The Effect of Topic Modelling on Prediction of Criticality Levels of Software Vulnerabilities

Prarna Mehta[1], Shubhangi Aggarwal[2], Abhishek Tandon[1*]
E-mail: pmmphilscholar@gmail.com, shubhagg1206@gmail.com, atandon@or.du.ac.in
[1]Deparment of Operational Research, University of Delhi, Delhi, India.
[2]Bhagwan Parshuram Institute of Technology, Delhi.
*****Corresponding author

Keywords: software vulnerabilities, topic modelling, machine learning, supervised learning, CVSS, text mining

Received: August 30, 2021

In this day and age, software is an indispensable part of our per diem endeavours, thereby keeping a check on exploitable vulnerabilities has become a vital function of a software firm. The motivation of this paper is to have better understanding of vulnerabilities, creating a tool for the industry practitioners to identify a critical vulnerability that could be detrimental for the firm's assets. In this article, 1999 vulnerabilities related to Google Chrome was analysed to understand the behaviour of vulnerabilities. The identification of trends and patterns using topic modelling technique led to extraction of topics. The extricated topics were then implemented in 10 classifiers to foresee the criticality of the vulnerability. The resulting performances were also assessed with the classifiers without implementing topic modelling techniques. A 10-fold validation was conducted on the suggested prediction model.

Povzetek: Metoda za ugotavljanje občutljivosti programske opreme je narejena s pomočjo tem.

1 Introduction

Enslavement towards software has been ferociously intensifying by leaps and bounds in the present era, consequently, a call for unswerving software system has become the need of the hour. The snowballing complexities in order to meet the demands of user and to survive in the industry, often escalates the vulnerabilities in the software. Any software employed in a project/application is subjected to some inadvertent shortcomings, in other words vulnerabilities that might turn out to be a liability. Such exposure encourages an/a attacker/hacker to disturb the software project/application, hampering the security of the system. A secure system is, thus a highly demanded pursuit for a developer as well as a consumer guaranteeing a smooth working even under any outbreak. Nevertheless, in order to avoid any attack, these vulnerabilities have to be deeply analysed by a software development team in order to fortify a system. Vulnerabilities in a software project/application liberates an attacker to squander vital data as well as interfere with the security. Countless episodes of losses due to vulnerability attack has been reported causing not only monetary loss but as well as eminence of a company. For instance, due to virus, namely, Code Red Worm, a loss of $2.6 billion was incurred as reported in the study by (Telang & Wattal, 2007). The National Vulnerability Database (NVD) aims at amassing statistics on software vulnerabilities and has a record of 152780 vulnerabilities till date[1]. The incidents due to vulnerabilities have been reported to Computer Emergency Response Team (CERT) and around 53117 security incidents were handled by Indian CERT team in the year 2017, nonetheless, the number hiked to 208456 in 2018 whereas it was 394499 in 2019[2]. Looking at the alarming rate of proliferating records on vulnerabilities, it draws attention of researchers to examine the scenario for the betterment of the industry.

The risk attached to these software vulnerabilities, given the fact that gigantic amount of classified data is getting accrued on the daily basis, if corrective measures not taken can lead to serious collisions whereas, on the other hand, mammoth-volume, textual data on vulnerability accumulating each year needs to be tamed for better analysis and research in the field of software vulnerabilities (Malhotra, 2021). Moreover, this gives a direction to a software maintenance team concentrate on highly vulnerable part in the software project/application curtailing false positive as well (Stuckman et al., 2016). This brings the focus to develop an efficient algorithm that condenses the corpus as well as converges the limited resources towards a highly vulnerable part. In this paper, topic modelling, state of the art technique is deployed to reduce the textual descriptions into meaningful clusters

[1] Source: https://nvd.nist.gov/vuln/search/results?form_type=Basic&results_type=overview&search_type=all

[2] https://www.cert-in.org.in

called topics. Three different Topic modelling algorithms were considered for this study, namely Latent Semantic Indexing (LSI), Latent Dirichlet Allocation (LDiA) and lastly, Non-Negative Matrix Factorization (NMF), to asses each of their performance when combined with the prediction model.

The colossal quantity of vulnerability data can be reduced by labelling them as critical and non-critical. The prediction of the criticality of vulnerabilities aids software maintenance team to drive the limited resources towards the critical vulnerabilities. However, the vulnerability prediction model as two aspects to it explicitly, the features of the vulnerability data and the classifier. For this study, Logistic Regression (LR), Linear Discriminant Analysis (LDA), K-Nearest Neighbours (KNN), Decision Tree (DT), Artificial Neural Network (ANN), Naïve Bayes' (NB), Linear Support Vector Machine (LSVM), Support Vector Machine (SVM), Random Forest (RF), and lastly, Gaussian Naïve Bayes' (GNB).

This study is noteworthy for the fact that it helps in mathematically modelling vulnerability text data, thereby furnishing with meaningful results obtained empirically. The core objective of this study is to build a highly accurate vulnerability prediction model to categorize vulnerability data into meaningful topics that trains state-of-the art classifiers to renders enriched prediction model. In order to achieve this goal, vulnerability data of Google Chrome, mined from the National Vulnerability Database (NVD), was pre-processed to configure topics using three Topic modelling techniques. The deduced topics contributed as training set for the learning algorithms to envisage the criticality of the vulnerability identified in Google Chrome. The models are validated using k-fold validation technique and compared with prediction model without considering procuring topics as a feature reduction scheme. The objectives of the research study is fulfilled by resolving the following research questions (RQ) that were investigated in this study,

RQ1. What is the performance of topic modelling when combined with classifiers?

RQ2. What is the performance of the classifier without incorporating any of the topic modelling technique?

RQ3. Which of the Machine Learning (ML) classifiers shows improvement in the performance?

In our knowledge, there has not been any work based on integration and comparative study of topic modelling techniques and machine learning classifiers. The dataset used to perform this study has also not been implemented in any previous literature. The key contribution manifested in this research article are, (1) to develop vulnerability prediction model using different topic modelling techniques and Machine learning classifiers, (2) to examine the performance of the developed models (3) to reconnoitre the effect of not incorporating topic modelling (4) adding new aspects to the literature for the experts to benefit from.

The paper is spread over five sections: Past literature articles are discussed elaborately in section 2, in order to overcome the research gap, a methodology is propositioned in section 3, the proposed model is illustrated and validated in section 4, Threats to internal as well as external Validity is examined in section 5 and to sum up the study section 5 concludes the study.

2 Related work

There are plenty of literature on vulnerability prediction in software project/application using machine learning techniques as well as feature reduction tools, establishing suitable results. (Walden et al., 2014) compared the effect of software metrics with that of bag of words on the vulnerability prediction model. A lot of work has been done in other areas of vulnerability like developing conventional models, optimisation model, release plans, cost models. (Kansal et al., 2016) developed a mathematical model for vulnerability detection and a cost model for patching after the detection. (Zerkane, 2018) examined the effect of vulnerabilities in software defined networking using CVSS score. (Kansal et al., 2018) made an effort of optimising the cost of after release maintenance issue by combining vulnerability fixing and fault fixing into single patch.

Many mathematical optimization techniques have been used to optimally prioritise vulnerabilities. (Sharma et al., 2019) uses MCDM techniques, namely VIKOR and TOPSIS to prioritise vulnerabilities. A novel optimization tool, VULCAN was developed by (Farris et al., 2018) to manage vulnerabilities with respect to exposure and remediation. A comparative study between best worst method and AHP was studied by (Anjum, Agarwal, et al., 2020), following which, (Anjum, Kapur, et al., 2020) integrated MCDM and ML technique to develop bi-objection optimization problem prioritising the most critical vulnerability. (Narang et al., 2018) incorporated the effect of software vulnerabilities inter dependency attribute in prioritising them in accordance to their critical levels with the help of DEMATEL.

Some Researchers are examining different feature reduction schemes to enhance the performances of the vulnerability prediction model. (Stuckman et al., 2016) examined the influence of dimension reduction techniques like PCA, Feature Synthesis, and their respective variant, on foreseeing vulnerabilities located in open source applications in PHP. (Ji et al., 2018) describes briefly different technologies implemented along with discussing pioneer work in the areas of automatic vulnerability detection, exploitation and patching. (Theisen & Williams, 2020) have used different software metrics along with features obtained through text mining and analysed the performance of Random Forest, Decision Trees, Logistic Regression and Naive Bayes. (Kalouptsoglou et al., 2020) develops model using deep learning and software metrics with promising results taking into consideration multiple projects for generalised results. A vulnerability prediction model was developed by (Filus et al., 2020) using RNN and CNN. An inter-

comparative study was performed by (Wu et al., 2017) to asses deep learning techniques like LSTM, CNN as well as reviewing the conventional machine learning techniques. (Shahriar & Haddad, 2016) implemented LSI to obtain smaller source code causing object injection vulnerability in a system. (Kudjo et al., 2020) framed a model using bellwether analysis to select subset for vulnerability prediction. (Rehurek & Sojka, 2010) discusses the importance of applying topic modelling techniques. A framework called SySeVR was developed by (Li et al., 2021) using deep learning techniques to identify semantics and syntax characteristics to spot the vulnerabilities in C/C++ source codes. A correlation was established between software metrics and the prevailing vulnerabilities by (Alves et al., 2016) determining answers to multiple research questions. A complete structure was suggested by (Kumar & Sharma, 2017) to manage vulnerabilities in an optimal manner.

3 Proposed methodology

In this section, the framework of the study is explained step by step as depicted by figure 1. In figure 1, orange depicts phase 1 that is extraction of vulnerability datasets, green represents phase 2 that is the extracted dataset is pre-processed and prepared for further analysis, blue represents phase 3 that is feature mining and training of classifiers using tokenised data as well as the generated topics as features and lastly black represents phase 4 where, the performance of the prediction model is evaluated.

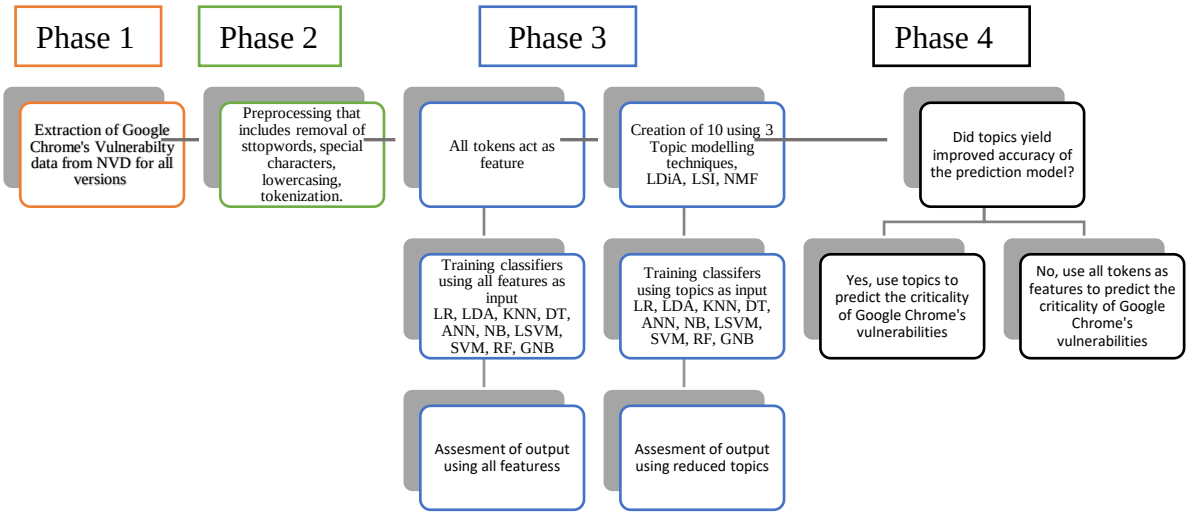


Figure 1: A general framework of the proposed study.

Exploring huge text data manually is taxing and arduous job which have greater chances to be erroneous and have discrepancies whereas shrinking data into relevant topics with the help of topic modelling can be considered as solution. Topic modelling is considered to be the most efficient unsupervised data-mining algorithm, discovering relationships between text data (Vanamala et al., 2020). This condenses the dimension of data by amputating superfluous features that do not weigh in further analysis. For our analysis, we have considered three topic modelling algorithms, LSI, LDiA and NMF. The rudimentary concept behind topic modelling is to convert large corpus into vectors with the help of term frequency or inverse term frequency thereby, dividing and optimized by a probability model or matrix factorization into topics which is an array of words or tokens.

LSI is a robust topic mining technique, having a knack for noise resistance and transforming large dimensional vector spaces to smaller dimensional vector spaces with the help of singular value decomposition. (Papadimitriou et al., 2000) endeavours to study the mathematics behind the LSI performance and its's ability to divulge in statistical properties of corpus. LSI and LDiA both have probabilistic approach where as NMF is a matrix factorization paradigm that decomposes high dimensional array to a non-negative and low dimensional one. Non-Negative being the only criteria, NMF uses term frequency-inverse document frequency (TF-IDF) whereas LDiA and LSI uses frequency of bag of words or term frequency (TF) for feature extraction since the paradigm reads only positive integer frequencies and not a real number.

The topics hence generated directs toward ameliorated results when read as an input by machine learning classifiers. For the experiment we have selected 10 classifiers that are most commonly used to assess any suggested model, namely Logistic Regression (LR), Linear Discriminant Analysis (LDA), K-Nearest Neighbours (KNN), Decision Tree (DT), Artificial Neural Network (ANN), Naïve Bayes' (NB), Linear Support

Vector Machine (LSVM), Support Vector Machine (SVM), Random Forest (RF), and lastly, Gaussian Naïve Bayes' (GNB).

By far, there are many evaluation markers to assess machine learning tools and traditionally ones are True Positives, True Negatives, False Negatives and False Positives that form the confusion matrix. In this study, the performance of the 10 classifiers were assessed with the help of Accuracy, F-measure, Recall and Precision. These measure were well explained by (Bulut et al., 2019) in their corresponding study and mostly used in past literature (Dam et al., 2016; Theisen & Williams, 2020).

4 Numerical analysis

4.1 Data collection

For the numerical, a dataset of 1999 vulnerabilities captured in Google's product "chrome" was collected manually from National Vulnerability Database (NVD). Google Chrome web application was chosen because its abundantly utilized web browser in the market (http://www.netmarketshare.com) for e-banking, social media, information sharing, consequently making it highly exploitable application to access sensitive data of a user. Additionally, many researchers have used google chrome to conduct their respective experiment, for instance, (Kudjo et al., 2020; Nguyen et al., 2016; Roumani et al., 2015).

The data consists of Vulnerabilities Ids, Summary of the vulnerabilities and CVSS Severity for all the versions (more than 20 available). The CVSS is computed in two ways, CVSS 3.0 and CVSS 2.0. For our analysis, the latter score has been considered, since the score value and the severity level were available for all listed vulnerabilities. The CVSS score quantities the criticality of a vulnerability numerically between 1 to 10. The criticality level of the stated vulnerabilities was tagged into three categories, namely, High, Medium and Low. For easy computation and binary classification, the medium severity level was united with Low severity level as non-critical vulnerabilities, whereas High severity level was termed as critical vulnerability. Table 1 describes the dataset.

Table 1: Description of vulnerability dataset.

Project	Google Chrome
No. of Vulnerabilities	1999
Range of years	2021-2011
Versions	<20
No. of critical Vulnerabilities	510
No. of Non-critical Vulnerabilities	1489

4.2 Data pre-processing

Subsequently, the vulnerability description is mined to extract useful information with the help of pre-processing methods, thereby optimising the results. Special characters, punctuation, blank spaces occupy memory spaces as well as hamper the result of the experiment, hence removing such irrelevant information acts as a corrective measure (Vijayarani et al., 2015). Next, with the help of Python packages, Natural Language Toolkit (NLTK) and pandas, the words more than 3 letter in the vulnerability description column were retrieved in lowercase, replacing other special characters by a blank space, the stop words were eliminated and each document was tokenized into list of words for further experiment. The other packages put to use were 'numpy' and 'matplotlib' for data management and visualization whereas 'sklearn' library abetted in importing TFIDF and Count vectorizer for feature extraction, LDiA, LSI and NMF for topic modelling and lastly, machine learning classifiers to determine the criticality of the vulnerabilities.

4.3 Topic modelling

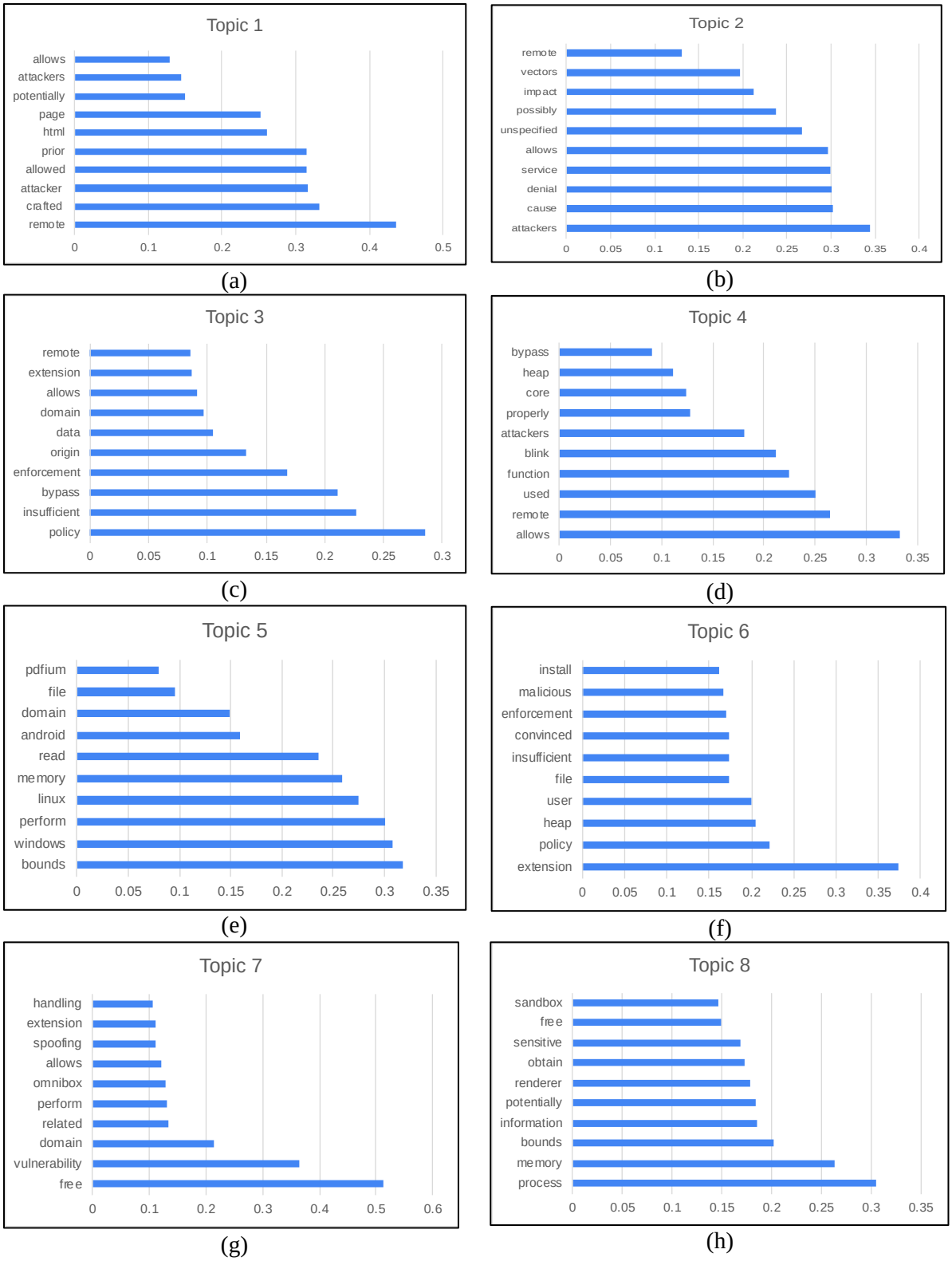
The list of words or token obtained after pre-processing vulnerability data is considered as features of the respective study. The description of each vulnerability is converted to their respective feature vector that stores frequency of each token in a particular vulnerability document. Following which, Count Vectorizer and TF-IDF screens these features further by assigning a weight of importance. This not only resolves the tedious job to handle large corpus but also cuts down the expense involved and the computation time.

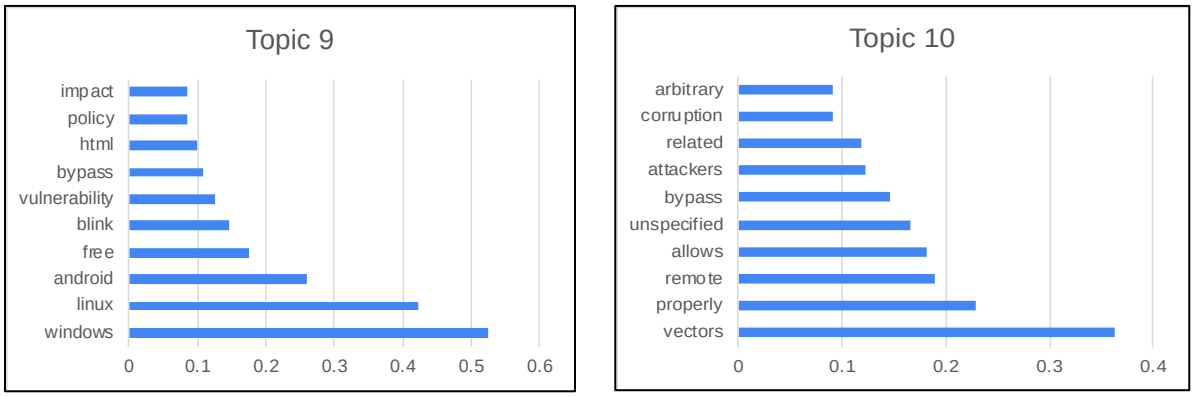
To improve the performance of the prediction models, all the vulnerability documents are iterated to capitulate unique tokens as dictionary or bag-of-words. The subset of 100 words was considered as input for LDiA, LSI and NMF topic models along with number of topics as 10. The number of words and topics was chosen as it is observed to work well in the past literature (Dam et al., 2016; Mounika et al., 2019; Vanamala et al., 2020). Each topic created using topic modelling

techniques is a linear combination of unique words and their respective weightage. For example, Topic 0 obtained from LSI topic modelling technique is represented as:

[('**remote**' * 0.4358473796441817) + ('**crafted**' * 0.3319362678157508) + ('**attacker**' * 0.3159864058882791) + ('**allowed**' * 0.31502006087883194) + ('**prior**' * 0.31419760562296395) + ('**html**' * 0.26029878325415207) + ('**page**' * 0.25179755400065057) + ('**potentially**' * 0.14941933391958886) + ('**attackers**' * 0.14537824144767378) + ('**allows**' * 0.12949807741439195)]

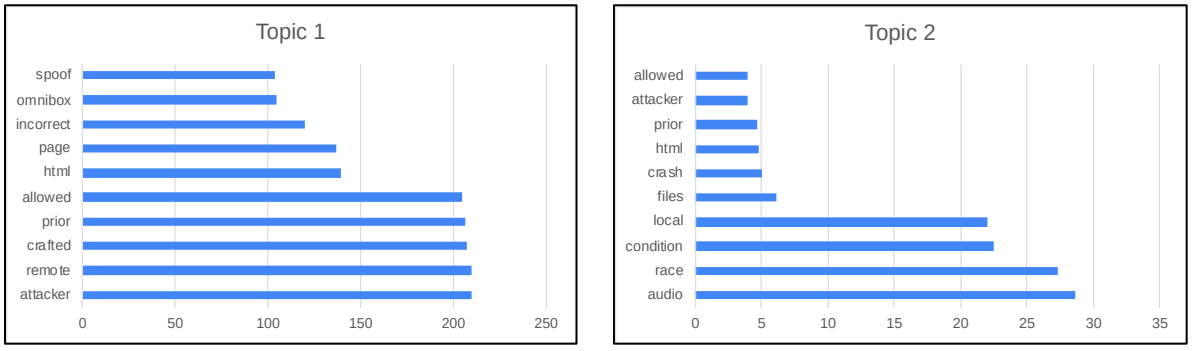
From the above equation, it can be noted that the tokens: "**remote, crafted, attacker, allowed, prior, html, page, potentially, attackers, allows**" conjointly form Topic 0 as conferred by LSI topic modelling result. The numerical part attached to each token in Topic 0 signifies the weightage of the word in the respective topic. The topics created by LSI, LDiA and NMF and each token's relative importance in their respective topics is depicted in figure 2, 3 and 4.



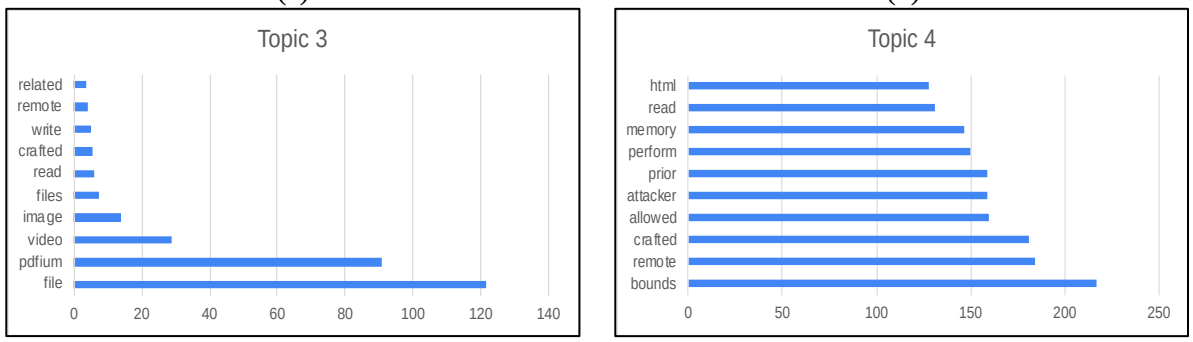


(i) (j)

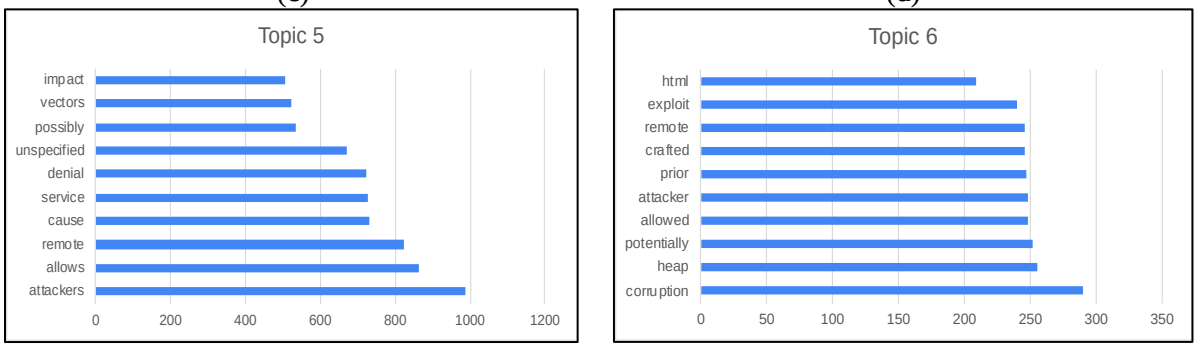
Fig 2: Relative importance of tokens in respective topics when LSI was performed.



(a) (b)



(c) (d)



(e) (f)

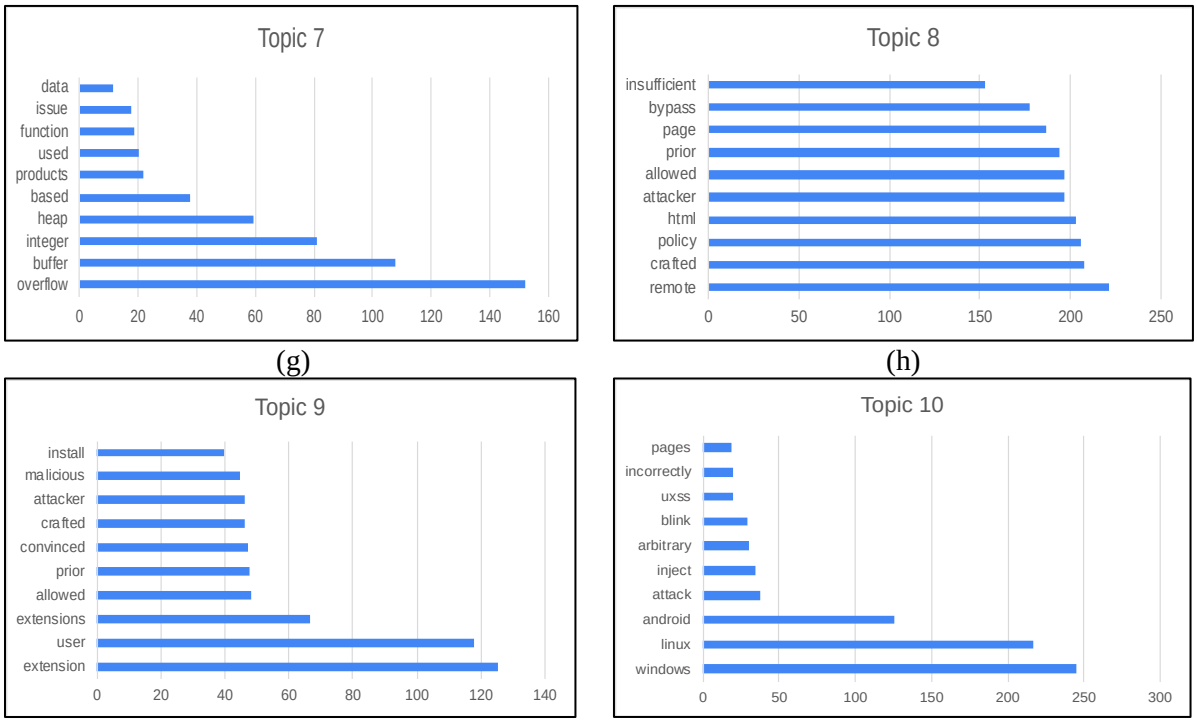
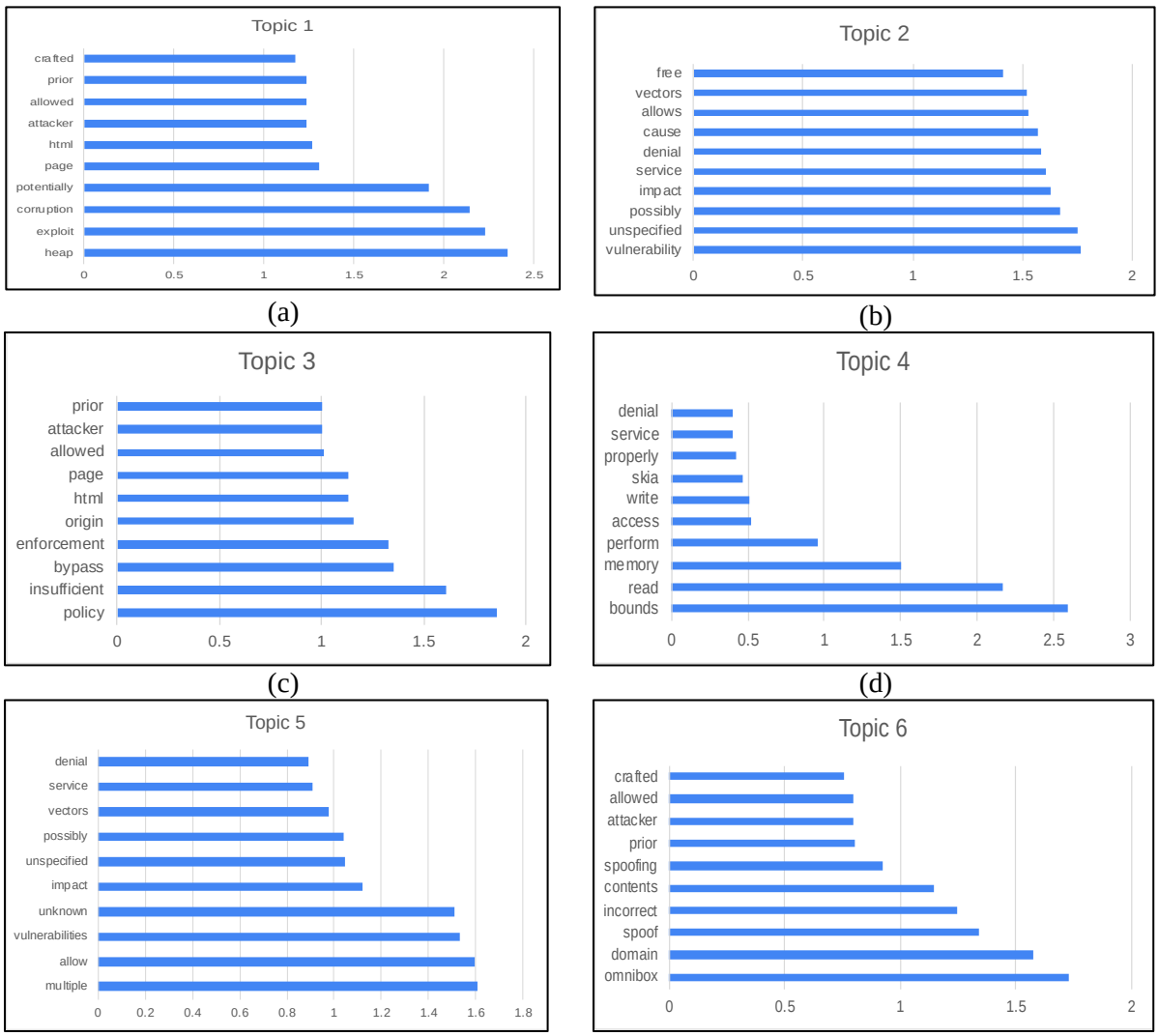


Fig 3: Relative importance of tokens in respective topics when LDA was performed.



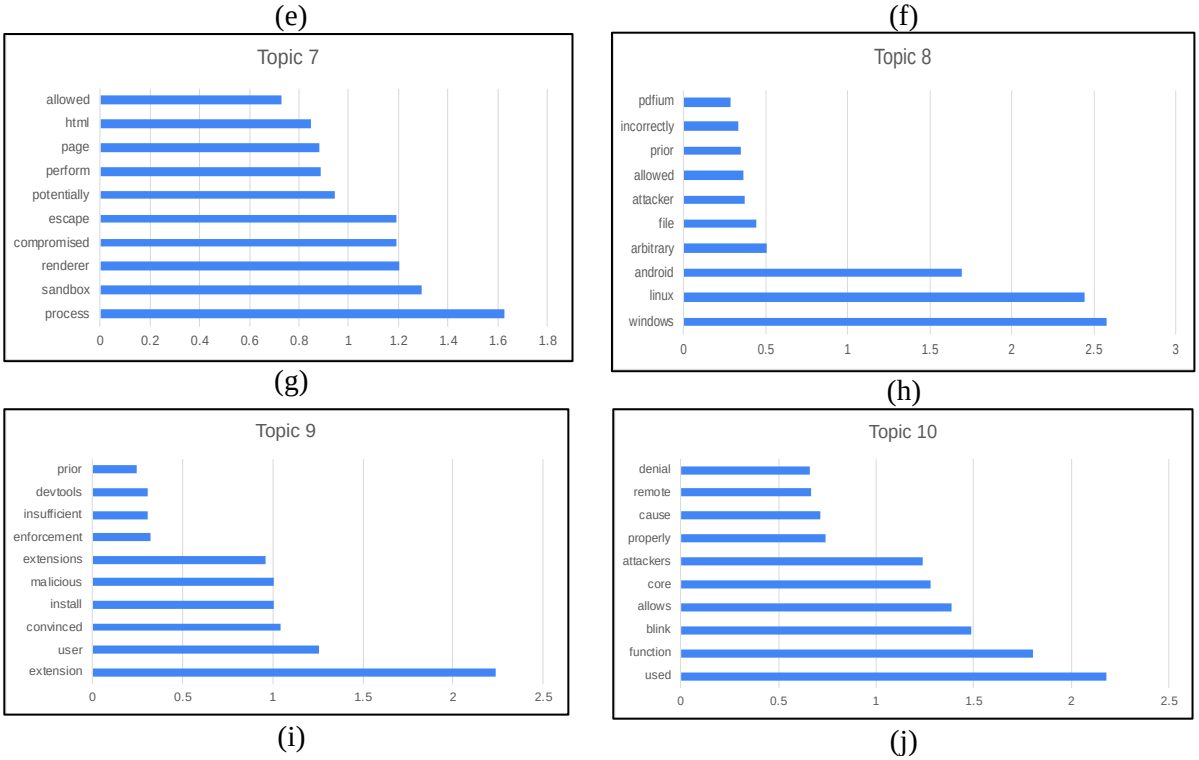


Figure 4: Relative importance of tokens in respective topics when NMF was performed.

From fig 2 (a)-(j), the tokens: "**remote, attackers, policy, allows, bounds, extension, free, process, windows, vendors**" in the respective topics have the highest weightage ranging between 0.285 and 0.526 when LSI was performed. The most influencing tokens in respective topics as depicted in fig 3(a)-(j) obtained from LDA were "**attacker, audio, file, bounds, attackers, corruption, overflow, remote, extension, windows**". Lastly, from NMF analysis, tokens "**heap, vulnerability, policy, bounds, multiple, omnibox, process, windows, extension, used**" weighed the most in their corresponding topic clusters.

4.4 Evaluations

RQ1. What is the performance of topic modelling when combined with classifiers?

The results of topics extracted and used as input for the 10 classifiers is given by table 2. From the table, it can be observed that when LR, KNN, DT, ANN, NB, RF and GNB combined with LSI gives an accuracy level of 0.8175, 0.82, 0.8025, 0.8175, 0.7975, 0.8175 and 0.7975 respectively. NMF combined with LDA, LSVM and SVM gives an accuracy level of 0.8275, 0.82, and 0.785. Lastly, it can be observed that LDiA has the poorest performance when combined the 10 classifiers with the accuracy level ranging between 0.7175 and 0.7525. A pictorial representation of accuracy levels of all classifiers with respect to topic modelling technique is given by figure 5.

From figure 6 and table 2, F1- measure can be analysed for different classifiers subject to a given topic modelling technique. Classifiers with LDiA has overall same level of F1-measure except for the classifier KNN that shows highest level of F1-measure at 0.7269. On the other hand, LSI's performance with the classifiers has an average F1-measure around 0.8 with highest at 0.8206 for ANN classifier and lowest for SVM at 0.7415. Last of all, NMF with the classifiers depicts a mixed performance of F1-measure ranging between the lowermost at 0.6462 for NB and reaching the peak at 0.8225 for LDA.

Lastly, the tabular results of performance measures, Recall and Precision are given by table 2 and line diagram given by figure 7 and figure 8. Classifier GNB with topic modelling technique, NMF results in lowest recall value at 0.6925 whereas lowermost recall value for classifier SVM with LSI was 0.7225, and lastly for classifier DT with LDiA, it was 0.7175. However, NB with the topic modelling technique NMF has the lowest Precision value at 0.5662, classifier DT with topic modelling technique LSI has the lowest precision value at 0.8005, but multiple classifiers had poor Precision value with LDiA at 0.5663. A low recall and high precision value imply how accurately the model is returning positive predicted value. For all the low recall values recorded, it was observed that they had more or less high precision values implying that the suggested model labels a critical vulnerability correctly, however the number of false negatives is high due to high precision which indicates that the model is sometimes missing out critical vulnerabilities. In general, one cannot help put notice, the opposite behaviour of F1-measure and precision, whereas accuracy is in parallel with recall implying the goodness fit of the proposed

model. But from the results it was also noted that many classifiers had a high recall and high precision values signifying the fact that the model was accurately labelling a critical vulnerability.

Table 2: Output of classifiers' performance measures.

NMF				LSI				LDiA				Classifiers ' Name
Precision	**Recall**	**F1-Score**	**Acc.**	**Precision**	**Recall**	**F1-Score**	**Acc.**	**Precision**	**Recall**	**F1-Score**	**Acc.**	
0.7451	0.77	0.7419	0.772	0.8194	0.817	0.8184	0.8175	0.5663	0.752	0.646	0.7525	**LR**
0.8204	0.82	0.8225	0.827	0.8173	0.812	0.8146	0.8125	0.5663	0.752	0.646	0.7525	**LDA**
0.7982	0.79	0.7978	0.797	0.8213	0.82	0.8206	0.82	0.7193	0.745	0.726	0.745	**KNN**
0.7729	0.78	0.7759	0.78	0.8005	0.802	0.8015	0.8025	0.6593	0.717	0.674	0.7175	**DT**
0.8005	0.80	0.8015	0.802	0.8251	0.817	0.8206	0.8175	0.5663	0.752	0.646	0.7525	**ANN**
0.5662	0.75	0.6462	0.752	0.8257	0.797	0.8061	0.7975	0.5663	0.752	0.646	0.7525	**NB**
0.8128	0.82	0.8152	0.82	0.8194	0.817	0.8184	0.8175	0.5663	0.752	0.646	0.7525	**LSVM**
0.7636	0.78	0.7535	0.785	0.8326	0.722	0.7415	0.7225	0.5663	0.752	0.646	0.7525	**SVM**
0.7584	0.78	0.7380	0.78	0.8125	0.817	0.8145	0.8175	0.5663	0.752	0.646	0.7525	**RF**
0.8271	0.692	0.7133	0.69	0.8257	0.797	0.8061	0.7975	0.6682	0.747	0.660	0.7475	**GNB**

RQ2. What is the performance of the classifier without incorporating any of the topic modelling technique?
The line graph illustrated by figure 5, 6, 7, 8 represents the accuracy, F1-measure, Recall and Precision levels for classifier when using topic modelling techniques and without topic. Modelling techniques. Even though accuracy level is oscillating between 0.71 and 0.89 whereas F-measure fluctuating between 0.7203 and 0.8904, it can be observed that the classifiers mostly show high precision and high recall values except for the classifier LDA. A high recall indicates that the model is predicting a vulnerability as non-critical but a critical vulnerability is not labelled as non-critical. However, high precision value with high recall is considered as perfect combination since the model results in high number of true positives implying that the critical vulnerabilities are predicted correctly.

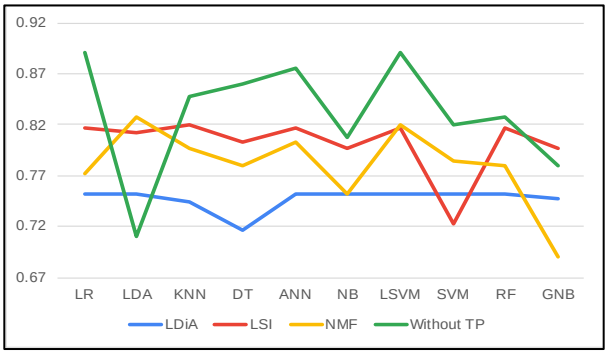


Figure 5: Comparative study of TP vs without TP using accuracy.

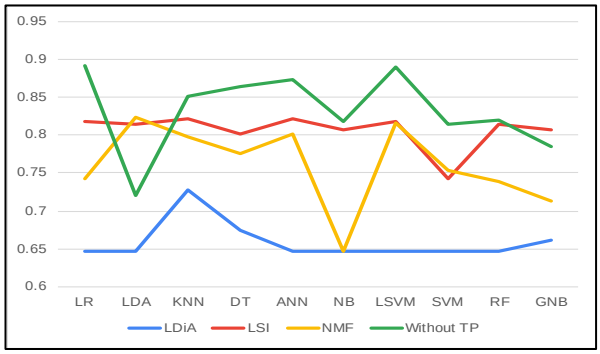


Figure 6: Comparative study of TP vs without TP using F1 score.

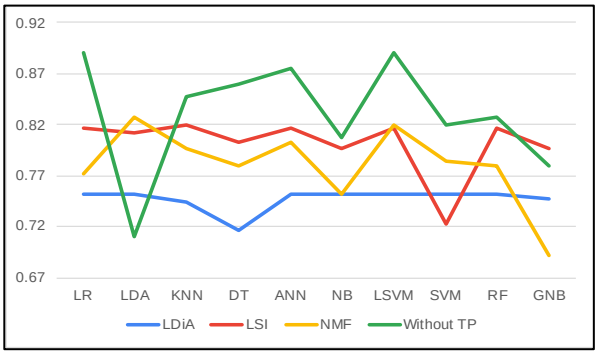


Figure 7: Comparative study of TP vs without TP using recall.

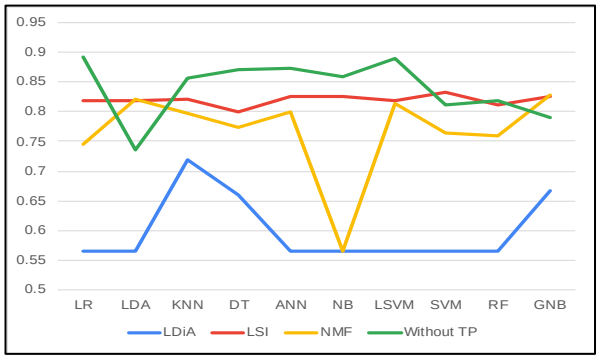


Figure 8: Comparative study of TP vs without TP using precision.

RQ3. Which of the Machine Learning (ML) classifiers shows improvement in the performance?
Looking at the figures 5,6,7,8 and table 2, the Machine learning classifier GNB performs the best when combined with topic modelling technique LSI and machine learning classifier LDA performs best
when combined with topic modelling technique NMF. While other classifiers for the given dataset show no sign of improvement when the features are reduced and combined into topics. The reason behind no improvement is simply due to over estimation while using topic modelling techniques.

4.6 Model validation

In order to study the impact of features extracted mechanically by topic modelling techniques on 10 classifiers while developing vulnerability prediction model, a 10 cross-fold validation experiment was conducted. The vulnerability dataset was divided into 10 folds: 9 parts as training set while 1 part to test the model. Hence for each unique topic modelling technique and each classifier, 10 different performances results were obtained. Figure 9, 10, 11 depicts the averaged-out performance measure for each classifier under three different topic modelling technique. Accuracy was used as performance measure for this validation experiment.

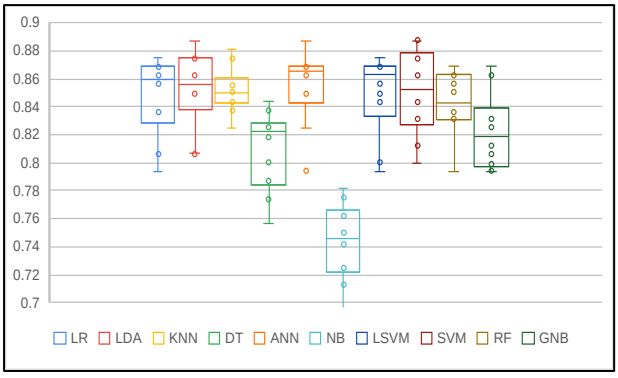


Figure 9: 10-fold validation for LSI model.

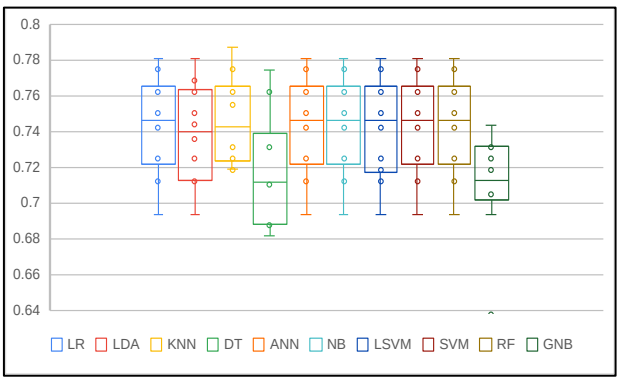


Figure 10: 10-fold validation for NMF model.

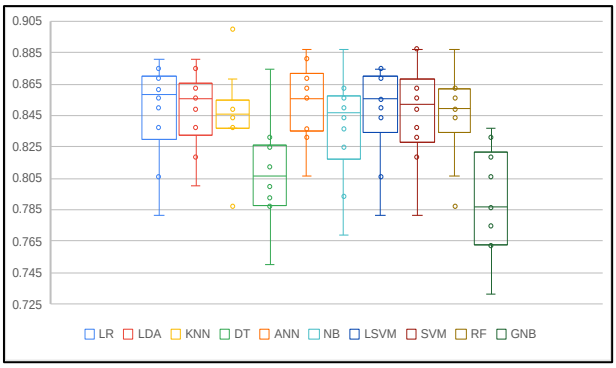


Figure 11: 10-fold validation for LDiA model.

From figure 9, it can be noted that vulnerability prediction model using LSI and ANN outperforms with accuracy being 0.8555 whereas LSI fused with classifier NB performs the least with accuracy level at 0.7429, while other classifiers with LSI performed between the range. Analysing figure 10, one cannot help but notice the poor performance of the classifier, GNB with accuracy at 0.7898, on the other hand LDA has the highest accuracy at 0.8499. Lastly, from figure 11, the accuracy level ranging between 0.7098 (GNB) and 0.7456 (KNN) for LDiA is observed. The classifiers, namely ANN, NB, LSVM, SVM and RF have almost same accuracy level as 0.7429. Overall, after 10-fold validation, LSI is most impactful feature reduction tool when conjointly performed with the machine learning tool, ANN.

5 Threats to validity

A Pragmatic study can be intimidated by number of limitations internally as well as externally, making it important to be worth of discussion. While a threat to internal validity describes the elements that might have an impact on the study's output on the other hand a threat to external validity aims at the generalizing the output. In this study, the vulnerability description was mined to extract features and CVSS score to determine the criticality of the respective vulnerability for the prediction model however, other factors like CVSS metrics, were not taken into considerations, which might have an impact on the performance of the prediction model. Another threat to internal validity of the study is the vulnerability records was not documented during the period of this study. A statistical test was not conducted to verify the statistical significance of the results which gives a direction for future work.
Subsequently, the threats to external validity in this study was the dataset was limited to one project which cannot infer generalized results for other datasets, adding biasness to the output. The reason behind this is that a vulnerability of high criticality level is not inevitably of same criticality in a different project dataset. In this study

we have worked on a web application's vulnerability dataset, but the results may differ for other applications in written in different languages or Android application. The performance measures to assess the learning algorithms for the prediction model were Accuracy, F-measure, recall and precision, nonetheless there other measures as well for instance, Area under Receiver operating characteristics curve, Welch t-test, cliff's delta effect size etc. For the empirical results, 10 machine learning algorithms were deployed, but there are many more algorithms to be validated for universal result.

6 Conclusion

This study focuses on the impact of topic modelling techniques on the performances of the classifiers labelling vulnerabilities as critical or non-critical. The topic extracted from the vulnerability description condenses the textual data, thereby captures the significance portion and eradicating the irrelevant text. In order to perform the analysis, we have extracted a vulnerability dataset for the most used web application, Google Chrome. The topics were generated with the help of three topic modelling techniques namely, LSI, NMF, LDiA. These spawned topics were used as input in 10 most commonly used classifiers. The results of the suggested methodology were compared with that of the classifiers without integrating topic modelling inputs.

All in all, one can conclude from the performed experiment that most of the classifiers perform best when not combined with topic modelling techniques except for GNB and LDA. Classifier GNB with LSI has an accuracy of 0.7975 whereas when LDA performs with NMF has an accuracy of 0.8275. However, individually considering the classifiers performance with topic modelling technique one can state that the performances are at par excellence.

Future work can be directed toward three courses. Firstly, the proposed methodology can be validated on software application database such as PHP application, web applications, mobile applications and applications from various fields like finance, education, banking, energy utility etc. The second direction is incorporating techniques to balance the datasets. An imbalanced dataset does not result in high accuracy and performance of the prediction model. Hence incorporating sampling techniques can enhance the results. The third approach is that for this study, the vulnerability description is used to extract features, but there are multiple factors that improve and deliver a generalised result.

References

[1] Alves, H., Fonseca, B., & Antunes, N. (2016). Software metrics and security vulnerabilities: dataset and exploratory study. 2016 12th European Dependable Computing Conference (EDCC),

[2] Anjum, M., Agarwal, V., Kapur, P., & Khatri, S. K. (2020). Two-phase methodology for prioritization and utility assessment of software vulnerabilities. International Journal of System Assurance Engineering and Management, 11(2), 289-300.

[3] Anjum, M., Kapur, P., Agarwal, V., & Khatri, S. K. (2020). Evaluation and Selection of Software Vulnerabilities. International Journal of Reliability, Quality and Safety Engineering, 27(05), 2040014.

[4] Bulut, F. G., Altunel, H., & Tosun, A. (2019). Predicting software vulnerabilities using topic modeling with issues. 2019 4th International Conference on Computer Science and Engineering (UBMK),

[5] Dam, H. K., Tran, T., & Pham, T. (2016). A deep language model for software code. in workshop on Naturalness of Software (NL+SE), co- located with the 24th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE),

[6] Farris, K. A., Shah, A., Cybenko, G., Ganesan, R., & Jajodia, S. (2018). Vulcon: A system for vulnerability prioritization, mitigation, and management. ACM Transactions on Privacy and Security (TOPS), 21(4), 1-28.

[7] Filus, K., Siavvas, M., Domańska, J., & Gelenbe, E. (2020). The random neural network as a bonding model for software vulnerability prediction. Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems,

[8] Ji, T., Wu, Y., Wang, C., Zhang, X., & Wang, Z. (2018). The coming era of alphahacking?: A survey of automatic software vulnerability detection, exploitation and patching techniques. 2018 IEEE third international conference on data science in cyberspace (DSC),

[9] Kalouptsoglou, I., Siavvas, M., Tsoukalas, D., & Kehagias, D. (2020). Cross-project vulnerability prediction based on software metrics and deep learning. International Conference on Computational Science and Its Applications,

[10] Kansal, Y., Kapur, P., & Kumar, D. (2016). Assessing optimal patch release time for vulnerable software systems. 2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH),

[11] Kansal, Y., Kumar, U., Kumar, D., & Kapur, P. K. (2018). Fixing of Faults and Vulnerabilities via Single Patch. In Quality, IT and Business Operations (pp. 175-190). Springer.

[12] Kudjo, P. K., Chen, J., Mensah, S., Amankwah, R., & Kudjo, C. (2020). The effect of Bellwether analysis on software vulnerability severity prediction models. Software Quality Journal, 1-34.

[13] Thanh Tung Khuat, My Hanh Le (2016). Optimizing Parameters of Software Effort Estimation Models using Directed Artificial Bee Colony Algorithm. Informatica 40 (2016) 427–436

[14] Kumar, M., & Sharma, A. (2017). An integrated framework for software vulnerability detection, analysis and mitigation: an autonomic system. Sādhanā, 42(9), 1481-1493.

[15] Li, Z., Zou, D., Xu, S., Jin, H., Zhu, Y., & Chen, Z. (2021). SySeVR: A framework for using deep learning to detect software vulnerabilities. IEEE Transactions on Dependable and Secure Computing.

[16] Malhotra, R. (2021). Severity Prediction of Software Vulnerabilities Using Textual Data. Proceedings of International Conference on Recent Trends in Machine Learning, IoT, Smart Cities and Applications.

[17] Mounika, V., Yuan, X., & Bandaru, K. (2019). Analyzing CVE Database Using Unsupervised Topic Modelling. 2019 International Conference on Computational Science and Computational Intelligence (CSCI),

[18] Narang, S., Kapur, P., Damodaran, D., & Majumdar, R. (2018). Prioritizing types of vulnerability on the basis of their severity in multi-version software systems using DEMATEL technique. 2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO),

[19] Nguyen, V. H., Dashevskyi, S., & Massacci, F. (2016). An automatic method for assessing the versions affected by a vulnerability. Empirical Software Engineering, 21(6), 2268-2297.

[20] Hrvoje Karna, Sven Gotovac and Linda Vicković (2020). Data Mining Approach to Effort Modeling on Agile Software Projects. Informatica 44 (2020) 231–239

[21] Papadimitriou, C. H., Raghavan, P., Tamaki, H., & Vempala, S. (2000). Latent semantic indexing: A probabilistic analysis. Journal of Computer and System Sciences, 61(2), 217-235.

[22] Abhishek Tandon, Neha & Anu G. Aggarwal (2020). Testing coverage-based reliability modelling for multi-release open-source software incorporating fault reduction factor. *Cycle Reliab Saf Eng* **9**, 425–435 (2020). https://doi.org/10.1007/s41872-020-00148-7

[23] Rehurek, R., & Sojka, P. (2010). Software framework for topic modelling with large corpora. In Proceedings of the LREC 2010 workshop on new challenges for NLP frameworks,

[24] Roumani, Y., Nwankpa, J. K., & Roumani, Y. F. (2015). Time series modeling of vulnerabilities. Computers & Security, 51, 32-40.

[25] Ouanes Aissaoui, Abdelkrim Amirat and Fadila Atil. (2014). A Model-Based Framework for Building Self-Adaptive Distributed Software. Informatica 38 (2014) 289–306.

[26] PK Kapur, Anu G Aggarwal, Abhishek Tandon (2012). A unified approach for developing two-dimensional software reliability model. International Journal of Operational Research Vol. 13, No. 3, pp- 318-337.

[27] Sharma, R., Sibal, R., & Sabharwal, S. (2019). Software vulnerability prioritization: A comparative study using TOPSIS and VIKOR techniques. In System performance and management analytics (pp. 405-418). Springer.

[28] Stuckman, J., Walden, J., & Scandariato, R. (2016). The effect of dimensionality reduction on software vulnerability prediction models. . IEEE Transactions on Reliability, 66(1), 17-37.

[29] Telang, R., & Wattal, S. (2007). An empirical analysis of the impact of software vulnerability announcements on firm stock price. IEEE Transactions on Software engineering, 33(8), 544-557.

[30] Abhishek Tandon, Anu G Aggarwal, Nidhi Nijhawan (2016). An NHPP SRGM with change point and multiple releases. International Journal of Information Systems in the Service Sector (IJISSS), Vol 8 (4), pp: 56-68.

[31] Theisen, C., & Williams, L. (2020). Better together: Comparing vulnerability prediction models. Information and Software Technology, 119, 106204.

[32] Vanamala, M., Yuan, X., & Roy, K. (2020). Topic Modeling and Classification Of Common Vulnerabilities And Exposures Database. 2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD),

[33] Vijayarani, S., Ilamathi, M. J., & Nithya, M. (2015). Preprocessing techniques for text mining-an overview. International Journal of Computer Science & Communication Networks, 5(1), 7-16.

[34] PK Kapur, RB Garg, Udayan Chanda, Abhishek Tandon (2010). Development of software reliability growth model incorporating enhancement of features and related release policy. International Journal of Systems Assurance Engineering and Management. Vol (1), pp-52-58.

[35] Walden, J., Stuckman, J., & Scandariato, R. (2014). Predicting vulnerable components: Software metrics vs text mining. 2014 IEEE 25th international symposium on software reliability engineering,

[36] Wu, F., Wang, J., Liu, J., & Wang, W. (2017). Vulnerability detection with deep learning. 2017 3rd IEEE International Conference on Computer and Communications (ICCC),

[37] Zerkane, S. (2018). Security Analysis and Access Control Enforcement through Software Defined Networks Brest].

[38] Roman Yu. Tsarev, Alexey S. Chernigovskiy, Elena N. Shtarik and Andrey V. Shtarik (2017). Modular Integrated Probabilistic Model of Software Reliability Estimation. Informatica 40 (2016) 125–132.

[39] Kapur, P., Tandon, A., & Kaur, G. (2010). Multi up-gradation software reliability model. Paper presented at the 2010 2nd International Conference on Reliability, Safety and Hazard-Risk-Based Technologies and Physics-of-Failure Methods (ICRESH).

[40] A.G. Aggarwal, N. Gandhi, V. Verma, A. Tandon

[41] Multi-release software reliability growth assessment: an approach incorporating fault reduction factor and imperfect debugging Int. J. Math. Oper. Res., 15 (4) (2019), pp. 446-463.

[42] A. Tandon, A.G. Aggarwal. Testing coverage-based reliability modelling for multi-release open-source software incorporating fault reduction factor
Life Cycle Reliability and Safety Engineering, 9 (4) (2020), pp. 425-435