A tree-based machine learning methodology to automatically classify software vulnerabilities

Georgios Aivatoglou*, Mike Anastasiadis*, Georgios Spanos*, Antonis Voulgaridis*,
Konstantinos Votis*, Dimitrios Tzovaras*
*Information Technologies Institute, Centre for Research and Technology Hellas, Thessaloniki, Greece
{aivatoglou,manastas,gspanos,antonismv,kvotis,Dimitrios.Tzovaras}@iti.gr

2021 IEEE International Conference on Cyber Security and Resilience (CSR) | 978-1-6654-0285-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/CSR51186.2021.9527965

***Abstract*—Software vulnerabilities have become a major problem for the security analysts, since the number of new vulnerabilities is constantly growing. Thus, there was a need for a categorization system, in order to group and handle these vulnerabilities in a more efficient way. Hence, the MITRE corporation introduced the Common Weakness Enumeration that is a list of the most common software and hardware vulnerabilities. However, the manual task of understanding and analyzing new vulnerabilities by security experts, is a very slow and exhausting process. For this reason, a new automated classification methodology is introduced in this paper, based on the vulnerability textual descriptions from National Vulnerability Database. The proposed methodology, combines textual analysis and tree-based machine learning techniques in order to classify vulnerabilities automatically. The results of the experiments showed that the proposed methodology performed pretty well achieving an overall accuracy close to 80%.**

***Index Terms*—Software Vulnerability categorization, Cybersecurity, Machine Learning, Decision Trees, Random Forests, Gradient Boosting**

I. Introduction

Software vulnerabilities constitute a major problem in the current era of technology. The lack of public vulnerability databases led MITRE Corporation to present the Common Vulnerability and Exposures[1] (CVE) in January 1999. Furthermore, due to the vast increase of software flaws, there was a need for a categorization system in order to group similar and related vulnerabilities. Thus, the Common Weakness Enumeration[2] (CWE) system was introduced in order to categorize the software vulnerabilities and weaknesses. According to MITRE corporation[3] there are more than 900 types of CWEs and a tree-like relationship between the vulnerabilities exists at different levels of abstraction. The top level of the tree consists of 10 abstract concepts which are defined as weaknesses that are independent of any specific language or technology.

The aforementioned manual categorization usually is a time consuming procedure. Indeed, according to Spanos et al. [15], there are delays from the registration time of a vulnerability (accompanied with the technical description) to the time this vulnerability has been characterized (receiving severity score and weaknesses enumeration). More specifically, this procedure starts with the vulnerability registration by any corresponding security expert, while it ends with the aforementioned characterization by the security experts of National Vulnerability Database (NVD)[4].

The purpose of the current research is to automatically predict the software vulnerability category from the technical description of the vulnerabilities, in order to enhance the manual categorization conducted from the security experts. More precisely, the methodology tries to classify a vulnerability into one of the top-level entries of the weaknesses tree introduced by MITRE. The proposed methodology considers a similar approach with Aghaei et al. [1], since it uses the tree-like hierarchical approach, but applies tree-based machine learning algorithms instead of neural networks in favor of interpretability. Finally, this work has been implemented as part of the Platform Analysis plugin, integrated in the ECHO Early Warning System (E-EWS) platform, provided by RHEA Group and developed under the umbrella of the ECHO[5] project.

The paper is organized as follows: in Section II the related work is described, whereas in the third Section the proposed methodology is analyzed. Finally, the fourth Section presents the results of the experiments and the final Section summarizes the conclusions of the present study and provides future work directions.

II. Related work

Several research studies during the past decade took advantage of the technical descriptions of the vulnerabilities in order to predict three main aspects of vulnerabilities: a) severity, b) characteristics and c) type.

Regarding the estimation of vulnerability characteristics and severity, Yamamoto et al. [20] estimated accurately the Common Vulnerability Scoring System (CVSS) metrics by using Natural Language Processing (NLP) techniques. The algorithms that were compared in the work were Naive Bayes (NB), Latent Dirichlet Allocation (LDA), Latent Semantic Indexing (LSI), and Supervised LDA in a training set of 60000 vulnerabilities since 2013. Moreover, Le et al. [8] in their study reported that the existing NLP-based approaches suffer from concept drift and thus, they proposed a new

[1]https://cve.mitre.org/
[2]https://cwe.mitre.org/
[3]https://cwe.mitre.org/data/definitions/1000.html
[4]http://nvd.nist.gov
[5]https://echonetwork.eu/

978-1-6654-0285-9/21/$31.00 ©2021 IEEE

methodology combining both character and word features in order to address this problem and improve results. Finally, Spanos et al. [15] proposed a multi-target approach to predict the vulnerability characteristics and calculate from them, the corresponding vulnerability score. The results of their work were encouraging.

Considering the problem of vulnerability type estimation and identification, which is more related to the current research study, Neuhaus and Zimmerman [12] used topic modeling in order to analyze vulnerability trends. In their experiments, they used up to 39393 CVEs until 2009 and they found 28 topics in the CVE entries using LDA. Furthermore, Liu et al. [10] conducted experiments in text classification using three machine learning algorithms: K-Nearest Neighbors (K-NN), Naive Bayes (NB) and Support Vector Machines (SVM). They used more than 1000 vulnerabilities collected from cyber-security websites and articles, and the experiments showed that the accuracy of the SVM algorithm was over 90%. It is worth mentioning that the vulnerabilities were classified into 4 predefined categories, namely SQL injection, Buff Errors, Code injection and Access Issues. Additionally, Na et al. [11] utilized 10000 CVE entries between 1999 and 2016 from NVD. They considered the classification of CVE into CWE using the textual descriptions and conducted 3 experiments for the classification using a) the top 3 CWEs , b) the top 5 CWEs and c) the top 10 CWEs. The results showed 95.2% Precision for the first experiment, 84.2% for the second and 75% for the third one. Moreover, Aota et al. [2] utilized 28253 CVEs from NVD in order to train an RF classifier for CWE prediction. The target variable of their work consisted of 19 different CWE classes since they removed CWEs consisted of less than 100 CVEs in the dataset and thus, their results showed 96.92% classification accuracy. Finally, Aghaei et al. [1] presented a tool that automatically classifies CVEs into CWEs, called `ThreatZoom`. Using both statistical and semantic features extracted from CVE descriptions and an adaptive hierarchical neural network, which adjusts its weights based on text analytic scores and classification errors, they tried to estimate the hierarchy scheme of the CWE classes. The results of their work have indicated that the `ThreatZoom` has achieved its purpose.

It is obvious from the aforementioned, that there is no research work until now that combines the hierarchy-based approach of vulnerability categorization and tree-based machine learning algorithms in order to estimate the vulnerability type from the technical description.

III. Methodology

The proposed methodology combines NLP and machine learning techniques in order to classify vulnerabilities. In the following subsections, the steps of the proposed methodology are described extensively.

A. Data Preprocessing

The dataset was collected from the NVD database, consisting of all the vulnerabilities up to 2021. The dataset was divided into a train-set, consisting of all the vulnerabilities up to 2018 (68187 vulnerabilities), and a test-set, consisting of the vulnerabilities from 2019 up to 2021 (26687 vulnerabilities). The most recent CVE analysed was CVE-2021-3184, while the oldest one was the CVE-1999-0001. This chronological separation of the dataset aligns with previous related works of Yamamoto et al.[20] and Spanos et al. [15], and it simulates better a real classification problem, since a model which has been built from an old database is going to predict future vulnerabilities.

Each vulnerability is described through a textual description, written from security analysts. In order to proceed with the vulnerability textual descriptions, NLP [6] techniques were implemented. To begin with, a sequence pattern replacement was developed in order to squeeze words which had the same repetitive character more than two times. Thus, typographic errors were normalized, leading to a more compact Document-Term matrix (DTM) [19]. In the present paper, DTM includes the term frequencies (TF) for each document, which is in consistence with the related literature [15, 16]. Moreover, words were removed if: a) did not exceed at least three characters, or b) included other than English letters. Hence, noise, inconsistent data and errors were removed from the dataset. Additionally, lemmatization was implemented for all words included in the vocabulary. The basic principle of the lemmatization is to group together similar meaning words and analyze them as a single entity, identified by each word's lemma, leading to a much smaller vocabulary. Finally, the stop-words were removed and all the characters were converted to lowercase. Again, that led to a more meaningful vocabulary as any redundant words with low or not at all semantic meaning were filtered out of the dataset. The aforementioned procedure follows a similar consideration with related works in text mining [16, 18].

B. CWE matchmaking

Regarding the CVE to CWE matchmaking, which is a procedure that was inspired from the related work of Aghai et al. [1], in the present paper the CVEs were correlated with the most abstract CWEs provided by MITRE.

The most abstract weaknesses according to MITRE are the following:

- Improper Access Control (id:284)
- Improper Interaction Between Multiple Correctly-Behaving Entities (id:435)
- Improper Control of a Resource Through its Lifetime (id:664)
- Incorrect Calculation (id:682)
- Insufficient Control Flow Management (id:691)
- Protection Mechanism Failure (id:693)
- Incorrect Comparison (id:697)
- Improper Check of Handling of Exceptional Conditions (id:703)
- Improper Neutralization (id:707)
- Improper Adherence to Coding Standards (id:710)

The dataset of the proposed methodology consists of over 150000 vulnerabilities up to 2021. In order to correlate each of the CVEs of the dataset with the corresponding abstract CWE, CWE research concepts from the MITRE Corporation[6] were used. CVEs without correlation with a CWE were left out of the dataset, leading to 94874 vulnerabilities. Hence, by using the related weaknesses information existed in NVD it was feasible the labeling of each CVE with the top 10 abstract CWE category that it belongs to. For example, the CVE-2017-5753 is tagged in NVD as CWE-200 category. In order to correlate the example CVE-2017-5753 with one of the most abstract CWEs a recurring procedure was conducted to find all the parents of CWE-200, which in this example is the Improper Neutralization (id:707). Specifically, the number of vulnerabilities existing in each different CWE category can be found in Table I. It should be noted, that there is not a vulnerability in the dataset that was correlated with CWE-682.

TABLE I
CWE CATEGORIES

#CWE	Vulnerabilities
CWE-664	40064
CWE-707	37381
CWE-284	9575
CWE-693	4580
CWE-691	1565
CWE-710	1301
CWE-703	284
CWE-435	92
CWE-697	32

C. Machine Learning algorithms

In the current work, three machine learning algorithms that belong to the tree-based family were implemented and thoroughly compared: Random Forests, Decision Trees and XGBoost [5] from the package of scikit-learn [13].

1) Random Forests: Random Forests (RF) is a supervised ensemble machine learning algorithm proposed by Breiman [4]. The algorithm combines decision trees with additional randomness in order to achieve higher diversity and generalization. Thus, each decision tree is composed as in the Bagging algorithm [3] regarding the data sample selection but also, only a randomly selected subset of the total features is being selected by the algorithm for splitting a node. In the proposed methodology, four different numbers of total decision trees are tested (50, 100, 200 and 300) and two different numbers of maximum features ($\sqrt{totalfeatures}$ and $\sqrt{totalfeatures}/2$). After validating the different implementations, we concluded to 200 as the best parameter for the total number of decision trees and $\sqrt{totalfeatures}$ as the number of total features for the model.

2) Decision Trees: Decision Trees (DT) is a predictive algorithm and the base for many ensemble classifiers (including RF). The idea behind this algorithm is to break down the dataset into smaller sets of data and at the same time to develop an associated decision tree. The leaf nodes of the tree represent a decision while the decision nodes are features from the dataset. In the proposed methodology, an optimized version of the CART [4] algorithm was used for the experiments. Finally, as in RF algorithm the best number of total features for the classification, is again $\sqrt{totalfeatures}$.

3) XGBoost: Gradient boosting is a machine learning technique which produces a prediction model in the form of an ensemble of weak prediction models. The main idea of the algorithm is based on the slow learning that is conducted during the building of the trees. XGBoost [5] is an optimized gradient boosting library that uses parallel tree boosting. In the proposed methodology, the scikit-learn [13] wrapper interface for XGBoost [5] was used. Additionally, the numbers of gradient boosted trees tested were 50, 100, 200 and 300, concluding to 200 after the tuning.

The proposed methodology was implemented in Python programming language using scikit-learn library [13] for the classification task and spacy [17] for the text analysis. Moreover, in order to make the results interpretable, the ELI5 [7] library was used. ELI5 is a library for algorithm visualization and interpretability, giving the opportunity to explain and understand black-box models. Finally, the experiments were conducted on a PC with 32.00 GB of memory, running Ubuntu 20.04.2 LTS operating system.

Algorithm 1: Methodology

Input CVE descriptions and CWE types
0. for *each CWE tagged in CVEs of the dataset* **do**
  Take the top level parent as target
end
1. Create a new dataset with the top CWE parents
2. CVE description pre-processing using NLP techniques
3. Split the dataset into train set (up to 2018) and test set ($2019 - 2020 - 2021$)
4. Calculate TF value for each word and create the DTM matrix from train set
5. Classification using DT, RF, XGBoost
Output The predicted CWE

D. Validation

In order to validate the models and tune the corresponding parameters of the problem, a 10-Fold Cross Validation was implemented for all the classification algorithms. Thus, after the validation, the best classifier was tested on the test set.

Additionally, the following well-established evaluation metrics were used:

- Accuracy: $\sum_{i=1}^{q} \frac{TP_i}{TP_i+FP_i+TN_i+FN_i}$, where q is the number of classes, TP_i are the True Positives, FP_i are the False Positives, TN_i are the true Negatives and FN_i are the False Negatives for class i.

[6]https://cwe.mitre.org/data/downloads.html

[7]https://eli5.readthedocs.io/en/latest/overview.html

- Precision: $\frac{1}{q}\sum_{i=1}^{q}\frac{TP_i}{TP_i+FP_i}$,
- Recall: $\frac{1}{q}\sum_{i=1}^{q}\frac{TP_i}{TP_i+FN_i}$,
- F1-score: $\frac{1}{q}\sum_{j=1}^{q}\frac{2*Precision_i*Recall_i}{Precision_i+Recall_i}$

Precision, Recall and F1-score are more reliable metrics in comparison to Accuracy when the dataset is unbalanced. Specifically, overall Accuracy metric equals to the correct predictions divided by the total examples in the dataset, Precision for each class corresponds to the true positives divided by the number of examples that were predicted as positive, Recall for each class equals to the true positives divided by the number of examples that should have been identified as positive and finally, the F1-score for each class is the harmonic mean of the Precision and Recall. For each of the metrics Precision, Recall and F1-score, the overall result corresponds to the average of the respective metric for each class.

IV. Results

In this section, the results of the proposed methodology will be presented. The structure of this section is the following: initially, the tree-based approaches will be compared according to Accuracy, Precision, Recall and F1 metrics. After that, the best algorithm of the three will be compared to the methodology proposed by Aghaei et al [1]. Specifically, the confusion matrices and classification metrics of both approaches will be discussed and compared thoroughly. Finally, an analysis related to the interpretability of the methodology is presented in the last subsection.

A. Tree-based approaches

Figure 1 displays the results related to the tree-based algorithms. The comparison is performed according to the 4 validation metrics presented in the previous Section III-D. Overall, it is obvious that the XGBoost algorithm was superior compared to the Random Forest and Decision Trees at all the validation metrics. Specifically, the Accuracy of XGBoost was 0.76, while Random Forest and Decision Trees scored 0.76 and 0.66 accordingly. Moreover, the Precision of XGBoost was 0.73, while the Precision of Random Forest and Decision Trees was 0.65 and 0.44 accordingly. For the Recall metric, XGBoost scored 0.53, while the other two algorithms scored 0.46 and 0.4 accordingly. Finally, for the F1-measure, which is the harmonic average of the Precision and Recall, the XGBoost scored 0.57 while Random Forest and Decision Trees scored 0.5 and 0.41 accordingly.

B. XGBoost vs Aghaei et al. methodology

In order to compare the two approaches, a part of the work of Aghaei et al. [1] was implemented. Specifically, their Neural Network for the top-level CWE classification was developed, namely, a one layer Perceptron consisting of 9 neurons and the sigmoid function for the activation of the neurons.

As observed in Section IV-A, the XGBoost algorithm was the superior method among all the tree-based methods and this

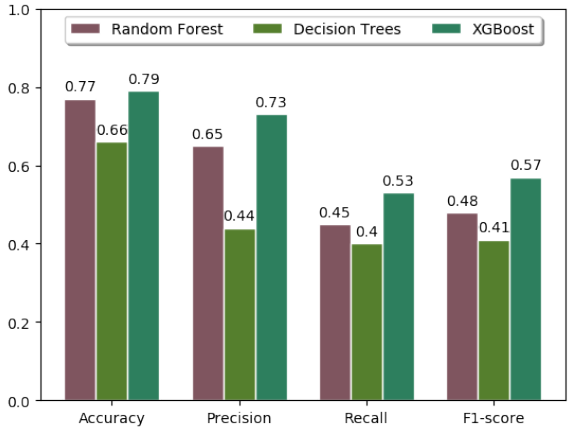

Fig. 1. Comparison of the tree-based algorithms.

is the algorithm that is compared to the study of Aghaei et al. The classification metrics for the two algorithms are depicted in Figure 2. The proposed methodology using XGBoost scored 0.79, 0.73, 0.53 and 0.57 for the Accuracy, Precision, Recall, F1-score respectively, while the methodology of Aghaei et al. scored 0.78, 0.53, 0.47 and 0.49 respectively. These results show that the proposed methodology outperforms the previous work.

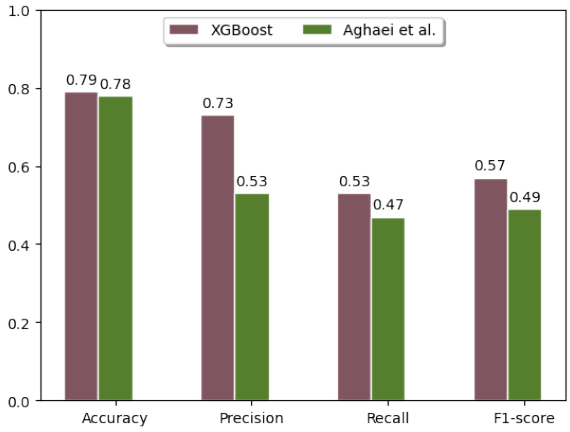

Fig. 2. XGBoost vs Aghaei et al. methodology

In order to test the performance of the two approaches more thoroughly, the confusion matrices and the classification metrics are presented. The Confusion Matrix allows the visualization of the performance of an algorithm, and thus, it is feasible to compare these two approaches. Each row of the matrix represents the instances of the true class and each column represents the instances of the predicted class. The main diagonal of the matrix represents the correct predicted cases, while the off-diagonal cells represent the misclassified cases.

As depicted in Figures 3 and 4, the overall performance of the XGBoost [5] was superior at all classes except class CWE-284. Additionally, XGBoost [5] was able to predict correctly

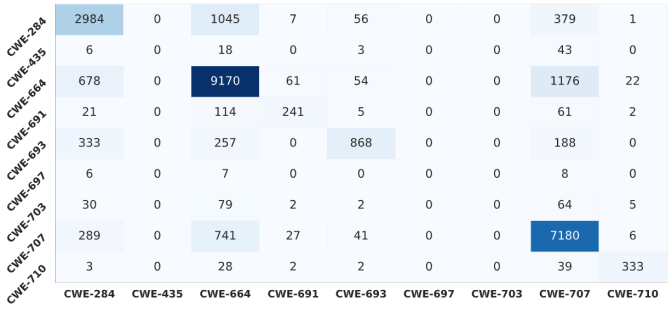

Fig. 3. Confusion Matrix of Aghaei et al. methodology.

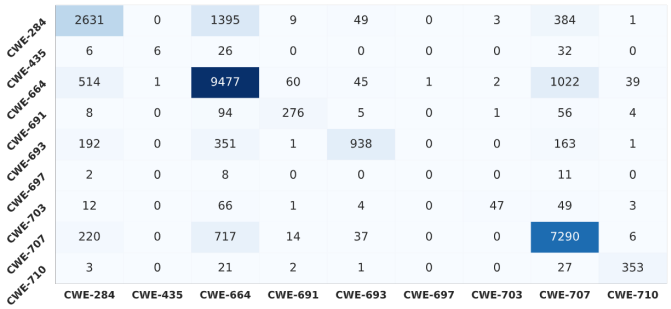

Fig. 4. Confusion Matrix of the XGBoost.

6 vulnerabilities from class CWE-435 and 47 vulnerabilities from class CWE-703. As analyzed in Table I, the classification problem is unbalanced since the classes CWE-703, CWE-435 and CWE-697 contain very few vulnerabilities.

C. Methodology interpretability

Interpretability in machine learning is a very recent and active area for many researchers and industries [7]. The goal is to make the predictions of an algorithm interpretable to a human-being in order to understand the cause of a decision. The XGBoost [5] algorithm is easily interpretable as it is based on decision trees [9]. Thus, from the analysis of the trees, a conclusion of the most prominent features that contribute the more to a specific decision could be conducted.

The most important word-features of the proposed methodology can be found in Table II. The importance of each feature is the weight that made the classifier to take a specific decision. Hence, the existence or not of words like *xss* and *injection* plays more prominent role in the decision of the algorithm in relation to words like *sensitive* or *certificate*. For example, the CVE-2019-0149 from NVD database[8] contains the following description: *Insufficient input validation in i40e driver for Intel(R) Ethernet 700 Series Controllers versions before 2.8.43 may allow an authenticated user to potentially enable a denial of service via local access.* The true class of the vulnerability is CWE-707, and the proposed methodology predicted the vulnerability category. The most important features of class CWE-707 can be found in Table III. It should be noted that the feature <BIAS> is the expected average score of the model, based only on the distribution of the training set (like the intercept parameter in a regression model). The words *validation*, *input* and *denial* from the description of the vulnerability, are giving a major contribution for that decision as depicted in Table III.

[8]https://nvd.nist.gov/vuln/detail/CVE-2019-0149

TABLE II
FEATURE IMPORTANCE

Weight	Feature
0.0787	xss
0.0727	injection
0.0656	trusted
0.0368	csrf
0.0264	traversal
0.0252	inclusion
0.0144	buffer
0.0138	dereference
0.0132	cross
0.0103	free
0.0101	memory
0.0094	race
0.0090	integer
0.0085	sensitive
0.0073	certificate

Overall, by using interpretability techniques in order to understand classification models, many security analysts and enthusiasts could be assisted in order to understand and evaluate better the results since there are many benefits that interpretability brings such as: a) reliability, b) better debugging and c) trust [14].

TABLE III
CWE-707 FEATURE IMPORTANCE

Contribution	Feature
1.829	<BIAS>
1.083	validation
0.978	input
0.177	buffer
0.131	denial
0.060	certificate
0.051	read

V. CONCLUSIONS AND FUTURE WORK

In this paper, a methodology for CWE classification through the textual descriptions of the vulnerabilities was proposed. More extensively, a matchmaking technique in order to tag each vulnerability with the most abstract CWEs that belongs to was developed. Following this consideration, the classification target consisted of 9 classes and the tree-based algorithms used in order to conclude to the best model were XGBoost, Random Forests and Decision Trees. The results highlighted that the proposed methodology achieved its purpose, since the prediction accuracy of the vulnerability category was adequately high, especially for the XGBoost algorithm.

Furthermore, in order to validate more thoroughly the methodology, a comparison between the proposed methodology with XGBoost and part of Aghaei et al. [1] work was

performed. The results showed that the proposed methodology was able to predict CWEs more precisely in comparison with the previous related work. The superiority of the proposed methodology is clearer in the prediction of classes consisting of very few vulnerabilities.

Moreover, as mentioned in Section I, the proposed methodology will be adopted in E-EWS, which is a system for Cyber-Threat Intelligence information sharing.

Finally, since the area of CVE classification through textual descriptions is at an early stage, as reflected by the related work, experimental studies could be performed in order to achieve better results from more robust algorithms and methodologies. Following this consideration, as future work, three research directions will be examined:

a Extension of the proposed methodology in order to predict not only the most abstract CWE but all the CWE hierarchy tree a vulnerability can belong to.

b Test and evaluation of more sophisticated machine and deep learning algorithms.

c Elaboration on the unbalanced dataset.

ACKNOWLEDGMENT

This work is partially funded by the European Union's Horizon 2020 Research and Innovation Programme through ECHO (https://echonetwork.eu/) project under Grant Agreement No. 830943. This paper reflects only the authors views. The European Union is not liable for any use that may be made of the information contained therein.

REFERENCES

[1] Ehsan Aghaei, Waseem Shadid, and Ehab Al-Shaer. "ThreatZoom: CVE2CWE using Hierarchical Neural Network". In: *arXiv preprint arXiv:2009.11501* (2020).

[2] Masaki Aota et al. "Automation of Vulnerability Classification from its Description using Machine Learning". In: *2020 IEEE Symposium on Computers and Communications (ISCC)*. IEEE. 2020, pp. 1–7.

[3] Leo Breiman. "Bagging predictors". In: *Machine learning* 24.2 (1996), pp. 123–140.

[4] Leo Breiman. "Random forests". In: *UC Berkeley TR567* (1999).

[5] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[6] Gobinda G Chowdhury. "Natural language processing". In: *Annual review of information science and technology* 37.1 (2003), pp. 51–89.

[7] Leilani H Gilpin et al. "Explaining explanations: An overview of interpretability of machine learning". In: *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*. IEEE. 2018, pp. 80–89.

[8] Triet Huynh Minh Le, Bushra Sabir, and Muhammad Ali Babar. "Automated software vulnerability assessment with concept drift". In: *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE. 2019, pp. 371–382.

[9] Yawen Li et al. "Application of interpretable machine learning models for the intelligent decision". In: *Neurocomputing* 333 (2019), pp. 273–283.

[10] Chungang Liu, Jianhua Li, and Xiuzhen Chen. "Network vulnerability analysis using text mining". In: *Asian Conference on Intelligent Information and Database Systems*. Springer. 2012, pp. 274–283.

[11] Sarang Na, Taeeun Kim, and Hwankuk Kim. "A study on the classification of common vulnerabilities and exposures using naive bayes". In: *International Conference on Broadband and Wireless Computing, Communication and Applications*. Springer. 2016, pp. 657–662.

[12] Stephan Neuhaus and Thomas Zimmermann. "Security trend analysis with cve topic models". In: *2010 IEEE 21st International Symposium on Software Reliability Engineering*. IEEE. 2010, pp. 111–120.

[13] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[14] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Model-agnostic interpretability of machine learning". In: *arXiv preprint arXiv:1606.05386* (2016).

[15] Georgios Spanos and Lefteris Angelis. "A multi-target approach to estimate software vulnerability characteristics and severity scores". In: *Journal of Systems and Software* 146 (2018), pp. 152–166.

[16] Georgios Spanos, Lefteris Angelis, and Dimitrios Toloudis. "Assessment of vulnerability severity using text mining". In: *Proceedings of the 21st Pan-Hellenic Conference on Informatics*. 2017, pp. 1–6.

[17] Shiliang Sun, Chen Luo, and Junyu Chen. "A review of natural language processing techniques for opinion mining systems". In: *Information fusion* 36 (2017), pp. 10–25.

[18] Dimitrios Toloudis, Georgios Spanos, and Lefteris Angelis. "Associating the Severity of Vulnerabilities with their Description". In: *International Conference on Advanced Information Systems Engineering*. Springer. 2016, pp. 231–242.

[19] G Williams. "Hands-on data science with R text mining". In: *no. January* (2016).

[20] Yasuhiro Yamamoto, Daisuke Miyamoto, and Masaya Nakayama. "Text-mining approach for estimating vulnerability score". In: *2015 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)*. IEEE. 2015, pp. 67–73.