Machine Learning Classification of Internet Worms and Ransomware Attacks and Effect of BGP Feature Properties

by

Hardeep Kaur Takhar

B.A.Sc. (Honours), Simon Fraser University, 2020

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Applied Science

in the
School of Engineering Science
Faculty of Applied Sciences

Declaration of Committee

Name:	**Hardeep Kaur Takhar**
Degree:	**Master of Applied Science (Engineering Science)**
Title:	**Machine Learning Classification of Internet Worms and Ransomware Attacks and Effect of BGP Feature Properties**
Committee:	**Chair: Shawn Sederberg** Assistant Professor, Engineering Science
	Ljiljana Trajković Supervisor Professor, Engineering Science
	Uwe Glässer Committee Member Professor, Computing Science
	Qianping Gu Examiner Professor, Computing Science

Abstract

Cyberattacks cause significant disruptions to communication networks and it is cruicial to detect and prevent such malicious behaviors to provide secure and reliable network connections. Detecting these intrusions is challenging and conventional intrusion detection techniques are insufficient to identify such malicious activities. Machine learning techniques offer effective intrusion detection due to their computational abilities. In this thesis, we apply machine learning techniques to classify anomalies such as Internet worms and ransomware attacks. We employ Border Gateway Protocol (BGP) datasets that contain routing records from Réseaux IP Européens (RIPE) collection sites.

Supervised machine learning algorithms Support Vector Machine, Long Short-Term Memory (LSTM), Gradient Boosting Decision Tree (GBDT) algorithms are employed for classifications. Dynamic learning rate scheduling and attention mechanism are employed to enhance the performance of LSTM models to classify ransomware attacks. While LSTM models proved effective in classifying attacks using sequential BGP data, their performance may degrade in case of lengthy data sequences. Feature transformation and selection techniques are applied to enhance performance of GDBT models. We perform feature selection to determine the most important features and identify the best fitting distributions. Experimental results indicate that a number of features follow heavy-tailed distributions. We evaluate performance of models generated using worms (Code Red, Nimda, and Slammer) and ransomware attack (WannaCrypt and WestRock) BGP datasets. Models generated using principal component analysis (PCA) transformed BGP data led to improved classification performance using the WestRock BGP dataset. Selecting important features using extra-trees algorithm led to the best classification performance of GBDT models. GBDT models offer short training time and may be suitable for designing scalable and real-time anomaly detection systems.

Keywords: Communication networks; cybersecurity; intrusion detection; malware; worms; ransomware attacks; anomaly detection; Border Gateway Protocol; goodness of fit test; supervised and unsupervised machine learning; feature selection; gradient boosting algorithms.

Dedication

To my parents, sister, and brother.

Acknowledgements

I would like to thank my family and friends for their unconditional love and support. I give my sincere thanks and appreciation to my advisor Prof. Ljiljana Trajković for her guidance that has greatly influenced my career. I thank her for all her efforts and time spent guiding me in my research journey and providing me with all the opportunities to learn and grow in my career. I would like to give my special thank to my former colleague Zhida Li for his guidance and support.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AdaBoost	Adaptive Boosting
AES	Advanced Encryption Standard
AI	Artificial Intelligence
ANNs	Artificial Neural Networks
ARPAnet	Advanced Research Projects Agency Network
ASes	Autonomous Systems
Bagging	Bootstrap Aggregation
BGP	Border Gateway Protocol
Bi-GRU	Bidirectional GRU
BLS	Broad Learning System
CART	Classification and Regression Trees
CatBoost	Categorical Boosting
CBC	Cipher Block Chaining
CDF	Cumulative Distribution Function
CIDR	Classless Inter-Domain Routing
CNNs	Convolutional Neural Networks
CV	Critical Value
DCNs	Data Center Networks
DDoS	Distributed Denial of Service
DoS	Denial of Service
DT	Decision Trees
ECG	Electrocardiogram
EFB	Exclusive Feature Bundling
EGP	Exterior Gateway Protocol
Extra-Trees	Extremely Randomized Trees

GBDT	Gradient Boosting Decision Tree
GBMs	Gradient Boosting Machines
GOSS	Gradient-Based One-Side Sampling
GRU	Gated Recurrent Unit
ID3	Iterative Dichotomiser 3
IDSs	Intrusion Detection Systems
IIS	Internet Information Service
IP	Internet Protocol
IRR	Internet Routing Registry
ISPs	Internet Service Providers
IXP	Internet Exchange Point
K-S	Kolmogorov–Smirnov
LANs	Local Area Networks
LightGBM	Light Gradient Boosting Machine
LSTM	Long Short-Term Memory
MLP	Multilayer Perceptron
MOAS	Multiple Origin AS
MRT	Multi-threaded Routing Toolkit
MVS	Minimal Variance Sampling
NCC	Network Coordination Centre
NIDS	Network Intrusion Detection Systems
NLRI	Network Layer Reachability Information
NML	Neural Machine Translation
NSA	National Security Agency
OS	Operating System
PCA	Principal Component Analysis
PDFs	Probability Distribution Functions
RaaS	Ransomware-as-a-Service
RBF	Radial Basis Function
RFE	Recursive Feature elimination
RIPE	Réseaux IP Européens

RIRs	Regional Internet Registries
RIS	Routing Information Service
RNNs	Recurrent Neural Networks
RRCs	Remote Route Collectors
SMBv1	Server Message Block version 1
SQL	Structured Query Language
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TCP	Transport Control Protocol
UDP	User Datagram Protocol
VNE	Virtual Network Embedding
XGBoost	eXtreme Gradient Boosting

Chapter 1

Introduction

The Internet is a global network that facilitates communication, collaboration, and access to information for businesses, educational institutions, healthcare institutes, governments, and non-profit organizations. Corporations and individuals rely on the Internet to perform their daily professional, social, and personal activities [2]. Therefore, secure and reliable Internet connectivity is crucial for effectively sustaining daily operations. Digital presence and device connectivity have immensely grown due to continued Internet expansion and to meet the increased user demands. This increase in digital presence exposes users to numerous exploitation incidents and security risks. Hence, managing secure connections of devices connecting to the Internet is a major challenge. The users are misled to browse compromised websites and are persuaded to download malicious software and files with hidden malicious intentions. Once downloaded on host devices, the software often automatically executes and corrupts the system. The evolution of attacks and increase in the complexity of devices connected to the Internet have outpaced current infrastructural and cybersecurity solutions. Therefore, new network architectures are being designed and implemented to address challenges in network security [3]. Network designers and administrators also monitor network traffic for malicious activities and manage permissions to prevent unauthorized system access and protect users against potential threats.

1.1 Motivation

Attackers invade enterprise networks by employing sophisticated and advanced techniques including connections via remote command-and-control (C&C) servers to gain access and steal users' data. C&C servers are malicious servers that first establish connections with infected hosts to enable communication. The C&C server then sends commands to infected machines and installs additional malicious software to gain complete control over compromised hosts and launch any commands [4]. Devices connected to a network are compromised by exploiting system vulnerabilities. Large corporations such as RSA, Target, Sony Pictures Entertainment, and Anthem were targeted in 2011 and 2014 [5]. Severe damages were caused due to stolen security tokens, personal and credit card information, proprietary documents, and malfunction of industrial operational equipment. Cyberattacks jeopardize the integrity, confidentiality, and availability of a system to legitimate users. Examples of cyberattacks include sniffing, phishing, man-in-the-middle attacks, end-point and source masquerading, trojans, viruses, worms, botnets, Denial of Service (DoS), Distributed Denial of Service (DDoS), and ransomware attacks [2, 5]. Network security is an essential topic in the field of computer networking given a surge in cyberattacks and the use of sophisticated attack techniques. Traditional intrusion detection approaches are insufficient to detect these attacks, given the plethora of attack techniques and immense volume of data. Hence, machine learning approaches might offer feasible solutions for network security.

1.2 Background

Malware, attack vectors, and exploits are employed to launch a successful attack. Malware is malicious software designed to harm or misuse any programmable device, service, or network. Spyware, keyloggers, rootkits, malicious adware, bots, trojans, viruses, worms, and ransomware are examples of malware [6, 7]. Spyware is software that collects users' information such as browsing history, search queries, login credentials, and other sensitive data without the consent of users. Keyloggers collect personal data by recording keyboard strokes. Rootkits are malicious softwares that are used to obtain administrative privileges in a system to perform harmful actions. Attackers employ rootkits to hide the presence of malware in compromised devices, steal sensitive data, create backdoors, and evade security systems [6]. Malicious adware are malicious softwares that are embedded in digital advertisements. They are designed to install additional malware on a device to monitor user activity or perform illicit activities without users' permission. They are also used as clickbait to steal money, redirect the user to malicious websites, and create bots for additional attacks. Bots are software programs that are designed to automatically execute repetitive tasks. Bots target vulnerable devices and employ them to send spam or perform illicit activities. Trojans are malicious softwares that are disguised as legitimate software. They gain remote access to systems by exploiting backdoors. Viruses and worms are a type of self-replicating malware. Viruses require user action to activate and infect devices. Worms do not require user interaction to infect and propagate. Ransomware is a type of malware that locks users' devices by employing encryption techniques and demanding ransom [7].

An attack vector is a method of obtaining unauthorized access to a system to perform an illicit activity by delivering malware in the form of attachments and by applying social engineering techniques. SQL injection, exploits, phishing, and passive monitoring are widely used attack vectors [8]. SQL injection is a technique of using malicious SQL code to obtain access to data from the backend databases. Exploits are tools that search for vulnerabilities in a system to launch an attack by installing malware. For example, attack vectors such as phishing emails are used to spread infected attachments. Phishing is a social engineering technique where the attacker masquerades as a credible person and tricks the victims into revealing sensitive information. Phishing emails usually contain malicious attachments and URLs. After the user opens the infected attachment or URL, the hidden exploit then searches the system for vulnerabilities to execute malware. Disruptions to the regular functioning of a network cause significant delays, financial damages, and data losses. Various passive attack vectors such as social engineering based techniques and active vectors such as malware are employed to gain unauthorized access to a network [9]. These malicious activities are inevitable in communication networks and may be detected using Intrusion Detection Systems (IDSs).

IDSs are categorized as host-based and network-based. Host-based IDSs are directly installed on host devices to monitor operating system files and processes [10]. Network Intrusion Detection Systems (NIDS) are passive devices that monitor traffic flows for malicious activities by inspecting packet flow or packet headers. Traffic flows are monitored from an internal port of the firewall. NIDs rely on matching signatures or detecting anomalous patterns in traffic data. Signature-based approaches match the attack signature against the database of pre-known events to identify threats, while anomaly-based approaches search for an activity deviating from an expected behavior [9]. Sample architectures of the host-based and network-based IDSs are shown in Fig. 1.1.

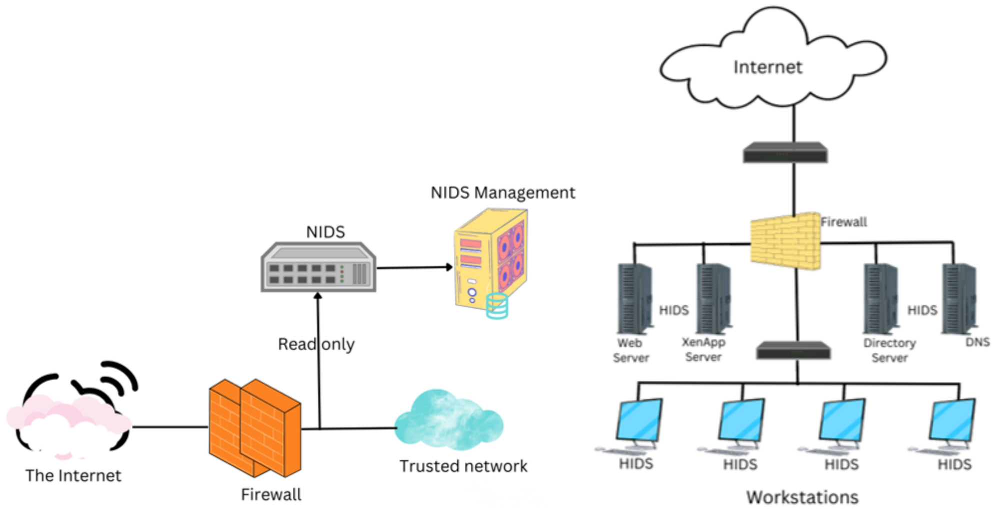


Figure 1.1: Examples of IDS architectures: host-based (left) and network-based (right).

1.2.1 Anomaly Detection

Anomalies are deviations of data from expected behavior. Such non-adhering patterns depending on the application domains are called outliers, discordant observations, exceptions, aberrations, surprises, peculiarities, or contaminants. The process of detecting these patterns is called anomaly detection. Anomaly detection is the most critical task of the Internet Service Providers (ISPs) to maintain network security and reliability. It is applied in diverse fields such as cybersecurity (network intrusion detection), finance (credit card and insurance fraud), healthcare (disease detection), and equipment operation (malfunction or failure detection). For example, an anomaly in network traffic data indicates a cyberattack or intrusion; theft in credit card and insurance transactions; an underlying health issue in medical imaging data; or an equipment malfunction for industrial operational data. Detecting anomalies is a challenging task because the regular behavior of a system keeps evolving, the context of behavior might vary for different domains, noise may reassemble as an anomaly, and the limited availability of labeled training data. Due to these challenges, anomaly detection solutions are designed specifically for the application domains.

Anomalies may be categorized as point, contextual, or collective anomalies. Point anomalies are data points that do not resemble the behavior of other data points in a dataset. For example, if a user residing in Canada makes a transaction in another country and that unusual transaction from the user's usual spending activity may be flagged as an anomaly. Contextual anomalies are data points that are considered anomalous in a specific setting and are otherwise regular. For example, a temperature monitor displaying high temperatures in summer months is considered regular, while it would indicate a faulty recording device during winter months. Collective anomalies are data points that are collectively considered anomalous in a dataset while they are otherwise individually considered regular. For example, a low value for a relatively long period in an Electrocardiogram (ECG) output might indicate an underlying heart disease. Point anomalies may occur in any dataset while contextual anomalies may only occur in datasets where instances are related and contextual attributes are available. Network activity should be critically monitored to flag suspicious activities and prevent attacks from recurring [11]. Anomaly detection is applied for intrusion detection in communication networks by employing rule-based and machine learning approaches. The cost of misclassifying data points depends on the application domain. In this thesis, we consider network anomalies in time series data collected from communication networks.

1.2.2 Border Gateway Protocol

The Internet Data Network model consists of five layers: application, transport, network, link, and physical layer. Routing protocols and algorithms are implemented in the network layer. The network layer is divided into the control plane and the data plane. All the routing decisions are implemented in the control plane, while packets are routed in the data plane. Border Gateway Protocol (BGP) [12] is a de-facto interdomain routing protocol that is developed from experience gained with Exterior Gateway Protocol (EGP) [13]. It is implemented in the network layer to facilitate Internet Protocol (IP) traffic routing between source and destination Autonomous Systems (ASes). Routing algorithms are employed to calculate an optimal path for packets to traverse through the Internet. BGP is an incremental protocol, that employs path vector, a variant of distance vector protocol.

ASes are groups of networks of routers that are managed by a single administration domain. They perform tasks such as packet delivery and assist with network connectivity [14]. Each AS has a unique identifier known as the AS number. The Internet is composed of such network of networks (ASes) that adhere to common routing policies designed by a network provider. The current version of BGP (BGP-4) supports Classless Inter-Domain Routing (CIDR) by removing the network class option. The BGP destination address is a CIDR network prefix of a subnet or collection of subnets. The existence and reachability information of each subnet is known to the Internet through an exchange of routing information by BGP. BGP facilitates the selection of efficient routing paths within the constraints of various network policies. It includes a new set of mechanisms for CIDR such as enabling IP prefix advertisement. It also allows the aggregation of routes, incremental additions, and an option to set routing policies [12].

The BGP connections are established between routers via Transport Control Protocol (TCP) connection using port 179 [14]. BGP peering session is the process where different ASes connect and exchange routing information. During a peering session, routers exchange BGP routing tables and apply existing routing policies to discover the best path for a packet to traverse. There are two types of BGP sessions: external BGP (eBGP) and internal BGP (iBGP). The gateway routers use eBGP to obtain reachability information for peers residing in distinct ASes. The gateway router then further shares routing information with all peers within the AS using iBGP. In summary, BGP enables communication between organizations by sharing network reachability information.

1.2.3 BGP Messages

BGP messages exchanged by BGP routers are *open, keepalive, update, and notification.* *Open* messages are exchanged after establishing a TCP connection. They include the BGP version, sender AS number, hold time (maximum number of elapsed seconds between consecutive keepalive and update messages), BGP identifier (IP address of the BGP speaker), and optional parameter length. After a BGP session is established, keepalive messages are exchanged among peers to ensure active BGP connections. BGP *update* messages are then exchanged to distribute information of available routes among routers. BGP is an incremental routing protocol and, hence, *update* messages are exchanged when network reachability or topological paths change. Otherwise, *update* messages of the existing prefix withdrawals are exchanged. BGP update messages include critical information about protocol status and configuration. Therefore, datasets may be generated by extracting fields of BGP update messages during an event of interest. Notification messages are exchanged to close an existing peering connection when there is a disagreement about the configuration parameters [14, 15].

A sample BGP *update* message is shown in Table 1.1. Important routing information such as AS-path, Next-hop, and announced NLRI prefix may be extracted from the BGP *update* message to examine traffic activities [16]. For example, the BGP update message is shown in Table 1.1 illustrates that the packet is routed from ASes listed in the AS-path field using the shortest path. The packet has a source IP address “192.65.185.195” and is en-route to the destination IP address “192.65.185.40” within the same AS with the subnet prefix “192.65.185”. Network prefix subnet and length are listed in the Network Layer Reachability Information (NLRI) field. The network may be reached via neighboring BGP routers [15,17].

Table 1.1: Sample BGP *update* message [1].

Field	**Value**
TIME	2022-1-16 01:40:06
TYPE	BGP4MP/BGP4MP_MESSAGE_AS4 AFI_IP
FROM	192.65.185.195
TO	192.65.185.40
BGP PACKET TYPE	UPDATE
ORIGIN	IGP
AS-PATH	15547 6939 4788 45259 10094 10030
NEXT-HOP	192.65.185.195
ANNOUNCED NLRI PREFIX	183.171.121.0/24
ANNOUNCED	183.171.120.0/24

BGP is prone to errors, misconfigurations, and lacks security mechanisms to validate legitimate route updates [18]. Events such as genuine policy changes or route flapping that are caused by the swift retraction of advertised routes might be perceived as anomalous BGP behaviors [19]. Power outages and routing table leaks destabilize the global network routing and disrupt Internet connectivity. An attacker may eavesdrop in a network by re-routing the traffic to an illegitimate AS or may disrupt traffic flows by dropping packets. This may lead to large-scale Internet disconnections.

1.2.4 BGP Anomalies

BGP was designed on a trust mechanism to exchange routing information between routers in different ASes. The protocol does not have any inbuilt mechanism to validate the authenticity of shared routing information. Due to this established trust between routers, it is susceptible to attacks. An intruder may misuse the trust to propagate false routing information across the Internet leading to eavesdropping or disconnections. These undesired changes in traffic patterns result in significant disruptions to network operations. Such unusual incidents are referred to as BGP anomalies. BGP anomalies may be caused due to hijacking attacks, misconfigurations, routing overloads, link failures, and changes in network topologies [20]. BGP hijacking attacks are launched by redirecting traffic from a legitimate

AS to desired ASes for malicious reasons. Attackers accomplish this by falsely advertising the ownership of IP prefixes that belong to other ASes to re-route the traffic. The re-routed traffic may be directed to non-existing paths, monitored for eavesdropping, obstructed by dropping packets, and prone to spoofing. Spoofing is a method of masquerading legitimate IP addresses for spam purposes. Multiple Origin AS (MOAS) is an example of a BGP hijacking event. MOAS conflict occurs when an IP prefix has more than one point of origin. Each IP prefix should belong to only one originating AS [21]. BGP misconfigurations lead to routers announcing used or unused prefixes. Both cases pose danger and may result in traffic overload on routers or loss of packets due to routing traffic through illegitimate or non-existing routes. BGP link failures are caused due to loss of connectivity and reachability between private and public BGP peers. The accurate identification of a BGP anomaly is a complex and challenging task. In the case of network anomaly, the cost of misclassifying anomalous data points is very high. Hence, it is critical to correctly identify anomalies and implement measures to prevent damage.

1.3 BGP Data Collections

The interdomain level routing information may be collected from BGP trace collectors, route servers, looking glasses, and Internet Routing Registry (IRR) databases. BGP trace collectors peer with ISPs to receive routing tables and updates from their peers [22]. The exchanged information is periodically stored in archives. BGP data may be generated by extracting fields of BGP update messages. Réseaux IP Européens (RIPE) [23] and Route Views [24] are two publicly available BGP trace collection sites. These repositories collect data from the BGP control plane.

RIPE Network Coordination Centre (NCC) is one of the five globally known Regional Internet Registries (RIRs) that provide service to Europe, the Middle East, and parts of Central Asia. Since 2001 RIPE Routing Information Service (RIS) project is managed by RIPE NCC, a non-profit organization. Route Views is a research project at the University of Oregon founded by the advanced network technology center. These organizations collect and store chronologically the collected routing data. BGP update messages collected by RIPE and Route Views repositories are publicly available to the community to download. They are stored in the Multi-threaded Routing Toolkit (MRT) format.

1.3.1 Réseaux IP Européens

RIPE RIS project was established to collect BGP routing data via globally distributed Remote Route Collectors (RRCs) at major Internet Exchange Point (IXP) sites. The rrcs employ Quagga routing software [25] to collect data as dump and update files. Dump files store the state of the routing system at a given time instance. The update files store all the changes in the routing system for a given period. A total of 25 rrcs are globally located in the following locations: Europe (16), North America (4), Asia (2), South America (2), and Africa (1) [26]. Gateway routers collect the routing data for their peers at the IXP peering Local Area Networks (LANs) collector. The routing data may also be collected by collectors that peer via a multihop mechanism. The following naming scheme is available to download files: "https://data.ris.ripe.net/rrcXX/YYYY.MM/TYPE.YYYYMMDD.HHmm.gz", where XX is the rrc number, YYYY is the year, MM is the month, TYPE is the file type (bview for dumps or update for updates), DD is the date, HH is the hour, and mm is the minute.

1.3.2 Route Views

The Route Views project was initially established to collect real-time routing data to enhance the understanding of the global routing system. Data are collected from various backbone routers at globally distributed locations. Route Views [24] project employs three types of collectors: FRRouting, Quagga, and Cisco. The FRRouting and Quagga collectors are based on Zebra routing software. BGP update messages and routing tables are collected by FRRouting and Quagga collectors every 15 mins and 2 hrs, respectively. The Cisco collectors obtain data every 2 hrs starting at 00:00. There are 36 Route Views collectors: 29 FRRouting, 6 Quagga, and 1 Cisco. These rrcs are distributed across the five RIRs: ARIN (14), LACNIC (6), APNIC (7), AFRINIC (4), and RIPE NCC (5). The available data are used to visualize AS paths and utilization of IPv4 address space. Various studies of Internet topology have also been performed to map IP addresses to the origin AS.

1.4 Machine Learning for Anomaly Detection

Network intrusions are detected by employing various approaches from the fields of machine learning and data mining. In the past, signature-based models have been widely adopted to detect anomalies. However, they lack adaptability as the techniques employed to attack keep evolving with time and it is not feasible to keep track of all the signatures of attacks. The limitations include a lack of contextual knowledge about the constantly evolving threats, detection systems that rely on expert training, and a single point of failure if the IDS is compromised. Therefore, intelligent cybersecurity solutions have been adapted to develop automated systems to address the increase in data volume and intrusions. Machine learning is a branch of Artificial Intelligence (AI) and statistics. It relies on data to build models for pattern recognition. Machine learning models are built on the foundation of human learning: memorizing, adapting, and generalizing.

A well-defined machine learning problem consists of a learning task, performance measure, and task experience [27, 28]. The four learning tasks in machine learning include classification, association, clustering, and regression. In classification tasks, the output variable contains finite discrete output categories. Binary classification is a task of identifying two distinct classes: regular and anomaly. In association, a relationship between observations is required. In clustering, the algorithm is applied to group alike observations. In regression, the output variable consists of multiple continuous output variables. The learning task of a machine learning model depends on the input training data. The machine learning models are trained to reduce the difference between the expected and predicted outcome. Machine learning approaches are categorized based on the availability of the data labels as unsupervised, supervised, and semi-supervised [29]. Unsupervised machine learning approaches are used when data labels are not available. They are employed to perform clustering. Supervised machine learning techniques are used when data labels are available. They are applied to perform classification and regression. Semi-supervised machine learning approaches use partially labeled training data to generate models. They may be implemented to solve a variety of problems such as classification, regression, clustering, and association. We perform binary classification of BGP anomalies using various unsupervised and supervised machine learning approaches.

1.5 Research Publications

[1] **H. K. Takhar**, A. L. Gonzalez Rios, and Lj. Trajković, "Comparison of virtual network embedding algorithms for data center networks," in *Proc. IEEE Int. Symp. Circuits Syst.*, Austin, TX, USA, May 2022, pp. 1660–1664.

Software defined networks are a new Internet architecture paradigm that allows coexistence of heterogeneous network architectures. They optimize network management (maintenance, operability, and effective content delivery) by provisioning a centralized network intelligence. Virtual Network Embedding (VNE) algorithms improve scalability and utilization of physical resources in Data Center Networks (DCNs). In this paper, we implement various DCN topologies and evaluate performance of VNE algorithms using the VNE-*Sim* simulator. We compare performance by implementing both server-centric and switch-centric DCN topologies.

[2] **H. K. Takhar** and Lj. Trajković, "BGP feature properties and classification of Internet worms and ransomware attacks," in *Proc. IEEE Trans. Syst. Man Cybern.*, submitted.

Machine learning approaches for anomaly detection heavily depend on the training data and their properties. In this paper, we analyze the impact of data probability distribution on the performance of machine learning models developed based on worms and ransomware attack Border Gateway Protocol datasets. We perform feature selection to determine the most important features and identify the best fitting distributions. Experimental results indicate that a number of features follow heavy-tailed distributions. Classification of traffic anomalies is evaluated using the gradient boosting decision tree models that offer short training time thus being suitable for designing real-time and scalable intrusion detection systems.

1.6 Overview of Related Work

In this Section, we provide a literature review dealing with signature-based and machine learning intrusion detection techniques. Network intrusion attacks severely disrupt the Internet connectivity and result in dire financial losses. Therefore, it is crucial to investigate the cause of these attacks and employ measures to terminate the attack and prevent network intrusions from recurring. An efficient approach involves first correctly identifying intrusions and then employing specific steps to terminate the attack. Since different attacks target different system and software vulnerabilities, therefore it is crucial to first correctly identify the type of attack before deploying resources to terminate the attack. This methodology will preserve resources and time in the case of an attack.

The resources employed in an attack may include numerous active (malware attacks) and passive (social engineering techniques) attack vectors. Active attack vectors are designed to exploit a system's resources by modifying or hindering its functionality. Vulnerabilities are exploited by employing techniques such as botnets, trojans, malware installation, hijacking, phishing attacks, and scanning systems. Various attack vectors employ social engineering techniques to mislead users and attack systems by gaining unauthorized access. The victims are forced to pay a ransom, attempt to remove malware, or restart the device to recover the system. Data retrieval is not guaranteed after the payment of ransom [30, 31].

A survey conducted by Sophos gathered statistics about different sized organizations ranging between 0.1k–1k and 1k–5k employees in 26 countries [32]. Sophos calculated the social and economic impact of ransomware attacks. It discovered that from the surveyed countries, 51% of the companies were attacked and the successful encryption rate of the attacks was 73%. Ransom was paid by 26% of the attacked companies to regain access to their data while other organizations relied on backups to recover the encrypted files. Overall, the revenue lost by organizations that paid a ransom ranged between \$ 0.7 – 1.4 million USD.

Various surveying methods and tools to detect network intrusions in routing data have been reported [33]. BGP is the interdomain routing protocol of the Internet. Attacks launched to compromise its regular operation affect the Internet integrity. Network anomalies and numerous approaches were surveyed to detect BGP anomalies [20]. The report highlights the vulnerability of the BGP protocol and difficulties to differentiate between BGP anomaly and reliability. For example, measures taken by ISPs to optimize utilization of the network resources may be falsely perceived as an anomaly. The techniques suggested to detect BGP anomalies are categorized as cryptographic-based prevention, anomaly mitigation, mitigation of unstable route propagation, and anomaly detection. Cryptographic techniques employ public key infrastructure (PKI) to authenticate routing announcements. PKI are encryption methods that employ public keys to confirm the validity of digital transactions. Anomaly mitigation relies on deferring suspicious updates of routes to prevent damage. The unstable routes as referred to in the report, are routes that do not inhibit the ability of BGP to share routing information. The mitigation of unstable route propagation approach refers to deferring sharing of unstable routes. Anomaly detection techniques rely on searching unusual patterns in traffic data [20].

Due to the shortcomings of the existing security approaches, the Internet is still exposed to hijacks, misconfigurations, and a lack of deployable security infrastructure. A rule-based approach [34] is designed using machine learning to detect network anomalies based on IP flow statistics. Network traffic may be analyzed by inspecting packet headers or packet payloads for pre-known behaviors known as signatures. The authors analyze and identify the challenges associated with signature-based detection systems such as Snort [35] that inspects packet signatures. A snort detection system is difficult to deploy in a large-scale network with high-capacity links due to high computational costs. It is favorable to analyze patterns in traffic for anomalous activity due to their smaller size compared to packet signatures. The suggested architecture uses the correlation between packet payload and flow data (flow signatures) to develop a scalable machine learning based detection system. The rules are defined by analyzing the IP flow traffic to replicate the actions of packet-based detection systems. Support Vector Machine (SVM), AdaBoost, and maximum entropy were selected for binary classification. The models were successfully able to detect anomalous traffic by monitoring flow signatures [34].

A generalized approach [36] was proposed to automatically generate features from raw data rather than applying manual feature extraction techniques that depend on observations. Techniques employed to manually generate features rely on extracting statistics such as the number of BGP update messages and AS-Path length. The authors criticize this feature extraction approach for lack of generality because the process is dependent on the observations. The datasets used were obtained using data collected by RIPE rrc04 located in Geneva. The BGP update messages are transformed using mathematical relationships into regularized vectors that are mapped to word indexes. Bag-of-words approach is then employed to generate sentence vectors. The neural networks are used to automatically extract features from the generated sentence vectors. The experimental results show improved results than the standard manual approach for extracting features. However, the approach has an extensive computational cost for sentence vectors with large dimensions.

Security mechanisms such as secure-BGP (S-BGP) and BGPsec are developed to secure the networks from vulnerabilities of BGP. A different type of attack called TIGER [37] evades the existing security mechanisms and falsely advertises non-existent network routes. TIGER attacks are launched by building fraudulent BGP sessions via tunneling between two routers residing in different ASes. After establishing a connection, non-existing connections are advertised to re-route the traffic for attackers' gain. TIGER attacks are difficult to detect because the routers exchange valid signatures for the forged routing paths. The routers also receive verification for the existence of forged routing paths in the data plane. A monitoring service Anti-TIGER was proposed to detect these attacks that rely on building neighbor AS graphs (NAGs). The direct spread spectrum technique is employed to add a watermark bit to the probing packets for detecting attacks. Covert channels are also built between intermediate and victim ASes to decode the hijacked traffic. The proposed approach [37] provides an effective solution to the condition when routers collude to attack a system by evading the existing BGP security schemes.

Implementation and configuration of routing policies is a challenging task. Unintended routing behavior degrades the efficacy of resources allocated by the operators and may lead to global connectivity issues. Networks suffer from unreliable connections due to prefix misconfigurations or fraudulent routing policies. Such anomalous behavior may be prevented by visualizing assigned prefixes. A BGP visibility toolkit [38] was developed for ISPs to monitor the connectivity status of interdomain prefixes. This visual tool is used by network providers to monitor assigned prefixes and detect faulty configurations by analyzing the global BGP routing tables. A global routing table contains the complete routing list provided by the ISPs to customers. The visibility of a prefix depends on the presence of an active route between a prefix and all ASes. Performance is evaluated using RIPE and Route Views data to illustrate that the tool provides reliable triggers for anomalous behaviors. In the past, routing leak events such as Dodo (February 2022) and YouTube prefix hijack in the case of Pakistan Telecom resulted in the inability of users to connect to the Internet. These disruptions can be reduced or prevented from recurring by utilizing BGP visibility toolkits.

Fast training time is desired for machine learning models to facilitate real-time anomaly detection, scalability with large-scale datasets, and to be computationally effective. The training time to generate intrusion detection models is investigated [39] using Light Gradient Boosting Machine (LightGBM) [40], Gated Recurrent Unit (GRU), Bidirectional GRU (Bi-GRU) [41], Broad Learning System (BLS) [42], and its extension algorithms. The models are generated using BGP [43], NSL-KDD [44], and CIC [45] datasets. The authors discover that the shortest training time as desired by most real-time IDSs is obtained using BLS models.

1.7 Roadmap

The thesis is organized as follows: In Chapter 1, we describe the motivation and background related to cyberattacks, anomaly detection, routing protocol BGP, and BGP messages. We also discuss the data collection repositories and the importance of applying machine learning in intrusion detection systems. We then list the research publications emanating from this project, followed by an overview of related work, and present the roadmap of the thesis. In Chapter 2, we describe network anomalies (viruses, worms, power blackouts, and ransomware attacks) and datasets employed in this thesis. Dimension reduction, data clustering, and feature selection approaches are discussed in Chapter 3. Various machine learning approaches such as Support Vector Machine (SVM), deep learning networks Long Short-Term Memory (LSTM), ensemble learning boosting algorithms, learning rate scheduling, and attention mechanism are described in Chapter 4. We perform feature selection by evaluating correlation coefficients and supervised machine learning tree-based models in Chapter 5 and identify underlying feature distributions of BGP features using goodness of fit test. Performance of models generated by using SVM, LSTM, and gradient boosting algorithms are evaluated in Chapter 6. We conclude with Chapter 7.

Chapter 2

Description of BGP Datasets

Malicious attempts exploit vulnerabilities of systems for monetary gains, political agendas, and the intentional spread of false information. Infected email attachments, malicious advertisements, infected USB drives, phishing emails, and texts are used to spread malware. These attacks result in major network disruptions and network anomalies. Network anomalies may be categorized based on their effect on network performance and security. Performance-related anomalies include file server failures, network congestion, and packet flooding that may result in network failures. Security-related anomalies include attacks such as trojans, viruses, worms, DoS, DDoS, rootkits, and ransomware attacks [46–49]. In this Chapter, we first describe network anomalies such as viruses, worms, power outages, and ransomware attacks. We then describe the process employed to generate datasets used in this thesis.

2.1 Viruses and Worms

Network viruses and worms are self-replicating malicious programs that compromise systems by consuming excessive network resources thus making the targeted resources and systems inaccessible to legitimate users [46]. Viruses are embedded as payloads in downloadable files and other software programs [50]. They invade a system through the activation of a host file prompted by a user. The first known virus Creeper was designed in 1971 as a test program. It harmlessly propagated through the Advanced Research Projects Agency Network (ARPAnet) (predecessor of the Internet) and displayed the message "I'm the creeper, catch me if you can". The virus gained access to servers and propagated by self-replication [51,52]. Worms are malicious programs that contaminate a system by self propagation and replication. Worms rely on social engineering techniques and target system vulnerabilities to invade networks. The world's first known worm Morris was launched on November 2, 1988 [53]. It was first detected by a student at the University of California, Berkeley. The worm targeted computers running a specific Unix OS and was designed to remain undetected. It compromised approximately $6,000$ computers out of the $60,000$ computers connected to the

ARPAnet. It employed multiple attack vectors to spread across the network and exploited loopholes in e-mail and user authentication systems. The casualties of the worm amounted to approximately a million dollars and resulted in major delays for essential operations and email deliveries. Measures such as erasing data of systems and disconnecting devices for weeks were taken to repair damages. Although creeper was originally called a virus, it propagated through networks without host activation, thus having the propagation properties of a worm.

Viruses and worms infect a system upon download of a malicious file. An executable file is embedded in virus infected files and a user action is required to activate the executable for the virus to infect. In contrast, in the case of worm infected files, the download of an infected file is sufficient to infect a system by exploiting vulnerabilities of a system. Email applications and scan engines are employed to spread malware (viruses and worms) in a network [54,55]. Code Red [56], Nimda [57,58], and Slammer [59,60] are well-known worms that caused severe disruptions by flooding networks with BGP update announcements and by exploiting vulnerabilities of Microsoft Internet Information Service (IIS) web servers.

The unchecked buffer in IIS web servers was located in the section of code that regulated the input URL. And by overflowing this buffer, the attacker gains complete control of the server and executes any code. Code Red worm exploited vulnerable IIS servers by causing buffer overflows. The code gained system-level access and generated a list of random IP addresses using a random seed generator and then spread the worm by probing these IP addresses [61]. The IP addresses generated each time using the random seed generator were unpredictable. Hence, the worm rapidly spread and infected thousands of vulnerable systems.

Nimda exploited vulnerabilities of Internet Explorer 5 running on IIS web servers and propagated via infected attachments using email messages, websites, and shared network drives [58]. Slammer infected systems by exploiting buffer overflow vulnerability in the MS Structured Query Language (SQL) servers and propagated using User Datagram Protocol (UDP) on port 1434 [59].

2.2 Power System Blackouts

Societies depend on the reliable availability of electrical power for conducting their daily activities. Power system blackouts are the loss of electrical power to end users. They are caused by transmission line failures, overload of transmission lines, malfunction of the protection equipment, human error, and natural calamities. Power outages pose a threat to public safety, have severe secondary repercussions, and cascading effects. The absence of reliable power backup may also affect ISPs during large-scale power outages. During a component failure, the neighboring components experience increased load leading to failures of multiple components. Therefore, it is crucial to study these events to maintain public and environmental safety. For example, the Moscow power system blackout occurred in May 2005 due to a transformer failure in the Changino substation of the Moscow energy ring. The power outage disconnected MSK-IX, a major Internet traffic exchange point. Users experienced disruption in the Internet connectivity for long periods because of disconnection between ISP peers and IXP [62].

2.3 Ransomware Attacks

Ransomware is a malicious software program that targets networks with weak security. System files are encrypted to restrict access and ransom payment is demanded from owners to regain access to the system. A variety of code obfuscation techniques like runtime packers and encryption methods are employed to evade security systems. Obfuscation may be employed to conceal information to make it difficult for a user to analyze. Code obfuscation techniques are implemented by attackers to prevent cybersecurity personnel from analyzing and reverse engineering the process employed to launch the malware. Runtime packers are software programs that are employed to obfuscate the content of files by using data compression. A new executable file, called runtime packer, is generated by compressing the content of the original malicious executable file. The compressed content is embedded as a payload in the newly generated executable file. Upon execution of this obfuscated executable file, the malicious executable is automatically decompressed in the system memory, hence, compromising the system security. Attackers prefer runtime packers to obfuscate malware because security systems may not be able to read the content of compressed files.

Ransomware attacks employ repeated actions to attack and encrypt the victim's data and, hence, may depict repeating local patterns. Detecting ransomware attacks by using endpoint protection systems that rely on matching signatures and using automated unpacking tools are challenging. Ransomware attack variants have distinct attack signatures and the new variant and old variants may differ. Hence, it is inefficient to detect ransomware attacks by matching signatures. Automated unpacking tools are employed to decompress executable files and detect malicious files. However, ransomware packages are designed using advanced techniques to evade automated unpacking tools [32].

Propagation methods are different approaches employed by attackers to enter a system. Ransomware attacks may be classified based on their propagation method as Ransomware-as-a-Service (RaaS), cryptoworm, and automated active adversary.

- RaaS are readily available ransomware kits, that are easy to deploy. Hackers with minimum skills purchase RaaS kits to exploit system vulnerabilities. Attackers then threaten to publish sensitive and confidential data of victims that could be protected in exchange for ransom payment [32]. Advanced cryptography techniques are employed to lock users' data. However, the decryption of data is not guaranteed [63,64]. Examples of RaaS are DarkSide, REvil, Dharma, and LockBit [65].

- Cryptoworm is a type of ransomware that reproduces itself for optimal reach and effect. It encrypts victims' data and self-replicates without host file activation, thus imitating the properties of a worm. The stolen sensitive information may then be sold on the dark web as a distribution kit. WannaCrypt is a cryptoworm ransomware variant.

- During an automated active adversary ransomware attack, intruders employ tools to automatically search systems for backdoors and inadequate security. The tools obtain access to devices and launch attacks for damage.

Ransomware attacks may also be categorized as locker, crypto, and scareware [66, 67]:

- Locker ransomware attacks disrupt primary access to users' devices by disabling basic operation functions in a system. The goal of locker ransomware attacks is to restrict access by locking system displays and keyboards [68].

- Crypto ransomware attacks encrypt files to make data inaccessible to legitimate users [30, 69]. Ransom payments are demanded in exchange for regaining system access [66, 70]. Symmetric, asymmetric, or hybrid encryption techniques are used in crypto ransomware attacks [71]. Symmetric encryption is a fast approach that employs one key for both encryption and decryption. The encryption key is embedded in the encrypted file and the victim might recover the key by using reverse engineering tools. Asymmetric encryption uses pair of keys to encrypt (public key) and decrypt (private key) data. RSA is a popular asymmetric encryption algorithm used in crypto ransomware attacks. In this case, the private key cannot be recovered by having access to the public key. Asymmetric encryption methods are time-consuming and require plenty of memory on the host system. Therefore, the encryption process may be interrupted by the host's IDS before the complete execution of an attack. Hybrid encryption uses a combination of symmetric and asymmetric encryption techniques. It is one of the most difficult mechanisms to decrypt. A random key is first generated to encrypt users' files and a command–and–control server is then used to generate public-private key pair [69, 72]. The private key is encrypted using the public key and data may only be decrypted using the private key.

- Scareware is employed to threaten users without posing a danger. Hackers disguise themselves as authoritarian individuals and falsely accuse victims of performing illegal activities. Victims are convinced to pay ransom payments to avoid legal repercussions such as litigation and social repercussions.

The PC Cyborg is the first known ransomware attack that targeted systems in December 1989 [73, 74]. Infected floppy discs were mailed to contacts of hijacked subscriber mailing lists (World Health Organization AIDS conference and PC Business World magazine). The encryption software was disguised as a survey. The malware hid hard drive files and encrypted file names making the system unusable. Ransom was demanded to regain access and protect compromised sensitive information from being leaked.

Spam emails, exploits, compressed executables, signed code, and code obfuscation techniques are employed to download illegitimate files and software packages. Attackers employ compressed executables to launch attacks and avoid being detected by security systems. The administrative privileges are gained by exploiting backdoors and vulnerabilities in the Operating System (OS) by using tools such as EternalBlue, Windows Event Viewer process, and CVE-2018-8453 [32,75]. The data encrypted during a ransomware attack is usually stored on the same disk (overwritten) or copied on a separate disk. Data may be partially or fully manipulated during the encryption process. Examples of ransomware attacks include Tescrypt, Crowti, Cerber, and Locky [76] as well as WannaCrypt [32] and Petya [77]. Several expert-based and machine learning approaches are employed to enhance detection of ransomware attacks. Machine learning approaches have strong computational capabilities to process large-scale datasets and more effectively address the detection of ransomware attacks than conventional techniques [76]. Deep learning models such as RNNs [78] have been advocated in the literature for their efficient performance in detecting anomalies.

2.3.1 WannaCrypt

In May 2017, WannaCrypt [32] cryptoworm infected over 230, 000 computers in 150 countries. It is also known as a network worm because it self-replicates throughout a network. System vulnerabilities are probed to gain administrative privileges by employing EternalBlue [79] exploit, and DoublePulsar [80] backdoor tool to replicate and execute the ransomware. ExploitBlue is a cyberattack tool developed by the US National Security Agency (NSA). It was leaked publicly by hackers on 14 April, 2017. ExploitBlue exploits the loophole in Server Message Block version 1 (SMBv1) [81] protocol to invade networks. SMBv1 is MS Windows 7 application layer protocol that allows users to obtain shared access to files and printers. Microsoft was notified by US NSA about the potential leak of the cyberattack tool. Subsequently, a security patch was released by Microsoft. However, not all computers were updated with this security patch and attackers succeeded in deploying ExploitBlue as a backdoor to infect vulnerable computers. The cryptoworm then self-propagated and infected large numbers of systems. The cryptoworm copied and encrypted important documents using encryption methods. In exchange, the attacker demanded bitcoins as ransom payments.

A symmetric encryption algorithm Advanced Encryption Standard (AES) based on 128-bit Cipher Block Chaining (CBC) mode was employed to encrypt files [32]. A single-threaded encryption process was used to encrypt files one at a time. After encryption, the cryptoworm either copies files or overwrites manipulated data. The encrypted files are renamed and replaced by adding the ".wncry" extension. The keyword "WannaCry!" is added to the combination of the AES key and encrypted data. Older versions of the data are stored in the system using Volume Shadow Service. During a ransomware attack, the volume shadow copies are deleted from the system by using a Windows utility VSSADMIN.EXE or WBADMIN.exe to prevent data recovery [32, 79, 82, 83]. The ransomware targets the boot system data and prohibits the MS Windows diagnostics and repair tool to run automatically after the third failed boot. It then attempts to execute a normal boot in case of a failed boot, shutdown, or checkpoint. Entire encrypted data is saved in a storage drive and buffers are flushed from the system. The wallpaper of the computer is replaced with a message stating that the data are locked and that ransom payment is required to recover data. WannaCrypt variants continue to remain a high risk for IT systems.

2.3.2 WestRock

WestRock [84], a USA manufacturing company with over 320 manufacturing facilities worldwide was attacked in late January 2021. The company's information (IT) and operational (OT) systems were targeted resulting in major shipments and production delays. The attack was detected on 23 January, 2021 and lasted over six days. The company immediately initiated response protocols to identify, contain, and recover from the attack. The process included systematically shutting down systems for security. Additional measures were employed to scan and enhance the existing security infrastructure. The company executed alternative manual processes for delivering packaging solutions. The shutdown systems were slowly brought back online in controlled phased stages. The production was 85, 000 tons lower than the projected goals [85, 86].

2.4 Extraction of BGP Features

Performance of machine learning models relies on the quality of data used for training and testing models. In this thesis, we use BGP data from communication networks. Datasets are generated by processing BGP update messages downloaded from the RIPE repositories. Data are captured by these collection sites using the Quagga [25] software suite derived from the multi-server routing software Zebra [87]. The BGP update messages are available for download in MRT format. The downloaded files are converted to ASCII format using the zebra-dump-parser [88] tool. The 37 BGP numerical features are generated by extracting statistics of the features from BGP update messages using the C# [89] tool. These numerical features are continuous, categorical, and binary [90]. Each data point corresponds to the collected messages over one minute time intervals. The extracted BGP features listed in Table 2.1 [1] are categorized as AS-path and volume features.

The AS-path field contains lists of routers that a packet traverses to reach its destination. Features that are extracted from the AS-path field of the BGP update message fields are called AS-path features. The remaining extracted features are categorized as volume

Table 2.1: List of features (*AS*-path and *volume*) extracted from BGP update messages.

Feature	**Name**	**Category**
1/2	Number of announcements/withdrawals	volume
3/4	Number of announced/withdrawn NLRI prefixes	volume
5/6/7	Average/maximum/unique AS-path length	AS-path
8/10	Number of duplicate announcements/withdrawals	volume
9	Number of implicit withdrawals	volume
11/13	Maximum/average edit distance	AS-path
12	Arrival rate	volume
14-23/	Maximum AS-path length = n/edit distance = n,	
24-33	where n = (11, ... , 20)	AS-path
34/35/36	Number of IGP, EGP or, incomplete packets	volume
37	Packet size (B)	volume

features. The definition of the extracted features is listed in Table 2.2. For example, the feature F1 (volume) corresponds to the number of announcements is the number of routes that are available for packet delivery. The feature F5 (AS-path) is the average path size of the respective AS. During a worm attack, the routers experience a surge of EGP packets. The incomplete BGP messages indicate that the incoming packet has an unknown source of origin. BGP is highly vulnerable to misconfigurations and attacks due to the trust relationship with BGP peers. During an attack, a manipulator may propagate false routing information throughout a network. This propagation of false information results in an increased number of announcements (volume feature). The volume features compose 65% of the influential features. The AS-path features show high variance during an anomalous event [16].

The details of the Code Red, Nimda, and Slammer worm training and test RIPE datasets are listed in Table 2.3. The details of training and test RIPE datasets for WannaCrypt and WestRock ransomware attacks are listed in Table 2.4. The Route Views data collection for Internet worms are only available for the Slammer dataset because Route Views have begun collecting update messages from October 2001. Numerous experiments were performed to select the partition for the datasets that obtain the best performance. The training and test datasets are created such that they contain 60% and 40% of the anomalous data points, respectively [1].

The data are normalized using a z-score to have a mean 1 and a standard deviation 0. The z-score is calculated as:

$$z = \frac{x - \mu}{\sigma}, \tag{2.1}$$

where x is the input data, μ is the mean, and σ is the standard deviation.

The datasets contain two labels: regular (0) and anomaly (1). For simplicity, all data collected during the periods of reported anomalous events are labeled as anomalies. The remaining data points are labeled as regular. However, data points in the considered time window for anomalous events may also contain regular data points. It may not be very pre-

Table 2.2: Definition of *volume* and *AS-path* features extracted From BGP *update* messages [1].

Feature	**Name**	**Definition**
1	Number of announcements	Routes available for delivery of data
2	Number of withdrawals	Routes no longer reachable
3/4	Number of announced/ withdrawn NLRI prefixes	BGP *update* messages that have type field set to announcement/withdrawal
5/6/7	Average/maximum/ average unique *AS-path* length	Various *AS-path* lengths
8/10	Number of duplicate announcements/withdrawals	Duplicate BGP *update* messages with type field set to announcement/withdrawal
9	Number of implicit withdrawals	BGP *update* messages with type field set to announcement and different *AS-path* attribute for already announced NLRI prefixes
11/13	Maximum/average edit distance	Maximum/average of edit distances of messages
12	Arrival rate	
14–23/24–33	Maximum AS-path length/ edit distance	Various *AS-path* lengths/maximum edit distances
34/35/36	Number of IGP, EGP, or, incomplete packets	BGP *update* messages generated by IGP, EGP, or unknown sources

cise to label all data points in a given time window as anomalous. Therefore, the considered anomalous data points may be further analyzed to discover hidden regular data points. This process of analyzing data to precisely label data points is called label refinement. Unsupervised clustering algorithms such as k-means and isolation forest may be employed to refine data labels by identifying regular data points in the considered anomalous data.

Table 2.3: BGP Internet worm datasets: number of data points. Note that Route Views data collection began in 2003 [1].

Collection site	**Dataset**	**Regular** (min)	**Anomaly** (min)	**Regular** (training)	**Anomaly** (training)	**Regular** (test)	**Anomaly** (test)	**Collection date** **Start**	**End**
RIPE	Code Red	6,600	600	3,679	361	2,921	239	17.07.2001 00:00:00	21.07.2001 23:59:59
	Nimda	7,308	1,301	3,673	827	3,635	474	16.09.2001 00:00:00	21.09.2001 23:59:59
	Slammer	6,331	869	3,210	530	3,121	339	23.01.2003 00:00:00	27.01.2003 23:59:59
Route Views	Slammer	6,319	869	3,198	530	3,121	339	23.01.2003 00:00:00	27.01.2003 23:59:59

Table 2.4: BGP ransomware attack datasets: Number of data points [1].

Collection site	**Dataset**	**Regular** (min)	**Anomaly** (min)	**Regular** (training)	**Anomaly** (training)	**Regular** (test)	**Anomaly** (test)	**Collection date** **Start**	**End**
RIPE/ Route Views	WannaCrypt ransomware	5,760	5,760	2,880	3,420	2,880	2,340	10.05.2017 00:00:00	17.05.2017 23:59:59
	WestRock	5,832	10,008	2,952	6,008	2,880	4,000	21.01.2021 00:00:00	31.01.2021 23:59:59

Chapter 3

Dimension Reduction and Feature Selection

Machine learning is the process of learning and identifying patterns in data. Machine learning algorithms are trained using data to generate models. The dataset used during the training process is called training dataset. Training data are employed to fine-tune models to obtain the best performing hyperparameters for creating generalized models. The dataset used to evaluate the performance of a model is called test dataset. Models generated using the best performing hyperparameters are evaluated using test datasets. The input datasets for the algorithms are feature matrices with rows representing data points and columns representing features. The robustness of machine learning models relies on the quality of data. Selecting features that do not capture relationships between input data may lead to poor classification results. The generated model may also misclassify test data points if the training data are unbalanced or contains redundancies (noise). In the case of unbalanced datasets, the classifier may be biased to accurately identify data points belonging to the majority class and may misclassify data points belonging to the minority class. The presence of redundancies in datasets increases training time, computational complexity of models, and memory usage. These factors negatively impact performance of machine learning models. Therefore, it is critical to select relevant features and appropriate combinations of features to enhance classification performance [91, 92]. Spatial separation between features may be enhanced by selecting an appropriate combination of features [16].

Techniques such as dimension reduction are employed to eliminate irrelevant features. Dimension of data may be reduced by either transforming features or by selecting a subset of the dataset. Data are transformed to a lower dimension so that their characteristics are preserved. The transformed data may then be used as input for supervised machine learning algorithms [93]. Data with lower dimension may also be used to visualize high dimensional data. Examples of unsupervised dimension reduction techniques include Principal Component Analysis (PCA) and autoencoders [94].

Relevant features may be identified by measuring importance of features and ranking them based on a pre-defined criteria. Feature selection is a pre-processing step employed to select a subset of features to enhance performance of machine learning models. It reduces the dimension of the original dataset. Data with lower dimension lead to reduced computational complexity of a model, decreased training time, and prevent data overfitting. By selecting a sufficient number of relevant features, generalized models may be built to classify data with a lower error rate. Some widely used feature selection techniques include correlation, Decision Trees (DT), RF, and extra-trees [95, 96].

Traffic anomalies cause significant disruptions to communication networks and, hence, various techniques including machine learning have been used to prevent such malicious activities. We apply dimension reduction and feature selection techniques to pre-process data and identify important features. The important features are then used as an input for the machine learning algorithms. We perform BGP anomaly detection as a binary classification task using time series BGP data. The datasets employed in this thesis contain numerical features. The expected classification outputs of machine learning models are categorical (binary) values containing regular or anomaly class labels. We compare the important features selected by feature transformation and selection approaches. Performance of models generated using new features are evaluated to identify the best performing models. The experiments performed using various feature transformation and selection approaches to analyze the classification performance are discussed in Chapter 6. Performance is evaluated by measuring performance metrics such as training time, accuracy, F-Score, precision, recall, and confusion matrix.

3.1 Unsupervised Machine Learning Approaches

Unsupervised machine learning approaches are employed to extract the underlying properties of data when data labels are unknown. These approaches may also be applied to reduce the data dimension and to perform data clustering. We employ PCA and correlation techniques to reduce data dimension.

3.1.1 Dimension Reduction Using Principal Component Analysis

PCA is a popular dimension reduction approach. It is applied to generate new features called principal components [97] that are orthogonal to each other. It is employed to identify linear relationships between features. A span of vectors is all linear combinations of a given set of vectors. In a given feature matrix, the span of feature vectors represents a column space. The linearly dependent vectors should be eliminated from a column space to identify the most relevant vectors. Vector space is a set of vectors that may be added together and multiplied with a scalar to generate new vectors. A subset of a vector space (subspace) includes zero vector and addition and scalar multiplication to generate new vectors. Orthogonal subspaces include a set of vectors that are orthogonal to each other. Principal components are eigenvectors of orthogonal subspace. The PCA principal components are generated by projecting the original features onto orthogonal subspaces ranked in descending order of variance. The first and last principal components contain the highest and the least data variance, respectively.

The PCA is employed to retain important information from the data by eliminating dependent features. PCA is a computationally expensive approach especially when used with larger datasets because it employs Singular Value Decomposition (SVD) [97]. The data should be normalized using a z-score to have zero mean and unit variance before applying PCA to transform features. The evaluated PCA variance ratio (left) and cumulative variance (right) with the increase in a number of principal components, using the WannaCrypt RIPE training dataset are shown in Fig. 3.1. The number of principal components to be selected for dimension reduction depends on retaining the desired level of variance. In our experiments, we select the number of principal components to preserve approximately 70% of the data variance [98]. In the case of WannaCrypt data, the first 10 principal components of PCA preserve 71.70% of the data variance.

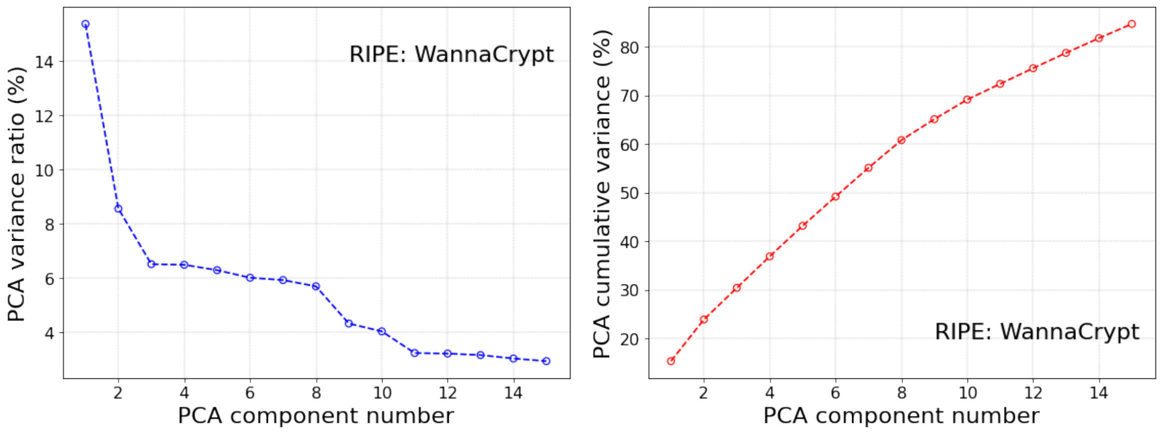


Figure 3.1: WannaCrypt RIPE training dataset: variance ratio (left) and cumulative variance ratio (right) of PCA components vs. the number of components. The principal components are generated so that the variance in data is preserved in descending order. The first 10 principal components preserve over 70% of data variance.

The separation between two classes of WannaCrypt data is visualized using the first and second principal components in two-dimensional and three-dimensional plots. The scatter plot of the first two principal components in two-dimensional space is illustrated in Fig. 3.2. We observe that the majority of data points belonging to regular and anomalous class overlap. Hence, the data are visualized in a higher dimension to examine data separation. The scatter plot of the first three principal components in a three-dimensional space is shown in Fig. 3.3. The orange colored (*) data points represent the anomalous class while the blue colored (o) data points represent the regular class. Four clusters are observed where data points belonging to regular and anomaly classes still overlap. Note that the majority of the regular and anomalous points are overlapping after the PCA transformation. Hence, other classification algorithms may further be employed to enhance the separation between clusters.

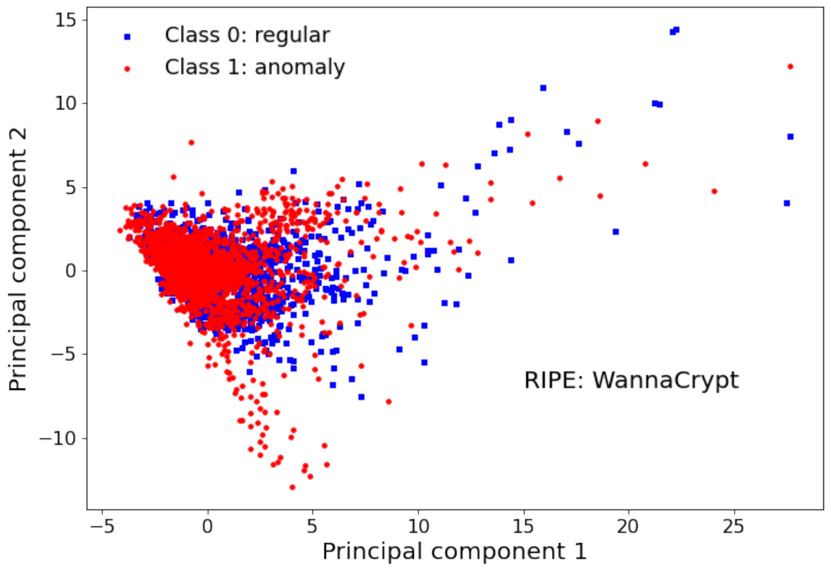


Figure 3.2: WannaCrypt RIPE training dataset: scatter plot of regular and anomalous clusters based on principal components 1 and 2. The majority of data points belonging to both classes after PCA transformation still overlap.

3.1.2 Data Clustering Using k–Means

The k–means [99] algorithm is a widely used unsupervised iterative machine learning clustering algorithm. It is employed to separate data points into pre-defined number of clusters. The algorithm uses Euclidean distance as a measure to cluster data. The k centroids are randomly initialized. Data are then assigned to the nearest clusters to minimize their intra-cluster distance from the centroid (inertia). Intra-cluster distance is the average distance between each data point within a cluster and its centroid. Inter-cluster distance is the average distance between all clusters. After data points are assigned to the clusters based on the nearest centroid, the centroids are recalculated by taking the average distance of all data points in a given cluster. The intra-cluster distances of data points are re-calculated to re-assign data points to the nearest cluster. This process is repeated until the recalculated centroids remain unchanged. The process of centroid recalculations and data reassignments to the nearest clusters may also be controlled by pre-defining the maximum number of iterations. The algorithm will repeat the process until one of these conditions is met. Clusters with the lowest possible intra-cluster distance are called homogeneous. Domain knowledge or evaluation criteria are employed to select the optimal number of clusters, since the labels

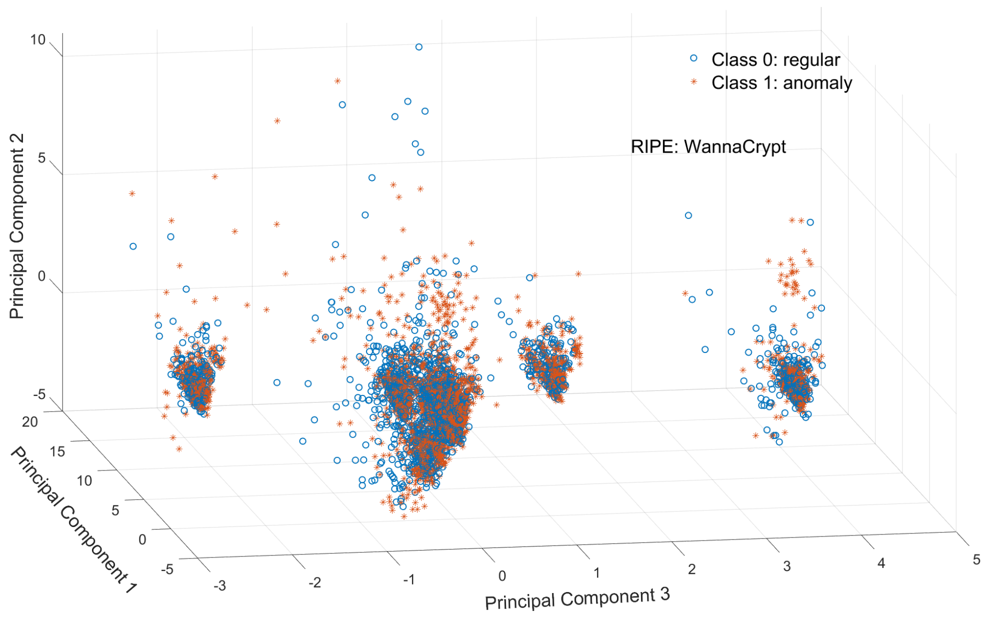


Figure 3.3: WannaCrypt RIPE training dataset: scatter plot of the first three principal components that belong to regular and anomalous classes in three-dimensional.

are unknown in unsupervised learning approaches. Evaluation measures for the k-means algorithm include elbow curve, silhouette analysis, inertia, and Dunn index [100].

- The elbow curve method is used to evaluate the sum of the squared distance between cluster centroids and data points within a cluster. The sum of squared distance and number k is plotted to identify the optimal number of clusters. The optimal k value is selected as the first lowest sum of squared distance before the graph flattens.

- The silhouette analysis is applied to infer the inter-cluster and intra-cluster distances. The silhouette coefficient is calculated as:

$$Silhouette\ coefficient = \frac{b-a}{max(a,b)}, \tag{3.1}$$

  where a is the average intra-cluster distance and b is the average inter-cluster distance. The value of the silhouette coefficient $\in [-1,1]$. Silhouette coefficient 0 indicates that the data points are very close to the neighboring clusters and are poorly clustered. The value 1 indicates that the clusters are homogeneous and well separated. The value -1 indicates that the points are assigned to the wrong clusters. Silhouette plots are analyzed to illustrate the quality of the clusters. The thickness of the silhouette plots indicates the size and quality of the clusters. A value of k is selected such that the thickness of the plots is uniform and the silhouette coefficient value is above the average silhouette score.

- Inertia is a measure of the sum of distances between data points and the centroid of a cluster (intra-cluster distances). A homogeneous cluster should have a small value of inertia.

- The Dunn index for given clusters is a ratio of the minimum inter-cluster and the maximum intra-cluster distances. A high value of the Dunn index is desired to form homogeneous and well separated clusters. Evaluation of the Dunn index is computationally expensive for large-sized datasets.

The k-means algorithm performs effectively for data that form spherical clusters. High-dimensional transformation approaches may be considered to enhance data separation in the case when data cannot be spherically clustered. The scatter plot of the number of announcements and the number of withdrawals is shown in Fig. 3.4. The two clusters that belong to regular and anomalous class overlap and might not be identified with high accuracy by machine learning models. Data may contain different types of anomalies that may be

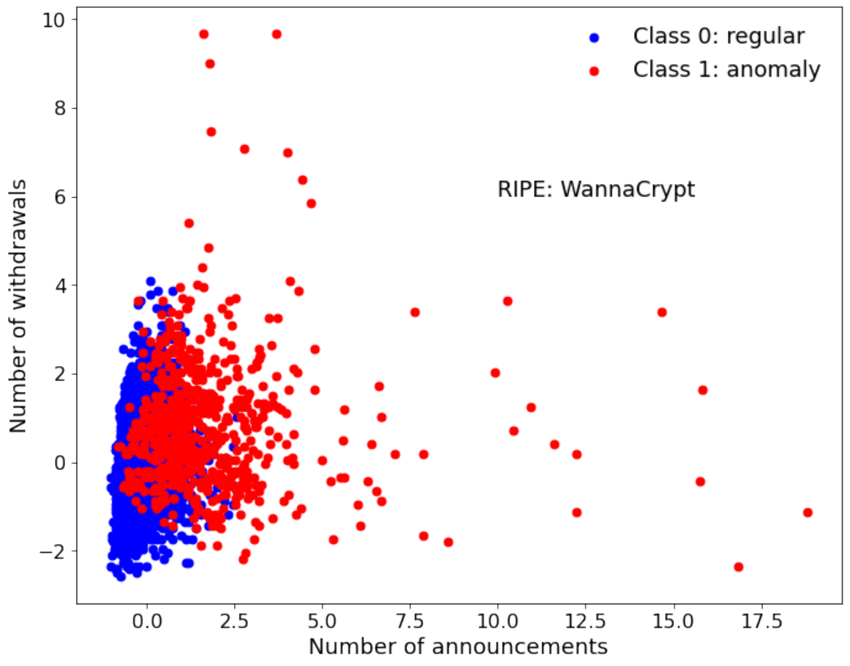


Figure 3.4: WannaCrypt RIPE training dataset: scatter plot number of announcements vs. number of withdrawals. While blue data points belong to the regular class and red data points belong to the anomalous class. Partial separation is observed between the two clusters.

categorized into different classes by performing multi-class classification. We perform binary classification to classify anomalous and regular data points. Experiments are performed using WannaCrypt RIPE training data by varying the number of clusters (k) to achieve better clustering. The data are normalized using a z-score to contain zero mean and unit variance. The Fig. 3.5 illustrates a three-dimensional scatter plot with the number of announcements vs. number of withdrawals vs. number of announced NLRI prefixes clustered into multiple clusters. The data are clustered is into two (top left), three (top right), four (bottom left), and five (bottom right) clusters. Several additional experiments are performed to cluster data into a larger number of clusters. The k-means is an unsupervised clustering machine learning algorithm the labels predicted by the model are assigned by the algorithm based on the distance and do not represent any of the classes. Therefore, the data points included in the clusters may belong to more than one class. Note that a cluster O shown in Fig. 3.5 (top left) of similar shape and size appears in all figures using a varied number of clusters. This cluster with a consistent size may represent the regular data points.

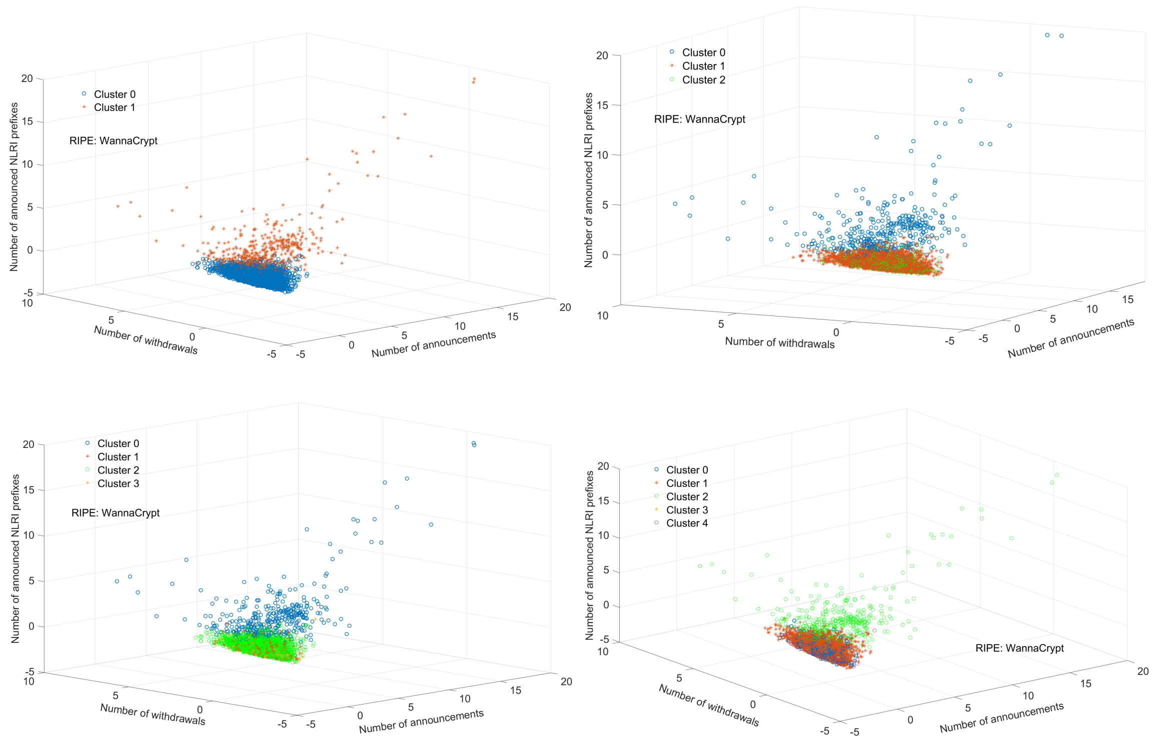


Figure 3.5: Scatter plot of three features of the WannaCrypt RIPE training data segregated into multiple clusters (k = 2, 3, 4, and 5).

The silhouette coefficients are analyzed to evaluate the quality of clusters. The silhouette plots using WannaCrypt RIPE training data using $k = 2, 3, ..., 6$ clusters and the generated clusters are shown in Fig. 3.6 and Fig. 3.7 left and right columns, respectively. The dashed line indicates the average value of the silhouette coefficient. The silhouette coefficients evaluated for k = 4, 5, and 6 clusters are lower than the average value of the coefficient. A major imbalance in the cluster size is observed in the silhouette plot for $k = 2$. The cluster labeled 0 in Fig. 3.6 (left, top) has a similar shape in silhouette plots for k = 3, 4, 5, and 6 cluster values. It indicates similar data properties and that this cluster of data may represent regular data. As the number of clusters increases, the cluster labeled 0

($k = 2$) is subdivided into smaller clusters. The shape of clusters consisting of a majority of data points is preserved for all selected values of k.

3.1.3 Cluster Refinement Using PCA and *k*-Means Algorithms

PCA (a statistical dimension reduction technique) and k-means (an unsupervised machine learning algorithm) are combined to improve the clustering of data by increasing the separation between data points belonging to regular or anomalous classes [101]. The silhouette coefficient of the two clusters generated using k-means with $k = 2$ as shown in Fig. 3.4 formed using the number of announcements and the number of withdrawals is 0.3393. The silhouette score is rather low and may indicate that the test data may not be adequately classified. We employ PCA to transform the original dataset to enhance the separation between the two classes. Various experiments are performed to identify the best number of PCA components to obtain higher silhouette scores and lower inertia as shown in Fig. 3.8. Transforming the data to 3 PCA components generates the highest silhouette coefficient 0.5222. The silhouette coefficients decrease with the increase in the number of PCA components.

Clustering performed using the PCA transformed features is shown in Fig. 3.9. The silhouette score increases to 0.5222 by clustering PCA transformed features using the first 2 principal components. Clustering after using PCA shows a visible separation between the two classes.

3.2 Feature Selection

Feature selection is a pre-processing step that involves selecting a subset of features from a given dataset based on ranking criteria. Important features are then employed as input for the machine learning algorithms. Classification performance of machine learning models may be enhanced by selecting important (relevant) features. Datasets with correlated features may negatively influence the model's performance. In machine learning the features should be correlated with the output to obtain high performance. Therefore, correlated features may be eliminated from the datasets before training. We evaluate Pearson and Spearman correlation coefficients to identify correlated features and perform feature selection by eliminating correlated features. Supervised machine learning tree-based algorithms such as random forests and extra-trees are employed to identify important features.

3.2.1 Person Correlation

Covariance is a measure of the strength and direction of a relationship between variables:

$$Cov(X,Y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1}, \tag{3.2}$$

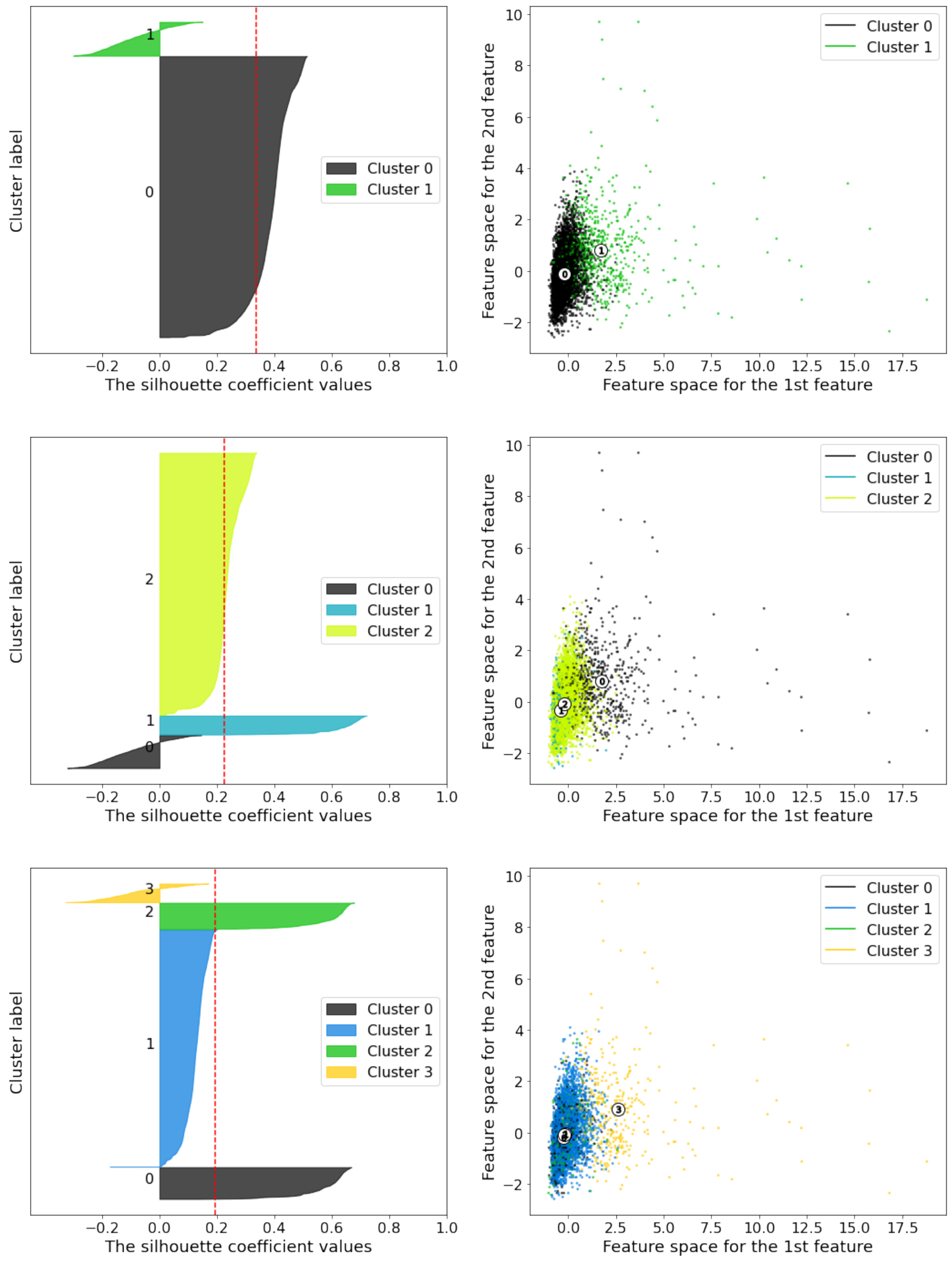


Figure 3.6: WannaCrypt RIPE dataset: silhouette coefficient (left) and cluster scatter (right) plots for $k = 2$, 3, and 4.

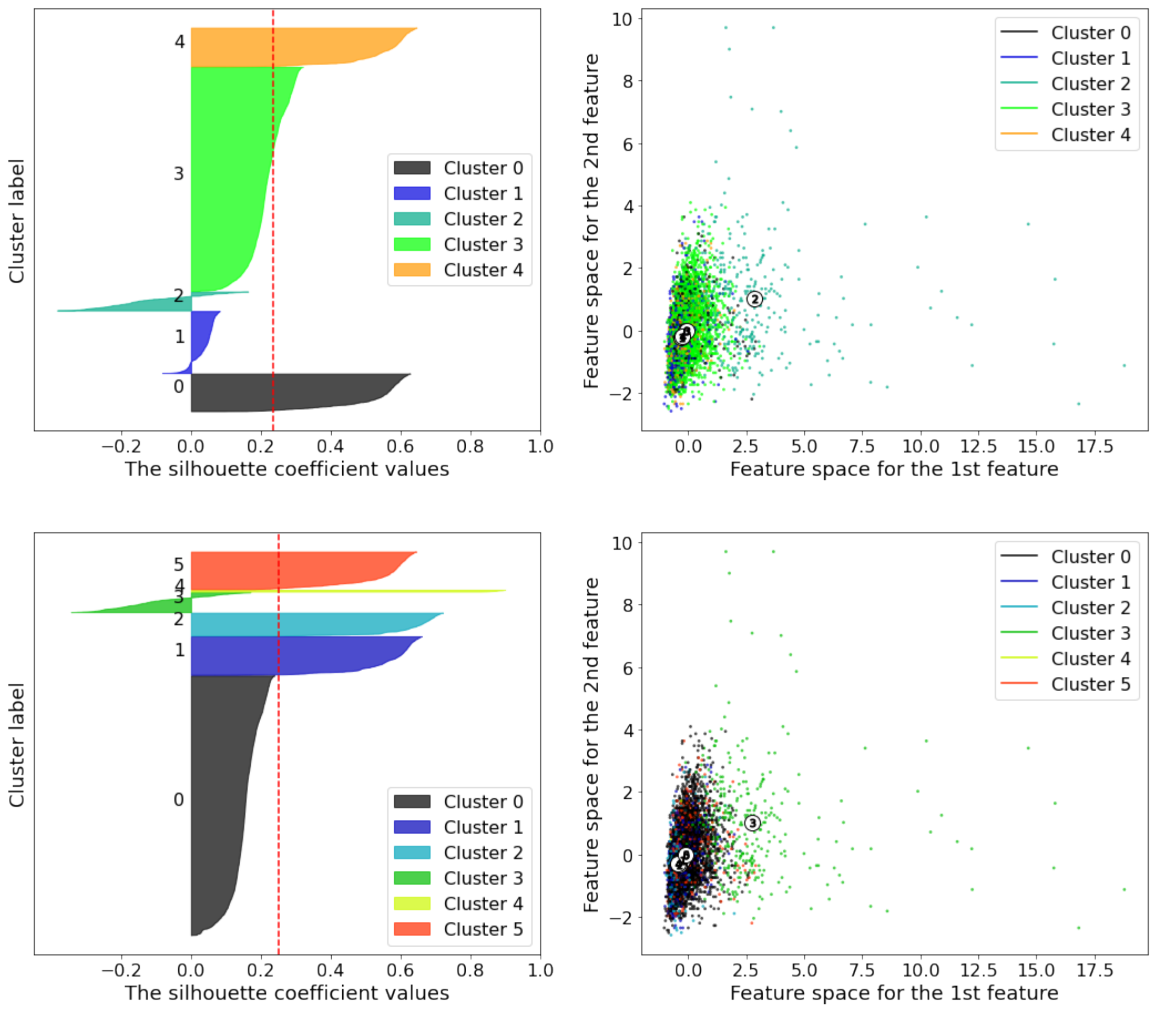
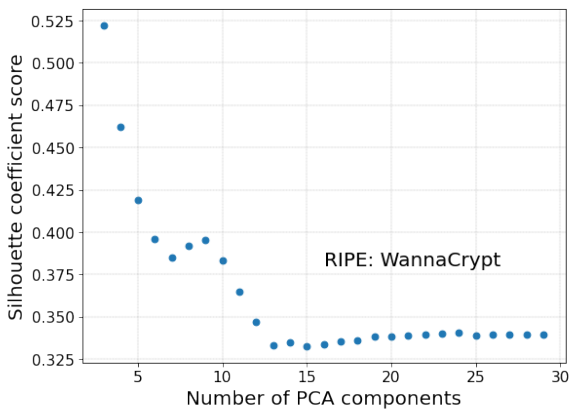


Figure 3.7: WannaCrypt RIPE training dataset: silhouette coefficient (left) and cluster scatter (right) plots for $k = 5$ and 6.



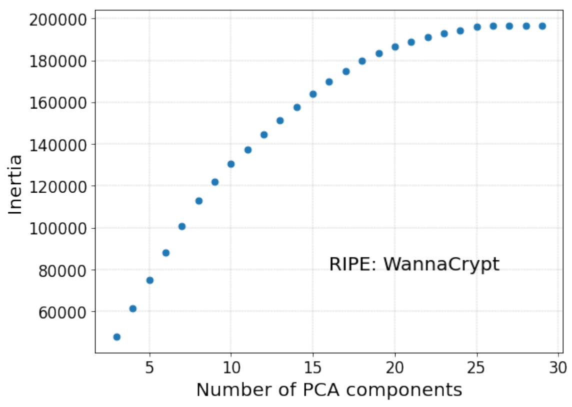


Figure 3.8: WannaCrypt RIPE data: k–means clustering measures: Silhouette coefficient (left) and inertia (right) vs. number of PCA components. The silhouette coefficients decrease with the increase in the number of PCA components. They indicate that clusters are more balanced for a smaller number of PCA components.

where X and Y are variables. Column vectors X and Y consist of n samples ($n \neq 1$) and i is the index variable. Variables x_i and y_i are i^{th} elements of vectors X and Y, respectively. Variables $\bar{x}$ and $\bar{y}$ are the mean values of X and Y, respectively.

Correlation is a statistical measure of linear relationships between vectors:

$$Correlation(X,Y) = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}, \tag{3.3}$$

where $Cov(X,Y)$ is the covariance of the two vectors X and Y while σ_X and σ_Y are standard deviations of vectors X and Y, respectively. The correlations between features are measured in terms of the Pearson and Spearman correlation coefficients.

Pearson correlation coefficient ρ is:

$$\rho = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}, \tag{3.4}$$

where n is the number of elements in vectors X and Y while i is the index term. The Pearson correlation coefficient is calculated to measure the linear correlation between variables. The Pearson correlation coefficient $\rho \in [-1, 1]$, where -1 indicates a strong negative linear relationship, 0 indicates no linear relationship, and $+1$ indicates a strong positive linear relationship between variables. Positive correlation values represent that the features will increase and decrease in the same direction. Negative correlation represents that features are negatively related to each other when both features will move in directions opposite to each other. For example, if two variables are negatively correlated, the increase in the value of one feature will decrease the value of the other feature. If the features are related to each other, they will have high correlation values and lead to the problem of multicollinearity [102, 103]. In machine learning, we are training models to find relationships between input data

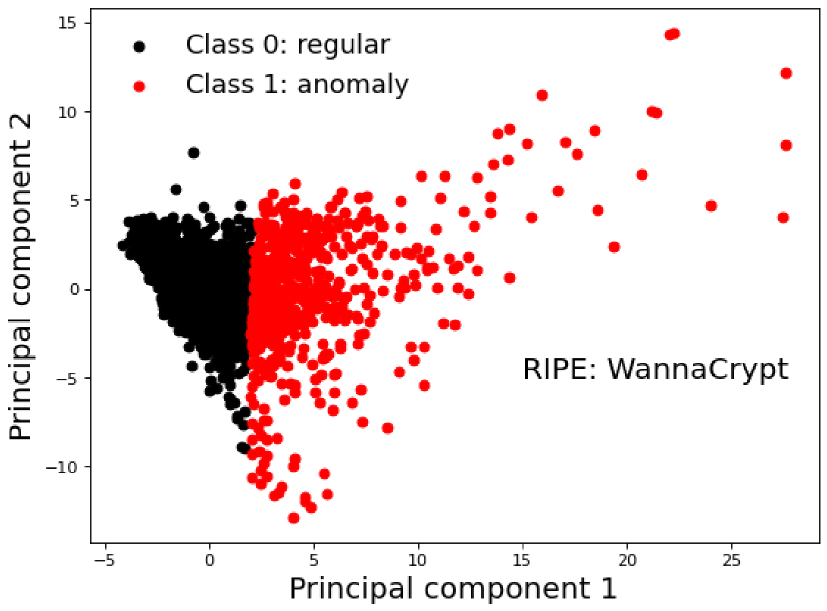


Figure 3.9: WannaCrypt RIPE data: k–means clustering on PCA (components = 3) transformed features. The PCA transformation applied to the training data has enhanced the separation between clusters.

and output labels. When input features are highly correlated with each other and not with the output label, they may not classify data points accurately. Therefore, to reduce redundancies, correlated features may be eliminated from datasets to generate new datasets for training machine learning models. Correlated features increase computational model cost and do not carry additional information relevant to the output. Therefore, they should be eliminated from the datasets to eliminate redundancies and enhance the performance of models. Pearson correlation coefficients are calculated under the assumption that the considered two variables are dependent on each other and are normally distributed.

3.2.2 Spearman Correlation

Pearson correlation coefficient captures only linear relationships, while nonlinear relationships between variables may be captured by measuring Spearman correlation coefficients. Other statistical measures may be evaluated to identify hidden data patterns and properties beyond linear data relationships. In monotonically related variables, change in one variable results in change in the other variable either in the same or opposite directions. Spearman correlation coefficient is a non-parametric statistical measure that evaluates monotonic relationships between variables [104]. It is employed to measure rank correlations (degree of similarity) between BGP features.

In order to calculate the coefficient of two vectors X and Y of size n, the data values should be ranked. The data are sorted in descending order, and values are then ranked so that the highest value has rank 1 and the lowest value has rank n. Spearman correlation coefficient is calculated as:

$$r_s(X, Y) = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)}, \tag{3.5}$$

where X and Y are column vectors of size $n > 1$, d_i is the i^{th} difference between the ranks of the i^{th} values of X and Y. Any correlated features should be eliminated from the datasets because they do not contain new information and may generate models that overfit.

The value of Spearman correlation coefficient $r_s \in [-1, 1]$. Values of $+1$ and -1 represent a very strong positive and negative monotonic relationship between the two variables, respectively. A value of 0 indicates no monotonic relationship between the two variables. The monotonic features might be selected to generate models based on the given problem. For example, in the case of small-sized datasets, generating models using monotonically related features might increase the risk of overfitting. Therefore, correlated features may be eliminated from the datasets as they do not provide additional information to generate a generalized model.

3.2.3 Supervised Machine Learning

Selecting relevant features to generate models may enhance the performance of models and reduce their computational cost. Supervised machine learning techniques are employed if data labels are available. The supervised feature selection methods are categorized as wrapper, filter, and intrinsic.

- Wrapper methods employ subsets of features to generate various models using machine learning algorithms. The performance of the generated models is inferred based on evaluation criteria, and the process is repeated until the features that generate the best performing model are obtained. Recursive Feature elimination (RFE) is a wrapper feature selection method.
- Filter methods evaluate relationships between input and target variables by calculating scores using statistical methods. The features are then ranked based on the score. They include statistical and feature importance techniques.
- Intrinsic feature selection methods are machine learning approaches that perform feature selection while generating models as a part of the training process. Examples of intrinsic feature selection models are tree and rule-based methods.

Decision trees [105] are a widely employed algorithm in data mining. The trees are built by recursively splitting nodes based on evaluation criteria. The decision tree model has a root node, branches, and internal and leaf nodes. The topmost node in a decision tree model is the root node. Tree branches direct data from the previous node to the next node after evaluating splitting criteria. Each internal node is labeled with an input feature. The nodes are split using a threshold value. Leaf nodes are nodes in a tree that contain predicted classification or regression output value.

During each iteration, the best features are selected by evaluating criteria to split the data further. Trees rely on evaluating metrics such as *Gini impurity*, *information gain*, and *entropy* [106]. Algorithms such as Iterative Dichotomiser 3 (ID3), C4.5, and Classification and Regression Trees (CART) are employed to determine the splitting criteria of nodes. The ID3 algorithm uses information gain and entropy to determine the splitting criteria. The C4.5 algorithm is a variation of the ID3 algorithm. It evaluates information gain to split nodes in decision trees. CART algorithm evaluates Gini impurity to identify the ideal splitting criteria.

The training dataset serves as the root node and each internal node (decision node) is labeled with a feature. The decision nodes are evaluated based on the decision criteria to best further split the data into sub-nodes. The internal nodes are selected so that they reduce the impurity of a tree. The tree branches until leaf nodes containing the expected category are obtained. The impurity of a decision tree is measured by calculating metrics such as Gini impurity, entropy, and information gain. Entropy is a measure of randomness (variance) in data. Data with high randomness will have high entropy. Information gain is a measure of randomness (variance) removed in data. Therefore, in decision trees, the desired spitting should be performed to remove the randomness in data so that each branch contains data belonging to a specific class [107]. For example, decision trees with the ID3 algorithm employ a greedy approach to obtain nodes with minimum entropy and maximum information gain. We employ random forests and extra-trees algorithms to identify relevant features.

3.2.3.1 Random Forest

Random forest [108] algorithm employs multiple decision trees to form predictions and Bootstrap Aggregation (Bagging) to generate multiple uncorrelated decision trees. Bagging is a technique of selecting samples with replacement, and a single sample may be selected multiple times to generate a tree. Each model is independently trained in parallel using samples selected by bagging. After decision trees are built, each classifier makes a prediction, and the outcome with a majority vote is selected as the output.

Random forests employ a random approach to select a subset of features and threshold values for splitting. The quality of a split is measured using the *Gini impurity*:

$$\begin{aligned} H(Q_m) &= \sum_k p_{mk}(1 - p_{mk}) \\ p_{mk} &= \frac{1}{n_m} \sum_{y \in Q_m} I(y = k), \end{aligned} \tag{3.6}$$

where Q_m is the data at node m, k is the class label, m is the node number, p_{mk} is the proportion of observation in node m. Random forests are generated by splitting data so that the data after the split result in reduced Gini impurity.

3.2.3.2 Extra-Trees

Extremely Randomized Trees (Extra-Trees) [109] algorithm, derived from random forest algorithm, is an ensemble method that uses multiple decision trees to calculate the final output. Extra-trees have faster execution time than random forest algorithms. Each decision tree is trained using a complete dataset without employing sampling techniques. The split point for each decision tree is selected randomly instead of searching for an optimal split point. The quality of the split is not evaluated, unlike the random forests algorithm. The generated trees are more diversified trees that are less prone to high bias and variance. The feature scores are calculated using *Gini importance* [110]:

$$Importance(\boldsymbol{X}_c) = \frac{1}{N_T} \sum_{T} \sum_{t \in T : v(s_t) = \boldsymbol{X}_c} p(t) \Delta i(s_t, t), \tag{3.7}$$

where $\boldsymbol{X}_c$ is the subset of input data $\boldsymbol{X}$ corresponding to a single feature, N_T is the number of trees, t is the node index in a tree, s_t is the direction of the split, $v(s_t)$ is a randomly generated threshold, $p(t)$ is the weight, and $\Delta i(s_t, t)$ is the decrease of the node impurity equivalent to its importance.

3.3 Feature Analysis

Data distributions are mathematically represented using Probability Distribution Functions (PDFs). The characteristics of probability distributions are defined by their first four moments: mean, standard deviation, skewness, and kurtosis [111]:

- The mean of a probability distribution is its average value.
- Standard deviation is a measure of the spread of a distribution from the mean value. Therefore, mean and standard deviations are location and scale parameters, respectively.
- Skewness and kurtosis are employed to infer the shape of a given distribution [112]. The skewness of a probability distribution is a measure of its symmetry. The shift in the shape of the data distribution to either the left or right side from the symmetrical (Gaussian) probability distribution curve determines its skewness. A distribution is positively skewed (right-tailed) or negatively skewed (left-tailed) if a majority of data points are located on the right or the left of the distribution median. The median value of a probability distribution is a central point of sorted values in a distribution.

The mean of a positively skewed distribution is larger than its median while the mean of a negatively skewed distribution is less than its median.

- Kurtosis is the measure of the number of data points residing further from the mean value. It identifies the heaviness of tails. Heavy kurtosis also indicates the presence of outliers. For a given distribution, the kurtosis is measured w.r.t the normal distribution. Note that the kurtosis of a normal distribution is three, and all measurements for kurtosis are obtained by subtracting three from the calculated value.

Machine learning approaches for anomaly detection heavily depend on the training and test data and their properties including probability distributions. Therefore, we analyze BGP features to estimate the best fitting distributions by measuring the skewness of worms and ransomware attack datasets.

3.3.1 Probability Distributions of BGP Features

The PDFs of Gaussian (normal), exponential, gamma, Weibull, Rayleigh, Burr, t Location-Scale, log-normal, and log-logistic distributions are used to estimate the best fit for BGP features. The evaluated distributions are listed in Table 3.1 [113]. The Gaussian, exponential, and gamma distributions are variations of a Poisson process [114]. The Gaussian (normal) distribution of a random variable X is defined as $X \sim N(\mu, \sigma^2)$, where μ is the mean (location parameter) and σ is the standard deviation (scale parameter). The Gaussian distribution is one of the most widely used symmetric distributions [115]. The exponential distribution of a random variable X is defined as $X \sim Exp(\lambda)$ where λ is a positive rate parameter. It is memoryless, and the event at time t is independent of the event occurring at time $t+1$. Its coefficient of variance is 1. It is used to calculate the probability of the first event's waiting time [116, 117]. The gamma distribution of a random variable X is defined as $X \sim \Gamma(k, \theta)$. It is a probability of the k^{th} future event. Its parameters are k shape and θ scale. When $k \leq 1$, the distribution resembles exponential distribution. With the increase in k, the distribution resembles a normal distribution.

The heavy-tailed distributions such as Weibull, Rayleigh, Burr, t Location-Scale, log-normal, and log-logistic [118, 119] are of interest because many processes in communication networks are heavy-tailed. The Weibull distribution is used to model skewed (left or right) data. It is defined by shape parameter γ, location parameter μ, and scale parameter α. The shape of the Weibull distribution falls between exponential and normal distributions based on the selection of parameters [116]. A widely used Rayleigh [120] distribution is a special case of two-parameter Weibull distribution [117, 121, 122] and has the scale parameter b. Burr [123] distribution is widely used to model heavy-tailed statistical events [124]. Burr (type XII) distribution is a right-skewed distribution also known as Pareto-IV or the Singh–Maddala distribution [115]. It is formed by combining Weibull and gamma distributions with scale parameter α and shape parameters c and k. The t Location-Scale is a heavy-tailed distribution that approaches the normal distribution as its shape parameter approaches infinity. The distribution uses location parameter μ, scale parameter σ, and shape parameter ν. A right-skewed probability distribution log-normal [125] is used to model distributions whose logarithm yields a normal distribution [126]. The PDF of log-normal distribution employs the location μ and the scale σ parameters. A special case of Burr distribution is the log-logistic heavy-tailed distribution [117]. It also has the location μ and the scale σ parameters.

Table 3.1: Probability distribution functions.

Distribution	**Expression**
Normal	$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-1}{2}(\frac{x-\mu}{\sigma})^2}$
Exponential	$f(x\|\lambda) = \lambda e^{-\lambda x}$; $x > 0$
Gamma	$f(x\|k,\theta) = \frac{1}{\theta^k \Gamma(k)} x^{k-1} e^{\frac{-x}{\theta}}$
	$\Gamma(k) = (k-1)!$
Weibull	$f(x\|\gamma,\mu,\alpha) = \frac{\gamma}{\alpha}(\frac{x-\mu}{\alpha})^{\gamma-1} e^{-(\frac{x-\mu}{\alpha})^{\gamma}}$
Rayleigh	$f(x\|b) = \frac{x}{b^2} e^{(\frac{-x^2}{2b^2})}$
Burr	$f(x\|\alpha,c,k) = \frac{\frac{ck}{\alpha}(\frac{x}{\alpha})^{c-1}}{(1+(\frac{x}{\alpha})^c)^{k+1}}$
t Location-Scale	$f(x\|\mu,\sigma,\nu) = \frac{\Gamma(\frac{\nu+1}{2})}{\sigma\sqrt{\nu\pi}\,\Gamma(\frac{\nu}{2})} \left[\frac{\nu+(\frac{x-\mu}{\sigma})^2}{\nu}\right]^{-(\frac{\nu+1}{2})}$
Log-normal	$f(x\|\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} \exp \frac{-(log(x)-\mu)^2}{2\sigma^2}$
Log-logistic	$f(x\|\mu,\sigma) = \frac{1}{\sigma x} \frac{e^z}{(1+e^z)^2}$
	where $z = \frac{log(x)-\mu}{\sigma}$

3.3.2 Goodness of Fit Test

The goodness of fit Kolmogorov–Smirnov (K-S) test is used to compare the reference probability distributions with the distribution of sampled data [127–129]. The goodness of fit tests that do not make assumptions about parameters of the sampled data are called non-parametric tests. They make no assumptions about the sample parameters. K-S test calculates the difference (D_n) between the Cumulative Distribution Function (CDF) of a reference probability distribution and the distribution of considered data. They are employed to compare sampled data distributions for given n sample data points [130, 131]. The D_n is calculated as:

$$max|F_n^1(x) - F_n^2(x)|, \tag{3.8}$$

where F_n^1 and F_n^2 are CDFs of random variable x. The two CDFs may be calculated using sampled data distributions or reference data and sampled data distributions. The CDF is calculated as:

$$F_n(a) = P(x \leq a), \tag{3.9}$$

where P is the probability function applied to a random variable x and the selected threshold value a [132].

Chapter 4

Machine Learning Approaches for Anomaly Detection

Machine learning is a process of generating models from data for predictions (classification). The performance of machine learning models is influenced by data quality, selected learning algorithm and loss function, presence of bias, variance, and noise [133]. Loss function measures the difference between the expected and predicted outputs during the training phase. Bias is the difference between expected and predicted outputs using the test data during the testing phase. Models with high bias and low variance do not learn the properties of the training data and, therefore, do not make accurate predictions using unseen data resulting in high prediction errors. Variance is the difference between the expected and predicted outputs achieved by models trained using different training datasets. Models with low bias and high variance learn only the properties of the employed training data. However, they do not make accurate predictions using unseen test data. In the case of underfitting, the model does not learn data properties during training to make accurate predictions using unseen data. While in the case of overfitting, the model closely learns the data properties during training. The performance of overfitted models is tailored to the properties of the training data, and they perform poorly on unseen data. Models with high bias and low variance underfit the training data, while models with high variance and low bias overfit the training data. These models are not generalized and lead to poor classification results. Therefore, a robust machine learning model should have low bias and low variance (referred to as the bias-variance trade-off) to generalize using the unseen data [134].

4.1 Support Vector Machine

SVM is a supervised machine learning algorithm that generates a decision boundary (hyperplane) to separate data points into distinct classes for classification tasks [29]. The decision surface is generated to maximize the distance (margin) between the closest data points belonging to distinct classes. The data points that lie close to the decision boundary are called support vectors. Data are either linearly or non-linearly separable. Kernel functions are designed to transform non-linearly separable data into higher dimensions such that the data becomes linearly separable in higher dimensional feature space. Linear hyperplanes are designed to linearly separate data to either side of the decision boundary. Examples of non-linear kernel functions include polynomial, Gaussian Radial Basis Function (RBF), and sigmoid kernels [135]. The decision boundary of SVM models is modified using the regularization parameter. Hard and soft margin kernels in SVM models are generated using high and low regularization values, respectively. Hard margin kernels are prone to overfitting.

4.2 Deep Learning Networks

Deep learning approach employs depth to map input and output values using an estimator function. Complex data relationships may be modeled by using nonlinear mapping called activation functions that transform input data to learn important features. A deep learning model consists of three or more layers: input, hidden, and output layers [93]. Artificial Neural Networks (ANNs) are deep learning models that consist of interconnected nodes with weighted connections. The ANNs interconnections may be cyclic or acyclic. Feedforward neural networks are networks with unidirectional connections. Multilayer Perceptron (MLP) network is an example of a widely used feedforward neural network. It is a multi-layer network with interconnected nodes between each subsequent layer. The learning is divided into steps by first mapping input to the subsequent layer and then mapping data from the intermediate layer to the final layer. The mappings of nonlinear data relationships are optimized using iterative approaches that employ the gradient of the loss function to update model hyperparameters. The important data features are learned by adjusting the weights of each layer using backpropagation to calculate the gradient of a loss function w.r.t the weight vector. The weights are adjusted in each layer of a network to obtain the minimum loss [96].

Deep learning is widely advocated for its performance using large datasets. With sufficient number of hidden units and appropriately selected activation function, neural networks approximate continuous functions. These activation functions are required to be nonlinear, bounded, decreasing, and continuous. Neural networks are also known as universal function approximators [136]. Examples of deep learning architectures include deep neural networks, deep belief networks, deep reinforcement learning [137], Recurrent Neural Networks (RNNs) [138], and Convolutional Neural Networks (CNNs) [139]. Deep learning is widely applied in computer vision, natural language processing, and anomaly detection [140]. Performance of deep learning networks depends on the selected number of hidden nodes, number of hidden layers, activation functions, and optimization algorithms.

4.2.1 Recurrent Neural Networks

RNNs are a class of ANNs that use recurrent (cyclic) connections between nodes and are employed to extract contextual information in sequential data. The memory of previous inputs is preserved in the internal state of the network to predict output with the help of recurrent connections. Although RNN models perform effectively using sequential data, their architecture suffers from vanishing or exploding gradient problem. The calculated gradient may be extremely large or small depending on the activation function used to train model layers. The chain rule is applied to compute gradients using backpropagation. As the n (number of layers) increases, the gradients will either explode or vanish. Approaches such as simulated annealing and discrete error propagation [141] were introduced in the 1990s to handle the vanishing gradient problem. However, they introduced other challenges such as delays, added time constants [142], and hierarchical sequence compression [143].

4.2.1.1 Long Short-Term Memory

LSTM algorithm was introduced to solve the vanishing gradient problem [144]. LSTM is an RNN known for its effective performance using sequential and temporal data [78]. It utilizes information between sequential events. An LSTM block consists of a memory block and has three gates: input (i), forget (f), and output (o). A single LSTM cell is shown in Fig. 4.1. The hidden states in RNNs are replaced with memory blocks in LSTM models. LSTM memory cells store sequential information to form memory. The new information is regulated using the input gate and it is updated only if the input gate is open. Otherwise, the old information is retained [145]. However, as the length of the sequence increases, the models are unable to retain information from previous sequences. Therefore, they might not perform as efficiently using data containing long sequences.

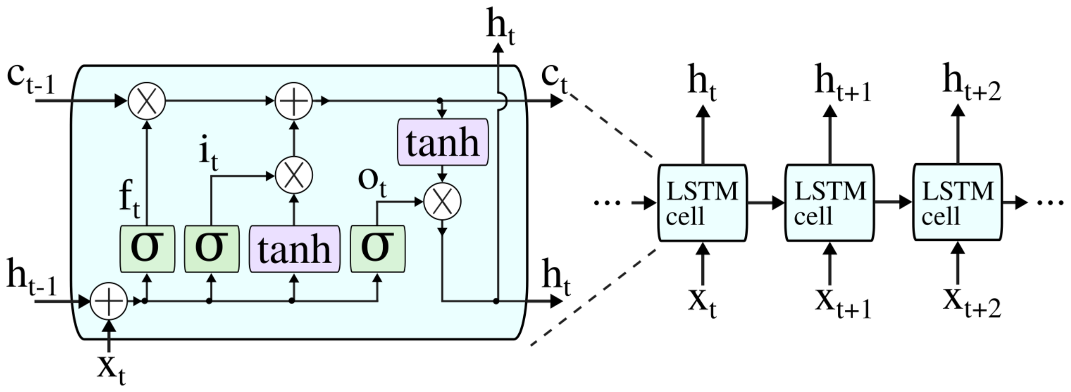


Figure 4.1: Neural network composed of LSTM cells. The current input x_t, previous cell state c_{t-1}, and output h_{t-1} are employed to calculate the current output h_t and cell state c_t [1].

The outputs of input (i_t), forget (f_t), and output (o_t) gates at time t are [146]:

$$\begin{aligned} i_t &= \sigma(W_{ii}x_t + b_{ii} + U_{hi}h_{t-1} + b_{hi}) \\ f_t &= \sigma(W_{if}x_t + b_{if} + U_{hf}h_{t-1} + b_{hf}) \\ o_t &= \sigma(W_{io}x_t + b_{io} + U_{ho}h_{t-1} + b_{ho}), \end{aligned} \tag{4.1}$$

where $\sigma(\cdot)$ is the sigmoid activation function with values $\in [0, 1]$, x_t is the current input, W_{ii}, U_{hi}, W_{if}, U_{hf}, W_{io}, and U_{ho} are the weight matrices, and b_{ii}, b_{hi}, b_{if}, b_{hf}, b_{io}, and b_{ho} are the bias vectors. The current cell state (c_t) is calculated using:

$$c_t = f_t * c_{t-1} + i_t * tanh(W_{ic}x_t + b_{ic} + U_{hc}h_{t-1} + b_{hc}), \tag{4.2}$$

where $*$ denotes element-wise multiplications, $tanh$ is the activation function with values $\in [-1, 1]$, W_{ic}, U_{hc} are weight matrices, b_{ic}, b_{hc} are bias vectors. The output of the LSTM cell is

$$h_t = o_t * tanh(c_t). \tag{4.3}$$

4.2.1.2 Gated Recurrent Unit

GRU is designed to solve the problem of vanishing or exploding gradients. It is a simplified version of the LSTM model and has two gates: update (z) and reset (r). The update gate regulates the information from previous time steps to calculate the next output. The reset gate regulates past information to be forgotten. The architecture of a GRU cell is shown in Fig. 4.2

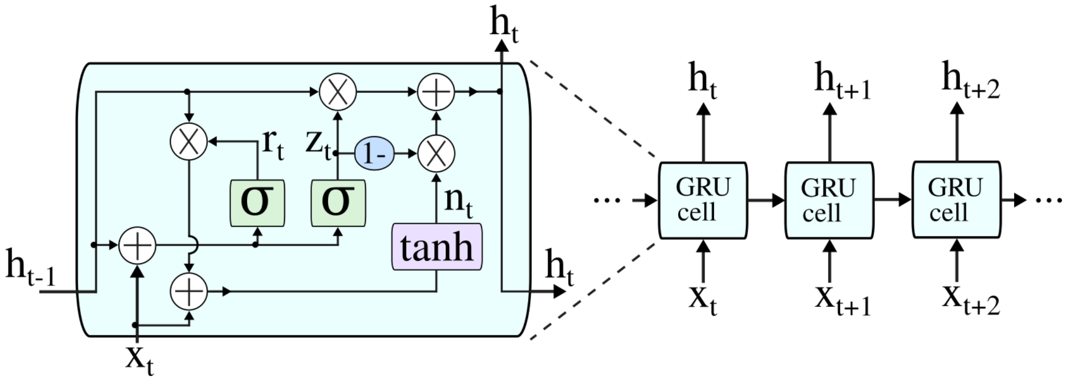


Figure 4.2: Neural network composed of GRU cells. The expanded architecture of the GRU cell with input x_t at time t, current output h_t, and past output h_{t-1} [1].

The outputs of the update gate (z_t) and the reset gate (r_t) at time t are [146]:

$$\begin{aligned} z_t &= \sigma(W_{iz}x_t + b_{iz} + U_{hz}h_{t-1} + b_{hz}) \\ r_t &= \sigma(W_{ir}x_t + b_{ir} + U_{hr}h_{t-1} + b_{hr}), \end{aligned} \tag{4.4}$$

where $(\cdot)$ is the sigmoid activation function with values $\in [0, 1]$, x_t is the current input, h_{t-1} is the previous cell output, W_{iz}, U_{hz}, W_{ir}, and U_{hr} are the weight matrices, and b_{iz}, b_{hz}, b_{ir}, and b_{hr} are the bias vectors. The output of the GRU cell is:

$$h_t = (1 - z_t) * n_t + z_t * h_{t-1}, \tag{4.5}$$

where n_t is:

$$n_t = tanh(W_{in}x_t + b_{in} + r_t * (U_{hn}h_{t-1} + b_{hn})), \tag{4.6}$$

tanh is the activation function with values $\in [-1, 1]$, W_{in} and U_{hn} are the weight matrices, and b_{in} and b_{hn} are the bias vectors.

4.3 Learning Rate Scheduling

Machine learning models are generated by learning patterns in the training dataset. The goal of machine learning is to train models so that they achieve the lowest possible loss. The loss functions are optimized using analytical or iterative approaches. Analytical optimization approaches are employed if the expected output and model hyperparameters are linearly related. If the relationship between expected output and model hyperparameters is nonlinear, they are optimized using iterative approaches. Examples of optimization approaches are least squares (analytical) and gradient descent (iterative).

The learning rate is an optimizer hyperparameter employed to solve the gradient-based minimization of loss functions. It is the rate at which model parameters such as weights are updated during backpropagation to reduce the training loss. Ideally, the loss function is desired to be differentiable and convex. However, the loss function may be multi-modal and include saddle points. The objective is to discover the minima of the loss function using optimization techniques. Gradient-based optimization approaches may iteratively converge to a minimum of the loss function. The optimization algorithm relies on update steps while performing iterations and searching the grid for the best performing values. The model may converge very slowly to the minima or even remain in a saddle point if small steps are taken to optimize the model. Conversely, the model may diverge from the optimal point if very large steps are taken to optimize the loss function. An ideal approach gradually converges to a minima point in a finite time. Therefore, managing the speed at which machine learning models learn is critical.

Leaning rate schedulers are pre-defined static or dynamic functions employed to update the learning rate parameter [147]. Static schedulers use constant learning rates initialized before training models. Dynamic schedulers are pre-defined functions that dynamically update learning rates in each training iteration. It is beneficial to initialize the learning rates to high values to obtain suitable model weights. The model is then fine-tuned by slowly reducing the learning rate to gradually converge to a minima. Popular learning rate schedulers include time-based decay, step decay, exponential decay, and adaptive optimizers.

Cosine annealing is a periodic dynamic learning rate scheduler. It is applied to periodically initialize the learning rate to a high value and then rapidly decrease its value. This periodic restart of the learning rate is employed to benefit from the restart of the learning process and to reuse the learned weights obtained using training data. The subsequent cycle uses weights learned during the previous cycle (warm restart) instead of starting with a new set of weights. The warm restart process is applied to optimize the learning rate and enhance the convergence rate [147].

4.4 Ensemble Learning

Ensemble learning is the branch of machine learning that combines multiple independent models called weak (base) learners to generate a final high performing prediction model [148]. This technique relies on using the strengths of each weak base learner to generate a final generalized (strong) model with a lower bias and variance. Ensemble learning is categorized as bagging, boosting, and stacking.

4.4.1 Bagging, Boosting, and Stacking

Datasets with missing values when used for training may generate models with high variance resulting in overfitting. Bagging (bootstrap aggregation) is employed to generate lower variance models than the individual weak learners to avoid overfitting. A single base learner is selected to train models with various datasets. Decision trees are the most commonly used base learners. Bagging employs the bootstrapping re-sampling technique that uniformly samples data using replacement to generate training sets. A data point might appear multiple times in a given training set. The selected bootstrap samples are then used to independently train multiple base learners in parallel. Bagging models employ deterministic approaches such as aggregation (regression) and majority vote (classification) to obtain the final output. A major drawback of this technique is the high computational cost associated with large-sized datasets due to the increase in the number of iterations [149,150]. Bagging is a widely adopted solution in healthcare, information technology, environmental studies, and finance. Random forests are an example of an algorithm that employs bagging [151].

Boosting is an effective approach to reduce data overfitting because it combines multiple classifiers and datasets. It is a variant of ensemble learning that sequentially combines multiple weak base learners to obtain a high performing model. This approach generates models by iteratively fitting data to multiple base learners. Heavier weights are assigned to the misclassified data points and the following model is generated by using these weighted data points for training. This approach for generating models by emphasizing misclassified data points generates low bias models. Two types of boosting techniques are: adaptive and gradient boosting. Adaptive Boosting (AdaBoost) initially assigns equal weights to all data points and then re-assigns weights after assessing the performance of the decision tree. Data points with heavier weights have a higher probability to be selected for the following model's training. Models are trained iteratively until the residual error is below the threshold [152, 153].

Gradient Boosting Machines (GBMs) [154] are greedy algorithms that generate models using forward stage-wise additive modeling. Models are sequentially added one by one in the case of forward stage-wise additive modeling. GBMs employ gradient using the loss function that should be differentiable. Its objective is to reduce the residuals of previous models. The negative gradient is calculated and the model hyper-parameters are then updated in opposite directions to optimize the model performance. Gradient boosting decision tree algorithms are a variant of GBMs.

Stacking is a process that uses a combination of multiple models trained using various algorithms for a given dataset. It consists of two-level architecture: first-level (level-0) and second-level (level-1) models. The first-level models are individual models that are trained on training datasets, while second-level models are used to combine the individual first-level models to generate the final output. In this thesis, we apply GBDT algorithms to generate classifiers using BGP data and evaluate model performances.

4.4.2 Gradient Boosting Decision Trees

Gradient boosting algorithms that use decision trees as their base learner are called Gradient Boosting Decision Tree (GBDT) algorithms. Widely used GBDT algorithms are: eXtreme Gradient Boosting (XGBoost) [155], LightGBM [40], and Categorical Boosting (CatBoost) [156]. The predicted output ($\hat{y}$) of a GBDT model during training is:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \tag{4.7}$$

where K is the number of estimators, f_k represents the k^{th} decision tree, X is the input data matrix, and x_i is the i^{th} row vector of matrix X. The output ($\hat{y}_i^{(k)}$) of the k^{th} iteration is:

$$\hat{y}_i^{(k)} = \hat{y}_i^{(k-1)} + f_k(x_i), \tag{4.8}$$

where $\hat{y}_i^{(k-1)}$ is the previous predicted output. The objective function of a GBDT model is:

$$L^{(k)} = \sum_{i=1}^{N} l(y_i, \hat{y}_i^{(k)}) + \Omega(f_k), \tag{4.9}$$

where $l(.)$ is the loss function, y_i is the expected label of the i_{th} input, and $\Omega(f_k)$ is the optional regularization term.

4.4.2.1 XGBoost

XGBoost is a scalable tree boosting algorithm that optimizes the loss function by adding the regularization term. The loss function cannot be optimized in the Euclidean space due to the presence of functions as parameters. Hence, an iterative additive approach is employed to train the model. The L^2 regularization term ($\Omega(.)$) is added to the loss function to reduce overfitting:

$$\Omega(f_k) = \gamma T + \frac{1}{T}\lambda||\omega||^2, \tag{4.10}$$

where γ and λ are the regularization coefficients, T is the number of leaves in the tree, and ω are the leaf weights. XGBoost employs second-order Taylor series approximation to simplify the loss function and discover optimal weights. The algorithm employs a histogram-based splitting using each feature and then selects the optimal split. The trees are generated using an asymmetric level-wise growth approach that splits the tree branches at a given level of the tree depth irrespective of the variance gain [157]. This approach may lead to overfitting and increased model complexity due to large tree depth and excessive number of branches. It might also construct highly biased models towards the majority class if generated using imbalanced datasets. Growing trees level-wise using large-sized datasets may be computationally expensive. Categorical features are features that have discrete values belonging to a defined number of categories. They may be encoded using a one-hot encoding that increases the number of binary features in a dataset if the number of categories for a given feature is large [158]. XGBoost employs one-hot encoding to convert categorical features to numerical features. XGBoost employs a cache-aware block structure to generate models faster by using parallel and distributed computing [155].

4.4.2.2 LightGBM

The LightGBM algorithm uses histogram-based Gradient-Based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) to enhance the training speed [40]. It requires lower memory than the XGBoost algorithm. It employs a histogram-based approach to discover faster the best splitting point for each feature. Feature histograms are created by bundling together mutually exclusive features. The feature bundling results in a lower dimension than the original feature space and increases computational speed. The GOSS technique sorts training data in descending order based on the absolute gradient values. Top N_t data points are selected using the sampling ratio a with the largest gradient to create a subset A. The random sampling is performed using sampling ratio b on the remaining low gradient data points to create a subset B. Note that during training gradients are calculated in each iteration. GOSS retains data points with high gradient values and does not use data points with low gradients to train the subsequent models because they contribute to high information gain. This approach is more effective for calculating information gain rather than a uniform random sampling of data points. LightGBM trees employ asymmetric leaf-wise growth by selecting the leaf with the maximum loss. LightGBM is more effective than using level-wise tree growth. LightGBM trees are asymmetrically grown by splitting the leaves and, hence, the model may potentially overfit the data. LightGBM employs gradient statistics to handle categorical features. The gradient statistics in LightGBM are calculated in each iteration and for each category. This might increase the associated computational cost and memory consumption. Training GBDT models based on the residuals increase the bias and may increase overfitting.

4.4.2.3 CatBoost

CatBoost is a scalable algorithm designed to handle categorical features [156]. It employs an ordered target statistic approach to convert the categorical features to numerical features based on the expected output labels (classes) while keeping the dimension of the original data. The ordered boosting employs permutations to train and evaluate decision trees using different sets of samples. It is employed to prevent data overfitting. CatBoost trees are grown level-wise using the same splitting criteria for a given tree level (oblivious trees). The generated trees are symmetric and less prone to overfitting. The CatBoost models have faster training time and perform effectively using smaller datasets. Minimal Variance Sampling (MVS) is a random sampling technique employed to sample data to train stochastic gradient boosting models. CatBoost employs a weighted version of MVS. The data are sampled so that each data point is selected at least once. This might reduce the model variance and, thus, optimize model performance. CatBoost also performs effectively when used with numerical and text features.

4.5 Attention Mechanism

The length of the sequence may become a challenge when dealing with sequential and temporal data. As the duration of the sequence increases, gradient-based learning algorithms may suffer from vanishing gradient problem and the past information of long sequences is not retained. For example, in an encoder-decoder model, the entire encoded input information is encapsulated into a context vector of fixed length (usually smaller than the input vector). The context vector is the last hidden state of the encoder model. The decoder uses this context vector to generate the decoded output. However, as the length of the input sentence increases, the fixed-length context vector does not encode information from earlier input sequences [159].

Attention mechanism [160, 161] overcomes the shortcoming of the fixed-length context vector that loses information as the size of the input vector increases. To form predictions, it relies on using selective information from a sequence instead of using the entire sequence [159]. To calculate the output, the relevant information is retained from the entire input by using weighted combinations of hidden states of the encoder. This weighted combination (context vector) helps emphasize the contribution of the local sequences in detecting patterns in long sequences. The context vector of each input keeps relative information of the previous sequences to obtain the desired output. This mechanism has been implemented in diverse fields such as computer vision, speech processing [162], image captioning [163], and Neural Machine Translation (NML) [164]. Attention mechanism has been proved to enhance the model performance by highlighting the desired sequences during the training phase.

The context vector c_t at time t is calculated using alignment scores $e_{t,i}$ and attention weights $\alpha_{t,i}$:

$$\begin{aligned} e_{t,i} &= a(s_{t-1}, h_i) \\ \alpha_{t,i} &= softmax(e_{t,i}) \\ c_t &= \sum_{i=1}^{T} \alpha_{t,i} h_i, \end{aligned} \tag{4.11}$$

where $a(.)$ and softmax are the activation functions, s_{t-1} is the previous hidden decoder state, h_i is the current encoded hidden state, and T is the number of hidden states.

4.6 Performance Metrics

Datasets may be balanced or unbalanced based on the number of data instances that belong to each class. Balanced datasets consist of a uniform number of data instances in each class while the data instances belonging to a particular class outnumber the data points belonging to the other classes in unbalanced datasets. Hence, there is a higher probability of classifying data points that belong to the majority class.

Performance of models may be measured using training time, confusion matrix, precision, recall, accuracy, and F-Score 4.12. The confusion matrix is generated by calculating: true positive (TP), true negative (TN), false positive (FP), and false negative (FN).

$$\text{precision} = \frac{TP}{TP + FP}, \tag{4.12}$$

$$\text{sensitivity (recall)} = \frac{TP}{TP + FN}, \tag{4.13}$$

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{4.14}$$

$$\text{F-Score} = 2 \times \frac{\text{precision} \times \text{sensitivity}}{\text{precision} + \text{sensitivity}}. \tag{4.15}$$

True positives and true negatives are data points that are correctly classified as anomalies and regular data points, respectively. False positives and negatives are data points that are incorrectly classified as anomalies and regular data points, respectively.

- Precision is the measure of correctly identified positive cases (TPs) from all identified positives (TPs + TNs) by the classifier model. It is beneficial to calculate when the cost of classifying false positives is high.
- Recall is the measure of correctly identified positive cases (TPs) from all the identified positives (TPs + FPs). It is an important measure when the cost of classifying false negatives is high.
- Accuracy is the measure of correctly identified anomalies (TPs) and regular data points (TNs) from all classified data. It may be employed when the cost of identifying each class is equal. It may be a misleading performance measure in the case of unbalanced datasets where the probability of identifying data belonging to one class is higher.
- F-Score is the harmonic mean of the precision and recall. It measures the correctly classified true positive data points (TPs). Therefore in the case of unbalanced datasets, F-Score is a better performance metric than accuracy.

4.7 Cross-Validation

Cross-validation generates a distribution of training and validation datasets from a single dataset. The input dataset is partitioned into k equally sized subsets $S_i : S_1, ..., S_k$, where $i = 1, ..., k$ are known as folds. After partitioning the dataset, the learning algorithm is applied k times to the selected pair of training and validation datasets. Cross-validation is a time-consuming process employed to select hyperparameters of the best-performing model. Various approaches may be applied to select subsets of training and test datasets in each iteration. In this thesis, we employ the 10-fold time series split cross-validation: the i^{th} subset in the i^{th} iteration is selected as the test dataset where $i = 1, ..., k = 10$. The i^{th} and $(i+1)^{th}$ subsets are selected as the training and test datasets, respectively. The cross-validation techniques are employed to obtain generalized models. Time series split cross-validation is employed to maintain the time sequence of temporal data. An example of the 5-fold time series split cross-validation is shown in Fig 4.3.

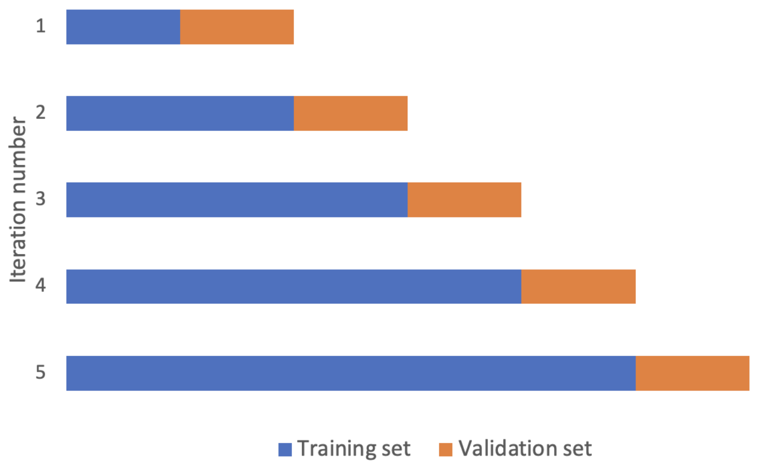


Figure 4.3: Time series split cross validation for $k = 5$. In the first iteration, the first subset is used for training while the second subset is used for validation.

Chapter 5

Performance of Algorithms Used for Dimension Reduction and Feature Selection

Dimension reduction and feature selection techniques are applied to eliminate redundancies and thus identify the most relevant features. Dimension reduction techniques such as autoencoders and PCA may be employed for data compression. Autoencoders are unsupervised neural networks that are employed to learn a representation of a dataset [96]. They consist of an encoder, a central layer, and a decoder. The encoder and decoder networks consists of an equal number of layers and a number of nodes. The central layer in an autoencoder is called the latent or context vector. The central layer is shared between the encoder and decoder network. The encoder and decoder networks are used for data compression and reconstruction, respectively. The Input data is mapped to a lower dimension using an encoder network to the central layer. This compressed data (output of the central layer) is then used as input to the decoder that reconstructs the original input. Autoencoders consisting of a single fully-connected hidden layer, linear activation function, and mean squared loss function are identical to PCA. We employ PCA to transform data to a lower dimension. PCA generates principal components that are orthogonal and uncorrelated to each other. Pearson and Spearman correlations are evaluated to identify highly correlated features. Features with high correlations are eliminated to identify the relevant features, thus selecting important features. Supervised feature selection techniques such as tree-based machine learning algorithms (random forests and extra-trees) are employed to identify the most important features using rank-based criteria.

5.1 Dimension Reduction Using Principal Component Analysis

Experimental data used for PCA are first normalized using z-score. In experiments, we select the number of principal components to preserve $\approx 70\%$ of the variance [98]. The data variance in each BGP training dataset using 10 PCA components is listed in Table 5.1. Performance of machine learning models depends on the combinations of selected features [16]. It is important to have a spatial separation between features to accurately classify data. PCA is applied to enhance the data separation between features because the generated principal components are orthogonal to each other. If the features are well separated the data points belonging to regular and anomaly classes may be well separated. This may enhance the classification performance of models.

Table 5.1: Variance retained in BGP training datasets that consists of the 10 selected principal components.

Dataset	**Variance (%)**
Code Red	69.58
Nimda	68.46
Slammer	67.70
WannaCrypt	71.70
WestRock	69.02

5.1.1 Feature Selection Using Correlation

Correlation is a statistical technique employed to measure the strength of a relationship between two features. It is desirable in machine learning techniques to have features related to the output. Models generated using data consisting of features correlated with each other they may contain duplicate information and may not be able to generate high performing models. Therefore, feature selection may be performed by eliminating highly correlated features. We calculate Pearson and Spearman correlation coefficients to identify correlated features in worms and ransomware attack BGP datasets. Pearson and Spearman correlation coefficients are employed to identify linear and nonlinear relationships between features, respectively.

5.1.1.1 Pearson Correlation

A Pearson correlation matrix consists of Pearson correlation coefficients computed for all combinations of feature pairs. The Pearson correlation matrix calculated using WannaCrypt RIPE training data is shown in Fig. 5.1. The diagonal of the Pearson correlation matrix is equal to 1. The value of the Pearson coefficient corresponds to the color and filling of squares. The squares filled with dark blue and red colors correspond to $+1$ and -1, respectively.

The intensity of the color fades as the value of the coefficient decreases. Weakly correlated features are identified with smaller shaded portions.

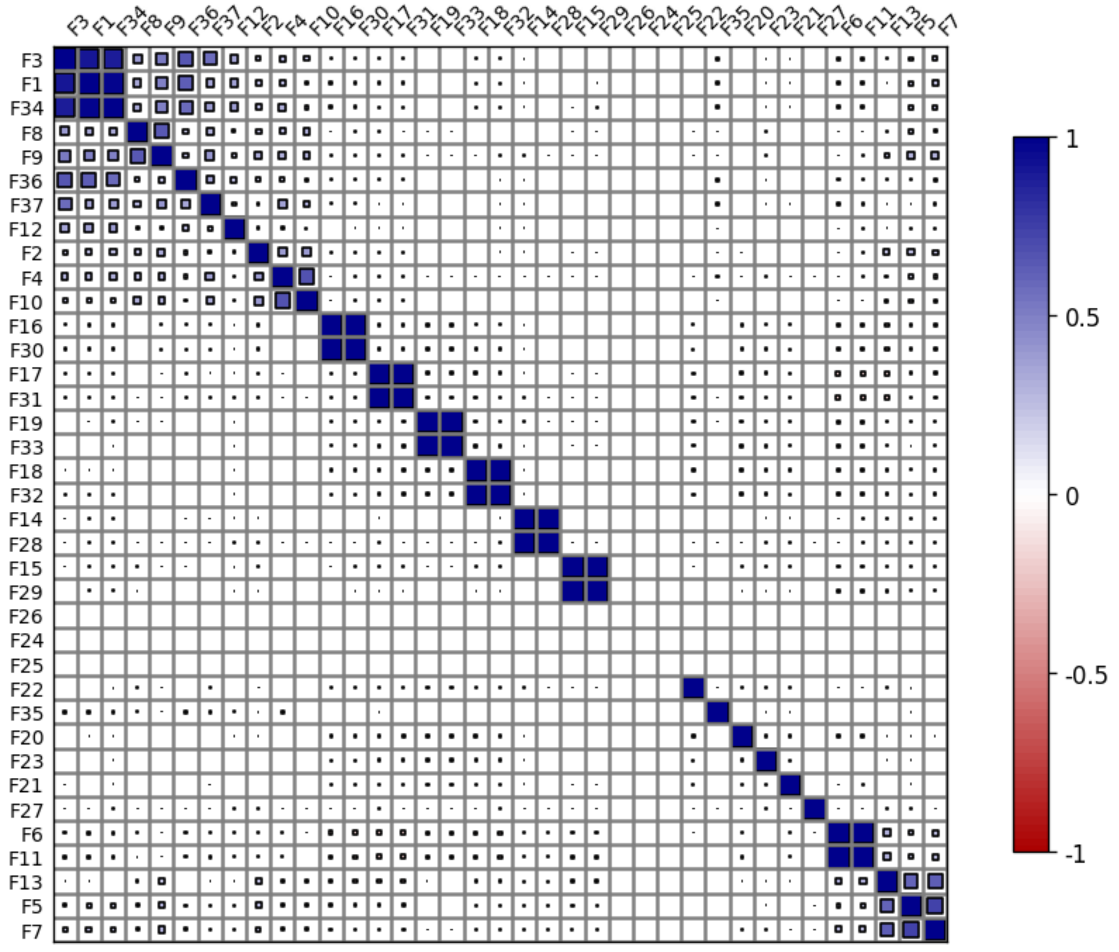


Figure 5.1: WannaCrypt RIPE training dataset: Pearson Correlation matrix. The strength of the correlation between features is indicated by color strengths. The squares filled with dark blue and red colors correspond to $+1$ and -1, respectively.

The Pearson correlation matrix is symmetric and, hence, the lower triangle of the matrix shown in Fig. 5.2 is used to identify strongly correlated features. Features with a Pearson correlation coefficients ≥ 0.90 are considered highly correlated. Pearson correlation coefficient of features $F24$, $F25$, and $F26$ are 0 because the values of these features, which represent the maximum edit distance for $n = 7,\ 8$, and 9, are 0.

The pairs of positively correlated features belonging to BGP worms and ransomware attack datasets with correlation coefficient ≥ 0.90 are listed in Tables 5.2 and 5.3, respectively. One feature from each correlated feature pair is selected as an important feature in both training and test datasets. For example, $F1$ of the Code Red dataset is highly correlated with $F3$ and $F36$ with a high correlation coefficient of 0.97. Correlated features contain du-

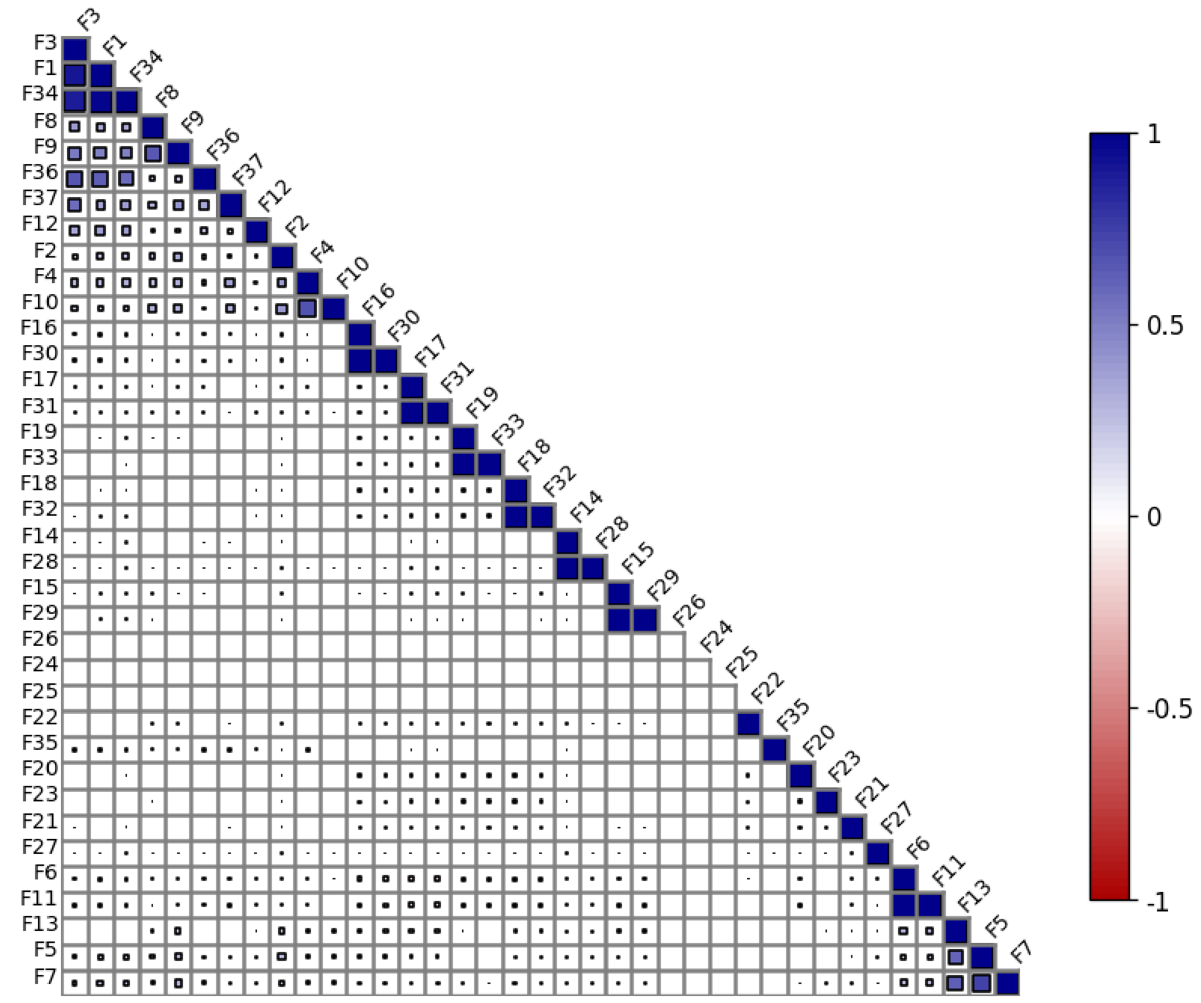


Figure 5.2: WannaCrypt RIPE training dataset: lower Pearson correlation matrix. The intensity and size of the filled color fades as the value of the coefficient decreases.

plicate information, therefore, $F1$ is selected to be removed from training and test datasets. The majority of the highly correlated features are identical for the three worms. Linear relationships are observed between the maximum AS-path and edit distance features, indicating an increase in AS-path lengths due to the announced malicious prefixes that may reside at a geographically far location. The feature $F34$ (number of IGP packets) is highly correlated to $F36$ (number of incomplete packets), $F3$ (number of announced NLRI prefixes), and $F1$ (number of announcements). This may indicate that during the worm attacks the increase in the announced prefixes and announcements from within the AS and number of packets that have an unknown source of origin increases. It may indicate that during these worm attacks, the infected systems residing in a given AS are used to propagate the malicious prefixes. These NLRI prefixes to be announced are generated by worms and therefore have no source of origin. The majority of correlated features for ransomware attack datasets are

AS-path features. Linear relationships are also observed between the maximum AS-path and edit distance, indicating an increase in AS-path lengths during ransomware attacks. It is also observed that feature $F34$ (number of IGP packets) is highly correlated with $F1$ (number of announcements). This may indicate that upon infection the false information is immediately widely spread within the AS.

Table 5.2: Worms BGP RIPE datasets: Feature pairs with strong positive Pearson correlation coefficients ($\rho \geq 0.90$).

Dataset	**Feature**	**Correlated features**	**Correlation value**
Code Red	F1	F3, F36	0.97, 0.97
	F7	F5	0.90
	F11	F6	0.99
	F28	F14	0.99
	F29	F15	0.99
	F30	F16	0.99
	F31	F17	0.99
	F32	F18	0.99
	F33	F19	1.0
	F34	F3, F36, F1	0.97, 0.97, 0.99
	F36	F3	0.96
Nimda	F1	F3, F36	0.99, 0.98
	F4	F2	0.99
	F7	F5	0.90
	F11	F6	0.99
	F28	F14	0.99
	F29	F15	0.99
	F30	F16	0.99
	F31	F17	0.99
	F32	F18	0.99
	F33	F19	1.0
	F34	F3, F36, F1	0.99, 0.98, 0.99
	F36	F3	0.98
Slammer	F1	F36, F3	0.93, 0.97
	F11	F6	0.99
	F28	F14	0.99
	F29	F15	0.99
	F30	F16	1.0
	F31	F17	0.99
	F32	F18	0.99
	F33	F19	0.99
	F34	F36, F3, F1	0.92, 0.97, 0.99

Table 5.3: Ransomware attack BGP RIPE datasets: Feature pairs with strong positive Pearson correlation coefficients ($\rho \geq 0.90$).

Dataset	Feature	Correlated features	Correlation value
WannaCrypt	F11	F6	0.99
	F28	F14	1.0
	F29	F15	1.0
	F30	F16	1.0
	F31	F17	0.99
	F32	F18	1.0
	F33	F19	0.99
	F34	F1	0.98
WestRock	F1	F3	0.97
	F11	F6	1.0
	F28	F14	1.0
	F29	F15	1.0
	F30	F16	1.0
	F31	F17	1.0
	F32	F18	1.0
	F33	F19	1.0
	F34	F3, F1	0.94, 0.96

5.1.1.2 Spearman Correlation

Spearman correlation coefficient is calculated to identify and measure the strength of the nonlinearly related features. We consider pairs of features with Spearman correlation coefficient ≥ 0.90 as strongly correlated features. One feature from each correlated feature pair is selected as an important feature in both training and test datasets. Experiments using training datasets are performed to identify nonlinear relationships between BGP features. The code used is listed in "appendix_spearmancorrelation.m" Appendix B. The pairs of features with high Spearman correlation coefficients for worm and ransomware datasets are listed in Table. 5.4 and 5.5, respectively. A number of correlated features are common in the considered experimental datasets indicating their importance in analyzing BGP anomalies. In the case of worm datasest, feature $F34$ is nonlinearly correlated with features $F1$ and $F12$ indicating a change in the arrival rate of packets results in a change in the number of IGP packets. In the case of ransomware attack datasets, the feature $F34$ is also nonlinearly correlated to $F12$. This may be influenced due to the rate at which the infected hosts are announcing paths within an AS.

5.1.2 Feature Selection Using Random Forests

Important features are selected based on *Gini impurity* using random forests. We generate random forests model using only one parameter number of estimators and values of other parameters are kept default [165]. The best number of estimators are obtained by using

Table 5.4: Worm BGP RIPE datasets: Spearman correlation coefficient ≥ 0.90.

Feature	**Correlated features**	**Correlation value**
Code Red		
F11	F6	1.0
F12	F1	0.95
F28	F14	0.99
F29	F15	1.0
F30	F16	1.0
F31	F17	1.0
F32	F18	1.0
F33	F19	1.0
F34	F1, F12	0.98, 0.93
Nimda		
F3	F1	0.93
F11	F6	1.0
F12	F1, F3	0.97, 0.90
F28	F14	1.0
F29	F15	1.0
F30	F16	1.0
F31	F17	1.0
F32	F18	1.0
F33	F19	1.0
F34	F1, F3, F12	0.99, 0.92, 0.96
Slammer		
F3	F1	0.92
F11	F6	1.0
F12	F1, F3	0.99, 0.91
F28	F14	1.0
F29	F15	1.0
F30	F16	1.0
F31	F17	1.0
F32	F18	1.0
F33	F19	0.99
F34	F1, F3, F12	0.99, 0.91, 0.98

10-fold time series split cross-validation. The models employed to obtain important features are trained based on accuracy or F-Score. In our experiments, the obtained best number of estimators performed using 10-fold time series split cross-validation based on accuracy or F-Score are identical. The top 16 features and number of estimators based on accuracy or F-Score are listed in Table. 5.6 and 5.7, respectively. The features are listed in decreasing order of importance. The selected features and the number of estimators based on accuracy or F-Score are identical. Unlike ransomware attacks datasets, the important features of the BGP worm datasets are identical for all worms.

Table 5.5: Ransomware attack BGP RIPE datasets: Spearman correlation coefficient ≥ 0.90.

Feature	Correlated features	Correlation value
WannaCrypt		
F11	F6	1.0
F12	F1	0.91
F28	F14	1.0
F29	F15	1.0
F30	F16	1.0
F31	F17	1.0
F32	F18	1.0
F33	F19	1.0
F34	F1, F12	0.94, 0.96
WestRock		
F11	F6	1.0
F28	F14	1.0
F29	F15	1.0
F30	F16	1.0
F31	F17	1.0
F32	F18	1.0
F33	F19	1.0
F34	F12	0.96

Table 5.6: BGP datasets: important features and number of estimators using random forest based on accuracy.

Dataset	Feature numbers in order of importance	No. of estimators
Code Red	34, 1, 3, 4, 12, 36, 9, 37, 8, 10, 2, 5, 11, 6, 7, 13	10
Nimda	1, 4, 34, 3, 12, 36, 9, 37, 8, 10, 6, 2, 11, 13, 7, 5	110
Slammer	1, 34, 36, 12, 4, 3, 10, 2, 8, 9, 13, 37, 6, 11, 7, 5	210
WannaCrypt	4, 8, 10, 3, 9, 2, 1, 34, 36, 37, 12, 6, 11, 13, 35, 7	90
WestRock	36, 1, 8, 3, 9, 34, 10, 37, 4, 11, 2, 6, 12, 22, 5, 13	180

Table 5.7: BGP datasets: important features and number of estimators using random forest based on F-Score.

RIPE Dataset	Feature numbers in the order of importance	No. of estimators
Code Red	34, 1, 3, 4, 12, 36, 9, 37, 8, 10, 2, 5, 11, 6, 7, 13	10
Nimda	1, 4, 34, 3, 12, 36, 9, 37, 8, 10, 6, 2, 11, 13, 7, 5	110
Slammer	1, 34, 36, 12, 4, 3, 10, 2, 8, 9, 13, 37, 6, 11, 7, 5	210
WannaCrypt	4, 8, 10, 3, 9, 2, 1, 34, 36, 37, 12, 6, 11, 13, 35, 7	90
WestRock	36, 1, 8, 3, 9, 34, 10, 37, 4, 11, 2, 6, 12, 22, 5, 13	180

5.1.3 Feature Selection Using Extra-Trees

Experiments are performed using worm and ransomware attack BGP datasets to identify the most important features. The top 16 important features listed in Table 5.8 ranked based on *Gini importance* [110].

The importance of top 16 features of ransomware attack BGP datasets are shown in Fig. 5.3. The 10-fold time series split cross-validation is performed to obtain the best hyper-parameter using extra-trees models. The number of estimators and maximum tree depth are 500 and 20, respectively. The important features obtained using extra-trees algorithm are obtained using the BGPGuard tool [166] and its "feature_select_plot_cnl.py" file. While the most important features are common for the three worm datasets, the important features are identical for the two ransomware datasets.

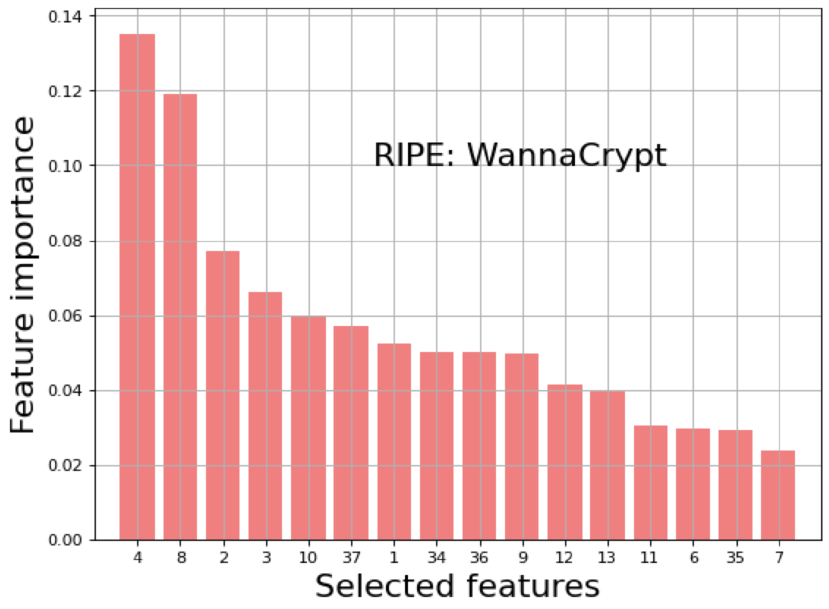
Table 5.8: BGP datasets: important features based on accuracy or F-Score evaluated using the extra trees.

Dataset	**Features in order of importance**
Code Red	34, 1, 4, 3, 12, 2, 9, 37, 36, 8, 10, 13, 5, 7, 35, 6
Nimda	1, 34, 3, 4, 9, 36, 12, 37, 8, 23, 10, 2, 13, 7, 11, 5
Slammer	36, 1, 9, 34, 10, 8, 3, 4, 2, 20, 11, 12, 6, 13, 5, 7
WannaCrypt	4, 8, 2, 3, 10, 37, 1, 34, 36, 9, 12, 13, 35, 11, 6, 7
WestRock	8, 9, 3, 37, 2, 1, 36, 34, 10, 4, 12, 35, 13, 6, 11, 7

5.2 Goodness of Fit Test

Experiments are performed using the BGP RIPE datasets to identify the best-fitting feature distributions. We perform goodness of fit K-S tests using pre-selected PDFs and selected features of Code Red, Nimda, and Slammer worms, and WannaCrypt and WestRock ransomware attack datasets. The extra-trees algorithm is used to select the top 10 features shown in Table 5.8. The best extra-trees parameters are selected by performing 10-fold time series cross-validation. The number of estimators and maximum tree depth are 500 and 20, respectively. The curve fitting is then performed on individual features using MATLAB. The PDFs of BGP features and Gaussian (normal), exponential, gamma, Weibull, Rayleigh, Burr, t Location-Scale, log-normal, and log-logistic are shown in Fig. 5.4. Visual inspection of distributions is used to select the suitable candidates for best fitting distributions to perform the K-S test.

Exponential, Weibull, Burr, t location-scale, log-normal, and log-logistic PDFs that closely fit the important BGP features are selected by visual inspection before performing the K-S test for various significance levels (α). The K-S test results of the evaluated BGP features for worm and ransomware datasets are shown in Table 5.9 and Table 5.10, respectively. The K-S test statistic measures h, p, k, and c are calculated. The h value of



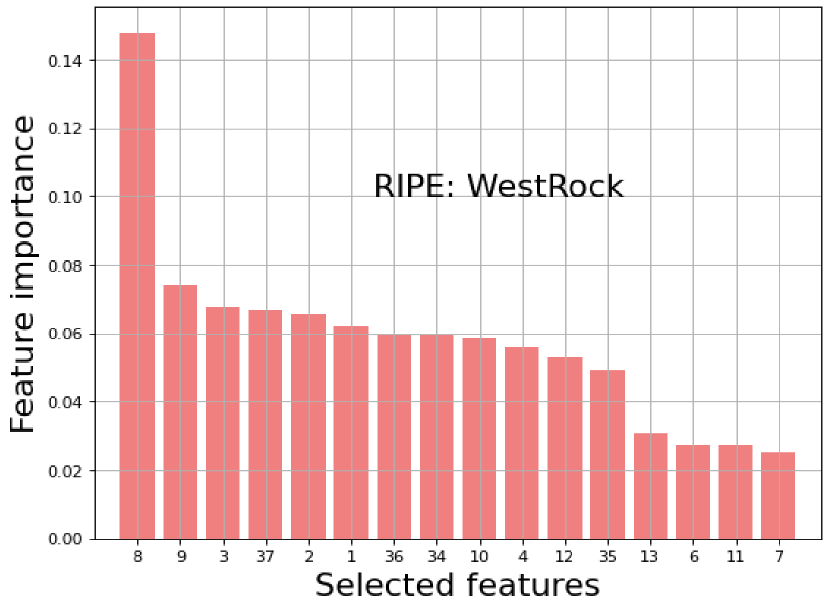


Figure 5.3: Feature importance of top 16 important features of WannaCrypt and WestRock datasets calculated using extra-trees algorithm. Note that the top 16 important features for both datasets are identical for ransomware attacks that occurred a decade apart.

0 indicates that the null hypothesis (H_0) is accepted. It is rejected otherwise. The p-value of the hypothesis test measures the probability of obtaining the observed results when the null hypothesis is true. Whenever the p-value is $\geq \alpha$ the null hypothesis is accepted. The Critical Value (CV) of the hypothesis test, derived from the p-value, is the value after which the null hypothesis is rejected. A high cv highlights the differences between distributions.

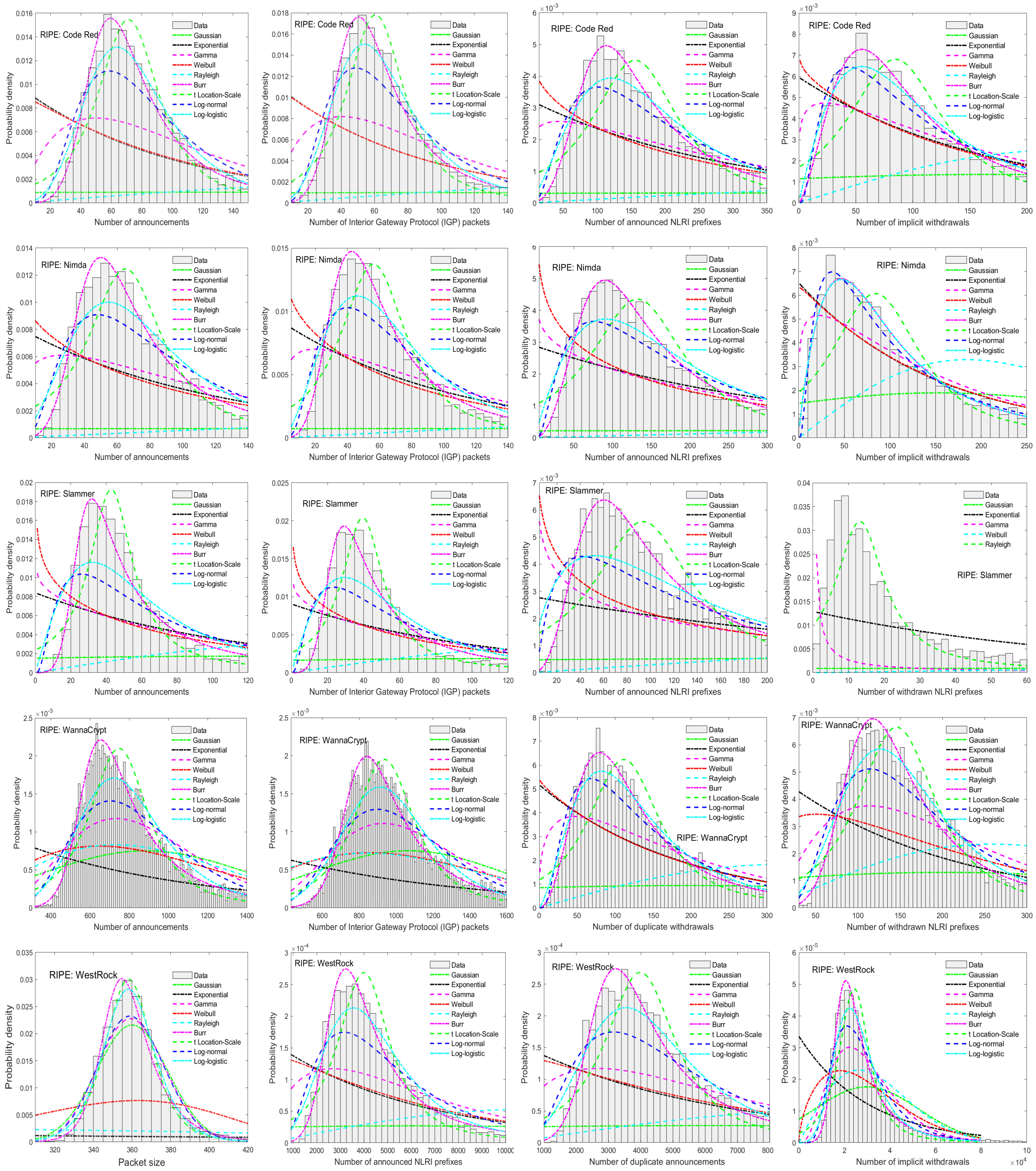


Figure 5.4: Features with at least one PDF accepted using the K-S test. BGP data: Code Red (1st row), Nimda (2nd row), Slammer (3rd row), WannaCrypt (4th row), and WestRock (5th row). Nine PDFs are considered. Heavy-tailed PDFs are the best fit for features F1, F3, F4, F9, F10, and F34.

The k measure is the maximum difference between the CDFs of the data and the reference distributions [167–169].

The fitting distributions and features with accepted H_0 are listed in Table 5.11. Highlighted are distributions and features with high p-values. The estimated parameters for best fitting Burr, log-normal, and log-logistic distributions are listed in Table 5.12. The Burr distribution is the closest fit for the accepted features of Code Red and WannaCrypt datasets due to the right-skewed nature of data.

- Code Red dataset: The null hypothesis for Burr distribution is accepted for features $F34$ (number of IGP packets), $F1$ (number of announcements), $F3$ (number of announced NLRI prefixes), and $F9$ (number of implicit withdrawals) of Code Red dataset.

- Nimda dataset: The null hypothesis is accepted for Burr, log-normal, and log-logistic distributions for feature $F9$ (number of implicit withdrawals) of the Nimda dataset. The log-normal distribution is the best suitable fit for feature $F9$ (number of implicit withdrawals) based on the high p-value. The Burr and log-logistic distributions are rejected due to lower p-values.

- Ransomware datasets: The null hypothesis for Burr distribution is accepted for features $F3$ (number of announced NLRI prefixes), $F10$ (number of duplicate withdrawals), $F1$ (number of announcements), and $F34$ (number of IGP packets) of the WannaCrypt dataset. Features $F4$, $F36$, and $F9$ are rejected based on lower p-values. Burr distribution is accepted for feature $F9$ (number of implicit withdrawals) of the WestRock dataset. The Burr and log-logistic distributions are accepted for WestRock feature $F4$ (number of withdrawn NLRI prefixes). The log-logistic distribution is the best fit for WestRock feature $F9$ (number of implicit withdrawals) since it has a higher p-value.

Code Red and WannaCrypt datasets have common features accepted by Burr distribution indicating underlying similarities between the two datasets. Note that worm BGP datasets do not have common features for a given probability distribution. WannaCrypt is a cryptoworm [32] that propagates through a network using similar techniques as Code Red. It encrypts victim's files upon infection and then self-replicates and propagates through a network without user activation. The feature $F9$ (number of implicit withdrawals) is the number of newly advertised AS-paths for the already announced NLRI prefixes. It follows the Burr distribution in the case of Code Red and WestRock datasets. Feature F9 indicates the attack that may have been re-routed by the attackers through desired AS-paths. We observe that a number of BGP features follow heavy-tailed distributions. The Q-Q plots were also analyzed for important features of worms and ransomware datasets to validate the skewness of the BGP features. The Q-Q plots also confirmed that the features for which

Burr distribution was accepted using the K-S test are right-skewed. Sample Q-Q plots for feature $F1$ (number of announcements) are shown in Fig. 5.5. The majority of data points in $F1$ Q-Q plots for Code Red (top row (left)) and WannaCrypt datasets (middle row (right)) confirm that the feature follows Burr distribution. The remaining plots indicate that the Burr distribution is not a suitable fit for feature $F1$.

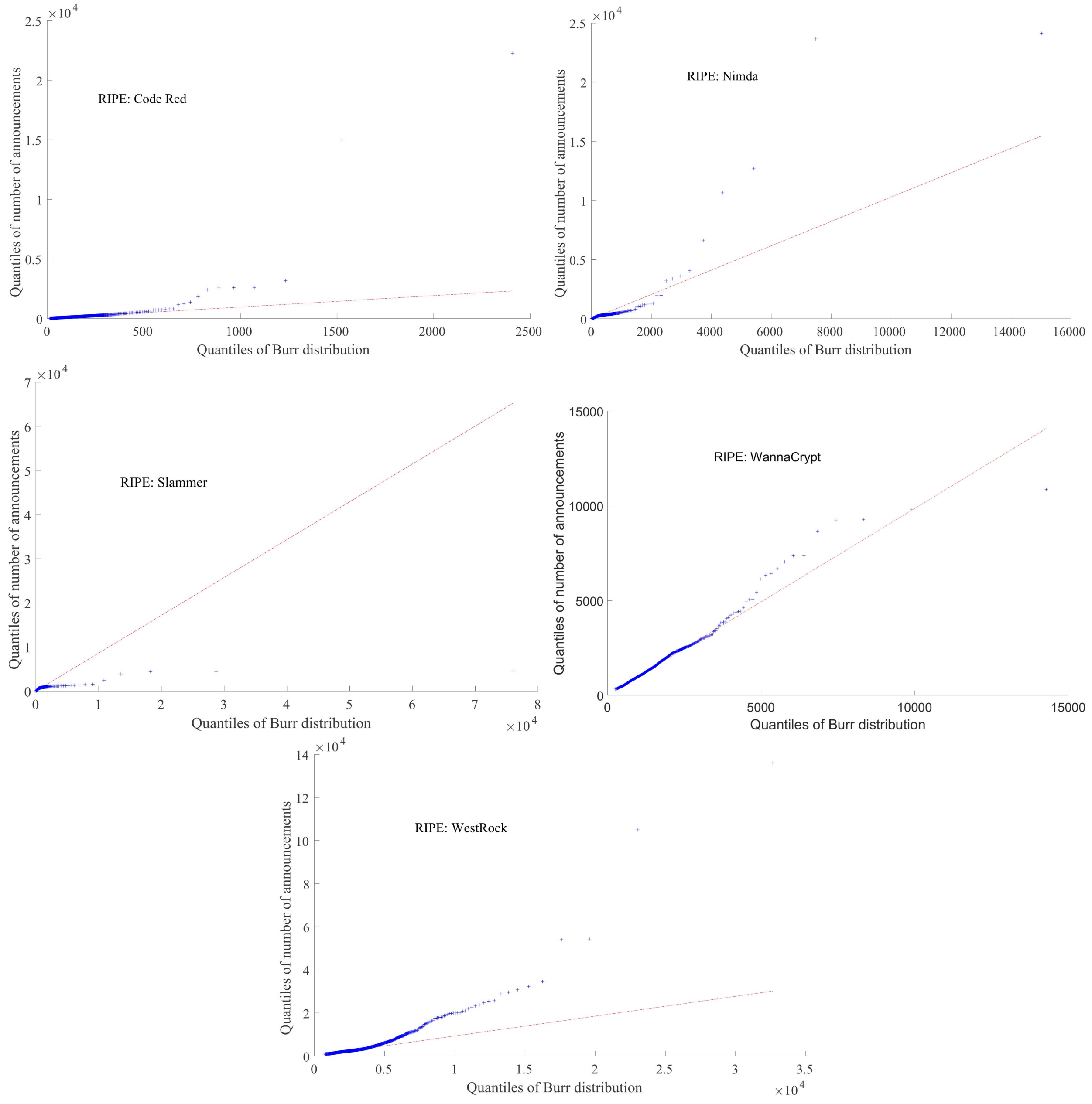


Figure 5.5: Burr distribution Q-Q plots of number of announcements (F1): Code Red, top row (left); Nimda top row (right); Slammer, middle row (left); WannaCrypt, middle row (right); and WestRock, bottom (row).

Table 5.9: K-S test results for Burr, log-normal, and log-logistic distribution using worm BGP RIPE datasets.

Feature	α **= 0.5**	α **= 0.1**	α **= 0.05**	α **= 0.01**
F34: Code Red			**Burr**	
h	1	0	0	0
p	0.120659	0.120659	0.120659	0.120659
k	0.01860	0.01860	0.01860	0.01860
c	NaN	0.019214	0.021325	0.025565
F1: Code Red				
h	1	0	0	0
p	0.246485	0.246485	0.246485	0.246485
k	0.016048	0.016048	0.016048	0.016048
c	NaN	0.019214	0.021325	0.025565
F3: Code Red				
h	1	0	0	0
p	0.292473	0.292473	0.292473	0.292473
k	0.015371	0.015371	0.015371	0.015371
c	NaN	0.019214	0.021325	0.025565
F9: Code Red				
h	1	0	0	0
p	0.101251	0.101251	0.101251	0.101251
k	0.019173	0.019173	0.019173	0.019173
c	NaN	0.019214	0.021325	0.025565
F37: Code Red				
h	1	1	1	0
p	0.023385	0.023385	0.023385	0.023385
k	0.023423	0.023423	0.023423	0.023423
c	NaN	0.019214	0.021325	0.025565
F9: Nimda			**Burr**	
h	1	1	0	0
p	0.056480	0.056480	0.056480	0.056480
k	0.019871	0.019871	0.019871	0.019871
c	NaN	0.018207	0.020208	0.024225
F9: Nimda			**Log-normal**	
h	1	0	0	0
p	0.142046	0.142046	0.142046	0.142046
k	0.017104	0.017104	0.017104	0.017104
c	NaN	0.018207	0.020208	0.024225
F9: Nimda			**Log-logistic**	
h	1	1	0	0
p	0.05679	0.05679	0.05679	0.05679
k	0.019855	0.019855	0.019855	0.019855
c	NaN	0.018207	0.020208	0.024225
F3: Slammer			**Burr**	
h	1	1	1	0
p	0.034104	0.034104	0.034104	0.034104
k	0.023285	0.023285	0.023285	0.023285
c	NaN	0.019968	0.022162	0.026569

Table 5.10: K-S test results for Burr distribution of ransomware attack BGP RIPE datasets.

Feature	$\alpha = \mathbf{0.5}$	$\alpha = \mathbf{0.1}$	$\alpha = \mathbf{0.05}$	$\alpha = \mathbf{0.01}$
F4: WannaCrypt			**Burr**	
h	1	1	0	0
p	0.054291	0.054291	0.054291	0.054291
k	0.016892	0.016892	0.016892	0.016892
c	NaN	0.015393	0.020479	0.017084
F10: WannaCrypt				
h	1	0	0	0
p	0.219681	0.219681	0.219681	0.219681
k	0.013209	0.013209	0.013209	0.013209
c	NaN	0.015393	0.020479	0.017084
F36: WannaCrypt				
h	1	1	1	0
p	0.010246	0.010246	0.010246	0.010246
k	0.020432	0.020432	0.020432	0.020432
c	NaN	0.015393	0.020479	0.017084
F9: WannaCrypt				
h	1	1	1	0
p	0.011156	0.011156	0.011156	0.011156
k	0.020266	0.020266	0.020266	0.020266
c	NaN	0.015393	0.020479	0.017084
F9: WestRock			**Burr**	
h	1	0	0	0
p	0.224965	0.224965	0.224965	0.224965
k	0.011020	0.011020	0.011020	0.011020
c	NaN	0.012911	0.017176	0.014329
F4: WestRock			**Burr**	
h	1	0	0	0
p	0.147570	0.147570	0.147570	0.147570
k	0.012041	0.012041	0.012041	0.012041
c	NaN	0.012911	0.017176	0.014329
F4: WestRock			**Log-logistic**	
h	1	0	0	0
p	0.284391	0.284391	0.284391	0.284391
k	0.010407	0.010407	0.010407	0.010407
c	NaN	0.012911	0.017176	0.014329

Table 5.11: Features and distributions: accepted null hypothesis.

Dataset	Distribution	Feature
Code Red	Burr	F34, F1, F3, F9, F37
Nimda	Burr/Log-normal /Log-logistic	F9
Slammer	Burr	F3
WannaCrypt	Burr	F4, F3, F10, F1, F34, F36, F9
WestRock	Burr	F9 , F4
	Log-logistic	F4

Table 5.12: Parameters of Burr, log-normal, and log-logistic distributions.

Dataset	**Feature**	α	**c**	**k**
Burr				
Code Red	F34	48.78573	4.98972	0.476546
	F1	56.93167	5.31064	0.452353
Nimda	F9	92.1486	1.80949	0.98291
Slammer	F3	57.7592	3.15328	0.283176
WannaCrypt	F4	114.162	4.75842	0.457034
	F3	1209.79	5.57019	0.341062
WestRock	F9	613.496	6.077945	0.72901
Log-normal		μ	σ	
Nimda	F9	4.545695	0.976093	
Log-logistic		μ	σ	
Nimda	F9	4.538061	0.556037	
WestRock	F9	6.502269	0.186274	

Chapter 6

Performance of Classification Models

We perform experiments using Python on Google Colab notebooks. The use of publicly available platforms and libraries enhances the reproducibility of research results. Machine learning libraries (Scikit-learn, NumPy, Pandas, Matplotlib) and the Keras framework are applied to perform data analysis and classification. BGP data used in this thesis are publicly available [43]. These datasets are generated by extracting BGP update messages available from RIPE collection sites. The training and test datasets contain 60% and 40% of anomalous data points, respectively. We employ SVM, LSTM, and GBDT machine learning algorithms to perform binary classification of regular and anomalous data points. We also apply learning rate scheduler and attention mechanism machine learning approaches to LSTM models to enhance their classification performance. We then examine the effect of dimension reduction and feature selection approaches on the performance of GBDT models. The GBDT algorithms XGBoost, LightGBM, and CatBoost are employed for classification due to their efficient training time. The best performing model hyperparameters are obtained by performing 10-fold time series split cross-validation using training datasets. The cross-validation experiments are performed based on accuracy or F-Score. Performance of the generated models is evaluated based on training time, accuracy, F-Score, precision, recall, and confusion matrix.

6.1 Performance Enhancement Using Machine Learning Approaches

Machine learning approaches such as SVM, learning rate scheduling and attention mechanism with LSTM models are employed to enhance the classification performance. We employ SVM to identify the separation between data belonging to regular and anomaly clusters. Learning rate scheduling and attention mechanisms are applied to enhance the performance of the LSTM model to classify anomalies.

6.1.1 Support Vector Machine

We evaluate F-Score of SVM models based on BGP datasets as shown in Table 6.1. Models are generated using various kernel functions and coefficient values. SVM models using linear kernel offer better F-Score using WestRock dataset. This may indicate that the WestRock data is linearly separable. The SVM model generated using linear kernel obtained 0 F-Score using Code Red dataset. It may represent that the data are not linearly separable. Although the models generated using high regularization coefficient values obtain high F-Scores. However, these models suffer from overfitting due to hard margin kernels and should not be considered for classification.

Table 6.1: BGP datasets: F-Scores of SVM models generated using various kernel functions and regularization coefficients.

Kernel/Regularization	**Code Red**	**Nimda**	**Slammer**	**WannaCrypt**	**WestRock**
RBF/C = 1	38.96	30.46	66.51	59.08	71.35
RBF/C = 100	73.64	39.44	57.90	38.74	66.76
RBF/C = 1000	69.45	38.16	57.28	40.56	62.82
Linear/C = 1	0.0	25.55	53.14	56.74	73.54
Linear/C = 100	0.0	26.26	52.83	56.55	73.54
Linear/C = 1000	0.0	26.08	52.89	56.82	73.54
Poly/C = 1	31.37	20.64	62.22	51.97	71.12
Poly/C = 100	69.85	36.57	60.40	50.68	68.22
Poly/C = 1000	70.77	35.05	56.90	46.14	65.20
Sigmoid/C = 1	25.99	23.77	49.77	52.16	66.87
Sigmoid/C = 100	20.87	23.69	43.86	50.88	65.47
Sigmoid/C = 1000	19.54	23.54	43.72	50.80	65.43

6.1.2 Long Short Term Memory

LSTM model is known for its effective performance using time series data. They learn long-term dependencies effectively in temporal data. Experiments are performed [63] to evaluate LSTM and GRU models using BGP datasets. The best F-Score to classify anomalies in WannaCrypt BGP RIPE data was obtained using an LSTM model while the best accuracy was achieved using the GRU model with Route Views data [63]. Therefore, we apply learning rate scheduling and attention mechanism to enhance the performance of LSTM model to classify anomalies using WannaCrypt BGP RIPE data.

6.1.2.1 Learning Rate Scheduling

We perform experiments to enhance the performance of the best performing LSTM model [63] generated using ransomware datasets. Cosine cyclic learning rate scheduling [170] is applied to a simplified architecture of the LSTM model. The modified hyperparameters of the LSTM model are listed in Table 6.2. The performance of the LSTM model using WannaCrypt dataset increased from 65.48% and 63.22% [63] to 68.06% and 66.27% for accuracy and F-Score, respectively. The results represent that the performance of machine learning models is enhanced by using cyclic rather than constant learning rates. The use of a warm start to learn weights using cyclic learning enhances the model performance.

Table 6.2: LSTM models: Best hyperparameters before and after applying cyclic learning rate scheduler.

	Best value	
Parameter	**Before**	**After**
Length of sequence	100	100
No. of epochs	30	30
No. of hidden nodes	$FC_1 = 64$	$FC_1 =64$
	$FC_2 = 32$	$FC_2 = 2$
	$FC_3 = 16$	$FC_3 = 16$
Dropout rate	0.4	0.4
Learning rate	0.01	0.01

6.1.2.2 Attention Mechanism

We analyze performance of LSTM models by applying attention mechanism with LSTM layer using the ransomware attack BGP datasets. An attention class is implemented in Keras by inheriting the Layer class [171]. Models are generated using: a sequential input layer (nodes = 37), an LSTM layer (nodes = 37), an attention layer, and an output dense layer (nodes = 2). Binary cross-entropy and root mean square propagation (RMSprop) are selected as loss function and optimizer, respectively. The random seeds for NumPy and TensorFlow libraries are set to 0 and 1. The number of nodes for the input and LSTM layer matches the number of features in the dataset while the two nodes in the output layer are used for binary classification. The input is transformed to three-dimensional because the LSTM layer in Keras accepts a three-dimensional input. The output binary labels (0 or 1) are transformed into two-dimensional to input it to the dense output layer. The output labels are transformed into two-dimensions because we are employing a binary cross entropy loss function. The best hyperparameters and the performance results are listed in Table 6.3 and 6.4, respectively. The model with attention mechanism using ransomware attack datasets did not improve the F-Score of previously reported results [1].

Table 6.3: Ransomware attack datasets: best performing hyperparameters and F-Score.

Dataset	**Batch size**	**Epochs**	**Learning rate**
WannaCrypt	100	50	0.001
WestRock	100	10	0.1

Table 6.4: Ransomware attack datasets: performance results using attention mechanism.

Dataset	**Training Time**	**Accuracy**	**F-Score**	**Precision**	**Sensitivity**
WannaCrypt	23.066	60.62	60.77	60.58	60.96
WestRock	13.053	76.92	76.92	76.92	76.92

6.2 GBDT Models

Experiments are performed to evaluate the effect of data normalization using z-score on the classification performance of GBDT models. The best performing hyperparameters obtained based on accuracy or F-Score are listed in Table 6.5 while the performance results are shown in Table. 6.6. High accuracy or F-Score are obtained for both datasets by models generated using unnormalized data. The LightGBM model generated using normalized WestRock dataset is an underfitted model that could not detect any data points belonging to the regular class and is unsuitable for anomaly detection. The decision tree models use absolute data values to create tree branches and do not require features to be scaled. Therefore, the unnormalized data are used to perform experiments using tree-based algorithms.

Table 6.5: Best hyperparameters: number of estimators and learning rate (LR) based on accuracy using normalized data using z-score and unnormalized data.

Dataset	**Algorithm**	**Estimators**	**LR**	**Estimators**	**LR**
		Accuracy		**F-Score**	
Normalized data					
	XGBoost	260	0.10	260	0.10
WannaCrypt	LightGBM	150	0.10	150	0.10
	CatBoost	300	0.10	300	0.10
	XGBoost	10	0.01	10	0.01
WestRock	LightGBM	20	0.01	60	0.10
	CatBoost	20	0.01	220	0.05
Unnormalized data					
	XGBoost	270	0.10	270	0.10
WannaCrypt	LightGBM	130	0.10	130	0.10
	CatBoost	280	0.10	280	0.10
	XGBoost	330	0.10	330	0.10
WestRock	LightGBM	50	0.10	50	0.10
	CatBoost	170	0.05	170	0.05

We now discuss the classification results using GBDT models generated using various dimension reduction (PCA) and feature selection techniques such as correlation, random forests, and extra-trees.

Table 6.6: Performance of models generated using hyperparameters based on accuracy or F-Score using normalized (z-score) and unnormalized BGP ransomware attack data.

Dataset	**Algorithm**	**Training time (s)**	**Accuracy (%)**	**F-Score (%)**
Normalized data				
	XGBoost	3.0562	55.00	41.49
WannaCrypt	LightGBM	0.7293	55.54	43.07
	CatBoost	2.2449	54.94	40.70
	XGBoost	0.4972	55.41	70.88
WestRock	LightGBM	0.5477	53.58	68.78
	CatBoost	2.7002	54.67	70.16
Unnormalized data				
	XGBoost	1.9838	60.48	61.43
WannaCrypt	LightGBM	0.3276	59.52	60.81
	CatBoost	1.7270	60.02	61.30
	XGBoost	3.3058	57.79	71.48
WestRock	LightGBM	0.1643	57.35	71.07
	CatBoost	2.8835	56.31	70.32

6.2.1 Principal Component Analysis

PCA is separately applied to training and test datasets to transform the data to contain 10 PCA components. The transformed data are used as input for the GBDT models. The training datasets are employed to perform 10-fold time series split cross-validation to obtain the best hyperparameters of GBDT models. Datasets used in this thesis are unbalanced and therefore the cross-validation experiments using PCA transformed data are based on F-Score. The BGPGuard [166] tool is used to perform time series cross-validation. Training time, accuracy, F-Score, precision, and sensitivity of the GBDT models are listed in Table 6.7. The best hyperparameters and confusion matrix results are listed in Table 6.8. The code used to perform experiments is listed in Appendix A. Dimension reduction using PCA improved the classification performance of GBDT models using the WestRock BGP dataset. Only a few true positives are classified using PCA transformed data resulting in a low F-Score for all BGP datasets except for WestRock. The LightGBM and CatBoost models generated using the Code Red dataset did not learn the properties of the training dataset and hence obtain 0 F-Score values. The transformed datasets have lower dimension leading to lower training time. However, they do not capture all relevant information of the original datasets and result in poor classification performance. Dimension reduction performed using autoencoders achieved better performance using the LightGBM model (WestRock data) [166].

Table 6.7: GBDT models: performance based on the F-Score using datasets with 10 PCA components.

Dataset	Algorithm	Training time (s)	Accuracy (%)	F-Score (%)	Precision (%)	Sensitivity (%)
	XGBoost	0.4635	92.15	16.78	43.10	10.42
Code Red	LightGBM	0.0452	92.41	0	0	0
	CatBoost	0.0600	92.41	0	0	0
	XGBoost	0.0402	82.21	19.23	20.19	18.35
Nimda	LightGBM	0.2507	85.11	17.96	24.63	14.14
	CatBoost	2.6072	84.47	20.25	24.85	17.09
	XGBoost	0.1941	91.01	59.77	53.35	67.94
Slammer	LightGBM	0.5597	91.65	43.00	65.27	32.06
	CatBoost	0.9329	91.50	52.73	58.16	48.24
	XGBoost	1.4300	54.31	41.13	48.68	35.60
WannaCrypt	LightGBM	0.5153	54.20	39.97	48.45	34.02
	CatBoost	1.5911	53.56	40.91	47.62	35.85
	XGBoost	0.1462	56.42	71.81	57.55	95.48
WestRock	LightGBM	0.4394	47.57	60.73	53.79	69.73
	CatBoost	1.8117	50.17	63.35	55.34	74.08

Table 6.8: GBDT models: best hyperparameters based on the F-Score and confusion matrix using datasets with 10 PCA components.

Dataset	Algorithm	Estimators	Learning rate	TP	FP	TN	FN
	XGBoost	10	0.01	25	33	2,887	215
Code Red	LightGBM	10	0.01	0	0	2,920	240
	CatBoost	10	0.01	0	0	2,920	240
	XGBoost	10	0.01	87	344	3,291	387
Nimda	LightGBM	110	0.05	67	205	3,430	407
	CatBoost	210	0.10	81	245	3,390	393
	XGBoost	60	0.10	231	202	2,918	109
Slammer	LightGBM	270	0.05	109	58	3,062	231
	CatBoost	200	0.10	164	118	3,002	176
	XGBoost	300	0.10	833	878	2,002	1,507
WannaCrypt	LightGBM	260	0.10	796	847	2,033	1,544
	CatBoost	300	0.10	839	923	1,957	1,501
	XGBoost	10	0.01	3,819	2,817	63	181
WestRock	LightGBM	200	0.10	2,789	2,396	484	1,211
	CatBoost	300	0.10	2,963	2,391	489	1,037

6.2.2 Features Selection

Important features are identified using various unsupervised and supervised feature selection techniques. The effect of the selected features is measured by evaluating performance of GBDT models generated using worms and ransomware attack datasets.

6.2.2.1 Pearson Correlation

The best performing hyperparameters based on F-Score listed in Table 6.9 are obtained using the 10-fold time series cross-validation. Performance of models generated using the selected features by eliminating highly correlated features is shown in Table 6.10. High F-Scores are obtained for the Code Red and WestRock datasets using XGBoost and LightGBM models, respectively. The F-Score for the models generated using WestRock dataset are similar for XGBoost and LightGBM models while the training time is shorter for the LightGBM model. The short training time is attributed to the histogram-based GOSS technique used in LightGBM. The sensitivity of GBDT models based on the WestRock dataset is also high indicating their efficient ability to detect true positives at a high rate. The GBDT models generated using the ransomware attack BGP datasets have low accuracy as they result in a large number of false positives.

Table 6.9: GBDT models: hyperparameters based on F-score after eliminating highly correlated features.

Dataset	**Algorithm**	**Estimators**	**Learning rate**
	XGBoost	10	0.01
Code Red	LightGBM	10	0.01
	CatBoost	10	0.01
	XGBoost	150	0.10
Nimda	LightGBM	190	0.10
	CatBoost	250	0.05
	XGBoost	150	0.10
Slammer	LightGBM	190	0.10
	CatBoost	250	0.05
	XGBoost	240	0.10
WannaCrypt	LightGBM	270	0.10
	CatBoost	300	0.10
	XGBoost	170	0.10
WestRock	LightGBM	280	0.05
	CatBoost	250	0.10

6.2.2.2 Spearman Correlation

The best performing hyperparameters based on F-Score listed in Table 6.12 are obtained using the 10-fold time series cross-validation. Performance and confusion matrix results are listed in Tables 6.13 and 6.14, respectively.

High F-Scores are obtained for the Code Red and WestRock datasets using CatBoost and XGBoost models, respectively. The LightGBM model using Code Red has a 0 F-Score because the model did not learn data properties due to unbalanced data. Performance results obtained using datasets without Spearman and Pearson correlated features are similar.

Table 6.10: GBDT model performance based on F-score with eliminated high Pearson correlated features.

Dataset	**Algorithm**	**Training time**	**Accuracy**	**F-Score**	**Precision**	**Senstivity**
		(s)	(%)	(%)	(%)	(%)
	XGBoost	0.3638	96.99	80.65	78.88	82.50
Code Red	LightGBM	0.1342	92.41	0	0	0
	CatBoost	0.2255	95.60	62.53	88.55	48.33
	XGBoost	0.7468	81.55	41.87	32.89	57.59
Nimda	LightGBM	0.9619	81.26	40.59	32.00	55.49
	CatBoost	1.2594	82.87	44.13	35.37	58.65
	XGBoost	0.4429	93.96	57.95	91.72	42.35
Slammer	LightGBM	0.9248	94.10	59.36	91.98	43.82
	CatBoost	1.5159	94.05	58.80	91.88	43.24
	XGBoost	3.5598	58.66	60.21	52.95	69.79
WannaCrypt	LightGBM	3.4422	59.25	60.46	53.50	69.49
	CatBoost	1.2406	59.98	61.11	54.14	70.13
	XGBoost	5.6716	57.60	70.76	59.06	88.22
WestRock	LightGBM	4.3026	57.81	70.94	59.16	88.60
	CatBoost	2.2074	55.89	69.77	57.99	87.55

Table 6.11: Confusion matrix of GBDT models generated with eliminated high Pearson correlated features from BGP datasets.

Dataset	**Algorithm**	**TP**	**FP**	**TN**	**FN**
	XGBoost	198	53	2,867	42
Code Red	LightGBM	0	0	2,920	240
	CatBoost	116	15	2,905	124
	XGBoost	273	557	3,078	201
Nimda	LightGBM	263	559	3,076	211
	CatBoost	278	508	3,127	196
	XGBoost	144	13	3,107	196
Slammer	LightGBM	149	13	3,107	191
	CatBoost	147	13	3,107	193
	XGBoost	1,633	1,451	1,429	707
WannaCrypt	LightGBM	1,626	1,413	1,467	714
	CatBoost	1,641	1,390	1,490	699
	XGBoost	3,529	2,446	434	471
WestRock	LightGBM	3,544	2,447	433	456
	CatBoost	3,502	2,537	343	498

6.2.2.3 Random Forests

The GBDT algorithms are employed to generate classification models. We use the entire dataset with 37 features and its subsets of 8 and 16 features. The best performing hyperparameters listed in Table 6.15 are obtained by performing 10-fold time series split cross-validation. The results are obtained based on accuracy or F-Score. The model hyperparameters based on accuracy or F-Score are identical for the three worm datasets. Performance results are listed in Tables 6.16, 6.17, and 6.18. The model generated using

Table 6.12: Hyperparameters for models generated using datasets with eliminated high Spearman correlated features.

Dataset	Algorithm	Estimators	Learning rate
	XGBoost	10	0.01
Code Red	LightGBM	10	0.01
	CatBoost	10	0.01
	XGBoost	200	0.10
Nimda	LightGBM	170	0.05
	CatBoost	140	0.10
	XGBoost	180	0.05
Slammer	LightGBM	70	0.05
	CatBoost	130	0.10
	XGBoost	250	0.10
WannaCrypt	LightGBM	250	0.05
	CatBoost	290	0.10
	XGBoost	150	0.05
WestRock	LightGBM	90	0.10
	CatBoost	300	0.05

Table 6.13: GBDT models: performance using datasets with eliminated Spearman correlated features.

Dataset	Algorithm	Training time (s)	Accuracy (%)	F-Score (%)	Precision (%)	Senstivity (%)
	XGBoost	0.4193	96.33	76.52	74.41	78.75
Code Red	LightGBM	0.1516	92.41	0	0	0
	CatBoost	0.2432	96.71	76.58	83.33	70.83
	XGBoost	2.5078	81.67	41.13	32.67	55.49
Nimda	LightGBM	1.2397	81.07	40.25	31.64	55.27
	CatBoost	1.3672	80.99	39.78	31.35	54.43
	XGBoost	2.0316	93.87	56.56	93.24	40.59
Slammer	LightGBM	0.3239	93.76	55.19	93.66	39.12
	CatBoost	0.6340	94.13	58.82	94.77	42.65
	XGBoost	2.4791	59.41	60.56	53.64	69.53
WannaCrypt	LightGBM	1.1649	59.69	60.80	53.90	69.74
	CatBoost	1.8676	59.71	61.08	53.87	70.51
	XGBoost	3.0025	58.60	72.04	59.31	91.72
WestRock	LightGBM	1.4164	57.14	70.49	58.77	88.05
	CatBoost	4.4326	56.37	70.57	58.05	89.98

LightGBM for the Code Red dataset has 0 F-Score because the model did not learn the data properties during cross-validation due to the unbalanced dataset. The confusion matrix represents that these models did not detect any anomalies (true positives and false positives). The highest F-Score 81.30 is obtained for the model generated using the CatBoost model for Code Red dataset with 37 features. CatBoost models generated using WestRock dataset exhibit higher performance than WannaCrypt GBDT models.

Table 6.14: GBDT models: confusion matrix using datasets with eliminated high Spearman correlated features.

Dataset	Algorithm	TP	FP	TN	FN
	XGBoost	189	65	2,855	51
Code Red	LightGBM	0	0	2,920	240
	CatBoost	170	34	2,886	70
	XGBoost	263	542	3,093	211
Nimda	LightGBM	262	566	3,069	212
	CatBoost	258	565	3,070	216
	XGBoost	138	10	3,110	202
Slammer	LightGBM	133	9	3,111	207
	CatBoost	145	8	3,112	195
	XGBoost	1,627	1,406	1,474	713
WannaCrypt	LightGBM	1,632	1,396	1,484	708
	CatBoost	1,650	1,413	1,467	690
	XGBoost	3,669	2,517	363	331
WestRock	LightGBM	3,522	2,471	409	478
	CatBoost	3,599	2,601	279	401

Table 6.15: GBDT models: best hyperparameters based on accuracy or F-Score. Important features are selected using random forest.

		Accuracy		F-Score	
Dataset	**Algorithm**	**Estimators**	**LR**	**Estimators**	**LR**
Code Red	XGBoost	10	0.01	10	0.01
	LightGBM	10	0.01	10	0.01
	CatBoost	10	0.01	10	0.01
Nimda	XGBoost	260	0.01	260	0.01
	LightGBM	280	0.01	280	0.01
	CatBoost	240	0.10	240	0.10
Slammer	XGBoost	140	0.05	140	0.05
	LightGBM	170	0.01	170	0.01
	CatBoost	60	0.10	60	0.10
WannaCrypt	XGBoost	270	0.10	270	0.10
	LightGBM	100	0.10	130	0.10
	CatBoost	280	0.10	280	0.10
WestRock	XGBoost	270	0.05	260	0.10
	LightGBM	270	0.10	270	0.10
	CatBoost	80	0.10	170	0.05

6.2.2.4 Extra-Trees

Performance of models is evaluated using 37 features and its subsets 16 and 8. Experiments are performed to identify the best parameters using grid-search and 10-fold time series split cross validation based on accuracy or F-Score. Grid search is an exhaustive ad hoc approach to evaluate combination of parameters and obtain the best performance re-

Table 6.16: GBDT models: best performance based on F-Score. Important features are selected using random forest.

Dataset	Features	Training time (s)			Accuracy (%)			F-Score (%)		
		XGBoost	LightGBM	CatBoost	XGBoost	LightGBM	CatBoost	XGBoost	LightGBM	CatBoost
	37	0.0514	0.1716	0.1431	96.84	92.41	97.28	78.54	0	81.30
Code Red	16	0.02646	0.0255	0.1100	96.84	92.41	96.87	78.54	0	78.05
	8	0.0205	0.0175	0.0495	96.84	92.41	96.99	79.84	0	79.21
	37	1.1529	0.8916	1.5392	80.58	81.67	82.14	39.08	40.94	42.11
Nimda	16	0.5964	0.5486	1.5917	80.58	81.50	81.70	39.08	40.72	41.07
	8	0.4222	0.5022	0.8698	80.24	80.99	80.65	39.58	39.32	40.09
	37	0.4438	0.3631	0.4054	93.76	93.06	94.08	55.37	46.67	58.59
Slammer	16	0.8141	0.2647	0.2046	93.82	93.03	93.29	55.79	46.33	50.00
	8	0.2762	0.3982	0.1521	93.67	92.89	93.01	53.89	44.09	46.70
	37	2.7926	1.8305	3.7370	60.48	59.52	60.02	61.43	60.81	61.30
WannaCrypt	16	2.6863	0.4210	3.1641	60.13	59.66	59.90	61.05	60.81	61.43
	8	1.1894	0.1934	1.1166	60.31	60.23	60.15	61.21	61.34	61.82
	37	3.3410	1.0118	2.4799	50.28	50.04	51.31	66.19	65.65	67.35
WestRock	16	1.1452	0.8360	0.7299	50.80	50.13	51.74	66.65	65.60	67.79
	8	0.8088	0.4180	1.9548	50.89	50.76	51.83	66.90	66.42	67.88

Table 6.17: GBDT models: precision and recall based on accuracy or F-Score. Important features are selected using the random forest.

	Features	Precision (%)			Recall (%)		
		XGBoost	LightGBM	CatBoost	XGBoost	LightGBM	CatBoost
	37	80.97	0	85.00	76.25	0	77.92
Code Red	16	80.97	0	83.41	76.25	0	73.33
	8	77.34	0	83.41	82.50	0	75.42
	37	30.62	32.58	33.63	54.01	55.06	56.33
Nimda	16	30.62	32.30	32.67	54.01	55.06	55.27
	8	30.57	31.12	31.18	56.12	53.38	56.12
	37	93.06	95.45	93.55	39.41	30.88	42.65
Slammer	16	93.75	95.41	93.55	39.71	30.59	34.12
	8	94.81	97.00	92.98	37.65	28.53	31.18
	37	54.60	53.72	54.14	70.21	70.04	70.64
WannaCrypt	16	54.31	53.86	54.00	69.70	69.83	71.24
	8	54.46	54.36	54.18	69.87	70.38	71.97
	37	54.73	54.69	55.19	83.72	82.12	86.38
WestRock	16	55.00	54.76	55.39	84.58	81.80	87.35
	8	55.00	55.02	55.43	85.35	83.78	87.55

sults. Its complexity depends on the selected grid granularity. Even though it is faster than cross-validation, grid search is an ad hoc approach, its results are not generalized, and its performance greatly depends on the employed training datasets.

The analysis of experiments illustrate that using F-Score is a suitable performance parameter due to unbalanced datasets. 10-fold time series cross-validation is a more reliable approach because it uses subsets of a given dataset to train the model over multiple iterations and create generalized models. Hence, the hyperparameter values of models obtained by performing time series cross-validation are more reliable and generalized. The best performing hyperparameters obtained based on accuracy or F-Score are listed in Table 6.19. The values obtained for models based on accuracy or F-Score are identical. The performance results are listed in Table 6.20.

Table 6.18: GBDT Models: Confusion matrix based on accuracy or F-Score. Important features are selected using the random forest.

Dataset	**Features**	**XGBoost**				**LightGBM**				**CatBoost**			
		TP	**FP**	**TN**	**FN**	**TP**	**FP**	**TN**	**FN**	**TP**	**FP**	**TN**	**FN**
	37	183	43	2,877	57	0	0	2,920	240	187	33	2,887	53
Code Red	16	183	43	2,877	57	0	0	2,920	240	176	35	2,885	64
	8	198	58	2,862	42	0	0	2,920	240	181	36	2,884	59
	37	256	580	3,055	218	261	540	3,095	213	267	527	3,108	207
Nimda	16	256	580	3,055	218	261	547	3,088	213	262	540	3,095	212
	8	266	604	3,031	208	253	560	3,075	221	266	587	3,048	208
	37	134	10	3,110	206	105	5	3,115	235	145	10	3,110	195
Slammer	16	135	9	3,111	205	104	5	3,115	236	116	8	3,112	224
	8	128	7	3,113	212	97	3	3,117	243	106	8	3,112	234
	37	1,643	1,366	1,514	697	1,639	1,412	1,468	701	1,653	1,400	1,480	687
WannaCrypt	16	1,631	1,372	1,508	709	1,634	1,400	1,480	706	1,667	1,420	1,460	673
	8	1,635	1,367	1,513	705	1,647	1,383	1,497	693	1,684	1,424	1,456	656
	37	3,349	2,770	110	651	3,285	2,722	158	715	3,455	2,805	75	545
WestRock	16	3,383	2,768	112	617	3,272	2,703	177	728	3,494	2,814	66	506
	8	3,414	2,793	87	586	3,351	2,739	141	649	3,502	2,816	64	498

Table 6.19: GBDT models: best hyperparameters based on accuracy and F-Score. Important features are selected using extra-trees.

		Accuracy		**F-Score**	
Dataset	**Algorithm**	**Estimators**	**LR**	**Estimators**	**LR**
Code Red	XGBoost	10	0.01	10	0.01
	LightGBM	10	0.01	10	0.01
	CatBoost	10	0.01	10	0.01
Nimda	XGBoost	260	0.01	260	0.01
	LightGBM	280	0.01	280	0.01
	CatBoost	240	0.10	240	0.10
Slammer	XGBoost	140	0.05	140	0.05
	LightGBM	170	0.01	170	0.01
	CatBoost	60	0.10	60	0.10
WannaCrypt	XGBoost	270	0.10	270	0.10
	LightGBM	130	0.10	130	0.10
	CatBoost	280	0.10	280	0.10
WestRock	XGBoost	330	0.10	330	0.10
	LightGBM	50	0.10	50	0.10
	CatBoost	170	0.05	170	0.05

Table 6.20: GBDT models: best performance results based on F-Score. Important features are selected using extra-trees.

Dataset	**Features**	**Training time (s)**			**Accuracy (%)**			**F-Score (%)**		
		XGBoost	**LightGBM**	**CatBoost**	**XGBoost**	**LightGBM**	**CatBoost**	**XGBoost**	**LightGBM**	**CatBoost**
Code Red	37	0.0470	0.0425	0.2110	96.84	92.41	97.28	78.54	0.00	81.30
	16	0.0262	0.0253	0.0525	96.84	92.41	97.22	78.54	0.00	81.03
	8	0.0231	0.0272	0.0468	96.90	92.41	96.58	80.32	0.00	75.78
Nimda	37	1.0583	0.4607	2.4636	80.58	81.67	82.14	39.08	40.94	42.11
	16	0.6187	0.4359	2.3066	80.58	81.46	81.97	39.08	40.56	41.97
	8	0.4749	0.3122	0.7893	80.24	80.99	80.65	39.58	39.32	40.09
Slammer	37	0.4645	0.3848	0.2644	93.76	93.06	94.08	55.37	46.67	58.58
	16	0.2710	0.2114	0.1824	93.55	92.95	93.15	53.05	45.05	47.91
	8	0.1968	0.1599	0.1652	93.41	92.75	93.09	51.07	42.3	47.01
WannaCrypt	37	1.9838	0.3276	1.7270	60.48	59.52	60.02	61.43	60.81	61.30
	16	2.3483	0.3074	1.3041	60.13	59.66	59.90	61.05	60.81	61.43
	8	0.7880	0.1973	1.0541	61.03	60.54	60.13	61.95	61.47	61.75
WestRock	37	3.3058	0.1643	2.8835	57.79	57.35	56.31	71.48	71.07	70.32
	16	1.7448	0.1480	1.1840	57.73	57.53	56.58	71.33	71.26	70.90
	8	0.9798	0.0907	1.4420	59.56	58.02	56.38	72.96	71.67	71.07

Chapter 7

Conclusion

Detecting ransomware attacks is a challenging task. The Internet is a vast interconnected network with billions of connected devices. Users heavily rely on the Internet to perform their daily activities. Network traffic consists of activities performed by many malicious traffic actors targeting various vulnerable systems. Anomaly detection is a crucial task in cybersecurity due to the constant increase in network attacks and intrusions. It is important that intrusion detection systems employ machine learning models that have low training time to be scalable and deployed in real-time environments. Machine learning intrusion detection techniques offer effective solutions to identify and detect cyberattacks. They handle a large amount of data effectively due to their computational abilities compared to traditional intrusion detection systems.

In this thesis, we applied various dimension reduction and feature selection techniques to identify important features and enhance the performance of the classifier models generated using BGP RIPE datasets of worms and ransomware attacks. Statistical approaches such as PCA, Pearson correlation, and Spearman correlation were applied to reduce the data dimension and eliminate redundancies. The data dimension was reduced using PCA to transform the data to contain 10 PCA components. Correlated features were eliminated from the datasets because they do not contain new information relevant to building generalized models. Supervised machine learning tree based algorithms such as random forest and extra-trees were applied to identify the important features based on importance criteria. Extra-trees models offered the best performance for selecting important features. We also estimated the data distributions of the 10 important features selected using extra-trees by performing K-S test and measure the influence of the data distributions on the performance of the machine learning models. The K-S test results indicated that heavy-tailed distributions are a suitable fit for a number of BGP features. The Burr distribution was accepted for common features in Code Red and WannaCrypt datasets highlighting their underlying similarities.

We performed binary classification using BGP data collected for various network anomalies: worms and ransomware attacks collected from RIPE repositories. Machine learning algorithms such as SVM, LSTM, and GBDT were applied to generate classifier models.

SVM models generated using linear kernels obtained high F-Scores using WestRock dataset indicating the data are linearly separable. Dynamic learning rate such as cosine annealing was applied to enhance the performance of LSTM model. They employ warm restarts to enhance the learning capabilities of models. Accuracy and F-Score of the best performing LSTM model was enhanced using the dynamic learning rate scheduler. Therefore, they may be applied rather than constant learning rates. Attention mechanism was also applied to enhance the model performance to classify ransomware attack datasets. However, it did not enhance the classification performance of the best performing LSTM models. The GBDT models generated using PCA transformed data offered a high F-Score using WestRock dataset. Experimental results indicated that GBDT models have a short training time that is desired for real-time IDSs. LSTM models remain the best performing models to classify ransomware attacks in BGP datasets. Identifying anomalies based on a high F-Score in the case of the WannaCrypt ransomware attack remains a challenging task.

In the future, transformers and generative adversarial networks may be applied to detect anomalies in networks. Transformers are deep learning models that employ attention mechanism and are well-known for their effective performance in natural language processing models. Reinforcement learning may also be applied to classify anomalies. Conditional generative adversarial networks (cGANs) are popular for generating conditional data and they may also offer effective performance to classify anomalies.

Bibliography

[1] Z. Li, “Machine learning for classifying anomalies and intrusions in communication networks,” Ph.D. dissertation, School of Engineering Science., Simon Fraser Univ., Burnaby, BC, Canada, 2022.

[2] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, 8th ed. New Jersey, NY, USA: Pearson, 2021.

[3] R. Vinayakumar, K. P. Soman, P. Poornachandran, and S. Akarsh, “Application of deep learning architectures for cyber security,” in *Cybersecurity and Secure Information Systems: Challenges and Solutions in Smart Environments*, A. E. Hassanien and M. Elhoseny, Eds. Cham: Springer International Publishing, 2019, pp. 125–160.

[4] (2023, Apr.) Command and control explained. [Online]. Available: https://www.paloaltonetworks.com/cyberpedia/command-and-control-explained

[5] S. Donaldson, S. Siegel, C. K. Williams, and A. Aslam, *"Enterprise Cybersecurity: How to Build a Successful Cyberdefense Program Against Advanced Threats"*. Apress, 2015.

[6] S. Talukder, “Tools and techniques for malware detection and analysis,” *arXiv preprint arXiv:2002.06819*, 2020.

[7] F. T. Ngo, A. Agarwal, R. Govindu, and C. MacDonald, *Malicious Software Threats*. Cham: Springer International Publishing, 2020, pp. 793–813.

[8] F. Ullah, M. Edwards, R. Ramdhany, R. Chitchyan, M. A. Babar, and A. Rashid, “Data exfiltration: A review of external attack vectors and countermeasures,” *J. Netw. Comput. Appl.*, vol. 101, pp. 18–54, 2018.

[9] E. Conrad, S. Misenar, and J. Feldman, “Domain 7: security operations,” in *Eleventh Hour CISSP*, 3rd ed., E. Conrad, S. Misenar, and J. Feldman, Eds. Syngress, 2017, pp. 145–183.

[10] T. B. Azad, “Locking down your xenapp server,” in *Securing Citrix Presentation Server in the Enterprise*, T. B. Azad, Ed. Burlington: Syngress, 2008, pp. 487–555.

[11] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: a survey,” *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, July 2009.

[12] (2023, Apr.) RFC 1771 - A Border Gateway Protocol 4 (BGP-4). [Online]. Available: https://datatracker.ietf.org/doc/html/rfc1771.

[13] (2023, Apr.) RFC 827 - Exterior Gateway Protocol (EGP). [Online]. Available: https://www.rfc-editor.org/rfc/rfc827.

[14] (2023, Apr.) The cisco learning network. [Online]. Available: https://www.cloudflare.com/learning/network-layer/what-is-an-autonomous-system/.

[15] (2023, Apr.) BGP network layer reachability information (nlri). [Online]. Available: https://bit.ly/2Lwod2c.

[16] Q. Ding, Z. Li, S. Haeri, and Lj. Trajković, "Application of machine learning techniques to detecting anomalies in communication networks: datasets and feature selection algorithms," in *Cyber Threat Intelligence*, A. Dehghantanha, M. Conti, and T. Dargahi, Eds. Berlin: Springer, 2018, pp. 47–70.

[17] (2023, Apr.) What is an autonomous system? | what are asns? [Online]. Available: https://www.cloudflare.com/learning/network-layer/what-is-an-autonomous-system/.

[18] (2023, Apr.) YouTube Hijacking: A RIPE NCC RIS case study. [Online]. Available: http://www.ripe.net/internet-coordination/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study.

[19] (2023, Apr.) RFC 4632 - Classless inter-domain routing (CIDR): The Internet address assignment and aggregation plan. [Online]. Available: https://datatracker.ietf.org/doc/rfc4632/.

[20] B. Al-Musawi, P. Branch, and G. Armitage, "BGP anomaly detection techniques: a survey," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 377–396, 2017.

[21] (2023, Apr.) RFC 1930 - Guidelines for creation, selection, and registration of an Autonomous System (AS). [Online]. Available: https://www.rfc-editor.org/rfc/pdfrfc/rfc1930.txt.pdf.

[22] B. Zhang, R. Liu, D. Massey, and L. Zhang, "Collecting the Internet AS-level topology," *ACM Comput. Commun. Rev.*, vol. 35, no. 1, pp. 53–62, 2005.

[23] (2023, Apr.) RIPE NCC routing information service. [Online]. Available: https://www.ripe.net/publications/docs/ripe-200.

[24] (2023, Apr.) University of Oregon Route Views projects. [Online]. Available: http://www.routeviews.org.

[25] (2023, Apr.) Quagga routing suite. [Online]. Available: https://www.quagga.net/index.html.

[26] (2023, Apr.) Route Collection Raw Data: MRT Files. [Online]. Available: https://ris.ripe.net/docs/20_raw_data_mrt.html.

[27] M. Gopal, "Well-posed machine learning problems," in *Applied Machine Learning.* New York, NY: McGraw-Hill Education, 2019.

[28] (2023, Apr.) Machine Learning and Cyber Security: An Introduction. [Online]. Available: https://www.vmray.com/cyber-security-blog/machine-learning-and-cyber-security-an-introduction/.

[29] C. M. Bishop, *Pattern Recognition and Machine Learning.* Secaucus, NJ, USA: Springer-Verlag, 2006.

[30] (2023, Apr.) Ransomware attacks and types - how encryption trojans differ. [Online]. Available: https://usa.kaspersky.com/resource-center/threats/ransomware-attacks-and-types.

[31] (2023, Apr.) What is an Attack Vector? 16 Common Attack Vectors in 2022. [Online]. Available: https://www.upguard.com/blog/attack-vector.

[32] (2023, Apr.) How ransomware attacks, SophosLabs. [Online]. Available: https://www.sophos.com/en-us/medialibrary/pdfs/technical-papers/sophoslabs-ransomware-behavior-report.pdf.

[33] M. Bhuyan, D. Bhattacharyya, and J. Kalita, "Network anomaly detection: methods, systems and tools," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 303–336, Mar. 2014.

[34] N. Duffield, P. Haffner, B. Krishnamurthy, and H. Ringberg, "Rule-based anomaly detection on ip flows," in *IEEE INFOCOM*, 2009, pp. 424–432.

[35] (2023, Apr.) What is Snort? [Online]. Available: https://www.snort.org

[36] M. Xu and X. Li, "BGP anomaly detection based on automatic feature extraction by neural network," in *IEEE 5th Information Technol. Mechatronics Eng.*, 2020, pp. 46–50.

[37] Q. Li, X. Zhang, X. Zhang, and P. Su, "Invalidating idealized bgp security proposals and countermeasures," *IEEE Trans. Dependable Secure Computing*, vol. 12, no. 3, pp. 298–311, 2015.

[38] A. Lutu, M. Bagnulo, C. Pelsser, O. Maennel, and J. Cid-Sueiro, "The BGP visibility toolkit: detecting anomalous internet routing behavior," *IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 1237–1250, Apr. 2016.

[39] Z. Li, A. L. Gonzalez Rios, and Lj. Trajković, "Machine learning for detecting anomalies and intrusions in communication networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2254–2264, July 2021.

[40] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: a highly efficient gradient boosting decision tree," in *Proc. Int. Conf. Neural Inform. Process. Syst.*, Long Beach, CA, USA, Dec. 2017, pp. 3146–3154.

[41] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[42] C. L. P. Chen, Z. Liu, and S. Feng, "Universal approximation capability of broad learning system and its structural variations," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 4, pp. 1191–1204, Apr. 2019.

[43] (2023, Apr.) Border Gateway Protocol Routing Records from Réseaux IP Européens (RIPE) and BCNET. [Online]. Available: http://ieee-dataport.org/1977.

[44] (2023, Apr.) NSL-KDD Data Set. [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html.

[45] (2023, Apr.) Canadian Institute for Cybersecurity datasets. [Online]. Available: https://www.unb.ca/cic/datasets/index.html.

[46] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network traffic anomaly detection techniques and systems," in *Network Traffic Anomaly Detection and Prevention: Concepts, Techniques, and Tools.* Cham: Springer International Publishing, 2017, pp. 115–169.

[47] M. Thottan and J. Chuanyi, "Anomaly detection in IP networks," *IEEE Trans. Signal Process.*, vol. 51, no. 8, pp. 2191–2204, 2003.

[48] A. Sheikh, *Certified ethical hacker (CEH) preparation guide: lesson-based review of ethical hacking and penetration testing.* Berkeley, CA: Apress L. P, 2021.

[49] G.-J. Mun, Y.-M. Kim, D. Kim, and B.-N. Noh, "Network intrusion detection using statistical probability distribution," in *Proc. Comput. Sci. Appl.*, M. L. Gavrilova, O. Gervasi, V. Kumar, C. J. K. Tan, D. Taniar, A. Laganá, Y. Mun, and H. Choo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 340–348.

[50] T. M. Chen and J.-M. Robert, "The evolution of viruses and worms," in *Statistical Methods in Computer Security.* CRC press, 2004, pp. 289–310.

[51] (2023, Mar.) A brief history of computer viruses & what the future holds. [Online]. Available: https://www.kaspersky.com/resource-center/threats/a-brief-history-of-computer-viruses-and-what-the-future-holds.

[52] (2023, Mar.) 1970s. [Online]. Available: https://encyclopedia.kaspersky.com/knowledge/years-1970s/.

[53] (2023, Mar.) Morris worm. [Online]. Available: https://www.fbi.gov/history/famous-cases/morris-worm.

[54] (2023, Apr.) What's the difference between a virus and a worm? [Online]. Available: https://www.kaspersky.com/resource-center/threats/computer-viruses-vs-worms

[55] D. J. Marchette, "Computer viruses and worms," in *Computer Intrusion Detection and Network Monitoring: A Statistical Viewpoint.* New York, NY: Springer New York, 2001, pp. 215–240.

[56] (2023, Apr.) The Code Red worm, SANS Institute Information Security Reading Room. [Online]. Available: https://www.sans.org/reading-room/whitepapers/malicious/code-red-worm-85.

[57] (2023, Apr.) Responding to the Nimda worm: recommendations for addressing blended threats, Symantec, Cupertino, CA, USA. [Online]. Available: https://vx-underground.org/archive/Symantec/nimda-worm-recommendations-blended-threats-01-en.pdf.

[58] (2023, Apr.) A challenging response to Nimda, SANS Institute GIAC Certifications. [Online]. Available: https://www.giac.org/paper/gcih/273/challenging-response-nimda/102847.

[59] (2023, Mar.) Attack of Slammer worm - a practical case study, SANS Institute GIAC Certifications. [Online]. Available: https://www.giac.org/paper/gcih/414/attack-slammer-worm-practical-case-study/103632.

[60] (2023, Mar.) Attack of Slammer worm - a practical study, SANS Institute. [Online]. Available: www.giac.org/paper/gcih/414/attack-slammer-worm-practical-case-study.

[61] (2023, Apr.) The Mechanisms and effects of the Code Red worm, SANS Institute. [Online]. Available: https://www.sans.org/white-papers/87/.

[62] K. Vichova and M. Hromada, "Power outage in the hospitals," in *Proc. 2019 Int. Conf. Intell. Medicine Image Process.*, 2019, pp. 15–20.

[63] Z. Li, A. L. Gonzalez Rios, and Lj. Trajković, "Detecting internet worms, ransomware, and blackouts using recurrent neural networks," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Toronto, Canada, Oct. 2020, pp. 2165–2172.

[64] K. Gaur, N. Kumar, A. Handa, and S. K. Shukla, "Static ransomware analysis using machine learning and deep learning models," in *Proc. Int. Conf. Advances Cyber Secur.*, M. Anbar, N. Abdullah, and S. Manickam, Eds., Penang, Malaysia, Dec. 2020, pp. 450–467.

[65] (2023, Mar.) What is ransomware-as-a-service (raas)? [Online]. Available: https://www.cloudflare.com/en-ca/learning/security/ransomware/ransomware-as-a-service/.

[66] J. Gómez-Hernández, L. Álvarez González, and P. García-Teodoro, "R-locker: thwarting ransomware action through a honeyfile-based approach," *Comput. Secur.*, vol. 73, pp. 389–398, 2018.

[67] S. H. Kok, A. Abdullah, N. Jhanjhi, and M. Supramaniam, "Prevention of crypto-ransomware using a pre-encryption detection algorithm," *Comput.*, vol. 8, no. 4, p. 79, Nov. 2019.

[68] U. Adamu and I. Awan, "Ransomware prediction using supervised learning algorithms," in *Proc. 2019 7th Int. Conf. Future Internet Things Cloud*, 2019, pp. 57–63.

[69] C. Beaman, A. Barkworth, T. D. Akande, S. Hakak, and M. K. Khan, "Ransomware: recent advances, analysis, challenges and future research directions," *Comput. Secur.*, vol. 111, p. 102490, Dec. 2021.

[70] I. Nadir and T. Bakhshi, "Contemporary cybercrime: a taxonomy of ransomware threats & mitigation techniques," in *Proc. 2018 Int. Conf. Comput., Math. Eng. Technologies*, 2018, pp. 1–7.

[71] F. Cicala and E. Bertino, "Analysis of encryption key generation in modern crypto ransomware," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 2, pp. 1239–1253, Mar. 2022.

[72] P. Bajpai, A. K. Sood, and R. Enbody, “A key-management-based taxonomy for ransomware,” in *Proc. 2018 APWG Symp. Electron. Crime Res.*, 2018, pp. 1–12.

[73] Monika, P. Zavarsky, and D. Lindskog, “Experimental analysis of ransomware on windows and android platforms: evolution and characterization,” *Procedia Comput. Sci.*, vol. 94, pp. 465–472, 2016.

[74] J. Tailor and A. Patel, “A comprehensive survey: ransomware attacks prevention, monitoring and damage control,” *Int. J. Scientific Innov.*, vol. 4, 06 2017.

[75] (2023, Apr.) CVE-2018-8453 Detail. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2018-8453.

[76] I. Bello, H. Chiroma, U. A. Abdullahi, A. Y. Gital, F. Jauro, A. Khan, J. O. Okesola, and S. M. Abdulhamid, “Detecting ransomware attacks using intelligent algorithms: recent development and next direction from deep learning and big data perspectives,” *J. Ambient Intell. Humanized Comput.*, vol. 12, no. 9, pp. 8699–8717, Sep 2021.

[77] (2023, Apr.) Stemming the exploitation of ict threats and vulnerabilities, United Nations Institute for Disarmament Research (UNIDIR). [Online]. Available: https://unidir.org/files/publications/pdfs/stemming-the-exploitation-of-ict-threats-and-vulnerabilities-en-805.pdf.

[78] R. Agrawal, J. W. Stokes, K. Selvaraj, and M. Marinescu, “Attention in recurrent neural networks for ransomware detection,” in *Proc. 2019 IEEE Int. Conf. Acoustics, Speech Signal Process.*, 2019, pp. 3222–3226.

[79] (2023, Apr.) EternalBlue: a prominent threat actor of 2017–2018, Virus Bulletin. [Online]. Available: https://www.virusbulletin.com/uploads/pdf/magazine/2018/201806-EternalBlue.pdf.

[80] (2023, Mar.) What is EternalBlue and why is the MS17-010 Exploit still relevant? [Online]. Available: https://bit.ly/3td13iz.

[81] (2023, Apr.) About the SMBv1 retirement. [Online]. Available: https://kb.iu.edu/d/aumn.

[82] (2023, Mar.) Reverse Engineering of WannaCry Worm and Anti Exploit Snort Rules, SANS Institute. [Online]. Available: https://www.sans.org/reading-room/whitepapers/malicious/reverse-engineering-wannacry-worm-anti-exploit-snort-rules-38445.

[83] (2023, Apr.) Technical whitepaper tracking the WannaCry ransomware, Nominet. [Online]. Available: https://satisnet.co.uk/wp-content/uploads/2019/04/WannaCry-Whitepaper.pdf.

[84] (2023, Mar.) Westrock. [Online]. Available: https://ir.westrock.com.

[85] (2023, Apr.) Packaging Giant WestRock Says Ransomware Attack Impacted OT Systems. [Online]. Available: https://www.securityweek.com/packaging-giant-westrock-says-ransomware-attack-impacted-ot-systems.

[86] (2023, Apr.) WestRock Provides Update on Ransomware Incident. [Online]. Available: https://ir.westrock.com/press-releases/press-release-details/2021/WestRock-Provides-Update-on-Ransomware-Incident-8dfde2fca/default.aspx.

[87] (2023, Apr.) Zebra. [Online]. Available: http://www.zebra.org.

[88] (2023, Apr.) Zebra-dump-parser. [Online]. Available: https://github.com/rfc1036/zebra-dump-parser.

[89] (2023, Apr.) BGP C sharp tool. [Online]. Available: https://github.com/communication-networks-laboratory/BGP_c_sharp_tool.

[90] Y. Li, H. J. Xing, Q. Hua, X. Z. Wang, P. Batta, S. Haeri, and Lj. Trajković, "Classification of BGP anomalies using decision trees and fuzzy rough sets," in *Proc. IEEE Trans. Syst. Man Cybern.*, San Diego, CA, USA, Oct. 2014, pp. 1331–1336.

[91] G. H. John, R. Kohavi, and K. Pfleger, "Irrelevant features and the subset selection problem," in *Proc. Int. Conf. Mach. Learn.*, New Brunswick, NJ, USA, July 1994, pp. 121–129.

[92] M. N. A. Kumar and H. S. Sheshadri, "On the classification of imbalanced datasets," *Int. J. Comput. Appl.*, vol. 44, no. 8, pp. 1–7, Apr. 2012.

[93] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* Cambridge, MA, USA: The MIT Press, 2016.

[94] M. Espadoto, R. M. Martins, A. Kerren, N. S. T. Hirata, and A. C. Telea, "Toward a quantitative survey of dimension reduction techniques," *IEEE Trans. Visualization Comput. Graph.*, vol. 27, no. 3, pp. 2153–2173, 2021.

[95] P. Cunningham, "Dimension reduction," in *Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval*, M. Cord and P. Cunningham, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 91–112.

[96] K. P. Murphy, *Machine Learning: A Probabilistic Perspective.* Cambridge, MA, USA: The MIT Press, 2012.

[97] A. Zheng, *Feature engineering for machine learning : principles and techniques for data scientists / Alice Zheng and Amanda Casari.*, 1st ed. O'Reilly, 2018.

[98] (2023, Apr.) Principal components (pca) and exploratory factor analysis (efa) with spss. [Online]. Available: https://stats.oarc.ucla.edu/spss/seminars/efa-spss/.

[99] H. Steinhaus, "Sur la division des corps matériels en parties," *Bulletin Polish Academy Sciences*, vol. 4, no. 12, pp. 801–804, October 1956.

[100] A. A. Patel, *Hands-on unsupervised learning using Python : how to build applied machine learning solutions from unlabeled data*, 1st ed. O'Reilly, 2019.

[101] C. Ding and X. He, "K-means clustering via principal component analysis," in *Proc. 21^{st} Int. Conf. Mach. Learn.*, 2004, p. 29.

[102] M. P. Allen, "The problem of multicollinearity," in *Understanding Regression Analysis.* Boston, MA: Springer US, 1997, pp. 176–180.

[103] K. I. Sundus, B. H. Hammo, M. B. Al-Zoubi, and A. Al-Omari, "Solving the multicollinearity problem to improve the stability of machine learning algorithms applied to a fully annotated breast cancer dataset," *Informatics in Medicine Unlocked*, vol. 33, p. 101088, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352914822002246

[104] G. P. Dubey and D. R. K. Bhujade, "Optimal feature selection for machine learning based intrusion detection system by exploiting attribute dependence," *Mater. Today*, vol. 47, pp. 6325–6331, 2021.

[105] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, Mar. 1986.

[106] X.-Z. Wang, L. C. Dong, and J. H. Yan, "Maximum ambiguity based sample selection in fuzzy decision tree induction," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 8, pp. 1491–1505, Aug. 2012.

[107] (2023, Apr.) A simple explanation of information gain and entropy. [Online]. Available: https://victorzhou.com/blog/information-gain/.

[108] L. Breiman, "Random forests," *Mach. Lear.*, vol. 45, no. 1, pp. 5–32, Jan. 2001.

[109] P. Geurts, D. Ernst, and L.Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, Apr. 2006.

[110] G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts, "Understanding variable importances in forests of randomized trees," in *Proc. Int. Conf. Neural Inform. Process. Syst.*, Lake Tahoe, NV, USA, Dec. 2013, pp. 431–439.

[111] B. E. Baaquie, "Probability distribution functions," in *Mathematical Methods and Quantum Mathematics for Economics and Finance.* Singapore: Springer Singapore, 2020, pp. 217–243.

[112] M. R. Spiegel and L. J. Stephens, *Schaum's outline of statistics.* New York, NY: McGraw-Hill Education., 2018.

[113] (2023, Mar.) Continuous distributions. [Online]. Available: https://www.mathworks.com/help/stats/continuous-distributions.html.

[114] (2023, Mar.) Gamma distribution - intuition, derivation, and examples. [Online]. Available: https://towardsdatascience.com/gamma-distribution-intuition-derivation-and-examples-55f407423840.

[115] M. King, *Statistics for Process Control Engineers: a practical approach.* New York: John Wiley & Sons, Incorporated, 2017.

[116] N. T. Thomopoulos, "Exponential," in *Statistical Distributions: Applications and Parameter Estimates.* Cham: Springer International Publishing, 2017, pp. 21–29.

[117] A. C. Harvey, “Location/scale models for non-negative variables,” in *Dynamic Models for Volatility and Heavy Tails: With Applications to Financial and Economic Time Series*, ser. Econometric Society Monographs. Cambridge University Press, 2013, pp. 149–186.

[118] “Heavy-tailed distribution,” in *Encyclopedia of Operations Research and Management Science*, S. I. Gass and M. C. Fu, Eds. Boston, MA: Springer US, 2013, pp. 693–694.

[119] M. Ibragimov, R. Ibragimov, and J. Walden, “Introduction,” in *Heavy-Tailed Distributions and Robustness in Economics and Finance.* Cham: Springer International Publishing, 2015, pp. 1–9.

[120] L. Rayleigh", “Xii. on the resultant of a large number of vibrations of the same pitch and of arbitrary phase,” *The London, Edinburgh, Dublin Philosophical Magazine J. Sci.*, vol. 10, no. 60, pp. 73–78, 1880.

[121] F. Merovci and I. Elbatal, “Weibull Rayleigh distribution: theory and applications,” *Appl. Math. Inf. Sci*, vol. 9, no. 5, pp. 1–11, 2015.

[122] S. J. Fletcher, “Univariate distribution theory,” in *Data Assimilation for the Geosciences.* Elsevier, 2017, pp. 29–124.

[123] I. W. Burr, “Cumulative frequency functions,” *Ann. Math. Statist.*, vol. 13, no. 2, pp. 215–232, 1942.

[124] A. Urbán, A. Groniewsky, M. Malý, V. Józsa, and J. Jedelský, “Application of big data analysis technique on high-velocity airblast atomization: searching for optimum probability density function,” *Fuel*, vol. 273, p. 117792, 2020.

[125] F. Galton, *Memories of my life.* New York: E.P. Dutton, 1909.

[126] N. T. Thomopoulos, “Lognormal,” in *Probability Distributions : With Truncated, Log and Bivariate Extensions.* Cham: Springer International Publishing, 2018, pp. 135–148.

[127] R. A. Bisland and E. Scheuermann, “A goodness of fit algorithm for empirical data,” in *Proc. Annu. Southeast Regional Conf.*, Apr 1978, pp. 30–33.

[128] D. L. Byrkett, “Classroom goodness of fit testing using microsoft windows,” in *Proc. 23rd Conf. Winter Simul.*, Phoenix, AZ, USA, Dec. 1991, pp. 882–886.

[129] Y. Dodge, *The Concise Encyclopedia of Statistics.* New York, NY: Springer Science & Business Media, 2008.

[130] “Kolmogorov–smirnov test,” in *The Concise Encyclopedia of Statistics.* New York, NY: Springer New York, 2008, pp. 283–287.

[131] F. J. Massey, “The Kolmogorov-Smirnov test for goodness of fit,” *J. American Statistical Assoc.*, vol. 46, no. 253, pp. 68–78, 1951.

[132] (2023, Apr.) Nonparametric statistics and model selection (2015). [Online]. Available: http://www.mit.edu/~6.s085/notes/lecture5.pdf.

[133] B. Neal, S. Mittal, A. Baratin, V. Tantia, M. Scicluna, S. Lacoste-Julien, and I. Mitliagkas, “A modern take on the bias-variance tradeoff in neural networks,” *arXiv preprint arXiv:1810.08591*, 2018.

[134] E. Briscoe and J. Feldman, “Conceptual complexity and the bias/variance tradeoff,” *Cognition*, vol. 118, no. 1, pp. 2–16, 2011.

[135] M. M. Hossain and M. S. Miah, “Evaluation of different SVM kernels for predicting customer churn,” in *Proc. 18th Int. Conf. Comput. Inform. Technol.*, Dhaka, Bangladesh, Dec. 2015, pp. 1–4.

[136] D. A. Winkler and T. C. Le, “Performance of deep and shallow neural networks, the universal approximation theorem, activity cliffs, and qsar,” *Molecular Informatics*, vol. 36, no. 1–2, p. 1600118, 2017.

[137] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Mach. Learn.*, vol. 8, no. 3, pp. 229–256, May 1992.

[138] A. Graves, A. Mohamed, and G. E. Hinton, “Speech recognition with deep recurrent neural networks,” in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, Vancouver, BC, Dec. 2013, pp. 6645–6649.

[139] Y. LeCun, Y. Bengio, and G. E. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[140] L. Deng and D. Yu, “Deep learning: methods and applications,” *Foundations and Trends in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[141] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, 1994.

[142] M. C. Mozer, “Induction of multiscale temporal structure,” in *Proc. Advances Neural Information Processing Systems*, J. Moody, S. Hanson, and R. Lippmann, Eds., vol. 4. Morgan-Kaufmann, 1992, pp. 275–282.

[143] J. Schmidhuber, “Learning complex, extended sequences using the principle of history compression,” *Neural Computation*, vol. 4, no. 2, pp. 234–242, Mar. 1992.

[144] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Oct. 1997.

[145] A. Graves, “Long short-term memory,” in *Supervised Sequence Labelling with Recurrent Neural Networks.* Berlin, Heidelberg: Springer, 2012, pp. 37–45.

[146] (2022, Apr.) PyTorch. [Online]. Available: https://pytorch.org.

[147] I. Loshchilov and F. Hutter, “SGDR: stochastic gradient descent with warm restarts using statistical probability distribution,” in *Proc. 5th Int. Conf. Learn. Representations.* Toulon, France: OpenReview.net, Apr. 2017.

[148] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, “A survey on ensemble learning,” *Frontiers Comp. Sci.*, vol. 14, pp. 241–258, Apr. 2020.

[149] (2023, Apr.) Bagging. [Online]. Available: https://www.ibm.com/cloud/learn/bagging.

[150] (2023, Apr.) Bagging in Machine Learning: Step to Perform And Its Advantages. [Online]. Available: https://www.simplilearn.com/tutorials/machine-learning-tutorial/bagging-in-machine-learning.

[151] D. P. Mohandoss, Y. Shi, and K. Suo, "Outlier prediction using random forest classifier," in *Proc. 2021 IEEE 11th Annual Computing Communication Workshop Conf.*, NV, USA, 2021, pp. 27–33.

[152] (2023, Apr.) Ensemble methods: bagging, boosting and stacking. [Online]. Available: https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205.

[153] (2023, Apr.) What is Boosting? [Online]. Available: https://aws.amazon.com/what-is/boosting/.

[154] J. Friedman, "Greedy function approximation: a gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, Apr. 2001.

[155] T. Chen and C. Guestrin, "XGBoost: a scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, San Francisco, CA, USA, Aug. 2016, pp. 785–794.

[156] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: unbiased boosting with categorical features," in *Proc. Int. Conf. Neural Inform. Process. Syst.*, Montreal, Quebec, Canada, Dec. 2018, pp. 6639–6649.

[157] J. Korstanje, *Gradient boosting with XGBoost and LightGBM.* Berkeley, CA: Apress, 2021, pp. 193–205.

[158] D. Micci-Barreca, "A preprocessing scheme for high-cardinality categorical attributes in classifcation and prediction problems," *ACM SIGKDD Explorations Newsletter*, vol. 3, no. 1, pp. 27–32, 2001.

[159] (2023, Apr.) Attention mechanism in deep learning, explained. [Online]. Available: https://bit.ly/3GWI7ug

[160] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. U. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Advances Neural Inf. Process. Syst.*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[161] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. Int. Conf. Learn. Representations*, San Diego, CA, May 2015.

[162] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention–based models for speech recognition," *Advances Neural Inf. Process. Syst.*, vol. 28, 2015.

[163] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio, “Show, attend and tell: neural image caption generation with visual attention,” in *Proc. 32nd Int. Conf. Mach. Learn.*, vol. 37, Lille, France, July 2015.

[164] (2023, Apr.) A comprehensive guide to attention mechanism in deep learning for everyone. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/11/comprehensive-guide-attention-mechanism-deep-learning/.

[165] (2023, Apr.) Random forest classifier. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[166] Z. Li, A. L. Gonzalez Rios, and Lj. Trajković, “Machine learning for detecting the westrock ransomware attack using BGP routing records,” *IEEE Comm. Magazine*, pp. 1–7, 2022, early access.

[167] (2023, Apr.) Statistical tests — when to use which ? [Online]. Available: https://towardsdatascience.com/statistical-tests-when-to-use-which-704557554740.

[168] (2023, Apr.) K-S Test. [Online]. Available: https://www.mathworks.com/help/stats/kstest.html.

[169] (2023, Apr.) Understanding hypothesis tests: significance levels (alpha) and p-values in statistics. [Online]. Available: https://bit.ly/3AP6UkH.

[170] (2023, Apr.) A simple LSTM-based time-series classifier. [Online]. Available: kaggle.com/code/purplejester/a-simple-lstm-based-time-series-classifier/notebook

[171] (2023, Apr.) Adding a custom attention layer to a recurrent neural network in keras. [Online]. Available: https://machinelearningmastery.com/adding-a-custom-attention-layer-to-recurrent-neural-network-in-keras/.

Appendix A

Principal Component Analysis

Listed is Python code developed for data transformation using PCA and performance evaluation of machine learning GBDT algorithms.

Listing A.1: Data transformation using PCA algorithm and performance evaluation of GBDT classifiers.

```python
/label{}
#Step 1:

#load libraries

import numpy as np
from keras import optimizers
import matplotlib.pyplot as plt
from keras import backend as K
from keras.utils import np_utils
import tensorflow as tf

from tensorflow.keras import regularizers
from sklearn.model_selection import KFold
from scipy.stats import zscore
import pdb
from sklearn.preprocessing import StandardScaler
import pandas as pd
import math
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
np.random.seed(1)

file= 'Code_Red_I.csv' #change file name as desired

df_data=pd.read_csv(file,header=None)
df_data_org = df_data

df_data=np.asfarray(df_data)
cols_num = df_data.shape[1]

df_labels=df_data[:, -1].reshape(-1,1) #it is a column vector
df_labels = np.where(df_labels == -1, 0, df_labels)
```

```
print("**********************************************************")
print(df_labels)
labels = df_labels[:, -1]
print("***********************************************************")

df_data=df_data[:, : cols_num-1] #drop the labels from the main dataframe
print('size of the dataframe is:- ', df_data.shape)

idx_days=[]
anomaly_days=[]
temp=24*60
dates=[10]
dict_data={}

#Generates loop for claculating indexes of data points and important days
for i in range(9):
  val = temp*i
  if i ==0:
    idx_days.append(val)
  else:
    idx_days.append(val-1)
    dates.append(dates[i-1]+1)

#creates key value pair between data and the respective dates
for i in range(9):
  date=dates[i]
  idx_days
  if i ==0:
    dict_data[str(date)] = df_data[0 : idx_days[i+1]]
  elif i>0 and i<8 :
    print(i, i+1)
    dict_data[str(date)] = df_data[idx_days[i]+1 : idx_days[i+1]]

print("======================== dict_data:- ================================
    ")
#print('dict-data: ', dict_data)
print("======================== dict_data:- ================================
    ")
print('val of idxs:- ',idx_days)
print('val of dates:- ',dates)

print("======================== df_data:- ================================="
    )
print(df_data.shape[1])
print(df_data[:, -1])
print(df_labels.shape)

#lets calculate the total num of anomalous points:-
total_points=-2880+8640+1
split_percnt= np.ceil(5760*0.60)

split_idx=split_percnt
partition_idx=split_idx+2879
print("partition idx: ");
print(partition_idx)


#slammer: 3740
```

```python
# Nimda: 4500
# Code Red: 4040
# WannaCrypt: 6300
# WestRock: 8960


partition_idx = 4040
x_train = df_data[:int(partition_idx),: cols_num-1]
y_train = df_labels[:int(partition_idx),: cols_num-1]


y_train_org = y_train

print('x train val is:- ', x_train[-1,0])
x_test = df_data[int(partition_idx):, : cols_num-1]
y_test = df_labels[int(partition_idx):, : cols_num-1]
print('x test val is:- ', x_test[0,0])

x_train_o=x_train
x_test_o=x_test

# #standardizing the data
scaler_train_S = StandardScaler()
x_train = scaler_train_S.fit_transform(x_train)
print('df-train val1: ', x_train[0,0])

scaler_test_S = StandardScaler()
x_test = scaler_test_S.fit_transform(x_test)
print('df-train val1: ', x_test[0,0])

print(x_train.shape)
print(x_test.shape)

print(y_train.shape)
print(y_train[-1])
print(y_test.shape)
print("=======================================================")

print('df-train val1: ', df_data[0,0])
x_train_std=x_train
x_test_std=x_test

#reshaping
#x_train = np.reshape(x_train, (x_train.shape[0], 1, x_train.shape[1]))
#x_test = np.reshape(x_test, (x_test.shape[0], 1, x_test.shape[1]))

y_train_o = y_train
y_test_o = y_test


print("===============================================================")

print("y_train_o: ", y_train_o)

print("===============================================================")


print("the y_train and test orignal vectors: ", y_train_o.shape, " " ,
    y_test_o.shape)
y_train = np_utils.to_categorical(y_train, 2)
y_test = np_utils.to_categorical(y_test, 2)

print(y_train)

print(y_train.shape)

print('modified shape: ', y_train.shape)
print("=====================================================================")

learning_rate = 0.001
opt = tf.optimizers.Adam(learning_rate)

#Step 2:

from sklearn.datasets import load_digits
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import numpy as np
import matplotlib.pyplot as plt
import xgboost
import time
from sklearn.metrics import f1_score

from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from lightgbm import LGBMClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, f1_score,
    precision_score, recall_score, roc_curve, roc_auc_score
from sklearn.metrics import silhouette_samples, silhouette_score

sscore=[]
inertia_list=[]
bce = tf.keras.losses.BinaryCrossentropy(from_logits=True)

for i in range(10,11):

    pca = PCA(i)
    pca_fit = pca.fit(x_train)
    print('The index value is: ', str(i))
    print('check shape of pca fit: ', pca_fit)
    pca_transform = pca.transform(x_train)
    print('check shape of pca transform: ', pca_transform.shape)

    df = pca.fit_transform(x_train)
    df_test = pca.fit_transform(x_test)
    print('Index: ', i, ' check shape PCA fit and transform: ', df.shape, "
    ", df_test.shape)
    print(df.shape)


    time_start = time.time()
    # model = xgboost.XGBClassifier(criterion=bce, random_state=0,
    learning_rate=0.1, n_estimators=10) #objective='binary:logistic',
    criterion= 'mse'; bce = tf.keras.losses.BinaryCrossentropy(from_logits=
    True)
    model = LGBMClassifier(learning_rate = 0.1, random_state = 0,
   n_estimators = 270)

    clf = model.fit(df,y_train_o)
    time_end = time.time()

    #Predict the response for test dataset
    y_pred = clf.predict(df_test)
    print(y_pred)
    print('org: ', y_pred)

    print("Accuracy y_pred: ", y_pred.shape , "   ", df_test.shape)

    f=f1_score(y_test_o, y_pred)
    accuracy = accuracy_score(y_test_o, y_pred)
    print('Accuracy', accuracy)
    print('F-Score: ', f1_score(y_test_o, y_pred))

    cm = confusion_matrix(y_test_o,y_pred).ravel()
    print('confusion matrix: ',cm)

    f=f1_score(y_test_o,y_pred)
    print('fscore: ', f)
    print('precision: ',precision_score(y_test_o, y_pred).ravel())
    print('Training time: ', time_end-time_start )
    print('recall: ', recall_score(y_test_o, y_pred).ravel())

    print('roc: ', roc_curve(y_test_o, y_pred))

    print('auc: ',roc_auc_score(y_test_o, y_pred).ravel())
```

Appendix B

Correlation Coefficient

Listing B.1: Spearman correlation coefficient calculation.

```
close all;
clear all;
% #slammer: 3740
%      # Nimda: 4500
%      # Code Red: 4040
%      # WannaCrypt: 6300
%      # WestRock: 8960

partition= 8960
Data = csvread('ransomware_rrc14_2021_westRock.csv'); %
    ransomware_rrc14_2021_westRock

[RHO,PVAL] = corr(Data(1:partition,5:41),'Type','Spearman');
RHO(RHO==NaN)=0;

A=RHO;
A(A<=0.7)=0;
h2 = heatmap(A, 'XLabel', 'Feature number', 'YLabel', 'Feature number')
```

Listing B.2: Spearman correlation coefficient calculation.

```
from numpy.random import rand
from numpy.random import seed
from scipy.stats import spearmanr
import seaborn as sns
import matplotlib
# seed random number generator
seed(1)

# calculate spearman's correlation
coef, p = spearmanr(df_data_org)

df = pd.DataFrame(DF_train)
arr_label=[]
for i in range(37):
    arr = 'F'+str(i+1)
    arr_label.append(arr)

```

```python
df.columns = arr_label

corr = df.corr(method = 'spearman')
matplotlib.rcParams['figure.dpi'] = 120
matplotlib.rcParams['figure.figsize'] = (20,18)
ax= sns.heatmap(corr, annot = True, linewidths=.3)

for t in ax.texts:
    if float(t.get_text())>=0.9:
        t.set_text(t.get_text()) #if the value is greater than 0.4 then I set the text
    else:
        t.set_text("")

plt.show()
```

Listing B.3: Correlation coefficient visualization.

```python
!pip install biokit

import matplotlib
matplotlib.rcParams['figure.dpi'] = 120
matplotlib.rcParams['figure.figsize'] = (12,10)

df = pd.DataFrame(DF_train)
arr_label=[]
for i in range(37):
    arr = 'F'+str(i+1)
    arr_label.append(arr)

df.columns = arr_label

from biokit.viz import corrplot
c= corrplot.Corrplot(df)
c.plot(method='text', fontsize=6, colorbar=False)#, tl.pos="n")
```