University of Otago

Department of Computer Science

COSC480 Project Report

An Exploration of the Common Vulnerability Scoring System

Author:
Jake Norton (5695756)

Supervisor(s):
Dr. David Eyers
Dr. Veronica Liesaputra

July 30, 2024

Abstract

The Common Vulnerability Scoring System (CVSS [14]) is designed to produce scores for software vulnerabilities. Such a system is needed in order to triage the sheer number of new vulnerabilities being released every year. We cannot keep up with the amount of CVSS scores that need to be produced, as such we need a way to automatically predict them. There is precedent to using machine learning, specifically in more recent times, large language models (LLMs) to accurately predict these CVSS scores [7]. However, there is a general focus on only using the National Vulnerability Database (NVD [7, 1, 4]), it would be ideal if there was more than one source for the ratings, not only for cross validation, but also for an increase in training data volume. Before we use any extra data sources, it will be interesting to do a comparison between the different sources, to see if we can get an estimated accuracy for each of the metrics within the scoring system. Additionally we should know how good of a system CVSS is and whether or not there are better alternatives. Unfortunately CVSS (version 3.1 [25]) is a flawed metric, hopefully once version 4.0 begins to be used commonly, it will solve some of these issues. However, the ability to predict a metric based on a short text description is still useful, and a focus on the interpretability of such a system remains important.

1 Introduction

Last year there were 29,065 new vulnerabilities as show by Figure 1. This is a number that is only going up year on year. These vulnerabilities are recorded using the Common Vulnerabilities and Exposure system (CVE [19]). From these CVEs, CVSS scores can be computed. The National Vulnerability Database (NVD [24]) takes CVEs and enriches them with CVSS data. They are not the only place to do so, however in terms of research they are often the main or sole data provider (E.g. [7, 1, 4]). I explored other options, and landed on the MITRE database [21], as it is the main database for CVEs, with a decent number enriched with CVSS scores. As a guideline to my investigation between the two databases, I used the same method as the paper *Can the Common Vulnerability Scoring System be Trusted? A Bayesian Analysis* [18]. This paper explores the extent to which different data sources agree on the scoring of a CVE, aiming to gain insight into

the potential for establishing a ground truth value. Unfortunately, since that paper was released back in 2016, many of the data sources they compared are either unavailable or in archival status. However, following their method still allows for insight between the two chosen databases, NVD and MITRE. This analysis shows that the databases do fundamentally rate CVEs differently (see Figure 2). The uncertainty between the two can therefore be an indicator going forward when analysing generated CVE scores, as it is likely that the model will also struggle in similar places to where the human evaluators did. In addition to this analysis, I looked into the CVSS system itself. There have been many complaints laid against CVSS (E.g. [12, 31, 30]), these will be discussed in Section 6, but broadly, CVSS is used for more than it is designed to do, leading to an inefficient system.

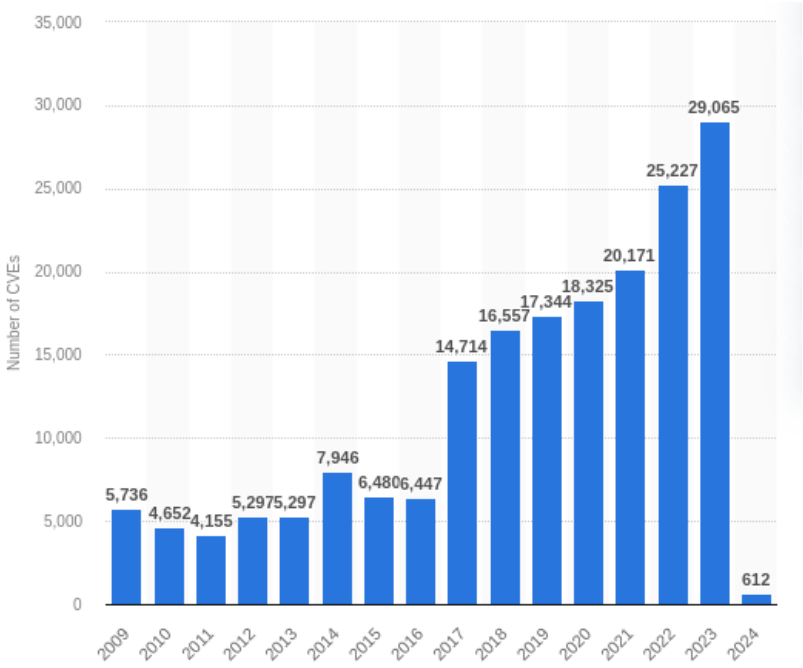

Figure 1: Number of new CVEs by year

2 Background

Vulnerabilities are stored in a consistent system called Common Vulnerabilities and Exposures (CVE [19]).

Here is an example CVE

- Unique Identifier: CVE-2024-38526
- Source: GitHub, Inc.
- Published:06/25/2024
- Updated:06/26/2024
- Description: pdoc provides API Documentation for Python Projects. Documentation generated with `pdoc --math` linked to JavaScript files from polyfill.io. The polyfill.io CDN has been sold and now serves malicious code. This issue has been fixed in pdoc 14.5.1.

Sourced from NVD CVE-2024-38526 Detail [23]

This has a unique identifier, which is given by one of the CVE numbering authorities (CNA [20]), such as GitHub, Google or any of these, CVE list of partners [6]. The description is the most important part in our case. This should provide information about the vulnerability. What can be exploited (device / software component)? How is the product affected if the vulnerability is exploited? Ideally there would be a part of the description that relates to every metric, unfortunately these descriptions are not necessarily suited to machine learning as the people writing the descriptions are expecting a lot of intrinsic knowledge.

The Common Vulnerability Scoring System

CVSS scoring is a high level way to break up vulnerabilities into different categories. Organisations can use it to choose which vulnerability to focus on first. CVSS is broken up into three distinct sections: base, temporal and environmental scores.

For brevity I will only show the specifics of CVSS 3.1 [25] as this is by far the most commonly used version, even if it is not the most recent.

Base Score

- Attack Vector: Defines the avenues of attack that the vulnerability is open to. The more open a component is, the higher the score. This can have the values: `Network`, `Adjacent`, `Local` and `Physical`.
- Attack Complexity: Describes how complex the attack is to orchestrate. Encompasses questions like, what are the prerequisites? How much domain knowledge / background work is necessary? How much effort does the attacker need to invest to succeed? This can have the values: `Low` or `High`. `Low` gives a higher base score.
- Priviledges Required: The degree of privileges the user needs to complete the attack. Ranging from: `None`, `Low` (e.g. User level privilege), `High` (e.g. Administrator). The lower the privilege the higher the base score.
- User Interaction: Describes if the exploit requires another human user to make the attack possible, E.g. clicking a phishing link. This is either `None` or `Required`, the score is highest when no user interaction is required.
- Scope: Defines if the attack can leak into other security scopes. E.g. access to one machine gives the ability to elevate privileges on other parts of the system. This can take `Unchanged` or `Changed`, the score being highest when a scope change occurs.
- Confidentiality Impact: Detemines what is the impact on the information access / disclosure to the attacker. This can be: `High`, `Low` or `None` with `High` adding the most to the base score.
- Integrity Impact: Refers to the integrity of the information within the component. I.e. could the data have been modified by the attacker. This has: `High`, `Low` or `None`, as categories with `High` adding the most to the base score.
- Availability Impact: Refers to the impact of the attack on the availability of the component. E.g. the attacker taking the component off the network, denying the users access. This can be: `High`, `Low` and `None` with `High` adding the most to the base score.

This is a summarized version of the 3.1 specification document provided by the Forum of Incident Response and Security Teams (FIRST) [25].

Temporal

- Exploit Code Maturity: The state of the attack itself, e.g. has this exploit been pulled off in the wild or is it currently academic.
- Remidiation Level: Whether the exploit in question has a patch availabile.
- Report Confidence: The degree of confidence in the CVE report itself, the report may be in early stages where not all of the information is known.

This is a summarized version of the 3.1 specification document provided by the Forum of Incident Response and Security Teams (FIRST) [25].

Temporal metrics would be useful in general for a CVSS score, however NVD do not store these temporal metrics. As far as I can tell there is no reason given for this specifically, though discourse (Stack exchange post) [3] around the subject suggests that this is due to a lack of verifiable reporting. From my perspective, both remidiation level and report confidence feel like they could have scores attributed to them, however finding verifiable reports on the exploits seen in the wild is difficult. There are two relatively new organisations on this front, Cybersecurity & Infrastructure Security Agency (CISA, public sector) and inthewild.org (private sector [5]).

2.1 Data Options

I will be using NVD and MITRE as the sources of data. In 2016 when Johnson et al. did their paper on CVSS [18], they had access to five different databases. Unfortunately only two of these remain for modern data. There are others, but they are either in archival or proprietary status.

National Vulnerability Database

The National Vulnerability Database is the defacto standard dataset used for CVSS generation research [7, 1, 4]. This makes a lot of sense as it is built for the purpose with a team dedicated to enriching CVEs with CVSS scores. The dataset I am using was retrieved using the NVD API in March 2024 and contains ∼100000 CVEs enriched with CVSS scores. This comes in a consistently formatted JSON dump.

MITRE Database

MITRE is the defacto database for the storage of CVEs themselves, their database contains ∼40000 CVEs enriched with CVSS 3.1 scores. These are in a JSON dump retrieved in March 2024. The format for usage is a bit more cumbersome to use. The CVSS scores are only stored as CVSS vector strings (a simple text encoding [28]). These are not hard to parse, though they are stored slightly different between versions, as well as sometimes being inconsistent (∼5000 had temporal metrics within the vector strings in the MITRE database).

2.1.1 Priliminary Data exploration

The scorers for both NVD and MITRE do rate CVEs reasonably similar, one pattern you can see as shown by Fig 2, is that NVD generally give the most common categorical output more ratings. They are less spread out across the full range of values. In addition, if we look at the `Attack Complexity` metric, there is a reasonably large difference in how they are rated, MITRE rate a lot more of the metrics with a `Low` score. This points to some of the difficulty with this kind of rating system, while in theory there is a true value for these metrics, it requires knowledge of the whole space around each of the vulnerabilities, this knowledge will always vary marker to marker. The model will not have direct access to this knowledge; however, it is hoped that it will be able to trace relationships between the different vulnerabilities and learn this intrinsically.

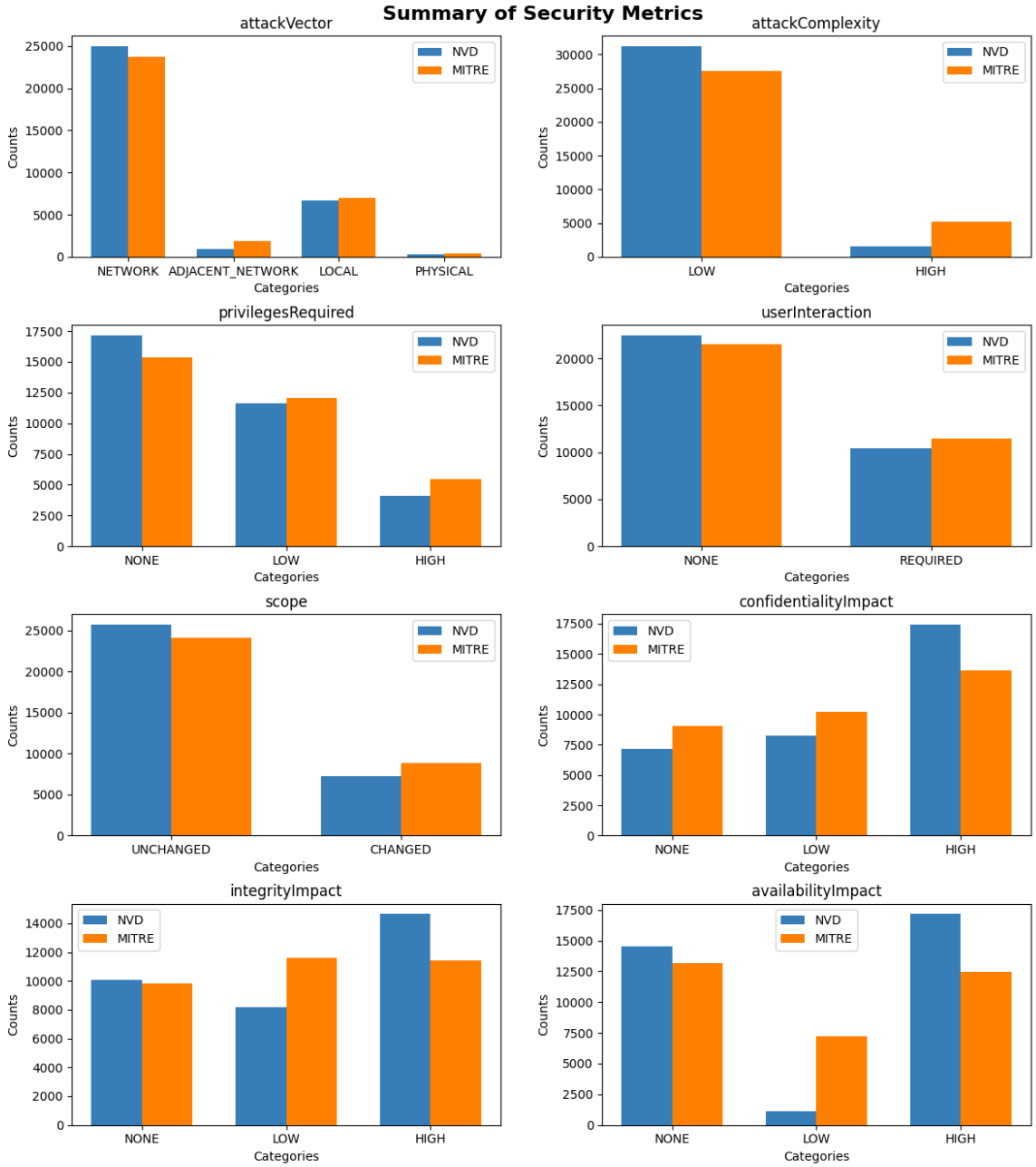

Figure 2: Comparison of CVSS ratings between MITRE and NVD

3 Related Work

Johnson et al. in the paper, *Can the Common Vulnerability Scoring System be Trusted? A Bayesian Analysis* [18], conducted a study into the state of CVSS databases and their accuracy in 2016. They found NVD to be the most correct database, and we can trust the Common Vulnerability Scoring System as a whole as scorers rate CVEs consistently. This paper will be used as the basis for the database comparison in Section 4.1.

Costa et al. in the paper, *Predicting CVSS Metric via Description Interpretation* [7], tested generation of CVSS from CVE descriptions with a range of large language encoder-only models. They achieved state-of-the-art results with the DistilBERT model [29]. They also improved the score with text preprocessing (e.g. lemmatization) and looked into interpretability of the models using Shapley values. This paper is relevant to Section 4.2 as much of the process around training and inferring CVSS scores is based on their work.

Jiang and Atif in the paper, *An Approach to Discover and Assess Vulnerability Severity Automatically in Cyber-Physical Systems* [17], create a novel pipeline for vulnerability assessment. They used multiple data sources and a majority voting system to decrease the chance of badly scored CVEs. This paper relates in that it shows a use case for multiple data sources, though less so on the methods used.

Henry Howland in the paper, *CVSS: Ubiquitous and Broken* [12] broke down issues with the CVSS system, namely, "lack of justification for its underlying formula, inconsistencies in its specification document, and no correlation to exploited vulnerabilities in the wild, it is unable to provide a meaningful metric for describing a vulnerability's severity" [12]. This paper mainly relates to Section 6 exposing the issues and general impression of CVSS.

4 Methods

4.1 Hierarchical Bayesian Model

Analysis between the NVD [24] and MITRE [21] databases is done with a hierarchical bayesian model. This type of model is suitable when you expect the population to be similar in some respects but different in others. In this

case, they share common knowledge towards the vulnerability, but differ in the experience of the people rating the metrics [18]. The model is similar to the original model (see section 4.1 of [18]), it assumes that there exists a true value for each CVSS dimension, but the database sample may not be that true value. These inaccuracies are represented with a confusion matrix (see equation 1). The difference from the original paper is that there is no longer consistently three variables for each CVSS metric, varying from two to four categorical choices.

Below is the confusion matrix of the CVSS dimension `Confidentiality Impact`

$$\Pi_c i = \begin{bmatrix} \pi_{nn} & \pi_{nl} & \pi_{nh} \\ \pi_{ln} & \pi_{ll} & \pi_{lh} \\ \pi_{hn} & \pi_{lh} & \pi_{hh} \end{bmatrix} \tag{1}$$

where π_{nn} denotes the probability that the current database correctly assigns the random vulnerability the score `None` when the actual value is the same. π_{nl} and π_{nh} represent when `None` was not the actual truth value, i.e. the database produced the wrong score.

4.1.1 Priors

The priors for the categorical variables were set up with uninformative priors using a Dirichlet distribution. This will give a uniform prior over the probability space for all categorical options. This is done to not colour the outcome of the results based on prior belief, though, this prior will have little impact for any categorical metric which has more than the number of options for that metric.

The confusion matrices also need priors, as before using Dirichlet distribution. However, we do want to add some initial belief to this, as the people producing scores are not acting completely randomly, and are likely to be right more often than not. These are still weak priors as the number of observations is in the thousands.

4.1.2 Estimation

This follows the Bayesian approach where to estimate the parameters: you take the prior beliefs, take an observation, and update these beliefs to produce posterior beliefs. In this case, the method involves using Markov Chain Monte Carlo methods. Broadly, this allows for simulating the data, based on the previously created distribution, by sampling values that the model has high belief would be from the target distribution. This allows for accurate sampling without having all of the data. The original paper used JAGS [26] and R. As I am more familiar with Python I did try pyjags (a python wrapper around JAGS [35]), however I did not find great success. Instead I used the pymc library [27] to help with the modelling. It fulfilled the same tasks that JAGS did with the original paper [18].

4.2 CVSS Prediction

Cody Airey –a classmate of mine– has been working on a similar problem. He has been repoducing results from Costa et al. [7]. My choice of model for CVSS prediction will very much bootstrap off his work. So far, a strong contender for state-of-the-art model for predicting CVSS metrics from CVE descriptions is the DistilBERT model [29]. This is a variant of BERT [9], a pre-trained deep bidirectional transformer large language model developed by Google. DistilBERT has advantages over other BERT models in terms of performance, but also on speed of training as well as size / memory footprint of the model.

4.2.1 Training

The model is trained separately for each metric. Following Airey's method, each of the eight models were trained on five different data splits to allow for a standard deviation to be calculated, in order to aid in reducing the chance of a *lucky* data split effecting the results. The difference between Costa et al. & Airey's work, and mine is that this model was trained on a combination of NVD and MITRE data as opposed to just using NVD. This was converted to the same format, a CSV containing the descriptions and the CVSS scores. This does mean there are now ∼40000 duplicate CVEs and ∼140000 CVEs enriched with CVSS scores total.

5 Results

5.1 Bayesian Analysis

The results for the database analysis will be shown through the confusion matrices for the estimated accuracy of both NVD and MITRE. Unfortunately it is difficult to compare my results to the original paper [18] as they are on a totally different dataset. However, I will note that in general the estimated accuracy for both datasets is much lower than the scores from the original study. Across the board NVD often had ∼90% accuracy for the highest accuracy field of any metric, with Confidentiality as a clear outlier as seen from Table 5.1.

Some general trends are that NVD (see Figure 3) have more extreme estimated accuracies. They do better for the higher frequency options, for the `Low` option on `Attack Complexity` for example, NVD have 98% estimated accuracy and 66% for `High` versus MITRE (see Figure 4), `Low` scored 87% and 75% for `High`. Instead of further analysing in this way, I will point out some of my worries around these results, not that they are wrong per se, but that they do not really tell us anything extra than that shown by in Figure 2, except perhaps it is helpful to see the results in a percentage estimated accuracy instead of a proportion. Unfortunately this is outside of my strengths, I did some cursory exploration into if doing this sort of analysis between two populations like this does make sense, the discourse here [11] suggests that such a thing can be done, though it also suggests the need for more informative priors. This may apply in my case, and I intend on getting an expert opinion closer to home, however that will be after this report.

Johnson et al. talk about the reliability of their results saying this in section 7.1 of [18]:

"reliability concerns are discussed. In this study we use five different scoring instruments - the databases. If some of these are generally correct, but some are generally incorrect, will not the scores of the incorrect ones still affect our beliefs about the actual values, thus worsening reliability? It turns out that this is not the case. In a simulation, two scorers were set up to be completely aligned, scoring 90% complete impact in one of the CIA dimensions, while a third was unaligned and scoring only 10% complete impact. Initially, all scorers are equally credible, but the gradual accrual of

evidence impacts scorer credibility (as well as beliefs about the actual CVSS score distributions). Thus, as the third scorer rarely matched the other two, its credibility eroded, thus shrinking the weight of its advise [18]."

Unfortunately in my case I do not have the advantage of a potential third or more scorer. This leads me to believe that there is a chance that incorrect scores can end up corrupting the results.

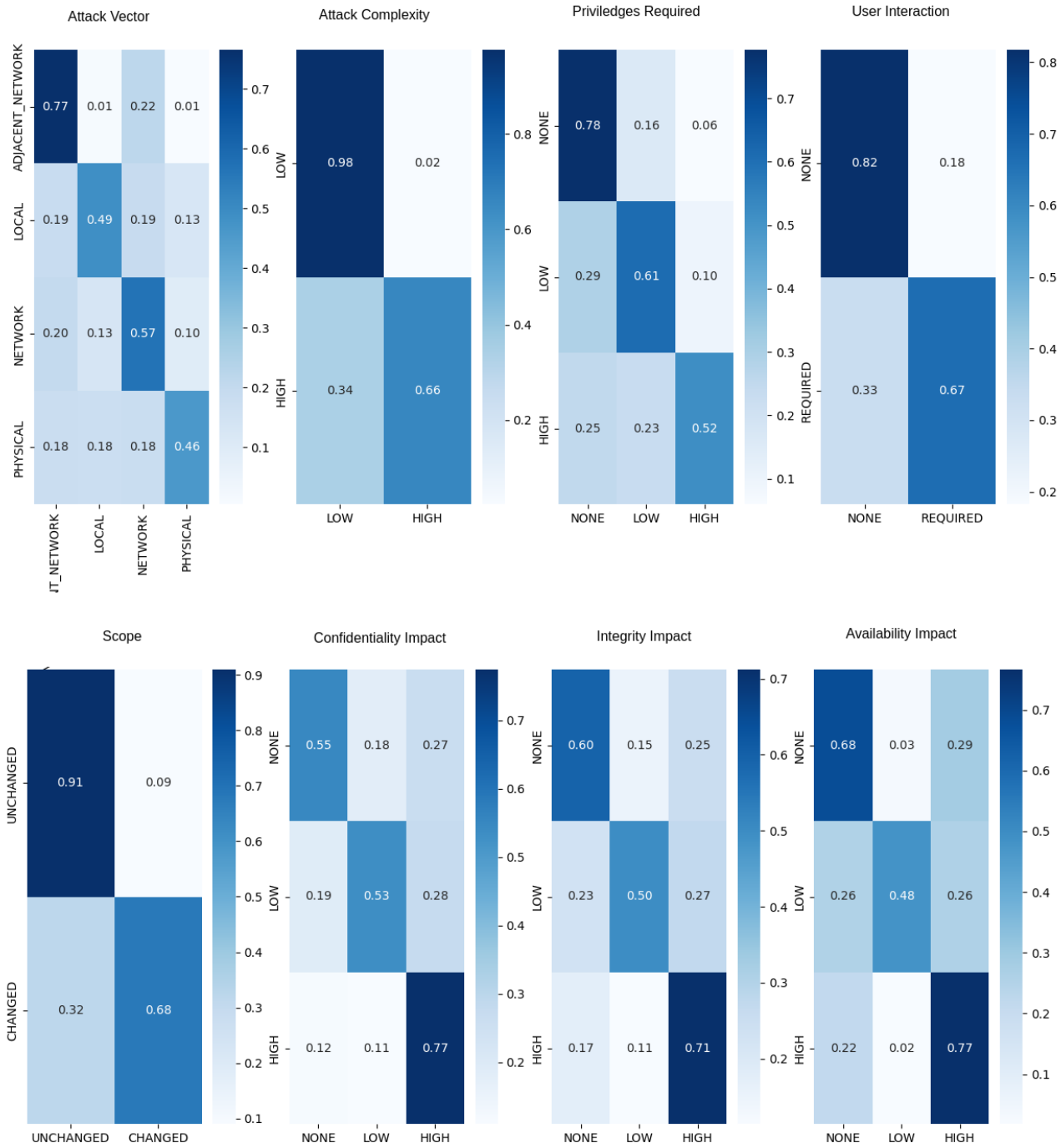

Figure 3: Confusion Matrices for NVD for CVSS version 3.1

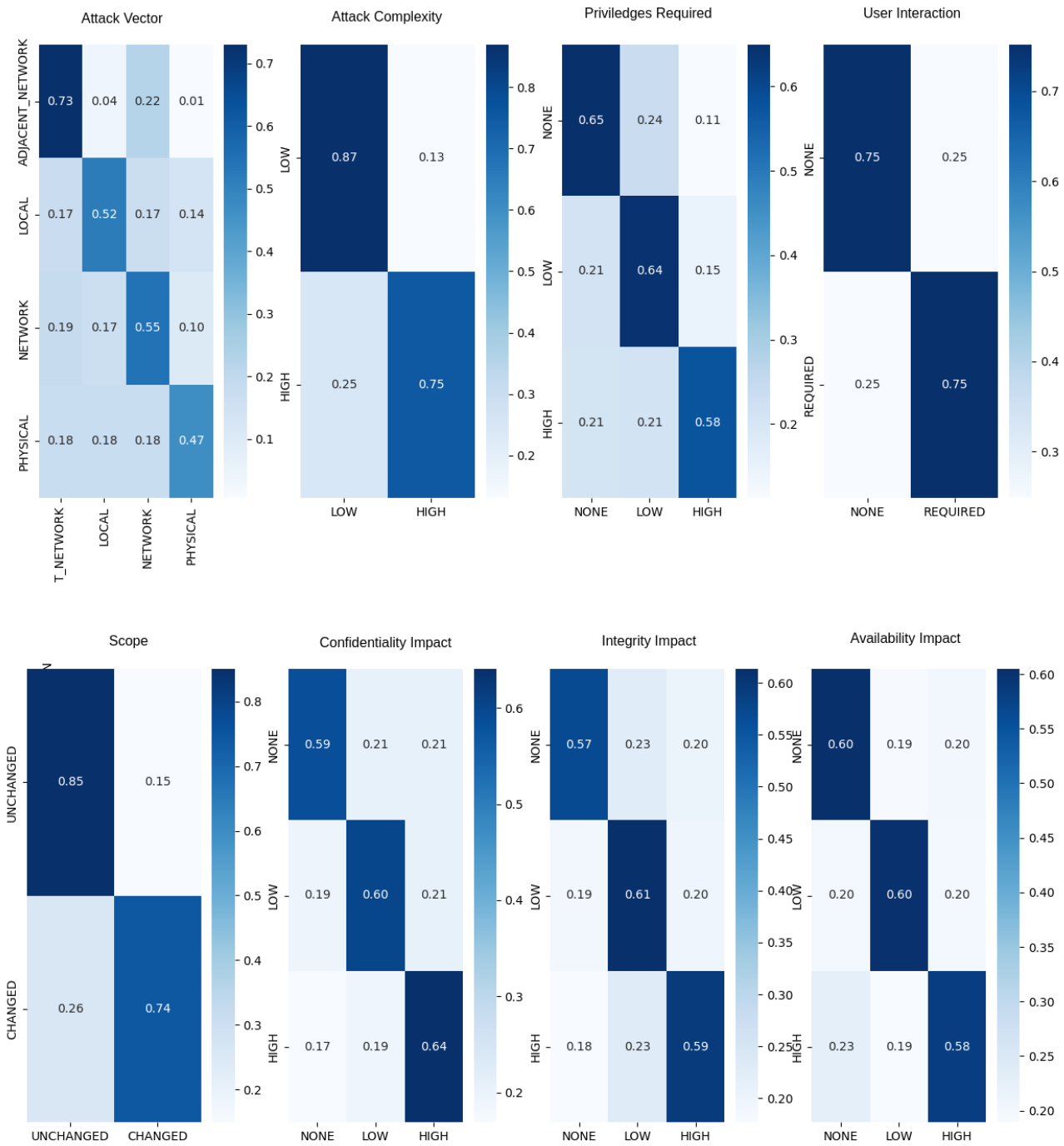

Figure 4: Confusion Matrices for MITRE for CVSS version 3.1

Table 1: Confusion Matrices for NVD on CVSS version 2.0 from Johnson et al. [18]

Access vector			
	N	A	L
N	0.99	0	0
A	0.21	0.71	0.08
L	0.05	0.0	0.95

Access complexity			
	L	M	H
L	0.54	0.29	0.17
M	0.02	0.88	0.1
H	0.01	0.08	0.91

Authentication			
	N	S	M
N	0.99	0.01	0
S	0.04	0.95	0.01
M	0.19	0.2	0.6

Confidentiality			
	C	P	N
C	0.4	0.2	0.2
P	0.2	0.4	0.19
N	0.2	0.21	0.4

Integrity			
	C	P	N
C	0.91	0.08	0.01
P	0.05	0.93	0.02
N	0.02	0.07	0.92

Availability			
	C	P	N
C	0.92	0.07	0.01
P	0.07	0.91	0.02
N	0.02	0.11	0.87

5.2 CVSS Prediction

Below, Table 2 & Table 3 show the results of the DistilBERT model trained on a combination of NVD and MITRE data. Unfortunately this has a purely negative effect on all metrics, with the caveat that some of the standard deviations are lower. Additional note is that the balanced accuracy for some metrics looks a bit weird, I believe that is due to the model not outputing some of the categorical options for that metric. This applies to all the balanced accuracy scores, apart from the Priorities Required (PR) and Confidentiality (C) as seen on Table 2 & Table 3. As this was done only recently I have not looked into this issue in-depth to see why this is happening, but I will in the future. As to why the model performs worse, my theory is that the added data, and therefore the overlapping CVEs with different scores confuse the model. I thought that adding the additional data may have given the model a better chance of generalising, however this does not appear to be the case. The uncertainty between the two databases can serve as an

indicator when analyzing generated CVE scores. It suggests that the model may encounter similar challenges to those faced by human evaluators. The lowest score for my model is in the Availability Impact category, which is also low for Cody's model. This observation indicates a correlation between the machine learning models' difficulties and the issues faced by human scorers. Attack Complexity also presents challenges, likely due to data imbalance, as the dataset is heavily skewed towards the `Low` score, thereby incentivizing the model to output `Low` scores (see Figures 2, 3, 4 for reference).

Metric	Model	AV	AC	PR	UI
Accuracy	DistilBERT-Cody	**91.28 ± 0.26**	**95.64 ± 0.68**	**82.77 ± 0.24**	**93.86 ± 0.19**
	DistilBERT-Jake	72.81 ± 0.32	92.62 ± 0.15	81.18 ± 0.18	66.35 ± 0.24
F1	DistilBERT-Cody	**90.98 ± 0.31**	**93.85 ± 1.39**	**82.53 ± 0.26**	**93.82 ± 0.19**
	DistilBERT-Jake	61.36 ± 0.42	89.08 ± 0.22	80.96 ± 0.19	52.93 ± 0.31
Bal Acc	DistilBERT-Cody	**67.88 ± 2.11**	**55.82 ± 7.23**	**75.98 ± 0.47**	**92.46 ± 0.21**
	DistilBERT-Jake	25.00 ± 0.00	50.00 ± 0.00	75.18 ± 0.31	50.00 ± 0.00

Table 2: Comparison of the effects of the X pre-trained models on the CVSS v3.1 dataset (Part 1).

Metric	Model	S	C	I	A
Accuracy	DistilBERT-Cody	**96.38 ± 0.09**	**86.24 ± 0.20**	**87.15 ± 0.10**	**88.70 ± 0.10**
	DistilBERT-Jake	80.21 ± 0.16	82.45 ± 0.11	45.71 ± 0.26	52.53 ± 0.23
F1	DistilBERT-Cody	**96.30 ± 0.10**	**86.09 ± 0.21**	**87.11 ± 0.10**	**88.04 ± 0.11**
	DistilBERT-Jake	71.40 ± 0.22	82.34 ± 0.12	28.68 ± 0.28	36.18 ± 0.27
Bal Acc	DistilBERT-Cody	**91.57 ± 0.43**	**82.70 ± 0.36**	**85.81 ± 0.10**	**64.01 ± 0.13**
	DistilBERT-Jake	50.00 ± 0.00	79.85 ± 0.23	33.33 ± 0.00	33.33 ± 0.00

Table 3: Comparison of the effects of the X pre-trained models on the CVSS v3.1 dataset (Part 2).

6 Discussion

Case Study: Curl Library Vulnerability

A practical example of the issues with CVSS scoring can be seen with a vulnerability in the curl library [34]. The vulnerability was described as

an integer overflow in `tool_operate.c`, which could result in undefined behaviour if a user entered a value exceeding the integer limit [22]. Initially, this vulnerability received a CVSS score of 9.8. Daniel Stenberg, the original founder of curl, disagreed with this high score, arguing that it was a bug fixed back in 2019 [33]. Despite his efforts, the CVE (Common Vulnerabilities and Exposures) entry remained. Stenberg successfully disputed the score and NVD lowered the score to 3.3 [32]. Stenberg was eventually somewhat successful in mitigating this issue, however it took a good chunk of his valuable time, as well as all others involved. This example highlights the time-consuming nature of rectifying CVSS scores and the potential for initial misratings.

6.1 Evolution of CVSS and Its Identity Crisis

When CVSS 2.0 was released, it was promoted as a framework to help IT management prioritize and remediate vulnerabilities posing the greatest risk. The initial goal was to provide a comprehensive method for assessing risk, as indicated by its original documentation:

> *"Currently, IT management must identify and assess vulnerabilities across many disparate hardware and software platforms. They need to prioritize these vulnerabilities and remediate those that pose the greatest risk. But when there are so many to fix, with each being scored using different scales, how can IT managers convert this mountain of vulnerability data into actionable information? The Common Vulnerability Scoring System (CVSS) is an open framework that addresses this issue."* [13]

However, by the time CVSS 3.1 was released, the framework's focus had shifted, partly due to complaints about CVSS being a poor judge of risk. The authors stated:

> *"CVSS measures severity, not risk."* [25]

The identity crisis is a problem because the original stance, that it can be used as a primary prioritisation tool, has lured parts of the industry into doing just that. As mentioned by Henry Howland in [12], there are many large, mainly US based places mandating the sole use of CVSS base score for remediation. Payment Card Industry Data Security Standard [2], the Department of Defense Joint Special Access Program implementation

Guide [8] to name a few.

This change in stance has created confusion about the true purpose of CVSS.

6.1.1 CVSS Formula

How CVSS is computed under-the-hood is confusing at best. CVSS 3.1 is not explained to the same depth as version 2.0, but my understanding is that it followed a similar process. This is that process summarised from CVSS version 2 FAQ: [10]

1. Divide the six metrics into two groups:
    - **Impact** (3 metrics)
    - **Exploitability** (3 metrics)
2. Create sub-vectors for each group:
    - **Impact sub-vector**: 27 possible values (3^3)
    - **Exploitability sub-vector**: 26 possible values ($3^3 - 1$ for no impact)
3. Develop and apply a series of rules to order the sub-vectors. These rules are primarily based on the severity of components, e.g. vectors with more `Complete` components are rated higher.
4. Assign scores to the ordered sub-vectors based on the derived rules and review by the Special Interest Group (SIG).
5. Apply weightings to the sub-vectors:
    - **Impact**: 0.6
    - **Exploitability**: 0.4
6. Develop a formula that approximates the scores derived from the ordered sub-vectors. Ensure the formula produces scores with ± 0.5 error from the originally defined vector score and does not exceed the maximum value of 10.

7. Test and refine the formula through iterations, ensuring it aligns with desired values and corrects any issues, such as scores exceeding 10.

This is process is inherently inaccurate, it is not a system designed to give precise scores. If we look at 6 above, the formula (Appendix .2 shows the CVSS 3.1 base score formula for reference) which produces the score, does not match exactly the experts decision. There is a lot of rounding and approximation going on. This is designed to make a system which is easy to use and quick to complete by security professionals. There is a space for CVSS, however this along with the other mentioned reasons outlines the issues with using CVSS as a sole metric for prioritization. Perhaps an option is to triage the large swathes of new vulnerabilites coming in with an initial CVSS score, then move on to a deeper dive. This could be in the form of the extra CVSS metrics (Temporal score & Environmental score), or a look into other potential options like the Exploit Prediction Scoring System (EPSS [15]).

6.1.2 Exploit Prediction Scoring System

EPSS is developed by FIRST, the same group who govern the CVSS standard. It has a different take on the problem, focusing on a data driven model designed to give "a daily estimate of the probability of exploitation activity being observed over the next 30 days [15]." If the data shown on the model page is to be believed, it is a promising system (some of their findings Figure 10, Figure 9). Unfortunately, while good to keep in mind for the industry, it is less useful for our purposes. This is a pretrained and uninterpretable model, from the outside at least. Analysis could be done on the output of the model in relation to CVEs, but that will not be a focus going forward.

6.2 Future Work

Despite the disadvantages of CVSS, I will continue to focus on it in my studies. My focus will pivot towards the interpretability of large language models, particularly from the perspective of data. The CVE dataset is messy, with many poorly written CVE descriptions that are not useful for machine learning models. I plan to clean up this dataset by identifying and removing low-quality entries. To achieve this, I will perform clustering and general analysis of the CVE descriptions to better understand the dataset and develop heuristics for filtering it.

7 Conclusion

The Common Vulnerability Scoring System (CVSS) is integral in prioritizing and managing the ever-growing number of software vulnerabilities. While the current approach heavily relies on manually assigned scores from the National Vulnerability Database (NVD), it may be worth using the MITRE database as well. However, the findings are very much inconclusive, it is possible that adding the extra data during model training is worth it, but so far it has only been to the detriment of performance. One key issue is the variability in CVSS scores between different databases, such as NVD and MITRE, which suggests a lack of consistency in the scoring process. This inconsistency can confuse automated systems and reduce the overall reliability of the predicted scores.

In conclusion, while CVSS remains a crucial tool for vulnerability management, its current implementation has limitations that need to be addressed. The integration of machine learning models offers a promising solution to automate and enhance the accuracy of CVSS scoring. However the focus should be on predicting the distinct categorical variable for each metric, as this is a universally interesting classification task and will be more able to apply to changes in the CVSS standard. Future research should focus on refining these models, and focusing on interpretability, whether through LLM interpretability methods, or through more traditional databased clustering.

References

[1] M Ugur Aksu et al. "Automated generation of attack graphs using NVD". In: *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*. 2018, pp. 135–142.

[2] Ellen Richey et al. *Payment card industry (PCI) data security standard.* https://www.pcisecuritystandards.org/document_library?category=pcidss&document=dss4aag. [Online; accessed July-2024]. 2018.

[3] Anonymous. *CVSS v3 and v3.1 missing temporal metrics exploit code maturity and remediation.* https://security.stackexchange.com/questions/270257/cvss-v3-and-v3-1-missing-temporal-metrics-exploit-code-maturity-and-remediation. [Online; accessed July-2024]. 2024.

[4] Hodaya Binyamini et al. “A framework for modeling cyber attack techniques from security vulnerability descriptions”. In: *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 2021, pp. 2574–2583.

[5] CISA. *Known Exploited Vulnerabilities Catalog*. https://www.cisa.gov/known-exploited-vulnerabilities-catalog. [Online; accessed June-2024]. 2024.

[6] The MITRE Corporation. *List of Partners*. https://www.cve.org/PartnerInformation/ListofPartners. [Online; accessed June-2024]. 2024.

[7] Joana Cabral Costa et al. “Predicting CVSS Metric via Description Interpretation”. In: *IEEE Access* 10 (2022), pp. 59125–59134. DOI: 10.1109/ACCESS.2022.3179692.

[8] Kenneth Brown David Beèn. *Department of Defense (DOD) Joint Special Access Program (SAP) Implementation Guide (JSIG)*. https://www.dcsa.mil/portals/91/documents/ctp/nao/JSIG_2016April11_Final_(53Rev4).pdf. [Online; accessed July-2024]. 2016.

[9] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL]. URL: https://arxiv.org/abs/1810.04805.

[10] FIRST. *CVSS Frequently Asked Questions*. https://www.first.org/cvss/v2/faq#Explanation-of-CVSS-v2-formula-and-metric-valued-development. [Online; accessed July-2024]. 2007.

[11] Andrew Gelman. *Hierarchical modeling when you have only 2 groups: I still think it's a good idea, you just need an informative prior on the group-level variation*. https://statmodeling.stat.columbia.edu/2015/12/08/hierarchical-modeling-when-you-have-only-2-groups-i-still-think-its-a-good-idea-you-just-need-an-informative-prior-on-the-group-level-variation/. [Online; accessed July-2024]. 2024.

[12] Henry Howland. “CVSS: Ubiquitous and Broken”. In: *Digital Threats* 4.1 (Feb. 2022). DOI: 10.1145/3491263. URL: https://doi.org/10.1145/3491263.

[13] Forum of Incident Response and Security Teams (FIRST). *A Complete Guide to the Common Vulnerability Scoring System*. https://www.first.org/cvss/v2/guide. [Online; accessed June-2024]. 2024.

[14] Forum of Incident Response and Security Teams (FIRST). *CVSS landing page*. https://www.first.org/cvss/. [Online; accessed February-2024]. 2024.

[15] Forum of Incident Response and Security Teams (FIRST). *The EPSS Model*. https://www.first.org/epss/model. [Online; accessed June-2024]. 2024.

[16] Forum of Incident Response and Security Teams (FIRST). *The EPSS User Guide*. https://www.first.org/epss/user-guide. [Online; accessed June-2024]. 2024.

[17] Yuning Jiang and Yacine Atif. "An Approach to Discover and Assess Vulnerability Severity Automatically in Cyber-Physical Systems". In: *13th International Conference on Security of Information and Networks*. SIN 2020: 13th International Conference on Security of Information and Networks. Merkez Turkey: ACM, Nov. 4, 2020, pp. 1–8. ISBN: 978-1-4503-8751-4. DOI: 10.1145/3433174.3433612. URL: https://dl.acm.org/doi/10.1145/3433174.3433612 (visited on 02/28/2024).

[18] Pontus Johnson et al. "Can the Common Vulnerability Scoring System be Trusted? A Bayesian Analysis". In: *IEEE Transactions on Dependable and Secure Computing* 15.6 (2018), pp. 1002–1015. DOI: 10.1109/TDSC.2016.2644614.

[19] MITRE. *Common Vulnerabilities and Exposures — CVE® The Standard for Information Security Vulnerability Names*. https://cve.mitre.org/docs/cve-intro-handout.pdf. [Online; accessed July-2024]. 2024.

[20] MITRE. *CVE Numbering Authorities (CNAs)*. https://www.cve.org/ProgramOrganization/CNAs. [Online; accessed May-2024]. 2024.

[21] MITRE. *MITRE landing page*. https://cve.mitre.org/. [Online; accessed February-2024]. 2024.

[22] NVD. *CVE-2020-19909 Detail*. https://nvd.nist.gov/vuln/detail/CVE-2020-19909. [Online; accessed July-2024]. 2023.

[23] NVD. *CVE-2024-38526 Detail*. https://nvd.nist.gov/vuln/detail/CVE-2024-38526. [Online; accessed June-2024]. 2024.

[24] NVD. *NVD landing page*. https://nvd.nist.gov/. [Online; accessed February-2024]. 2024.

[25] Sasha Romanosky Peter Mell Karen Scarfone. *Common Vulnerability Scoring System v3.1: Specification Document*. https://www.first.org/cvss/v3.1/specification-document. [Online; accessed February-2024]. 2024.

[26] Martyn Plummer. *JAGS: Just Another Gibbs Sampler*. https://sourceforge.net/projects/mcmc-jags/. [Online; accessed April-2024]. 2024.

[27] PyMC. *PyMC landing page*. https://www.pymc.io/welcome.html. [Online; accessed May-2024]. 2024.

[28] Qualys. *CVSS Vector Strings*. https://qualysguard.qualys.com/qwebhelp/fo_portal/setup/cvss_vector_strings.htm. [Online; accessed July-2024]. 2024.

[29] Victor Sanh et al. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. 2020. arXiv: 1910.01108 [cs.CL]. URL: https://arxiv.org/abs/1910.01108.

[30] Jonathan Spring et al. "Time to Change the CVSS?" In: *IEEE Security & Privacy* 19.2 (2021), pp. 74–78. DOI: 10.1109/MSEC.2020.3044475.

[31] Jonathan Spring et al. "Towards improving CVSS". In: *SEI, CMU, Tech. Rep* (2018).

[32] Daniel Stenbeck. *BOGUS CVE FOLLOW-UPS*. https://daniel.haxx.se/blog/2023/09/05/bogus-cve-follow-ups/. [Online; accessed July-2024]. 2023.

[33] Daniel Stenbeck. *CVE-2020-19909 IS EVERYTHING THAT IS WRONG WITH CVES*. https://daniel.haxx.se/blog/2023/08/26/cve-2020-19909-is-everything-that-is-wrong-with-cves/. [Online; accessed July-2024]. 2023.

[34] Daniel Stenberg. *Curl landing page*. https://curl.se/. [Online; accessed July-2024]. 2023.

[35] Michael Nowotny Tomasz Miasko. *PyJAGS: The Python Interface to JAGS*. https://github.com/michaelnowotny/pyjags. [Online; accessed April-2024]. 2024.

Appendix .1 CVSS figures from other versions

Below is a collection of confusion matrics which are results from the Bayesian analysis of CVSS versions 2.0 & 3.0 for both the NVD and MITRE databases. Version 2.0 for MITRE is especially rough as there was very little data.

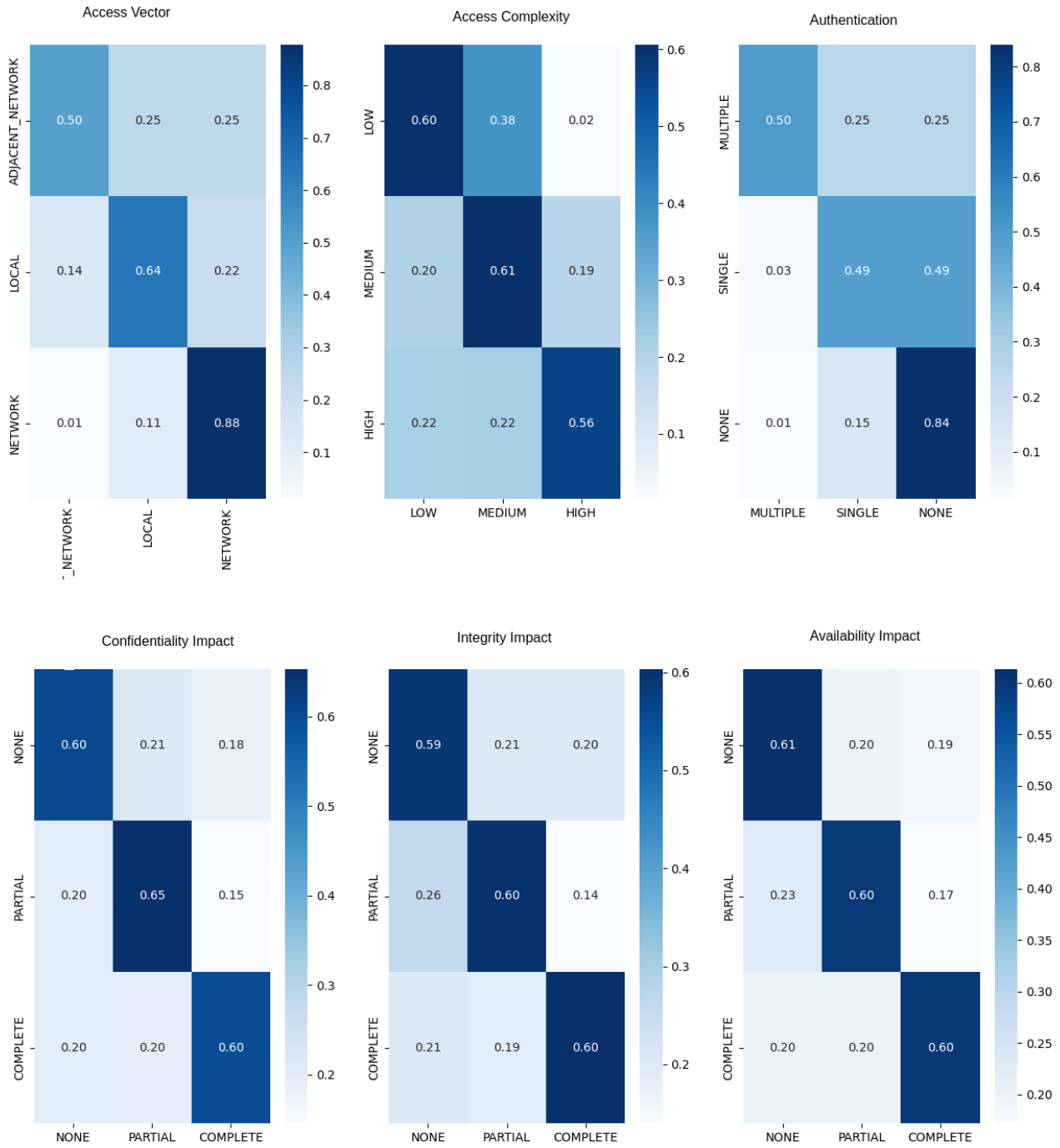

Figure 5: Confusion matrix of estimated accuracy for CVSS metrics for version 2.0 for NVD

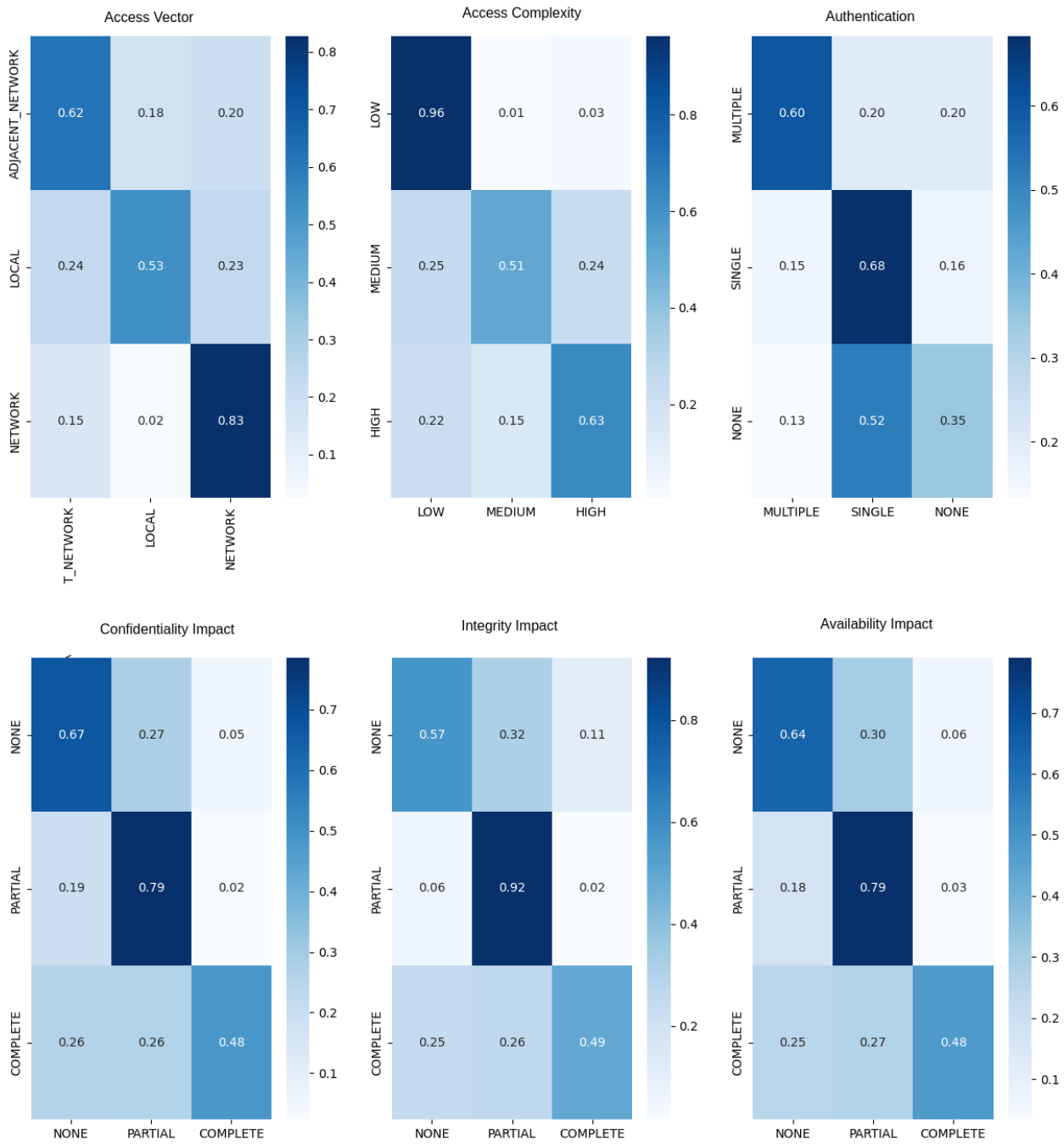

Figure 6: Confusion matrix of estimated accuracy for CVSS metrics for version 2.0 for MITRE

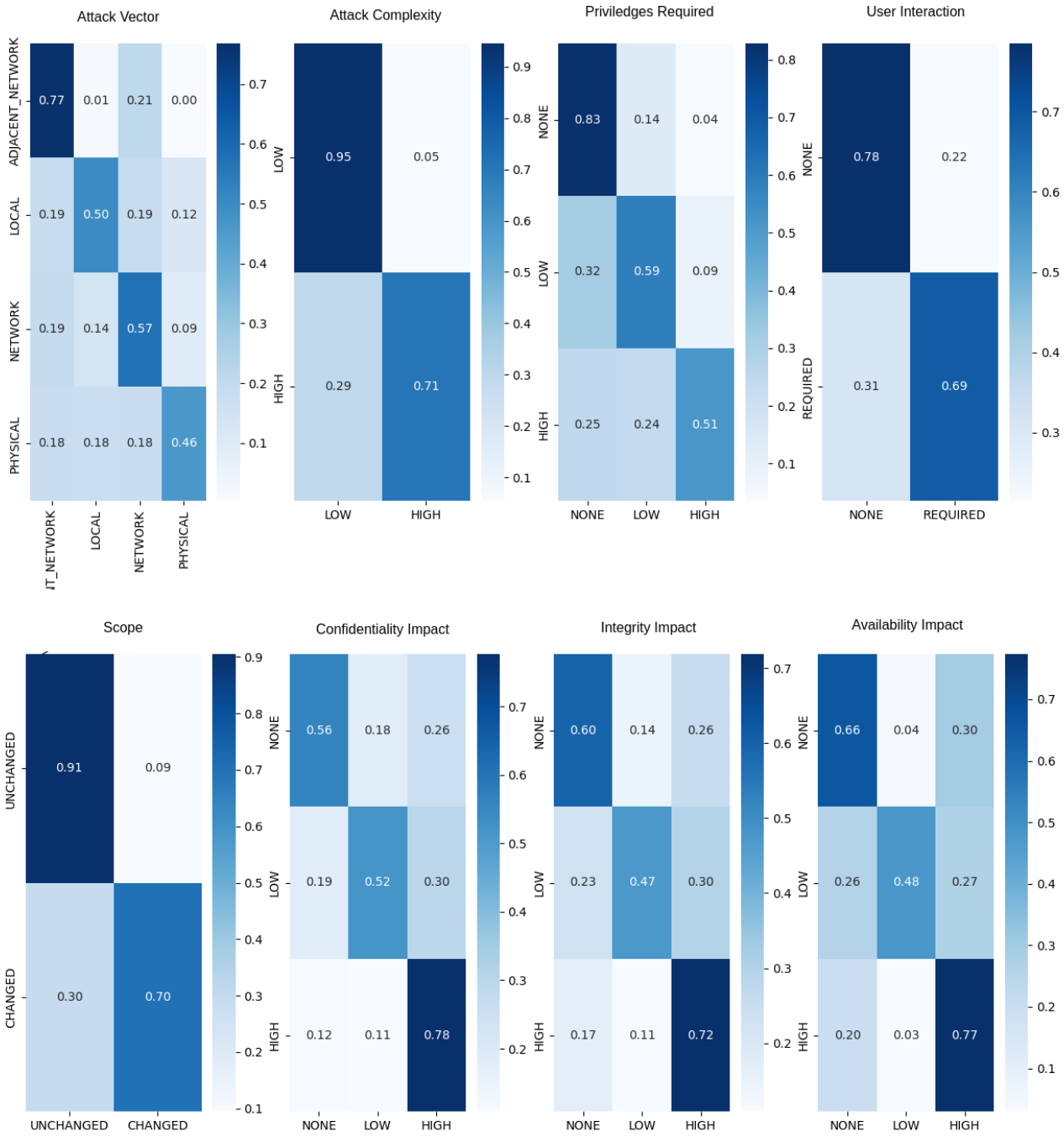

Figure 7: Confusion matrix of estimated accuracy for CVSS metrics for version 3.0 for NVD

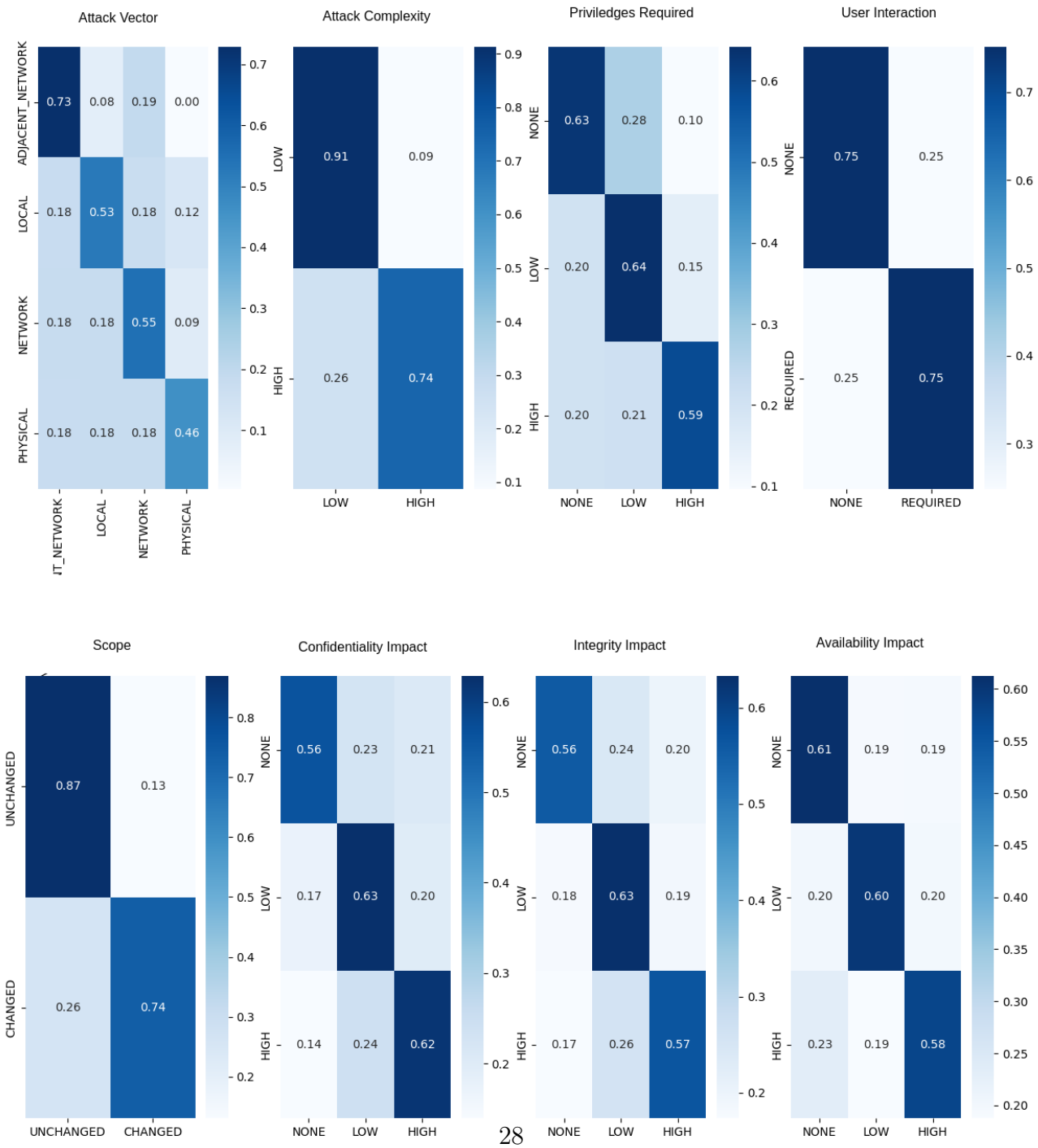

Figure 8: Confusion matrix of estimated accuracy for CVSS metrics for version 3.0 for MITRE

Appendix .2 CVSS 3.1 Base Score formula

Below is a formulaic representation of the CVSS 3.1 base score formula.

$$ISS = 1 - ((1 - Confidentiality) \times (1 - Integrity) \times (1 - Availability)) \tag{2}$$

$$Impact = \begin{cases} 7.52 \times (ISS - 0.029) - 3.25 \times (ISS - 0.02)^{15} & \text{if } Scope \text{ is } Changed \\ 6.42 \times ISS & \text{if } Scope \text{ is } Unchanged \end{cases} \tag{3}$$

$$Exploitability = 8.22 - AttackVector \times AttackComplexity \times PrivilegesRequired \times UserInteraction \tag{4}$$

$$BaseScore = \begin{cases} 0 & Impact \leq 0 \\ Roundup(Minimum(1.08 \times (Impact + Exploitability), 10)) & \text{if } Scope \text{ is } Changed \\ Roundup(Minimum((Impact + Exploitability), 0)) & \text{if } Scope \text{ is } Unchanged \end{cases} \tag{5}$$

Appendix .3 Exploit Prediction Scoring System diagrams for reference

Below are two graphs from FIRST, the creator of EPSS. These show some of the results they have found for this system, especially when comparing EPSS to CVSS in terms of effeciency of effort if you use EPSS to guide remediation of vulnerabilities.

Appendix A Aims and Objectives

Original

Aims The primary aim of this research is to develop sophisticated predictive models capable of accurately determining the severity levels of security threats based on the CVSS. This will involve a comprehensive review and comparison of current datasets, with a focus on leveraging natural language descriptions provided in security vulnerability reports. The project intends to utilize advanced transformer-based models to achieve this goal, contributing to the field of cybersecurity by enhancing the precision of threat severity assessments.

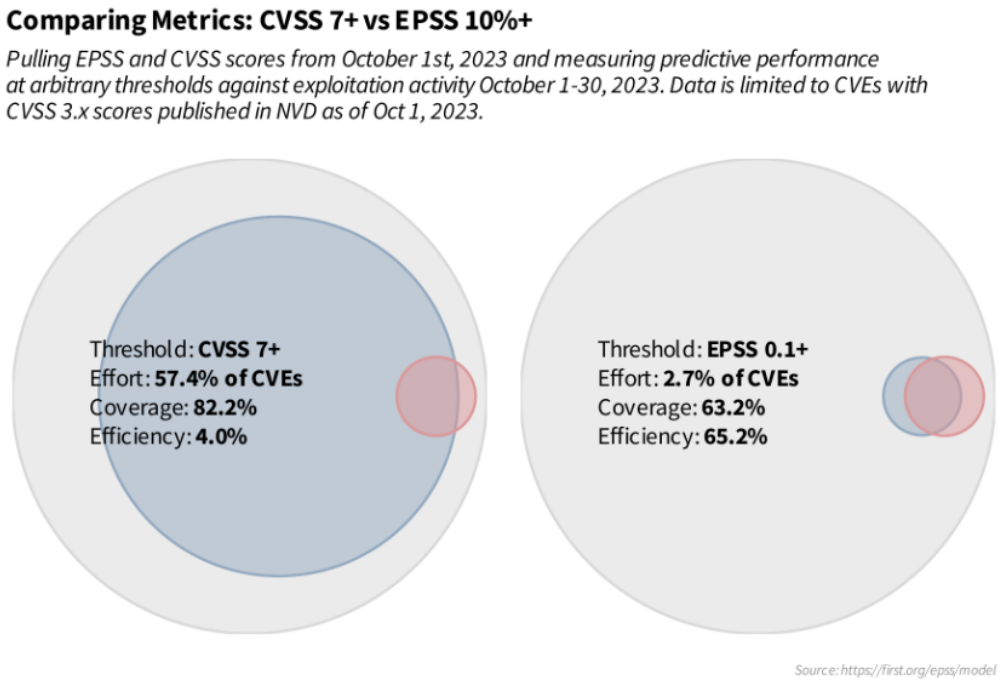

Figure 9: Comparing Metrics: CVSS 7+ vs. EPSS 10%+ sourced from [15]

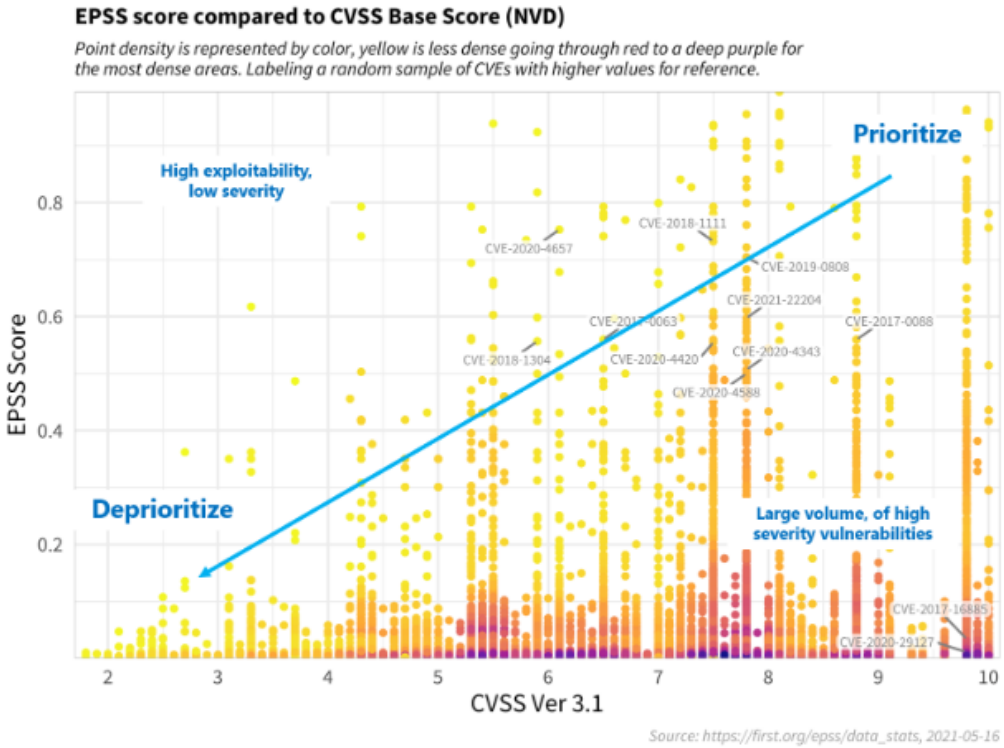

Figure 10: EPSS score compared to CVSS Base Score (NVD) sourced from [16]

Objectives

- Conduct a comprehensive literature review to understand the current landscape of CVSS score prediction and the methodologies employed in existing models.
- Replicate successful methodologies to verify the accuracy of CVSS score databases, with a particular focus on alignment with recent CVSS standards and datasets.
- Explore opportunities for enhancing existing methodologies, including the investigation of data amalgamation from multiple databases to ascertain improvements in model performance.
- Experiment with various model architectures to identify the most effective approach in terms of predictive accuracy, specifically focusing on metrics such as the F1 score and balanced accuracy.

Timeline

- March: Initiate the project with a literature review, system environment setup, and resource gathering.
- March-April: Replicate existing methodologies to validate findings and ensure alignment with current standards.
- May-June: Generate preliminary results and compile an interim report detailing findings and methodologies.
- July-August: Conduct experiments with various data source combinations and model architectures to identify optimal configurations.
- September-October: Finalize experimental work, analyze results, and prepare the comprehensive final report.

Revised

Aims The primary aim of this research is to develop sophisticated predictive models capable of accurately determining the severity levels of security threats based on the CVSS. This will involve a comprehensive review and comparison of current datasets, with a focus on leveraging natural language descriptions provided in security vulnerability reports. The project intends to utilize advanced transformer-based models to achieve this goal, contributing to the field of cybersecurity by enhancing the precision of threat severity assessments.

Objectives

- Conduct a comprehensive literature review to understand the current landscape of CVSS score prediction and the methodologies employed in existing models.
- Replicate successful methodologies to verify the accuracy of CVSS score databases, with a particular focus on alignment with recent CVSS standards and datasets.
- Explore opportunities for enhancing existing methodologies, including the investigation of data amalgamation from multiple databases to ascertain improvements in model performance.
- Look into data cleaning and clustering, to improve the efficacy of the models, as well as a look into interpretability though data analysis.