I. J. Computer Network and Information Security, **2022, 4, 27-42**
Published Online August 2022 in MECS (http://www.mecs-press.org/)
DOI:10.5815/ijcnis.2022.04.03

A Multiclass Approach to Estimating Software Vulnerability Severity Rating with Statistical and Word Embedding Methods

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: 28 January 2022; Revised: 01 March 2022; Accepted: 23 March 2022; Published: 08 August 2022

Abstract: The analysis and grading of software vulnerabilities is an important process that is done manually by experts today. For this reason, there are time delays, human errors, and excessive costs involved with the process. The final result of these software vulnerability reports created by experts is the calculation of a severity score and a severity rating. The severity rating is the first and foremost value of the software's vulnerability. The vulnerabilities that can be exploited are only 20% of the total vulnerabilities. The vast majority of exploitations take place within the first two weeks. It is therefore imperative to determine the severity rating without time delays. Our proposed model uses statistical methods and deep learning-based word embedding methods from natural language processing techniques, and machine learning algorithms that perform multi-class classification. Bag of Words, Term Frequency Inverse Document Frequency and Ngram methods, which are statistical methods, were used for feature extraction. Word2Vec, Doc2Vec and Fasttext algorithms are included in the study for deep learning based Word embedding. In the classification stage, Naive Bayes, Decision Tree, K-Nearest Neighbors, Multi-Layer Perceptron, and Random Forest algorithms that can make multi-class classification were preferred. With this aspect, our model proposes a hybrid method. The database used is open to the public and is the most reliable data set in the field. The results obtained in our study are quite promising. By helping experts in this field, procedures will speed up. In addition, our study is one of the first studies containing the latest version of the data size and scoring systems it covers.

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

1. Introduction

Software vulnerability can be defined as cases where errors occurring during the development process of the software cause security policies to be violated in an explicit or implicit way[1]. Information technologies are used in every aspect of modern life. The vast majority of these technologies are supported by software systems. Compensation will occur vulnerabilities in software systems can lead to impossible results. A security report published in the year 2021 will occur cybercrime result of property damage have been reported size would cost 6 trillion dollars[2]. This shows the importance of security vulnerabilities in software.

It is very important to detect and prevent software vulnerabilities without being exploited. Especially recent studies reveal that concerns about the abuse risks of security vulnerabilities are increasing[3]. Vulnerabilities detected in the official process are published in Common Vulnerabilities and Exposures(CVE), which is accessible to everyone[4]. CVE lists are used by many academic and empirical studies [5,6]. Half of the exploits that took place occurred within two weeks of posting the vulnerabilities[7]. This shows the importance of quickly analyzing security vulnerabilities and accurately determining their severity. There is always a risk for abuse of posted vulnerabilities, and it is necessary for all stakeholders to be aware of the extent of the risk and to take precautions. For this reason, the vulnerabilities in the

CVE lists are regularly evaluated by experts and published in the National Vulnerability Database (NVD) [8].

Analysis of software vulnerabilities is an important issue, and it is necessary to determine which vulnerabilities are of highest priority. A universally accepted scoring system is needed for this process. Although different systems have been proposed, the Common Vulnerability Scoring System (CVSS), which is officially used by NVD, has been accepted by all IT professionals and has become the standard in this field. The system, which is constantly updated, is currently available in 2.0 and 3.1 versions [7–10]. The main goal of CVSS is to create a common structure that accurately determines the seriousness and impact of security vulnerabilities [9]. A lot of research is being done to find new techniques for vulnerabilities [13]. In this way, it can be determined which software are more vulnerable to risk [14].

The most important feature of CVSS used in NVD security reports is the severity score, which gives initial information on the vulnerability. This score has a value between 0-10; the greater the value, the greater the danger posed by the vulnerability. The determination of severity scores is performed manually by NVD experts, which slows down the prioritization procedure. Ruohonen et al. [15] examined time delays in adding CVSS information to CVEs published in NVD. According to empirical results based on regular regression analysis of over 80,000 archived vulnerabilities, the effort spent calculating CVSS values statistically negative affects time delays. Furthermore, manual classification is prone to human error and increases cost [16].

Detected software vulnerabilities have been accumulating on CVE and NVD lists for years. This indicates that it has become a large chunk of unstructured data. It is impossible to analyze data of this size with statistical methods. Therefore, it is clear that there is undiscovered information [17]. Similarly, not all reported vulnerabilities carry the same risk. Few of the existing vulnerabilities are exploited by attackers [18]. This rate is about 20% of all software vulnerabilities [7]. This situation clearly shows us the advantage of being able to anticipate the deficits that merit focus.

One of the most important duties of the units responsible for monitoring and security of corporate networks and systems such as the Security Operations Center (SOC) against threats and vulnerabilities is to analyze the security violations. In such cases, analysts first conduct intelligence research through open source systems [19]. In particular, vulnerability databases and security scores stand out as the most preferred references at this stage (SCAP, OVAL, etc.) [20]. At this point, the model we propose will be integrated into the relevant systems and will constitute the basic reference for vulnerability reports whose security score has not yet been determined (zero-day etc.) [21]. It will also be useful for the comparison of manually assigned scores.

The main purpose of this study is to develop a method that automatically estimates the severity of security reports published by NVD. The proposed method is not intended to replace people. The goal is to support people's decision-making processes and speed up the process. It is also to enable end-users to access severity ratings faster. For this, a model using natural language processing techniques and machine learning algorithms using technical explanations of security reports has been proposed.

According to the problems above, the main research questions of our study;

RQ 1. Can the classification and scoring of security vulnerabilities, which is a manual procedure, be supported by machine learning algorithms?

RQ 2. Can a model be developed using technical descriptions of vulnerability reports written in natural language to achieve this goal?

RQ 3. Is it possible to develop a system that can help the stakeholders of the field (software developers, cyber security experts, managers and even end users) with the proposed method in this study?

Other sections of the study are organized as follows. In the second section, the solution methods proposed for the problem in the literature are examined. In the third section, the dataset used in the study is presented, while in the fourth section, vulnerability scoring systems are introduced. The research methodology is presented in the fifth section, and the findings obtained are broadly described in the sixth section. A discussion of the findings is presented in the seventh section, and threats to the validity of the study are stated in the eighth section. In the last section, the results of the study are presented in comparison with other work and future research directions are suggested.

2. Related Work

Analysis and evaluation of software vulnerabilities are important issues. Estimating the severity of a detected vulnerability accurately and quickly is very useful for the stakeholders of the issue. This process is necessary for both proactive measures and planning security, update, and patch procedures.

Spanos et al.[16], proposed a multi-target approach that predicts the characteristics of software vulnerabilities using the 99,091 security reports available in the NVD. They used text analysis and multi-target classification algorithms in their approach. Although their research has high performance values, the authors predict that the success will increase with higher data size and different algorithms. In addition, the authors do not directly estimate the severity of software vulnerabilities. They preferred to estimate the values of the security vectors and calculate the severity of software vulnerabilities over values of the security vectors. CVSS has been updated twice since their work was done. In this aspect of their work, they are not available for the current versions whose security scheme has changed.

Sharma et al.[22], prioritized with the datasets they created from 10,000 vulnerability reports from their popular vendors in their CVE lists. They used word embedding technique and convolutional neural network (CNN) for this process. However, the data set they use is limited. They also estimated the CVSS scores in three categories, with an average success rate of 87%.

Malthotra et al.[23], using textual descriptions of the vulnerabilities, they estimated their severity. They used chi-square and information acquisition methods with different classification algorithms. They achieved a 92% success rate with the Naive Bayes method they used with their knowledge acquisition. They created the data set for their work from the vulnerabilities of the Apache Tomcat program. This situation limited the scope of their studies.

Russo vd. [24], automatically created a summary of security vulnerabilities in their studies, using students and professionals who are experts in the field of cyber security, to evaluate the results. Their suggestions can also classify and categorize their vulnerabilities. In their study, they used 3369 pre-tagged vulnerability records.

Yasasin et al. [25], addresses the issue of runtime security vulnerabilities that arise after software release. They used time series in their studies. They also examine the advantages and disadvantages of error metrics on vulnerability analysis.

Aota et al. [26], tried to predict the categories of vulnerabilities using text mining techniques in their study. Their models were 96% successful. They also found that many of the vulnerability reports were misclassified. They report that their errors are caused by manual labeling procedures.

Ghaffarian et al. [1], published a comprehensive analysis of the uses of machine learning and data mining techniques in the field of software vulnerability. They gathered analyses of software vulnerabilities under four different categories. They particularly found that the use of feature extraction and machine learning algorithms in this field yielded successful results.

Izonin et al. [27,28] clearly reveal the effect of hybrid models on classification success. They extract features with the Probabilistic Neural Network. They train classifiers such as Support Vector Machine and Logistic regression with the outputs they obtain. They compared the results of their model with the literature. Their results show that hybrid models increase the classification accuracy.

Şahin et al. [29] as a result of the increasing complexity and diversity of software development, software vulnerabilities have become a very difficult unit to manage. They proposed approaches using SYNbiotic genetics algorithms, metaheuristic optimization and deep learning. They claim that their approach improves the quality and security of software.

Miyomoto et al. [30] aimed to predict the ratings in the Common Vulnerability Scoring System. They used 60,000 data in the NVD dataset for training purposes and 1300 for testing vulnerabilities. They use natural language processing techniques and Naive Bayes, LDA, Latent Semantic Indexing and Supervised LDA algorithms in their studies.

Kekül et al. [31], 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 method that gave the best results. The results they obtained are quite promising. It is also one of the first studies to use the current version of scoring systems.

The studies we have reviewed show us that there is no study similar to the scope of our current study or to the hybrid use of the different methodologies proposed. Generally, the studies we reviewed were conducted with certain vendors and limited data sets. Our study covers all existing official security vulnerabilities. In addition, we could find no study that used the 3.1 version of CVSS. The studies have especially focused on security vector values. No study of this size has been found, which is a multi-class classification that directly calculates the severity of vulnerability. Our study is original, with the dimensions of the dataset we used, and one of the first studies using the CVSS 3.1 system, direct estimation of severity, and calculation with a hybrid model consisting of different feature extraction methods and different multi-class classification algorithms.

3. Software Vulnerability Databases

Software vulnerabilities pose a significant risk in today's society equipped with information technologies. The importance of this risk has started to be recognized more recently [32]. For this reason, an effective information sharing and coordination among the stakeholders of the subject is essential. Important problems can be avoided by taking preventive measures. Management of software vulnerabilities is therefore very important [33]. It is important to know what data is collected and how it is reported.

Individuals, companies, and other organizations that detect a flaw in any software that violates their security policies, fill out a security report to report it. This notification can be made through any vulnerability database provider. However, an international standard and a procedure funded by the US Department of Homeland Security is applied for the official announcement of a vulnerability when it is detected. The authorized institution for this transaction is the not-for-profit company MITRE [34]. The official process is started by recording the vulnerabilities detected in the Common Vulnerabilities and Exposures (CVE) database by the Computer Emergency Response Team (CERT) affiliated with many countries that are members of this organization. A list of current vulnerability databases available to researchers is given in Table 1.

Table 1. SoftwareVulnerability Databases

Databases	Data Size*
Common Vulnerabilities and Exposures – CVE[4]	214.988
National Vulnerability Database - NVD[8]	170.788
Exploit-DB [35]	44.421
SecurityFocus [36]	102.330
Rapid7[37]	171.816
Snyk[38]	6.012
Software Assurance Reference Dataset Project-SARD[39]	177.184

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

3.1. Database Used

In order to facilitate the evaluation and create a suitable estimation model, we have selected NVD, the most reliable and public data set. This enables the proposed methodology to be validated by different study groups and to produce comparable results in future studies. NVD was created in 2000 under the National Institute of Standards and Technology (NIST). However, it contains security vulnerabilities that have been detected since 1988 [8]. Within the scope of this study, 156,096 records published up through 31 December 2020 were selected. In the analysis made, it was determined that some reports did not have severity scores. Reports that do not have official values have been excluded from the data set because verification cannot be made. As a result, 147,275 records with CVSS 2.0 values were included in the data set of 73,907 records with CVSS 3.1 values. Models for the two scoring systems were applied separately. For this, the data sets are separated by 70% for training-verification and 30% for testing. Training-verification and test sets consisted of 103,092 - 44,183 records for CVSS 2.0. For CVSS 3.1, 51,735 records were reserved for training-verification and 22,172 records were reserved for testing. While dividing the data set for training and testing, attention was paid to ensure that the class distributions were homogeneous. Table 2 shows an example data set record.

Table 2. An example data set record

Description	Dokodemo eye Smart HD" SCR02HD Firmware 1.0.3.1000 and earlier allows remote attackers to bypass access restriction to view information or modify configurations via unspecified vectors.	
CVSS 2.0	*Score*	6.4
	Severity Rating	Medium
CVSS 3.1	*Score*	9.1
	Severity Rating	Critical

4. Software Vulnerability Scoring Systems

Since the day the detection and exploitation of software vulnerabilities started, the need for classification has arisen. For this reason, many scoring systems have been proposed over time. The first studies in this sense include the Microsoft Threat Scoring System, Symantec Threat Scoring System, CERT Vulnerability Scoring, and SANS Critical Vulnerability Analysis Scale [40]. However, they could not be used due to their lack of general acceptance, their limited scope and their independent structure. As a solution, the universally accepted CVSS has been proposed with its open source and customizable structure [9]. Being chosen as the official scoring system of NVD has increased the popularity of CVSS [8]. In this way, it has managed to become a standard in its field [16].

4.1. CVSS - Common Vulnerability Scoring System

With its structure suitable for development, CVSS is regularly updated in line with the needs of the time and changing security perceptions. The first version, 1.0, was released in 2004[9]. In 2007, version 2.0, which has been used successfully for many years, was announced [10]. CVSS versions 3.0 and 3.1, using the same security schemes, were announced in 2016 and 2019, respectively[9,10]. Today, NVD uses both versions 2.0 and 3.1.

CVSS specifies the severity scores of security vulnerabilities in both versions with a value between 0 and 10. However, there are important differences in the classification of values. While CVSS 2.0 divides vulnerabilities into three categories, CVSS 3.1 uses five. Tables 3 and 4 show the respective categories of CVSS versions 2.0 and 3.1. The major change between CVSS 3.0 and 3.1 is the clarification of concepts. There was no change in the security scheme or calculations. For this reason, records with CVSS 3.0 vector values were not changed in the reports in the NVD dataset. New records are processed as CVSS 3.1 vectors. Therefore, in the current study, all records are expressed as CVSS 3.1. Today, CVSS versions 2.0 and 3.1 are used together. Two systems in use were also included in our study.

Table 3. Qualitative severity rating scale for v2.0 [8]

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

Table 4. Qualitative severity rating scale for v3.1 [8]

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 proposed method can be summarized as providing a classification of software vulnerabilities by estimating their severity. Natural language processing techniques, statistical feature extraction methods, word embedding and machine learning algorithms capable of multi-class classification will be used for this process. The features found in the vulnerability reports to be used in this process are technical descriptions written by experts. In order to use this text written in natural language, it is necessary to pre-process it. Basically, the general framework of the proposed model consists of preprocessing, feature extraction, feature selection, and classification stages. These stages are suggested in all models that classify using text analysis [41]. The stages of the proposed methodology are shown in Figure 1. In order to better understand the proposed methodology, the first-used data set, CVSS, preprocessing, feature extraction methods, and multi-class classification algorithms are explained respectively. The parameter values used in the algorithms are given in Table 5.

Table 5. Parameter values used in algorithms

Algorithm	Parameter List
Word2Vec	sentences,size=300,min_count=10,negative=0,workers=8,sg=0,hs=0
Doc2Vec	vector_size=300, min_count=1, epochs=100
FastText	size=300, window=3, min_count=10, sentences=sentences, iter=10
DecisionTreeClassifier	random_state=0,max_depth=20
KNeighborsClassifier	n_neighbors=10
MLPClassifier	random_state=1, max_iter=1000
RandomForestClassifier	random_state=0,max_depth=20

5.1. Pre-Processing

Studies have shown that preprocessing is as important as other stages in text analysis [42]. The first step in the preprocessing phase of texts is to remove punctuation marks and extra spaces. The second step is to remove stop words that don't make sense. The final step separates the roots of the word lemmatization, allowing the processing of words that appear different but have the same root as a single form [43]. After this stage, the texts are ready for the feature extraction process.

5.2. Feature Extraction

Researchers have been working on text analysis for a long time. As a result, many different feature extraction techniques have been proposed. These methods fall into two categories: statistical and deep learning based approaches [44]. Statistical methods ignore the semantic relationships between words, and they have high computational requirements due to their direct processing of word frequencies. However, they are highly suitable for comparing the performance of classification models [41]. Since the proposed methodology focuses on the performance of different machine learning algorithms, we have chosen the statistical methods described below for our study.

Bag of Words (BoW): In this model, the document is represented as a vector consisting of the frequency of words. It is a common method used to extract features of texts [45].

Term Frequency Inverse Document Frequency (tf-idf): While a vector is created with the frequency of the words for each text in the BoW model, in the tf-idf model, a vector is created by calculating the importance of a word in the document and the corpus. For this, the absolute value of the logarithm of the Number of Documents / Number of Documents that the Word Passes Through is taken [46].

Ngram: Similar to the BoW model, this model counts words by selecting them in groups of two, three, or more. With the N value, text vectors with different word groups can be created. If “1” is selected as the N value, the model created will be the same as the BoW model [47]. In our study, we chose “2” and “3” as N values. However, it has been observed that the performance decreases as the N value increases. For these reasons, the results obtained by taking N value as two were reflected in the tables.

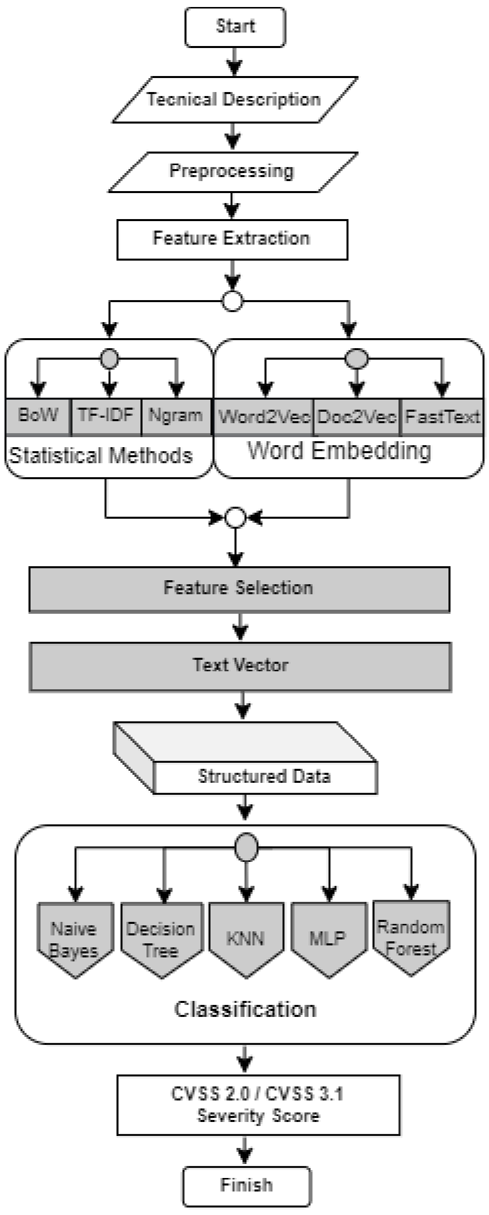

Fig.1. The general structure of the proposed methodology.

Traditional statistical methods focusing on word frequency continue to be used in feature extraction from text-based data. However, the use of word embedding algorithms, one of the new deep learning-based methods that focus on semantic relationships between words, is increasing [48]. For this reason, word embedding algorithms Word2Vec, Doc2Vec and FastText were used in the feature extraction stage for the creation of text vectors in the proposed methodology.

Word2Vec: It is an unsupervised method that allows words to be displayed as vectors. It is focused on learning the relationship between words using deep neural networks. It creates different vectors for each word. The generated vectors are a vector space determined by similarities to other words. After the model is trained, synonymous words and semantic relationships between words can appear. It uses two different methods, Continous Bag of Words (CBOW) and

Skip-Gram [49, 50].

Doc2Vec: By nature, Word2Vec creates only word vectors, not document vectors. Sentences within documents do not contain a specific semantic connection, unlike words. However, a simple and clever solution was produced by adding a new layer to the Word2Vec model. The Doc2Vec structure, which uses the CBOW structure and adds document-specific feature vectors besides words, is proposed to create a vector representation of a document regardless of its length.

FastText: A method used to represent words in a document in vector space. It uses the Skip-Gram structure of the Word2Vec method. The focus of words on morphology makes it different from other methods. In this way, it can better represent the logical relationships between words [51].

5.3. Feature Selection

Applying the proposed methodology, we determined the number of different words for the technical explanations in the data set used for feature extraction models to be 86, 544. The data set consisting of vectors to be created by using all words creates a large sparse matrix. While this situation increases system requirements, it does not have a positive benefit on performance. Studies have shown that the optimum word number is 300 [52]. In our experimental studies, we observed that using more than 300 words in vectors will not affect performance. Therefore, we have limited the number of words used in vectors to 300. In the vectorization process, idf and ngram values were calculated from the 300 most used words and the training set. With these calculated values, the vector values of the test data were found.

Among the word embedding methods, Wor2Vec and FastText only extract word vectors. Therefore, they cannot be used directly in the classification of documents. In the proposed approach, word vectors were averaged in the datasets created for the classification of vulnerability reports. Vectors representing vulnerability reports were generated from the results found. At the end of this process, we created a structured data set that can be used with machine learning algorithms.

5.4. Multiclass Classification Algorithms

After we created the structured data set, we began the classification process. At this stage, our target classes consisted of three classes for CVSS 2.0 and five classes for CVSS 3.1. Because of this structure, our model is a multi-class classification problem. Therefore, we chose estimators with built-in strategies for multi-class classification that have proven successful in text analysis, including Naive Bayes (NB) [53], Decision Tree (DT) [54], K-Nearest Neighbors (KNN) [55], Multi-Layer Perceptron (MLP) [56], and Random Forest (RF) [57]. We have used the cross validation method to determine whether our model has overfitting or selection bias problems and to measure accuracy rates [58, 59]. We chose a value of 10 as the cross validation parameter.

5.5. Validation

In multi-class classification models, if the target classes in the data set are not homogeneous, the performance value depends on the prediction success of the dominant class [60]. This situation makes it difficult to evaluate the model correctly. In order to avoid this problem, we chose the verification methods suggested in the literature to evaluate our model [61].

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

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (1)$$

Recall: shows how many of the values that should have been predicted positive were predicted positive.

$$Recall = \frac{TP}{TP+FN} \quad (2)$$

Precision: shows how many of the values guessed positive were actually positive.

$$Precision = \frac{TP}{TP+FP} \quad (3)$$

F1 Score: shows the harmonic mean of the precision and recall values.

$$F_1 = 2 * \frac{Precision*Recall}{Precision+Recall} \quad (4)$$

In order to evaluate multi-class models, we evaluated the model with the accuracy, recall, precision, and F1-score methods recommended in the literature. In this respect, the evaluation principles of our model can be compared with the criteria of other studies [62].

6. Results

In order to evaluate the results of the proposed methodology correctly, we present the data of the training performance of our model in detail in Table 6. We explained in detail in the previous section that 70% of our data set was reserved for training and 30% for testing.

When the accuracy and standard deviation values of our model (trained using the cross validation technique) are examined from Table 6, the consistency of the results can be seen. In addition, it is understood that there is no overfitting or selection bias. We emphasize that predicting a problem with a multi-probability nature from a limited technical explanation is defined as a difficult task in the literature [16].

Table 6. Cross Validation train results of vulnerability score rating

			NB	**DT**	**KNN**	**MLP**	**RFC**
CVSS 2.0	BoW	*Mean*	68	79	62	57	64
		Std	0.0028	0.0035	0.0049	0.0005	0.003
	TF-IDF	*Mean*	68	78	74	74	62
		Std	0.0028	0.0038	0.0047	0.0037	0.0022
	2Ngram	*Mean*	68	76	72	77	67
		Std	0.0033	0.0056	0.0078	0.0065	0.0035
	Word2Vec	*Mean*	0.51	0.73	0.74	0.76	0.63
		Std	0.0047	0.0038	0.0024	0.0032	0.0023
	Doc2Vec	*Mean*	0.59	0.63	0.70	0.69	0.68
		Std	0.0051	0.0023	0.0032	0.0063	0.0029
	FastText	*Mean*	0.76	0.92	0.93	0.93	0.93
		Std	0.0038	0.0016	0.0014	0.0029	0.0018
CVSS 3.1	BoW	*Mean*	0.54	0.72	0.68	0.72	0.53
		Std	0.0044	0.0076	0.0058	0.0061	0.0037
	TF-IDF	*Mean*	0.54	0.72	0.67	0.69	0.53
		Std	0.0044	0.005	0.0085	0.0072	0.0038
	2Ngram	*Mean*	0.46	0.68	0.61	0.68	0.5
		Std	0.0064	0.006	0.0142	0.0033	0.0049
	Word2Vec	*Mean*	0.52	0.59	0.64	0.66	0.66
		Std	0.0056	0.0079	0.0073	0.0074	0.0068
	Doc2Vec	*Mean*	0.44	0.48	0.56	0.55	0.55
		Std	0.0053	0.0029	0.0060	0.0061	0.0039
	FastText	*Mean*	0.46	0.65	0.68	0.69	0.69
		Std	0.0036	0.0064	0.0048	0.0066	0.0064

6.1. *CVSS 2.0 Severity Rating Results*

We present the results of the classification of severity ratings for CVSS 2.0 to evaluate the proposed methodology. This model has attempted to make the correct prediction among three different classes, shown in table 3. Table 7 shows the best performance values of all models. We evaluated results according to accuracy, precision, recall, and F1-score scales. The precision and recall values specified in the table are the macro values found by taking the average of the performance of each class. In addition, we calculated the F1-score value with the harmonic mean of the precision and recall values.

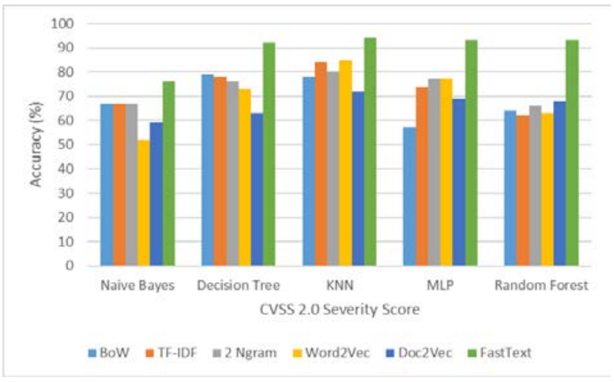

Fig.2. The bar-plot of the severity rating prediction for CVSS 2.0

To perform an accurate evaluation, each text vectorization model should first be evaluated separately according to the classification algorithms. Figure 2 shows the classification performance of all models. As can be seen, the most successful models are the ones using FastText. It obtained the most successful values in all classification algorithms. TF-IDF using KNN is the statistical model with the highest success. Figure 3 is a radar graph showing the classification performance of the models in a comparative manner. From here, we can see that KNN used with FastText performs more successful classifications than other algorithms.

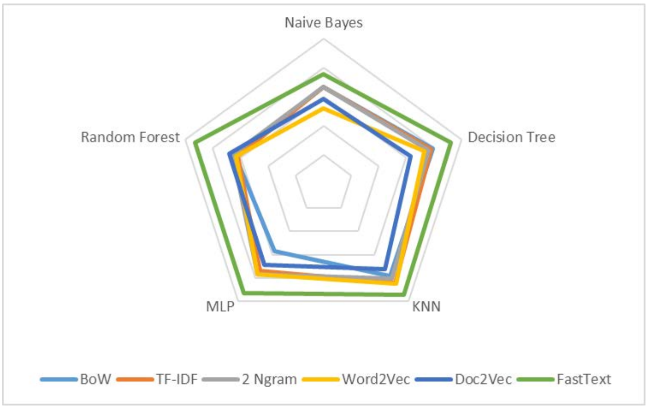

Fig.3. The comparative accuracies radar graphs of the models for CVSS 2.0.

Table 7. CVSS 2.0 The results of the predictive value

Metric		Alg.	Acc.	Prec.	Recall	F1
Statistical Methods	BoW	NB	0.67	0.60	0.64	0.61
		DT	**0.79**	**0.77**	**0.70**	**0.73**
		KNN	0.78	0.71	0.71	0.71
		MLP	0.57	0.19	0.33	0.24
		RF	0.64	0.48	0.42	0.39
	tf-idf	NB	0.67	0.60	0.64	0.61
		DT	0.78	0.77	0.69	0.72
		KNN	**0.84**	**0.80**	**0.80**	**0.80**
		MLP	0.74	0.68	0.68	0.68
		RF	0.62	0.48	0.39	0.36
	Ngram	NB	0.67	0.60	0.64	0.62
		DT	0.76	0.73	0.67	0.69
		KNN	**0.80**	**0.76**	**0.73**	**0.74**
		MLP	0.77	0.75	0.67	0.70
		RF	0.66	0.48	0.44	0.42
Word Embedding Methods	Word2Vec	NB	0.52	0.49	0.54	0.48
		DT	0.73	0.72	0.60	0.64
		KNN	**0.85**	**0.81**	**0.81**	**0.81**
		MLP	0.77	0.73	0.70	0.72
		RF	0.63	0.48	0.41	0.38
	D oc2Vec	NB	0.59	0.53	0.58	0.54
		DT	0.63	0.58	0.44	0.44
		KNN	**0.72**	**0.66**	**0.59**	**0.61**
		MLP	0.69	0.62	0.62	0.62
		RF	0.68	0.80	0.47	0.48
	FastText	NB	0.76	0.53	0.72	0.57
		DT	0.92	0.91	0.70	0.77
		KNN	**0.94**	**0.90**	**0.77**	**0.82**
		MLP	0.93	0.80	0.81	0.80
		RF	0.93	0.93	0.71	0.79

In addition, when Table 7 is examined, the consistency of the validation values of our estimation model will be seen. The FastText model has been the most successful with all algorithms. It showed the highest success rate of 94% with the KNN algorithm. Moreover, the success rates of DT, MLP and RF algorithms are quite high.

The success rate of the NB algorithm, where the FastText method has the lowest value, is 76%. When the results of other Word Embedding methods are examined, it is seen that the MLP algorithm used with Word2Vec achieved the highest success rates of 85%, and the KNN algorithms used with Doc2Vec achieved the highest success rates of 77%. Although these two methods have shown successful classification performances, they are far behind FastText.

Of the statistical methods, KNN and TF-IDF classification model has a success rate of 84%. It became the second most accurate model by achieving 80% classification success with Ngram KNN from other models. The BoW model achieved the highest success rate with 79% with the DT algorithm. When Word Embedding models are compared against each other, it is clear that FastText outperforms other models. However, the results obtained by the Tf-Idf and Word2Vec models with the KNN algorithm are promising.

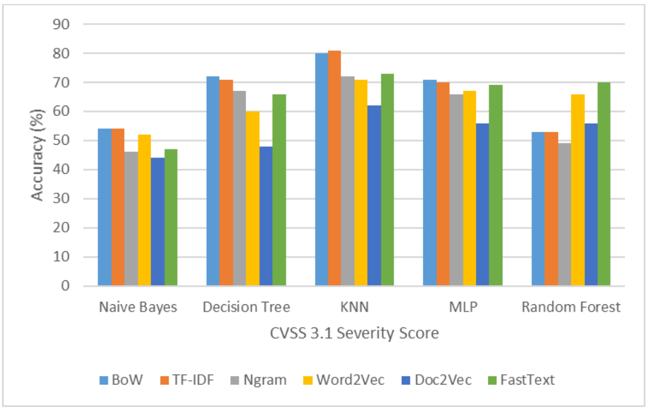

Fig.4. The bar-plot of the severity rating prediction for CVSS 3.1

Table 8. CVSS 3.1 The results of the predictive value

Metric		Alg.	Acc.	Prec.	Recall	F1
Statistical Methods	BoW	NB	0.54	0.43	0.47	0.43
		DT	0.72	0.68	0.57	0.61
		KNN	**0.80**	**0.73**	**0.73**	**0.73**
		MLP	0.71	0.69	0.55	0.57
		RF	0.53	0.30	0.31	0.27
	tf-idf	NB	0.54	0.43	0.47	0.43
		DT	0.71	0.67	0.56	0.59
		KNN	**0.81**	**0.74**	**0.74**	**0.74**
		MLP	0.70	0.59	0.58	0.58
		RF	0.53	0.31	0.31	0.27
	Ngram	NB	0.46	0.41	0.44	0.37
		DT	0.67	0.61	0.53	0.55
		KNN	**0.72**	**0.64**	**0.62**	**0.62**
		MLP	0.66	0.70	0.50	0.52
		RF	0.49	0.33	0.29	0.23
Word Embedding Methods	Word2Vec	NB	0.52	0.41	0.45	0.41
		DT	0.60	0.65	0.41	0.43
		KNN	**0.71**	**0.69**	**0.51**	**0.54**
		MLP	0.67	0.55	0.47	0.48
		RF	0.66	0.73	0.45	0.47
	Doc2Vec	NB	0.44	0.33	0.35	0.34
		DT	0.48	0.31	0.30	0.28
		KNN	**0.62**	**0.60**	**0.43**	**0.45**
		MLP	0.56	0.42	0.42	0.42
		RF	0.56	0.74	0.34	0.32
	FastText	NB	0.47	0.41	0.46	0.38
		DT	0.66	0.66	0.47	0.50
		KNN	**0.73**	**0.69**	**0.55**	**0.58**
		MLP	0.69	0.59	0.55	0.57
		RF	0.70	0.74	0.50	0.53

6.2. *CVSS 3.1 Severity Rating Results*

The latest version of the vulnerability scoring system is CVSS 3.1. For this reason, few studies have used it to this point. This study will complete an important shortcoming in our field. With the CVSS 3.1 version, the number of classes to which the severity ratings belong has increased to five. These are presented in table 4. As stated in the previous section, creating a prediction model using a limited technical explanation becomes more difficult as the number of classes increases. Despite all the difficulties, the results of our model, in which the number of data is 50% less, are quite satisfactory. Table 8 shows the results of all models and classification algorithms. As can be seen from the table, we present the values of all validation methods so that the test results can be evaluated correctly.

Classification performances of models and algorithms can be seen in figure 4. CVSS 3.1 is the TF-IDF and KNN model that achieved the highest success in predicting severity ratings. The lowest performance value is seen as 3Ngram and NB model. Figure 5 presents radar graphs showing the comparative performance of the models.

When we look at the details of the classification performances of the models in Table 8, we see that the TF-IDF and BoW models have performance values close to those of KNN algorithms. The TF-IDF and KNN model has been the most successful model with a classification success rate of 81%, leaving behind BoW and KNN models that achieved 80% classification success. The second BoW and KNN models were followed by FastText and KNN with a 73% success rate. Ngram-KNN and BoW-DT models are other models that seem successful with an accuracy rate of 72%. Among the other models, the models with 70% or more success were respectively BoW-MLP, TFIDF-DT/MLP, Word2Vec-KNN and FastText-RF. The model with the lowest success rate is Doc2Vec with 44% classification success using NB. When the results are examined, it is clearly seen that all models, except Doc2Vec, show over 70% prediction performance with at least one algorithm. It is seen that the most successful vectorization methods are BoW and TF-IDF, and the most successful algorithm is KNN.

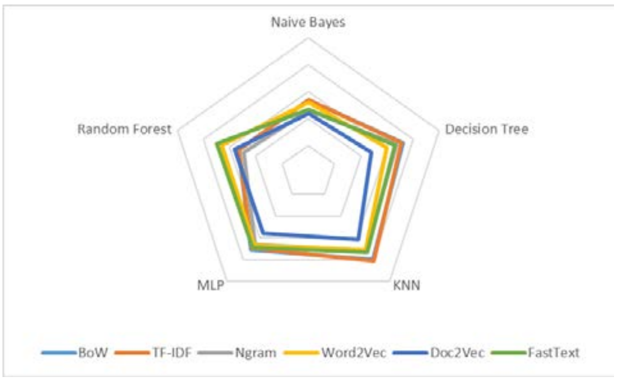

Fig.5. The comparative accuracies radar graphs of the models for CVSS 3.1.

Table 9 shows the highest classification success of all models. It is clear from this table that the most successful classifier is the KNN algorithm. Only the DT algorithm has achieved the highest success with BoW. For all other models, KNN was the algorithm with the highest success rate. Among the vectorization methods, FastText had the highest success rate for CVSS 2.0, while the highest success rate for CVSS 3.1 was achieved with TF-IDF. When statistical methods and word embedding methods are compared within themselves, the TF-IDF and FastText methods stand out. Success rates among statistical methods are close. However, in Word embedding methods, Doc2Vec has lagged behind other methods.

Table 9. Highest Value Results of Forecast Models

CVSS	Metric	Alg.	Acc.	Prec.	Recall	F1
2.0	BoW	DT	0.79	0.77	0.70	0.73
	TF-IDF	KNN	0.84	0.80	0.80	0.80
	Ngram	KNN	0.80	0.76	0.73	0.74
	Word2Vec	KNN	0.85	0.81	0.81	0.81
	Doc2Vec	KNN	0.72	0.66	0.59	0.61
	FastText	**KNN**	**0.94**	**0.90**	**0.77**	**0.82**
3.1	BoW	KNN	0.80	0.73	0.73	0.73
	TF-IDF	**KNN**	**0.81**	**0.74**	**0.74**	**0.74**
	Ngram	KNN	0.72	0.64	0.62	0.62
	Word2Vec	KNN	0.71	0.69	0.51	0.54
	Doc2Vec	KNN	0.62	0.60	0.43	0.45
	FastText	KNN	0.73	0.69	0.55	0.58

7. Discussion

The severity ratings of software vulnerabilities have been estimated from technical explanations written in natural language. In solving this problem, six different algorithms from statistical and word embedding methods are used for feature extraction. In addition, five multi-class classification algorithms and methods were evaluated separately. The results of this hybrid structure are presented in the sixth section. All models produced acceptable results to some degree. However, FastText for CVSS 2.0 and tf-idf for CVSS 3.1 achieved the highest success rate in estimating the severity of the two scoring systems.

Although the same technical explanations are used in training the models, there are serious size differences in the data sets. For this reason, it is necessary to evaluate the results of vulnerability scoring systems and model performances separately for each version. The dataset size used for CVSS 2.0 is approximately twice the CVSS 3.1 dataset size. When the CVSS 2.0 results are examined, it is clear that the FastText method has a superior success with all classification algorithms except NB. Moreover, the Word2Vec method is the second most successful method with the KNN algorithm. The success of word embedding methods is evident in this particular problem. When the statistical methods are examined, the TF-IDF method has been the most successful statistical model with the KNN algorithm. However, the results of other statistical methods are also very close and at a level that can be considered successful.

The estimation values of CVSS 3.1, which is a relatively new version and on which no mature academic studies have been made yet, are considerably lower than CVSS 2.0.

It should be noted that CVSS version 3.1 represents a more difficult problem in terms of both the number of classes to be estimated and the size of the dataset. Since this study is the first to use CVSS version 3.1, it will constitute a reference point. Especially in the CVSS 2.0 version, the success of the word embedding values comes to the fore, while the estimation performance of the statistical methods in the CVSS 3.1 version draws attention. In particular, the performance of TF-IDF and BoW methods with the KNN algorithm comes to the fore. Moreover, the KNN algorithm also shows a successful result in the Ngram method. Although FastText and Word2Vec methods, which gave very successful results in the prediction of CVSS 2.0 version, produced relatively successful results with the KNN algorithm in CVSS 3.1 version, they lagged far behind statistical models. Doc2Vec seems to be the method that produces the most unsuccessful results in terms of the results obtained.

If statistical methods are ranked according to their success, they are listed as TF-IDF, BoW and Ngram. Word Embeddig methods are listed as FastText, Word2Vec and Doc2Vec. Doc2Vec models had the lowest average success values for the two systems. When the Ngram methods are evaluated within themselves, we see that the increase in the value of n decreases the prediction success. For this reason, only results with N value of 2 are reflected in the tables.

When the algorithms used for classification are evaluated, we see that the most successful is KNN. It has the best results In eleven out of twelve predictions for all models. The KNN algorithm was followed by DT, which has the best value in only one model. However, in the other models, the classification algorithms with the second highest value were always MLP seven times and DT five times. The poor results of the NB and RF algorithms put them in fourth and fifth place, respectively. In summary, in order of success rates, the classification algorithms are KNN, DT, MLP, NB, and RF.

8. Threats to Validity

As with every study, there are elements that threaten the validity of the results of the current study. Threat to validity is the exclusive use of statistical methods for feature extraction from text data. Of course, many different methods can be considered for feature selection. However, one of the aims of the proposed methodology is to compare the performance of multiclass classification algorithms on this problem. For this reason, different vectorization methods were not considered. In addition, the method is based on reports such as Table 2 written by experts. Therefore, it is still possible for human error to be made in writing reports.

9. Conclusions

Software vulnerabilities are increasing rapidly. Security reports, which have become a large accumulation of unstructured data, need to be evaluated accurately and on time. There is a need for methods to automate this process, which is done manually by humans. Therefore, our study fills an important gap. The proposed model will support and speed up the procedures performed by experts. Our methodology offers a broad overview of natural language processing techniques and algorithms capable of multi-class classification.

The results of the proposed methodology show that the severity of software vulnerability reports can be accurately predicted with a very high success rate of 94% for CVSS 2.0 and 81% for CVSS 3.1. On a model basis, statistical methods for CVSS 2.0 showed a remarkable success of 84% with tf-idf. However, this value was well below the 94% estimation success achieved with the word embedding algorithm FastText. For CVSS 3.1, a 73% prediction success was achieved with the word embedding algorithm FastText. The prediction success of the Word2Vec model is 71%. In this model, where the size of the data set was reduced by almost half, tf-idf obtained 81% and BoW 80% prediction success from statistical methods.

When we compare our study with other studies in the literature, our study is the largest in terms of the size of the data used. We evaluated all reports with official values, and instead of focusing on a specific vendor or vulnerability category, we took a holistic approach. In this respect, our study is the most comprehensive research in its field. It is also one of the first studies to use the most up-to-date version of study scoring systems, which is original and innovative. The results of the study are very promising given the wide scope of the non-working. As with any study, the current study has limitations. First of all, these threats originate from the database used. The NVD database we use is the largest public and official database in the field. However, some researchers state that the explanations in the NVD reports are insufficient and should be more specific when describing security vulnerabilities. Therefore, it can increase the success of evaluating technical descriptions of different vulnerability databases.

When the results of our study are analyzed, it is understood that statistical methods and word embedding methods show variable results on different data sets. Especially as the size of the data set increases, the success rates of Word embedding methods increase. However, it is seen that the results of statistical methods are less affected by the data size. This clearly shows us that word embedding methods produce more successful results with high-dimensional data. FastText and Word2Vec methods work with word frequencies. Doc2Vec methods, on the other hand, are focused on finding the document frequency. It is understood that the short explanations in the vulnerability reports have a negative effect on document frequency extraction and therefore negatively affect the success of the Doc2Vec method. In addition, it is seen that statistical methods are not much affected by the data size due to their structure. Moreover, the findings show that the feature extraction method directly affects the prediction success on classification success.

Text mining and machine learning algorithms have many methods other than those used in the proposed methodology. We would like to try different techniques in our future work, especially deep-learning-based feature extraction techniques and classification methods, which we feel could increase success rates. We also want to test our methodology on different data sets. We believe this will unlock the potential of security reports from different data providers.

CRediT authorship contribution statement

Hakan KEKÜL: Conceptualization, Methodology, Validation, Formal analysis, Resources, Data Curation, Writing - Original Draft, Writing - Review & Editing , Visualization. **Burhan ERGEN:** Conceptualization, Methodology, Validation, Formal analysis, Writing - Review & Editing, Supervision, Project administration. **Halil ARSLAN:** Conceptualization, Methodology, Validation, Formal analysis, Writing - Original Draft, Writing - Review & Editing, Supervision

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

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

References

[1] S.M. Ghaffarian, H.R. Shahriari, Software vulnerability analysis and discovery using machine-learning and data-mining techniques: A survey, ACM Comput. Surv. 50 (2017). https://doi.org/10.1145/3092566.
[2] L.P. Kobek, The State of Cybersecurity in Mexico: An Overview, Wilson Centre's Mex. Institute, Jan. (2017).
[3] T.W. Moore, C.W. Probst, K. Rannenberg, M. van Eeten, Assessing ICT Security Risks in Socio-Technical Systems (Dagstuhl Seminar 16461), Dagstuhl Reports. 6 (2017) 63–89. https://doi.org/10.4230/DagRep.6.11.63.
[4] CVE, CVE, Common Vulnerabilities Expo. (2020). https://cve.mitre.org (accessed July 25, 2020).
[5] Samuel Ndichu, Sylvester McOyowo, Henry Okoyo, Cyrus Wekesa, "A Remote Access Security Model based on Vulnerability Management", International Journal of Information Technology and Computer Science, Vol.12, No.5, pp.38-51, 2020.
[6] H. Kekül, B. Ergen, H. Arslan, Yazılım Güvenlik Açığı Veri Tabanları, Avrupa Bilim ve Teknol. Derg. (2021) 1008–1012.
[7] H. Yang, S. Park, K. Yim, M. Lee, Better not to use vulnerability's reference for exploitability prediction, Appl. Sci. 10 (2020). https://doi.org/10.3390/app10072555.
[8] NVD, NVD, Natl. Vulnerability Database. (2020). https://nvd.nist.gov (accessed July 25, 2020).
[9] M. Schiffman, 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 January 1, 2021).

[10] P. Mell, K. Scarfone, S. Romanosky, A Complete Guide to the Common Vulnerability Scoring System Version 2.0, FIRSTForum Incid. Response Secur. Teams. (2007) 1–23. https://www.first.org/cvss/cvss-v2-guide.pdf (accessed January 1, 2021).

[11] Common Vulnerability Scoring System v3.0: User Guide, (n.d.). https://www.first.org/cvss/v3.0/user-guide (accessed January 1, 2021).

[12] Common Vulnerability Scoring System v3.1: User Guide, (n.d.). https://www.first.org/cvss/v3.1/user-guide (accessed January 1, 2021).

[13] Muhammad Noman Khalid, Muhammad iqbal, Kamran Rasheed, Malik Muneeb Abid, "Web Vulnerability Finder (WVF): Automated Black- Box Web Vulnerability Scanner", International Journal of Information Technology and Computer Science, Vol.12, No.4, pp.38-46, 2020.

[14] Hakan Kekül, Burhan Ergen, Halil Arslan, " A New Vulnerability Reporting Framework for Software Vulnerability Databases", International Journal of Education and Management Engineering, Vol.11, No.3, pp. 11-19, 2021.

[15] J. Ruohonen, A look at the time delays in CVSS vulnerability scoring, Appl. Comput. Informatics. 15 (2019) 129–135. https://doi.org/10.1016/j.aci.2017.12.002.

[16] G. Spanos, L. Angelis, A multi-target approach to estimate software vulnerability characteristics and severity scores, J. Syst. Softw. 146 (2018) 152–166. https://doi.org/10.1016/j.jss.2018.09.039.

[17] C. Theisen, L. Williams, Better together: Comparing vulnerability prediction models, Inf. Softw. Technol. 119 (2020). https://doi.org/10.1016/j.infsof.2019.106204.

[18] Y. Fang, Y. Liu, C. Huang, L. Liu, Fastembed: Predicting vulnerability exploitation possibility based on ensemble machine learning algorithm, PLoS One. 15 (2020) 1–28. https://doi.org/10.1371/journal.pone.0228439.

[19] F.D. János, N. Huu Phuoc Dai, Security Concerns Towards Security Operations Centers, in: 2018 IEEE 12th Int. Symp. Appl. Comput. Intell. Informatics, 2018: pp. 273–278. https://doi.org/10.1109/SACI.2018.8440963.

[20] K. Kritikos, K. Magoutis, M. Papoutsakis, S. Ioannidis, A survey on vulnerability assessment tools and databases for cloud-based web applications, Array. 3–4 (2019) 100011. https://doi.org/10.1016/j.array.2019.100011.

[21] X. Wu, W. Zheng, X. Chen, F. Wang, D. Mu, CVE-assisted large-scale security bug report dataset construction method, J. Syst. Softw. 160 (2020) 110456. https://doi.org/10.1016/j.jss.2019.110456.

[22] R. Sharma, R. Sibal, S. Sabharwal, Software vulnerability prioritization using vulnerability description, Int. J. Syst. Assur. Eng. Manag. 12 (2021) 58–64. https://doi.org/10.1007/s13198-020-01021-7.

[23] R. Malhotra, Vidushi, Severity Prediction of Software Vulnerabilities Using Textual Data, in: V.K. Gunjan, J.M. Zurada (Eds.), Proc. Int. Conf. Recent Trends Mach. Learn. IoT, Smart Cities Appl., Springer Singapore, Singapore, 2021: pp. 453–464.

[24] E.R. Russo, A. Di Sorbo, C.A. Visaggio, G. Canfora, Summarizing vulnerabilities' descriptions to support experts during vulnerability assessment activities, J. Syst. Softw. 156 (2019) 84–99. https://doi.org/10.1016/j.jss.2019.06.001.

[25] E. Yasasin, J. Prester, G. Wagner, G. Schryen, Forecasting IT security vulnerabilities – An empirical analysis, Comput. Secur. 88 (2020) 101610. https://doi.org/10.1016/j.cose.2019.101610.

[26] M. Aota, H. Kanehara, M. Kubo, N. Murata, B. Sun, T. Takahashi, Automation of Vulnerability Classification from its Description using Machine Learning, in: 2020 IEEE Symp. Comput. Commun., 2020: pp. 1–7. https://doi.org/10.1109/ISCC50000.2020.9219568.

[27] I. Izonin, R. Tkachenko, M. Gregus, L. Ryvak, V. Kulyk, V. Chopyak, Hybrid Classifier via PNN-based Dimensionality Reduction Approach for Biomedical Engineering Task, Procedia Comput. Sci. 191 (2021) 230–237. https://doi.org/https://doi.org/10.1016/j.procs.2021.07.029.

[28] M.G.Z.D. 3 N.S. Ivan Izonin Roman Tkachenko, PNN-SVM Approach of Ti-Based Powder's Properties Evaluation for Biomedical Implants Production, Comput. Mater. \& Contin. 71 (2022) 5933–5947. https://doi.org/10.32604/cmc.2022.022582.

[29] C.B. Şahin, Ö.B. Dinler, L. Abualigah, Prediction of software vulnerability based deep symbiotic genetic algorithms: Phenotyping of dominant-features, Appl. Intell. 51 (2021) 8271–8287. https://doi.org/10.1007/s10489-021-02324-3.

[30] D. Miyamoto, Y. Yamamoto, M. Nakayama, Text-mining approach for estimating vulnerability score, Proc. - 2015 4th Int. Work. Build. Anal. Datasets Gather. Exp. Returns Secur. BADGERS 2015. (2017) 67–73. https://doi.org/10.1109/BADGERS.2015.12.

[31] H. Kekül, B. Ergen, H. Arslan, A multiclass hybrid approach to estimating software vulnerability vectors and severity score, J. Inf. Secur. Appl. 63 (2021) 103028. https://doi.org/https://doi.org/10.1016/j.jisa.2021.103028.

[32] A. Kuehn, M. Mueller, Shifts in the cybersecurity paradigm: Zero-day exploits, discourse, and emerging institutions, in: Proc. 2014 New Secur. Paradig. Work., 2014: pp. 63–68.

[33] O. Bozoklu, C.Z. Çil, Yazılım Güvenlik Açığı Ekosistemi Ve Türkiye'deki Durum Değerlendirmesi, Uluslararası Bilgi Güvenliği Mühendisliği Derg. 3 (2017) 6–26.

[34] Mitre Corporation, (2020). https://www.mitre.org (accessed July 25, 2020).

[35] ExploitDB, Exploit Database, (2020). https://www.exploit-db.com (accessed July 25, 2020).

[36] SecurityFocus, SecurityFocus, (2020). https://www.securityfocus.com (accessed July 25, 2020).

[37] Rapid7, Rapid7, (2020). https://www.rapid7.com/db/ (accessed July 25, 2020).

[38] Snyk, Snyk, (2020). https://snyk.io (accessed July 25, 2020).

[39] SARD, SARD-Software Assurance Reference Dataset Project, (2020). https://samate.nist.gov (accessed July 25, 2020).

[40] V.-V. Patriciu, I. Priescu, S. Nicolaescu, Security metrics for enterprise information systems, J. Appl. Quant. Methods. 1 (2006) 151–159.

[41] A. Fesseha, S. Xiong, E.D. Emiru, M. Diallo, A. Dahou, Text Classification Based on Convolutional Neural Networks and Word Embedding for Low-Resource Languages: Tigrinya, Information. 12 (2021). https://doi.org/10.3390/info12020052.

[42] A.K. Uysal, S. Gunal, The impact of preprocessing on text classification, Inf. Process. Manag. 50 (2014) 104–112. https://doi.org/https://doi.org/10.1016/j.ipm.2013.08.006.

[43] A.A. Jalal, B.H. Ali, Text documents clustering using data mining techniques., Int. J. Electr. Comput. Eng. 11 (2021).

[44] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, D. Brown, Text classification algorithms: A survey, Information. 10 (2019) 150.
[45] Y. Zhang, R. Jin, Z.-H. Zhou, Understanding bag-of-words model: a statistical framework, Int. J. Mach. Learn. Cybern. 1 (2010) 43–52.
[46] A. Aizawa, An information-theoretic perspective of tf--idf measures, Inf. Process. Manag. 39 (2003) 45–65.
[47] S. Banerjee, T. Pedersen, The design, implementation, and use of the ngram statistics package, in: Int. Conf. Intell. Text Process. Comput. Linguist., 2003: pp. 370–381.
[48] M. Aydoğan, A. Karci, Turkish Text Classification with Machine Learning and Transfer Learning, in: 2019 Int. Artif. Intell. Data Process. Symp., 2019: pp. 1–6. https://doi.org/10.1109/IDAP.2019.8875919.
[49] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Adv. Neural Inf. Process. Syst., 2013: pp. 3111–3119.
[50] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, ArXiv Prepr. ArXiv1301.3781. (2013).
[51] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, Trans. Assoc. Comput. Linguist. 5 (2017) 135–146.
[52] Z. Yin, Y. Shen, On the dimensionality of word embedding, ArXiv Prepr. ArXiv1812.04224. (2018).
[53] S. Aggarwal, D. Kaur, Naïve Bayes Classifier with Various Smoothing Techniques for Text Documents, in: 2013.
[54] L. Breiman, J. Friedman, C.J. Stone, R.A. Olshen, Classification and regression trees, CRC press, 1984.
[55] E. Fix, Discriminatory analysis: nonparametric discrimination, consistency properties, USAF school of Aviation Medicine, 1951.
[56] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (1943) 115–133.
[57] L. Breiman, Random Forests, Mach. Learn. 45 (2001) 5–32. https://doi.org/10.1023/A:1010933404324.
[58] R. Kohavi, others, A study of cross-validation and bootstrap for accuracy estimation and model selection, in: Ijcai, 1995: pp. 1137–1145.
[59] G.C. Cawley, N.L.C. Talbot, On over-fitting in model selection and subsequent selection bias in performance evaluation, J. Mach. Learn. Res. 11 (2010) 2079–2107.
[60] D. Ballabio, F. Grisoni, R. Todeschini, Multivariate comparison of classification performance measures, Chemom. Intell. Lab. Syst. 174 (2018) 33–44. https://doi.org/https://doi.org/10.1016/j.chemolab.2017.12.004.
[61] M. Sokolova, G. Lapalme, A systematic analysis of performance measures for classification tasks, Inf. Process. Manag. 45 (2009) 427–437. https://doi.org/https://doi.org/10.1016/j.ipm.2009.03.002.
[62] C. Bielza, G. Li, P. Larrañaga, Multi-dimensional classification with Bayesian networks, Int. J. Approx. Reason. 52 (2011) 705–727. https://doi.org/https://doi.org/10.1016/j.ijar.2011.01.007.

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 2018, 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, "A Multiclass Approach to Estimating Software Vulnerability Severity Rating with Statistical and Word Embedding Methods", International Journal of Computer Network and Information Security(IJCNIS), Vol.14, No.4, pp.27-42, 2022. DOI:10.5815/ijcnis.2022.04.03