With the support of the Erasmus+ Programme of the European Union

LEEDS BECKETT UNIVERSITY

School of Built Environment, Engineering, and Computing

Data Analytics and Visualization

Border Gateway Routing Protocol Anomaly Analysis and Classification

Student: Manasik Hassan, c7302653

Supervisor: Dr. Ah-Lian Kor

Abstract

Over the past years, the Internet experienced multiple events that caused massive disruptions for the internet services, such as cyber-worm attacks and power outages. Border Gateway Protocol (BGP), as the de facto routing protocol for the internet, was subject to these vulnerabilities, and BGP messages and handshakes exchanged between routers suffered from drastic variations. Understanding the BGP behavior during these anomalies is of great interest for the networking community to develop anomaly detection tools to observe the anomaly in the early stage of its occurrence. In favor of this, this work analyzes the BGP records of five well-known Internet incidents that caused large-scale internet instabilities (WannaCry, Nimda, Slammer, Moscow Blackout, and Code Red I). Four Data Analytics levels were applied: descriptive analysis, inferential analysis, machine learning, and deep learning. Descriptive and Inferential analysis results indicated some relevance of BGP behavior between different events. Six ML classifiers and two DL classifiers were used to detect regular and irregular BGP behavior. The classification results showed no significant differences in the obtained accuracies by MLP, RF, 1D-CNN, and GRU. Yet, MLP and RF achieved the highest accuracies among all models. Lastly and most significantly, the accuracies obtained by our proposed models showed superior performance to many other solutions suggested by authors in the literature review.

With the support of the Erasmus+ Programme of the European Union

LEEDS BECKETT UNIVERSITY

Table of Contents

1 Introduction

Now more than ever, the Internet is believed to be the most important invention that eased communications between humans; it affects all modern life aspects and helps people enhance the quality of their lives. Therefore, the capability of detecting any abnormal events that cause interruption of internet services is of enormous importance.

1.1 Background

Internet is commonly referred to as Interconnected Autonomous Systems (AS) that exchange information using standardized protocols to communicate with end devices, and thus border Gateway Protocol (BGP) is one of the essential elements to maintain the formation of the internet as the de facto internet routing protocol (Siganos and Faloutsos, 2004), (Rekhter, Hares and Li, 2006). Figure 1 shows the standard header format for BGP communications. Four possible messages can be sent on a BGP header (OPEN, UPDATE, NOTIFICATION, and KEEPALIVE), among which UPDATE is the most important. It's used to broadcast multiple information and properties between routers allowing them to choose the optimum path within many different possible ways to reach a specific destination (Rekhter, Hares and Li, 2006)

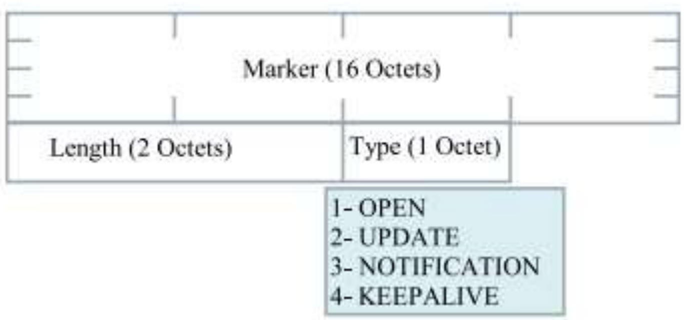


Figure 1. Common Message Header Format (Al-Musawi, Branch and Armitage, 2017).

BGP anomalies refer to any abnormal behavior of BGP UPDATES (Peng *et al.*, 2021), such as invalid routing announcements, unexpected networks IP addresses, huge increment or decrement in the no. of updates, or any behavior that doesn't follow regular routing schemes (Wübbeling, Elsner and Meier, 2014). BGP anomalies are classified into different types; indirect abnormalities and Link failure are on the list. While indirect anomalies relate to harmful activities that target the internet and Link failures are associated with internet backbone links malfunctioning, both anomalies were observed to cause immense instabilities in BGP UPDATES (Al-Musawi, Branch and Armitage, 2017) and thus studying BGP behaviors during these anomalies will help in developing detection tools to ensure networks stability, confidential data exchange, and the required quality of service for Internet operations.

1.2 Scope of the Research

Many BGP anomalies have been detected over the past years of the internet age. The scope of this research is limited to studying datasets for the following five well-known internet anomalies that caused massive inconsistency in BGP updates:

- **WannaCry**: WannaCry was a famous malware attack in May 2017. It works by gaining access to systems running Microsoft Windows 7 and encrypting the whole operating system and files in a device until the attacked person or organization pays the attacker a ransom to get the decryption key (Mohurle and Patil, 2017).
- **Moscow Blackout:** on 25 May 2005, the Chagino energy station in Moscow witnessed an operational failure that led to a total blackout in the Moscow energy pool (Voropai and Efimov, 2008). This blackout affected MSK-IX Internet Exchange and led to interrupting many ISPs in Russia (Al-Musawi, Branch and Armitage, 2017), (Trajkovic, 2020).
- **Slammer:** Slammer is a computer worm attack designed to affect Microsoft SQL servers, it took place in January 2003, and BGP message updates witnessed tremendous instability during this accident (Lad *et al.*, 2003).
- **Nimda and Code Red I:** are two computer worm attacks that occurred in September 2001 and July 2001, respectively, they targeted Microsoft OS machines, and BGP updates experienced significant abnormal behavior (*11th Annual USENIX Security Symposium — Technical Paper*, no date)**,** (Al-Musawi, Branch and Armitage, 2017)**.**

1.3 Aims and Objectives

1.3.1 Research Aims

As this research presents four levels of Data Analytics, it aims to deliver an in-depth understanding of various data analytics and visualization methodologies by using Internet anomalies datasets as a case for applying these methodologies.

1.3.2 Research Objectives

RO 1: Level 1 Analysis

To investigate and understand how BGP features are represented and varied over regular and anomaly periods using descriptive statistics, meeting the following sub-objectives:

RO 1.1: Explore the datasets and prepare the data for the following statistical operations if needed.

RO 1.2: Obtain numerical and graphical measures for central tendency and dispersion for different features in the BGP anomalies datasets.

RO 1.3: Explore the varied behavior of BGP updates messages before, during, and after the anomaly event.

RO 2: Level 2 Analysis

To compare and measure the differences between BGP behaviors in five famous internet anomalies using inferential statistics and to meet the following sub-objectives:

RO 2.1: Apply a one-way ANOVA test to determine whether BGP behaves the same way in all five anomalies.

RO 2.2: Apply two-sample t-tests to investigate the similarity in BGP behavior between every two anomalies.

RO 2.3: Examine the level of linear correlations between BGP features in all five datasets.

RO 3: Level 3 Analysis

To apply different Machine Learning models for binary classification of the internet traffic, meeting the following sub-objectives:

RO 3.1: Test the classification performance of six different ML classifiers (MLP, DT, KNN, RF, SVC, and NB) for our datasets.

RO 3.2: Compare the performance of these different classifiers in terms of accuracy, F1-score, and recall.

RO 3.3: Suggest an optimum Machine learning classifier for each type of anomaly.

RO 4: Level 4 Analysis

To apply different Deep Learning models for binary classification of the internet traffic, meeting the following sub-objectives:

RO 3.1: Test the classification performance of Convolutional Neural Networks and Gated Recurrent Units classifiers for our datasets.

RO 3.2: Evaluate the performance of these deep learning classifiers in terms of accuracy and loss for each anomaly type.

RO 3.3: Compare the achieved accuracies of machine learning classifiers vs. deep learning classifiers and investigate which classification models are more precise.

1.4 Rationale of the Research

Given that many BGP anomalies have been detected over the past years of the internet age, the scientific community has already made great efforts to tackle BGP behaviors and develop anomaly detection mechanisms. Nevertheless, there is no evidence for a comprehensive multi-level analysis that contains thorough insights into BGP behavior during anomalies. Starting from scratch, this work will apply and combine four data analytics levels (descriptive statistics, inferential statistics, machine learning, and deep learning) to identify numerous patterns of the interrelations of BGP features with anomalies and to build various models able to achieve high accuracy for this binary classification problem.

1.5 Contribution of the Research

This report contributes to the efforts being made to ensure internet stability and quality of service. By understanding BGP behavior in both regular periods and anomalies, how do different anomalies affect BGP behavior, the association of BGP features with each other's, and how to accurately classify regular and irregular traffic; this report provides sound recommendations for developing highly precise internet anomaly detection tools.

1.6 Outline of the Report

The rest of this report is organized as follows:

In the Literature Review previous work related to BGP anomaly detection using different data analytics methods is shown. In the Methodology section, various macro and micro methodologies conducted in this research will be demonstrated, followed by the Results and Discussion section to interpret the findings of applying four data analysis levels, and then Recommendations will be given. Finally, we conclude our work and propose some future investigations in the Conclusion and Future Work section.

2 Literature Review

Recently, many efforts have been made by the scientific community to develop tools for internet anomaly detection and traffic classifications using several statistical methods. Previous work-related to internet anomaly detection using BGP features will be explored in this section.

To begin with, descriptive analysis and central tendency measures can be used to highlight the significance of some network components, characteristics, and trends (Kolaczyk and Csárdi, 2014), e.g. (Borgnat et al., 2009) developed a tool to observe abnormal network traffic in which standard deviation was the primary measure. However, advanced statistical methods are required for a precise anomaly identification process (Marnerides, Schaeffer-Filho and Mauthe, 2014).

Al-Rousan and Trajković, (2012) investigated the performance of Support Vector Machine (SVM) and Hidden Markov Models (HMMs) for BGP anomaly detection. They used Slammer, Nimda, and Code Red I datasets in their works and attained the best F-Scores of 86.1% and 84.4% for SVM and HMM, respectively. The same authors also investigated the use of Naïve Bayes classifiers (Al-Rousan, Haeri and Trajković, 2012). (Dai, Wang and Wang, 2019) constructed an SVM binary classification model and compared the performance of different SVM kernel functions. For their model training, they applied Fisher-Markov algorithms to extract highly correlated features in Slammer, Nimda, and Code Red datasets; in their findings, SVM with RBF function achieved the best F-score of 96.03%. (Ding et al., 2016) worked with the same previous datasets to develop an accurate BGP detection tool; they applied SVM and long short term (LSTM) algorithms and got f-scores of 72.32% and 58.15% for both models, respectively. Furthermore, (Li et al., 2014) applied decision tree and extreme learning machine (ELM) methods for the previous BGP anomaly events and obtained accuracies of 78.8% and 80%, respectively. Better accuracy of 98% for these datasets was acquired by an MLP classifier using only eight extracted BGP features (Karimi et al., 2019).

Many other studies have been carried out to compare the performance of multiple supervised ML classifiers to find the optimum BGP anomaly classification model, e.g., Cosovic and Junuz, (2019) examined several ensemble machine learning methods for Slammer, Nimda, Code Red I, and Moscow blackout events, they find out that Random Forests classifiers give better accuracy than boot-strap bagging and adaptive boosting algorithms. Sanchez et al., (2019) followed a novel approach; they trained four ML models (SVM, NB, MLP, and RF) by using graphical features of BGP message updates (betweenness, closeness, eigenvector, etc.) instead of the commonly used statistical characteristics and pointed out that MLP and SVM outperformed NB and RF. Similarly, Hoarau, Tournoux and Razafindralambo, (2021) tested the accuracy of five ML algorithms (SVM, MLP, DT, NB and KNN) when using graphical features compared to statistical features, they concluded that although ML models based on statistical data are more accurate, using graphic elements still gives an adequate performance.

Alternatively, unsupervised machine learning models proposals were also spotted in the literature review. For instance, a Real-time BGP anomaly detection engine based on the DenStream unsupervised clustering approach was developed by (Putina et al., 2018). Edwards, Cheng and Kadam, (2019) used K-means and DBSCAN unsupervised clustering algorithms. They used Nimda, Slammer, two BGP misconfiguration events & Moscow blackout datasets in the training phase and tested the performance of the models on near real-time data. Their models achieved good accuracies for detecting anomaly occurrence; however, they emphasized that human involvement in the monitoring process is required along with their model.

Although the majority of BGP anomaly detection techniques found in the literature review are Machine Learning models, the adoption of Deep Learning algorithms in anomaly detection work is significantly evident, Xu and Li, (2020) used Neural Networks to select features with the highest correlation to abnormal behaviors in Slammer, Code Red, Nimda and AS leak events, they fed these features into a 3-layer fully connected NN and an LSTM classification models to obtain best F-scores of 86.4% and 90.1% respectively. McGlynn, Acharya and Kwon, (2019) trained DL auto-encoders on regular BGP traffic and tested its accuracy on anomalous traffic, achieving an F-score of 83%. Additionally, Artificial Neural Networks were used to identify several BGP anomalies by (Cosovic, Obradovic and Junuz, 2018). a Multi-Scale LSTM model was introduced by (Cheng et al., 2016); they proposed an architecture composed of a preprocessing layer followed by an LSTM model, the LSTM model itself is constructed from a single LSTM layer, a mean polling layer, and a logistic regression layer, they tested this model against three BGP misconfiguration anomalies. They showed that 99.5% accuracy could be achieved with an optimal time scale of 8. Same authors later in 2021 presented a different MS LSTM model consisting of a discrete wavelet transform layer and two LSTM layers. They applied it to four BGP events (AS Leak, Code Red I, Nimda, and Slammer), and it showed better performance than the baseline methods stated in their paper (Cheng et al., 2021).

3 Methodology

3.1 Macro Level Methodology

With the vast amount of data being generated in various fields, increasing efforts have been made to observe and extract meaningful patterns and valuable information from big data using data mining techniques. (Kurgan and Musilek, 2006) stated: 'Before any attempt can be made to extract this helpful knowledge, an overall approach that describes how to extract knowledge needs to be established'. Hence different Knowledge Discovery and Data Mining (KDDM) methodologies were presented.

The most applied models by data mining professionals are Knowledge Discovery in Databases (KDD), Cross-Industry Standard Process for Data Mining (CRISP-DM), and SEMMA (Sample, Explore, Modify, Model, Assess). KDD is a nine-step generalized DM model related to extracting useful information from databases iteratively and interactively. CRISP-DM was firstly introduced in 1999, and as the name implies, it is an industry-related standard framework for DM with six phases. SEMMA is another famous DM methodology consisting of five steps developed by the SAS Institute (Shafique and Qaiser, 2014).

The steps in the introduced models are related to each other's and following any model will be useful. CRISP-DM is the most widely used DM framework (Sharma, Osei-Bryson and Kasper, 2012); however, the KDD framework will be adopted for achieving the objectives of this academic report, as it's a broader model, unlike CRISP-DM and SEMMA which are considered industry-oriented models (Shafique and Qaiser, 2014).

The overall phases in the KDD model are illustrated in Figure 2.

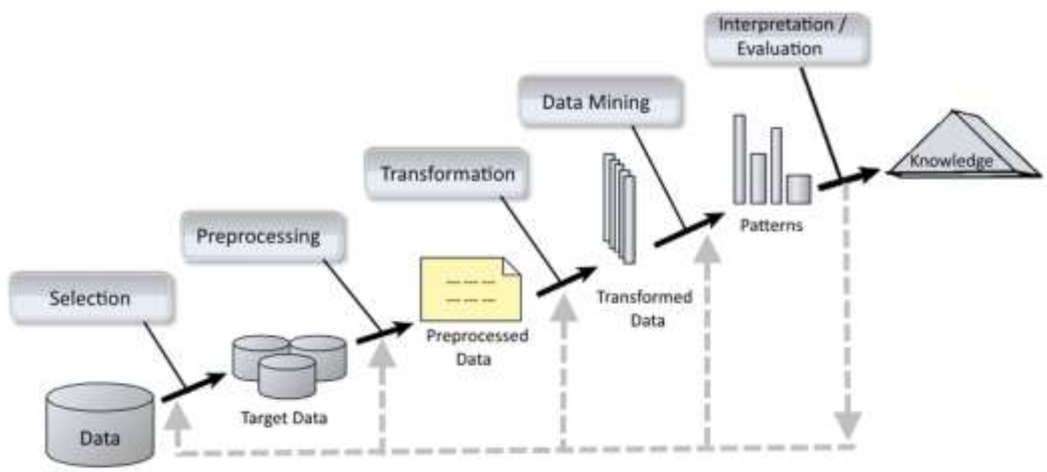


Figure 2. Steps in KDD process (Fayyad, Piatetsky-Shapiro and Smyth, 1996)

3.2 Micro Level Methodology

3.2.1 Descriptive analysis

As the name implies, descriptive statistics are the numerical and graphical representations used to analyze and describe the characteristics of a set of variables in a sample or population to help find out data patterns.

The Following Descriptive statistics will be applied to our datasets:

- Central tendency measures such as mean, median, and mode to describe our datasets and point out the central position of frequency distribution within that set of data (Conner and Johnson, no date), (Kaur, no date).
- Dispersion measures such as range, variance, quartiles, and standard deviation to determine the spread and width of values in our datasets (Conner and Johnson, no date), (Kaur, no date).
- Skewness and Kurtosis to measure the symmetrical behavior of BGP feature distribution to a normal distribution.

Figure 3 shows the mathematical formulas for famous descriptive statistics.

Mean	$\bar{x} = \frac{\sum x}{n}$	x = Observations given n = Total number of observations

Median	If n is odd, then
	M = $(\frac{n+1}{2})^{th}$ term
	If n is even, then
	M = $\frac{(\frac{n}{2})^{th} term + (\frac{n}{2}+1)^{th} term}{2}$
		n = Total number of observations

Mode	The value which occurs most frequently	
Variance	$\sigma^2 = \frac{\sum(x-\bar{x})^2}{n}$	x = Observations given $\bar{x}$ = Mean n = Total number of observations
Standard Deviation	$S = \sigma = \sqrt{\frac{\sum(x-\bar{x})^2}{n}}$	x = Observations given $\bar{x}$ = Mean n = Total number of observations

Figure 3. Formulas for multiple descriptive statistical methods (Statistics formulas-Mean, Median, Mode, Variance and Standard deviation, no date)

3.2.2 Inferential Statistics

Inferential Statistics methods are applied for finding broad generalizations from a sample valid to describe an entire population the sample belongs to. These methods rely on the probability theory and testing the null hypothesis against the research hypothesis. Inferential Statistics investigates whether the differences between two samples are significant or not by using the p-value, the product of hypothesis testing(Marshall and Jonker, 2011), (Allua and Thompson, 2009).

Choosing the most commonly used confidence interval by the scientific community of 95% with α = 0.05 for the hypothesis testing (Marshall and Jonker, 2011), The following Inferential methods will be applied to our datasets:

- One-way Analysis of Variance (ANOVA) test to investigate whether there is a significant difference in the mean values for BGP features in our five datasets.
- Two-Sample T-test to investigate if there is a significant difference in the mean values for BGP features between every two events.
- Pearson product-moment coefficient to study the strength of linear correlations between BGP features in each dataset.

3.2.3 Machine Learning Methods

Machine Learning is a sub-branch of Artificial intelligence and computer science that uses statistical models and algorithms to develop computer systems that learn, adapt, and make decisions without explicit instructions. One of the usual tasks in ML is supervised classification, in which the ML model is trained with labeled datasets to perceive a general phenomenon for each class, so it can use this knowledge to predict the classes for any similar situation that a model has no prior knowledge about.

Many techniques are used for binary classification tasks, and the following six ML Classifiers will be applied to our datasets:

- Multi-Layered Perceptrons classifier (MLP) is a class of feedforward artificial neural networks that computes the sum of weighted inputs and tests it against a threshold to output a decision. It has been developed to work on non-linearly separated classes. Its performance depends on the input layer, the activation function for the hidden layers, and the weight optimization of each input connection (Kotsiantis, Zaharakis and Pintelas, 2006).
- Decision Trees classifiers (DT) are constructed in a branched trees layout. Each node represents a feature in the datasets, and each branch connecting two nodes is associated with a cost. The classification starts from the root node, and the categorization depends on the values of the features, thus navigating the tree branches to conclude a final decision (Kotsiantis, Zaharakis and Pintelas, 2006).
- K-Nearest Neighbor classifiers (KNN) principle is to store the training data points. At the classification phase, the most frequent class among a number k of training samples close in the distance (neighbors) to the new input determines the new sample label (*Machine Learning, no date*).
- Support Vector Machines classifiers (SVM) use the idea of maximizing the margin at both sides of a hyperplane that separates different classes. It uses kernel functions to create the hyperplanes for linearly or non-linearly separated classes (Kotsiantis, Zaharakis and Pintelas, 2006).
- Random Forest classifiers (RF) are meta classifiers that construct multiple decision trees and make a final decision based on the average results achieved by all sub-decision trees (*sklearn*).
- Naïve Bayes classifiers (NB) are simple Bayesian networks that are probabilistic graphical models used to represent dependencies among different features using a directed acrylic graph (Kotsiantis, Zaharakis and Pintelas, 2006).

For evaluating the performance of different classification models, a confusion matrix and a classification report are shown for each case. Table 1 shows the confusion matrix, which demonstrates the no. of correctly classified testing points and the no. of misclassified ones. Table 2 shows several evaluation measures generated from the confusion matrix.

Table 1. A Confusion Matrix (M and M.N, 2015)

	Actual Positive Class	**Actual Negative Class**
Predicted Positive Class	True positive (*tp*)	False negative (*fn*)
Predicted Negative Class	False positive (*fp*)	True negative (*tn*)

Table 2. Several Evaluation Metrics for ML Classifiers Performance (M and M.N, 2015)

Metrics	**Formula**	**Evaluation Focus**
Accuracy (acc)	$\frac{tp + tn}{tp + fp + tn + fn}$	In general, the accuracy metric measures the ratio of correct predictions over the total number of instances evaluated.
Error Rate (err)	$\frac{fp + fn}{tp + fp + tn + fn}$	Misclassification error measures the ratio of incorrect predictions over the total number of instances evaluated.
Sensitivity (sn)	$\frac{tp}{tp + fn}$	This metric is used to measure the fraction of positive patterns that are correctly classified
Specificity (sp)	$\frac{tn}{tn + fp}$	This metric is used to measure the fraction of negative patterns that are correctly classified.
Precision (p)	$\frac{tp}{tp + fp}$	Precision is used to measure the positive patterns that are correctly predicted from the total predicted patterns in a positive class.
Recall (r)	$\frac{tp}{tp + tn}$	Recall is used to measure the fraction of positive patterns that are correctly classified
F-Measure (FM)	$\frac{2 * p * r}{p + r}$	This metric represents the harmonic mean between recall and precision values

3.2.4 Deep Learning Methods

Deep Learning is a sub-branch of machine learning. DL models are more complex than ML models, with multiple neural network layers and dense structures. DL tries to mimic the human brain's behavior by learning patterns and drawing conclusions from massive data. Same as ML, DL models are used for supervised classification tasks.

The following DL Classifiers will be applied to our datasets:

- Convolutional Neural Networks (CNN): CNN is a type of neural networks usually used for classification and image recognition tasks. CNN consists of three layers: A convolution layer, followed by a pooling layer, and a final fully connected layer. The complexity of the model increases throughout these layers, and hence its perception of the intended problem evolves (What are Convolutional Neural Networks?, 2021).
- Gated Recurrent Unit (GRU): GRU was developed to solve the short-term memory issue for the recurrent neural networks. It consists of two gates: the reset gate and the update gate to determine which portion of historical information to carry or drop (Phi, 2020).

To evaluate the performance of the DL models, loss and accuracy measures are obtained for each dataset. The loss represents the summation of errors made during model training and validation phases, while the accuracy measures the ratio of correctly made predictions to the total no. of predictions.

3.3 Datasets Description

The Datasets for the BGP Anomalies Detection were found on the IEEEdataport website. They were uploaded by the authors in (Trajkovic, 2020), who extracted 37 features of BGP update messages from BGP routing records published by Reseaux IP Europeens (RIPE) captured in Autonomous System No. 513 in Geneva, Switzerland (*Index of /rrc04*, no date).

The datasets contain data recordings for BGP message updates exchanges every minute during the internet anomalies as follows:

- Eight days records for WannaCry anomaly, four days of the attack, and two days before and after.
- Five days records for Moscow blackout, Slammer, and Code Red I, including the day of the attack and two days before and after the attack.
- Six days for Nimda, including two days of the attack and two days before and after the attack.

Table 3 gives information about the datasets dictionary (Al-Rousan and Trajković, 2012).

Table 3. Dataset Dictionary

Columns No.	Feature	Description
1	H+M	Time: Hour + Minute
2	H	Time: hour
3	M	Time: Minute
4	S	Time: Seconds
5	NOA	Number of announcements
6	NOW	Number of withdrawals
7	NOANP	Number of announced NLRI prefixes (Network Layer Reachability Information)
8	NOWNP	Number of withdrawn NLRI prefixes
9	AAPL	Average AS-path length (Average No. of AS peers in AS path)
10	MAPL	Maximum AS-path length (Maximum No. of AS peers in AS path)
11	AUAPL	Average unique AS-path length
12	NODA	Number of duplicate announcements (packets that have the same NLRI and AS-path attributes)
13	NODW	Number of duplicate withdrawals (packets that have the same NLRI and AS-path attributes)
14	NOIW	Number of implicit withdrawals (BGP announcements that have different AS-PATHs for previously announced NLRI prefixes)
15	AED	Average edit distance (the edit distance between two AS-PATH elements is the minimum number of modifications that need to be carried out to match the two attributes)
16	MED	Maximum edit distance
17	IAT	Inter-arrival time
18 – 28	MED1-11	Maximum edit distance = n, n = 7, . . . , 17
29 – 37	MAL1-9	Maximum AS-path length = n, n = 7, . . . , 15
38	IGP packets	Number of Interior Gateway Protocol (IGP) packets
39	EGP packets	Number of Exterior Gateway Protocol (EGP) packets
40	Incomplete packets	Number of incomplete packets
41	packet size	Packet size (B)
42	Classification	labels for the regular (-1) and anomalous (1) data.

4 Results and Discussion

This Section discusses and shows the results of applying the different Data Analysis methods mentioned in Section 3.2 to the datasets described in 0.

4.1 Level 1: Descriptive Analysis

4.1.1 Dataset layout

The pandas library was uploaded to the Jupyter notebook to visualize and check datasets layouts.

First, to find out the data types and the count of categorical and numerical columns in our datasets, the pandas dtypes() method is used to get the following results:

```
Data types for WannaCrypt Anomaly are:
int64       41
float64      1
dtype: int64


Data types for Nimda Anomaly are:
int64       41
float64      1
dtype: int64


Data types for Slammer Anomaly are:
int64       41
float64      1
dtype: int64


Data types for Moscow_blackout Anomaly are:
int64       41
float64      1
dtype: int64


Data types for Code_Red_I Anomaly are:
int64       41
float64      1
dtype: int64
```

These results show that all five datasets have the same structure and datatypes of numerical columns.

Next, our dataset's top and bottom five records were displayed using head() and tail() methods. The results for WannaCry Dataset are shown next to demonstrate the five datasets.

Table 7. Top 5 Rows of WannaCry Dataset-1

	H+M	H	M	S	NOA	NOW	NOANP	NOWNP	AAPL	MAPL	AUAPL	NODA	NODW	NOIW	AED	MED	IAT	MED1	MED2	MED3	MED4
0	0	0	0	0	1268	53	2592	239	6	20	6	898	6087	132	20	27	7	0	0	0	0
1	1	0	1	0	1147	51	2593	177	6	19	7	789	8776	110	19	22	8	0	0	0	0
2	2	0	2	0	796	70	1461	193	6	17	6	644	5028	223	17	18	7	0	0	0	0
3	3	0	3	0	647	46	1154	125	6	15	6	801	4157	94	15	13	7	0	0	0	0
4	4	0	4	0	880	52	1851	285	6	17	6	943	7951	520	17	18	7	0	0	0	0

Table 6. Top 5 Rows of WannaCry Dataset-2

MED5	MED6	MED7	MED8	MED9	MED10	MED11	MAL1	MAL2	MAL3	MAL4	MAL5	MAL6	MAL7	MAL8	MAL9	IGP packets	EGP packets	Incomplete packets	packet size	Classification
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1620	0	33	319	-1
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1181	0	132	300	-1
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	948	0	104	300	-1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	635	0	117	277	-1
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	873	0	199	307	-1

Table 5. Bottom 5 Rows of WannaCry Dataset-1

	H+M	H	M	S	NOA	NOW	NOANP	NOWNP	AAPL	MAPL	AUAPL	NODA	NODW	NOIW	AED	MED	IAT	MED1	MED2	MED3	MED4
11515	2355	23	55	0	739	48	1901	230	6	22	6	548	5521	230	22	16	7	0	0	0	0
11516	2356	23	56	0	713	66	1467	188	5	14	6	603	5074	214	14	17	6	0	0	0	1
11517	2357	23	57	0	946	81	3272	750	6	14	6	1011	13257	933	14	21	7	0	0	0	1
11518	2358	23	58	0	914	43	3014	237	6	15	6	600	10059	97	15	19	7	0	0	0	0
11519	2359	23	59	0	868	46	1921	102	6	20	6	378	4440	65	20	19	7	0	0	0	0

Table 4. Bottom 5 Rows of WannaCry Dataset-2

MED5	MED6	MED7	MED8	MED9	MED10	MED11	MAL1	MAL2	MAL3	MAL4	MAL5	MAL6	MAL7	MAL8	MAL9	IGP packets	EGP packets	Incomplete packets	packet size	Classification
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	911	0	48	305	-1
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	888	0	83	299	-1
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1134	0	46	333	-1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1056	0	48	322	-1
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1106	0	33	303	-1

The tables above illustrate that each dataset has a total no. of 42 columns. However, the rows count varies across our five datasets. The first four columns represent the time per minute for each BGP message exchanged between routers, and the remaining 38 columns represent BGP features described in section 3.3.

4.1.2 Missing Data Investigation

Missing Values are a common issue with datasets; investigating and handling these missing values is crucial before further analysis. For doing so, isnull().sum() methods were applied to address this issue. Results are shown in the table below:

Table 8. No. of Missing Values from the datasets.

Missing Data Counts	WannaCrypt	Nimda	Slammer	Moscow_blackout	Code_Red_I
H+M	0	0	0	0	0
H	0	0	0	0	0
M	0	0	0	0	0
S	0	0	0	0	0
NOA	0	0	0	0	0
NOW	0	0	0	0	0
NOANP	0	0	0	0	0
NOWNP	0	0	0	0	0
AAPL	0	0	0	0	0
MAPL	0	0	0	0	0
AUAPL	0	0	0	0	0
NODA	0	0	0	0	0
NODW	0	0	0	0	0
NOIW	0	0	0	0	0
AED	0	0	0	0	0
MED	0	0	0	0	0
IAT	0	0	0	0	0
MED1	0	0	0	0	0
MED2	0	0	0	0	0
MED3	0	0	0	0	0
MED4	0	0	0	0	0
MED5	0	0	0	0	0
MED6	0	0	0	0	0
MED7	0	0	0	0	0
MED8	0	0	0	0	0
MED9	0	0	0	0	0
MED10	0	0	0	0	0
MED11	0	0	0	0	0
MAL1	0	0	0	0	0
MAL2	0	0	0	0	0
MAL3	0	0	0	0	0
MAL4	0	0	0	0	0
MAL5	0	0	0	0	0
MAL6	0	0	0	0	0
MAL7	0	0	0	0	0
MAL8	0	0	0	0	0
MAL9	0	0	0	0	0
IGP packets	0	0	0	0	0
EGP packets	0	0	0	0	0
Incomplete packets	0	0	0	0	0
packet size	0	0	0	0	0
Classification	0	0	0	0	0
dtype: int64					

Since no missing values were detected in our five datasets, we can proceed with further analysis. Because features MED1-MED11 and MAL1-MAL9 were calculated based on MAL and MED by Al-Rousan and Trajković, (2012), they will be excluded from the subsequent analysis.

4.1.3 Descriptive Insights

Pandas DataFrame Describe() method was used to calculate statistical characteristics for the five datasets, such as the mean, standard deviation, and the percentiles. Results are shown below.

Table 9. Descriptive Data for WannaCry Dataset.

	NOA	NOW	NOANP	NOWNP	AAPL	MAPL	AUAPL	NODA	NODW	NOIW	AED	MED	IAT	IGP packets	EGP packets	Incomplete packets	packet size
count	11520	11520	11520	11520	11520	11520	11520	11520	11520	11520	11520	11520	11520	11520	11520	11520	11520
mean	957.1252	50.87969	2667.405	222.3499	6.007378	20.13481	6.095833	969.9092	7685.904	195.1491	20.13611	23.84803	6.746701	1147.24	0.727951	65.96233	314.4057
std	765.5633	15.84267	3420.785	1349.433	0.441801	9.228093	0.411042	907.3461	7495.694	411.3156	9.227637	29.01832	0.655459	754.6031	4.144614	86.03598	22.60828
min	304	18	417	31	4	10	4	82	919	6	10	7.8	4	404	0	11	255
25%	648	42	1322	103	6	16	6	625	4262	73	16	15	6	833	0	40	301
50%	791	49	1820	144	6	18	6	854	5828.5	116	18	18	7	984	0	53	310
75%	1013	58	2758	211	6	22	6	1089	8493.25	198	22	22	7	1221.25	0	71	322
max	16760	732	76993	101180	10	154	9	24275	200955	17142	154	520	10	17035	137	4327	651

WannaCry Dataset has 11520 data points. AAPL, AUAPL, and IAT have a slight standard deviation from the mean. The remaining features have higher standard deviations, indicating wider data spreads.

Table 10. Descriptive Data for Nimda Dataset.

	NOA	NOW	NOANP	NOWNP	AAPL	MAPL	AUAPL	NODA	NODW	NOIW	AED	MED	IAT	IGP packets	EGP packets	Incomplete packets	packet size
count	8609	8609	8609	8609	8609	8609	8609	8609	8609	8609	8609	8609	8609	8609	8609	8609	8609
mean	127.4205	6.381926	361.3479	83.3077	6.054826	12.76501	6.064932	9.652457	168.9671	6.710652	12.76838	4.795923	6.291091	113.1987	0.13091	14.09084	258.7894
std	523.0959	12.55541	1583.485	1537.644	0.556948	4.058781	0.551138	35.11279	201.0903	16.22644	4.058502	29.84849	0.772624	482.7031	1.182836	40.19764	41.60616
min	10	0	13	0	4	5	4	0	1	0	5	0.2	3	10	0	0	198
25%	61	5	122	19	6	11	6	1	65	1	11	1.1	6	52	0	6	238
50%	85	6	203	32	6	12	6	4	115	3	12	1.5	6	74	0	11	252
75%	126	7	360	56	6	14	6	9	205	7	14	2.2	7	111	0	17	270
max	24122	675	78495	82739	12	107	12	1476	5471	650	107	800	15	22285	48	1807	1455

Nimda Dataset has fewer BGP records than WannaCry, with only 8609 rows. Standard deviations values vary across these seventeen features, as shown in the above table.

Table 11. Descriptive Data for Slammer Dataset.

	NOA	NOW	NOANP	NOWNP	AAPL	MAPL	AUAPL	NODA	NODW	NOIW	AED	MED	IAT	IGP packets	EGP packets	Incomplete packets	packet size
count	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200
mean	94.95347	4.9	274.1017	56.72542	6.252361	12.47986	6.245833	11.93264	136.1463	9.515139	12.49944	4.232833	6.243889	87.39681	0.158472	7.398194	255.9406
std	179.543	1.984101	587.4939	329.2427	0.617662	3.173264	0.621966	27.8664	244.9844	20.74425	3.170131	25.31366	0.824824	168.3496	0.772332	12.02555	44.40558
min	5	0	6	0	4	5	4	0	0	0	5	0.1	3	2	0	0	189
25%	37	4	74	11	6	10	6	1	33	1	10	0.6	6	33	0	2	232
50%	52	5	124	21	6	12	6	3.5	63	3	12	0.9	6	47	0	4	249
75%	81	6	242	45	7	15	7	10	129	8	15	1.4	7	74	0	8	271
max	4584	69	14830	15993	10	29	10	385	3278	342	29	800	11	4317	12	269	2565

Slammer Dataset has 7200 records. The above table shows that standard deviations and other descriptive measures vary across BGP features.

Table 12. Descriptive data for Moscow_blackout Dataset.

	NOA	NOW	NOANP	NOWNP	AAPL	MAPL	AUAPL	NODA	NODW	NOIW	AED	MED	IAT	IGP packets	EGP packets	Incomplete packets	packet size
count	7199	7199	7199	7199	7199	7199	7199	7199	7199	7199	7199	7199	7199	7199	7199	7199	7199
mean	202.7359	8.453118	748.8811	72.14169	6.027782	13.38728	6.077372	336.3342	265.5128	39.17961	13.4045	6.889804	6.251563	198.5384	0.251424	5.805806	251.8272
std	737.9139	2.673383	3630.482	406.5333	0.814945	4.512005	0.849456	1855.766	466.5511	171.3477	4.51465	33.50234	1.215889	725.2316	1.176842	22.22752	43.2197
min	12	0	15	0	4	5	4	4	2	0	5	0.2	2	12	0	0	190
25%	57	7	106	16	6	10	6	37	83	8	10	1.1	6	56	0	0	230
50%	77	8	162	29	6	13	6	59	145	18	13	1.4	6	75	0	2	243
75%	112	10	287	48	6	15	7	106	273	31	15	2.1	7	109	0	5	264
max	15607	37	84500	13591	11	42	12	47382	8253	6824	42	850	14	14918	25	936	1736

Moscow_blackout dataset has a similar number of records to the previously investigated Slammer Dataset (7199 data points).

Table 13. Descriptive Data for Code_Red_I Dataset

	NOA	NOW	NOANP	NOWNP	AAPL	MAPL	AUAPL	NODA	NODW	NOIW	AED	MED	IAT	IGP packets	EGP packets	Incomplete packets	packet size
count	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200	7200
mean	97.09667	5.875556	281.1085	54.40458	6.170694	12.29167	6.169722	9.511944	152.9006	5.995278	12.30056	4.022528	6.430694	85.03611	0.124167	11.93639	257.37
std	352.8306	3.647789	1041.582	411.7199	0.637009	2.664233	0.621935	44.15499	286.709	18.21359	2.661836	23.21934	0.825303	325.0795	0.820541	28.19621	54.4336
min	12	0	15	0	4	5	5	0	1	0	5	0.2	3	11	0	0	195
25%	49	5	91	15	6	10	6	1	46	1	10	0.9	6	42	0	5	233
50%	67	6	143	25	6	12	6	3	81	2	12	1.2	6	57	0	9	248
75%	94	7	253	47	6	14	6	8	155	6	14	1.7	7	81	0	14	268
max	22261	243	63283	29446	10	25	9	2724	11507	590	25	700	10	20454	35	1777	2382

Code_Red_I dataset also has the same number of records as Moscow_blackout and Slammer datasets. The above table shows that standard deviations and other descriptive measures for Code_Red_I vary across the features.

To investigate whether these datasets are balanced or imbalanced, row counts grouped by the classification are obtained.

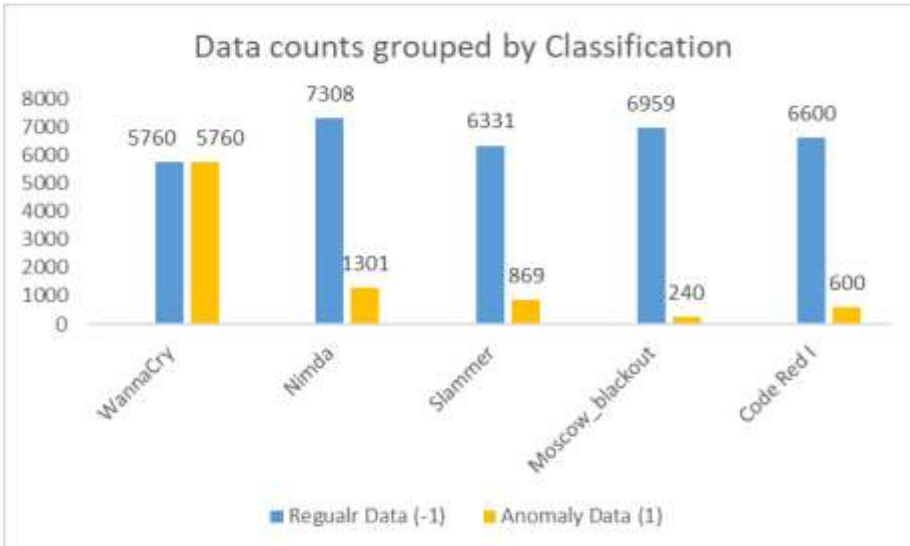


Figure 4. Imbalance of Datasets.

Results shown in Figure 4 demonstrate that only Moscow Dataset is balanced. The remaining datasets are imbalanced in which the irregular class has fewer records than the regular class.

For a better understanding of the distribution of BGP features and a more precise visualization of frequency accumulation, values positions, skewness, and kurtosis, histograms were plotted for all five datasets.

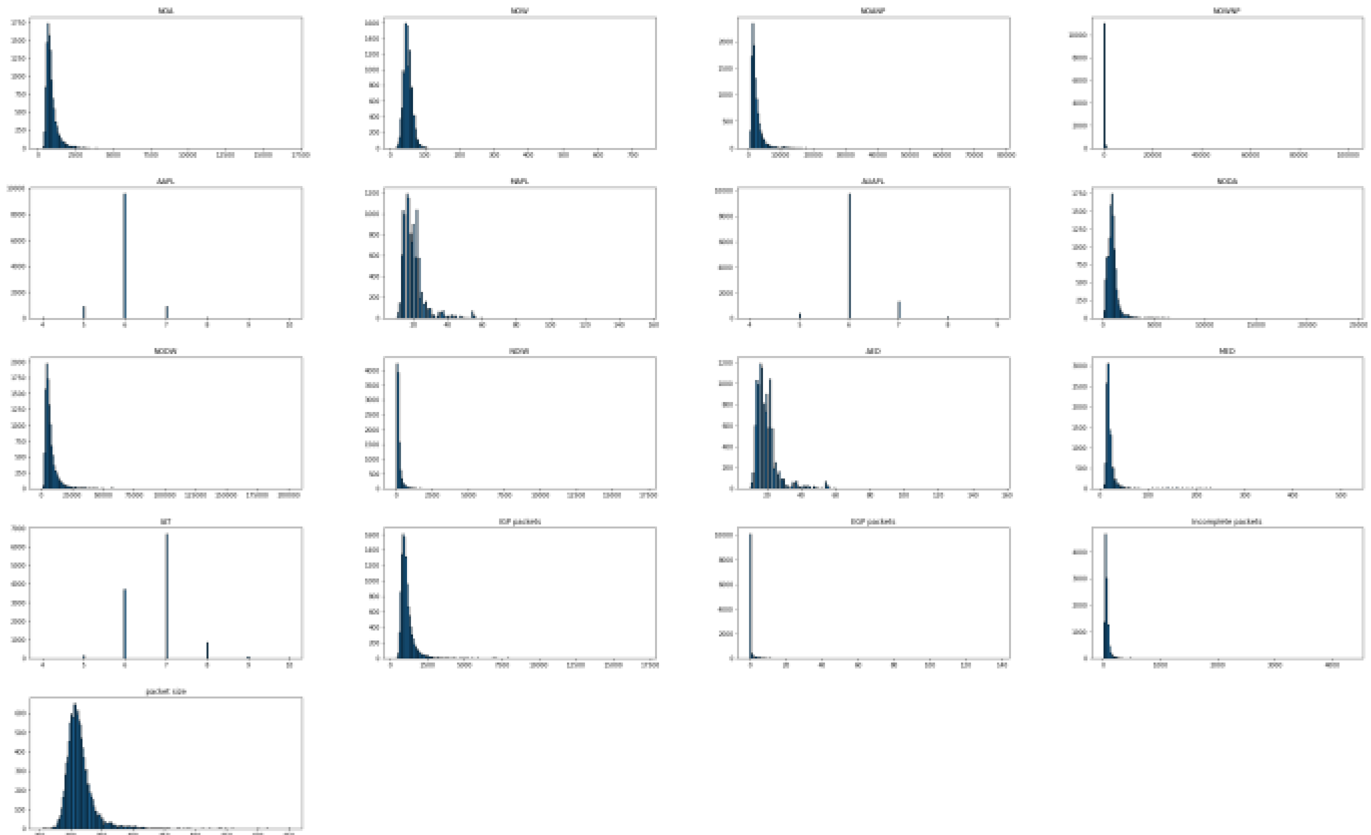


Figure 5. Histograms of features in WannaCry Dataset.

In WannaCry Dataset, BGP features show a distribution with one peak representing unimodal data. No Significant outliers are observed. Except for AAPL, IAT, and AUAPL, distributions for all remaining features are positively skewed.

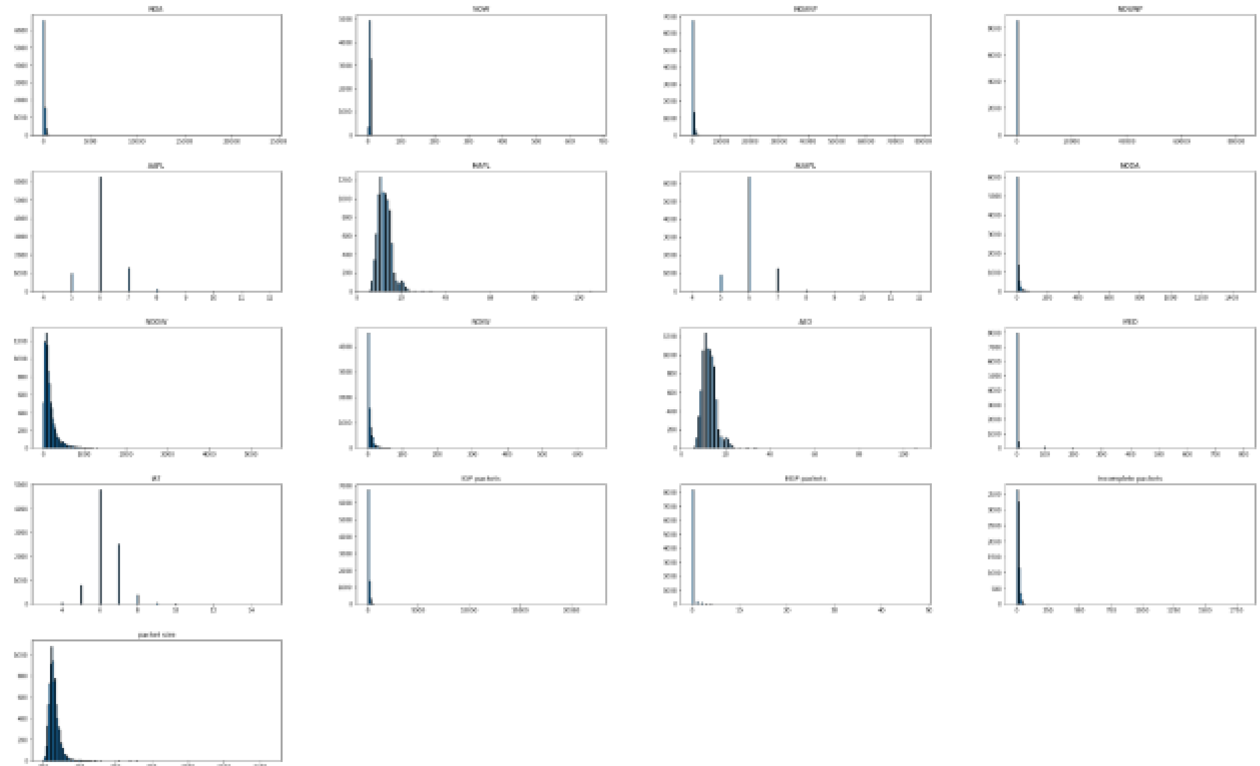


Figure 6. Histograms of features in Nimda Dataset.

In Nimda Dataset, BGP features represent unimodal data with no observed outliers. NOWNP shows the mode. Except for AAPL, IAT, and AUAPL, distribution fits for all remaining features are positively skewed.

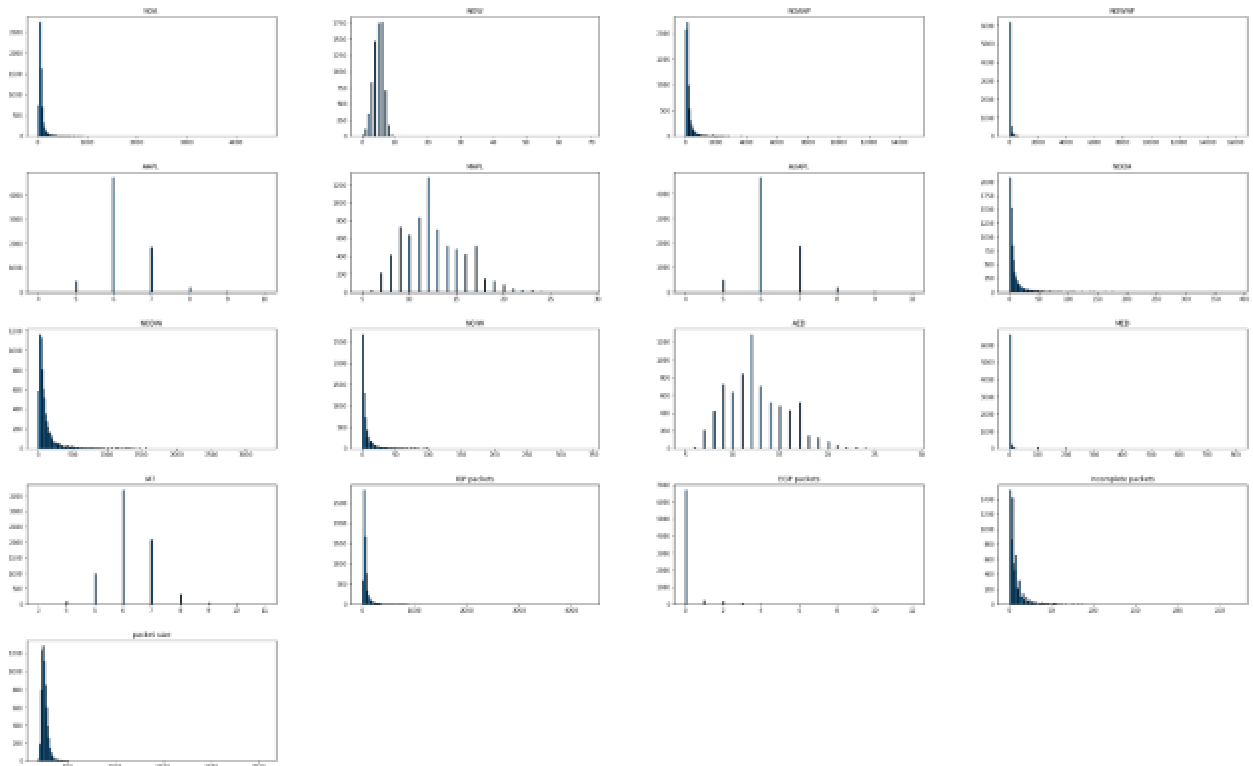


Figure 7. Histograms of features in Slammer Dataset.

In Slammer, BGP features are unimodal with only one peak and no outliers. MAPL and AED have slight positive skewness, and except for AAPL, IAT, and AUAPL, the skewness for all remaining features varies from moderate to high positive skewness.

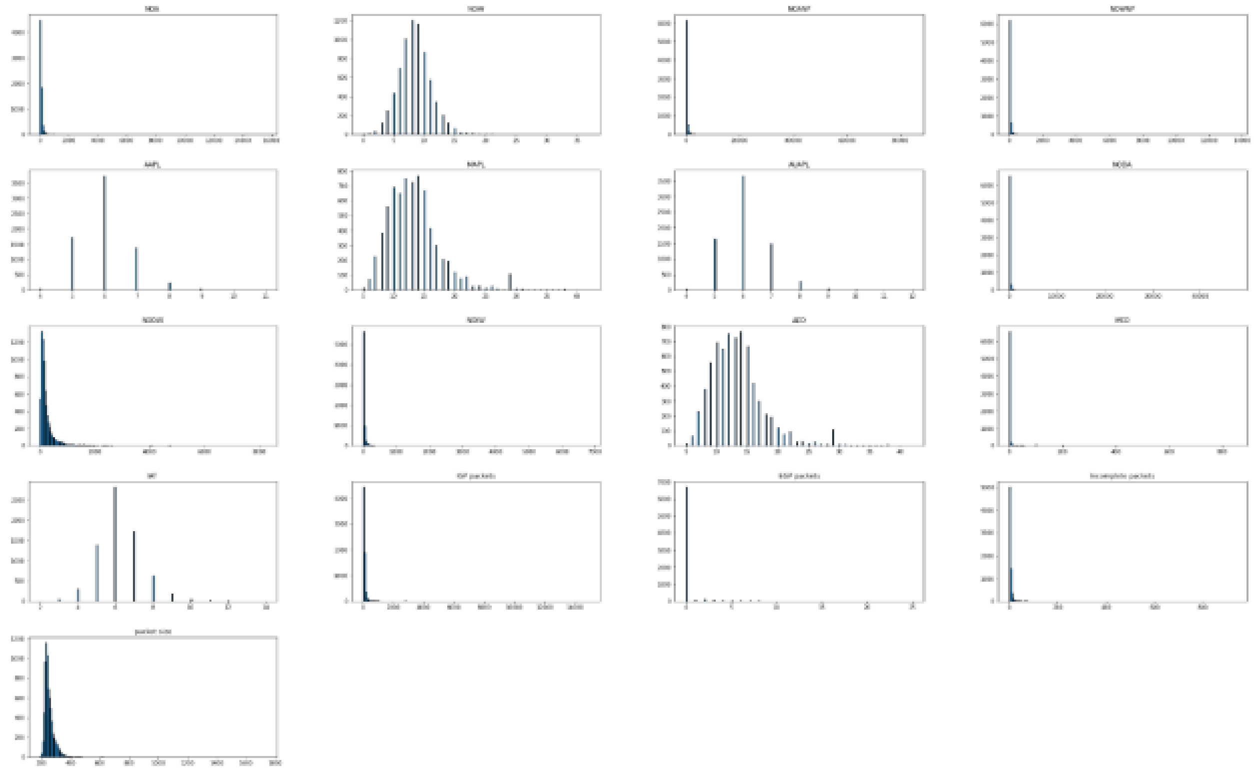


Figure 8. Histograms of features in Moscow blackout Dataset.

BGP features in the Moscow blackout are unimodal data with only one peak, and no outliers were observed. Features distribution spread varies; some features have narrow spreads, and others have wider spreads.

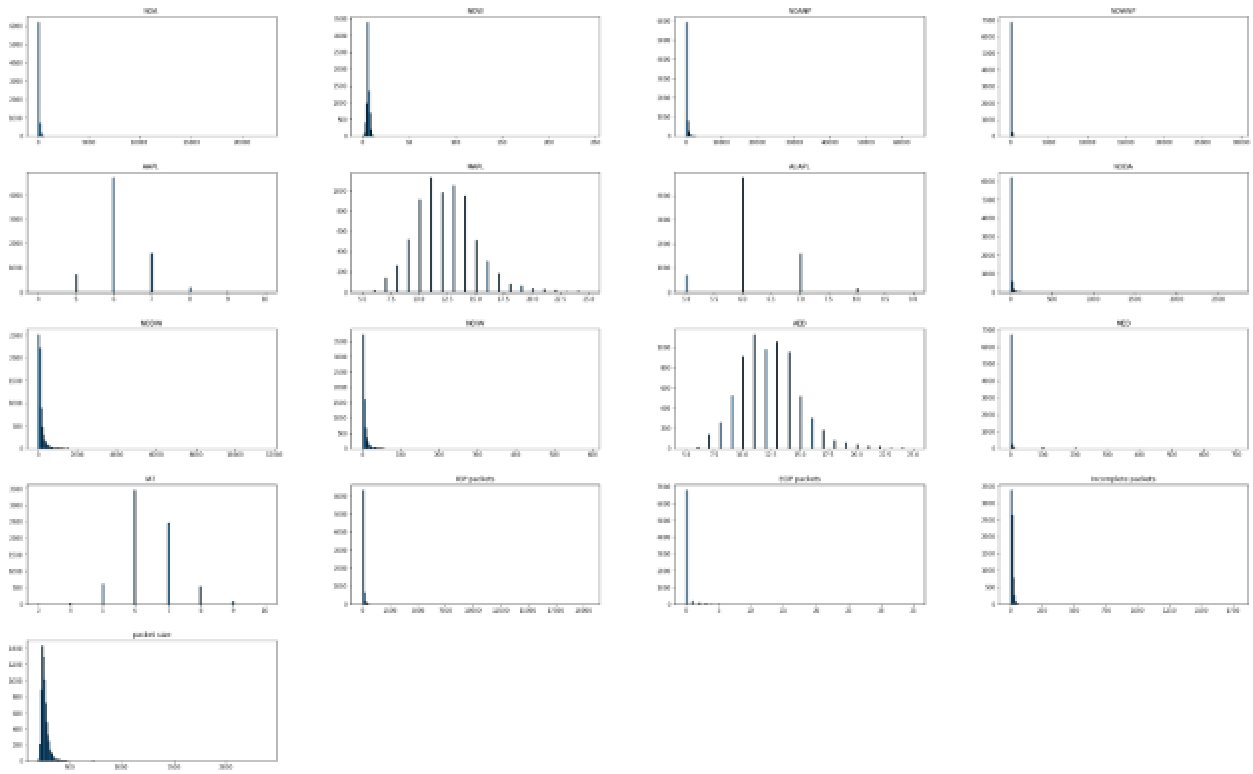


Figure 9. Histograms of features in Code_Red_I Dataset.

In Code_Red_I Anomaly, which has 7200 records, BGP features are unimodal with only one observed peak and no outliers. The features distribution spread varies; some features have narrow spreads, and others have wider spreads.

To give a more accurate understanding of data distributions compared to a normal distribution, statistical values for skewness and Kurtosis were computed Using skew() and Kurt() methods for pandas DataFrame. The results are shown in Table 14 and Table 15.

Table 14. Skewness of features in all 5 datasets.

Skewness of the features	WannaCrypt	Nimda	Slammer	Moscow_blackout	Code_Red_I
NOA	8.080534	34.30332	10.8026	10.013465	46.643827
NOW	13.62299	44.86883	12.89653	0.60392	43.737898
NOANP	7.891836	33.90088	10.37446	11.434288	39.665336
NOWNP	70.061616	45.69574	43.53205	19.406611	57.861797
AAPL	0.428155	0.642601	0.707574	0.625314	0.550057
MAPL	6.102853	10.06035	0.528055	1.463638	0.644212
AUAPL	1.455717	0.744978	0.62669	0.774486	0.525006
NODA	11.311492	27.91385	5.773591	12.838751	39.974538
NODW	8.133619	8.004618	5.29278	7.580668	13.617159
NOIW	16.671388	20.82186	5.258298	23.160514	15.218224
AED	6.103414	10.06118	0.52974	1.466225	0.645758
MED	7.213487	14.54075	17.15838	11.692974	15.329775
IAT	0.321239	0.57954	0.33131	0.698671	0.253641
IGP packets	7.748035	34.28587	10.90387	10.060817	46.426307
EGP packets	13.040842	26.7648	7.615124	7.278822	21.408347
Incomplete packets	25.852429	33.02528	7.773976	19.819381	44.629457
packet size	2.945426	9.115799	20.93263	13.229372	18.330276

Table 15. Kurtosis of features in all 5 datasets.

Kurtosis of the features	WannaCrypt	Nimda	Slammer	Moscow_blackout	Code_Red_I
NOA	97.46597	1352.42	200.7345	131.247929	2625.663768
NOW	540.002444	2137.524	410.3566	2.939472	2645.557215
NOANP	96.96194	1360.828	183.15	171.013607	2128.189146
NOWNP	5160.48867	2188.768	2049.426	450.256027	3847.091641
AAPL	5.789632	3.62749	1.575054	1.044257	1.260472
MAPL	61.82912	218.753	0.050524	3.785863	1.186462
AUAPL	5.97368	4.394896	1.447613	1.686194	1.072042
NODA	209.529828	1068.552	45.29213	224.322555	2204.366565
NODW	124.288627	143.2432	38.02377	82.605177	392.180322
NOIW	451.620656	710.7501	41.10148	698.98775	337.232761
AED	61.83844	218.781	0.05368	3.778531	1.192144
MED	65.272294	265.7447	420.209	191.4266	322.689287
IAT	1.173705	3.677377	1.202379	2.09663	0.915373
IGP packets	90.173756	1352.637	203.576	132.192492	2603.537385
EGP packets	249.175869	915.9844	72.8335	77.425522	744.17892
Incomplete packets	1090.98265	1269.211	117.4097	609.263238	2528.437762
packet size	20.312422	185.017	1024.856	372.534991	636.581297

From the previous histograms and Skewness and Kurtosis tables, the following summary can describe the key characteristics of features in all five datasets:

- **Multiple modes and outliers:** Almost all features in the five datasets represent unimodal data showing one peak in the distribution. No significant outliers have been observed from the histogram's visualizations.
- **Spread and Centers**: WannaCry has the widest distribution spread for almost all features among all datasets. Nevertheless, determining whether the features mean in different datasets are significantly distinct is hard to be concluded from histograms visualization; hence this will be investigated in the following inferential analysis section.
- **Distribution Fit:** Most features' distribution fit is positively skewed, varying from moderate to high skewness. The following features are exceptions with reasonably symmetrical skewness: AAPL and IAT in all datasets, MAPL and AED in Slammer and Code Red, NOW in Moscow, and AUAPL in Code Red.

- **Kurtosis:** MAPL and AED in Slammer can be said to have nearly zero kurtosis, indicating its distribution is following a normal distribution (mesokurtotic). However, the remaining features have a positive kurtosis, meaning their peaks are higher, and their tails are fatter than a normal distribution (leptokurtotic).

To further understand the behavior of BGP features during regular times and anomaly events, the following line graphs were plotted to show features variations over the collection period of different anomalies.

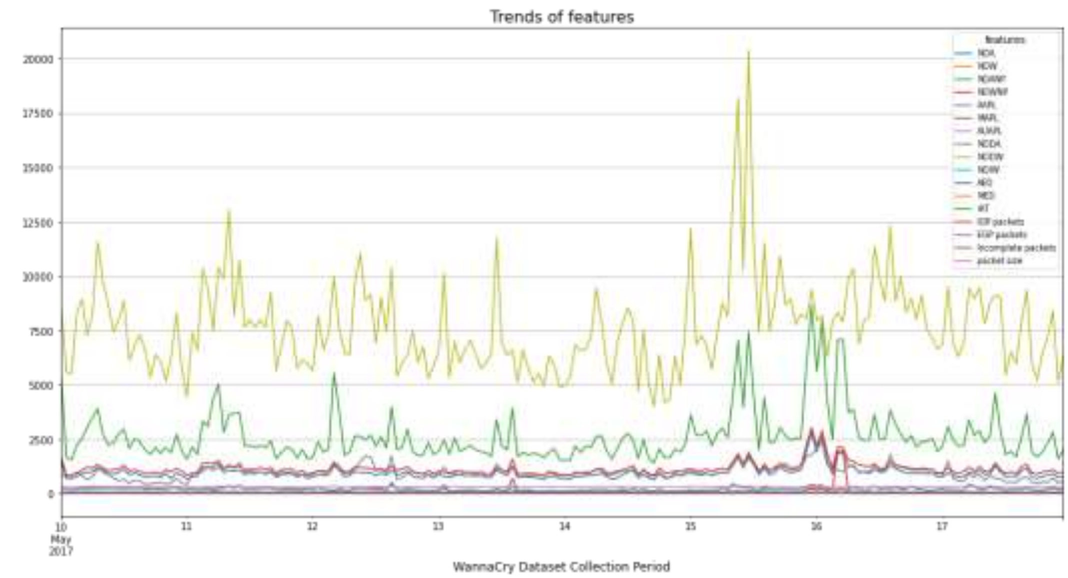


Figure 10. BGP Features variation over the collection period of WannaCry anomaly.

The WannaCry attack took place on 12 May 2017. The attack lasted for four days until 16 May 2017, and BGP records for this anomaly include the attack period and data for two days before and two days after the event (total records for eight days). Multiple BGP features showed a significant increase only on the last day of the attack, particularly between 15 and 16 May 2017. However, no significant difference is observed between the regular and the anomaly traffic for the remaining attack period.

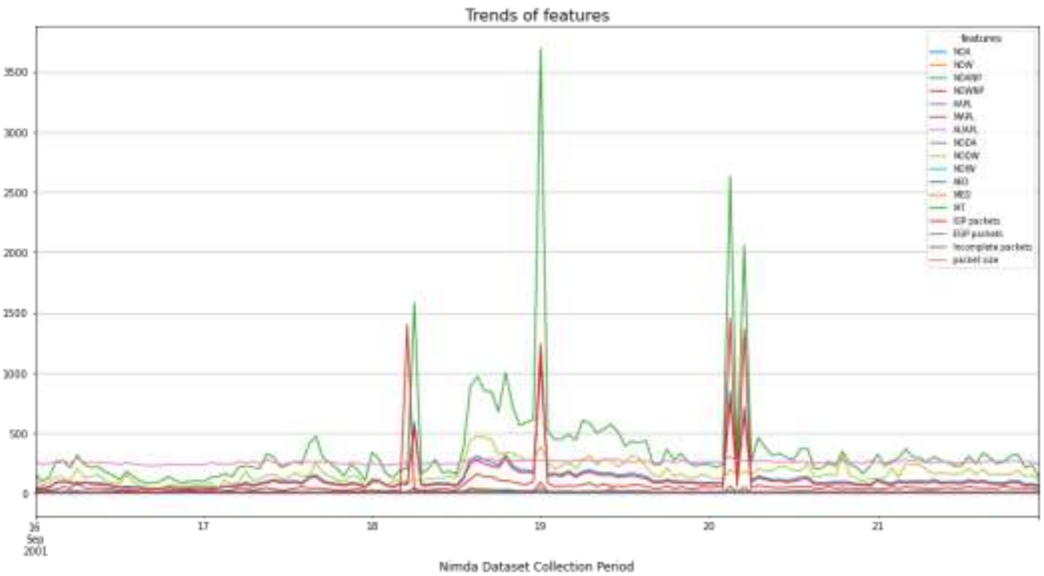


Figure 11. BGP Features variation over the collection period of Nimda anomaly.

Nimda worm attack took place on 18 September 2001. It lasted for two days until 20 September 2001. The BGP anomaly dataset for this anomaly records the attack period and data for two days before and two days after the event (total records for six days). A significant increase in multiple BGP features was detected during the attack period, and BGP updates experienced abnormal behavior compared to regular traffic.

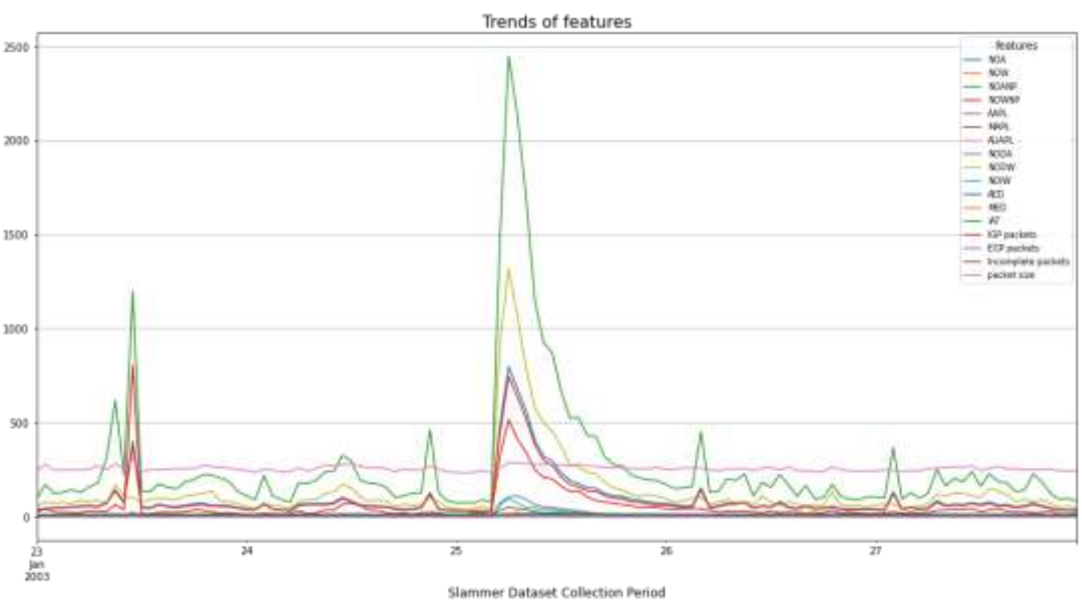


Figure 12. BGP Features variation over the collection period of Slammer anomaly.

The Slammer worm attack occurred on 25 June 2003 and lasted only for one day. The BGP records in this dataset include the attack period and data for two days before and two days after the anomaly (total BGP updates for five days). On 25 of June and during the attack, BGP message updates witnessed tremendous instability. Significant increments in several features can be observed in the above figure.

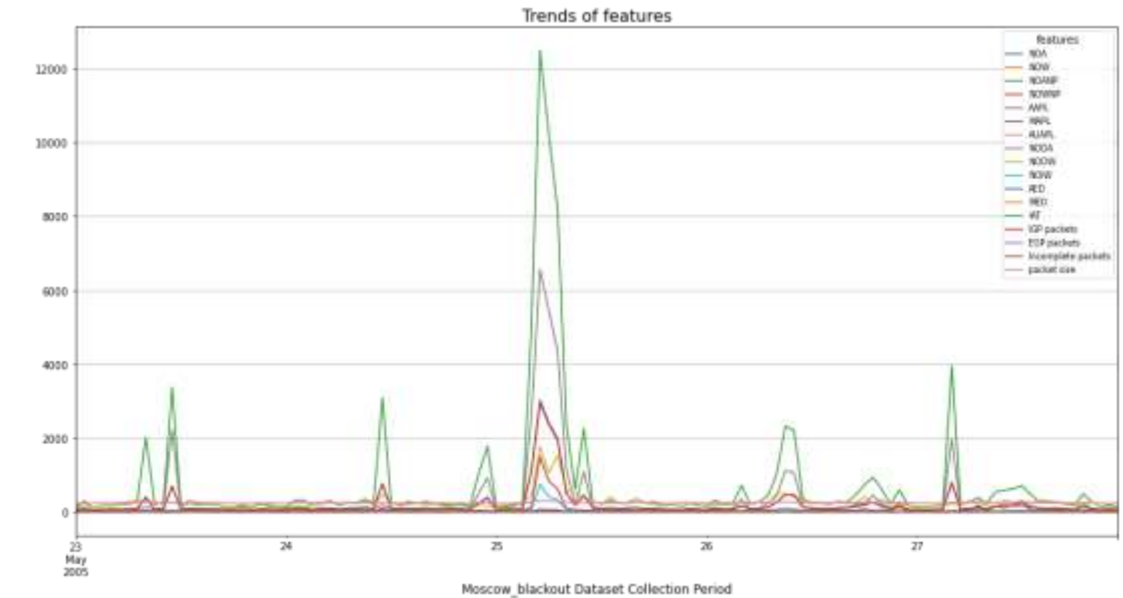


Figure 13.BGP Features variation over the collection period of Moscow_Blackout anomaly

Moscow_blackout happened on 25 May 2005 and lasted for one day. The BGP records in this dataset include data during the blackout period and data for two days before and two days after the Moscow power blackout (total BGP updates for five days). On 25 May and during the power blackout, BGP message updates witnessed enormous instability. Significant increments in several features can be seen in the above figure.

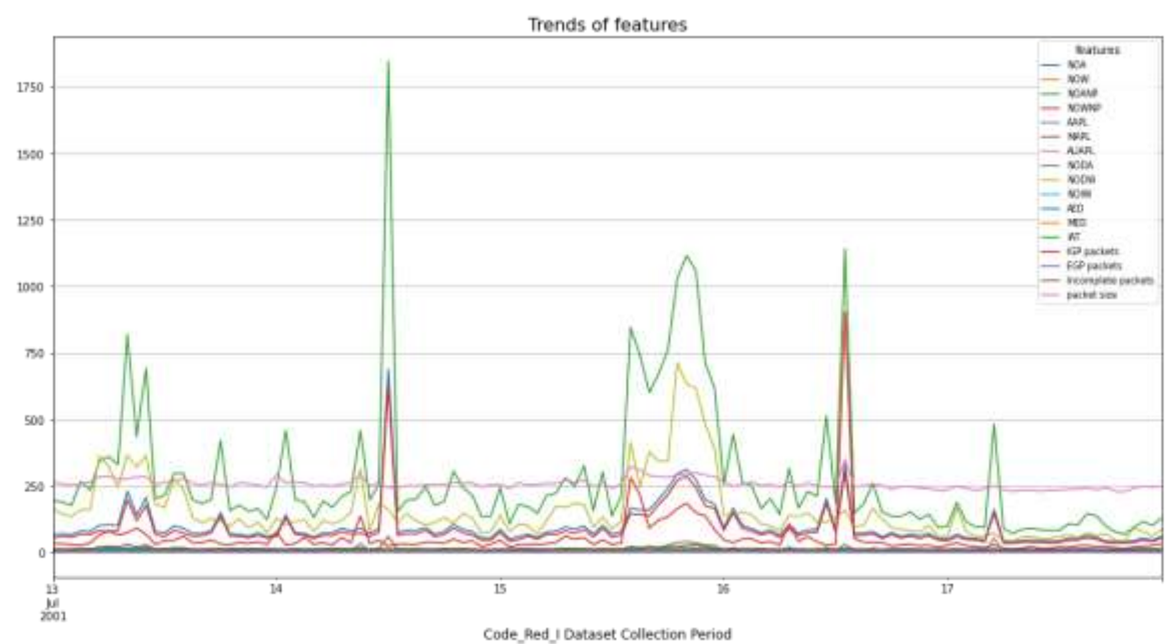


Figure 14. BGP Features variation over the collection period of Code Red anomaly.

Code Red worm attack occurred on 15 July 2001 and lasted for one day. The BGP records in this dataset include data during the attack period and data for two days before and two days after the attack (total BGP updates for five days). An observable odd behavior for BGP features can be noticed during the anomaly period from the previous figure. However, other spikes that happened during regular times are evident.

The following points can be concluded from the previous line graphs:

- Despite the evidence of some spikes in regular times, BGP features in all five datasets witnessed a noticeable change in multiple features during the anomaly that doesn't follow normal BGP behavior and is hence defined as an anomaly.
- No. of Announcements and Withdrawals significantly increased during all types of anomalies. However, announcements and withdrawals are not only sufficient to detect the abnormalities, and more features are required to efficiently detect the anomalies (Fonseca et al., 2019).
- No. of IGP packets also witnessed an increment during all anomalies. This indicates BGP is experiencing instabilities to announce trusted routes; thus, routing announcements from other routing protocols arose.

4.2 Level 2: Inferential Analysis

4.2.1 One-way ANOVA test

To answer the question: Does BGP behave similarly in different anomaly events? One-way ANOVA tests were performed for the mean values of BGP features in all five datasets to study whether there is a significant difference in the mean values regarding the different events. Our Null Hypothesis (H_0) is as follows:

- Null Hypothesis (H_0): All the means of all the features in different internet anomaly events are equal.
- Alternative Hypothesis (H_a): At least one of the means is not equal to the other.

Working with the one-tail ANOVA test and confidence level of (95%), alpha value, α = 0.05, the following results were obtained.

```
One-way ANOVA testing for NOA
F Value=637.759, P Value=0
The null hypothesis can be rejected

One-way ANOVA testing for NOW
F Value=5961.26, P Value=0
The null hypothesis can be rejected

One-way ANOVA testing for NOANP
F Value=435.992, P Value=0
The null hypothesis can be rejected

One-way ANOVA testing for NOWNP
F Value=35.2448, P Value=3.0045e-29
The null hypothesis can be rejected

One-way ANOVA testing for AAPL
F Value=196.178, P Value=2.2921e-161
The null hypothesis can be rejected

One-way ANOVA testing for MAPL
F Value=199.848, P Value=2.78181e-164
The null hypothesis can be rejected

One-way ANOVA testing for AUAPL
F Value=216.058, P Value=4.12437e-177
The null hypothesis can be rejected

One-way ANOVA testing for NODA
F Value=1245.67, P Value=0
The null hypothesis can be rejected

One-way ANOVA testing for NODW
F Value=620.564, P Value=0
The null hypothesis can be rejected

One-way ANOVA testing for NOIW
F Value=102.717, P Value=1.27119e-85
The null hypothesis can be rejected

One-way ANOVA testing for AED
F Value=199.671, P Value=3.84729e-164
The null hypothesis can be rejected

One-way ANOVA testing for MED
F Value=94.4524, P Value=8.68831e-79
The null hypothesis can be rejected

One-way ANOVA testing for IAT
F Value=593.535, P Value=0
The null hypothesis can be rejected

One-way ANOVA testing for IGP packets
F Value=939.76, P Value=0
The null hypothesis can be rejected

One-way ANOVA testing for EGP packets
F Value=7.62177, P Value=3.98631e-06
The null hypothesis can be rejected

One-way ANOVA testing for Incomplete packets
F Value=274.524, P Value=1.08806e-222
The null hypothesis can be rejected

One-way ANOVA testing for packet size
F Value=706.383, P Value=0
The null hypothesis can be rejected
```

Figure 15. One-way ANOVA Testing for BGP features in all 5 datasets

BGP Feature	P-value
NOA	0
NOW	0
NOANP	0
NOWNP	3E-29
AAPL	2.3E-161
MAPL	2.8E-164
AUAPL	4.1E-177
NODA	0
NODW	0
NOIW	1.27E-85
AED	3.8E-164
MED	8.69E-79
IAT	0
IGP packets	0
EGP packets	3.99E-06
Incomplete packets	1.1E-222
packet size	0

Table 16. P-Values for One-way ANOVA testing

All P-values reject the Null Hypothesis.

We can reject the null hypothesis for all features in all datasets, and it can be concluded that for each fe ature, at least the mean values in one event are significantly different from the remaining events.

4.2.2 Two-Sample T-test

After Rejecting the Null Hypothesis of the ANOVA test, it's essential to conduct two-sample t-tests for features in every two datasets to have more profound knowledge about whether BGP tends to have the same behavior in different events. Working with a two-tail t-test and confidence level of (95%), alpha value, α = 0.05, Results are shown next.

1st T-test: WannaCry and Nimda

- Null Hypothesis (H_0): Features mean in WannaCry Anomaly = Features mean in Nimda Anomaly
- Alternative Hypothesis (H_a): Features mean in WannaCry Anomaly ≠ Features mean in Nimda Anomaly

```
Two-way T-test for fetures in WannaCrypt and Nimda
p value=4.39864e-199, t value=31.0968
The null hypothesis for NOA can be rejected

p value=0, t value=101.798
The null hypothesis for NOW can be rejected

p value=1.69406e-81, t value=19.3718
The null hypothesis for NOANP can be rejected

p value=0.123339, t value=1.54109
The null hypothesis for NOWNP is accepted

p value=3.2603e-07, t value=-5.11265
The null hypothesis for AAPL can be rejected

p value=1.13934e-76, t value=18.7604
The null hypothesis for MAPL can be rejected

p value=0.771527, t value=-0.290389
The null hypothesis for AUAPL is accepted

p value=0, t value=43.0551
The null hypothesis for NODA can be rejected

p value=1.98438e-228, t value=33.4963
The null hypothesis for NODW can be rejected

p value=7.60457e-40, t value=13.2933
The null hypothesis for NOIW can be rejected

p value=1.09838e-76, t value=18.7624
The null hypothesis for AED can be rejected

p value=2.50421e-35, t value=12.4717
The null hypothesis for MED can be rejected

p value=1.91463e-99, t value=21.5084
The null hypothesis for IAT can be rejected

p value=0, t value=42.1709
The null hypothesis for IGP packets can be rejected

p value=1.07676e-05, t value=4.40433
The null hypothesis for EGP packets can be rejected

p value=1.67059e-109, t value=22.6281
The null hypothesis for Incomplete packets can be rejected

p value=0, t value=49.7221
The null hypothesis for packet size can be rejected
```

Figure 16. Two-sample t-test for WannaCry and Nimda.

BGP Features	P-Value
NOA	4.4E-199
NOW	0
NOANP	1.69E-81
NOWNP	0.123339*
AAPL	3.26E-07
MAPL	1.14E-76
AUAPL	0.771527*
NODA	0
NODW	2E-228
NOIW	7.6E-40
AED	1.1E-76
MED	2.5E-35
IAT	1.9E-99
IGP packets	0
EGP packets	1.08E-05
Incomplete packets	1.7E-109
packet size	0

Table 17. P-Values for Two-sample t-test for WannaCry and Nimda

The asterisk (*) indicates the P-value accepts the Null Hypothesis

Although WannaCry and Nimda are both cyber worm attacks, mean values for NOWNP and AUAPL fall in the area of accepting H_0. We can reject the H_0 for all the rest.

2^{nd} T-test: WannaCry and Slammer

- Null Hypothesis (H_0): Features mean in WannaCry Anomaly = Features mean in Slammer Anomaly
- Alternative Hypothesis (H_a): Features mean in WannaCry Anomaly ≠ Features mean in Slammer Anomaly.

```
Two-way T-test for fetures in WannaCrypt and Slammer
p value=6.32998e-140, t value=25.7975
The null hypothesis for NOA can be rejected

p value=0, t value=95.3185
The null hypothesis for NOW can be rejected

p value=3.76693e-51, t value=15.1742
The null hypothesis for NOANP can be rejected

p value=0.00640074, t value=-2.72738
The null hypothesis for NOWNP can be rejected

p value=2.18162e-40, t value=-13.3943
The null hypothesis for AAPL can be rejected

p value=1.33325e-36, t value=12.7132
The null hypothesis for MAPL can be rejected

p value=2.56905e-30, t value=-11.4995
The null hypothesis for AUAPL can be rejected

p value=1.28598e-235, t value=34.1519
The null hypothesis for NODA can be rejected

p value=3.22125e-148, t value=26.6007
The null hypothesis for NODW can be rejected

p value=2.46704e-19, t value=9.01806
The null hypothesis for NOIW can be rejected

p value=1.3709e-36, t value=12.711
The null hypothesis for AED can be rejected

p value=4.8931e-32, t value=11.8434
The null hypothesis for MED can be rejected

p value=5.48702e-31, t value=11.6347
The null hypothesis for IAT can be rejected

p value=6.7321e-254, t value=35.5795
The null hypothesis for IGP packets can be rejected

p value=0.223529, t value=1.21731
The null hypothesis for EGP packets is accepted

p value=5.18567e-91, t value=20.5486
The null hypothesis for Incomplete packets can be rejected

p value=0, t value=46.3666
The null hypothesis for packet size can be rejected
```

Figure 17.Two-sample t-test for WannaCry and Slammer.

BGP Features	P-Value
NOA	6.3E-140
NOW	0
NOANP	3.77E-51
NOWNP	0.006401
AAPL	2.18E-40
MAPL	1.33E-36
AUAPL	2.57E-30
NODA	1.3E-235
NODW	3.2E-148
NOIW	2.47E-19
AED	1.37E-36
MED	4.89E-32
IAT	5.49E-31
IGP packets	6.7E-254
EGP packets	0.223529*
Incomplete packets	5.19E-91
packet size	0

Table 18. P-Values for Two-sample t-test for WannaCry and Slammer

The asterisk (*) indicates the P-value accepts the Null Hypothesis

The only feature that accepts the H_0 between WannaCry and Slammer events is the no. of EGP packets. We can reject the H_0 for all the rest.

3rd T-test: WannaCry and Moscow blackout

- Null Hypothesis (H_0): Features mean in WannaCry Anomaly = Features mean in Moscow blackout Anomaly.
- Alternative Hypothesis (H_a): Features mean in WannaCry Anomaly ≠ Features mean in Moscow blackout Anomaly.

```
Two-way T-test for fetures in WannaCrypt and Moscow_blackout
p value=5.4502e-123, t value=-24.1439
The null hypothesis for NOA can be rejected

p value=0, t value=46.1146
The null hypothesis for NOW can be rejected

p value=3.7576e-157, t value=-27.5238
The null hypothesis for NOANP can be rejected

p value=4.23113e-80, t value=-19.2404
The null hypothesis for NOWNP can be rejected

p value=1.96254e-104, t value=22.1361
The null hypothesis for AAPL can be rejected

p value=2.08679e-09, t value=6.00013
The null hypothesis for MAPL can be rejected

p value=8.25284e-144, t value=26.2451
The null hypothesis for AUAPL can be rejected

p value=0, t value=-43.4097
The null hypothesis for NODA can be rejected

p value=4.96038e-36, t value=12.6155
The null hypothesis for NODW can be rejected

p value=4.01547e-12, t value=-6.95086
The null hypothesis for NOIW can be rejected

p value=2.06197e-09, t value=6.00208
The null hypothesis for AED can be rejected

p value=3.20324e-41, t value=-13.5496
The null hypothesis for MED can be rejected

p value=0, t value=42.839
The null hypothesis for IAT can be rejected

p value=1.39501e-84, t value=-19.8012
The null hypothesis for IGP packets can be rejected

p value=0.930147, t value=0.0876631
The null hypothesis for EGP packets is accepted

p value=7.76474e-07, t value=4.94643
The null hypothesis for Incomplete packets can be rejected

p value=1.68386e-50, t value=15.0857
The null hypothesis for packet size can be rejected
```

Figure 18. Two-sample t-test for WannaCry and Moscow Blackout.

BGP Features	P-Value
NOA	5.5E-123
NOW	0
NOANP	3.8E-157
NOWNP	4.23E-80
AAPL	2E-104
MAPL	2.09E-09
AUAPL	8.3E-144
NODA	0
NODW	4.96E-36
NOIW	4.02E-12
AED	2.06E-09
MED	3.2E-41
IAT	0
IGP packets	1.4E-84
EGP packets	0.930147*
Incomplete packets	7.76E-07
packet size	1.68E-50

Table 19. P-Values for Two-sample t-test for WannaCry and Moscow

The asterisk (*) indicates the P-value accepts the Null Hypothesis

In this test, the only feature that accepts the H_0 between WannaCry and Moscow Blackout events is the no. of EGP packets. We can reject the H_0 for all the rest.

4th T-test: WannaCry and Code Red I

- Null Hypothesis (H_0): Features mean in WannaCry Anomaly = Features mean in Code Red I Anomaly.
- Alternative Hypothesis (H_a): Features mean in WannaCry Anomaly ≠ Features mean in Code Red I Anomaly.

```
Two-way T-test for fetures in WannaCrypt and Code_Red_I
p value=4.98289e-140, t value=25.8343
The null hypothesis for NOA can be rejected

p value=0, t value=77.21
The null hypothesis for NOW can be rejected

p value=4.51586e-44, t value=14.0316
The null hypothesis for NOANP can be rejected

p value=0.0494736, t value=1.96486
The null hypothesis for NOWNP is accepted

p value=2.39192e-08, t value=-5.58802
The null hypothesis for AAPL can be rejected

p value=3.13087e-57, t value=16.1055
The null hypothesis for MAPL can be rejected

p value=0.00641026, t value=-2.72693
The null hypothesis for AUAPL can be rejected

p value=3.81297e-175, t value=29.123
The null hypothesis for NODA can be rejected

p value=2.11366e-106, t value=22.3304
The null hypothesis for NODW can be rejected

p value=2.1138e-18, t value=8.77781
The null hypothesis for NOIW can be rejected

p value=4.43189e-57, t value=16.0831
The null hypothesis for AED can be rejected

p value=1.53783e-21, t value=9.56687
The null hypothesis for MED can be rejected

p value=1.41553e-54, t value=15.7074
The null hypothesis for IAT can be rejected

p value=3.43121e-232, t value=33.9386
The null hypothesis for IGP packets can be rejected

P value=0.00432156, t value=2.85471
The null hypothesis for EGP packets can be rejected

p value=7.92437e-76, t value=18.6769
The null hypothesis for Incomplete packets can be rejected

p value=2.21431e-57, t value=16.1277
The null hypothesis for packet size can be rejected
```

Figure 19. Two-sample t-test for WannaCry and Code Red I.

BGP Features	P-Value
NOA	4.98E-140
NOW	0
NOANP	4.52E-44
NOWNP	0.0494736*
AAPL	2.39E-08
MAPL	3.13E-57
AUAPL	0.0064103
NODA	3.81E-175
NODW	2.11E-106
NOIW	2.11E-18
AED	4.43E-57
MED	1.54E-21
IAT	1.42E-54
IGP packets	3.43E-232
EGP packets	0.0043216
Incomplete packets	7.92E-76
packet size	2.21E-57

Table 20. P-Values for Two-sample t-test for WannaCry and Code Red

The asterisk (*) indicates the P-value accepts the Null Hypothesis

Although WannaCry and Code Red I are both cyber worm attacks, only mean values for No. of withdrawn NLRI prefixes (NOWNP) falls in the area of accepting H_0. We can reject the H_0 for all the rest.

5th T-test: Nimda and Slammer

- Null Hypothesis (H_0): Features mean in Nimda Anomaly = Features mean in Slammer Anomaly.
- Alternative Hypothesis (H_a): Features mean in Nimda Anomaly ≠ Features mean in Slammer Anomaly.

```
Two-way T-test for fetures in Nimda and Slammer
p value=0.00784274, t value=-2.66124
The null hypothesis for NOA can be rejected

p value=0.0806049, t value=1.748
The null hypothesis for NOW is accepted

p value=0.0367692, t value=-2.08961
The null hypothesis for NOANP is accepted

p value=0.238301, t value=-1.17957
The null hypothesis for NOWNP is accepted

p value=1.3396e-15, t value=-8.05104
The null hypothesis for AAPL can be rejected

p value=2.18107e-14, t value=-7.69208
The null hypothesis for MAPL can be rejected

p value=2.70605e-19, t value=-9.06528
The null hypothesis for AUAPL can be rejected

p value=8.15653e-34, t value=-12.3309
The null hypothesis for NODA can be rejected

p value=1.07124e-34, t value=-12.5049
The null hypothesis for NODW can be rejected

p value=5.2818e-103, t value=-22.7602
The null hypothesis for NOIW can be rejected

p value=1.97745e-14, t value=-7.70494
The null hypothesis for AED can be rejected

p value=0.990878, t value=0.011434
The null hypothesis for MED is accepted

p value=2.86005e-09, t value=-5.96437
The null hypothesis for IAT can be rejected

p value=0.00452592, t value=-2.84194
The null hypothesis for IGP packets can be rejected

p value=8.05936e-06, t value=-4.47439
The null hypothesis for EGP packets can be rejected

p value=0.721668, t value=-0.356277
The null hypothesis for Incomplete packets is accepted

p value=0.942493, t value=0.0721451
The null hypothesis for packet size is accepted
```

Figure 20. Two-sample t-test for Nimda and Slammer.

BGP Features	P-Value
NOA	0.0078427
NOW	0.0806049*
NOANP	0.0367692*
NOWNP	0.238301*
AAPL	1.34E-15
MAPL	2.18E-14
AUAPL	2.71E-19
NODA	8.16E-34
NODW	1.07E-34
NOIW	5.28E-103
AED	1.98E-14
MED	0.990878*
IAT	2.86E-09
IGP packets	0.0045259
EGP packets	8.06E-06
Incomplete packets	0.721668*
packet size	0.942493*

Table 21. P-Values for Two-sample t-test for Nimda and Slammer

The asterisk (*) indicates the P-value accepts the Null Hypothesis

Comparing BGP behavior during Nimda and Slammer, the following features accept H_0: NOW, NOANP, NOWNP, MED, No. of Incomplete packets, and packet size. The rest fails to accept H_0.

6th T-test: Nimda and Moscow blackout

- Null Hypothesis (H_0): Features mean in Nimda Anomaly = Features mean in Moscow blackout Anomaly.
- Alternative Hypothesis (H_a): Features mean in Nimda Anomaly ≠ Features mean in Moscow blackout Anomaly.

```
Two-way T-test for fetures in Nimda and Moscow_blackout
p value=8.51465e-118, t value=-25.2254
The null hypothesis for NOA can be rejected

p value=0.00659362, t value=-2.7204
The null hypothesis for NOW can be rejected

p value=2.58214e-130, t value=-26.8184
The null hypothesis for NOANP can be rejected

p value=2.46118e-07, t value=-5.18389
The null hypothesis for NOWNP can be rejected

p value=9.67482e-95, t value=22.1711
The null hypothesis for AAPL can be rejected

p value=2.06882e-10, t value=-6.39914
The null hypothesis for MAPL can be rejected

p value=4.79051e-94, t value=22.076
The null hypothesis for AUAPL can be rejected

p value=1.2465e-194, t value=-34.6108
The null hypothesis for NODA can be rejected

p value=1.76988e-84, t value=-20.7432
The null hypothesis for NODW can be rejected

p value=1.46124e-56, t value=-16.5195
The null hypothesis for NOIW can be rejected

p value=2.14303e-10, t value=-6.39361
The null hypothesis for AED can be rejected

p value=1.64442e-22, t value=-9.91638
The null hypothesis for MED can be rejected

p value=1.69229e-135, t value=27.4678
The null hypothesis for IAT can be rejected

p value=1.79935e-125, t value=-26.2065
The null hypothesis for IGP packets can be rejected

p value=0.000100721, t value=-3.89906
The null hypothesis for EGP packets can be rejected

p value=7.83065e-07, t value=-4.95999
The null hypothesis for Incomplete packets can be rejected

p value=1.79781e-10, t value=-6.42112
The null hypothesis for packet size can be rejected
```

Figure 21.Two-sample t-test for Nimda and Moscow Blackout.

BGP Features	P-Value
NOA	8.51E-118
NOW	0.0065936
NOANP	2.58E-130
NOWNP	2.46E-07
AAPL	9.67E-95
MAPL	2.07E-10
AUAPL	4.79E-94
NODA	1.25E-194
NODW	1.77E-84
NOIW	1.46E-56
AED	2.14E-10
MED	1.64E-22
IAT	1.69E-135
IGP packets	1.80E-125
EGP packets	0.0001007
Incomplete packets	7.83E-07
packet size	1.80E-10

Table 22. P-Values for Two-sample t-test for Nimda and Moscow

All P-values reject the Null Hypothesis.

Comparing Nimda and Moscow blackout events, All BGP features fall in the area of rejecting the Null Hypothesis.

7th T-test: Nimda and Code Red I

- Null Hypothesis (H_0): Features mean in Nimda Anomaly = Features mean in Code Red I Anomaly.
- Alternative Hypothesis (H_a): Features mean in Nimda Anomaly ≠ Features mean in Code Red I Anomaly.

```
Two-way T-test for fetures in Nimda and Code_Red_I
p value=0.262089, t value=1.1218
The null hypothesis for NOA is accepted

p value=0.821473, t value=-0.225683
The null hypothesis for NOW is accepted

p value=0.852812, t value=-0.185558
The null hypothesis for NOANP is accepted

p value=0.891106, t value=-0.136923
The null hypothesis for NOWNP is accepted

P value=0.0720483, t value=-1.79982
The null hypothesis for AAPL is accepted

p value=2.42878e-23, t value=10.0863
The null hypothesis for MAPL can be rejected

p value=0.0323574, t value=-2.14154
The null hypothesis for AUAPL is accepted

p value=0.0927043, t value=-1.68215
The null hypothesis for NODA is accepted

p value=9.40074e-12, t value=-6.85819
The null hypothesis for NODW can be rejected

p value=0.000195379, t value=-3.73223
The null hypothesis for NOIW can be rejected

p value=4.86129e-23, t value=10.0143
The null hypothesis for AED can be rejected

p value=0.979916, t value=0.0251773
The null hypothesis for MED is accepted

p value=0.829325, t value=0.215597
The null hypothesis for IAT is accepted

p value=0.257813, t value=1.13192
The null hypothesis for IGP packets is accepted

p value=0.690291, t value=-0.398521
The null hypothesis for EGP packets is accepted

p value=0.309529, t value=1.01648
The null hypothesis for Incomplete packets is accepted

p value=1.06887e-15, t value=-8.08812
The null hypothesis for packet size can be rejected
```

Figure 22.Two-sample t-test for Nimda and Code Red I.

BGP Features	P-Value
NOA	0.262089*
NOW	0.821473*
NOANP	0.852812*
NOWNP	0.891106*
AAPL	0.0720483*
MAPL	2.43E-23
AUAPL	0.0323574*
NODA	0.0927043*
NODW	9.40E-12
NOIW	0.0001954
AED	4.86E-23
MED	0.979916*
IAT	0.829325*
IGP packets	0.257813*
EGP packets	0.690291*
Incomplete packets	0.309529*
packet size	1.07E-15

Table 23. P-Values for Two-sample t-test for Nimda and Code Red

The asterisk (*) indicates the P-value accepts the Null Hypothesis

Comparing Nimda and Code Red I events, all the following BGP features fall in the area of accepting the Null Hypothesis: NOA, NOW, NOANP, NOWNP, AAPL, AUAPL, NODA, MED, IAT, No. of IGP packets, No. of EGP packets and No. of incomplete packets. This indicates the relevance of BGP behavior in Nimda and Code Red I.

8^{th} T-test: Slammer and Moscow Blackout

- Null Hypothesis (H_0): Features mean in Slammer Anomaly = Features mean in Moscow blackout Anomaly.
- Alternative Hypothesis (H_a): Features mean in Slammer Anomaly ≠ Features mean in Moscow blackout Anomaly.

```
Two-way T-test for fetures in Slammer and Moscow_blackout
p value=3.63627e-128, t value=-27.6203
The null hypothesis for NOA can be rejected

p value=5.04269e-128, t value=-27.6003
The null hypothesis for NOW can be rejected

p value=3.60778e-114, t value=-25.637
The null hypothesis for NOANP can be rejected

p value=9.30094e-22, t value=-9.78899
The null hypothesis for NOWNP can be rejected

p value=1.38976e-106, t value=24.5466
The null hypothesis for AAPL can be rejected

p value=0.0248039, t value=-2.24751
The null hypothesis for MAPL can be rejected

p value=3.80972e-101, t value=23.7563
The null hypothesis for AUAPL can be rejected

p value=1.36518e-131, t value=-28.1013
The null hypothesis for NODA can be rejected

p value=6.62316e-30, t value=-11.7019
The null hypothesis for NODW can be rejected

p value=1.83132e-33, t value=-12.4643
The null hypothesis for NOIW can be rejected

p value=0.025354, t value=-2.239
The null hypothesis for AED is accepted

p value=1.14288e-22, t value=-10.0172
The null hypothesis for MED can be rejected

p value=9.9032e-150, t value=30.6295
The null hypothesis for IAT can be rejected

p value=8.39678e-130, t value=-27.8503
The null hypothesis for IGP packets can be rejected

p value=0.210769, t value=-1.25218
The null hypothesis for EGP packets is accepted

p value=4.44543e-24, t value=-10.3624
The null hypothesis for Incomplete packets can be rejected

p value=5.17553e-18, t value=-8.79969
The null hypothesis for packet size can be rejected
```

Figure 23. Two-sample t-test for Slammer and Moscow Blackout.

BGP Features	P-Value
NOA	3.64E-128
NOW	5.04E-128
NOANP	3.61E-114
NOWNP	9.30E-22
AAPL	1.39E-106
MAPL	0.0248039
AUAPL	3.81E-101
NODA	1.37E-131
NODW	6.62E-30
NOIW	1.83E-33
AED	0.025354*
MED	1.14E-22
IAT	9.90E-150
IGP packets	8.40E-130
EGP packets	0.210769*
Incomplete packets	4.45E-24
packet size	5.18E-18

Table 24. P-Values for Two-sample t-test for Slammer and Moscow

The asterisk (*) indicates the P-value accepts the Null Hypothesis

For Slammer and Moscow blackout, only AED and No. of EGP packets features of BGP accept the Null Hypothesis, and we can reject the rest.

9th T-test: Slammer and Code Red I

- Null Hypothesis (H_0): Features mean in Slammer Anomaly = Features mean in Code Red I Anomaly.
- Alternative Hypothesis (H_a): Features mean in Slammer Anomaly ≠ Features mean in Code Red I Anomaly.

```
Two-way T-test for fetures in Slammer and Code_Red_I
p value=2.72269e-23, t value=10.114
The null hypothesis for NOA can be rejected

p value=1.9169e-19, t value=-9.14595
The null hypothesis for NOW can be rejected

p value=4.42682e-05, t value=4.09633
The null hypothesis for NOANP can be rejected

p value=8.57373e-06, t value=4.46616
The null hypothesis for NOWNP can be rejected

p value=1.9124e-06, t value=4.78176
The null hypothesis for AAPL can be rejected

p value=1.28982e-68, t value=18.4647
The null hypothesis for MAPL can be rejected

p value=1.40635e-07, t value=5.29029
The null hypothesis for AUAPL can be rejected

p value=3.81632e-20, t value=9.32875
The null hypothesis for NODA can be rejected

p value=0.0117549, t value=2.52258
The null hypothesis for NODW can be rejected

p value=2.8355e-29, t value=11.48
The null hypothesis for NOIW can be rejected

p value=2.06195e-68, t value=18.4335
The null hypothesis for AED can be rejected

p value=0.986665, t value=0.0167171
The null hypothesis for MED is accepted

p value=7.20542e-08, t value=5.41364
The null hypothesis for IAT can be rejected

p value=1.79549e-24, t value=10.3955
The null hypothesis for IGP packets can be rejected

p value=5.26296e-05, t value=4.05562
The null hypothesis for EGP packets can be rejected

p value=2.16139e-05, t value=4.26135
The null hypothesis for Incomplete packets can be rejected

p value=1.40339e-14, t value=-7.77573
The null hypothesis for packet size can be rejected
```

Figure 24.Two-sample t-test for Slammer and Code Red I.

BGP Features	P-Value
NOA	2.72E-23
NOW	1.92E-19
NOANP	4.43E-05
NOWNP	8.57E-06
AAPL	1.91E-06
MAPL	1.29E-68
AUAPL	1.41E-07
NODA	3.82E-20
NODW	0.0117549
NOIW	2.84E-29
AED	2.06E-68
MED	0.986665*
IAT	7.21E-08
IGP packets	1.80E-24
EGP packets	5.26E-05
Incomplete packets	2.16E-05
packet size	1.40E-14

Table 25. P-Values for Two-sample t-test for Slammer and Code Red

The asterisk (*) indicates the P-value accepts the Null Hypothesis

The only feature that accepts the H_0 between Slammer and Code Red I events is MED. We can reject the H_0 for all the rest.

10^{th} T-test: Moscow blackout and Code Red I

- Null Hypothesis (H_0): Features mean in Moscow blackout Anomaly = Features mean in Code Red I Anomaly.
- Alternative Hypothesis (H_a): Features mean in Moscow blackout Anomaly ≠ Features mean in Code Red I Anomaly.

```
Two-way T-test for fetures in Moscow_blackout and Code_Red_I
p value=4.10416e-104, t value=25.0984
The null hypothesis for NOA can be rejected

p value=3.8083e-23, t value=10.2096
The null hypothesis for NOW can be rejected

p value=1.95427e-84, t value=21.9027
The null hypothesis for NOANP can be rejected

p value=1.15896e-17, t value=8.75002
The null hypothesis for NOWNP can be rejected

p value=8.01438e-67, t value=-18.9325
The null hypothesis for AAPL can be rejected

p value=4.64034e-37, t value=13.3625
The null hypothesis for MAPL can be rejected

p value=7.96641e-67, t value=-18.9329
The null hypothesis for AUAPL can be rejected

p value=4.7221e-94, t value=23.4755
The null hypothesis for NODA can be rejected

p value=9.55468e-26, t value=10.8479
The null hypothesis for NODW can be rejected

p value=1.19612e-26, t value=11.063
The null hypothesis for NOIW can be rejected

p value=6.6527e-37, t value=13.33
The null hypothesis for AED can be rejected

p value=6.27219e-15, t value=7.94444
The null hypothesis for MED can be rejected

p value=4.31238e-91, t value=-22.994
The null hypothesis for IAT can be rejected

p value=7.75229e-105, t value=25.2145
The null hypothesis for IGP packets can be rejected

p value=3.4124e-06, t value=4.67575
The null hypothesis for EGP packets can be rejected

p value=2.54042e-26, t value=10.9854
The null hypothesis for Incomplete packets can be rejected

p value=0.45621, t value=-0.745449
The null hypothesis for packet size is accepted
```

Figure 25.Two-sample t-test for Moscow blackout and Code Red I.

BGP Features	P-Value
NOA	4.10E-104
NOW	3.81E-23
NOANP	1.95E-84
NOWNP	1.16E-17
AAPL	8.01E-67
MAPL	4.64E-37
AUAPL	7.97E-67
NODA	4.72E-94
NODW	9.55E-26
NOIW	1.20E-26
AED	6.65E-37
MED	6.27E-15
IAT	4.31E-91
IGP packets	7.75E-105
EGP packets	3.41E-06
Incomplete packets	2.54E-26
packet size	0.45621*

Table 26. P-Values for Two-sample t-test for Moscow and Code Red

The asterisk (*) indicates the P-value accepts the Null Hypothesis

The only feature that accepts the H_0 between Moscow blackout and Code Red I events is packet size. We can reject the H_0 for all the rest.

The results of the Two-sample t-tests conducted previously to study BGP features' tendency to behave in the same way during different anomaly events can be summarized in Figure 26 as follows:

- The highest relevance is between Code Red I and Nimda events. Out of 17 features, the mean values of 12 features during the anomaly period are equal between Code Red I and Nimda events. This can be understood since both incidents are worm attacks targeting Microsoft Internet Information Services.
- Although WannaCry is considered a worm attack, BGP behavior during this anomaly doesn't relate to other worm attacks (Slammer, Nimda, and Code Red I).
- Moscow Blackout, a link failure event, doesn't relate to the other four cyber worm attack anomalies.

WannaCry	17				
Nimda	1	17			
Slammer	1	6	17		
Moscow	1	0	2	17	
Code Red I	1	12	1	1	17
	WannaCry	Nimda	Slammer	Moscow	Code Red I

Figure 26. No. of features accepting the Null Hypothesis for two-sample t-test between different events.

A complete table with all the reported P-values for all t-tests is attached in the appendix section.

4.2.3 Correlation

To represent the strength of the linear relationship between the BGP features, NumPy, pandas, seaborn, and scipy libraries were used to perform the Pearson correlation test for the five datasets. The results for Pearson correlation coefficients are shown in the following Heatmaps.

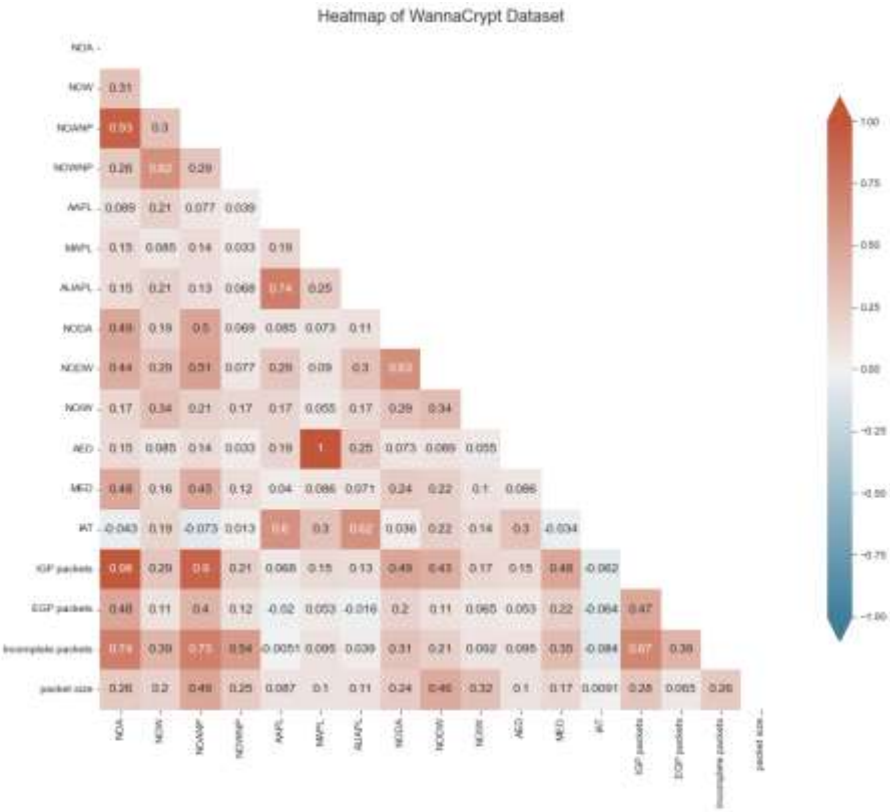


Figure 28. Heatmap for Pearson Correlation coefficients for WannaCry dataset.

Pearson correlation coefficient results for the WannaCry dataset shown on the left side indicate the following summary:

- There is a perfect positive correlation between MAPL and AED. Very high positive correlations were also detected between NOA and IGP pakcepacketsNOANP.
- Several considerable positive correlations between multiple features are also evident. However, negative correlations are minimal and very poor.

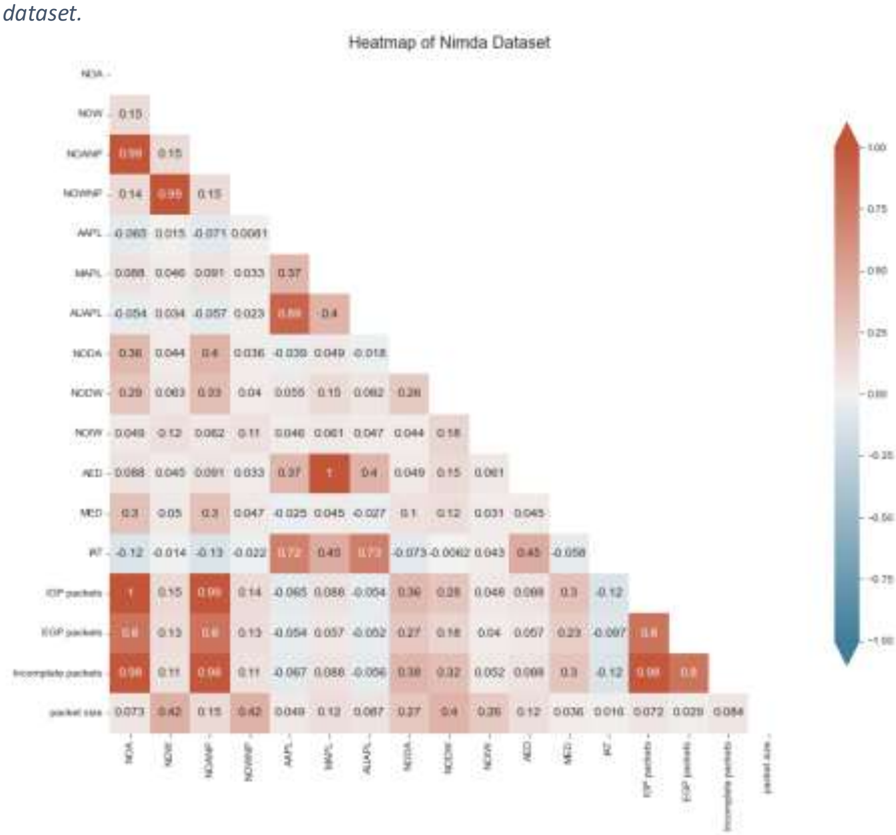


Figure 27.Heatmap for Pearson Correlation coefficients for Nimda dataset.

Pearson correlation coefficient results for the Nimda dataset illustrated on the right side point out the following summary:

- More Strong positive correlations are evident in Nimda compared to WannaCry.
- There is a perfect positive correlation between MAPL and AED.
- NOA and NOANP have strong positive correlations to IGP, EGP, and incomplete packet features.
- Several considerable positive correlations between multiple features. However, as in WannaCry, negative correlations are minimal and very poor.

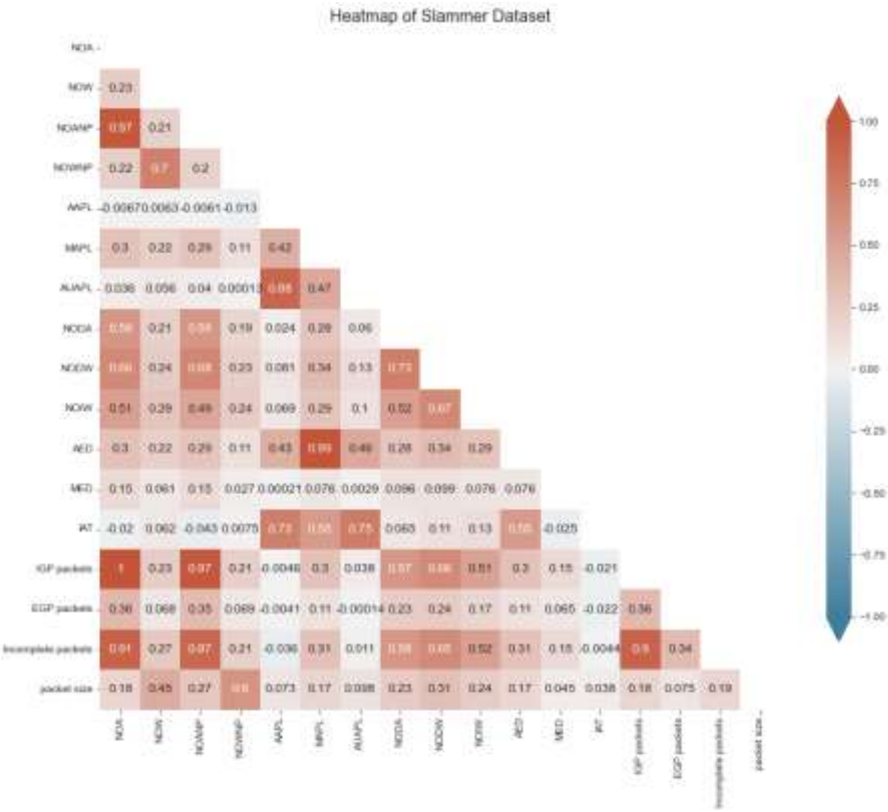


Figure 29.Heatmap for Pearson Correlation coefficients for Slammer dataset.

Pearson correlation coefficient results for the Slammer dataset illustrated on the left side indicate the following summary:

- Strong positive correlations are evident between NOA and IGP packets, MAPL and AED, NOANP and IGP packets, NOA and NOANP, and between incomplete packets and NOA and NOANP
- Several other significant positive correlations between multiple features can be seen. However, as in WannaCry and Nimda, negative correlations are minimal and poor.

Pearson correlation coefficient results for the Moscow blackout dataset illustrated on the right side signify the following:

- Strong positive correlations are evident between NOA and IGP packets, MAPL and AED, NOANP and IGP packets, NOA and NOANP, AAPL and AUAPL, and NODA and IGP packets.
- Several other significant positive correlations between multiple features can be noticed. However, as in previous datasets, negative correlations are minimal and poor.

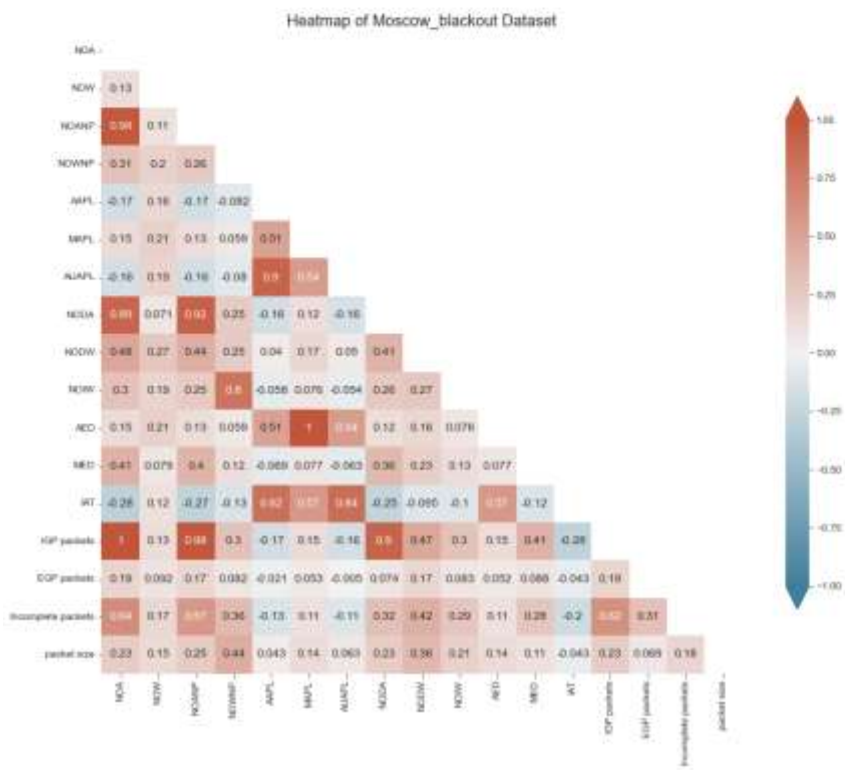
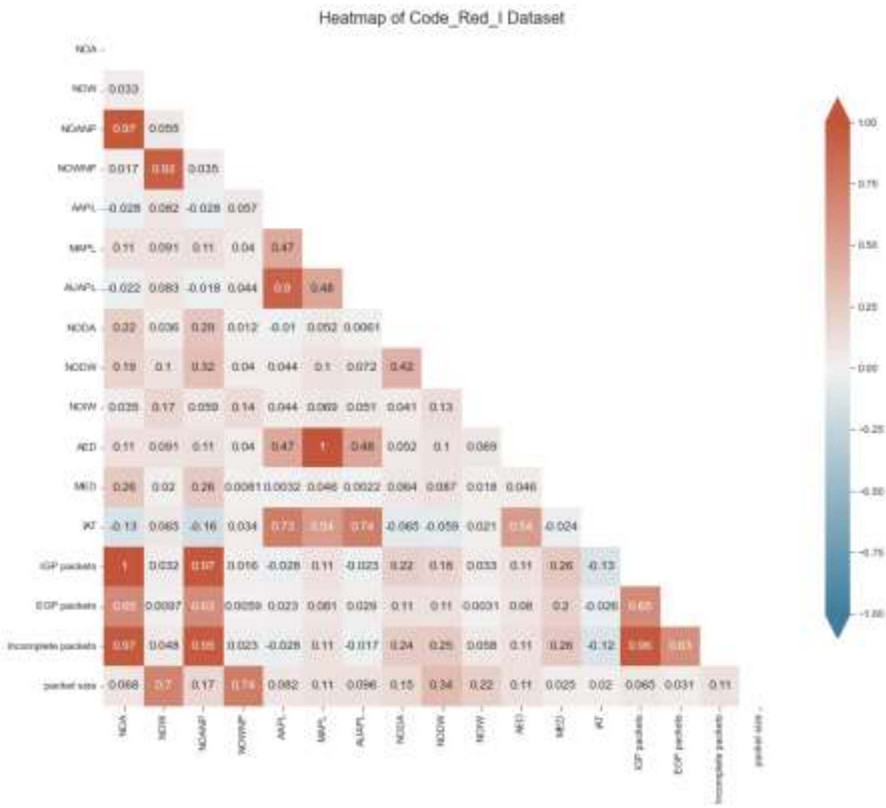


Figure 30.Heatmap for Pearson Correlation coefficients for Moscow Blackout dataset.



Pearson correlation coefficient results for the Code Red dataset shown on the left side indicate the following:

- Strong positive correlations are evident between MAPL and AED, NOA and NOANP, AAPL and AUAPL, and NOW and NOWNP.
- Same as in the Nimda dataset, NOA and NOANP have strong positive correlations to IGP, EGP, and incomplete packet features
- few other significant positive correlations between multiple features can be found. However, as noticed in all BGP datasets, negative correlations are minimal and poor.

Figure 31.Heatmap for Pearson Correlation coefficients for Code Red I dataset.

Pearson correlation results for all datasets make it possible to conclude the following points:

- Overall, there are not many high correlations. The majority of the strong correlations are positive, and negative correlations are very weak.
- Average Edit Distance (AED) and Maximum AS-path length (MAPL) are of perfect positive correlation for all datasets.
- No. of Announcement of new routes (NOA), No. of routes advertised from IGP protocols, and No. of routes advertised from unknown sources (incomplete) are high to perfect positively correlated to each other for almost all datasets.
- Inter arrival time (IAT) of packets is positively correlated to features describing the AS path length (AAPL, AUAPL, MAPL).

4.3 Level 3: Machine Learning

This section presents and discusses the results of applying six different ML classifiers to our five anomaly events. SK-learn library was uploaded to the Jupyter notebook to use different ML classifiers, GridSearchCV function from (*sklearn.model_selection.GridSearchCV*, no date) was used for hyperparameters tuning to find the model with the optimal values for each dataset. Results are shown in a normalized confusion matrix and a classification report in which (-1) represents regular traffic and (1) means anomalous traffic.

4.3.1 Multi-Layered Perceptron Classifier (MLP)

MLP Classifier with default one hidden layer of 100 perceptrons was used. The maximum iteration parameter was set to 2000. Gridsearchcv was used to choose the optimal activation function for the hidden layer {'identity', 'logistic', 'tanh', 'relu'} and the optimal solver for weight optimization {'lbfgs', 'sgd', 'adam'}, all rest parameters remained as default.

- **WannaCry Dataset**

```
Best parameters found for WannaCrypt Dataset:
 {'activation': 'logistic', 'solver': 'adam'}

              precision    recall  f1-score   support

          -1       0.69      0.67      0.68      1439
           1       0.68      0.70      0.69      1441

    accuracy                           0.68      2880
   macro avg       0.68      0.68      0.68      2880
weighted avg       0.68      0.68      0.68      2880
```

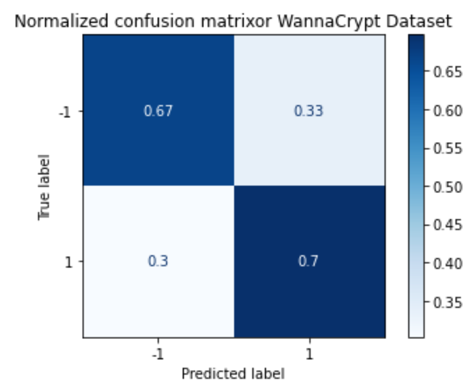


Figure 32. Performance Results of MLP Classifier with WannaCry Dataset.

The logistic activation function and adam solver were the optimal hyperparameters for the WannaCry da taset. The achieved accuracy by an MLP classifier is 68%.

- **Nimda Dataset**

```
Best parameters found for Nimda Dataset:
 {'activation': 'logistic', 'solver': 'adam'}

              precision    recall  f1-score   support

          -1       0.94      0.96      0.95      1843
           1       0.74      0.61      0.67       310

    accuracy                           0.91      2153
   macro avg       0.84      0.79      0.81      2153
weighted avg       0.91      0.91      0.91      2153
```

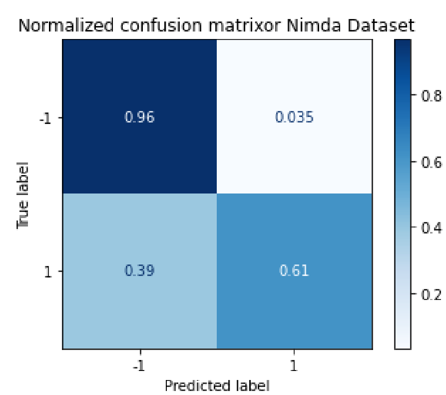


Figure 33. Performance Results of MLP Classifier with Nimda Dataset.

The logistic activation function and adam solver were the optimal hyperparameters for the Nimda datase t. The achieved accuracy by an MLP classifier is 91%. However, there is a noticeable difference in the ach ieved f1-scores between the two classes.

- **Slammer Dataset**

```
Best parameters found for Slammer Dataset:
 {'activation': 'relu', 'solver': 'adam'}
```

	precision	recall	f1-score	support
-1	0.98	0.99	0.99	1584
1	0.93	0.87	0.90	216
accuracy			0.98	1800
macro avg	0.96	0.93	0.94	1800
weighted avg	0.98	0.98	0.98	1800

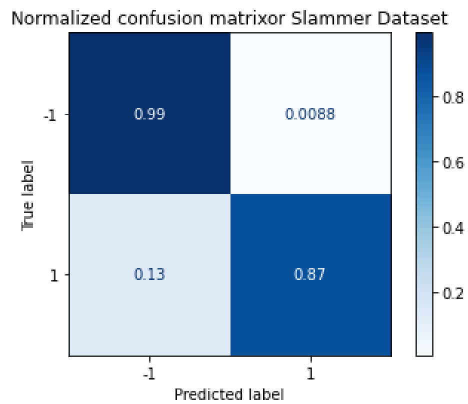


Figure 34. Performance Results of MLP Classifier with Slammer Dataset.

The relu activation function and adam solver were the optimal hyperparameters for the Slammer datase t. The achieved accuracy by an MLP classifier is 98%. However, there is a slight difference in the achieved f1-scores between the two classes.

- **Moscow Blackout Dataset**

```
Best parameters found for Moscow_blackout Dataset:
 {'activation': 'relu', 'solver': 'adam'}
```

	precision	recall	f1-score	support
-1	0.99	0.99	0.99	1738
1	0.82	0.66	0.73	62
accuracy			0.98	1800
macro avg	0.90	0.83	0.86	1800
weighted avg	0.98	0.98	0.98	1800

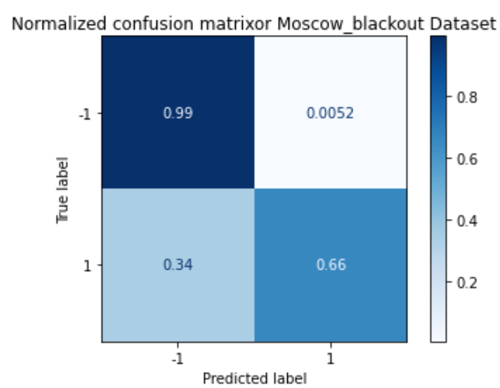


Figure 35.Performance Results of MLP Classifier with Moscow Blackout Dataset.

The relu activation function and adam solver were the optimal hyperparameters for the Moscow blackou t dataset. The achieved accuracy by an MLP classifier is 98%. However, there is a considerable difference in the achieved f1-scores between the two classes.

- **Code Red I Dataset**

```
Best parameters found for Code_Red_I Dataset:
 {'activation': 'logistic', 'solver': 'adam'}
```

	precision	recall	f1-score	support
-1	0.96	0.99	0.97	1645
1	0.79	0.55	0.65	155
accuracy			0.95	1800
macro avg	0.88	0.77	0.81	1800
weighted avg	0.94	0.95	0.94	1800

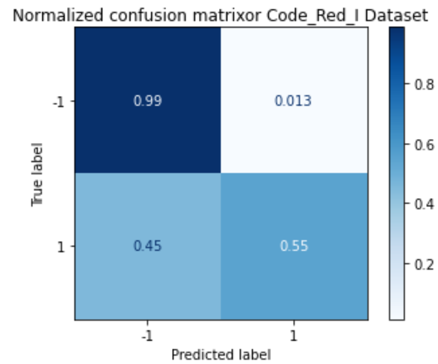


Figure 36. Performance Results of MLP Classifier with Code Red I Dataset.

The logistic activation function and adam solver were the optimal hyperparameters for the Code Red dat aset. The achieved accuracy by an MLP classifier is 95%. However, there is a significant difference in the a chieved f1-scores between the two classes.

4.3.2 Decision Tree Classifier

DT classifier from SK-learn library was used. GridSearchCV was used to choose the optimal function to measure the quality of split {‘gini’, ‘entropy’}, and all rest parameters remained as default.

- **WannaCry Dataset**

```
Best parameters found for WannaCrypt Dataset:
 {'criterion': 'gini'}
```

	precision	recall	f1-score	support
-1	0.62	0.63	0.63	1740
1	0.62	0.61	0.62	1716
accuracy			0.62	3456
macro avg	0.62	0.62	0.62	3456
weighted avg	0.62	0.62	0.62	3456

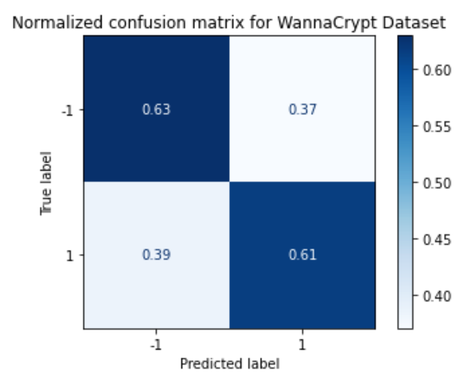


Figure 37. Performance Results of DT Classifier with WannaCry Dataset.

Gini was the optimal function for the WannaCry dataset. The achieved accuracy by a DT classifier is 62%. DT shows worse performance than MLP for this dataset.

- **Nimda Dataset**

```
Best parameters found for Nimda Dataset:
 {'criterion': 'entropy'}
```

	precision	recall	f1-score	support
-1	0.91	0.92	0.91	2165
1	0.56	0.53	0.54	418
accuracy			0.86	2583
macro avg	0.73	0.72	0.73	2583
weighted avg	0.85	0.86	0.85	2583

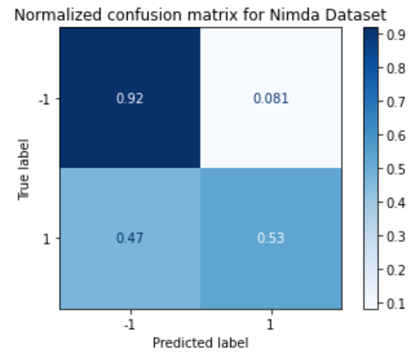


Figure 38. Performance Results of DT Classifier with Nimda Dataset

Entropy was the optimal function for the Nimda dataset. The achieved accuracy by a DT classifier is 86%. DT shows worse performance than MLP for this dataset, and there is a significant difference in the achie ved f1-scores between the two classes.

- **Slammer Dataset**

```
Best parameters found for Slammer Dataset:
 {'criterion': 'gini'}
```

	precision	recall	f1-score	support
-1	0.98	0.97	0.98	1904
1	0.80	0.86	0.83	256
accuracy			0.96	2160
macro avg	0.89	0.92	0.90	2160
weighted avg	0.96	0.96	0.96	2160

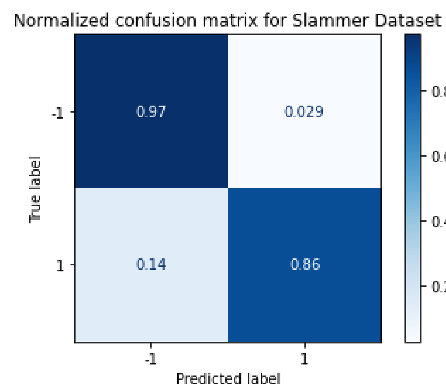


Figure 39.Performance Results of DT Classifier with Slammer Dataset

Gini was the optimal function for the Slammer dataset. The achieved accuracy by a DT classifier is 96%, a slightly worse accuracy than MLP for this dataset, and there is a considerable difference in the achieved f 1-scores between the two classes.

- **Moscow Blackout Dataset**

```
Best parameters found for Moscow_blackout Dataset:
 {'criterion': 'entropy'}

              precision    recall  f1-score   support

          -1       0.99      0.99      0.99      2088
           1       0.60      0.64      0.62        72

    accuracy                           0.97      2160
   macro avg       0.79      0.81      0.80      2160
weighted avg       0.97      0.97      0.97      2160
```

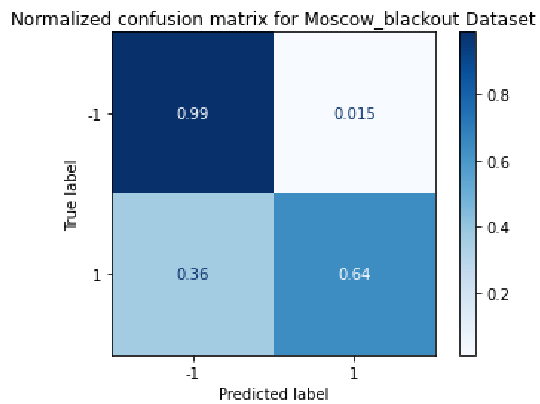


Figure 40.Performance Results of DT Classifier with Moscow Blackout Dataset

Entropy was the optimal function for the Moscow dataset. The achieved accuracy by a DT classifier is 97 %, a slightly less accuracy than MLP for this dataset. Additionally, there is a considerable difference in th e achieved f1-scores between the two classes.

- **Code Red I Dataset**

```
Best parameters found for Code_Red_I Dataset:
 {'criterion': 'entropy'}

              precision    recall  f1-score   support

          -1       0.97      0.94      0.95      1985
           1       0.47      0.62      0.53       175

    accuracy                           0.91      2160
   macro avg       0.72      0.78      0.74      2160
weighted avg       0.93      0.91      0.92      2160
```

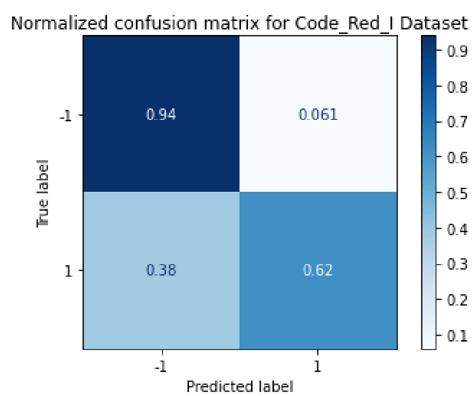


Figure 41. Performance Results of DT Classifier with Code Red I Dataset

Entropy was the optimal function for the Code Red dataset. The achieved accuracy by a DT classifier is 91 %, a worse accuracy than MLP for this dataset. Moreover, there is a considerable difference in the achiev ed f1-scores between the two classes.

4.3.3 K-Nearest Neighbor Classifier

KNN classifier from SK-learn library was imported. GridSearchCV was used to choose the optimal no. of neighbors in (range (1,100)), and the weight function used in prediction {‘uniform’, ‘distance’}, all rest parameters remained as default.

- **WannaCry Dataset**

```
Best parameters found for WannaCrypt Dataset:
 {'n_neighbors': 48, 'weights': 'distance'}

              precision    recall  f1-score   support

          -1       0.65      0.62      0.63      1701
           1       0.64      0.67      0.66      1755

    accuracy                           0.65      3456
   macro avg       0.65      0.65      0.65      3456
weighted avg       0.65      0.65      0.65      3456
```

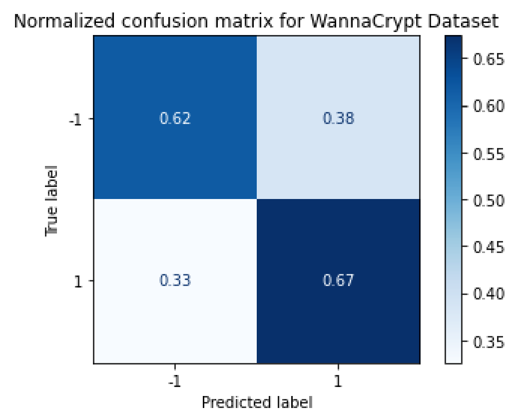


Figure 42. Performance Results of KNN Classifier with WannaCry Dataset

The optimal number of neighbors in the KNN classifier for the WannaCry dataset was 48. The achieved a ccuracy by a KNN classifier is 65% which is better than the DT classifier but not as good as the MLP classif ier.

- **Nimda Dataset**

```
Best parameters found for  Nimda  Dataset :
 {'n_neighbors': 14, 'weights': 'distance'}

              precision    recall  f1-score   support

          -1       0.91      0.97      0.94      2222
           1       0.68      0.45      0.54       361

    accuracy                           0.89      2583
   macro avg       0.80      0.71      0.74      2583
weighted avg       0.88      0.89      0.88      2583
```

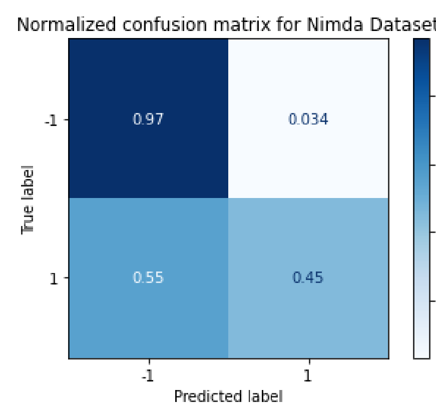


Figure 43. Performance Results of KNN Classifier with Nimda Dataset

The optimal number of neighbors in the KNN classifier for the Nimda dataset was 14. The achieved accu racy by a KNN classifier is 89% which is better than the DT classifier but not as good as the MLP classifier. A considerable difference in the achieved f1-scores between the two classes can be observed from the c onfusion matrix.

- **Slammer Dataset**

Best parameters found for Slammer Dataset:
{'n_neighbors': 11, 'weights': 'distance'}

	precision	recall	f1-score	support
-1	0.97	0.99	0.98	1900
1	0.92	0.78	0.84	260
accuracy			0.96	2160
macro avg	0.94	0.88	0.91	2160
weighted avg	0.96	0.96	0.96	2160

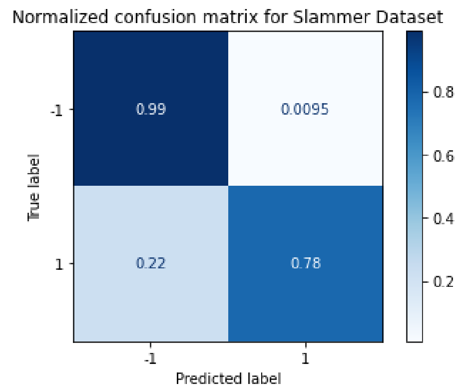


Figure 44. Performance Results of KNN Classifier with Slammer Dataset

The optimal no. of neighbors in the KNN classifier for the Slammer dataset was 11. The achieved accurac y by a KNN classifier is 96%, the same as the accuracy obtained from the DT classifier and slightly worse t han MLP accuracy. A noticeable difference in the achieved f1-scores between the two classes can be see n from the confusion matrix.

- **Moscow Blackout Dataset**

Best parameters found for Moscow_blackout Dataset:
{'n_neighbors': 2, 'weights': 'uniform'}

	precision	recall	f1-score	support
-1	0.99	1.00	0.99	2089
1	0.93	0.56	0.70	71
accuracy			0.98	2160
macro avg	0.96	0.78	0.85	2160
weighted avg	0.98	0.98	0.98	2160

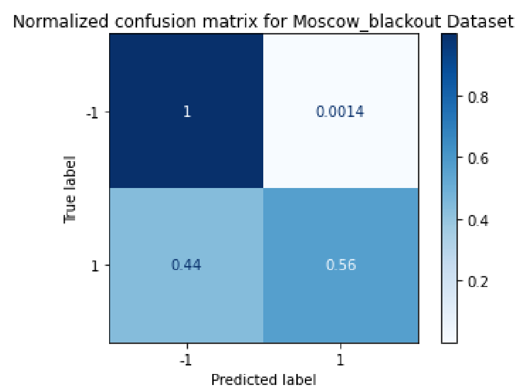


Figure 45. Performance Results of KNN Classifier with Moscow Blackout Dataset

The optimal no. of neighbors in the KNN classifier for the Moscow dataset was only 2. The achieved accu racy by a KNN classifier is 98 %, the same as the accuracy obtained from the MLP classifier and slightly be tter than DT accuracy. However, there is a difference in the achieved f1-scores between the two classes.

- **Code Red I Dataset**

```
Best parameters found for Code_Red_I Dataset:
 {'n_neighbors': 14, 'weights': 'distance'}
```

	precision	recall	f1-score	support
-1	0.94	0.99	0.97	1964
1	0.80	0.40	0.53	196
accuracy			0.94	2160
macro avg	0.87	0.69	0.75	2160
weighted avg	0.93	0.94	0.93	2160

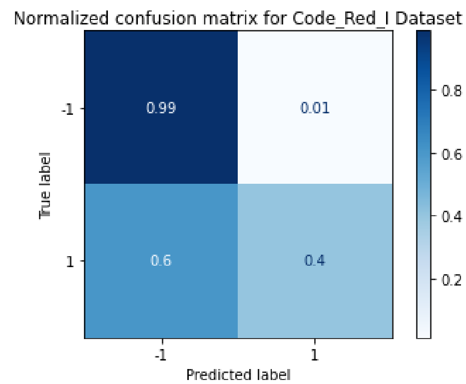


Figure 46. Performance Results of KNN Classifier with Code Red I Dataset

The optimal no. of neighbors in the KNN classifier for the Moscow dataset was only 14. The achieved acc uracy by a KNN classifier is 94 %, better accuracy than the DT classifier but not as good as the MLP classif ier. However, there is a difference in the achieved f1-scores between the two classes.

4.3.4 Support Vector Machines Classifier

SV classifier from SK-learn library was imported. GridSearchCV was used to choose the optimal kernel type {'linear', 'poly', 'rbf', 'sigmoid'}, all rest parameters remained as default.

- **WannaCry Dataset**

```
Best parameters found for WannaCrypt Dataset:
 {'kernel': 'rbf'}
```

	precision	recall	f1-score	support
-1	0.70	0.66	0.68	1758
1	0.67	0.71	0.69	1698
accuracy			0.68	3456
macro a**vg**	0.68	0.68	0.68	3456
weighted avg	0.68	0.68	0.68	3456

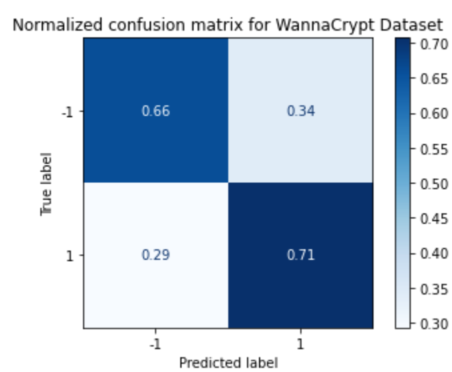


Figure 47. Performance Results of SVM Classifier with WannaCry Dataset

RBF was the optimal kernel for the SVM classifier in the WannaCry dataset. The achieved accuracy by an SVM classifier is 68%, the same accuracy as the MLP classifier and the best-achieved score so far.

- **Nimda Dataset**

```
Best parameters found for  Nimda  Dataset :
 {'kernel': 'rbf'}
```

	precision	recall	f1-score	support
-1	0.91	0.98	0.94	2199
1	0.83	0.42	0.55	384
accuracy			0.90	2583
macro avg	0.87	0.70	0.75	2583
weighted avg	0.89	0.90	0.89	2583

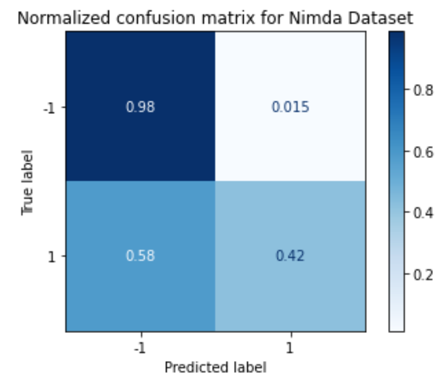


Figure 48.Performance Results of SVM Classifier with Nimda Dataset

RBF was the optimal kernel for the SVM classifier in the Nimda dataset. The achieved accuracy by an SV M classifier is 90%, a very close accuracy to an MLP classifier and better than the DT and the KNN classifi ers. However, there is a notable difference in the achieved f1-scores between the two classes.

- **Slammer Dataset**

```
Best parameters found for Slammer Dataset:
 {'kernel': 'rbf'}
```

	precision	recall	f1-score	support
-1	0.97	0.99	0.98	1889
1	0.95	0.78	0.86	271
accuracy			0.97	2160
macro avg	0.96	0.89	0.92	2160
weighted avg	0.97	0.97	0.97	2160

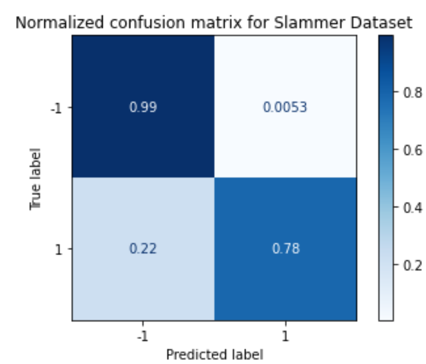


Figure 49. Performance Results of SVM Classifier with Slammer Dataset

RBF was the optimal kernel for the SVM classifier in the Slammer dataset. The achieved accuracy by an S VM classifier is 97%, slightly worse than MLP and slightly better than DT and KNN. However, there is a dif ference in the achieved f1-scores between the two classes.

- **Moscow Blackout Dataset**

```
Best parameters found for Moscow_blackout Dataset:
 {'kernel': 'linear'}
```

	precision	recall	f1-score	support
-1	0.98	0.99	0.99	2089
1	0.76	0.54	0.63	71
accuracy			0.98	2160
macro avg	0.87	0.76	0.81	2160
weighted avg	0.98	0.98	0.98	2160

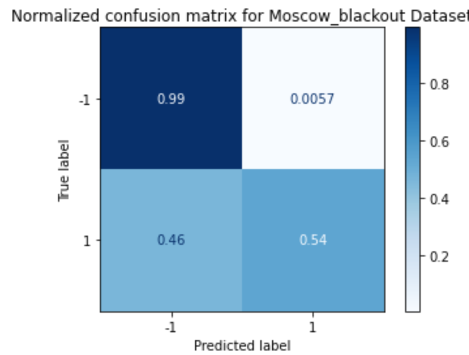


Figure 50. Performance Results of SVM Classifier with Moscow Blackout Dataset

Linear was the optimal kernel for the SVM classifier in the Moscow dataset. The achieved accuracy by an SVM classifier is 98%, the same as MLP and KNN. Yet, there is a significant difference in the achieved f1-scores between the two classes, as seen from the confusion matrix.

- **Code Red I Dataset**

```
Best parameters found for  Code_Red_I  Dataset :
 {'kernel': 'rbf'}
```

	precision	recall	f1-score	support
-1	0.95	0.99	0.97	1989
1	0.87	0.42	0.56	171
accuracy			0.95	2160
macro avg	0.91	0.70	0.77	2160
weighted avg	0.95	0.95	0.94	2160

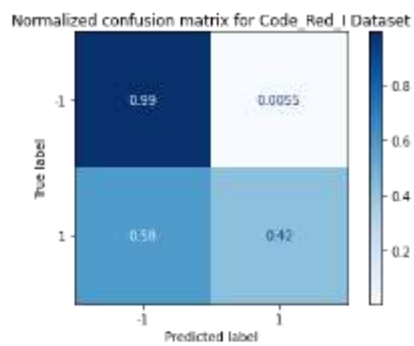


Figure 51. Performance Results of SVM Classifier with Code Red I Dataset

RBF was the optimal kernel for the SVM classifier in the Code Red dataset. The obtained accuracy by an SVM classifier is 95%, the same as MLP and better than DT and KNN for this dataset. Yet, there is significant variation in the achieved f1-scores between the two classes, as can be seen from the confusion matrix.

4.3.5 Random Forest Classifier

RF classifier from SK-learn library was imported. GridSearchCV was used to choose the optimal function to measure the split quality {'gini', 'entropy'} and the no. of trees in the forest in (range (50,151)), all rest parameters remained as default.

- **WannaCry Dataset**

```
Best parameters found for WannaCrypt Dataset:
 {'criterion': 'entropy', 'n_estimators': 131}
```

	precision	recall	f1-score	support
-1	0.69	0.67	0.68	1731
1	0.68	0.70	0.69	1725
accuracy			0.69	3456
macro avg	0.69	0.69	0.69	3456
weighted avg	0.69	0.69	0.69	3456

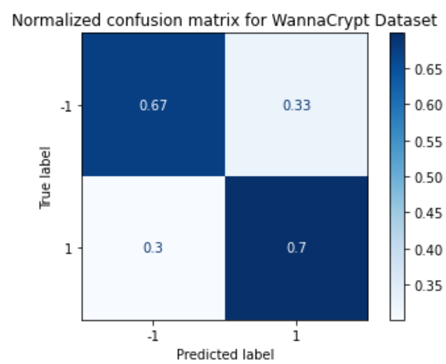


Figure 52. Performance Results of RF Classifier with WannaCry Dataset

With 131 estimators in the forest and the entropy split function, the RF classifier achieved the best accur acy of 69% for the wannaCry dataset. The highest achieved accuracy for this dataset so far.

- **Nimda Dataset**

```
Best parameters found for Nimda Dataset:
 {'criterion': 'entropy', 'n_estimators': 60}
```

	precision	recall	f1-score	support
-1	0.93	0.97	0.95	2215
1	0.78	0.58	0.66	368
accuracy			0.92	2583
macro avg	0.86	0.78	0.81	2583
weighted avg	0.91	0.92	0.91	2583

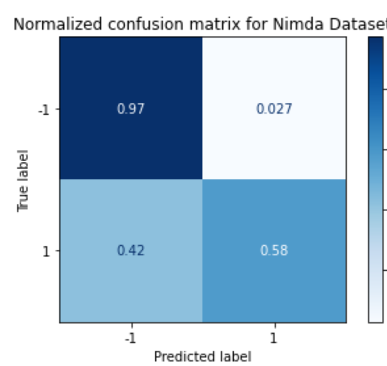


Figure 53. Performance Results of RF Classifier with Nimda Dataset

With 60 estimators in the forest and the entropy split function, the RF classifier achieved the best accura cy of 92% for the Nimda dataset. The highest achieved accuracy for this dataset so far. However, there is significant variation in the achieved f1-scores between the two classes, as seen from the confusion matri x.

- **Slammer Dataset**

```
Best parameters found for  Slammer  Dataset :
 {'criterion': 'gini', 'n_estimators': 58}
```

	precision	recall	f1-score	support
-1	0.98	0.99	0.99	1908
1	0.93	0.85	0.89	252
accuracy			0.97	2160
macro avg	0.96	0.92	0.94	2160
weighted avg	0.97	0.97	0.97	2160

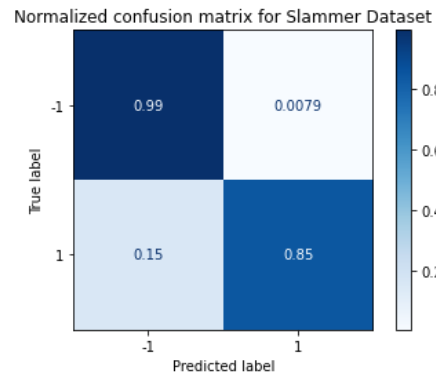


Figure 54. Performance Results of RF Classifier with Slammer Dataset

With 58 estimators in the forest and the Gini split function, the RF classifier achieved the best accuracy o f 97% for the Slammer dataset, the same as the RF classifier. However, there is noticeable variation in th e achieved f1-scores between the two classes, as seen from the confusion matrix.

- **Moscow Blackout Dataset**

```
Best parameters found for  Moscow_blackout  Dataset :
 {'criterion': 'entropy', 'n_estimators': 54}
```

	precision	recall	f1-score	support
-1	0.99	1.00	0.99	2085
1	0.92	0.63	0.75	75
accuracy			0.99	2160
macro avg	0.95	0.81	0.87	2160
weighted avg	0.98	0.99	0.98	2160

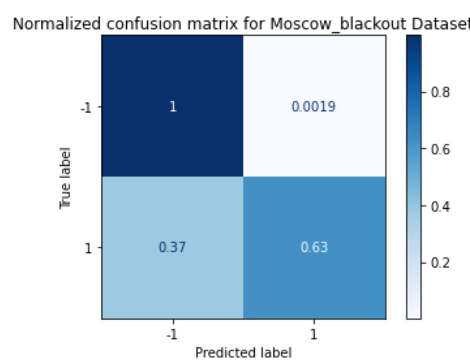


Figure 55. Performance Results of RF Classifier with Moscow Blackout Dataset

With 54 estimators in the forest and the entropy split function, the RF classifier achieved the best accura cy of 99% for the Moscow dataset; the highest achieved accuracy for this dataset so far. Nevertheless, th ere is a considerable variation in the achieved f1-scores between the two classes, as seen from the confu sion matrix.

- **Code Red I Dataset**

```
Best parameters found for  Code_Red_I  Dataset :
 {'criterion': 'gini', 'n_estimators': 128}
```

	precision	recall	f1-score	support
-1	0.96	0.99	0.97	1977
1	0.83	0.55	0.66	183
accuracy			0.95	2160
macro avg	0.89	0.77	0.82	2160
weighted avg	0.95	0.95	0.95	2160

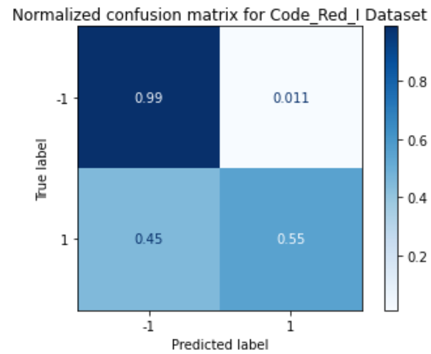


Figure 56. Performance Results of RF Classifier with Code Red I Dataset

With 128 estimators in the forest and the Gini split function, the RF classifier achieved the best accuracy of 95% for the Code Red dataset. The highest achieved accuracy for this dataset and the same as MLP an d RF classifiers. Nevertheless, there is a considerable variation in the achieved f1-scores between the two classes.

4.3.6 Naïve Bayes Classifier

MultinominalNB classifier from the SK-learn library was imported. Results were obtained with default parameters.

- **WannaCry Dataset**

	precision	recall	f1-score	support
-1	0.67	0.52	0.58	1731
1	0.61	0.74	0.67	1725
accuracy			0.63	3456
macro avg	0.64	0.63	0.63	3456
weighted avg	0.64	0.63	0.63	3456

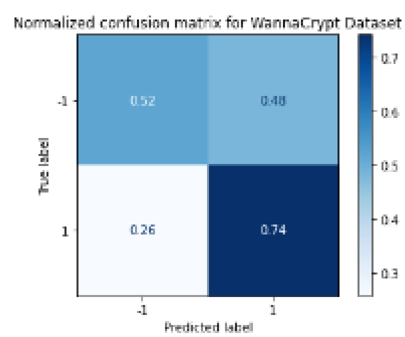


Figure 57. Performance Results of NB Classifier with WannaCry Dataset

The obtained classification accuracy from an NB classifier for the WannaCry dataset was 63%. Except for the DT classifier, which achieved 62% accuracy, NB performance for the WannaCry dataset is worse than all remaining classifiers.

- **Nimda Dataset**

	precision	recall	f1-score	support
-1	0.93	0.93	0.93	2215
1	0.59	0.60	0.60	368
accuracy			0.88	2583
macro avg	0.76	0.77	0.76	2583
weighted avg	0.88	0.88	0.88	2583

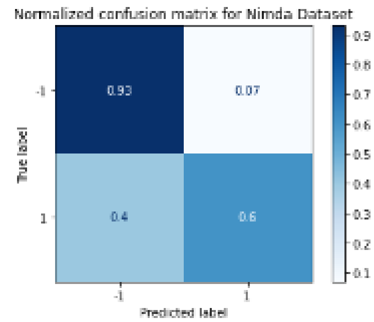


Figure 58. Performance Results of NB Classifier with Nimda Dataset

The achieved classification accuracy from an NB classifier for the Nimda dataset was 88%. Except for the DT classifier, which reached 86% accuracy, NB performance for the Nimda dataset is worse than all rema ining classifiers.

- **Slammer Dataset**

	precision	recall	f1-score	support
-1	0.96	0.96	0.96	1908
1	0.71	0.72	0.72	252
accuracy			0.93	2160
macro avg	0.84	0.84	0.84	2160
weighted avg	0.93	0.93	0.93	2160

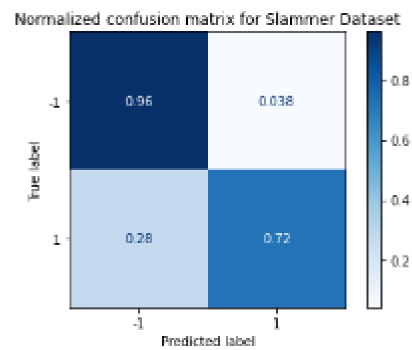


Figure 59. Performance Results of NB Classifier with Slammer Dataset

The classification accuracy from an NB classifier for the Slammer dataset was 93%. This performance is t he worst compared to all other classifiers.

- **Moscow Blackout Dataset**

	precision	recall	f1-score	support
-1	0.99	0.98	0.99	2085
1	0.58	0.69	0.63	75
accuracy			0.97	2160
macro avg	0.79	0.84	0.81	2160
weighted avg	0.97	0.97	0.97	2160

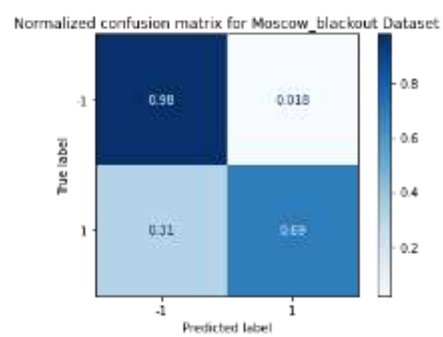


Figure 60. Performance Results of NB Classifier with Moscow Blackout Dataset

The obtained classification accuracy from the NB classifier for the Moscow dataset was 97%. Same the D T classifier accuracy and slightly worse than other classifiers.

- **Code Red I Dataset**

	precision	recall	f1-score	support
-1	0.96	0.92	0.94	1977
1	0.42	0.64	0.50	183
accuracy			0.89	2160
macro avg	0.69	0.78	0.72	2160
weighted avg	0.92	0.89	0.90	2160

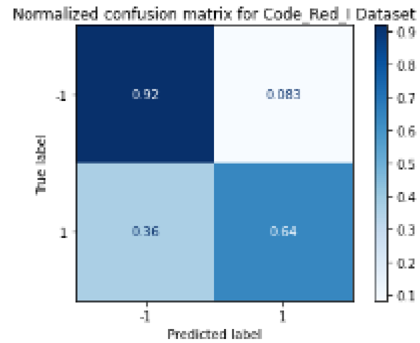


Figure 61. Performance Results of NB Classifier with Code Red I Dataset

The obtained classification accuracy from the NB classifier for the Code Red dataset was 89%. This perfor mance is the worst compared to all other classifiers.

The table and graph below summarize the performance of all ML models for the different datasets. A further Detailed table with all the reported accuracies for all datasets is attached in the appendix section.

Table 27. Weighted Average F1-Score for All models

	MLP	DT	KNN	SVC	RF	NB
WannaCry	0.68	0.62	0.65	0.68	**0.69**	0.63
Nimda	**0.91**	0.85	0.88	0.89	**0.91**	0.88
Slammer	**0.98**	0.96	0.96	0.97	0.97	0.93
Moscow	**0.98**	0.97	**0.98**	**0.98**	**0.98**	0.97
Code Red	0.94	0.92	0.94	0.94	**0.95**	0.9

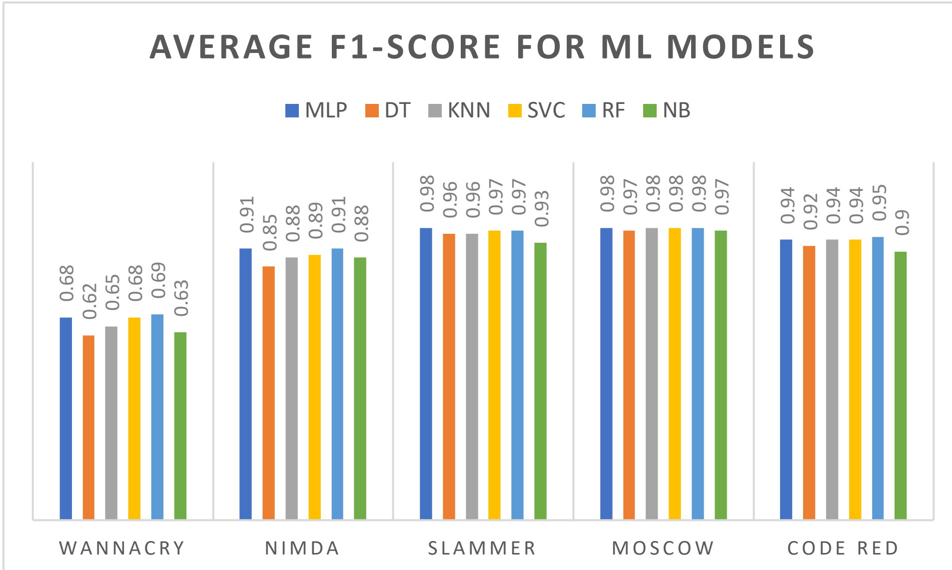


Figure 62. Weighted Average F1-Score for The Models

The results shown above make it possible to conclude the following:

- There is no significant difference in the achieved accuracies between our six proposed ML models. However, MLP and RF classifiers achieved the highest F1-Scores for all the datasets, while NB and RF are the least accurate models.
- By using the GridSearch optimization function, our proposed ML classifiers achieved better performance than the solutions suggested by the following authors: (Al-Rousan and Trajković., 2012), (Dai, Wang and Wang, 2019), (Ding et al., 2016), (Li et al., 2014) and (Karimi et al., 2019).
- Among all various events, ML models achieved the highest F1-Scores of 98% for Slammer and Moscow events, and significantly lower scores were obtained for the WannaCry dataset. These results, accompanied by features trends Figures 10 to 14, might highlight the issue of misleading spikes in regular traffic and thus the importance of filtering the noisy data before developing ML models (Peng et al., 2021).
- Except for the balanced WannaCry Dataset, there is a noticeable difference in F1-Scores between the classes (regular and anomalous) for all four imbalanced datasets. The classifier achieves higher scores for the majority class (-1 regular) because the minority class (1 irregular) doesn't have enough data points for the classifier to generalize a typical irregular behavior. Hence, the performance of the classifier drop (Japkowicz and Stephen, 2002).
- An Interesting observation is that although WannaCry is a balanced dataset with the highest no. of records for each class, the lowest accuracy and F1 scores were attained for this dataset (68% compared to more than 90% accuracies for the remaining dataset). The reason for this might be the noisiness of the dataset, as shown in Figure 10. However, further investigations are required to develop better models.

4.4 Level 4: Deep Learning

This section discusses the binary classification results obtained from CNN and GRU deep learning models. Keras and TensorFlow libraries were uploaded to the Jupyter notebook to use different DL layers. The classification label for regular traffic was changed from (-1) to (0) as Keras Library models require that. Multiple Hyperparameters were experimented to find the DL model with the optimal accuracy for each dataset. Results are shown in terms of loss and accuracy.

4.4.1 CNN Model

Table 28 shows the experimented Hyperparameters for our CNN model. Different combinations were tested to achieve the highest accuracy and lowest loss for each dataset.

Table 28. CNN Hyperparameters Search Space

Hyperparameters Names	Hyperparameters Experimented Values
No. of Convolution Layers	1,2
No. of Filters in the Convolution Layers	128, 64, 32
No. of Dense Layers	1, 2
No. of Neurons in Dense Layers	10,20,50
Optimizers	Adam, RMSprop, SGD
No. of Epochs	50, 100, 150, 200, 250, 300

- **WannaCry Dataset**

For the WannaCry dataset, the highest achieved accuracy was 68.6% with a CNN model that has one Convolutional Layer with 32 filters, two dense layers, 250 epochs, and the adam optimizer.

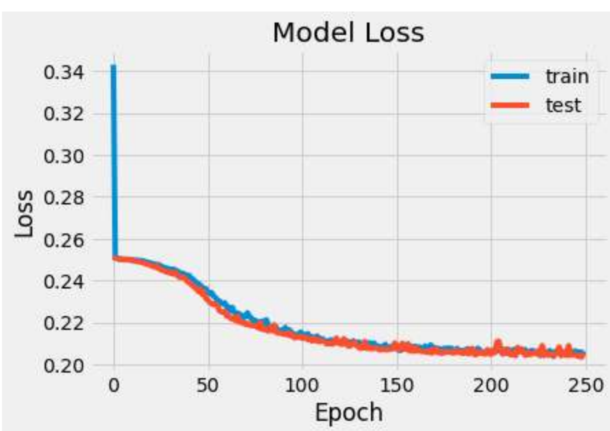


Figure 64. CNN Loss Curve for WannaCry Dataset

Figure 63. CNN Accuracy Curve for WanaCry Dataset

```
Model Loss: 0.2009 - Model Accuracy: 0.6861
```

The model accuracy is 68.6%, and the training and validation curves indicate a consistent model converg ence behavior.

- **Nimda Dataset**

The CNN model's best-achieved accuracy for the Nimda dataset was 89.39%. In this case, the optimum CNN model consisted of one Convolutional Layer with 32 filters, two dense layers with a drop out of 0.2 between them, 300 epochs, and the adam optimizer.

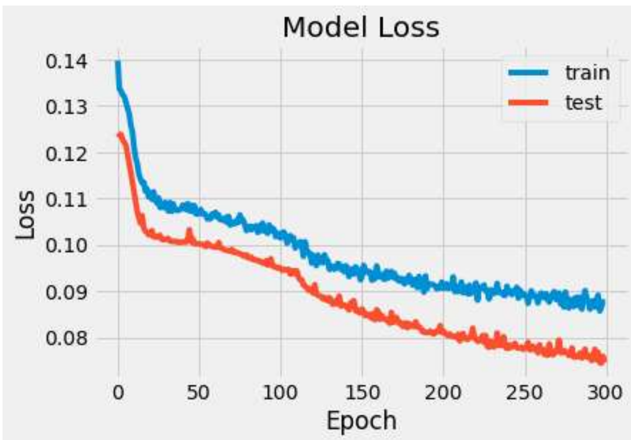


Figure 65. CNN Loss Curve for Nimda Dataset

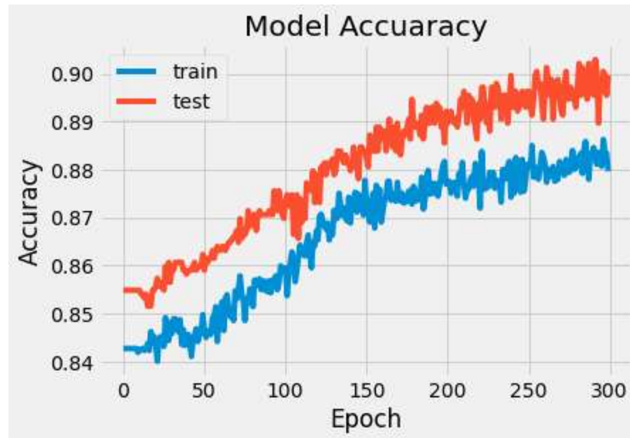


Figure 66. CNN Accuracy Curve for Nimda Dataset

```
Model Loss: 0.0788 - Model Accuracy: 0.8940
```

The model accuracy is 89.4%. The training and validation curves indicate that the model struggles with g eneralizing a common behavior for regular and abnormal data in the Nimda dataset. Moreover, the mod el convergence is relatively slow and maybe requires more epochs for full convergence.

- **Slammer Dataset**

The best-achieved accuracy for Slammer dataset by the CNN model was 97.2%. The optimum CNN model consisted of two Convolutional Layers with 64 and 32 no. of filters, respectively, followed by two dense layers. The optimum optimizer was adam, and the model converged with 200 epochs.

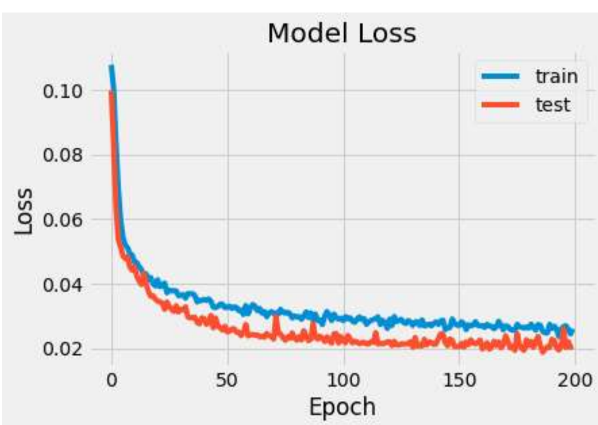


Figure 68. CNN Loss Curve for Slammer Dataset

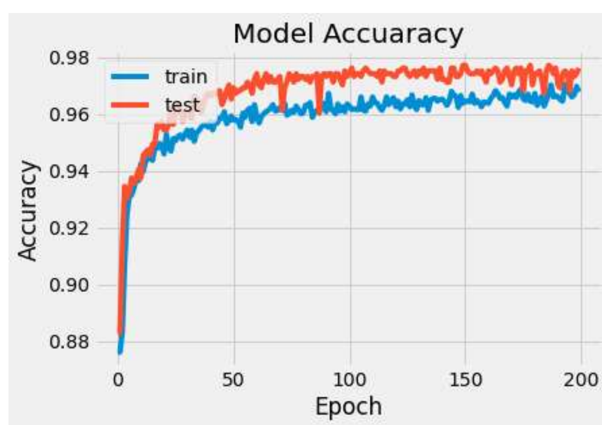


Figure 67. CNN Accuracy Curve for Slammer Dataset

`Model Loss: 0.0209 -Model Accuracy: 0.9726`

The model accuracy shows adequate precision, and the learning curves for the training and validation sets indicate stable model behavior and decent convergence.

- **Moscow Blackout Dataset**

For the Moscow Blackout dataset, the highest obtained accuracy was 98.6%. The CNN model consisted of one Convolutional Layer with 32 filters, followed by two dense layers with a drop out of 0.2 between them. In this scenario, the no. of ephocs was100, and the adam optimizer gave this optimum performance.

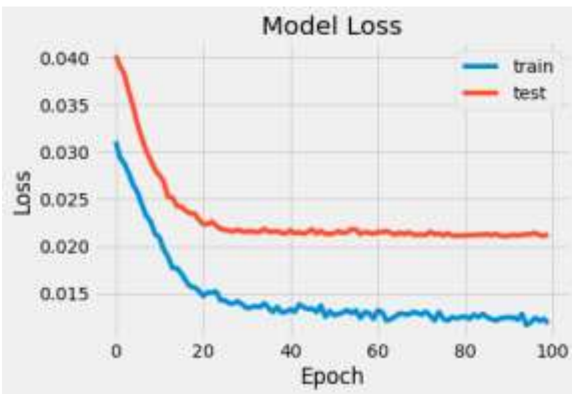


Figure 69. CNN Loss Curve for Moscow Dataset

Figure 70. CNN Accuracy Curve for Moscow Dataset

`Model Loss: 0.0132 - Model Accuracy: 0.9863`

Although this CNN model's loss and accuracy curves show differences between the training and validatio n sets, these variations are minimal, and the model performance is acceptable.

- **Code Red Dataset**

The best-achieved accuracy from the CNN model for the Code Red dataset was 95.3%. The CNN model consisted of one convolutional layer with 32 filters, two dense Layers, 300 epochs, and adam optimizer.

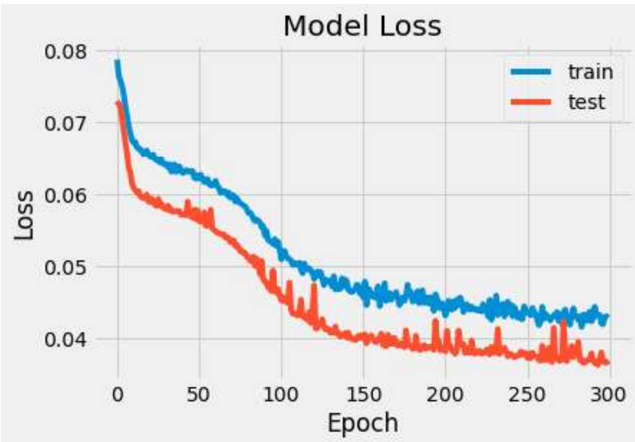


Figure 72. CNN Loss Curve for Code Red dataset

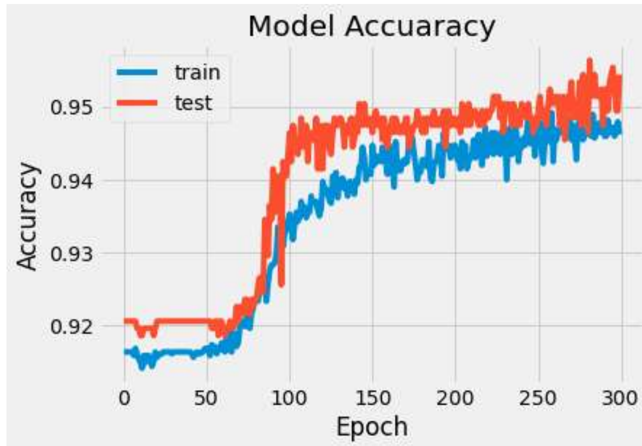


Figure 71. CNN Accuracy Curve for Code Red Dataset

```
Model Loss: 0.0378 - Model Accuracy: 0.9532
```

The model accuracy is 95.3%, and the loss is 3.7%. There are slight differences between the training and validation curves; this may require additional parameters tuning to enhance this performance further.

4.4.2 GRU Model

Table 29 shows the experimented Hyperparameters for the GRU model. Different combinations were tested to achieve the highest accuracy and lowest loss for each dataset.

Table 29. GRU Hyperparameters Search Space

Hyperparameters Names	Hyperparameters Experimented Values
No. of GRU Layers	1,2
No. of units in the GRU Layers	128, 64,
No. of Dense Layers	100, 80, 50, 25
No. of Neurons in Dense Layers	10,20
Optimizers	Adam, RMSprop, SGD
No. of Epochs	50, 100, 150, 200, 250, 300

- **WannaCry Dataset**

The best possible accuracy achieved with GRU classification for the WannaCry dataset was 58.2%. The GRU model consisted of two GRU layers with 50 and 25 units, respectively, followed by a single dense layer. The optimal optimizer was the RMS prop, and the model was trained with 250 epochs.

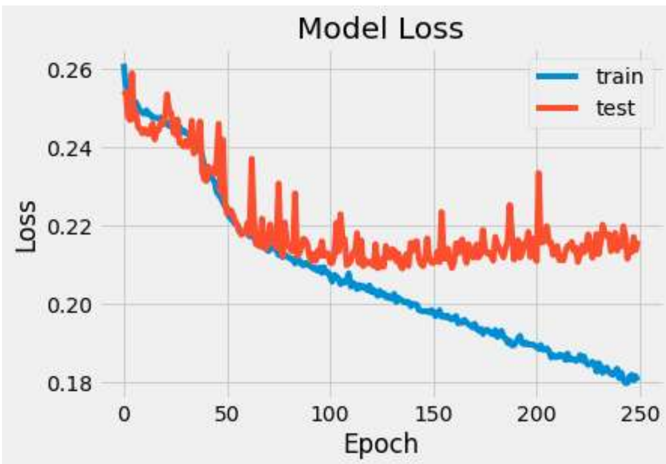


Figure 74. GRU Loss Curve for WannaCry Dataset

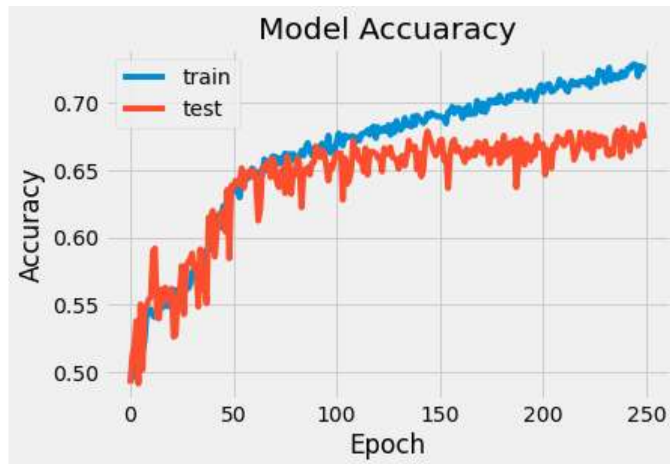


Figure 73. GRU Accuracy Curve for WannaCry Dataset

```
Model Loss: 0.2753 - Model Accuracy: 0.5822
```

The achieved accuracy by the GRU model for the WannaCry dataset is significantly worse than the accuracy obtained by the CNN model. Moreover, GRU loss and accuracy curves show overfitting behavior, which indicates that the GRU model might not perform accurately with real-time (unseen) data.

- **Nimda Dataset**

For Nimda Dataset, GRU's best-achieved accuracy was 90.04%. This dataset's optimum GRU model specifications were one GRU layer with 50 units, one dense layer, and adam optimizer.

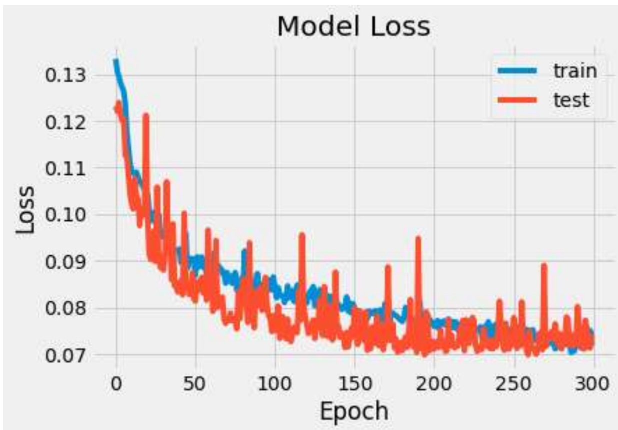


Figure 76. GRU Loss Curve for Nimda Dataset

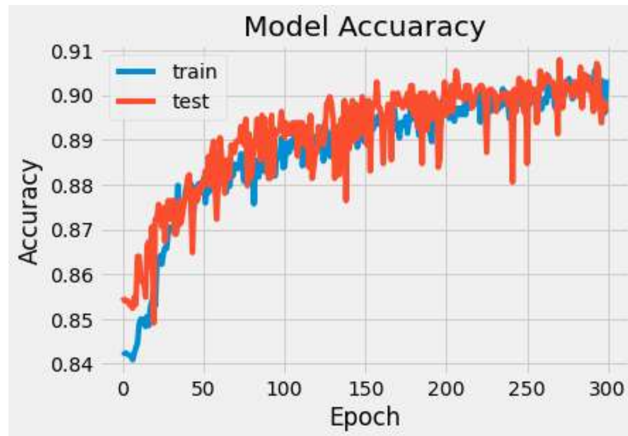


Figure 75. GRU Accuracy Curve for Nimda Dataset

```
Model Loss: 0.0764 - Model Accuracy: 0.9046
```

This obtained accuracy is slightly better than the accuracy obtained from the CNN model. Although Loss and Accuracy curves are not perfectly smooth, the GRU model converges in a better way than the CNN m odel for this Dataset.

- **Slammer Dataset**

GRU model achieved the best score of 96.7% for the slammer dataset. GRU model parameters in this scenario are as follows: two GRU layers with 50 and 25 units, respectively, one dense layer, adam optimizer, and 200 epochs.

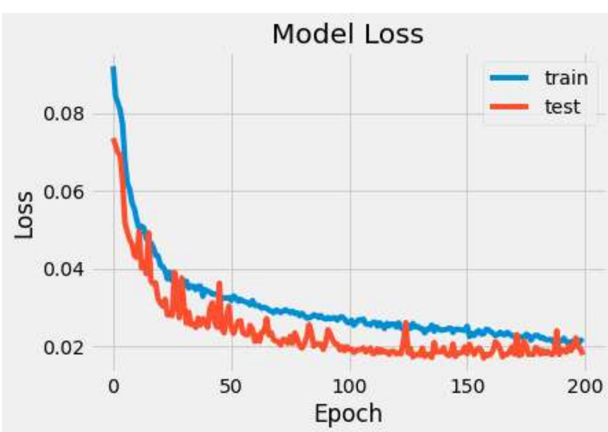


Figure 78. GRU Loss Curve for Slammer Dataset

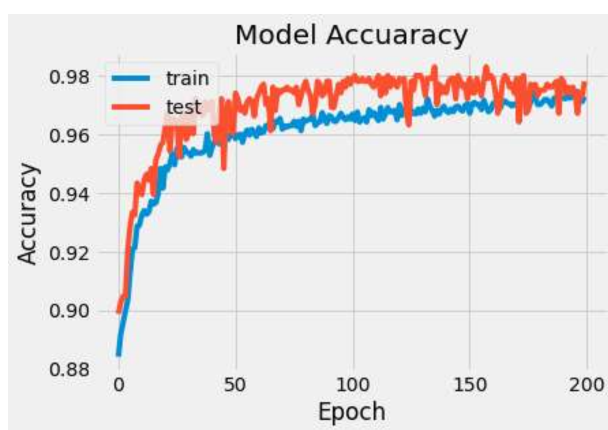


Figure 77. GRU Accuracy Curve for Slammer Dataset

```
Model Loss: 0.0234 - Model Accuracy: 0.9674
```

GRU accuracy for Slammer Dataset is slightly worse than CNN accuracy; the learning curves for GRU are less stable and less converged than the learning curves of the CNN.

- **Moscow Dataset**

The GRU model combination that worked best with the Moscow dataset was two GRU layers with 50 and 25 units. One Dense layer, RMSprop optimizer, and 100 epochs. The optimum accuracy in this scenario was 98.3%.

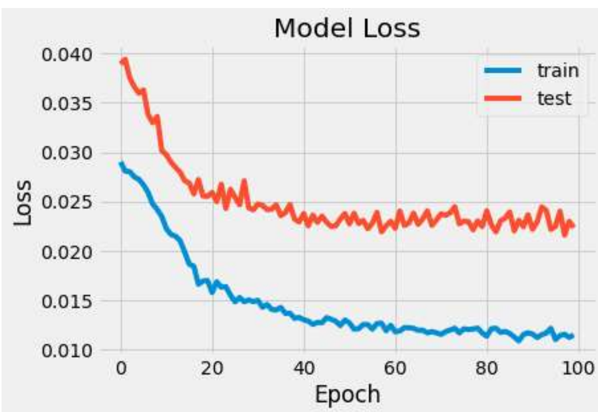


Figure 79. GRU Loss Curve for Moscow Dataset

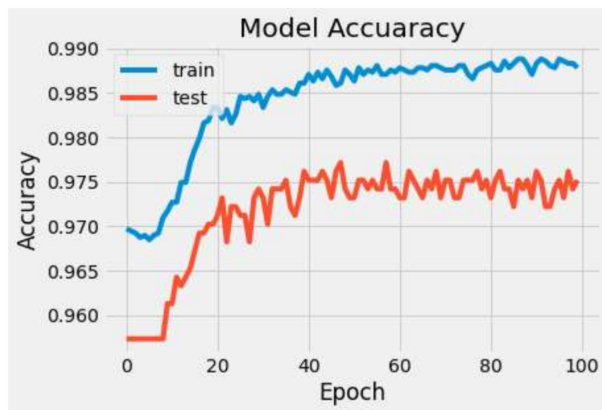


Figure 80. GRU Accuracy Curve for Moscow Dataset

```
Model Loss: 0.0157 - Model Accuracy: 0.9839
```

The behaviors of GRU and CNN models with Moscow blackout are similar; the accuracies for both models are 98%. The loss and accuracy curves show differences between the training and validation sets. Still, this variation is minimal, and the model performance is highly acceptable.

- **Code Red Dataset**

For Code Red Dataset, GRU scored the highest accuracy of 92.6% with a model consisting of one GRU lay er with 80 units, one dense layer, adam optimizer, and no. of epochs equal to 300.

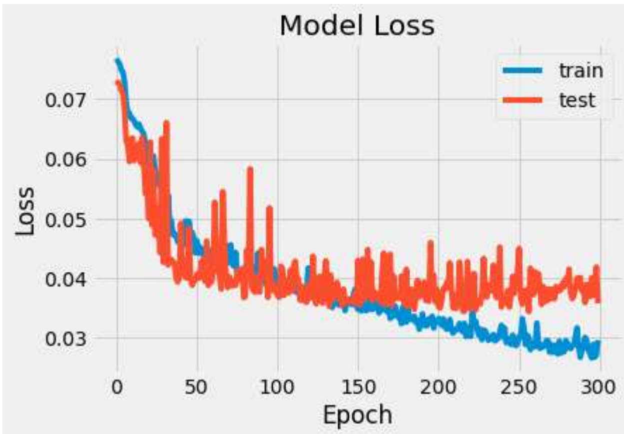


Figure 82. GRU Loss Curve for Code Red Dataset

Figure 81. GRU Accuracy Curve for Code Red Dataset

```
Model Loss: 0.0633 - Model Accuracy: 0.9269
```

The obtained accuracy for the Code Red dataset by the GRU model is noticeably worse than the one achi eved by the CNN model. The training and testing curves are diverging, and overfitting behavior is observ ed.

The following Graph illustrates the performances of GRU and CNN models with our BGP anomalies datas ets.

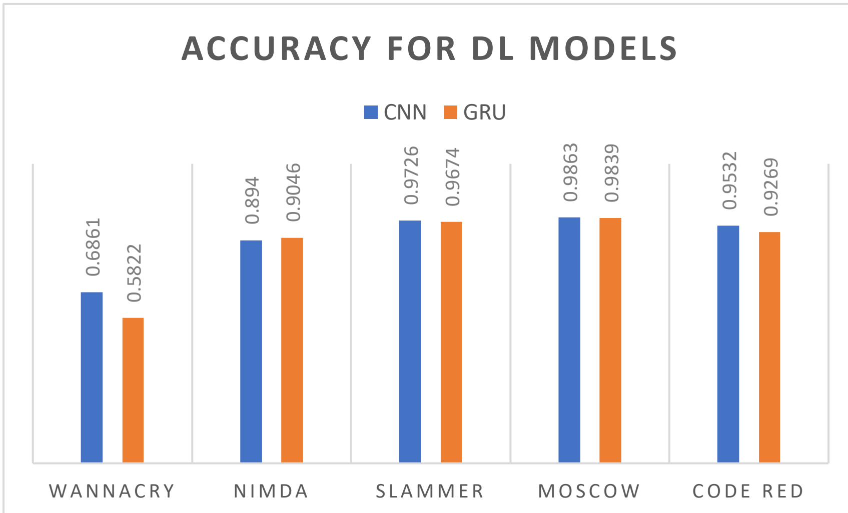


Figure 83. GRU and CNN models accuracy comparison

The results shown previously for the DL models make it possible to point out the following conclusions:

- There is no significant difference between GRU and CNN performances for Nimda, Slammer, and Moscow datasets; both models achieved decent accuracies.
- CNN outperformed GRU for both WannaCry and Code Red Dataset.
- GRU experienced an overfitting behavior for WannaCry and Code Red datasets, and hence it's not feasible in real-time detection scenarios, especially for events similar to WannaCry and Code Red.
- In the WannaCry dataset, the highest obtained accuracy by a DL model was 68.6% compared to m ore than 90% accuracy for other datasets, the same observation with ML models. Further investig ations are required in this context.
- Although GRU and CNN achieved good scores for several datasets, some plotted loss and accurac y curves indicate that multiple parameter tunings are needed to improve models' performances.

The following graph compares the ML and the DL models' accuracies.

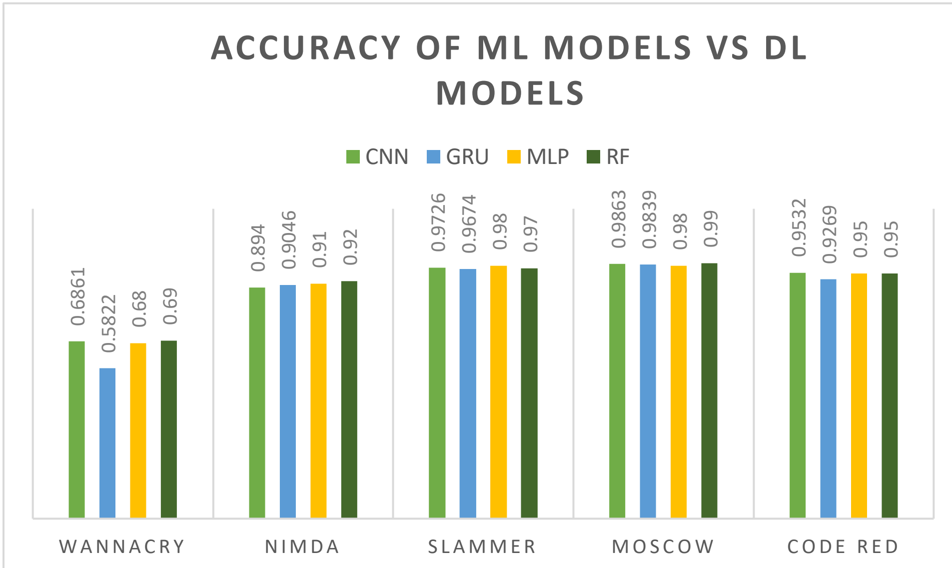


Figure 84. Comparison between DL and ML models classification accuracies

For simplicity, only the two most accurate ML models were included in this comparison.

Among CNN, GRU, MLP, and RF classification models, the MLP classifier achieved the highest accuracy for almost all the datasets. Nevertheless, there are insignificant differences in the achieved accuracies by ML models compared to DL modes.

Another conclusion is that the complexity level in our BGP classification problem does not demand deep learning tools, and machine learning models give appropriate accuracies. Moreover, since DL models are far more complex and more resource and time-consuming than ML models, it's more efficient to build anomaly detection tools based on ML.

5 Recommendations

5.1 Classification with Machine Learning Models

Six supervised ML models were tested on our five BGP anomaly datasets to determine the optimum classifier. The following recommendations are highlighted:

1- MLP-ANN classifier achieved the highest classification accuracies. Therefore, it's recommended to use MLP based classifier for developing BGP internet anomalies detection tools.
2- Unbalanced datasets pose real challenges for the ML classifiers. Hence, it's recommended to apply balancing mechanisms to help the model achieves balanced scores between the classes. Under-sampling the class with more data points is one of the most common approaches (Akbani, Kwek and Japkowicz, 2004). It's assumed by removing unnecessary replications from the majority class, the perceived generalization by the classifier for this class won't be affected (Pozzolo et al., 2015).
3- Although many efforts have been made in research to develop BGP anomaly detection tools using different datasets, no evident attention has been paid to WannaCry Dataset. With the highest achieved accuracy of 69%, it's of great interest to study the reason behind this performance by either examining more ML models or investigating the dataset itself.

5.2 Classification with Deep Learning Models

This work evaluated the classification accuracies of two robust deep learning models for our five datasets (CNN and GRU). The author's recommendations are as follows:

1- For WannaCry and Code Red datasets, it's recommended to use CNN for better classification accuracies. However, there is no significant difference in the achieved accuracies between CNN and GRU models for the remaining events.
2- Same as in ML models, DL models achieved the lowest accuracies for the WannaCry dataset compared to all remaining datasets. These results further strengthen our recommendations to investigate the WannaCry dataset and study the reason behind this poor performance.
3- The accuracies obtained by ML are pretty similar, if not higher than the accuracies obtained by the DL models. Therefore, it's recommended to use ML models rather than DL models in this binary classification problem for time and resource efficiency purposes.

6 Conclusion and Future Work

Internet services stability is a major concern for all network operators, and many efforts are being made to deal with this challenge. During the previous major internet incidents, researchers noticed that BGP routing updates experienced massive inconsistencies. Hence, BGP behavior is considered a good indicator of abnormal events happening on the internet. In this research, we investigated five well-known BGP datasets with BGP records before, during, and after five separate internet incidents (WannaCry, Nimda, Slammer, Moscow Blackout, and Code Red I). Our adopted methodology consisted of four data analytics levels: Descriptive Analysis, Inferential Analysis, Machine Learning, and Deep Learning.

- Descriptive Analysis results demonstrated that, for all different datasets, most BGP features distribution fit tend to have positive skewness and positive kurtosis. Only the WannaCry dataset is balanced, the remaining four datasets are imbalanced, and BGP records during regular periods are more than BGP records during the attack. Moreover, we plotted BGP features trends for each dataset. We confirmed that despite the evidence of some spikes during regular periods, BGP in all five datasets witnessed a noticeable change in multiple features during the anomaly.

- At the Inferential analysis level, we used one-way ANOVA testing to verify the following Null Hypothesis (H_0): The mean values of all BGP features in different internet anomaly events are equal. The results of one way ANOVA test reject the H_0 for all the datasets. Since ANOVA testing doesn't highlight which statistical sample is considerably different from others, we conducted two-sample t-tests to verify the same Null Hypotheses but this time for each pair of the datasets. T-test results showed that the highest relevance is between Code Red I and Nimda events. Out of 17 features, the mean values of 12 features during the anomaly period are equal between Code Red I and Nimda events. Moreover, we examined the correlations between BGP features by using Pearson correlation coefficients, and we found that there are not many high correlations between the 17 BGP features in each dataset. The majority of the strong correlations are positive, and negative correlations are very weak.

- At the third level of our analysis, we compared the accuracy of six different ML classifiers for each BGP anomaly dataset. Although there was no significant difference in the achieved accuracies of our six proposed ML models, MLP and RF classifiers achieved the highest F1-Scores for all the datasets. Our obtained F1-score are remarkably higher than many f1-scores found in the literature review. Another finding is that except for the balanced WannaCry dataset, there is a noticeable difference in F1-Scores between the regular and anomalous classes for all imbalanced datasets. However, for the WannaCry dataset, all ML classifiers achieved poor accuracy compared to other datasets.

- At our final analysis level, GRU and CNN classification models were applied to our datasets. We conclude that both models achieved similar accuracies for Nimda, Slammer, and Moscow datasets. However, CNN outperformed GRU for WannaCry and Code Red events. Moreover, Deep learning models experienced the same ML model mediocre performance on the WannaCry dataset.

- By comparing the accuracies of the ML models and the accuracies of the DL models, MLP achieved the optimum accuracy among all the tested models. We can conclude that the complexity level in

BGP anomaly datasets is not demanding deep learning algorithms. ML models give more appropriate results than DL models, and hence, it's more convenient to use.

In Future work, several investigations are intended as follows:

- For the WannaCry dataset, further investigations are planned to identify the reason behind the poor performance of the classification models.
- The imbalanced datasets problem is another challenge; balancing the datasets by under-sampling is one solution to be explored in the future.
- Although the performance of the suggested classifiers is satisfactory, additional Hyperparameter tunings for DL and ML are planned to enhance models performances and eliminate minimal obse rved inconsistencies in loss and accuracy curves.

References

8. k-Nearest Neighbor Classifier in Python | Machine Learning (no date). Available at: https://python-course.eu/machine-learning/k-nearest-neighbor-classifier-in-python.php (Accessed: 13 April 2022).

11th Annual USENIX Security Symposium — Technical Paper (no date). Available at: https://www.usenix.org/legacy/event/sec02/full_papers/staniford/staniford_html/ (Accessed: 16 March 2022).

Akbani, R., Kwek, S. and Japkowicz, N. (2004) 'Applying Support Vector Machines to Imbalanced Datasets', in Boulicaut, J.-F. et al. (eds) *Machine Learning: ECML 2004*. Berlin, Heidelberg: Springer (Lecture Notes in Computer Science), pp. 39–50. doi:10.1007/978-3-540-30115-8_7.

Allua, S. and Thompson, C.B. (2009) 'Inferential Statistics', *Air Medical Journal*, 28(4), pp. 168–171. doi:10.1016/j.amj.2009.04.013.

Al-Musawi, B., Branch, P. and Armitage, G. (2017) 'BGP Anomaly Detection Techniques: A Survey', *IEEE Communications Surveys Tutorials*, 19(1), pp. 377–396. doi:10.1109/COMST.2016.2622240.

Al-Rousan, N., Haeri, S. and Trajković, L. (2012) 'Feature selection for classification of BGP anomalies using Bayesian models', in *2012 International Conference on Machine Learning and Cybernetics*. *2012 International Conference on Machine Learning and Cybernetics*, pp. 140–147. doi:10.1109/ICMLC.2012.6358901.

Al-Rousan, N.M. and Trajković, L. (2012) 'Machine learning models for classification of BGP anomalies', in *2012 IEEE 13th International Conference on High Performance Switching and Routing*. *2012 IEEE 13th International Conference on High Performance Switching and Routing*, pp. 103–108. doi:10.1109/HPSR.2012.6260835.

Borgnat, P. *et al.* (2009) 'Seven Years and One Day: Sketching the Evolution of Internet Traffic', in *IEEE INFOCOM 2009*. *IEEE INFOCOM 2009*, pp. 711–719. doi:10.1109/INFCOM.2009.5061979.

Cheng, M. *et al.* (2016) 'MS-LSTM: A multi-scale LSTM model for BGP anomaly detection', in *2016 IEEE 24th International Conference on Network Protocols (ICNP)*. *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, pp. 1–6. doi:10.1109/ICNP.2016.7785326.

Cheng, M. *et al.* (2021) 'Multi-Scale LSTM Model for BGP Anomaly Classification', *IEEE Transactions on Services Computing*, 14(3), pp. 765–778. doi:10.1109/TSC.2018.2824809.

Conner, B. and Johnson, E. (no date) 'Use these tools to analyze data vital to practice-improvement projects.', p. 4.

Cosovic, M., Obradovic, S. and Junuz, E. (2018) ‘Deep Learning for Detection of BGP Anomalies’, in Rojas, I., Pomares, H., and Valenzuela, O. (eds) *Time Series Analysis and Forecasting*. Cham: Springer International Publishing, pp. 95–113. doi:10.1007/978-3-319-96944-2_7.

Dai, X., Wang, N. and Wang, W. (2019) ‘Application of machine learning in BGP anomaly detection’, *Journal of Physics: Conference Series*, 1176, p. 032015. doi:10.1088/1742-6596/1176/3/032015.

Ding, Q. *et al.* (2016) ‘Detecting BGP anomalies using machine learning techniques’, in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 003352–003355. doi:10.1109/SMC.2016.7844751.

Edwards, P., Cheng, L. and Kadam, G. (2019) ‘Border Gateway Protocol Anomaly Detection Using Machine Learning Techniques’, 2(1), p. 20.

Fonseca, P. *et al.* (2019) ‘BGP Dataset Generation and Feature Extraction for Anomaly Detection’, in *2019 IEEE Symposium on Computers and Communications (ISCC)*. *2019 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6. doi:10.1109/ISCC47284.2019.8969619.

Hoarau, K., Tournoux, P.U. and Razafindralambo, T. (2021) ‘Suitability of Graph Representation for BGP Anomaly Detection’, in *2021 IEEE 46th Conference on Local Computer Networks (LCN)*. *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, pp. 305–310. doi:10.1109/LCN52139.2021.9524941.

Index of /rrc04 (no date). Available at: https://data.ris.ripe.net/rrc04/ (Accessed: 15 March 2022).

Japkowicz, N. and Stephen, S. (2002) ‘The class imbalance problem: A systematic study’, *Intelligent Data Analysis*, 6(5), pp. 429–449. doi:10.3233/IDA-2002-6504.

Karimi, M. *et al.* (2019) ‘Border Gateway Protocol Anomaly Detection Using Neural Network’, in *2019 IEEE International Conference on Big Data (Big Data)*. *2019 IEEE International Conference on Big Data (Big Data)*, pp. 6092–6094. doi:10.1109/BigData47090.2019.9006201.

Kaur (no date) *Descriptive statistics*. Available at: https://www.ijam-web.org/article.asp?issn=2455-5568;year=2018;volume=4;issue=1;spage=60;epage=63;aulast=Kaur (Accessed: 19 March 2022).

Kolaczyk, E.D. and Csárdi, G. (2014) ‘Descriptive Analysis of Network Graph Characteristics’, in Kolaczyk, E.D. and Csárdi, G. (eds) *Statistical Analysis of Network Data with R*. New York, NY: Springer, pp. 43–67. doi:10.1007/978-1-4939-0983-4_4.

Kotsiantis, S.B., Zaharakis, I.D. and Pintelas, P.E. (2006) ‘Machine learning: a review of classification and combining techniques’, *Artificial Intelligence Review*, 26(3), pp. 159–190. doi:10.1007/s10462-007-9052-3.

Kurgan, L.A. and Musilek, P. (2006) ‘A survey of Knowledge Discovery and Data Mining process models’, *The Knowledge Engineering Review*, 21(1), pp. 1–24. doi:10.1017/S0269888906000737.

Lad, M. *et al.* (2003) ‘Analysis of BGP Update Surge during Slammer Worm Attack’, in Das, S.R. and Das, S.K. (eds) *Distributed Computing - IWDC 2003*. Berlin, Heidelberg: Springer, pp. 66–79. doi:10.1007/978-3-540-24604-6_7.

Li, Y. *et al.* (2014) ‘Classification of BGP anomalies using decision trees and fuzzy rough sets’, in *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1312–1317. doi:10.1109/SMC.2014.6974096.

Marnerides, A.K., Schaeffer-Filho, A. and Mauthe, A. (2014) ‘Traffic anomaly diagnosis in Internet backbone networks: A survey’, *Computer Networks*, 73, pp. 224–243. doi:10.1016/j.comnet.2014.08.007.

Marshall, G. and Jonker, L. (2011) ‘An introduction to inferential statistics: A review and practical guide’, *Radiography*, 17(1), pp. e1–e6. doi:10.1016/j.radi.2009.12.006.

McGlynn, K., Acharya, H.B. and Kwon, M. (2019) ‘Detecting BGP Route Anomalies with Deep Learning’, in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1039–1040. doi:10.1109/INFCOMW.2019.8845138.

Mohurle, S. and Patil, M. (2017) ‘A brief study of Wannacry Threat: Ransomware Attack 2017’. International Journal of Advanced Research in Computer Science. Available at: https://sbgsmedia.in/2018/05/10/2261f190e292ad93d6887198d7050dec.pdf.

Peng, S. *et al.* (2021) ‘A Multi-View Framework for BGP Anomaly Detection via Graph Attention Network’, *arXiv:2112.12793 [cs]* [Preprint]. Available at: http://arxiv.org/abs/2112.12793 (Accessed: 11 March 2022).

Phi, M. (2020) *Illustrated Guide to LSTM’s and GRU’s: A step by step explanation*, *Medium*. Available at: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21 (Accessed: 6 May 2022).

Pozzolo, A.D. *et al.* (2015) ‘Calibrating Probability with Undersampling for Unbalanced Classification’, in *2015 IEEE Symposium Series on Computational Intelligence*. *2015 IEEE Symposium Series on Computational Intelligence*, pp. 159–166. doi:10.1109/SSCI.2015.33.

Putina, A. *et al.* (2018) 'Unsupervised real-time detection of BGP anomalies leveraging high-rate and fine-grained telemetry data', in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1–2. doi:10.1109/INFCOMW.2018.8406838.

Rekhter, Y., Hares, S. and Li, T. (2006) *A Border Gateway Protocol 4 (BGP-4)*. Request for Comments RFC 4271. Internet Engineering Task Force. doi:10.17487/RFC4271.

Sanchez, O.R. *et al.* (2019) 'Comparing Machine Learning Algorithms for BGP Anomaly Detection using Graph Features', in *Proceedings of the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks*. New York, NY, USA: Association for Computing Machinery (Big-DAMA '19), pp. 35–41. doi:10.1145/3359992.3366640.

Shafique, U. and Qaiser, H. (no date) 'A Comparative Study of Data Mining Process Models (KDD, CRISP-DM and SEMMA)'.

Sharma, S., Osei-Bryson, K.-M. and Kasper, G.M. (2012) 'Evaluation of an integrated Knowledge Discovery and Data Mining process model', *Expert Systems with Applications*, 39(13), pp. 11335–11348. doi:10.1016/j.eswa.2012.02.044.

Siganos, G. and Faloutsos, M. (2004) 'Analyzing BGP policies: methodology and tool', in *IEEE INFOCOM 2004*. *IEEE INFOCOM 2004*, pp. 1640–1651 vol.3. doi:10.1109/INFCOM.2004.1354576.

sklearn.ensemble.RandomForestClassifier (no date) *scikit-learn*. Available at: https://scikit-learn/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (Accessed: 13 April 2022).

sklearn.model_selection.GridSearchCV (no date) *scikit-learn*. Available at: https://scikit-learn/stable/modules/generated/sklearn.model_selection.GridSearchCV.html (Accessed: 11 April 2022).

Trajkovic, L. (2020) 'Border Gateway Protocol (BGP) routing records from Reseaux IP Europeens (RIPE) and BCNET'. IEEE. Available at: https://ieee-dataport.org/open-access/border-gateway-protocol-bgp-routing-records-reseaux-ip-europeens-ripe-and-bcnet (Accessed: 11 March 2022).

Voropai, N.I. and Efimov, D.N. (2008) 'Analysis of blackout development mechanisms in electric power systems', in *2008 IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century*. *2008 IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century*, pp. 1–7. doi:10.1109/PES.2008.4596304.

What are Convolutional Neural Networks? (2021). Available at: https://www.ibm.com/cloud/learn/convolutional-neural-networks (Accessed: 6 May 2022).

Wübbeling, M., Elsner, T. and Meier, M. (2014) ‘Inter-AS routing anomalies: Improved detection and classification’, in *2014 6th International Conference On Cyber Conflict (CyCon 2014)*. *2014 6th International Conference On Cyber Conflict (CyCon 2014)*, pp. 223–238. doi:10.1109/CYCON.2014.6916405.

Xu, M. and Li, X. (2020) ‘BGP Anomaly Detection Based on Automatic Feature Extraction by Neural Network’, in *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*. *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*, pp. 46–50. doi:10.1109/ITOEC49072.2020.9141762.

Appendix

Codes for Level 1: Descriptive Analysis

```
#Check Datatypes
import pandas as pd

pd.set_option("display.max_columns", None)

WannaCrypt=pd.read_csv('WannaCrypt.csv')
Nimda=pd.read_csv('Nimda.csv')
Slammer=pd.read_csv('Slammer.csv')
Moscow_blackout = pd.read_csv('Moscow_blackout.csv')
Code_Red_I = pd.read_csv('Code_Red_I.csv')

print("Data types for WannaCrypt Anomaly are:")
print(WannaCrypt.dtypes.value_counts())
print('\n')
print("Data types for Nimda Anomaly are:")
print(Nimda.dtypes.value_counts())
print('\n')
print("Data types for Slammer Anomaly are:")
print(Slammer.dtypes.value_counts())
print('\n')
print("Data types for Moscow_blackout Anomaly are:")
print(Moscow_blackout.dtypes.value_counts())
print('\n')
print("Data types for Code_Red_I Anomaly are:")
print(Code_Red_I.dtypes.value_counts())
```

```
#Check missing Data
import pandas as pd


pd.set_option("display.max_columns", None)

WannaCrypt=pd.read_csv('WannaCrypt.csv')#sep=';',na_values=".")
Nimda=pd.read_csv('Nimda.csv')
Slammer=pd.read_csv('Slammer.csv')
Moscow_blackout = pd.read_csv('Moscow_blackout.csv')
Code_Red_I = pd.read_csv('Code_Red_I.csv')


print("Is there any missing data for WannaCrypt")
print(WannaCrypt.isnull().sum())
print('\n')
print("Is there any missing data for Nimda")
print(Nimda.isnull().sum())
print('\n')
print("Is there any missing data for Slammer")
print(Slammer.isnull().sum())
print('\n')
print("Is there any missing data for Moscow_blackout")
print(Moscow_blackout.isnull().sum())
print('\n')
print("Is there any missing data for Code_Red_I")
print(Code_Red_I.isnull().sum())
```

```
#WannaCry head and tail expolartion
import pandas as pd
WannaCrypt=pd.read_csv('WannaCrypt.csv')
WannaCrypt.head()
```

```
#WannaCry head and tail expolartion
import pandas as pd
WannaCrypt=pd.read_csv('WannaCrypt.csv')
WannaCrypt.tail()

```

```
#Statistical measures for WannaCry
import pandas as pd
import matplotlib.pyplot as plt

WannaCrypt=pd.read_csv('WannaCrypt.csv')
dataset=WannaCrypt.groupby("Classification")
print("Descriptive Statistics for WannaCrypt")
WannaCrypt[["NOA", "NOW", "NOANP", "NOWNP", "AAPL", "MAPL", "AUAPL", "NODA",
            "NODW", "NOIW", "AED", "MED", "IAT", "IGP packets", "EGP packets",
            "Incomplete packets", "packet size"]].describe().to_clipboard()
dataset[["Classification"]].agg(['count'])

```

```
#Statistical measures for Nimda
import pandas as pd
Nimda=pd.read_csv('Nimda.csv')
dataset=Nimda.groupby("Classification")
print("Descriptive Statistics for Nimda")
Nimda[["NOA", "NOW", "NOANP", "NOWNP", "AAPL", "MAPL", "AUAPL", "NODA",
       "NODW", "NOIW", "AED", "MED", "IAT", "IGP packets", "EGP packets",
       "Incomplete packets", "packet size"]].describe().to_clipboard()
dataset[["Classification"]].agg(['count'])
```

```
#Statistical measures for Slammer
import pandas as pd
Slammer=pd.read_csv('Slammer.csv')
dataset=Slammer.groupby("Classification")
print("Descriptive Statistics for Slammer")
Slammer[["NOA", "NOW", "NOANP", "NOWNP", "AAPL", "MAPL", "AUAPL", "NODA",
         "NODW", "NOIW", "AED", "MED", "IAT", "IGP packets", "EGP packets",
         "Incomplete packets", "packet size"]].describe().to_clipboard()
dataset[["Classification"]].agg(['count'])
```

```
#Statistical measures for Moscow Blackout
import pandas as pd
Moscow_blackout= pd.read_csv('Moscow_blackout.csv')
dataset=Moscow_blackout.groupby("Classification")
print("Descriptive Statistics for Moscow_blackout")
Moscow_blackout[["NOA", "NOW", "NOANP", "NOWNP", "AAPL", "MAPL", "AUAPL", "NODA",
                 "NODW", "NOIW", "AED", "MED", "IAT", "IGP packets", "EGP packets",
                 "Incomplete packets", "packet size"]].describe().to_clipboard()
dataset[["Classification"]].agg(['count'])
```

```
#Statistical measures for Code Red
import pandas as pd
Code_Red_I=pd.read_csv('Code_Red_I.csv')
dataset=Code_Red_I.groupby("Classification")
print("Descriptive Statistics for Code_Red_I")
Code_Red_I[["NOA", "NOW", "NOANP", "NOWNP", "AAPL", "MAPL", "AUAPL", "NODA",
            "NODW", "NOIW", "AED", "MED", "IAT", "IGP packets", "EGP packets",
            "Incomplete packets", "packet size"]].describe().to_clipboard()
dataset[["Classification"]].agg(['count'])
```

```
#Skewness, Kurotosis, and Histograms for WannaCry
WannaCrypt=pd.read_csv('WannaCrypt.csv')
WannaCrypt[["NOA", "NOW", "NOANP", "NOWNP", "AAPL", "MAPL", "AUAPL", "NODA", "NODW",
            "NOIW", "AED", "MED", "IAT", "IGP packets", "EGP packets", "Incomplete packets",
            "packet size"]].hist(figsize=(40, 25), bins=200, linewidth='1',edgecolor='k',grid=False)
print(WannaCrypt[["NOA", "NOW", "NOANP", "NOWNP", "AAPL", "MAPL", "AUAPL", "NODA", "NODW",
                  "NOIW", "AED", "MED", "IAT", "IGP packets", "EGP packets", "Incomplete packets",
                  "packet size"]].skew())
print(WannaCrypt[["NOA", "NOW", "NOANP", "NOWNP", "AAPL", "MAPL", "AUAPL", "NODA", "NODW",
                  "NOIW", "AED", "MED", "IAT", "IGP packets", "EGP packets", "Incomplete packets",
                  "packet size"]].kurt())

```

```python
#Skewness, Kurotosis, and Histograms for Nimda
import pandas as pd
Nimda=pd.read_csv('Nimda.csv')
Nimda[["NOA", "NOW", "NOANP", "NOWNP", "AAPL", "MAPL", "AUAPL", "NODA", "NODW",
       "NOIW", "AED", "MED", "IAT", "IGP packets", "EGP packets", "Incomplete packets",
       "packet size"]].hist(figsize=(40, 25), bins=100, linewidth='1',edgecolor='k',grid=False)
print(Nimda[["NOA", "NOW", "NOANP", "NOWNP", "AAPL", "MAPL", "AUAPL", "NODA", "NODW", \
             "NOIW", "AED", "MED", "IAT", "IGP packets", "EGP packets", "Incomplete packets",
             "packet size"]].skew())
print(Nimda[["NOA", "NOW", "NOANP", "NOWNP", "AAPL", "MAPL", "AUAPL", "NODA", "NODW",
             "NOIW", "AED", "MED", "IAT", "IGP packets", "EGP packets", "Incomplete packets",
             "packet size"]].kurt())
```

```python
#Skewness, Kurotosis, and Histograms for Slammer
Slammer=pd.read_csv('Slammer.csv')
Slammer[["NOA", "NOW", "NOANP", "NOWNP", "AAPL", "MAPL", "AUAPL", "NODA", "NODW",
         "NOIW", "AED", "MED", "IAT", "IGP packets", "EGP packets", "Incomplete packets",
         "packet size"]].hist(figsize=(40, 25), bins=200, linewidth='1',edgecolor='k',grid=False)
print(Slammer[["NOA", "NOW", "NOANP", "NOWNP", "AAPL", "MAPL", "AUAPL", "NODA", "NODW",
               "NOIW", "AED", "MED", "IAT", "IGP packets", "EGP packets", "Incomplete packets",
               "packet size"]].skew())
print(Slammer[["NOA", "NOW", "NOANP", "NOWNP", "AAPL", "MAPL", "AUAPL", "NODA", "NODW",
               "NOIW", "AED", "MED", "IAT", "IGP packets", "EGP packets", "Incomplete packets",
               "packet size"]].kurt())
```

```python
#Skewness, Kurotosis, and Histograms for Moscow
Moscow_blackout=pd.read_csv('Moscow_blackout.csv')
Moscow_blackout[["NOA", "NOW", "NOANP", "NOWNP", "AAPL", "MAPL", "AUAPL", "NODA", "NODW",
                 "NOIW", "AED", "MED", "IAT", "IGP packets", "EGP packets", "Incomplete packets",
                 "packet size"]].hist(figsize=(40, 25), bins=200, linewidth='1',edgecolor='k',grid=False)
print(Moscow_blackout[["NOA", "NOW", "NOANP", "NOWNP", "AAPL", "MAPL", "AUAPL", "NODA", "NODW",
                       "NOIW", "AED", "MED", "IAT", "IGP packets", "EGP packets", "Incomplete packets",
                       "packet size"]].skew())
print(Moscow_blackout[["NOA", "NOW", "NOANP", "NOWNP", "AAPL", "MAPL", "AUAPL", "NODA", "NODW",
                       "NOIW", "AED", "MED", "IAT", "IGP packets", "EGP packets", "Incomplete packets",
                       "packet size"]].kurt())
```

```python
#Skewness, Kurotosis, and Histograms for Code Red
Code_Red_I=pd.read_csv('Code_Red_I.csv')
Code_Red_I[["NOA", "NOW", "NOANP", "NOWNP", "AAPL", "MAPL", "AUAPL", "NODA", "NODW",
            "NOIW", "AED", "MED", "IAT","IGP packets", "EGP packets", "Incomplete packets",
            "packet size"]].hist(figsize=(40, 25), bins=200, linewidth='1',edgecolor='k',grid=False)
print(Code_Red_I[["NOA", "NOW", "NOANP", "NOWNP", "AAPL", "MAPL","AUAPL", "NODA", "NODW",
                  "NOIW", "AED", "MED", "IAT","IGP packets", "EGP packets", "Incomplete packets",
                  "packet size"]].skew())
print(Code_Red_I[["NOA", "NOW", "NOANP", "NOWNP", "AAPL", "MAPL", "AUAPL", "NODA", "NODW",
                  "NOIW", "AED", "MED", "IAT", "IGP packets", "EGP packets", "Incomplete packets",
                  "packet size"]].kurt())
```

```python
#BGP Features Trend before, during and after th edifferent anomalies
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm

WannaCrypt=pd.read_csv('WannaCrypt.csv')
Moscow_blackout = pd.read_csv('Moscow_blackout1.csv')
Nimda=pd.read_csv('Nimda1.csv')
Slammer=pd.read_csv('Slammer1.csv')
Code_Red_I = pd.read_csv('Code_Red_I1.csv')


Datasets = [WannaCrypt, Nimda,Slammer, Moscow_blackout,Code_Red_I ]
names = ['WannaCrypt', 'Nimda','Slammer', 'Moscow_blackout','Code_Red_I' ]
start = ['2017-05-10', '2001-09-16', '2003-01-23', '2005-05-23', '2001-07-13']
for i in range (len(Datasets)):

    dataset=Datasets[i][["NOA", "NOW", "NOANP", "NOWNP", "AAPL", "MAPL", "AUAPL", "NODA", "NODW",
                        "NOIW", "AED", "MED", "IAT", "IGP packets", "EGP packets", "Incomplete packets",
                        "packet size"]]
    index=pd.date_range(start=start[i], periods=dataset.NOA.count(), freq='min')
    dataset.insert(0, 'TimeStamp', index)
    dataset.tail()
    dataset.set_index('TimeStamp')
    month_average=dataset.resample('60min',on='TimeStamp').mean()
    month_average.head()

    plt.figure()
    month_average.plot(kind="line", figsize=(18, 9))
    plt.xlabel(names[i] + 'Dataset Collection Period', fontsize=12)
    plt.title('Trends of features', fontsize=16)
    #plt.ylabel('Unemployment Rate', fontsize=14)
    plt.grid(True)
    plt.legend(title="features",loc='upper right', fontsize=8, fancybox=True)

    plt.show()
```

Codes for Level 2: Inferential Statistics

```python
#Oneway-Anova test
import pandas as pd
import scipy.stats as stats

WannaCrypt=pd.read_csv('WannaCrypt.csv')
Nimda=pd.read_csv('Nimda.csv')
Slammer=pd.read_csv('Slammer.csv')
Moscow_blackout = pd.read_csv('Moscow_blackout.csv')
Code_Red_I = pd.read_csv('Code_Red_I.csv')

Array = ['NOA', 'NOW', 'NOANP', 'NOWNP', 'AAPL', 'MAPL', 'AUAPL', 'NODA', 'NODW',
        'NOIW', 'AED', 'MED', 'IAT', 'IGP packets', 'EGP packets', 'Incomplete packets', 'packet size']
for i in range (len(Array)):

    print('\033[1m'+'One-way Anova testing for '+Array[i]+'\033[0m')
    fvalue, pvalue = stats.f_oneway(WannaCrypt[Array[i]][WannaCrypt['Classification'] == 1],
                                    Nimda[Array[i]][Nimda['Classification'] == 1],
                                    Slammer[Array[i]][Slammer['Classification'] == 1],
                                    Moscow_blackout[Array[i]][Moscow_blackout['Classification'] == 1],
                                    Code_Red_I[Array[i]][Code_Red_I['Classification'] == 1])
    print("F Value={:g} ".format(fvalue) +','+ "P Value={:g} ".format(pvalue))

    alpha = 0.05

    if pvalue < alpha:  # null hypothesis: x comes from a normal distribution
        print("The null hypothesis can be rejected")
    else:
        print("The null hypothesis is accepted")
    print('\n')
```

```python
#Two-way T-tests
import pandas as pd
from scipy import stats

WannaCrypt=pd.read_csv('WannaCrypt.csv')
Nimda=pd.read_csv('Nimda.csv')
Slammer=pd.read_csv('Slammer.csv')
Moscow_blackout = pd.read_csv('Moscow_blackout.csv')
Code_Red_I = pd.read_csv('Code_Red_I.csv')

Datasets = [WannaCrypt, Nimda, Slammer, Moscow_blackout, Code_Red_I]
Array = ['NOA', 'NOW', 'NOANP', 'NOWNP', 'AAPL', 'MAPL', 'AUAPL', 'NODA', 'NODW',
         'NOIW', 'AED', 'MED', 'IAT', 'IGP packets', 'EGP packets', 'Incomplete packets', 'packet size']
Names = ['WannaCrypt', 'Nimda', 'Slammer', 'Moscow_blackout', 'Code_Red_I']

for i in range(len(Datasets)):

    for j in range(i+1, len(Datasets)):
        print('\033[1m'+"Two-way T-test for fetures in "+Names[i]+" and "+Names[j]+'\033[0m')
        for k in range(len(Array)):
            A_Classified= Datasets[i][Datasets[i]['Classification'] == 1][Array[k]]
            B_Classified = Datasets[j][Datasets[j]['Classification'] == 1][Array[k]]
            t2, p =stats.ttest_ind(A_Classified, B_Classified)
            #two-tail 2-sample t-test
            alpha_half = 0.025 #alpha is 0.05 or level of confidence is 95%

            print("p value={:g}".format(p)+','+" t value={:g}". format(t2))


            if p < alpha_half:  # null hypothesis: x comes from a normal distribution
                print("The null hypothesis for "+ Array[k]+" can be rejected")
            else:
                print("The null hypothesis for "+ Array[k]+" is accepted")
            print('\n')

```

```python
#Correlations
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

WannaCrypt=pd.read_csv('WannaCrypt.csv')
Nimda=pd.read_csv('Nimda.csv')
Slammer=pd.read_csv('Slammer.csv')
Moscow_blackout = pd.read_csv('Moscow_blackout.csv')
Code_Red_I = pd.read_csv('Code_Red_I.csv')

Datasets = [WannaCrypt, Nimda,Slammer, Moscow_blackout,Code_Red_I ]
names = ['WannaCrypt', 'Nimda','Slammer', 'Moscow_blackout','Code_Red_I' ]

for i in range(len(Datasets)):

    df = pd.DataFrame(Datasets[i], columns= ['NOA', 'NOW', 'NOANP', 'NOWNP', 'AAPL', 'MAPL', 'AUAPL',
                                             'NODA', 'NODW', 'NOIW', 'AED', 'MED', 'IAT', 'IGP packets',
                                             'EGP packets', 'Incomplete packets', 'packet size'])

    #create a correlation matrix for all the column sets except the target variable
    correlation = df.corr()
    mask = np.triu(np.ones_like(correlation, dtype=bool))
    f, ax = plt.subplots(figsize=(16, 12))
    sns.set_style("white")
    cmap = sns.diverging_palette(230, 20, as_cmap=True)
    heatmap = sns.heatmap(correlation, mask = mask, cmap=cmap, vmin = -1,
                          vmax = 1, center=0,
            square=True, cbar_kws={"shrink": .8, 'extend':'both'},
            annot = True, annot_kws = {"size": 12})
    plt.title("Heatmap of "+names[i]+" Dataset", fontsize = 16)
    sns.set_style({'xtick.bottom': True}, {'ytick.left': True})

```

P-values for all t-tests

	P-Value									
BGP Features	**WannaCry and Nimda**	**WannaCry and Slammer**	**WannaCry and Moscow**	**WannaCry and Code Red**	**Nimda and Slammer**	**Nimda and Moscow**	**Nimda and Code Red**	**Slammer and Moscow**	**Slammer and Code Red**	**Moscow and Code Red**
NOA	4.3986E-199	6.33E-140	5.4502E-123	4.98E-140	0.00784274	8.51E-118	0.262089*	3.64E-128	2.72E-23	4.10E-104
NOW	0	0	0	0	0.0806049*	0.00659362	0.821473*	5.04E-128	1.92E-19	3.81E-23
NOANP	1.69406E-81	3.76693E-51	3.7576E-157	4.52E-44	0.0367692*	2.58E-130	0.852812*	3.61E-114	4.43E-05	1.95E-84
NOWNP	0.123339*	0.00640074	4.23113E-80	0.0494736*	0.238301*	2.46E-07	0.891106*	9.30E-22	8.57E-06	1.16E-17
AAPL	3.2603E-07	2.18162E-40	1.9625E-104	2.39E-08	1.34E-15	9.67E-95	0.0720483*	1.39E-106	1.91E-06	8.01E-67
MAPL	1.13934E-76	1.33325E-36	2.08679E-09	3.13E-57	2.18E-14	2.07E-10	2.43E-23	0.0248039	1.29E-68	4.64E-37
AUAPL	0.771527*	2.56905E-30	8.2528E-144	0.00641026	2.71E-19	4.79E-94	0.0323574*	3.81E-101	1.41E-07	7.97E-67
NODA	0	1.286E-235	0	3.81E-175	8.16E-34	1.25E-194	0.0927043*	1.37E-131	3.82E-20	4.72E-94
NODW	1.9844E-228	3.2213E-148	4.96038E-36	2.11E-106	1.07E-34	1.77E-84	9.40E-12	6.62E-30	0.0117549	9.55E-26
NOIW	7.60457E-40	2.46704E-19	4.01547E-12	2.11E-18	5.28E-103	1.46E-56	0.000195379	1.83E-33	2.84E-29	1.20E-26
AED	1.09838E-76	1.3709E-36	2.06197E-09	4.43E-57	1.98E-14	2.14E-10	4.86E-23	0.025354*	2.06E-68	6.65E-37
MED	2.50421E-35	4.8931E-32	3.20324E-41	1.54E-21	0.990878*	1.64E-22	0.979916*	1.14E-22	0.986665*	6.27E-15
IAT	1.9146E-99	5.48702E-31	0	1.42E-54	2.86E-09	1.69E-135	0.829325*	9.90E-150	7.21E-08	4.31E-91
IGP packets	0	6.7321E-254	1.39501E-84	3.43E-232	0.00452592	1.80E-125	0.257813*	8.40E-130	1.80E-24	7.75E-105
EGP packets	1.07676E-05	0.223529*	0.930147*	0.00432156	8.06E-06	0.000100721	0.690291*	0.210769*	5.26E-05	3.41E-06
Incomplete packets	1.6706E-109	5.18567E-91	7.76474E-07	7.92E-76	0.721668*	7.83E-07	0.309529*	4.45E-24	2.16E-05	2.54E-26
packet size	0	0	1.68386E-50	2.21E-57	0.942493*	1.80E-10	1.07E-15	5.18E-18	1.40E-14	0.45621*

Codes for Level 3: Machine Learning

```
#MLP Classifier

import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.metrics import plot_confusion_matrix
from sklearn.model_selection import GridSearchCV

WannaCrypt=pd.read_csv('WannaCrypt.csv')
Nimda=pd.read_csv('Nimda.csv')
Slammer=pd.read_csv('Slammer.csv')
Moscow_blackout = pd.read_csv('Moscow_blackout.csv')
Code_Red_I = pd.read_csv('Code_Red_I.csv')

Datasets = [WannaCrypt, Nimda,Slammer, Moscow_blackout,Code_Red_I ]
names = ['WannaCrypt', 'Nimda','Slammer', 'Moscow_blackout','Code_Red_I' ]

for i in range(len(Datasets)):

    X = Datasets[i].drop(['H+M','H','M','S','MED1','MED2','MED3','MED4','MED5','MED6','MED7','MED8','MED9',
                          'MED10','MED11','MAL1','MAL2','MAL3','MAL4','MAL5','MAL6','MAL7','MAL8','MAL9',
                          'Classification'],axis=1)
    y = Datasets[i]['Classification']

    #Train, test and split the dataset. Random number generator, with popular integer see numbers are 0 and 42
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

    #Pre-processing - transformation, etc...
    scaler = StandardScaler()

    # Fit only to the training data
    scaler.fit(X_train)

    # Now apply the transformations to the data:
    X_train = scaler.transform(X_train)
    X_test = scaler.transform(X_test)
```

```
parameter_space = {
'activation' : ['identity', 'logistic', 'tanh', 'relu'],
'solver' : ['lbfgs', 'sgd', 'adam'] }

#Create an MLP model
clf = GridSearchCV(MLPClassifier(max_iter=2000), parameter_space, n_jobs=-1, cv=3)

#Fit the model
classifier = clf.fit(X_train,y_train)
#Prediction
y_pred = clf.predict(X_test)

print('Best parameters found for', names[i],'Dataset :\n', clf.best_params_)

#Model Evaluation
print("Confusion Matrix is ")
print(confusion_matrix(y_test, y_pred))
print("\n")
print(classification_report(y_test, y_pred))
print("\n")

# Plot non-normalized confusion matrix
titles_options = [("Confusion matrix, without normalization for "+ names[i] + " Dataset", None),
                  ("Normalized confusion matrix for "+ names[i] + " Dataset", 'true')]
for title, normalize in titles_options:
    disp = plot_confusion_matrix(classifier, X_test, y_test,
                                 cmap=plt.cm.Blues,
                                 normalize=normalize)
    disp.ax_.set_title(title)

    print(title)
    print(disp.confusion_matrix)

plt.show()
print(':\n')
```

```
#Decision Tree Classifier
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
from sklearn.metrics import confusion_matrix, classification_report, plot_confusion_matrix
from matplotlib import pyplot as plt
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.model_selection import GridSearchCV

#Load dataset and explore dataset
WannaCrypt=pd.read_csv('WannaCrypt.csv')
Nimda=pd.read_csv('Nimda.csv')
Slammer=pd.read_csv('Slammer.csv')
Moscow_blackout = pd.read_csv('Moscow_blackout.csv')
Code_Red_I = pd.read_csv('Code_Red_I.csv')

Datasets = [WannaCrypt, Nimda,Slammer, Moscow_blackout,Code_Red_I ]
names = ['WannaCrypt', 'Nimda','Slammer', 'Moscow_blackout','Code_Red_I' ]

for i in range(len(Datasets)):

    #Feature selection: split the dataset into features (independent variables) and target (dependent variable)
    X = Datasets[i].drop(['H+M','H','M','S','MED1','MED2','MED3','MED4','MED5','MED6','MED7',
                          'MED8','MED9','MED10','MED11','MAL1','MAL2','MAL3','MAL4','MAL5','MAL6',
                          'MAL7','MAL8','MAL9','Classification'],axis=1)
    y = Datasets[i]['Classification']

    # Split dataset into training set and test set,70% training and 30% test
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)

    parameter_space = {
        'criterion': ['gini', 'entropy'] }

    # Create Decision Tree classifer object
    clf = GridSearchCV(DecisionTreeClassifier(random_state=1234), parameter_space, n_jobs=-1, cv=3)

    # Train Decision Tree Classifer
    clf = clf.fit(X_train,y_train)

    #Predict the response for test dataset
    y_pred = clf.predict(X_test)

    print('Best parameters found for', names[i],'Dataset :\n', clf.best_params_)

    # Model Accuracy
    print("\n")
    print("Accuracy for 70% training set and 30% test set :",
              metrics.accuracy_score(y_test, y_pred))

    #Confusion matrix
    print("Confusion Matrix is")
    print(confusion_matrix(y_test, y_pred))
    print("\n")
    print(classification_report(y_test, y_pred))
    print("\n")

        # Plot non-normalized confusion matrix
    titles_options = [("Confusion matrix, without normalization for "+ names[i] + " Dataset", None),
                      ("Normalized confusion matrix for "+ names[i] + " Dataset", 'true')]
    for title, normalize in titles_options:
        disp = plot_confusion_matrix(clf, X_test, y_test,
                                     #display_labels=class_names,
                                     cmap=plt.cm.Blues,
                                     normalize=normalize)
        disp.ax_.set_title(title)

        print(title)
        print(disp.confusion_matrix)

    plt.show()
    print(':\n')
```

```python
# KNN classifier

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix, plot_confusion_matrix
from sklearn.model_selection import GridSearchCV

%matplotlib inline

#Import the data set
WannaCrypt=pd.read_csv('WannaCrypt.csv')
Nimda=pd.read_csv('Nimda.csv')
Slammer=pd.read_csv('Slammer.csv')
Moscow_blackout = pd.read_csv('Moscow_blackout.csv')
Code_Red_I = pd.read_csv('Code_Red_I.csv')

for i in range(len(Datasets)):
    Datasets = [WannaCrypt, Nimda,Slammer, Moscow_blackout,Code_Red_I ]
    names = ['WannaCrypt', 'Nimda','Slammer', 'Moscow_blackout','Code_Red_I' ]

    dataset = Datasets[i].drop(['H+M','H','M','S','MED1','MED2','MED3','MED4','MED5','MED6','MED7',
                                'MED8','MED9','MED10','MED11','MAL1','MAL2','MAL3','MAL4','MAL5','MAL6',
                                'MAL7','MAL8','MAL9','Classification'],axis=1)
    #Standardize the data set
    scaler = StandardScaler()
    scaler.fit(dataset)
    scaled_features = scaler.transform(dataset)
    scaled_data = pd.DataFrame(scaled_features, columns = dataset.columns)

    parameter_space = {
        'n_neighbors': list(range(1,100)),
        'weights': ['uniform', 'distance'] }

    #Split the data set into training data and test data
    X= scaled_data
    y = Datasets[i]['Classification']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3)

    #Train the model and make predictions
    model = GridSearchCV(KNeighborsClassifier(), parameter_space, n_jobs=-1, cv=3)
    #model = KNeighborsClassifier(n_neighbors = 1)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    print('Best parameters found for ',names[i],' Dataset :\n', model.best_params_)

    #Performance measurement
    print(classification_report(y_test, y_pred))
    print(confusion_matrix(y_test, y_pred))

    titles_options = [("Confusion matrix, without normalization for "+ names[i] + " Dataset", None),
                      ("Normalized confusion matrix for "+ names[i] + " Dataset", 'true')]
    for title, normalize in titles_options:
        disp = plot_confusion_matrix(model, X_test, y_test,
                                     #display_labels=class_names,
                                     cmap=plt.cm.Blues,
                                     normalize=normalize)
        disp.ax_.set_title(title)

        print(title)
        print(disp.confusion_matrix)

    plt.show()
    print(':\n')
```

```
#Suppor Vector MAchines Classifier
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix, plot_confusion_matrix
from sklearn.model_selection import GridSearchCV


#Import the dataset
WannaCrypt=pd.read_csv('WannaCrypt.csv')
Nimda=pd.read_csv('Nimda.csv')
Slammer=pd.read_csv('Slammer.csv')
Moscow_blackout = pd.read_csv('Moscow_blackout.csv')
Code_Red_I = pd.read_csv('Code_Red_I.csv')

for i in range(len(Datasets)):
    Datasets = [WannaCrypt, Nimda,Slammer, Moscow_blackout,Code_Red_I ]
    names = ['WannaCrypt', 'Nimda','Slammer', 'Moscow_blackout','Code_Red_I' ]

    dataset = Datasets[i].drop(['H+M','H','M','S','MED1','MED2','MED3','MED4','MED5','MED6','MED7',
                                'MED8','MED9','MED10','MED11','MAL1','MAL2','MAL3','MAL4','MAL5','MAL6',
                                'MAL7','MAL8','MAL9','Classification'],axis=1)
    #Standardize the data set
    scaler = StandardScaler()
    scaler.fit(dataset)
    scaled_features = scaler.transform(dataset)
    scaled_data = pd.DataFrame(scaled_features, columns = dataset.columns)

    parameter_space = {
        'kernel': ['linear', 'poly', 'rbf', 'sigmoid'] }

    #Split the data set into training data and test data
    X= scaled_data
    y = Datasets[i]['Classification']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3)

    #Train the model and make predictions
    model = GridSearchCV(SVC(), parameter_space, n_jobs=-1, cv=3)
    #model = KNeighborsClassifier(n_neighbors = 1)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)


    print('Best parameters found for ',names[i],' Dataset :\n', model.best_params_)

    #Performance measurement
    print(classification_report(y_test, y_pred))
    print(confusion_matrix(y_test, y_pred))

    titles_options = [("Confusion matrix, without normalization for "+ names[i] + " Dataset", None),
                      ("Normalized confusion matrix for "+ names[i] + " Dataset", 'true')]
    for title, normalize in titles_options:
        disp = plot_confusion_matrix(model, X_test, y_test,
                                     cmap=plt.cm.Blues,
                                     normalize=normalize)
        disp.ax_.set_title(title)

        print(title)
        print(disp.confusion_matrix)

    plt.show()
    print(':\n')
```

```
# Random Forests Classifier
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix, plot_confusion_matrix
from sklearn.model_selection import GridSearchCV

#Import the dataset
WannaCrypt=pd.read_csv('WannaCrypt.csv')
Nimda=pd.read_csv('Nimda.csv')
Slammer=pd.read_csv('Slammer.csv')
Moscow_blackout = pd.read_csv('Moscow_blackout.csv')
Code_Red_I = pd.read_csv('Code_Red_I.csv')

for i in range(len(Datasets)):
    Datasets = [WannaCrypt, Nimda,Slammer, Moscow_blackout,Code_Red_I ]
    names = ['WannaCrypt', 'Nimda','Slammer', 'Moscow_blackout','Code_Red_I' ]

    dataset = Datasets[i].drop(['H+M','H','M','S','MED1','MED2','MED3','MED4','MED5','MED6','MED7',
                                'MED8','MED9','MED10','MED11','MAL1','MAL2','MAL3','MAL4','MAL5','MAL6',
                                'MAL7','MAL8','MAL9','Classification'],axis=1)

    parameter_space = {
        'n_estimators': list(range(50,151)),
        'criterion': ['gini', 'entropy'] }

    #Split the data set into training data and test data
    X= dataset
    y = Datasets[i]['Classification']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=42)

    #Train the model and make predictions
    model = GridSearchCV(RandomForestClassifier(), parameter_space, n_jobs=-1, cv=3)
```

```
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    print('Best parameters found for ',names[i],' Dataset :\n', model.best_params_)

    #Performance measurement
    print(classification_report(y_test, y_pred))
    print(confusion_matrix(y_test, y_pred))

    titles_options = [("Confusion matrix, without normalization for "+ names[i] + " Dataset", None),
                      ("Normalized confusion matrix for "+ names[i] + " Dataset", 'true')]
    for title, normalize in titles_options:
        disp = plot_confusion_matrix(model, X_test, y_test,
                                     cmap=plt.cm.Blues,
                                     normalize=normalize)
        disp.ax_.set_title(title)

        print(title)
        print(disp.confusion_matrix)

    plt.show()
    print(':\n')
```

```python
#Naive Bayes Classifier
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB, MultinomialNB
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import confusion_matrix, plot_confusion_matrix
from sklearn.model_selection import GridSearchCV


#Import the dataset
WannaCrypt=pd.read_csv('WannaCrypt.csv')
Nimda=pd.read_csv('Nimda.csv')
Slammer=pd.read_csv('Slammer.csv')
Moscow_blackout = pd.read_csv('Moscow_blackout.csv')
Code_Red_I = pd.read_csv('Code_Red_I.csv')
c_SVC = np.logspace(start = 0, stop = 10, num = 100, base = 2 , dtype = 'float64')
Datasets = [WannaCrypt, Nimda,Slammer, Moscow_blackout,Code_Red_I ]
names = ['WannaCrypt', 'Nimda','Slammer', 'Moscow_blackout','Code_Red_I' ]

for i in range(len(Datasets)):


    dataset = Datasets[i].drop(['H+M','H','M','S','MED1','MED2','MED3','MED4','MED5','MED6','MED7',
                                'MED8','MED9','MED10','MED11','MAL1','MAL2','MAL3','MAL4','MAL5','MAL6',
                                'MAL7','MAL8','MAL9','Classification'],axis=1)

    #Split the data set into training data and test data
    X= dataset
    y = Datasets[i]['Classification']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=42)

    #Train the model and make predictions
    model = MultinomialNB()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)


    #Performance measurement
    print(classification_report(y_test, y_pred))
    print(confusion_matrix(y_test, y_pred))

    titles_options = [("Confusion matrix, without normalization for "+ names[i] + " Dataset", None),
                      ("Normalized confusion matrix for "+ names[i] + " Dataset", 'true')]
    for title, normalize in titles_options:
        disp = plot_confusion_matrix(model, X_test, y_test,
                                     #display_labels=class_names,
                                     cmap=plt.cm.Blues,
                                     normalize=normalize)
        disp.ax_.set_title(title)

        print(title)
        print(disp.confusion_matrix)

    plt.show()
    print(':\n')
```

Accuracies for all ML Model

		MLP				DT				KNN				SVC				RF				NB			
		precision	recall	f1-score	support	precision	recall	f1-score	support	precision	recall	f1-score	support	precision	recall	f1-score	support	precision	recall	f1-score	support	precision	recall	f1-score	support
WannaCry	**-1**	0.69	0.67	0.68	1439	0.62	0.63	0.63	1740	0.65	0.62	0.63	1701	0.7	0.66	0.68	1758	0.69	0.67	0.68	1731	0.67	0.52	0.58	1731
	1	0.68	0.7	0.69	1441	0.62	0.61	0.62	1716	0.64	0.67	0.66	1755	0.67	0.71	0.69	1698	0.68	0.7	0.69	1725	0.61	0.74	0.67	1725
	accuracy			0.68	2880			0.62	3456			0.65	3456			0.68	3456			0.69	3456			0.63	3456
	macro avg	0.68	0.68	0.68	2880	0.62	0.62	0.62	3456	0.65	0.65	0.65	3456	0.68	0.68	0.68	3456	0.69	0.69	0.69	3456	0.64	0.63	0.63	3456
	weighted avg	0.68	0.68	0.68	2880	0.62	0.62	0.62	3456	0.65	0.65	0.65	3456	0.68	0.68	0.68	3456	0.69	0.69	0.69	3456	0.64	0.63	0.63	3456
Nimda	**-1**	0.94	0.96	0.95	1843	0.91	0.92	0.91	2165	0.91	0.97	0.94	2222	0.91	0.98	0.94	2199	0.93	0.97	0.95	2215	0.93	0.93	0.93	2215
	1	0.73	0.64	0.68	310	0.56	0.53	0.54	418	0.68	0.45	0.54	361	0.83	0.42	0.55	384	0.78	0.58	0.66	368	0.59	0.6	0.6	368
	accuracy			0.91	2153			0.86	2583			0.89	2583			0.9	2583			0.92	2583			0.88	2583
	macro avg	0.84	0.8	0.82	2153	0.73	0.72	0.73	2583	0.8	0.71	0.74	2583	0.87	0.7	0.75	2583	0.86	0.78	0.81	2583	0.76	0.77	0.76	2583
	weighted avg	0.91	0.91	0.91	2153	0.85	0.86	0.85	2583	0.88	0.89	0.88	2583	0.89	0.9	0.89	2583	0.91	0.92	0.91	2583	0.88	0.88	0.88	2583
Slammer	**-1**	0.98	0.99	0.99	1584	0.98	0.97	0.98	1904	0.97	0.99	0.98	1900	0.97	0.99	0.98	1889	0.98	0.99	0.99	1908	0.96	0.96	0.96	1908
	1	0.93	0.87	0.9	216	0.8	0.86	0.83	256	0.92	0.78	0.84	260	0.95	0.78	0.86	271	0.93	0.85	0.89	252	0.71	0.72	0.72	252
	accuracy			0.98	1800			0.96	2160			0.96	2160			0.97	2160			0.97	2160			0.93	2160
	macro avg	0.96	0.93	0.94	1800	0.89	0.92	0.9	2160	0.94	0.88	0.91	2160	0.96	0.89	0.92	2160	0.96	0.92	0.94	2160	0.84	0.84	0.84	2160
	weighted avg	0.98	0.98	0.98	1800	0.96	0.96	0.96	2160	0.96	0.96	0.96	2160	0.97	0.97	0.97	2160	0.97	0.97	0.97	2160	0.93	0.93	0.93	2160
Moscow	**-1**	0.99	0.99	0.99	1738	0.99	0.99	0.99	2088	0.99	1	0.99	2089	0.98	0.99	0.99	2089	0.99	1	0.99	2085	0.99	0.98	0.99	2085
	1	0.82	0.66	0.73	62	0.6	0.64	0.62	72	0.93	0.56	0.7	71	0.76	0.54	0.63	71	0.92	0.63	0.75	75	0.58	0.69	0.63	75
	accuracy			0.98	1800			0.97	2160			0.98	2160			0.98	2160			0.99	2160			0.97	2160
	macro avg	0.9	0.83	0.86	1800	0.79	0.81	0.8	2160	0.96	0.78	0.85	2160	0.87	0.76	0.81	2160	0.95	0.81	0.87	2160	0.79	0.84	0.81	2160
	weighted avg	0.98	0.98	0.98	1800	0.97	0.97	0.97	2160	0.98	0.98	0.98	2160	0.98	0.98	0.98	2160	0.98	0.99	0.98	2160	0.97	0.97	0.97	2160
Code Red	**-1**	0.96	0.99	0.97	1645	0.97	0.94	0.95	1985	0.95	0.99	0.97	1978	0.95	0.99	0.97	1989	0.96	0.99	0.97	1977	0.96	0.92	0.94	1977
	1	0.79	0.55	0.65	155	0.47	0.62	0.53	175	0.78	0.48	0.6	182	0.87	0.42	0.56	171	0.83	0.55	0.66	183	0.42	0.64	0.5	183
	accuracy			0.95	1800			0.91	2160			0.94	2160			0.95	2160			0.95	2160			0.89	2160
	macro avg	0.88	0.77	0.81	1800	0.72	0.78	0.74	2160	0.87	0.74	0.78	2160	0.91	0.7	0.77	2160	0.89	0.77	0.82	2160	0.69	0.78	0.72	2160
	weighted avg	0.94	0.95	0.94	1800	0.93	0.91	0.92	2160	0.94	0.94	0.94	2160	0.95	0.95	0.94	2160	0.95	0.95	0.95	2160	0.92	0.89	0.9	2160

Codes for Level 4: Deep Learning

```
#1D CNN Classification

#Import the Libraries
import math
import numpy as np
import pandas as pd
import keras
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers import Dropout
from keras.layers import Dense
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report,confusion_matrix
from datetime import datetime
from tensorflow.keras.optimizers import Adam, RMSprop, SGD
from keras.layers import Flatten
from keras.layers.convolutional import Conv1D, MaxPooling1D, AveragePooling1D

start = datetime.now()

plt.style.use('fivethirtyeight')
Moscow_blackout = pd.read_csv('Moscow_blackout1.csv')
WannaCrypt=pd.read_csv('WannaCrypt1.csv')
Nimda=pd.read_csv('Nimda1.csv')
Slammer=pd.read_csv('Slammer1.csv')
Code_Red_I = pd.read_csv('Code_Red_I1.csv')

Datasets = [WannaCrypt, Nimda,Slammer, Moscow_blackout,Code_Red_I ]
names = ['WannaCrypt', 'Nimda','Slammer', 'Moscow_blackout','Code_Red_I' ]
epochs = [250,300,200,100,300]

for i in range(len(Datasets)):
    print("CNN model for ", names[i])

    dataset_x=Datasets[i].drop(['H+M','H','M','S','MED1','MED2','MED3','MED4','MED5','MED6','MED7',
                                'MED8','MED9','MED10','MED11','MAL1','MAL2','MAL3','MAL4','MAL5','MAL6',
                                'MAL7','MAL8','MAL9','Classification'],axis=1)
```

```
targets=Datasets[i]['Classification']

dataset_x = dataset_x.values
le = LabelEncoder()
dataset_y = le.fit_transform(targets)

#Split the dataset into training and testing sets (80%: 20%)
x_train,x_test,y_train,y_test=train_test_split(dataset_x, dataset_y, test_size=0.2, random_state=42)

#Scale the all of the data to be values between 0 and 1
X_scaler = MinMaxScaler()
y_scaler = MinMaxScaler()
scaled_xtrain = X_scaler.fit_transform(x_train)
scaled_xtest = X_scaler.fit_transform(x_test)
#scaled_ytrain = y_scaler.fit_transform(y_train)

#Convert to numpy arrays
scaled_xtrain, scaled_ytrain = np.array(scaled_xtrain), np.array(y_train)

#Reshape the data into 2-D array
scaled_x = np.reshape(scaled_xtrain, (scaled_xtrain.shape[0],scaled_xtrain.shape[1],1))
scaled_y = np.reshape(scaled_ytrain, (scaled_ytrain.shape[0],1))

model_cnn = Sequential()
model_cnn.add(Conv1D(filters=32, kernel_size=2, activation='relu', input_shape=(scaled_x.shape [1], 1)))
model_cnn.add(MaxPooling1D(pool_size=3))
model_cnn.add(Dropout(0.3))

model_cnn.add(Flatten())
model_cnn.add(Dense(20, activation='softmax'))
model_cnn.add(Dense(1))
model_cnn.compile(optimizer='adam', loss='mean_squared_error', metrics=['accuracy'])

# fit model
cnn_history = model_cnn.fit(scaled_x, scaled_y, epochs=epochs[i], batch_size=32, verbose=1, validation_split=0.20)
```

With the support of the Erasmus+ Programme of the European Union

LEEDS BECKETT UNIVERSITY

```
#Convert x_test to a numpy array
x_test = np.array(scaled_xtest)
#Reshape the data into 2-D array
x_test = np.reshape (x_test, (x_test.shape[0],x_test.shape[1],1))
#check predicted values
predictions = model_cnn.predict(x_test)
x_pred = model_cnn.predict(scaled_x)

print(cnn_history.history.keys() )
print("\n")
# summarize history for loss
plt.plot(cnn_history.history['loss'])
plt.plot(cnn_history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.ylim(0, 0.08)
plt.legend(['train', 'test'], loc='upper right')
plt.show()

# summarize history for accuracy
plt.plot(cnn_history.history['accuracy'])
plt.plot(cnn_history.history['val_accuracy'])
plt.title('Model Accuaracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.ylim(0.05, 0.99)
plt. legend(['train', 'test'], loc='upper left')
plt.show()

test_loss, test_acc = model_cnn.evaluate(scaled_x, scaled_y)
print("Accuracy: ", test_acc)

print("Training RMSE: ", np.sqrt(mean_squared_error(y_train,x_pred)))
print("Testing RMSE: ", np.sqrt(mean_squared_error(y_test, predictions)))
print("Accuarcy of CNN model for " + names[i] + " is ", test_acc)
total = datetime.now() - start
print("Time: ", total, " minutes")
print(':\n')
```

```
#GRU Classification

#Import the Libraries
import math
import numpy as np
import pandas as pd
import keras
from numpy import concatenate
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers import Dropout
from keras.layers import Dense
from keras.layers import GRU
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report,confusion_matrix
from datetime import datetime
from tensorflow.keras.optimizers import Adam, RMSprop, SGD
from keras.layers import Flatten
from keras.layers.convolutional import Conv1D, MaxPooling1D, AveragePooling1D

start = datetime.now()

plt.style.use('fivethirtyeight')
Moscow_blackout = pd.read_csv('Moscow_blackout1.csv')
WannaCrypt=pd.read_csv('WannaCrypt1.csv')
Nimda=pd.read_csv('Nimda1.csv')
Slammer=pd.read_csv('Slammer1.csv')
Code_Red_I = pd.read_csv('Code_Red_I1.csv')
epochs = [250,300,200,100,300]

Datasets = [WannaCrypt, Nimda,Slammer, Moscow_blackout,Code_Red_I ]
names = ['WannaCrypt', 'Nimda','Slammer', 'Moscow_blackout','Code_Red_I' ]

for i in range(len(Datasets)):
    print("GRU model for ", names[i])

```

```
dataset_x=Datasets[i].drop(['H+M','H','M','S','MED1','MED2','MED3','MED4','MED5','MED6','MED7',
                           'MED8','MED9','MED10','MED11','MAL1','MAL2','MAL3','MAL4','MAL5','MAL6',
                           'MAL7','MAL8','MAL9','Classification'],axis=1)
targets=Datasets[i]['Classification']

dataset_x = dataset_x.values
le = LabelEncoder()
dataset_y = le.fit_transform(targets)

#Split the dataset into training and testing sets (80%: 20%)
x_train,x_test,y_train,y_test=train_test_split(dataset_x, dataset_y, test_size=0.3, random_state=42)

#Scale the all of the data to be values between 0 and 1
X_scaler = MinMaxScaler()
y_scaler = MinMaxScaler()
scaled_xtrain = X_scaler.fit_transform(x_train)
scaled_xtest = X_scaler.fit_transform(x_test)
#scaled_ytrain = y_scaler.fit_transform(y_train)

#Convert to numpy arrays
scaled_xtrain, scaled_ytrain = np.array(scaled_xtrain), np.array(y_train)

#Reshape the data into 2-D array
scaled_x = np.reshape(scaled_xtrain, (scaled_xtrain.shape[0],scaled_xtrain.shape[1],1))
scaled_y = np.reshape(scaled_ytrain, (scaled_ytrain.shape[0],1))

model_gru = Sequential()
model_gru.add(GRU(60, return_sequences = False, input_shape=(scaled_x.shape [1], 1)))
model_gru.add(Dropout(0.2))
model_gru.add(Dense(units=1))

#optimizer = RMSprop(learning_rate=0.01)
model_gru.compile(optimizer='SGD', loss='mean_squared_error', metrics=['accuracy'])

# fit model
gru_history = model_gru.fit(scaled_x, scaled_y, epochs=epochs[i], batch_size=32, shuffle=True, verbose=1,
                            validation_split=0.20)
```

```
#Convert x_test to a numpy array
x_test = np.array(scaled_xtest)

#Reshape the data into 2-D array
x_test = np.reshape (x_test, (x_test.shape[0],x_test.shape[1],1))

#check predicted values
predictions = model_gru.predict(x_test)
x_pred = model_gru.predict(scaled_x)

# List all data in history
print(gru_history.history.keys() )
print("\n")
# summarize history for accuracy
plt.plot(gru_history.history['loss'])
plt.plot(gru_history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.show()

# summarize history for accuracy
plt.plot(gru_history.history['accuracy'])
plt.plot(gru_history.history['val_accuracy'])
plt.title('Model Accuaracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.ylim(0, 0.8)
plt. legend(['train', 'test'], loc='upper left')
plt.show()

test_loss, test_acc = model_gru.evaluate(x_test, y_test, verbose=2)

print("Training RMSE: ", np.sqrt(mean_squared_error(y_train, x_pred)))
print("Testing RMSE: ", np.sqrt(mean_squared_error(y_test, predictions)))
print("Accuarcy of GRU model for " + names[i] + " is ", test_acc)
total = datetime.now() - start
print("Time: ", total, " minutes")
print(':\n')
```