Bilkent University
Department of Electrical and Electronics Engineering

EEE 494 - Industrial Design Project II
CM4 Report

Anomaly-Based Network Intrusion Detection System using Big Data

Group Members of A3
Berfin Kavşut
Beste Aydemir
Ege Ozan Özyedek
Meltem Toprak
Şevki Gavrem Kulkuloğlu

Academic Advisor
Prof. Serdar Kozat
Company Advisor
Oğuzhan Karaahmetoğlu
Project TA
Arda Atalık

DATABOSS is a company which is active on topics of Big Data, Machine Learning, Artificial Intelligence, and Deep Learning. DATABOSS has a mission to produce the domestic and national solutions for our country's technological needs and it has a vision to global competition with high quality of goods [1].

May 20, 2021

Contents

Project Summary

The project is an anomaly-based network intrusion detection system on single source and multi-source streaming network traffic data. Anomaly detection is implemented in real-time with streaming big data. Anomalies on network traffic are monitored by an admin computer while network traffic packets are collected from connected computers. The project mainly has three parts: data acquisition, anomaly detection, and user interface. Data acquisition is agent-based and performed by a network protocol analyzer named Tshark [2]. To capture live packets in our program, a Python library called Pyshark [3], which is taking packets from Tshark directly, is used. Captured packets are processed by a transformer module one by one and then fed to the analysis module. Analysis module detects anomalies by models of a Python library named PySAD [4]. Models are trained with unsupervised learning because anomalies are unlabeled in real-life and novel anomalies are also desired to be caught. Anomalies are detected by models. Lastly, detected anomalies are visualized in UI via live graphs to alert the user. The user also receives an alert notification by mail. There is a confidence threshold selection option provided for the user. For simulation, NSL-KDD dataset [5], Kitsune Network Attack Dataset [6], and Intrusion Detection Evaluation Dataset (CIC-IDS2017) [7] are used where Kİtsune Network Attack Dataset is the most realistic dataset. The network traffic data has approximately 100 events per second for each host. The best achieved results are that the Area Under Receiver Operating Characteristic curve is 0.61, F1-Score is 0.203, and Precision-Recall curve is above no skill line.

List of Figures

List of Tables

1 Motivation and Novelty

With big data, there are countless streaming data coming from multiple sources in real-time. As a result, the number of network attacks has risen up and novel attacks are developed by attackers. The security of networked systems has become an important issue for individuals, companies, and governments [8]. Similarly, it is important to produce high-technology cybersecurity systems in the defense industry and many defense industry companies are carrying on projects on cybersecurity in Turkey.

There are two types of Intrusion Detection Systems (IDS): anomaly-based and signature-based. Signature-based IDS is built on pattern recognition of pre-stored anomalies and works with high accuracy for known anomalies [9]. However, its disadvantage is that it cannot detect any novel attack. Novel attacks should be predicted by developers of IDS beforehand and the efficiency of the system depends on the difference of speed of developers and attackers creating new signatures. Therefore, signature-based detection techniques are not capable of catching new, self-modifying network behavioural anomalies. Anomaly-based IDSs have the advantage to catch novel attacks [9]. Our system is an anomaly-based IDS for its ability to detect novel attacks.

Anomaly-based detection is based on defining network behaviour and network administrators should identify accepted network behaviour. However, after setting the rules for network behaviour, anomaly-based IDSs work fine [9], which is the challenging part of the project. In similar research projects, we have encountered interval-based features to identify network traffic behaviour: number of packets, byte sum of packet size, number of flows, number of source addresses, number of destination addresses, number of source ports, number of destination ports, the distance between source and destination addresses, the distance between a source port and destination port [10]. Another way to interpret network packet features and select the ones giving the most information about anomalies is using classical feature selection and extraction methods in machine learning applications. The performance of an IDS improves when the features are more discriminative and representative [8].

The company wants to build a system for online anomaly detection with big data. Following this, we built a system for intrusion detection on network traffic. At the end of the project, the expectation of our company was to have an independent, core system with anomaly detection on multi-source real-time streaming data. The system will not be integrated into another system. This core system is planned to be developed for other application areas not limited

to network traffic. It can be utilized as a real-time alert system for different purposes such as predictive maintenance and finance later. There is a transformer module for preparing streaming data before anomaly detection so that the company can use the system with different data by developing different transformer modules accordingly.

There are two target end-users of our product: developers in the firm and regular customers. We are building a core system for the company to continue developing later and use for their own purposes and data sources. Therefore, we give the user to select hyperparameters if they wish, however, it is not a must for the product. These options are provided for the users who have a technical background in Machine Learning and desires to optimize the functioning of the system. For the regular users who do not have a technical background, there will be a default setup. The product is a software product, therefore there is a need for a computer and network connection to use it. Other than that, there is no additional financial requirement for the user.

We used pre-built models of Python library PySAD (Python Streaming Anomaly Detection) which have online anomaly detection algorithms. For data acquisition, Tshark is used to extracts features from network traffic packets. For UI/UX, tools from Elasticsearch are used. Hence, it is unlikely to have a patent with our product because it is an implementation, i.e. combination, of existing solutions.

1.1 Similar Products

1.1.1 IBM QRadar Security Information and Event Management (SIEM)

IBM QRadar is a real-time cybersecurity program for the detection of network attacks. QRadar can monitor network flow data and identify anomalies on the network based on the applications, protocols, services, and ports they use. There are considerably high numbers of detected cyber attacks. Some identified attacks are security events (firewall, intrusion detection systems, intrusion prevention systems), network events from switches, routers, servers, hosts, network activity context, and cloud activity from Google Cloud. Cost information is not shared by the company. Product prices are offered differently according to the needs of firms and customers Therefore, product prices cannot be officially seen from the information on the website [11]. We will not be able to detect as much as IBM QRadar and our company advisor suggested classification of anomalies after the core project works well and anomalies can be detected with high-performance metrics. We may label anomalies that are encountered and stored beforehand, however, the main idea of our project is to be able to identify novel attacks with online anomaly detection techniques.

One disadvantage of IBM Qradar is that it is a black-box program and the user cannot interfere in the process of anomaly detection. In our UI, we will let the user select hyperparameters and confidence thresholds. In IBM Qradar, anomalies are detected in real-time and the user can see the report of anomalies with their assigned categories as shown in Figure 1. The program can detect not only known anomalies such as data exfiltration using large DNS, Denial of Service (DoS) attack, Firewall attack, but also unknown anomalies. This property is a significant advantage of the program since mostly signature-based anomaly detections are used in intrusion detection systems (IDS) [9], which is similar to our system.

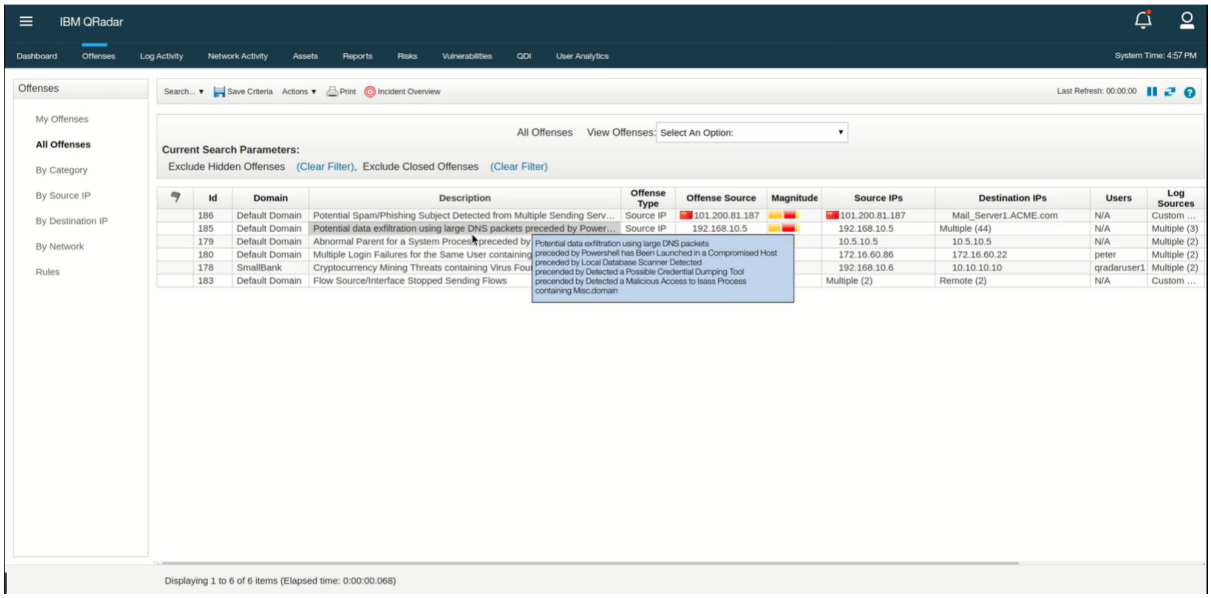
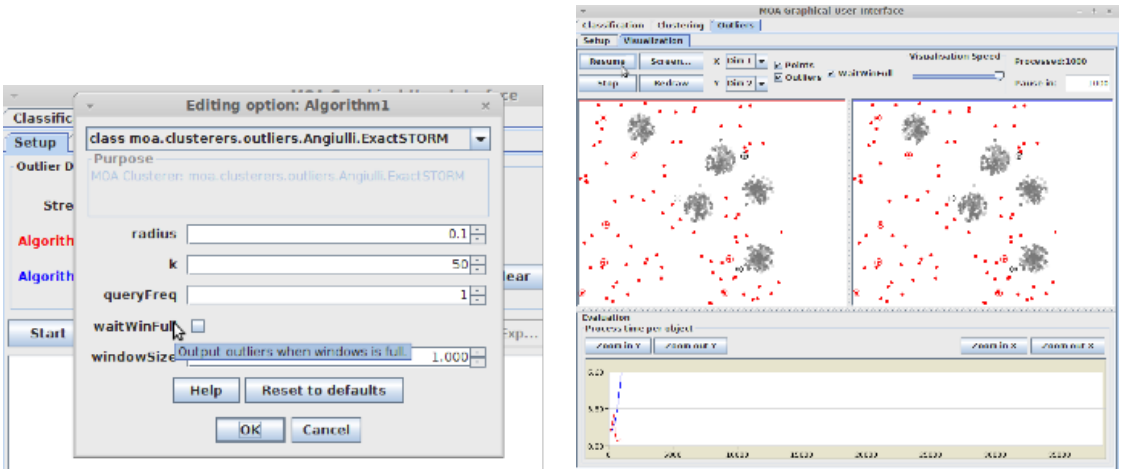


Figure 1: IBM QRadar SIEM [12].

1.1.2 MOA

MOA is an open-source program working on GUI for data stream mining. It is a free product and can be downloaded directly from its official website [13]. The cost of the product cannot be found. In the machine learning community, it is highly used especially by beginners. Available machine learning algorithms are classification, regression, clustering, outlier detection, concept drift detection, and recommender systems. There are also tools for evaluation. Algorithm parameters can be selected by the user, and two algorithms in the outlier detection part can be compared with their performances as shown in Figure 2. Outliers are visualized in real-time by mapping. Pre-saved data streams are coming with the installation of the program and there are also drift stream generators, rotating hyperplane, and random RBF generator [13]. MOA is similar to our product in terms of UI, however, it is more suitable for developers who want to experiment and have a comparison with different machine learning algorithms. It is implemented in Java while we will use Python as the programming language.



(a) Parameter selection in MOA

(b) Visualization of outliers by two algorithms

Figure 2: Example pictures from MOA [14].

2 Design and Performance Requirements

2.1 Functional Requirements

The functions and requirements of the final product are specified and listed below based on IEEE Std 1233-1998. Throughout the progress of the project, we tried to achieve these specifications. If some problems have arisen in this context, some modifications and configurations were made to them as stated in the system requirements specification in IEEE Standards Guide [15]. Then, we have expressed them accordingly. The changed statements **from CM3 up to CM4** are shown with bold letters.

Autonomous

1. The anomaly detection is performed online on multivariate data (by analyzing at least two features and their relations).
2. The final product is connected to multiple real-time data sources.
3. Different network traffic data types/information sources are monitored including at least security events, network events, network attacks.
4. Final Product is provided to the user via UI/UX on an application.
5. The anomaly statistics are displayed with live charts and histograms.

6. Model fitting of each model speed is at most $\sim$**0.5s** for each data point.

7. Anomaly detection works in real-time, with capturing live data, data processing, and anomaly detection performed under $\sim$2s for one network data packet.

8. Event per second (EPS): up to 200-250 with the source number: $\sim$4-5 (EPS observed in network traffic of one host: $\sim$50).

9. The chart's update time for each event detection is under 1 minute.

10. Data acquisition of 10 packets is under 1s.

    The models are fast to detect the anomalies early as much as possible and they are evaluated depending on the proper evaluation metrics. The performance metrics to evaluate the algorithms are provided by PySAD libraries as well as scikit-learn libraries.

    For testing purposes, since obtaining AUROC, F1 score and PR curve (the details of these evaluation methods are given in Anomaly Detection subsection of Methods Section) is difficult while performing online anomaly detection, we saved the anomaly scores in online detection and found these metrics after detection operation.

11. Area Under ROC > 0.6, also it is compared with some related anomaly detection works [16], [17]

12. F1-Score > 0.4 [18]

13. The PR curve of the algorithms is above the no skill line (number of anomalous samples/number of all samples) [19].

User Controlled

14. The user is able to determine a confidence percent threshold between 0-1 to distinguish anomalies depending on their anomaly output score.

15. The user is able to determine the hyperparameters of the algorithms, if they do not do so, the default parameters will be used.

16. The user is able to download the results in CSV format.

17. The UI is highly customizable in terms of time range and adding thresholds.

2.2 Non-Functional Requirements

Maximum Allowable Cost

Most of the methods and tools we are going to use are free except Kafka. Also, the maximum allowable cost for this project is 5000 TL according to a grant provided by TÜBİTAK.

Maximum Allowable Size, Weight

The maximum allowable size, weight is not relevant to the project since it is implemented on software.

Power Requirements/Constraints

The power consumption of a common laptop or notebook is up to 100W [20].

Environmental and Other Issues

The system can run on any operating system including Windows, macOS. The user interface is desired to be clean and user-friendly to facilitate the usage of the application. For this reason, IBM QRadar SIEM is an important example for the design of our project.

Safety Issues, Health Constraints and Requirements

There is no safety issue and health constraints related to our project since the project will be implemented on software only.

Global, Cultural and Social Factors That Can Affect the Design of the Project

The project aim is anomaly detection for diagnostics using big data. Indeed, the anomalies that we deal with provide a basis for cyber attacks. Nowadays, network threats and cyber attacks in information systems have become a global issue. In this context, this project can be used as UEBA (User and Entity Behavior Analysis) by combining with different SIEM tools. The UEBA market is expected to be worth 908.3 Million USD by 2021 [21]. Moreover, it is known that “databases and private information that is stored in governmental institutions, to power plants, communications networks and navigation systems” are under the threat of cyber-attacks [22]. Hence, governments take policies and preventative measures to provide cybersecurity.

3 Big Picture

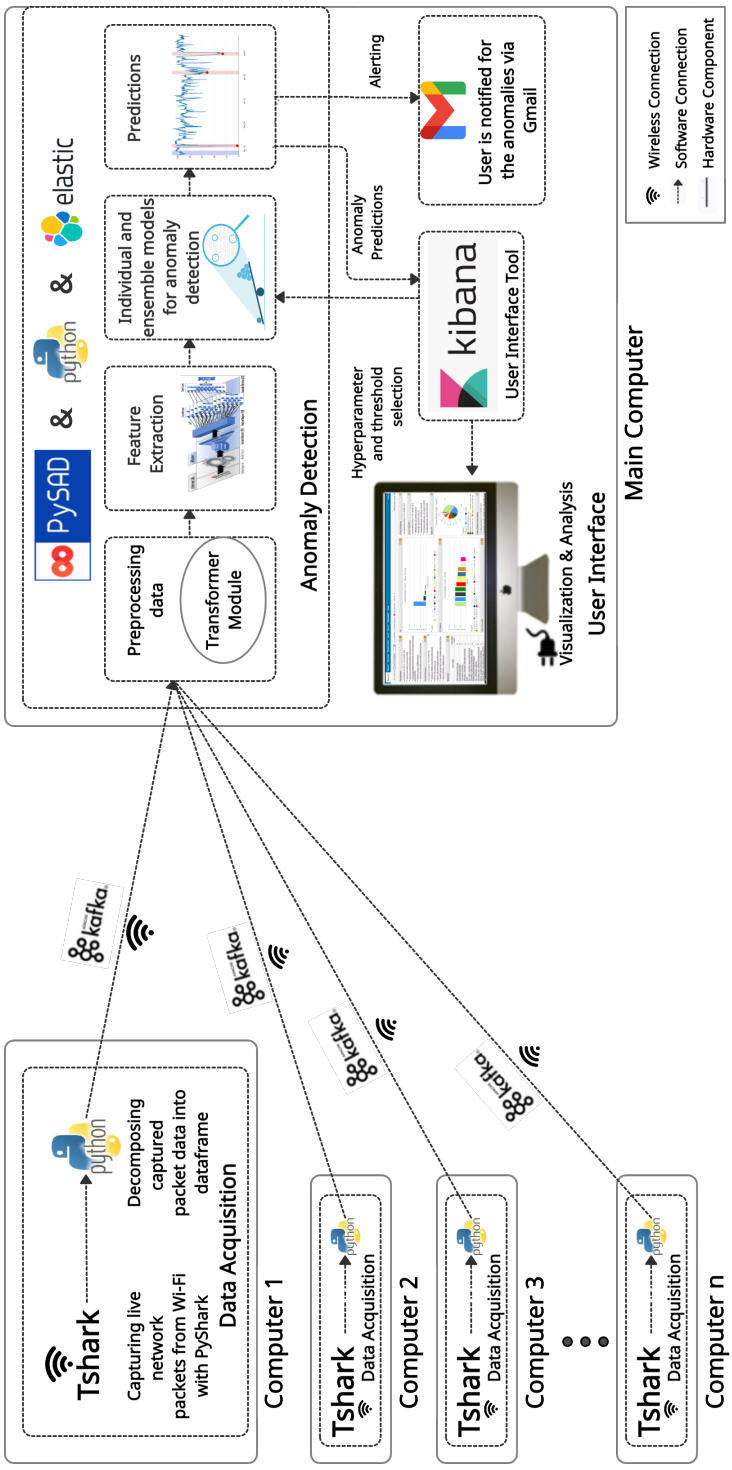


Figure 3: Big Picture of the project.

The starting point of the project is the data acquisition part. The **online streaming network traffic data** is acquired from multiple real network data sources via **wireless connection**. Network traffic data is acquired in each computer separately as displayed in Figure 3. Then, each network traffic data is sent to the admin computer for anomaly detection. It is preprocessed before the anomaly detection, then, it enters the anomaly detection module. The user is alerted via mail when an anomaly occurs. Finally, the anomaly predictions coming out from the anomaly detection module are displayed and analyzed on the User Interface application.

Data Acquisition

Then, the network traffic packets from online streaming network data are captured with **TShark** via **Wi-fi IEEE 802.11 a/b**. Packet-based features are extracted by the Python library named PyShark. The packets with raw features are put into dataframe and sent to the transformer module via **Confluent Kafka**. In the transformer module, the data is preprocessed so that it is suitable for the anomaly detection algorithms.

- For the sake of the performance and speed of the acquisition part, the data is obtained lively without converting it to CSV format.
- It is transformed to **dataframe.**
- Unnecessary packet features are filtered out.
- Categorical data types, e.g. strings, are converted to integers.
- The data is normalized by **min-max normalization** at the end.

Finally, the processed data is sent to the anomaly detection part.

Anomaly Detection

In the anomaly detection part, the **PySAD** library and Python libraries including Sklearn and Tensorflow are used. PySAD is used for online anomaly detection algorithms. **Sklearn and Tensorflow** libraries are used for the feature extraction part.

The relevant features are extracted from incoming data and given to the anomaly detection models. The outputs of the anomaly detection are the anomaly probabilities of the algorithms as the predictions. The predictions and some network packet information are sent to **Elasticsearch** to be indexed.

- The anomaly scores are obtained from the models and their ensembles.
- The scores are evaluated to determine whether they are anomalous or not.

User Interface

The anomaly predictions are displayed on a user interface provided via **Kibana**. They are visualized with live charts, graphs, and histograms.

- User can change the hyperparameters of the algorithms in the beginning and change the threshold in the graphs anytime.
- Time intervals of the graphs are customizable.
- If the system detects an anomaly, it alerts the user via e-mail.

4 Methods and Implementation Details

4.1 Work Breakdown Structure and Project Plan

The project is broken down into six work packages. There are three main parts: data acquisition, anomaly detection, UI/UX design and there are three side parts: planning, research, and testing. Milestones and success criteria are specified for all the work packages. The work breakdown structure (WBS) is in Figure4.

4.1.1 Planning

Tasks in this section aim to understand the project requirements by meeting with the project advisors, arrange expenses by preparing the equipment list and construct a timeline for the project by a Gantt chart.

Milestone: Making a tentative project plan.

Criteria for success: For all members to understand project requirements clearly, fair division of the work packages based on the interests and capabilities of the members, a reasonable equipment list with clear costs, and having a clear timeline for the whole project.

4.1.2 Research

Tasks in this section aim to understand the technical steps by researching anomaly detection through literature review, studying network data, and figuring out what anomalies mean in the network context. Since the network data can flow from many different sources, choosing the network sources to analyze is another work package. Research also contains studying similar products or articles recommended by the advisors.

Milestone: Clear understanding of network data & anomaly detection.

Criteria for success: For the relevant members to understand their parts of data acquisition/ anomaly detection and having found decent tools for the whole project.

4.1.3 Data Acquisition

This section contains tasks on obtaining the network data from different sources via Pyshark and decomposing network data packages to pandas dataframe format. This is directed to the transformer module via Kafka to be prepossessed. Sample datasets are prepared from existing simulated network attack datasets.

Milestone: Transformer module.

Criteria for success: For the transformer module to meet speed requirements and extract meaningful data from network data packages into a common format used in anomaly detection applications.

4.1.4 Anomaly Detection

The anomaly detection section aims to apply anomaly detection on the data coming from data acquisition transformer modules. Data visualization is done to understand the data better. ML models are used from the PySAD library. Ensemble models are created from combinations of the models.

Milestone: Anomaly detection module.

Criteria for success: For the module to meet preselected performance metrics in anomaly detection and anomaly classification while fulfilling speed requirements.

4.1.5 UI/UX Design

The anomaly detection module is connected to the Kibana (UI component) via Elastic. UI and UX is designed.

Milestone: Operating UI.

Criteria for success: Clean and user-friendly UI.

4.1.6 Testing

In this group, feedback is taken from the advisors and possible advancements in the performance of the anomaly detection algorithms and adjustments in the product is made.

Milestone: Final product.

Criteria for success: For the final product to meet all functional and non-functional requirements.

The WBS and The Gannt chart for the project containing the responsible members for each work package, time durations, and milestones are shown respectively in Figures 4 and 5.

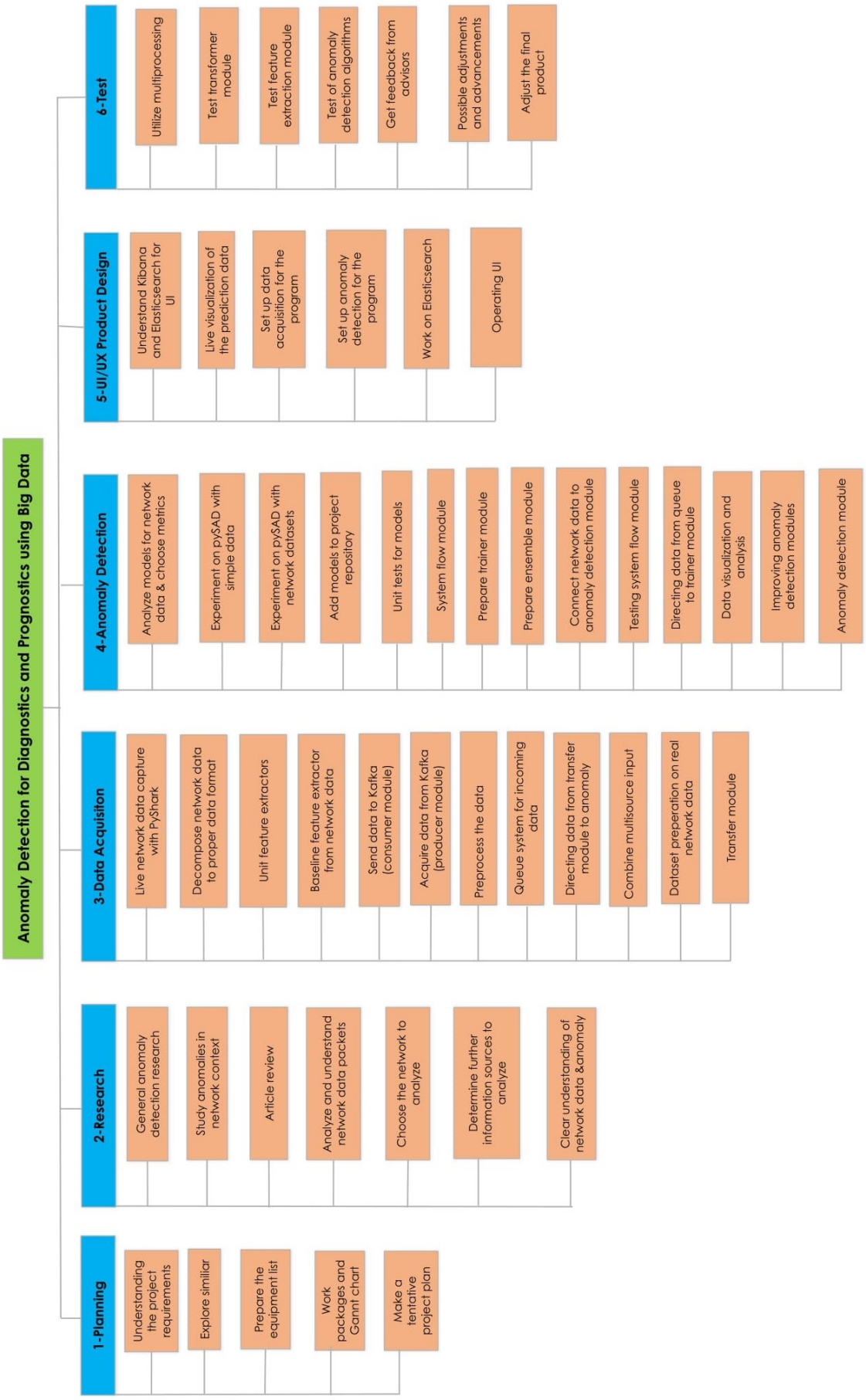


Figure 4: WBS for the project.

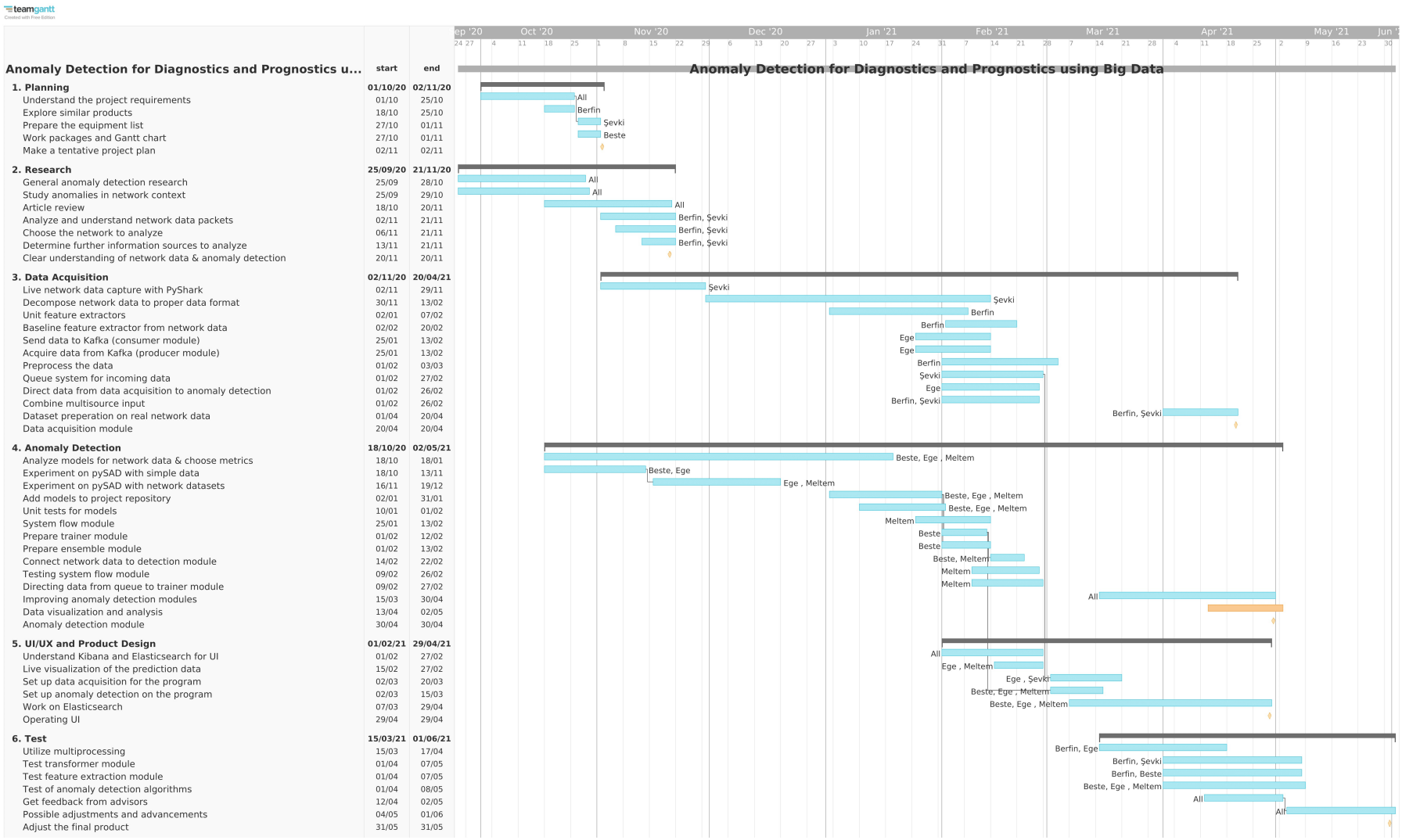


Figure 5: Gantt chart for the project.

4.2 Methods

Solution strategy and methods are explained for three main major tasks, i.e. data acquisition, anomaly detection, and UI/UX design. These sections are divided into two to represent the work done from the beginning up to CM3 and since CM3.

4.2.1 Data Acquisition

4.2.1.1 Up to CM3

Live network traffic data acquisition was completed. Packet-based features were studied and preprocessed in the transformer module accordingly. Available network traffic datasets were researched and the NSL-KDD dataset [23] was studied for preprocessing data. Feature selection in the NSL-KDD dataset was implemented for understanding dataset properties and model performance.

NSL-KDD Dataset
NSL-KDD is used in detecting novel attacks to overcome the weaknesses of signature-based IDSs, which is one of the main purposes of our product. Moreover, it is benchmark data for comparison with other studies [23].

NSL-KDD dataset was used for understanding network traffic data and creating a transformer module. PCAP and CSV files are available [5]. CSV files were already processed for developers and contain extracted features from network packets such as strings and numerical data.

There are 43 features available in CSV files. One feature is the difficulty level of classification of the packet with the target label. There are 21 predicted labels for attack types, which will be categorized into 5 classes as main attack types in a simple transformer module.

Capturing Live Network Data
Using Wireshark itself, we can transform PCAP files to CSV [24], but in this way all of the information that PCAP contains, whether it is important or not, will be transferred to the CSV file. Another way is using TShark's command [25] on CMD via Python. Instead of using TShark's command, we searched for wrapper libraries in Python for TShark and found PyShark [26]. We had simple trials with Pyshark for acquiring data in Live Capture mode and File Read mode.

We have decided to use PyShark for live network data capturing for two main reasons. Firstly, since it's a Python wrapper, we can directly get the data and extract network packet features without dealing with file reading and format changing, which saves significant time for our system. The other reason is file storage might be a problem in the future.

Here, you can see the network packet features which are extracted from packets via PyShark:

- **Packet information:** Date & time, duration, packet length, source hardware MAC address, destination hardware MAC address.
- **TCP:** Time, source port, destination port, data length, sequence, sequence raw, next sequence, acknowledgement, acknowledgement raw, TCP flags, RES flag, NS flag, CWR flag, ECN flag, URG flag, ACK flag, PSH flag, RES flag, SYN flag, FIN flag, STR flag, window size, checksum, checksum status, urgent pointer, stream, protocol name.
- **UDP:** Time, source port, destination port, data length, stream, protocol name, checksum, checksum status.
- **IP** Source IP, destination IP, protocol, differential service, flags, IP version.
- **IPv6:** Source IP, destination IP, IP version.
- **ARP:** Proto type, proto size, hardware type, hardware size, hardware opcode, source hardware MAC address, destination hardware MAC address.

As can be seen above, some of the layers are alternatives to each other, and therefore they are not captured at the same time. Like TCP and UDP protocols, they have some common features such as time, source port, destination port, etc., but TCP has more features. Because of that, when we obtain UDP layer, we are assigning 'None' to the remaining TCP features which are not common. IP and IPv6 are also alternative layers of each other. We are assigning 'None' to IP layer's non-common features. ARP is also an alternative layer for IP and IPv6. We are assigning 'None' to the non-common features of these alternatives.

Network Data Transformer Module

Network traffic packets contain features which are binary, nominal, integer, and real variables [27]. Their data types are both strings and numerical data, therefore preprocessing was necessary. One sample packet is shown in Figure6. This packet information is obtained on Python by Pyshark.

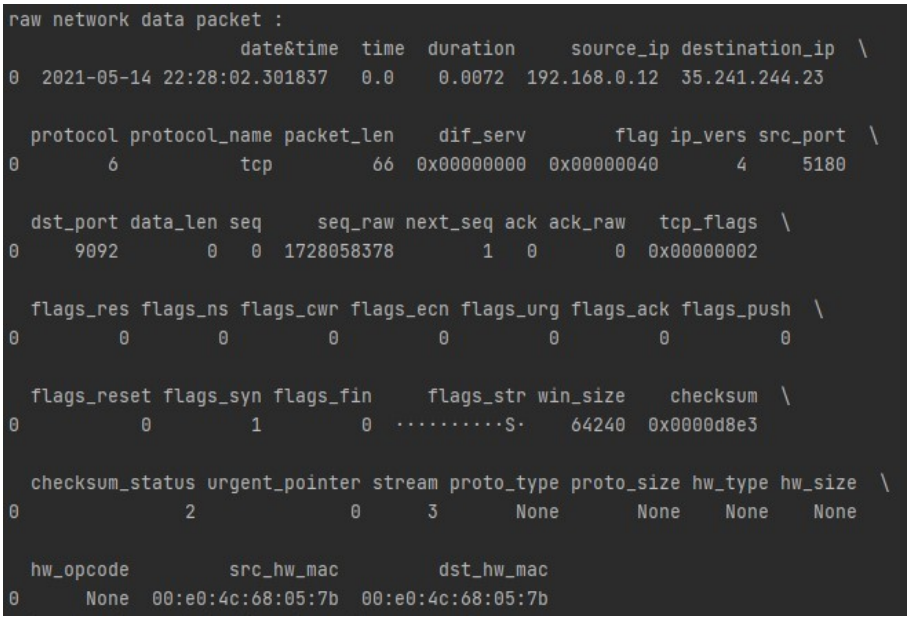


Figure 6: Raw network data packet.

For preprocessing, integer representation and min-max normalization (normalization in 0-1 range) was proposed in similar applications [8]. We needed to implement these methods on time-series data and use them in online learning.

For integer representation of protocols, global protocol numbers are extracted. For features in bits and hex, we used their integer equivalents. For IPv4, IPv6, and MAC addresses, we again calculated their integer values. For 'None' values, representative integers are assigned. For example, 0 and 1 are captured for the RES flag. When the UDP layer is captured, this flag will be 'None' because it is a property of the TCP layer. We are assigning 0.5 for 'None' in this case.

After integer representation of all features, we normalized them by dividing them to their maximum possible value assuming that all features were positive. For instance, if we were obtaining a 16-bit valued feature, we divided it by 65536.

Data Storage and Transferring

To bridge the connection between data acquisition and anomaly detection modules, or in the hardware perspective the connection between source devices and the main "admin" device (where all detection processes happen), we are using a data storage solution called Kafka [28]. Kafka stores times series data and unlike databases, it does not work as a table structure. It can be understood as a pointer that points to the next data instance as the data is used. Hence it's much easier to apply to the project. Additionally, the data is loaded to topics, which can be utilized to create a multi-source structure where each device has its own topic and its data can be reached from this topic.

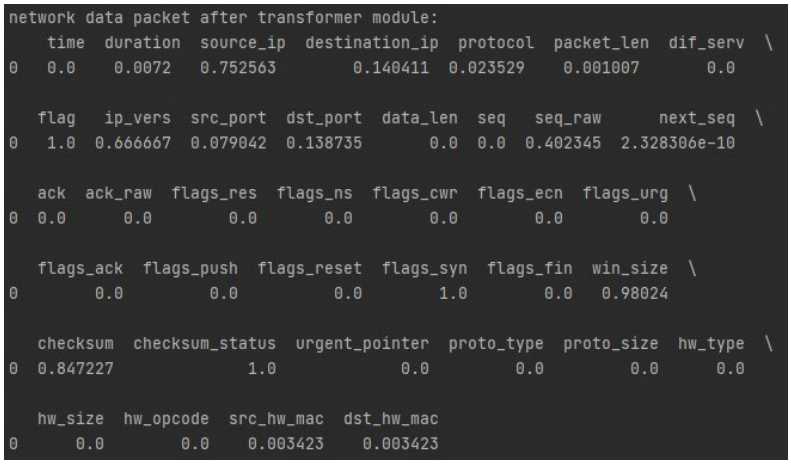


Figure 7: Network data packet after Transformer Module.

Feature Selection & Extraction

Feature selection and extraction are important in Machine Learning due to reducing overfitting, improving accuracy, avoiding the curse of dimensionality, and reducing training time. Extracting a new set of features in a lower dimension space is the main purpose here.

There are three main feature selection methods: filter-based, wrapper-based and embedded [29]. We worked on univariate filtering from filter-based feature selection. In univariate filtering, the statistical scores between target labels and each feature is analyzed separately, which means data should be labeled [30]. F-test and mutual information statistics were computed for all features in the NSL-KDD Dataset, which are methods of univariate filtering.

We observed that statistical information calculated from the whole dataset was relatively more important for attack detection as their F-test and mutual information scores were higher. This was also explained in similar studies. When different feature selection methods were applied, features such as a sum of connections to the same destination IP address were scored higher [27].

For feature extraction, autoencoder and PCA were proposed to be used with the aim of dimensionality reduction. Similar applications were seen in a relevant article on feature dimensionality reduction by using the CIC-IDS2017 network intrusion dataset [8]. Our company advisor suggested using various feature extractors and the following feature extractors were implemented: autoencoder, PCA, LSTM autoencoder, and K-means clustering.

4.2.1.2 Since CM3

The available network traffic datasets are usually in CSV format and are already processed. We have started with the NSL-KDD dataset for simulations but we have seen that statistical features were extracted in the NSL-KDD dataset [27]. Other network traffic datasets have similar feature extraction techniques. As a result, we had to create our own dataset from available PCAP files consisting of attacks so that we could use these datasets for testing purposes in our simulations.

Datasets

We used two datasets with simulated network attacks: Kitsune Network Attack Dataset and Intrusion Detection Evaluation Dataset (CIC-IDS 2017). Kitsune Network Attack Dataset is found both in PCAP and CSV format, but we are using PCAP format. CIC-IDS 2017 is in CSV format. We can read packets from these datasets one by one as we did in capturing the Live Network data. Our data acquisition module is performing exactly the same as in Live Capture mode. We are reading from datasets and sending them one by one using Kafka to the transformer module.

- **Kitsune Network Attack Dataset**

  Kitsune Attack dataset is prepared a controlled environment by simulating attacks [6]. It consists of 9 attacks: Active Wiretap, Fuzzing, Mira, OS Scan, SSL Renegotiation, Video Injection, SYN DoS, and SSDP Flood attacks. In these simulated attacks network packets are labeled as normal until the controlled attacks start after that they are labeled as anomalies. In other words, first, normal packets are coming constantly, and then attacks start at the end of the dataset. Using the raw PCAP files of Kitsune Dataset, we have created a dataset consisting of 45000 network packets of all 9 attacks. An anomaly rate in the dataset is 0.1. We have read the raw PCAP files and passed them from the transformer module. It had 39 features and final size was 45000x39. Then, we have used the dataset as both this form and the form after passing from the LSTM autoencoder.

- **Intrusion Detection Evaluation Dataset (CIC-IDS 2017)[31]**

  This dataset consists of some simulated attacks such as Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet, and DDoS. Compating with Kitsune in this dataset normal and attacked network packets are mixed. These datasets were prepared on daily basis for one week long. We have used Friday-WorkingHours-Afternoon-DDos in CSV format. In this dataset, features are already extracted with CIC-Flow Meter [32].

  It has 80 features and 225711 packets. The augmented features are based on flow-based packet features: forward/backward inter-arrival time, flow inter-arrival time, the amount of time a

flow was active/idle, flow bytes per second, and flow packets per second. Except for the last two features, statistical information of mean, minimum, maximum, and standard deviation are calculated by CIC-Flow Meter. An anomaly rate is 0.567 [32].

Feature Extraction with LSTM Autoencoder

Autoencoders are used to reduce dimensionality while preserving the information with minimum loss. Autoencoder is a neural network structure having a latent layer with less dimension as a representation of the input data. It is an unsupervised method since the input and output are identical. LSTM autoencoder is working with the same idea. For hidden layers, LSTMs are used to extract the temporal state information [33]. As shown in Figure8, the encoded data is taken from the latent vector. Here, both state information is extracted and dimension is reduced. In our project, the packet vectors are taken one by one, and the model is trained by the packet vectors.

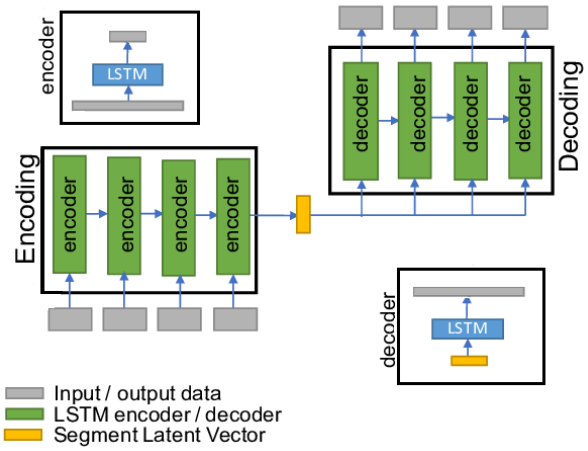


Figure 8: LSTM autoencoder structure [34].

4.2.1.3 Unfinished Tasks

Feature extraction based on windowing could not be finished because we mainly focused on dataset preparation. Also, the results raise the question that more feature extraction techniques, e.g. flow-based feature extraction, could be applied; however, this task requires good knowledge of the network domain. We may have failed to extract meaningful data which was an expected risk in the project.

Dataset preparation was a challenge from the beginning of our project as live network capture was our priority for the final product. It was hard to find a dataset suitable for our project.

The available datasets were based on separate attacks. For the ones which had various attacks together sequentially, e.g. CIC-IDS2017, PCAP file sizes were extremely large and beyond the memory capacity of our computers. We were planning to use datasets supplied by our company but this plan was suspended and we were suggested to create our own datasets from available PCAP files.

4.2.2 Anomaly Detection

4.2.2.1 Up to CM3

For this time period, we have mostly focused on research and tests on anomaly detection algorithms. We decided on the libraries that we would utilize throughout the project, which are PySAD and Kitsune. PySAD has many different algorithms such as xStream which gives us the option to choose the most optimal algorithm for the problem at hand [4]. We did simple tests on sample network datasets to understand how the library functions [35]. One example is the trials that we did on the NSL-KDD dataset. This dataset was only used to increase our intuition and understanding of the algorithms in this period and is not suitable for our problem since it contains additional extracted features from packet information that are not possible to extract from time-series network traffic data. Additionally, we have tried different algorithms such as Kitnet and LODA in this period in addition to the algorithms that we are presenting in this report [6]. However, these algorithms were not really solutions to our problem. Kitnet does semi-supervised learning, which we could not use since our project is based on unsupervised anomaly detection. LODA was fit to our problem at hand but unfortunately, because of internal problems in the library, it could not work properly. Hence we are presenting three models: xStream, iForest, and an Ensemble Model. The next section will detail these models.

4.2.2.2 Since CM3

iForest

iForest algorithm (Isolation Forest) [36], uses a tree structure to explain the anomalies by separating them away from the normal points. An ensemble of random trees is built to create a random forest. According to the algorithm, anomalous points are the points with the shortest average path lengths [37]. Ensemble of trees takes advantage of the rareness and dissimilarity of the anomalous points. Anomalous points are easily separated from the normal points early on in the forest. So, the average depth of the node is presented as the anomaly score. If its score is low, it has a higher probability of being anomalous. The general framework for the algorithm is given in Figure 9.

In addition to the forest structure, iForest algorithm utilizes time dependencies, which can be also informative in detecting algorithms. Each data point in the streaming system is evaluated according the the block of data points. The number of data points in the sliding window is fixed over time and is given to the algorithm as a hyperparameter. iForest does not use distance or density measures in order to detect anomalies. Therefore, iForest has a relatively low memory requirement, since it does not calculate nor store the measures. It also has linear time complexity, which is desirable.

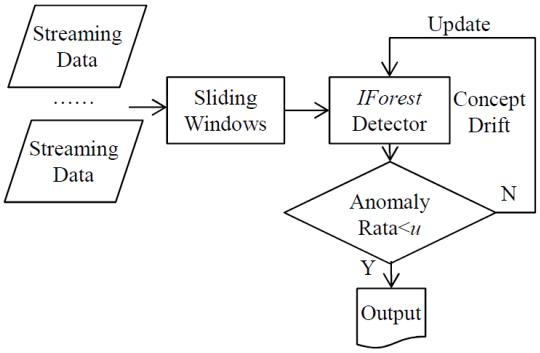


Figure 9: iForest framework that demonstrates the algorithm [36].

xStream

The algorithm is based on the "density-based ensemble outlier detector" working principle [38]. It can handle three settings: (1) static setting with known feature size and sample size, (2) row-stream setting with known feature size and unknown sample size (since it is streaming data), (3) feature evolving stream setting with unknown sample and feature size. The settings (2) and (3) are shown in Figure 10. We are not interested in the static mode because the data should be streaming in this project. In setting (3), the feature size and values may change over time and some new features may come out. However, the feature size is fixed in the project and the network packet comes one by one. Therefore, the algorithm operates in mode (2) in our project.

It "operates on projections of the data points" which have lower dimensionality and constant size [38]. To detect outliers, it makes density estimation with recursively and randomly constructed partitions. With these partitions, the data is divided into smaller parts as bins. Then, it performs as a windowing operation. The bin counts are collected and in the window to make evaluations on the next incoming data point. In this way, an outlierness score is generated. The window starts sliding when it has a predetermined number of data points. This predetermined number is the hyperparameter of the algorithm and it can be changed. We have decided to use this algorithm in this project since it applies to streaming data and can find the outliers in the data. Moreover, we have found out that it has many real-world applications including security.

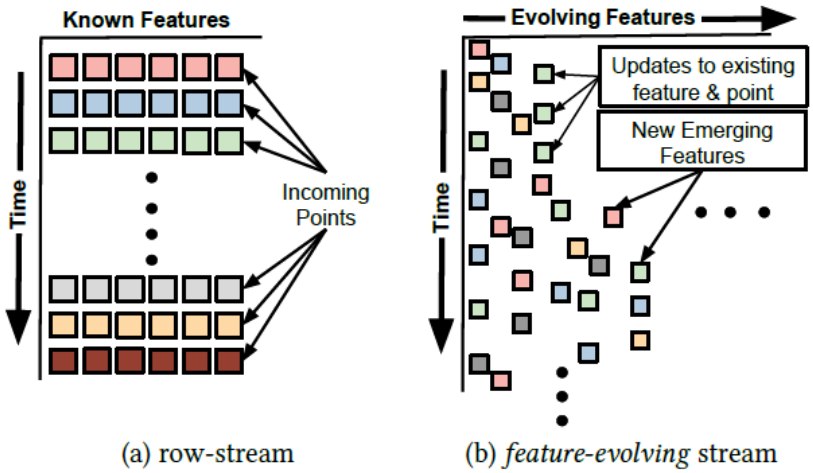


Figure 10: Two situations demonstrated that xStream can handle [38].

Ensemble Model

The ensemble model uses the output scores from the above two algorithms (xStream and iForest) as inputs of another model. This model is called the Average Ensembler Model and is part of PySAD similar to the algorithms that we are using [4]. The average ensembler gets the weighted average of the input scores and outputs a new score based on its weights. These weights change over-time and as new data is introduced, hence this part of the model also learns. The addition of these three models is our Ensemble Model.

4.2.2.3 Unfinished Tasks

Although we have reached most of the goals that we have set out to accomplish, there were still some improvements that we could've pursued. Scheduled training was something that we thought would improve the product. This would increase the efficiency of the general flow since the program would spend most of the time predicting and would learn at certain periods of the day. Another improvement could be packaging the program using a service such as Docker, this way the product can turn into a single executable for ease of use. Finally, group anomaly can be researched and added to the project.

4.2.3 UI/UX Design

4.2.3.1 Up to CM3

In this period, we mainly focused on research for the UI of the system. For the UI of the project we use Elasticsearch and Kibana. Elasticsearch is used to transfer data from the code

to be visualized in Kibana. Kibana can be used to visualize data in different ways. In this period we worked on utilizing Kibana to its full potential and connecting the code with Kibana using Elasticsearch. We have achieved our goal of showing different models in different plots, updating these plots live and also giving additional info to the user using tables in the UI.

4.2.3.2 Since CM3

Since we had mostly finished the flowing system and most parts of UI, we focused on the only incomplete requirement since CM3, which was concerning alerts and threshold selection. In order to set up an alert system, we implemented an e-mail sender code that can send e-mails to a predefined address when an anomaly is detected. If the probability of an anomaly is above the threshold, the user is alerted via e-mail. This is demonstrated in the Results section for UI. Before the user selects a threshold, it is set to a default value of 0.7. If the user decides to change the value of the threshold, the change request is taken via the Kibana interface and a JSON request. The code handles this request to change the threshold used in algorithm detection. The alarm system is also modified so that it operates according to the current value of the threshold.

4.2.3.3 Unfinished Tasks

We have finished all of our goals in the UI section of the project but some improvements can be pursued such as using Elasticsearch more efficiently. This would perhaps be for some kind of data storage solution or using Elasticsearch machine learning frameworks to achieve better results in the anomaly detection part. Also, the option of choosing the algorithms can be given to the user. Nonetheless, we have achieved all that we set out to do in this part.

5 Results, Discussions, and Future Directions

5.1 Results

The flowing systems start with one of the computers sending its network data to Kafka, retrieval of this data, the transformation of the data to a vector format, feature selection on this vector, anomaly detection of the obtained network packet vector, and the final result on Elasticsearch user interface Kibana.

Achieved Results: The functional requirements 1-5 which are about the general structure of the system are met in the final product. The final product is a flowing system both applicable

to live capture mode and file read mode for network traffic data. The system can implement point and contextual anomaly detection on multivariate data with single and multiple sources. The infrastructure to connect multiple network data sources is built up. Results are displayed via the User Interface application. In addition, all non-functional requirements are met. Other results are listed as performance results (for models on different datasets), computational speed results, and UI/UX results.

5.1.1 Data Visualization & Analysis

- **Kitsune Dataset**

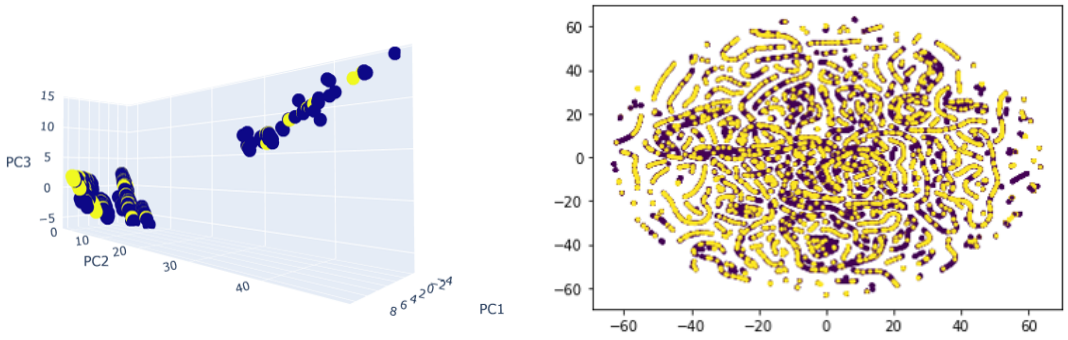


Figure 11: 3D PCA plot and t-SNE plot of Kitsune Dataset.

The 3D PCA plot and t-SNE plot of the Kitsune Dataset is provided in Figure 11. According to PCA and t-SNE plots, the data set is not linearly separable or separable when a kernel is used. The normal and anomaly class looks overlapping. Therefore, it affects the performance of the algorithms in a bad manner.

- **Kitsune Dataset (LSTM Autoencoder)**

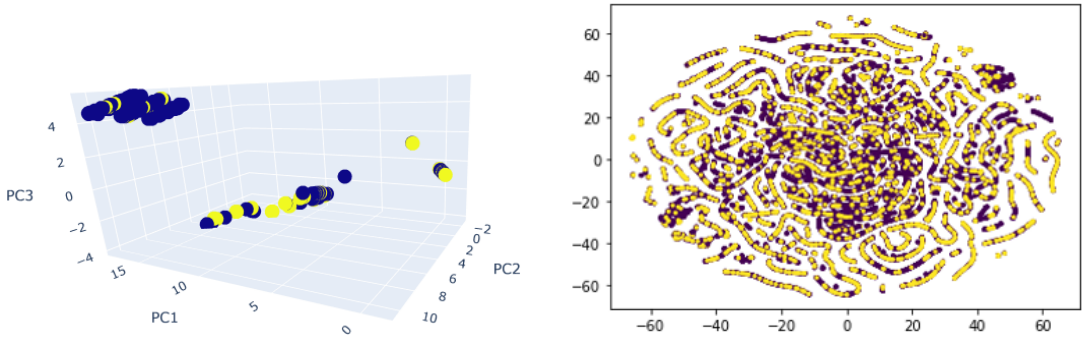


Figure 12: 3D PCA plot and t-SNE plot of Kitsune Dataset (LSTM Autoencoder).

The 3D PCA plot and t-SNE plot of the Kitsune Dataset (LSTM Autoencoder) is provided in Figure 12. The dataset does not seem linearly separable. The normal and anomaly class

looks overlapping. However, the algorithms give better and faster results since the feature size is diminished.

- **CIC-IDS 2017**

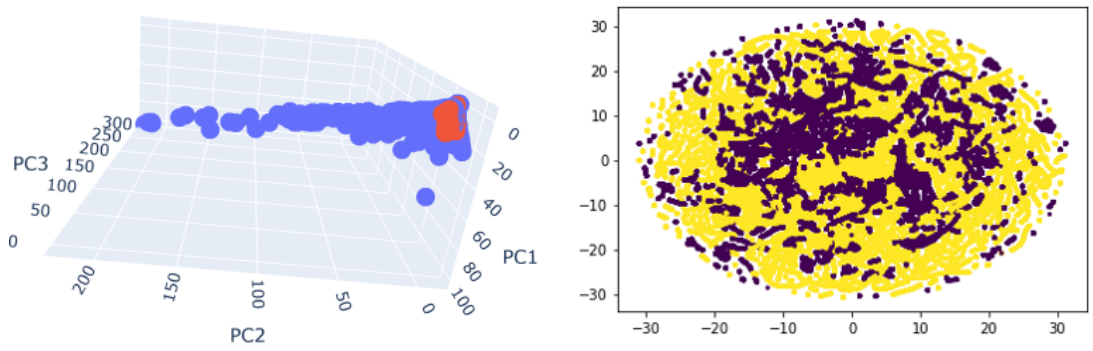


Figure 13: 3D PCA plot and t-SNE plot of CIC-IDS 2017.

The 3D PCA plot and t-SNE plot of the CIC-IDS 2017 are provided in Figure 13. The anomaly class seems gathered in a part of the plot but they are merged with the normal class. Thus, the normal and anomaly classes look overlapping. Therefore, it affects the performance of the algorithms. The speeds of the algorithms become slower since there are a large number of features.

5.1.2 Evaluation Metrics

Evaluation metrics are the Area Under ROC curve (AUROC), Precision-Recall curve (PR), and F1 score. These evaluators use frequency of true positives (TP), false positive (FP), true negatives (TN), and false negatives (FN) [39]. Using these values, the axes of the PR curve and F1 score can be found.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

$$precision = \frac{TP}{TP + FP} \tag{2}$$

$$recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1 = 2\left(\frac{p \times r}{p + r}\right) \tag{4}$$

The AUROC curve uses the false positive rate (FPR) and true positive rate (TPR) as its axes. These two values are found via equation (5) and (6).

$$FPR = \frac{FP}{FP + TN} \tag{5}$$

$$TPR = \frac{TP}{TP + FN} \tag{6}$$

Using these two values, the AUROC curve can be plotted and observed.

5.1.3 Performance Results

Below, the AUROC and PR curves, as well as the confusion matrix for each model can be found. For the functional requirement 11-13, the accuracy and the performance of the algorithms are considered. The results of online anomaly detection models, which are xStream, iForest, and their ensemble model, on different datasets, which are Kitsune Network Attack Dataset and CIC-IDS 2017 dataset, are explained here. Area Under ROC should be higher than 0.6, F1-score should be higher than 0.4, and the PR curve should be above the no skill line in order to meet the project requirements.

For better understanding, accuracy, precision, and recall are also calculated and displayed in Table 2 at the end of Performance Results part even though they are not included in the requirements list. Additionally, a confusion matrix guide can be observed in Figure 14. The Y-axis of the confusion matrix indicates actual values while the X-axis indicates predicted values.

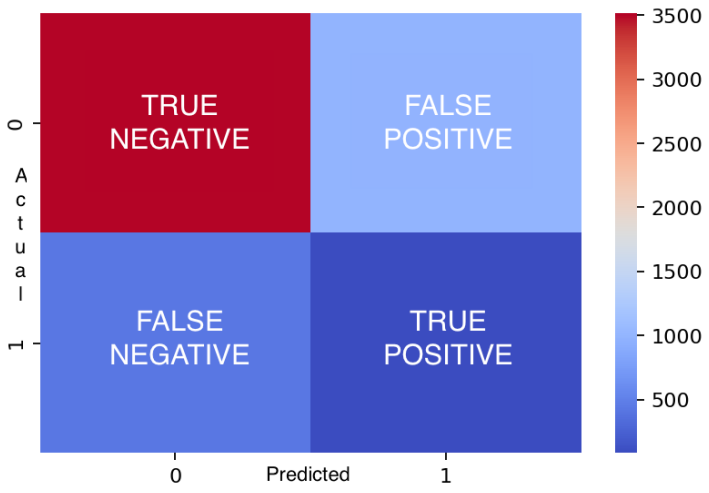


Figure 14: Confusion Matrix Guide.

- **Kitsune Dataset**

  The area under ROC curve is 0.44 for xStream algorithm and the ROC curve is under the random classifier curve. This means that the algorithm cannot find the anomalies and it learns normals as anomalies and anomalies as normals. Also, the PR curve is under the no skill line. xStream algorithm cannot work properly on this dataset.

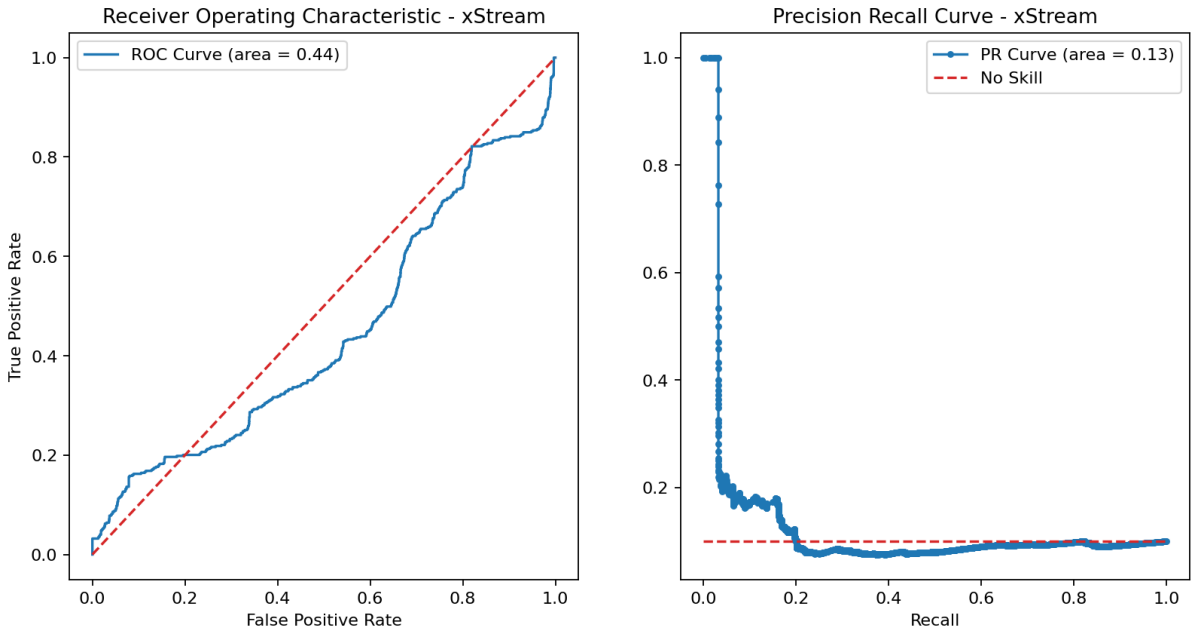


Figure 15: AUROC and PR curves for xStream.

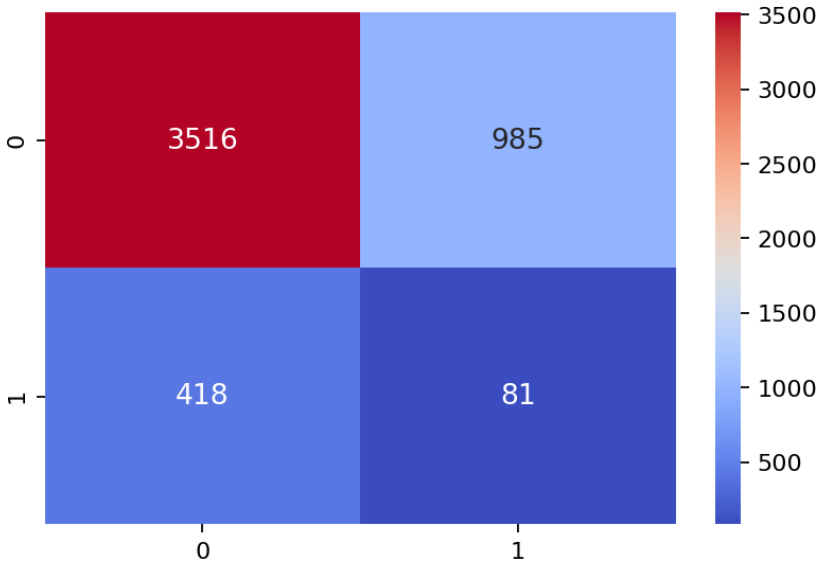


Figure 16: Confusion Matrix of xStream.

The area under ROC curve is 0.50 for iForest algorithm. The ROC curve is under the random classifier curve mostly. The algorithm cannot find the anomalies since it learns the normals as anomalies and the anomalies as normals similar to xStream algorithm. The PR curve is under the no skill line mostly. Therefore, iForest algorithm cannot work properly on this dataset, either. However, it seems relatively better than xStream when they are compared.

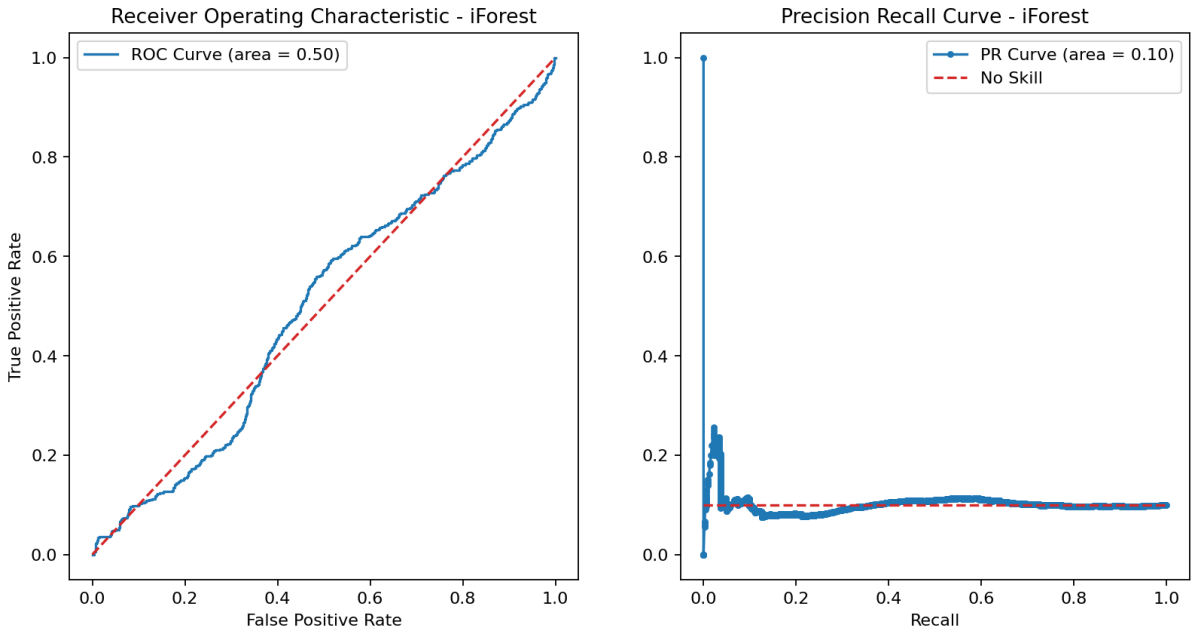


Figure 17: AUROC and PR curves for iForest.

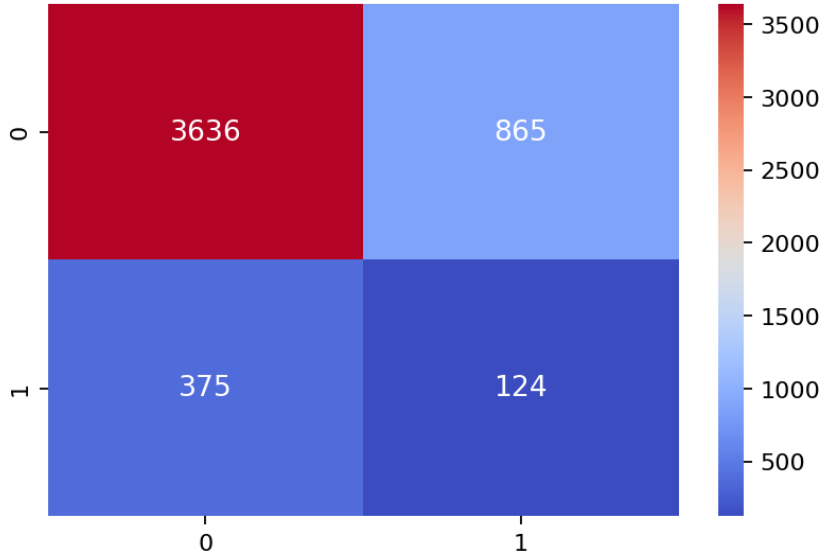


Figure 18: Confusion Matrix of iForest.

The area under ROC curve is 0.59 for the ensemble of xStream and iForest. The ROC curve is above the random classifier curve mostly. The results of xStream and iForest are weighted and averaged using the ensemble model so that it can give better results. The PR curve is above the no skill line mostly. However, it still does not meet the criterion since the area under ROC curve is not above 0.6. Therefore, the ensemble model cannot work properly on this dataset, either.

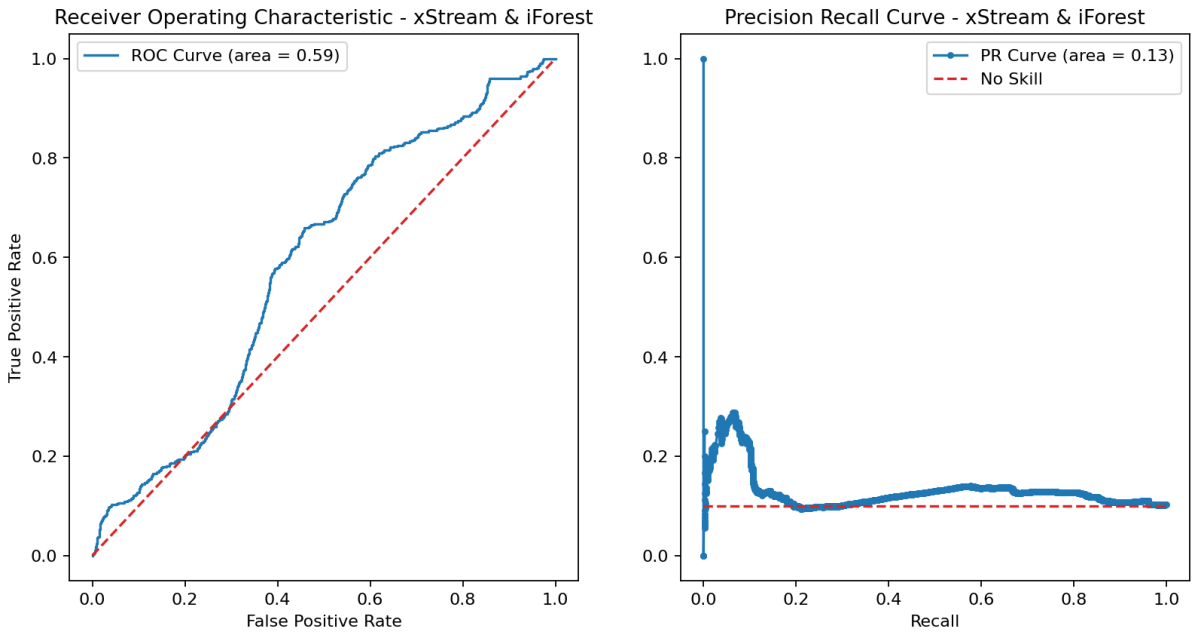


Figure 19: AUROC and PR curves for Ensemble Model.

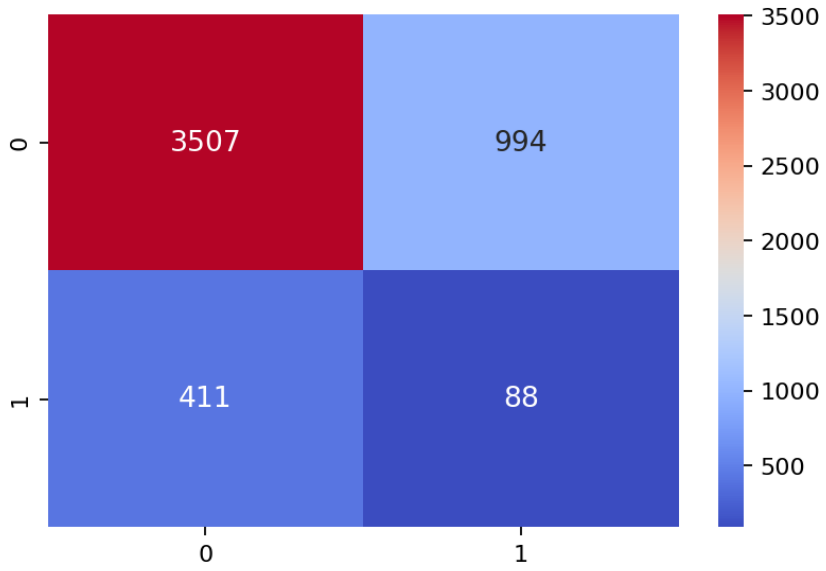


Figure 20: Confusion Matrix of Ensemble Model.

- **Kitsune Dataset (utilizing LSTM Autoencoder)**

  The area under ROC curve is 0.55 for xStream algorithm and the ROC curve is above the random classifier curve. This result is close to meet the criterion of AUROC > 0.6. Also, the PR curve is above the no skill line. xStream algorithm gives better results on this dataset.

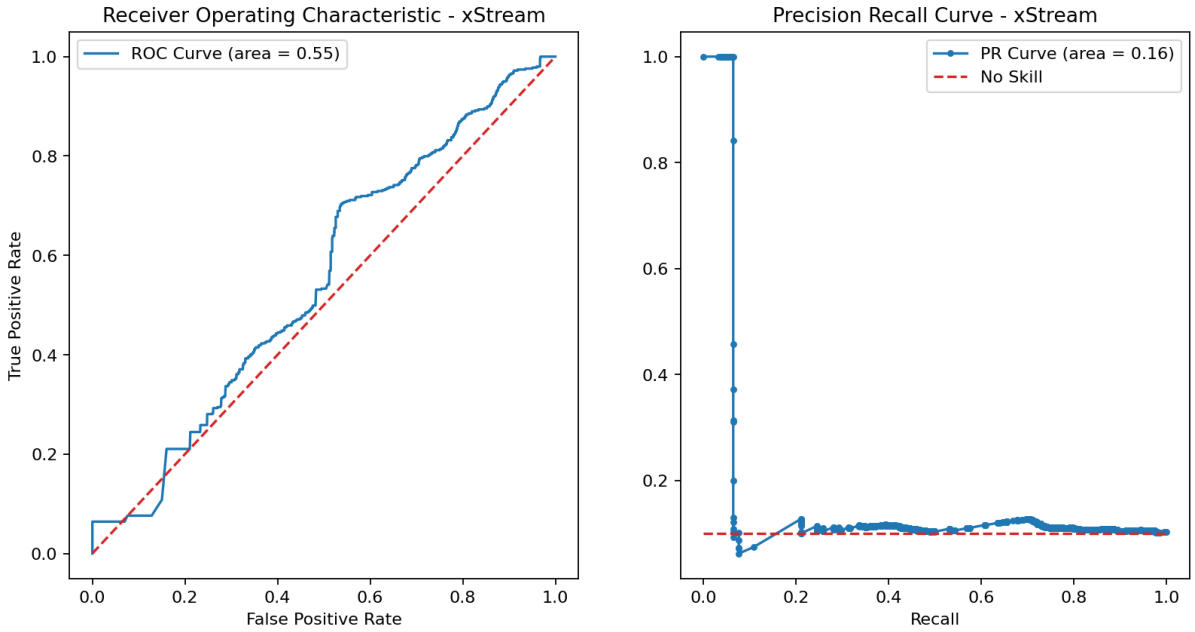


Figure 21: AUROC and PR curves for xStream.

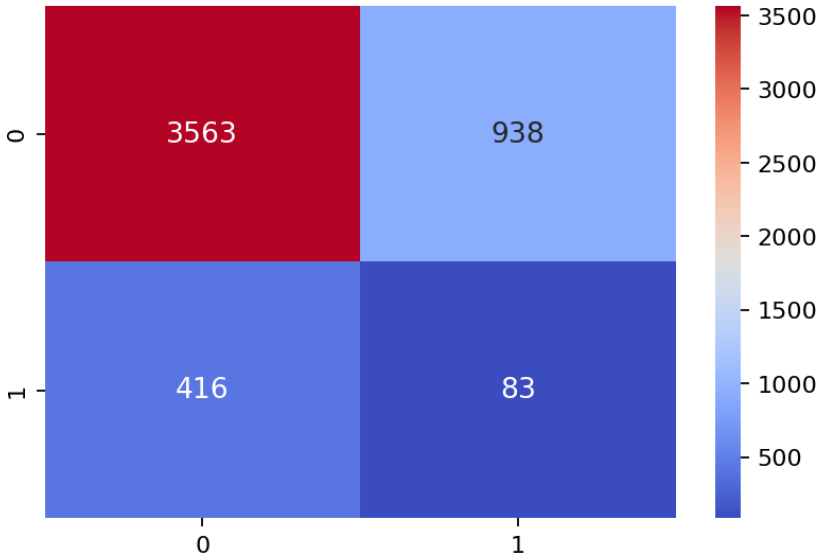


Figure 22: Confusion Matrix of xStream.

The area under ROC curve is 0.59 for iForest algorithm. The ROC curve is above the random classifier curve. The PR curve is above the no skill line. Therefore, the performance of iForest is better on this dataset compared to the Kitsune dataset without utilizing LSTM autoencoder. It seems relatively better to xStream when they are compared.

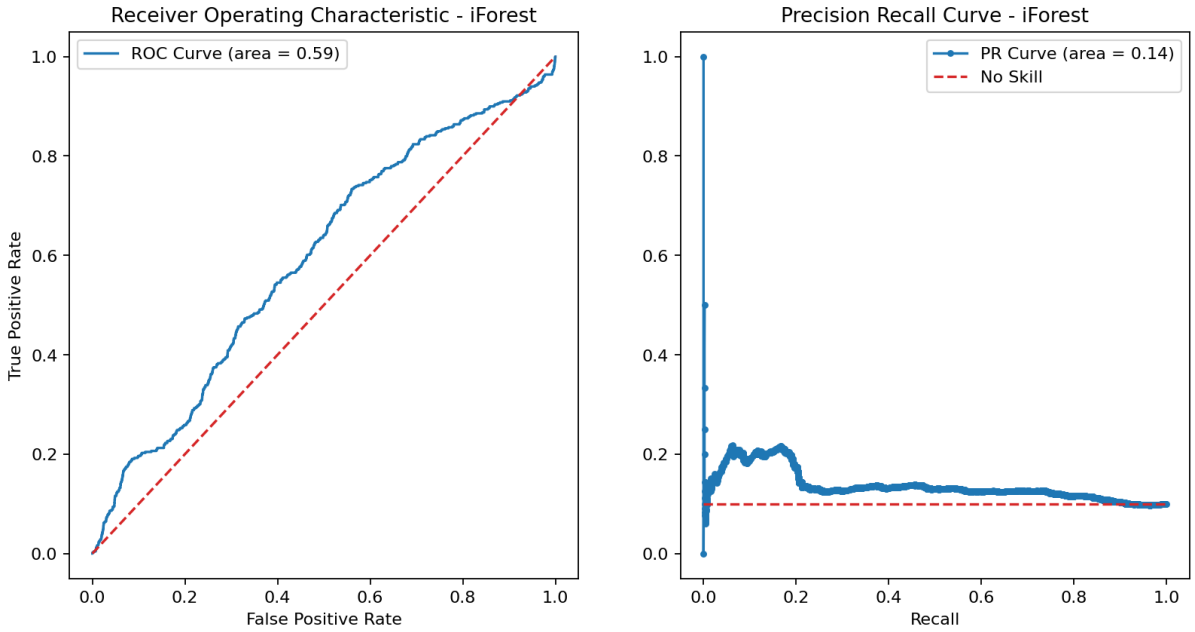


Figure 23: AUROC and PR curves for iForest.

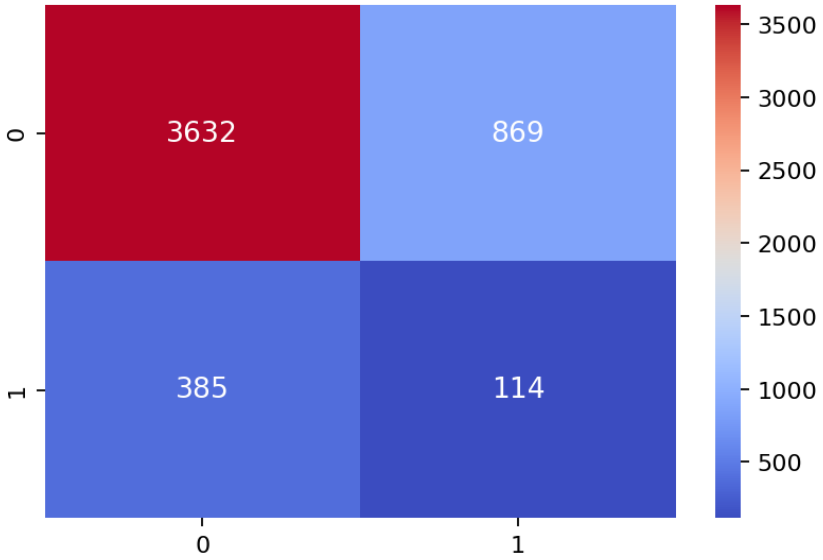


Figure 24: Confusion Matrix of iForest.

The area under ROC curve is 0.61 for the ensemble of xStream and iForest. The ROC curve is above the random classifier curve. The PR curve is above the no skill line. Also, it meets the criterion since the area under ROC curve is above 0.6. Therefore, the ensemble model can work properly on this dataset.

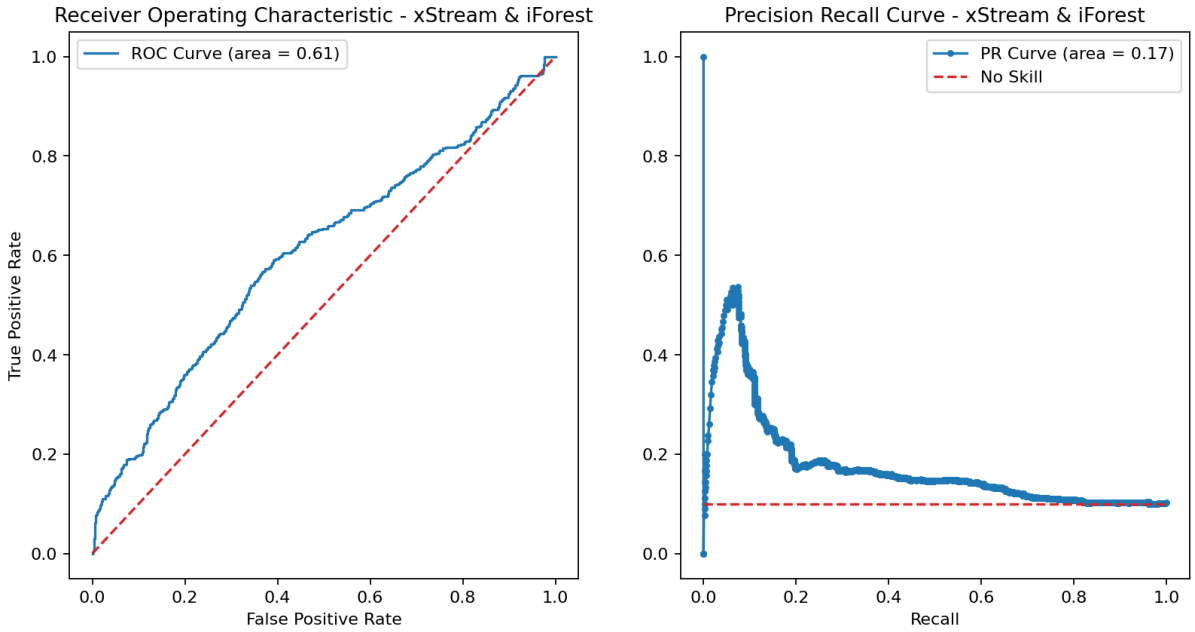


Figure 25: AUROC and PR curves for Ensemble Model.

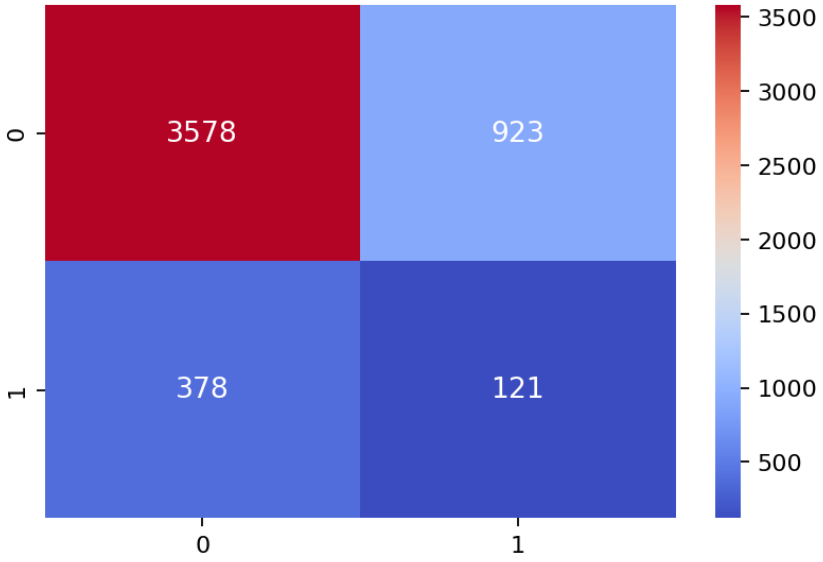


Figure 26: Confusion Matrix of Ensemble Model.

- **CIC-IDS 2017**

  The area under ROC curve is 0.40 for xStream algorithm and the ROC curve is under the random classifier curve. It does not meet the criterion of AUROC > 0.6. The algorithm cannot find the anomalies since it learns the normals as anomalies and the anomalies as normals. Also, the PR curve is under the no skill line. The algorithm cannot work properly on this dataset.

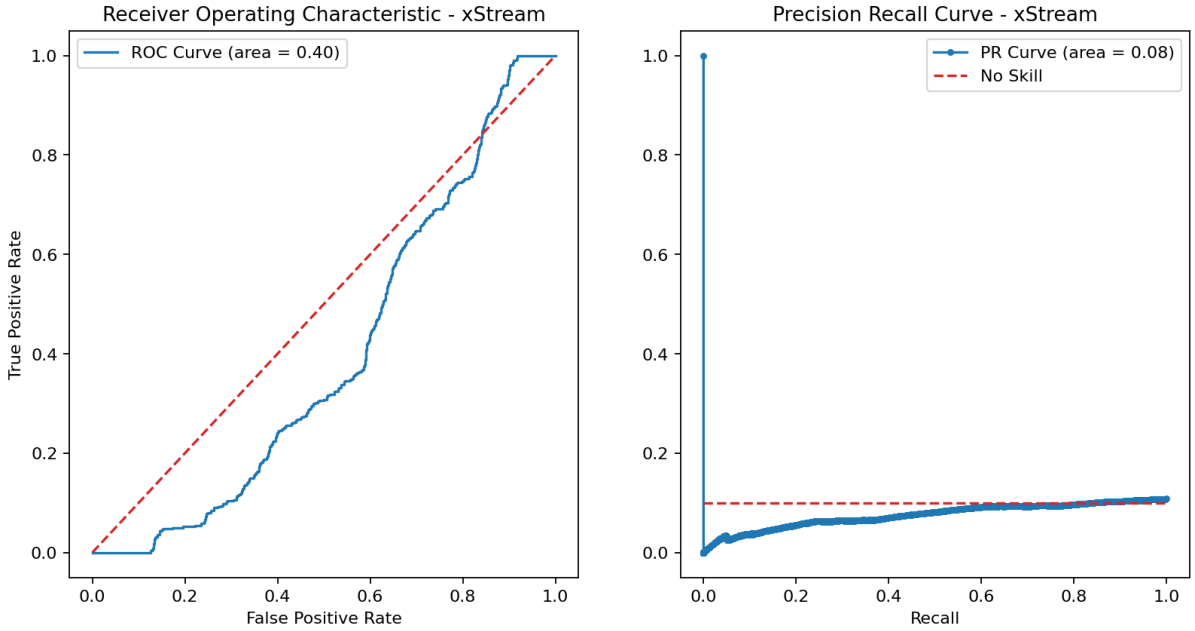


Figure 27: AUROC and PR curves for xStream.

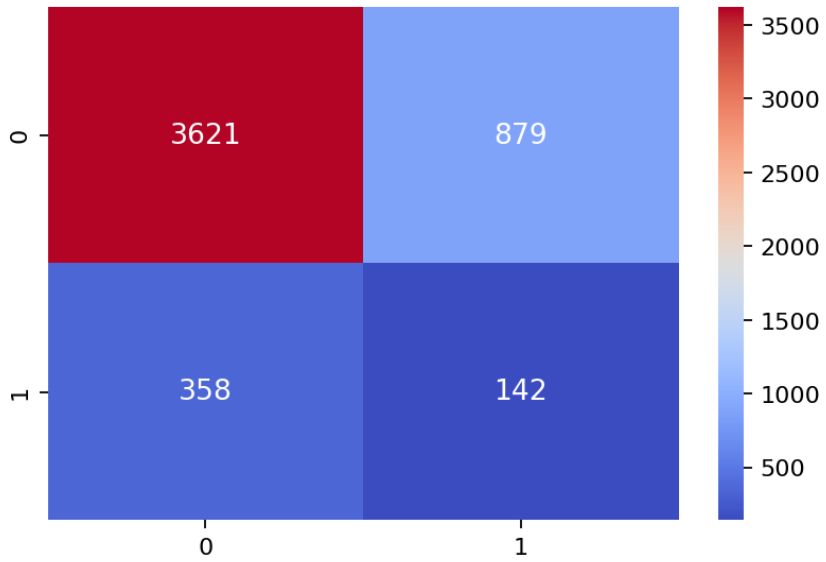


Figure 28: Confusion Matrix of xStream.

The area under ROC curve is 0.72 for iForest algorithm. The ROC curve is above the random classifier curve. The PR curve is above the no skill line. Therefore, the performance of iForest is best on this dataset. It does meet the criterion of AUROC > 0.6 as well.

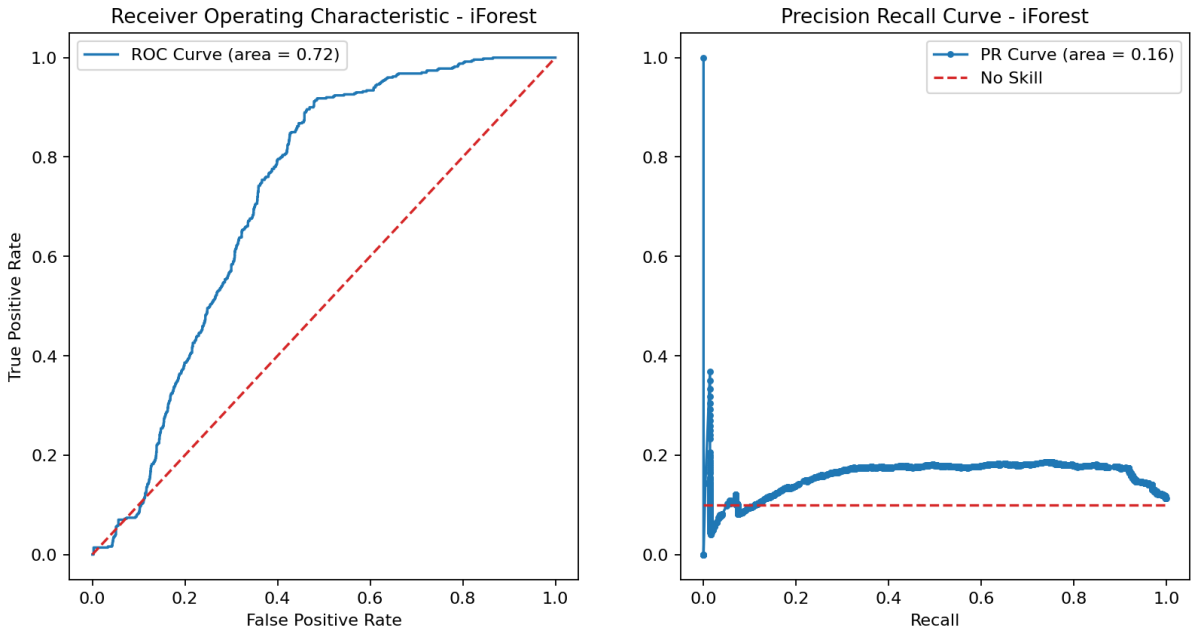


Figure 29: AUROC and PR curves for iForest.

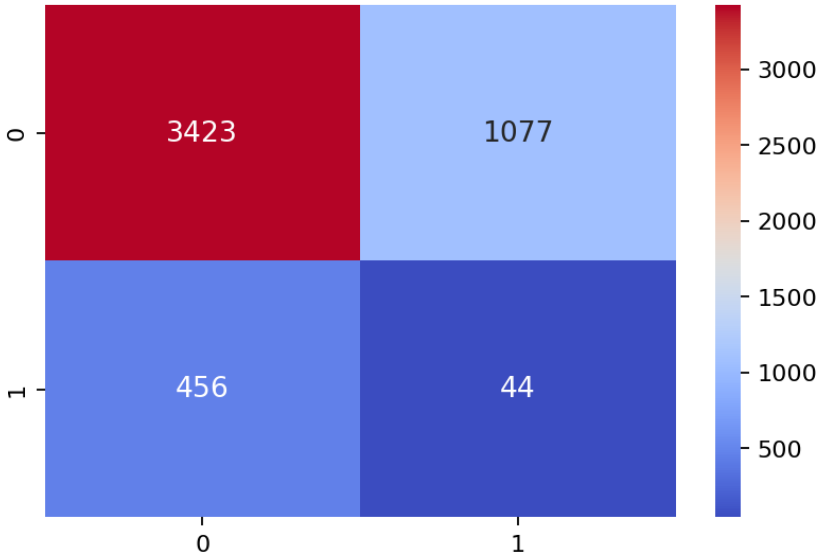


Figure 30: Confusion Matrix of iForest.

The area under ROC curve is 0.61 for the ensemble of xStream and iForest. The ROC curve is above the random classifier curve. The PR curve is above the no skill line. Also, it meets the criterion since the area under ROC curve is above 0.6. Therefore, the ensemble model can work properly on this dataset. However, it becomes slow when compared to the other datasets since it has more features.

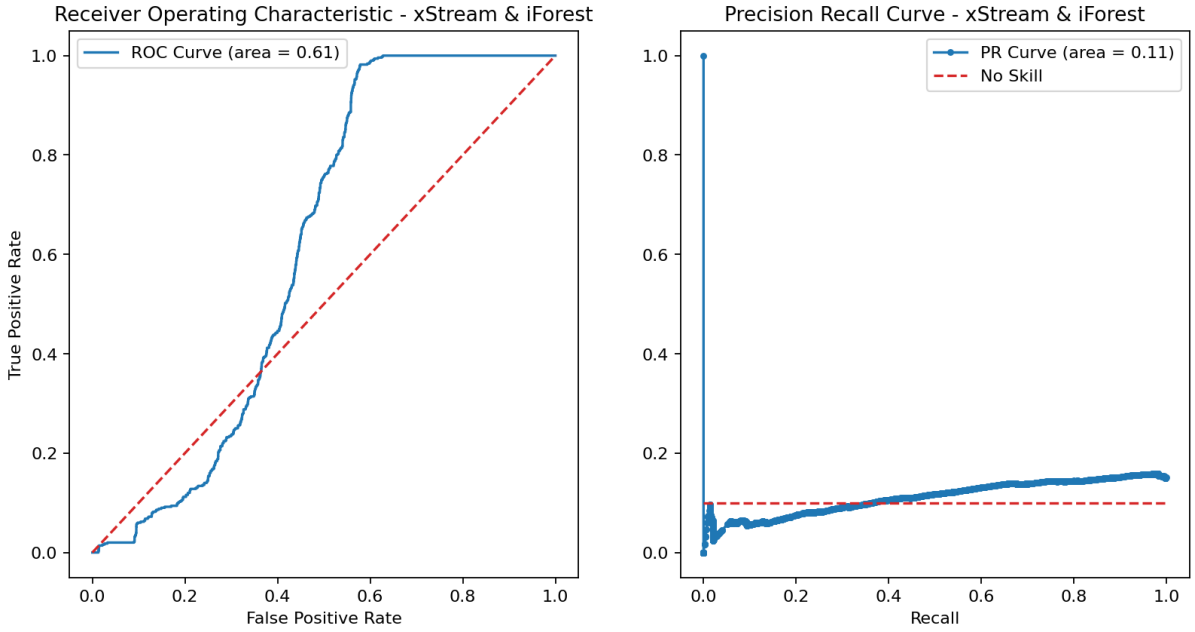


Figure 31: AUROC and PR curves for Ensemble Model.

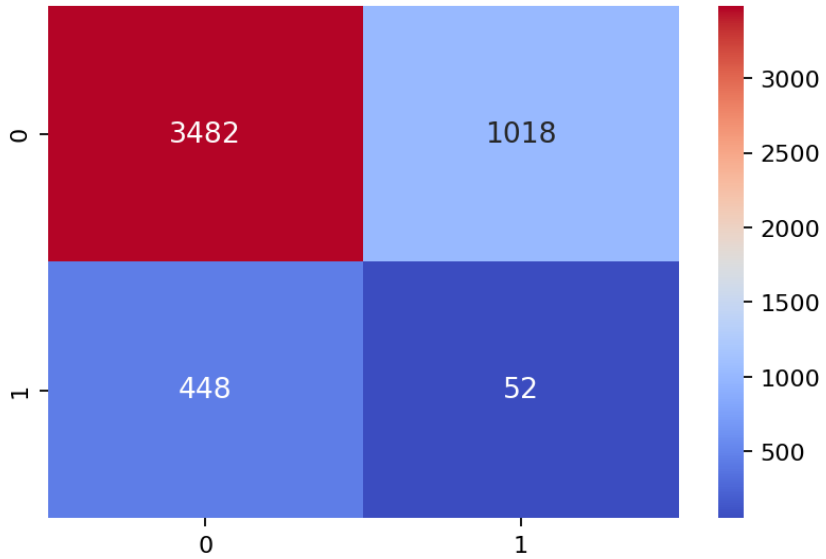


Figure 32: Confusion Matrix of Ensemble Model.

All of the F1 scores of the algorithms remained under the expected criterion (F1 Score > 0.4). This is because the datasets consist of multiple attack/anomaly types. However, we have labeled all the anomaly types as '1' and the normals as '0' and we expected from algorithms to make a binary classification. When we researched for the reason behind low F1 scores, we have come up with that F1 Scores of multiclass problems are usually low [40].

The results are summarized in Table 1. As it can be seen in Table 1, the criteria set is mostly satisfied with the Kitsune Dataset (LSTM Autoencoder). The best achieved results with this realistic dataset is highlighted in Table 1. Also, the performance of the algorithm is relatively better than the other datasets. As it was explained in Datasets part inside Methods & Progress section, Kitsune Dataset is more realistic for our project, hence it is also wise to consider the Kitsune Dataset results. Using LSTM Autoencoder increased the performance of the algorithms. Therefore, it was adapted to the part of the project that works with the live data as well.

Dataset	Model	AUROC	PR	F1
Kitsune Dataset	xStream	0.44	0.13	0.136
Kitsune Dataset	iForest	0.50	0.10	0.139
Kitsune Dataset	xStream & iForest	0.59	0.13	0.165
Kitsune Dataset (LSTM Autoencoder)	xStream	0.55	0.16	0.189
Kitsune Dataset (LSTM Autoencoder)	iForest	0.59	0.14	0.169
Kitsune Dataset (LSTM Autoencoder)	xStream & iForest	0.61	0.17	0.203
CIC-IDS 2017	xStream	0.40	0.08	0.119
CIC-IDS 2017	iForest	0.72	0.16	0.229
CIC-IDS 2017	xStream & iForest	0.61	0.11	0.146

Table 1: Final Results for AUROC, F1-Score, and PR.

The results for accuracy, precision, and recall are in Table 2. Although CIC-IDS 2017 gave best results for AUROC and PR with iForest algorithm, the accuracy, precision, and recall results are the worst among all the combinations of datasets and models. This also shows the challenge to evaluate the results in anomaly detection applications. One metric never tells the whole story behind the scenes, and various metrics should be checked at the same time. Kitsune Dataset (LSTM Autoencoder) gave the results that the accuracy is 0.7398, the precision is 0.1159, and the recall is 0.2425, which are quite good compared to all of the results.

Dataset	Model	Accuracy	Precision	Recall
Kitsune Dataset	xStream	0.7194	0.0760	0.1623
Kitsune Dataset	iForest	0.7520	0.1254	0.2485
Kitsune Dataset	xStream & iForest	0.7190	0.0813	0.1764
Kitsune Dataset (LSTM Autoencoder)	xStream	0.7292	0.0813	0.1663
Kitsune Dataset (LSTM Autoencoder)	iForest	0.7492	0.1160	0.2285
Kitsune Dataset (LSTM Autoencoder)	xStream & iForest	0.7398	0.1159	0.2425
CIC-IDS 2017	xStream	0.7526	0.1391	0.2840
CIC-IDS 2017	iForest	0.6934	0.0393	0.0880
CIC-IDS 2017	xStream & iForest	0.7068	0.0486	0.1040

Table 2: Final Results for Accuracy, Precision, and Recall.

Achieved Results: As the final result, the area under ROC curve (requirement 11) and PR curve (requirement 13) criteria are met with the ensemble model.

Unachieved Results: The F1 Score criterion (requirement 12) could not be satisfied as the reason was explained before.

5.1.4 Computational Speed Results

From Kafka	Transformer	Feature Extraction	Fitting	Prediction
0.05s	0.02s	0.07s	1.0 $\rightarrow$ 0.4s	0.1s

Table 3: Computational Speed Results for Processing One Packet.

Achieved Results: Data acquisition interval meets the functional requirement 10. Model fitting of each model speed is 0.4 s which is 0.5 s as stated in the functional requirement 6 for each data point. Anomaly detection time from transformer until prediction is 0.59 s as can be seen in Table 3, which is under 2 s and meets the functional requirement 7. Around 100 EPS observed in network traffic of one host, that meets the functional requirement 8.

5.1.5 User Interface

Achieved Results: The functional requirement 9 met since the chart's update time for each event detection is under 1 minute. As in functional requirement 14, the user can to determine a confidence percent threshold between 0-1 now. The user can download the results in CSV format, therefore functional requirement 16 is met. Thanks to the User Interface application Kibana, the UI is customizable; the functional requirement 17 is met.

Unachieved Results: The user cannot select hyperparameters of algorithms right now, therefore the functional requirement 15 is not achieved yet. This property can be added after the CM4 period and we are currently working on it.

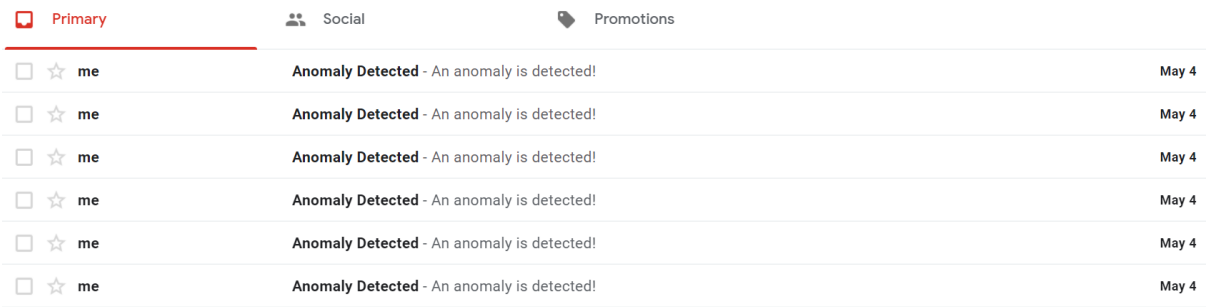


Figure 33: Mails coming from the system for the anomalies shown in Gmail Inbox.

When a prediction is above the given threshold, it is counted as an anomaly. If an anomaly is detected with any of the algorithms, the user is alerted via e-mail as shown in Figure 33.

The steps for changing threshold are shown in Figure 34. Initially, 'Dev Tools' bar should be selected from the menu. Then, the desired threshold value can be written instead of the existing one. Finally, when the code is run, the threshold will have been set to the desired value.

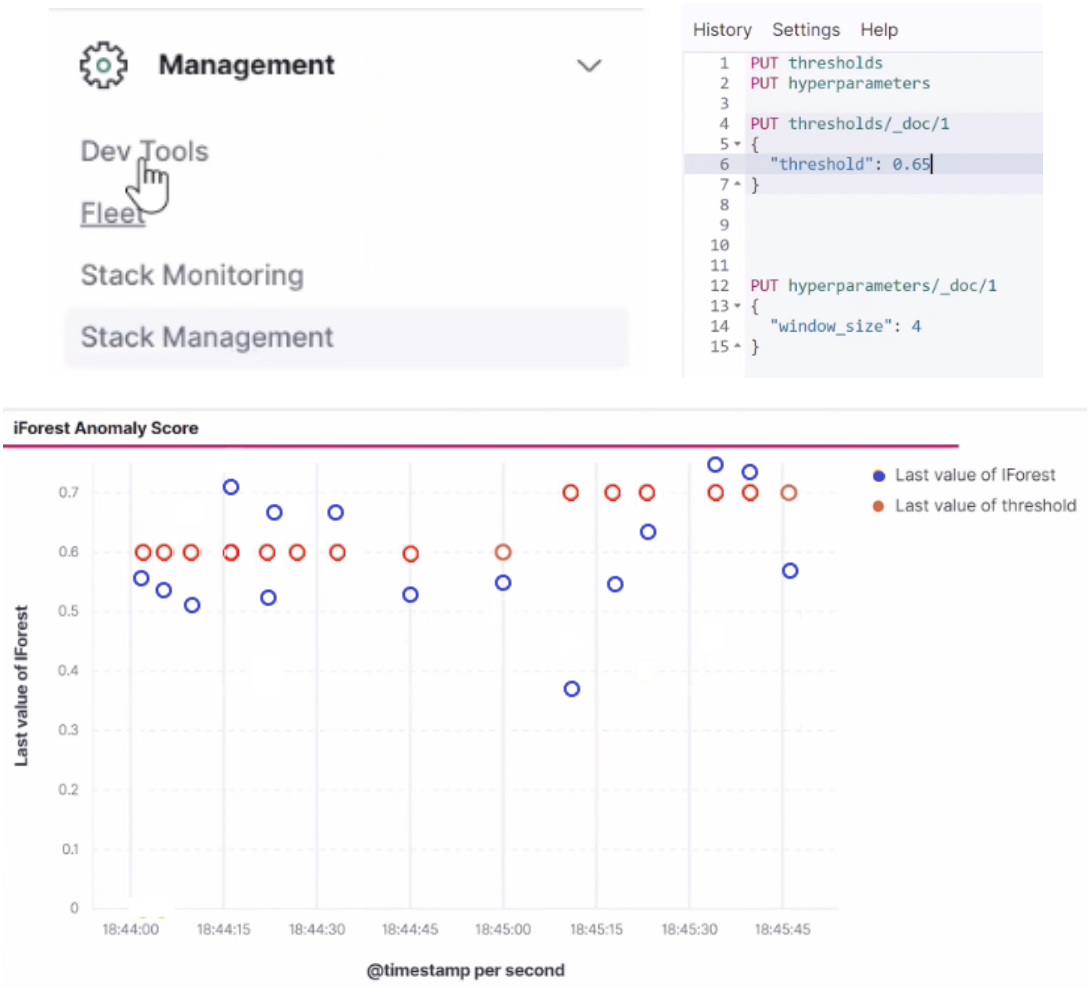


Figure 34: Steps for changing the threshold.

Figure 35: UI that contains score and threshold values for iForest, Loda, RRCF, and Ensemble (xStream and Loda) algorithms.

5.2 Discussions and Lessons Learned

- **Achievements and Positive Parts**

  For the finalized version of our project, we can claim that the essential part of the goals was attained. We were able to present a flowing system that can detect anomalies to a considerable degree while also meeting the requirements of the project. We think that the general system flow and the user interface of the product are a strong part of the project since we achieved the requirements and were content with the results.

  Our thoughtfulness and decency that we have shown each other was one of the most successful parts of the project. We were able to work and cooperate under stressful conditions without impairing our group or damaging the project.

  In addition, one thing we anticipated but did not happen was our lack of expenses for the project itself. We anticipated a larger processing power for training but we did not utilize or see the necessity for it. We see the minimal expenses as a success of the product.

- **Challenges and Hurdles**

  The most challenging part of our project was to understand the network data and anomalies in a network context. The network data analysis requires expertise and we were not familiar with the subject matter. In addition, the complexity of the network data also created hurdles for the anomaly detection part. This skill was missing from the project. Therefore, we can say this part did not go well and resulted in extended deadlines in our project plan given on the Gantt chart.

  The pandemic also affected our project. For a long time, we were not able to work at the company, which hindered our project and prevented effective communication with the company. This could be better handled if we were all able to visit the company in a normal situation. This situation was beyond our control and highly affected the trajectory of our project.

  We were surprised by the deficiencies of some tools that we planned to use: such as some libraries and software tools. We worked intensively to fix and sometimes reach the owners of relevant codes and tools and utilize them. In addition, the data set for the network analysis were mostly outdated and was not labeled or documented clearly. We tried to prepare our own data set, which was time-consuming.

- **Lessons Learned**

  Throughout the project, we were able to observe how a large-scale software project is developed along with the working conditions in a company. This particularly helped us in our decisions for future career plans and further education. Also, the help we got from our

company advisor was not limited to the project, he also offered invaluable advice as a new engineer with a similar background.

For our recommendations for a project of a similar type, we can highly advise learning domain knowledge for anomaly detection. Anomaly detection can be performed on a variety of systems, each having its unique intricacies and complexities. This knowledge is essential in understanding, interpreting, and observing any kind of anomaly, which is essential for detection. For this project, we focused on anomaly detection for the network data. Our lack of network data and network structure prevented us from understanding the anomalous behavior from the network context. So, our most important advice is to understand the system characteristics before trying to detect and analyze the anomaly, instead of being only data-driven.

5.3 Future Directions

Although we have reached the goals that we set for ourselves throughout the project, there still are many improvements and additions that can be done to further increase the usability and efficiency of the project. Some general things such as optimization of the code can be done to further speed up the algorithms and the general flow. Additionally, other detection models can be added to further increase the variety of models. Docker can be used to package the project into a single executable, which can help ease the distribution of the project. Scheduled training can be implemented. This way the algorithms will be trained on incoming data only at certain times of the day (times where there is less possibility of an attack happening), and at other times the program will only predict. This will speed up the general flow since training takes more time. Finally, multi-source input can be utilized to detect group anomalies. Group anomalies are anomalies that are detected using patterns from multiple sources.

The final product can be used in many different industries. As technology progresses, many companies are starting to utilize the internet. Since our product works with network data, it can be utilized to find network anomalies in servers and other hardware devices that are connected to the network. Hence, it can be utilized in many places such as banks, schools, and telecommunication companies.

6 Equipment List

We will be using PyCharm IDE for Python, both of them are obtained from the internet and free to use. WireShark/TShark will be used for data acquiring, both of them are obtained

from the internet and free to use. Python library PyShark will be used to capture live packets directly from WireShark. It is free to use and obtained from the internet. PySAD is an anomaly detection model and metrics which is obtained from the internet and free to use. Scikit-learn is a machine learning library for python which is used for predictive data analysis. Elasticsearch & Kibana is an easy UI integration due to the elastic python wrapper [41].

NSL-KDD and Kitsune Network Attack Dataset are two datasets as raw network data and feature format. Kitsune is a library in which feature extraction and anomaly detection can be done. CIC-IDS 2017 dataset is also in feature extracted format.

Tools	Obtained From	Cost
PyCharm	jetbrains.com [42]	Free to use
PySAD	github.com [4]	Free to use
Scikit-learn	scikit-learn.org[43]	Free to use
WireShark/TShark	wireshark.org [2]	Free to use
PyShark	github.com [3]	Free to use
Elastic & Kibana	elastic.com [41]	Free to use
Confluent Kafka	confluent.cloud [28]	Free to use*
Creately	creately.com [44]	$7 a month
NSL-KDD	unb.ca [31]	Free to use
Kitsune Network Attack Dataset	kaggle.com [6]	Free to use
CIC-IDS 2017	unb.ca [31]	Free to use
Kitsune	github.com [45]	Free to use
Tensorflow	tensorflow.org [46]	Free to use

Table 4: Equipment list.

*Free to use for up to $200 of consumption. After the free period, you are charged based on your consumption rate. More info on [28]

References

[1] *Databoss*, https://www.data-boss.com.tr/about-databoss/, (Accessed on 14/05/2021).

[2] *Wireshark*. [Online]. Available: https://www.wireshark.org/download.html?fbclid=IwAR20qjegznVqILbMsO4cYXFH5bRSS4QhOwL84GRO5URqGq5mqLORjjK7pEs.

[3] KimiNewt, *Kiminewt/pyshark*. [Online]. Available: https://github.com/KimiNewt/pyshark.

[4] Selimfirat, *Selimfirat/pysad*. [Online]. Available: https://github.com/selimfirat/pysad.

[5] *Nsl-kdd datasets*, https://www.unb.ca/cic/datasets/nsl.html, (Accessed on 05/14/2021).

[6] YisroelMirsky, *Kitsune network attack dataset*, Aug. 2020. [Online]. Available: https://www.kaggle.com/ymirsky/network-attack-dataset-kitsune.

[7] *Intrusion detection evaluation dataset (cic-ids2017)*, https://www.unb.ca/cic/datasets/ids-2017.html.

[8] R. Abdulhammed, H. Musafer, A. Alessa, M. Faezipour, and A. Abuzneid, "Features dimensionality reduction approaches for machine learning based network intrusion detection," *Electronics*, vol. 8, p. 322, Mar. 2019. DOI: 10.3390/electronics8030322.

[9] J. Veeramreddy, V. Prasad, and K. Prasad, "A review of anomaly based intrusion detection systems," *International Journal of Computer Applications*, vol. 28, pp. 26–35, Aug. 2011. DOI: 10.5120/3399-4730.

[10] K. Limthong and T. Tawsook, "Network traffic anomaly detection using machine learning approaches," Apr. 2012. DOI: 10.1109/NOMS.2012.6211951.

[11] *Ibm qradar siem - genel bakış*. [Online]. Available: https://www.ibm.com/tr-tr/products/qradar-siem.

[12] [Online]. Available: https://qradar-demo.mybluemix.net/?%5C_ga=2.227851933.1852877022.1605521871-1300051361.1603224723.

[13] [Online]. Available: https://moa.cms.waikato.ac.nz/.

[14] *Outlier detection*, Aug. 2013. [Online]. Available: https://moa.cms.waikato.ac.nz/details/outlier-detection/.

[15] "Ieee guide for developing system requirements specifications," *IEEE Std 1233, 1998 Edition*, pp. 1–36, 1998. DOI: 10.1109/IEEESTD.1998.88826.

[16] *Methods_solution*. [Online]. Available: https://notebook.community/nicococo/tilitools/lectures/methods%5C_solution.

[17] T. Ergen and S. S. Kozat, "Unsupervised anomaly detection with lstm neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 8, pp. 3127–3141, 2020. DOI: 10.1109/TNNLS.2019.2935975.

[18] N. Laptev, S. Amizadeh, and I. Flint, "Generic and scalable framework for automated time-series anomaly detection," New York, NY, USA: Association for Computing Machinery, 2015, ISBN: 9781450336642. DOI: 10.1145/2783258.2788611. [Online]. Available: https://doi.org/10.1145/2783258.2788611.

[19] D. Steen, *Precision-recall curves*, Sep. 2020. [Online]. Available: https://medium.com/@douglaspsteen/precision-recall-curves-d32e5b290248.

[20] *Electricity usage of a laptop, notebook or netbook*. [Online]. Available: http://energyusecalculator.com/electricity%5C_laptop.htm.

[21] *Global user and entity behavior analytics market (ueba) market worth usd 908.3 million by 2021 - rapid growth in mobile apps in apac - research and markets*, Aug. 2016. [Online]. Available: https://www.businesswire.com/news/home/20160823006004/en/Global-User-and-Entity-Behavior-Analytics-Market-UEBA-Market-Worth-USD-908.3-Million-by-2021---Rapid-Growth-in-Mobile-Apps-in-APAC---Research-and-Markets.

[22] I. I. B. Ltd., *Siber güvenliğin milli güvenlik açısından önemi ve alınabilecek tedbirler*, Jan. 1970. [Online]. Available: https://www.ceeol.com/search/article-detail?id=222649.

[23] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6. DOI: 10.1109/CISDA.2009.5356528.

[24] *5.7.exporting data*. [Online]. Available: https://www.wireshark.org/docs/wsug%5C_html%5C_chunked/ChIOExportSection.html.

[25] [Online]. Available: https://www.wireshark.org/docs/man-pages/tshark.html.

[26] KimiNewt, *Kiminewt/pyshark*. [Online]. Available: https://github.com/KimiNewt/pyshark/.

[27] Iglesias, F., Zseby, T, "Analysis of network traffic features for anomaly detection," *Mach Learn*, vol. 101, pp. 59–84, 2015. DOI: https://doi.org/10.1007/s10994-014-5473-9.

[28] apache kafka, *Confluent kafka*. [Online]. Available: https://www.confluent.cloud/environments/env-g1051/clusters.

[29] H. E. Kiziloz, "Classifier ensemble methods in feature selection," *Neurocomputing*, vol. 419, pp. 97–107, 2021, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.07.113. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231220313151.

[30] G. Azevedo, *Feature selection techniques for classification and python tips for their application*, Aug. 2019. [Online]. Available: https://towardsdatascience.com/feature-selection-techniques-for-classification-and-python-tips-for-their-application-10c0ddd7918b.

[31] *Search unb*. [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html.

[32] A. H. Lashkari, G. D. Gil, M. S. I. Mamun, and A. A. Ghorbani, "Proceedings of the 3rd international conference on information systems security and privacy," *Characterization of Tor Traffic using Time based Features*, 2017.

[33] S. Cosan, "Payload-based network intrusion detection using lstm autoencoders," M.S. thesis, Department of Electrical and Electronics Engineering, Bilkent University, Ankara, Turkey, 2020.

[34] *Lstm autoencoder architectures*, https://www.researchgate.net/figure/LSTM-Autoencoder-A-and-B-are-architectures-of-the-encoder-and-decoder-respectively_fig2_340474035, (Accessed on 05/19/2021).

[35] *Outlier detection datasets (odds)*. [Online]. Available: http://odds.cs.stonybrook.edu/.

[36] Z. Ding and M. Fei, "An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window," *IFAC Proceedings Volumes*, vol. 46, no. 20, pp. 12–17, 2013, 3rd IFAC Conference on Intelligent Control and Automation Science ICONS 2013, ISSN: 1474-6670. DOI: https://doi.org/10.3182/20130902-3-CN-3020.00044. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667016314999.

[37] *Unsupervised anomaly detection with isolation forest - elena sharova - youtube*, https://www.youtube.com/watch?v=5p8B2Ikcw-k, (Accessed on 05/18/2021).

[38] E. Manzoor, H. J. Heinz, and H. Lamba, "Xstream : Outlier dete ' x ' ion in feature-evolving data stream s," 2018.

[39] Y. Imamverdiyev and L. Sukhostat, "Anomaly detection in network traffic using extreme learning machine," Oct. 2016, pp. 1–4. DOI: 10.1109/ICAICT.2016.7991732.

[40] *F-1 score for multi-class classification | baeldung on computer science*, https://www.baeldung.com/cs/multi-class-f1-score, (Accessed on 05/19/2021).

[41] elastic, *Elastic*. [Online]. Available: https://www.elastic.co/.

[42] *Pycharm: The python ide for professional developers by jetbrains*. [Online]. Available: https://www.jetbrains.com/pycharm/.

[43] *Installing scikit-learn¶*. [Online]. Available: https://scikit-learn.org/stable/install.html.

[44] creately, *Creately*. [Online]. Available: https://creately.com/.

[45] Ymirsky, *Ymirsky/kitsune-py*. [Online]. Available: https://github.com/ymirsky/Kitsune-py.

[46] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.