2. Chapter 3 describes the generation process of malware datasets and their evaluation, aiming to assess the efficacy of malware defense systems. The characteristics of the generated dataset along with its performance evaluation are also reported.

3. Chapter 4 presents a supervised filter-based feature selector. The proposed method selects the optimal feature subset by considering the relevance and significance criteria of each feature. For this, rough set theory is used and also, a new criterion is introduced to select the optimal feature subset. The performance of the proposed method is also reported.

4. Chapter 5 presents a fast, yet reliable ransomware defense solution, referred to as ERAND, powered by an optimal feature selection method to discriminate the ransomware class as a whole, as well as the variants of the ransomware family from the goodware instances.

5. Chapter 6 presents an ensemble approach called FRAMC to identify the key features that contribute significantly to the detection of malware. The effectiveness of FRAMC is assessed using different types of classifiers on a number of real-world malware datasets.

6. Chapter 7 introduces TUKNN, a parallel version of the k-Nearest Neighbors (KNN) algorithm designed to improve speed and efficiency through parallel processing. The chapter includes an extensive experimental study on various proximity measures, recommending the most effective ones for better accuracy with TUKNN. Additionally, it identifies the optimal range of k values for malware and malware-based attack datasets to ensure the best performance.

7. Chapter 8 presents a Convolutional Neural Network (CNN)-based malware defense solution. This chapter explores various CNN methods for the classification and detection of malware, leveraging their advanced capabilities to identify and recognize complex patterns within the data.

8. Chapter 9 presents a Graph Neural Network (GNN)-based malware defense solution. This method leverages the unique capabilities of GNNs to model and analyze function call graphs of malware, enabling effective detection and classification of malware.

9. Finally, Chapter 10 summarizes the overall contributions of this thesis and identifies future research directions in the domain of malware detection and defense.

Chapter 2

Background

2.1 Networks

A network refers to a collection of interconnected devices that can communicate and share resources with each other. These devices could include computers, servers, routers, switches, printers, and other types of hardware. Computer networks are designed to enable the exchange of data, information, and resources between devices, regardless of their physical location. Computer networks come in various sizes and scopes. Local Area Networks (LANs) cover small areas like a single building or home, connecting devices in close proximity using technologies such as Ethernet or Wi-Fi. On a larger scale, Wide Area Networks (WANs) extend over greater distances and often utilize the internet to link multiple LANs. Falling between these, Metropolitan Area Networks (MANs) connect various locations within a city or metropolitan area, offering a compromise in geographical coverage. Networks can also be classified based on their purpose. In a Client-Server Network, dedicated servers provide services or resources to client devices. Clients, such as computers or smartphones, request services from servers, which respond to these requests. In a Peer-to-Peer network, all devices are considered equal, with each device acting as both a client and a server. Direct communication and resource sharing occur without reliance on a central server. Hybrid Networks represent a combination of client-server and peer-to-peer architectures, offering flexibility to accommodate various applications and goals [4].

2.2 Network Components and Protocols

Computer networks are systems that allow devices to share data and communicate with each other. At their core, these networks rely on a variety of components and protocols to ensure efficient and secure information flow. Network components encompass devices such as routers, switches, servers, and client machines, each playing a specific role in the network's functionality. Routers manage data traffic between different networks, while switches facilitate communication within a local network. Servers store and provide access to shared resources, and client devices request and utilize these resources. The smooth operation of these components is governed by a set of protocols, most notably the Transmission Control Protocol (TCP) and Internet Protocol (IP) suite. TCP/IP defines the rules for data transmission, addressing, and routing, providing a standardized framework for communication across diverse networks. Additionally, other protocols such as HTTP, HTTPS, and FTP dictate specific rules for web browsing and file transfer. Together, these components and protocols form the foundation of computer networks, enabling the seamless exchange of information in the digital world [4] [5]
.

2.3 Network Architectures and Topologies

Network architecture refers to the design and organization of a network's components, structures, and functionalities. It defines how devices are interconnected and how data flows within the network [4] [5]. There are two primary network architecture models:

- Client-Server Architecture: In a client-server architecture, the network is organized around a central server that provides resources, services, or data to client devices. Clients, such as computers or devices, make requests to the server, which responds by fulfilling those requests. This model centralizes control, simplifying resource management and data storage. It is commonly used in enterprise environments, where servers host databases, applications, or shared files while clients access and utilize these resources.

- Peer-to-Peer Architecture: In a peer-to-peer (P2P) architecture, devices in the network, known as peers, have equal status and can act both as clients and servers. Peers share resources, such as files or processing power, directly

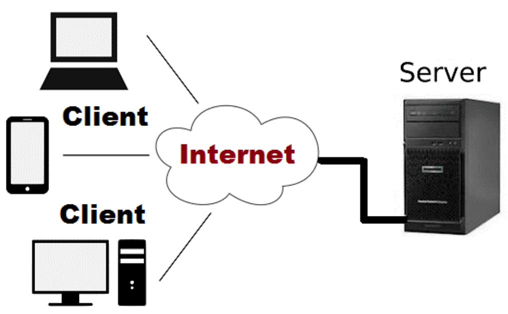

Figure 2-1: A conceptual framework of client-server architecture

with one another without relying on a centralized server. This decentralized approach fosters collaboration and can be more resilient, as there's no single point of failure. A P2P architecture is often found in file-sharing applications and distributed computing environments.

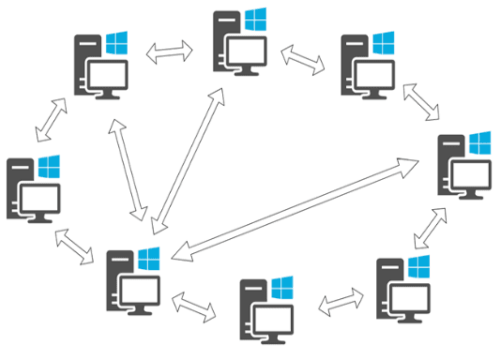

Figure 2-2: A conceptual framework of peer-peer architecture

Network architectures can also be categorized based on their topology, which describes the arrangement of nodes and links. Different topologies offer varying levels of redundancy, scalability, and fault tolerance.

- Star Topology: In a star topology, devices are connected to a central hub, which serves as a focal point for communication. The hub can simplify management and troubleshooting, but if the hub fails, the entire network may be affected.
- Mesh Topology: A mesh topology involves direct connections between all devices, providing high redundancy and fault tolerance. While resilient, this topology can be complex and resource-intensive to implement and maintain.
- Bus Topology: In a bus topology, devices are connected in a linear fashion along a single communication path. It is simple and cost-effective but can

suffer from performance degradation if multiple devices transmit simultaneously.

- Ring Topology: A ring topology features devices connected in a circular arrangement. Data circulates around the ring, and while it offers some fault tolerance, a single device failure can disrupt the entire network.
- Hybrid Topology: Hybrid topologies combine multiple basic topologies to suit specific needs. For instance, a combination of star and mesh topologies can provide redundancy and centralized management.

2.4 Network Devices

Network devices are essential components that facilitate the efficient flow of data within computer networks, ensuring seamless communication between connected devices. These devices play distinct roles in managing and directing data traffic, enhancing network performance, and enforcing security measures [4] [5]. Some of the common network devices are discussed below.

- Hubs: Hubs are simple networking devices operating at the physical layer (Layer 1) of the OSI model. Unlike more intelligent devices, hubs lack the ability to selectively transmit data to specific devices within a network. Instead, they broadcast incoming data to all connected devices, creating potential for increased network congestion. Hubs serve as basic connectivity points, linking multiple devices within a local network.
- Routers: Routers are crucial for directing data between different networks, determining the optimal path for information to travel. They operate at the network layer (Layer 3) of the OSI model and use routing tables to make decisions about the most efficient routes for data transmission.
- Switches: Switches are fundamental components in computer networking that operate at the data link layer (Layer 2) of the OSI model. Unlike traditional hubs, switches intelligently manage and forward data within local networks. These devices use MAC addresses to selectively direct data packets to the specific devices for which they are intended, reducing unnecessary network traffic and enhancing overall efficiency.
- Gateways: A network device that connects different networks, serving as a translator that enables communication between devices using distinct com-

munication protocols. Positioned at the intersection of disparate networks, gateways facilitate the seamless exchange of information by interpreting and converting data between incompatible formats. Essentially, gateways bridge the language gap between networks, ensuring that data can flow smoothly across heterogeneous environments.

- Access Points: Access Points (APs) are critical components in wireless networking that facilitate the connection between Wi-Fi-enabled devices and a wired local area network (LAN). Operating within the framework of wireless communication standards, such as Wi-Fi, APs serve as communication hubs, enabling seamless access for devices like laptops, smartphones, and tablets to the broader network infrastructure.

2.5 Operating Systems

An Operating System (OS) is a fundamental software component that acts as an intermediary between computer hardware and user applications [6]. It provides a set of essential services and functionalities, including process management, memory management, file systems, and user interface interactions. The OS is vital for coordinating hardware resources and enabling communication between software and hardware components. Various operating systems cater to different computing environments [7]. Windows, developed by Microsoft, is a widely used OS known for its user-friendly interface and extensive software compatibility. Linux, an open-source Unix-based OS, is acclaimed for its stability and security, commonly used in server environments. macOS, developed by Apple, powers their personal computers and emphasizes sleek design and integration. Android, an open-source mobile OS by Google, dominates the smartphone market, offering flexibility and a vast app ecosystem. iOS, also developed by Apple, powers iPhones and iPads, focusing on user privacy within a closed ecosystem. Operating systems are the backbone of computing, providing a platform for software applications and enabling efficient hardware use.

Each operating system creates a unique environment that can be targeted by various forms of malware. Windows, widely used in personal and enterprise settings, is a common target for malware due to its popularity, making it crucial for users to employ robust security measures like regular updates and antivirus software. Linux, known for its security features and open-source nature, is generally more resistant to malware but attracts sophisticated attacks in server environments. macOS, with its closed ecosystem and strict app store policies, experiences fewer malware incidents, but the growing popularity of Apple devices makes them increasingly attractive to attackers. Android, an open-source mobile OS, faces diverse threats, including malicious apps and phishing attacks. iOS, operating exclusively on Apple devices, benefits from a controlled environment, but users are not entirely immune to security risks.

2.6 Security Vulnerabilities

A security vulnerability is any weakness or flaw in a system's implementation, configuration, or design that could allow attackers to compromise the system's availability, integrity, or confidentiality. Both hardware and software can have vulnerabilities caused by factors like code errors, human mistakes, and outdated software [8].

Software vulnerabilities are weaknesses in the design, implementation, or configuration of software applications that can be exploited by malicious entities. These vulnerabilities pose significant risks to the security and integrity of computer systems, offering entry points for various cyber threats, including malware. A common software vulnerability is unpatched or outdated software. Failing to apply timely security updates and patches leaves systems exposed to known vulnerabilities that attackers can exploit [8].

Hardware vulnerabilities are potential weaknesses in the physical components of computing systems that can be exploited by malicious actors to compromise device security and functionality [9]. These vulnerabilities can arise from flaws in design and manufacturing processes or specific hardware components. Attacks targeting hardware vulnerabilities may exploit weaknesses in firmware or chipsets, compromising the device's integrity at a foundational level. Unauthorized physical access to devices also poses a significant risk, as attackers can manipulate or extract sensitive information directly from the hardware. Hardware vulnerabilities highlight the importance of robust physical security measures, including secure boot processes, encryption, and protection against tampering.

2.7 Malware Authors

In the complex world of cybersecurity, different groups of malware authors create and spread malicious software for various reasons and with varying degrees of skill. Script kiddies, who often lack deep technical expertise, use existing tools or scripts to launch attacks, driven more by a desire for fame than by financial gain. Hacktivists use malware to promote their ideological or political causes, aiming to disrupt or deface systems for social or political impact. Cybercriminals, motivated by financial gain, develop sophisticated malware to compromise systems, steal sensitive data, or engage in activities such as ransomware attacks. State-sponsored actors, backed by governments, use advanced and targeted malware for espionage or cyber warfare, focusing on gathering intelligence or disrupting adversaries. A growing concern is the rise of "malware-as-a-service" providers who offer malicious tools or services on the dark web, allowing individuals or groups to deploy malware without extensive technical expertise. Recognizing the diverse motivations and capabilities of malware authors is critical for developing effective cybersecurity strategies and mitigating the evolving threats posed by different actors in the digital realm.

2.8 Malware Attack Categories

Malware plays an important role in all kinds of network intrusion and security attacks. Any software that disrupts user data, computer, or network can be considered malware, including viruses, trojan horses, worms, rootkits, scareware, and spyware. We can categorize malware based on its ways of propagation, target platforms, infected objects, and evasion mechanisms. A taxonomy of malware is presented in Figure 2-3.

2.8.1 Propagation Methods

To invade the victim machine, the malware uses different transport mechanisms. These mechanisms enable malware propagation among information devices and systems, including mobile devices it seeks to infect. Some of the common malware propagation techniques are listed below. Table 2.1 lists the common propagation methods used by malware.

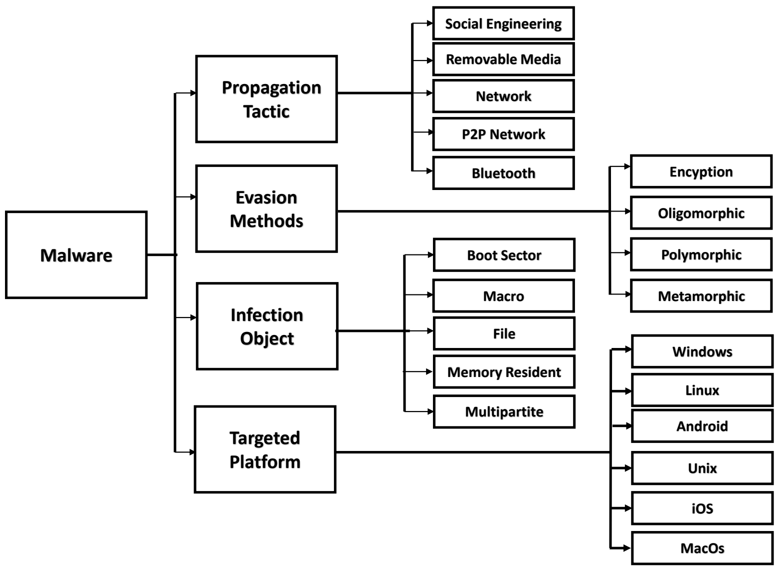

Figure 2-3: Taxonomy of malware

a) *Social engineering*: Attackers often employ techniques of social engineering to distribute malware over the Internet. It uses different elements of social engineering to lure victims to click on malicious links or to launch an infected file or to open a malicious email attachment. For example, the LoveLetter [1] worm uses the social engineering tactics to propagate and compromise the victim's system.

b) *Removable media*: This is the dominant propagation medium for malware. Malware often uses the USB and Flash drive devices to propagate. This propagation medium is particularly effective in crossing the physical air gap that exists between the Internet and some internal networks [10]. For example, the stuxnet[2] malware attaches itself to a USB device, and once on such a device; it propagates itself to machines that the USB device comes in contact with.

c) *Network*: Malware often propagates to a large number of systems connected to a network by exploiting the vulnerabilities in client side and server side applications. For example, the Blaster [3] malware spreads via network and infects the users system.

d) *P2P Network*: Due to increase in the users of peer to peer networks, attackers often use such networks to launch the malware onto the victim systems. One prerequisite to launch malware into a P2P network is the installation

[1]http://virus.wikidot.com/loveletter
[2]https://en.wikipedia.org/wiki/Stuxnet
[3]https://en.wikipedia.org/wiki/Blaster

of a vulnerable application on the client side. By exploiting vulnerabilities, malware can crawl through the network. For example, Alureon [4] malware uses the P2P network for propagation.

e) *Bluetooth*: Malware also uses Bluetooth as a medium to propagate among nearby Bluetooth enabled devices. Bluetooth is a wireless Personal Area Network (PAN), which can be compromised by malware using techniques such as bluejacking[5]. For example, the BlueBorne malware spreads via bluetooth.

2.8.2 Evasion Methods

Malware can be categorized based on the evasive techniques it uses to thwart detection systems. Below are the categories of malware evasive techniques. Table 2.2 lists some commonly used evasive techniques by malware to thwart malware detectors.

1. *Encrypted Malware*: This type of malware has two parts: encrypted body and decryptor code. The encrypted body contains the actual contents of the malware, and the decryptor code is used to decrypt the body into machine executable code. As the decryptor code is constant, this kind of malware can be easily detected with anti-malware programs. The CASCADE[6] malware is an example of encrypted malware.

2. *Oligomorphic Malware*: It is an enhanced version of the encrypted malware. It uses a collection of decryption keys which are used randomly for different target machines. Whale[7] is an example of oligomorphic malware.

3. *Polymorphic Malware*: It is designed to take advantage of weaknesses associated with signature based detection. Such malware creates a new decryption routine to unpack a constant malware body each time it executes. The functionality of the new decryption routine remains the same, but the sequence of instructions may be different [11]. It makes use of an encryption routine to encrypt the malware body and the new decryption routine is appended to create a variant of it. The first polymorphic malware family, the Chameleon

[4]https://en.wikipedia.org/wiki/Alureon
[5]https://en.wikipedia.org/wiki/Bluejacking
[6]https://en.wikipedia.org/wiki/Cascade
[7]http://malware.wikia.com/wiki/Whale

family (1260, V2P1, V2P2, and V2P6) was developed by Mark Washburn and Ralf Burger [12].

4. *Metamorphic Malware*: It is also known as body polymorphic malware [13]. It transforms its code in every iteration, making difficult for anti-malware programs to recognize it. Like polymorphic malware, it does not use any decryption routine to unpack a constant malware body, instead it generates a new body with the same fucntionality. It also uses different obfuscation techniques [14] to create new variants of a malware. Zmist[8] and Simile[9] are examples of metamorphic malware.

Table 2.1: Malware Propagation Methods

Propagation Methods	**Examples**
1) Social Engineering	LoveLetter worm
2) Removable Media	Stuxnet
3) Network	Blaster
4) P2P Network	Alureon
5) Bluetooth	Blueborne

Table 2.2: Common Malware Evasion Techniques

Evasion Methods	**Examples**
1) Encryption	CASCADE
2) Oligomorphic	Whale
3) Polymorphic	1260
4) Metamorphic	Zmist

2.8.3 Targeted Platform

Based on the target platform it is created for, malware can be categorized into various types. For example, there are malware groups for Windows, Linux, Unix, Android, MS-DOS, iOS and Mac OS X operating systems. Some modern malware can infect multiple platforms. For example, Banloader[10] and Adwind Rat[11] can infect multiple platforms.

[8]https://en.wikipedia.org/wiki/Zmist
[9]https://en.wikipedia.org/wiki/Simile
[10]https://securityintelligence.com/news/java-malware-becomes-a-cross-platform-threat/
[11]https://blog.malwarebytes.com/threat-analysis/2016/07/cross-platform-malware-adwind-infects-mac/

2.8.4 Infected Objects

Malware often targets an object to infect it. The type of infected objects varies among different forms of malware. Below, we present types of malware based on infected objects. Table 2.3 lists the common malware infectors.

a) *Boot Sector Infectors*: Such malware infects the boot sector or the Master Boot Record (MBR) of a hard disk to get control over the system. It is executed when the system is booted from an infected disk. The most common boot sector malware include Elk Cloner[12] and Stoned[13].

b) *Macro Infectors*: This type of malware is written in a macro language embedded inside a software application like Microsoft Word and Excel. It generates a sequence of malicious actions when the infected application is opened. The Melisa[14] virus is a macro virus.

c) *Direct File Infectors*: Such malware infects the host file in the system as soon as it is executed. It overwrites the host file with its code, and sometimes it attaches itself to the host file during infection. Rugrat[15] is a direct file infector virus.

d) *Memory Resident Infectors*: Such malware copies itself to the system memory and infects any file that is executed by the system. Sometimes it attaches itself to even anti-malware programs. The Onehalf[16] virus is a memory resident malware.

e) *Multipartite*: Multipartite malware poses the characteristics of more than one type, and infects both files and boot sector. It infects and spreads in multiple ways. Ghostball[17] is the first multipartite virus discovered by Fridrik Skulason in 1989.

[12]http://virus.wikidot.com/elk-cloner
[13]http://virus.wikidot.com/stoned
[14]https://en.wikipedia.org/wiki/Melissa
[15]https://www.f-secure.com/v-descs/rugrat.shtml
[16]https://en.wikipedia.org/wiki/OneHalf
[17]https://en.wikipedia.org/wiki/Ghostball

Table 2.3: Malware Infectors

Infected Objects	**Examples**
Boot Sector Infectors	Elk Cloner
Macro Infectors	Melisa
Direct File Infectors	Rugrat
Memory Resident Infectors	Onehalf
Multipartite	Ghostball

Although there are different classes of malware, the classes are not mutually exclusive. In other words, some malware instances may successfully integrate malicious techniques used by other contemporary malware classes.

2.9 Sandboxes and Its Types

Sandboxes are controlled and isolated environments designed for the safe execution and analysis of potentially malicious software, providing security professionals and researchers with a crucial tool for understanding and combating malware. These environments aim to minimize the risks associated with handling unknown or suspicious code by containing their impact within a confined space. Sandboxes typically employ various techniques to simulate a real-world computing environment while maintaining strict controls to prevent the spread of malware beyond the controlled environment [15]. The three types of sandboxes that are commonly found are explained below.

- Hardware-Based Sandboxes: Hardware sandboxes utilize physical devices, often separate from the host system, to create isolated environments for malware analysis. This approach ensures a high level of isolation, as the hardware sandbox is distinct from the actual computing infrastructure. However, it may involve more logistical challenges and expenses.
- Software-Based Sandboxes: Software sandboxes are virtualized environments created through software emulation or virtual machines. These sandboxes run on the host system and provide a controlled space for executing and observing potentially malicious code. They are more flexible and cost-effective than hardware-based alternatives but may be susceptible to certain evasion techniques.

- Cloud-Based Sandboxes: Cloud sandboxes leverage remote servers and resources to execute and analyze suspicious code. This approach allows for scalability and collaboration among researchers but may introduce latency and reliance on external networks. Cloud-based sandboxes are particularly useful for handling large-scale malware samples.

2.10 Honeynet System

A honeynet is a specialized network designed to emulate a real production network but is used solely for the purpose of observing and analyzing malicious activities. The primary goal of a honeynet is to attract and study attackers, understanding their tactics, techniques, and procedures without putting a legitimate network at risk [16].

2.10.1 Honeypots

Honeypots are essential components of a honeynet, designed to simulate vulnerable systems and services to attract attackers.

- Low-Interaction Honeypots: These emulators simulate vulnerable services and applications with limited functionality, providing basic information about an attacker's activities without exposing the system to significant risks.
- High-Interaction Honeypots: These are real systems or applications that are intentionally left vulnerable, allowing for in-depth analysis of attacker behavior. High-interaction honeypots provide a more realistic environment but come with greater risks.

2.10.2 Honeynet Architecture

The architecture of a honeynet involves a strategic setup of various components to effectively monitor and analyze malicious activities.

- Sensor Networks: Honeynets are often composed of a network of sensors strategically placed to monitor and capture malicious activities. These sensors can include both low and high-interaction honeypots.

- Data Capture and Logging: The honeynet system records and logs all activities within the network, capturing information about the attackers' methods, the malware used, and any other relevant details.

2.10.3 Deployment Strategies

Deployment strategies for honeynets vary depending on the specific goals and requirements of the deployment.

- Research Honeynets: These are set up by security researchers and organizations to study the global threat landscape. They are often large-scale and distributed, capturing a wide range of malicious activities.
- Production Honeynets: These are deployed within an organization's production network to detect and analyze internal threats. Production honeynets are valuable for understanding the risks and attack vectors specific to the organization.

2.11 Malware Analysis

To counter a network intrusion event, we require knowledge of malware instances and their behaviors. To acquire such information, we need to acquire and analyze the malware. Although, malware appears in many different forms, certain common techniques are used to analyse malware in order to discover the risks it poses and intention with which it was created. Malware analysis helps to identify the indicators of compromise, based on which the detection model can be built. There are four fundamental approaches to malware analysis: static, dynamic, automated and manual code reversing. This taxonomy of malware analysis presented in Figure 2-4.

2.11.1 Static Analysis

In static analysis, we analyse a program without actually executing it. Static properties of a binary program include its string signature, hash, byte n-gram

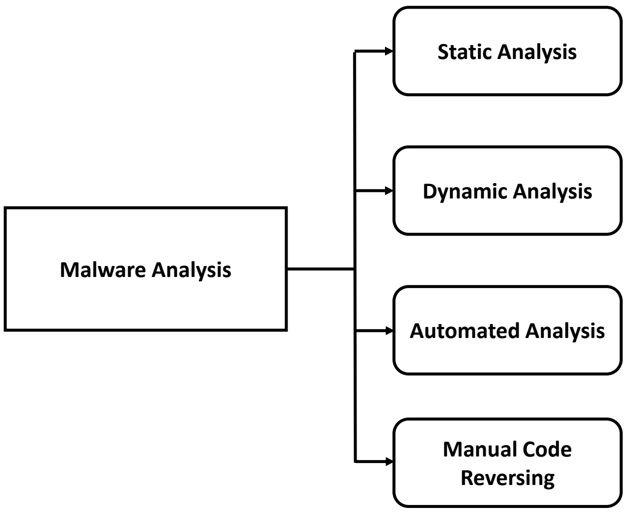

Figure 2-4: Types of malware analysis

sequences, packer signatures, and opcode frequency distributions. Such details can be obtained quickly because it does not require any execution. This is typically done by using tools such as disassemblers, debuggers, and decompilers to extract information from the file. However, static analysis can be easily evaded by malware authors using obfuscation techniques. As a result, properties that can be gained from the binary representation by static analysis are not extensive.

2.11.2 Dynamic Analysis

Dynamic analysis focusses on runtime behavior of executables. In dynamic analysis, we analyse a program as it executes. It is carried out by executing the binary program in an emulated environment so that the malicious program cannot infect a real system. The behavioral changes made to the file system, registry, processes and network communication are monitored and recorded for further investigation. Unlike static analysis, dynamic analysis allows one to observe the real functionality of malware binaries. Various techniques that can be applied to perform dynamic analysis include function call monitoring, function parameter analysis, information flow tracking, instruction traces and use of autostart extensibility points [17]. The control flow for dynamic analysis is presented in Figure 2-5.

Dynamic analysis is time consuming and it has difficulty in analyzing multipath malware. The rate of false alarm generated is also high compared to static analysis. In addition, the malware may behave differently in a virtual environment than in a real system. Some malware execution requires access to the external environment, and in such a case dynamic analysis cannot be performed.

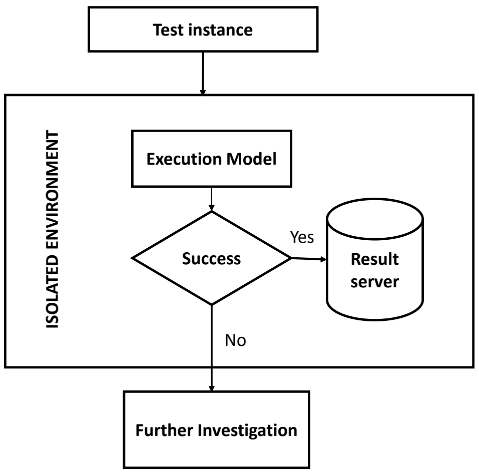

Figure 2-5: Flow diagram of dynamic analysis

2.11.3 Automated Analysis

The rapid growth of malware and its evolving nature have rendered manual malware analysis inefficient and time intensive. Hence, the focus of the security analysts has shifted towards automated malware analysis. Automated analysis tools aid in quick analysis and assessment of large scale threats that a malicious program poses. These tools detonate suspicious programs in a safe enivronment and generate detailed reports outlining the activities of such programs without user intervention. The automated analysis tools use static or dynamic analysis or combination of both to perfrom their tasks. However, automated analysis does not provide crucial insigths into suspicious programs as maunal analysis does. Several tools available for automated analysis are listed in table. The basic architecture of an automated malware analysis system is presented in Figure 2-6.

2.11.4 Manual Code Reversing

To understand the internal characteristics of a suspicious program, sometimes thorough analysis of the source code is essential. This can be achieved with the help of manual reverse engineering. It is carried out by taking apart the different components of the binary file without executing it, followed by examination of each component. The binary program is disassembled to obtain the assembly code. This assembly code is human readable and hence the job of the analysts

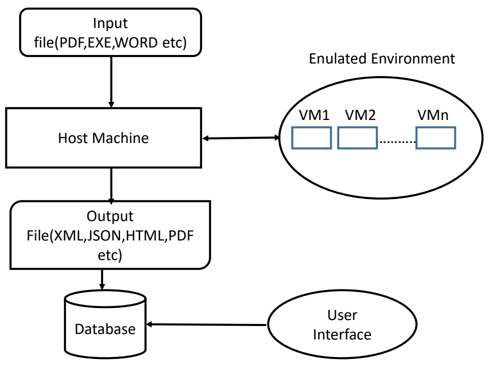

Figure 2-6: A generic architecture of an automated analysis system

becomes easier. The analyst can hypothesize what the program intends to do. The main limitation of manual analysis is that it is very time consuming. Sometimes it may take several days, weeks or months to obtain the complete source code of a single malware binary. For example Symantec took nearly six months to completely reverse engineer the stuxnet malware.

2.11.5 Recent advances in Malware Analysis

Recent advances in malware analysis have witnessed significant strides in the realm of cybersecurity, driven by the evolving sophistication of malicious threats. Machine learning and artificial intelligence have emerged as powerful tools, enabling more efficient and proactive detection of malware. Behavioral analysis, a cornerstone of modern malware analysis, has become increasingly nuanced, allowing security professionals to identify and understand the subtle patterns and tactics employed by advanced malware strains. Cloud-based malware analysis solutions offer scalability and real-time threat intelligence, enhancing the ability to handle large-scale and dynamic threats. Additionally, the integration of threat intelligence feeds, sandboxing, and automated incident response mechanisms has fortified the overall cybersecurity posture, enabling organizations to swiftly adapt to emerging threats. These collective advancements mark a pivotal shift towards a more proactive and adaptive approach in the ongoing battle against evolving and sophisticated malware threats.

2.12 Representation of Malware Data

Malware data can be represented in various ways for a malware defense system to effectively detect and mitigate cyber threats. These representations serve as input data for malware defense models and security measures. Malware data can be categorized into distinct formats, including tabular data, image data, graph data, sequence data, and text data.

- Tabular data: Tabular data represents malware features in a structured, spreadsheet-like format. Each row corresponds to a malware sample, and each column represents a specific attribute or feature of that sample such as file size, file type, API calls, or permission.
- Image data: Malware can be converted into image representations through a process known as binary visualization. In this technique, the binary code of the malware, which is typically a sequence of ones and zeros, is transformed into a pixel-based image. Each binary digit is mapped to a pixel's grey value or intensity value, creating a visual pattern. This conversion preserves the structural and sequential information of the malware, making it possible to analyze the malware's code and behavior in a more visual and interpretable format.
- Graph data: Malware can be represented as a control flow graph (CFG), which provides a visual and structural depiction of its execution path and behavior. In this representation, the various code blocks, functions, and instructions within the malware are represented as nodes in the graph. The edges between these nodes illustrate the order in which these elements are executed, creating a directed graph that reveals the flow of control and data within the malware.
- Sequence data: Malware activities can be represented as sequences of events or commands. These sequences may include system calls, API calls, or instructions executed by malware. Each sequence is a time-ordered list of actions.
- Text data: Text data represents malware in the form of textual code or strings. This may include disassembled code, scripts, or other textual representations of malware content.

2.13 Malware as a Service (MaaS) Model

Malware-as-a-Service (MaaS) is a modern form of cybercrime that allows anyone to initiate a cyberattack, irrespective of their technical expertise. Much like Software-as-a-Service (SaaS), where users pay to utilize software online instead of managing it themselves, MaaS provides easily accessible malware, tools, and infrastructure for a fee. In the MaaS model, individuals purchase access to malware and the required resources for carrying out cyberattacks [18][19]. Because MaaS providers frequently design their malicious software and tools to circumvent traditional security measures, this advancement simplifies the execution of complex and targeted attacks. Additionally, MaaS represents a lucrative opportunity for cybercriminals, posing a significant threat to both businesses and individuals due to the increased potential for successful, hard-to-detect cyberattacks. Consequently, it's crucial to acknowledge the MaaS threat and implement defensive measures. The Malware-as-a-Service (MaaS) ecosystem involves several key actors as shown in Figure 2-7, each playing a specific role in this underground cybercrime ecosystem. The common steps in the MaaS model are listed below.

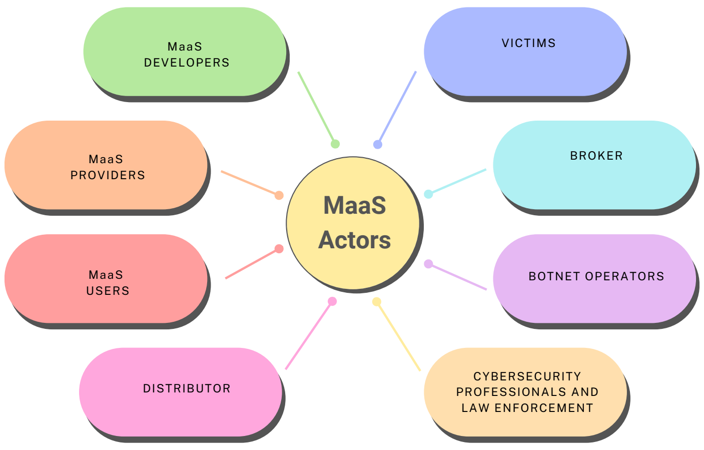

Figure 2-7: Actors involved in MaaS ecosystem

- Development of Malware: The first step in MaaS is the development of malicious software by cybercriminals or groups specializing in malware creation. This software can include various types of malware, such as ransomware, keyloggers, Trojans, and more. These malicious tools are designed to exploit vulnerabilities, compromise systems, or steal data.

- MaaS Platform Setup : Once the malware is developed, the creators establish a MaaS platform, often on the dark web or hidden parts of the internet. This platform serves as the hub for their criminal operations. Access to this platform is typically restricted, and potential users may need an invitation or some form of authentication to gain entry.

- Service Offerings: Once the MaaS platform is set up, the creators list their malicious tools and services for sale. These listings often include detailed descriptions, pricing, and the terms of use. Users can select from a menu of options, choosing the malware and services that suit their needs.
- Subscription or Payment: Users who want to utilize the malware and services must pay a fee. The payment structure can vary, with options like one-time payments, subscription models, or even pay-per-use arrangements.
- Access and Deployment: After paying for access, users gain access to the malware and related tools. They download or receive the malware, along with instructions on how to deploy it. This step might also involve downloading any additional tools or resources necessary for the attack, such as server infrastructure, botnets, or proxy services.
- Customization and Targeting: MaaS users can often customize the malware to suit their specific objectives. They may adjust settings, tailor the attack to specific targets, or modify the code to evade antivirus and intrusion detection systems.
- Delivery and Execution: With the malware ready, users deploy it through various means. This can include sending phishing emails, exploiting software vulnerabilities, or infecting websites. The goal is to compromise the target system or network.
- Monitoring and Data Theft: As the attack unfolds, users monitor its progress. They can steal data, access systems, or employ malware for their malicious purposes. They may also take steps to avoid detection and maintain control over the compromised systems.
- Payment for Services: In many cases, MaaS providers expect a share of the profits generated by their customers' criminal activities. This might be a percentage of any ransom paid in a ransomware attack, for instance. This revenue-sharing model incentivizes the MaaS provider to provide ongoing support and updates.
- Recurrence or Further Services: MaaS users can renew their subscriptions, purchase additional services, or engage in recurring attacks as needed. The

MaaS provider continues to offer support, updates, and new malware variants to keep their customers satisfied.

- Staying Anonymous: Throughout the process, MaaS users often take precautions to remain anonymous, such as using proxy servers, encrypted communication channels, and other techniques to conceal their identities.

2.14 Feature Selection

In machine learning and data science, a dataset comprises a set of data points, each of which is represented by a set of numbers or values. These numbers are called variables, attributes, or features of the data point. The cardinality of the feature set is also called the dimensionality of the dataset. The dimensionality of data plays an important role in building machine learning models. When the dimensionality of data is very high, a set of issues known as the curse of dimensionality arises, which affects the performance of the learning model. In addition, the high-dimensional feature space makes the learning model overfitted and also increases the memory requirements and computational cost. To address these issues, feature selection is used as a data preprocessing approach to reduce dimensionality. All the features present in the real world are not useful; the feature selection strategy helps to select a relevant feature subset using some evaluation criteria. Feature selection has several advantages like improving the performance and generalization ability of the learning model, the computational efficiency, and decreasing the memory requirements. Feature selection approaches are classified into four categories: filter approach, wrapper approach, embedded approach, and hybrid approach.

2.14.1 Filter Approach

With this approach, a subset of features is chosen without the need for a learning process. It is applied to numerous datasets with a large number of features. Feature selection techniques based on filters are quicker than those based on wrappers.

2.14.2 Wrapper Approach

This approach uses a learning algorithm to evaluate the accuracy produced by the use of the selected features in classification. Wrapper methods can give high classification accuracy for particular classifiers, but generally, they have high computational complexity.

2.14.3 Embedded Approach

The embedded approach integrates feature selection directly into the process of model training. Unlike filter and wrapper methods, which are separate preprocessing steps, embedded methods incorporate the feature selection process as part of the algorithm's learning phase. This integration allows the model to simultaneously select the most relevant features while being trained, leading to potentially better performance and efficiency.

2.14.4 Hybrid Approach

This approach is a combination of both filter and wrapper-based methods. The filter approach selects a candidate feature set from the original feature set, and the candidate feature set is refined by the wrapper approach. It exploits the advantages of these two approaches.

2.15 Machine Learning

Machine Learning (ML) is a subfield of artificial intelligence (AI) that focuses on developing algorithms and models that enable computers to learn and make decisions without explicit programming. This is achieved by training algorithms to identify patterns and improve their performance over time. Machine learning encompasses three main categories: supervised learning, where models are trained on labeled data; unsupervised learning, where models identify patterns in unlabeled data; and reinforcement learning, where models learn through trial and error. Machine learning encompasses three main categories: supervised learning, where models are trained on labeled data; unsupervised learning, where models identify patterns in unlabeled data; and reinforcement learning, where models learn through trial and error.

2.15.1 Supervised Learning

Supervised machine learning is a type of machine learning algorithm that learns from labeled data. In supervised learning, the algorithm is given a set of training data that consists of input data and corresponding output labels. The algorithm then learns to map the input data to the output labels. Once the algorithm has been trained, it can be used to make predictions on new, unseen data.

2.15.2 Unsupervised Learning

Unsupervised machine learning is a type of machine learning algorithm that learns from unlabeled data. In unsupervised learning, the algorithm is given a set of training data that does not have corresponding output labels. The algorithm then learns to identify patterns and structures in the data. Once the algorithm has been trained, it can be used to perform a variety of tasks. The primary goal is to allow algorithms to identify patterns within the data without predefined categories. Clustering and dimensionality reduction are common techniques in unsupervised learning, where algorithms autonomously find similarities or reduce the complexity of the data.

2.15.3 Reinforcement Learning

Reinforcement machine learning is a type of machine learning algorithm that learns through trial and error. In reinforcement learning, the algorithm is placed in an environment and interacts with it by taking actions. The algorithm receives feedback from the environment in the form of rewards and punishments. The goal of the algorithm is to learn a policy that maximizes the cumulative reward. Reinforcement machine learning is often used in situations where there is no labeled data or where the environment is too complex for supervised learning.

2.15.4 Ensemble Learning

Ensemble machine learning is a powerful technique in the field of machine learning that combines multiple models to achieve better predictive performance than any individual model alone. It is based on the principle that combining the predictions of multiple models can help reduce the overall error rate. Ensemble methods can be used for both supervised and unsupervised learning tasks. There are three main types of ensemble learning methods: Bagging, Boosting, and Stacking.

2.16 Deep Learning

Deep learning is a subset of machine learning that uses artificial neural networks (ANNs) with multiple layers to learn from data. ANNs are inspired by the structure and function of the human brain. They are composed of layers of interconnected nodes called neurons. Each neuron receives input from the previous layer neurons or the input layer, applies an activation function to the input, and produces an output. The output of one neuron becomes the input to other neurons in the next layer of the network, and this process continues until the final layer produces the output of the network. Deep learning is particularly well-suited for tasks that involve learning complex patterns from data, such as image and speech recognition, natural language processing, and machine translation.

2.16.1 Aritifical Neural Networks

An ANN is based on a collection of connected perceptrons. The ANN is organized as layers of neurons. The outputs of neurons in a layer serve as inputs to other neurons in the next layer. The most common layer type is the fully-connected layer in which neurons between two adjacent layers are fully pairwise connected, but neurons within a single layer share no connections. The basic architecture of an Artificial Neural Network is shown in figure 2-8. In the figure, the first layer of neurons is called the input layer and the last layer with a single neuron is called the output layer. The middle layer is called a hidden layer, since the neurons in this layer are neither inputs nor outputs. An artificial neural network can have multiple hidden layers. Such multiple layer networks, with only feedforward connections, are called Multilayer Perceptrons (MLPs). In such an artifical neural network, the output from one layer is used as input to the next layer. Such networks are

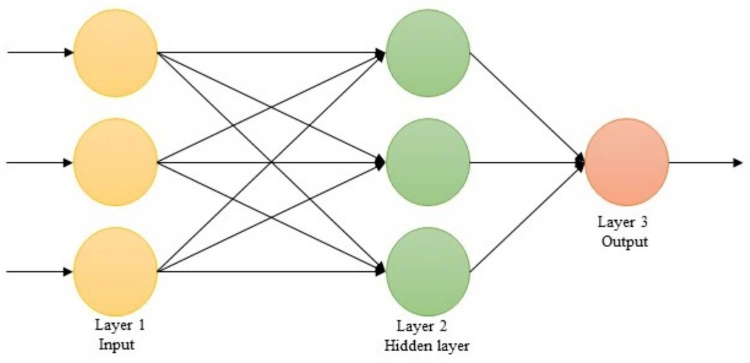

Figure 2-8: Artiicial Neural Network

also called as feedforward neural networks. This means there are no loops in the network - information is always fed forward, never fed back. However, there are other models of artificial neural networks in which feedback loops are possible. These models are called recurrent neural networks.

2.16.2 Convolutional Neural Network (CNN)

A Convolutional Neural Network is a type of artificial neural network that uses a special architecture which is particularly well-adapted to classify images. It can also be thought as a regularized version of the multilayer perceptron which prevents the overfitting problem. Three main types of layers are used in CNN architectures: Convolutional layers, Pooling layers, and Fully-Connected layers. Convolutional neural networks use three fundamental ideas: local receptive fields, shared weights, and pooling

2.16.2.1 Local receptive field

It is the region of the input space connected to a hidden neuron in the convolutional layer. It is a little window on the input pixels. As in figure 2-9, the input neurons are the pixel intensities of an input image, and on the right is a hidden neuron in the first hidden layer. Each neuron is connected to only a region of the input layer; thus region in the input image is called the local receptive field for the hidden neuron. In order to cover the entire input space, we slide the local receptive field across the entire input image horizontally as well as vertically. For each local receptive field, there is a different hidden neuron in the first hidden layer. To slide the local receptive field, different stride lengths are used. If stride length is one, the local receptive field is moved by one pixel at a time.

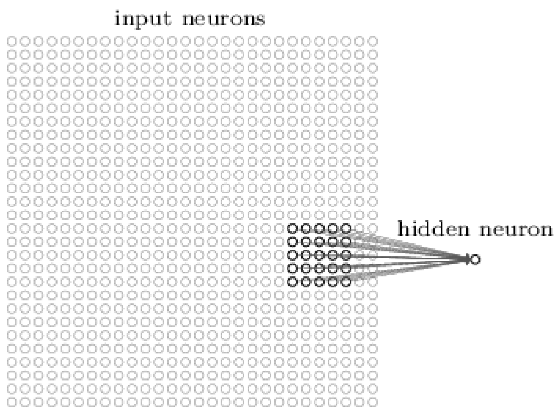

Figure 2-9: The local receptive field for a neuron

2.16.2.2 Shared weights and biases

In a CNN, the same weights and bias are used for each of the hidden neurons in a certain layer. This means that all neurons in the first hidden layer detect exactly the same feature, just at different locations in the input image. The shared weights and bias are often said to define a kernel or filter. The repeated use of a kernel reduces the number of weights that must be learned, which in turn reduces model training time and cost. It also makes feature search insensitive to feature location in the image. Several convolutional filters or kernel can be used in a convolutional layer, not just one.

2.16.2.3 Pooling layers

A Convolutional neural network also contains pooling layers. A pooling layer is usually used immediately after a convolutional layer. It simplifies the information in the output from the convolutional layer. A pooling layer takes the output from the convolutional layer and prepares a condensed version of the output. One common procedure for pooling is known as max-pooling. In max-pooling, a pooling unit simply outputs the maximum value among all the activations within its purview. In figure 2-10, a pooling unit outputs the maximum activation in the 2×2 input region. There are other techniques available for pooling like l2 pooling or average pooling.

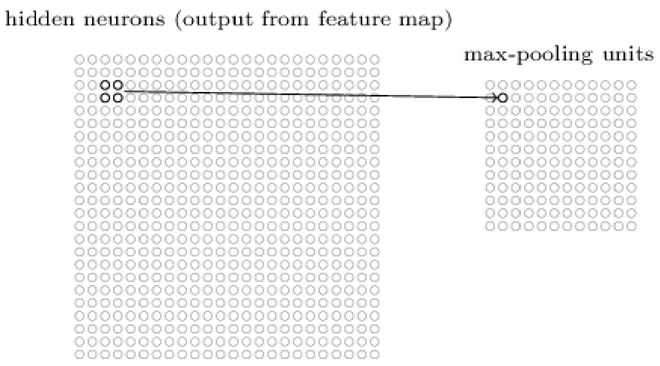

Figure 2-10: Illustration of max pooling

2.16.3 Generative Adversarial Networks

Generative Adversarial Networks [20] (GANs) are a type of machine learning model that can be used to generate new data, such as images, videos, or text. GANs are composed of two neural networks: a generator and a discriminator. The generator's job is to generate new data, and the discriminator's job is to determine whether or not the data is real or fake. GANs work by pitting the generator and discriminator against each other in a zero-sum game. The generator is trained by feeding it random noise and then evaluating the output using the discriminator. If the discriminator correctly identifies the output as synthetic, then the generator is rewarded. If the discriminator mistakenly identifies the output as real, then the generator is punished. In this way, the generator is encouraged to generate more realistic data. The discriminator is trained by feeding it real data and synthetic data and then evaluating its output. If the discriminator correctly identifies the input as real, then it is rewarded. If the discriminator mistakenly identifies the input as synthetic, then it is punished. In this way, the discriminator is encouraged to become better at distinguishing between real and synthetic data.

2.16.4 Graph Neural Networks

Graph neural networks (GNNs) are a type of machine learning model that can learn from and make predictions on graph data. Graphs are a powerful way to represent data with relationships between different entities, such as social networks, road networks, and molecular structures. GNNs can be used to solve a wide range of tasks, including node classification, edge prediction, graph classification, and graph generation.

GNNs work by iteratively aggregating information from neighboring nodes in a graph. In each iteration, each node updates its representation based on the representations of its neighbors. It allow GNNs to learn global patterns in the graph, even from local information. This process is often performed through a series of layers, similar to traditional neural networks, where each layer refines the node representations, enabling the network to capture complex patterns and dependencies in the graph structure.

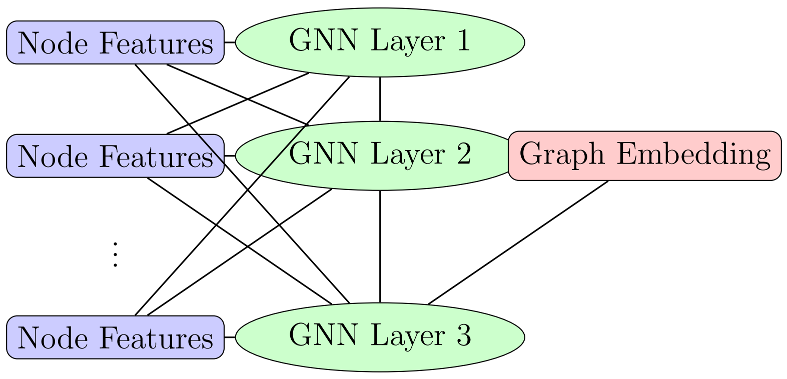

Figure 2-11: Graph Neural Network Architecture

2.17 Evaluation Metrics

Evaluation metrics are crucial tools in assessing the performance of models and algorithms across various domains, including machine learning, data science, and information retrieval. Different tasks and objectives require different metrics to provide meaningful insights into the effectiveness of a model.

2.17.1 Accuracy

Accuracy is a commonly used metric in classification, representing the ratio of correctly predicted instances to the total instances.

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Instances}} \tag{2.1}$$

2.17.2 Precision

Precision measures the ratio of true positive predictions to the total predicted positives, emphasizing the accuracy of positive predictions.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2.2}$$

2.17.3 Recall

Recall, also known as sensitivity or true positive rate, measures the ratio of true positive predictions to the total actual positives, emphasizing the model's ability to capture all relevant instances.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2.3}$$

2.17.4 F1 Score

The F1 score is the harmonic mean of precision and recall, providing a balance between the two metrics.

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.4}$$

2.17.5 AUC-ROC

The Area Under the Receiver Operating Characteristic Curve (AUC-ROC) is a performance metric used for binary classification models. It represents the area under the ROC curve, which illustrates the trade-off between the true positive rate (sensitivity) and the false positive rate (1-specificity).

$$\text{AUC-ROC} = \int_0^1 \text{TPR}(\text{FPR}^{-1}(t))\, dt \tag{2.5}$$

where TPR is the true positive rate (sensitivity) and FPR is the false positive rate (1-specificity).