

# **AI-Driven Vulnerability Prediction and Mitigation in Datacenter Environment**



**By**

**Aaliya Ali**

**(00000397850)**

**Supervised by**

**Cdre. Dr. Nadeem Kureshi**

**Department of Cyber Security**

**Pakistan Navy Engineering College (PNEC)**

**National University of Sciences & Technology (NUST)**

**Islamabad, Pakistan**

**(2024)**

**AI-Driven Vulnerability Prediction and Mitigation in  
Datacenter Environment**



**By**

**Aaliya Ali**

**(Registration No: 00000397850)**

**A Thesis Submitted to the National University of Sciences and Technology,  
Islamabad, in partial fulfillment of the requirements for the degree of**

**MASTER OF SCIENCE in  
CYBER SECURITY**

**Supervisor: Cdre. Dr.Nadeem Kureshi**

**Pakistan Navy Engineering College (PNEC)**

**National University of Sciences & Technology (NUST)**

**Islamabad, Pakistan**

**(2024)**

### Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at Department of Cyber Security at Pakistan Navy Engineering College (PNEC) or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at Pakistan Navy Engineering College (PNEC) or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Name of Student: Aaliya Ali

Date: 26 Jan 2024


**National University of Sciences and Technology**

**MASTER'S THESIS WORK**

We hereby recommend that the dissertation prepared under our supervision by LT Cdr Aaliya Ali PN (00000397850) Titled: AI- Driven Vulnerability Prediction and mitigation in Datacenter Environmet be accepted in partial fulfillment of the requirements for the award of Master's degree.


**EXAMINATION COMMITTEE MEMBERS**

1. Name: Dr. Ayaz Sherazi Signature: 

2. Name: Dr. Muhammad Usama Signature: 

Supervisor's name: Cdre Dr Nadeem Kureshi Signature: 

Date:   
**DR NADEEM KURESHI**  
Commodore  
DEAN MIS  
PNS JAUHAR


  
Head of Department

26-01-2024  
Date

**DR NADEEM KURESHI**  
Commodore  
DEAN MIS  
PNS JAUHAR

**COUNTERSIGNED**


Date: 26-01-2024


  
Dean / Principal


**DR NADEEM KURESHI**  
Commodore  
DEAN MIS  
PNS JAUHAR

### Thesis Acceptance Certificate

Certified that final copy of MS/MPhil thesis entitled "AI-Driven Vulnerability prediction and mitigation in Datacenter Environment" written by Aaliya Ali, (Registration No 00000397850), of Pakistan Navy Engineering College (PNEC) has been vetted by the undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/M Phil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Signature: \_\_\_\_\_   
Name of Advisor: Cdre. Dr. Nadeem Kureshi  
Date: 26-01-2024

Signature (HOD): \_\_\_\_\_   
Date: 26-01-24  
DR NADEEM KURESHI  
Commodore  
DEAN MIS  
PNS JAUMAP

Signature (Dean): \_\_\_\_\_   
Date: 26-01-24  
DR NADEEM KURESHI  
Commodore  
DEAN MIS  
PNS JAUMAP

**CERTIFICATE OF APPROVAL**

It is certified that the contents and form of the thesis entitled "AI-Driven Vulnerability prediction and mitigation in Datacenter Environment" submitted by Aaliya have been found satisfactory for the requirement of the degree.

Advisor: **Cdre. Dr. Nadeem Kureshi**

Signature: \_\_\_\_\_ 

Date: 26 - 01 - 2024

Committee Member 1: **Asst. Prof. Dr. M Usama**

Signature: \_\_\_\_\_ 

Date: 26/1/24

Committee Member 2: **Asst. Prof. Dr. Avaz Sherazi**

Signature: \_\_\_\_\_ 


Date: 26 - 01 - 24

## CERTIFICATE FOR PLAGIARISM

1. It is certified that PhD / M.Phil / MS Thesis Titled "AI- Driven Vulnerability Prediction and mitigation in Datacenter Environment" by Lt Cdr Aaliya Ali PN (2021-NUST-MS Cyber Security (CyS Fall 21)) has been examined by us. We undertake the follows:

- a. Thesis has significant new work / knowledge as compared already published or is under consideration to be published elsewhere. No sentence, equation, diagram, table, paragraph or section has been copied verbatim from previous work unless it is placed under quotation marks and duly referenced.
- b. The work presented is original and own work of the author (i.e. there is no plagiarism). No ideas, processes, results or words of others have been presented as Author own work.
- c. There is no fabrication of data or results which have been compiled / analyzed.
- d. There is no falsification by manipulating research materials, equipment, or processes, or changing or omitting data or results such that the research is not accurately represented in the research record.
- e. The thesis has been checked using TURNITIN (copy of originality report attached) and found within limits as per HEC Plagiarism Policy and instructions issued from time to time.

### Name & Signature of Supervisor



DR NADEEM KURESHI  
Commodore  
DEAN MIS  
PNS JAUMAR

*To the guiding lights of my life,*

*To my late father, whose unwavering support and wisdom shaped the foundation of my dreams. Your memory fuels my determination, and I dedicate this thesis to the lessons you imparted and the love that continues to inspire me. To my resilient mother, whose sacrifices and love are the foundation of my strength. To my siblings, the source of shared laughter and enduring bonds. To my precious children, your joy fuels my purpose. To my husband, your unwavering support is my anchor in life's journey. To my teachers, your guidance shaped my intellect. To my supervisor, your mentorship was invaluable. This thesis is dedicated to each of you—my pillars of love, support, and inspiration. I am grateful for the impact you've had on my life and academic journey.*



## ACKNOWLEDGEMENTS

I extend my deepest gratitude to Cdre Dr. Nadeem Kureshi, not only for his role as my esteemed supervisor but also as the Dean of the department. His guidance, support, and unwavering commitment to academic excellence have been instrumental in shaping this thesis. I would also like to express my sincere appreciation to the members of the GEC, Dr. Ayaz Sherazi and Dr. Muhammad Usama, for their valuable insights, constructive feedback, and dedication to fostering a rigorous academic environment. To my beloved children, whose understanding, patience, and encouragement sustained me throughout this academic journey. Your presence added a dimension of joy and purpose to each step of the way. To my husband, whose unwavering support and belief in my abilities were my constant motivation. Your encouragement was a driving force behind the completion of this thesis. To my siblings, thank you for being a source of inspiration and a pillar of support during both challenging and triumphant moments. Each of you played a pivotal role in this academic endeavor, and I am profoundly grateful for your contributions, encouragement, and belief in my capabilities. This achievement reflects our collective efforts and the strength derived from your support.

# TABLE OF CONTENTS

<b>ACKNOWLEDGEMENTS</b>	<b>xi</b>
<b>TABLE OF CONTENTS</b>	<b>viii</b>
<b>LIST OF TABLES</b>	<b>x</b>
<b>LIST OF FIGURES</b>	<b>xi</b>
<b>LIST OF SYMBOLS, ABBREVIATIONS AND ACRONYMS</b>	<b>XII</b>
<b>ABSTRACT</b>	<b>xiv</b>
<b>CHAPTER 1 : INTRODUCTION</b>	<b>1</b>
<b>1.1 Background</b>	<b>1</b>
<b>1.2 Research Objective</b>	<b>4</b>
<b>1.3 Contribution</b>	<b>5</b>
<b>1.4 Structure</b>	<b>5</b>
<b>CHAPTER 2 : LITERATURE REVIEW</b>	<b>7</b>
<b>2.1 Related Work</b>	<b>7</b>
<b>2.1.1 Data Center and Vulnerabilities</b>	<b>7</b>
<b>2.1.2 Diverse Approaches</b>	<b>8</b>
<b>2.1.3 Correlated Studies</b>	<b>9</b>
<b>2.2 Research Gap</b>	<b>11</b>
<b>2.3 Data Repositories</b>	<b>11</b>
<b>2.3.1 Common Vulnerability Enumeration</b>	<b>11</b>
<b>2.3.2 National Vulnerability Database</b>	<b>12</b>
<b>2.3.3 Other Sources</b>	<b>14</b>
<b>CHAPTER 3 : METHODOLOGY</b>	<b>16</b>
<b>3.1 Introduction</b>	<b>16</b>
<b>3.2 Framework Planned</b>	<b>16</b>
<b>3.3 Vulnerability Severity Metrics</b>	<b>18</b>
<b>3.3.1 Exploitability Metrics</b>	<b>18</b>
<b>3.3.2 Scope (S)</b>	<b>20</b>
<b>3.3.3 Impact Metrics</b>	<b>21</b>
<b>3.4 Vulnerability Computation</b>	<b>21</b>
<b>3.4.1 Version 2</b>	<b>21</b>
<b>3.4.2 Version 3</b>	<b>23</b>
<b>3.5 Workflow for proposed Framework</b>	<b>24</b>
<b>3.5.1 Data Extraction</b>	<b>24</b>
<b>3.5.2 Data Set Preparation</b>	<b>26</b>
<b>3.5.3 Data Cleaning</b>	<b>28</b>
<b>3.5.4 Arithmetic Mean (AM)</b>	<b>30</b>
<b>3.5.5 Vulnerability Severity Calculation</b>	<b>30</b>
<b>3.5.6 Evaluation</b>	<b>30</b>
<b>3.5.7 Validation</b>	<b>31</b>
<b>3.5.8 Feature Extraction</b>	<b>32</b>
<b>3.5.9 ML Model Application</b>	<b>32</b>
<b>3.5.10 Mathematical Modelling</b>	<b>33</b>

3.5.11 Library Used	35
3.6 Algorithms	39
3.6.1 Algorithm 1: Calculation of Base Score CVSS V3	40
3.6.2 Algorithm 2 Generation of Data Product wise from CVE details	41
3.6.3 Algorithm for Impact Calculation	42
3.6.4 Algorithm for Base Score Calculation CVSS V2	43
3.6.5 Algorithm to Calculate Arithmetic Mean	44
CHAPTER 4 : EXPERIMENTS AND RESULTS	45
4.1 Evaluation Metrics	45
4.1.1 Accuracy	45
4.1.2 Precision	46
4.1.3 Recall	46
4.1.4 F1 Score Balanced	46
4.1.5 Support	48
4.1.6 Reciprocal Rank and Mean Reciprocal Rank	48
4.2 Results of Random Forest Model	49
4.2.1 RF Classifier specification	49
4.2.2 Data Breakdown	50
4.2.3 Results and Comparison	59
4.2.4 Performance Study	66
4.3 Analysis	79
CHAPTER 5 : RECOMMENDATIONS FOR MITIGATION	81
5.1 Known Exploited Vulnerabilities (KEV)	81
5.2 Analysis for Patterns	82
5.3 Recommendations	87
CHAPTER 6 : CONCLUSION AND FUTURE WORKS	88
REFERENCES	90

# LIST OF TABLES

Table 3.1: Dataset features for each CVE record CVSS V2.....	26
Table 3.2: Dataset features for each CVE record – CVSS V3.....	27
Table 4.1: RF parameters .....	49
Table 4.2: Comparison Table LR & RF for CVSS V2 NVD Only .....	59
Table 4.3: Comparison Table LR & RF for CVSS V3 NVD Only .....	60
Table 4.4: Comparison Table RF with different parameters for CVSS V2 NVD Only ...	61
Table 4.5: Comparison Table RF with different parameters for CVSS V3 NVD Only ...	62
Table 4.6: Comparison Table RF with different parameters for CVSS V2 DC Data.....	63
Table 4.7: Comparison Table RF with different parameters for CVSS V3 DC Data.....	64
Table 4.8: RF for CVSS V2 DC correlated data.....	65
Table 4.9: RF for CVSS V3 DC correlated data.....	65
Table 4.10: Evaluation metrics for pure NVD dataset CVSS V2.....	66
Table 4.11: Evaluation metrics for pure NVD dataset CVSS V3.....	67
Table 4.12: Evaluation metrics for DC dataset CVSS V2.....	67
Table 4.13: Evaluation metrics for pure DC dataset CVSS V3.....	68
Table 4.14: Calculation on Evaluation Metrics for pure NVD dataset CVSS V2.....	69
Table 4.15: Calculation on Evaluation Metrics for pure NVD dataset CVSS V3.....	69
Table 4.16: Calculation on Evaluation Metrics for pure DC dataset CVSS V2.....	70
Table 4.17: Calculation on Evaluation Metrics for DC dataset CVSS V3 .....	70
Table 5.1: CISA catalog format .....	82
Table 5.2: Occurrences CVSS score range-wise .....	84
Table 5.3: Priority of Vulnerability mitigation .....	87

# LIST OF FIGURES

Figure 1.1: Basic calculation cycle of CVSS.....	3
Figure 2.1: CVE complete process .....	13
Figure 3.1: Detailed representation of CVSS score calculation with NVD.....	17
Figure 3.2: Broad representation of CVSS score calculation with NVD .....	19
Figure 3.3: DC data extraction cycle .....	25
Figure 3.4: Complete Data generation cycle.....	29
Figure 4.1: Count wise vulnerabilities of CVSS V2.....	50
Figure 4.2: Count wise vulnerabilities of CVSS V3.....	51
Figure 4.3: Count wise vulnerabilities of DC products .....	51
Figure 4.4: Count wise vulnerabilities of DC products .....	52
Figure 4.5: Percentage of different classes data in Access Vector – CVSS V2 .....	52
Figure 4.6: Percentage of different classes data in Access Complexity– CVSS V2 .....	53
Figure 4.7: Percentage of different classes data in Authentication – CVSS V2.....	53
Figure 4.8: Percentage of different classes data in Confidentiality Impact – CVSS V2 ..	54
Figure 4.9: Percentage of different classes data in Integrity Impact – CVSS V2.....	54
Figure 4.10: Percentage of different classes data in Availability Impact – CVSS V2 .....	55
Figure 4.11: Percentage of different classes data in Attack Vector – CVSS V3.....	55
Figure 4.12: Percentage of different classes data in Attack Complexity – CVSS V3.....	56
Figure 4.13: Percentage of different classes data in Privileges Required – CVSS V3.....	56
Figure 4.14: Percentage of different classes data in User Interaction – CVSS V3.....	57
Figure 4.15: Percentage of different classes data in Scope – CVSS V3.....	57
Figure 4.16: Percentage of different classes data in Confidentiality Impact – CVSS V3	58
Figure 4.17: Percentage of different classes data in Integrity – CVSS V3.....	58
Figure 4.18: Percentage of different classes data in Availability Impact – CVSS V3 .....	59
Figure 4.19: Confusion matrix for Access Vector – CVSS V2.....	71
Figure 4.20: Confusion matrix for Access Complexity – CVSS V2 .....	72
Figure 4.21: Confusion matrix for Authentication – CVSS V2.....	72
Figure 4.22: Confusion matrix for Confidentiality – CVSS V2.....	73
Figure 4.23: Confusion matrix for Integrity – CVSS V2.....	73
Figure 4.24: Confusion matrix for Availability – CVSS V2 .....	74
Figure 4.25: Confusion matrix for Attack Vector – CVSS V3.....	75
Figure 4.26: Confusion matrix for Attack Complexity – CVSS V3.....	76
Figure 4.27: Confusion matrix for User Interaction – CVSS V3 .....	76
Figure 4.28: Confusion matrix for Privileges Required – CVSS V3.....	77
Figure 4.29: Confusion matrix for Scope – CVSS V3.....	77
Figure 4.30: Confusion matrix for Confidentiality Impact – CVSS V3.....	78
Figure 4.31: Confusion matrix for Integrity Impact – CVSS V3 .....	78
Figure 4.32: Confusion matrix for Availability Impact – CVSS V3.....	79
Figure 5.1: Year wise occurrences of CVSS V2 vulnerabilities .....	83
Figure 5.2: Year wise occurrences of CVSS V3 vulnerabilities .....	83
Figure 5.3: KEV entries CVSS score wise V2 .....	85
Figure 5.4: KEV entries CVSS score wise V3 .....	85
Figure 5.5: KEV entries CVSS score category wise V2.....	86
Figure 5.6: KEV entries CVSS score category wise V3.....	86

## **LIST OF SYMBOLS, ABBREVIATIONS AND ACRONYMS**

AC	Attack Complexity / Access Complexity
AI	Artificial Intelligence
API	Application Programming Interface
AV	Attack Vector / Access Vector
CIA	Confidentiality Integrity Availability
CISA	Cyber Security and Infrastructure Security Agency
CVE	Common Vulnerability Enumeration
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
FIRST	Forum of Incident Response and Security Teams
KEV	Known Exploited Vulnerabilities
ML	Machine Learning
MRR	Mean Reciprocal Rank
NLP	Natural Language Processing
NVD	National Vulnerability Database
NIST	National Institute of Standards and Technology
RF	Random Forest
RR	Reciprocal Rank

S Scope

TF-IDF Term frequency – Inverse Document Frequency

## **ABSTRACT**

In this era of digitization where everything is shifted to computer systems, Data and Information are the key assets of any organization. To maintain huge amount of data, Data center (DC) is considered primary resource. However, as much as we rely on DC for storage of data and operations, it is at risk of vulnerability exploitation and attacks. Vulnerability assessment is difficult task because different procedures are employed to assess different levels of cyber-security weaknesses. There are various repositories maintaining database of vulnerabilities but one can see delays and inconsistency in the severity scores calculated by them. In this research, effort is put to assess role of ML in looking at patterns in the reported vulnerabilities and predicting CVSS scores. The error rate caused by the manual calculation procedures that are often employed in cybersecurity analysis is reduced by this method. Further, a method is formulated and applied to a case study on DC, where focus is on resolution of disparity between various scores provided by various sources including NVD, Vulner and VulDb with the help of CVE Details API for the retrieval of pertinent DC related equipment records and applied Arithmetic mean scoring system along ML model. To validate the suggested method, NVD data base is referred and CVSSv2 and CVSSv3 scores are calculated and increased accuracy is gained. In addition to that, CISA KEV catalog is analyzed and priority preferences to mitigate the vulnerabilities are provided to help in performing risk estimation.

**Keywords:** Artificial Intelligence, Data Center, CISA, CVE, CVSS, KEV, Machine Learning, NVD, Prediction.



# INTRODUCTION

## 1.1 Background

Development is moving at an unprecedented rate right now. Almost every significant aspect of our routine involves the use of computer systems, including everyday tasks like paying bills, doing vacations, purchasing, sharing documents, educating others or self, talking and video conferencing [1].

All these operations are carried out by using a large amount of data and information and data center is a home for computer systems where organizations keep every type of information and data. In addition to that, a data center manages all network workflow and data transmission. With the current state of technology, many organizations have come to the realization that to provide real-time, fast, and uninterrupted data transfer, they must fully set up their data center possible [2].

It is necessary that the information must always be available to an organization, regardless of the issues it faces. This has an impact on the assumption that data centers will operate properly to serve the company. Data centers are created as one of the organization's resources for handling data management, as well as for the dissemination of data and secondary storage media [3].

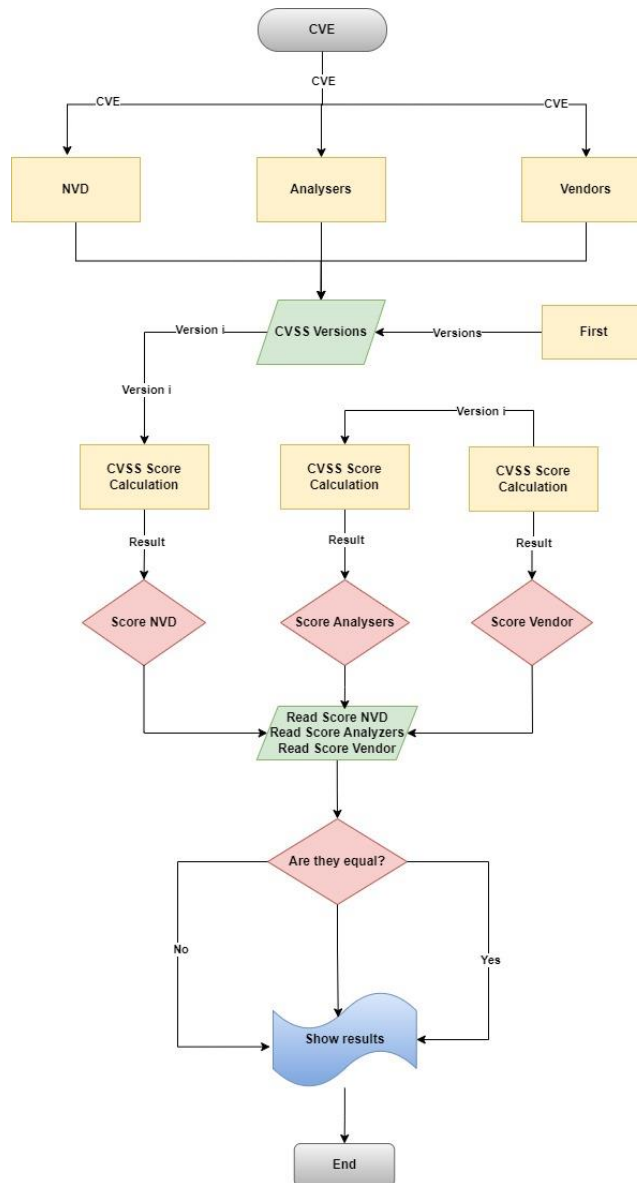
Vulnerabilities that accompany with increasing dependency on these systems in our daily lives also become a part of what we do. In terms of the risks they offer, if exploited, these vulnerabilities might range from being essentially innocuous to quite deadly. [1].

A vulnerability is an issue in a computer program or hardware which can be directly exploited by a hostile actor to obtain access to a system or network and manipulate it in an unfavorable way [4]. The frequency of vulnerability exposure is increasing due to the complexity of logic and the large number of source codes [5].

One may consider vulnerabilities to be a particular class of flaws. They can be more significant than bugs depending on the application, and their discovery calls for a different procedure than that of defects. Vulnerabilities typically go undetected while flaws are discovered by users or developers amid routine system operation. [6].

Researchers have created vulnerability prediction techniques; the proposed frameworks identify features that are expected to be vulnerable and use that information to direct security assessments, such as code audits or security assessments [5]. In addition to increasing the effectiveness of vulnerability recovery and management, properly classifying, and managing vulnerabilities can lower the likelihood of a system attack and harm, which is critical for the system's security performance [7].

To enhance patch selection operations and offer forecasting, contemporary security procedures encourage the use of statistical analytical techniques. In general, vulnerability scoring is carried out using Common Vulnerability Scoring System (CVSS), which necessitates human inputs from an expert, to assess a vulnerability instance based on predetermined parameters [8].



**Figure 0.1: Basic calculation cycle of CVSS**

CVSS is extensively used in educational research and business settings to evaluate vulnerability severities. [9,10] but, in practical use, CVSS presents several difficulties. [11]. Essentially, individual experts determine the severity of a vulnerability since it has been reported in CVE and have the possibility that it will take some time to do so. The longer it takes to assess vulnerabilities, the more likely it is that threats will turn into real breaches [12]. It is consequently expected that vulnerability scoring automation will eliminate the attack vector for zero-day vulnerabilities [13].

Common Vulnerabilities and Exposures dataset is a main source of data, National Vulnerability Database (NVD) additionally assigns CVSS scores and other quantitative metrics to vulnerability reports derived from CVE [8]. However, it may be an influenced and ambiguous choice to only rely on NVD only for assessing and handling vulnerabilities [14]. Furthermore, disparities exist across current iterations of the CVSS, resulting in metrics that are not consistent. These challenges were not sufficiently addressed in earlier research [13] subsequently producing contradictory results.

For instance, NVD uses CVSS version 3 scores to rate vulnerability instances reported only from 2015 onwards. The issues of varying CVSS cores as well as delays incurred are highlighted by the help of Figure 0.1, we can see that for any vulnerability, the score generated as taking formula from First.org and each organization generates their own data which is most of the time not same. [13]

## **1.2 Research Objective**

In this research, we suggest a vulnerability rating method to measure the severity of a reported vulnerability occurrence. By converting quantitative signs into actionable intelligence, the calculated scores will help to improve contextual awareness. This approach overcomes problems with compatibility between different CVSS versions and improves vulnerability prediction. Standard CVSS criteria serve as a foundation for rating vulnerabilities' exploitability as well as the effects of nefarious exploitation.

To achieve these goals, we correlate vulnerability scores and reports from several online cybersecurity data sources, such as NVD, CVE details, and other websites like Vulner and Vuldb. For the vulnerability-severity calculation approach based on Machine Learning (ML), this research works on generating ground truth. After that, it will utilize these cases to train machine learning model, then it further assesses using vulnerabilities found in vulnerability repositories like NVD, VulDb, and Vulner.

The research objectives are:

1. To investigate the potential benefits of AI-based vulnerability prediction of a data center

2. To explore the patterns of vulnerability in NVD and other sources using AI

The assessment of DC vulnerability and associated variables demonstrates a higher degree of automation in cybersecurity evaluations.

### **1.3 Contribution**

The following summarizes this research's primary contributions:

1. A unique machine learning framework for vulnerability assessment that deduces vulnerability instances' CVSS severity scores. As part of the suggested machine-learning paradigm, this technique uses an arithmetic mean system to handle compatibility difficulties of CVSS scores. To provide a shared computational semantic that enhances consistency in vulnerability assessment, the technique can be modified to support a preferred version of the CVSS.
2. A case study on DC vulnerability analysis that supports the suggested machine-learning-based method for vulnerability assessment.
3. A prioritization mechanism for mitigation of vulnerabilities based on analysis of CISA KEV catalog.

### **1.4 Structure**

The remaining portions of this study are arranged as follows:

1. Chapter 2, is the literature review of how other researchers work is related to VSS, NVD and Management of Data Center and the existing standards of CVSS calculations. It also discusses data sources of vulnerability scores.
2. Chapter 3,presents prototype for vulnerability assessment, which uses text analysis techniques to reconcile disparate CVSS versions on a collection of vulnerability reports and compares current CVSS scores against other security warning signs.

3. Chapter 4, is the analysis of vulnerability finding and assessment methods in DC applications primarily using NVD and CVE information API, through some investigation. It discusses the results of experimental work done to process correlated data and achieve accuracy.
4. Chapter 5, elaborates the processes done on the mitigation and prioritization work. It discusses the analysis done on CISA KEV catalog and suggests a way forward.
5. Chapter 6, offer some closing thoughts by providing conclusion of the research work carried out and provides several potential paths for further study.

# LITERATURE REVIEW

This chapter presents a detailed overview of the importance of vulnerability assessment in cyber security in general and Data center in particular by studying previous research work and exploring the possibility and implementation of AI and ML techniques. Section 2.1 explains the diversified work carried out for vulnerability severity assessment and importance of Data Center management. Section 2.2 introduces various Data set sources used to carry out the work. Section 2.3 Discusses Vulnerability Severity Metrics and how each of it weighs the final score calculation.

## 2.1 Related Work

### 2.1.1 Data Center and Vulnerabilities

According to Matthieu et. al [6], The goal of vulnerability projection modelling, a relatively new area of research, is to automatically categorize software entities as susceptible or not. By identifying the entities to concentrate on, this sort of approach aims to assist code reviewers. These methods are beneficial as they are independent of any precise information or dependency of the program, they are working on to garner the investigation. Therefore, they can mold into varying projects by using relevant training data. One way to think of vulnerability prediction is as a branch of flaw prediction. Hall et al [15] carried out a thorough survey and provides a summary of these strategies.

D. Achmadi et. Al. [16] says that, In today's digital epoch, the most important and pivotal entity is data center, especially for a business. Since a point where data goes through progression, broadcast and accumulated, it becomes of a great deal of value that it should be available in terms of service and support.

Levy et. Al. [17] claims that as world is heavily dependent on data centers for its operations, the need to intelligently supervise and administer them with improvement in its efficiency and output increases tremendously. In addition to that, the risk of collapse will be very less. According to Colombelli et. Al. [18] Vulnerabilities are flaws in

systems, processes, and strategies that result in risks. Later on, the researchers [18] with other researcher gives perspective on vulnerabilities in data center as they said that vulnerabilities are available in data centers and we need to tackle them as early as they are discovered as they can pose severe threat to the stability and operation of organization.

Most of the attacks are not carried out by exploiting zero-day vulnerabilities rather they are done using known exploited vulnerabilities which are out in public since a long time like months or years [19].

### **2.1.2 Diverse Approaches**

Sharma et al [1] performed vulnerability prioritization based on the textual description of vulnerabilities. The total number of vulnerabilities in the dataset used is 10,000 and they are divided into three categories based on their CVSS scores. They have made 5 different data sample, the dataset includes data from three vendors viz, Linux (V1), Microsoft (V2) and Google(V3). Each of this data sample contains 2000 vulnerabilities and 90,282, 70,594, 76,006 words in description. The other two data samples (MV1 and MV2) also contain 2000 vulnerabilities each, but they are not vendor specific and contains vulnerabilities detected in products from different vendors. MV1 contains 91,750 words and MV2 contains 78,893 words.

Ziems et. al. [20] explores use of Natural Language processing (NLP) deep learning models to automatically detect vulnerabilities on National Vulnerability Database and used C language as programming tool. They claim 93% accuracy in detecting security vulnerabilities. Whereassharma et. al. [21] developed a semi-automatic model that is based on CVSS score and evaluated Critical threat intelligence feed (CTI) and claims that their model provides strength to mitigate vulnerabilities in a network.

Amarasinghe et. al. [22] worked on the prediction of cyber-attacks by using convolutional neural networks and algorithm SARIMA. Their approach works in two steps, first is the detection of vulnerabilities and second is the prevention by using AI



based models. Dash claims that prediction of an attack is possible by looking at patterns in the Data, resultantly help to protect system from vulnerabilities.

Blinowski& Piotrowski [23] did classification of vulnerability record into seven unique types: the SCADA system, Commercial & Networks, cell phones, PC equipment, other products which are not for home use and Small Office / Home Office. An SVM classifier has been used with the handpicked dataset in order to forecast the categories of novel vulnerabilities. with this they achieved classification accuracy and recall rates of approximately fifty percent or less for fewer-populated groups and seventy to eight for more populated categories. SVM is accurate for classifying text data, albeit the results are not optimal.

Zhang et. Al [24] work primarily focuses on predicting the time in days until the next vulnerability. Utilizing the MulVAL attack-graph analyzer, a quantitative risk assessment methodology based on known vulnerabilities, they have implemented the prediction model. They think that another useful metric for estimating the risk-level of zero-day vulnerabilities may be the quantity of zero-day vulnerabilities (of each software). Researchers also need to consider the severity of each vulnerability for each application, since the number of active zero-day vulnerabilities could not be sufficient to evaluate the risk-level of zero-day vulnerabilities.

S. Na et al [25] have given proposal to use the naïve Bayes classifier to classify CVE entries into vulnerability types. The experimental dataset has been categorized into two groups, CWE, 119 and CWE 79, based on the highest number of CWEs. They claim that their categorization model's accuracy was 99.8%. The accuracy for subsequent experiment's classification which was done for top three and top five CWEs was found to be 95.1% and 84.5%, respectively.

### **2.1.3 Correlated Studies**

The integration of diverse viewpoints from several stakeholders can be achieved by correlation studies linking multiple cybersecurity data sources, linking intricate analysis into more comprehensive statistical relationships [8]. Some popularly used

vulnerability database, cyber-attack and advisory sources are Mitre [26], CVE [27], NVD [28], VulDb [29], Canadian Center for Cyber Security [30]. Afterwards, other websites like Oday.today [31], ExploitDB.com [32], and cvedetails[33] use these sources.

Using PoCs taken from ExploitDB.com as ground truth, the researchers [34] use SecurityFocus and NVD to forecast the exploitability and use of vulnerabilities. The researchers [35] examine the initial public release dates of several data sources, such as SecurityFocus, ExploitDB, NVD and certain companies like Cisco, Wireshark and Microsoft. In comparison to other data sources, they note that vulnerability occurrences disclosed in NVD are behind schedule by one to seven days.

Jimenez et al [6] put an effort in differentiating between components that are vulnerable and non-vulnerable, researchers have evaluated the efficacy of vulnerability prediction models. They have used regular expression to gather data from git URLs. The vulnerable files that were obtained are specifically relevant to the Linux Kernel and are written in C. A total of 1,640 with 743 different types of vulnerabilities have been collected since 2005. They demonstrated that vulnerability models for prediction can be more accurate and more effective than random selection when applied to the Linux kernel. They employed libSVM module of the Weka3 core library as their model.

Jacobs et. al. [36] performed a thorough study and proposed their own exploitability scoring. They extracted common multi-word expressions from the raw text using Rapid Automatic Keyword Extraction and manually culled and normalized a list of 191 tags encoded as binary features for each vulnerability. Exploit code was extracted from Exploit DB, weaponized exploits were found by looking at the modules in Rapid 7's Metasploit framework, D2 Security's Elliot Framework, and the Canvas Exploitation Framework. The Outcome variable, information about whether the vulnerability was exploited in the wild, comes from Proofpoint, Fortinet and AlienVault. (exploitations within the first 12 months after the CVE was published). They used 159 vulnerabilities from June 1, 2016, and June 1, 2018 from MITRE's CVE and NVD including CVSS base severity score, sub-metrics, CPE and used logistic regression model.

Jiang& Atif [8] proposed a mechanism to predict CVSS scores by using NVD data and other third parties. They used majority voting system for selection of CVSS scores and performed a case study on CPS based system. They claim that disadvantage of earlier AI-based CVSS calculating methods is that their models may be biased because they use the vulnerability reports and CVSS scores from NVD only as their training data. Instead, the researchers [8] have combined multiple data sources with NVD like other related vendor reports with independent cyber security analyzers like CERT reports. Further they performed a case study on Cyber Physical Systems and gathered vulnerabilities related to them.

This research is inspired by their work and is an effort on implementing the future work proposed by them and applying another ML model. In addition to that, suggesting mitigation ways by performing in-depth analysis of Cyber Security and Infrastructure Security Agency's Known Exploited Vulnerabilities catalog.

## **2.2 Research Gap**

The dataset is rich and contains options to explore in various ways. One of them is to analyze vulnerabilities, Risks and cyber-attacks in Data center environment, and give recommendations according to study carried out on CISA KEV catalog analysis. In addition to that, previous researchers [6] use majority voting scheme and that can neglect work of other sources. Therefore, scheme of calculating arithmetic mean of various sources scores can be implemented to analyze the results. Further, it is observed that the source used [6] Security focus is discontinued as it is now owned by Symantec is not publicly available. The aim of this research work is to overcome the issue of non-availability of other sources by incorporation of available vulnerability databases and performing different ways of correlating data with different ML model and Arithmetic means, resultantly, acquiring better results in predicting vulnerabilities. Moreover, analyzing CISA KEV for providing a mechanism to prioritize vulnerabilities to mitigate.

## **2.3 Data Repositories**

### **2.3.1 Common Vulnerability Enumeration**

In 1999, the MITRE Corporation organized a collaborative effort to create the CVE List [27]. For giving each disclosed vulnerability a unique identification, MITRE Organization releases the CVE [26] industry standard. Furthermore, it keeps an openly searchable list of all identifiers via CNA which stands for Common Vulnerability Enumeration Numbering Authority.

Every vulnerability on the CVE List gets a distinct CVE Record. After a vulnerability is found, it is reported to the CVE program. A CVE ID is then requested by the reporter and set aside for the disclosed vulnerability. The record is added to the CVE List once the vulnerability that was reported has been verified by determining the minimal set of data pieces needed to create a CVE Record. Worldwide partners of the CVE Program publish CVE Records. Process of CVE is shown in Figure 0.1.

The fields that are usually present in a CVE entry are: an identification number, a synopsis of the reported vulnerability, and any relevant references. The crucial factor that sets one security vulnerability apart from another is its own CVE ID. By doing this, CVE IDs offer a dependable means of interacting with these various databases to obtain further details on the security vulnerabilities that have been disclosed.

### **2.3.2 National Vulnerability Database**

NVD [28] expands on the data included in CVE entries to offer more detailed information for every entry, including impact ratings and severity scores that are determined using the CVSS standard. The structured JSON or XML formats are created from the unstructured CVE data by NVD. NVD offers sophisticated searching options, including OS, manufacturer, product, edition number, vulnerability type, and severity, as part of its improved information. Affected product names and versions have corresponding string entries in CPE entries, among these additional features. The Common Weakness Enumeration (CWE) [32] repository provides features for vulnerability categories by abstracting reported errors and weaknesses into common groupings of vulnerabilities and adding data about expected behaviors, impacts, and extra implementation details. The CVSS version 3 and version 2 guidelines are used to determine the vulnerability severity score.

# CVE Process

## Discover

A person or organization discovers a new vulnerability.

## Report

Discoverer reports a vulnerability to a CVE Program participant

## Request

CVE Program participant requests a CVE Identifier (CVE ID).

## Reserve

The ID is reserved, which is the initial state of a CVE Record. The Reserved state means that CVE stakeholder(s) are using the CVE ID for early-stage vulnerability coordination and management, but the CNA is not yet ready to publicly disclose the vulnerability

## Submit

CVE Program participant submits the details. Details include but are not limited to affected product(s); affected or fixed product versions; vulnerability type, root cause, or impact; and at least one public reference

## Publish

CVE Program participant submits the details. Details include but are not limited to affected product(s); affected or fixed product versions; vulnerability type, root cause, or impact; and at least one public reference

**Figure 0.1: CVE complete process**

### 2.3.3 Other Sources

#### 2.3.3.1 CVE Details

It is a website that provides information about Common Vulnerabilities and Exposures (CVE). CVE is a system that assigns unique identifiers, called CVE IDs, to vulnerabilities in software and hardware. These identifiers help standardize the process of tracking and addressing security vulnerabilities. Typically, the workflow of it is as follows:

1. **Database of Vulnerabilities:** A thorough database of known vulnerabilities is kept up to date by CVE Details. Every vulnerability has its own CVE ID. **Search and Browse:** Users can peruse the database by vendor, product, or vulnerability type, or they can search for individual vulnerabilities.
2. **Vulnerabilities Details:** The website offers comprehensive information on each CVE entry, including a description of the problem, an estimate of its severity, the date it was found, and links to relevant security warnings.
3. **Statistics and Trends:** Statistics on the number of vulnerabilities per manufacturer, product, or severity level are frequently displayed on the website. Users may use this to evaluate the security posture of various hardware and software components.
4. **CVSS Scores:** For every vulnerability, scores from the Common Vulnerability Scoring System (CVSS) are often given. A methodology called CVSS is used to evaluate the seriousness of security flaws and assigns a number to each one that represents the risk involved.
5. **References And Links:** CVE Details usually provide links to other resources, including security advisories and patches, where users may obtain further details about the vulnerability and instructions on how to address or mitigate it.

#### 2.3.3.2 VulDb

It's a leading platform for vulnerability management and threat information, was developed to help businesses remain ahead of security threats. By offering a comprehensive database of known vulnerabilities and exposures, VulDB makes it easier for enterprises to quickly identify, assess, and mitigate security risks [29].

### 2.3.3.3 Vulners

Several essential elements and characteristics are:

1. **Vulnerability Database:** It keeps an extensive database of known vulnerabilities, along with information on severity levels, descriptions, and links to relevant security warnings.
2. **Search Engine:** It allows users to search for specific vulnerabilities, exploits, or other security-related information. The platform pulls pertinent information from its large database using its search engine.
3. **API Access:** Developers and security experts may programmatically access vulnerability information with the help of Vulners.com's API (Application Programming Interface). Integrating vulnerability data into security tools, apps, or scripts can benefit from this.
4. **Monitoring and alerts:** Users can keep track of new vulnerabilities or exploits that align with their interests by configuring alerts and monitoring based on certain criteria.
5. **Integration with Security tools:** it is frequently utilized in tandem with additional security tools and resources. Threat intelligence platforms, security information and event management (SIEM) systems, and other security solutions can include its data.
6. **OpenSource Contributions:** It is connected to several open-source cybersecurity initiatives. This covers scripts and tools for automating the collection and analysis of vulnerability data.

# METHODOLOGY

## 3.1 Introduction

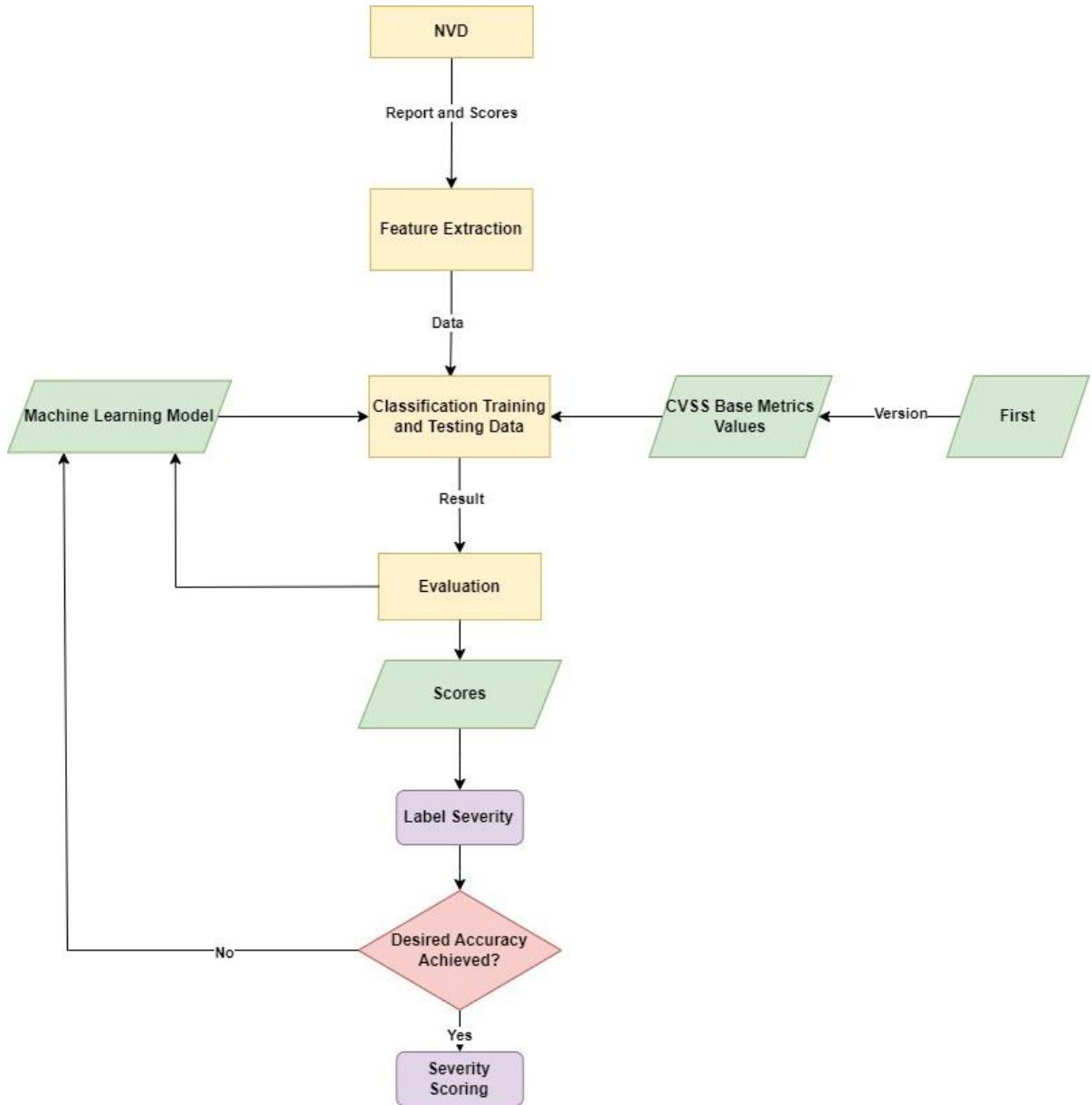
Based on the previous chapter's research gap, this chapter presents a framework for AI based vulnerability scoring prediction in general and using correlated data to predict score for Datacenter Environment in particular. This chapter provides an in-depth exploration of the tasks executed at each point of planned framework. The ensuing argument comprehensively describes the Vulnerability dataset, software, Websites / other Sources, and libraries used in this research.

## 3.2 Framework Planned

The planned framework uses ML model to predict vulnerability scoring by ensuring sufficient training. The dataset used to train ML model in the proposed framework has been adopted from [28] and the case study carried out on DC is done with the help of [27][33][37]. The method employed is basically adopted from [8] and the research gap after identifying in [8] as highlighted in the previous chapter is met by performing arithmetic mean of multiple sources. The framework uses NVD Dataset from the year 2002-2022 for training and 2023 for validation. The TF-IDF [38] is used for extracting features appropriate from the textual description of the vulnerabilities provided with each entry of NVD record. The files are downloaded from [27] in JSON format using Jupyter notebook by python code and then changed into Common Separated Values (CSV) format. The Figure 0.1 shows a broader picture of Vulnerability Severity Scores Calculation for the proposed model that employs ML technique to predict severity scores with the help of data provided from NVD [28]. First data is extracted from NVD website and necessary fields are stored in JSON format. Most import fields are description of vulnerabilities, associated scores, and labels of the classes of different parameters associated with each CVE. At the second step of Figure, the collected data's description is passed through feature extraction process. At third step, the data is classified into training and testing parts. At this step we can see that the data is processed with the help



of Machine learning algorithm and simultaneously the metrics properties and methods are taken from FIRST website. The obtained results at the next step are evaluated and appropriate labels are associated with the predictions. At the next step, accuracy of the model is calculated and if the desired accuracy is not achieved then data is again processed through machine learning model with different set of parameters. Finally, after achieving desired results, severity scores are finalized.



**Figure 0.1: Detailed representation of CVSS score calculation with NVD**

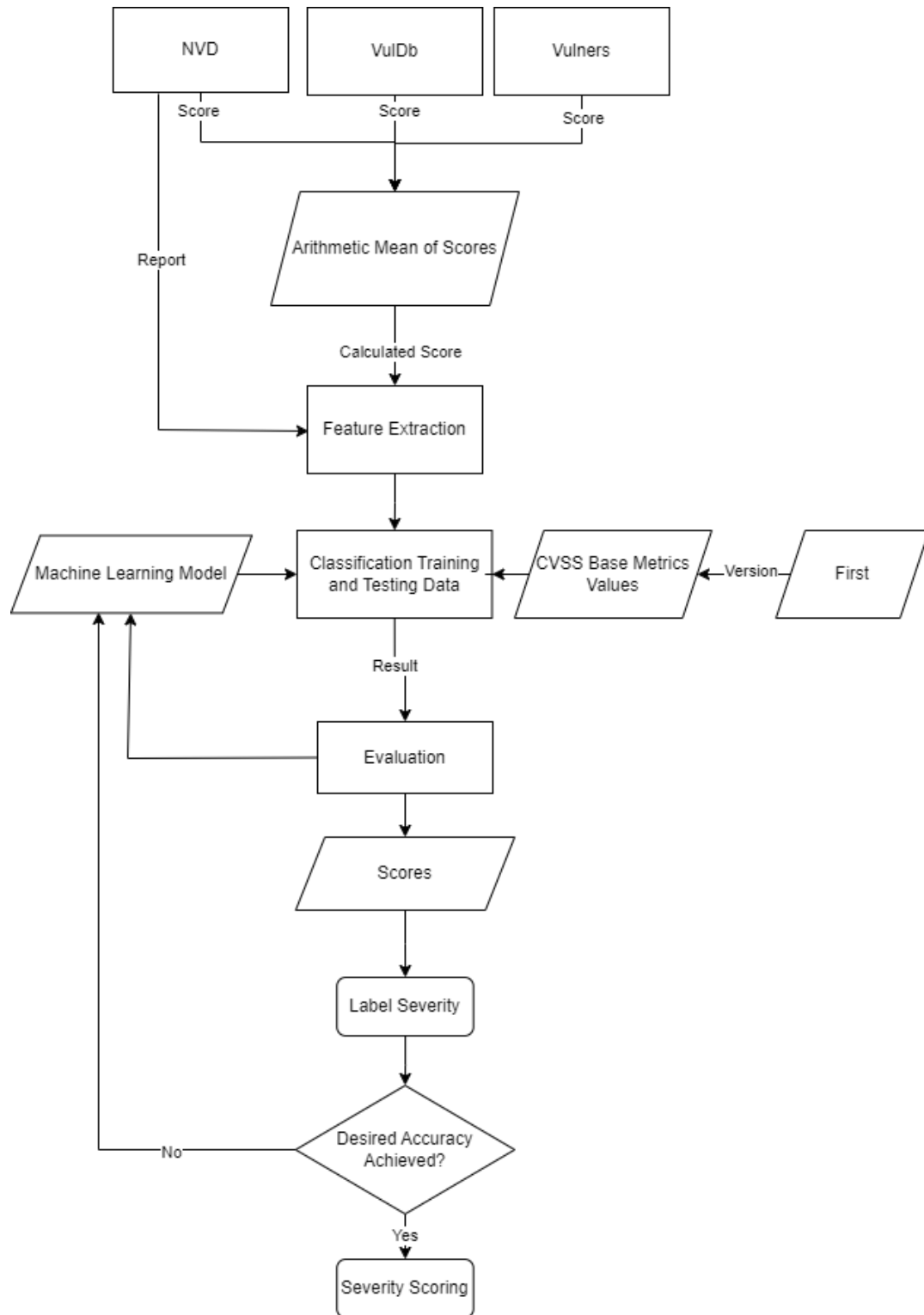
TheFigure 0.2 shows detailed picture of Vulnerability Severity Scores Calculation for the corelated model that employs ML technique to predict severity scores with the help of data provided from sources [28][29] [37]. It has a different scores calculation mechanism as it is at first step taking scores from 3 sources including Vulners, VulDb and NVD. It then takes arithmetic mean of these scores and then use those calculated scores for further calculations same as previous methodology.

### **3.3 Vulnerability Severity Metrics**

An open framework called the Common Vulnerability Scoring System (CVSS) is used to communicate the features and intensity of software flaws. CVSS has various versions including 3.1, 3.0, 2 and 1. This scoring system is developed by Forum of Incident Response and Security Teams (FIRST). The three metric groupings that make up CVSS are Base metrics, Temporal metric, and Environmental metric. Widely used CVSS V2 and V3 calculates the Base Score considers the probable worst-case effect across various deployed contexts and represents the degree of severity of a vulnerability based on its basic properties, which remain consistent over time. Based on variables that vary over time, including the accessibility of exploit code, the Temporal Metrics modify the Base severity of a vulnerability. The Base and Temporal severities are adjusted to a particular computer environment by the Environmental Metrics. They consider things like the environment's mitigations. The organization that maintains a vulnerable item or a third party assessing on their behalf often generate Base Scores. [39]. The base score metrics for CVSS V2 are six which are Access Vector (AV), Access Complexity (AC), Authentication (Au), Confidentiality (C), Integrity (I) and Availability (A). For V3, the base score metrics are eight including AV, AC, C, I, A, Authentication (Au) in place of Privileges Required, and additionally User interaction (UI) and Scope (S). Moreover, Access Vector and Access Complexity are called Attack Vector and Attack Complexity in V3.

#### **3.3.1 Exploitability Metrics**

The features of what is vulnerable—formally referred to as the vulnerable system—are reflected in exploitability metrics.



**Figure 0.2: Broad representation of CVSS score calculation with NVD**

#### 3.3.1.1 Attack Vector (AV)

This measure illustrates the environment in which vulnerability exploitation might occur. The further an attacker may be from the susceptible system, both physically and conceptually, the greater the metric value and, as a result, the severity that results. AV has further four classes Network, Adjacent Network, Local and Physical.

#### 3.3.1.2 Attack Complexity (AC)

Attack complexity measures the tangible actions that an attacker needs to do in order to actively avoid or go around current security-enhancing circumstances put in before they can produce a functional exploit. AC has further two classes Low and High.

#### 3.3.1.3 Authentication (Au)

This metric counts how many times an attacker needs to authenticate with an intended user before they can take advantage of a vulnerability. Au has further three classes Multiple, Single and None.

#### 3.3.1.4 Privileges Required (PR)

This measure indicates the minimum degree of privilege required by an attacker to effectively take advantage of the vulnerability. When no privileges are needed, the Base Score is at its highest. PR is further divided two classes Low and High.

#### 3.3.1.5 User Interaction (UI)

This measure indicates the minimum degree of privilege required by an attacker to effectively take advantage of the vulnerability. When no privileges are needed, the Base Score is at its highest. UI is further divided into two classes None and Required.

### **3.3.2 Scope (S)**

When a vulnerability in one component affects resources in other components that are not inside its security scope, the scope metric records this information. S is further divided into two classes Unchanged and Changed.

### 3.3.3 Impact Metrics

Impact metrics quantify the consequences of a successfully exploited vulnerability on the most directly and reliably attacked component that experiences the worst possible outcome. Impacts should be limited to a feasible, ultimate result that analysts are certain an attacker can accomplish.

#### 3.3.3.1 Confidentiality (C)

This statistic assesses how an effectively exploited vulnerability impacts the confidentiality of the data assets that a software component manages. C is further divided into three classes High, Low and None.

#### 3.3.3.2 Integrity (I)

This metric assesses how effectively vulnerabilities that are exploited affects integrity. Integrity is the quality of being able to rely on and verify information. I is further divided into three classes High, low and None.

#### 3.3.3.3 Availability (A)

This metric calculates the effect that an effectively exploited vulnerability has on the affected component's availability. A is further divided into three classes High, low and None.

## 3.4 Vulnerability Computation

### 3.4.1 Version 2

To calculate the base score for version 2, first Impact is calculated, Equation 3.1 shows the relation of confidentiality, integrity and availability (CIA) in calculation of impact:

$$Impact = 10.41 * (1 - (1 - C) * (1 - I) * (1 - A)) \quad (3.1)$$

Equation 3.2 shows how to measure exploitability by multiplying attack vector, attack complexity, authentication with twenty.

$$\textit{Exploitability} = 20 * AV * AC * Au \quad (3.2)$$

This condition describes that function of impact is equal to zero  $f(\text{Im})=0$ , if calculated impact is zero  $\text{Im}=0$ , otherwise it will be 1.176

AV is different for each class,

If local access = 0.395

Else if adjacent network accessible = 0.646

Else network accessible = 1.0

AC is different for each class,

If high = 0.35

Else if medium = 0.61

Else low = 0.71

Authentication is different for each class,

If multiple = 0.45

Else if single = 0.56

Else none = 0.704

CIA

C is different for each class,

If none = 0.0

Else if partial = 0.275

Else = 0.660

I is different for each class,

If none = 0.0

Else partial = 0.275

Else complete = 0.660

A is different for each class,

If none = 0.0

Else if partial = 0.275

Else complete = 0.660

Finally, all the values are put into the Equation 3.3 for Base Score calculation which is:

$$\text{Base Score} = \text{Round}(((0.6 * Im) + (0.4 * E) - 1.5) * f(Im)) \quad (3.3)$$

### 3.4.2 Version 3

In CVSS V3 calculations, the sub score calculations for impact (ISS) and exploitability determine the Base Score. Equation 3.4 is showing calculation for ISS:

$$ISS = I - [(1 - C) * (1 - I) * (1 - A)] \quad (3.4)$$

Where as Impact (Im) is dependent on Scope (S), which has two conditions, If S is changed, then:

$$Im = 6.44 \times ISS$$

If S is Unchanged, then Equation 3.5 shows the calculation:

$$Im = 7.52 \times (ISS - 0.029) - 3.25 \times (ISS - 0.02)^{15} \quad (3.5)$$

Whereas, the exploitability is calculated as Equation 3.6:

$$E = 8.22 \times AV \times AC \times PR \times UI \quad (3.6)$$

Finally, the Base Score is calculated as, if  $I_m \leq 0$ , then Base Score = 0,

Base score is dependent on Scope (S), which has two conditions, If S is changed, then Else if S is changed, and Equation 3.7 shows calculation of Base Score:

$$\text{Base Score} = \text{Round} (\text{Min} [(I + E), 10]) \quad (3.7)$$

Else if S is Unchanged, then Equation 3.8 shows calculation of Base Score:

$$\text{Base Score} = \text{Round} (\text{Min} [1.08 \times (I + E), 10]) \quad (3.8)$$

### **3.5 Workflow for proposed Framework**

This section explains the techniques employed for data analysis to predict vulnerability effectively. It is pertinent to mention that the prediction is divided into two parts, Parent is where the predictions are carried out on pure data set from NVD and child part which makes the case study about the Data Center and that part is the correlated study. In both the cases, the process consists of different stages including Data extraction, Data Cleaning, Arithmetic mean, Vulnerability severity calculation, Evaluation, Validation, Feature extraction, ML Model application and Libraries used. TF-IDF method of Natural Language Processing (NLP) is used extract features from textual descriptions taken from the reports of each CVE. Data is divided into training and testing parts and input to Machine Learning Models.

#### **3.5.1 Data Extraction**

##### **3.5.1.1 For Overall Data Extraction**

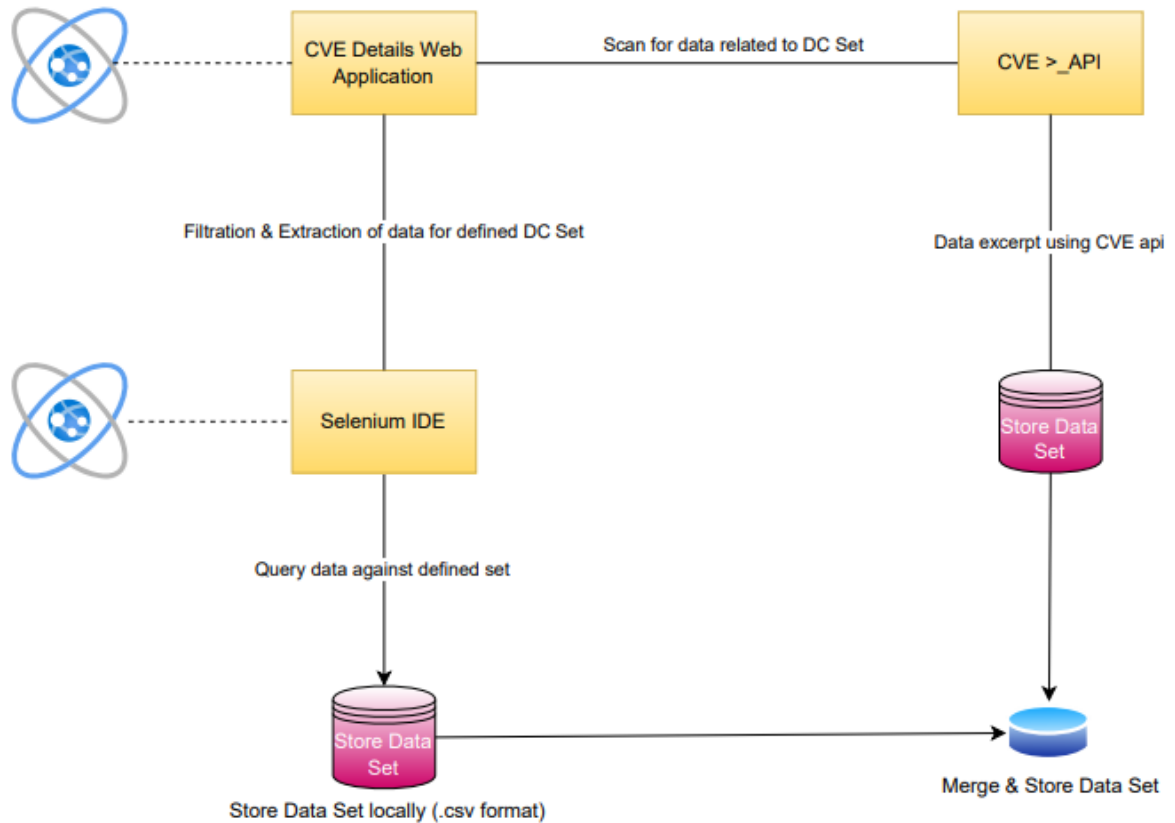
Direct downloads of NVD data streams are made, and they are saved in JSON format in a database on the computer. For CVSS V2, NVD website was crawled, 171219



vulnerability data for the years 2002–2023 are obtained and for V3, 123838 vulnerability data is obtained. Value pairs for characteristics that are understandable by humans, makes up the open standard file format known as JSON, which is used for exchanging data. JSON objects have a tendency to be nested inside other JSON objects, and every nested object within the tree-like structure has an individual access path. NVD includes a wide range of data, including impact measurements (like CVSS), security checklist references (like the CVE dataset), software vulnerabilities connected to security (like CWE), and more. When using NVD, it is typical for researchers to use it according to the requirement from set of values [42].

### 3.5.1.2 For DC Data Extraction

In this research numerous methods have been used to extract vulnerability data from online available platforms or sources, as represented in Figure 0.3.



**Figure 0.3: DC data extraction cycle**

Initially, the CVE data details is collected from the official website [33] in accordance with the defined DC set. Multiple selection mechanism employed for this data extraction phase including CVE api option named as '>\_API' [33] along with Selenium IDE [43] that used mainly for the automation of steps that have been executed for data filtration and export page wise information as per the selection criterion. The Selenium IDE (where IDE stands for Integrated Development Environment) enables web application automation and the recording, editing, replaying, and debugging of functional web app use cases [43]. Total number of vulnerabilities found for the selected categories of DC (Servers, IPS, Router, Switches and Firewall) are for V2 there are 377 records and for V3 there are 214 records.

### 3.5.2 Data Set Preparation

#### 3.5.2.1 For pure NVD Dataset

After performing the data extraction procedure, the data generated for the preparation of data sets that is used for the experimental work in the form of test and train stacks. Precisely, the NVD data feeds that are directly downloaded is JSON format are than converted and stored in a local machine in .CSV format. Microsoft Excel can create and edit a unique kind of file called Comma Separated Value - CSV file. Information is stored in CSV files separated by commas as opposed to columns. Transferring text and numbers between programs is simple when they are stored in a CSV file [44]. Table of the record for each CVE kept in V2 dataset is shown in Table 0.1 and for V3 in Table 0.2.

**Table 0.1: Dataset features for each CVE record CVSS V2**

S No	Data Features
1.	CVE_ID
2.	Report
3.	CVSSV3

4.	Access Vector
5.	Access Complexity
6.	Authentication
7.	ConfidentialityImpact
8.	AvailabilityImpact
9.	IntegrityImpact

**Table 0.2: Dataset features for each CVE record – CVSS V3**

S No	Data Features
1.	CVE_ID
2.	Report
3.	CVSSV3
4.	AttackVector
5.	AttackComplexity
6.	PrivilegesRequired
7.	User Interaction
8.	Scope
9.	ConfidentialityImpact
10.	AvailabilityImpact
11.	IntegrityImpact

### 3.5.2.2 For DC correlated Dataset

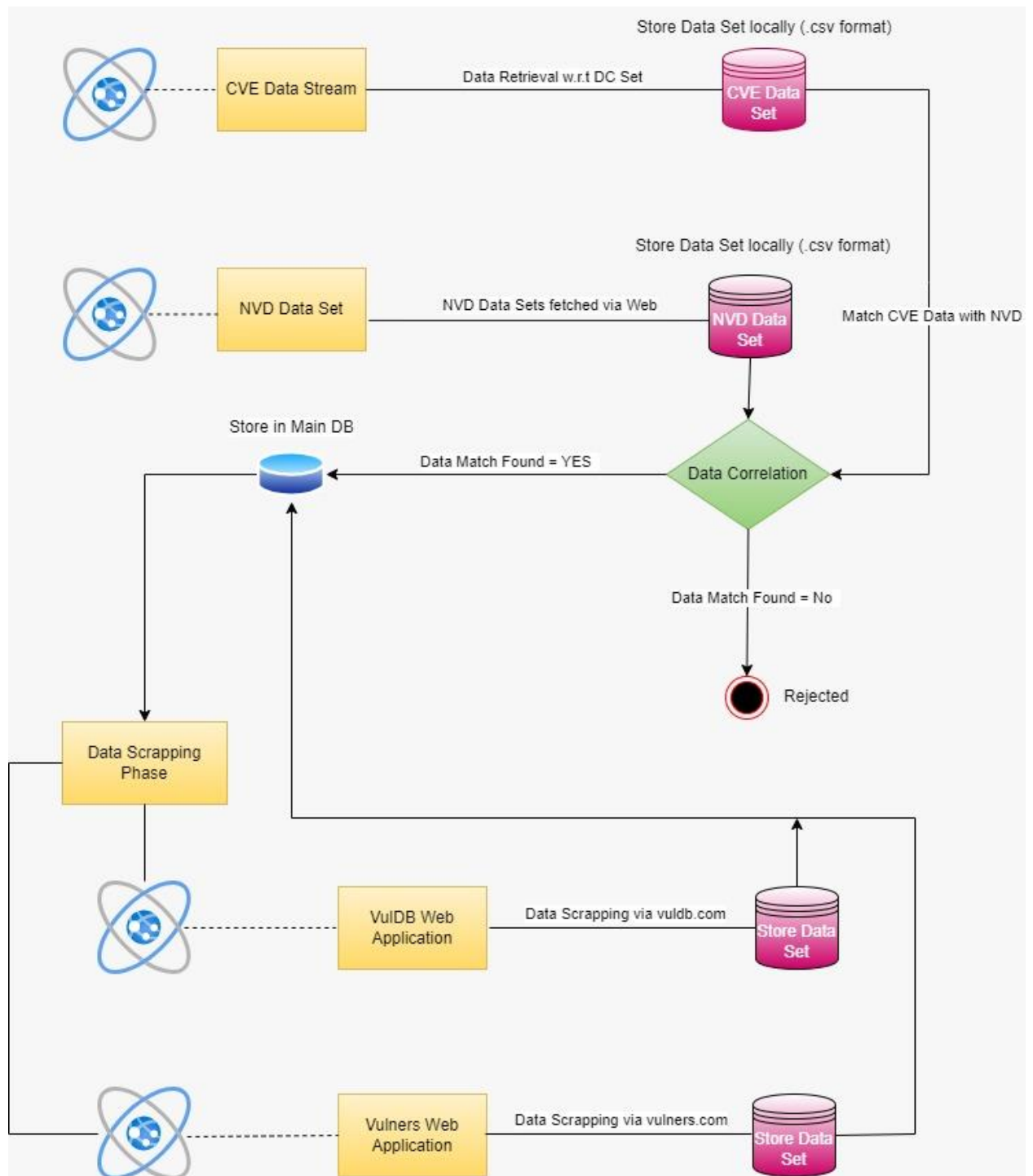
Now, the CVE data set prepared in the ‘Data Extraction’ phase (section 3.5.1.2) is matched with NVD data set. The NVD data sets downloaded from the official website [28] starting from the year 2002 till 2023 for each year separately. The Data Correlation is performed and the matched results within NVD Data sets included in the main data set and those that are not found are rejected.

Moreover, the scraping technique (i.e. the process of extracting information from a website that after being gathered is transported into a form that is more beneficial to the user either an application programming interface or a spreadsheet [45] is employed to obtain vulnerability data released in vendor web applications, like *VulDB*[29], *Vulners*[37], and other vendor applications. Following the process of retrieving and extracting specific information, the data is formatted appropriately for its intended use. Consider data that is kept in the CSV (comma-separated values) format. For V2 correlated records are 352 and for V3 197.

### 3.5.3 Data Cleaning

Data cleaning involves the process of identifying and addressing errors, inadequate structure, duplication, or missing values within the dataset. There are various ways through which data duplication or incorrect classification might occur during the integration of different data sources [40]. The lack of a widely accepted methodology for defining each stage of the data cleaning process may lead to anomalies in dataset. The reliability of results and algorithms can be compromised by the presence of inaccurate data, despite achieving accuracy. Establishing a standardized framework for data cleaning process is imperative to ensure accuracy and consistency across all iterations. The data set finally compiled and used in this research is cleaned by dropping duplicate, missing, Rejected values. Reports that receive a REJECT mark are not given any more attention. Hence, after removal for pure NVD data set V2 vulnerabilities become 171180 and V3 123837. For case study V2 vulnerabilities become 351 and V3 they remain 197.

Complete methodology for the process of data creation from scratch is illustrated in Figure 0.4.



**Figure 0.4: Complete Data generation cycle**

### 3.5.4 Arithmetic Mean (AM)

Utilizing NVD scores as the only basis for training models may introduce partiality into vulnerability evaluation [41]. This is because the human scoring method is thought to be the reason of tiny fraction of inaccuracies in score records of NVD [11]. Different perspectives on vulnerability grading are provided by different data sources such as vulDb, Vulners, and CVE details, in along with statistical vulnerability patterns extracted from CVE files. In this research, The true score, also known as the ground truth score, is the arithmetic mean of the scores from all data sources  $([CVSSV1, \dots, CVSSVn, \dots, CVSSVN])$  where  $0 < n < N, N > 2$  that agree on.

Equation 3.9 shows arithmetic mean for calculation of scores is:

$$AM = \{\text{Sum of Scores}\} \div \{\text{Total numbers of Scores}\} \quad (3.9)$$

If there are just two score sources  $[CVSSV1, CVSSV2]$ , an average score is calculated.

Here, as an example, a vulnerability CVE-2023-5552 is given score of 7.5 by NVD, 6 by VulDb and 7.3 by Vulners. By taking the Arithmetic Mean we got the value 6.9.

### 3.5.5 Vulnerability Severity Calculation

Using text-mining techniques, collected vulnerability reports from current cybersecurity repositories are compared against vulnerability descriptors. Data is categorized using a pipeline of ML algorithms to fill CVSS score gaps.

### 3.5.6 Evaluation

Training and evaluating the classification performance is based on the difference between predicted severity that the original labels. To evaluate this comparison, we use the Accuracy, Balanced Accuracy metrics in addition to the F1- score, Reciprocal rank and mean reciprocal rank.

The performance connotation takes into consideration classes that are not balanced, like the Authentication (Au) class, where the sample size for the "Single" and "None" group is significantly bigger than that of the "Multiple" categories.

Multi-class associations may be involved in the authentication (Au) classification process. In this case, the mean value across class associations is calculated using a micro-average. The distinction between macro- and micro-averages is that the former takes the average inputs from all classes, whilst the latter collects the weighted contributions of all classes. Thus, for multi-class classification issues with class imbalance, the micro-average is preferred. The method is also used for other instances involving several classes. Binary classifiers, such as the one used for User Interface (UI), deduce the balanced-accuracy and F1-score values of the classification using a confusion matrix.

### **3.5.7 Validation**

The validation method involves conducting two evaluation tests using data from crawled websites and data for reports that are acquired from pre-existing repositories. A portion of data is provided in the first experiment to test the validation, and in the second experiment, textual description is provided to provide CVE ratings. In addition, we evaluated the trained algorithm using a validation set of vulnerabilities that were released in 2023. This enables us to extensively evaluate our classifiers with data that was undoubtedly not utilized in any of the training stages.

The second method employed to validate the data works by using some textual descriptions as they were passed through a function and CVSS V2 base score, impact score and exploitability scores were calculated for different CVEs. For example, a textual description "Parsing malformed project files in Omron CX-One versions 4.42 and prior, including the following applications: CX-FLnet versions 1.00 and prior, CX-Protocol versions 1.992 and prior, CX-Programmer versions 9.65 and prior, CX-Server versions 5.0.22 and prior, Network Configurator versions 3.63 and prior, and Switch Box Utility versions 1.68 and prior, may cause a stack-based buffer overflow." associated with CVE\_ID "CVE-2018-7514" was passed through the function. The calculated scores at NVD websites for the vulnerability is base score 4.6, Impact sub score (ISS) is 6.4 and

Exploitability (E) score is 3.9. For the same CVE, scores generated by our model are: Base score is 4.7, ISS is 6.4 and E is 3.9. If we compare our generated score with the website and the generated scores we can see that only slight difference is present in base score which is of  $4.7 - 4.6 = 0.1$ .

For CVSS V3, textual description “Huawei products IPS Module; NGFW Module; NIP6300; NIP6600; NIP6800; Secospace USG6300; Secospace USG6500; Secospace USG6600; USG9500 with versions of V500R001C00; V500R001C20; V500R001C30; V500R001C50; V500R001C60; V500R001C80; V500R005C00; V500R005C10; V500R005C20; V500R002C00; V500R002C10; V500R002C20; V500R002C30 have an improper authentication vulnerability. Attackers need to perform some operations to exploit the vulnerability. Successful exploit may obtain certain permissions on the device.” was taken from a csv file for CVE\_ID “CVE-2020-9099”. This CVE has base score 9.8 on NVD website and the calculated score by our ML model is also 9.8.

### **3.5.8 Feature Extraction**

The proposed ML model uses these scores as ground truths for training purposes. To create the machine-learning pipeline, which includes feature extraction and other data operations, the Python package pipeline in the Scikit-learn library was utilized. Therefore, a simplified transformation of severity scores from various CVSS versions is performed. Tokenization and subsequent feature extractions using the tools CountVectorizer and TfidfTransformer are the first steps in processing the data from NVD vulnerability reports. Then, using word characteristics, TF-IDF (Term Frequency–Inverse Document Frequency) values are computed to create a TF-IDF matrix. Using the train\_test\_split technique, data records are randomly distributed into training (75%) and testing (25%) datasets.

### **3.5.9 ML Model Application**

#### **3.5.9.1 For NVD Dataset**



New vulnerability reports are categorized using machine learning classifiers within anticipated severity patterns [8]. Since Random Forest (RF) classification technique is a well-known and often used algorithm for classification issues, that is why in this research RF is used. Using the Random Forest (RF) classifier, the findings for our case study and generic prediction were achieved. Table 3.3 shows the prediction performances of the CVSS classifiers for the testing datasets.

The results ensure satisfactory performances when compared to closely related CVSS classification researches from [9], who trained CVSS version 3 classifiers using Naive Bayes and Neural Networks algorithms on CVE vulnerability reports published before and within 2016 and [8] who applied Logistic Regression on NVD dataset and correlated dataset produced from year 2002 to 2020.

Overall, we are more accurate. For instance, the Attack Complexity classifier's accuracy is 96% when utilizing correlated data and 96% while using solely NVD vulnerability information. By contrast, the Logistic Regression-based Attack Complexity classifier [8] has a 95.31% accuracy rate. The overall accuracy achieved in this research by using RF is 89%.

#### 3.5.9.2 For DC Dataset

For the DC data set, it is unique that not much of literature is found in this particular domain related vulnerabilities prediction and correlation. The accuracy achieved after correlating is overall 93%.

### **3.5.10 Mathematical Modelling**

A method for lowering an estimated prediction function's variance is called bootstrap aggregation or bagging. A categorization technique called Random Forest uses several decision trees. Using this technique, a randomized decision tree is constructed for every bagging method cycle [46].

#### 3.5.10.1 Classification

Formal definition of classification tree is:

A decision tree is a classification tree in which every node makes a binary decision on whether  $x_i < a$  or not to for a fixed  $a$  (node-dependent) item [47].

All the instances  $(x_k$  and  $y_k)$  are contained in the top node, and the set of examples is further distributed among each node's offspring based on the categorization at that node. The examples are further subdivided until there are only examples from a single class at each node at the bottom. The features  $x_i$  and threshold  $a$  at each node are selected to reduce the 'diversity' that results in the offspring nodes. The *Gini criterion* is frequently used to gauge this diversity.

### 3.5.10.2 Gini Criterion

The subdivision process is repeated until each node at the bottom has only one class—assigned to input  $x$  as a prediction.

Gini Criterion: Define class  $C_1 = \text{Yes}$ ,  $C_2 = \text{No}$

In relation to these two groups, how can the variance of samples inside a node be measured.

Assume that at our present node, we have instances in set  $S$  for classes  $C_1$  and  $C_2$ .

Creating child nodes now, partition  $S = S_1 \cup S_2$

Each sample  $S_1$  and  $S_2$  is partitioned into 2 classes  $C_1$  and  $C_2$

Recall  $|S| = \text{Number of objects in set } S$

$$\hat{P}(S_j) = \frac{|S_j|}{|S|} = \text{proportion of } S_j \text{ in } S \quad (3.10)$$

$$\hat{P}(C_i | S_j) = \frac{|S_j \cap C_i|}{|S_j|} = \text{proportion of } S_j \text{ which in } C_i \quad (3.11)$$

Variation  $g(S_j)$  in set  $S_j$  to be:

$$g(S_j) = \sum_{i=1}^2 \hat{P}(C_i|S_j)(1 - \hat{P}(C_i|S_j)) \quad (3.12)$$

Variation is largest, if set is equally divided among  $C_i$ . It's smallest when all of  $S_j$ 's just one of the  $C_i$

We define the variation of this full subdivision of the  $S_j$  to be the Gini Index = G if:

$$G = \hat{P}(S_1)g(S_1) + \hat{P}(S_2)g(S_2) \quad (3.13)$$

*= weighted sum of variations  $g(S_1), g(S_2)$*

### 3.5.11 Library Used

The Random Forest meta estimator use averaging to enhance predicted accuracy and manage over-fitting by training several decision tree classifiers on different sub-samples of the dataset.

`Sklearn.ensemble.RandomForestClassifier`

If `bootstrap=True` (the default), the sub-sample size is managed using the `max_samples` argument; if not, each tree is constructed using the whole dataset.

#### 3.5.11.1 Parameters

1. No. of trees in forest

*$n\_estimators$  : int, default = 100*

2. This is used to quantify the value of split. Supported criteria are “log\_loss” and “entropy” both for the Shannon information gain and “gini” for the Gini impurity.

*Criterion: can be [`'gini'`, `'entropy'`, `log_loss`'] and by default it is `'gini'`*

3. Maximum depth of tree, if it's none then nodes will grow till last leaves, or all leaves has less than `min_samples_split`

*Max\_depth*, this is integer value and by default it is none

4. To fragment an internal node, the least number of samples mandatory:

for interger value, *min\_samples\_split* provides least value

for floating point value, *min\_samples\_split* is a fraction and the least samples for each split for  $\text{ceil}(\text{min\_samples\_split} * n\_samples)$

5. Below is least samples required to be at a leaf node. If in each left and right node, *min\_samples\_leaf* training sample remains, that will be considered split point. It will affect on smoothing of the model.

for interger value, *min\_samples\_leaf* provides least value

for floating point value, *min\_samples\_leaf* is a fraction and the least samples for each node is  $\text{ceil}(\text{min\_samples\_leaf} * n\_samples)$

6. Samples will be considered equally if *sample\_weight* is not provided. Least weighted fraction for sum total of weights for all input samples are mandatory to be a leaf node.

*min\_weight\_fraction\_leaf*: float, default = 0.0

7. Best setting of features for split are:

For integer, *max\_features* features

For floating point values, *max\_features* is fraction and  $\text{max}(1, \text{int}(\text{max\_features} * n\_features\_in\_))$  for each split

For square root,  $\text{max\_features} = \text{sqrt}(n\_features)$

For logarithm2,  $\text{max\_features} = \text{log2}(n\_features)$

For none,  $\text{max\_features} = n\_features$

8. To produce trees with *max\_leaf\_nodes* in best-first style. Best nodes are distinct as comparative decrease in impurity. For none, infinite figure of leaf nodes

*max\_leaf\_nodes : int, default =None*

9. If a split makes a reduction in impurity better than or identical to its worth, than the node will be split. The subjective impurity reduction equation is as follows:

$$\frac{N_t}{N} * \left( impurity - \frac{N_{t_R}}{N_t} * right_{impurity} - \frac{N_{t_L}}{N_t} * left_{impurity} \right)$$

$N$  is total # of samples,  $N_t$  is # of samples at the current node,  $N_{t_L}$  # of samples in the left child, and  $N_{t_R}$  is # of samples in the right child

For `sample_weight` is passed  $N$ ,  $N_t$ ,  $N_{t_L}$  and  $N_{t_R}$  all refer to the weighted sum

*min\_impurity\_decrease: float, default = 0.0*

10. Bootstrap samples are used, for it used as `False`, entire dataset is used to build each tree

`bootstrap : bool, default = true`

11. It is used to select out-of-bag samples to estimate the generalization score. By default, `accuracy_score` is used. Provide a callable with signature `metric(y_true, y_pred)` to use a custom metric and can be used if `bootstrap` is true.

*oob\_score: boolean pr callable, default=False*

12. It defines # of jobs to run simultaneously. `Fit`, `predict`, `decision_path` and `apply` are all parallelized over the trees. None means 1 unless in a `joblib.parallel_backend` context. -1 means using all processors

`n_jobs : integer, default = none`

13. It takes care of randomness of bootstrap process and feature sampling for best split which is `max_features < n_features`

*random\_state : integer, randomState instance or None, by default it is none*

14. It takes care of verbosity while fitting and predicting

*verbose :integer, by default it is equal to 0*

15. It reutilize the result of previous call to fit and inserts added estimators to ensemble for if set to `True`, else, it simply fits a complete novel forest

*warm\_start : Boolean, by default it is false*

16. Class-specific weights in the format `{class_label: weight}`. All classes are expected to have weight one if it is not provided. A list of dicts can be supplied in

the same order as the columns of  $y$  for multi-output issues. It should be noted that weights for multioutput (including multilabel) should be defined in a separate dict for each class of each column. For instance, rather than  $[\{1:1\}, \{2:5\}, \{3:1\}, \{4:1\}]$ , the weights for the four-class multilabel classification should be  $[\{0: 1, 1: 1\}, \{0: 1, 1: 5\}, \{0: 1, 1: 1\}, \{0: 1, 1: 1\}]$ . The balanced mode uses  $n_{\text{samples}} / (n_{\text{classes}} * \text{np.bincount}(y))$  to automatically modify weights inversely proportionate to class frequencies in the input data. The *balanced\_subsample* mode is identical to the "balanced" mode; however, weights are calculated using the bootstrap sample for each tree that is generated. The weights in each column of  $y$  will be multiplied for multi-output. keep in mind that if *sample\_weight* is provided, these weights will be multiplied by *sample\_weight* (using the fit method).

*Class\_weight* : {“balanced”, “balanced\_sample”}, dict or list of dicts, by default it is none

- Minimal Cost-Complexity Pruning's complexity parameter. The subtree that is smaller than *ccp\_alpha* and has the highest cost complexity will be selected. By default, there is no pruning done

*ccp\_alpha*: non-negative float, default = 0.0

- The # of samples to take from  $X$  for training each base estimator, it is used if bootstrap is true. If it is none which is default value than draw  $X.\text{shape}[0]$  samples. For it as integer, than takes sample value from *max\_samples* and for floating point, it takes  $\max(\text{round}(n_{\text{samples}} * \text{max\_samples}), 1)$  samples. Therefore, *max\_samples* has to be in the interval (0.0, 1.0)

*max\_samples* : integer or float, default = none

### 3.5.11.2 Attributes

- The set of fitted sub-estimators was produced using the child estimator template

*estimator\_*: *DecisionTreeClassifier*

- Estimator used to the ensemble growth

*base\_estimator\_*: *DecisionTreeClassifier*

3. The grouping of adjusted sub-estimators

*estimator\_*: *list of DecisionTreeClassifier*

4. A list of arrays containing class labels (multi-output problem) or the class labels themselves (single output problem)

*classes\_*: *ndarray of shape (n\_classes,)* or *a list of such arrays*

5. A list with the number of classes for each output (multi-output problem) or the number of classes (single output problem)

*n\_classes\_*: *int or list*

6. # of features seen during fit

*n\_features\_in*: *int*

7. Names of the characteristics observed during the fit. Defined exclusively in the case that X has all string feature names

*feature\_names\_in\_*: *ndarray of shape (n\_features\_in,)*

8. The quantity of outputs produced by a fit

*n\_outputs*: *integer*

9. The feature importances dependent on impurities

*feature\_importances\_*: *ndarray of shape (n\_features,)*

10. The training dataset's score was calculated using an out-of-bag estimate. Only when *oob\_score* is True does this property exist

*oob\_score\_*: *float*

11. On the training set, the decision function was calculated using an out-of-bag estimate. It is conceivable that a data point was never omitted during the bootstrap if *n\_estimators* is tiny. NaN may be present in *oob\_decision\_function\_* in this instance. Only when *oob\_score* is True does this property exist.

*oob\_decision\_function\_*: *ndarray of shape (n\_samples, n\_classes) or (n\_samples, n\_classes, n\_outputs)*

## 3.6 Algorithms

Algorithms are used in data storage, sorting, processing, and machine learning to determine the optimal solution for a given issue. They increase a program's efficiency in the process [49]. Therefore, for each piece of computation during this research, algorithms are written.

### 3.6.1 Algorithm 1: Calculation of Base Score CVSS V3

**START PROCESS** CVSSCalculation (SEML, a, A, CVSS, CVSS')

1. SEML is a model of Machine Learning with function  $f()$
2.  $fBasescore()$  is the CVSS evaluator function
3.  $[CVSS1, \dots, CVSSd, \dots, CVSSD]$  ( $0 < d \leq D, D > 2$ ) is a data sources array, where each element has vulnerability instances 'R'. An array of severity scores,  $[ssi,1, \dots, ssi,d, \dots, ssi,D]$ , and a set of CVSS vectors,  $[Ti,1, \dots, Ti,d, \dots, Ti,D]$ , have been allocated to every vulnerability instance  $vi$  ( $0 < i \leq R$ ).
4. A list of vulnerability instances without any severity rating or CVSS measurements denoted as  $CV'$  of  $cp$  ( $0 < p \leq R'$ ).
5. The set  $a$  consists of CVSS metrics  $aj$  ( $0 < j \leq A$ ) in which every metric  $aj$  possesses an array of  $B$   $aj$  classes to be mapped that correspond to the value  $T(aj) i \in \{c1(aj), \dots, ck(aj), \dots, cB(aj)\}$  ( $0 < b(aj) \leq B(aj)$ ).
6.  $D = |[CV1, \dots, CVd, \dots, CVD]|$ ,  $R = |CVd|$ ,  $R' = |CV'|$ ,  $A = |a|$ ,  $B(aj) = |\{c1(aj), \dots, ck(aj), \dots, cB(aj)\}|$

**FOR** vulnerability instance  $vi$  ( $i = 1, \dots, R$ )

**DO**

**FOR** CVSS metric  $aj$  ( $j = 1, \dots, A$ )

**DO**

**SET**  $T(aj)i = \arg \max B(aj) [card\{\{c1(aj), \dots, ck(aj), \dots, cB(aj)\}\} | T(aj) i, d ] (0 < d \leq D)$  as the reference point for measuring CVSS



**END FOR**

$Ti = [T(a1) i, \dots, T(a_j) i, \dots, T(aA) i] (j = 1, \dots, A)$

$SET_{ssi} = f_{Basescore}(Ti)$  as the base information for the severity score

**END FOR**

**FOR**  $j = 1, \dots, A$  CVSS metric  $aj (j = 1, \dots, A)$

**DO**

$Train(SEML)$  //SEML a model for training and testing of historic datasets

$f(aj)(vi) = \arg \max b(aj) f(aj) b(aj)(vi)$

**END FOR**

**FOR** vulnerability instance  $cp (p = 1, \dots, R')$

**DO**

**FOR** CVSS metric  $aj (j = 1, \dots, A)$

**DO**

$X(aj)p = f(aj)(cp)$

**END FOR** //Obtain the estimated CVSS measurement from SEML as a result

$Xp = [X(a1)p, \dots, X(aj)p, \dots, X(A)p] (j = 1, \dots, A)$

**END FOR**

The predicted resultant score  $xp = f_{Basescore}(Xp)$

**END PROCESS**

### 3.6.2 Algorithm 2 Generation of Data Product wise from CVE details

**START PROCESS***CveDetailsDataGenerator*( $D_n, P_s, CVE_e$ )

1.  $D_n$  is an empty dictionary to store vulnerabilities organized by product type  $P_s$ , where  $s = \{0, \dots, 4\}$
2. The five product types that are imported:  $P_1$  for Switch:  $P_2$  for Router:  $P_3$  for IPS,  $P_4$  for Firewall:  $P_5$  for Server
3.  $CVE_e$  is each entry of CVE in CVE details

**FOR** *each entry in CVE Details*

**DO**

*Extract the  $P_s$  from the  $CVE_e$  entry*

**IF** ( $P_s$  not in  $D_n$ )

**THEN**

*Discard that  $P_s$*

**Else**

*Add the  $CVE_e$  entry of corresponding  $P_s$  in  $D_n$*

**END IF**

*return the  $D_n$*

**END FOR**

**SORT** *dictionary  $D_n$  by  $P_s$*

**END PROCESS**

### 3.6.3 Algorithm for Impact Calculation

**START PROCESS***ImpactCalculation*( $I_m, S, UC, ISS, C, E$ )

1. CVSS calculations, the sub score calculations for impact (ISS) and exploitability determine the Base Score. Formula for ISS is:  $ISS = 1 - [(1 - C)(1 - I)(1 - A)]$   
*Where C = Confidentiality, I = Integrity, A = Availability*
2. *WHEN  $I_m$  is dependent on S: where S = Scope, UC = Unchanged, I<sub>m</sub> = Impact, C = Changed, E = Exploitability*
3. *AV is Access Vector: AC is the Access Complexity: UI is User Interaction: PR is Privileges Required*

***IF S is UC***

***THEN***

$$I_m = 6.44 * ISS$$

***ELSE IF S is C***

***THEN***

$$I_m = 7.52 * (ISS - 0.029) - 3.25 * (ISS - 0.02) ^ 15$$

$$WHEREAS E = 8.22 * AV * AC * PR * UI$$

***END PROCESS***

### **3.6.4 Algorithm for Base Score Calculation CVSS V2**

***START PROCESS***BaseScoreCalculation( $I_m, S, C, UC, ISS, E$ )

***IF  $I_m < = 0$***

***THEN***

$$Base\ Score = 0$$

***ELSE IF S is C***

***THEN***

*Base Score = Round (min [ (I<sub>m</sub> + E), 10])*

**ELSE IF** *S is UC*

**THEN**

*Base Score = Round (min [1.08 \* (I + E), 10])*

*([CVSSV1,...,CVSSVn,...,CVSSVN] where 0 < n < N, N > 2)*

**END PROCESS**

### **3.6.5 Algorithm to Calculate Arithmetic Mean**

**START PROCESS** *ArithmeticMeanCalculation(I<sub>m</sub>, S, C, UC, ISS, E)*

1. *Define SC := Score*
2. *Define S := sum()*
3. *Define C := count()*
4. *Define AM := Arithmetic Mean, Avg := Average*

**IF** *SC > = 0*

**THEN** *AM = {S (SC) / (C (SC))}*

**ELSE IF**

*SC = From Two Score Sources [CVSSV1, CVSSV2]*

**THEN**

*AM = Avg (SC) //An average score is calculated*

*Avg = S (All Sources Scores) / 2*

**END PROCESS**

# EXPERIMENTS AND RESULTS

Several experimental findings are provided in this chapter to assess the suggested machine-learning model covered in Chapter 3. Experiments have been conducted using the NVD dataset. By extracting the metrics to assess the classification performance, the effectiveness of the suggested machine learning model is ascertained. There is also a comparison of various tried-and-true methods that have been documented in the body of literature.

The results of the experiment are presented in tabular and graphical forms, which facilitated the comparison of the algorithms' performance on different data sets. The statistics and class imbalances are shown using bar charts, which clearly indicated occurrence of vulnerabilities over the period. The accuracy of model is shown using confusion matrix for each Label and its classes and keeping the normalization true to have results in more understandable form.

## 4.1 Evaluation Metrics

Assessment metrics including precision, recall, F1 score, and accuracy which are frequently used in a variety of domains, including vulnerability detection—are used to evaluate the suggested models. Each metric's specifics are presented in the sections that follow.

### 4.1.1 Accuracy

The degree of accuracy demonstrated by a model's predictions is referred to as its "accuracy." The percentage of correctly classified instances including true positives and negatives to every instance is referred to as the accuracy rate. Equation 4.1 below is utilized in its computation.

$$Accuracy = (T_P + T_N)/(T_P + T_N + F_P + F_N) \quad (4.1)$$

Denoting,

1. True Positive - TP = The total number of positive (threat-related) instances that have been accurately identified (threats).
2. True Negative – TN = The total number of cases that were accurately classified as negative (non-threats).
3. False Positive - FP = The number of cases that were misclassified as positive—that is, non-threats that seemed erroneously thought to be threats.
4. False Negative – FN = The number of cases that are mislabelled as negative (that is, threats that are mislabelled as non-threats).

#### **4.1.2 Precision**

The level of accuracy in positive estimations is referred to as precision. The percentage of accurately foretold positive instances, including both true and false positives, to the total number of positively predicted positive instances is known as the true positive rate. The following is the precision formula:

$$Precision = (T_P)/(T_P + F_P) \quad (4.2)$$

#### **4.1.3 Recall**

The number of correctly estimated positive events (true positives) about all actual positive scenarios (true positives along with false negatives) is known as 'Recall', also referred to as sensitivity as well as true positive rate. Equation 4.3 provides the mathematical expression.

$$Precision = (T_P)/(T_P + F_P) \quad (4.3)$$

#### **4.1.4 F1 Score Balanced**

A measure of accuracy that combines recall and precision is referred to as the F1 score. Particularly in the case of an unequal class distribution, the F1 score frequently provides more information than raw accuracy. Therefore, reaching a state of balance

between each of the measurements. It is widely used to assess a classification model's performance in machine learning, deep learning, and statistical analysis. Equation 4.3 and 4.4, below demonstrates the F1 score in mathematical form:

$$F1\ Score = (2) / \left( \frac{1}{Precision} + \frac{1}{Recall} \right) \quad (4.3)$$

$$F1\ Score = (2) \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (4.4)$$

There are several variants of the F1 score, such weighted, macro, and micro F1 scores, which are better suited for situations involving multiple classes or when you wish to assign various weights to the classes.

#### 4.1.4.1 Accuracy Macro Score

The macro method involves calculating the F1 score for every class separately and averaging them. This method assumes that every class matters equally, which isn't necessarily the case. Equation 4.5 shows calculation of Macro F1:

$$Macro\ F1 = \frac{1}{N} \sum_{i=1}^N F1_i \quad (4.5)$$

where the F1 score for each class is denoted by  $F1_i$ , and  $N$  is the total number of classes.

#### 4.1.4.2 Accuracy Micro Score

In the micro method, the average F1 score is calculated by adding up the input from each class. Smaller classes are given the identical weight as bigger classes in this strategy, which is helpful if your dataset has an imbalance in classes. The steps involved in calculating F1 Score are:

1. Calculate True positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN).
2. Use the calculated TP, TN, FP and FN to calculate recall and precision.

3. Use recall and precision values to calculate F1 Score.

$$\text{Micro F1 Score} = (2) \cdot \frac{\text{Precision}_{\text{micro}} \cdot \text{Recall}_{\text{micro}}}{\text{Precision}_{\text{micro}} + \text{Recall}_{\text{micro}}} \quad (4.6)$$

#### 4.1.5 Support

The number of real instances of the class in the given dataset is known as support. It shows how many actual instances there are of each class. In multiclass classification settings, support is frequently used to show how cases are distributed throughout several classes.

#### 4.1.6 Reciprocal Rank and Mean Reciprocal Rank

Reciprocal rank is a statistic that is used in ranking assessment and information retrieval to gauge how well a search engine or system for retrieving information is working. It is especially prevalent when assessing how well systems function when they return ranked lists of responses to user queries.

Given a ranked list of objects, Reciprocal Rank (RR) is determined as the reciprocal of the rank at which the first relevant item is discovered. If the first relevant item is at position  $k$ , the Reciprocal Rank (RR) is as shown in Equation 4.7:

$$RR = \frac{1}{k} \quad (4.7)$$

In the field of data extraction, an exploration or system of recommendations often generates the ranked list, and things are ranked according to certain criteria (e.g., if an item fulfills a user's information requirement or desire).

To calculate Mean Reciprocal Rank (MRR), we use RR calculated using formula of Reciprocal Rank (RR). Then we take sum of all RR we get (Reciprocal Rank for the  $i^{\text{th}}$  instance) and divide it by the sum of total instances (which is  $n$ ). Equation 4.8 shows calculation for MRR:



$$MRR = \frac{\sum_{i=1}^n RR_i}{n} \quad (4.8)$$

## 4.2 Results of Random Forest Model

### 4.2.1 RF Classifier specification

In this research, practical work is done in two parts: First part is where pure NVD dataset is preprocessed, and ML model RF is applied on it and secondly the correlated data for DC environment is taken and RF is applied on it. All the work is done using Jupyter Notebook and coding is done using Python language. The specification or parameters applied of RF classifier is shown is **Error! Reference source not found..1**.

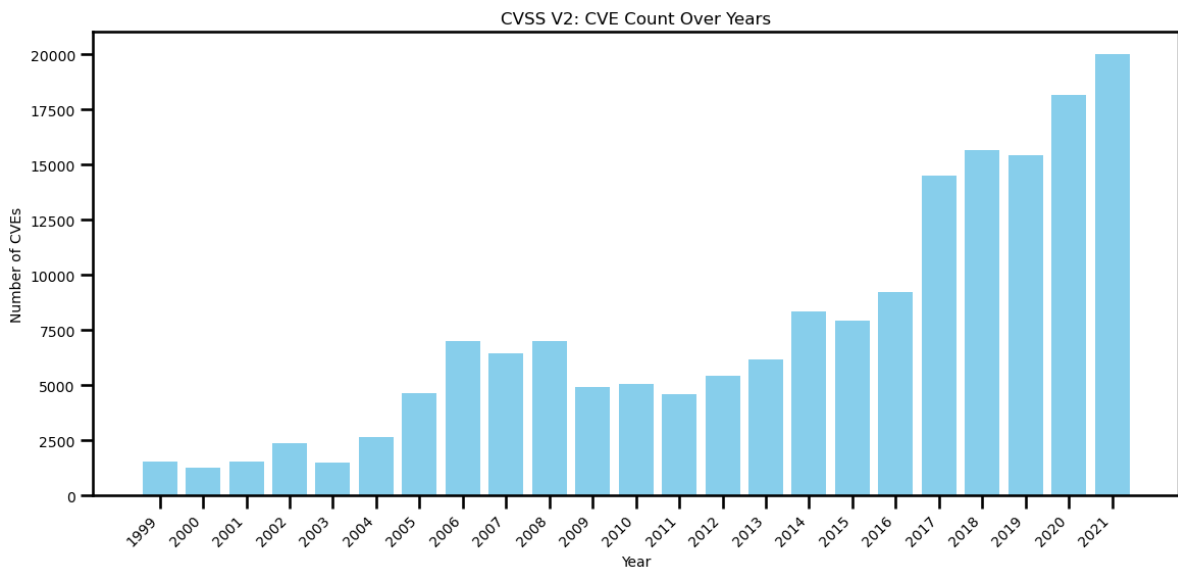
**Table 0.1: RF parameters**

S No	RF Parameters	Values
1.	n_estimators	40-180
2.	Criterion	gini
3.	Max_depth	none
4.	min_weight_fraction_leaf	0.0
5.	max_features	n_features
6.	max_leaf_nodes	none
7.	min_impurity_decrease	0.0
8.	bootstrap	true
9.	oob_score	false
10.	n_job	none

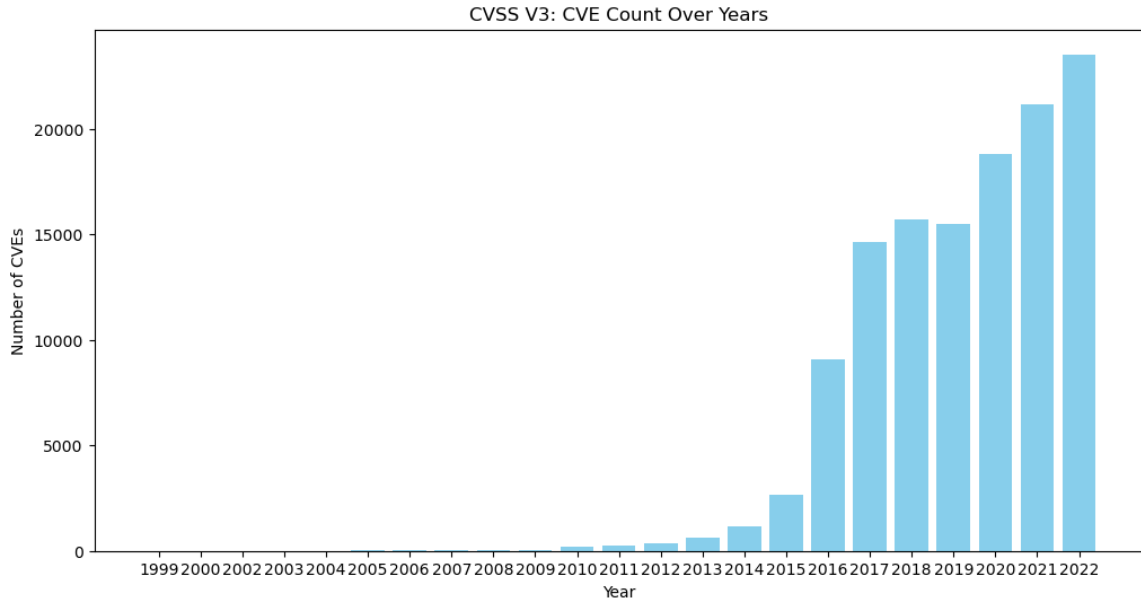
11.	random_state	none
12.	verbose	0
13.	warm_start	false
14.	Class_weight	none
15.	ccp_alpha	0.0
16.	max_samples	none

#### 4.2.2 Data Breakdown

No of CVSS V2 vulnerabilities spread over period of years are shown in the Figure 0.1 and CVSS V3 in Figure 0.2 below:

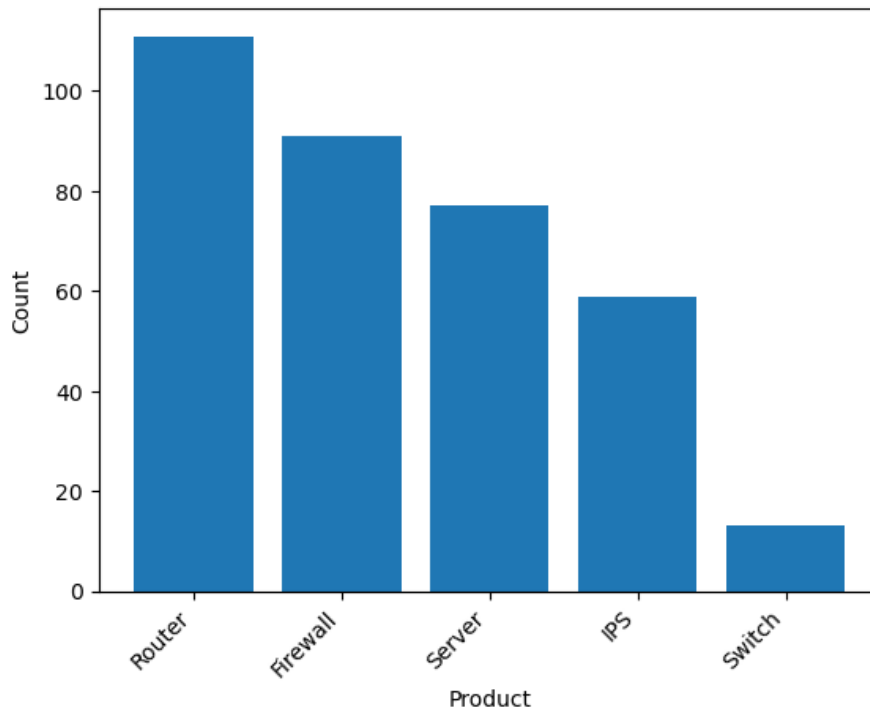


**Figure 0.1: Count wise vulnerabilities of CVSS V2**



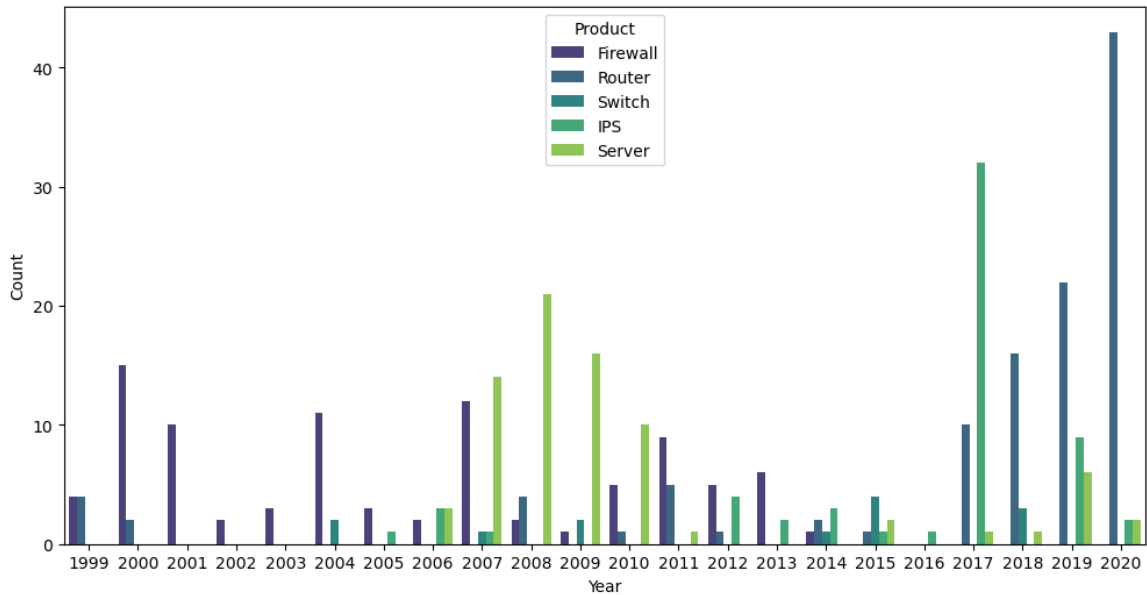
**Figure 0.2: Count wise vulnerabilities of CVSS V3**

Spread of vulnerabilities product wise for the case study of DC is shown in Figure 0.3:



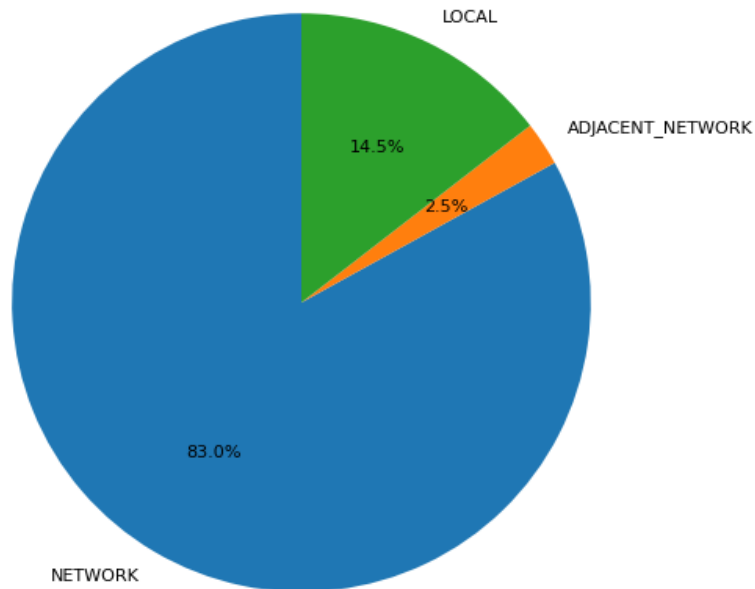
**Figure 0.3: Count wise vulnerabilities of DC products**

Spread of vulnerabilities over years product wise for the case study of DC is shown in Figure 0.4:

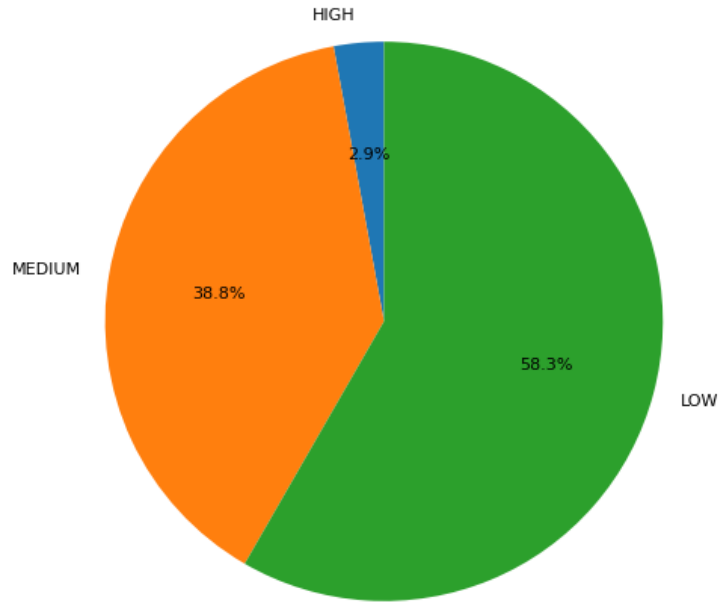


**Figure 0.4: Count wise vulnerabilities of DC products**

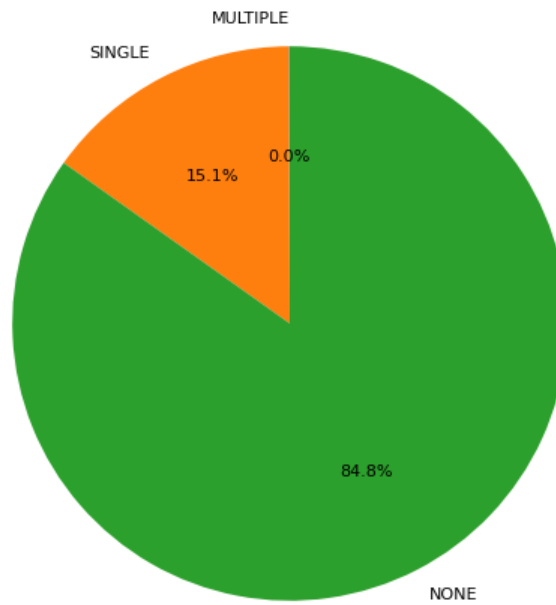
The CVSS V2 data of NVD has clear class imbalances which can be seen by Figure 0.5 to Figure 0.10.



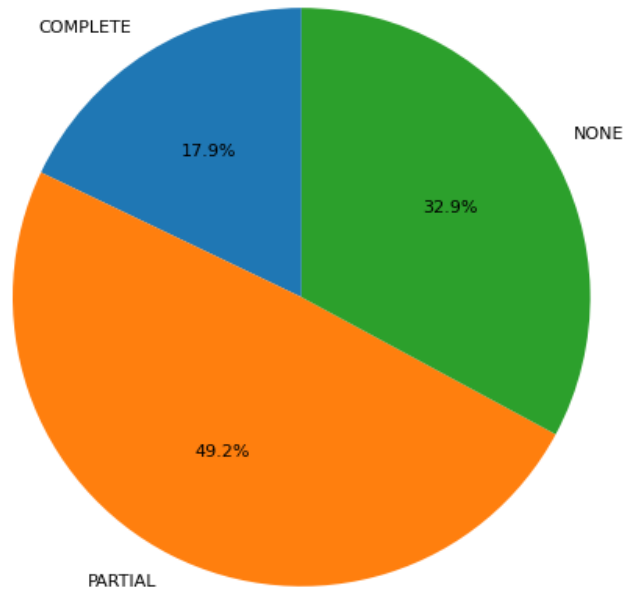
**Figure 0.5: Percentage of different classes data in Access Vector – CVSS V2**



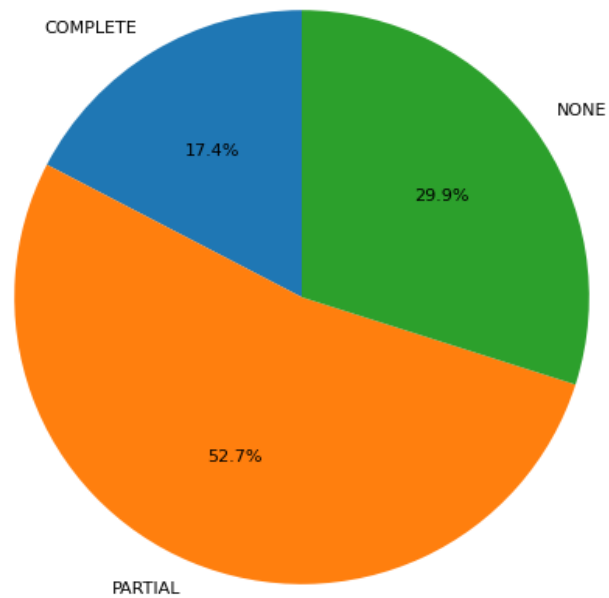
**Figure 0.6: Percentage of different classes data in Access Complexity– CVSS V2**



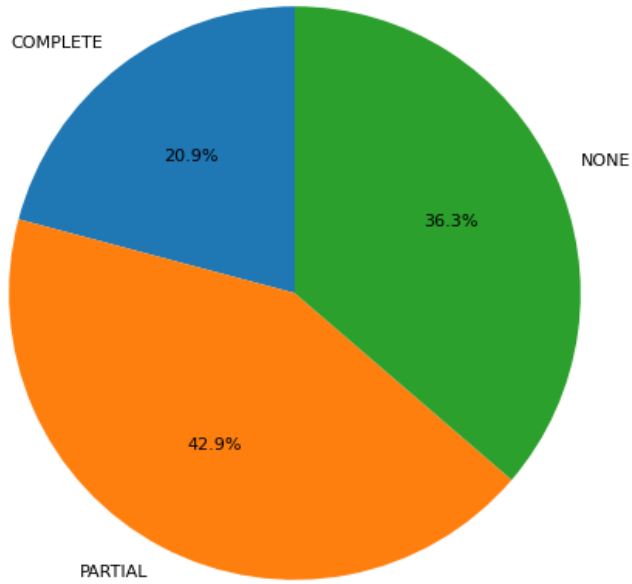
**Figure 0.7: Percentage of different classes data in Authentication – CVSS V2**



**Figure 0.8: Percentage of different classes data in Confidentiality Impact – CVSS V2**

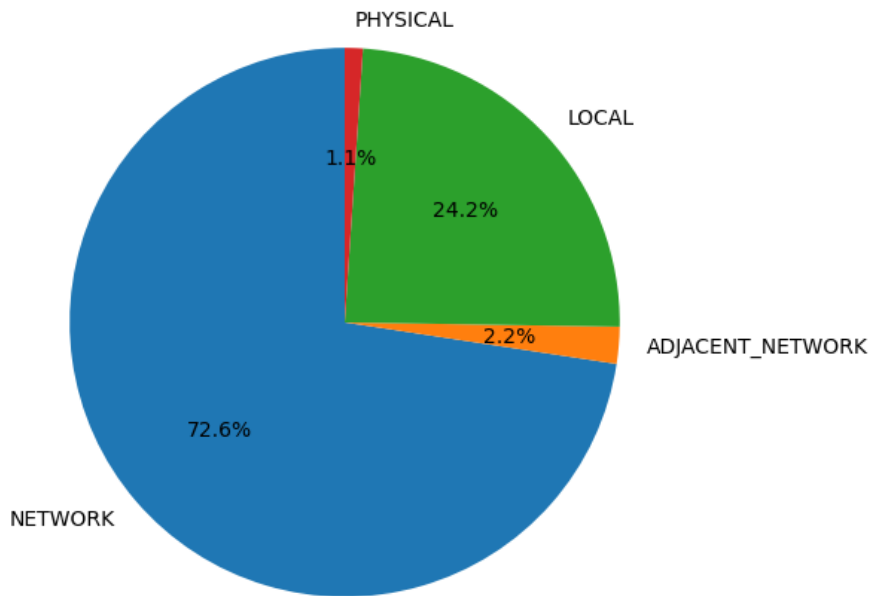


**Figure 0.9: Percentage of different classes data in Integrity Impact – CVSS V2**

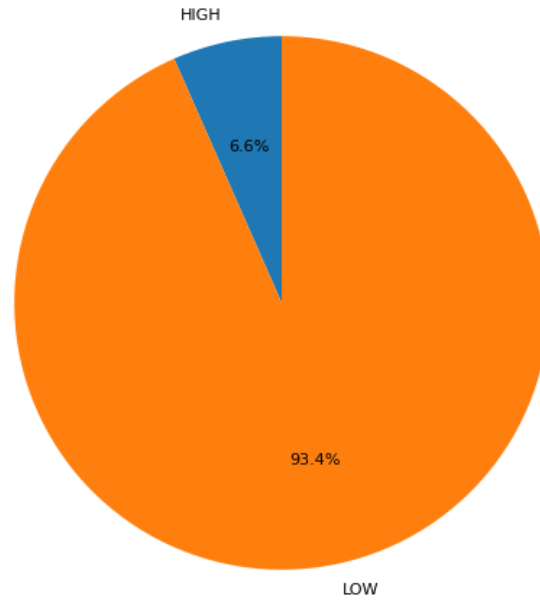


**Figure 0.10: Percentage of different classes data in Availability Impact – CVSS V2**

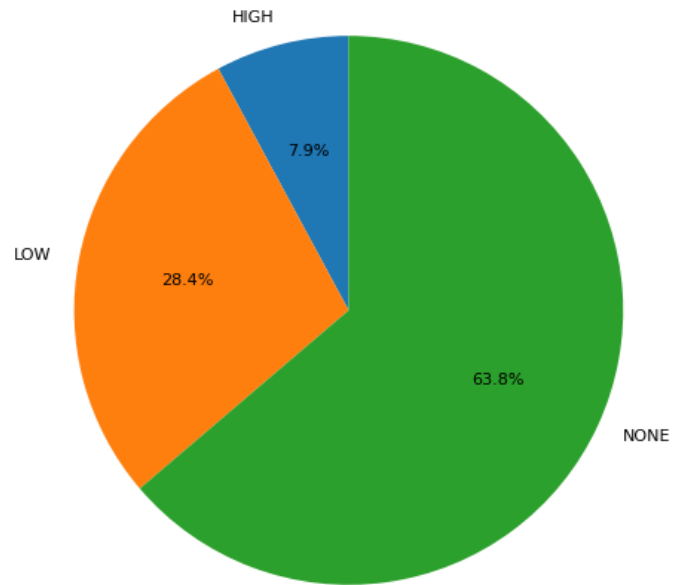
The CVSS V3 data of NVD has clear class imbalances which can be seen by fig 4.11 to fig 4.18



**Figure 0.11: Percentage of different classes data in Attack Vector – CVSS V3**

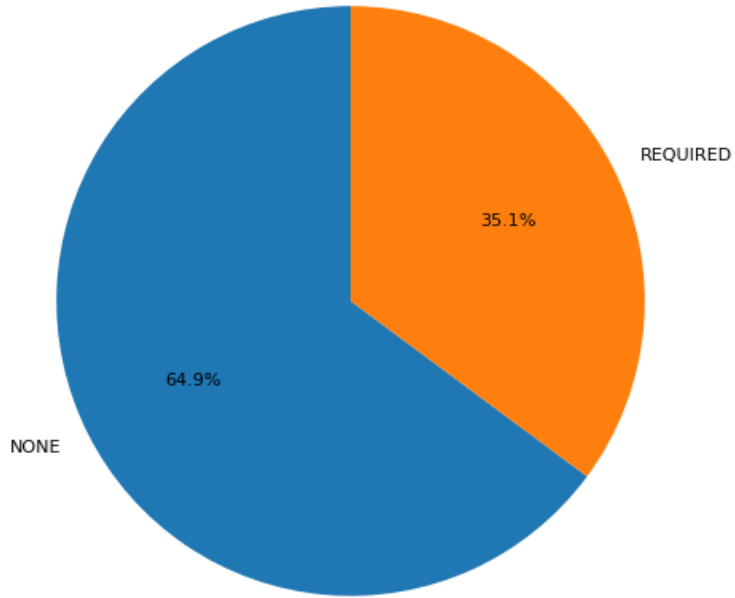


**Figure 0.12: Percentage of different classes data in Attack Complexity – CVSS V3**

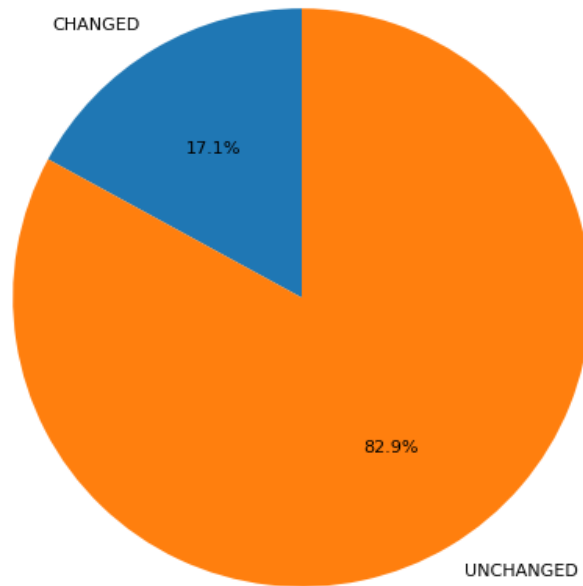


**Figure 0.13: Percentage of different classes data in Privileges Required – CVSS V3**

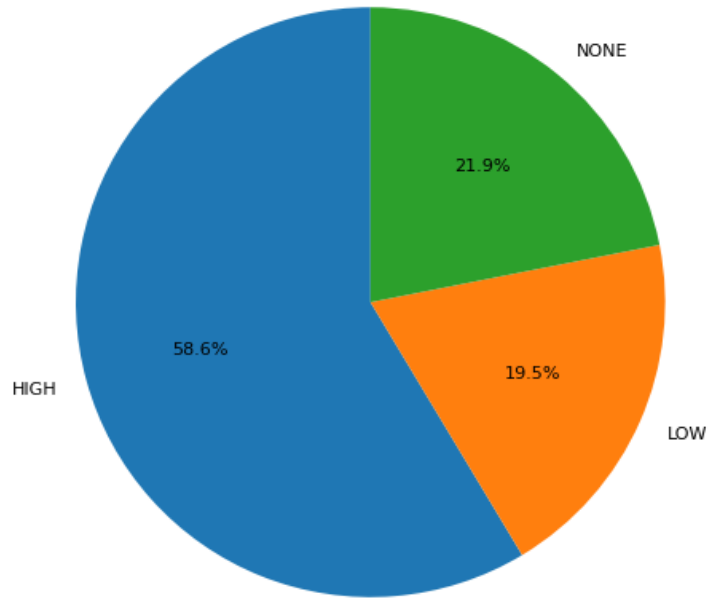




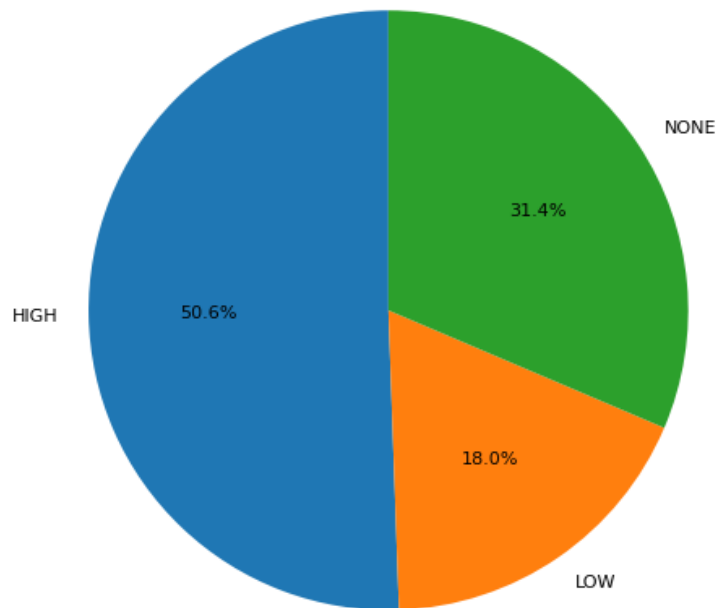
**Figure 0.14: Percentage of different classes data in User Interaction – CVSS V3**



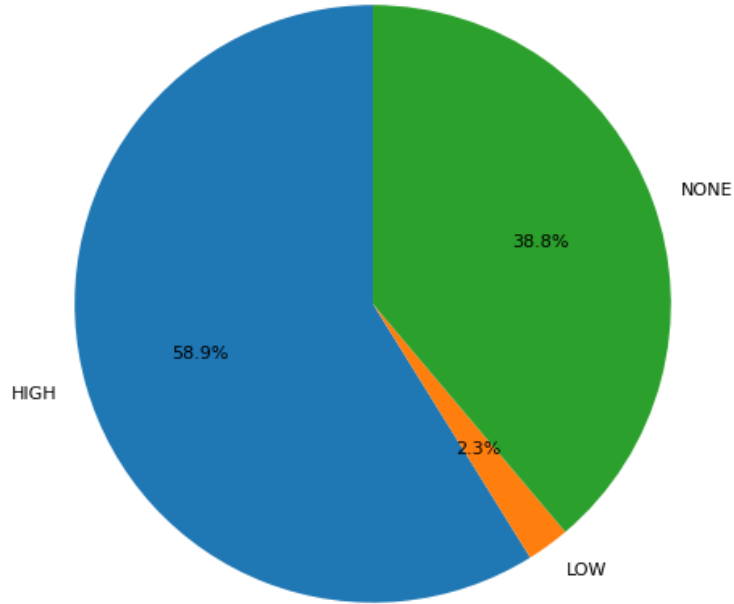
**Figure 0.15: Percentage of different classes data in Scope – CVSS V3**



**Figure 0.16: Percentage of different classes data in Confidentiality Impact – CVSS V3**



**Figure 0.17: Percentage of different classes data in Integrity – CVSS V3**



**Figure 0.18: Percentage of different classes data in Availability Impact – CVSS V3**

#### 4.2.3 Results and Comparison

In this research the main performance gauging indicator was the comparison of results of the work of [8] when performing CVSS score prediction for pure NVD dataset. For CVSS V2 and V3 score calculation, the accuracy achieved is more than [8] which can be seen in Table 0.2 and Table 0.3.

**Table 0.2: Comparison Table LR & RF for CVSS V2 NVD Only**

Attributes	Logistics Regression	Random Forest
Access Vector	95.09%	95.36%
Access Complexity	84.02%	85.12%
Authentication	93.92%	93.90%
Confidentiality Impact	82.45%	82.86%

<b>Integrity Impact</b>	84.43%	84.86%
<b>Availability Impact</b>	80.53%	81.71%

**Table 0.3: Comparison Table LR & RF for CVSS V3 NVD Only**

<b>Attributes</b>	<b>Logistics Regression</b>	<b>Random Forest</b>
<b>Attack Vector</b>	90.36%	90.47%
<b>Attack Complexity</b>	95.31%	96.01%
<b>Privileges Required</b>	85.77%	85.82%
<b>User Interaction</b>	92.11%	92.62%
<b>Scope</b>	96.29%	96.16%
<b>Confidentiality Impact</b>	86.67%	86.81%
<b>Integrity Impact</b>	87.45%	87.47%
<b>Availability Impact</b>	89.18%	89.38%

**To obtain the highest accuracy, a range of parameters were given to the model and resultantly best accuracy is acquired. Details of the parameters and its effect on accuracy is as shown in Table 0.4for CVSS V2 pure NVD Dataset,**

**Table 0.5 for CVSS V3 pure NVD Dataset, Table 0.6for CVSS V2 DC corelated dataset and**

Table 0.7 for CVSS V3 DC corelated dataset.

**Table 0.4: Comparison Table RF with different parameters for CVSS V2 NVD Only**

<b>n_estimators</b>	<b>50</b>	<b>60</b>	<b>70</b>	<b>80</b>	<b>90</b>	<b>100</b>
<b>AV</b>	92.08%	93.71%	92.11%	94.55%	94.62%	95.36%
<b>AC</b>	84.21%	84.55%	84.31%	85.27%	85.63%	85.12%
<b>Au</b>	90.33%	91.73%	91.01%-	92.89%	92.68%	93.90%
<b>C</b>	80.37%	80.74%	81.03%	83.06%	81.54%	82.86%
<b>I</b>	83.23%	83.88%	84.02%	84.41%	83.32%	84.86%
<b>A</b>	80.81%	81.28%	81.02%	81.72%	81.08%	81.71%
<b>Acc</b>	82.65%	83.05%	83.47%	85.84%	88.24%	89.09%
	<b>110</b>	<b>120</b>	<b>130</b>	<b>140</b>	<b>150</b>	
<b>AV</b>	94.16%	94.28%	94.28%	94.46%	94.44%	
<b>AC</b>	84.42%	85.41%	85.23%	85.11%	85.15%	
<b>Au</b>	93.44%	93.44%	92.66%	92.87%	92.99%	
<b>C</b>	81.63%	81.63%	82.10%	82.44%	82.97%	
<b>I</b>	83.88%	83.88%	84.12%	84.23%	84.52%	

<b>A</b>	80.97%	80.97%	81.56%	81.80%	81.85%
<b>Acc</b>	88.36%	88.36%	86.89%	85.17%	85.67%

**Table 0.5: Comparison Table RF with different parameters for CVSS V3 NVD Only**

<b>n_estimators</b>	<b>50</b>	<b>60</b>	<b>70</b>	<b>80</b>	<b>90</b>	<b>100</b>
<b>AV</b>	88.41%	89.21%	89.44%	89.91%	90%	90.47%
<b>AC</b>	92.75%	94.94%	95.04%	95.23%	95.74%	96.01%
<b>PR</b>	80.58%	82.96%	83.22%	83.89%	84.60%	85.82%
<b>UI</b>	90.87%	91.77%	92.13%	92.23%	92.46%	92.62%
<b>S</b>	94.79%	95.03%	95.31%	95.68%	95.97%	96.16%
<b>C</b>	84.11%	85%	85.14%	85.36%	85.58%	86.81%
<b>I</b>	84.14%	86.27%	86.56%	86.97%	87.08%	87.47%
<b>A</b>	86.01%	88.54%	88.89%	89.04%	89.26%	89.38%
<b>Acc</b>	87%	89%	89.17%	89.88%	90%	90.45%
	<b>110</b>	<b>120</b>	<b>130</b>	<b>140</b>	<b>150</b>	
<b>AV</b>	90.21%	90.08%	89.83%	89.49%	89.12%	
<b>AC</b>	95.44%	95%	94.97%	94.63%	93.86%	

<b>PR</b>	84.32%	84.13%	83.98%	83.54%	83.23%
<b>UI</b>	93.27%	93.04%	92.59%	92.34%	92.17%
<b>S</b>	95.42%	95.33%	95.01%	94.96%	94.67%
<b>C</b>	85.47%	85.13%	85%	84.88%	83.41%
<b>I</b>	88.42%	88.18%	87.63%	87.42%	86.32%
<b>A</b>	87.65%	87.05%	86.86%	86.55%	84.94%
<b>Acc</b>	90.33%	89.65%	89.32%	89.07%	88.56%

**Table 0.6: Comparison Table RF with different parameters for CVSS V2DC Data**

<b>n_estimators</b>	<b>AV</b>	<b>AC</b>	<b>Au</b>	<b>C</b>	<b>I</b>	<b>A</b>	<b>Accuracy</b>
<b>20</b>	96.12%	89.32%	90.45%	81.65%	80.55%	77.12%	77.11%
<b>30</b>	96.45%	89.67%	90.88%	81.65%	80.55%	77.12%	77.11%
<b>40</b>	96.33%	89.77%	90.89%	81.82%	80.68%	77.27%	77.27%
<b>50</b>	96.43%	89.86%	90.90%	82.95%	81.82%	78.41%	81.82%
<b>60</b>	96.59%	89.77%	90.91%	80.68%	79.55%	77.27%	95.45%
<b>70</b>	96.36%	89.63%	90.65%	81.56%	80.65%	77.26%	80.22%
<b>80</b>	96.27%	89.36%	90.32%	81.87%	80.65%	77.34%	77.77%
<b>90</b>	96.13%	89.02%	90%	81.03%	80.23%	77.12%	76.56%

<b>100</b>	95.45%	89.77%	90.91%	80.68%	81.82%	73.86%	75.45%
------------	--------	--------	--------	--------	--------	--------	--------

**Table 0.7: Comparison Table RF with different parameters for CVSS V3DC Data**

<b>n_esti</b>	<b>AV</b>	<b>AC</b>	<b>PR</b>	<b>UI</b>	<b>S</b>	<b>C</b>	<b>I</b>	<b>A</b>	<b>Accuracy</b>
<b>20</b>	92%	94%	95.6 %	77.56 %	88.22 %	77.35%	81.21%	90.12%	88%
<b>30</b>	92%	94%	96%	78%	88.69 %	78.02%	82%	90.50%	90%
<b>40</b>	92%	94%	96%	78%	90%	78%	82%	90%	93.33%
<b>50</b>	92.36 %	94.45 %	96.12 %	78.23 %	90%	78.14%	82.32%	90.02%	94.66%
<b>60</b>	96%	95%	96.33 %	79%	90.23 %	78.64%	82.66%	91%	95.36%
<b>70</b>	92.12 %	94.32 %	96%	78.12 %	89.66 %	78%	82.21%	90%	94.03%
<b>80</b>	92%	94%	95.65 %	78%	89.43 %	78%	82.04%	88.36%	93%
<b>90</b>	92%	94%	95.5%	77.3 %	89%	77.3%	82.3%	88.6%	89.24%



<b>100</b>	91.54 %	93%	95.3%	77%	89%	77%	81%	88%	88%
------------	------------	-----	-------	-----	-----	-----	-----	-----	-----

Results of CVSS score prediction for V2 for correlated data of DC dataset is shown in Table 0.8 and for V3 is shown in Table 0.9

**Table 0.8: RF for CVSS V2 DC correlated data**

<b>Attributes</b>	<b>Random Forest</b>
<b>Access Vector</b>	95.59%
<b>Access Complexity</b>	89.77%
<b>Authentication</b>	90.91%
<b>Confidentiality Impact</b>	80.68%
<b>Integrity Impact</b>	79.55%
<b>Availability Impact</b>	77.27%
<b>Validation</b>	95.45%

**Table 0.9: RF for CVSS V3 DC correlated data**

<b>Attributes</b>	<b>Random Forest</b>
-------------------	----------------------

<b>Attack Vector</b>	96%
<b>Attack Complexity</b>	95%
<b>Privileges Required</b>	96.33%
<b>User Interaction</b>	79%
<b>Scope</b>	90.23%
<b>Confidentiality Impact</b>	78.64%
<b>Integrity Impact</b>	82.66%
<b>Availability Impact</b>	91%
<b>Validation</b>	95.36

#### 4.2.4 Performance Study

To conduct performance analysis of the classification and prediction performed by the model, in this research, precision, recall, F1 Score and Support for each class is calculated using libraries of Machine Learning. These scores are shown in Table 0.10 and

Table 0.11.11 for pure NVD dataset CVSS V2, V3, and Table 0.12.12,

Table 0.13.13 for DC data CVSS V2, V3, where the best accuracy is achieved.

**Table 0.10: Evaluation metrics for pure NVD dataset CVSS V2**

<b>Attributes</b>	<b>Precision</b>	<b>Recall</b>	<b>F Score</b>	<b>Support</b>
<b>AV</b>	93.69%	73.67%	80.77%	14265

<b>AC</b>	87.46%	64.17%	68.86%	14265
<b>Au</b>	61.19%	50.32%	53.89%	14265
<b>C</b>	84.60%	76.64%	79.33%	14265
<b>I</b>	84.91%	78.32%	80.76%	14265
<b>A</b>	81.82%	78.34%	79.45%	14265

**Table 0.11: Evaluation metrics for pure NVD dataset CVSS V3**

<b>Attributes</b>	<b>Precision</b>	<b>Recall</b>	<b>F Score</b>	<b>Support</b>
<b>AV</b>	91.34%	61.11%	69.83%	7740
<b>AC</b>	94.46%	72.06%	78.97%	15480
<b>PR</b>	86.33%	70.41%	75.67%	10320
<b>UI</b>	93.48%	90.40%	91.65%	15480
<b>S</b>	96.79%	89.52%	92.67%	15480
<b>C</b>	89.71%	79.12%	83.12%	10320
<b>I</b>	89.38%	84.65%	86.62%	10320
<b>A</b>	90.20%	70.11%	74.87%	10320

**Table 0.12: Evaluation metrics for DC dataset CVSS V2**

<b>Attributes</b>
-------------------

<b>AV</b>	65.38%	61.11%	62.98%	29.333
<b>AC</b>	62.92%	50.00%	53.57%	29.333
<b>Au</b>	60.82%	54.43%	56.79%	29.333
<b>C</b>	78.45%	76.76%	77.14%	29.333
<b>I</b>	76.96%	70.08%	72.78%	29.333
<b>A</b>	81.04%	69.01%	71.38%	29.333

**Table 0.13: Evaluation metrics for pure DC dataset CVSS V3**

<b>Attributes</b>	<b>Precision</b>	<b>Recall</b>	<b>F Score</b>	<b>Support</b>
<b>AV</b>	97.04%	69.05%	78.25%	16.667
<b>AC</b>	47.00%	50.00%	48.45%	25
<b>PR</b>	55.88%	54.67%	52.92%	16.667
<b>UI</b>	97.83%	83.33%	88.89%	25
<b>S</b>	94.68%	68.75%	74.46%	25
<b>C</b>	73.18%	65.50%	67.43%	16.667
<b>I</b>	81.67%	74.54%	73.97%	16.667
<b>A</b>	93.27%	90.52%	91.52%	16.667

**The data acquired in**

**Table 0.11-**

Table 0.13 is further used to calculate F1 Score. Similarly, RR is calculated and further used to calculate MRR. In addition to that, balanced accuracy score is also calculated. All these calculations of F1 Score, RR, MRR and Balanced accuracy score are shown in Table 0.14 and Table 0.15 for pure NVD data set CVSS V2, V3 and Table 0.16, Table 0.17 for DC related dataset CVSS V2, V3.

**Table 0.14: Calculation on Evaluation Metrics for pure NVD dataset CVSS V2**

Attributes	F1 Score	MRR	Balanced Accuracy
AV	82.48%	94.28%	73.67%
AC	74.03%	85.41%	64.17%
Au	55.22%	92.13%	50.32%
C	80.42%	82.37%	76.64%
I	81.48%	84.02%	78.32%
A	80.04%	81.48%	78.34%

**Table 0.15: Calculation on Evaluation Metrics for pure NVD dataset CVSS V3**

Attributes	F1 Score	MRR	Balanced Accuracy
AV	73.23%	90.47%	61.11%
AC	81.75%	96.01%	72.06%
PR	77.56%	84.42%	70.41%
UI	91.91%	92.62%	90.40%
S	93.02%	96.16%	89.52%
C	84.08%	86.01%	79.12%

<b>I</b>	86.95%	87.17%	84.65%
<b>A</b>	78.90%	89.38%	70.11%

**Table 0.16: Calculation on Evaluation Metrics for DC dataset CVSS V2**

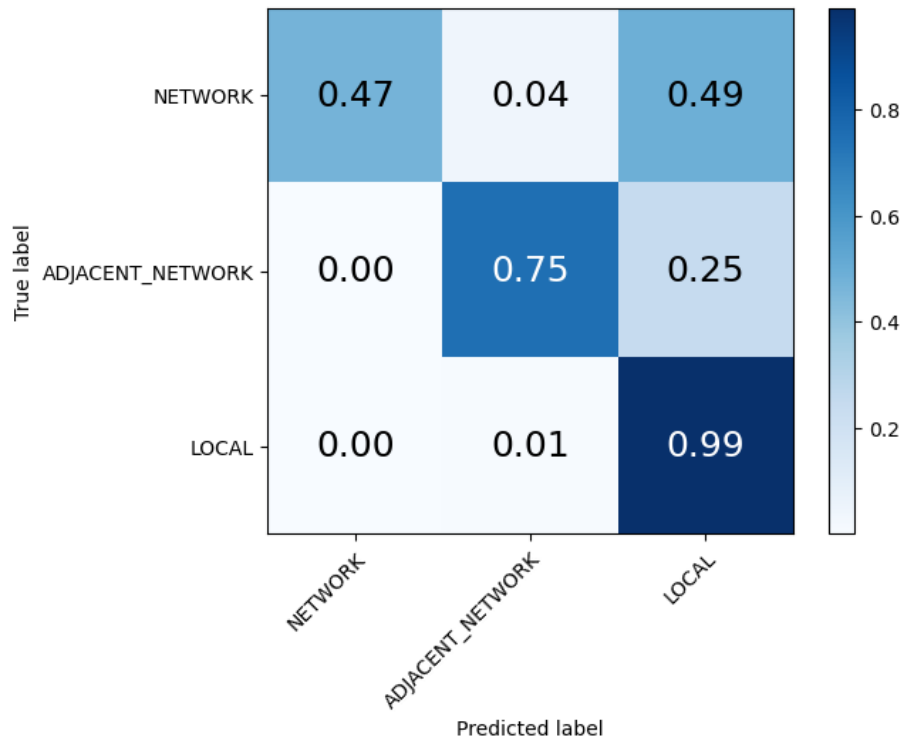
<b>Attributes</b>	<b>F1 Score</b>	<b>MRR</b>	<b>Balanced Accuracy</b>
<b>AV</b>	63.18%	96.59%	61.11%
<b>AC</b>	55.72%	89.77%	50.00%
<b>Au</b>	57.44%	90.91%	54.43%
<b>C</b>	77.60%	80.68%	76.76%
<b>I</b>	73.36%	79.55%	70.08%
<b>A</b>	74.54%	77.27%	69.01%

**Table 0.17: Calculation on Evaluation Metrics for DC dataset CVSS V3**

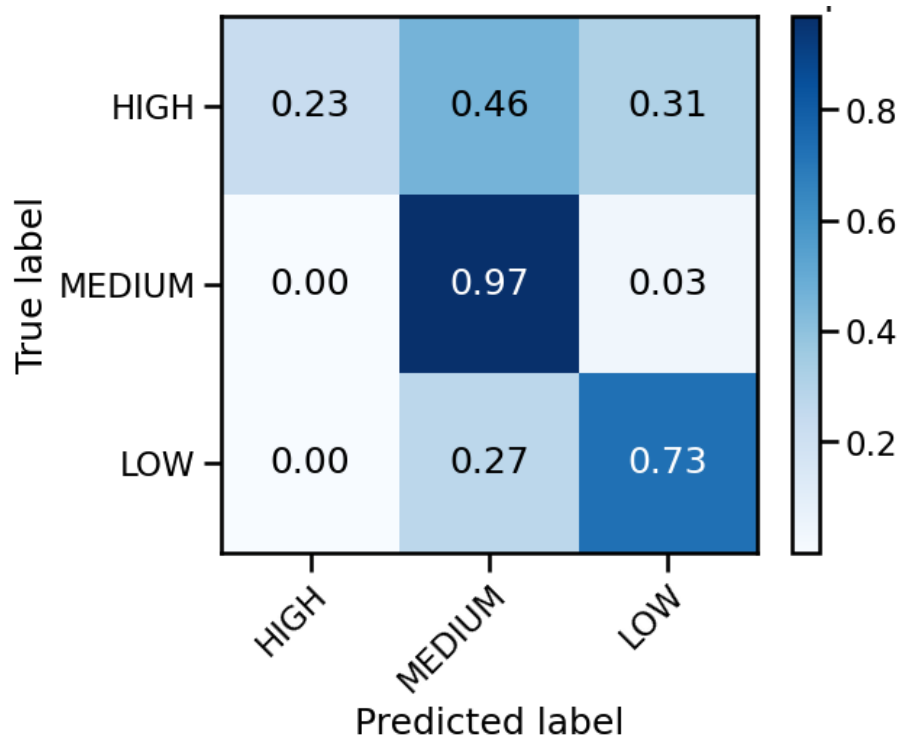
<b>Attributes</b>	<b>F1 Score</b>	<b>MRR</b>	<b>Balanced Accuracy</b>
<b>AV</b>	80.68%	92.00%	69.05%
<b>AC</b>	48.45%	94.00%	50.00%
<b>PR</b>	55.27%	78.00%	54.67%
<b>UI</b>	90.00%	96.00%	83.33%
<b>S</b>	79.66%	90.00%	68.75%
<b>C</b>	69.13%	78.00%	65.50%
<b>I</b>	77.94%	82.00%	74.54%

<b>A</b>	91.87%	90.00%	90.52%
----------	--------	--------	--------

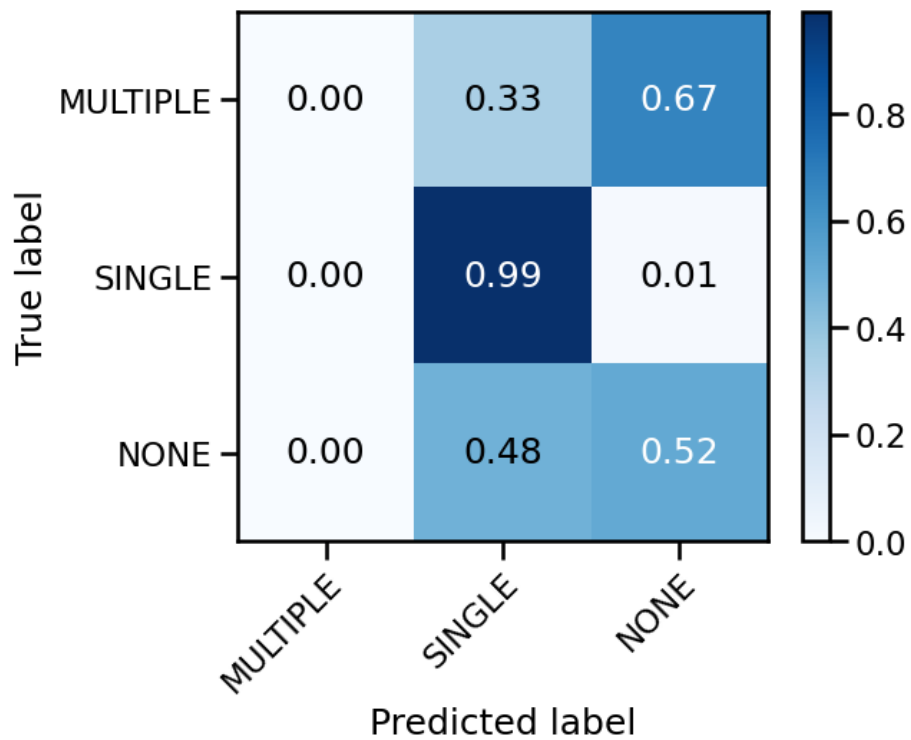
The computed confusion matrix is also generated for each label. The in-depth analysis of model's behavior on CVSS V2 is shown by Figure 0.19 to Figure 0.24.



**Figure 0.19: Confusion matrix for Access Vector – CVSS V2**

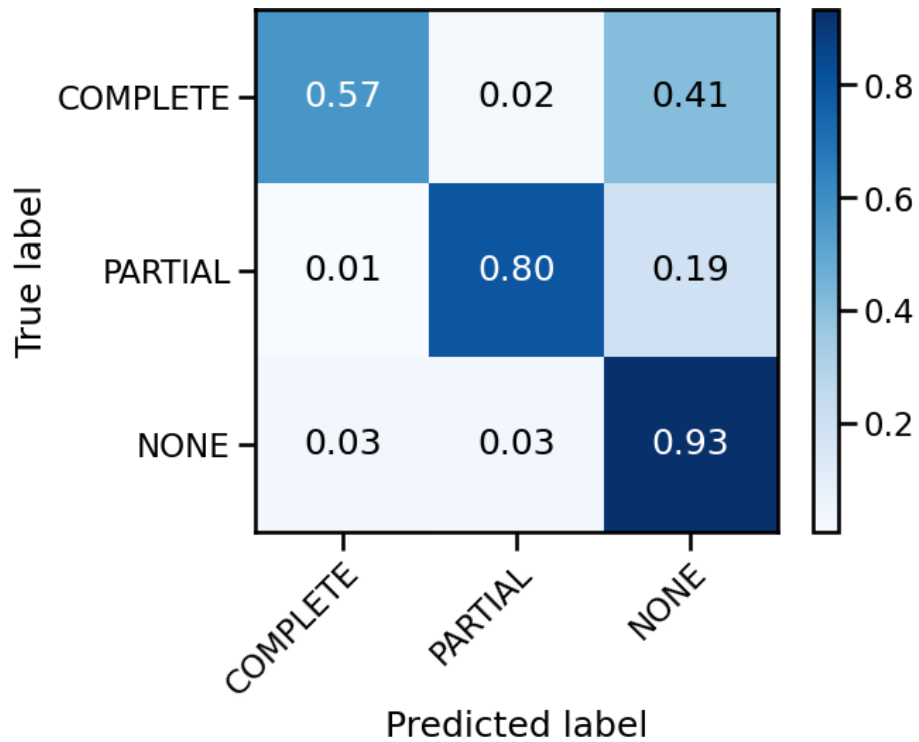


**Figure 0.20: Confusion matrix for Access Complexity – CVSS V2**

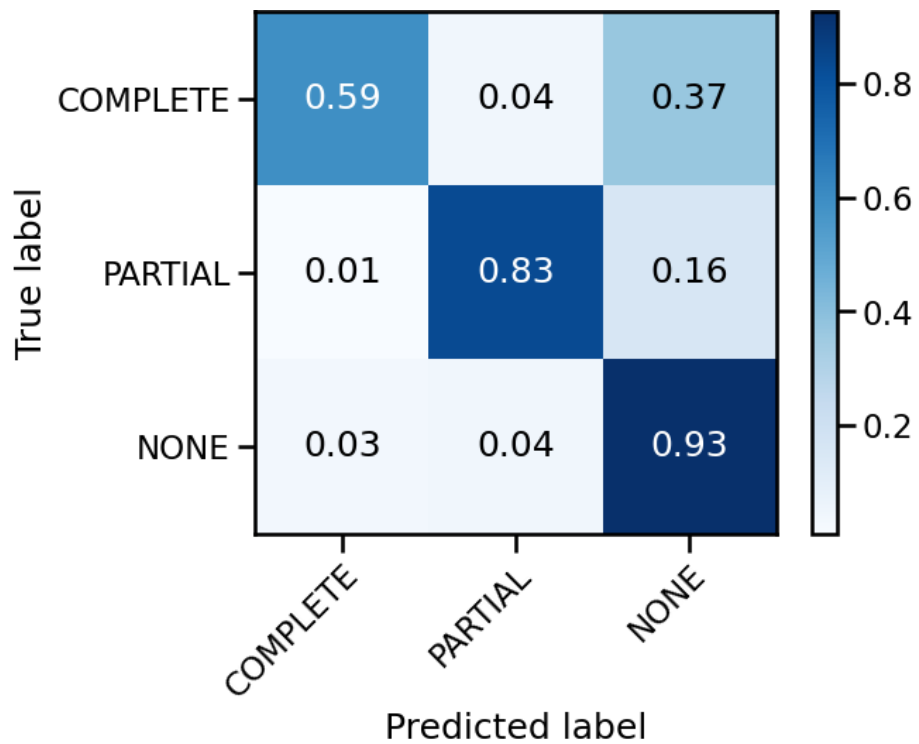


**Figure 0.21: Confusion matrix for Authentication – CVSS V2**

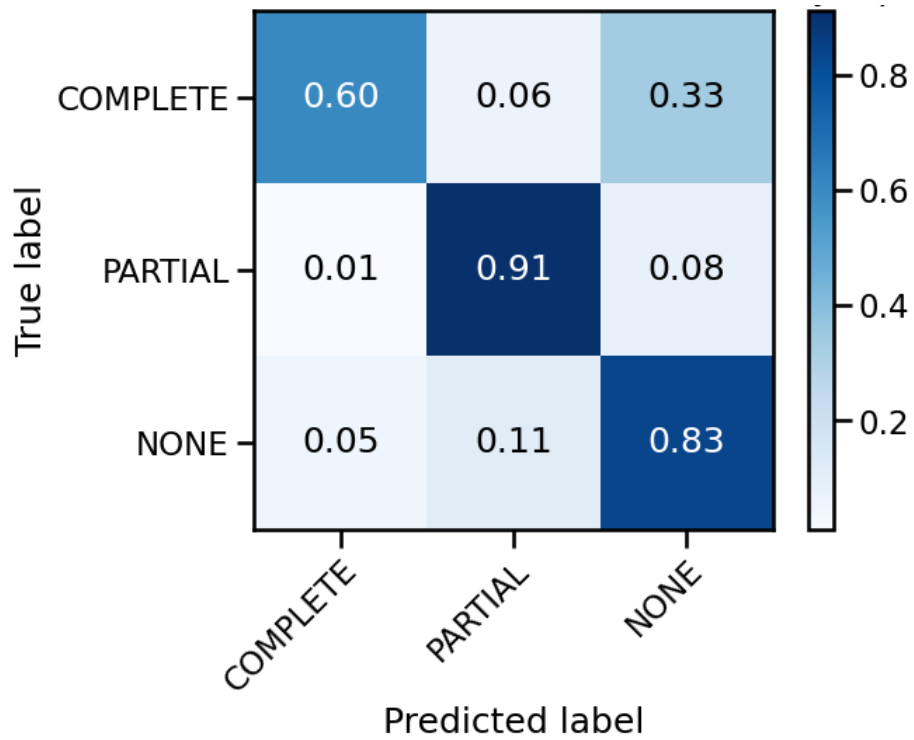




**Figure 0.22: Confusion matrix for Confidentiality – CVSS V2**

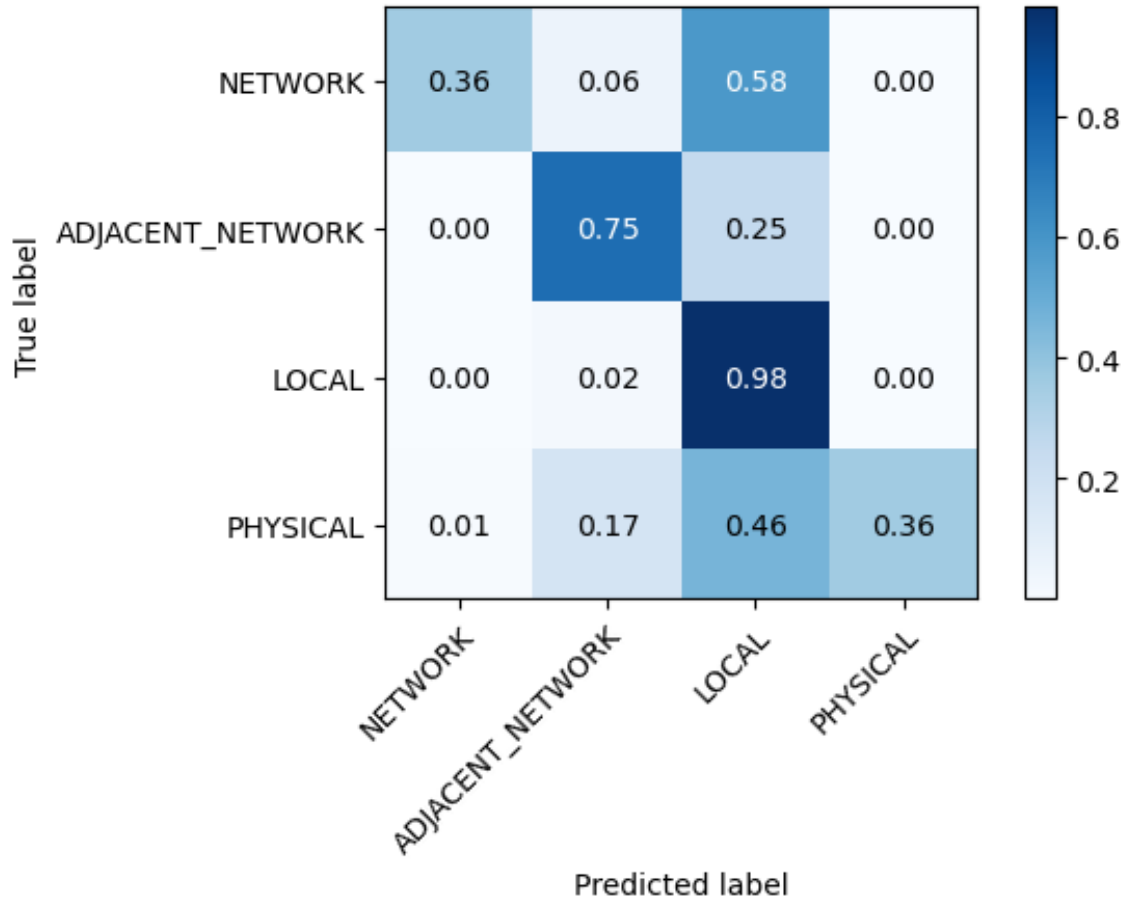


**Figure 0.23: Confusion matrix for Integrity – CVSS V2**

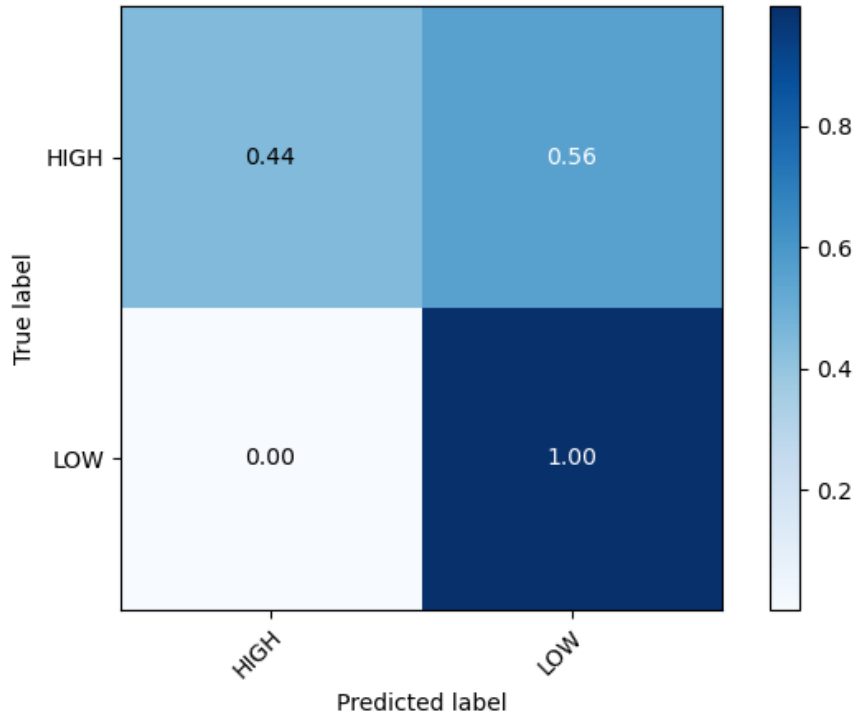


**Figure 0.24: Confusion matrix for Availability – CVSS V2**

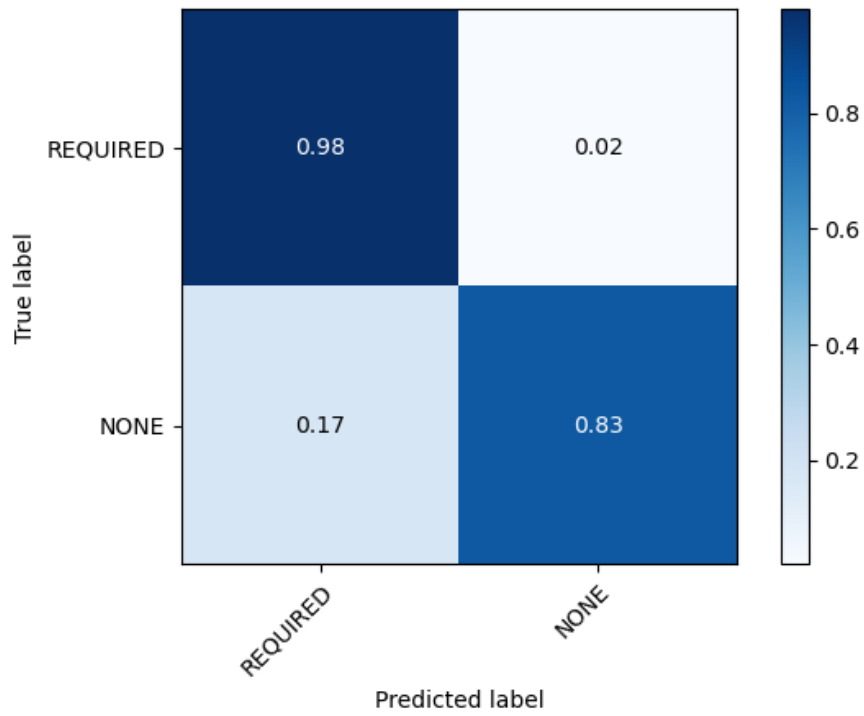
The Figure 0.25 from Figure 0.32 shows the in-depth analysis of model's behavior on CVSS V3.



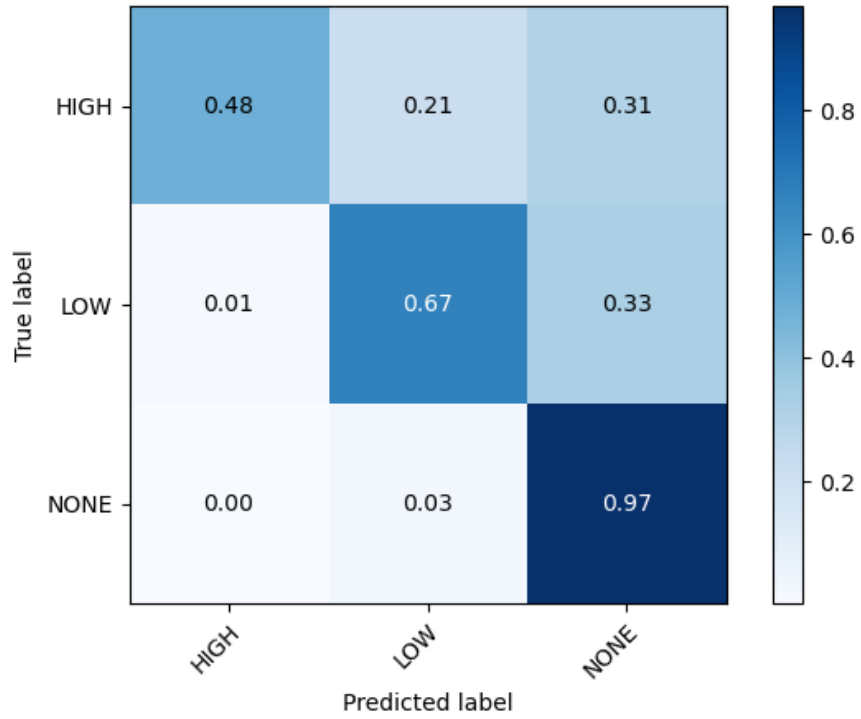
**Figure 0.25: Confusion matrix for Attack Vector – CVSS V3**



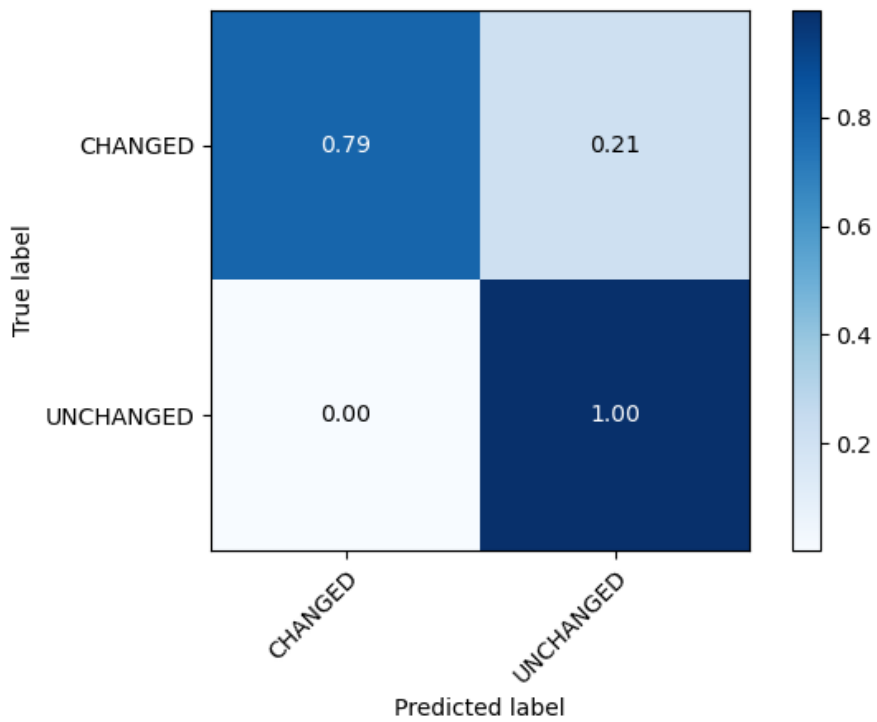
**Figure 0.26: Confusion matrix for Attack Complexity – CVSS V3**



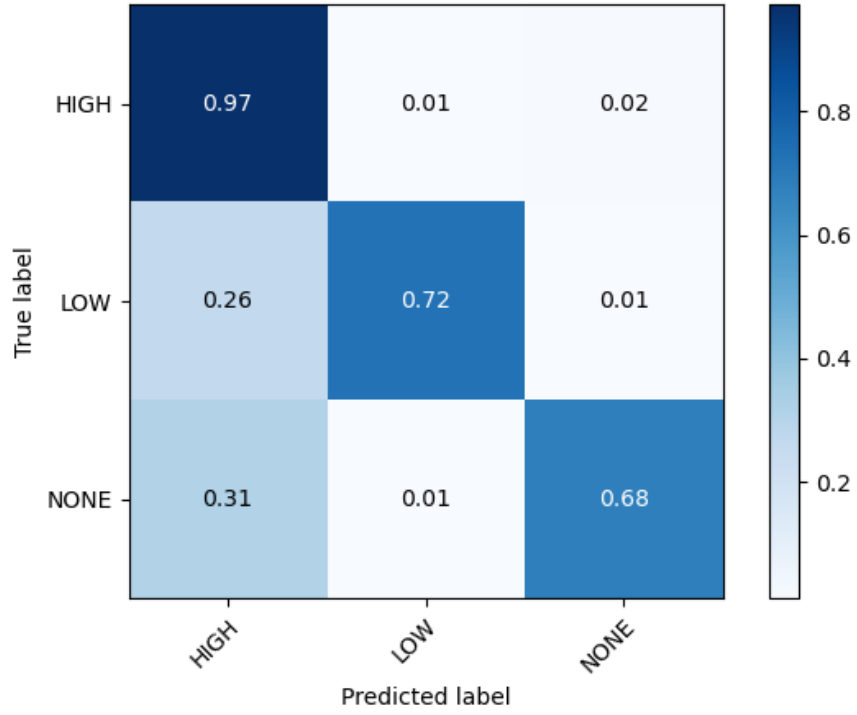
**Figure 0.27: Confusion matrix for User Interaction – CVSS V3**



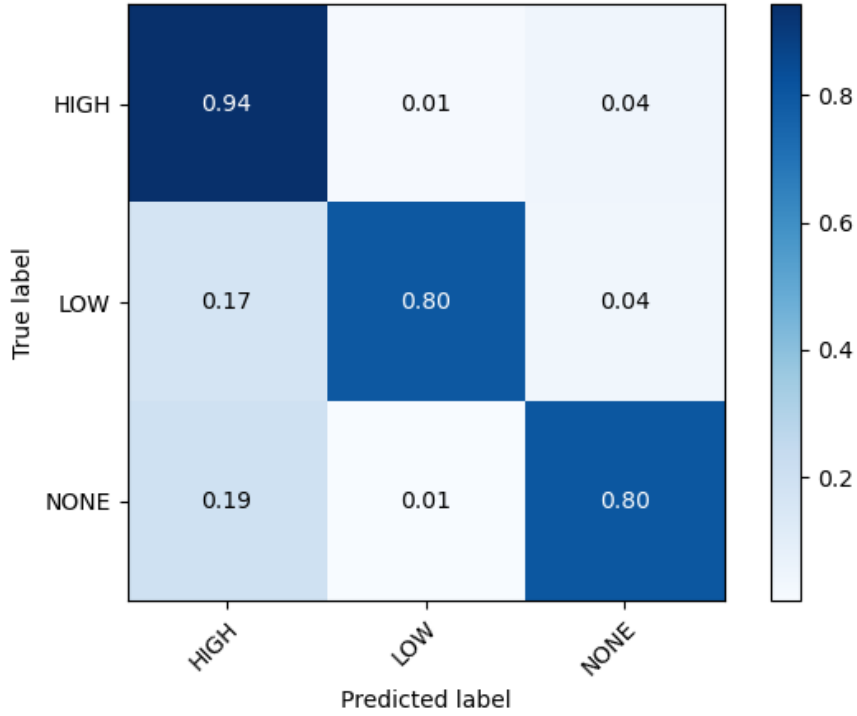
**Figure 0.28: Confusion matrix for Privileges Required – CVSS V3**



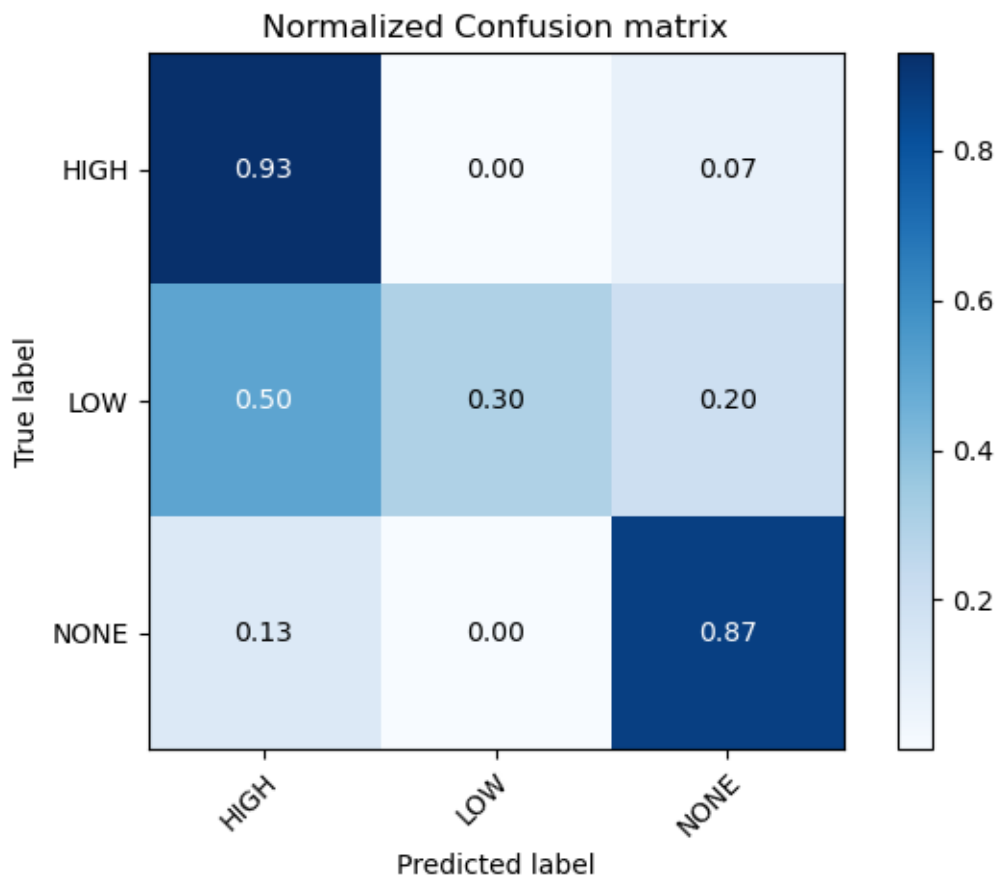
**Figure 0.29: Confusion matrix for Scope– CVSS V3**



**Figure 0.30: Confusion matrix for Confidentiality Impact – CVSS V3**



**Figure 0.31: Confusion matrix for Integrity Impact – CVSS V3**



**Figure 0.32: Confusion matrix for Availability Impact – CVSS V3**

### 4.3 Analysis

The Random Forest's tree count may be adjusted using the `n_estimators` argument. Up to a certain point, a more stable and precise model is often produced by increasing the number of trees. But there are decreasing gains, and overfitting and higher computing costs might result from using too many trees. In this research, experiments were conducted to examine the effect of changing the number of trees in the RF model. We adjusted the `n_estimators` parameter using the grid-search strategy, which involves training and assessing our model on varying numbers of trees and use the model's hyperparameters to get the best accuracy for our set of data. The specifications of the RF model are presented in detail Table 4.1. Results with different hyperparameters for pure NVD and DC case study are shown in Tables 4.4, 4.5, 4.6 and 4.7. It can be observed

that, n\_estimators ranging from 50 to 150 for NVD and 20 to 100 for DC are used and are giving different results. For CVSS V2, the model attained the highest accuracy of 89.09% for pure NVD dataset at n\_estimators 100 and 95.45% for DC correlated data at n\_estimators 60. Whereas for CVSS V3, the highest accuracy of 90.45% is achieved for pure NVD dataset at n\_estimators 100 and 95.36% for DC related data at n\_estimators 60. For the DC case study, the correlated data is gathered from multiple sources and arithmetic mean is taken of various sources score to improve prediction.

For pure NVD data, LR of [8] gives 84.02% accuracy for label “Access Complexity” whereas RF is giving 85.12% accuracy for the same. Similarly, [8] has “Availability Impact” with accuracy 80.53% while this study calculated it at 81.71%. Overall accuracy of [8] is 85.69% for CVSS V2, which is lesser than that of RF which is 89.09%.



# RECOMMENDATIONS FOR MITIGATION

A good amount of breakdown was presented in Chapter 3 regarding the mechanism of vulnerability scoring, data gathering and correlating, evaluation metrics, ML model and its algorithm. Analysis with the help of comparison tables of results, parameters and figures depicting proportions of class imbalances and confusion matrix representing accuracy attained by the model in calculating CVSS scores were shown in chapter 4. After obtaining the ratings, one of the most difficult tasks will be to mitigate the vulnerabilities.

Specifically, determining the severity of a specific vulnerability—often with the help of the CVSS—is a typical method of determining the order in which vulnerability mitigation should be prioritized. Organizations that store or process credit cards are required by the payment card industry data security standard (PCI-DSS) to remediate vulnerabilities greater than CVSS four. Additionally, back in 2019, the Department of Homeland Security published an official operational orders directing federal departments to fix high and critical vulnerabilities conferring to CVSS standard [36].

In this chapter, an in-depth analysis of Known Exploited Vulnerabilities (KEV) catalog is carried out which is issued by Cyber and Information Security Agency (CISA). CISA maintains the official source of vulnerabilities that have been exploited in the wild for the usage of network administrators and cybersecurity professionals, as well as to assist every company in better managing vulnerabilities and staying up to date with threat activities [50]. CISA advice that KEV catalog should be incorporated by organizations into their vulnerability management priority system [50]. Experiments have been conducted using the KEV catalog to analysis the trends and patterns in the dataset of exploited vulnerabilities and keeping in view the analysis recommendations are outlined to mitigate the vulnerabilities.

## 5.1 Known Exploited Vulnerabilities (KEV)

KEV is a 10 labels-based catalog, which is maintained by CISA has by the end of year 2023, 1055 entries. The catalog is available on website in 2 formats CSV and JSON (CISA provides JSON Schema too). The catalog has different labels which are shown inTable 0.1.

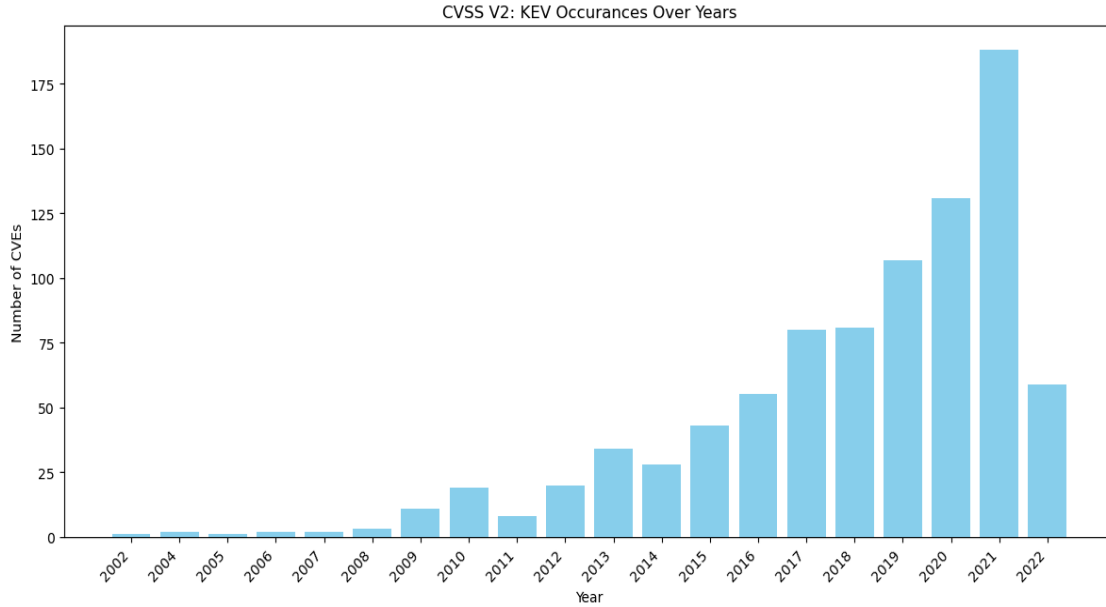
**Table 0.1: CISA catalog format**

Lables				
Cve ID	Vendor Project	Product	Vulnerability Name	Date Added
Short Description	Required Action	Due Date	Known Ransomware Campaign Use	Notes

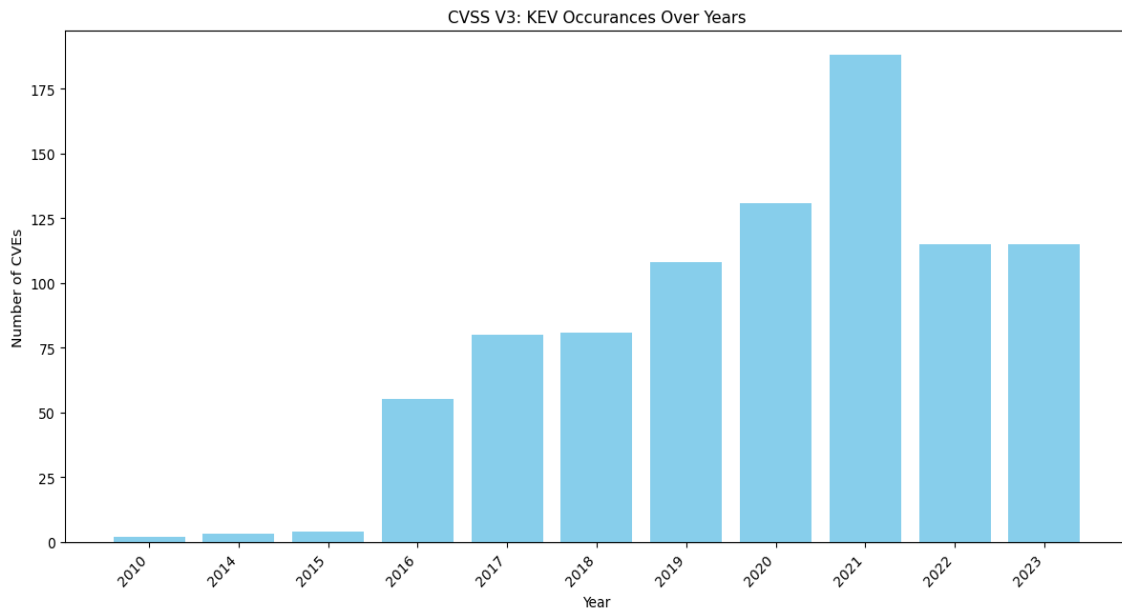
## 5.2 Analysis for Patterns

In this research, an in-depth analysis of KEV is carried out to observe any pattern in KEV keeping CVSS score as base parameter. To do so, the catalog was appended with another label of CVSS score fetched from pure NVD dataset and a year label was included which was extracted from CVE\_ID label.

To look at the pattern and trend of vulnerabilities exploitation, analysis is done with the help of chart showing trend year wise in Figure 0.1 and Figure 0.2.By looking at both charts we can see the increasing trend of vulnerabilities exploitation year wise.



**Figure 0.1: Year wise occurances of CVSS V2 vulnerabilities**

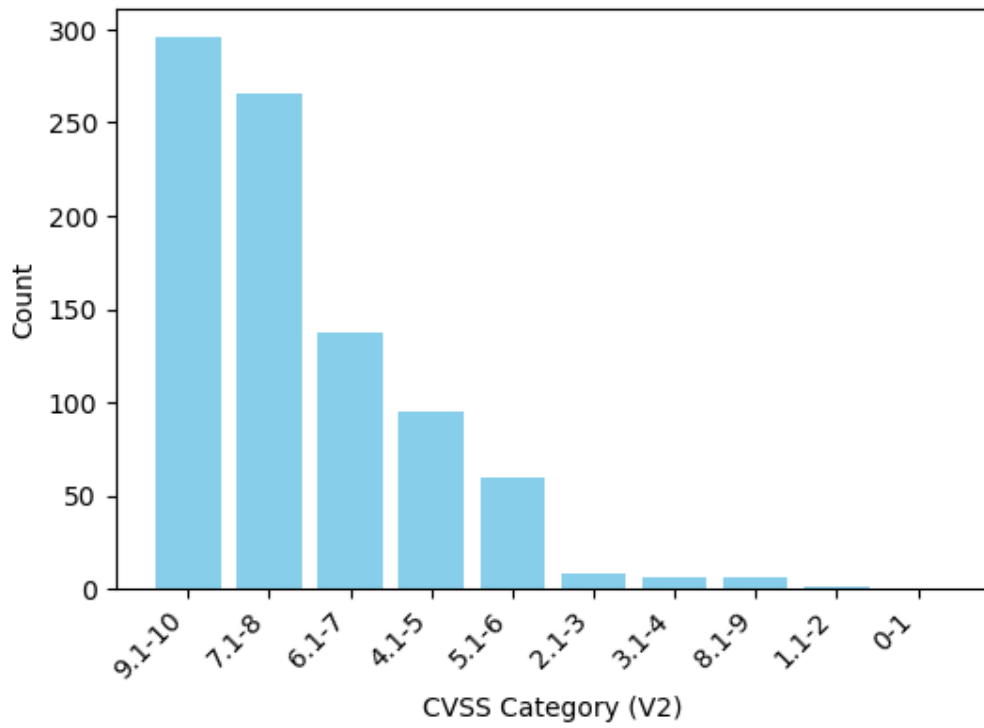


**Figure 0.2: Year wise occurances of CVSS V3 vulnerabilities**

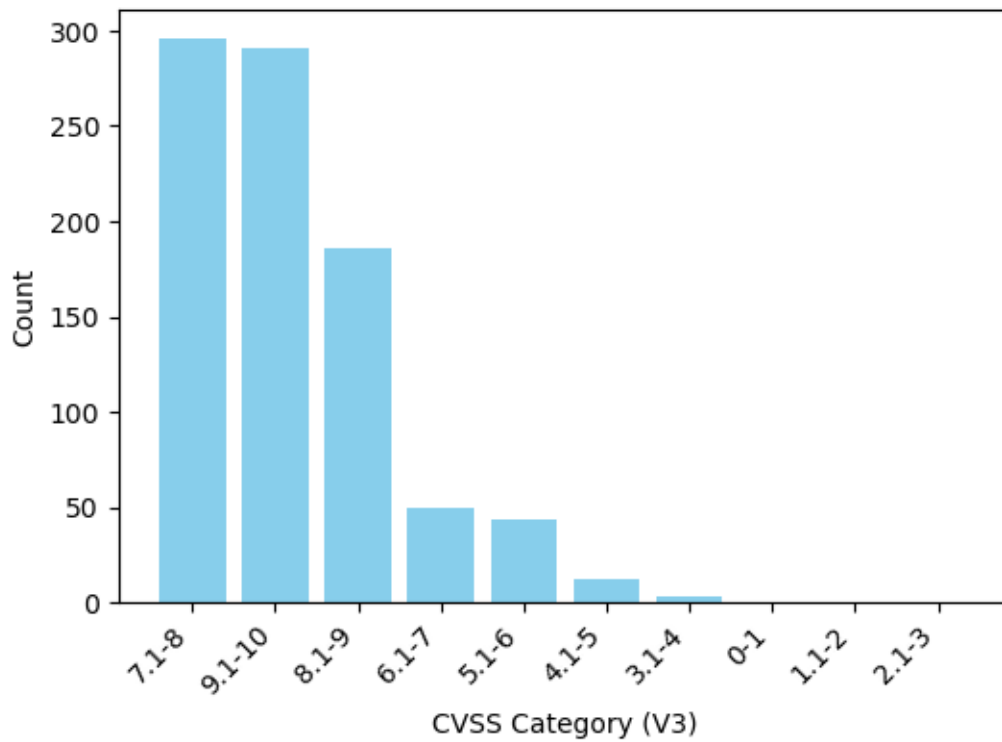
To further analyse to find out which CVSS score vulnerabilities were more exploited count of CVSS V2 and V3 vulnerabilities was done. Table 0.2 shows the count of occurrences grouped by the category of CVSS scores they fall into and Figure 0.3 and Figure 0.4 shows the graphical representation.

**Table 0.2: Occurrences CVSS score range-wise**

<b>Category ranges CVSS Score</b>	<b>Count of CVSS V2 entries</b>	<b>Count of CVSS V3 entries</b>
<b>9.1-10</b>	296	291
<b>8.1-9</b>	6	186
<b>7.1-8</b>	266	296
<b>6.1-7</b>	137	50
<b>5.1-6</b>	60	44
<b>4.1-5</b>	95	12
<b>3.1-4</b>	6	3
<b>2.1-3</b>	8	0
<b>1.1-2</b>	1	0
<b>0-1</b>	0	0

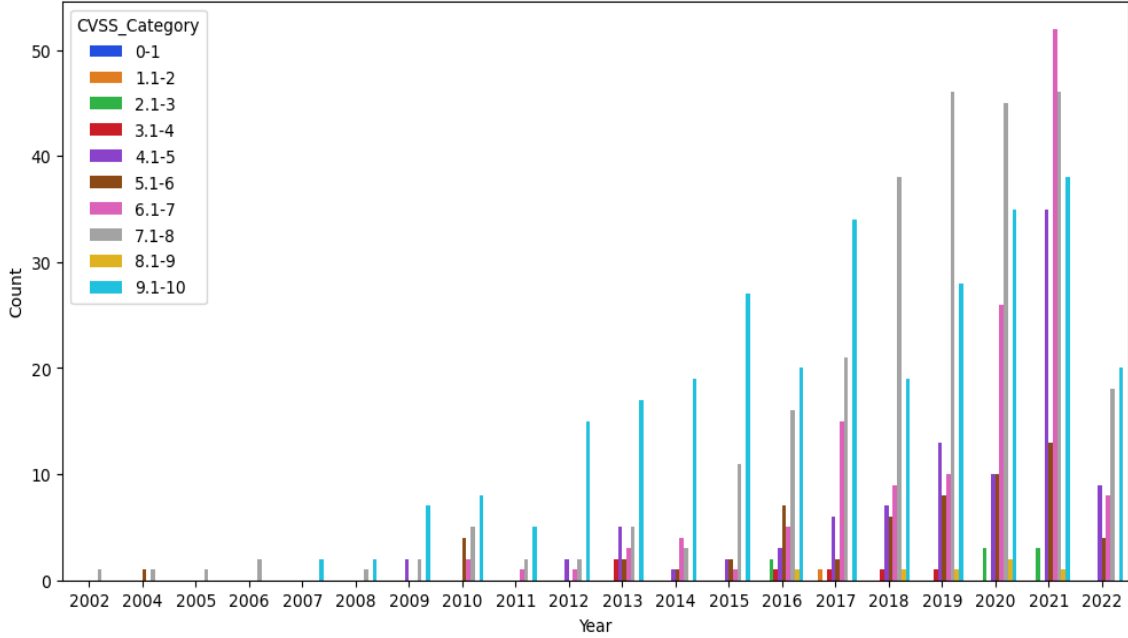


**Figure 0.3: KEV entries CVSS score wise V2**

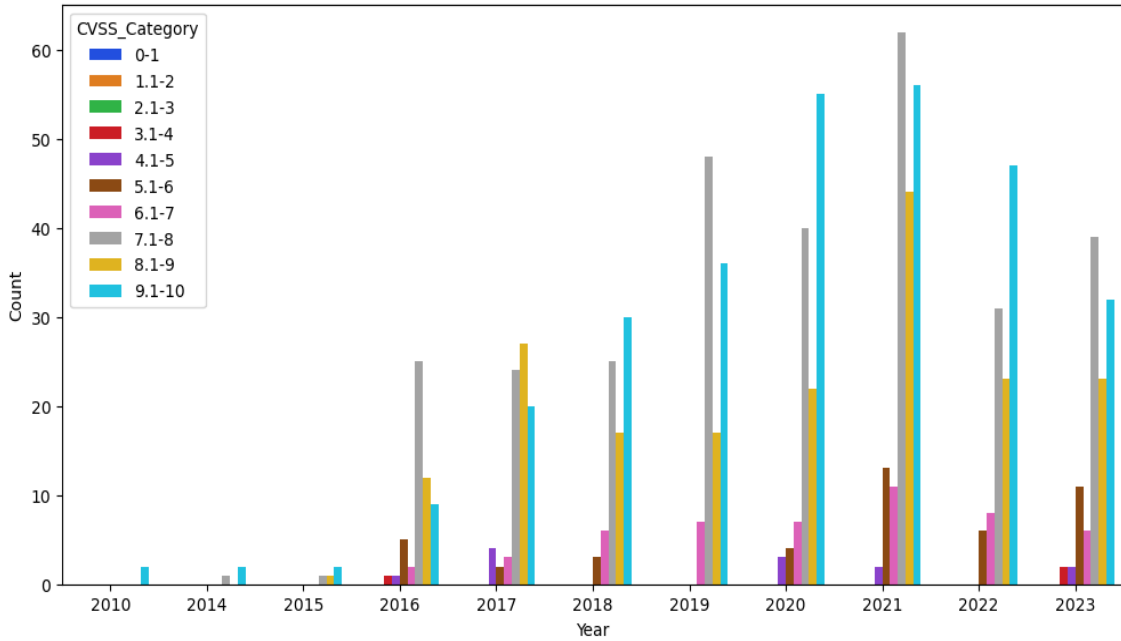


**Figure 0.4: KEV entries CVSS score wise V3**

The data set was further examined for the exploitation range of the score wise instances occurred each year. Figure 0.5 and Figure 0.6 are depicting occurrence of vulnerability exploitation in each year and with their corresponding range scale of CVSS.



**Figure 0.5: KEV entries CVSS score category wise V2**



**Figure 0.6: KEV entries CVSS score category wise V3**

### 5.3 Recommendations

By doing in-depth analysis of the data represented in section 5.2, at first, vulnerabilities of score 6.1-8, 9.1-10 are exploited on the higher side and vulnerabilities lying between 4.1 to 6 are second highest to be exploited. For CVSS V3 the score range 8.1-9 is also on the higher side but for CVSS V2, 8.1-9 are exploited on the lower side.

By looking at the statistics, recommended priorities to fix the vulnerabilities are displayed in Table 0.3

**Table 0.3: Priority of Vulnerability mitigation**

Score range	Priority
6.1-10	First
4.1-6	Second
0-4	Rest

# CONCLUSION AND FUTURE WORKS

Identifying and evaluating vulnerabilities is an essential and challenging task. To evaluate the severity of a stated vulnerability instance, it is suggested in this research to improve the effectiveness of vulnerability severity scoring systems that adhere to CVSS standards. This research approach reduces potential delays in calculation of severity scores by implying a machine-learning model, which is trained using suitable vulnerability instances as ground truth, which acts as a basis for scoring. When compared to similar studies, the performance of the suggested model demonstrates high precision as well as micro F1-score thresholds, resultantly higher producing accuracy.

To verify the suggested vulnerability assessment model, a case study that involve Data center vulnerability observations from multiple repositories is presented. The case study is conducted keeping in view the conflicts arising from different CVSS mechanisms and addresses erratic vulnerability severity scores contributed by different cybersecurity analysers. The arithmetic mean method was used to determine the score of incoherent indications for identical vulnerabilities in various cybersecurity repositories.

The case study's results further demonstrate that vulnerability scoring differ depending on the cybersecurity data sources utilized, which could possibly distort cybersecurity decisions about patch prioritization and funding. Therefore, to increase cybersecurity awareness even further, an analysis of vulnerabilities approach which correlates various sources of information is required.

In addition to that, an in-depth analysis of CISA's KEV is done to suggest which vulnerabilities scores should ne prioritize in mitigation of vulnerabilities according to scores.Exploring the application of Deep learning-based models for predicting CVSS scores and the types of threats associated with each vulnerability will enable the proposed research to be investigated further. By applying computational intelligence techniques, such startup settings could be dynamically adjusted in addition to being provided by security experts.



To assess the dependability of the scores obtained from various sources, arithmetic means of multiple scores from more sources could be another potential future pathway. Subsequently, this research has been planned to keep in mind investigating the importance of vulnerabilities and their mitigation parameters in general and specifically for data center environment.

In addition to that, further study of CISA KEV entries with CVSS scores and their relevant labels for different versions including AV, AC, Au, PR, S, C, I, A and their sub-classes can be carried out to analyse the relationship and impact between scores and classes.

## REFERENCES

- [1] R. Sharma, R. Sibal, and S. Sabharwal, “Software vulnerability prioritization using vulnerability description,” *International Journal of System Assurance Engineering and Management*, vol. 12, no. 1, pp. 58–64, Jul. 2020, doi: <https://doi.org/10.1007/s13198-020-01021-7>.
- [2] K. A. Saed, N. Aziz, A. W. Ramadhani, and N. H. Hassan, “Data Governance Cloud Security Assessment at Data Center,” *IEEE Xplore*, Aug. 2018, doi: <https://doi.org/10.1109/ICCOINS.2018.8510612>.
- [3] J. S. Suroso, A. Sutikno, F. G. Br. Ginting, and N. Angelica, “Risk Management & Mitigation Plan for Data Center Environment,” <https://www.ijtre.org>, Mar. 2020. <https://www.ijrte.org/wp-content/uploads/papers/v8i6/F7656038620.pdf>
- [4] Víctor Mayoral Vilches, Juan, Bernhard Dieber, Unai Ayucar Carbajo, and Endika Gil-Uriarte, “Introducing the Robot Vulnerability Database (RVD),” *arXiv (Cornell University)*, Dec. 2019, doi: <https://doi.org/10.48550/arxiv.1912.11299>.
- [5] P.-C. Wang, Y. Zhou, B. Sun, and W. Zhang, “Intelligent Prediction of Vulnerability Severity Level Based on Text Mining and XGBboost,” *IEEE Explore*, Jun. 2019, doi: <https://doi.org/10.1109/icaci.2019.8778469>.
- [6] M. Jimenez, M. Papadakis, and Yves Le Traon, “Vulnerability Prediction Models: A Case Study on the Linux Kernel,” *IEEE*, Oct. 2016, doi: <https://doi.org/10.1109/scam.2016.15>.
- [7] G. Huang, Y. Li, Q. Wang, J. Ren, Y. Cheng, and X. Zhao, “Automatic Classification Method for Software Vulnerability Based on Deep Neural Network,” *IEEE Access*, vol. 7, pp. 28291–28298, 2019, doi: <https://doi.org/10.1109/access.2019.2900462>.
- [8] Y. Jiang and Y. Atif, “An Approach to Discover and Assess Vulnerability Severity Automatically in Cyber-Physical Systems,” *13th International Conference on Security of Information and Networks*, Nov. 2020, doi: <https://doi.org/10.1145/3433174.3433612>.
- [9] M. Gawron, F. Cheng, and C. Meinel, “Automatic Vulnerability Classification Using

Machine Learning,” *Lecture Notes in Computer Science*, pp. 3–17, Jan. 2018, doi: [https://doi.org/10.1007/978-3-319-76687-4\\_1](https://doi.org/10.1007/978-3-319-76687-4_1).

[10] P. Johnson, R. Lagerstrom, M. Ekstedt, and U. Franke, “Can the Common Vulnerability Scoring System be Trusted? A Bayesian Analysis,” *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 1002–1015, Nov. 2018, doi: <https://doi.org/10.1109/tdsc.2016.2644614>.

[11] K. Scarfone and P. Mell, “An analysis of CVSS version 2 vulnerability scoring,” *IEEE Xplore*, 2009. <https://ieeexplore.ieee.org/abstract/document/5314220>

[12] J. Ruohonen, “A look at the time delays in CVSS vulnerability scoring,” *Applied Computing and Informatics*, vol. 15, no. 2, pp. 129–135, Jul. 2019, doi: <https://doi.org/10.1016/j.aci.2017.12.002>.

[13] Y. Jiang and Y. Atif, “Towards automatic discovery and assessment of vulnerability severity in cyber–physical systems,” *Array*, vol. 15, p. 100209, Sep. 2022, doi: <https://doi.org/10.1016/j.array.2022.100209>.

[14] G. D. Stone, “Field versus Farm in Warangal: Bt Cotton, Higher Yields, and Larger Questions,” *World Development*, vol. 39, no. 3, pp. 387–398, Mar. 2011, doi: <https://doi.org/10.1016/j.worlddev.2010.09.008>.

[15] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, “A Systematic Literature Review on Fault Prediction Performance in Software Engineering,” *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp. 1276–1304, Nov. 2012, doi: <https://doi.org/10.1109/tse.2011.103>.

[16] D. Achmadi, Y. Suryanto, and K. Ramli, “On Developing Information Security Management System (ISMS) Framework for ISO 27001-based Data Center,” *IEEE Xplore*, May 01, 2018. <https://ieeexplore.ieee.org/document/8471700>

[17] M. Levy and A. Subburaj, “Emerging Trends in Data Center Management Automation,” *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, Jan. 2021, doi: <https://doi.org/10.1109/ccwc51732.2021.9375837>.

[18] F. Colombelli, Thayne Woycinck Kowalski, and M. Recamonde-Mendoza, “A

hybrid ensemble feature selection design for candidate biomarkers discovery from transcriptome profiles,” *Knowledge Based Systems*, vol. 254, pp. 109655–109655, Oct. 2022, doi: <https://doi.org/10.1016/j.knosys.2022.109655>.

[19] S. Moore, “Focus On The Biggest Security Threats Not The Most Publicized,” *Gartner*, Nov. 02, 2017. <https://www.gartner.com/smarterwithgartner/focus-on-the-biggest-security-threats-not-the-most-publicized>

[20] N. Ziems and S. Wu, “Security Vulnerability Detection Using Deep Learning Natural Language Processing,” *IEEE Xplore*, May 01, 2021. [https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9484500&casa\\_token=JQ15PvhszS0AAAAA:YSjzYXzfRQW09Rc2nkEmNkvLONxVQlHpJJRHhVqsxoJxMvvOAcgbqvGhXvclCwjW-jIU9W8](https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9484500&casa_token=JQ15PvhszS0AAAAA:YSjzYXzfRQW09Rc2nkEmNkvLONxVQlHpJJRHhVqsxoJxMvvOAcgbqvGhXvclCwjW-jIU9W8)

[21] G. Sharma, S. Vidalis, C. Menon, and N. Anand, “Analysis and implementation of semi-automatic model for vulnerability exploitations of threat agents in NIST databases,” *Multimedia Tools and Applications*, Nov. 2022, doi: <https://doi.org/10.1007/s11042-022-14036-y>.

[22] A. M. S. N. Amarasinghe, W. A. C. H. Wijesinghe, D. L. A. Nirmana, A. Jayakody, and A. M. S. Priyankara, “AI Based Cyber Threats and Vulnerability Detection, Prevention and Prediction System,” *IEEE Xplore*, Dec. 01, 2019. <https://ieeexplore.ieee.org/document/9103372> (accessed Mar. 30, 2022).

[23] G. J. Blinowski and P. Piotrowski, “CVE based classification of vulnerable IoT systems,” *arXiv (Cornell University)*, Jun. 2020, doi: <https://doi.org/10.48550/arxiv.2006.16640>.

[24] S. Zhang, X. Ou, and D. Caragea, “Predicting Cyber Risks through National Vulnerability Database,” *Information Security Journal: A Global Perspective*, vol. 24, no. 4–6, pp. 194–206, Nov. 2015, doi: <https://doi.org/10.1080/19393555.2015.1111961>.

[25] S. Na, T. Kim, and H. Kim, “A Study on the Classification of Common Vulnerabilities and Exposures using Naïve Bayes,” *Advances on Broad-Band Wireless Computing, Communication and Applications*, pp. 657–662, Oct. 2016, doi: [https://doi.org/10.1007/978-3-319-49106-6\\_65](https://doi.org/10.1007/978-3-319-49106-6_65).

[26] MITRE, “CVE - Common Vulnerabilities and Exposures (CVE),” *Mitre.org*, 2019. <https://cve.mitre.org/>

- [27] “cve-website,” *www.cve.org*. <https://www.cve.org/>
- [28] NIST, “NVD - Home,” *Nist.gov*, 2019. <https://nvd.nist.gov/>
- [29] “Vulnerability Database,” *Vuldb.com*, 2019. <https://vuldb.com/>
- [30] C. C. for C. Security, “Canadian Centre for Cyber Security,” *Canadian Centre for Cyber Security*, Oct. 28, 2022. <https://www.cyber.gc.ca/en>
- [31] “0day.today Agreement - 0day.today Exploit Database : vulnerability : 0day : new exploits : buy and sell private exploit : shellcode by 0day Today Team,” *151.80.37.64*. <http://151.80.37.64/> (accessed Dec. 19, 2023).
- [32] Offensive Security, “Offensive Security’s Exploit Database Archive,” *Exploit-db.com*, 2019. <https://www.exploit-db.com/>
- [33] CVE Details, “CVE security vulnerability database. Security vulnerabilities, exploits, references and more,” *Cvedetails.com*, 2009. <https://www.cvedetails.com/>
- [34] Y. Fang, Y. Liu, C. Huang, and L. Liu, “FastEmbed: Predicting vulnerability exploitation possibility based on ensemble machine learning algorithm,” *PLOS ONE*, vol. 15, no. 2, p. e0228439, Feb. 2020, doi: <https://doi.org/10.1371/journal.pone.0228439>.
- [35] L. Rodriguez, J. Trazzi, V. Fossaluzza, R. Campiolo, and D. Batista, “Analysis of Vulnerability Disclosure Delays from the National Vulnerability Database,” *SBC*, May 2018, Available: <https://sol.sbc.org.br/index.php/wscdc/article/view/2394>
- [36] J. Jacobs, S. Romanosky, B. Edwards, I. Adjerid, and M. Roytman, “Exploit Prediction Scoring System (EPSS),” *Digital Threats: Research and Practice*, vol. 2, no. 3, pp. 1–17, Jul. 2021, doi: <https://doi.org/10.1145/3436242>.
- [37] “CVE Database - Security Vulnerabilities and Exploits | Vulners.com,” *vulners.com*. <https://www.vulners.com> (accessed Dec. 20, 2023).
- [38] M. Das, Selvakumar Kamalanathan, and Alphonse, “A Comparative Study on TF-

IDF feature Weighting Method and its Analysis using Unstructured Dataset,” *arXiv (Cornell University)*, Aug. 2023, doi: <https://doi.org/10.48550/arxiv.2308.04037>.

[39] “CVSS v4.0 Specification Document,” *FIRST — Forum of Incident Response and Security Teams*. <https://www.first.org/cvss/v4.0/specification-document>

[40] S. M. Kasongo and Y. Sun, “Performance Analysis of Intrusion Detection Systems Using a Feature Selection Method on the UNSW-NB15 Dataset,” *Journal of Big Data*, vol. 7, no. 1, Nov. 2020, doi: <https://doi.org/10.1186/s40537-020-00379-6>.

[41] A. Anwar, A. Abusnaina, S. Chen, F. Li, and Aziz Mohaisen, “Cleaning the NVD: Comprehensive Quality Assessment, Improvements, and Analyses,” *arXiv (Cornell University)*, Jun. 2020, doi: <https://doi.org/10.48550/arxiv.2006.15074>.

[42] X. Li, S. Moreschini, Z. Zhang, F. Palomba, and D. Taibi, “The anatomy of a vulnerability database: A systematic mapping study,” *Journal of Systems and Software*, vol. 201, pp. 111679–111679, Mar. 2023, doi: <https://doi.org/10.1016/j.jss.2023.111679>.

[43] “Selenium IDE · Open source record and playback test automation for the web,” *selenium.dev*. <https://www.selenium.dev/selenium-ide/>

[44] *Microsoft.com*, 2019. <https://support.microsoft.com/en-au>

[45] M. Perez, “What is Web Scraping and What is it Used For? | ParseHub,” *ParseHub Blog*, Aug. 06, 2019. <https://www.parsehub.com/blog/what-is-web-scraping/>

[46] A. Khazaei, M. Ghasemzadeh, and V. Derhami, “An automatic method for CVSS score prediction using vulnerabilities description,” *Journal of Intelligent & Fuzzy Systems*, vol. 30, no. 1, pp. 89–96, Aug. 2015, doi: <https://doi.org/10.3233/ifs-151733>.

[47] Boston University, “Homepage | Boston University,” *Boston University*, 2019. <https://www.bu.edu/>

[48] Scikit-learn, “sklearn.ensemble.RandomForestClassifier — scikit-learn 0.20.3 documentation,” *Scikit-learn.org*, 2018. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

[49] C. S. GMT +01:00 Route des Acacias 48, CH-1227 Carouge, SUISSE- Tel : +41 22 308 48 60- Fax : +41 22 308 48 68- Timezone:, “Algorithm & computer science: definition and understanding,”*www.iig.ch*.<https://www.iig.ch/en-en/blog/computer-science/algorithm-computer-science-definition-and-understanding#:~:text=Algorithms%20are%20used%20to%20find>

[50] Cybersecurity and Infrastructure Security Agency, “Known Exploited Vulnerabilities Catalog | CISA,” *www.cisa.gov*, 2023. <https://www.cisa.gov/known-exploited-vulnerabilities-catalog>