I. J. Engineering and Manufacturing, **2022, 3, 1-13**
Published Online June 2022 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijem.2022.03.01

Estimating Missing Security Vectors in NVD Database Security Reports

Hakan KEKÜL
University of Fırat, Institute of Science, Elazığ Turkey
Sivas Information Technology Technical High School, Diriliş Mahallesi Rüzgarli Sokak No 21 Sivas, Turkey
E-mail: hakankekul@gmail.com

Burhan ERGEN
University of Fırat, Faculty of Engineering, Computer Engineering Department, Elazığ Turkey
E-Mail: bergen@firat.edu.tr

Halil ARSLAN
University of Sivas Cumhuriyet, Faculty of Engineering, Computer Engineering Department, Sivas Turkey
E-Mail: harslan@cumhuriyet.edu.tr

Received: 11 March 2022; Accepted: 28 April 2022; Published: 08 June 2022

Abstract: Detection and analysis of software vulnerabilities is a very important consideration. For this reason, software security vulnerabilities that have been identified for many years are listed and tried to be classified. Today, this process, performed manually by experts, takes time and is costly. Many methods have been proposed for the reporting and classification of software security vulnerabilities. Today, for this purpose, the Common Vulnerability Scoring System is officially used. The scoring system is constantly updated to cover the different security vulnerabilities included in the system, along with the changing security perception and newly developed technologies. Different versions of the scoring system are used with vulnerability reports. In order to add new versions of the published scoring system to the old vulnerability reports, all analyzes must be done manually backwards in accordance with the new security framework. This is a situation that requires a lot of resources, time and expert skill. For this reason, there are large deficiencies in the values of vulnerability scoring systems in the database. The aim of this study is to estimate missing security metrics of vulnerability reports using natural language processing and machine learning algorithms. For this purpose, a model using term frequency inverse document frequency and K-Nearest Neighbors algorithms is proposed. In addition, the obtained data was presented to the use of researchers as a new database. The results obtained are quite promising. A publicly available database was chosen as the data set that all researchers accepted as a reference. This approach facilitates the evaluation and analysis of our model. This study was performed with the largest dataset size available from this database to the best of our knowledge and is one of the limited studies on the latest version of the official scoring system published for classification of software security vulnerabilities. Due to the mentioned issues, our study is a comprehensive and original study in the field.

Index Terms: Software Security, Software Vulnerability, Information security, Text Analysis, Multiclass Classification

1. Introduction

Increasingly and unstoppably, software-supported informatics infrastructures are exposed to cyber threats. Many financial losses are experienced as a result of the abuse of the threatened systems. Vulnerability concept is an error or defect of an information system or component that causes confidentiality, integrity, or usability failure and violates security protocols. The probability and frequency of exploitation of a vulnerability is called a threat. The potential size of the total impact or damage that may result from the exploitation of a vulnerability is called risk [1]. Not all vulnerabilities may have risks to be exploited. In addition, some security vulnerabilities may need to be fixed with urgent patches rather than workarounds. At this stage, it is an important process for institutions/organizations with limited resources to determine which security vulnerabilities they will prioritize. In this process, each vulnerability needs an accurate and standard classification method.

Many solutions have been proposed to the need to classify the risks that may occur in information systems with a framework that is accepted by everyone. Microsoft Threat Scoring System, Symantec Threat Scoring System, CERT Vulnerability Scoring and SANS Critical Vulnerability Analysis Scale are recommended systems [2]. These systems

have been proposed in the context of the institutions in which they were developed. This has caused them to be independent and incompatible with different systems. Their limited nature prevents them from being a standard scale. The Common Vulnerability Scoring System (CVSS) has been proposed as a solution to this situation [3]. Offering an open source and universal framework, CVSS has eliminated the disadvantages of other systems. The Weighted Impact Vulnerability Scoring System (WIVSS) system, which is based on the CVSS system, has been proposed and tried to be developed in the literature [4,5]

Providing a common framework for classifying vulnerabilities, CVSS has been selected as the official scoring system by the National Vulnerability Database (NVD) [6]. As a result, the CVSS system was quickly adopted by the IT security community. The popularity of the system has grown rapidly and with widespread acceptance it has become a standard in this field [7].

The main difference of CVSS from its competitors is that it allows customization and creates a consistent environment that can be used for ranking thanks to its open framework [2]. Thanks to this structure, CVSS is open to development. One year after the first version was introduced in 2004, CVSS 1.0 version was officially announced [3]. The announced version has been updated with regular improvements. As a result of this update, CVSS 2.0 version was introduced in 2007 [1]. For many years CVSS version 2.0 was widely adopted and remained in use. However, the increase in the number of detected security vulnerabilities and changes in security perception necessitated new updates. As a result, CVSS 3.0 version was announced in 2016 [8]. The current CVSS version 3.1 was published in 2019 [6]. Today, CVSS versions 2.0 and 3.1 are used together. CVSS versions 1.0 and 3.0 are no longer used. Therefore, the versions that are in use are included in our study.

Two proposed systems stand out for determining the scores of software vulnerabilities. These are CVSS 2.0 and CVSS 3.1. Generally, calculations are made with these two systems [7]. Vulnerability assessment activities are important in order to decide which vulnerabilities have the highest priority. This process is done manually by experts. This is time consuming and imprecise. In addition, manual detection and classification of security vulnerabilities by experts is costly and contains weaknesses caused by human nature. Vulnerability scoring systems use a computational system consisting of different security metrics developed to classify the vulnerability. Definitions consisting of these metrics are called security vectors.

As mentioned above, scoring systems contain differences and similarities. As a result, scoring methods that stand out with their different features have been proposed. NVD is one of the largest vulnerability databases and is publicly available. The main problem is that NVD experts cannot fully understand the effects of CVSS versions and the security vulnerabilities of the constantly changing scores, which system can produce more accurate results over time. CVSS scoring system is a system that is constantly updated and new versions are released. Currently, the versions officially supported by NVD are CVSS 2.0 and CVSS 3.1. When the security vulnerabilities are examined, it is noticed that there are serious differences between the values obtained as a result of the versions. Also, many vulnerabilities found in NVD lists are missing security vectors.

According to the problems described above, the main purpose of our study is to contribute to the process of evaluating security vulnerabilities by clearly revealing the relationship between scoring systems. It is to provide a complete data set to the use of researchers by completing the missing data in the NVD lists. This can be used to support developers and experts in the field and help them make the right decisions. One of the most important innovations of our study is that it is one of the first studies conducted with CVSS 3.1, the newest version of scoring systems. Moreover, the NVD database will open a large unused data stack to the use of researchers due to missing data.

The main research questions of our study are as follows;

RQ 1. Can the missing data in the NVD data set be reconstructed in a way that can be used by researchers in their studies?

Increasingly, vulnerabilities are becoming an unstructured data pile. Evaluation of vulnerabilities is a time-consuming process as it is manually analyzed by humans. Therefore, experts who fulfill this task give priority to newly dated deficits. Especially when a new version of the vulnerability scoring system is released, there are large gaps in the analysis of old reports retrospectively. Since new security vulnerabilities are constantly entering the system, it does not seem possible to fix this situation manually. However, analyzing the old vulnerabilities is essential for the correct analysis of the new vulnerabilities. In order to solve this problem, it is thought that a model that uses natural language processing techniques and machine learning algorithms, which are used successfully in many fields, can be developed.

RQ 2. Can models developed using technical descriptions of software vulnerability reports be used for this purpose?

When the studies in the literature are examined in order to create the predicted model, it is seen that the technical explanations of the vulnerability reports are generally used. These technical explanations appear to be found in all reports that are missing safety scores and metrics. This means that it can also be used in our study. In addition, it has been determined that the methods that achieve the highest prediction success rate in the literature are Tf-Idf and KNN algorithms. Therefore, in this study, a model using these methods has been proposed.

Other parts of the study are organized as follows. In the second section, the literature that guided the study was examined in detail. In the third section, databases and the data set used are explained in detail, and CVSS is introduced in the fourth section. The research methodology is presented in the fifth section, and the findings are given in detail in the sixth section. In section seven there is a discussion of the findings. Threats to the validity of the study are stated in

the eighth section, and in the last section, the results of the study are presented and future studies are expressed.

2. Related Work

Information technologies have become an indispensable part of human life. In particular, the storage and transmission of financial and personal data via digital systems is an inevitable result of this situation. Personal data trafficking and malicious approaches threaten modern systems as a security problem. This situation shows the importance of ensuring compliance with security principles during the development and use of software, which forms the basic structure of technological systems.

Spanos et al. [7], the main purpose of their research is to develop, accelerate and support the manual procedure of vulnerability characteristic assignment. To achieve this goal, a model was developed that combines text analysis and multi-target classification techniques. This model estimates vulnerability features and then calculates vulnerability severity scores from predicted features. A dataset containing 99,091 records from the vulnerability database in the publicly available National Vulnerability Database (NVD) was used to perform the present research. The number of vulnerabilities in the NVD database has almost doubled since the date of the study. In addition, the 2nd version of the CVSS scoring system is used in the study. The shortcoming of the study is that it does not include new data and does not calculate according to the CVSS 3.1 version, which is the new version of the CVSS scoring system. In addition, the study ignores the relationship between the categories of vulnerabilities. The authors suggest that the classification success can be increased with new data added and different algorithms. Our study will fill the academic gap of this study in terms of the size of the data set it will cover and the different vulnerability scoring systems.

Ghaffarian et al. [10], their work has filled an important gap in the field. They investigated the use of machine learning and data mining algorithms, which have been successfully applied in many fields, in the analysis and discovery of security vulnerabilities. They are conducting a literature review that they divide into four different categories. They suggest the use of deep learning-based methods in their studies.

Morrison et al. [11], aimed to support the use of security measurement in the software lifecycle of the practitioner and researcher by cataloging the security measures presented in the literature, their validity, and the issues they measured. In their study, they moaned 71 articles through a systematic mapping study. For each metric, they identified the subject being measured, how the metric was validated, and how the metric was used. They divided the metrics into categories and gave examples of metrics for each category. As a result, it is recommended that researchers check their vulnerability count definitions when making comparisons between articles. They report there are research opportunities for greater attention to vulnerability scoring. It is expected that our project will include different scoring standards such as CVSS 2.0-3.1 and WIVSS, and will create a different perspective for the academic community and researchers.

Aota et al. [12], used text mining methods to classify security vulnerabilities categorically. The results of their studies show that they achieved a success rate of 96%. One of the results they obtained was that they detected many reports that were assigned to the wrong category. As the reason for this situation, they show that the classification and categorization processing is done manually by humans.

Moore et al. [13], emphasize that it is necessary to quickly determine whether a vulnerability can be exploited. The reason for this is the concerns arising from the rapid increase in the abuse of security vulnerabilities in recent years. They argue that as a result of their foresight, exploitation of a vulnerability can be prevented.

Theisen et al. [14], have examined the problems in processing vulnerability reports, which have become a rapidly growing and large unstructured data pile. They emphasize that it is not possible to evaluate large-scale vulnerability data with statistical methods. They say that there may be points that cannot be discovered if analysis is done using only statistical methods. Their suggestions are the use of machine learning and data mining algorithms that have been successfully applied in solving many problems. They say that in this way, many unexplored points can be revealed.

Fang et al. [15], state that listing the security vulnerabilities published in the NVD database causes time delays due to the institutional structure of NVD. In their study, which emphasizes that only a small portion of security vulnerabilities are exposed to exploitation, they explain the importance of correct and rapid classification of vulnerabilities that can and cannot be exploited.

In this experimental paper by Ruohone [16], he examines the time delays between Common Vulnerability Scoring System (CVSS) information added to CVEs published in NVD. According to empirical results based on regular regression analysis of over eighty thousand archived vulnerabilities, CVSS content is statistically unaffected by time delays. Ruohonen et al. [9], in a different study, observes time delays between the assignment of CVEs and their subsequent appearance in the National Vulnerability Database (NVD). Considering a period from 2008-2016, more than five thousand CVE examples were used to model delays with approximately fifty explanatory metrics. The main reason for these delays is the manual nature of the evaluation process. Our work will speed up the evaluation process and help experts.

Malhotra et al. [17], used the definitions of Apache Tomcat vulnerabilities as input, reducing the data size using chi-square and information gain methods. They estimated the severity of security vulnerabilities with the data sets they created. They used Bagging Technique, Random Forest, Naive Bayes, Support Vector Machine as classification algorithm. They reported that the Naive Bayes algorithm together with the information gain technique produced the best

results and achieved approximately 92% success.

Kekül et al. [18], estimated vulnerability vectors and severity levels using statistical feature extraction methods and different classification algorithms from technical descriptions written in natural language. In their study, they presented a hybrid mix of methods that worked best. The results they obtained are quite promising. It is also one of the first studies to use the current version of scoring systems.

When the above studies are examined, it is seen that a structure similar to the method suggested by the current study is not used. In addition, when the data sets of the studies are examined, it is seen that the missing data are generally removed from the data set. It is seen that the lack of security vectors of new versions in the existing data will increase, especially if the scoring systems are regularly updated. In this study, it is aimed to estimate the missing safety vector data of the scoring systems officially used in NVD. Since this study is the first applied in this field, it is an original study and will fill an important academic gap.

3. Software Vulnerability Databases

Kekül et al. [17] examined in detail and systematically the databases that were used in the literature, that were not out of date, and that were open to access. In this study, they compared seven different databases available to researchers. These databases and comparison criteria are shown in Table 1.

Table 1. Comparison of Vulnerability Databases

Database	Security Score	Solution	Exploit Code	Test	Reporting	Business Model	Data Size *
CVE	×	×	×	×	Everyone	Public	214.988
NVD	✓	×	(✓)	×	Members	Public	170.788
Exploit-DB	×	×	✓	✓	Everyone	Public	44.421
SecurityFocus	×	✓	(✓)	✓	Everyone	Public	102.330
Rapid7	✓	×	×	×	Employees	Commercial	171.816
Snyk	✓	×	×	×	Employees	Commercial	6.012
SARD	×	×	×	✓	Everyone	Public	177.184

✓: There is - ×: None - (✓): For some data there is for some not [19].

* data sizes are the values as of 31.08.2021 and it continues to increase [19].

The databases examined here are; It has been evaluated according to the criteria of whether or not there is a vulnerability scoring, whether it contains a solution method for the vulnerability, whether there are exploit codes, whether or not there is a test for security vulnerabilities, who reports it, business model and data size. Security scores are used to express the impact value of vulnerabilities. When Table 1 is examined, it is seen that the databases from which this score information can be obtained are NVD, Rapid7 and Snyk. It is seen that databases based mainly on CVE lists provide datasets of different sizes due to the methodologies for interpreting and evaluating these data [19].

3.1. Database Used

In this study, the NVD database, which has been making the most reliable vulnerability data available to researchers since 1988, has been used. The NVD includes security checklist references, security-related software flaws, misconfigurations, product names, and impact metrics information. Supported by the National Cyber Security Division of the US Department of Homeland Security. The main task of NVD staff is to analyze the vulnerability lists published in the CVE dictionary. At this stage, they use all the additional data they can collect the explanations and references found in the CVE [20,21]. As a basis for the data published in the NVD database, associated impact metrics (Common Vulnerability Scoring System - CVSS), vulnerability types (Common Vulnerability Enumeration - CWE), applicability statements (Common Platform Enumeration - CPE) and other relevant metadata are added. However, NVD does not perform vulnerability testing for the attributes it assigns. According to new information, CVSS scores and applicability expressions of the data may change [6].

Within the scope of this study, 173,241 records published until 31.10.2021 were selected. In the analyzes made, it was determined that some reports did not have importance scores. Registries with official values were used to train, validate and test the proposed system. As a result, 163,229 records with CVSS 2.0 values and 89,868 records with CVSS 3.1 values were included in the data set to be used to establish the model. Models were applied separately for the two scoring systems. For this, the data set is divided into 70% for training-validation and 30% for testing. The training-validation and test sets consisted of 114,260 – 48,967 records for CVSS 2.0. For CVSS 3.1, 53,907 records are reserved for training-validation and 26,961 records are reserved for testing. The number of records that do not have official values and whose safety vectors will be estimated using the established model is 10.012 for CVSS 2.0 and 83.373 for CVSS 3.1. Table 2 shows an example dataset record. The dataset used in our study has been made available to researchers on Github.

Table 2. A Dataset Record

CVSS 3.1		CVSS2.0	
CVE ID	CVE-2021-22940		
Description	Node.js before 16.6.1, 14.17.5, and 12.22.5 is vulnerable to a use after free attack where an attacker might be able to exploit the memory corruption, to change process behavior.		
Attack Complexity	LOW	Access Vector	NETWORK
Attack Vector	NETWORK	Access Complexity	LOW
Privileges Required	NONE	Authentication	NONE
User Interaction	NONE	NaN	
Scope	UNCHANGED	Nan	
Confidentiality Impact	NONE	NONE	
Integrity Impact	HIGH	PARTIAL	
Availability Impact	NONE	NONE	
Exploitability Score	3.9	10	
Impact Score	3.6	2.9	
Base Score	7.5	5.0	
Severity	CRITICAL	HIGH	
Vector String	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	AV:N/AC:L/Au:N/C:P/I:P/A:P	

4. Common Vulnerability Scoring System – CVSS

The software vulnerability scoring system is explained in the CVSS Introduction. CVSS officially uses three different metric groups in both versions. These metric groups are called Basic, Temporal, and Environmental. Each metric group has an independent security vector and calculation. However, the Base Score does not change over time. In addition, this score is usually created by the software developer company or the organizations that maintain the software product for which they are authorized. This ensures that the Base scores are unchanged over time and produce a standard value for all platforms. Therefore, only Core Metric values are published [9].

With the CVSS 2.0 base set of metrics, it specifies the characteristics of a software vulnerability that will not change over time as the user environment or platform changes. This metric group has six independent metric groups. Although no sub-division is clearly specified in this metric group, the metrics that determine how the access to the vulnerability is provided and whether additional conditions are needed to exploit this vulnerability are Access Vector (AV), Access Complexity (AC), and Authentication (Au). The other three metrics are called Impact metrics, which indicate the potential effects of the vulnerability and the potential harm it can cause when exploited. These are called Confidentiality Impact (CI), Integrity Impact (II) and Availability Impact (AI), which indicates different aspects of possible effects independently of each other. The standalone effect means that while a security vulnerability causes loss of usability, it may not cause any privacy impact [1].

In CVSS 3.1, as in previous versions, basic metrics are a group of metrics that do not change over time and in different environments. Thanks to this feature, it provides general information about the vulnerability. Subgroups that are not explicitly mentioned in CVSS version 2.0 are officially defined in CVSS version 3.1. Basic metrics are divided into two as Exploitability and Impact metrics. It also has an independent metric called Scope. Availability metrics are Attack Vector (AV), Attack Complexity (AC), Privileges Required (PR), and User Interaction (UI). Impact Metrics are Confidentiality (C), Integrity (I), and Availability (A) [9].

Table 3. CVSS 2.0 Base Metrics Groups [8]

Vector	Description	Values	Weights	Category
Access Vector	Specifies the method by which the vulnerability can be exploited. The further the attacker is, the greater the value.	Local (L) Adjacent Network (A) Network (N)	0.395 0.646 1.0	Exploitability
Access Complexity	It is a measure of the complexity of the attack required to exploit the vulnerability. The lower the complexity, the higher the value.	High (H) Medium (M) Low (L)	0.35 0.61 0.71	Exploitability
Authentication	The authentication value required to exploit the vulnerability. The lower the value, the higher the score.	Multiple (M) Single (S) None (N)	0.45 0.56 0.704	Exploitability
Confidentiality Impact	Expresses the effect of exploiting the vulnerability on the privacy of the system. Scores the extent to which privacy will be affected as a result of exploitation in the system. The higher the effect, the higher the score.	None (N) Partial (P) Complete (C)	0.0 0.275 0.660	Impact
Integrity Impact	It refers to the effect a successful attack will have on system integrity. Increasing the integrity effect increases its score.	None (N) Partial (P) Complete (C)	0.0 0.275 0.660	Impact
Availability Impact	It specifies how the resources used by the system are affected in case of a possible attack. The higher the effect, the higher the vulnerability score.	None (N) Partial (P) Complete (C)	0.0 0.275 0.660	Impact

Table 4. CVSS 3.1 Base Metrics Groups [10]

Vector	Description	Values	Weights	Category
Attack Vector	Describes the situation required for the attacker to exploit the vulnerability. The further it is possible to benefit from the vulnerable system, the more points increase.	Network (N) Adjacent (A) Local (L) Physical (P)	0.85 0.62 0.55 0.2	Exploitability
Attack Complexity	It refers to the external conditions necessary for the attacker to perform a successful abuse. The score has the highest value for low complexity attacks.	Low (L) High (H)	0.77 0.44	Exploitability
Privileges Required	It refers to the level of privilege the attacker must have in order to exploit the vulnerability. For a non-privileged vulnerability, the value is the highest. ***if Scope/Modified Scope is Changed**	None (N) Low (L) High (H)	0.85 0.62 or 0.68* 0.27 or 0.5*	Exploitability
User Interaction	Indicates whether any user's action is required to exploit the vulnerability. If the action of a different user is not required, the score is high.	None (N) Required (R)	0.85 0.62	Exploitability
Confidentiality Impact	Expresses the effect of exploiting the vulnerability on the privacy of the system. Scores the extent to which privacy will be affected as a result of exploitation in the system. The higher the effect, the higher the score.	High (H) Low (L) None (N)	0.56 0.22 0.0	Impact
Integrity Impact	It refers to the effect a successful attack will have on system integrity. Increasing the integrity effect increases its score.	High (H) Low (L) None (N)	0.56 0.22 0.0	Impact
Availability Impact	It specifies how the resources used by the system are affected in case of a possible attack. The higher the effect, the higher the vulnerability score.	High (H) Low (L) None (N)	0.56 0.22 0.0	Impact
Scope (S)	Expresses whether components other than the vulnerable component affect the resources it uses. If a scope change occurs, the score will increase.	Unchanged(U) Changed (C)		

The CVSS v2.0 equations are defined below.

$$BaseScore = RoundTo1Decimal((0.6 * Impact + 0.4 * Exploitability - 1.5) * f(Impact)) \quad (1)$$

$$Impact = 10.41 * \left(1 - (1 - ConfImpact) * (1 - IntegImpact) * (1 - AvailImpact)\right) \quad (2)$$

$$Exploitability = 20 * AccessComplexity * Authentication * AccessVector \quad (3)$$

$$f(Impact) = 0 \; if \; Impact = 0; \; 1.176 \; otherwise \quad (4)$$

The CVSS v3.1 equations are defined below.

$$If \; (Impact \; sub \; score <= 0) \quad 0 \; else,$$

$$Scope \; Unchanged \quad Base \; Score = Roundup(Minimum[(Impact + Exploitability), 10]) \quad (5)$$

$$Scope \; Changed \quad Base \; Score = Roundup(Minimum[1.08 \times (Impact + Exploitability), 10]) \quad (6)$$

$$Scope \; Unchanged \; 6.42 \times ISCBase \quad (7)$$

$$Scope \; Changed \; 7.52 \times [ISCBase - 0.029] - 3.25 \times [ISCBase - 0.02]^{15} \quad (8)$$

$$ISCBase = 1 - [(1 - ImpactConf) \times (1 - ImpactInteg) \times (1 - ImpactAvail)] \quad (9)$$

$$Exploitability = 8.22 \times AttackVector \times AttackComplexity \times PrivilegeRequired \times UserInteraction \quad (10)$$

The basic scores of the vulnerabilities detected by the vulnerability scoring systems, whose formulas and metric values are given above, are calculated. The calculated scores are a value between 0.0 – 10.0. These found values are classified using the qualitative severity rating. Tables 5 and 6 show the scales of CVSS 2.0 and 3.1, respectively.

Table 5. Qualitative severity rating scale for v2.0 [6]

Severity Rating	Base Score Range
Low	0.0 - 3.9
Medium	4.0 - 6.9
High	7.0 – 10.0

Table 6. Qualitative severity rating scale for v3.1 [6]

Severity Rating	Base Score Range
None	0.0
Low	0.1 - 3.9
Medium	4.0 - 6.9
High	7.0 - 8.9
Critical	9.0 - 10.0

5. Research Methodology

The main purpose of this study is to estimate missing security vectors found in vulnerability reports. Estimating software vulnerability vectors from the limited technical descriptions of security reports is a difficult task. The aim of our work is to find the best solution to this difficult problem. For this, text analysis and multi-class classification algorithms are used. The technical descriptions of the vulnerability reports appear to be documents written by experts and no longer than a few sentences, as indicated in Table 2. Especially in the old reports, the fact that the explanations are very short texts makes the estimation of security vectors very difficult. However, there is no data that can be used in the implementation of the proposed methodology other than the technical explanations of the safety reports. In addition, technical explanations are written in natural language and are a large chunk of unstructured data. The first thing to do is to convert this large chunk of unstructured data into a structured data set. The basic steps required in the classification of text data also apply to the proposed method. These are preprocessing, creation of text vectors, and classification [22]. In Figure 1, all stages of the proposed method are shown in detail.

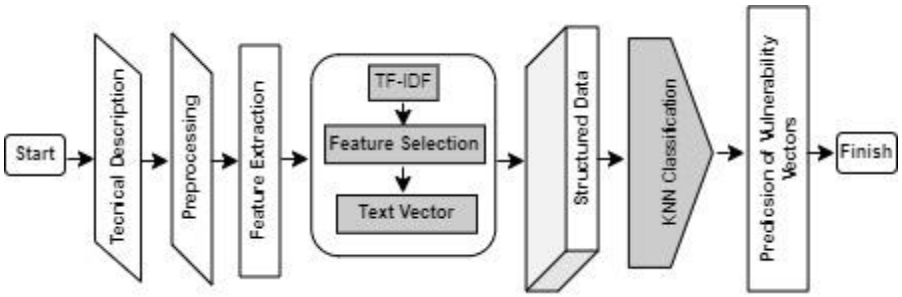

Fig.1. The general structure of the proposed methodology.

Within the scope of our study, the non-profit NVD database funded by the US Department of Homeland Security was selected. In addition, this database uses CVSS, a security scoring system generally accepted by IT professionals. The NVD database was preferred in this study, as it is open to the public and provides a generally accepted standard framework. Moreover, this allows us to compare the results of our study with different studies. As can be seen in Figure 1, first of all, the technical explanations of the vulnerability reports need to be pre-processed. At this stage, the NVD database provides all reports in JSON and XML format.

At the first stage, the technical explanations of the reports obtained from expressions such as stop words, punctuation marks, etc. are cleared. The cleaned data is not directly used in classification algorithms. At this stage, technical descriptions need to be represented as a numerical vector using natural language processing methods. Many different methods have been proposed in the literature for this process. However, Tf-Idf method was preferred within the scope of this study. At this stage, it has been seen that a 300-word vector is the most optimal value in experimental studies and literature analysis. For this reason, all technical explanations are represented by the 300 words with the highest frequency values[23]. Our model, which was trained with the data set whose document vectors were created, then estimated the missing data in our data set. The data set with the missing values and the vulnerability vectors predicted by the proposed model has been presented to the use of the researchers.

5.1. Text Preprocessing

Text preprocessing is the process of cleaning text data, removing noise and revealing important features. When the effect of text preprocessing on the classification performance is examined, it has been revealed that it is as important as the other stages [24]. This phase starts with cleaning up punctuation marks, extra spaces and special characters in the text data. The second preprocessing step is parsing the words. This process is called tokenization. With this process, it is aimed to examine the words in the sentence one by one and to distinguish meaningful or meaningless items [25], [26]. The next step is to remove the words, which are expressed as stopwords in documents reserved for tokens, and which are in every language but do not have significant meaning, from the data set. After this stage, the data set is made up of only meaningful word groups and is free from many noises. However, it is not yet in a completely clean structure. This is because a word can be used in different forms in many languages. These words, which have the same root but appear

in a different structure, should be reduced to their roots and analyzed as the same word. This process is called lemmatization [27]. After completing all the steps, the pre-processing phase is completed.

5.2. Creating Text Vectors

The preprocessed text data has to be converted into a structured data that can be used in machine learning algorithms. This step is also called as feature inference in different classification problems. In text analysis, on the other hand, it is the stage where textual data is converted into numerical vectors. Feature extraction methods in text classification can be divided into two groups. These can be expressed as statistical methods and deep learning-based word embedding methods [28]. Bag of words (BoW) [29], term frequency inverse document frequency (TF-IDF)) [30] and ngram [31] can be given as examples of statistical methods in general.

Statistical methods have been successfully applied in the literature to solve many problems and are still used. Basically, they have two disadvantages. These are the fact that they only focus on the frequencies of the words in the document and corpus during the vector creation phase. As a result, it causes the loss of semantic relations and morphological connections between words. Another disadvantage is that the computational costs are high. Despite their disadvantages, they can be used to compare the performance of classification problems [22]. In this study, TF-IDF was chosen from the traditional statistical approaches, which was the most successful method used in the literature for classification of vulnerability vectors [18].

5.3. K-Nearest Neighbors (KNN) Classification Algorithm

Classification of text data, which has become a huge data pile today, is an important research area. In particular, some criteria were determined in the selection of the algorithm used in the classification of the proposed model. First of all, since the problem of interest has a multi-class structure and high computation is required, algorithms that can make multi-class classification within themselves have been tried to be selected. As another criterion, an algorithm that has been used before, especially in text classification problems and classification of security vulnerabilities, has been included in the model. For the classification stage of the methodology, the K-Nearest Neighbors (KNN) [32] algorithm, which is a built-in strategy grouped estimator with multiple learning support, was chosen. While creating the classification model, the cross validation method was used. A value of 10 was chosen as the cross validation parameter. Cross validation is a statistical analysis that checks the accuracy of the model on independent data sets [33]. Its main purpose is to estimate the accuracy of a system in practice. Thus, overfitting or selection bias problems can be detected by determining the sensitivity of the model to new data [34].

The KNN algorithm is an easy-to-apply and high-performance method that is widely used in solving data mining and statistical problems [35]. The algorithm calculates which class it belongs to among the k nearest neighbors in the vector space given during the training phase. While elements belonging to the same class have high similarity values, different classes have low similarity values [36]. Although the algorithm has advantages such as simplicity and accuracy, it has high computational costs [37]. It is one of the algorithms frequently used in text classification [38]. In addition, as far as it is known in the literature, it is the classification algorithm that achieves the highest classification values in the classification of security vulnerabilities [18].

5.4. Validation

Used in multi-class classification problems and the following well-known evaluation criteria were used to evaluate the proposed methodology [39].

Accuracy: Refers to the ratio of the number of correctly predicted tags for a sample set to the total data set.

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \tag{11}$$

Recall: It is a metric that shows how much of the transactions that need to be positive predicted are positive predicted.

$$Recall = \frac{TP}{TP+FN} \tag{12}$$

Precision: It shows how many of the values we guess positive are actually positive.

$$Precision = \frac{TP}{TP+FP} \tag{13}$$

F1 Score: It shows the harmonic mean of the Precision and Recall values.

$$F_1 = 2 * \frac{Precision*Recall}{Precision+Recall} \tag{14}$$

Precision, recall and f1 score are more reliable measurements than accuracy. In addition, this set of measurements is in line with the suggestions of other researchers working on the evaluation of multi-class classification models [40].

6. Results

The data set used in our study is presented in detail in Section 3. There are approximately 175,000 vulnerability reports in our dataset. The basic approach of the proposed methodology is to calculate the missing vulnerability vector values of vulnerability reports. We present some statistical information to better understand our results. Kekul et al. [18], the size of the dataset they used in their study is approximately 165,000. According to the authors, when the statistical properties of the datasets are examined, it is seen that the vulnerability vectors become complex. In their study, which they evaluated separately for CVSS versions, they state that there are 355 different vector numbers for CVSS 2.0 and that vectors with a frequency value of 1% and above are only the first 20 most frequently used vectors. Also, the cumulative sum of these vectors is about 83%. The frequency distribution of the vectors is in the range of 0.01% - 14.00%. They emphasize the same research that for CVSS 3.1 vector values, there are a total of 1322 vectors out of 2592 probabilities, which are distributed in the range of 0.01% - 8.85% and cumulative sums are 60%. In addition, only the first 22 of the vectors have a frequency value of 1% or more. This illustrates the fact that most of the vulnerability vectors are encountered with very low frequencies. They describe the problem of estimating vulnerability vectors as a rather difficult task [7]. This clearly shows that the data set has turned into a difficult problem to construct a prediction model.

6.1 Traning Results

For a more accurate analysis of the results of the method proposed in this study, we present the training results of the model. 70% of our dataset detailed in Section 3.1 was used for training purposes. Moreover, the accuracy of the model was checked by using the cross validation technique during the training phase. Cross validation value was chosen as 10. One of the first metrics to look at in the evaluation of classification problems is accuracy. Accuracy corresponds to the ratio of correct classifications to the total number of samples. In multi-class classification problems, the uneven distribution of classes can affect the results in the direction of the performance of the dominant class when evaluating the models. This is also true for our problem. Although Accuracy is an important indicator, it may not be sufficient on its own for multi-class problems [41]. For this reason, our model was also evaluated with macro precision, recall and f1 score scales, which were created with the averages of the performances of each subclass. Values are presented as percentages.

6.2 Results of Vulnerability Vectors

At this stage, we present the classification results obtained with the proposed method in detail. As detailed in Tables 4 and 5, security vectors of vulnerability scoring systems consist of 6 and 8 metrics for different versions. Each metric value consists of different numbers of subclasses. This creates a situation where there are 729 different possibilities for CVSS 2.0 and 2592 different possibilities for CVSS 3.1. In the previous section, the difficulty of estimation of this multi-probability structure was explained. The values of all predicted metrics of CVSS 2.0 security vectors are given in Table 8. Results were evaluated according to Accuracy, Precision, Recall and F1 Score scales. Precision and recall values specified in the tables are macro values found by averaging the performance of each class. In addition, the F1-score value is calculated with the harmonic average of the precision and recall values.

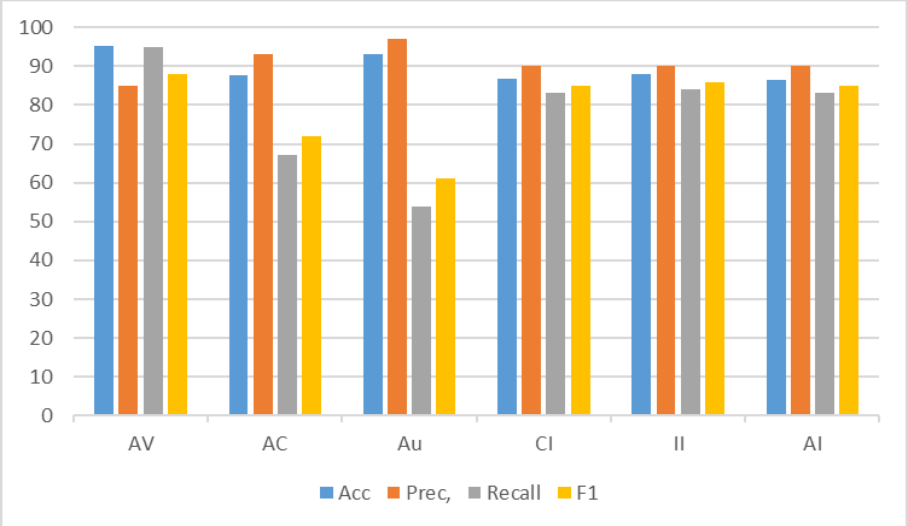

Fig. 2. CVSS 2.0 Security Vector prediction results

Table 7. The results of the CVSS 2.0 model with the highest predictive value

Metric	Acc.	Prec.	Recall	F1
AV	95,32	85	95	88
AC	87,8	93	67	72
Au	93,11	97	54	61
CI	86,78	90	83	85
II	88,07	90	84	86
AI	86,59	90	83	85

When Table 7 and Figure 2 are examined, it is seen that the proposed method produces very successful results for CVSS 2.0. Especially in AV and Au metric values, a very high estimation result of 95% and 93%, respectively, was obtained. For other metric values, satisfactory results were obtained between 87% and 88%. This shows that the vulnerability vector that will be created for missing data as a result of the estimations will be very sensitive. With a prediction performance of these ratios, 10,012 CVSS 2.0 records missing safety vector values were predicted.

Table 8. The results of the CVSS 3.1 model with the highest predictive value

Metric	Acc.	Prec.	Recall	F1
AV	92,45	98	70	78
AC	95,48	92	74	80
PR	88,11	92	73	79
UI	93,31	95	91	92
S	96,11	98	89	92
CI	88,16	86	86	86
II	88,23	88	87	87
AI	91,29	87	74	78

When Table 8 and Figure 3 are examined, it is seen that the proposed method is quite successful in estimating the metrics of the CVSS 3.1 scoring system. The scope metric was the most successful prediction of the model with a predictive value of 96.11%. In addition, the AC metric was estimated with a very high value of 95.48%. Among the predicted metrics, AV, UI and AI metrics were estimated with 92.45%, 93.31% and 91.29% predictive values, respectively. As can be seen, the prediction performance of five of the eight metrics is above 90%. The prediction performance of other vulnerability metrics, on the other hand, is very close to this value, although it is below 90%. PR was calculated as 88.11%, CI as 88.16% and II as 88.23%. When all the values are examined together, it is seen that the results obtained are quite promising.

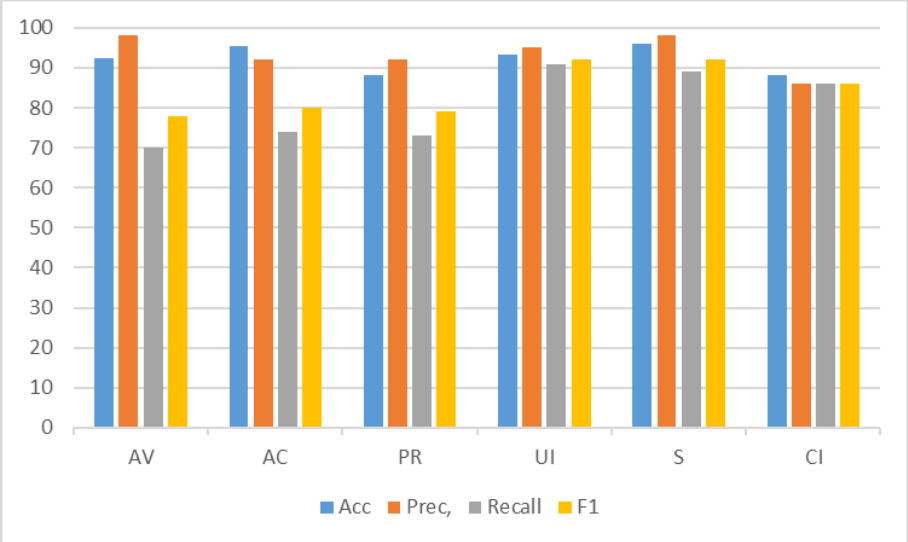

Fig. 3 CVSS 3.1 Security Vector prediction results

7. Discussion

The number of software security vulnerabilities has been increasing in recent years. As a result of this situation, there has been an increase in the abused systems recently [42]. It is important to share software vulnerabilities with industry and researchers in order to be able to analyze them correctly and prevent their recurrence. By using past experiences, it can be ensured that the systems to be developed in the future will be more secure. However, there are problems due to the size of the security vulnerabilities accumulated for many years and their analysis by people. The

biggest problem at this stage is that the first analyzes cannot fully predict potential threats or time delays are experienced. Considering that the vast majority of security vulnerabilities are exploited within the first two weeks, it is clear that there is no compensation for the errors to be experienced in these transactions. In addition, this manually operated procedure makes it impossible to reanalyze old reports by using new features in scoring systems. Even in the oldest version CVSS 2.0 system, which is one of the scoring systems we examined within the scope of this study, no information was entered in the 10.012 security report. For CVSS 3.1, the last version in use, this figure is 83,373. This corresponds to almost 50% of all deficits. This means ignoring a lot of knowledge and experience in software vulnerability reports. When the findings of the studies are examined, it is seen that the model established with the proposed natural language processing methods and classification algorithms produces very successful results. However, for an accurate analysis, it is necessary to discuss the vulnerability scoring systems, which have quite different security schemes and classifications, separately.

Older than software vulnerability scoring systems, CVSS 2.0 has a larger data size and fewer missing values. The security vector consists of 6 different metrics. Of these metrics, AV had the highest accuracy with 95.32% prediction success. According to the prediction success, AI was the metric with the lowest predictive value with 86.59%. When other metrics are ordered according to forecast success; Au 93.11%, II 88.07%, AC 87.8% and CI 86.78%. The security vector of CVSS 3.1, the latest version of the software vulnerability scoring system, has 8 different metrics. When these metrics are examined, it is seen that the highest prediction success belongs to the S metric with 96.11%. This metric was followed by AC, UI, AV and AI metrics with values of 95.48%, 93.31%, 92.45%, and 91.29%, respectively. II, CI and PR metrics, which have very close estimation results, have values of 88.23%, 88.16%, and 88.11%, respectively. These results show us that when the two versions of our model are included, the highest predicted metric with 96.11% is the S metric of the CVSS 3.1 version. The lowest accuracy value is the AI metric of CVSS 2.0 version with 86.59%.

9. Conclusions

It is not possible for a software product to be completely error-free. For this reason, each new information system released has the potential for different security gaps. The number of software vulnerabilities reported and published as a result of increasing software products in recent years has been increasing rapidly. Manual analysis of detected software vulnerabilities becomes difficult. In addition, the perception of security changes over time and newly developed technologies bring different security vulnerabilities. As a result of this situation, the systems used in the analysis and classification of software security vulnerabilities are constantly updated. The backward reflection of these processes made with human power causes negligence as it requires too much effort. This situation is exacerbated with each new software vulnerability scoring system version or method released. However, transferring meaningful information and experience in old software vulnerabilities can guide the analysis of new vulnerabilities. Moreover, it will speed up the process by increasing the sensitivity of the analysis. The findings obtained as a result of the study can be used to make the development processes of software more secure.

With this study, a meaningful and measurable model that will fill the gap in the field has been put forward for the first time. It is seen that very successful results have been obtained with the proposed model. The technical descriptions of the software vulnerability reports were converted into numerical vectors using the TF-IDF method. The obtained values were used as input and a multi-class classification was carried out. KNN algorithm was used for classification. The results found indicate very high accuracy rates. The lowest value for the metrics of both versions of vulnerability scoring systems was 86.59%. This shows that the security vectors created as a result of the prediction can be estimated with very high precision. The average accuracy value calculated for CVSS 2.0 of the new data set created by the estimation of missing data was found to be 89.61. According to the calculations made for CVSS 3.1, the average accuracy value was 91.49%. These values show that the new data set created is reliable and usable in many respects. The predicted values for CVSS 2.0 are, in order; AV - 95.32%, Au - 93.11%, II - 88.07%, AC - 87.8%, CI - 86.78% and AI - 86.59%. The predicted values for CVSS 3.1 are, in order; S - 96.11%, AC - 95.48%,UI - 93.31%,AV - 92.45%,AI - 91.29%,II - 88.23%,CI - 88.16% and PR - 88.11%

Today, the software vulnerability data set has gained a very large size. As a result, it becomes difficult to reanalyze old reports with newly published safety scoring systems. This situation causes incomplete data to occur in many reports. This problem gets worse with each new version of the vulnerability scoring system released. Especially in recent years, the analysis of software security vulnerabilities with machine learning and data mining techniques has increased. Generally, missing data is ignored in these studies. The number of vulnerability reports with the latest software vulnerability scoring system is about half of the total reports. This situation reveals that there is quite a large amount of data that researchers ignore and do not include in their analysis. With the proposed method, the missing vulnerability vectors were obtained. A new data set was created to present the obtained safety vector values to the use of researchers. The new dataset is available on GitHub (https://github.com/hakankekul/CvssDataSet). Especially recently, all researchers recommend that studies on deep learning be conducted. We are planning to expand our work with deep learning methods in our next studies. We also want to improve our work with new models that use different feature extraction and classification algorithms.

Funding

This study is supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) with project number 121E298.

References

[1] P. Mell, K. Scarfone, and S. Romanosky, "A Complete Guide to the Common Vulnerability Scoring System Version 2.0," *FIRSTForum of Incident Response and Security Teams*, 2007. https://www.first.org/cvss/cvss-v2-guide.pdf (accessed Jan. 01, 2021).
[2] V.-V. Patriciu, I. Priescu, and S. Nicolaescu, "Security metrics for enterprise information systems," *J. Appl. Quant. Methods*, vol. 1, no. 2, pp. 151–159, 2006.
[3] M. Schiffman and C. I. A. G. Cisco, "A Complete Guide to the Common Vulnerability Scoring System (CVSS) v1 Archive," 2005. https://www.first.org/cvss/v1/guide (accessed Jan. 01, 2021).
[4] G. Spanos, A. Sioziou, and L. Angelis, "WIVSS: A New Methodology for Scoring Information Systems Vulnerabilities," in *Proceedings of the 17th Panhellenic Conference on Informatics*, 2013, pp. 83–90, doi: 10.1145/2491845.2491871.
[5] G. Spanos and L. Angelis, "Impact metrics of security vulnerabilities: Analysis and weighing," *Inf. Secur. J. A Glob. Perspect.*, vol. 24, no. 1–3, pp. 57–71, 2015.
[6] NVD, "NVD," *National Vulnerability Database*, 2020. https://nvd.nist.gov (accessed Jul. 25, 2020).
[7] G. Spanos and L. Angelis, "A multi-target approach to estimate software vulnerability characteristics and severity scores," *J. Syst. Softw.*, vol. 146, pp. 152–166, 2018, doi: 10.1016/j.jss.2018.09.039.
[8] "Common Vulnerability Scoring System v3.0: User Guide." https://www.first.org/cvss/v3.0/user-guide (accessed Jan. 01, 2021).
[9] "Common Vulnerability Scoring System v3.1: User Guide." https://www.first.org/cvss/v3.1/user-guide (accessed Jan. 01, 2021).
[10] S. M. Ghaffarian and H. R. Shahriari, "Software vulnerability analysis and discovery using machine-learning and data-mining techniques: A survey," *ACM Comput. Surv.*, vol. 50, no. 4, 2017, doi: 10.1145/3092566.
[11] P. Morrison, D. Moye, R. Pandita, and L. Williams, "Mapping the field of software life cycle security metrics," *Inf. Softw. Technol.*, vol. 102, no. July 2017, pp. 146–159, 2018, doi: 10.1016/j.infsof.2018.05.011.
[12] M. Aota, H. Kanehara, M. Kubo, N. Murata, B. Sun, and T. Takahashi, "Automation of Vulnerability Classification from its Description using Machine Learning," in *2020 IEEE Symposium on Computers and Communications (ISCC)*, 2020, pp. 1–7, doi: 10.1109/ISCC50000.2020.9219568.
[13] T. W. Moore, C. W. Probst, K. Rannenberg, and M. van Eeten, "Assessing ICT Security Risks in Socio-Technical Systems (Dagstuhl Seminar 16461)," *Dagstuhl Reports*, vol. 6, no. 11, pp. 63–89, 2017, doi: 10.4230/DagRep.6.11.63.
[14] C. Theisen and L. Williams, "Better together: Comparing vulnerability prediction models," *Inf. Softw. Technol.*, vol. 119, no. August 2019, 2020, doi: 10.1016/j.infsof.2019.106204.
[15] Y. Fang, Y. Liu, C. Huang, and L. Liu, "Fastembed: Predicting vulnerability exploitation possibility based on ensemble machine learning algorithm," *PLoS One*, vol. 15, no. 2, pp. 1–28, 2020, doi: 10.1371/journal.pone.0228439.
[16] J. Ruohonen, "A look at the time delays in CVSS vulnerability scoring," *Appl. Comput. Informatics*, vol. 15, no. 2, pp. 129–135, 2019, doi: 10.1016/j.aci.2017.12.002.
[17] R. Malhotra and Vidushi, "Severity Prediction of Software Vulnerabilities Using Textual Data," in *Proceedings of International Conference on Recent Trends in Machine Learning, IoT, Smart Cities and Applications*, 2021, pp. 453–464.
[18] H. Kekül, B. Ergen, and H. Arslan, "A multiclass hybrid approach to estimating software vulnerability vectors and severity score," *J. Inf. Secur. Appl.*, vol. 63, p. 103028, 2021, doi: https://doi.org/10.1016/j.jisa.2021.103028.
[19] H. Kekül, B. Ergen, and H. Arslan, "Yazılım Güvenlik Açığı Veri Tabanları," *Avrupa Bilim ve Teknol. Derg.*, no. 28, pp. 1008–1012, 2021.
[20] C. W. Samuel Ndichu, Sylvester McOyowo, Henry Okoyo, "A Remote Access Security Model based on Vulnerability Management," *Int. J. Inf. Technol. Comput. Sci.*, vol. 12, no. 5, pp. 38–51, 2020, doi: 10.5815/ijitcs.2020.05.03.
[21] H. Kekül, B. Ergen, and H. Arslan, "A New Vulnerability Reporting Framework for Software Vulnerability Databases," *Int. J. Educ. Manag. Eng.*, vol. 11, no. 3, pp. 11–19, 2021, doi: 10.5815/ijeme.2021.03.02.
[22] A. Fesseha, S. Xiong, E. D. Emiru, M. Diallo, and A. Dahou, "Text Classification Based on Convolutional Neural Networks and Word Embedding for Low-Resource Languages: Tigrinya," *Information*, vol. 12, no. 2, 2021, doi: 10.3390/info12020052.
[23] Z. Yin and Y. Shen, "On the dimensionality of word embedding," *arXiv Prepr. arXiv1812.04224*, 2018.
[24] A. K. Uysal and S. Gunal, "The impact of preprocessing on text classification," *Inf. Process. Manag.*, vol. 50, no. 1, pp. 104–112, 2014, doi: https://doi.org/10.1016/j.ipm.2013.08.006.
[25] G. Gupta and S. Malhotra, "Text document tokenization for word frequency count using rapid miner (taking resume as an example)," *Int. J. Comput. Appl*, vol. 975, p. 8887, 2015.
[26] T. Verma, R. Renu, and D. Gaur, "Tokenization and filtering process in RapidMiner," *Int. J. Appl. Inf. Syst.*, vol. 7, no. 2, pp. 16–18, 2014.
[27] A. A. Jalal and B. H. Ali, "Text documents clustering using data mining techniques.," *Int. J. Electr. Comput. Eng.*, vol. 11, no. 1, 2021.
[28] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," *Information*, vol. 10, no. 4, p. 150, 2019.
[29] Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model: a statistical framework," *Int. J. Mach. Learn. Cybern.*, vol. 1, no. 1–4, pp. 43–52, 2010.
[30] A. Aizawa, "An information-theoretic perspective of tf--idf measures," *Inf. Process. Manag.*, vol. 39, no. 1, pp. 45–65, 2003.

[31] S. Banerjee and T. Pedersen, "The design, implementation, and use of the ngram statistics package," in *International Conference on Intelligent Text Processing and Computational Linguistics*, 2003, pp. 370–381.
[32] E. Fix, *Discriminatory analysis: nonparametric discrimination, consistency properties*. USAF school of Aviation Medicine, 1951.
[33] R. Kohavi and others, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, 1995, vol. 14, no. 2, pp. 1137–1145.
[34] G. C. Cawley and N. L. C. Talbot, "On over-fitting in model selection and subsequent selection bias in performance evaluation," *J. Mach. Learn. Res.*, vol. 11, pp. 2079–2107, 2010.
[35] Y. Yang, "An evaluation of statistical approaches to text categorization," *Inf. Retr. Boston.*, vol. 1, no. 1, pp. 69–90, 1999.
[36] X. Deng, Y. Li, J. Weng, and J. Zhang, "Feature selection for text classification: A review," *Multimed. Tools Appl.*, vol. 78, no. 3, pp. 3797–3816, 2019.
[37] Z. Chen, L. J. Zhou, X. Da Li, J. N. Zhang, and W. J. Huo, "The Lao Text Classification Method Based on KNN," *Procedia Comput. Sci.*, vol. 166, pp. 523–528, 2020, doi: https://doi.org/10.1016/j.procs.2020.02.053.
[38] Y. Tan, "An Improved KNN Text Classification Algorithm Based on K-Medoids and Rough Set," in *2018 10th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, 2018, vol. 01, pp. 109–113, doi: 10.1109/IHMSC.2018.00032.
[39] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Inf. Process. Manag.*, vol. 45, no. 4, pp. 427–437, 2009, doi: https://doi.org/10.1016/j.ipm.2009.03.002.
[40] C. Bielza, G. Li, and P. Larrañaga, "Multi-dimensional classification with Bayesian networks," *Int. J. Approx. Reason.*, vol. 52, no. 6, pp. 705–727, 2011, doi: https://doi.org/10.1016/j.ijar.2011.01.007.
[41] D. Ballabio, F. Grisoni, and R. Todeschini, "Multivariate comparison of classification performance measures," *Chemom. Intell. Lab. Syst.*, vol. 174, pp. 33–44, 2018, doi: https://doi.org/10.1016/j.chemolab.2017.12.004.
[42] L. P. Kobek, "The State of Cybersecurity in Mexico: An Overview," *Wilson Centre's Mex. Institute, Jan*, 2017.

Authors' Profiles

Hakan Kekül is currently working as a teacher at Sivas Information Technology Technical High School. In 2006, he received his undergraduate degree from Sakarya University, Department of Electronics and Computer Education. In 2006, he received his bachelor's degree in Computer Engineering from Cumhuriyet University. In 2017, he received his Master's degree from Cumhuriyet University. Since 2018, he is a PhD candidate at Fırat University, Department of Computer Engineering.

Burhan Ergen is currently Prof. in Department of Computer Engineering at Fırat University. He received his BS degree in Electronics Engineering from Karadeniz Technical University in 1993. He received his master's degree from Karadeniz Technical University in 1996 and his doctorate degree from Fırat University in 2004. He is currently working at Fırat University.

Halil Arslan is currently a faculty member at Cumhuriyet University Computer Engineering Department. He received his undergraduate, graduate and doctorate degrees from Sakarya University Electronics and Computer Education Department in 2006, 2008 and 2016, respectively. He is currently working at Cumhuriyet University

How to cite this paper: Hakan KEKÜL, Burhan ERGEN, Halil ARSLAN, " Estimating Missing Security Vectors in NVD Database Security Reports", International Journal of Engineering and Manufacturing (IJEM), Vol.12, No.3, pp. 1-13, 2022. DOI: 10.5815/ijem.2022.03.01

