VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE
CYBERSECURITY LABORATORY

Masters Thesis

Evaluation of vulnerability inventories using a classification algorithm

Pažeidžiamumo aprašų vertinimas naudojant klasifikavimo algoritmą

Done by:
Arnas Čižikovas

Supervisor:
dr. Linas Bukauskas

Vilnius
2025

Contents

Abstract

Cybersecurity vulnerabilities pose a significant threat to organisations, so it is essential to identify and prioritise them to mitigate potential risks. Newly emerging breaches are classified slowly, yet they keep increasing every year. Treat assessment is an extraordinarily long and complex process, and it can take months to start taking adequate measures to protect against cyber attacks.

This study compares classification algorithms for vulnerability inventories using the Common Vulnerabilities and Exposures (CVE), the Common Weakness Enumeration (CWE), and the Common Vulnerability Scoring System (CVSS). We compare and contrast the CVEs, CWEs, and CVSSs to evaluate the performance of different classification algorithms in classifying CVEs by their CWE mapping and their Common Vulnerability Scoring System (CVSS) vectors. The research provides a broader overview of the fundamental concepts of CVE, CWE, and CVSS, their differences and applicability, how they relate, and how they are interrelated. The analysis aims to identify the optimal methods for developing a new classification algorithm faster and more accurately than existing algorithms and methodologies. This study contributes to developing automated vulnerability assessment and prioritisation methods to help organisations improve their cybersecurity posture.

Key words: CVE, CWE, CVSS, Random Forest Classifier, SVC, Logistic Regression Classifier.

Santrauka

Pažeidžiamumo aprašų vertinimas naudojant klasifikavimo algoritmą

Kibernetinio saugumo pažeidžiamosios vietos kelia didelę grėsmę organizacijoms, todėl labai svarbu jas nustatyti ir nustatyti prioritetus, kad būtų sumažinta galima rizika. Naujai atsirandantys pažeidimai klasifikuojami lėtai, tačiau jų kasmet vis daugėja. Apsaugos vertinimas yra nepaprastai ilgas ir sudėtingas procesas, todėl gali prireikti kelių mėnesių, kad būtų galima pradėti imtis tinkamų priemonių apsisaugoti nuo kibernetinių išpuolių.

Šiame tyrime lyginami pažeidžiamumų aprašų klasifikavimo algoritmai, naudojami pagal Bendrą pažeidžiamumų ir rizikų sąrašą (angl. Common Vulnerabilities and Exposures, CVE), Bendrą silpnybių sąrašą (angl. Common Weakness Enumeration, CWE) ir Bendrą pažeidžiamumų vertinimo sistemą (angl. Common Vulnerability Scoring System, CVSS). Lyginame ir sugretiname CVE, CWE ir CVSS, kad įvertintume skirtingų klasifikavimo algoritmų efektyvumą klasifikuojant CVE pagal jų CWE atvaizdavimą ir jų Bendrosios pažeidžiamumų vertinimo sistemos (CVSS) vektorius. Tyrime plačiau apžvelgiamos pagrindinės CVE, CWE ir CVSS sąvokos, jų skirtumai ir pritaikomumas, jų sąsajos ir tarpusavio ryšiai. Analizės tikslas - nustatyti optimalius metodus, leidžiančius greičiau ir tiksliau identifikuoti aprašus, nei esami algoritmai ir metodikos kuriant naują klasifikavimo algoritmą. Šis tyrimas prisideda prie automatizuotų pažeidžiamumų vertinimo ir prioritetų nustatymo metodų, padedančių organizacijoms gerinti kibernetinio saugumo būklę, kūrimo.

Raktiniai žodžiai: CVE, CWE, CVSS, Atsitiktinių miškų klasifikavimas, Palaikomųjų vektorių klasifikatorius, Logistinės regresijos klasifikatorius.

Introduction

Cybersecurity vulnerabilities represent critical weaknesses in software or hardware systems that malicious actors can exploit to gain unauthorized access, disrupt services, or inflict damage. The identification and prioritization of these vulnerabilities are paramount for organizations aiming to manage cybersecurity risks effectively and safeguard their digital assets. However, traditional methods of conducting manual vulnerability assessments can be labor-intensive, time-consuming, and susceptible to human error, which can lead to overlooked vulnerabilities and increased risk exposure

This study aims to enhance the vulnerability assessment process by comparing existing vulnerability descriptions with their corresponding Common Weakness Enumeration (CWE) references and Common Vulnerability Scoring System (CVSS) vectors. By doing so, we seek to provide potential values for newly created Common Vulnerabilities and Exposures (CVE) cases. Our approach categorizes vulnerabilities based on their CVSS vectors, utilizing CVE and CWE as the primary sources of information to ensure a comprehensive analysis.

The Common Vulnerability Scoring System (CVSS) is a widely recognized standard for measuring the severity of vulnerabilities. It employs a scoring system that takes into account various factors, including the attack vector, the complexity of the attack, and the potential impact on the system. The Common Vulnerabilities and Exposures (CVE) database lists publicly disclosed cybersecurity vulnerabilities, serving as a critical resource for organizations to stay informed about known threats. In parallel, the Common Weakness Enumeration (CWE) is a community-driven standard that aids in identifying and categorizing software weaknesses, allowing organizations to prioritize vulnerabilities even if they have not yet been publicly disclosed.

Threat modeling is an essential component of cybersecurity that involves identifying potential threats and vulnerabilities within a system and assessing their likelihood and potential impact. Attack graphs are valuable tools in this context, as they illustrate possible attack paths within a system, helping security professionals identify the most vulnerable components and prioritize their remediation efforts.

In recent years, machine learning techniques have emerged as powerful tools for analyzing vulnerability data. These techniques apply algorithms to learn patterns and relationships within the data, enabling the prediction of the likelihood and impact of new vulnerabilities. However, despite their potential, these methods also face limitations, including subjectivity in feature selection, complexity in model training, and challenges related to scalability when applied to large datasets.

In this study, we will conduct a thorough analysis of classification algorithms designed to assess vulnerability inventories based on criteria such as severity, exploitability, and potential impact. We will evaluate the effectiveness of these algorithms using a comprehensive dataset of known vulnerabilities, comparing our results with those obtained from existing methods. By doing so, we aim to contribute to the ongoing efforts to improve vulnerability management practices and enhance the overall security posture of organizations in an increasingly complex threat landscape.

The tasks to achieve the aim are:

- Analyze the state of the art in the vulnerability definition inventories and classification methods;
- Understand CWE, CVE and CVSS differences;
- Compare classification methods for CVSS and CWE;

- Evaluate vulnerability inventory using a classification algorithm.

The Common Vulnerabilities and Exposures (CVE) system provides a standardized method for documenting publicly known information security vulnerabilities and exposures, serving as a comprehensive database of cyber threats. Each CVE entry is assigned a unique identifier and includes attributes such as a description, potential impact, affected software or hardware, and links to remediation resources. This structured information helps organizations assess risks and implement appropriate measures to mitigate vulnerabilities.

However, there is currently no efficient method for automatically classifying new CVE records into specific attack categories. The existing manual classification process is time-consuming and prone to human error, making it challenging for organizations to prioritize their response to emerging threats effectively. This gap can lead to critical vulnerabilities being overlooked.

To address these challenges, there is a need for advanced methodologies that leverage machine learning and natural language processing to automate the classification of CVE records. Such solutions would enhance vulnerability management, improve response times, and strengthen overall cybersecurity posture by providing a consistent framework for understanding and addressing vulnerabilities in a complex cyber landscape.

1 Related work

1.1 Meaning of CVE

Common Vulnerabilities and Exposures, commonly known as CVE, is a publicly available list of security vulnerabilities in computer systems. This list aids IT teams in prioritising security, sharing information, and proactively addressing problem areas. Thus, it enhances the security of systems and networks and helps prevent harmful cyberattacks. [1]

CVE entries include a description, an identification number, and a public link. Each CVE details a specific vulnerability or impact. According to the CVE website [4], a vulnerability is a software code error that allows hackers direct access to a system or network. Conversely, an exposure refers to an error that gives an attacker indirect access. [1]

In general, the CVE project creates a system of vulnerabilities and organisation. To make a list of CVEs, several steps must be followed. The first step in creating a CVE list is identifying the vulnerability or impact. In the second step, the computer network attack (CNN) [13] assigns a CVE identification number to the vulnerability. Afterwards, the CNA records a description of the problem and provides links. In the final step, the completed CVE entry is added to the CVE list and published on the CVE website. [1]

CVE provides a single and unique identifier for each individual vulnerability. In general, CVE is more like a dictionary than a database. The description of each entry should be brief and shall not include technical data, information on specific effects or corrective information.

To summarise and accurately identify the concept of CVE, it is best to create a list of key components and advantages of the CVE:

- Single naming for a single vulnerability or exposure to be described;
- Single standardised description of each vulnerability or exposure;
- It is more of a dictionary than a database;
- A way to unite different databases and tools in the same language;
- A way to ensure interoperability and better security;
- The basis for evaluating tools and databases.

The main purpose of the catalogue is to standardise the method of identifying each known vulnerability or exposure. This is important because standard identifiers allow security administrators to quickly obtain technical information about a specific threat from many CVE-compatible information sources. [1]

Information technology and cyber-security professionals can use the CVE and its records to understand, prioritise, and address vulnerabilities in their organisations. They can also use the CVE to engage in useful discussions with colleagues and coordinate their vulnerability mitigation efforts.

1.2 A brief history of CVE

The CVE standard [12] was launched in 1999 when most cyber-security measures used their databases with the names of their security vulnerabilities. At that time, there were no significant differences

between products and no easy way to determine when different databases indicated the same problem. The consequences were possible gaps in the security coverage and effective interactions between different databases. In addition, the seller of each instrument used different indicators to indicate the names of the vulnerabilities detected, which means there was no standardised basis for evaluating the measures. [12]

The CVE common and standardised identifiers provided an opportunity to solve these problems. CVE is now a tool and a standard for naming some vulnerabilities. CVE identifiers provide reference points for data exchange so that cyber-security products and services can communicate with each other. They also provide a basis for assessing the coverage of tools and services so that users can determine which tools are most appropriate, effective, and suitable for their organisation's needs. In short, CVE-compatible products and services provide better coverage, easier interaction, and better quality and security. [12].

1.3 CVE usability and performance

As an industry standard, CVE identifiers are used in many cyber-security products and services worldwide. These CVE products include vulnerability databases, security tips and archives, vulnerability reporting, evaluation and remediation products, intrusion detection, management, monitoring and response products, incident management products, data and event correlation products, firewalls, patch management products, policy compliance products and security information management tools.

When a vulnerability is detected, the standardised life cycle [23] of the CVE must be passed before it is published. Each step and sequence are shown in the Figure 1.

- **Discover**: The process begins when a person or organisation detects a vulnerability;
- **Inform**: Vulnerability discovered person or organisation submits a report to the CVE numbering authorities;
- **Fill out an application**: CVE numbering authorities (CNA) issues a vulnerability ID;
- **Reserve**: The ID is reserved for that specific vulnerability and is used in the early stages of the CVE assessment and in all relevant cross-country relationships;
- **Submit**: The CVE numbering authorities evaluate the vulnerability documentation provided, which should contain all the information necessary to prove the existence, root cause, threat type, and impact of the vulnerability or impact;
- **Publish**: After checking all the information in the documents, the CNA publishes the CVE, thus making it publicly available.

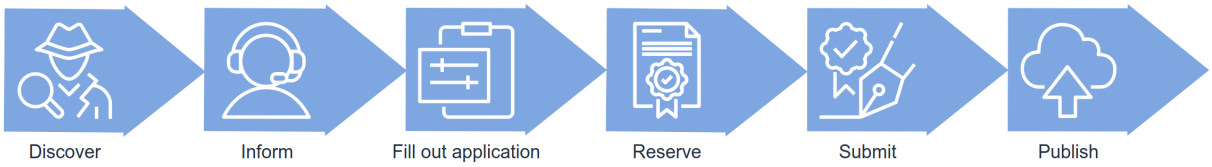

Figure 1. CVE lifecycle

1.4 CVE compatibility

The basic premise of the CVE list is that vulnerability or impact must be identified in one name. A CVE-compatible product or service must understand the CVE vulnerability names and allow users to interact with the product/service under these CVE names. Support for the following CVE names is at the heart of the CVE compatibility concept. CVE-compatible tools, websites, databases or services must use CVE names in such a way that users can associate their information with other repositories, tools, and services that also use CVE names, such as shown in Figure 2 [23].

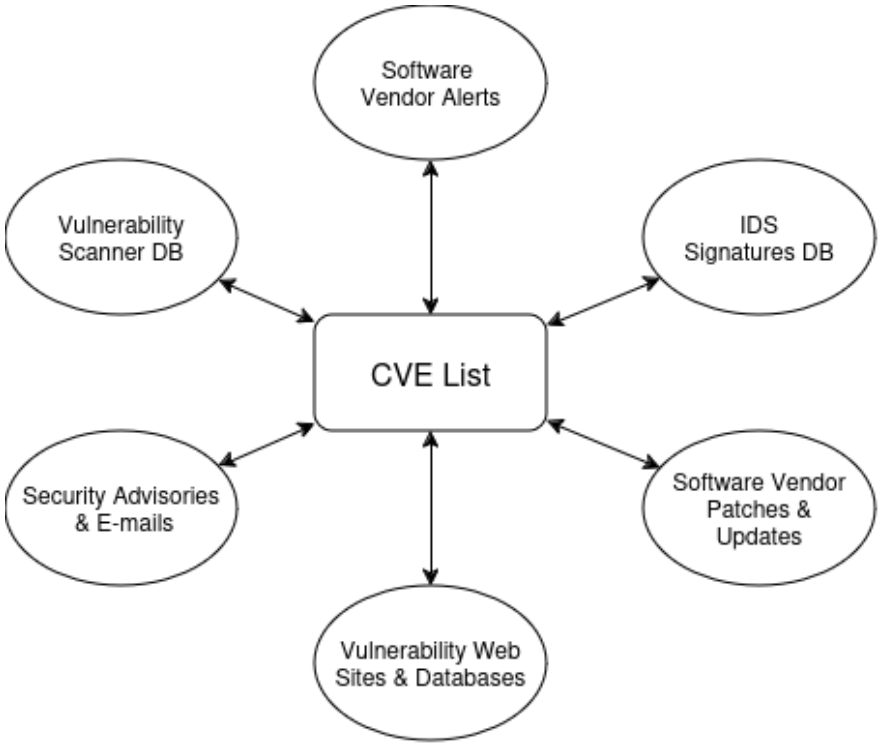

Figure 2. Cross linking

Integrating vulnerability services, databases, websites, and tools that include CVE titles will provide the organisation with more comprehensive and effective coverage of security issues. For example, a vulnerability scanning tool report using CVE names will allow an organisation to quickly and accurately find patch information in one or more CVE-compliant databases and websites [7]. It can also pinpoint which vulnerabilities and exposures are covered by each CVE-related compliant tool or service, as the CVE list provides a basis for comparison. Identifying which CVE entries apply to platforms, operating systems, and commercial software packages allows the organisation to compare this subset of the CVE list with the coverage of any specific tool or service. Network and security logs already mention that support for CVE names is a desirable feature. The National Institute of Science and Technology (NIST) has issued a recommendation to all federal government agencies and services to use CVE-compatible products and services whenever possible [16] [15].

CVE compatibility [16] includes four main requirements:

- Customers can use CVE names to learn about scope, content or coverage and then get any relevant information;
- Clients can receive output that includes all relevant CVE names;

- The owner of an element shall make a good faith effort to ensure that the mapping from its own elements to the CVE names remains accurate as the CVE list and the compatible element are updated over time;

- The standard documentation shall provide a description and details of CVE compatibility, how customers can access the CVE-related elements of their instrument, database, website, or service features.

Vendors are generally given the flexibility to implement the requirements in different ways. Users may determine which features or implementations best meet their needs [8].

1.5 Comparison between CWE, CVE and CVSS

To protect your digital infrastructure from threats, it is important to be familiar with security standards such as CWE, CVE, and CVSS. These terms are developed and maintained by MITRE, a non-profit organisation that manages U.S. government-sponsored research and development (R&D) centres.

Several methods have been proposed for vulnerability assessment and prioritisation. CWE is often used with CVSS to provide a more comprehensive view of vulnerabilities. CVSS provides a common language for describing vulnerabilities and helps organisations prioritise their efforts to address the most critical ones. CWE provides a common language for describing software weaknesses and helps organisations prioritise their efforts to address the most critical ones. CWE has several categories: input validation, authentication, and authorisation.

Threat modelling can be performed using various methods, such as attack trees, graphs, and misuse cases. Threat modelling can help organisations identify potential threats and vulnerabilities and prioritise their efforts to address them.

Attack graphs can be generated using various methods, such as model checking, graph theory, and machine learning. Attack graphs can help organisations identify potential attack paths and prioritise their efforts to address them.

Machine learning techniques can be supervised, unsupervised, or semi-supervised. Machine learning techniques can handle large datasets and complex relationships between features. However, they may be more difficult to interpret and require more computational resources.

CWE stands for "Common Weakness Enumeration," a community-developed tool that has become a standard for describing and categorizing weaknesses that can lead to vulnerabilities. These vulnerabilities may include system misconfiguration and code bugs that have not yet been exploited. CWE facilitates the identification and remediation of software vulnerabilities, design flaws, programming errors, and configuration problems. Each vulnerability on the CWE list is assigned a unique 3–4 digit identifier to each vulnerability and provides detailed descriptions, examples, and recommendations on how to reduce or avoid the vulnerability. [11].

The following are some examples of CWEs related to web-oriented systems that can extend the external attack surface. [11].

- **CWE-200: Disclosure of information**: this deficiency occurs when personal information, systems data, network configurations, and other sensitive information is inadvertently disclosed or disclosed in error messages, logs, or other system outputs, resulting in potential hackers gain valuable information that can be used to exploit the system.

- **CWE-326: Insufficient encryption strength**: CWE-326 is associated with weak encryption algorithms or insufficient key length on external resources.
- **CWE-346: Origin CWE-346: Origin Validation Error**: This error can occur when the network validation is not properly validated.
- **CWE-434: Unrestricted upload of a dangerous file type**: This vulnerability occurs when an application allows users to upload files without proper authentication, which could lead to malicious code, file overwriting, or other security vulnerabilities.
- **CWE-601: URL redirection to an untrusted website ("Open Redirect")**: This vulnerability weakness occurs when an application redirects users to another website or URL without confirming the target. This can lead to phishing attacks, malware downloads or other malicious activities.

The above are just a few examples. There are currently more than a thousand CWEs.

CVE, abbreviated as "Common Vulnerabilities and Exposures," is also a standardised method for identifying and monitoring publicly known vulnerabilities in systems. However, unlike CWEs, which focus on vulnerability or deficiency at a higher level, CVE analyses vulnerabilities in the context of a specific product or system. Each CVE has a unique CVE ID of 8 or more digits and follows the syntax CVE-YYYYY-NNNN, where 'YYYYY' stands for the year and 'NNNN' represents freely chosen digits. CVE is a generic reference to a specific system, platform, or technology-level vulnerability. Hence, security practitioners, researchers, and organisations make it easier for security organisations to collaborate and share much-needed information. [1].

Here are some examples of CVEs.

- **CVE-2021-34523**: This vulnerability in Microsoft Exchange Server can be exploited to access the victim's SSL/TLS keys, allowing attackers to intercept and modify encrypted data during transmission secretly.
- **CVE-2020-1350**: This CVE, known as "SIGRed", affects Windows DNS servers. It allows hackers to remotely execute their own code on the server to compromise the DNS service's reliability and security.
- **CVE-2018-6789**: This vulnerability affects Exim mail servers and is called the "Exim Mail Server" vulnerability. It allows remote attackers to execute commands on the server, potentially gaining unauthorised access or malicious actions.

There are currently more than 240,000 CVEs, and the list continues to grow as new ones are discovered and added vulnerabilities.

The Common Vulnerability Scoring System (CVSS) is a standardised framework for assessing the severity and impact of vulnerabilities in cyber security. It classifies vulnerabilities as digital scores based on indicators such as the impact on the availability and integrity of the affected system, the ease of access, and the ease with which attackers can exploit it. CVSS scores range from 0 to 10, with 10 being the most severe. This The system is useful for the cybersecurity community and security teams [9] [5]:

1. They can prioritise responses to vulnerability.

2. Make vulnerability assessments objective and consistent.

3. Enable effective communication and cooperation between different stakeholders.

From 13 July 2022, NVD will no longer generate vector strings of qualitative severity ratings or CVSS v2 severity scores. The existing CVSS v2 information will remain in the database, but the NVD will no longer actively populate the CVSS v2 for the newly created CVEs. The change has been issued due to the CISA policy that after the full transition from CVSS v2, it will be fully dependent on the NVD data [14].

The NVD analysts will continue to use the reference information provided with the CVE and any publicly available information to reference tags for CWE, CVSS v3.1 and CPE Applicability statements [14].

CVSS v2 has been included in the NVD since 2007, and v3 and v3.1 have been included since their release in 2015 and 2019, respectively. Led by FIRST's CVSS-SIG team, CVSS v4 development is underway. NVD expects to introduce CVSS v4 components from 2023 [14].

When comparing CWE with CVE and CVSS, it is important to note that although they are vulnerability management tools and can appear similar, they have different objectives and purposes. The table below highlights some key differences between CWE, CVS and CVSS [5] Table 1.

Table 1. Differences between CWE, CVS and CVSS

Deadline	**Purpose**	**Focus**	**Representations**
Overall vulnerabilities (CWE)	To provide a standardised method for identifying and classifying weaknesses.	To identify weaknesses or vulnerabilities. Identification of vulnerabilities.	3-4 digit ID (e.g., CWE-200)
General Vulnerabilities and risk factors (CVE)	Facilitate communication, tracking and referencing vulnerabilities of each product or system level.	For specific vulnerabilities that have been publicly disclosed. Identification of vulnerabilities that have been identified.	IDs with 8 or more digits (e.g., CVE-2021-34523)
Common Vulnerabilities Assessment Framework (CVSS)	To provide a standardised vulnerability assessment method to enable accurate prioritisation.	Quantitative vulnerability severity assessment includes exploitation potential, impact, and exploitation complexity.	Score range 0 to 10

They may vary, but CWE, CVE, and CVSS all contribute to better vulnerability management, secure coding practices, risk prioritisation, and information sharing, ultimately improving overall performance.

The security posture of software and systems supports the overall cybersecurity community efforts.

1.6 CVSS calculation metrics and values

In subsection 1.5, there was a slight hint of CVSS and its different versions. The entire meaning of the CVSS score is to determine the criticality of the vulnerability. As of the writing of this thesis, CVSS v2 is no longer applicable to the new CVE appearances since version 2 was discontinued in 2022, and it is no longer used for further research. Version 3.1 is the current standard applicable for the new CVE entries. Additionally, version 4 has appeared since 2023 and provided some changes that overhauled the calculation of the metrics.

The current standard CVSS v3.1 scoring system consists of Base, Temporal, and Environmental metrics, Scoring range, and multiple metric categories such as attack vector (AV) or scope (S). The new metric system added sub-metrics to the base metrics to improve the contextual scoring. Introduced new impact metrics for refined confidentiality, integrity and availability impact assessments. Incorporated optional and supplemental metrics designed to support automation and improve the scoring accuracy for diverse contexts. Refined attack complexity and added more nuance to the likelihood of success of an attack, adapting to more complex environments. It was also designed with modern technology in mind, offering stronger support for IoT, cloud, and distributed networks. Both versions of CVSS calculate three main metrics: basic, temporal, and environmental.

1.6.1 Base metrics

The Base Metrics represent the core characteristics of a vulnerability that remain constant over time and across environments. They are used to calculate the Base Score, which reflects the severity of the vulnerability, independent of any specific organisation's environment. The Base Metrics are divided into two main categories Exploitability and Impact Metrics.

The Exploitability Metrics describe the characteristics of the vulnerable thing, formally called the vulnerable component. When scoring Base metrics, it should be assumed that the attacker has advanced knowledge of the target system's weaknesses, including general configuration and default defence mechanisms.

Attack Vector (AV) described in Table 2 provides the context by which vulnerability exploitation is possible. This metric value will be larger the more remote (logically, and physically) an attacker can be in exploiting the vulnerable component

Metric Value	**Description**	**Numerical Value**
Network (N)	The vulnerability can be exploited remotely over a network.	0.85
Adjacent (A)	The attack requires access within the same local network.	0.62
Local (L)	The attacker must have local access to the system.	0.55
Physical (P)	Physical access to the device is required.	0.2

Table 2. Attack Vector (AV)

Attack Complexity (AC) in Table 3 describes the conditions beyond the attacker's control that must exist to exploit the vulnerability.

Metric Value	Description	Numerical Value
Low (L)	The attack does not require specific conditions or is easily repeatable.	0.77
High (H)	The attack requires access within the same local network.	0.44

Table 3. Attack Complexity (AC)

Privileges Required (PR) in Table 4 describes the level of privileges an attacker must possess before successfully exploiting the vulnerability.

Metric Value	Description	Numerical Value
None (N)	No privileges are required; any user can exploit the vulnerability.	0.85
Low (L)	Low-level privileges are needed (e.g., a standard user).	0.62 (S:C 0.68)
High (H)	High-level privileges are required (e.g., administrator access).	0.27 (S:C 0.5)

Table 4. Privileges Required (PR)

User Interaction (UI) in Table 5 describes the requirement for a human user, other than the attacker, to participate in the successful compromise of the vulnerable component.

Metric Value	Description	Numerical Value
None (N)	No user interaction is needed; the attacker can exploit the vulnerability without user action.	0.85
Required (R)	User action (e.g., clicking a link or opening a file) is required for exploitation.	0.62

Table 5. User Interaction (UI)

Scope (S) in Table 6 captures whether a vulnerability in one vulnerable component impacts resources in components beyond its security scope.

Metric Value	Description	Numerical Value
Unchanged (U)	The exploit only affects the vulnerable component.	1
Changed (C)	The exploit can extend beyond the vulnerable component, impacting other systems or components.	1.08

Table 6. Scope (S)

Impact Metrics describe the effects of a successfully exploited vulnerability on the component that suffers the worst outcome that is most directly and predictably associated with the attack. Only

the increase in access, privileges gained, or another negative outcome as a result of successful exploitation should be considered when scoring the Impact metrics of a vulnerability. If a scope change has not occurred, the Impact metrics should reflect the Confidentiality, Integrity, and Availability impacts to the vulnerable component. However, if a scope change has occurred, then the Impact metrics should reflect the Confidentiality, Integrity, and Availability impacts to either the vulnerable component, or the impacted component, whichever suffers the most severe outcome.

Confidentiality Impact (C) in Table 7 measures the impact on the confidentiality of the information resources managed by a software component due to a successfully exploited vulnerability.

Metric Value	**Description**	**Numerical Value**
None (N)	No loss of confidentiality.	0
Low (L)	Limited impact on confidentiality, with partial data exposure.	0.22
High (H)	Complete loss of confidentiality, exposing all data.	0.56

Table 7. Confidentiality Impact (C)

Integrity Impact (I) in Table 8 measures the impact on integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and veracity of information.

Metric Value	**Description**	**Numerical Value**
None (N)	No loss of integrity.	0
Low (L)	Limited impact on data integrity, with some data alteration possible.	0.22
High (H)	Complete loss of integrity, leading to total data compromise.	0.56

Table 8. Integrity Impact (I)

Availability Impact (A) in Table 9 measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability.

Metric Value	**Description**	**Numerical Value**
None (N)	No impact on availability.	0
Low (L)	Reduced performance or intermittent disruptions.	0.22
High (H)	Complete loss of availability, rendering the system or service unusable.	0.56

Table 9. Availability Impact (A)

Calculating the Base Score. The Base Metrics combine into a Base Score, which ranges from 0.0 to 10.0. This score reflects the intrinsic severity of the vulnerability, independent of any specific

environment. The Base Score is computed through a mathematical formula that uses the values of each Base Metrics.

The Base Score is categorised into the following severity levels:

$$\begin{aligned} &None : 0.0, \quad && Low : 0.1 - 3.9, \quad && Medium : 4.0 - 6.9, \\ & && High : 7.0 - 8.9, \quad && Critical : 9.0 - 10.0 \end{aligned}$$

To calculate the base score there had to be multiple sub-calculations made. The Base Score formula Algorithm 4 depends on sub-formulas for Impact Sub-Score Algorithm 1, Impact Algorithm 2, and Exploitability Algorithm 3.

Algorithm 1. Impact Sub-Score (ISS) calculation

$ISS \coloneqq 1 - [(1 - Confidentiality) \times (1 - Integrity) \times (1 - Availability)]$

Algorithm 2. Impact calculation

if $Scope$ is $unchanged$ **then**
    $Impact \coloneqq 6.42 \times ISS$
else if $Scope$ is $changed$ **then**
    $Impact \coloneqq 7.52 \times (ISS - 0.029) - 3.25 \times (ISS - 0.02)^{15}$
end if

Algorithm 3. Exploitability calculation

$Exploitability \coloneqq 8.22 \times AttackVector \times AttackComplexity \times PrivilegesRequired \times UserInteraction$

Algorithm 4. Base Score calculation

if $Impact \leq 0$ **then** Base Score $\coloneqq 0$
    if $Scope$ is $unchanged$ **then**
        $BaseScore \coloneqq \min\lceil(Impact + Exploitability), 10\rceil)$
    else if $Scope$ is $changed$ **then**
        $BaseScore \coloneqq \min\lceil 1.08 \times (Impact + Exploitability), 10\rceil)$
    end if
    $=0$

1.6.2 Temporal metrics

In the Common Vulnerability Scoring System (CVSS), the Temporal Metrics allows adjustment of the Base Score based on factors that change over time. These metrics provide a way to refine a vulnerability's score as new information, such as exploit availability or remediation, becomes available. The three Temporal Metrics in CVSS v3.1 Exploit - Code Maturity (E), Remediation Level (RL), and Report Confidence (RC).

Exploit Code Maturity (E). This metric reflects the likelihood and availability of exploit code or techniques for the vulnerability. The more mature or available the exploit code is, the higher the score Table 10.

Metric Value	Description	Numerical Value
Not Defined (X)	No impact on availability.	1
High (H)	Exploit code or techniques are widely available and mature.	1
Functional (F)	Exploit code is functional but may be less mature.	0.97
Proof-of-Concept (P)	There is proof-of-concept code, but it may not be fully functional or reliable.	0.94
Unproven (U)	No exploit code is available or publicly accessible.	0.91

Table 10. Exploit Code Maturity (E)

Remediation Level (RL). This metric reflects the availability and level of remediation for the vulnerability. The more readily available and complete the remediation, the lower the Temporal Score Table 11.

Metric Value	Description	Numerical Value
Not Defined (X)	Default; no adjustment to the Base Score.	1
Unavailable (U)	No remediation, such as a patch or workaround, is available.	1
Workaround (W)	A workaround is available, but it may not completely resolve the issue.	0.97
Temporary Fix (T)	A temporary or partial fix is available.	0.96
Official Fix (O)	An official patch or solution is available and addresses the vulnerability effectively.	0.95

Table 11. Remediation Level (RL)

Report Confidence (RC). This metric assesses the level of confidence in the vulnerability's existence and technical details based on the quality and source of the report Table 12.

Metric Value	Description	Numerical Value
Not Defined (X)	Default; no adjustment to the Base Score.	1
Confirmed (C)	The vulnerability is confirmed by reliable sources, such as the vendor.	1
Reasonable (R)	There is reasonable evidence for the vulnerability, but some uncertainty remains.	0.96
Unknown (U)	The report is unverified, with limited information about the vulnerability.	0.92

Table 12. Report Confidence (RC)

Temporal Score Calculation. The Temporal Score is calculated by adjusting the Base Score using the Temporal Metrics Algorithm 5

Where E, RL, and RC are the numerical values associated with the selected Temporal Metrics.

Algorithm 5. Temporal Score calculation

1: $TemporalScore := BaseScore \times E \times RL \times RC$=0

Each metric value has a defined weight that decreases the score as the availability of exploits, level of remediation, or confidence in the vulnerability decreases. Temporal Metrics provide a dynamic layer to the CVSS score by considering factors that evolve as a vulnerability matures or is mitigated. By applying Temporal Metrics, organisations can prioritise vulnerabilities to reflect their current severity and mitigations available, which is useful for keeping vulnerability management up-to-date with real-world conditions.

1.6.3 Environmental Metrics

The Environmental Metrics in the Common Vulnerability Scoring System (CVSS) allow organisations to adjust the Base and Temporal Scores of a vulnerability based on factors specific to their environment. These metrics help prioritise vulnerabilities by considering the impact of the vulnerability on an organisation's particular systems, configurations, and requirements. The Environmental Metrics include two main categories - Security Requirements, and Modified Base Metrics.

The Security Requirements metrics enable organisations to specify the importance of the confidentiality, integrity, and availability impacts in their environment. Each requirement can be set to indicate whether that aspect is critical, moderately important, or unimportant for the organisation.

Each requirement of Confidentiality Requirement (CR), Integrity Requirement (IR), and Availability Requirement (AR) has the values depicted in Table 13.

Metric Value	Description	Numerical Value
Not Defined (X)	Default; no adjustment to the Base Score.	1
High (H)	The requirement is of high importance to the organization.	1.5
Medium (M)	The requirement is of medium importance to the organization.	1
Low (L)	The requirement is of low importance to the organization.	0.5

Table 13. CR, IR, AR values

Modified Base Metrics. The Modified Base Metrics allow an organisation to adjust the original Base Metrics according to the specific conditions of their environment. Each Modified Base Metric corresponds to a Base Metric, providing a way to reflect environment-specific characteristics that might influence the exploitability or impact of a vulnerability.

These values are modified versions of Attack Vector, Attack Complexity, Privileges Required, User Interaction, Scope, Confidentiality Impact, Integrity Impact, and Availability Impact metrics with corresponding labelling: MAW, MAC, MPR, MUI, MS, MC, MI, MA. Each Modified Base Metric value has the same possible choices as the original Base Metric values.

Environmental Score Calculation. The Environmental Score is calculated by adjusting the Base Score using the Environmental Metrics Algorithm 6.

Algorithm 6. Environmental Score calculation

1: $EnvironmentalScore := TemporalScore \times ModifiedImpact$

Where the Modified Impact reflects the importance of confidentiality, integrity, and availability impacts as specified by the organisation. And calculation considers all Modified Base Metrics and Security Requirements in the specific environment.

The Environmental Metrics enable organisations to prioritise vulnerabilities based on their environment's specific impact and requirements. By taking into account factors like security requirements and environmental adjustments to the Base Metrics, organisations can customise vulnerability scores to more accurately reflect their operational needs.

1.6.4 Comparison of CVSS v4 and CVSS v3.1 Calculations.

The calculation methods for CVSS v4 and CVSS v3.1 aim to quantify the severity of vulnerabilities, but CVSS v4 introduces enhancements to improve precision and flexibility.

In CVSS v3.1, the overall score is calculated using Base Metrics (Exploitability and Impact) to determine the Base Score, which is then modified by the Temporal and Environmental Metrics Algorithm 5.

The final Environmental Score then adjusts the Temporal Score using Security Requirements and Modified Base Metrics Algorithm 6.

CVSS v4 introduces additional metrics and modifiers to address modern security needs, such as those presented by IoT, cloud environments, and supply chains. Key enhancements include:

- New Metrics: CVSS v4 adds Attack Requirements and Safety Impact to capture more specific exploit conditions and impacts.
- Granularity of Severity Ratings: CVSS v4 refines scoring for more nuanced prioritisation, allowing for more precise severity categories.
- Dynamic Scoring Adjustments: The scoring formula now includes options for automatically adjusting scores based on the evolving exploit and remediation landscape.

In addition to these modifications, CVSS v4 scoring formulas are adapted to balance the increased range of metrics without disproportionately affecting the score. The final Environmental Score in CVSS v4 is calculated by incorporating these new metrics directly into the Base Score and then applying modified Temporal and Environmental values as applicable. This structure allows for a more tailored vulnerability assessment that better aligns with real-world risk.

In summary, CVSS v4 maintains the core principles of CVSS v3.1 but enhances the calculation methods and adds new metrics for greater precision and applicability in diverse environments.

These calculations will be useful in determining the CVE description severity from its description. First, CVSS v3.1 will be predicted, and then a CVSS v4 calculation will be performed on top of it. The only problem is determining how to classify the CVE descriptions.

1.7 CVE classification methods and tools

Classifying Common Vulnerabilities and Exposures (CVE) and Common Weakness Enumeration (CWE) is a critical component in the ongoing effort to enhance software security. As the number of

vulnerabilities continues to grow, the need for effective classification becomes increasingly important. By accurately categorizing these vulnerabilities, organizations can prioritize their responses, allocate resources more effectively, and ultimately reduce the risk of exploitation. The integration of machine learning (ML) techniques into this classification process has emerged as a promising solution, enabling automation that leads to faster and more efficient identification and management of vulnerabilities.

Several machine learning methods have been explored for the classification of CVE and CWE, each offering unique advantages:

- **Decision Tree Classifier:** This algorithm constructs a model that predicts the class of vulnerabilities based on features extracted from the code. Decision trees are intuitive and easy to interpret, making them a popular choice for initial classification tasks. They work by recursively splitting the data into subsets based on feature values, ultimately leading to a decision about the class label [25].

- **Random Forest Classifier:** As an ensemble method, the Random Forest Classifier aggregates predictions from multiple decision trees to enhance classification accuracy. By combining the outputs of various trees, this method reduces the risk of overfitting and improves generalization to unseen data. It is particularly effective in handling large datasets with numerous features, making it well-suited for vulnerability classification [3].

- **Support Vector Classifier:** This method employs a hyperplane to separate different classes in a high-dimensional space, making it a robust approach for classifying vulnerabilities. By focusing on the most informative data points (support vectors), this classifier can effectively manage complex datasets and is particularly useful when the classes are not linearly separable [6].

- **Logistic Regression Classifier:** A widely used statistical method, logistic regression is suitable for both binary and multi-class classification tasks. It estimates the probability of a vulnerability being present based on input features, making it a valuable tool for understanding the likelihood of different vulnerabilities occurring [10].

To optimize the performance of these machine learning models, hyperparameter tuning is essential. This process involves adjusting the parameters that govern the learning process to achieve the best possible results. Properly tuned hyperparameters can significantly enhance the model's performance, leading to higher accuracy on both training and validation datasets. Tuning hyperparameters can also help in understanding how different settings affect model behavior. Different algorithms have various hyperparameters that can affect their performance.

- **Randomized Search Cross Validation:** This technique enhances model performance by randomly sampling from a predefined range of hyperparameters. By exploring a diverse set of parameter combinations, it allows for efficient optimization without the exhaustive computational cost associated with evaluating every possible configuration [2].

- **Grid Search Cross Validation:** In contrast, grid search is a systematic approach that evaluates all possible combinations of specified hyperparameters. While it can be computationally intensive, it provides a thorough examination of the parameter space, ensuring that the best-performing model is identified. This method is particularly useful when the number of hyperparameters is limited, allowing for a comprehensive search of the optimal.

The use of machine learning techniques to classify Common Vulnerabilities and Exposures (CVEs) and predict Common Vulnerability Enumeration (CWE) images and Common Vulnerability Scoring System (CVSS) indicators is a significant advancement in software security. Traditional machine learning algorithms, including decision trees, random forests, support vector machines (SVMs), and logistic regression, are utilized to analyze and categorize vulnerabilities in software systems. Applying hyperparameter optimization techniques such as grid search cross-validation (CV) and random search CV can greatly improve the performance of these algorithms. These techniques help identify the best model parameters, enhancing prediction accuracy and model robustness. Optimizing model parameters is crucial to ensure that the models can effectively generalize from unseen data, which is vital for effective vulnerability management.

Automating the grading process and integrating machine learning into vulnerability management systems is a proactive methodology that improves the identification and management of vulnerabilities. This approach also helps prioritize remedial actions based on estimated CVSS scores, allowing organizations to react more quickly to emerging threats and improve their overall security posture. Additionally, integrating machine learning into vulnerability management systems enhances the resilience of software systems, increasing their ability to withstand potential attacks and protect sensitive data. As the cybersecurity landscape becomes more complex, utilizing advanced machine learning techniques for vulnerability classification becomes crucial in developing robust defenses against complex threats. This emphasizes the significance of using data-driven methodologies to enhance software security and effectively mitigate risk.

2 Implementation

All of the experiments and data processing were done using Python programming language and JupyterLab environment. Multiple python files were created that includes custom methods, classes and defined constants. Custom method class contains methods and classes that are used in data processing, CVSS metric mapping and metrics calculations. A custom class of CVEClassifier was developed to predict CWE mapping and CVSS metrics from a JSON object.

All of the code uses data processing libraries, that work with JSON objects, pandas for data management, numpy for calculations, logging, to track issues with the code. Matplotlib pyplot and seaborn are used for data visualization and pickle for dumping trained model data into file. Sklearn is primary library that hold necessary model, metrics and feature extraction tools for developing custom training model.

In Section, 2, more detailed information will be provided on how the data were processed. What additional steps were taken to make the data more suitable for all the experiments and analysis. From the description alone, we will gain more insights into what models were used to classify CWE or CVSS metric predictions. Some details on comparing multiple classification models and their outcomes. How were they compared by comparing results before and after hyperparameter adjustments. What differences were when comparing training models with different data sizes. What kind of data pre-processing was necessary.

2.1 Data description

All data used in each experiment was taken from National Vulnerability Databases. Data are accessible in huge quantities, starting in early 1999, when NVD launched the CVE project, and ending with the latest submitted entries in 2024. In total, over 236 thousand CVE entries were recorded in 25 years. All necessary data comes in different folders separated by year and stored in JSON format. Most datasets are usually in CSV format for faster pre-processing, yet this one is in JSON because all CVE entries are widely used. Since JSON is a common standard for API development and endpoint creation in today's web service technologies, it is only natural that such data be in the most accessible format.

In Figure 3 it can be seen that all CVE entries follow the same structure. Id is a string attribute that refers to the CVE's identification number, which consists of the CVE appendix, followed by the year that the CVE was registered and the unique number that was reserved for analysis from the number queue. The "SourceIdentifier" attribute is a string that refers to the reported source email. The published and "lastModified" string attributes refer to the date and time that the CVE was registered or updated, respectively. The "VulnStatus" string attribute refers to the status of the CVE that it currently is. An array of descriptions refers to the descriptions that a particular CVE may have; it is possible to have multiple descriptions in different languages. Commonly, English and Spanish languages appear on some CVE descriptions, but English remains the primary language. The "Metrics" attribute, which refers to the CVSS score and data, can be separated into a unique JSON object with its specific fields. Some descriptions lack this attribute, especially those with "vulnStatus" set as "Avaiting Analysis." "Weaknesses" is also an optional array that describes the CWE entry value. This attribute is able to have multiple possible weaknesses in accordance with the description or even metrics. The "Configurations" array is optional and refers to the possible configurations that the reported CVE might affect. The "References" optional array attribute provides additional information that can be related to the CVE entry.

Figure 3. CVE Structure

The metrics attribute is divided further because multiple versions of CVSS metrics can be attached to the CVE entry. This also applies to data reported by multiple sources with the same description. The same CVE entry gets updated, and additional calculated CVSS metrics are added to the metrics attribute.

When the theses were written, CVSS version 2 was already deprecated and no longer supported. Yet there are cases where some new CVE entries had CVSS version 2 metrics attached. In comparison, CVSS version 3.1 had less attributed compared to version 2. All of the attributes regarding privileges were moved inside the CVSS data attribute under a single attribute called "privilegesRequired" Figure 4.

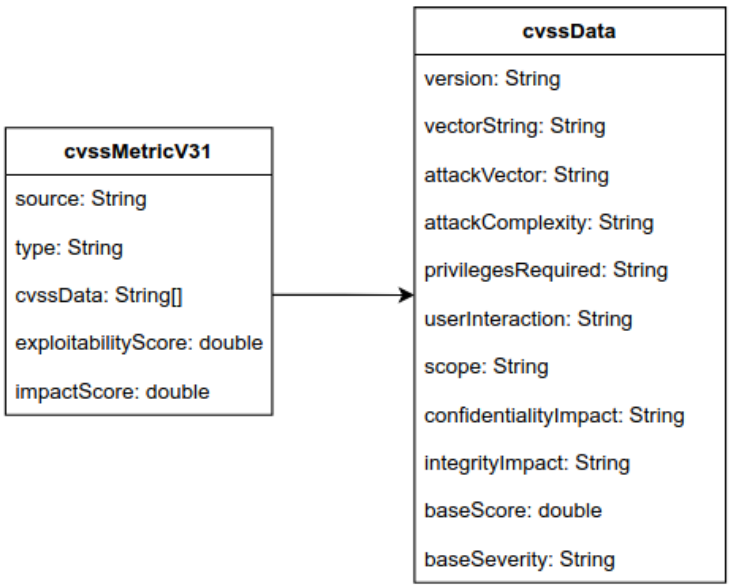

Figure 4. CVSS version 3.1 object structure

The main difference between version 3.1 and version 2.0 CVSS data is that multiple CVSS metrics attributes were moved into the data. CVSS data "vectorString" has more keywords and an appended version number Figure 5.

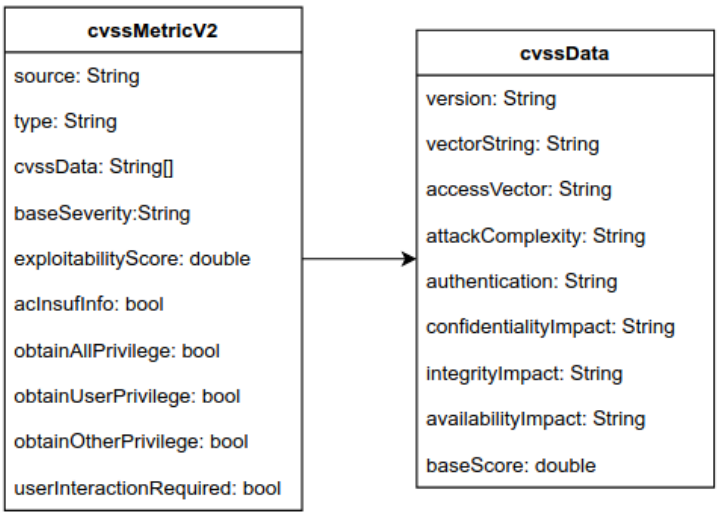

Figure 5. CVSS version 2 object structure

Since 2023, a new standardized version 4 of CVSS has been released, but it has yet to be used in current CVE entries. CVSS version 4 will introduce a new and updated metric system that overhauls all the calculations of the previous metric by introducing multiple different attributes and fields.

2.2 Data preparation for CWE classification

The primary part of pre-processing the data is to extract the most needed key points for further classification methods. When predicting CWE entries for newly created CVE entries, it is best to associate the CWE label with specific keywords that refer to them. The earliest implementation was focused on extracting the top 25 commonly occurring CWE entries and compiling a dictionary based on their general description that would refer to the specific CWE entry if such a keyword was detected in the CVE description.

The first implementation approach was not very efficient and showed very poor results. After several more failed attempts to improve overall prediction accuracy, the idea of using only the 25 most frequently occurring CWE entries and their keywords was completely abandoned. The earliest implementation involved completely deconstructing a JSON object and mapping every component to its designated field or even an entire class. Such an approach was impractical since it created the same object only in a different format and obstructed further data processing. Because the data was unprocessed overall training results were very poor. By using unprocessed data and implementing Decision Tree Classifier model, the accuracy of prediction was less than 50%. Additional models like Neural Networks barely made any decent changes that could be reliable and overall accuracy was still less than 50%. Predicting with such models was very inefficient, and good predictions were too sparse to be able to determine if there was any successful prediction. A new approach was taken in making successful predictions.

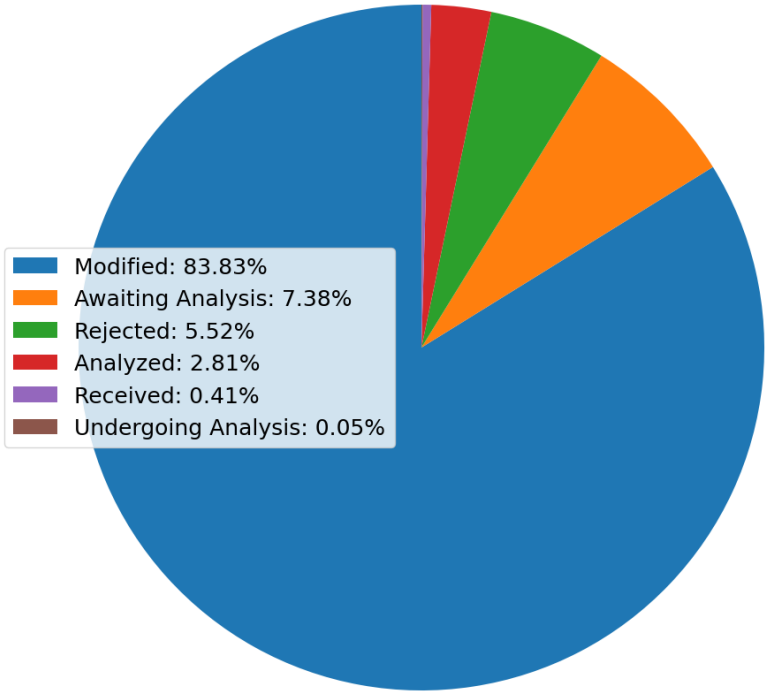

Figure 6. CVE distribution by status of 246,993 entries

Several key features had to be addressed, including the status of the CVE entry itself. In Figure 6 there can be seen six possible statuses for CVEs were distinguished among the over 240,000 entries. The most common and notably useful status is "Modified," which accounts for nearly 84% of all CVEs listed since 1999. Entries with the status "Rejected" cannot be used because there was some reason why they were rejected. Some causes are that such an entry is a duplicate of another CVE entry and should be referred to it. Some entries were not used, withdrawn, or associated by the CNA for a certain time. Additionally, entries with the status "Awaiting Analysis," "Undergoing Analysis," and "Received" are considered new entries, and they have yet to receive their associated weaknesses and metrics. Therefore, only entries with "Modified" and "Analyzed" statuses can be used for further research.

In Figure 7, it can be seen that there are multiple CWE naming patterns. The standard is a CWE label appended with a number, but there are two additional entries that takes around 22% of the data. Such entries have their weakness label set to "NVD-CWE-info" or "NVD-CWE-other." Such weaknesses have their descriptions and are not needed for further study. Their descriptions are as follows:

- **NVD-CWE-noinfo** - There is insufficient information about the issue to classify it, details are unkown or unspecified.
- **NVD-CWE-other** - NVD only uses a subset of CWE for mapping instead of the entire CWE, which does not cover the weakness type.

One of the final data processing was cleaning up CVE entry descriptions. Since humans create descriptions, they are prone to errors, which can result in more unnecessary words or letters than

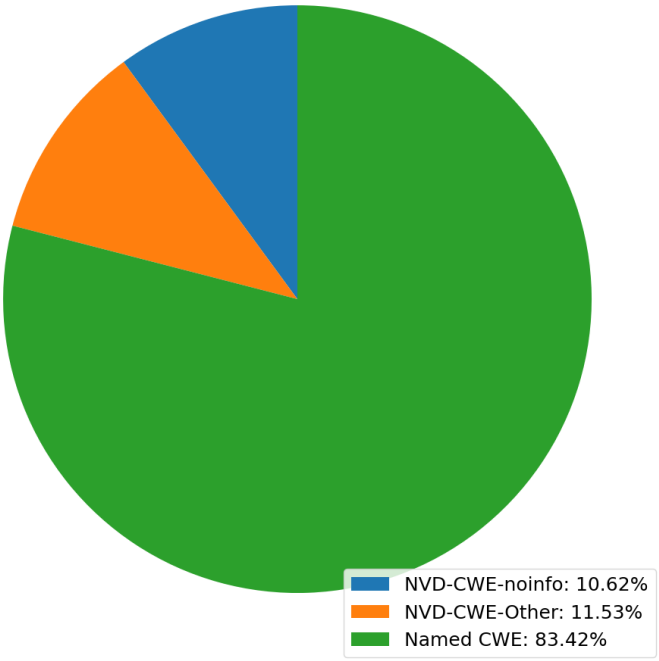

Figure 7. CWE distribution by category of 246,993 entries

necessary. Every CVE description had to undergo text processing and tokenization, splitting the entire description and removing unnecessary words. And finally, these words joined again, but without any unnecessary words that could obscure the data. A list of keywords was created in a separate file for this case. This list was populated with the most common words that appear in any sentence and have no particular meaning or impact on predicting descriptions. There are many keywords, starting with random letters and ending with words that do not add any meaning to the description context Table 14.

Table 14. Sample list of Stop Words

Stop Words						
i	me	my	myself	we	our	ours
ourselves	you	your	yours	yourself	yourselves	he
him	his	himself	she	her	hers	herself
it	its	itself	they	them	their	theirs
themselves	what	which	who	whom	this	that
...	...	...	...	...	...	...
...	...	...	...	...	...	...
how	all	any	both	each	few	more
most	other	some	such	no	nor	not
only	own	same	so	than	too	very
s	t	can	will	just	don	should

2.3 Data preparation for CVSS classification

Considering CVSS classification, the data had to be prepared similarly to CWE data. Both datasets have the same description preparation steps. The difference between the two data sets is that vital CVEs with defined weaknesses but no metrics might be removed. Some CVE entries may have multiple metric entries because there were several versions of metric calculations over 25 years. Since there were four main versions throughout the CVE lifetime, there are some things to consider. The primary concern is that the current standard of the CVE metric version is 3.1. This is the latest and final metrics iteration before moving on to version 4. Older metric implementations consist of version 2 and version 3.0. 3.0 barely has any difference from the current standard. Therefore, its conversion poses no issues. Version 4 is the newest metric iteration, drastically changing overall evaluation and calculation. Since it is a rather new metric, barely any entries contain this version.

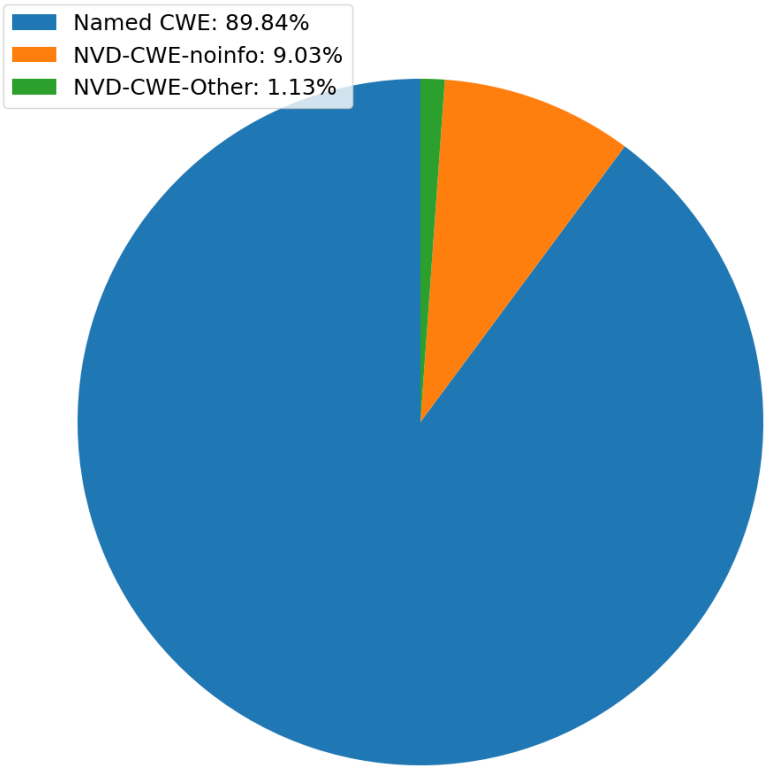

Figure 8. Metrics versions used by 246,993 CVE entries

Figure 8 shows that among 219756 entries, 76.61% comprises of version 2. Versions 3 and 3.1 are rather similar, and they do not have any major differences between each other. They both individually appear in 20.82% entries for version 3.0 and 44.45% for version 3.1; in total, they are present in nearly 65% of total entries. Additionally, the newest version is present only in 0.05 of the entries, it is bound to change since this is the newest iteration of the metrics.

Some sort of standardization of the metrics had to be done in order to prepare data for making predictions. CVSS version 2 is the earliest standard of metric calculation and the most dominant among the entries, yet it is deprecated and no longer used. For this reason, some sort of data conversion is required in order to use a broader data spectrum for the model training. Every CVE entry with only version 2 metrics had to be converted into version 3.1, the latest.

The Version 2 standard had attributes that are no longer used by the newer versions; additionally, it lacks a couple of critical attributes used by its successors. In Figure 9, it can be seen that the metrics taken from the same CVE entry have multiple metrics versions. In comparison with one another, the entire metrics structure was reduced in version 3.1, and its data was increased from version 2. The most significant change was the introduction of 3 new attributes: privileges required, user interaction and scope. Additionally, the access vector has been changed to an attack vector.

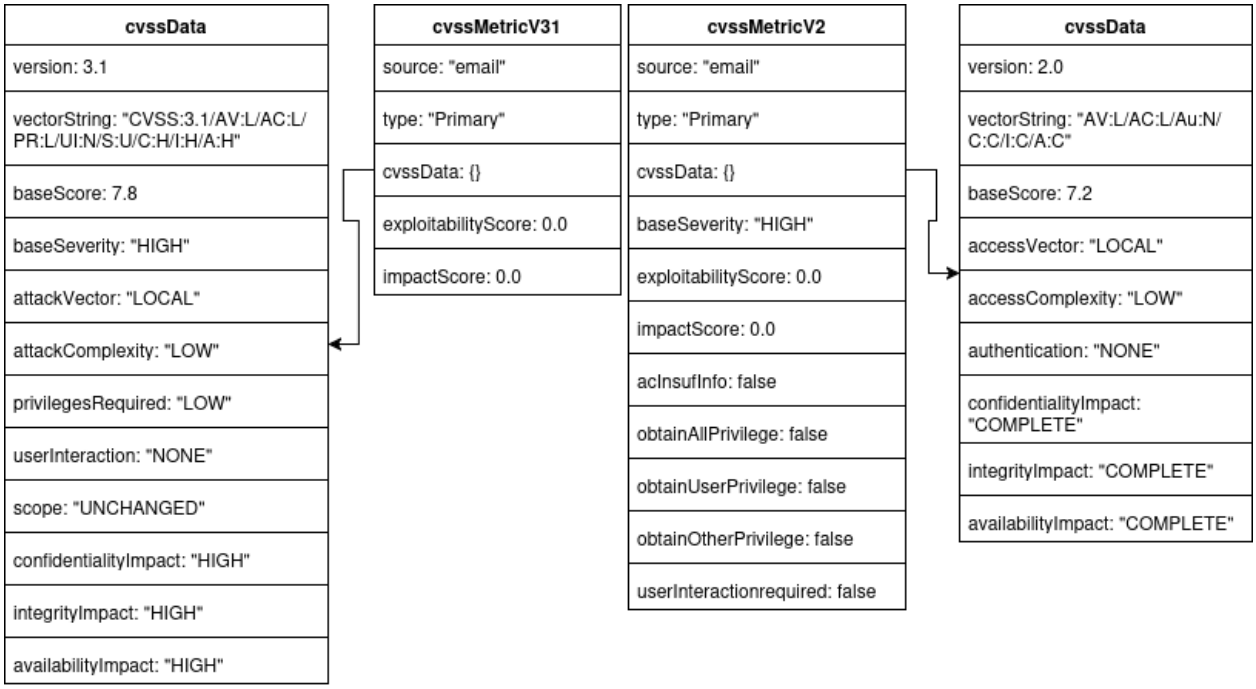

Figure 9. CVSS metrics compared between version 3.1 and version 2

To convert CVSS version 2 into version 3.1, attributes, such as privileges required and user interaction were set to the value of none, while scope was set to unchanged. This represents the lowest values in the newly introduced attributes. Additionally, version 2 data that had attributes with a value of partial was changed to low, while complete was changed to high. There are some additional cases where the value medium is mapped to high or low. This is determined by the CVSS version 2 vector itself. The remaining values were mapped according to their respective attributes. After mapping the primary values and creating a new version 3.1 vector, the remaining scoring values were recalculated using version 3.1 calculation methods and newly assigned data values.

Since there are over 240,000 entries it is a bit too much to perform any experiments. Figure 10 shows how many entries have been identified every year. Judging by the yearly increase of CVE entries it is best to take the latest year for experimentation. Since the year 2024 is the latest one, there can be too many entries that are still waiting for further analysis. Additionally, this year can be used for manual verification of the predictions. For further experiments, the year 2023 will be taken since it is the second-largest data set consisting of 23,502 entries in total.

2.4 Pseudocode of an algorithm

In pseudo-code Algorithm 7 there is described the model that is being developed. The developed model should take in a CVE entry in the form of a JSON file, and extract its description. Vectorize the description and transform it to both CWE model and CVSS model. Vectorized CWE description

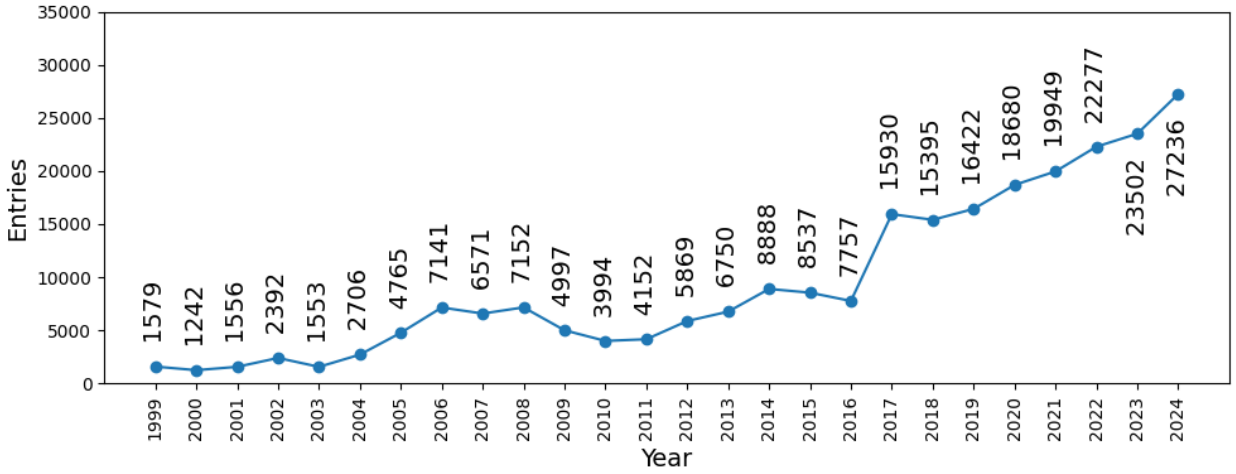

Figure 10. CVE entries per year

then is predicted and the results are encoded into the corresponding label creating a new weakness entry for the CVE to have. Vectorized CVSS description is predicted using CVSS model. Predicted values are encoded and mapped to the corresponding metrics. Based on predicted metric values, scoring metrics are calculated by the predicted value weights. A new metrics entry is created for the CVE to have. Updates the loaded JSON file with newly predicted weakness and metrics.

Algorithm 7. Basic model pseudocode

Input: CVE JSON file
Output: Updated CVE JSON file with CWE and CVSS predictions
Load the CVE JSON file
Initialize the CVEClassifier with the loaded data
Extract the description from the classifier
Vectorize the description for the CWE model
Vectorize the description for the CVSS model
Predict the CWE mapping using the CWE model
Encode the predicted CWE label into textual format
Predict the CVSS metric values using the CVSS model
Calculate the CVSS scoring metrics based on the predicted metrics
Construct the attack vector string from the predicted metrics
Update the loaded CVE JSON file with the new CWE prediction
Update the loaded CVE JSON file with the new CVSS metric predictions

3 Experiments

After data preparation, multiple experiments were performed on both CWE and CVSS datasets. During experimentation, observations were made to determine the most accurate model by comparing them without any hyperparameter adjustments.

3.1 CWE experiments

To train models for CWE prediction only 2 attributes were needed: description and weakness number. For training all of the description keywords will be assigned to the corresponding CWE label. In order to increase the prediction accuracy there had to be data limitations added in order to keep more clearer model training and predictions. There are many different CWE entries in a single year, there are some cases where a single CWE entry might appear very rarely. These occurrences are infrequent it is possible to assume that they can be set as a clear threshold at what occurrence of a threat can be accounted for in training. By filtering data from such entries, basic model training can be done. For this experiment, a 1% threshold has been chosen. Any summed-up CWE entry if it falls under 1% of the threshold they are dropped out of the dataset. By removing CVE entries that do not have valid CWE entries we get a total of 19,678 valid entries. With the application of 1% threshold, the dataset was reduced to 12,347 valid CVE entries in total.

To compare the accuracy with both datasets, training was done with Decision Tree Classifier [18]. In Figure 11 it is seen that unfiltered data with Decision Tree Classification accuracy is barely 67.45%, while filtered data with the same classification model shows 82.63%. Since filtered data provides better results it is decided for further experiments to use filtered data.

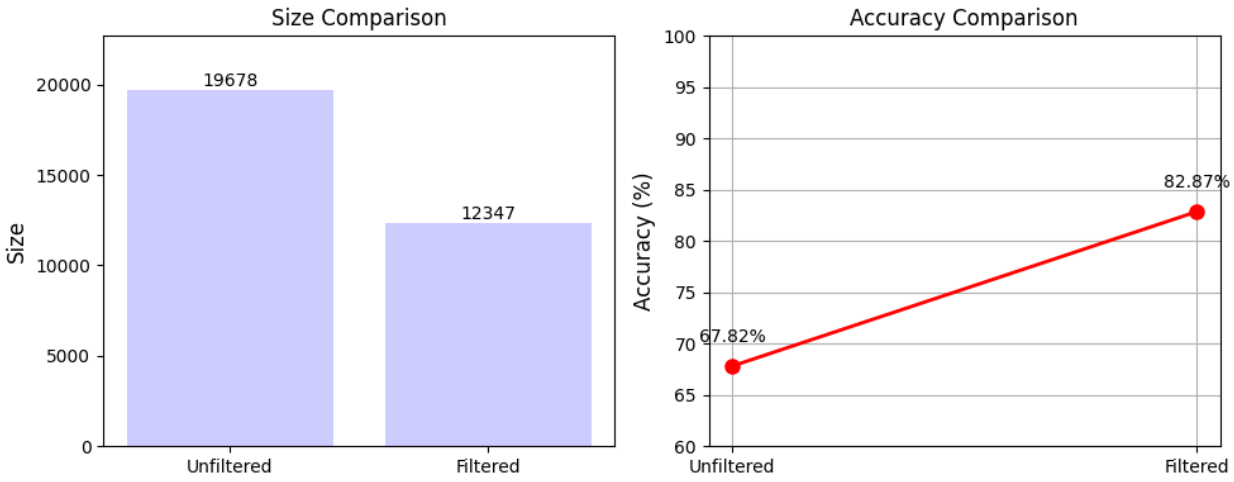

Figure 11. Comparison of unfiltered and filtered data

Next comparison was made between several Linear models in order to determine which model provides higher accuracy without any hyperparameter adjustments. The models in question for CWE prediction were Decision Tree Classifier [18], Support Vector Classifier [20], Random Forest Classifier [19], and Logistic Regression classifier [17].

The same training method was used on all four classifier models, as it was used to determine the accuracy of filtered and unfiltered data. In terms of accuracy, in Figure 12 Support Vector model performed the worst with only 80.12% accuracy, while Logistic Regression and Decision Tree models had roughly the same accuracy 82.27% and 82.51% respectively. Random Forest model

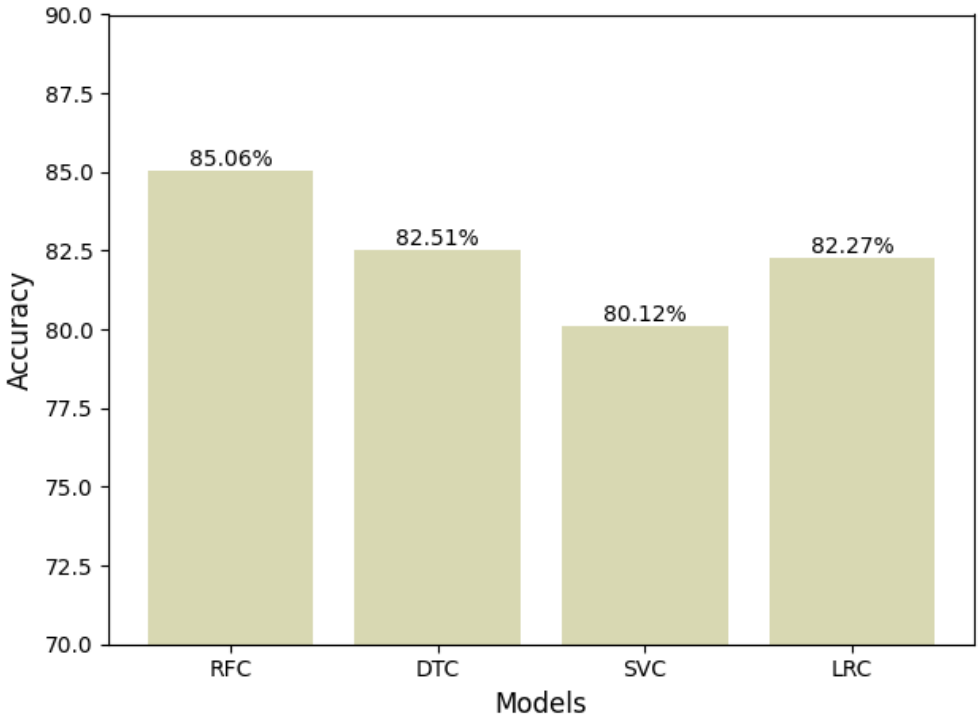

Figure 12. Training accuracy comparison between models

managed to have the highest accuracy of 85.06%. In terms of accuracy alone, Random Forest classifier is the best model to use, yet there are other metrics to be considered.

Figure 13 shows the performance of every model without any adjustments. When comparing other metrics in terms of precision, Logistic Regression showed the best performance with 82.23%, while Decision Tree has the lowest Precision of 71.51%. Recall and F1 Score show very low results for every model except Random Forest which exceeds other models by 3 and even up to 10% at some cases. Overall all of the models perform nearly the same, while Decision Tree falls under the lowest performers it will be abandoned for the next experiment. Random Forest favours higher accuracy, but Logistic Regression shows a higher precision rate, which is very important as well. For the next experiment, Random Forest Classifier and Logistic Regression will be used to make predictions with hyperparameter adjustments.

By utilizing the scikit's Random Search cross [21] validation and Gradient Search cross validation [22] the best estimator was picked for both models. In Figure 14 it is seen that both models gained some sort of performance boost, even if it is non-significant. In regards of Random Forest, there is barely any change seen, while Logistic Regression shows a whole 5% accuracy boost, with a 4% decrease in precision. Although both recall and f1 score had from 7% up to 9% boost which is very good in predicting more accurately.

In order to verify the accuracy of the models in predicting the CWE from the CVE description, there were 10 CVE entries taken from the year 2024 at random. In Table 15 a list of CVE entries is provided with ID in the first column, real CWE value in the second column, in the third column - prediction result from adjusted Random Forest model and in the final column prediction result from Logistic Regression model. Random Forest model predicted 7 out of 10 entries, while Logistic Regression model predicted 6 out of 10 entries.

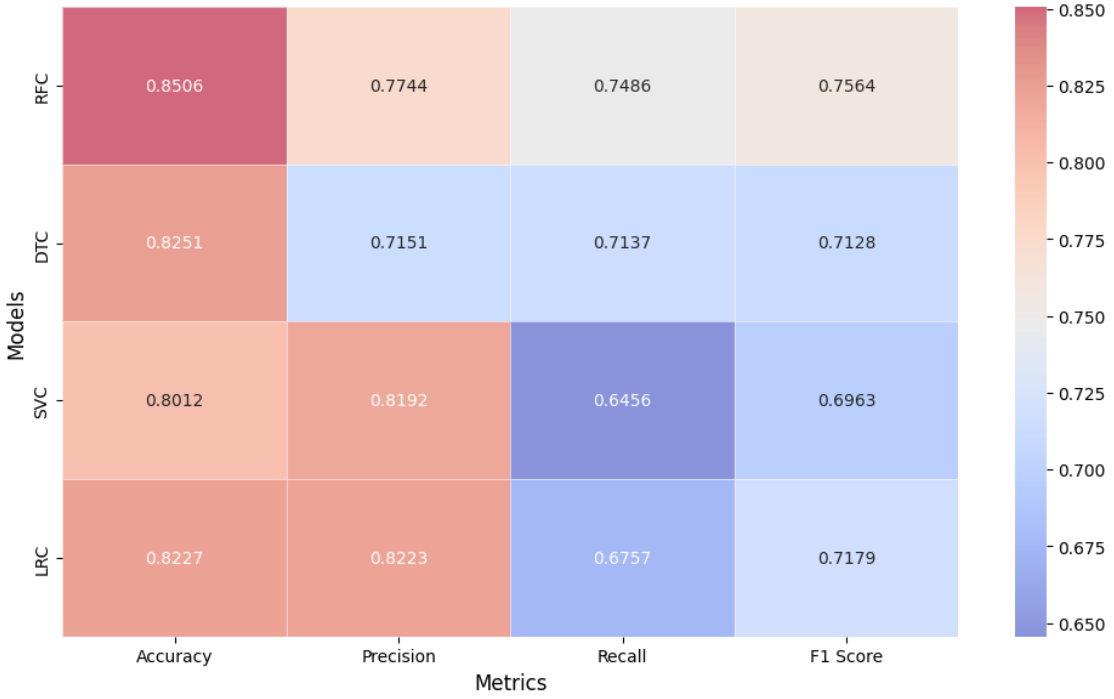

Figure 13. Performance metrics heatmap

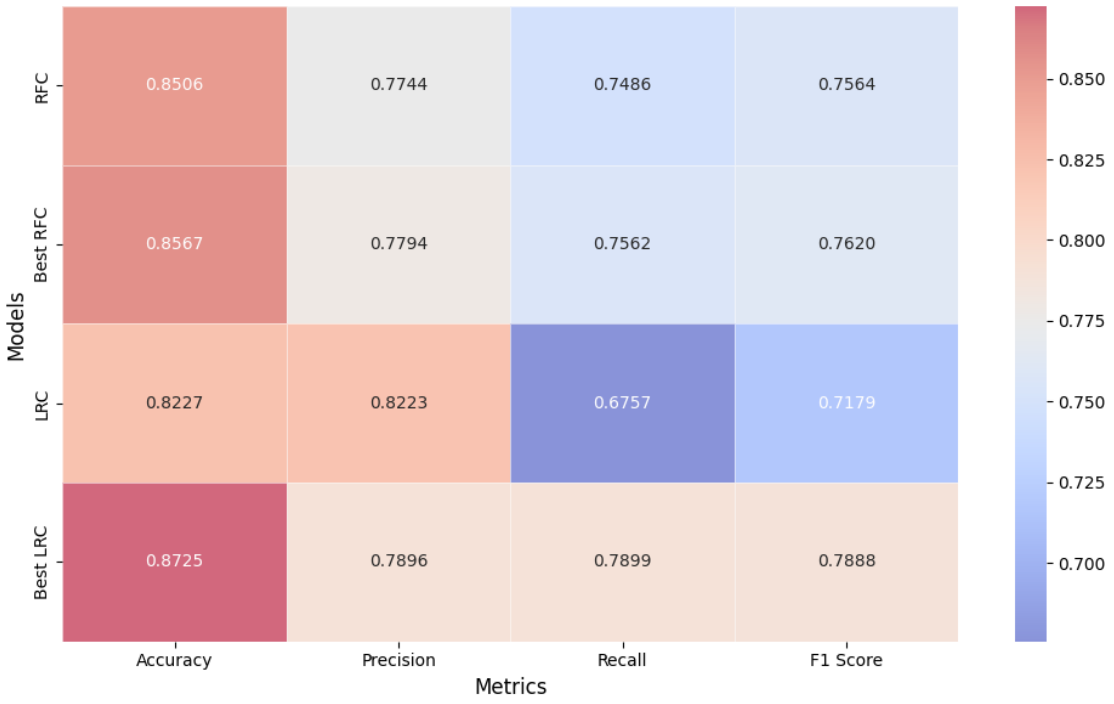

Figure 14. Performance metrics heatmap of Random Forest and Logistic Regression

Table 15. 10 tested CVEs and their descriptions

CVE	Real CWE	Predicted CWE (RF)	Predicted CWE (LR)
CVE-2024-7454	CWE-89	CWE-89	CWE-89
CVE-2024-9799	CWE-79	CWE-79	CWE-79
CVE-2024-8073	CWE-20	CWE-20	CWE-20
CVE-2024-23764	CWE-269	CWE-400	CWE-400
CVE-2024-23494	CWE-89	CWE-89	CWE-89
CVE-2024-27242	CWE-79	CWE-79	CWE-79
CVE-2024-3402	CWE-79	CWE-79	CWE-79
CVE-2024-3153	CWE-400	CWE-400	CWE-434
CVE-2024-6996	CWE-362	CWE-416	CWE-79
CVE-2024-8252	CWE-98	CWE-22	CWE-22

It is clear that after this manual data validation, Random Forest shows better results; however, these experiments were done with data worth a single year. This will change if the entire dataset is used to train both models. Since the dataset contains a lot of entries, it will take a significant amount of time and computing power to perform such a task. But according to Figure 14 it is more likely that the Logistic Regression model will be more favourable for predicting labels even with a bigger dataset.

3.2 CVSS Experiments

For CVSS prediction, more attributes were needed, with a cleaned-up and tokenized description as the primary attribute. The secondary attributes were the attributes taken from CVSS data, these attributes are attack vector, attack complexity, privileges required, user interaction, scope, confidentiality impact, integrity impact and availability impact. These attributes are used to calculate metric scoring like base score, exploitability score and impact score. These metrics use designated calculation methods provided by the NVD, and each attribute value has a certain weight attached to it. Vector string is a combination of the metric version, and each of the attribute's first letter of each word and the first letter of the assigned value.

Every entry regardless of year will be recalculated to use CVSS version 3.1 mapping. The data for the CVSS prediction will not undergo any kind of filtering, since all the entries not containing any metrics will not be present in the dataset. To make each attribute much easier to identify, the value of every attribute was encoded with a positional value of the corresponding attribute. Every attribute has specific values assigned to it. They are constant, so the only time they can change is when there is a major update and the entire metric system is overhauled or improved. The possible values for each metric attribute and the weight of each metric value are described in detail by the First [24] companies documentation on CVSS version 3.1.

After changing values for each attribute, it is much easier and faster to perform model training. The year 2023 was used for training Decision Tree, Random Forest, Support Vector and Logistic Regression models.

In Figure 15 a comparison between all the models is shown. Each of the models predicted every designated metric, which is in total 8 of them. After predictions were made an average of all

the metrics for each type were summed up. This resulted in average data for each model and their statistics. By comparing untuned models, it is seen that Decision Tree performed overall the worst. Although an average of 82% in each of the metrics is a very high value, other models produced better results. While Logistic Regression showed a very high percentage of nearly 87%, the Support Vector classifier managed to outperform the Logistic Regression model in each category by over 0.5% in each category.

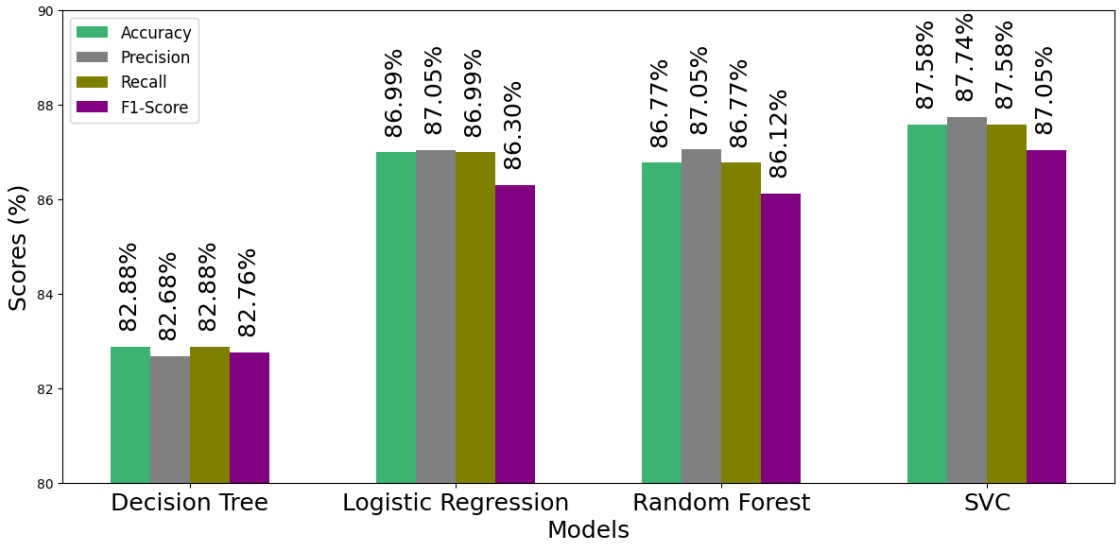

Figure 15. CVSS average model statistics

These statistics were even similar clearly showing, that the Support Vector Classifier is the best choice for further experiments, to predict each attribute of the CVSS metric. The next experiment involves tuning the model hyperparameters. For this purpose, the Grid Search cross-validation was used to search for the most optimal parameter to increase the prediction rate. Main adjustments were done by adjusting the regularization parameter, starting from the lower value of 0.1 which creates a larger margin with the risk of misclassifying. Going to the larger value will lower the margin to increase the prediction rate, yet it makes the model more complex. Additional changes were made to the kernel, to make the model work efficiently with the high-dimensional data by using linear type. As well as using the default Radial Basis Function that can handle non-linear relationships of the data. Gamma adjustments were tested to determine the decision boundaries between a larger radius which leads to smoother decision boundaries and a smaller radius, which leads to more complex decision boundaries. Lastly, class weight adjustments had to be experimented on, since there is no guarantee that dataset classes are imbalanced or not. Because of that reasoning, the class weight of none had to be tested with consideration that all classes are of the same weight, and the class weight of balanced, had to be tested which automatically adjusts the weights of classes.

After making parameter adjustments the results were compared with the untuned model. By analysing differences in training models depicted in Figure 16 it is clear to see that the tuned Support Vector model gains higher performance in every aspect. Precision gained the lowest increase, only 0.28%, recall and accuracy had the same boost of 0.59%, and f1-score had the biggest increase in its performance, gaining 0.84% increase.

The final step in the CVSS experiment was to manually check whether the predictions were accurate enough. Predicting multiple outputs from the same string might cause one or even more metrics to be predicted differently. Table 16 lists 10 tested CVE entries which are the same as the ones used in CWE prediction. In the table there are multiple columns, CVE indicates the CVE id,

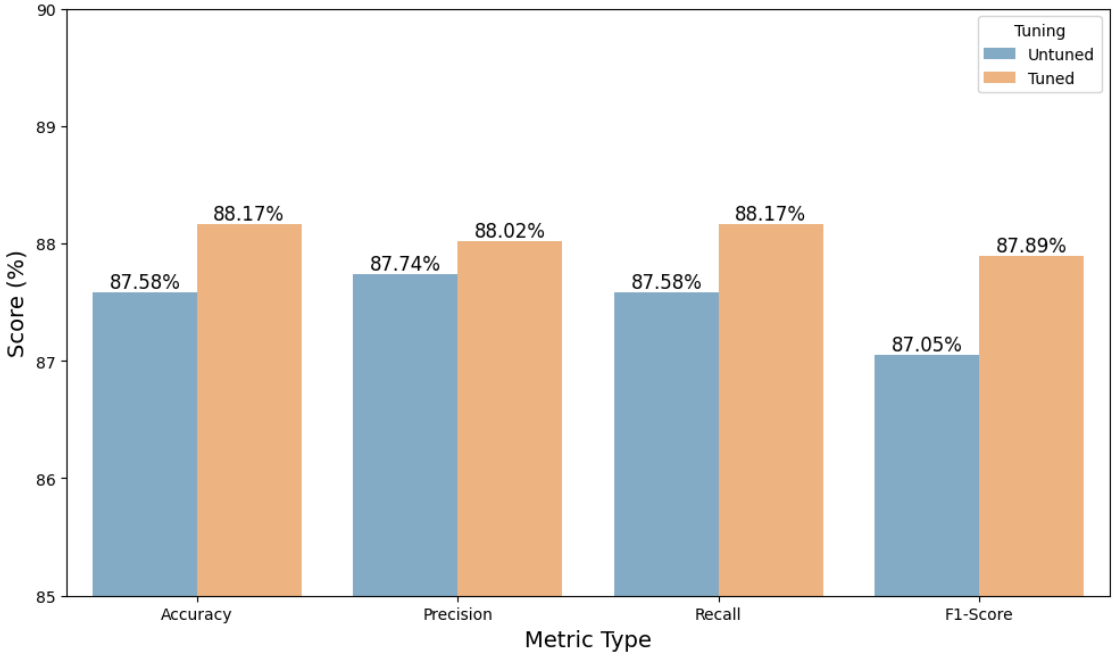

Figure 16. Support Vector tuning result comparison

and Data represents whether the data provided is original, or predicted. The remaining columns are acronyms of the metrics, AV - attack vector, AC - attack complexity, PR - privileges required, UI - User Interaction, S - scope, CI - confidentiality impact, I - integrity impact, AI - availability impact, ES - exploitability score, IS - impact score, BS - basic score. From these 10 tested CVEs, only 4 were predicted perfectly. CVEs: CVE-2024-979, CVE-2024-8073, CVE-2024-23494, and CVE-2024-3403 were predicted identical in comparison to their original entries. CVE-2024-0079, CVE-2024-4453, and CVE-2024-8252 had a single metric predicted incorrectly, and only the CVE-2024-0079 entry had the worst calculated base score, but this is because scope changes affect metrics dramatically. CVE-2024-23764 and CVE-2024-27242 had 2 wrongly predicted attributes, the base score was not impacted much, but CVE-2024-23764 entries exploitability score went up by 1 point. CVE-2024-3153 entry had the worst predicted values and because of that, the scoring is incredibly bad. The reason for this poor prediction of this entry is tied to its assigned CWE mapping which is CWE-400 15. CWE-400 entries are not very common and it is highly discouraged by Mitre [12] themselves, to use this mapping with the real-world vulnerabilities and that is because this mapping is frequently misused. It is highly possible that after training the adjusted Support Vector Classifier model, the prediction rate should increase. The current dataset which is comprised of a single year contains limited samples, and yet it predicts with a very high precision and accuracy.

Table 16. Tested CVE metrics using tuned Support Vector Classifier

CVE	Data	AV	AC	PR	UI	S	CI	I	AI	ES	IS	BS
CVE-2024-979	Original	N	L	L	R	U	N	L	N	2.1	1.4	3.5
CVE-2024-979	Predicted	N	L	L	R	U	N	L	N	2.1	1.4	3.5
CVE-2024-0079	Original	L	L	L	N	C	N	N	H	2.0	4.0	6.5
CVE-2024-0079	Predicted	L	L	L	N	U	N	N	H	1.8	3.6	5.5
CVE-2024-8073	Original	N	L	N	N	U	H	H	H	3.9	5.9	9.8
CVE-2024-8073	Predicted	N	L	N	N	U	H	H	H	3.9	5.9	9.8
CVE-2024-23764	Original	L	L	H	N	U	H	H	H	0.8	5.9	6.7
CVE-2024-23764	Predicted	L	L	L	N	U	H	N	H	1.8	5.9	7.8
CVE-2024-23494	Original	N	L	L	N	U	H	H	H	2.8	5.9	8.8
CVE-2024-23494	Predicted	N	L	L	N	U	H	H	H	2.8	5.9	8.8
CVE-2024-27242	Original	N	L	L	R	C	N	N	L	2.3	1.4	4.1
CVE-2024-27242	Predicted	N	L	L	R	C	N	L	N	2.1	1.4	3.8
CVE-2024-3403	Original	N	L	N	N	U	H	N	N	3.9	3.6	7.5
CVE-2024-3403	Predicted	N	L	N	N	U	H	N	N	3.9	3.6	7.5
CVE-2024-3153	Original	N	L	L	N	U	N	N	H	2.8	3.6	6.5
CVE-2024-3153	Predicted	N	L	L	N	C	H	H	H	2.8	6.0	9.6
CVE-2024-4453	Original	L	L	N	R	U	H	H	H	1.8	5.9	7.8
CVE-2024-4453	Predicted	N	L	N	R	U	H	H	N	2.8	5.2	8.1
CVE-2024-8252	Original	N	L	L	N	U	H	H	H	2.8	5.9	8.8
CVE-2024-8252	Predicted	N	L	L	N	U	H	H	N	2.8	5.2	8.1

Conclusions

Security is one of the most important issues of our time. Every day, security breaches ranging from physical break-ins or fraud to cybercrime are recorded around the world. To prevent the physical consequences of security breaches, special instances have been set up which can take appropriate action. In the context of cyber security, security is highly valued and there is an increasing level of activity in this area, but unfortunately, not everyone is willing to take the time to take steps against cybercrime. In most cases, the concern is only belated.

The study focused on the analysis of security descriptions, which can help to protect against cyber attacks. Of course, the detection of security vulnerabilities is not new, but the processing and analysis of the information is a rather lengthy process. In the course of this work, the following objectives have been set and the following conclusions have been reached:

1. Analysing CVEs is a fairly lengthy process with many steps. Due to the number of steps involved, it can take months or even years to analyse an inventory. One of the main reasons for this would be the human factor that exists between the creation of the inventory and its final analysis. The manual creation of inventories leads to many grammatical and logical errors. It is common to have one sentence to describe an violation, but that sentence is made up of a number of keywords that are not related to each other. This means that the violation cannot be analysed from the text alone. It is therefore common that a recorded infringement has to be replicated manually, which requires more resources and resources. Such tedious resource use requires new tools and techniques to speed up the process.

2. n the case of the CVE, CWE and CVSS descriptors, they are all intended to describe vulnerabilities. Each of these keywords has its own specific function, which is to inform the world about how threatening a breach is and what influences the existence of a breach and how to deal with it. In general terms, each of the descriptors can be nicely organised into a fairly simple structure, the CVE being the parent element that describes the entire breach. The CWE is an additional short description that points in a specific direction to have a general idea of how to deal with the vulnerability. The CVSS describes how bad the breach can be by providing additional calculations and indicating which categories are affected by the vulnerability.

3. There are many different classification methods and techniques, each focusing on a different category or species. The study used linear-based models ranging from Decision Trees, Random Forests, and Logistic Regression to Support Vectors. These models proved to be the most appropriate for classifying big data. For the classification of CWEs, random forest and logistic regression models performed best. These two models kept each other in check in all experiments. Both before and after optimisation, the results of these models were similar. Meanwhile, the same models performed worse when predicting several different values for the CVSS prediction. In contrast, the support vector model showed particularly good results. After optimisation, the likelihood of the model increased locally, even up to 9%. Of course, it should not be excluded that the results of the models change with larger amounts of data. Even though the experiments were carried out on data that represented almost 10% of the total data, the results were very good.

4. Given the good performance of the models, the profiles of 10 randomly selected CVEs were provided as part of the testing of the real entries. These descriptions were not included in the

model training to allow their use as a test. During the validation, hyper-parameter-enhanced Random Forest and Logistic Regression models were used to predict CWEs. The Random Forest model had a very high accuracy in the experiments, while the Logistic Regression model had a slightly lower accuracy but a higher precision compared to the Random Forest model. Each model predicted 10 randomly selected records and the Random Forest model was able to accurately predict 7 out of 10 entries, while the Logistic Regression model predicted only 6. Although the logistic regression had a better average in a general sense, the accuracy of the Random Forests was higher. Meanwhile, for CVSS predictions, the Support Vector Model was superior to all others. In achieving over 88% accuracy, out of 10 entries, it accurately predicted 4 entries, predicted 3 entries with 1 discrepancy, predicted 2 entries with 2 discrepancies, and predicted 3 discrepancies for 1 entry. The predictions in this case are really good because most of the records that are harder to predict are the ones that are rarer. The support vector model predicted the most recurrent vulnerabilities very well in terms of CWE value.

References

[1] Taylor Amerding. "what is cve, it's definition and purpose?". https://www.csoonline.com/article/562175/what-is-cve-its-definition-and-purpose.html.

[2] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[3] LEO BREIMAN. Random forests. *Machine Learning*, 45:5–32, 2001.

[4] The MITRE Corporation. About the cve program. https://www.cve.org/About/Overview.

[5] Philipp Kühn, David N. Relke, and Christian Reuter. Common vulnerability scoring system prediction based on open source intelligence information sources. *Computers & Security*, 131(103286), 2023.

[6] K.W. Lau and Q.H. Wu. Online training of support vector classi er. *Pattern Recognition*, 36:1913–1920, 2003.

[7] D. E. Mann and S.M. Christey. Towards a common enumeration of vulnerabilities, 199. http://cve.mitre.org/docs/cerias.html.

[8] Robert A Martin. Managing vulnerabilities in networked systems. *Computer*, 34(11):32–38, 2001.

[9] Peter Mell, Karen Scarfone, and Sasha Romanosky. A complete guide to the common vulnerability scoring system version 2.0. 2007. https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=51198).

[10] PhD Michael P. LaValley. Logistic regression. *International Conference on Cybernetics and Computational Intelligence*, 117:2395–2399, 2008.

[11] MITRE. Common weakness enumeration — cwe™ a community-developed dictionary of software weakness types. https://cwe.mitre.org.

[12] Mitre.org. Common vulnerabilities and exposures — cve ® the standard for information security vulnerability names. https://cve.mitre.org/docs/cve-intro-handout.pdf.

[13] NIST. computer network attack (cna). https://csrc.nist.gov/glossary/term/computer_network_attack.

[14] NIST. National vulnerability database. https://nvd.nist.gov/.

[15] NIST National Institute of Standards and Technology. National vulnerability database (nvd), 2019. https://nvd.nist.gov/.

[16] A. Saita. Cve-use recommendations open for comment, 2002. http://www.INFOSECURITYMAG.COM/digest/2002/02-04-02.shtml#1b).

[17] scikit learn.org. 1.1. linear models — scikit-learn 1.6.0 documentation.

[18] scikit learn.org. 1.10. decision trees — scikit-learn 1.6.0 documentation.

[19] scikit learn.org. 1.11. ensembles: Gradient boosting, random forests, bagging, voting, stacking — scikit-learn 1.6.0 documentation.

[20] scikit learn.org. 1.4. support vector machines — scikit-learn 1.6.0 documentation.

[21] scikit learn.org. 3.2. tuning the hyper-parameters of an estimator — scikit-learn 1.6.0 documentation.

[22] scikit learn.org. 3.2. tuning the hyper-parameters of an estimator — scikit-learn 1.6.0 documentation.

[23] Arfan Sharif. What is cve? common vulnerabilities & exposures, 2022. https://www.crowdstrike.com/cybersecurity-101/common-vulnerabilities-and-exposures-cve/.

[24] CVSS v3.1 Specification Document. Common vulnerability scoring system version 3.1 - specification document - revision 1.

[25] Song YY and Lu Y. Decision tree methods: applications for classification and prediction. *Machine Learning*, 27:130–135, 2015.