and whether noise or outliers are present, and how large is our data set. Another area is whether the algorithm requires user-input of parameters like the number of clusters, etc. Most clustering algorithms suffer from the problems of difficult parameter selection, insufficient robustness to noise in the data, and distributional assumptions about the clusters themselves [39]. Algorithms that require user-specified parameters like number of clusters rely on domain knowledge about the number of clusters that are expected.

3.5.1 Comparison of Clustering Methods

k-means is a proximity-based technique whose objective function can depend on the proximity measure used, and for Euclidean measure the objective is to minimize the sum of

squared errors (SSE):

$$\text{SSE} = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \mu_i\|^2$$

The error is the distance to the nearest cluster center, $dist^2(C_i, X) = \|\mathbf{x} - \mu_i\|^2$. The centroid, C_i, which is usually the mean, μ_i of the i^{th} cluster is defined as [38]:

$$\mathbf{c}_i = \frac{1}{m_i} \sum_{x \in C_i} \mathbf{x}$$

It works by iteratively assigning data points to their closest centroid and updating the centroid of the new clusters until either convergence or threshold being reached. It is a simple and fast algorithm that scales easily to with large datasets but has limitations with non-globular shaped custers and clusters with varying sizes and densities. The basic K-means algorithm:

Step	Description
1	Select K points as initial centroids.
2	**repeat**.
3	Form K clusters by assigning each point to its closest centroid.
4	Recompute the centroid of each cluster.
5	**until** Centroids do not change.

Table 1: K-means algorithm

GMM is a prototype method that is similar to k-means but instead of selecting and updating centroids, model parameters are chosen. An iterative process then follows where each data point is assigned a probability of belonging to each distribution and then assigned to the cluster (distribution) where they have the highest probability. If the mixture model is Gaussian, the expectation-maximization algorithm can be used. It is usually useful when you have clusters that are overlapping, no clear-cut boundaries, as as each point is assigned a probability of belonging to each cluster. Can be slow to converge and can get stuck in local minima if initialization was unlucky.

$$\mathcal{L}(\boldsymbol{\theta}|\mathbf{X}) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)$$

The objective is to maximize this log-likelihood function with respect to the parameters θ, using the Expectation-Maximization (EM) algorithm.

Step	Description
1	Select an initial set of parameters.
2	**repeat**.
3	**Expectation Step** For each object, calculate the probability that each object belongs to each distribution..
4	**Maximization Step**Given the probabilities from the Expectation step, find the new estimates of the parameters that maximize the expected likelihood.
5	**until** The parameters do not change or a specified threshold is reached.

Table 2: Expectation-Maximization algorithm

A hierarchical clustering is a recursive partitioning of a dataset into successively smaller clusters. In turn hierarchical clustering can further be divided into agglomerative and divisive. Agglomerative clustering is a bottom-up approach starting with all data points being recognized as individual cluster merging them as we go up the tree. Divisive clustering is a top-down approach starting with with data points being included in one cluster and splitting the clusters as we go down the tree.

We will look at three types of agglomerative hierarchical clustering namely Single (MIN), Complete (MAX), and Average. These techniques have a fairly general algorithm that mainly differs according to how the proximity measures are used to merge the clusters. Single clustering merge clusters based on the closest single points. It minimizes distance between closest points. Complete clustering merge clusters based on the farthest points. It minimizes maximum distance. Average merge clusters based on the average distance between points. It minimizes average distance. Merging decisions in agglomerative clustering are final.

Step	**Description**
1	Compute the proximity matrix, if necessary.
2	**repeat**.
3	Merge the closest two clusters.
4	Update the proximity matrix to reflect the proximity between the new cluster and the original clusters.
5	**until** Only one cluster remains.

Table 3: Basic Agglomerative Hierarchical Clustering algorithm

A data point cannot be assigned another cluster once it has been assigned one. Interpretability and the ability to cluster arbitrary shapes are some of the advantages. Lack of

scalability with large datasets, sensitivity to excessive noise and outliers, non-reversibility once a decision has been are some of its weaknesses.

Spectral clustering was designed to clustering problems that have non-convex clusters like concentric circles [40]. Spectral clustering takes a spectral graph partitioning approach by using the top eigenvectors of a matrix derived from the distance between points simultaneously [41]. A similarity graph is created with nodes representing data points and edges connect them if the similarity is positive or greater than zero. The similarity graph is sparsified by removing edges below a threshold. Sparsification can be achieved by setting many low-similarity (high-disimilarity) values to zero. An adjacency matrix of the similarity graph is the matrix of edge weights $\mathbf{W} = w_{ii'}$. The diagonal matrix $\mathbf{D}$, is a diagonal matrix whose diagonal elements are the sum of the weights of the edges connected to a node where $d_i = \sum_{i'} w_{ii'}$. The graph Laplacian matrix can be defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W}$$

Using eigendecomposition, we have

$$\mathbf{L} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-\mathbf{1}}$$

where $\mathbf{V}$ is an nxn with columns being eigenvectors of $\mathbf{L}$ and Λ being the diagonal matrix with its diagonal elements being the eigenvalues of $\mathbf{L}$. In short, the eigenvectors of the graph Laplacian matrix contain information that can be used to partition the graph into its underlying components [38]. Spectral clustering looks for k eigenvectors corresponding to the smallest eigenvalues of $\mathbf{L}$. K-means clustering is then used to cluster the eigenvectors to extract clusters.

Step	Description
1	Create a sparsified similarity graph $\mathbb{G}$.
2	Compute the graph Laplacian for $\mathbb{G}$, $\mathbf{L}$.
3	Create a matrix $\mathbf{V}$ from the first k eigenvectors of $\mathbf{L}$.
4	Apply K-means clustering on $\mathbf{V}$ to obtain the k clusters.

Table 4: Spectral Clustering algorithm

Spectral clustering has the following advantages: ability to cluster varying size and arbitrary shape clusters including non-linearly separable clusters like concentric clusters, and it is also robust to noise and outliers. Computational complexity, $O(n^3)$ and memory requirements can be challenges for large data sets unless sparsification is implemented.

Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) incrementally and dynamically clusters incoming multi-dimensional metric data points to try to produce the

best quality clustering with the available resources (i.e., available memory and time constraints) [42]. Clustering Feature (CF) and CF Tree are the two central concepts used in BIRCH for incremental clustering. A Clustering Feature is a triple summarizing the information that we maintain about a cluster. A cluster can be thought of as a set of data points, but with only the CF vector stored as summary. This CF summary is not only efficient because it stores much less than all the data points in the cluster, but also accurate because it is sufficient for calculating all the measurements that we need for making clustering decisions in BIRCH. A CF tree is a height-balanced tree with two parameters: branching factor B and threshold T. The CF tree is a very compact representation dataset. Birch has two main steps: Building the CF tree and global clustering. It ideal for very large datasets.

Step	Description
1	Load the data into memory by creating a CF tree that summarizes the data.
2	(Optional, if needed in 3) Build a smaller CF tree.
3	Perform global clustering.
4	(Optional and offline) Cluster refining - using the centroids of the clusters produced in Phase 3 as seeds, the data points are redistributed to their closest seed to obtain a set of new clusters.

Table 5: BIRCH Clustering algorithm

The main advantage of BIRCH is that it was designed to deal with very large datasets where resources like memory are minimal. It can scale easily and is memory efficient. It is also robust to noise. Lack of interpretability, non-convex or irregularly shaped clusters, and the need to tune parameters are some of the challenges.

Density-based Spatial Clustering of Applications with Noise (DBSCAN) is a density-based method that was designed to solve the problem when users lack domain knowledge about the number of clusters that might exist, when clusters have arbitrary shapes, and when the datasets are large [43]. DBSCAN is based on the notion of data points being labeled as either a core point, border point, or noise point. It has two main parameters epsilon and minPts. epsilon is the distance (radius) from a point that is specified in order for points within this radius to be considered inside the core. minPts is the minimum number of points within epsilon that are required for it to be considered a core.

The parameters epsilon and minPts are global values with the same values used for all clusters. *epsilon* is a distance-based threshold while minimum number of points is a density-based threshold. Smaller *epsilon* can miss sparse density clusters while large *epsilon* can potentially merge two high density clusters.

DBSCAN has the following advantages: robust to outliers and noise, can handle arbitrary

Step	Description
1	Label all points as core, border, or noise points.
2	Eliminate noise points.
3	Put an edge between all core points within a distance Eps of each other.
4	Make each group of connected core points into a separate cluster.
5	Assign each border point to one of the clusters of its associated core points.

Table 6: DBSCAN algorithm

shapes and sizes, it is scalable to large data sets, and it does not require specification of the number of clusters Main disdvantage is its inability to cluster when clusters are have varying densities.

Ordering Points To Identify the Clustering Structure (OPTICS) is a clustering algorithm based on density reachability and core distances. It does not explicitly produce clusters but creates an augmented ordering of the dataset representing its density-based clustering structure. This ordering contains information that is equivalent to the density-based clusterings corresponding to a broad range of parameter settings. This addresses DBSCAN's weakness of failing to find clusters when their densities are varying. The developers of OPTICS, which was developed by the same team that developed DBSCAN, defined core distance and reachability distance. The OPTICS algorithm generates the augmented cluster-ordering consisting of the ordering of the points, the reachability-values and the core-values [44]. A reachability plot can be produced by combining reachability distances and data set ordering.

OPTICS has the following strengths: automatically detects clusters of arbitrary shape and size in the dataset without requiring prior knowledge of the number of clusters, through its augmented ordering of points, it can detect clusters of varying densities. It is robust to noise and outliers. Weaknessses are its memory and computational requierements the datset is large. High-dimensional data can also pose a challenge.

Step	Description
1	Calculate reachability distances.
2	Build the reachability plot
3	Extract cluster hierarchies.
4	Identify core points and clusters.
5	Set the MinPts parameter and extract the clusters.

Table 7: OPTICS algorithm

The Chameleon algorithm's key feature is that it accounts for both interconnectivity and

closeness in identifying the most similar pair of clusters. It uses an approach that does not depend on a static, user-supplied model and can automatically adapt to the internal characteristics of the merged clusters. Chameleon uses a dynamic modeling framework to determine the similarity between pairs of clusters by looking at their relative interconnectivity (RI) and relative closeness (RC). CHAMELEON takes into account features intrinsic to the clusters [45].

The algorithm has two stages: a graph-partitioning algorithm to cluster the data items into several relatively small sub clusters, and an algorithm to find the genuine clusters by repeatedly combining these subclusters using the concepts of Rrelative interconnectivity and relative closeness. Relative interconnectivity between clusters is their absolute interconnectivity normalized with respect to their internal interconnectivities. Absolute interconnectivity between clusters C_i and C_j in terms of edge cut is the sum of the weight of the edges that straddle the two clusters.The two stages are preceded by a preprocessing step that produces a graph representation of the data using the k-nearest neighbor graph approach.

$$\mathbf{RI}(\mathbf{C_i}, \mathbf{C_j}) = \frac{\mathbf{EC}(\mathbf{C_i}, \mathbf{C_J})}{\frac{1}{2}\left(\mathbf{EC}(\mathbf{C_i} + \mathbf{EC}(\mathbf{C_j})\right)}$$

Relative closeness is between a pair of clusters is the absolute closeness normalized with respect to the internal closeness of the two clusters.Absolute closeness of clusters is the average weight of the edges that connect vertices in C_i to those in C_j.

$$\mathbf{RC}(\mathbf{C_i}, \mathbf{C_j}) = \frac{\bar{\mathbf{S}}_{\mathbf{EC}}(\mathbf{C_i}, \mathbf{C_J})}{\frac{m_i}{m_i+m_j}\bar{\mathbf{S}}_{\mathbf{EC}}(\mathbf{C_i}) + \frac{m_i}{m_i+m_j}\bar{\mathbf{S}}_{\mathbf{EC}}(\mathbf{C_j})}$$

The main advantage of CHAMELEON algorithm is that it adapts to the local intrinsic characteristics of the clusters and thereby able to form clusters that have varying shapes, sizes, and densities. It is robust to noise and outliers. Like other hierarchical methods, once a decision is made, it cannot be undone.

Step	Description
1	Build k-nearest neighbor graph.
2	Partition the graph using a multilevel graph partitioning algorithm
3	**repeat**.
4	Merge the clusters that preserve the cluster self-similarity with respect to relative interconnectivity and relative closeness.
5	**until** No more clusters can be merged.

Table 8: Chameleon algorithm

Shared nearest neighbor density based clustering (SNN density-based) combines the concepts of shared nearest neighbor similarity and density-based clustering. This method was developed to deal with the limitations faced with clustering high dimensional data. It also takes into account variations in density. Shared nearest neighbor similarity is the number of shared neighbors as long as the two objects are on each other's nearest neighbor lists [46]. Using this similarity measure, density at a data point is defined as the sum of the similarities of a point's nearest neighbors [47]. SNN density combined with DBSCAN creates SNN Density-based clustering.

Step	Description
1	Compute the SNN similarity graph.
2	Apply DBSCAN with user-specified parameters eps and minPts.

Table 9: SNN density-based algorithm

Strengths are that it can handle high-dimensional data, robust to noise and outliers, and can handles varying sizes, shapes, and densities. Time complexity of $O(n^2)$, and splitting true clusters or joining separate clusters are some of its weaknesses.

Following the comparison in the next section, HDBSCAN [48] [49] was the clustering method that performed better in many areas than the other methods. We will briefly look at HDBSCAN [39] and describe its properties and how it works. HDBSCAN was mainly developed to improve on the main weakness of DBSCAN that is it does not perform well when clusters have varying densities. OPTICS was an improvement to DBSCAN and introduced the reachability graph as a solution to the varying densities problem. HDBSCAN is an improvement over OPTICS. Both OPTICS and HDBSCAN are density-based hierarchical clustering algorithms.

The HDBSCAN algorithm has the following steps:

Step 1 involves the computation of core distances for every data point with respect to the minimum number of samples. The distance between a data point and its k^{th} nearest neighbor is computed.

$$d_{core}(x_p) = d(x_p, x_*)$$

The mutual reachability distance of two points, x_p and x_q is

$$d_{mrd}(x_p, x_q) = \max\{d_{core}(x_p), d_{core}(x_q), d(x_p, x_q)\}$$

Step	Description
1.	Compute the core distance w.r.t. mpts for all data objects in X.
2.	Compute an MST of G_{mpts}, the Mutual Reachability Graph.
3.	Extend the MST to obtain MSText, by adding for each vertex a “self edge" with the core distance of the corresponding object as weight.
4.	Extract the HDBSCAN hierarchy as a dendrogram from MSText: (a) For the root of the tree assign all objects the same label (single “cluster"). (b) Iteratively remove all edges from MSText in decreasing order of weights (in case of ties, edges must be removed simultaneously): i. Before each removal, set the dendrogram scale value of the current hierarchical level as the weight of the edge(s) to be removed. ii. After each removal, assign labels to the connected component(s) that contain(s) the end vertex(-ices) of the removed edge(s), to obtain the next hierarchical level: assign a new cluster label to a component if it still has at least one edge, else assign it a null label (“noise") [69].

Table 10: HDBSCAN clustering algorithm

We can then construct a mutual reachability graph defined for a fixed choice of min_samples by associating each sample with a vertex of the graph, and thus edges between points are the mutual reachability distance between them.

Step 3 involves constructing of a minimum spanning tree (MST) which is equivalent to Single Linkage Clustering and a hierarchical clustering is obtained. At this point we are dealing with distance-based clustering.

Step 4 involves pruning the clustering tree into a condensed tree. To switch to density-based approach, we need to use density parameters which can be estimated by $\lambda = \frac{1}{\epsilon}$. Varying the value of λ extracts clusters at those density values.

Step 5 involves extracting a flat clustering tree with the optimal flat clustering being one that maximizes the persistence score (stability) over chosen clusters, subject to the constraint that clusters must not overlap. If the set of clusters is $C_1, C_2, \ldots, C_n$ then we wish to select $I \subseteq 1, 2, \ldots, n$ to maximize

$$\sum_{i \in I} \sigma(C_i)$$

subject to the constraint that, for all $i, j \in I$ with $i = j$,we have

$$C_i \cap C_j = \emptyset$$

The constraint not only prevents clusters to overlap but it also prevents nested clusters from being selected.

The stability of a cluster, $\sigma(C_i)$, is the sum of the range of λ values for points in a cluster.

$$\sigma(C_i) = \sum_{X_j \in C_i} (\lambda_{max,C_i} C_i(X_j) - \lambda_{min,C_i} C_i(X_j))$$

3.5.2 Evaluation of Clustering Methods

Initially, the following methods were considered for investigation: k-means, k-prototypes, GMM, hierarchical measures (MIN, MAX, Average), CURE, CHAMELEON, BIRCH, spectral clustering, DBSCAN, SNN Density-based, DENCLUE, and DBSCAN. DENCLUE, CURE and CHAMELEON were dropped because a readily available implementation was not found. Spectral clustering, OPTICS, average, and complete for UNSW-NB15 were dropped as they either had memory issues even with access to high-performance computing resources with 364GB RAM and 40 cores. k-protoypes method was not used when a decision was made to use the reduced data. SNN had runtime issues and was also dropped from the investigation.

We first tried to perform clustering using the Gower matrix as input as had proximity measures for mixed data types. But since the matrix was either 148517 X 148517 or 257673 X 257673, RAM was easily consumed. It was then decided that the data in low dimension would be used. The UMAP with 3 components was then used as the input to the clustering algorithms.

Performing a visual inspection revealed that HDBSCAN was able to pick both small and large groups in the NSL KDD dataset. It was also able to distinguish dense groupings from sparse groupings in the UNSW-NB15 dataset. See Fig 11. The first and third rows are the UMAP output with colored by attack types. The second and fourth rows are the HDBSCAN outputs for the respective datasets. Cluster evaluation metrics were inconclusive. For the comparison of the cluster output see see appendices E-H.

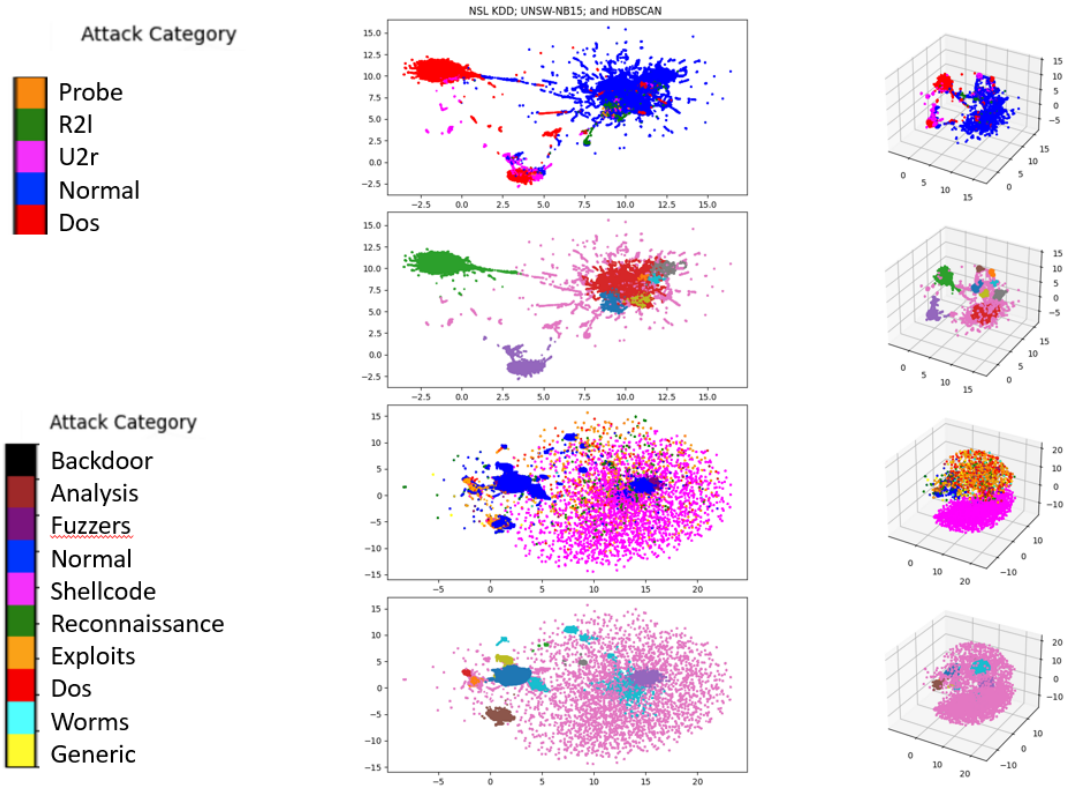


Figure 8: 2D and 3D HDBSCAN outputs for the NSW KDD and UNSW-NB15 data sets.

NSL KDD Clustering n_samples 148517, n_features 3

	Internal Evaluation Metrics		External Evaluation Metrics			
	Silh	CHI	DBI	ARI	AMI	Compl
k-means	0.627	361536.639	0.653	0.359	0.390	0.323
Average	0.692	242307.475	0.587	0.364	0.402	0.355
Complete	0.564	247649.877	0.721	0.346	0.381	0.315
DBSCAN	-0.111	1042.106	1.027	0.041	0.107	0.278
HDBSCAN	0.622	117506.749	1.206	0.303	0.390	0.293
OPTICS	0.269	41870.398	0.722	0.432	0.430	0.501
BIRCH	0.560	290185.196	1.014	0.251	0.374	0.300
GMM	0.620	339042.620	0.670	0.359	0.384	0.318

silh - silhouette score, CHI - Callinski score, DBI - David Bouldin score
sARI - adjusted rand score, AMI - adjusted mutual information, Compl - completeness_score

(a) NSL KDD

UNSW-NB15 Clustering n_samples 257673, n_features 3

	Internal Evaluation Metrics		External Evaluation Metrics			
	Silh	CHI	DBI	ARI	AMI	Compl
k-means	0.470	169100.817	0.887	0.191	0.355	0.314
Single	-0.102	196.730	0.905	-0.006	0.013	0.178
GMM	0.438	150617.444	0.940	0.201	0.365	0.323
DBSCAN	-0.364	508.856	0.913	-0.002	0.029	0.188
HDBSCAN	0.280	21893.812	1.378	0.198	0.309	0.269
BIRCH	0.373	98620.247	1.025	0.213	0.339	0.330

silh - silhouette score, CHI - Callinski score, DBI - David Bouldin score
sARI - adjusted rand score, AMI - adjusted mutual information, Compl - completeness_score

(b) UNSW-NB15

Figure 9: Clustering evaluation metrics

3.6 Anomaly Detection

The earlier usage of anomaly detection in cybersecurity can be traced to Dorothy Denning's paper in which a model based on the hypothesis that security violations can be detected by monitoring a system's audit records for abnormal patterns of system usage was proposed

[50]. "A value is an *exception* whenever it falls outside defined standards, expectations, or any other definition of normal. *Outlier*, by contrast, is a statistical term that refers to values that fall outside the norm based on a statistical calculation, such as anything beyond three standard deviations from the mean" [51]. Data points are *noise* when they contain some random component of a measurement error. These random or irrelevant fluctuations distorts the signal. Formally, anomaly detection approaches have been grouped into statistical-based, proximity-based, density-based, classification-based, density-based, clustering-based, and more recently reconstruction-based and information theoretic based. But recently, they are being grouped into three min approaches namely supervised, semi-supervised, and unsupervised approaches. In this paper we will look at Local Outlier Factor (LOF) [52], One Class Support Vector Machine [53], Isolation Forest (iForest) [54], and autoencoders [55] [56].

3.6.1 Comparison of Anomaly Detection Methods

LOF is an anomaly detection method that takes a density-based approach. It was developed by most of the developers of DBSCAN and OPTICS clustering methods and thus shares the concepts of *core distance* and *reachability distance* with the two algorithms. The LOF of an object is the measure assigned to an object of the degree of how much outlying that object is. Because the degree depends on how isolated an object is with respect to the surrounding neighborhood, it is local. The LOF of an object is based on the single parameter of MinPts, which is the number of nearest neighbors used in defining the local neighborhood of the object. Borrowing from definitions in OPTICS [44], *core distance* of an object p is simply the smallest distance ϵ' between p and an object in its ϵ-neighborhood such that p would be a core object with respect to ϵ' if this neighbor is contained in $N_\epsilon(p)$. Otherwise, the core-distance is UNDEFINED .

$$d_{core}(x_p) = d(x_p, x_*)$$

The *reachability-distance* of an object p with respect to another object o is the smallest distance such that p is directly density-reachable from o if o is a core object.

$$rd(x_p, x_o) = \max\{d_{core}(x_p), d(x_p, x_o)\}$$

core-distance neighborhood of an object p, contains every object whose distance from p is not greater than the *core distance*, These objects q are called the k-nearest neighbors of p. The *local reachability density* of p is defined as

$$lrd_{MinPts}(x_p) = \frac{1}{\left(\frac{\sum_{o \in N_{MinPts}(x_p} rd(x_p, x_o)}{|N_{MinPts}(x_p)|}\right)}$$

The *local reachability density* of p is the inverse of the average reachability distance of the object p from its neighbors, MinPts. MinPts can also be known as k-neighbors. The *local*

outlier factor of p is defined as

$$LOF_{MinPts}(x_p) = \left(\frac{\sum_{o \in N_{MinPts}(x_p)} \frac{lrd(x_o)}{lrd(x_p)}}{|N_{MinPts}(x_p)|} \right)$$

LOF is the average of the ratio of the *local reachability density* of p and those of p's MinPts-nearest neighbors, the average of the ratios of the density of sample p and the density of its nearest neighbors. LOF depends on one parameter only, MinPts or k nearest neighbors and therefore is sensitive to the value of the parameter chosen.

LOF considers an object as an outlier by the degree to which the object is isolated from its surrounding neighborhood. LOF of approximately 1 means the object has similar density as its neighbors, less than 1 has higher density than its neighbours and thus it is an inlier, and LOF greater than 1 means the object has lower density than its neighbors and is thus an outlier[52].

One Class Support Vector Machine (OC-SVM) for anomaly detection was developed to address the need of the difficulty in obtaining network traffic data that is not from normal traffic i.e. malicious traffic data [53]. It uses the OC-SVM algorithm which is an extension of the SVM algorithm to the case of unlabeled data [57].

SVM is a discriminative classification model that learns linear or nonlinear decision boundaries in the attribute space to separate classes. It represents the decision boundary using only a subset of the training data points that are most difficult to classify i.e. those data points that are on the edge. It has the advantage that only the instances closest to the margin determines class belongingness.

Understanding the following terms is important in understanding any type of SVM: hyperplane, margin, decision boundary, support vector, and kernels. In a p-dimensional space, a hyperplane is a flat affine subspace of dimension p-1. It is a line in two dimensions, and a plane in three dimensions. We can have infinitely many separating hyperplanes that separate data points into classes. A decision boundary (maximal margin hyperplane) is the separating hyperplane that is farthest away from the training data points. Support vectors are the data points that are located on the margin boundaries i.e. the perpendicular distance between the decison boundary and the data points on the margin boundaries. Only the support vectors determine the maximal margin boundary and nothing else. A kernel is a function that quantifies the similarity of two observations and is a generalization of the inner product of the form

$$K(x, y) = (\Phi(x), \Phi(y))$$

where Φ is a feature map $\mathbb{X} \mapsto \mathbb{F}$. A non-separable case for SVM has the following objective

function

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{subject to} \quad \xi_i \geq 0, \quad y_i(w^T x_i - b) \geq 1 - \xi_i \quad \forall i$$

A nonlinear SVM has the following objective function

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{subject to} \quad \xi_i \geq 0, \quad y_i(w^T \Phi(x) - b) \geq 1 - \xi_i \quad \forall i$$

where in both cases, w is the weight parameter, C is the regularizing parameter that controls the trade-off between the slack variables penalty and the margin, ξ_i are the slack variables that allow some data points to be within the margins or outside the decision boundary,

For OC-SVM, Scholkopf [57] proposed the following objective function

$$\min_{w,\xi,\rho} \quad \frac{1}{2}\|w\|^2 + \frac{1}{\nu l}\sum_{i=1}^{l}\xi_i - \rho$$
$$\text{subject to} \quad \xi_i \geq 0, \quad w^T \Phi(x) - b \geq \rho - \xi_i$$

and ρ is the offset parameter that determines how strict the model is in considering points as normal or anomalous. A higher ρ value makes the model less sensitive to anomalies, meaning fewer data points are classified as anomalies. Conversely, a lower ρ value makes the model more sensitive, classifying more points as anomalies. $\nu \in (0, 1]$ is the regularization parameter that is an upper bound on the fraction of outliers and a lower bound on the fraction of support vectors. OC-SVM aims to maximize the distance of the separating plane from the origin.It basically separates all the data points from the origin where the hyperplane has the form

$$w\Phi(x) - \rho = 0$$

and the distance from the hyperplane to the origin is $\frac{\rho}{||w||}$.

iForest is an unsupervised machine learning algorithm for anomaly detection that unlike other methods that start by profiling normal instances start by isolating anomalous instances. It is an ensemble method. The main terms in an iForest algorithm are *isolation tree* and *path length*. *isolation tree* or *iTree* is defined as: Let T be a node of an isolation tree. T is either an external -node with no child, or an internal-node with one test and exactly two child nodes (T_l, T_r). A test consists of an attribute $q < p$ divides data points into T_l and T_r. *path length* is defined as: **Path Length** h(x) of a point x is measured by the number of edges x traverses an iTree from root node until traversal is terminated at an external node. The important parameters are sub-sampling size φ that controls the tree

training data size, and number of tree t that controls the ensemble size. n_estimators and max_samples respectively in *sklearn*.

An iTree is built by selecting a random sub-sample of the data and splitting it using a randomly selected feature. A threshold for splitting is randomly selected from a value that is between that feature's minimum and maximum values. t iTrees are built during the training stage and during the evaluation stage, each instance passes through the t iTrees and a path length h(x) for that instance computed. E(h(x)), the average or expected path length is then computed for that instance. An anomaly score s for each instance is then computed which is defined as

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}$$

c(n) is the average of h(x) given n which is used to normalize h(x) is defined as

$$c(n) = 2H(n-1) - (2(n-1)/n)$$

Instances that are anomalies have distinguishable attribute values and are likely to be separated in early partioning. Anomalies will more more susceptible to islation and hence have short path lengths.

iForest works well for large data sets, high dimensional data with a large number of irrelevant attributes, and where we have only normal instances and no anomalies.

An autoencoder is a type of artificial neural network used to learn efficient codings of unlabeled data (unsupervised learning) [58]. The model extracts the main features and distribution of the input data by decoding and encoding the input data. Initially, they were used for dimensionality reduction [59] but now they are also used for feature extraction, learning generative models of data, and can be used for compression. For anomaly detection they are used through the incorporation of a reconstruction error. The reconstruction error is used to evaluate the performance of the autoencoder and measures how well the input is reconstructed by measuring the input and output difference [60].

It has the following setup: input layer together the hidden encoding layers, bottleneck layer, and output layer together with the hidden decoding layers. The importance of autoencoders in recreating the input comes from the bottleneck layer. Autoassociative networks are useful precisely because they contain an internal constraint that prevents them from learning the identity mapping perfectly [56].

The input and output layers have the same the number of nodes that is equal to the number of input attributes/features. The layers after the input layer but before the bottleneck layer form the encoder part and the layers after the bottleneck layer and before the output layer form the decoder part. The encoder needs to compress the representation of an input from a higher dimension to a lower dimension. The decoder recreates the input as closely as possible by using the lower dimensional result in the bottleneck layer. The lower dimension in the bottleneck layer should contain only the most important information.

The steps in anomaly detection using reconstruction error can be summarized as:

Step	Description
1.	Let $\mathbf{x}$ be the original data object.
2.	Find the representation of the object in a lower dimensional space. (Encoding).
3.	Project the object back to the original space. (Decoding).
4.	Call this object $\hat{\mathbf{x}}$

Table 11: Autoencoder steps

The reconstruction error is then

$$\text{Reconstruction Error}(\mathbf{x}) = \|\mathbf{x} - \hat{\mathbf{x}}\|$$

Objects with large reconstruction errors are anomalies.

3.6.2 Evaluation of Anomaly Detection Methods

From visual inspection, iForest was able to identify most cyberattack types that have fewer instances while it determined Dos attacks as normal which is understandable by looking at the number of instances. OC-SVM came second. Anomaly detection using autoencoders managed to pick the r2l (green color) cyberattack only. Since anomaly detection is the detection of rare events, this method can be said to be the best as it has managed to pick out the cyberattack that had fewer instances.

4 Conclusion

For visualization purposes of cybersecurity data that is usually high-dimensional, there is need to use dimensionality reduction methods. UMAP was the method that came out the best. To cluster the data, HDBSCAN performed better. Isolation Forest managed to pick out a number of cyberattacks but autoencoder managed to pick a cyberattack that had fewer instances compared to the other attacks. In choosing a dimensional reduction method, one should look at how the chosen method preserves both local and global structure of the data. In choosing a clustering method the following should be taken into account: size, shape, density, noise and outliers, and the size of the dataset. Noise and outliers are very important in cybersecurity as the anomalies usually indicate a cyberattack therefore one should avoid methods that discard noise. Knowing the essentials of both cybersecurity and data analytics would help practitioners to efficiently analyze data and design and implement tools that can help in protecting information systems. Knowing how the various algorithms work can help in choosing the best candidate algorithms to test without wasting time on trying all the available algorithms before finding one that works best.

5 References

References

[1] Bresniker K et al. "Grand Challenge: Applying Artificial Intelligence and Machine Learning to Cybersecurity". In: *Computer* 52 (12), p. 2019.

[2] (ISC)² Blog. *(ISC)² Blog*. Oct. 2018. URL: https://blog.isc2.org/isc2_blog/2018/10/cybersecurity-skills-shortage-soars-nearing-3-million.html.

[3] Mongeau S and Hajdasinski A. *Cybersecurity Data Science: Best Practices in an Emerging Profession*. Cham, Switzerland: Springer, 2021.

[4] Sarker I. "Machine Learning for Intelligent Data Analysis and Automation in Cybersecurity: Current and Future Prospects". In: *Annals of Data Science* 10 (199), pp. 1473–1498.

[5] Greengard S. "Cybersecurity gets smart". In: *Communications of the ACM* 59 (5 May 2015), pp. 29–31.

[6] National Cyber Security Centre (NCSC). "NCSC Warns That AI is Already Being Used by Ransomware Gangs". In: (Jan. 2024). Blog Article. URL: https://www.tripwire.com/state-of-security/ncsc-warns-ai-already-being-used-ransomware-gangs.

[7] Abnormal blog. *3 Cybersecurity Threats Caused by Generative AI*. Blog Post. July 2023. URL: https://abnormalsecurity.com/blog/cybersecurity-threats-generative-ai.

[8] CompTIA. *What is a Network Protocol?* Accessed: 24 1 2024. 2024. URL: https://www.comptia.org/content/guides/what-is-a-network-protocol.

[9] Imperva. *The OSI Model Explained: How to Understand (and Remember) the 7-Layer Network Model*. 2024. URL: https://www.imperva.com/learn/application-security/osi-model/.

[10] Stallings W. In: *Cryptography and Network Security* (2017).

[11] Lu W. In: *4th International Conference on Wireless, Intelligent and Distributed Environment for Communication* (2021), p. 23.

[12] Lu W. In: *4th International Conference on Wireless, Intelligent and Distributed Environment for Communication* (2021).

[13] NIST. *Cyber Attack*. Accessed: 2 4 2024. 2024. URL: https://csrc.nist.gov/glossary/term/cyber_attack.

[14] Janeja V. *Data Analytics for Cybersecurity*. Cambridge University Press, 2022, p. 20.

[15] Chapple N, Stewart J, and Gibson D. "Certified Information System Security Professional Office Study Guide". In: (2021), p. 1004.

[16] Sarker I et al. "Cybersecurity data science: an overview from machine learning perspective". In: *Journal of Big Data* 7 (1 2020), pp. 1–29.

[17] Sarraute C, Miranda F, and Orlick J. "Simulation of Computer Network Attacks". In: (2010).

[18] Verma R and Marchette D. *Cybersecurity Analytics*. Taylor & Francis Group, 2020, pp. 52–53.
[19] Janeja V. *Data Analytics for Cybersecurity*. Cambridge University Press, 2022, p. 21.
[20] Janeja V. *Data Analytics for Cybersecurity*. Cambridge University Press, 2022, p. 25.
[21] Mcinnes L, Healy J, and Melville J. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction". In: (). arXiv: arxiv:1802.03426v3.
[22] Sheskin D. In: *Handbook of Parametric and Nonparametric Statistical Procedures* (2011), p. 1647.
[23] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey". In: *ACM Comput. Surv.* 41.3 (July 2009). ISSN: 0360-0300. DOI: 10.1145/1541880.1541882. URL: https://doi.org/10.1145/1541880.1541882.
[24] M. Tavallaee et al. "A Detailed Analysis of the KDD CUP 99 Data Set". In: *Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*. 2009.
[25] Moustafa N and Slay J. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)". In: *Military Communications and Information Systems Conference (MilCIS)* (2015).
[26] Saporito G. "Towards Data Sceince". In: (Sept. 2019).
[27] Gower J. "A General Coefficient of Similarity and Some of Its Properties". In: *Biometrics* 27 (4 1971), pp. 857–871.
[28] Jérôme Pagès. *Multiple Factor Analysis by Example Using R*. 1st. Chapman and Hall/CRC, 2014. Chap. 3. DOI: 10.1201/b17700. URL: https://doi-org.ezp2.lib.umn.edu/10.1201/b17700.
[29] Dice L. "Measures of the Amount of Ecologic Association Between Species". In: *Ecology* 26 (3 1945), pp. 297–302.
[30] Michael Christoph Thrun. "Methods of Projection". In: *Projection-Based Clustering through Self-Organization and Swarm Intelligence: Combining Cluster Analysis with the Visualization of High-Dimensional Data*. Wiesbaden: Springer Fachmedien Wiesbaden, 2018, pp. 33–42. ISBN: 978-3-658-20540-9. DOI: 10.1007/978-3-658-20540-9_4. URL: https://doi.org/10.1007/978-3-658-20540-9_4.
[31] Nguyen L and Holmes S. "Ten quick tips for effective dimensionality reduction". In: *PLOS Computational Biology* 15 (6 2019).
[32] J. "Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis". In: *Psychometrika* 29 (1 1964).
[33] Tenenbaum J, Silva V, and Langford J. "A Global Geometric Framework for Nonlinear Dimensionality Reduction". In: *Science* 290 (5500 2000), pp. 2319–2323.
[34] Van Der Maaten L and Hinton G. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9 (11 2008), pp. 2579–2605.
[35] Alan Agresti. *Categorical Data Analysis*. John Wiley & Sons, 2013, p. 396.

[36] J. Pagès. "Analyse factorielle de données mixtes". In: *Revue de Statistique Appliquée* 52.4 (2004), pp. 93–111. URL: http://archive.numdam.org/item/RSA_2004__52_4_93_0/.

[37] Wikipedia contributors. *Factor analysis of mixed data*. https://en.wikipedia.org/w/index.php?oldid=specific_version_id. Accessed: 10 4 2024. 2024.

[38] Tan P.-N et al. *Introduction to Data Mining*. Pearson India Education Services, 2023.

[39] Mcinnes L and Healy J. "Accelerated Hierarchical Density Based Clustering". In: *IEEE International Conference on Data Mining Workshops (ICDMW)* (2017).

[40] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2009.

[41] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. "On Spectral Clustering: Analysis and an Algorithm". In: *Advances in Neural Information Processing Systems*. 2002.

[42] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. "BIRCH: An Efficient Data Clustering Method for Very Large Databases". In: *SIGMOD Record* 25.2 (1996), pp. 103–114. DOI: 10.1145/235968.233324. URL: https://doi.org/10.1145/235968.233324.

[43] Martin Ester et al. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*. AAAI Press. 1996, pp. 226–231.

[44] Mihael Ankerst et al. "OPTICS: Ordering Points To Identify the Clustering Structure". In: *SIGMOD Rec.* 28.2 (1999), pp. 49–60. DOI: 10.1145/304181.304187.

[45] George Karypis, Eui-Hong Han, and Vipin Kumar. "Chameleon: Hierarchical Clustering Using Dynamic Modeling". In: *IEEE Computer* 32.8 (Aug. 1999), pp. 68–75.

[46] Tan P.-N et al. *Introduction to Data Mining*. Pearson India Education Services, 2023, p. 695.

[47] Levent Ertoz, Michael S. Steinbach, and Vipin Kumar. "A New Shared Nearest Neighbor Clustering Algorithm and its Applications". In: 2002. URL: https://api.semanticscholar.org/CorpusID:115462989.

[48] Campello R, Moulavi D, and Sander J. "Density-Based Clustering Based on Hierarchical Density Estimates". In: *Advances in Knowledge Discovery and Data Mining. PAKDD 2013* (2013).

[49] Campello R et al. "Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection". In: *ACM Transactions on Knowledge Discovery from Data* 10 (1 July 2015), pp. 1–51.

[50] Dorothy E. Denning. "An Intrusion-Detection Model". In: *IEEE Transactions on Software Engineering* SE-13.2 (1987), pp. 222–232. DOI: 10.1109/TSE.1987.233024.

[51] Stephen Few. *Now You See It: Simple Visualization Techniques for Quantitative Analysis*. Oklahoma City, OK: Analytics Press, 2009, p. 134.

[52] Markus M. Breunig et al. “LOF: Identifying Density-Based Local Outliers”. In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: ACM, 2000, pp. 93–104. DOI: 10.1145/342009.335388.

[53] X. Zhang, J. Xu, and L. Gong. “An Anomaly Detection Model Based on One-class SVM to Detect Network Intrusions”. In: *2015 11th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*. Shenzhen, China: IEEE, 2015, pp. 231–237. DOI: 10.1109/MSN.2015.47.

[54] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. “Isolation Forest”. In: *2008 Eighth IEEE International Conference on Data Mining*. Pisa, Italy: IEEE, 2008, pp. 413–422. DOI: 10.1109/ICDM.2008.17.

[55] Mark A. Kramer. “Nonlinear Principal Component Analysis Using Autoassociative Neural Networks”. In: *AIChE Journal* 37.2 (1991), pp. 233–243. DOI: 10.1002/aic.690370209.

[56] M. A. Kramer. “Autoassociative Neural Networks”. In: *Computers I& Chemical Engineering* 16.4 (1992), pp. 313–328. DOI: 10.1016/0098-1354(92)80074-K.

[57] Bernhard Schölkopf et al. “Estimating the Support of a High-Dimensional Distribution”. In: *Neural Computation* 13.7 (July 2001), pp. 1443–1471. ISSN: 0899-7667. DOI: 10.1162/089976601750264965. eprint: https://direct.mit.edu/neco/article-pdf/13/7/1443/814849/089976601750264965.pdf. URL: https://doi.org/10.1162/089976601750264965.

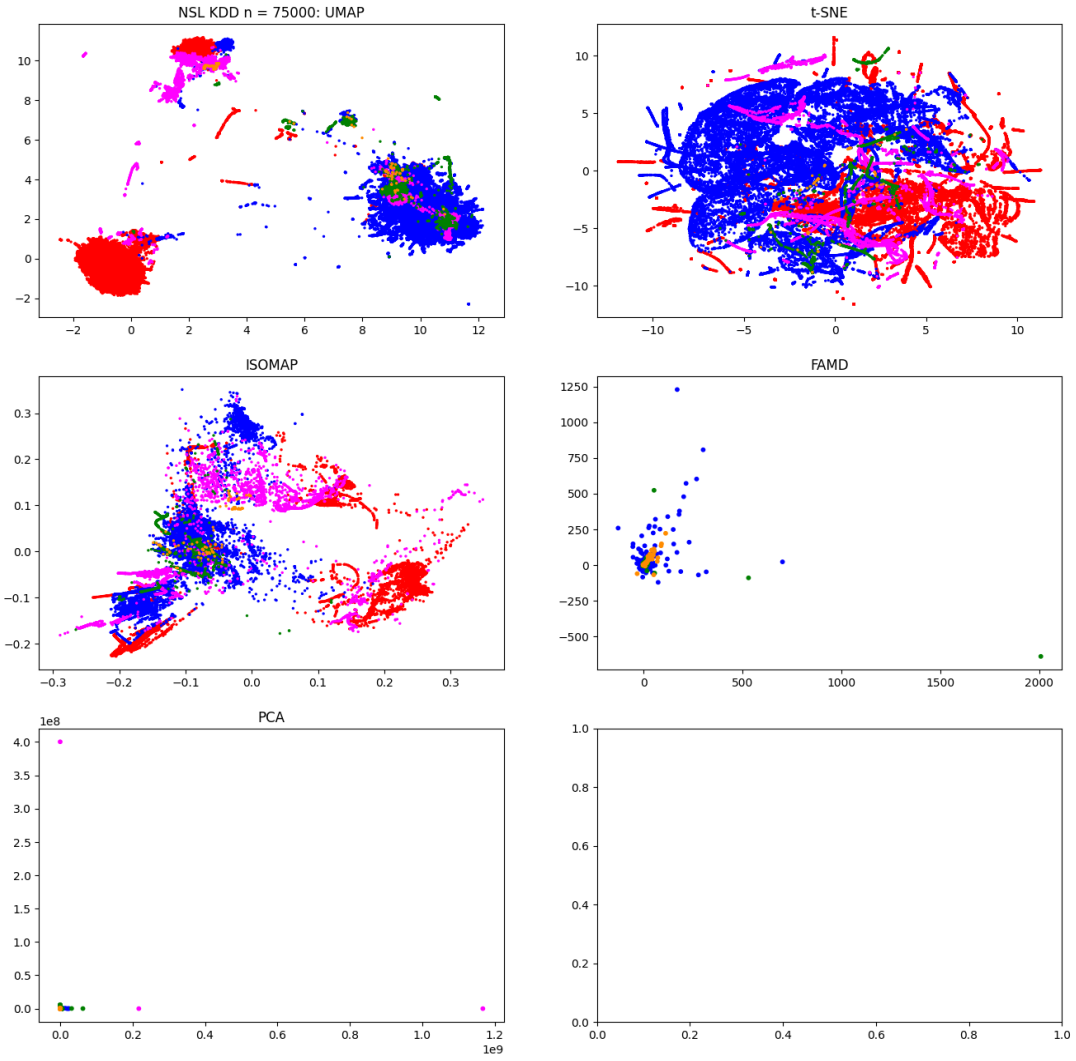
[58] *Autoencoder — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/wiki/Autoencoder. [Online; accessed 26-June-2024]. 2024.

[59] G. E. Hinton and R. R. Salakhutdinov. “Reducing the Dimensionality of Data with Neural Networks”. In: *Science* 313.5786 (2006), pp. 504–507. DOI: 10.1126/science.1127647. eprint: https://www.science.org/doi/pdf/10.1126/science.1127647. URL: https://www.science.org/doi/abs/10.1126/science.1127647.

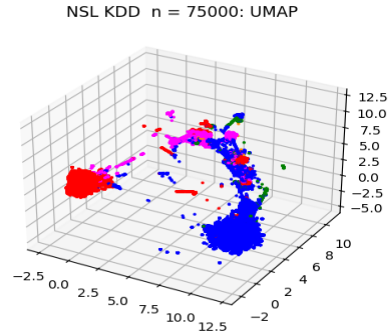
[60] Najmi Rosley et al. “Autoencoders with Reconstruction Error and Dimensionality Reduction for Credit Card Fraud Detection”. In: *CITIC 2022*. Atlantis Press, 2022, pp. 503–512. DOI: 10.2991/978-94-6463-094-7_40.

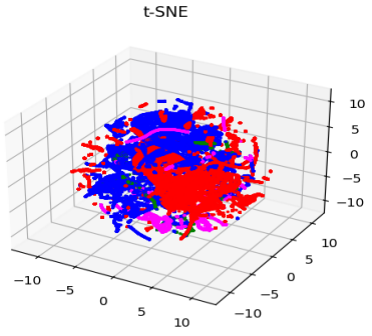
A Appendix

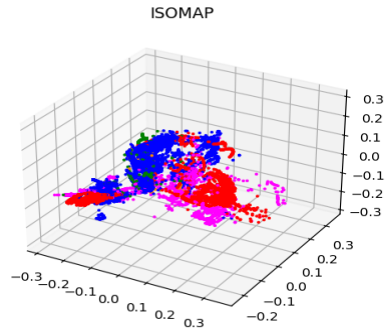
A.1 NSL KDD n=75,000, 2D dimensionality reduction outputs

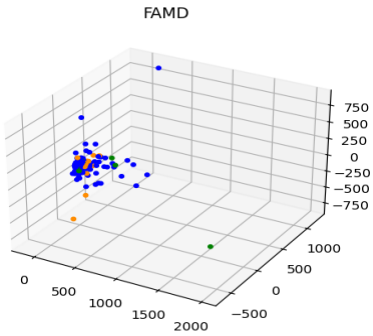


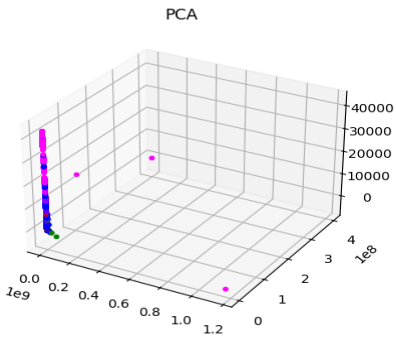
A.2 NSL KDD n=75,000, 3D dimensionality reduction outputs

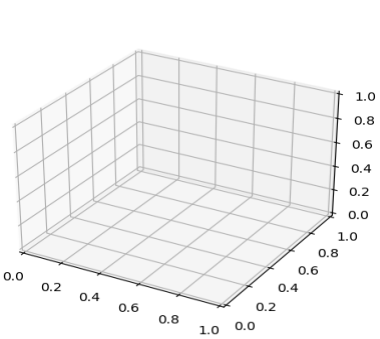




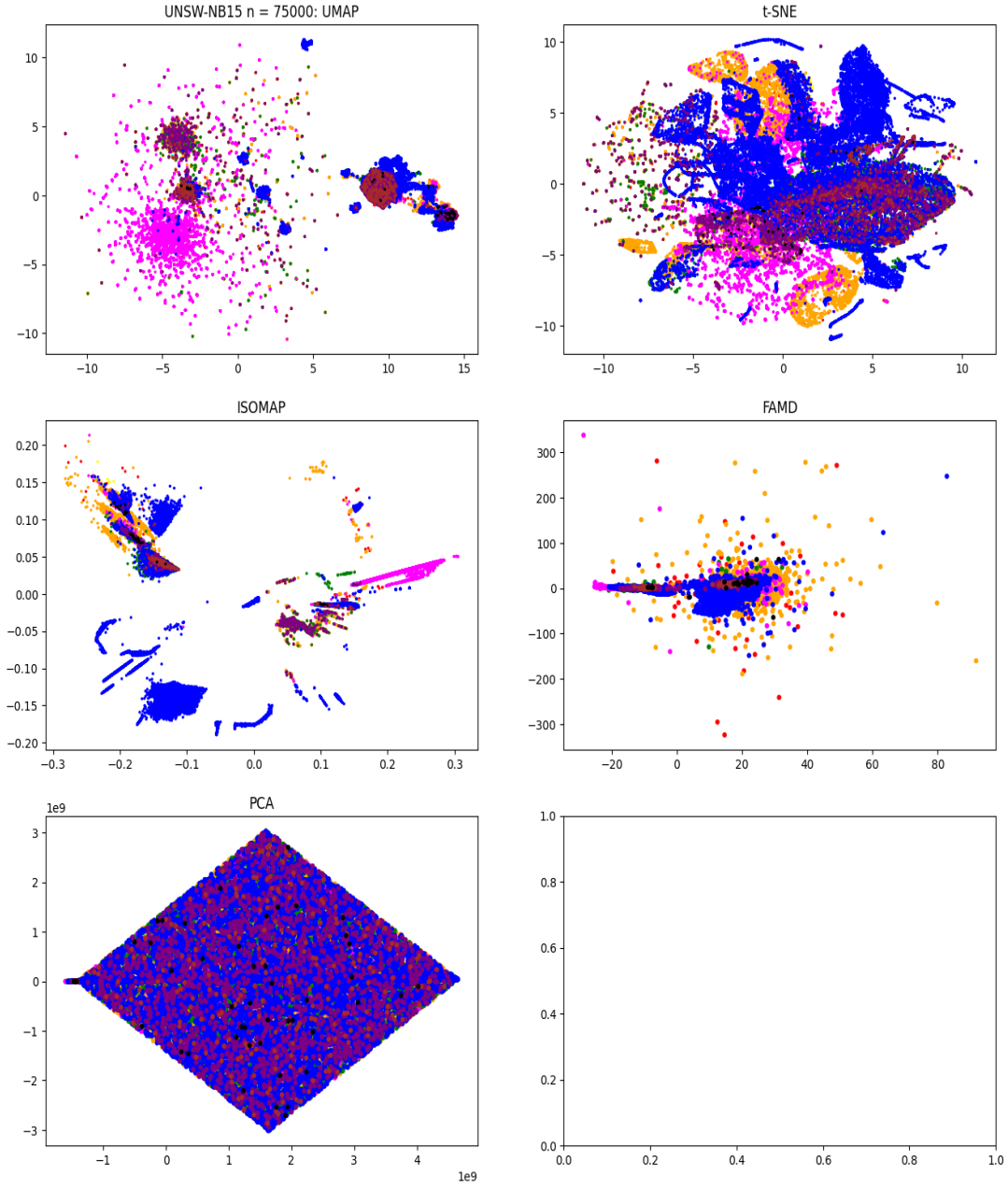




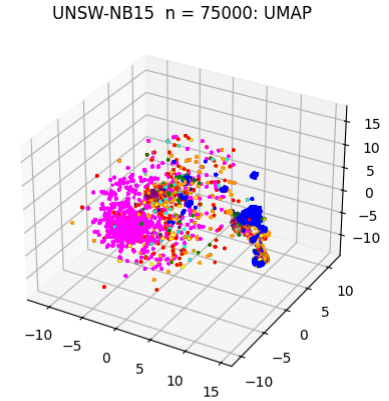


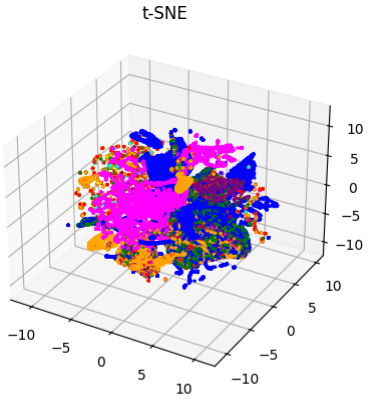


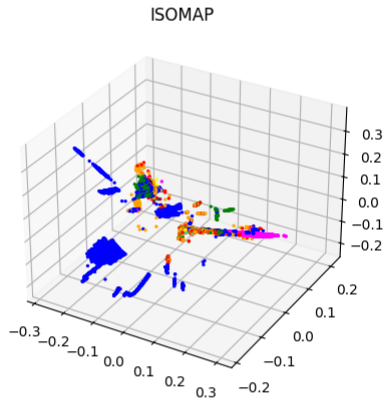
A.3 UNSW-NB15 n=75,000, 2D dimensionality reduction outputs

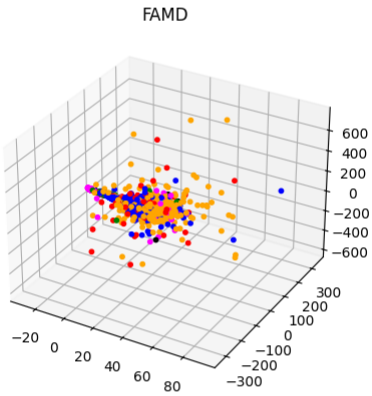


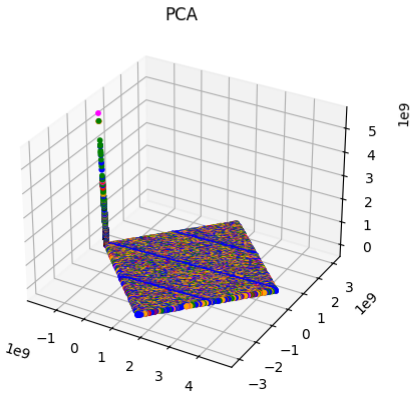
A.4 UNSW-NB15 n=75,000, 3D dimensionality reduction outputs

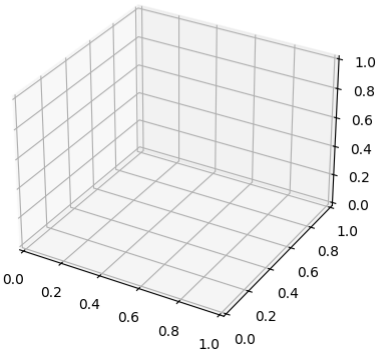




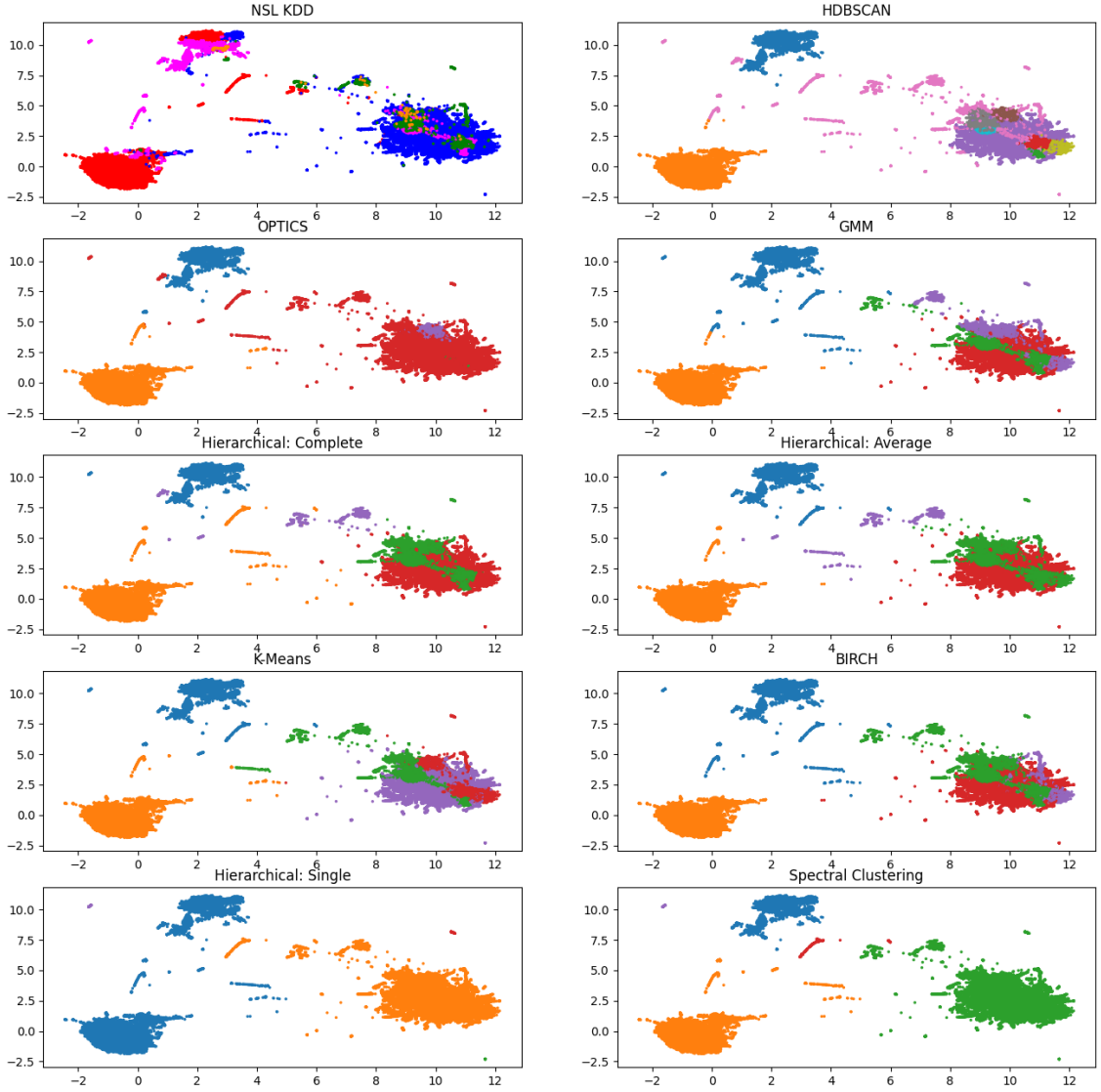




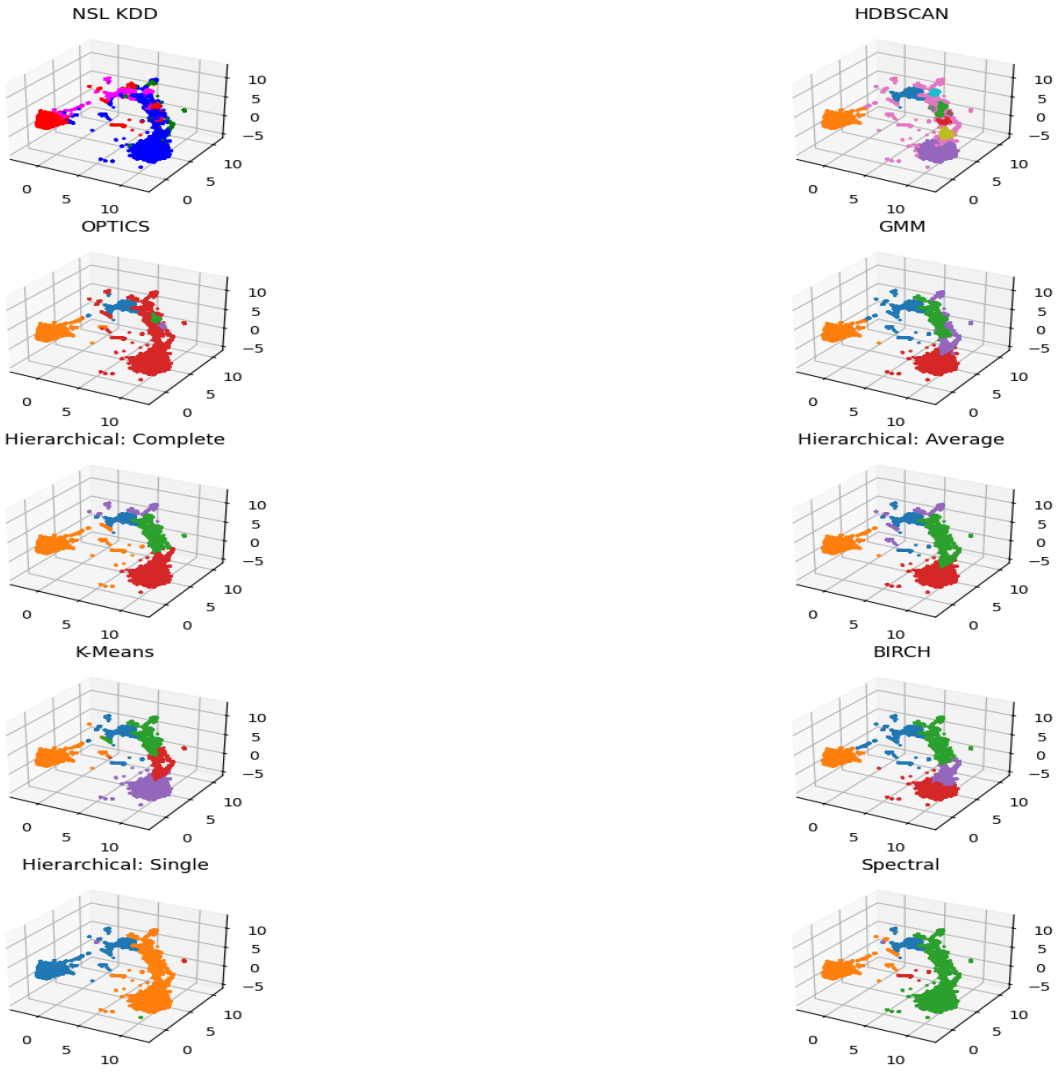




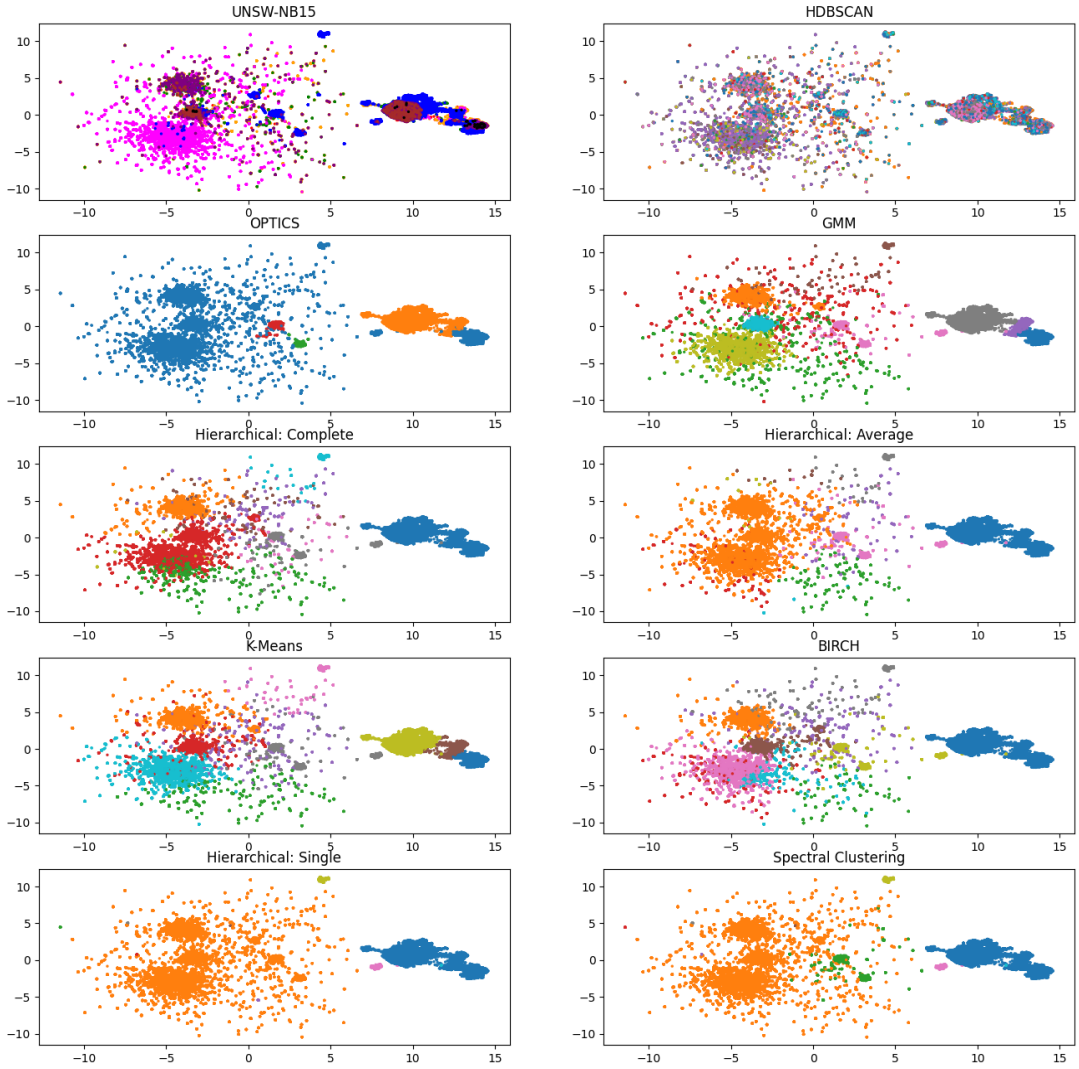
A.5 NSL KDD n=75,000, 2D clustering outputs



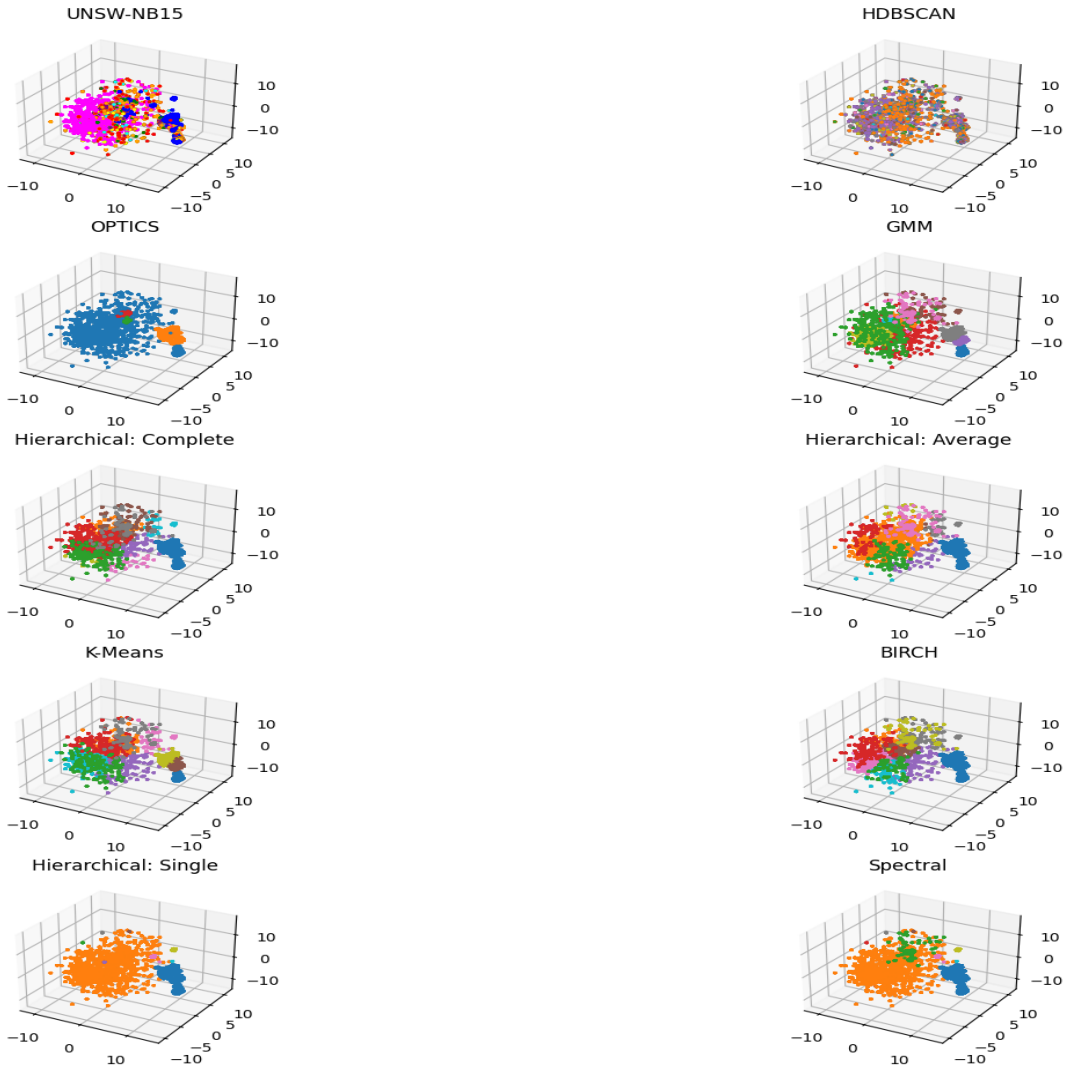
A.6 NSL KDD n=75,000, 3D clustering outputs



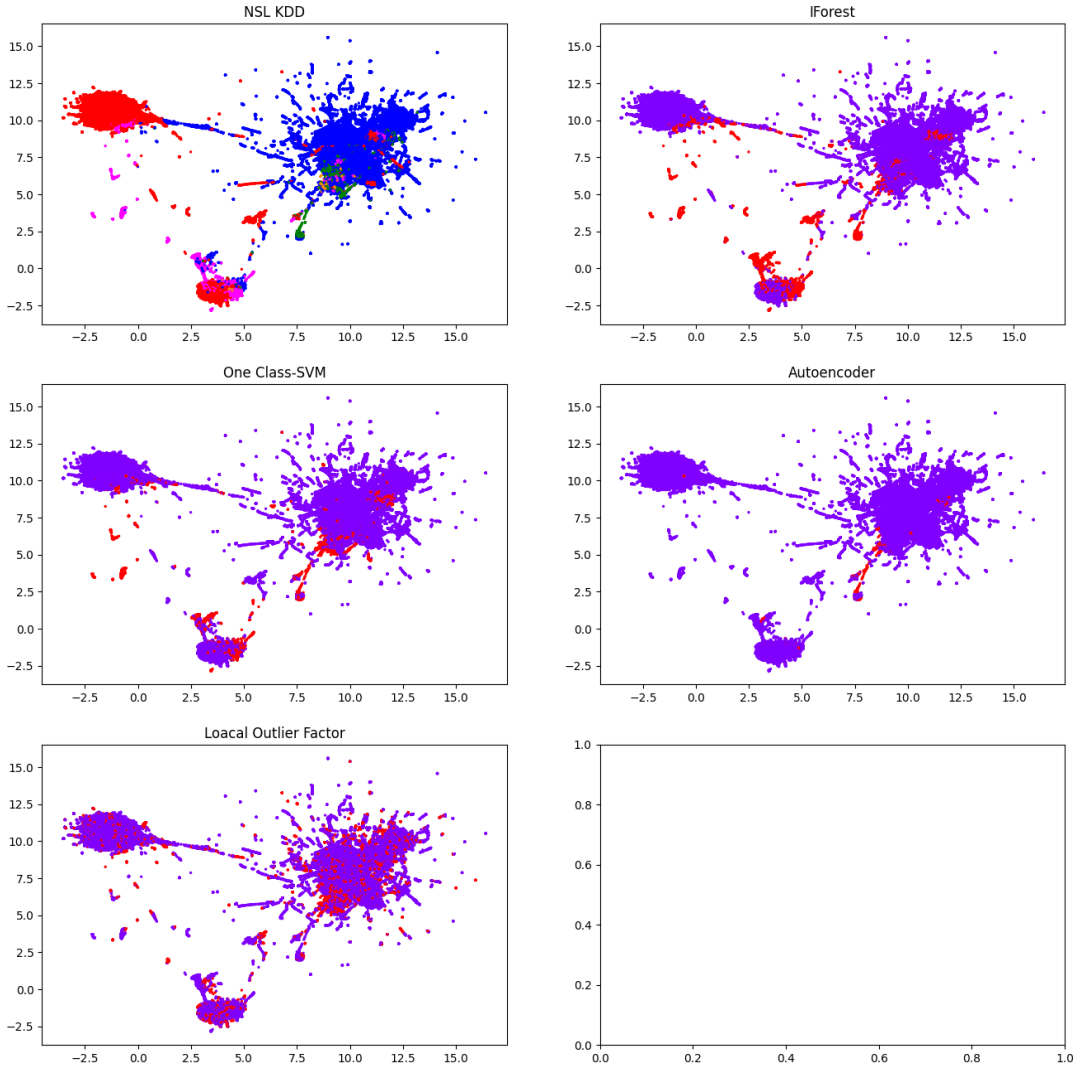
A.7 UNSW-NB15 n=75,000, 2D clustering outputs



A.8 UNSW-NB15 n=75,000, 3D clustering outputs



A.9 NSL KDD, 2D anomaly detection outputs



A.10 NSL KDD, 3D anomaly detection outputs