Empirical Software Engineering (2024) 29:113
https://doi.org/10.1007/s10664-024-10519-8

Explaining poor performance of text-based machine learning models for vulnerability detection

Kollin Napier[1] · **Tanmay Bhowmik**[2] · **Zhiqian Chen**[2]

Accepted: 17 June 2024 / Published online: 22 July 2024

Abstract
With an increase of severity in software vulnerabilities, machine learning models are being adopted to combat this threat. Given the possibilities towards usage of such models, research in this area has introduced various approaches. Although models may differ in performance, there is an overall lack of explainability in understanding how a model learns and predicts. Furthermore, recent research suggests that models perform poorly in detecting vulnerabilities when interpreting source code as text, known as "text-based" models. To help explain this poor performance, we explore the dimensions of explainability. From recent studies on text-based models, we experiment with removal of overlapping features present in training and testing datasets, deemed "cross-cutting". We conduct scenario experiments removing such "cross-cutting" data and reassessing model performance. Based on the results, we examine how removal of these "cross-cutting" features may affect model performance. Our results show that removal of "cross-cutting" features may provide greater performance of models in general, thus leading to explainable dimensions regarding data dependency and agnostic models. Overall, we conclude that model performance can be improved, and explainable aspects of such models can be identified via empirical analysis of the models' performance.

Keywords Text-based analysis · Machine learning models · Explainability

Communicated by: Yuan Zhang

✉ Kollin Napier
kollin.napier@mgccc.edu

Tanmay Bhowmik
tbhowmik@cse.msstate.edu

Zhiqian Chen
zchen@cse.msstate.edu

[1] Mississippi Artificial Intelligence Network (MAIN), Mississippi Gulf Coast Community College, Perkinston, Mississippi, USA

[2] Department of Computer Science and Engineering, Mississippi State University, Mississippi State, Mississippi, USA

1 Introduction

As the demand for new features and capabilities within software grows, maintaining systems becomes increasingly challenging due to their rising complexity. This is especially true in areas related to software security (Shin et al. 2010; Alenezi and Zarour 2020). Lapses in software security can open pathways for known or unknown vulnerabilities within a system. Resources such as the Common Vulnerability and Exposures (CVE) database[1] and National Vulnerability Database (NVD)[2] offer public knowledge about known vulnerabilities, including details on mitigation. Exploiting such vulnerabilities can lead to data breaches affecting individuals and companies (Saleem and Naveed 2020).

As developers create and maintain systems, ensuring complete knowledge of and accounting for possible security vulnerabilities that currently exist or may emerge in the future can be overwhelming (Czerwonka et al. 2015). Developers may inadvertently introduce vulnerabilities through implementation, reuse of components, or other oversights (Oliveira et al. 2014; Gkortzis et al. 2021). Processes such as code reviews are generally implemented to help alleviate such concerns, but may be ineffective (Mäntylä and Lassenius 2008; Edmundson et al. 2013). These approaches often require extensive manual effort and still leave room for potential vulnerabilities due to various reasons such as lack of awareness or focus (Braz et al. 2022).

Efforts to combat vulnerabilities have been explored in various capacities (Liu et al. 2012; Zeng et al. 2020). Currently, research related to the use of newer methods involving machine learning models is gaining traction (Jie et al. 2016; Ijaz et al. 2019; Spreitzenbarth et al. 2015; Yamaguchi et al. 2011; Ghaffarian and Shahriari 2017; Grieco et al. 2016; Lin et al. 2017). Generally, models are trained and tested within a specific system to determine their effectiveness in detecting vulnerabilities based on provided vulnerable and non-vulnerable samples. Such efforts may be limited to settings of single systems (Chernis and Verma 2018; Shar et al. 2014; Harer et al. 2018). Expanded efforts explore the potential of using models trained on one or more systems and tested across other systems (Scandariato et al. 2014; Ban et al. 2019; Lin et al. 2019; Liu et al. 2020).

Although models may perform well in certain scenarios, the general consensus is that machine learning research for vulnerability detection is still in an immature state, and further exploration is needed, such as comprehensibility of models (Ghaffarian and Shahriari 2017; Lin et al. 2020). To expand the current field of research, some focus is shifting towards understanding how a model functions. These efforts go beyond determining what problem can be solved and ask why or how a model is making a prediction (Zhang et al. 2020). This concept leads to an area defined as explainability or interpretability of machine learning models (Lipton 2018). While models have become popular and are continuously improving, there is a significant lack of information regarding why a model performs the way it does, leaving those who implement or use the models with unanswered questions (Sotgiu et al. 2022).

Recent studies, such as a survey conducted by Burkart and Huber (2021), provide details about the aspect of explainability. They outline dimensions that may affect the interpretation of prediction results from supervised machine learning models. One such dimension may include data dependency, where a model's performance is contingent upon the data it is trained and tested on. Other dimensions focus on explainability determined before or after a model is built and executed, how models function compared to others, and more.

[1] https://cve.mitre.org/

[2] https://nvd.nist.gov/

Little work has been done on the explainability of machine learning models in the context of vulnerability detection. Mosolygó et al. (2021) proposed a line-level JavaScript vulnerability prediction model emphasizing both explainability and accuracy. Their unique approach leverages word2vec embedding and vector distances to identify code patterns. Despite providing significant insights, their research is constrained by limitations related to the dataset size and the subjectivity of the evaluation process.

In conducting an empirical study on using text-based machine learning models for vulnerability detection, Napier et al. (2023) interpreted extracted functions from instances of "fixed" and "vulnerable" GitHub source code as text. Their motivation was based on the availability of vulnerable code snippets from online Q&A platforms, such as Stack Overflow. Their approach experimented with a variety of models and techniques related to training and testing models within and across projects and CWE vulnerability types. They conclude that text-based machine learning models may not be effective. Despite providing analysis and discussion in their empirical study, they did not explore any specific approach towards explainability.

Napier and Bhowmik (2022) offered further analysis by determining a correlation between the average precision scores observed from the models and the overlap of features still present throughout the data. They labeled these features as being "cross-cutting" due to their overlap between the "fixed" and "vulnerable" function pairings used for training and testing. They observed a negative correlation related to both projects and CWE vulnerability type data. Despite suggesting that these features may contribute to the lack of effectiveness, they acknowledged that correlation does not imply causation and that other aspects could play a role.

In order to improve our understanding and the performance of text-based machine learning models for vulnerability detection, we perform a series of experiments related to defining explainable dimensions and the removal of "cross-cutting" features. This analysis is grounded in the empirical results from Napier et al. (2023) and correlation studies from Napier and Bhowmik (2022). From these considerations, we pose the following research question:

– **What effect does the removal of "cross-cutting" features have on the performance of text-based machine learning models for vulnerability detection?**

In this study, we embark on a detailed exploration of text-based machine learning models, with an emphasis on their role in vulnerability detection. We conduct comprehensive experiments, probing the consequences of eliminating "cross-cutting" features from the training and testing datasets on the models' performance. Alongside, we delve into the inherent explainability of these models, identifying the significant dimensions that guide this process. We focus on two key dimensions: "Agnostic" and "Data Dependent". The term "Agnostic" refers to an approach where the model's explanation is extracted in a post-hoc manner, treating the initial model as a "black box". This means that the explanation is flexible and not limited to the specifics of the model, its explanation, or its representation. On the other hand, "Data Dependent" refers to the explainability that stems from the dependency of a model on the data being used, where the nature and quality of data play a crucial role in interpreting the model's output. We also consider other potential explainable dimensions, which, though not directly relevant to our study, could apply in a broader context. Our findings point to a significant improvement in model performance upon the systematic removal of "cross-cutting" features. That is, our proposed method may improve accuracy and interpretability if the "cross-cutting" feature does so. Thus, our paper's primary contributions are:

– **An investigation into the role and impact of "cross-cutting" features on the performance of text-based machine learning models for vulnerability detection**

- **A demonstration of how the systematic removal of "cross-cutting" features from training and testing datasets can enhance model performance**
- **An exploration into the dimensions of explainability in the context of text-based models for vulnerability detection, aimed at identifying key dimensions that can potentially explain the general performance of such models**

The remainder of this paper is organized as follows: Section 2 delves into related work on machine learning models for vulnerability detection, emphasizing explainability. Section 3 outlines the methodology of our study, including the data, models, and techniques utilized. Section 4 details our experiments and assesses the results under two scenarios of "cross-cutting" feature removal. Section 5 presents our research question and formulates hypotheses. Section 6 offers a thorough discussion on the explainability aspect of our models, identifying significant dimensions of explainability. Section 7 addresses potential threats to the validity of our study, and Section 8 concludes our work, proposing potential future research directions.

2 Related Work

2.1 Machine Learning Models For Vulnerability Detection

Usage of machine learning models for vulnerability detection has been explored in a variety of scenarios. Hovsepyan et al. (2012) present initial usage of raw source code as text for vulnerability prediction. Scandariato et al. (2014) explore usage of text mining source code and applying models to predict vulnerabilities across applications. Perl et al. (2015) introduced new methods of detecting vulnerabilities by using code-metric analysis and create a vulnerability dataset. Li et al. (2018) expand such analysis methods by introducing representations of software known as code gadgets. Lin et al. (2019) explore an alternate approach of representations using abstract syntax trees (ASTs). Recent approaches such as those presented by Li et al. (2018); Ban et al. (2019); Li et al. (2021a, b), among others explore further usage of models including newer methods of neural networks with expanded efforts related to cross-domain vulnerability detection. Although such works present unique results given their scenarios, there is very limited work on important aspects, such as detailed empirical analysis and explainability of the models' performance.

Napier et al. (2023) present an empirical study of text-based machine learning models for vulnerability detection. They perform a series of experiments regarding the usage of such models given datasets of vulnerable source code. To avoid issues of class imbalance, they elect to generate pairings based on extracted functions from instances of "fixed" and "vulnerable" source code. Using a variety of machine learning models, natural language processing techniques, and their own data processing methods, they determine an appropriate combination of these approaches for usage in vulnerability detection. Based on their results and statistical analysis, they conclude that text-based machine learning models are not effective in detecting vulnerabilities within or across projects and vulnerability types. They further state that additional analysis is needed in determining the cause and possible improvements.

Napier and Bhowmik (2022) provide analysis of empirical cross-domain vulnerability prediction results from Napier et al. (2023) to examine potential causes of poor performance. Although the empirical data was pre-processed prior to training and testing models, they determine an overlap of features may exist between various function pairings of "fixed" and "vulnerable" data. For example, a set of features may be unique to a "fixed"/"vulnerable"

pairing but only existing in the list of "fixed" features, but in another pairings similar features may occur in the list of "vulnerable" features. For such occurrences, they deem these features as "cross-cutting". Using the Pearson correlation coefficient, they observe correlations between the "cross-cutting" features and the model prediction scores. Their analysis provides instances of negative correlations for both projects and CWE vulnerability types. They conclude that such "cross-cutting" features may play a role in lack of performance for such models. However, they state that the correlation does not mean causation and further exploration is needed regarding explainability.

2.2 Explainability of Machine Learning Models

Zhang et al. (2020) present a survey on explainable recommendation of machine learning models to bridge the gap between explanations and recommendations when developing models. The approach of explainable recommendation is used to help answer questions about why a model performs in a manner and how better humans can understand it. Inclusion of such an explainability aspect can introduce certain traits, including transparency and trustworthiness. Typically, the lack thereof may exist either from model output or the mechanism not being understood and can lead to problems from a user perspective. For their survey, they review previous literature and provide a taxonomy for classifying such research. They also provide a summary of how explainable recommendation can be applicable for different scenarios. Overall, they expect that new perspectives throughout the field can help advance such development.

Burkart and Huber (2021) provide an in-depth survey regarding the explainability of supervised machine learning models. In their work, they discuss the needs for explainability as it applies to a variety of domains. Explainability may be necessary to provide clarity from the human perspective as to why a model is performing in a specific manner and providing a result which may or may not be desired. Related to the work of utilizing machine learning models for vulnerability detection, the domain of recommendation systems is generally applicable given the potential of future research in this area. In such a system, it is essential to understand why it identifies a particular vulnerability or offers a specific resolution (Zhang et al. 2020). They also provide problem definitions regarding supervised machine learning and ways to gain interpretability. Overall, they provide necessary details regarding the relevance of the problems and the necessity to overcome them while working with models.

Zhou et al. (2021) conduct a comprehensive investigation into the explainability of supervised machine learning models, addressing the importance of explainability in various domains. The necessity for explainability is underscored, especially from a human perspective, to understand why a model behaves in a specific way and produces certain results. Particularly, in the realm of machine learning models used for vulnerability detection, the principles applicable to recommendation systems are relevant, given the anticipated future research in this area. It becomes crucial to comprehend why the system identifies a particular vulnerability or proposes a specific resolution. They also offer problem definitions pertaining to supervised machine learning and strategies to achieve interpretability. On the whole, they furnish crucial details regarding the significance of these issues and the imperative to overcome them while dealing with models. This comprehensive study is a valuable resource for understanding the evaluation of machine learning explanations, accentuating the importance of human-centric evaluations and guiding future research in this field.

Duval (2019) provides a detailed review of the explainability of supervised machine learning models, emphasizing its importance across various domains. The author focuses on understanding why a model operates in a certain way from a human perspective. The relevance of the author's work is underscored in the realm of recommendation systems, particularly in understanding the recognition of vulnerabilities or proposal of solutions. This work outlines problem definitions for supervised machine learning and methods to achieve interpretability. The paper also inspects post-hoc explanation techniques that elucidate a model's behavior and results after training, highlighting that the boundary between interpretation methods isn't rigid. Although an in-depth discussion on interpretability dimensions and interpretation methods is lacking, the author claims to provide a rigorous mathematical definition of dominant interpretation methods. The author thereby underlines the necessity to overcome these challenges in working with models.

Mosolygó et al. (2021) introduce a prototype for JavaScript vulnerability prediction that emphasizes both granularity of prediction and explainability. Instead of traditional function or file-level analysis, their line-level model utilizes word2vec embeddings and vector distances to identify vulnerabilities with a particular focus on semantically similar but syntactically different code patterns. Unlike many state-of-the-art methods, their model provides intelligible explanations for its predictions, enhancing its usefulness to developers. The authors claim that their approach was able to flag up to 60% of known vulnerabilities in some instances, reaching 100% in specific cases, with only a fraction of the code base. Contrasting this, our study focuses on function-level C/C++ vulnerability analysis. We use previously paired "fixed" and "vulnerable" functions, remove "cross-cutting" features, and compare our models to prior empirical work. Our technique of identifying and removing "cross-cutting" features, which may affect the data for models, can be generalized across all languages. Our approach concentrates on exploring explainable dimensions within text-based machine learning models for vulnerability detection.

Given the vast number of vulnerabilities that are disclosed each day, it can be a tedious process to gather adequate information and present potential severity. To combat this problem, Shahid and Debar (2021) introduce an approach to utilize natural language processing (NLP) to identify the severity of a vulnerability from its textual description. Their process uses multiple Bidirectional Encoder Representations from Transformers (BERT) models corresponding to the various metrics needed to establish a common vulnerability scoring systems (CVSS) vector. Their model was able to achieve severity scores which were like those provided by human experts. Their explainability was based on usage of the most relevant input terms for prediction. Such terms can mimic those selected by human experts creating an explanation comprehensive for users to interpret.

With the lack of explainability for machine learning models, Sotgiu et al. (2022) present a systematic method to help mitigate such issues by highlighting problems in model design and training. With a combination of the Devign dataset (Zhou et al. 2019) as well as the CodeBERT model (Feng et al. 2020), they fine tune the process in identifying source code likely to introduce vulnerabilities in each project. They identified three insights from their considered model including: 1) may heavily rely on features which do not improve discovery yet can be necessary for structure identification; 2) token may be interpreted in different ways depending on context; and 3) vulnerable prediction when encountering unknown tokens can create bias. Overall, their work focuses on usage of tokens rather than model specific details and provides insight into how model improvement may be achieved with explainable efforts.

To enhance the explainability of supervised learning models in machine learning (ML) and artificial intelligence (AI), various dimensions have been delineated by Burkart and Huber (2021). These dimensions help in establishing a taxonomy and framing problem definitions. Key dimensions include ante-hoc versus post-hoc, instance/local versus model/global, specific versus agnostic, and data-independent versus data-dependent explanations. In the following sections, we briefly describe each of these dimensions, emphasizing their distinctions and interactions, particularly in the context of contrasting paired opposites.

Ante-Hoc: These models are inherently interpretable, designed to be straightforward and comprehensible from their inception. They are "white-box" models, where their internal workings are clear and transparent, allowing for immediate understanding without additional explanation. This innate clarity is a key characteristic of Ante-Hoc models (Burkart and Huber 2021).

Post-Hoc: In contrast, Post-Hoc interpretability applies to "black-box" models, which are not inherently transparent. Explanations in Post-Hoc models are generated after the model has been created, providing insight into their operation. This approach involves creating explanators that can interpret the model's output and decisions, making the originally opaque model understandable in hindsight (Burkart and Huber 2021).

Comparison and Contrast: Ante-Hoc and Post-Hoc models present a contrasting approach to explainability in ML and AI. Ante-Hoc models are transparent from the beginning, akin to a clear glass through which one can see the inner workings. They require no extra effort for understanding their processes. On the other hand, Post-Hoc models are like a mystery box, initially opaque, and their operations become clear only after explanations are applied post-development. This distinction is crucial in our taxonomy, as it highlights the difference between models that are inherently understandable versus those that require additional tools for interpretability, addressing different needs and contexts in ML and AI applications.

Instance/Local: These explanations are tailored to individual instances within a machine learning model. They provide detailed insights for specific data points or predictions, allowing for an understanding of the model's decision-making process on a case-by-case basis. This "microscopic" view focuses on the specifics of individual outputs, using various forms of data representation (Burkart and Huber 2021).

Model/Global: In contrast, Model/Global explanations provide a holistic overview of the model's functioning. They offer insights into the model's behavior across all instances, enabling a general understanding of its mechanisms and decision patterns. This "macroscopic" perspective looks at the model as a whole, often using parameters or general characteristics of the model (Burkart and Huber 2021).

Comparison and Contrast: Instance/Local and Model/Global explanations serve different purposes in ML and AI. While Instance/Local is like zooming in with a microscope to understand individual decisions of a model, Model/Global is like zooming out to view the entire landscape of the model's functioning. The former is crucial for detailed, specific insights, and the latter for overarching patterns and general model behaviors, together providing a comprehensive understanding of ML and AI systems.

Specific: This type of interpretability is unique to a particular model or class of models. It means that the explanation method works only for a specific set of models, often characterized by inherent transparency or "white-box" nature. These models are designed to be understood in their specific context, with explanations tailored to their unique structure and functioning (Burkart and Huber 2021).

Agnostic: Agnostic interpretability, on the other hand, is versatile and adaptable. It applies to a wide range of models, typically "black-box" models, where the internal workings are not immediately transparent. This approach extracts explanations in a post-hoc manner, offering flexibility in understanding various models irrespective of their specific design (Ribeiro et al. 2016; Molnar 2018; Burkart and Huber 2021).

Comparison and Contrast: Specific interpretability is akin to a custom-made suit, tailored precisely for a specific model, providing clarity and understanding for that particular model only. In contrast, Agnostic interpretability is like a one-size-fits-all garment, adaptable to various models, providing a broad understanding that spans across different model architectures. This contrast highlights the trade-off between tailored precision and versatile applicability in ML and AI explainability.

Data Dependent: This approach to explainability hinges on the specific data used by a model. The interpretation of the model's behavior is directly linked to the characteristics of the data it processes. In scenarios like vulnerability detection using source code, the quality and relevance of the data play a critical role in how the model's decisions are interpreted and understood (Burkart and Huber 2021).

Data Independent: In contrast, Data Independent interpretability does not rely on the specifics of the data used. Instead, the model's explainability is derived from other factors inherent to the model or external explanators. This approach offers a broader understanding of the model's functionality, independent of the data it processes (Burkart and Huber 2021).

Comparison and Contrast: Data Dependent explainability is like using a recipe specifically tailored to the ingredients at hand, where the outcome depends on those specific ingredients. Data Independent explainability, however, is more like a cooking technique that can be applied regardless of the ingredients used, focusing more on the method than the specifics of the ingredients.

Overall, the research of explainability towards supervised machine learning models is still developing. This is especially true for text-based models and models used in vulnerability detection, given the limited work. This paper attempts to reduce this gap by expanding upon the discussion of explainability through identification of explainable dimensions. Such dimensions are observed in our scenario through experimentation regarding the impact of model performance given the removal of "cross-cutting" features among training and testing data. As such, our focus is primarily on vulnerability detection utilizing text-based models.

3 Study Setup

In this work, we utilize data from Napier et al. (2023) and Napier and Bhowmik (2022) to investigate the impact of removing "cross-cutting" features on the use of text-based machine learning models for vulnerability detection. We further evaluate this effort as it relates to explainability of such models as defined by explainable dimensions. We focus on the top 10 ranked projects and CWE vulnerability types, as defined by the previous studies, which are displayed in Table 1. These rankings are based on the number of pairings generated when extracting functions from "fixed" and "vulnerable" source code. The extraction of functions was motivated by the availability of vulnerable code snippets that generally lack a complete context. Pairing these functions was a strategy employed to avoid the class imbalance problem, which arises when instances of data outweigh each other and create potential bias. From these rankings, the projects are labeled as P1 through P10, where P1 corresponds to

Table 1 Top 10 projects and CWE vulnerability types based on function pairings (adapted from Napier and Bhowmik 2022)

Rank	Project	CWE
1	ffmpeg/ffmpeg	119
2	bonzini/qemu	20
3	xen-project/xen	264
4	torvalds/linux	189
5	chromium/chromium	399
6	ellson/graphviz	200
7	jktjkt/trojita	94
8	libvirt/libvirt	125
9	adaptivecomputing/torque	120
10	inspircd/inspird	17

ffmpeg/ffmpeg. Similarly, CWE vulnerability types are labeled as C1 through C10, with C1 representing *CWE 119*. We apply these function pairings, corresponding to the fixed" and "vulnerable" functions of projects and CWE vulnerability types, to a series of machine learning models. The following subsections will provide further details regarding the data, models, techniques, and metrics used throughout the remainder of this study.

3.1 Data

The data used throughout this study is publicly available in a GitHub repository.[3] The training and testing datasets have already been processed and used for the empirical study by Napier et al. (2023). For their methodology, they combined two datasets (Perl et al. 2015; Fan et al. 2020) based on vulnerability data and created equal function pairings in an effort to avoid unequal samples, otherwise known as the class imbalance problem. During this process, they generated and experimented with three data processing methods, *Full Context* (*FC*), *No Context* (*NC*), and *Lines Context* (LC). For their final experiments the training data consisted of *No Context* (*NC*) while the testing data consisted of *Full Context* (*FC*) . We will use the *NC* and *FC* datasets for the purpose of our study.

In generating these data processing methods, functions were extracted from instances of "fixed" and "vulnerable" source code corresponding to vulnerabilities of GitHub projects. The data was split into two instances based on the project as a whole or a mapped CWE vulnerability type based on mapped CVE identifiers. In an effort to avoid class imbalance, the functions were equally paired. The *FC* data represents function pairings of "full context" functions in their raw state minus comments. That is, functions as it appear within the context of a given source code file. Meanwhile, the *NC* data is generated by splitting function pairings into pieces based on whitespace where each piece is compared against the counterpart to determine if it exists in both places (fixed versus vulnerable). If a match is found, then that piece is discarded. Therefore, only the differences between a pairing are leftover with the features being considered unique for that pairing. Given the removal of context within the function, the label "No Context (*NC*)" was deemed appropriate. The *FC* data was used for testing to represent a "real world" scenario since generating *NC* data would not be possible unless both the "fixed" and "vulnerable" data were present.

[3] https://github.com/krn65/explainability_data

Table 2 Example of data processing methods handling function differences (adapted from Napier and Bhowmik 2022)

Method	Type (Length)	# Features	`Fixed` Function
FC	String (1)	10	`static int raw_truncate(BlockDriverState *bs, int64_t offset) { return bdrv_truncate(bs->file, offset); }`
NC	List (4)	4	`['bdrv_truncate(bs->file,', 'offset);']`
Method	Type (Length)	# Features	`Vulnerable` Function
FC	String (1)	17	`static int raw_truncate(BlockDriverState *bs, int64_t offset) { BDRVRawState *s = bs->opaque; if (s->type != FTYPE_FILE) return -ENOTSUP; if (ftruncate(s->fd, offset) < 0) return -errno; return 0; }`
NC	List (18)	12	`['BDRVRawState', '*s', '=', 'bs->opaque;', 'if', '(s->type', '!=', 'FTYPE_FILE)', '-ENOTSUP;', 'if', '(ftruncate(s->fd,', 'offset)', '<', '0)', 'return', '-errno;', 'return', '0;']`

Table 2 provides an example of the *NC* and *FC* data processing methods and their representations. In the top portion of the table, the first line represents how a "fixed" function is represented using the *FC* method. This line of code is represented a single string with 10 features that can be extracted. The counterpart "vulnerable" function is present in the lower portion of the table showing a similar representation, but more features given that there is a difference between the two instances. The *NC* representation shows a list of portions from the original *FC* function because it has since been modified as described previously. In this case, the "fixed" version has been reduced to 4 features whereas the "vulnerable" version has been reduced to 12. That is, similar portions of the function between the two instances has been reduced.

3.2 "Cross-Cutting" Features

The analysis by Napier and Bhowmik (2022) indicated a negative correlation in most observations between two data points: the overlap of features between the "fixed" and "vulnerable" categories and the average precision scores from the model's predictions. This overlap can cause confusion during the prediction process, as the model might associate a feature with a "fixed" label (1), only to encounter the same feature later with a "vulnerable" label (0). Consequently, the model learns that this feature can be both "fixed" and "vulnerable", leading it to resort to a "best guess" approach. This guesswork results in a 50% accuracy rate, as observed in the initial empirical results by Napier et al. (2023). Eliminating such overlapping features can reduce noise in the training and testing data. Table 3 provides a representation of *NC* data for two randomly chosen function pairings and their associated features. In this example, the feature sets in each row are unique to the specific columns: "fixed" and "vulnerable." However, when comparing the function pairings, an overlap of features becomes apparent. Specifically, the features *av_log* and *av_log_error* are labeled as both "fixed" and "vulnerable". Therefore, the data contains "cross-cutting" features.

Table 3 Example of function pairings with "cross-cutting" features (adopted from Napier and Bhowmik 2022)

	Fixed	Vulnerable
Features	['**av_log**', '**av_log_error**', 'avctx', 'fail', 'goto', 'has', 'length', 'more', 'of', 'one', 'plane', 'slice', 'symbol', 'than', 'yet', 'zero']	['0x80', 'continue', 'dest', 'for', 'send', 'sstart', 'step', 'stride', 'width']
Features	['av_assert0', 'av_malloc_array', 'avpix-elformat', 'break', 'choices', 'do', 'enum', 'for', 'memcpy', 'setup_hwaccel', 'sizeof', 'unsigned', 'while']	['an', '**av_log**', '**av_log_error**', 'av_mallocz', 'avhwaccel', 'codec_id', 'could', 'err', 'find', 'find_hwaccel', 'format', 'hwaccel', 'hwaccel_priv_data', 'if', 'init', 'int', 'internal', 'not', 'pixel', 'priv_data_size', 'return', 'the']

3.3 Models and Techniques

From the results of the empirical study by Napier et al. (2023), we elect to use their methods to present an accurate comparative view. This includes five of their seven machine learning models and the natural language processing (NLP) technique Term Frequency-Inverse Document Frequency (TF-IDF). We exclude the Text Convolutional Neural Network (TextCNN) from consideration due to similarities between it and the initial Convolutional Neural Network (CNN) model being used such as convolutional and pooling layers. Additionally, since we are using the TF-IDF technique, we exclude the BERT model due to a required integrated processing technique for training. These models, along with four of the five we proceed with, were only utilized during the preliminary experiments of the prior empirical study by Napier et al. (2023). Only the CNN model was utilized during the latter experiments based on statistical analysis due to to its better performance, coupled with the TF-IDF technique and the defined data processing technique, *NC*.

The remaining five models include: Random Forest (RF), Multi-layer Perceptron (MLP), Linear Support Vector Classification (LSVC), Bidirectional Long Short-Term Memory (BiLSTM), and Convolutional Neural Network (CNN). Using these models allows us to evaluate the previous results as well as provide an approach consisting of traditional and newer methods. Additionally, due to the prior empirical results, the TF-IDF technique was selected since it outperforms other techniques used with the models based on statistical analysis.

All the models are implemented from a base standpoint allowing an "out-of-the-box" approach. Additional processing can be made to the models as desired given their variety of parameters. The models are comprised from the following Python[4] libraries: scikit-learn,[5] Keras,[6] and TensorFlow.[7] The TF-IDF technique is also implemented via scikit-learn.

3.4 Evaluation Metrics

To generate and evaluate the results of our model, we employ a comparative analysis approach, following the evaluation method proposed by Napier et al. (2023). In their empirical study, they utilized the Average Precision (AP) scoring (APS) metric to assess their model

[4] https://python.org

[5] https://scikit-learn.org/stable/

[6] https://keras.io

[7] https://tensorflow.org

experiments. This metric combines precision and recall to measure the quality of a ranked list. AP is calculated by computing precision and recall values at different thresholds, summing up the precision values at each unique recall value, and dividing by the total number of positive instances. It provides a comprehensive evaluation by considering the trade-off between precision and recall and rewards models that consistently achieve high precision and recall. AP is a reliable measure of a model's ability to accurately rank items, making it valuable in tasks such as information retrieval and recommendation systems.[8] The Average Precision (AP) metric is defined as:

$$AP = \frac{1}{N} \sum_{k=1}^{N} \left(\frac{k}{\text{rank}_k} \right) \cdot \text{rel}(k) \tag{1}$$

- AP represents the Average Precision.
- N is the total number of items in the ranked list.
- k is the index representing the position of an item in the ranked list, ranging from 1 to N.
- rank_k denotes the position at which the k-th item is ranked in the list.
- $\text{rel}(k)$ is an indicator function that equals 1 if the item at rank k is relevant, and 0 otherwise.

While precision, recall, and the F1-score are commonly used in model performance evaluations (Scandariato et al. 2014; Ban et al. 2019; Liu et al. 2020), we opted for AP in our study. Although the F1-score is effective in scenarios where an equal balance between precision and recall is sought, it does not capture the nuances of ranking quality, which is crucial in our context, such as in vulnerability detection models (van Rijsbergen 1979; Murphy 2012). APS, with its emphasis on evaluating the overall ranking quality and rewarding models with consistent high precision and recall across varying thresholds, aligns better with our objective of assessing models in tasks that necessitate not only precision and recall but also the quality of ranking (Zhu 2004; Turpin and Scholer 2006). This rationale underpins our choice of AP over metrics like the F1-score, which, while valuable in certain contexts, falls short in addressing the specific requirements of our study.

3.5 Prior Empirical Results

To investigate the impact of removing "cross-cutting" features from the training and testing data, we refer to prior empirical results presented by Napier et al. (2023). Their study was primarily focused on the use of text-based machine learning models for vulnerability detection. They utilized code snippets to form function pairings consisting of "fixed" and "vulnerable" source code. By employing a variety of machine learning models, natural language processing techniques, and data processing methods, they conducted preliminary experiments to identify effective combinations. Based on statistical analysis, they determined that a combination of Convolutional Neural Network (CNN), Term-Frequency Inverse Document Frequency (TF-IDF), and their unique *No Context* (*NC*) data processing method achieved the best results. In addressing their research questions, they conducted experiments on within and cross-testing using the previously mentioned combination on a selection of data representing different projects and CWE vulnerability types. They found that condensed functions with fewer features might yield better prediction results when testing within projects rather than across. However, they concluded that text-based machine learning models are not particularly

[8] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.average_precision_score.html

effective in detecting vulnerabilities either within or across different projects and vulnerability types.

Table 4 provides an example of how Napier et al. (2023) presented their empirical results. Each column and row of the table represents a combination of ranked data based on the number of generated function pairings from the "fixed" and "vulnerable" source code. In this example, P1 through P10 corresponds to the top 10 projects as ranked in Table 1. The authors adopted a strategy of training and testing within and across different datasets for each project. For example, P1 (*ffmpeg/ffmpeg*) would be trained and tested on itself using the *NC* training data and *FC* testing data. Subsequently, P1 would be tested on P2 (*bonzini/qemu*) and so forth, until all items had been evaluated against the trained model for each combination. This process for repeated for the CWE vulnerability types as well, corresponding to C1 through C10. As a result, we can observe a table of results, highlighting model performance across different combinations, with the results of within testing emphasized in bold. Given these findings, we provide a comparison of the average precision scores using the same tabular format.

4 Experiments

In this section, we explore how removing certain features might improve the performance of machine learning models in identifying code vulnerabilities. We focus on "cross-cutting" features - those found in both "fixed" and "vulnerable" snippets of source code, based on function pairings (Section 4.1). In the first scenario (S1), we remove these "cross-cutting" features from a "*No Context* (*NC*)" training set, creating two unique sets of features for both "fixed" and "vulnerable" instances (Section 4.2). Then, we extend this process to a "*Full Context* (*FC*)" testing set in the second scenario (S2). Here, we eliminate all overlapping features between the unique *NC* training set and *FC* testing set (Section 4.3). Throughout these scenarios, we compare our results with previous studies using a Convolutional Neural Network (CNN). In the following sections, we discuss our findings and their implications for enhancing the ability of text-based machine learning models to detect code vulnerabilities.

Table 4 Average precision scores within and across top 10 projects using NC data with TF-IDF technique and CNN model (adopted from Napier et al. 2023)

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
P1	**53.02**%	50.21%	50.37%	49.73%	51.07%	49.77%	49.72%	50.83%	51.33%	49.79%
P2	50.46%	**57.93**%	50.26%	50.58%	50.17%	49.36%	48.52%	51.28%	51.87%	49.21%
P3	50.24%	49.92%	**54.65**%	50.39%	50.44%	50.52%	49.62%	51.04%	51.89%	52.32%
P4	50.18%	50.19%	50.16%	**52.07**%	50.37%	49.48%	49.88%	50.01%	50.20%	48.28%
P5	50.25%	50.01%	50.38%	50.76%	**62.06**%	50.91%	53.09%	50.06%	51.16%	48.16%
P6	50.16%	50.38%	50.13%	49.71%	50.36%	**63.36**%	49.74%	50.10%	49.23%	48.84%
P7	50.28%	50.22%	50.26%	50.06%	50.15%	50.04%	**62.59**%	50.57%	51.08%	48.54%
P8	50.31%	50.25%	50.31%	50.38%	50.23%	50.43%	51.78%	**53.53**%	50.56%	48.92%
P9	50.13%	49.91%	50.09%	50.45%	50.48%	51.21%	49.80%	50.22%	**56.25**%	50.24%
P10	50.15%	49.96%	50.26%	49.85%	50.30%	49.62%	51.49%	50.11%	50.91%	**56.28**%

4.1 "Cross-Cutting" Feature Removal

Our removal process operates on a selection from the top 10 items in Table 1, generating a unique set of features for "fixed" and "vulnerable" data types. These features are derived from two distinct data processing methods: "*No Context* (*NC*)" and "*Full Context* (*FC*)", which are employed for training and testing models respectively. During this process, we identify overlaps in features between different data sets. If such overlaps exist, we excise them, thereby forming a unique set corresponding to the respective data types. As a result, we attain a set of features that are exclusive to the "fixed" and "vulnerable" instances within the given data processing method. While modifying such data, we do not claim to provide any novel techniques beyond standard data processing techniques such as removal and extraction. This approach can be replicated or presented in an alternative manner. We do acknowledge our advantage in overcoming potential data limitations by utilizing an existing dataset with an equal split of data presented in the form of function pairings from extracted source code.

Our primary aim is to identify and eliminate "cross-cutting" features - those present in both "fixed" and "vulnerable" data types - within each processing method. However, we approach this in distinct scenarios. The first scenario (S1) focuses on the removal of "cross-cutting" features strictly from the *NC* training data. For instance, when *ffmpeg/ffmpeg* (P1) is the training data, the procedure discards all shared features within the *NC* context from both "fixed" and "vulnerable" data types, ensuring a clear distinction between features derived from each data processing method and thereby preventing any overlap. We then examine whether changes to the training data alone can impact model predictions when tested across diverse projects and CWE vulnerability types. This process corresponds to our first scenario (S1), as depicted in Section 4.2.

To further scrutinize the removal of "cross-cutting" features, we build upon the unique sets of features generated in the first scenario (S1). This involves leveraging the "*No Context* (*NC*)" training data that has been cleared of "cross-cutting" features and comparing it to the "*Full Context* (*FC*)" testing data. We conduct this comparison and extraction to identify any features that are common, or "cross-cutting", between these two data sets. Upon detection, these overlapping features are eliminated, ensuring that only unique elements persist within both the training and testing datasets. This process not only safeguards the distinctiveness of the features within each dataset but also optimizes them for their respective roles in training and testing. We then evaluate how model predictions are affected when both the unique *NC* training data from S1 and the *FC* testing data undergo this processing. This extended process corresponds to our second scenario (S2), as illustrated in Section 4.3.

In the subsequent subsections, we present a comparison table juxtaposing previous empirical results with data from the two scenarios, using the Convolutional Neural Network (CNN) model for projects and CWE vulnerability types. A comparison of the previous and current data is furnished only for the CNN model, due to its usage in the final experiments of the empirical study by Napier et al. (2023). However, results for the remaining models are also compiled for each scenario for additional scrutiny. To minimize the use of tables within the main text related to these auxiliary models, the data pertaining to these models are provided in the Appendix.

4.2 S1: Removal from Training

As described for our first scenario (S1), we remove any "cross-cutting" features with a common occurrence between the "fixed" and "vulnerable" *NC* training data for projects and

CWE vulnerability types. This process creates a new set of unique features for both the "fixed" and "vulnerable" data. In this case, we perform an evaluation to determine if processing the training data will lead to better prediction performance from our models.

For the initial aspect of S1, we present the previous empirical results from Napier et al. (2023) for comparison with the processed training *NC* data for the projects and CWE vulnerability types. These results stem from the usage of the CNN model, coupled with the TF-IDF technique and the *NC* training data. The testing data is the original *FC* counterpart as previously described. Table 5 provides a comparative view of the original empirical results on the top half, with the results from S1 of removing "cross-cutting" features from only the "fixed" and "vulnerable" *NC* training data of the top 10 projects.

From the results in Table 5 we observe similar trends among the results from the two experiments, the previous empirical and S1 data. Overall, the average of the scores when training and testing within a project stay around the 55% range. Comparing the average within scores between the two experiments, only a 0.5% difference is present, with previous empirical results offering a slightly higher score. However, a 2% difference in seen in the maximum value (P5, *chromium/chromium*) and 0.5% in the minimum value (P1, *ffmpeg/ffmpeg*) for the S1 "cross-cutting" experiment. When testing within a particular project (i.e., P1), the difference between S1 and empirical experiments reaches a maximum of 6% increase in P4 (*torvalds/linux*). However, a loss of 8% occurs in P7 (*jktjkt/trojita*). Although there are positive increases, the within scoring provides equal occurrences of increased and decreased values across the top 10 projects. The cross-testing scores mostly show a decrease

Table 5 Comparison of average precision scores: Convolutional Neural Network (CNN) model in empirical study and removal of "cross-cutting" features in training data for projects (adapted from Napier et al. 2023)

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
P1	**53.02**%	50.21%	50.37%	49.73%	51.07%	49.77%	49.72%	50.83%	51.33%	49.79%
P2	50.46%	**57.93**%	50.26%	50.58%	50.17%	49.36%	48.52%	51.28%	51.87%	49.21%
P3	50.24%	49.92%	**54.65**%	50.39%	50.44%	50.52%	49.62%	51.04%	51.89%	52.32%
P4	50.18%	50.19%	50.16%	**52.07**%	50.37%	49.48%	49.88%	50.01%	50.20%	48.28%
P5	50.25%	50.01%	50.38%	50.76%	**62.06**%	50.91%	53.09%	50.06%	51.16%	48.16%
P6	50.16%	50.38%	50.13%	49.71%	50.36%	**63.36**%	49.74%	50.10%	49.23%	48.84%
P7	50.28%	50.22%	50.26%	50.06%	50.15%	50.04%	**62.59**%	50.57%	51.08%	48.54%
P8	50.31%	50.25%	50.31%	50.38%	50.23%	50.43%	51.78%	**53.53**%	50.56%	48.92%
P9	50.13%	49.91%	50.09%	50.45%	50.48%	51.21%	49.80%	50.22%	**56.25**%	50.24%
P10	50.15%	49.96%	50.26%	49.85%	50.30%	49.62%	51.49%	50.11%	50.91%	**56.28**%
P1	**52.67**%	50.13%	50.29%	49.65%	49.67%	50.15%	49.68%	50.71%	48.78%	50.26%
P2	50.02%	**55.39**%	50.07%	50.16%	49.65%	51.23%	50.10%	50.38%	50.65%	48.58%
P3	50.08%	49.68%	**54.94**%	50.54%	50.21%	50.09%	50.83%	50.01%	50.28%	48.89%
P4	50.04%	50.11%	50.21%	**58.28**%	50.46%	49.06%	49.45%	50.12%	50.49%	50.04%
P5	50.13%	50.13%	49.89%	50.36%	**65.36**%	49.87%	50.97%	50.03%	50.64%	50.65%
P6	49.95%	49.90%	49.81%	50.63%	50.24%	**58.30**%	50.98%	49.67%	50.07%	49.41%
P7	50.02%	49.73%	49.83%	49.74%	50.07%	50.18%	**54.35**%	49.47%	49.34%	48.73%
P8	50.17%	49.96%	50.20%	50.69%	50.55%	50.13%	50.36%	**53.94**%	50.10%	49.49%
P9	50.07%	49.83%	49.95%	49.58%	50.23%	51.39%	50.91%	50.77%	**55.51**%	49.72%
P10	50.09%	50.39%	50.13%	50.11%	50.26%	50.36%	49.73%	50.41%	51.82%	**58.18**%

when averaging, however, this value is less than 0.5% in all but one case. In the cases of positive increase, we observe a maximum of 0.2%.

From these results, we noted minimal or no improvement for the CNN model when removing "cross-cutting" features from the *NC* training data. However, in S1, we only examined a single model for comparison. The results for the four remaining machine learning models begin with Table 25 in the Appendix. Tables 25, 26, 27, and 28 display the outcomes for the remaining four machine learning models after the *NC* training data processing was applied. Since these models were not advanced beyond the preliminary stage in the previous empirical study by Napier et al. (2023), they are not applicable for direct comparison. Nevertheless, to analyze the effect of removing "cross-cutting" features and to identify explainable dimensions for using these models, we have generated and provided these results.

We observed trends similar to the CNN model with the remaining models (Tables 25, 26, 27, and 28). For instance, the within scores are 50% or higher in all cases. However, when testing across models, the scores typically hover around 50%, with few instances of slightly higher results. This suggests that the models are likely providing a "best guess" due to residual confusion. In S1, we conjecture that "cross-cutting" features may still be present and could be influencing the results, as suggested by the correlation analysis by Napier and Bhowmik (2022).

Similar to the projects, Table 6 provides a comparative perspective for CWE vulnerability types based on the previous empirical results by Napier et al. (2023) and the results of S1

Table 6 Comparison of average precision scores: Convolutional Neural Network (CNN) model in empirical study and removal of "cross-cutting" features in training data for CWE vulnerability types (adapted from Napier et al. 2023)

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
C1	**55.54**%	51.10%	52.13%	50.95%	51.60%	51.77%	52.48%	50.16%	51.46%	50.02%
C2	50.97%	**58.00**%	51.38%	50.31%	50.46%	50.59%	53.05%	49.84%	51.02%	51.15%
C3	50.92%	50.92%	**58.45**%	50.50%	51.25%	50.87%	50.06%	49.73%	50.51%	50.82%
C4	50.55%	50.38%	50.45%	**55.19**%	50.99%	50.45%	50.19%	50.56%	50.39%	50.18%
C5	50.57%	50.59%	50.86%	50.99%	**57.42**%	50.56%	50.16%	50.20%	50.31%	50.63%
C6	50.59%	50.38%	50.40%	50.25%	50.54%	**58.61**%	51.34%	50.40%	50.93%	50.04%
C7	50.59%	50.74%	50.58%	50.25%	50.34%	50.99%	**61.61**%	49.90%	49.99%	50.07%
C8	50.30%	50.17%	50.37%	50.50%	50.32%	50.25%	50.15%	**56.02**%	50.71%	50.33%
C9	50.30%	50.41%	50.50%	50.77%	50.66%	50.15%	49.83%	50.28%	**58.23**%	50.55%
C10	50.17%	50.12%	50.45%	50.14%	50.14%	50.08%	49.90%	50.38%	49.69%	**58.59**%
C1	**56.02**%	50.74%	50.54%	50.43%	50.59%	50.81%	51.39%	50.10%	50.13%	50.09%
C2	50.96%	**59.32**%	50.24%	50.02%	49.98%	50.82%	51.64%	49.66%	50.86%	51.03%
C3	50.74%	50.57%	**58.70**%	50.63%	50.41%	50.76%	50.32%	50.02%	50.31%	50.36%
C4	50.38%	50.35%	50.21%	**57.79**%	50.19%	50.52%	49.85%	50.56%	50.04%	49.40%
C5	50.77%	50.30%	50.64%	50.46%	**59.21**%	50.36%	50.93%	51.12%	50.88%	49.92%
C6	50.35%	50.62%	50.39%	50.19%	50.26%	**63.11**%	50.67%	50.12%	49.85%	49.37%
C7	50.44%	50.53%	50.42%	50.44%	50.16%	50.02%	**61.85**%	50.19%	50.19%	50.27%
C8	50.01%	50.13%	50.13%	50.61%	50.60%	50.04%	49.65%	**55.85**%	49.94%	49.85%
C9	50.50%	50.55%	50.15%	50.65%	50.57%	49.82%	51.52%	50.60%	**60.16**%	49.94%
C10	50.10%	50.47%	50.32%	50.22%	50.03%	49.80%	50.03%	47.72%	49.71%	**58.40**%

after eliminating "cross-cutting" features within the *NC* training data using the CNN model. When training and testing within a specific CWE vulnerability type, we noted an increase in several values, albeit not significant. The difference between the two experiment scores averages a 1.3% increase with a maximum increase of 4.5% in C6 (*CWE 200*) and the largest decrease in C10 (*CWE 17*) by 0.2%. Similar to the cross-testing scores shown for the projects, the CWE vulnerability types exhibit minimal or no difference. When averaging the scores across each combination for the previous empirical results and S1 results, most instances show a slight decrease.

Tables 29, 30, 31, and 32 present the results of the remaining four machine learning models after applying the processed *NC* training data. Similar to our analysis with the projects, we examine the performance of these models with CWE vulnerability types. Once again, we note "best guess" attempts from each of the models, with average precision scores hovering around 50% both when testing within and across other CWE vulnerability types. A few instances of 60% or higher are present in Table 32 for the BiLSTM model, but these are not observed in any other model apart from the CNN, as shown in the comparison from Table 6.

For S1, we eliminated "cross-cutting" features from the *No Context* (*NC*) training data for both the projects and CWE vulnerability types. Upon applying this processed data towards training various text-based machine learning models, we observed no impact on the prediction compared to the previous results. We hypothesize that model confusion still plays a role in both situations and that further experimentation with the removal of "cross-cutting" features is required. From this, we introduce our second scenario (S2), in which we employ the unique *NC* training data we created and apply it towards further cross-testing experiments. For S2, we also remove "cross-cutting" features from the *Full Context* (*FC*) testing data, resulting in a unique set for both training and testing data. In this case, we examine how the models react to the complete removal of such "cross-cutting" features between the *NC* training data and the *FC* testing data.

4.3 S2: Removal from Training and Testing

As described for our second scenario (S2), we utilize the previous results from the first scenario (S1) and extend that to the current testing data known as Full Context (*FC*) . From this, we remove any "cross-cutting" features with a common occurrence between the training (*NC*) and testing (*FC*) data for projects and CWE vulnerability types. As a result, we generate a new set of unique features for both sets of data. In this case, we perform an evaluation to determine if processing both training and testing data will lead to better prediction performance from our models.

For S2, we present the average precision score results of the CNN model for the projects in Table 7, akin to Table 5 for S1. We draw a comparison between the empirical results presented by Napier et al. (2023) and the S2 data, in which both training and testing datasets have been processed to eliminate "cross-cutting" features. Upon a comprehensive review, we observe similar trends when testing within projects, but significant increases in average precision scores when testing across different projects. The highest average precision score noted is 80.43%, observed between P3 (*xen-project/xen*) and P9 (*adaptivecomputing/torque*). Across the columns for P6 (ellson/graphviz), P7 (jktjkt/trojita), and P9, we register scores that are markedly higher, exhibiting an increase of over 20% compared to the corresponding columns in the earlier empirical results (top portion of Table 7).

Table 7 Comparison of average precision scores: Convolutional Neural Network (CNN) model in empirical study and removal of "cross-cutting" features in training and testing data for projects (adapted from Napier et al. 2023)

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
P1	**53.02**%	50.21%	50.37%	49.73%	51.07%	49.77%	49.72%	50.83%	51.33%	49.79%
P2	50.46%	**57.93**%	50.26%	50.58%	50.17%	49.36%	48.52%	51.28%	51.87%	49.21%
P3	50.24%	49.92%	**54.65**%	50.39%	50.44%	50.52%	49.62%	51.04%	51.89%	52.32%
P4	50.18%	50.19%	50.16%	**52.07**%	50.37%	49.48%	49.88%	50.01%	50.20%	48.28%
P5	50.25%	50.01%	50.38%	50.76%	**62.06**%	50.91%	53.09%	50.06%	51.16%	48.16%
P6	50.16%	50.38%	50.13%	49.71%	50.36%	**63.36**%	49.74%	50.10%	49.23%	48.84%
P7	50.28%	50.22%	50.26%	50.06%	50.15%	50.04%	**62.59**%	50.57%	51.08%	48.54%
P8	50.31%	50.25%	50.31%	50.38%	50.23%	50.43%	51.78%	**53.53**%	50.56%	48.92%
P9	50.13%	49.91%	50.09%	50.45%	50.48%	51.21%	49.80%	50.22%	**56.25**%	50.24%
P10	50.15%	49.96%	50.26%	49.85%	50.30%	49.62%	51.49%	50.11%	50.91%	**56.28**%
P1	**52.67**%	62.82%	62.02%	57.25%	62.23%	74.84%	74.62%	66.20%	77.22%	37.97%
P2	58.42%	**55.39**%	61.70%	56.94%	62.36%	74.84%	74.22%	67.62%	77.78%	36.61%
P3	58.53%	62.94%	**54.94**%	57.10%	62.66%	74.98%	74.79%	66.63%	80.43%	35.85%
P4	58.58%	62.90%	62.17%	**58.28**%	62.61%	74.53%	74.62%	65.06%	77.51%	37.11%
P5	58.56%	62.98%	62.21%	57.22%	**65.36**%	75.39%	74.62%	66.04%	77.88%	37.34%
P6	58.66%	62.99%	62.24%	57.22%	62.77%	**58.30**%	75.57%	66.04%	78.65%	38.12%
P7	58.58%	62.98%	62.19%	57.25%	62.75%	74.87%	**54.35**%	66.36%	78.65%	35.48%
P8	58.58%	62.95%	62.21%	57.26%	62.77%	74.61%	75.00%	**53.94**%	78.31%	37.89%
P9	58.58%	62.95%	62.19%	57.29%	62.76%	74.53%	74.62%	65.71%	**55.51**%	38.91%
P10	58.57%	62.96%	62.19%	57.19%	62.77%	74.77%	75.07%	66.36%	78.31%	**58.18**%

However, P10, which had the fewest function pairings from the top 10 rankings, consequently had fewer features. When processing the features between the training and testing data, a substantial decrease in applicable features and labeling is evident. Consequently, the scores drop significantly. Although, given a more robust data set, we hypothesize that these scores would likely improve, as observed with the other projects. For the CNN model, scoring from the processed data has increased the efficacy of vulnerability detection. Supplemental data for the remaining S2 models using project data is available in the Appendix Tables 33, 34, 35, and 36.

Figure 1 depicts the average cross-testing score for the CNN model for each project, incorporating results from the previous empirical study as well as our S1 and S2 experiments. In this analysis, we examine how each model fares when tested against data representing a variety of other projects. For instance, if the current training project is P1, we calculate the average cross-testing scores when P1 is tested against other projects. Thus, there are nine data points for each possible pairing combination (e.g., P1-P2, P1-P3, etc.). We note the resemblances between the previous empirical findings and our first scenario (S1) results, both approximating a 50% average precision score, which we interpret as a baseline "best guess" attempt by the models. However, the introduction of our second scenario (S2) results in a significant uplift in the average precision scores across all models. Additional plots for the remaining models, presenting S1 and S2 data, can be found in the Appendix (Figs. 4, 5, 6, and 7). These findings suggest that eliminating "cross-cutting" features from the project's

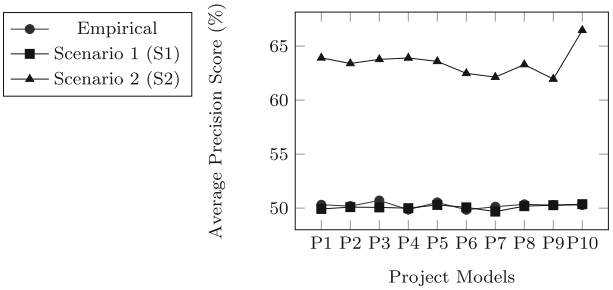


Fig. 1 Average of cross-testing scores for trained project CNN models

training and testing data might enhance the performance of text-based machine learning models in vulnerability detection.

Upon examining the comparative results in Table 8 for CWE vulnerability types, we note a substantial increase in the cross-testing results for almost all combinations. In contrast, the within-testing scores demonstrate a modest increase in most combinations. The most notable average precision scores are associated with C8 (*CWE 125*), surpassing 66% in multiple combinations. The only exception appears to be data for C7 (*CWE 94*), which either remains consistent or slightly declines. This can be assumed to mirror previously observed trends,

Table 8 Comparison of average precision scores: Convolutional Neural Network (CNN) model in empirical study and removal of "cross-cutting" features in training and testing data for CWE vulnerability types (adapted from Napier et al. 2023)

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
C1	**55.54**%	51.10%	52.13%	50.95%	51.60%	51.77%	52.48%	50.16%	51.46%	50.02%
C2	50.97%	**58.00**%	51.38%	50.31%	50.46%	50.59%	53.05%	49.84%	51.02%	51.15%
C3	50.92%	50.92%	**58.45**%	50.50%	51.25%	50.87%	50.06%	49.73%	50.51%	50.82%
C4	50.55%	50.38%	50.45%	**55.19**%	50.99%	50.45%	50.19%	50.56%	50.39%	50.18%
C5	50.57%	50.59%	50.86%	50.99%	**57.42**%	50.56%	50.16%	50.20%	50.31%	50.63%
C6	50.59%	50.38%	50.40%	50.25%	50.54%	**58.61**%	51.34%	50.40%	50.93%	50.04%
C7	50.59%	50.74%	50.58%	50.25%	50.34%	50.99%	**61.61**%	49.90%	49.99%	50.07%
C8	50.30%	50.17%	50.37%	50.50%	50.32%	50.25%	50.15%	**56.02**%	50.71%	50.33%
C9	50.30%	50.41%	50.50%	50.77%	50.66%	50.15%	49.83%	50.28%	**58.23**%	50.55%
C10	50.17%	50.12%	50.45%	50.14%	50.14%	50.08%	49.90%	50.38%	49.69%	**58.59**%
C1	**56.02**%	61.38%	62.15%	59.78%	61.80%	59.49%	49.97%	62.71%	56.67%	59.73%
C2	60.56%	**59.32**%	61.36%	60.42%	61.87%	60.57%	50.63%	64.55%	56.30%	59.08%
C3	60.78%	62.28%	**58.70**%	61.67%	63.76%	59.31%	50.65%	66.10%	58.09%	59.61%
C4	60.80%	62.74%	63.11%	**57.79**%	64.34%	60.43%	50.38%	64.94%	58.42%	60.76%
C5	60.92%	62.77%	63.18%	62.24%	**59.21**%	60.80%	51.26%	65.31%	57.84%	60.10%
C6	61.12%	62.93%	62.01%	61.27%	63.33%	**63.11**%	51.82%	66.05%	58.80%	61.22%
C7	61.40%	62.67%	63.09%	61.70%	63.98%	62.00%	**61.85**%	66.10%	57.52%	60.87%
C8	61.05%	63.00%	62.86%	61.74%	63.65%	61.38%	50.98%	**55.85**%	58.90%	61.07%
C9	60.98%	62.70%	62.28%	61.38%	63.19%	60.66%	49.68%	66.65%	**60.16**%	60.74%
C10	61.17%	63.09%	62.76%	61.63%	63.38%	61.30%	51.09%	66.08%	59.15%	**58.40**%

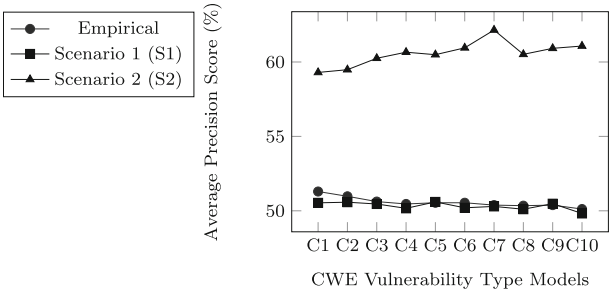


Fig. 2 Average of cross-testing scores for trained CWE vulnerability types CNN models

as seen with Column P10 in Table 7, where insufficient feature data was present due to the "cross-cutting" removal process. For training and testing within a specific CWE, most scores generally increase by a few percentage points. Further data for the remaining S2 models using project data can be found in the Appendix Tables 37, 38, 39, and 40.

As depicted in Fig. 2, the average cross-testing scores for the CNN model are provided for each CWE vulnerability type. The patterns observed here parallel those from our prior experiment on models trained and tested on projects (as seen in Fig. 1). Both the past empirical findings and the results from our first scenario (S1) demonstrate a mean precision score close to 50%. On the other hand, the data from our second scenario (S2) reveals a substantial improvement. Supplementary figures, plotting S1 and S2 data for the remaining models, are included in the Appendix (Figs. 8, 9, 10, and 11). These findings propose that eradicating "cross-cutting" features from the CWE vulnerability training and testing datasets might boost the effectiveness of text-based machine learning models in identifying vulnerabilities.

5 Answering the Research Question

5.1 What effect does the removal of "cross-cutting" features have on the performance of text-based machine learning models for vulnerability detection?

Based on the analysis provided by Napier and Bhowmik (2022), we processed data used in Napier et al. (2023) by removing "cross-cutting" features. We present two experiment scenarios in which the such features are removed from projects and CWE vulnerability types: 1) remove from *No Context* (*NC*) training data; 2) remove from *No Context* (*NC*) training data and *Full Context* (*FC*) testing data. Our objective is to compare independent samples of data from these scenarios and previous empirical study to answer our central research question. As demonstrated by the correlation analysis conducted by Napier and Bhowmik (2022), negative correlations were evident in three out of the four scatter plots relating to the projects and CWE vulnerability types. This suggests that the data does not follow a normal distribution. For more details, please see the scatter plots included in Appendix A, specifically Fig. 3. As a result, traditional parametric tests such as Analysis of Variance (ANOVA), may not be appropriate. Therefore, to find our answers with statistical evidence, we elect to use Kruskal-Wallis (Kruskal and Wallis 1952). Kruskal-Wallis is a non-parametric statistical ranked-based test and previously used by Napier et al. (2023). Such tests will be used throughout the remainder of this study. From these tests we present findings on

Table 9 Kruskal-Wallis - ranks of CNN model for projects

Model	N	Mean Rank
CNN (Empirical)	10	12.40
CNN (Scenario 1)	10	8.60
CNN (Scenario 2)	10	25.50
Total	30	

the statistical difference related to cross-testing among the scenarios and models based on hypotheses. Such results are applied towards explainable dimensions related to explainability of supervised machine learning models. We examine a variety of explainable dimensions for this work, but focus on the most relevant, agnostic and data dependency. In this effort, we address our research question to determine if removal of such features provides performance improvement towards our text-based models for vulnerability detection. To that end, we formulate the following null and alternative hypotheses:

Data Dependent:
H_{10}: There is no significant difference between the machine learning models when removing "cross-cutting" features (not data dependent).
H_{11}: There is a significant difference between the machine learning models when removing "cross-cutting" features (data dependent).
Agnostic:
H_{20}: There is no significant difference between increase in the performance of the models when removing "cross-cutting" features (model agnostic).
H_{21}: There is a significant difference between increase in the performance of the models when removing "cross-cutting" features (not model agnostic).

5.2 Investigating the Data Dependent Aspect

We execute Kruskal-Wallis tests as it relates to our formulated hypotheses regarding the explainable dimension of data dependency. Data from the scenarios in which "cross-cutting" features were removed from the top 10 projects and top 10 CWE vulnerability types, as shown in Table 1, will be used. The tests examine the results of within and cross-testing between the scenarios for the CNN model. We can determine if the models are data dependent based on the usage of processed training and testing data.

Our first test focuses on the results of the CNN model which was used for comparison between the scenario experiments and the previous empirical study by Napier et al. (2023) for top 10 projects data, as shown in Tables 5 and 7. The independent variables are the CNN models from the empirical study and our two scenarios. The dependent variables are their related average precision scores. The results for this test are presented in Tables 9 and 10. From the results related to the CNN model in the different scenarios, removing "cross-cutting"

Table 10 Kruskal-Wallis - test statistics of CNN model for projects

	APS
Kruskal-Wallis H	20.286
df	2
Asymp. Sig.	< 0.001

Table 11 Kruskal-Wallis - ranks of CNN model for CWE vulnerability types

Model	N	Mean Rank
CNN (Empirical)	10	12.40
CNN (Scenario 1)	10	8.60
CNN (Scenario 2)	10	25.50
Total	30	

features from the training and testing data for projects offered the higher mean rank. Table 10 states there is a statistically significant difference between the cross-testing scores between the CNN model scenarios for projects at $\alpha = 0.05$ level of significance with $p = < 0.001$.

The second test is similar in that is focuses on the compared results between the CNN model in the previous empirical study and our scenario experiments for the top 10 CWE vulnerability types, as shown in Tables 6 and 8. The independent variables are the CNN models from the empirical study and our two scenarios. The dependent variables are their related average precision scores. Using the CNN models from the three scenarios, we can determine if there is a statistically significant difference related the observed average precision scores (APS) for prediction. As with the projects, the highest mean rank among the scenarios for the CNN model is second scenario, as shown in Table 11. Table 12 states there is a statistically significant difference between the cross-testing scores between the CNN model scenarios for CWE vulnerability types at $\alpha = 0.05$ level of significance with $p = < 0.001$.

We define a new Kruskal-Wallis test for the remaining models The independent variables are the remaining models and their associated cross-testing average precision scores. The dependent variables are the two scenarios in which "cross-cutting" features are removed. Table 13 shows the APS in the second scenario where "cross-cutting" features were removed from both training and testing data offers higher mean rank. Table 14 provides further clarification in stating there is a statistically significant difference between the cross-testing scores of the remaining models for projects at $\alpha = 0.05$ level of significance with $p = < 0.001$. Similarly with the CWE vulnerability types, we observe a higher mean rank for scenario two as shown in Tables 15. Table 16 shows that there is a statistically significant difference in the cross-testing scores at $\alpha = 0.05$ level of significance with $p = < 0.001$.

Based on the results for all the models related to the cross-testing scores from the scenarios, we can address our hypotheses for the explainable dimension of data dependency. We reject our null hypothesis (H_{10}) and accept our alternative hypothesis (H_{11}). We conclude that there is a statistically significant difference between the text-based machine learning models when "cross-cutting" features are removed. Therefore, such models can be deemed data dependent in our setting.

5.3 Investigating the Agnostic Aspect

From the results of our scenario experiments, we suspect the models of being agnostic. That is, all models perform in a similar manner and therefore are not limited to specifics. Based

Table 12 Kruskal-Wallis - test statistics of CNN model for CWE vulnerability types

	APS
Kruskal-Wallis H	20.295
df	2
Asymp. Sig.	< 0.001

Table 13 Kruskal-Wallis - ranks of APS for projects

	Scenario	N	Mean Rank
APS	S1	40	20.50
	S2	40	60.50
	Total	80	

Table 14 Kruskal-Wallis - test statistics of APS for projects

	APS
Kruskal-Wallis H	59.288
df	1
Asymp. Sig.	< 0.001

Table 15 Kruskal-Wallis - ranks of APS for CWE vulnerability types

	Scenario	N	Mean Rank
APS	S1	40	20.50
	S2	40	60.50
	Total	80	

Table 16 Kruskal-Wallis - test statistics of APS for CWE vulnerability types

	APS
Kruskal-Wallis H	59.286
df	1
Asymp. Sig.	< 0.001

Table 17 Kruskal-Wallis - ranks of models for projects

	Scenario	N	Mean Rank
Model	S1	40	40.50
	S2	40	40.40
	Total	80	

on the projects and CWE vulnerability types represented in Section 4 and the Appendix, similar scores are observed throughout the models when we conducted cross-testing. For the remaining project models, Scenario 2 (S2) outperforms Scenario 1 (S1) in all cases as shown in Figs. 4, 5, 6, and 7. As with the projects, the remaining CWE vulnerability type models show S2 achieving greater AP scores as shown in Figs. 8, 9, 10, and 11. Overall, the remaining models between our two scenarios shows them performing within the same ranges regardless of the model used. To further confirm the conclusion on the model agnostic aspect, we will include another Kruskal-Wallis test where the model is the independent variable and the difference in AP scores from S1 to S2 is the dependent variable.

Table 17 shows almost the same mean ranking among the models between the two scenarios in which "cross-cutting" features were removed from the project data. Table 18 shows there is no significant difference between increase in the performance of the models when removing "cross-cutting" features from projects data at $\alpha = 0.05$ level of significance with $p = 1.000$.

Tables 19 and 20 provide results based on the data from CWE vulnerability types. Table 19 also shows almost the same mean ranking among the models between the two scenarios. Table 20 shows there is no significant difference between increase in the performance of the models when removing "cross-cutting" features from CWE vulnerability type data at $\alpha = 0.05$ level of significance with $p = 1.000$.

Based on the results of the models from the two scenarios, we fail to reject our null hypothesis (H_{20}) for the performance of machine learning models. Therefore, this shows the models can be generalized in that they offer the same type of performance regardless of their structure even with the processed data. To further evaluate this result, we analyze the resulting difference between the two scenarios and apply additional Kruskal-Wallis tests for both projects and CWE vulnerability type data.

The results are shown in Tables 21 and 22. Based on the mean rankings of the remaining models, they all fall within a similar range. Table 22 states that there is no significant difference between the models at $\alpha = 0.05$ level of significance with $p = 0.974$. For the CWE vulnerability type data, Table 23 shows similar ranking ranges while Table 24 states that there is no significant difference between the models at $\alpha = 0.05$ level of significance with $p = 0.678$.

Based on the results for both projects and CWE vulnerability types, we fail to reject our null hypothesis (H_{20}). We conclude that there is no statistically significant difference

Table 18 Kruskal-Wallis - test statistics of models for projects

	Model
Kruskal-Wallis H	0.000
df	1
Asymp. Sig.	1.000

Table 19 Kruskal-Wallis - ranks of models for CWE vulnerability types

	Scenario	N	Mean Rank
Model	S1	40	40.50
	S2	40	40.40
	Total	80	

Table 20 Kruskal-Wallis - test statistics of models for CWE vulnerability types

	Model
Kruskal-Wallis H	0.000
df	1
Asymp. Sig.	1.000

Table 21 Kruskal-Wallis - ranks of models for projects

	Model	N	Mean Rank
Difference	Random Forest (RF)	10	21.60
	Linear Support Vector (LSVC)	10	20.80
	Multi-layer Perceptron	10	20.40
	Bidirectional Long Short-Term Memory (BiLSTM)	10	19.20
	Total	40	

Table 22 Kruskal-Wallis - test statistics of models for projects

	Difference
Kruskal-Wallis H	0.220
df	3
Asymp. Sig.	0.974

Table 23 Kruskal-Wallis - ranks of models for CWE vulnerability types

	Model	N	Mean Rank
Difference	Random Forest (RF)	10	22.30
	Linear Support Vector (LSVC)	10	21.80
	Multi-layer Perceptron	10	21.30
	Bidirectional Long Short-Term Memory (BiLSTM)	10	16.60
	Total	40	

Table 24 Kruskal-Wallis - test statistics of models for CWE vulnerability types

	Difference
Kruskal-Wallis H	1.520
df	3
Asymp. Sig.	0.678

between the increase in the performance of the models when removing "cross-cutting" features. We provide further discussion related to the results and explainability as it pertains to the dimensions in the next section.

6 Discussion

From the experiment results presented in Section 4, we make several conclusions and assumptions regarding the usage of text-based machine learning models for vulnerability detection. As observed in the previous correlation analysis by Napier and Bhowmik (2022), an interaction between the overlap of features, deemed "cross-cutting", and the AP scores being observed provided a negative trend. That is, the higher the overlap, the lower the model prediction scores. When attempting to remove such "cross-cutting" features from the data, we generate two scenarios. The first (S1) being removal of "cross-cutting" features from the "fixed" and "vulnerable" *No Context* (*NC*) training data and second (S2) being the removal of "cross-cutting" features from the *NC* training data as well as the *Full Context* (*FC*) testing data. From the initial results of our experiments, we show that removal of such "cross-cutting" features in both projects and CWE vulnerability types does provide improvement when cross-testing as it pertains to the second scenario (S2). In turn, we can state that removal of "cross-cutting" features from the training data does not offer enough evidence to that it provides an effective approach.

To further explore statistical evidence, we applied Kruskal-Wallis. These tests were conducted to test certain hypotheses formulated about the scenarios for both projects and CWE vulnerability types. Our hypotheses were focused on dimensions of explainability as it pertains to supervised machine learning models as defined by Burkart and Huber (2021). In this study, we observe the dimensions of data dependency and agnostic models to be the most relevant. However, we also provide discussion as it pertains to the other explainable dimensions and how they may or may not be applicable to our study.

6.1 Data Dependent

Our models offered similar trends in results when applying processed data for both projects and CWE vulnerability types. The first scenario offered limited to no improvement as shown in a comparison view with the CNN model from Tables 5 and 6. The second scenario offered a significant increase when cross-testing in all models, starting with data shown in Tables 7 and 8. Similar results are seen throughout the remaining models. In further evaluating these results, our Kruskal-Wallis tests provide confirmation that there is indeed a statistically significant difference among the AP scores observed based on the scenario data.

From such results analysis, we can conclude that supervised text-based machine learning models for vulnerability detection may suffer from an explainability dimension regarding data dependency. As stated by Burkart and Huber (2021), creating explainability towards a model may stem from the dependency of data in which the model is built upon. As observed from the comparative view of our results and results from the empirical study by Napier et al. (2023), the CNN model performs better after data has been further processed. Even though additional comparative data from the empirical study is not present, we can assume likewise results given the data we have presented. Given this insight, usage of different datasets or even further processing towards the data being observed in this study may provide additional benefit towards the effectiveness of such models.

6.2 Agnostic

In conjunction with the conclusion that data dependency is an explainable dimension in which text-based machine learning models may lack, agnostic models are also present. Although we observed an increase in cross-testing scores for both projects and CWE vulnerability types with the various models, further analysis was needed in determining if such results are specific to a setting. For this analysis we presented Kruskal-Wallis tests based on formulated hypotheses in which we determine if there is a statistically significant difference in the models. From the results shown in Tables 18, 20, 22, and 24, we concluded that there was no statistically significant difference among the models. That is, no single model outperforms the other and we can generalize such a result in stating that the models are agnostic.

As stated by Ribeiro et al. (2016) and Molnar (2018), such explanation is derived in a post-hoc fashion. That is, when observing all results from the different types of data and models, we generalize the result in that the model may or may not be effective given a specific scenario. We further support this claim due to the variety of models used and assume that regardless of model, the result will be the same for such as text-based approach to vulnerability detection. Therefore, regardless of the type of model used, the results can be similar. Even with a data dependency dimensions, improvement upon one model likely leads to improvement with another for this setting.

6.3 Ante-Hoc

We cannot consider ante-hoc to be an applicable dimension towards our study. This derives from the aspect that interpretability was not built into our models from the beginning. As a result, we achieved insights through post-experiment analysis from our scenarios. In modifying out data, this occurs during a pre-processing step and not a characteristic of the models themselves. Even though this step does aid in the models performance, it does not make the models inherently interpretable from the beginning which is the primary aspect of this dimension.

6.4 Post-Hoc

The dimension of "post-hoc" may be applicable to our experiments because we are deriving insights from our models after they have been trained and tested. That is, we analyze the feature overlaps and their correlation with performance. In our study, we utilize the same models from Napier et al. (2023), train and test them on modified data and analyze their performance to derive insights. The process of drawing interpretations after the model has been created, trained, and evaluated fits with the explainable dimensions as defined by Burkart and Huber (2021).

However, we still aim to clarify that while post-hoc methods can provide useful insights, they often do not offer a full understanding of how the model marks decisions, especially for complex models like deep learning or neural networks. As such, the interpretation of "post-hoc" may vary. As an example, we could consider post-hoc as any insights gained after the models are trained, whereas someone else might only consider it post-hoc if we utilizied the exact same models after training. Therefore, this dimension may apply, but we cannot say it is completely relevant.

6.5 Instance/Local

Given our variety of models, we are not interpreting individual predictions made by them. Instead, we have a general pattern that relates to how the models perform based on the overlap of features within the data, deemed "cross-cutting". Furthermore, the observations and insights we have drawn from our study are consistent across multiple models and the entirety of the dataset, reinforcing a possible global, rather than local scope of our insights. While it's conceivable that the phenomenon of feature overlap influences individual predictions, our primary focus is on the broad, overall impact this has on our models, not on the interpretation of individual predictions or small clusters of data. Therefore, this dimension does not apply to our scenario.

6.6 Model/Global

Throughout our work, we gain an understanding that affects all models used, but the understanding pertains to how the models perform with the input data and not about the model's internal workings itself. We observed previous analysis data related to negative correlations between the overlap of features in the data and the performance of the models. This insight applies to the models as a whole because its about how the models performs on an overall basis, given certain characteristics of the input data. Regardless of the specific instance or data point the models are evaluating, the insight about the feature overlap and its impact on performance applies. In our case, the insights primarily pertain to how the input data characteristics affect the overall performance of the models and do not provide a detailed understanding of the inner workings of the models or how it processes input data to make predictions. Therefore, we cannot state that this explainable dimension is relevant.

6.7 Specific

Although we deemed agnostic as a relevant explainable dimension of our experiments, we can still describe how the dimension of "specific" may still apply. That is, depending on whether the insights we have gained can be applied to other models or if they are specific to the ones we used. If we believe that similar behavior would be exhibited by all models given the same data, then it applies. Otherwise, we can state that agnostic applies. For example, in our study, if our insights are applicable only to the particular model classes we observed - RF, MLP, LSVC, BiLSTM, and CNN - then the specific dimension applies. However, if these insights are applicable regardless of the model class, then the specific dimension does not apply.

We also note that the two dimensions, specific and agnostic, are not mutually exclusive but rather denote different aspects of the explainability. As such, a certain insight or mechanism for interpretability could be specific to a certain model class, but it could also be agnostic if it is found to hold true across many model classes. Therefore both dimensions could apply based on the level of generality of our findings. However, given that the feature overlap affects performance across the variety of models we observed, then we elect to deem agnostic as the more relevant dimension.

6.8 Data Dependent

Having already established the relevance of the data dependent dimension, it logically follows that the data independent dimension isn't applicable in our context. Our process involves

modifications to the data, which have proven fundamental in gaining insights into our models. These insights, in turn, have generated superior performance metrics when compared with initial attempts. This was particularly evident when we removed overlapping ("cross-cutting) features, a process highly dependent on the data. The direct impact this process had on the performance of the models presents the significance of the data and its structure in our pursuit of interpretability. Therefore, it's clear that the models are intensely reliant on the data specifics, further emphasizing the data dependent nature of our insights and negating the applicability of the data independent dimension in our study.

6.9 Limitations

In our pursuit of understanding the explainability factor in text-based machine learning models for vulnerability detection, we identify the potential significance of explainable dimensions. However, a concept of a trade-off arises between an enhancement in interpretability and a probable decrease in model accuracy as described in related work such as Duval (2019), Burkart and Huber (2021), and Zhou et al. (2021).

The papers underline several challenges and limitations pertaining to the interpretability and accuracy of machine learning models. They all mention a significant trade-off: more interpretable models, which rely on simpler and more transparent algorithms, tend to be less accurate, as these models might not fully capture the complexity of the data. In contrast, more accurate models, which employ complex algorithms to identify subtle patterns in the data, are usually less interpretable due to their complexity Burkart and Huber (2021).

Further limitations include the potential for explanations to be misleading or incomplete, especially in scenarios where the model is complex or the data is noisy. Such models might inherently be opaque, adding to the difficulty of providing a comprehensive explanation of their behavior Duval (2019). Moreover, the effectiveness of explanations may vary among users, as different levels of expertise require different types of explanations. Some aspects of a model's behavior, particularly those of highly complex models or those operating in dynamic environments, may not be captured at all in the explanations Zhou et al. (2021).

Overall, explainable research emphasizes that, while explanations are fundamental to understanding machine learning models, they inherently possess limitations. This is exemplified in the context of software vulnerability detection, where machine learning models often lack explainability, particularly text-based models that interpret source code. However, our study illustrates that by removing overlapping or "cross-cutting" features present in training and testing datasets, we can potentially improve model performance and reveal explainable aspects of the models' behavior. Consequently, it highlights that explanations, though critical, should be employed with prudence and their development must be continually explored through empirical analysis to uncover the models' performance dynamics, and to address these limitations and enhance their overall effectiveness.

7 Threats to Validity

In this paper, we present results related to providing explainability towards text-based machine learning models as it pertains to vulnerability detection. Using data provided by Napier et al. (2023) and Napier and Bhowmik (2022), we experiment with removal of "cross-cutting" features from a series of projects and CWE vulnerability types. In doing so, we create two scenarios in which such features are removed and evaluate prediction results. During this

process we apply processed data to a series of machine learning models and conduct statistical tests to achieve our findings. However, our work does have limitations which can affect the validity of this research.

7.1 Construct Validity

Construct validity relates to determining if the measurement used is an accurate representation of a variable (Bates and Cozby 2017). In our study, we analyzed both "fixed" and "vulnerable" source code from various projects and CWE types as plain text to establish a baseline for model performance. This approach allowed us to discern general patterns in text-based machine learning models for vulnerability detection. We define two scenarios in which data is processed and re-evaluated for a series of machine learning models. The presence of "cross-cutting" features, discussed in Section 3.2, to some extent, indicates the individual vulnerabilities' distinct features. However, the primary aim of our methodology was to offer a broader overview of vulnerability detection rather than focusing on the specifics of individual cases.

Having access to both training and testing data enabled us to tailor data processing to our needs, including the removal of "cross-cutting" features. While this task may seem straightforward in our controlled environment, applying such a model to "real world" data presents challenges. For instance, when applying a trained model to new data, it might not be possible to determine if the features of the new data were already labeled during model training. Therefore, we cannot fully attest to the results of applying the model in such scenarios. While our results may be subjective, they provide valuable insights into how similar problems could be addressed in a more generalized setting.

7.2 Internal Validity

Internal validity is defined from the accuracy of the conclusions drawn (Bates and Cozby 2017). This study performs experiments on text-based machine learning models based on processed data with features deem as "cross-cutting" removed. We acknowledge that results from the removal of such features may not be fully reliable as alternative data may offer differing results and conclusions. In such a case, this may further prove our conclusions of the explainable dimension of models as it related towards data dependency.

7.3 External Validity

External validity comes from the ability to generalize the accuracy of the findings (Bates and Cozby 2017). Although we suspect "cross-cutting" features play a role in the poor performance of text-based machine learning models, we cannot claim the significance of the role they play. That is, the significance of such features may differ among the data represented by the projects and CWE vulnerability types observed in this setting. Similarly, given the structure in which such data is constructed, features for specific projects or CWE vulnerability types may provide less or more significance. It is also possible that other modifications towards model parameters or processed data may offer greater performance where additional removal may not. Additionally, utilizing other data sources may offer alternative views as it pertains to explainability.

7.4 Reliability

Reliability refers to the stability of a measurement (Bates and Cozby 2017). Based on the empirical data by Napier et al. (2023), we utilize the metric of average precision (AP) scores for comparison purposes. It is possible that other metrics such as precision, recall, among others may be used. In such an effort, additional insights may be presented to which alternative conclusions may be drawn.

8 Conclusion and Future Work

This paper delivers critical insights into the under-performance of text-based machine learning models used for vulnerability detection. We explored various aspects of explainability by identifying its dimensions and examining their limitations. While machine learning has demonstrated effectiveness in specific scenarios for vulnerability detection, it generally lacks explainability. We have sought to understand the dynamics of these models and reasons behind their performances.

We conducted a study using data from a pre-existing empirical study, which suggested the inefficacy of text-based models for vulnerability detection. Later research, building on this empirical study, performed a correlation analysis, hinting that an overlap of features within the training and testing data might contribute to the models' poor performance. These overlapping features, termed "cross-cutting", intersect the available "fixed" and "vulnerable" source code function pairings constituting the training and testing data. In our research, we leveraged these findings, eliminating "cross-cutting" features in two scenarios related to various projects and CWE vulnerability types.

Our analysis yielded divergent outcomes for our scenarios. In our first scenario, S1, the removal of "cross-cutting" features from the training data showed no improvement over previous empirical study results. However, we noted statistically significant enhancements in the models following the elimination of "cross-cutting" features from both training and testing data. Further statistical analysis revealed that text-based machine learning models used in these settings suffer from data dependency and the limitation of being agnostic models. These models are heavily reliant on data and, regardless of the model type, can improve overall, thus generalizing their results.

In this study, we focused on two main dimensions of explainability: Agnostic and Data Dependent. However, as discussed in Section 6, other explainable dimensions could be applicable depending on interpretation, further analysis, or data processing. We also discussed the potential drawback of increasing explainability, where an enhancement in interpretability might result in a decline in accuracy.

Our analysis of both "fixed" and "vulnerable" source code from various projects and CWE types as plain text allowed us to establish a baseline for model performance and discern general patterns in text-based machine learning models for vulnerability detection. As we identified and eliminated "cross-cutting" features, a task facilitated by our data discussed in Section 7, we acknowledged that individual vulnerabilities may possess distinct features, as indicated by the presence of "cross-cutting" features discussed in Section 3.2.

Looking forward, we envision exploring the creation of a model capable of identifying common features as "cross-cutting", enabling their removal to improve effectiveness regardless of the data or programming language in use. To that end, recognizing the diversity in the projects used in our study, future research could focus on enhancing precision by

tailoring features to specific vulnerabilities. This approach represents a crucial direction for future research. By doing so, we intend to delve more deeply into the nuances of text-based vulnerability detection models, extending our exploration of explainability and potentially improving the precision of these models in diverse contexts.

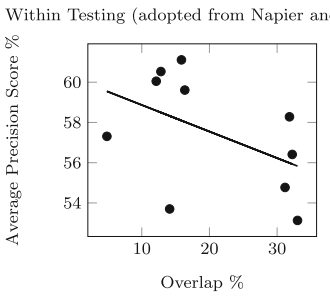
Appendix A: Additional Data and Results

Table 25 Average precision scores from Random Forest (RF) model when removing "cross-cutting" features from *NC* training data for projects

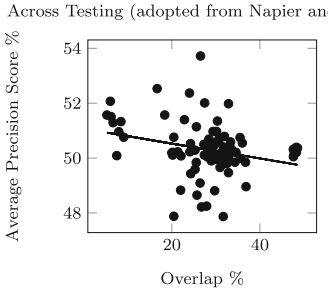
	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
P1	**50.93**%	50.10%	50.00%	49.78%	49.99%	50.24%	50.00%	50.00%	48.84%	50.38%
P2	49.98%	**51.91**%	50.05%	50.07%	49.76%	50.41%	50.51%	50.00%	50.00%	48.92%
P3	49.97%	49.91%	**51.74**%	50.11%	50.01%	50.06%	50.00%	50.26%	49.85%	49.45%
P4	49.95%	49.96%	49.99%	**52.62**%	49.90%	49.71%	50.00%	49.87%	49.70%	50.57%
P5	49.99%	50.03%	49.91%	50.00%	**55.73**%	49.94%	50.25%	50.13%	50.61%	49.81%
P6	49.97%	50.15%	50.08%	49.94%	49.99%	**54.95**%	49.50%	49.75%	50.00%	50.57%
P7	50.03%	49.98%	49.99%	49.94%	49.90%	49.94%	**55.87**%	50.13%	50.00%	50.76%
P8	50.03%	49.98%	50.05%	49.68%	50.00%	50.00%	50.13%	**51.19**%	50.30%	49.63%
P9	50.03%	50.10%	50.00%	50.11%	50.18%	51.57%	50.00%	49.75%	**50.46**%	50.00%
P10	50.08%	49.95%	50.00%	49.77%	50.17%	50.12%	50.00%	49.62%	49.56%	**52.55**%

Table 26 Average precision scores from Linear Support Vector Classification (LSVC) model when removing "cross-cutting" features from *NC* training data for projects

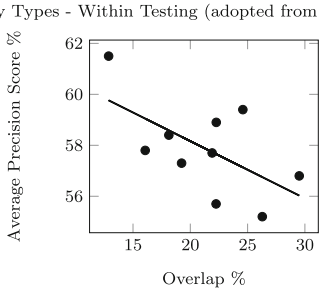
	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
P1	**51.16**%	50.07%	50.00%	49.78%	49.94%	50.24%	50.00%	50.00%	48.84%	50.00%
P2	49.98%	**52.56**%	50.06%	49.98%	49.74%	50.24%	50.38%	50.00%	50.00%	49.45%
P3	49.96%	49.93%	**52.30**%	50.04%	50.06%	50.06%	50.25%	50.26%	50.15%	49.63%
P4	49.97%	49.98%	50.03%	**53.52**%	49.82%	49.83%	50.13%	50.00%	50.00%	50.38%
P5	49.98%	50.03%	49.89%	50.11%	**57.67**%	49.94%	50.13%	50.13%	50.30%	49.63%
P6	49.98%	50.17%	50.08%	49.95%	49.96%	**55.76**%	49.26%	49.87%	50.30%	50.57%
P7	50.01%	49.98%	49.98%	49.84%	50.00%	49.83%	**56.19**%	49.87%	50.00%	51.75%
P8	50.01%	49.98%	50.05%	49.76%	50.03%	50.06%	50.25%	**51.72**%	50.46%	50.19%
P9	50.03%	50.10%	50.00%	50.06%	50.13%	50.35%	50.00%	50.00%	**51.07**%	49.63%
P10	50.07%	49.91%	49.99%	49.81%	50.04%	50.18%	49.75%	50.00%	49.56%	**55.08**%



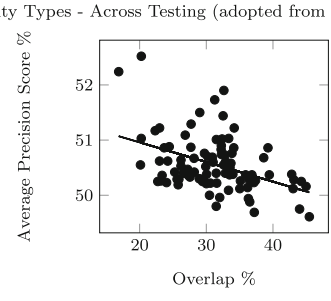
(a)



(b)



(c)



(d)

Fig. 3 Correlation of overlap % of features and APS % in different scenarios. (a) Testing within projects. (b) Testing across projects. (c) Testing within CWE vulnerability types. (d) Testing across CWE vulnerability types

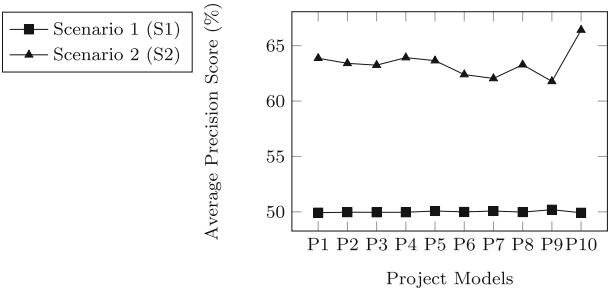


Fig. 4 Average of cross-testing scores for trained projects RF models

Table 27 Average precision scores from Multi-layer Perceptron (MLP) model when removing "cross-cutting" features from *NC* training data for projects

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
P1	**51.09**%	50.06%	49.99%	49.78%	49.94%	50.29%	50.00%	50.00%	48.84%	50.00%
P2	49.98%	**52.54**%	50.03%	50.04%	49.75%	50.18%	50.38%	50.13%	50.00%	50.19%
P3	49.97%	49.91%	**52.28**%	50.04%	49.99%	50.00%	50.38%	50.13%	50.30%	50.00%
P4	49.98%	50.00%	50.04%	**53.54**%	49.86%	49.77%	50.13%	50.00%	50.15%	50.38%
P5	50.00%	50.01%	49.92%	50.00%	**57.17**%	49.94%	50.13%	50.13%	50.15%	50.00%
P6	49.98%	50.17%	50.08%	49.94%	50.07%	**55.00**%	49.38%	49.87%	50.61%	50.38%
P7	50.00%	49.98%	49.98%	49.84%	49.96%	49.77%	**55.24**%	50.26%	50.00%	50.76%
P8	50.03%	49.99%	50.07%	49.89%	49.99%	50.12%	49.87%	**52.01**%	50.30%	50.19%
P9	50.02%	50.10%	49.99%	50.04%	50.10%	50.06%	50.00%	50.00%	**50.81**%	49.81%
P10	50.06%	50.02%	50.12%	50.04%	50.18%	50.00%	50.65%	50.13%	49.56%	**54.25**%

Table 28 Average precision scores from Bidirectional Long Short-Term Memory (BiLSTM) model when removing "cross-cutting" features from *NC* training data for projects

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
P1	**52.67**%	50.20%	50.34%	49.65%	49.94%	50.17%	49.78%	51.08%	48.93%	50.92%
P2	50.02%	**55.60**%	50.06%	50.13%	49.66%	51.00%	50.13%	50.18%	50.36%	50.02%
P3	50.14%	49.69%	**54.93**%	50.61%	49.98%	50.33%	50.31%	50.03%	50.37%	49.26%
P4	50.10%	50.19%	50.05%	**59.44**%	50.45%	49.71%	49.51%	50.18%	50.76%	49.94%
P5	50.10%	49.96%	50.06%	50.43%	**67.48**%	49.61%	50.87%	50.26%	50.58%	50.45%
P6	50.03%	49.79%	50.27%	49.82%	49.76%	**55.65**%	49.19%	49.17%	50.06%	50.53%
P7	50.07%	50.36%	50.11%	50.24%	50.83%	49.38%	**64.06**%	50.57%	51.11%	51.42%
P8	49.90%	49.84%	50.18%	49.95%	49.96%	50.24%	50.97%	**60.50**%	50.02%	49.13%
P9	49.86%	49.83%	49.83%	49.45%	50.30%	52.94%	50.42%	49.86%	**56.23**%	50.34%
P10	50.18%	50.26%	50.07%	50.35%	50.68%	50.20%	50.75%	50.39%	49.93%	**61.43**%

Table 29 Average precision scores from Random Forest (RF) model when removing "cross-cutting" features from *NC* training data for CWE vulnerability types

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
C1	**51.93**%	50.17%	50.05%	50.00%	50.26%	50.25%	50.61%	49.89%	50.05%	49.46%
C2	50.13%	**52.67**%	49.90%	49.99%	50.00%	50.18%	50.39%	49.87%	50.13%	50.40%
C3	50.17%	50.05%	**52.82**%	50.00%	50.09%	50.51%	50.06%	50.03%	49.68%	49.98%
C4	50.07%	50.09%	50.05%	**52.39**%	50.02%	50.03%	50.07%	50.19%	49.82%	49.73%
C5	50.20%	50.12%	50.02%	50.13%	**52.95**%	50.15%	50.38%	50.14%	50.16%	50.38%
C6	49.99%	50.02%	50.23%	49.99%	50.14%	**53.90**%	50.36%	50.06%	49.89%	49.83%
C7	50.11%	50.06%	50.08%	50.10%	50.10%	50.34%	**54.41**%	49.97%	49.92%	49.92%
C8	49.90%	49.95%	49.97%	50.21%	50.19%	49.97%	49.77%	**52.75**%	49.90%	49.73%
C9	50.12%	50.05%	50.04%	50.29%	50.32%	49.92%	50.72%	50.03%	**53.36**%	49.88%
C10	49.90%	50.05%	50.12%	49.86%	49.91%	49.71%	49.95%	49.87%	49.82%	**53.12**%

Table 30 Average precision scores from Linear Support Vector (LSVC) model when removing "cross-cutting" features from *NC* training data for CWE vulnerability types

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
C1	**52.71**%	50.16%	50.09%	50.04%	50.34%	50.27%	50.60%	49.94%	49.57%	49.47%
C2	50.24%	**53.84**%	49.93%	49.98%	50.06%	50.20%	50.32%	49.73%	50.06%	50.36%
C3	50.23%	50.06%	**54.15**%	50.07%	50.16%	50.60%	50.04%	50.08%	49.79%	50.02%
C4	50.10%	50.08%	50.02%	**53.39**%	49.98%	50.12%	50.13%	50.13%	49.94%	49.83%
C5	50.24%	50.00%	50.11%	50.10%	**54.41**%	50.31%	50.45%	50.14%	50.13%	50.25%
C6	50.01%	50.08%	50.13%	49.93%	50.11%	**56.17**%	50.35%	50.08%	49.62%	49.88%
C7	50.12%	50.17%	50.03%	50.11%	50.10%	50.33%	**56.72**%	49.98%	49.82%	49.96%
C8	49.93%	49.93%	49.96%	50.24%	50.18%	49.96%	49.87%	**54.02**%	49.94%	49.67%
C9	50.12%	50.09%	50.04%	50.45%	50.49%	49.94%	50.77%	50.11%	**55.21**%	49.88%
C10	49.93%	50.03%	50.13%	49.88%	49.84%	49.73%	50.04%	49.86%	49.94%	**54.89**%

Table 31 Average precision scores from Multi-layer Perceptron (MLP) model when removing "cross-cutting" features from *NC* training data for CWE vulnerability types

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
C1	**52.64**%	50.15%	50.11%	50.04%	50.27%	50.36%	50.54%	49.91%	49.86%	49.79%
C2	50.30%	**53.70**%	49.97%	50.01%	50.04%	50.17%	50.31%	49.94%	50.14%	50.13%
C3	50.21%	50.00%	**54.07**%	50.10%	50.10%	50.64%	50.06%	50.02%	49.76%	49.92%
C4	50.10%	50.11%	50.09%	**53.70**%	49.96%	50.07%	49.86%	50.19%	50.11%	49.63%
C5	50.20%	49.99%	50.20%	50.15%	**54.37**%	50.29%	50.49%	50.38%	49.97%	50.21%
C6	50.02%	50.08%	50.10%	49.92%	50.03%	**55.90**%	50.56%	50.06%	49.59%	50.10%
C7	50.08%	50.23%	50.04%	50.18%	50.17%	50.21%	**56.63**%	49.73%	49.86%	49.85%
C8	49.95%	49.91%	49.99%	50.26%	50.17%	49.97%	49.95%	**53.97**%	49.97%	49.71%
C9	50.10%	50.08%	50.10%	50.47%	50.38%	50.04%	50.62%	50.22%	**54.90**%	49.83%
C10	49.92%	49.98%	50.12%	49.87%	49.91%	49.74%	49.99%	49.84%	49.97%	**54.74**%

Table 32 Average precision scores from Bidirectional Long Short-Term Memory (BiLSTM) model when removing "cross-cutting" features from *NC* training data for CWE vulnerability types

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
C1	**56.16**%	50.72%	50.53%	50.46%	50.52%	50.92%	51.22%	50.07%	50.11%	49.95%
C2	50.84%	**59.19**%	50.40%	50.01%	49.89%	50.94%	51.78%	49.58%	51.04%	50.78%
C3	50.83%	50.53%	**58.86**%	50.51%	50.42%	50.90%	50.32%	50.14%	50.22%	50.40%
C4	50.51%	50.17%	50.26%	**57.76**%	50.11%	50.28%	49.77%	50.42%	50.32%	49.21%
C5	50.69%	50.27%	50.60%	50.29%	**59.94**%	50.46%	50.96%	50.95%	51.56%	49.95%
C6	50.33%	50.38%	50.47%	50.04%	50.19%	**63.11**%	50.84%	50.20%	50.04%	49.99%
C7	50.51%	50.73%	50.15%	50.60%	50.30%	50.67%	**63.62**%	49.27%	50.31%	50.03%
C8	50.13%	49.95%	50.02%	50.63%	50.31%	50.05%	49.43%	**58.48**%	50.15%	50.03%
C9	50.43%	50.72%	50.41%	50.85%	50.71%	50.24%	50.91%	50.30%	**61.90**%	49.81%
C10	49.99%	50.21%	50.53%	50.15%	49.95%	49.80%	50.59%	48.89%	50.14%	**60.53**%

Table 33 Average precision scores from Random Forest (RF) model when removing "cross-cutting" features from *NC* training data and *FC* testing data for projects

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
P1	**50.93**%	62.83%	62.03%	57.22%	62.38%	74.84%	74.62%	65.69%	77.22%	37.97%
P2	58.45%	**51.91**%	61.80%	56.87%	62.43%	74.84%	74.23%	67.62%	77.78%	36.61%
P3	58.49%	62.93%	**51.74**%	57.09%	62.66%	74.69%	74.62%	66.02%	76.92%	35.74%
P4	58.56%	62.90%	62.17%	**52.62**%	62.61%	74.53%	74.81%	65.07%	77.52%	37.11%
P5	58.56%	62.94%	62.21%	57.22%	**55.73**%	75.39%	74.81%	66.04%	78.31%	37.34%
P6	58.64%	62.95%	62.19%	57.22%	62.74%	**54.95**%	75.57%	66.04%	78.05%	38.12%
P7	58.58%	62.95%	62.19%	57.22%	62.74%	74.77%	**55.87**%	66.36%	78.05%	35.48%
P8	58.58%	62.95%	62.21%	57.22%	62.74%	74.61%	75.00%	**51.19**%	78.31%	37.89%
P9	58.58%	62.95%	62.19%	57.25%	62.63%	74.53%	74.62%	65.71%	**50.46**%	37.51%
P10	58.57%	62.94%	62.19%	57.19%	62.77%	74.77%	74.62%	66.36%	78.31%	**52.55**%

Table 34 Average precision scores from Linear Support Vector Classification (LSVC) model when removing "cross-cutting" features from *NC* training data and *FC* testing data for projects

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
P1	**51.16**%	62.83%	62.03%	57.22%	62.38%	74.84%	74.62%	65.69%	77.22%	37.97%
P2	58.45%	**52.56**%	61.80%	56.87%	62.43%	74.84%	74.23%	67.62%	77.78%	36.61%
P3	58.49%	62.93%	**52.30**%	57.09%	62.66%	74.69%	74.62%	66.02%	76.92%	35.74%
P4	58.56%	62.90%	62.17%	**53.52**%	62.61%	74.53%	74.81%	65.07%	77.52%	37.11%
P5	58.56%	62.94%	62.21%	57.22%	**57.67**%	75.39%	74.81%	66.04%	78.31%	37.34%
P6	58.64%	62.95%	62.19%	57.22%	62.74%	**55.76**%	75.57%	66.04%	78.05%	38.12%
P7	58.58%	62.95%	62.19%	57.22%	62.74%	74.77%	**56.19**%	66.36%	78.05%	35.48%
P8	58.58%	62.95%	62.21%	57.22%	62.74%	74.61%	75.00%	**51.72**%	78.31%	37.89%
P9	58.58%	62.95%	62.19%	57.25%	62.63%	74.53%	74.62%	65.71%	**51.07**%	37.51%
P10	58.57%	62.94%	62.19%	57.19%	62.77%	74.77%	74.62%	66.36%	78.31%	**55.08**%

Table 35 Average precision scores from Multi-layer Perceptron (MLP) model when removing "cross-cutting" features from *NC* training data and *FC* testing data for projects

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
P1	**51.09**%	62.83%	62.03%	57.22%	62.38%	74.84%	74.62%	65.69%	77.22%	37.97%
P2	58.45%	**52.54**%	61.80%	56.87%	62.43%	74.84%	74.23%	67.62%	77.78%	36.61%
P3	58.49%	62.93%	**52.28**%	57.09%	62.66%	74.69%	74.62%	66.02%	76.92%	35.74%
P4	58.56%	62.90%	62.17%	**53.54**%	62.61%	74.53%	74.81%	65.07%	77.52%	37.11%
P5	58.56%	62.94%	62.21%	57.22%	**57.17**%	75.39%	74.81%	66.04%	78.31%	37.34%
P6	58.64%	62.95%	62.19%	57.22%	62.74%	**55.00**%	75.57%	66.04%	78.05%	38.12%
P7	58.58%	62.95%	62.19%	57.22%	62.74%	74.77%	**55.24**%	66.36%	78.05%	35.48%
P8	58.58%	62.95%	62.21%	57.22%	62.74%	74.61%	75.00%	**52.01**%	78.31%	37.89%
P9	58.58%	62.95%	62.19%	57.25%	62.63%	74.53%	74.62%	65.71%	**50.81**%	37.51%
P10	58.57%	62.95%	62.20%	57.19%	62.77%	74.77%	75.00%	66.36%	78.31%	**54.25**%

Table 36 Average precision scores from Bidirectional Long Short-Term Memory (BiLSTM) model when removing "cross-cutting" features from *NC* training data and *FC* testing data for projects

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
P1	**52.67**%	62.82%	62.04%	57.32%	62.23%	74.84%	74.62%	65.35%	77.22%	37.97%
P2	58.32%	**55.60**%	61.42%	56.54%	62.15%	73.88%	73.81%	66.31%	79.17%	36.03%
P3	58.53%	62.94%	**54.93**%	57.09%	62.64%	74.69%	74.62%	66.02%	76.92%	35.16%
P4	58.53%	62.90%	62.17%	**59.44**%	62.65%	74.53%	75.07%	65.06%	77.51%	37.11%
P5	58.56%	62.94%	62.21%	57.22%	**67.48**%	75.39%	75.19%	66.04%	77.62%	37.34%
P6	58.70%	63.00%	62.27%	57.22%	62.84%	**55.65**%	77.34%	67.31%	81.54%	38.12%
P7	58.58%	62.95%	62.24%	57.25%	62.75%	74.69%	**64.06**%	66.36%	78.05%	35.48%
P8	58.58%	62.95%	62.21%	57.26%	62.75%	74.61%	75.00%	**60.50**%	78.31%	37.89%
P9	58.60%	62.96%	62.23%	57.36%	62.72%	75.24%	75.78%	66.35%	**56.23**%	37.74%
P10	58.57%	62.96%	62.22%	57.19%	62.77%	74.77%	74.62%	66.36%	78.31%	**61.43**%

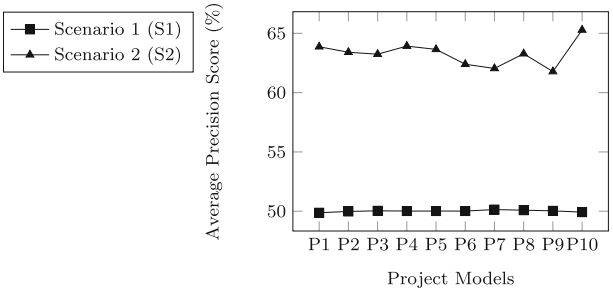


Fig. 5 Average of cross-testing scores for trained projects LSVC models

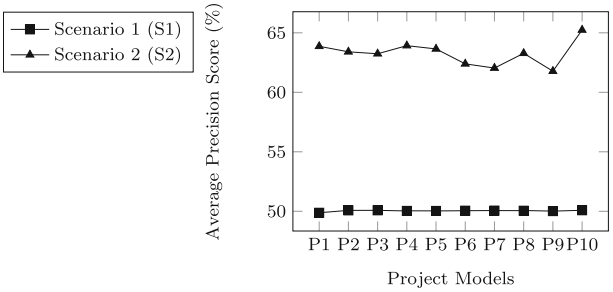


Fig. 6 Average of cross-testing scores for trained projects MLP models

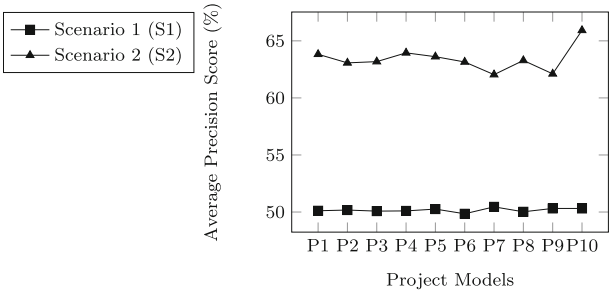


Fig. 7 Average of cross-testing scores for trained projects BiLSTM models

Table 37 Average precision scores from Random Forest (RF) model when removing "cross-cutting" features from *NC* training data and *FC* testing data for CWE vulnerability types

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
C1	**51.93**%	61.71%	62.22%	59.98%	62.36%	59.91%	49.51%	63.38%	57.16%	59.95%
C2	60.51%	**52.67**%	61.57%	60.60%	62.03%	60.58%	50.21%	64.75%	56.64%	59.16%
C3	60.86%	62.41%	**52.82**%	61.92%	63.96%	59.54%	50.75%	66.10%	58.00%	59.80%
C4	60.91%	62.75%	63.18%	**52.39**%	64.44%	60.62%	50.41%	65.15%	58.58%	60.73%
C5	60.92%	62.79%	63.14%	62.52%	**52.95**%	60.89%	51.25%	65.39%	58.05%	60.28%
C6	61.09%	62.95%	62.18%	61.24%	63.40%	**53.90**%	51.78%	65.94%	58.84%	61.12%
C7	61.41%	62.69%	63.11%	61.85%	64.07%	62.12%	**54.41**%	66.15%	57.57%	60.94%
C8	61.04%	63.03%	62.86%	61.74%	63.67%	61.35%	50.94%	**52.75**%	58.90%	61.07%
C9	61.02%	62.86%	62.50%	61.74%	63.38%	61.24%	49.89%	66.41%	**53.36**%	60.98%
C10	61.11%	63.09%	62.82%	61.63%	63.43%	61.34%	51.09%	65.95%	59.24%	**53.12**%

Table 38 Average precision scores from Linear Support Vector Classification (LSVC) model when removing "cross-cutting" features from *NC* training data and *FC* testing data for CWE vulnerability types

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
C1	**52.71**%	61.71%	62.20%	59.98%	62.36%	59.91%	49.51%	63.38%	57.16%	59.95%
C2	60.51%	**53.84**%	61.57%	60.59%	62.03%	60.58%	50.21%	64.75%	56.64%	59.16%
C3	60.86%	62.41%	**54.15**%	61.92%	63.96%	59.54%	50.75%	66.10%	58.00%	59.80%
C4	60.91%	62.75%	63.18%	**53.39**%	64.44%	60.62%	50.41%	65.15%	58.58%	60.73%
C5	60.91%	62.79%	63.14%	62.52%	**54.41**%	60.89%	51.25%	65.39%	58.05%	60.28%
C6	61.09%	62.95%	62.18%	61.24%	63.40%	**56.17**%	51.78%	65.94%	58.84%	61.12%
C7	61.41%	62.69%	63.11%	61.85%	64.07%	62.12%	**56.72**%	66.15%	57.57%	60.94%
C8	61.04%	63.03%	62.86%	61.74%	63.67%	61.35%	50.94%	**54.02**%	58.90%	61.07%
C9	61.02%	62.86%	62.50%	61.74%	63.38%	61.24%	49.89%	66.41%	**55.21**%	60.98%
C10	61.11%	63.09%	62.82%	61.63%	63.43%	61.34%	51.09%	65.95%	59.24%	**54.89**%

Table 39 Average precision scores from Multi-layer Perceptron (MLP) model when removing "cross-cutting" features from *NC* training data and *FC* testing data for CWE vulnerability types

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
C1	**52.64**%	61.71%	62.20%	59.98%	62.36%	59.91%	49.51%	63.38%	57.16%	59.95%
C2	60.51%	**53.70**%	61.57%	60.59%	62.03%	60.58%	50.21%	64.75%	56.64%	59.16%
C3	60.86%	62.41%	**54.07**%	61.92%	63.96%	59.54%	50.75%	66.10%	58.00%	59.80%
C4	60.91%	62.75%	63.18%	**53.70**%	64.44%	60.62%	50.41%	65.15%	58.58%	60.73%
C5	60.91%	62.79%	63.14%	62.52%	**54.37**%	60.89%	51.25%	65.39%	58.05%	60.28%
C6	61.09%	62.95%	62.18%	61.24%	63.40%	**55.90**%	51.78%	65.94%	58.84%	61.12%
C7	61.41%	62.69%	63.11%	61.85%	64.07%	62.12%	**56.63**%	66.15%	57.57%	60.94%
C8	61.04%	63.03%	62.86%	61.74%	63.67%	61.35%	50.94%	**53.97**%	58.90%	61.07%
C9	61.02%	62.86%	62.50%	61.74%	63.38%	61.24%	49.89%	66.41%	**54.90**%	60.98%
C10	61.11%	63.09%	62.82%	61.63%	63.43%	61.34%	51.09%	65.95%	59.24%	**54.74**%

Table 40 Average precision scores from Bidirectional Long Short-Term Memory model when removing "cross-cutting" features from *NC* training data and *FC* testing data for CWE vulnerability types

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
C1	**56.16**%	61.41%	62.06%	59.74%	61.88%	59.49%	50.01%	62.97%	56.83%	59.78%
C2	60.51%	**59.19**%	61.31%	60.45%	61.84%	60.46%	50.11%	64.80%	56.27%	59.28%
C3	60.81%	62.24%	**58.86**%	61.67%	63.68%	59.22%	50.57%	66.20%	58.01%	59.43%
C4	60.85%	62.70%	63.09%	**57.76**%	64.31%	60.49%	50.20%	64.92%	58.26%	60.58%
C5	60.60%	62.29%	62.67%	61.54%	**59.94**%	60.13%	50.73%	64.71%	57.38%	59.06%
C6	61.14%	62.91%	62.01%	61.18%	63.28%	**63.11**%	51.68%	66.05%	58.62%	61.22%
C7	61.42%	62.69%	63.08%	61.76%	63.96%	62.07%	**63.62**%	66.16%	57.55%	60.87%
C8	61.03%	63.00%	62.86%	61.73%	63.65%	61.35%	50.94%	**58.48**%	58.90%	61.14%
C9	61.02%	62.83%	62.47%	61.62%	63.43%	61.27%	49.78%	66.35%	**61.90**%	60.98%
C10	61.11%	63.07%	62.82%	61.63%	63.48%	61.36%	51.09%	65.95%	59.24%	**60.53**%

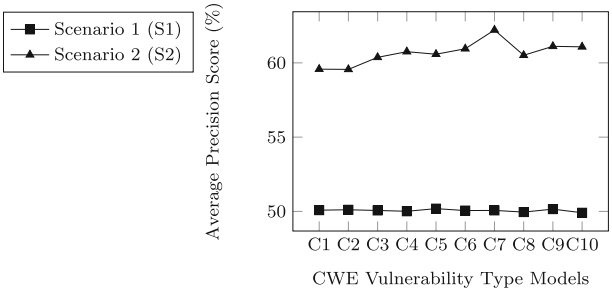


Fig. 8 Average of cross-testing scores for trained CWE vulnerability types RF models

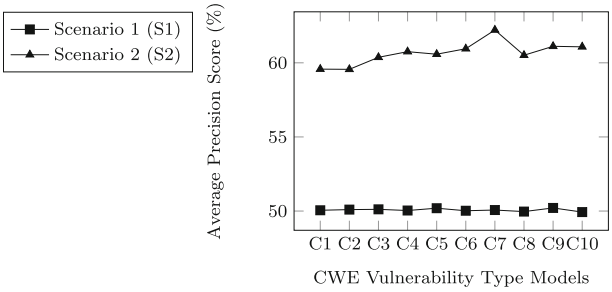


Fig. 9 Average of cross-testing scores for trained CWE vulnerability types LSVC models

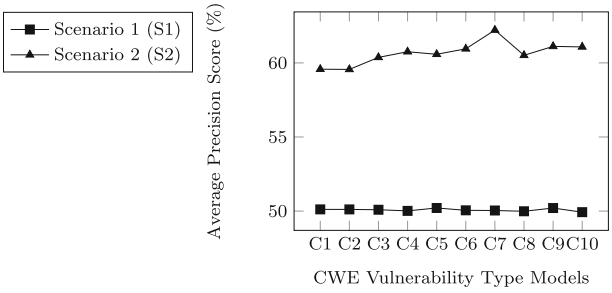


Fig. 10 Average of cross-testing scores for trained CWE vulnerability types MLP models

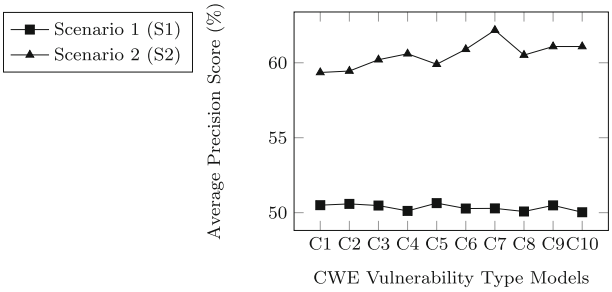


Fig. 11 Average of cross-testing scores for trained CWE vulnerability types BiLSTM models

Data Availability The datasets generated during and/or analyzed during the current study are available in the "explainability_data" repository, https://github.com/krn65/explainability_data

Declarations

Conflicts of Interest The authors of this manuscript have no conflicts of interest.

References

Alenezi M, Zarour M (2020) On the relationship between software complexity and security. arXiv preprint arXiv:2002.07135

Ban X, Liu S, Chen C, Chua C (2019) A performance evaluation of deep-learnt features for software vulnerability detection. Concurr Comput Pract Exp 31(19):e5103. https://doi.org/10.1002/cpe.5103

Bates S, Cozby P (2017) Methods in behavioral research. McGraw-Hill Education

Braz L, Aeberhard C, Çalikli G, Bacchelli A (2022) Less is more: supporting developers in vulnerability detection during code review. In: Proceedings of the 44th international conference on software engineering, pp 1317–1329

Burkart N, Huber MF (2021) A survey on the explainability of supervised machine learning. J Artif Intell Res 70:245–317

Chernis B, Verma R (2018) Machine learning methods for software vulnerability detection. In: Proceedings of the fourth acm international workshop on security and privacy analytics, pp 31–39. https://doi.org/10.1145/3180445.3180453

Czerwonka J, Greiler M, Tilford J (2015) Code reviews do not find bugs. How the current code review best practice slows us down. In: 2015 IEEE/ACM 37th IEEE International conference on software engineering, vol 2. IEEE, pp 27–28

Duval A (2019) Explainable artificial intelligence (xai). N/A N/A:N/A. https://doi.org/10.13140/RG.2.2.24722.09929

Edmundson A, Holtkamp B, Rivera E, Finifter M, Mettler A, Wagner D (2013) An empirical study on the effectiveness of security code review. In: International symposium on engineering secure software and systems. Springer, pp 197–212

Fan J, Li Y, Wang S, Nguyen TN (2020) Ac/c++ code vulnerability dataset with code changes and cve summaries. In: Proceedings of the 17th international conference on mining software repositories, pp 508–512. https://doi.org/10.1145/3379597.3387501

Feng Z, Guo D, Tang D, Duan N, Feng X, Gong M, Shou L, Qin B, Liu T, Jiang D et al (2020) Codebert: a pre-trained model for programming and natural languages. arXiv preprint arXiv:2002.08155

Ghaffarian SM, Shahriari HR (2017) Software vulnerability analysis and discovery using machine-learning and data-mining techniques: a survey. ACM Comput Surv (CSUR) 50(4):1–36. https://doi.org/10.1145/3092566

Gkortzis A, Feitosa D, Spinellis D (2021) Software reuse cuts both ways: an empirical analysis of its relationship with security vulnerabilities. J Syst Softw 172:110653

Grieco G, Grinblat GL, Uzal L, Rawat S, Feist J, Mounier L (2016) Toward large-scale vulnerability discovery using machine learning. In: Proceedings of the sixth acm conference on data and application security and privacy, pp 85–96. https://doi.org/10.1145/2857705.2857720

Harer JA, Kim LY, Russell RL, Ozdemir O, Kosta LR, Rangamani A, Hamilton LH, Centeno GI, Key JR, Ellingwood PM et al (2018) Automated software vulnerability detection with machine learning. arXiv preprint arXiv:1803.04497, https://arxiv.org/abs/1803.04497

Hovsepyan A, Scandariato R, Joosen W, Walden J (2012) Software vulnerability prediction using text analysis techniques. In: Proceedings of the 4th international workshop on Security measurements and metrics, pp 7–10. https://doi.org/10.1145/2372225.2372230

Ijaz M, Durad MH, Ismail M (2019) Static and dynamic malware analysis using machine learning. In: 2019 16th International bhurban conference on applied sciences and technology (IBCAST). IEEE, pp 687–691. https://doi.org/10.1109/IBCAST.2019.8667136

Jie G, Xiao-Hui K, Qiang L (2016) Survey on software vulnerability analysis method based on machine learning. In: 2016 IEEE First International conference on data science in cyberspace (DSC). IEEE, pp 642–647. https://doi.org/10.1109/DSC.2016.33

Kruskal WH, Wallis WA (1952) Use of ranks in one-criterion variance analysis. J Am Stat Assoc 47(260):583–621. https://doi.org/10.1080/01621459.1952.10483441

Li Z, Zou D, Xu S, Chen Z, Zhu Y, Jin H (2021a) Vuldeelocator: a deep learning-based fine-grained vulnerability detector. IEEE Trans Dependable Secure Comput. https://doi.org/10.1109/TDSC.2021.3076142

Li Z, Zou D, Xu S, Jin H, Zhu Y, Chen Z (2021b) Sysevr: a framework for using deep learning to detect software vulnerabilities. IEEE Trans Dependable Secure Comput. https://doi.org/10.1109/TDSC.2021.3051525

Lin G, Zhang J, Luo W, Pan L, De Vel O, Montague P, Xiang Y (2019) Software vulnerability discovery via learning multi-domain knowledge bases. IEEE Trans Dependable Secure Comput. https://doi.org/10.1109/TDSC.2019.2954088

Lin G, Wen S, Han QL, Zhang J, Xiang Y (2020) Software vulnerability detection using deep neural networks: a survey. Proc IEEE 108(10):1825–1848. https://doi.org/10.1109/JPROC.2020.2993293

Lin G, Zhang J, Luo W, Pan L, Xiang Y (2017) Poster: vulnerability discovery with function representation learning from unlabeled projects. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security, pp 2539–2541. https://doi.org/10.1145/3133956.3138840

Lipton ZC (2018) The mythos of model interpretability: in machine learning, the concept of interpretability is both important and slippery. Queue 16(3):31–57

Liu S, Lin G, Qu L, Zhang J, De Vel O, Montague P, Xiang Y (2020) Cd-vuld: cross-domain vulnerability discovery based on deep domain adaptation. IEEE Trans Dependable Secure Comput. https://doi.org/10.1109/TDSC.2020.2984505

Liu B, Shi L, Cai Z, Li M (2012) Software vulnerability discovery techniques: a survey. In: 2012 fourth international conference on multimedia information networking and security. IEEE, pp 152–156. https://doi.org/10.1109/MINES.2012.202

Li Z, Zou D, Xu S, Ou X, Jin H, Wang S, Deng Z, Zhong Y (2018) Vuldeepecker: a deep learning-based system for vulnerability detection. arXiv preprint arXiv:1801.01681, https://doi.org/10.14722/ndss.2018.23158

Mäntylä MV, Lassenius C (2008) What types of defects are really discovered in code reviews? IEEE Trans Software Eng 35(3):430–448. https://doi.org/10.1109/TSE.2008.71

Molnar C (2018) A guide for making black box models explainable. https://christophmgithubio/interpretable-ml-book

Mosolygó B, Vándor N, Antal G, Hegedűs P, Ferenc R (2021) Towards a prototype based explainable javascript vulnerability prediction model. In: 2021 International conference on code quality (ICCQ). IEEE, pp 15–25

Murphy KP (2012) Machine learning: a probabilistic perspective. The MIT Press

Napier K, Bhowmik T, Wang S (2023) An empirical study of text-based machine learning models for vulnerability detection. Empir Softw Eng 28(2):38

Napier K, Bhowmik T (2022) Text-based machine learning models for cross-domain vulnerability prediction: Why they may not be effective? In: 2022 IEEE 23rd International conference on information reuse and integration for data science (IRI). IEEE, pp 158–163

Oliveira D, Rosenthal M, Morin N, Yeh KC, Cappos J, Zhuang Y (2014) It's the psychology stupid: how heuristics explain software vulnerabilities and how priming can illuminate developer's blind spots. In: Proceedings of the 30th annual computer security applications conference, pp 296–305

Perl H, Dechand S, Smith M, Arp D, Yamaguchi F, Rieck K, Fahl S, Acar Y (2015) Vccfinder: finding potential vulnerabilities in open-source projects to assist code audits. In: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security, pp 426–437. https://doi.org/10.1145/2810103.2813604

Ribeiro MT, Singh S, Guestrin C (2016) Model-agnostic interpretability of machine learning. arXiv preprint arXiv:1606.05386

Saleem H (2020) Naveed M (2020) Sok: anatomy of data breaches. Proc Priv Enhancing Technol 4:153–174

Scandariato R, Walden J, Hovsepyan A, Joosen W (2014) Predicting vulnerable software components via text mining. IEEE Trans Software Eng 40(10):993–1006. https://doi.org/10.1109/TSE.2014.2340398

Shahid MR, Debar H (2021) Cvss-bert: explainable natural language processing to determine the severity of a computer security vulnerability from its description. In: 2021 20th IEEE International conference on machine learning and applications (ICMLA). IEEE, pp 1600–1607

Shar LK, Briand LC, Tan HBK (2014) Web application vulnerability prediction using hybrid program analysis and machine learning. IEEE Trans Dependable Secure Comput 12(6):688–707. https://doi.org/10.1109/TDSC.2014.2373377

Shin Y, Meneely A, Williams L, Osborne JA (2010) Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities. IEEE Trans Software Eng 37(6):772–787

Sotgiu A, Pintor M, Biggio B (2022) Explainability-based debugging of machine learning for vulnerability discovery. In: Proceedings of the 17th international conference on availability, reliability and security, pp 1–8

Spreitzenbarth M, Schreck T, Echtler F, Arp D, Hoffmann J (2015) Mobile-sandbox: combining static and dynamic analysis with machine-learning techniques. Int J Inf Secur 14(2):141–153. https://doi.org/10.1007/s10207-014-0250-0

Turpin A, Scholer F (2006) User performance versus precision measures for simple search tasks. In: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, pp 11–18

van Rijsbergen C (1979) Information Retrieval, 2nd edn. Butterworths, London

Yamaguchi F, Lindner F, Rieck K (2011) Vulnerability extrapolation: assisted discovery of vulnerabilities using machine learning. In: Proceedings of the 5th USENIX conference on Offensive technologies, pp 13–13. https://doi.org/10.5555/2028052.2028065

Zeng P, Lin G, Pan L, Tai Y, Zhang J (2020) Software vulnerability analysis and discovery using deep learning techniques: a survey. IEEE Access 8:197158–197172

Zhang Y, Chen X et al (2020) Explainable recommendation: a survey and new perspectives. Foundations and Trends® in Information Retrieval 14(1):1–101

Zhou J, Gandomi AH, Chen F, Holzinger A (2021) Evaluating the quality of machine learning explanations: a survey on methods and metrics. Electronics 10(5):593

Zhou Y, Liu S, Siow J, Du X, Liu Y (2019) Devign: effective vulnerability identification by learning comprehensive program semantics via graph neural networks. Adv Neural Inf Process Syst 32

Zhu M (2004) Recall, precision and average precision. Department of Statistics and Actuarial Science, University of Waterloo, Waterloo 2(30):6

Kollin Napier is the Director of the Mississippi Artificial Intelligence Network (MAIN) and an employee at Mississippi Gulf Coast Community College (MGCCC). MAIN, the nation's first statewide AI initiative, enhances Mississippi's workforce through AI awareness, education, training, and innovation. In his role, Dr. Napier collaborates with community colleges and higher education institutions across Mississippi to drive these efforts. His research focuses on leveraging artificial intelligence (AI) and machine learning (ML) models in cybersecurity and software engineering, particularly for vulnerability detection. This includes the use of natural language processing (NLP) and large language models (LLM). Dr. Napier earned his Ph.D. in Computer Science from Mississippi State University in 2023.

Tanmay Bhowmik received his Ph.D. and Master's in Computer Science from Mississippi State University in 2015 and 2010, respectively. He received his B. Tech. Degree in Computer Science and Engineering from National Institute of Technology, Durgapur, India, in 2007. Dr. Bhowmik joined the Department of Computer Science and Engineering at Mississippi State University in 2016. Prior to joining Mississippi State, he worked at Northwest Missouri State University for a year as a tenure-track Assistant Professor. Dr. Bhowmik's research interests lie in the area of requirements engineering (RE), with three main thrusts: (1) developing methods for capturing and evaluating creative software requirements, (2) bridging RE and software testing, also known as "requirements engineering and testing (RET)," by examining the role of environment assumptions in testing activities, and (3) linking RE with secure software engineering through vulnerability predictions for software requirements. His research endeavors leverage Natural Language Processing (NLP), Machine Learning, and Large Language Models (LLMs) with an aim to bridge the gaps between theory and practice and develop automated solutions for RE-related problems.

Zhiqian Chen is an Assistant Professor in the Department of Computer Science and Engineering at Mississippi State University. Specializing in graph machine learning and its applications, Dr. Chen's research endeavors have garnered support from the National Science Foundation (NSF) and the United States Department of Agriculture (USDA). His accolades include an Outstanding Contribution Award from Toyota Research North America in 2016 and a Best Paper Award from ACM SIGSPATIAL. Dr. Chen has a portfolio of research published in esteemed journals and conferences such as AAAI, IJCAI, IEEE ICDM, EMNLP, ACM Computing Surveys, and Nature Communication. Beyond his scholarly contributions, he has been an active reviewer for prestigious academic platforms including AAAI, ICML, ICLR, NeuralPS, and SIGKDD.