DATA3001 Report Proposal

Detecting Anomalies in IoT Device Communications

by CyberNet:
Seoyeon Park (z5248219)
Dong Bin Min (z5297724)
Harish Vinoth Anna (z5309188)

December 2022

Contents

Introduction

Team Name: CyberNet

Motif from SkyNet an AI system for defence in Terminator, Cybernet's mission is to utilise current Intrusion Defence System (IDS) to develop analytic techniques to defend IoT (Internet of Things) and IIoT (Industrial Internet of Things) devices from cyber attacks.

Executive Summary

The following report effectively outlines methods to find out the device_mac and time at which IoT devices could have been under a cyber attack by looking at its outgoing byte count. For this project, 14 days of data containing byte and packet size for all outgoing and incoming ports were collected. Only outgoing ports were selected for analysis with the reasons being explained in the Data Preprocessing section. The report outlines different methods of analysis such as K-Nearest Neighbours, Time Series Forecast, Isolation Forest, One-class SVM, LSTM autoencoder and K-means. Amongst these models KNN and K-means performed the best with an average accuracy of above 99%.

1 Background

1.1 Problem Summary

The Internet of Things (IoT) is expanding into every facet of life. Everyday devices such as fire alarms, fridges, TVs and digital cameras now all have the ability to connect to the internet and collect and share data. There are already 14.4 billion IoT devices operational today, a figure estimated to grow to approximately 27 billion by 2025 [**1**], and IoT devices now make up more than half of all total devices connected to the internet [**2**]. This presents an increasingly challenging security problem as many of these devices have little in the way of cybersecurity, and are highly vulnerable to data breaches or being hijacked to carry out a large-scale attack on the network.

1.2 Project Aim

The project aims to deliver effective anomaly detection to clients. The success of the report was assessed by looking at these aspects.

1. Reliability: Code that works with any given port.
2. Linear Time: Finding anomalies as time progresses.
3. Time Complexity: Having time complexity of under 2 minutes to prevent further intrusion.
4. Memory: Using minimal memory.
5. Consultation: Effective solution from the findings.

1.3 Related Research

Several studies regarding the challenges and anomalies within IoT-based systems were regulated through various relevant research topics within distributed environment fields which will be mentioned throughout this section.

Intrusion techniques theorised on anomaly data utilises a basic data flow which uses normal data (from trust users) to compare it against user data to help detect anomalies within IoT's [3]. In this scenario malicious tools and applications create noise and other harmful situations causing various deviations from normal system behaviours. Unusual footprints within a computing environment are generally caused by hackers or intruders who produce abnormal system behaviour[4]. The main detection system is created through models that examine normal activity and behaviour within IoTs. Thus, the deviation from normal behaviour within IoTs can be detected through analysing the created model which updates often with data from its trust users [5].

Numerous Swarm Intelligence isolation tree algorithms have been contemplated to elucidate problems such as abnormality detection in a dispersed fashion [6]. As seen in [7], the clustering influenced advancement of the Ant colony optimisation (ACO) model was utilised.Furthermore [7] expanded on the pre-screening procedure of data in order to present outlier detection. Whilst the unselected data attempts to identify the outliers; the approach starts from a point selection matrix where the final results are obtained through another algorithm. One of the ACO approaches to distinguish intrusions was suggested in [8], in contrast to earlier studies where it was assisted in intrusion response.

Honeypots and ACO is portrayed in [9] where the ant colony is exploited to imprint the trail of attack and investigates the interests and the environment of aggressors. 'Hierarchical Particle Swarm Optimization Based Clustering', is a swarm-based clustering approach which is in [10], designed to distinguish between outliers.

This method achieves hierarchical agglomerative clustering to achieve this where the swarm of particles expands, developing many phases to observe outliers and clusters. Within the optimization task, [11] can be seen reorganising this clustering task. This advancement strengthens the scope of PSO assisting new knowledge into anomaly outlier.

underlines the possible flaws with respect to the security of IoT's and also portrays the various other flaws within this system and how it can be harmful for businesses and the general public. It explores the significant aspects that influence security mechanisms achievement, in regards to energy consumption, invalid positive rate, performance processing.

Whereas, [13] provides the history on the security challenges which can be experienced when applying anomaly detection to IoT data. In order to evaluate some parameters within the travel of a minute, the use of a simple machine [14] is used to represent the normal behaviour. The models are exploited to determine definitive failures if the data received does not apply to normalised patterns. This is where results are obtained for specific fault modalities; to use to define which data are compatible.

Through a histogram based inquiry, the relations amid the variables are achieved. In regards to the tailored access, the description of the anomalies observed can be efficiently established and the computational intricacy is shortened for each method.

With the aim to accomplish a distributed detection, [15] proposes a hybrid intrusion detection system for IoT established on MapReduce approach. A supervised and non-supervised path forest is made to find the error where the expected method empowers decision about apprehensive behaviours exploiting a voting mechanism.

Kumar et al. proposed a distinct method, the subspace ensemble method which utilised an average anomaly score made from several sub spaces to further improve the accuracy of its prediction [16]. Through this method [16], isolation forest was presented as a mean for anomaly detection within Iots, which is then built through data which has been sampled and produces a unique anomaly score through a depth of leaf. This method then suffered a lack of ability to find more complicated anomalies, for example an "axis-parallel anomaly", which led to a discovery by Zhang and Xiang et al. These researchers produced a detected method called LSHiForest to contest these more unique and complex anomalies, where after this Xiang et al. produced a learning-to-hash isolation forest method to decode these complex anomalies and to learn more data leaked by these anomalies [17]. These new found methods were more unique, efficient and accurate compared the previous isolation forest methods.

Whilst comparable to the research topics above, our proposed method is much more cost-efficient and adapts to a fast paced environment, which makes it a more dynamic approach towards flawed systems such as IoTs. Within our system, reliable and accurate results are always available.

This approach overcomes two-phase schemes (usually occurring in isolation forest methods) with a single online isolation forest phase. Furthermore, this approach guarantees serviceability, continuity and limited user actions but on the other hand a central management mechanism aids in resulting disadvantages among a central point of failure. Thus, our approach is efficient, low in cost, efficient and is distributed with a high level of resilience.

2 Scope

This project aimed to accurately analyse and flag potential cyberattacks by tracking abnormal data flows through different network protocols such as the Domain Name System (DNS) and Hypertext Transfer Protocols (HTTP). It produced a model that flags and reports abnormal data flows and a 30 page report on the construction process of the model, the analysis results and recommendations to the client on potential actions they could take to improve IoT device security in the future. The project will be presenting our model to the client at the conclusion of this project.
The model was produced using the data provided by the client, of which each row consisted of:

- Time of record
- 4x incoming port byte count
- 4x incoming port packet count
- 5x outgoing port byte count
- 5x outgoing port packet count
- 2x local port byte count
- 2x local port packet count
- Mac address

In total, the data consisted of 14,476,698 rows x 24 columns.

There were a number of assumptions that were required to make about the data after conversing with the client. The data provided to us is in a benign condition, and most spikes or dips in byte or packet count come within the range of normal operations for that port. Any significant gaps in data were assumed to be due to a power outage or similarly benign occurrence and subsequently removed from the data. Smaller gaps in data were extrapolated from the data points on either side. The project assumed that all components involved in the collection and measuring of this data stayed the same throughout the measurement period, and that for our model to be effective these components would remain the same going forward.

The success of this project was measured in three parts. First, the construction of a model capable of identifying potentially abnormal network communications from IoT devices based on the device name, flow, time and day of the communication compared to other communications made with similar parameters. Second, the ability to identify and flag any abnormal communications between devices within a time frame that allows the client to react swiftly. Third, the capacity to classify all identified potential anomalies based on the risk they pose to the client.

3 Project Plan

3.1 Formulating the Problem

As discussed in the background above, Cybernet carried out extensive research into the problem and work completed on similar problems in the past. Based on this research, our data pool of purely benign data and our project objectives, the team was able to select different methods to approach our project.

1. Classification Method

   (a) Port Classification
   Port Classification method utilize strip of single port data with a mixture of devices. It uses data from all the port as training set and classify the test set. If there is a device classified differently to its majority port, flag it as an anomaly. KNN was used for this method.

   (b) Device Classification
   Device Classification uses data from a device and compare with other devices of the same time. The project did not choose this method as there was too many devices and a specific device has some unique behaviour that could be seen as an anomaly compared to other devices.

2. Forecasting Method
   Forecasting model is used in this project only after knowing the port of the test data. After using port classification it utilise past data to predict the future data. ARMA, ARIMA and SARIMA models were used in this method.

3.2 Exploratory Data Analysis

3.2.1 Column Analysis

The given data set was made up of 24 columns, most of which represented the data flow across different ports between the network and the collection of IoT devices. Data was provided for four incoming ports (network to device), five outgoing ports (device to network) and two local ports (device to device). For each of these ports, both the byte and packet size were provided as separate columns, making up 22 columns in total. The incoming ports were made of three transmission control protocols (TCP) and a single user datagram protocol (UDP). The outgoing ports were made of three TCP's, one UDP and a single resource reservation protocol. Lastly, local ports were made of two UDP ports.

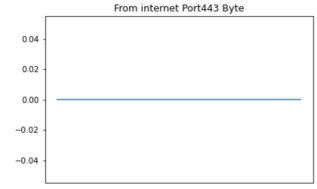
The data was recorded over a time period of 14 days with a new entry recorded every minute. This started from the 2nd of September 2022 at 00:25 to the 16th of September 2022 at 00:24, for a total of 20,160 unique timestamps. This time series data was provided in the data set as the column 'time', which was initially measured in nanoseconds and then converted into a more readable date-time format. Then, the data set was sorted by the following column to make sure it was in chronological order.

The name of each device measured for this data set was located in the column 'device_mac'. There were a total of 753 unique devices included in the data, and all devices had their communications recorded each minute for the whole 14-day period.

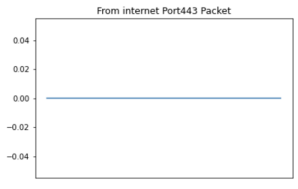
3.2.2 Data Visualisation

A simple visualisation of the activity of every data column in the data set in figures 1-3 showed that there are significant fluctuations in activity over the 14 day time period, as well as between different ports. An interesting

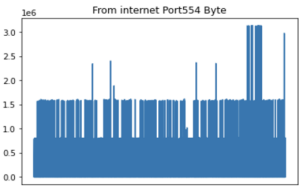
insight from these visualisations was that the byte and packet count for each port appeared to follow almost the exact same trend of activity. This is corroborated by calculating the correlation values between the byte size and packet size columns for each port, which was greater than 95% in all cases. Because of this, for all our models the project decided to use only the byte count for each port, essentially halving the number of features used in all models.



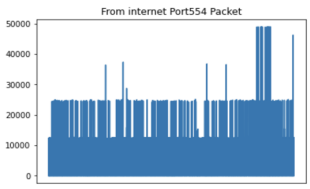
(a) Port443 Bytes



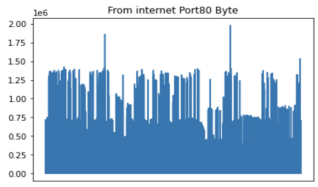
(b) Port443 Packets



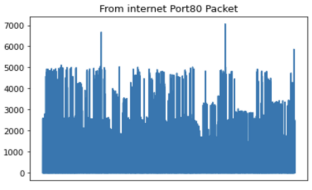
(c) Port554 Bytes



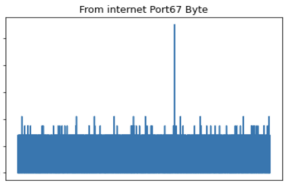
(d) Port554 Packets



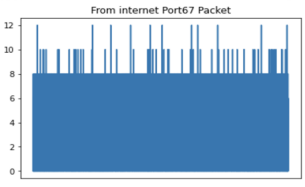
(e) Port80 Bytes



(f) Port80 Packets

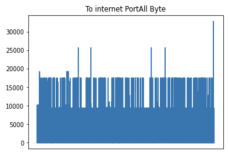


(g) Port67 Bytes

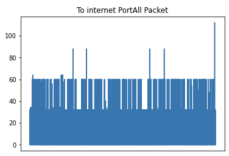


(h) Port67 Packets

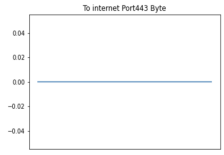
Figure 1: Network to Device Activity Map



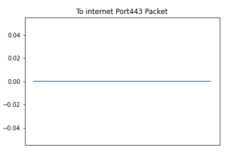
(a) PortAll Bytes



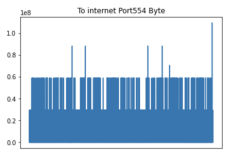
(b) PortAll Packets



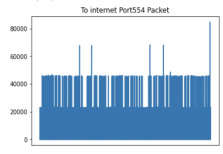
(c) Port443 Bytes



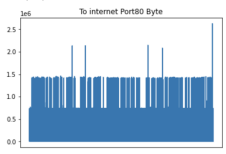
(d) Port443 Packets



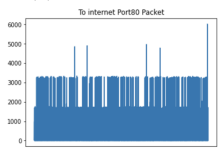
(e) Port554 Bytes



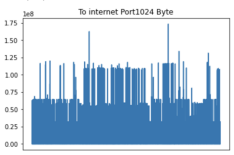
(f) Port554 Packets



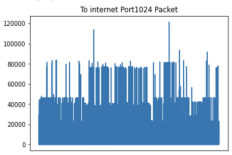
(g) Port80 Bytes



(h) Port80 Packets

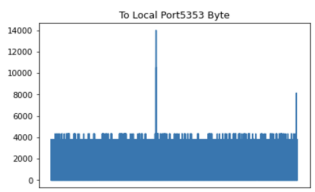


(i) Port1024 Bytes

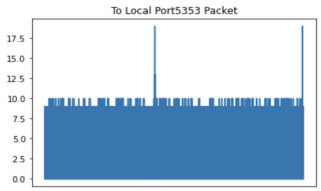


(j) Port67 Packets

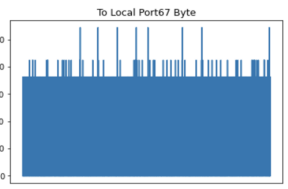
Figure 2: Device to Network Activity Map



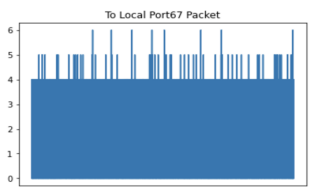
(a) Port5353 Bytes



(b) Port5353 Packets



(c) Port67 Bytes



(d) Port67 Packets

Figure 3: Device to Device Activity Map

3.2.3 Data Preprocessing

The original dataset contained too many rows and columns to consider. However, the project did not shorten the rows as it contains vital information such as time and device name. Therefore, it decided to reduce some columns that aren't so useful. The project observed average byte count and active transmission (Byte count is not zero) for UDP, TCP and RSVP for both incoming and outgoing port.

	TCP incoming	TCP outgoing
Average Byte count	47817.22	1712082.74
Active Transmission	12660492	12581479

Table 1: TCP comparison

	UDP incoming	UDP outgoing
Average Byte count	307.12	7646957.94
Active Transmission	11259261	8964665

Table 2: UDP comparison

For TCP there is 35% increase in byte count for outgoing port and similar active transmission. Meaning they TCP are actively communicating and more prone to attacks like remove access protocol. For UDP there is 25,000% increase in byte count and 0.79% increase in active transmission for outgoing port. This shows UDP just sends signal and doesn't receive any because they are connectionless protocol. In conclusion, TCP and outgoing port contains most important data that could be linked to cyber attacks.

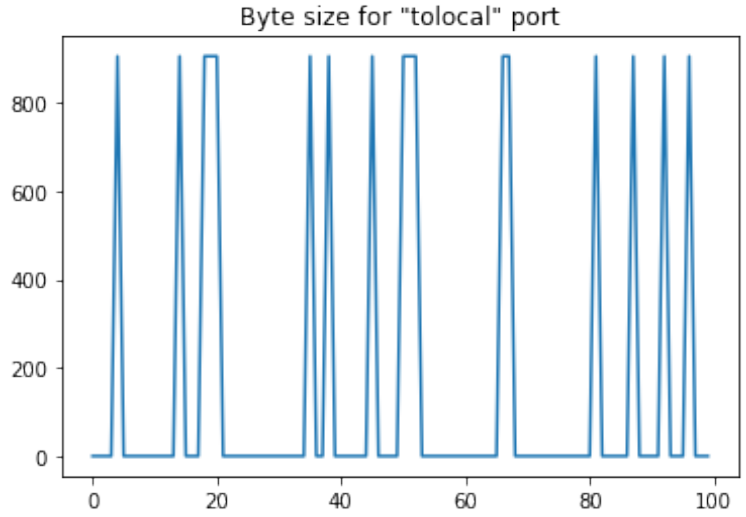


Figure 4: Local byte transmission =

Average Packet count	0.445
Average Byte count	201

Table 3: 'Tolocal' characteristic comparison

In figure red4 to_local port sends out fixed amount of byte size in a regular pattern and contains significantly low packet and byte compared to other ports. Combining this with the information that it is only UDP, a hypothesis can be made that it is sending out heartbeat to communicate it is working correctly.

Therefore in the project decided to use 'tointernet' port with TCP, UDP and RSVP to see anything interesting.

3.3 K nearest neighbours

The project started with a basic KNN classification process. Given a test data containing byte and packet size the project aimed to correctly classify each data to the correct port. If they were classified differently flag it as an anomaly. The project calculate the correct classification by looking at the majority ports in the test set.

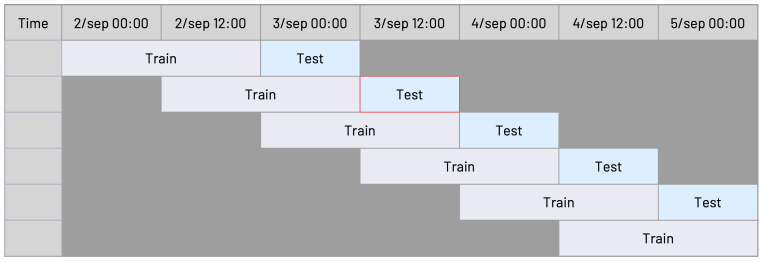


Figure 5: train test split representation for knn

The figure red5 shows method used for knn, the data is splited into 24 hours training data to classify the next 6 hours. This method only store the past 24 hours to maximise efficiency and memory storage.

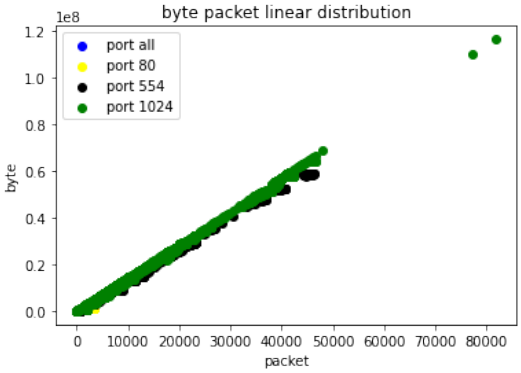


Figure 6: Byte and Packet linear relationship

Run time	Accuracy
< 30 minutes	$> 20\%$

Figure 7: Linear Relationship description

Initial process classified using the raw data. The figure red6 shows linear and clustered relationship between the packet and byte. This made the accuracy under 20% and run time for over 30 minutes. To overcome the cluster and linearity, division method is utilised where byte per packet was calculated $\frac{byte}{packet}$.

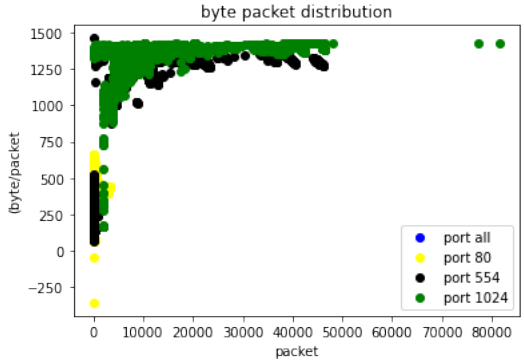


Figure 8: Byte over packet, packet relationship

Run time	Accuracy
20 seconds	98.68%

Figure 9: Byte per Packet Relationship description

all	1
443	0
554	5311
80	45
1024	25

Table 4: division port 554 classification results

Our hypothesis worked and the division relationship performed better. Increased the accuracy by plus 70% and reduce the runtime by 900%. However, as seen from the figure red8 port 1024 and port 554 still showed some cluster. Byte divided by packet squared ($\frac{byte}{packet}^2$) is used to further seperate the points.

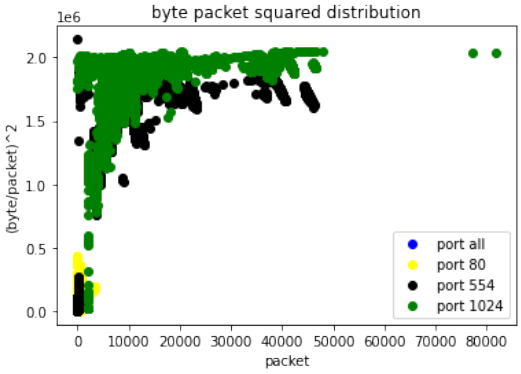
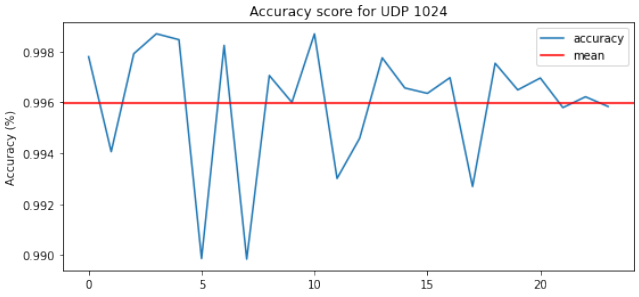


Figure 10: Byte over Packet squared relationship

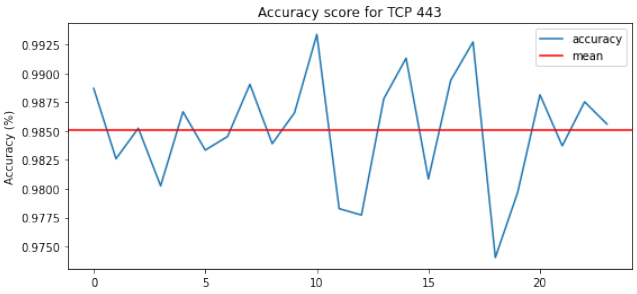
Run time	Accuracy
2 seconds	98.16%

Figure 11: Byte per Packet squared Relationship description

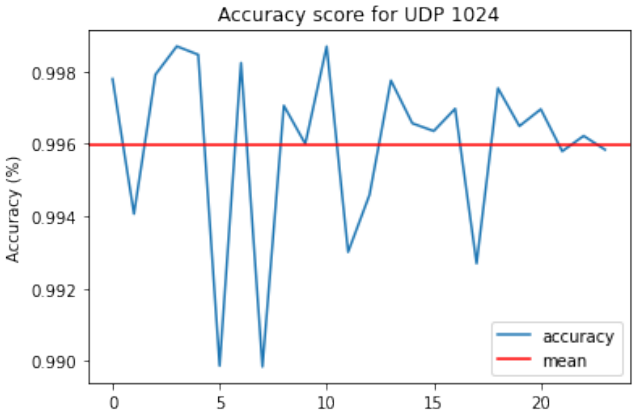
The figure red11 shows the successfully decreased the run time by 1,000% and have similar accuracy. For this reason Cybernet proceeded to use the divide and squared method for the project. The figure red12 shows accuracy scores for different ports over the entire data maintained above 99%.



(a) Flower one.



(b) Flower two.



(c) Flower two.

Figure 12: Accuracy for entire data iteration.

3.3.1 Insight of KNN

During our method of dividing and squaring the project used $\frac{byte}{packet}$ and had to remove all rows with 0 packet. This could lead to loss of vital information. However, as seen from the figure red13 whenever there is a 0 transmission the next minute doubles in byte size. This shows that 0 packet size is just caused from a network lag and by removing it no vital informatio is lost.

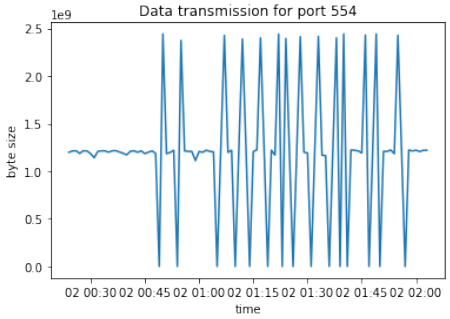
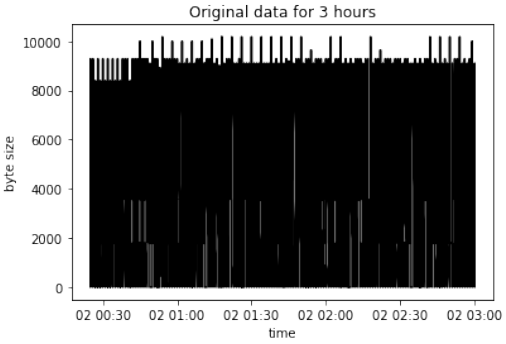


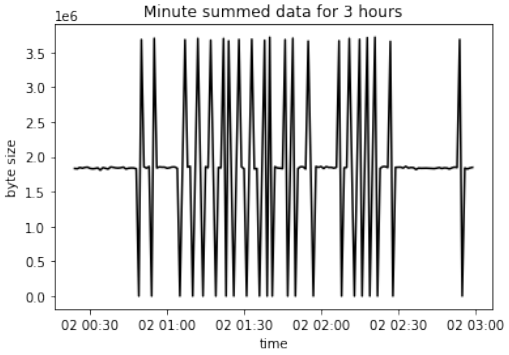
Figure 13: Data transmission describing for 0 transmission

3.4 Time Series forecasting

Our next step after classification of the port was to forecast the port. This method includes training the model to predict the value and compare it's euclidean distance. If the distance of our prediction and the actual *yvalue* is large, it would be flagged. The model aimed to use the same train and test from knn model. However, because of the running time was going over 3 hours the data was simplified to 3 hours training set and 1 hour test set. The failure of the forecasting model can be visualised this way for the report.//



(a) Byte plot for original data



(b) Byte plot for summed data

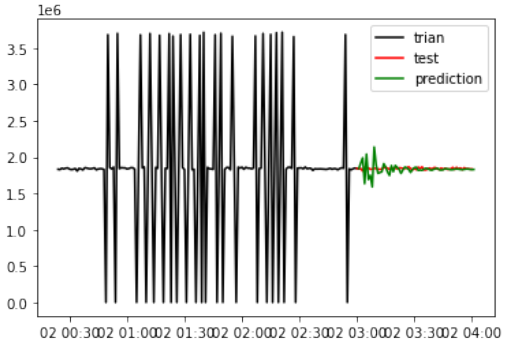
Figure 14: Difference between summed and original data byte plot

The forecasting model uses summed up the rows by minute such that it shows clearer patterns for the models to perform better. This can be visually seen from the comparison between red14a and red14b. Better accuracy of forecasting can be achieve through this method but lose which device is being attacked.

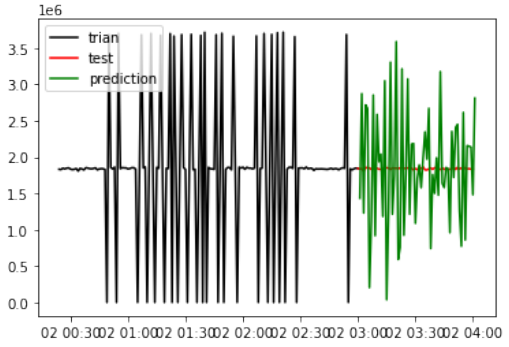
3.4.1 AutoRegressive Model

An autoRegressive model is a simple linear regression model that iterates it's past values to predict the future value. The AR model can be represented as:

$$AR(p) = \beta\_0 + \beta\_1 * y_{t-1} + ... + \beta\_p * y_{t-p}$$



(a) AR model with lag 13

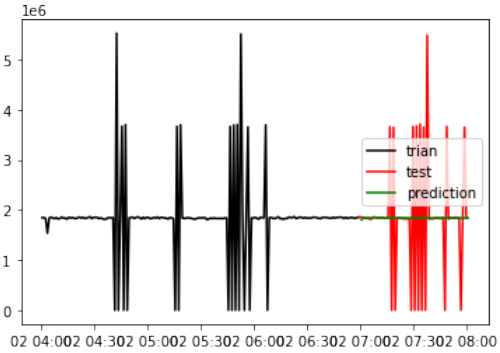


(b) AR model with lag 67

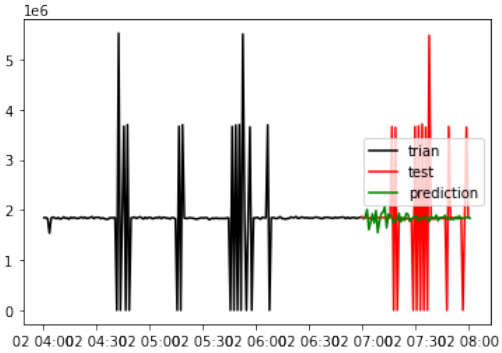
Figure 15: AR model for port all day 1 from 0:00 to 04:00

lag	13	67
RMSE	124852.651	792755.939

Table 5: RMSE of AR model



(a) AR model with lag 13



(b) AR model with lag 67

Figure 16: AR model for port all day 1 from 4:00 to 08:00

lag	13	67
RMSE	1074350.188	1078114.886

Table 6: RMSE of AR model

After the trained model, the training set shifts over by 1 hour and re-develop the model. As seen from the figure red5 and figure red6 our prediction failed. In the case where it had high variance and our prediction had low variance such as, figure red15a =, decreasing the parameter produced better results. For cases where it presented some pattern in training and no visible pattern in the test set, figure red29a the parameter tuning did not do much. The results did not come out great because AR model is very simple.

3.4.2 Moving Average model

ARMA is a combination of the AR model and MA model. Moving Average model is where y_t is dependent on a random error term. The MA model can be represented as:

$$MR(p) = \beta\_0 + \epsilon_t + \pi * \epsilon_{t-1} + ... + \pi_p * \epsilon_{t-p}$$

Intergration of both model produce:

$$ARMA(p) = \beta\_0 + \beta\_1 * y_{t-1} + ... + \beta\_p * y_{t-p} + \epsilon_t + \pi * \epsilon_{t-1} + ... + \pi_p * \epsilon_{t-p}$$

3.4.3 ARIMA

ARIMA intergrates ARMA to make it stationary. The ARIMA model calculates the data set's station-ability through an Augmented Dickey-Fuller (ADF) test and a Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test.

KPSS	Results
Test Statistic	0.255661
p-value	0.100000
Critical Value (10%)	0.347000
Critical Value (5%)	0.463000
Critical Value (2.5%)	0.574000
Critical Value (1%)	0.739000

ADF	Results
Test Statistic	-7.093052e+00
p-value	4.359830e-10
Critical Value (10%)	-2.567877e+00
Critical Value (5%)	-2.863619e+00
Critical Value (1%)	-3.435056e+00

Table 7: Stationary testing results

From the table red7, the KPSS p-value can be seen as less than the significance level of 0.05 and the ADF test statistic is greater than it's critical value. Therefore, both tests show our data to be highly stationary. ARIMA contains three parameters p, d and q. P is the number of time lags, d is the degree of differences and q is the moving-average. Therefore, data is proven as station with d fixed as 0.

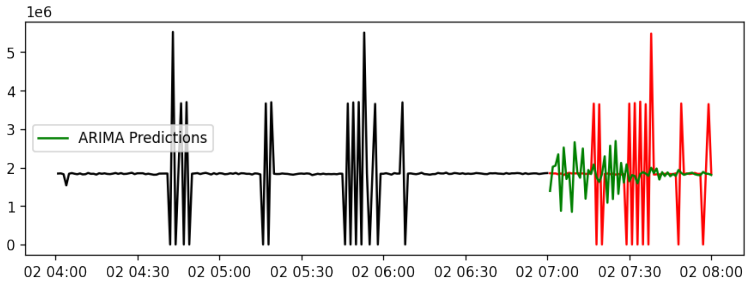


Figure 17: ARIMA prediction

3.4.4 Anomaly detection using ARIMA

In conclusion ARIMA performed the best out of all the forecasting model, but compared to knn it performed poorly. Similar to ARIMA, SARIMA and SARIMAX results were unusable. Our assumption was that the measured time was too short to have any seasonal patterns.

3.4.5 Limitations of ARIMA

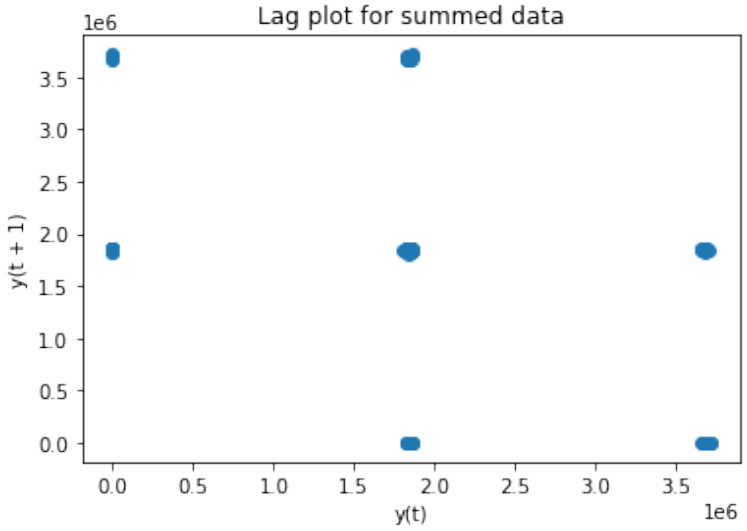


Figure 18: lag plot

The figure red18 The results from time forecasting model concludes that the data is a non AR-model. There were too may anomalies within the data-set. This mean that y_t is independent from $y_{t-n}, \forall n \in \mathbb{R}; n < t$. This represents that the data shows random walk property.

3.5 Multivariate statistical analysis

Multivariate statistics are useful, when the data is scaled up to the high-dimensional space [21]. To detect anomalies in a quantitative approach, the probability distribution p(x) from the data points need to be calculated first. Then, the p(x) will be compared with the threshold r when the new sample x comes in. If p(x) is smaller than r, it will be considered an anomaly. This is because the normal data tends to have a large p(x) while anomalous data have a small p(x) [22].

Mahalanobis distance (MD) refers to the distance from observation to center, taking into account the covariance matrix [23]. MD was used to classify a testset as either normal and anomaly. To achieve this, PCA was done for feature engineering and the number of components was set to 5. Subsequently, MD was calculated and if the distance was above the threshold, it was classified as anomaly.

Overall, the model detected 107 data points as anomalies and 34953 points as normal

	0	1	2	3	4
0	-0.032705	-0.013041	0.002666	-0.001024	0.000635
1	0.078739	-0.001995	-0.002349	0.011256	-0.003205
2	-0.032704	-0.013038	0.002657	-0.001024	0.000635
3	0.089726	-0.005352	-0.001754	0.012086	-0.002849
4	0.078382	-0.003297	-0.002700	0.011079	-0.002495
...	...	...	...	...	...
35055	0.081199	0.006649	0.000652	-0.007483	0.001534
35056	0.079462	0.000217	-0.001132	-0.012655	0.003440
35057	-0.032487	-0.012227	0.002894	-0.000122	0.000496
35058	0.079623	0.000797	-0.000977	-0.013052	0.003169
35059	-0.032489	-0.012234	0.002916	-0.000122	0.000495

(a) Dataset after PCA

	Mob dist	Thresh	Anomaly
0	0.962510	4.758264	False
1	2.493327	4.758264	False
2	0.962195	4.758264	False
3	2.652387	4.758264	False
4	2.371624	4.758264	False
...	...	...	...
35055	1.997671	4.758264	False
35056	2.655802	4.758264	False
35057	0.922578	4.758264	False
35058	2.650059	4.758264	False
35059	0.923482	4.758264	False

35060 rows × 3 columns

```
False    34953
True       107
Name: Anomaly, dtype: int64
```

(b) MD Anomaly Detection Result

Figure 19: Mahalanobis Distance

3.6 Isolation Forest

Another model discovered during the initial research that is assumed to be appropriate to satisfy our project objectives was the isolation forest. An isolation forest is an unsupervised model that was specifically designed to detect anomalies in large datasets, and can still be used effectively even when trained on a training set of fully benign data, such as the dataset provided. Since one of our project objectives was to identify devices acting outside of their normal range, and as the model is trained on the set of the data where the devices act within their normal range, isolation forest model would be useful in identifying extreme outliers in our dataset, while still passing off most data points in the dataset as normal.

Anomaly detection can be solved by supervised learning algorithms if enough information of anomalous behavior is given before modeling. Initially without the feedback it was difficult to identify. Thus, we model this as an unsupervised problem using algorithms such as an Isolation Forest [18]. Therefore, an isolation forest was chosen and aimed to detect the extreme outliers which are far from the main clusters [24].
First, the dimension reduction was done according to the correlations calculated before. As mentioned earlier, byte and packet were highly correlated and so only the byte column was extracted for modelling. To aid in this, the isolation forest was built. The hyper-parameters used for the model is as below:

1. n_estimators: the number of base estimators in the ensemble
2. max_samples: the number of samples to draw from X to train each base estimator
3. contamination: the proportion of outliers in the data set
4. bootstrap:
    - If True, individual trees are fit on random subsets of the training data sampled with replacement.
    - If False, sampling without replacement is performed.

Optimal set of hyper-parameters was found by various experiments which set different values for each parameter:

1. n_estimators: [30, 40, 50, 60, 70]
2. max_samples: [30, 40, 50, 60, 70]
3. contamination: [0.001, 0.002, 0.003, 0.004, 0.005]
4. max_features: [1,5,10]
5. bootstrap: [True, False]

Nextly, mean and standard deviation of the test score were calculated for each set and based on the scores the ranking was set.

params	split0_test_score	split1_test_score	split2_test_score	...	mean_test_score	std_test_score	rank_test_score	sp
{'contamination': 0.1, 'n_estimators': 100}	0.783086	0.896321	0.690958	...	0.797975	0.066998	2	
{'contamination': 0.1, 'n_estimators': 105}	0.780091	0.891329	0.699087	...	0.799971	0.063366	1	
{'contamination': 0.1, 'n_estimators': 110}	0.761409	0.899315	0.698659	...	0.797690	0.068139	3	
{'contamination': 0.2, 'n_estimators': 100}	0.648317	0.816172	0.594552	...	0.663206	0.078538	6	
{'contamination': 0.2, 'n_estimators': 105}	0.649601	0.815174	0.597547	...	0.664233	0.077398	5	
{'contamination': 0.2, 'n_estimators': 110}	0.652310	0.825014	0.601398	...	0.664689	0.081884	4	

Figure 20: Results of different hyper-parameters 1

	rank_test_score	mean_test_score	params
17	1	0.999886	{'bootstrap': True, 'contamination': 0.001, 'm...
257	1	0.999886	{'bootstrap': False, 'contamination': 0.001, '...
284	3	0.999715	{'bootstrap': False, 'contamination': 0.001, '...
44	4	0.999686	{'bootstrap': True, 'contamination': 0.001, 'm...
280	5	0.999601	{'bootstrap': False, 'contamination': 0.001, '...
...	...	...	...
212	475	0.980747	{'bootstrap': True, 'contamination': 0.005, 'm...
217	477	0.979549	{'bootstrap': True, 'contamination': 0.005, 'm...
457	477	0.979549	{'bootstrap': False, 'contamination': 0.005, '...
456	479	0.979350	{'bootstrap': False, 'contamination': 0.005, '...
216	479	0.979350	{'bootstrap': True, 'contamination': 0.005, 'm...

480 rows × 3 columns

Figure 21: Results of different hyper-parameters 2

The set of below:

1. bootstrap: True
2. contamination: 0.001
3. max_features: 5
4. max_samples: 30
5. n_estimators: 40

The best test score was achieved from the above set.
Then, the result was projected into a 3 dimensional space to visualise by PCA.

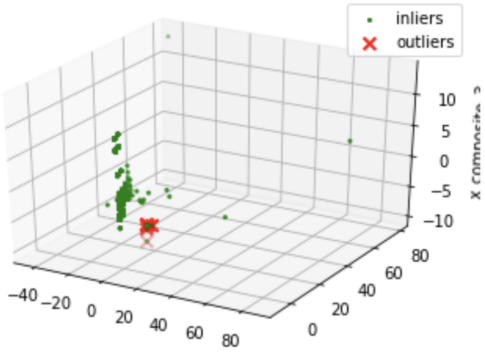


Figure 22: rank1 hyper-parameter's 3dim visualisation

Now, the result was projected into a 2 dimensional space for an intuitive understanding by utilising the PCA.

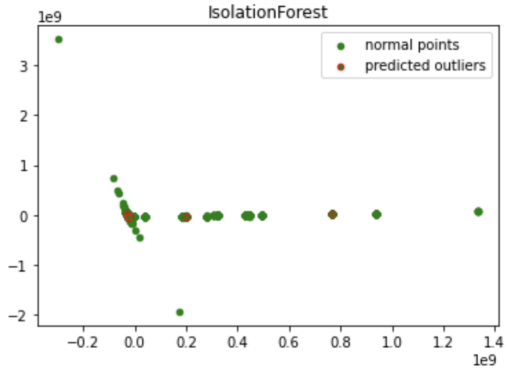


Figure 23: rank1 hyper-parameter's 2dim visualisation

As can be seen, most of the data points were classified as normal points including extreme outliers such as the point found on the coordinate [1,0] on Figure 23. Thus, the performance of anomaly detection was modest as extreme outliers were incorrectly classified.

Sequentially, the plots with different hyper-parameter settings were created to find the optimal set which satisfies the objective. And the set below was assumed to be optimal in the sense that it correctly detected the extreme outlier:

1. bootstarp: True
2. contamination: 0.004
3. max_features: 10
4. max_samples: 30
5. n_estimators: 40

Similarly, the result was projected on both 2 dimension and 3 dimension as below:

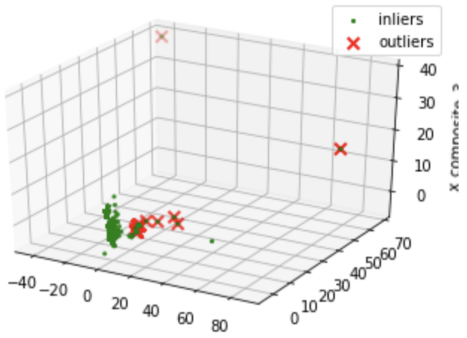


Figure 24: Optimal set of hyper-parameter's 3d visualisation

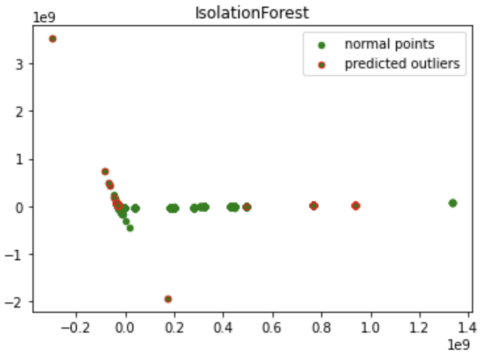


Figure 25: Optimal set of hyper-parameter's 2d visualisation

Though the extreme outlier such as the right-most point on figure 25 was misclassified, overall, the performance of extreme-outliers was satisfactory as such the 2nd and 3rd points from the right were correctly classified as outliers.

3.7 One-class SVM

Detecting outliers is important in the sense that it has the potential to detect cyber attacks which are also likely to behave like outliers.

One-Class Support Vector Machines are unsupervised models for finding the decision function and therefore useful for anomaly or outlier detection [30]. Unlike normal supervised SVM, one-class SVM does not require labeled outputs during the training process. Instead it learns the boundaries that define normal data-points and classifies any data not included within those boundaries as outliers/anomalies [31].

The provided dataset was unlabelled, hence an unsupervised learning model such as SVM was suitable. Furthermore, it is important to note that the provided dataset contains only benign datapoints. Thus, it is assumed that the count of outliers is small compared to the count of non-outlier data-points. In our model, the percentage of outliers was estimated to be 1%. In other words, the ratio of the majority and minority class was set to be 0.99 to 0.01. Therefore, one-class SVM was chosen to build the anomaly detection model.

Data Preprocessing
The data was extracted from the "ToInternetRSVPPortAllIPurn" port to build the one class network model with byte and packet values used as features.

In the next step, a downsizing of the dataset was required due to the practical issues of computing on a large dataset. Thus, a time-window of 50 was set, with the input and output values being the mean of the 50 consecutive data points. In addition, all the data-points with 0 value were dropped since it indicated that the device was not working and so not useful in detecting outliers. This downsizing resulted in the dataset being reduced from an original size of 14376698 datapoints, to 287533 datapoints.

Meanwhile, the consecutive order of the averaged data-points needed to be preserved in order to achieve real-time anomaly detection. Given the time-series nature of the dataset, the first 80% of sequenced entries (230,026) was used during the training process of the model with the last 20% (57,506) being used for testing purposes. The training data spanned the time between the 2nd and 13rd of September. The test data spanned from the 13th to the 16th.

Hyper parameter

Now, three hyper parameters were selected to train the one-class SVM model. Firstly "nu" was set to 0.01 which set the model's expected percentage of anomalies to 1 percent. This was because originally, 1% of the dataset was estimated to be outliers. Secondly, the radial basis function - "rbf" was chosen as the kernel type to map data from lower dimensional to higher dimensional space. Thirdly, the kernal coefficient "gamma" was set to auto as the rbf kernel was used. As a result, the kernel coefficient was set to 1 over the number of features.

Initial Result

Since, the data provided were all consists of benign data, it is assumed that no cyber attack data is given. Since the data provided consisted entirely of benign data, it was assumed that no cyber attack data was given. Thus, true positive and false negative were used as model metrics. Normal data points were defined as positive and anomalies defined as negative. The results obtained with the default threshold were as following:

	Prediction
normal	51210
anomaly	6297

	Test
TP	0.9
FN	0.1

Figure 26: results of the one-class SVM

Among 57,506 data points in the test dataset, 51210 were classified as normal and 6296 as anomalies and so the true positive ratio of the test data was 0.8 and false negative was 0.1.

Customise Threshold

Instead of using the default threshold for identifying outliers, the threshold was customized by calculating the percentile of the score. To achieve this, first, the one-class SVM scores of the test dataset were extracted. Then the new percentage of the percentile was set to calculate the customized score threshold.

For instance, 1.06 was set to be our new threshold by using scores for 4% of outliers. The red dots in the graph represent the outliers. Before customization, the one-class SVM did not have a clear boundary as the red dots were randomly distributed. As such, the red dots distributed near 4000 bytes were expected to be classified as normal. However, in actuality after customization, the outliers were concentrated on the edges of the dataset and fewer data points were labeled as anomalies.

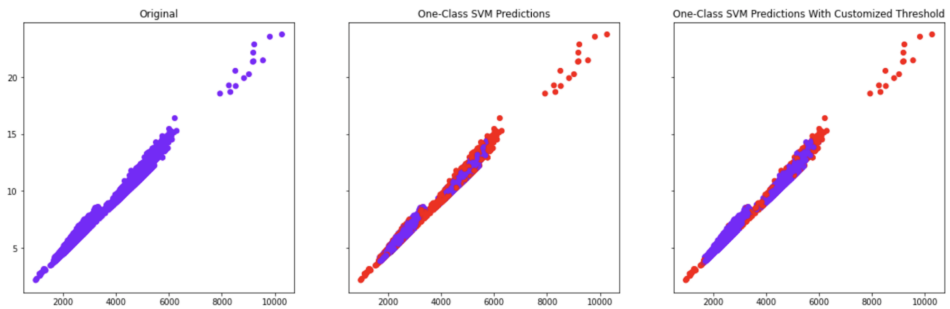


Figure 27: Customised threshold:1.06

Further reduction of the percentile to 0.1 decreased the number of outliers as can be seen below. With a new score threshold of 0.2, the model's performance in outlier detection was improved in the sense that most of the red dots were distributed near the edges of the dataset which had the most distance from the main cluster. On the other hand, the data points belonging to the main cluster located between 2000 and 7000 bytes were classified as normal.

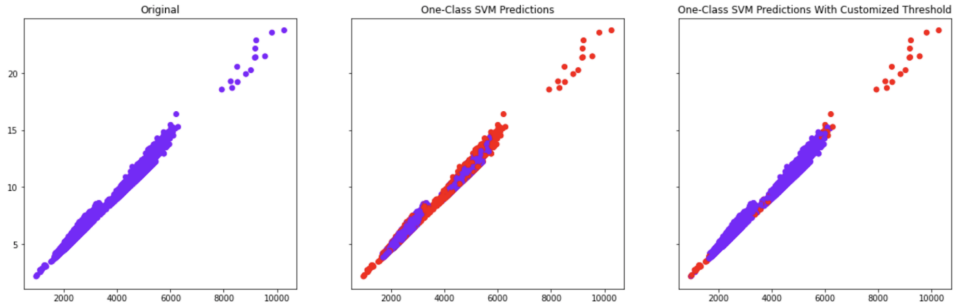


Figure 28: Customised threshold 0.2

3.8 LSTM Autoencoder

Autoencoders are neural networks that aim to copy their inputs to their outputs. It consists of two modules: [37]
1. Encoder: It compresses the input into a latent-space representation and so reduces the dimension
2. Decoder: It reconstructs the input from the latent space representation.

The LSTM Autoencoder is an implementation of an auto-encoder for sequential data using an Encoder-Decoder LSTM architecture. The Autoencoder trains the normal pattern and classifies the data as normal if it has a similar normal pattern as trained. Therefore, the reconstruction error will be small. However, if the decoder imperfectly reconstructs the input and so reconstruction error is high then the sample will be labeled as anomaly [36].

Data Preprocessing
The data was extracted from the "ToInternetPortAll" port following the time-series order provided. Then all the entries were grouped by the same time value and returned the mean. As a result, the data size was downsized from an original size of 14376698 datapoints, to 134192 datapoints. Furthermore, only byte data was used as the sole feature due to packet being highly correlated (corr = 0.99) with the byte feature. The data was then standardized using StandardScalar.

Train Test Split
The first 70% of the dataset was used for training with the rest of the 30% used for testing. The training data spanned the time between the 2nd and 8th of September. The test data spanned from the 8th to the 12th.

Data Loader
Nextly, the dataset was converted such that each entry would hold a time sequence value and a real byte value which could then be fed into an LSTM for training and testing. A window length of 890 entries was used for each temporal slice value which approximated to two hours of time.

time	Byte
2022-09-02 00:24:59.418	-0.468675
2022-09-02 00:24:59.419	-0.351945
2022-09-02 00:24:59.420	0.182284
2022-09-02 00:24:59.421	0.166706
2022-09-02 00:24:59.422	-0.370112
...	...
2022-09-02 02:24:00.392	-0.739051
2022-09-02 02:24:00.393	-0.568701
2022-09-02 02:24:00.394	-0.123840
2022-09-02 02:24:00.395	-0.023219
2022-09-02 02:24:00.396	1.675345

890 rows × 1 columns

(a) Data for 2hours

timesteps = 3

Data entries with single time value	Data entries with a temporal slice value
[Time A, Value A] [Time B, Value B] [Time C, Value C]	[[Value A, Value B, Value C], Value C] [[Value B, Value C, Value D], Value D] [[Value C, Value D, Value E], Value E]

(b) Example of dataloader

Figure 29

LSTM Architecture

The LSTM autoencoder predicts the output of the (t)th temporal slice based on the historical data up to the (t-1)th temporal slice. Then the predicted output values from the LSTM model will be compared with real test value outputs. The difference between the two represents the error. To accomplish this, a sequential model from keras was used.

Dropout layer randomly drops units in a neural network, hence it forces a neural network to learn more robust features that are useful in conjunction with many different random subsets of the other neurons. Therefore, it prevents overfitting [38]. As such, the dropout ratio is set to be 0.2 and so 0.2 of the output from the previous LSTM layer was dropped. Then the Repeatvector layer was created to prepare the input for the LSTM layer in the decoder. The timesteps was given as a parameter so that the Repeatvector replicates the feature vector as many times as its timesteps. Next, the LSTM layer was added with units size of 64 which is the same units size of the LSTM layer in the encoder. This is because the decoder layer was designed to unfold the encoding thus, the decoder layer is stacked in the reverse order of the encoder. And the parameter return_sequence was set to True to make each cell per timestep emit a signal. Lastly, TimeDistributed(Dense(number_of_features)) layer was added to get the output where length is equal to the number of features outputted from the previous layer.

```
model.add(keras.layers.LSTM(
   units=64,
   input_shape=(X_train.shape[1], X_train.shape[2])
))
model.add(keras.layers.Dropout(rate=0.2))
model.add(keras.layers.RepeatVector(n=X_train.shape[1]))
model.add(keras.layers.LSTM(units=64, return_sequences=True))
model.add(keras.layers.Dropout(rate=0.2))
model.add(keras.layers.TimeDistributed(keras.layers.Dense(units=X_train
.shape[2])))
model.compile(loss='mae', optimizer='adam')
```

Figure 30: LSTM Autoencoder Architecture

Hyper-parameter
The optimizer used was Adam as empirical results demonstrate an effective performance and efficiency compared to other stochastic optimization methods [39]. Robustness is the capacity of a model to remain stable as such only small changes happen when it is exposed to noise. Thus, the metric should be less affected by outliers. Mean Absolute Error refers to the average of all absolute errors [40] and so it is less sensitive to outliers compared to the other metrics such as Mean Squared Error. Since, the power function in MSE enforces the importance of outliers whereas absolute value in MAE accounts for the negative values properly. Therefore, MAE was used to calculate the loss [41]. The model was then fed with an epoch size 10, batch size of 20 and the time order maintained. Moreover, the parameter validation$_s plitissetto 0.1 toobtainthevalidationloss$.

Loss
As can be seen from the below plot, over the epochs, both train and validation loss have been decreased but more rapid loss reduction was observed during the training phase.

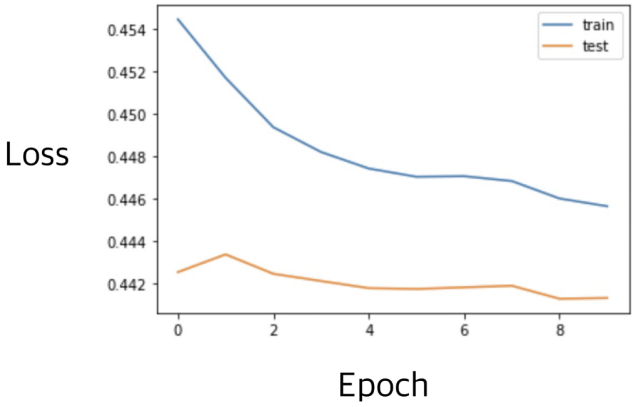


Figure 31: loss over the epoch

Threshold
The below plot shows the density of MAE during the training. As can be seen, the majority of loss was concentrated between 0.0 and 1.0 and after 1.0 significant density reduction was observed. Thus, the threshold for classification of anomalies was set at 1.0.

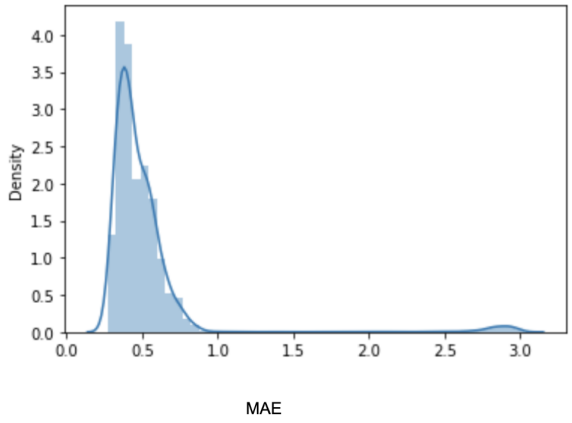


Figure 32: MAE density graph

Therefore, if test MAE was greater than the threshold 1.0 it was flagged as an anomaly. The below plot shows the test loss and threshold set.

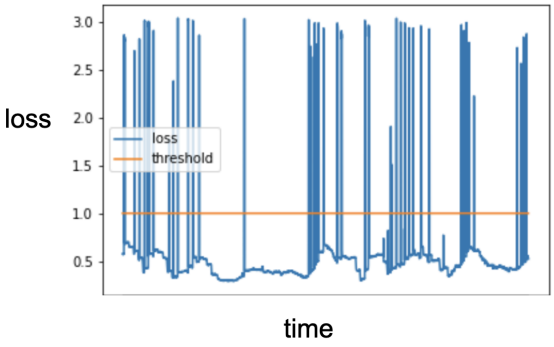


Figure 33: test loss with threshold

Anomaly detection visualisation
Before visualisation, the data standardised was inversed back to the original value as actual byte size value was the point of interest. In the diagram, the y-axis represents the byte size and the x-axis represents time. The red points represent the anomalies. And as below, the performance of anomaly detection was modest in the sense that the extreme outliers such as top right data points which are distant from the main distribution were not classified as anomalies.

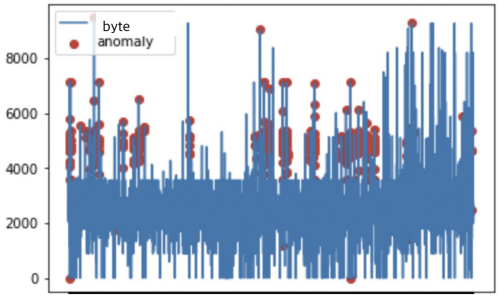


Figure 34: Anomaly Detection Visualisation

It is presumed that an over down-sizing of the dataset lost meaningful data. Moreover, a two-hour sequence of data is not enough to train the model optimally. Thus, further improvements would be incrementing the size of the time-window, data, and epoch.

Evaluation
Observing the result, 437 test data points were classified as anomalies. And 97.8% true positive and 2.2 false negative were achieved.

True Positive	97.8%
False Negative	2.2%

	time	loss	threshold	anomaly	Byte
	2022-09-07 01:08:00.400	1.171532	1	True	3.534104
	2022-09-07 01:08:00.401	2.260101	1	True	1.614252
	2022-09-07 01:08:00.403	2.303839	1	True	2.706339
	2022-09-07 01:10:00.365	2.717125	1	True	2.617004
	2022-09-07 01:10:00.366	2.779757	1	True	4.011659
	...	...	...	...	...
	2022-09-09 00:40:51.111	2.190702	1	True	-0.449115
	2022-09-09 01:00:51.155	1.584156	1	True	3.241191
	2022-09-09 01:00:51.156	2.425356	1	True	4.375789
	2022-09-09 01:00:51.157	2.827994	1	True	3.258182
	2022-09-09 01:01:51.169	2.875425	1	True	-0.070848

437 rows × 4 columns

Figure 35: LSTM Anomaly Detection results

3.9 K-means

Collective anomalies happen in datasets where the instances are related to each other either sequentially or spatially. K-means is an unsupervised model which groups the unlabeled dataset into different clusters. Contextual anomaly detection problems can be solved by applying k-means clustering to a time series dataset. The K-means model learns the normal pattern by creating k main clusters. Then, during the test phase, each datapoint is classified into one of the previously trained clusters. If the test data point is significantly distanced from its selected main cluster, it is then flagged as an anomaly [32].

Data Preprocessing
The data was extracted from the "ToInternetRSVPPortAllIPurn" port to build the one class network model with byte and packet values used as features. Time series data entries with the same time were grouped together with values being meaned to reduce the size of the dataset. In addition, all the datapoints with 0 values were dropped. This downsizing resulted in the dataset being reduced from an original size of 14376698 datapoints, to 134192 datapoints. The first 70% of sequenced entries (96649) which spanned from the 2nd to the 12nd of September, were used during the training process of the model with the last 30% (38263) spanning the time between the 12th and 16 being used for testing purposes.

Optimal number of clusters
Finding an optimal number of clusters is important since as 'k' (number of clusters) increases, the sum of squared distances from each point to the main center will converge to zero. In other words, the sum of squared distances will converge to zero as each sample will form its own cluster if the k is equal to the number of samples [33]. Thus, the Elbow method was used to determine the optimal number of clusters.

Inertia provides a measurement for how far points within a cluster are generally distanced. It is calculated by measuring the sum of the squared distances between each data point and its centroid across one cluster [34]. The optimal number of clusters 'k' is the Elbow point where the decrease in inertia is not significant for additional clusters [35]. Thus, upon experimentation, 3 was selected as the number of optimal clusters. This was because, when k equalled 3 the inertia had dropped to near 1.0 and subsequent rates of decrease in inertia was diminishing. Furthermore, as can be seen on the right plot, the score graph levels off after 3 clusters, implying that an addition of clusters does not explain much more of the variance.

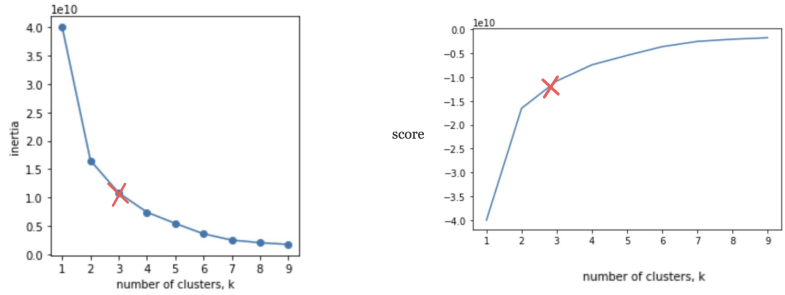


Figure 36: Elbow method

Result
As a result, the cluster distribution when k is 3 is as the following:

Cluster	Data count
0	87792
1	5365
2	3492

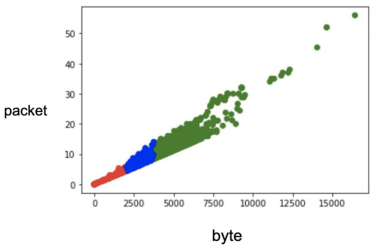


Figure 37: Cluster when k=3

As shown on the right plot, each color represents a different cluster and most of the data points were classified as cluster 0.

Hence, now all the data points were classified within three clusters. Subsequently, the distance of each point from the nearest centroid was calculated. The outlier fraction was estimated to 0.01 and the number of outliers was obtained. The threshold was set to be the minimum distance of outliers. And, if the distance was greater or equal to threshold, the data was flagged as an anomaly. Otherwise the data points were assumed as normal.

	Condition	Color
Normal	Distance < threshold	Blue
Anomaly	Distance ≥ threshold	Red

Figure 38: Anomaly detection

Visualisation
The blue points on the graph below represent the normal data points while the red represent the anomalies. As can be seen, the extreme outliers which were significantly distant from the main clusters located between 12500 and 15000 were classified as anomalies.

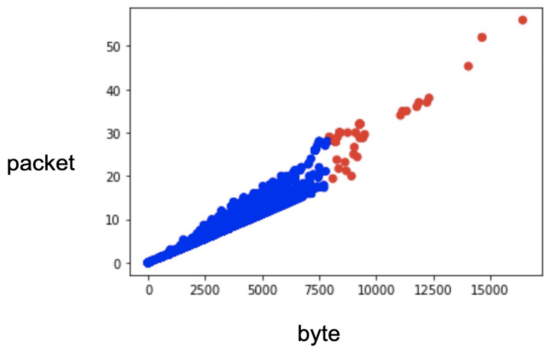


Figure 39: Byte-Packet visualisation

The below graph shows the byte and packet flow through time spanned from 12th of September and 16th. Observing the results, across the entire time period, the data points were mainly distributed between 0 and 7500 and 0 to 20 on byte and packet plot respectively. And, the extreme outliers which were isolated from the main distribution were classified as anomalies. For example, the top-left and top-right data points on both plots were flagged as anomalies.

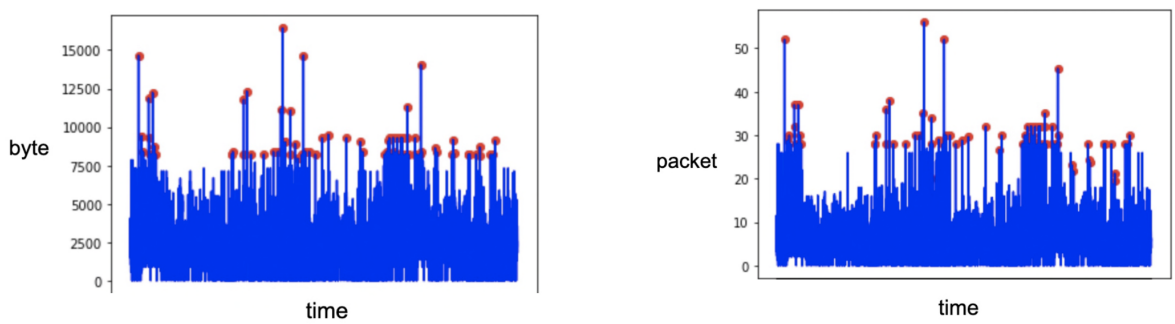


Figure 40: k=3 visualisation

As can be seen when k is 7, more data points were classified as outliers compared to when k=3. As such, the red dots near 0 were flagged as anomalies and it decreased the performance of anomaly detection. This is

because, throughout most of the time period, the data points were distributed near 0 meaning that it was expected to be classified as normal when in fact it was classified as anomalies. Thus, upon observation, the performance of outlier detection was higher at k=3 compared to k=7.

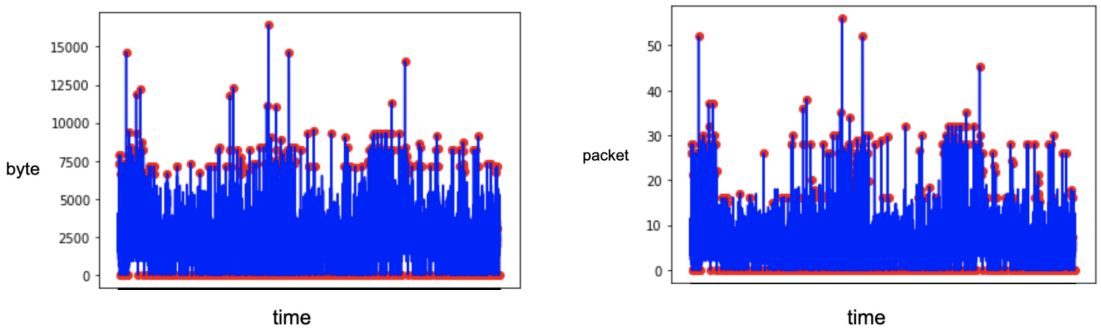


Figure 41: k=7 visualisation

4 Project Insights

The knn method best fit to our project aim. It uses minimal memory by storing 24 hours data, able to find attacks in 2 seconds and have above 99% accuracy.

In this project a variety of different models were used to attempt extreme outlier anomaly detection including one-class SVM, LSTM autoencoder and K-means. The metrics for model evaluation was the running time and false negatives. Additionally intuitive judgements were made based on the visualisation of anomaly detection by each model to determine the best approach.

In terms of running time, the K-means was the quickest followed by the one-class SVM and the LSTM autoencoder. Comparing visualisations of each model, the anomaly detection of K-means and one-class SVM was effective whereas the LSTM autoencoder was not. Finally, comparing the models on their FN, the one-class SVM had 0.1, K-means had 0.001 and LSTM was 0.22. So overall, K-means proved to be the most effective model.

4.1 Recommendations

Cybernet is able to deliver time, and devices that could be leaking out data by looking at the byte count. However, it is hard for the company to shut down the entire network to prevent the data to be uploaded to the internet. Therefore, we recommend back tracking TCP protocol to find the a network that contains virus/ attacked. Now looking at the network find out the type, method of attack and act accordingly. Simple methods include updating software and improving firewall.

References

[1] Hassan, M. (2022) State of IOT 2022: Number of connected IOT devices growing 18 to 14.4 billion globally, IoT Analytics. Available at: https://iot-analytics.com/number-connected-iot-devices/ (Accessed: November 6, 2022).

[2] Lueth, K.L. (2022) State of the IOT 2018: Number of IOT devices now at 7B – market accelerating, IoT Analytics. Available at: https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/ (Accessed: November 6, 2022).

[3] Bhuyan M.H, "Network anomaly detection: Methods, systems and tools", 2014, Scopus. Available at: https://www.scopus.com/record/display.uri?eid=2-s2.0

(Accessed: November 22, 2022).

[4] Hong J, "Integrated anomaly detection for cyber security of the substations", 2014,Scopus. Available at: https://www.scopus.com/record/display.uri?eid=2-s2.0-85053494115origin=inwardtxGid=5dd402e0f4eaa613685b4a2f5b508cf1 (Accessed: November 12, 2022).

[5] Mishra S, "warm intelligence in anomaly detection systems: an overview", 2021, Scorpus. Available at: https://www.scopus.com/record/display.uri?eid=2-s2.0-85053494115origin=inwardtxGid=5dd402e0f4eaa613685b4a2f5b508cf1 (Accessed: 21 November, 2022).

[6] Liu, B., Cai, M. and Yu, J. (2018) Swarm intelligence and its application in abnormal data detection, Swarm Intelligence and its Application in Abnormal Data Detection. College of Information Science and Technology, Jinan University. Available at: https://www.informatica.si/index.php/informatica/article/viewFile/752/602 (Accessed: November 6, 2022).

[7] Jindal R, "A new technique to increase the working performance of the ant colony optimization algorithm", 2013. Available at: , https://www.scopus.com/record/display.uri?eid=2-s2.0-84928253473origin=inwardtxGid=fb1aa413f483ff5872db2bab3292fbba (Accessed: November 21, 2022).

[8] Sourosh E, "A boosting ant-colony optimization algorithm for computer intrusion detection, in: Proceedings of the 2006 International Symposium on Frontiers in Networking with Applications", 2006. Available at: https://link.springer.com/chapter/10.1007/978-981-13-1810-8 37 (Accessed: November 26, 2022).

[9] C. -L. Tsai, C. -C. Tseng and C. -C. Han, "Intrusive behavior analysis based on honey pot tracking and ant algorithm analysis," 43rd Annual 2009 International Carnahan Conference on Security Technology, 2009, pp. 248-252. Available at: https://ieeexplore.ieee.org/document/5335531 (Accessed: November 13, 2022)

[10] Alam S, "A swarm intelligence based clustering approach for outlier detection", 2010. Available at: https://www.scopus.com/record/display.uri?eid=2-s2.0-84863137332origin=inwardtxGid=ebbeb8eff8a458ef9182dd1bdd8208c6 (Accessed: November 27, 2022).

[11] Mohemmed, A., Zhang , M. and Browne , W. (2010) Particle swarm optimisation for Outlier Detection: Proceedings of the 12th Annual Conference on Genetic and evolutionary computation, ACM Conferences. Available at: https://dl.acm.org/doi/10.1145/1830483.1830498 (Accessed: November 6, 2022).

[12] Elrawy, M.F., Awad, A.I. and Hamed, H.F.A. (2018) Intrusion Detection Systems for IOT-based Smart Environments: A Survey - Journal of Cloud Computing, SpringerLink. Springer Berlin Heidelberg. Available at: https://link.springer.com/article/10.1186/s13677-018-0123-6 (Accessed: November 16, 2022).

[13] A. A. Cook, G. Mısırlı and Z. Fan, "Anomaly Detection for IoT Time-Series Data: A Survey," in IEEE Internet of Things Journal, vol. 7, no. 7, pp. 6481-6494, July 2020. Available at: https://ieeexplore.ieee.org/document/8926446 (Accessed: 15 November, 2022).

[14] Shah, G. and Tiwari , A. (2018) Anomaly detection in iiot: Proceedings of the ACM india joint international conference on data science and management of data, ACM Other conferences. Available at: https://dl.acm.org/doi/10.1145/3152494.3156816 (Accessed: November 17, 2022).

[15] Bostani H, A hybrid intrusion detection architecture for internet of things, in: 8th International Symposium on Telecommunications, 2016. Available at: https://ieeexplore.ieee.org/abstract/document/7881893 (Accessed: November 24, 2022).

[16] Jaadi , Z. (2019) When and why to standardize your data, Built In. Available at: https://builtin.com/data-science/when-and-why-standardize-your-data (Accessed: November 13, 2022).

[17] [17] Adithya, K, "Anomaly Detection with Isolation Forest, Visualization", 2019. Available at: https://towardsdatascience.com/anomaly-detection-with-isolation-forest-visualization-23cd75c281e2 (Accessed: 23 November, 2022).

[18] Brownlee, J. (2021) Autoregression models for time series forecasting with python, MachineLearningMastery.com. Available at: https://machinelearningmastery.com/autoregression-models-time-series-forecasting-python/ (Accessed: November 3, 2022).

[19] Kumar, V.: Feature bagging for outlier detection. In: ACM Conference on Knowledge Discovery andData Mining (SIGKDD), (2005) (Accessed: November 8, 2022).

[20] Xiang, H, Zhang, X, Order preserving hashing based isolation forest for robust and scalable anomaly detection. In: ACM International Conference on Information and Knowledge Management (2020) (Accessed: November 6, 2022).

[21] Kaya, I. (2020) Anomaly detection and Mahalanobis distance, Medium. Medium. Available at: https://medium.com/@ikaya754/anomaly-detection-and-mahalanobis-distance-25b21b7cfe5b (Accessed: November 25, 2022).

[22] Vegard, F 2019, 'How to use machine learning for anomaly detection and condition monitoring'. Available at: https://towardsdatascience.com/how-to-use-machine-learning-for-anomaly-detection- (Accessed: November 13, 2022).

[23] Mishtert, T 2019, 'Outlier Detection (Part 2): Multivariate. Available at: https://medium.com/towards-artificial-intelligence/outlier-detection-part-2-multivariate-df486f658d09 (Accessed: November 27, 2022).

[24] Jain, S. (2022) Anomaly detection for time series data, Medium. Medium. Available at: https://medium.com/@sidjain1412/anomaly-detection-for-time-series-data-dc92a1d1a49f (Accessed: November 14, 2022).

[25] Krishnan, A. (2019) Anomaly detection with Time Series forecasting, Medium. Towards Data Science. Available at: https://towardsdatascience.com/anomaly-detection-with-time-series-forecasting-c34c6d04b24a (Accessed: November 27, 2022).

[26] R, S. (2022) Univariate time series anomaly detection using Arima model, Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/08/univariate-time-series-anomaly-detection-using-arima-model/ (Accessed: December 13, 2022).

[27] Dotis, A. (2019) Why use K-means for time series data? (part One), Medium. devconnected-DevOps, Sysadmins amp; Engineering. Available at: https://medium.com/schkn/why-use-k-means-for-time-series-data-part-one-a8f19964f538 (Accessed: November 20, 2022).

[28] Georgiou, A.D. (2021) Why use K-means for time series data? (part Two): Blog, InfluxData. Available at: https://www.influxdata.com/blog/why-use-k-means-for-time-series-data-part-two/ (Accessed: November 16, 2022).

[29] https://medium.com/towards-data-science/time-series-analysis-resampling-shifting-and-rolling-F5664ddef77e (Accessed: November 14, 2022).

[30] Amy (2022) One-class SVM for anomaly detection, Medium. GrabNGoInfo. Available at: https://medium.com/grabngoinfo/one-class-svm-for-anomaly-detection-6c97fdd6d8af (Accessed: November 6, 2022).

[31] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. (Accessed: November 13, 2022).

[32] Anais, D 2018, 'Why Use K-Means for Time Series Data? (Part one)'. Available at: https://medium.com/schkn/why-use-k-means-for-time-series-data-part-one-a8f19964f538 (Accessed: November 22, 2022).

[33] Amy, 2022, '5 ways for Deciding Number of Clusters in a Clustering Model'. Available at: https://medium.com/grabngoinfo/5-ways-for-deciding-number-of-clusters-in-a-clustering-model-5db993ea5e09 (Accessed: November 21, 2022).

[34] Amelia, A. (2018) K-means clustering: From A to Z, Medium. Towards Data Science. Available at: https://towardsdatascience.com/k-means-clustering-from-a-to-z-f6242a314e9a (Accessed: November 13, 2022).

[35] Tola, A 2022, 'How to determine the optimal number of cluster for k-means clustering'. Available at: https://blog.cambridgespark.com/how-to-determine-the-optimal-number-of-clusters-for-k-means-clustering-14f27070048f (Accessed: November 11, 2022).

[36] Chitta, R 2019, 'Extreme Rare Event Classification using Autoencoders in Keras'. Available at: https://towardsdatascience.com/extreme-rare-event-classification-using-autoencoders-in-keras-A565b386f098 (Accessed: November 17, 2022).

[37] Nathan, H 2018, Deep inside: Autoencoders, Medium. Towards Data Science Available at: https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f (Accessed: November 24, 2022).

[38] Vivek, Y 2016, 'Why dropouts prevent overfitting in Deep Neural Networks'.Available at: https://vivek-yadav.medium.com/why-dropouts-prevent-overfitting-in-deep-neural-networks-937e2543a701 (Accessed: November 25, 2022).

[39] Brownlee, J. (2017) Gentle introduction to the adam optimization algorithm for deep learning, MachineLearningMastery.com. Available at: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/ (Accessed: November 28, 2022).

[40] Keith, M. (2018) Mean absolute error mae [machine learning(ml)], Medium. Medium. Available at: https://medium.com/@2080/mean-absolute-error-mae-machine-learning-ml-b9b4afc63077 (Accessed: November 12, 2022).

[41] Trevisan, V. (2022) Comparing robustness of Mae, MSE and RMSE, Medium. Towards Data Science. Available at: https://towardsdatascience.com/comparing-robustness-of-mae-mse-and-rmse-6d69da870828 (Accessed: November 20, 2022).

5 Appendix

We would be interested to see more on how you fine-tuned some models, including AR and k-means algorithm.

For the K-means, the Elbow method was used to determine 'k' the number of clusters. For the Isolation Forest, different hyper-parameter sets such as the one shown below were used to get the test score. The set with the best results were used in the final model.

1. bootstrap: True
2. contamination: 0.001
3. max_features: 5
4. max_samples: 30
5. n_estimators: 40

Figure 42

Could you elaborate more on how you measured the accuracy of your models (evaluation metric)?
True positive refers to correctly classifying a normal data point while a false negative indicate the probability of a normal data being classified as an anomaly.

We are also curious about what defines an anomaly that would pose high-risk to an IoT device and how you would classify them in the future.
For each model, a threshold was chosen. Data points were compared against the threshold in order to determine whether they were anomalies. For example, in the K-means model, the minimum distance of the outliers was set as a threshold. And if the distance was greater or equal to the threshold, a datapoint would be classified as an anomaly.

In terms of improvement, from our tutor's advice, it is best your Executive Summary remains brief (I.e., less than half a page)
The executive summary was shortened to less than half a page but still containing the key aspects of our project.

Your scope or EDA section could benefit from a small amount of context for the nature of the columns (e.g., What is port 80? What are packets used for?) as well as hyperparameters (I.e., what does each hyperparameter influence?)
Regarding hyperparameters, the report provided extensive details of hyperparameters used in each model. For example, in the one-class SVM model details were provided about 'nu' which set the model's expected percentage of outliers and 'rbf' a kernel type used for multi-dimensional mapping.

Additionally, the team found that your model performances slightly difficult to keep track of and compare to one another thus a table summarising your models' performances would be useful.
A table summarizing the metric performance for each model was added into the report.