VULNERABILITY ANALYSIS USING UNSUPERVISED MACHINE LEARNING METHODS

Project work submitted to Avinashilingam Institute for Home Science and Higher

Education for Women

In partial fulfilment of the Requirements for the Degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

Submitted By

K. Megha sree Kannaiah (20PIT005)

Under the guidance of

Dr. D. Shanmugapriya, M.Sc., M.Phil., Ph.D.,

Head, Department of Information Technology

AVINASHILINGAM INSTITUTE FOR HOME SCIENCE AND HIGHER EDUCATION FOR WOMEN

SCHOOL OF PHYSICAL SCIENCES AND COMPUTATIONAL SCIENCES

DEPARTMENT OF INFORMATION TECHNOLOGY

COIMBATORE-641043

May - 2022

VULNERABILITY ANALYSIS USING UNSUPERVISED MACHINE LEARNING METHODS

Project work submitted to Avinashilingam Institute for Home Science and Higher

Education for Women

In partial fulfilment of the Requirements for the Degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

Submitted By

K. Megha sree Kannaiah (20PIT005)

Under the guidance of

Dr. D. Shanmugapriya, M.Sc., M.Phil., Ph.D.,

Head, Department of Information Technology

AVINASHILINGAM INSTITUTE FOR HOME SCIENCE AND HIGHER EDUCATION

FOR WOMEN

SCHOOL OF PHYSICAL SCIENCES AND COMPUTATIONAL SCIENCES

DEPARTMENT OF INFORMATION TECHNOLOGY

COIMBATORE-641043

May - 2022

DECLARATION

DECLARATION

I hereby declare that the project entitled **"VULNERABILITY ANALYSIS USING UNSUPERVISED MACHINE LEARNING METHODS"** is a record of the original work done by **K. Megha sree Kannaiah(20PIT005)** under the guidance of **Dr. D. Shanmugapriya, M.Sc., M.Phil., Ph.D., Head, Department of Information Technology**, School of Physical sciences and Computational sciences, Avinashilingam Institute for Home Science and Higher Education for Women in the partial fulfilment for the award of the degree of Master of Science in Information Technology, and this project has not formed the basis for any Degree/Diploma/Associates.

Place:

Date:

Signature of the Candidate

_____

Countersigned by

Dr. D. Shanmugapriya, M.Sc., M.Phil., Ph.D.,

Head, Department of Information Technology,

School of Physical Sciences and Computational Sciences.

CERTIFICATE

Avinashilingam Institute for Home Science and Higher Education for Women

(Deemed to be University under Estd. u/s 3 of UGC Act 1956, Category 'A' by MHRD)

Re-accredited with 'A++' Grade by NAAC. CGPA 3.65/4, Category 1 University by UGC

Coimbatore - 641 043, Tamil Nadu, India

CCI

DST - CURIE - AI Sponsored

Centre for Cyber Intelligence

MLI

CERTIFICATE OF APPRECIATION

This is to certify that Ms. Megha Sree K (20PIT005), M.Sc Information Technology, Avinashilingam Institute for Home Science and Higher Education for Women, has successfully completed the project entitled "Vulnerability Analysis using Unsupervised Machine Learning Methods" under Centre for Cyber Intelligence - Centre for Machine Learning and Intelligence - a DST - CURIE - AI facility during December 2021 - May 2022.

Dr. G. Padmavathi
Dean, School of PSCS
CCI - Principal Investigator

Dr. P. Subashini
Project Coordinator - DST - CURIE - AI

Dr. S. Kowsalya
Registrar

Certificate

CERTIFICATE

This is to certify that this project work entitled "**VULNERABILITY ANALYSIS USING UNSUPERVISED MACHINE LEARNING METHODS"** done by **K. Megha Sree Kannaiah(20PIT005)** has been submitted to Avinashilingam Institute for Home science and Higher education for women, Coimbatore-43 in partial fulfilment of the requirement for the award of the degree of **MASTER OF SCIENCE ININFORMATION TECHNOLOGY.** This Project has not found the basis for the award of any Degree/Associate/fellowship or similar title to any Candidate of any University. Certified as a bonafied record of the work submitted for the Viva voce held on _____.

Signature of the HOD

Signature of the Guide

Signature of External Examiner

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

I would like to express my sincere thanks to God Almighty, for his constant love and grace that he has showered upon me, which kept me in good health, and sound mind without which my project would not have reached a successful end.

We take this opportunity to express our deep sense of reverential gratitude and sincere thanks to **Shri, Dr. S. P. Thyagarajan, Chancellor,** Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for providing all facilities during my course of study.

I owe my great deal of gratitude to **V. Bharathi Harishankar, Ph.D., FRSA, Vice Chancellor,** Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for extending all resources that facilitated the smooth conduct of the project study.

I express my gratitude to **Dr. S. Kowsalya, Registrar, Avinashilingam** Institute for Home Science and Higher Education for Women, Coimbatore, for providing all facilities and support necessary for the study.

I wish to extend my sincere thanks **Dr. (Mrs.) G. Padmavathi M.Sc., M.Phil., Ph.D., Dean, School of Physical Sciences and Computational Sciences**, for her support and valuable guidance.

I am grateful and obliged wholeheartedly to my esteemed project guide **Dr. D. Shanmugapriya, M.Sc., M.Phil., Ph.D., Head, Department of Information Technology,** for her imparting tremendous supervision, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by her time to time shall carry us a long way in the journey of life on which we are about to embark.

I would like to thank all the faculty members, laboratory staffs of the Department of Information Technology, all our friends and well-wishers who had either directly or indirectly helped us to finish this project successfully in this esteemed university. Last yet importantly, I thank my parents for their kind inspiration, moral support, blessings and encouragement during the course of project. Above all I thank God almighty whose grace was sufficient at all times.

I would like to acknowledge the help rendered by **Centre for Cyber Intelligence, DST–CURIE-AI sponsored Phase II** for providing the laboratory facilities to execute my project.

ABSTRACT

ABSTRACT

Source code vulnerability is a weakness or a glitch in script used for software development purpose that make a way for an attacker to enter inside a network or system of an individual or a company. The broad usage of software projects has resulted in the possibility of emerging vulnerabilities and potential consequences for their exploits.

Existing code analysis methods are ineffectual at identifying vulnerabilities. This project investigates and presents vulnerabilities, particularly in source code. Vulnerabilities paves a way to businesses and individuals approachable to various kinds like malware and account takeovers.

Vulnerability analysis affords an organisation with the essential information, awareness, and risk background it needs to recognise and respond to threats to its environment. The project's intention is to execute a vulnerability analysis and tool framework.

A complete vulnerability evaluation can assist companies to enhance the safety of their structures. Vulnerability analysis also offers detailed steps for revealing current flaws and preventing future assaults. The analysis can also help improve your company's reputation and goodwill, inspiring greater confidence among customers. It can also assist in safeguarding the integrity of assets in the event of any malicious code being concealed in any of said assets.

The proposed framework consists of five phases, including data acquisition, data preprocessing, feature selection, model building (unsupervised machine learning models) and performance evaluation. According to the Positive Technologies report 2020, 31% of companies dredged endeavor to impose source code vulnerabilities; nearly one-third of discovered risks accommodate software exploit shots.

In this project, vulnerability analysis was done with unsupervised machine learning method using clustering techniques. Examining the vulnerabilities, especially in source code, is done and presented in this project. There are a variety of frequently used techniques, but clustering is the most appropriate. This algorithm focuses on identifying groups of data according to similarities. Hence, the method of clustering allows the data to form clusters. The fact that this is an unsupervised problem with no target class is one of the main reasons for its usage.

CONTENTS

TABLE OF CONTENT

CHAPTER NO.	TITLE	PAGE NO.
1	**INTRODUCTION** **1.1** Problem Statement **1.2** Motivation and Justification **1.3** Objectives	1
2	**SYSTEM REQUIREMENTS** **2.1**Hardware Specification **2.2**Software Specification **2.3** About the software	8
3	**LITERATURE STUDY** **3.1** Background study	14
4	**METHODOLOGY** **4.1** Dataset Description **4.2 Modules Description** **4.2.1** Data Acquisition **4.2.2**Data Preprocessing **4.2.3**Dimensionality Reduction (PCA) **4.2.4**Unsupervised Machine Learning Model Building	21

	4.2.5 Performance Evaluation	
5	**RESULTS AND DISCUSSIONS**	**46**
6	**CONCLUSION AND SCOPE FOR FUTURE ENHANCEMENT**	**58**
7	**REFERENCES**	**61**
8	**ANNEXURE** **8.1** Coding and Screenshots	**62**

LIST OF TABLES

S.NO	NAME OF THE TABLE	PAGE.NO
1	3.1 Summary of Literature Review	**18**
2	4.1 Attributes and its Description	**26**
3	4.2 Common Vulnerability Scoring System	**28**
4	4.3Basic Metric Components	**29**
5	4.4Performance Evaluation Metrics	**42**

LIST OF FIGURES

S.NO	FIGURE NAME	PAGE NO
1	4.1 Basic Architecture of ML	**23**
2	4.2 Methodology	**24**
3	4.3 Types of Unsupervised Learning	**33**
4	4.4Workflow of clustering in unsupervised ML	**34**
5	4.5Types of Clustering	**35**
6	4.6 Autoencoder Components	**36**
7	5.1 Heat map of Principal Components	**50**
8	5.2 Variance in Each Principal Components	**50**
9	5.3 Elbow Method	**52**
10	5.4 Results Obtained through silhouette coefficient	**56**
11	5.5 Results Obtained through Davies-Bouldin Index	**57**

INTRODUCTION

CHAPTER 1

INTRODUCTION

Even ordinary people are now bare to software. Millions of individuals around the sphere use smart devices. Everything is software-based. We need to write code in order to develop software. Smartphone users, for example, take images for social media. It is used by laptop users for writing purposes. Gamers use their home consoles to play games. This software may include functionality that the developer did not anticipate. These features can be exploited to harm a system if they are malicious. Such software flaws are a major problem in the software industry.

When building software, one of the most important considerations is security. Software is being produced and released now more than ever. The number of users will grow in parallel with the number of applications. All of these users are generating sensitive data. Not only are firms and organisations interested in processing and analysing this data, but so are hostiles. Security and information breaks are becoming normal, and consistently the results are developing. As per the NIST study, the quantity of IT weaknesses has been expanding throughout the course of recent years.

Endeavors to go after can represent an assortment of dangers. According to Business Insider, the top 21 greatest data breaches in 2018 exposed billions of user accounts. Adversaries can use this data to blackmail people and companies or sell it on the black market. For example, an attack by a group of hackers tries to capture the user data on of Ashley Madison, a dating service company in 2015.

As a result, cyber security is becoming increasingly essential. Unfortunately, developing software also entails the development of security flaws. As more software is written, the number of vulnerabilities in software grows. When attackers tries to exploit these vulnerabilities; they can pose a risk, with some posing a greater risk than others. These dangers can hurt both persons and businesses.

To address these issues, vulnerabilities in source code must be addressed as early as possible during software development, as bug fixes become more expensive with time.

Developers must frequently test and examine their code for vulnerabilities to ensure that the majority of vulnerabilities are not present after release. This might be done by the developers or by security professionals. However, this process might take a long time and cause delays in the rest of the software's development. As a result of the analysis, we are able to equip firms with awareness and knowledge in order to secure their products from becoming vulnerable.

Designers should every now and again test and analyze their code for weaknesses to guarantee that most of weaknesses are absent after discharge. These tasks are handled by the professionals. Moreover, this process of examining took some time and results in delays. As a result of the analysis, we are able to equip firms with awareness and knowledge in order to secure their products from becoming vulnerable.

Software developers use vulnerability assessment for checking the vulnerabilities without the specialists to speed up the process. Developers can respond appropriately by developing a prediction model to detect whether a deployment is susceptible.

The massive increase in network traffic in cyberspace over the last few decades has resulted in an increase in the number of security threats that might cause havoc in Internet. Various cyber security approaches can aid in the fight against this problem. Traditional cyber security systems are failing because they can't handle the massive number of logs generated and aren't capable of detecting zero-day threats.

Source code vulnerability is a flaw or weakness in code or design that can result in a network or endpoint security compromise. Hackers can exploit this flaw by adding an endpoint to your code and retrieving data, messing with your programme, or, in the worst-case scenario, wiping everything. According to security experts, approximately 76% of programmes contain at least one vulnerability. More than four vulnerabilities were detected in 34% of the situations.

Vulnerabilities provide attack vectors for a hacker to accomplish malicious code and to get an entry to a memory of destination system. Vulnerabilities allow an attacker to execute code or gain access to the memory of a target machine. Approaches like injecting codes, as well as hacking scripts and apps are used to attack vulnerabilities.

Software developers nowadays compete with one another to create new products for the market. During the development of a new product, software designers are under a lot of pressure. Developers should thoroughly design the product's system development life cycle during the development phase. They can minimise any software problems that may exist in this way. As a result, it is possible to create secure software.

Vulnerabilities in software can be exploited, resulting in a wide range of adverse consequences for users and their enterprises. The user and the system may be subject to cyber security assaults if this happens. The chances of a user being exposed to security threats like viruses and malware would substantially increase.

Detecting source code vulnerabilities before they are exploited has been a difficult task for decades. Traditional code analysis approaches have been offered, but they are generally ineffective. We analysed the data to provide a detailed summary of the flaws, which will aid the company in improving their security environment. The data was compiled using the National Vulnerability Database (NVD) at the National Institute of Standards and Technology (NIST), as well as the Common Vulnerabilities and Exposures (CVE) database, which is the current global standard for vulnerability numbering and proof of identity.

This project's purpose is to investigate common vulnerabilities and exposures (CVEs).

While there are other vulnerability databases, we'll concentrate on the National Vulnerability Database because it's the most widely utilised (NVD). It is the industry standard database perfectly capture all vulnerabilities revealed publicly.

Unsupervised and supervised machine learning techniques are the most regular used machine learning approaches. Unsupervised technique works directly on unlabeled data, whereas supervised approach go ahead with preceding knowledge of the wanted outcome in the form of labelled information, allowing the training process to be directed. the learning algorithm must find these labels even in its absence to lead the process.

Because of the contemporary growth in platform of automation, such as robotics, AI and in IOT, makes us more preparatory than ever to be aware of potential vulnerabilities to prevent from cyber-attacks. Vulnerabilities in source code have resulted in several disasters. These defects must be investigated so that the most critical ones can be addressed before they become disasters.

This analysis emphasizes the extent and tries to impose the vulnerability y, and how severe each type is by analyzing it. Data analysis and PCA (Principal Component Analysis) which is a dimensionality reduction technique, and via clustering algorithms such as K-means, K-medoids, and agglomerative hierarchical clustering, are used to do this. This analysis is used to determine the severity of vulnerabilities discovered in the dataset.

Every day, new security flaws are discovered and disclosed. These flaws can cause input-output problems, memory leaks, authentication breaches, network difficulties, and other problems. Vulnerability analysis aids in the detection of environmental weaknesses, suggesting which areas require further attention and how to mitigate security concerns. An attacker's chances of gaining access to the source code are reduced as a result.

There are a few developed methods for detecting the severity of an issue, such as the CVSS score (Common Vulnerability Scoring System). The CVSS score assigns a numerical value to the severity of a problem. This project's contribution is the analysis of the severity of vulnerabilities in the dataset and the identification of the top most serious vulnerabilities.

As a result, we suggested a vulnerability analysis model based on clustering techniques in order to analyse vulnerabilities in a very effective and fast manner and to assist organisations in overcoming all of these challenges. In unsupervised machine learning, determining the quality of the results obtained by clustering approaches is a critical challenge.

Finally, the major goal of this study is to employ clustering to analyse vulnerabilities. The analysis was carried out with the help of a dataset provided by the US government. First and foremost, preprocessing is carried out. Clustering is a machine learning technique that is being employed. Following the evaluation, the performance of clustering methods was compared, and a result was reached. As a result, performance evaluation is a critical component of this project in order to achieve the best outcome across all clustering models.

Many unsupervised learning approaches and algorithms have been introduced since the last decade, many of which are well-known and widely used algorithms for unsupervised learning. The growing interest in applying unsupervised learning techniques has led to great success in fields such as computer vision, natural language processing, speech recognition, and developing autonomous self-driving cars.

Unsupervised learning eliminates the need for labelled data and manual handcrafted feature engineering, enabling general, more flexible, and automated ML methods. This project focuses on unsupervised learning. In this, we conducted a systematic literature survey on the approaches and algorithms of unsupervised learning, the metric used for performance evaluation in unsupervised learning, and the merits and demerits of several studies employing unsupervised learning.

This project will also enable others to compare the effectiveness and efficacy of unsupervised learning algorithms used.

The rest of this work is organised in the following manner. Chapter II summarises previous work in vulnerability identification utilising Natural Language Processing (NLP), deep learning, and machine learning to extract information from computer code. Chapter III provides a more in-depth look at the methodology that has been offered. In Chapter IV, we go over everything we did for the project. The final chapter, Chapter V, contains the conclusion as well as the possibilities for future improvement.

1.1Problem Statement

As more and more software is developed every year, it needs source code to build software or products. The created software needs to be tested for vulnerabilities, as the exploitation of such vulnerabilities could probably cost more than the testing of the software. Also, no software user wants their data to be stolen and sold by hackers on the black market. Testing of the software is expensive because there are not enough specialised pentesters to hire, and testing the software is sometimes tedious and time-consuming.

However, automated testing has its own problems, namely that its results need to be more reliable and they have to truly find vulnerabilities instead of returning lots of false positives. Machine learning is one method of finding source code vulnerabilities that may give good results.

Formally, the problem statement of this project is defined as:

Analysing vulnerabilities in Common Vulnerabilities and Exposures (CVE) database.

1.2 Motivation and Justification

Researchers from the University of Cambridge, UK have identified a new attack called Trojan Source that allows threats to hide vulnerabilities in the source code of software projects at the beginning of November 2021. A new type of source code and supply chain assault in which

the source code evaluated by a human differs from the actual software generated by the compiler. The source code appears legal, but it is intended to harm, steal, or disrupt the system or network. Through the use of Unicode control characters, it distinguishes the source code read on the screen by a person from the binary code generated by a compiler.

The new "Trojan Source" flaws affect a who's who of the world's most popular programming languages, including the top five: Python, Java, JavaScript, C#, and C, putting massive numbers of computer programmes at risk. According to recent cases, more than one in every four businesses is exposed to WannaCry. The WannaCry ransomware in Microsoft Windows ended up exploiting weaknesses, coming about misfortunes in dollars in billions.

A detailed examination to understand the nature of vulnerabilities, which can have serious consequences for an organization if they go unnoticed due to a lack of analysis. Before vulnerabilities can be exploited, they must first be found and fixed. Vulnerability analysis is critical. Vulnerability analysis will assist organisations of all sizes, as well as people who are at risk of a cyber-attack, which will benefit most from vulnerability analysis.

A thorough study on doing a careful analysis of the problem using the NVD database, which offers clarity to those who rely on the organisation to handle their duties, Source code is required to construct a product at a certain level of application development. To deliver a vulnerable free product to the user, there is a requirement to build one. As a result, any company aiming to create a susceptible-free product will tremendously profit through this project.

1.3 Objectives

The primary objective of this project is to examine the most common vulnerabilities.

The secondary objectives are

1) To remove irrelevant data and use label encoding to convert categorical values into numerical values.
2) Using PCA to reduce dimensions and clustering algorithms, and then evaluate their performance to see which one worked best.

SYSTEM REQUIREMENTS

CHAPTER 2

SYSTEM REQUIREMENTS

2.1 Hardware Specification

- **RAM:** 8GB
- **Processor:** HP intel i7
- **Hard Disk:** 1TB minimum with 256 SSD

2.2 Software Specification

- **OS:** Windows10
- **Python IDE**: python3.7.x
- Jupyter Notebook
- Sklearn tools, NumPy, Pandas
- **Language**: Python

2.3About the Software

2.3.1Python

Python is a significant level and deciphered language. center way of thinking of python supports code coherence. Its components pointed in supporting software engineers recorded as a hard copy intelligent code for projects.

Python is a progressively composed programming stage. It upholds different programming ideal models, including organized, object-situated and useful programming. As it is a far-reaching standard library, it is frequently depicted as "batteries included" language.

Guido van Rossum began working in the last part of the 1980s in python and in 1991, Python 0.9.0 was sent off. Python 2.0 was distributed in 2000 with added new capacities trash assortment framework in view of reference forgetting about prior to being progressively eased

in 2020 with form 2.7.18. In 2008, Python 3.0 was delivered. It is routinely perhaps the most famous programming language. On account of its elements makes it well known and important.

A few aspects of python described below:

Simple Understanding

Rather than other programming dialects, Python is easy to learn. Its sentence structure is easy to learn and like the language of English. It is novice cordial language.

Linguistic That Articulates

Python can deal with muddled works with code of not many lines. Best model is hi world code. It will take one line to run when contrasted with Java will take various lines.

Interpretation

Python is a language of understanding, inferring that each line can run independently. Troubleshooting is straightforward as it is deciphered.

Cross-stage Language

Python can run namelessly on different stages. Along these lines, it is a versatile language. It permits developers to make programming for various adversary stages by making just a single program.

Accessible Open Source

Python is a free programming language to utilize. It is openly accessible in its true site. It has a gigantic local area around the circle scarcely attempting to create new capacities and modules.

Object-Oriented Programming

Python presents the ideas of classes and items by empowering object-situated programming. Gives offices like legacy, polymorphism, and epitome. This technique assists software engineers to compose reusable code and creating applications with negligible utilization of code.

Protractible

We can assemble the code in different dialects, and use it in any work on python. It changes it into byte code to run on any stage.

Quality Atheneum (Large)

It presents a numerous arrangement of libraries for different spaces in AI libraries ordinarily include TensorFlow, Pandas, NumPy, Keras, and Pytorch, to give some examples and in python web advancement systems.

Support for GUI Programming

GUI is utilized for planning work area applications. PyQT5, Tkinter, and Kivy were among the libraries used to assemble the web application.

Unified

It's easy to incorporate with C, C++, and JAVA dialects, among others. Python, similar to C, C++, and Java, executes code line by line. It makes troubleshooting the code a lot simpler.

Embeddable

Other writing computer programs dialects' code can be utilized here. Its source code is appropriate for a wide scope of present-day dialects. It has the capability of consolidating unknown dialects.

Dynamic Memory Allocation

Clients need not to proclaim the information sort of factors. At the point when we dole out a worth, the variable's memory is naturally designated at runtime. In the event that the number worth of 15 is set to x, don't have to compose int x = 15. x = 15 is sufficient.

2.3.2 Anaconda

Anaconda constrictor is a dispersion of Python and R programming dialects designated at making logical registering bundle the board more straightforward. Information science bundles for Windows, Linux, and macOS are remembered for the delivery. In 2012 by Peter Wang and Travis Oliphant made Anaconda, Inc.

Bundle the executive’s framework conda deals with the bundle renditions. This bundle supervisor was veered off as a particular open-source bundle since it ended up being significant all by itself, as well with respect to purposes other than Python. Miniconda is a conservative, bootstrapped variant of Anaconda that just incorporates conda, Python, and the projects they rely upon, as well as a couple of additional bundles.

More than 7,500 extra open-source bundles, as well as the conda bundle and virtual climate supervisor, can be imported by means of PyPI. Boa constrictor Navigator, a graphical UI, is an option in contrast to the order line interface (CLI).

No matter what the past establishment, it would introduce a bundle with any of its conditions. Brings about a client who recently had a practical establishment of Google Tensor Flow might find that it has stopped working subsequent to utilizing pip to introduce another bundle requires an alternate rendition of the dependent NumPy library than Tensor Flow. In certain conditions, the bundle would be helpful.

The conda introduce order can be utilized to introduce open-source bundles from the Anaconda store, Anaconda Cloud (anaconda.org), or the client's own confidential storehouse or mirror. Boa constrictor, Inc. incorporates and delivers the Anaconda archive's bundles and gives pairs to Windows 32/64 digit, Linux 64 cycle, and MacOS 64 bit. Custom bundles can be made utilizing the conda construct order and imparted to others through Anaconda Cloud, PyPI, or different stores. Anaconda2 accompanies Python 2.7 of course, while Anaconda3 accompanies Python 3.7. Nonetheless, any adaptation of Python included with boa constrictor can be utilized to develop new conditions.

2.3.3 Anaconda Navigator

Anaconda Navigator is a work area graphical UI (GUI) remembered for the Anaconda appropriation that allows clients to send off programs and oversee conda bundles, conditions, and channels without utilizing order line orders.

Navigator might search for bundles on Anaconda Cloud or in a neighborhood Anaconda Repository, introduce them, execute them, and update them. It runs on Windows, Mac OS X, and Linux.

Guide incorporates the accompanying projects of course:

- RStudio
- Jupyter Lab
- Jupyter Notebook
- Qt Console
- Spyder
- Glue

- Orange
- Visual Studio Code

2.3.4 Jupyter Notebook

The Jupyter Notebook is an open-source web application that you can use to make and share archives that contain live code, conditions, representations, and message. Utilizes include: information cleaning and change, mathematical re-enactment, factual demonstrating, information perception, AI, and significantly more.

Jupyter Notebooks is a fork of the IPython project, which used to have its own IPython Notebook project. Jupyter gets its name from the three primary programming dialects it upholds: Julia, Python, and R. Jupyter accompanies the IPython part, which permits you to assemble Python programs, despite the fact that there are by and by in excess of 100 elective bits accessible.

The Jupyter Notebook is comprised of three sections:

The journal web application:

The journal web application is an engaging web application that permits clients to make note pad pages as well as construct and run code.

Portions:

Portions are independent cycles sent off by the journal web application that run clients' code in a particular language and give results to the scratch pad web application. Calculations for intelligent gadgets, tab culmination, and thoughtfulness are totally taken care of by the bit.

Note pad reports:

Note pad reports are independent archives that contain all of the stuff introduced in the note pad online application, including calculation data sources and results, account writing, conditions, photos, and rich media portrayals of items. Each record in the Fernando Perez made IPython Notebook as an electronic connection point to the IPython bit.

LITERATURE STUDY

CHAPTER 3

LITERATURE STUDY

3.1 Background Study

In the evolving world of computer and network security, continuous advances have been made in tool development and techniques for both compromising and protecting computer systems. The growth in the number and quality of security products indicates the perceived need for further means to protect computer systems from compromise **[Baker D. W., Christey S. M., Hill W. H., Mann D. E, 1999].**

Venter et al. has identified a set of 13 Harmonized Vulnerability Categories, which represent the entire range of vulnerabilities that are currently known **[VENTER, H.S.; ELOFF, J.H.P, 2005].**

Neuhaus et al. studied the Mozilla vulnerability history and found that components with similar imports or function calls were likely to be vulnerable, **2007**.

H. s. Venter and Jan H. P. Eloff, 2008 proposed the possibilities of using a data clustering technique to intelligently categorize vulnerabilities. This makes it simpler and different than the normal approaches where categories are identified first and vulnerabilities are then assigned to the most appropriate categories. We also suggest a solution for standardization of the vulnerability categories. This makes it possible to compare and assess VSs, and to increase the efficiency of risk management as more abstract reports can now be produced.

Livshits et al. analysed common JAVA vulnerabilities. Li et al. applied static and dynamic analysis to discover code clone vulnerability based on released security patches, **2008**.

Perrand et al. investigated JAVA component vulnerabilities, **2009**.

Such reports are often large and place huge burdens on administrators to rectify the vulnerabilities. For example, the Nessus Security Scanner provides a report that gives detailed and organized information about all the vulnerabilities detected on the network based on the address of the hosts, ports and security issues regarding the port. It is still very hard for an administrator to go through it as it is 3 too long and does not highlight the problem areas of the network for a quick response. This could have a negative impact on the efficiency and

effectiveness of risk management, as vulnerabilities are often not rectified immediately **(Nessus, 2011).**

Malicious attackers constantly look for and exploit weaknesses in computer systems in order to attack or break these security services. Those weaknesses in security systems that might be exploited to cause harm or loss are referred to as vulnerabilities. This makes Internet security a challenging but interesting topic to research **(Pfleeger C. P, 2012).**

Shahzad et al. conducted a descriptive statistical analysis of a large software vulnerability dataset employing clustering on type-based vulnerability data, **2012**.

Bhanudas S. Panchabhai and **Dr. Ashok N. Patil, 2013** presented a way to classify the vulnerabilities in the CVE repository and propose a solution for standardization of the vulnerability categories using a data-clustering algorithm.

The CVE was initiated in 1999 to solve this naming inconsistency. It is a list or dictionary that provides common names for publicly known information security vulnerabilities and exposures. The CVE repository is downloadable from the CVE web site in HTML format, Text format, for Comma-separated format. Although CVE is not quite a vulnerability database, almost all the major Vulnerability Databases (VDs) and Vulnerability scanners have a reference to it. The fact that it provides an internationally accepted naming standard for common vulnerabilities makes it a more suitable repository to perform clustering than any other vulnerability database.

Similar work by **Scandariato et al**. analysed vulnerable components in Android apps, **2014**.

Another paper by **Pang et al** used a deep learning algorithm paired with a statistical feature selection algorithm to predict vulnerable classes in Java based android applications, **2017**.

A paper written by **Harer et al**. discussed about the use of deep learning models and also traditional machine learning models to detect vulnerabilities in C/C++. However, they use CFGs (Control Flow Graph) as their major feature to predict said vulnerabilities. Their features are extracted mainly based on the build and source of the code, **2018**.

Dong et al. analysed the inconsistencies in public security vulnerability reports, including the NVD, and found over claims and under claims in the affected software product, **2019**.

William et al. proposed a framework to discover evolutionary patterns in the vulnerabilities, **2020**.

Livshits et al, had presented a paper, where they discuss a solution to reduce software vulnerabilities using static analysis. They propose a static analysis tool which provides warnings about vulnerabilities based on a method by name Tainted Object Propagation analysis. In this technique they correlate the source and sink objects and their propagation in order to identify the threats. With this method, they are able to unearth 29 vulnerabilities successfully, **2008**.

An et al. examined C code vulnerabilities by similarity matching between the security characteristics and the code, **2018**.

Ponta et. al manually collected and curated a dataset of vulnerabilities of open-source Java software and their fixing commits. They used their dataset to train classifiers that could automatically identify security-relevant commits in code repositories, **2019**.

Huang et al. classified vulnerabilities employing several clustering algorithms to create a relatively objective classification criterion among the vulnerabilities, **2019**.

Verdi et al. investigated security vulnerabilities in C++ code snippets on Stack Overflow and developed a browser extension to check for vulnerabilities in code snippets when they upload them on the platform, **2020**.

Gkortzis et al. studied software vulnerabilities and showed projects without test code or continuous integration suffer from undetected vulnerabilities, **2021**.

Machine learning (ML) is a data-driven approach in which machines learn from the data without the involvement of humans. Several domains take advantage of mind-boggling applications of ML. There are three main learning problems in ML: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning involves the training of the model on a labelled dataset. Unsupervised learning involves the training of a model in an unlabelled dataset. The model learns on its own by learning the features of the training dataset. Based on that learning feature, the model makes predictions on test data. Several unsupervised learning approaches and algorithms range from clustering, k-means to agglomerative, principal component analysis, and Fuzzy C-means. Clustering involves the grouping of objects based on their similar features. The algorithms in clustering are categorized into two broad categories such as hierarchal clustering and partitioned clustering. **(Salim Dridi, 2021).**

S.NO	TITLE OF THE PAPER	AUTHORS	TECHNIQUES USED	RESULT
1.	Categorizing the Vulnerabilities using Data Clustering Techniques, 2004.	H. s. Venter, Jan H. P. Eloff	SOM (Self Organizing Feature Map)	Categorized the vulnerabilities in the CVE repository using SOM.
2.	Organize Vulnerabilities by Data Clustering Techniques, 2013.	Bhanudas S. Panchabhai, Dr. Ashok N. Patil	SOM (Self Organizing Feature Map)	Classified the vulnerabilities in the CVE using a data clustering algorithm and SOM.
3.	An Empirical Study on using the National Vulnerability Database to Predict Software Vulnerabilities, 2014.	Su Zhang, Doina Caragea, Xinming Ou	Data-mining	Applied data mining techniques On NVD data in order to build a prediction Model for TTNV (Time To Next Vulnerability).
4.	An Analysis of Common Vulnerability And Exposure (CVE) of software Products in the year 2016, 2017.	Afiq Bonandir, Salman Yussof	Data Analysis	Analysis on the product vulnerabilities that have been discovered in 2016.
5.	Automated Vulnerability Detection In Source Code using Deep Representation Learning, 2018.	Rebecca L. Russell, Louis Kim, Lei H. Hamilton, Tomo Lazovich, Jacob A. Harer, Onur	Deep Representation Learning	Used ML to detect software vulnerabilities directly from source code and applied a variety of ML techniques.

		Ozdemir Paul M. Ellingwood, Marc W.Mcconley		
6.	Predicting Software Vulnerabilities With Unsupervised Machine Learning Techniques, 2020.	Man, K.W., Verwer, Panichella, Lagendijk.	Clustering and Anomaly Detection(K-Means, KNN)	Machine Learning based process that finds software vulnerabilities.
7.	A Vulnerability Analysis and Prediction Framework, 2020.	Mark A. Williams, Roberto Camacho Barranco, Sheikh MotaharNaim , Sumi De y, M.Shahriar Hossain, and Monika Akbar	Deep-Neural Network, Regression algorithms	An analytical framework that uses a Topically Supervised Evolution Model (TSEM) and algorithms to discover hidden themes and trends in the National Vulnerability Database, corpus describing software vulnerabilities since 1996.
8.	Security Vulnerability Detection Using Deep Learning Natural Language Processing, 2021.	Noah Ziems, Shaoen Wu	Natural Language Processing (NLP)	Used Software Vulnerability detection as a Natural Language Processing (NLP) problem with source code with recent advanced deep

				Learning NLP models.
9.	Sysevr: A Framework for using Deep Learning to Detect Software Vulnerabilities, 2021.	Zhen Li, Deqing Zou, Shouhuai Xu, Hai Jin, Yawei Zhu, Zhaoxuan Chen	Deep Learning	Proposed first systematic framework for using deep learning to detect vulnerability in C/C++ programs with Source code.
10.	Predicting Software Vulnerabilities Using Software Code Metrics, 2021.	Ganesh, Sundarakrishnan	Naive Bayes Classifier, Logistic Regression, Decision Tree, K Nearest Neighbors	Used Machine Learning to predict potential security vulnerability by analyzing the source Code of the system.

Table 3.1: Summary of Literature Review

3.2 Literature Survey Findings

- Prior work concentrated on categorising and organising vulnerabilities.
- Worked with previous datasets until 2016.
- Most of papers that solved the problem did so mostly under supervised learning methods.
- Most of the papers used some other techniques like Deep Learning, Natural Language Processing (NLP), SOM (Self-Organizing Feature Map) and Data Mining instead of clustering.
- Usage of clustering techniques for categorising alone.
- Practically, there is no implementation of a performance evaluation phase.

METHODOLOGY

CHAPTER 4

METHODOLOGY

4.1 Dataset Description

Dataset/source: The source of the dataset is from National Vulnerability Database (NVD).

CVE/NVD Database

The NVD (public weakness data set) is based upon the cve data set; with significant characteristics incorporates seriousness score, weakness type, and effect on each cve section. c/c++ weakness tests are utilized from these data sets for testing our models. Accordingly, we utilize the weakness information bases from **CVE** and **NVD** in this undertaking.

A rundown of sections from genuine activities announced makes up the CVE data set. The weaknesses identified in programming items are addressed by the examples in the data set. CVE is a glossary of regularly involved names for transparently uncovered security blemishes. CVE Identifiers, or normal identifiers, makes it simpler for information to convey between different organization security data sets and devices, and give a sunrise for passing judgment on an association's security apparatuses inclusion.

CVE is:

- One vulnerability or exposure is given a single name and each vulnerability has a standardized description.
- The process of ensuring that various databases and technologies "speak the same language."
- The path to enhanced security coverage and interoperability.
- A benchmark for tool and database evaluation and it is available to the general public for free download and use.
- The CVE Numbering Authorities (CNAs), the CVE Editorial Board and CVE-Compatible Products have all given their endorsement to the venture.

4.2 Modules Description

4.2.1 Data Acquisition

An information researcher invests the vast majority of the energy for looking, cleaning, and for handling the information. As AI utilized generally, a few applications need an adequate number of named information. Indeed, even the best Machine Learning calculations can't fabricate models as expected without great information. The primary intension of the information obtaining covers looking for the datasets that can be utilized to prepare the Machine Learning models. Data acquisition's role is to find and capture the data that can be utilized to evaluate the business patterns, derive actionable insights, develop machine learning models and achieve accurate predictions.

The workflow of the Machine learning is shown in **Figure 4.1**. And the overall methodology for the vulnerability analysis is depicted in **Figure 4.2**.

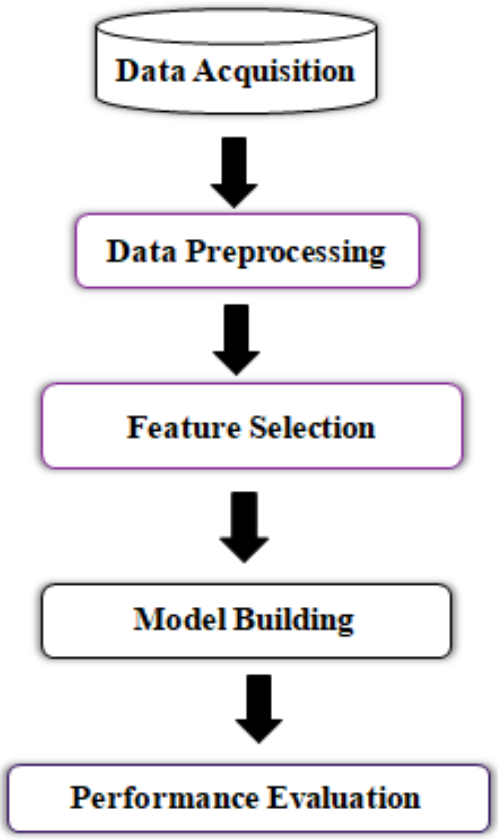


Figure 4.1: Basic Architecture of ML

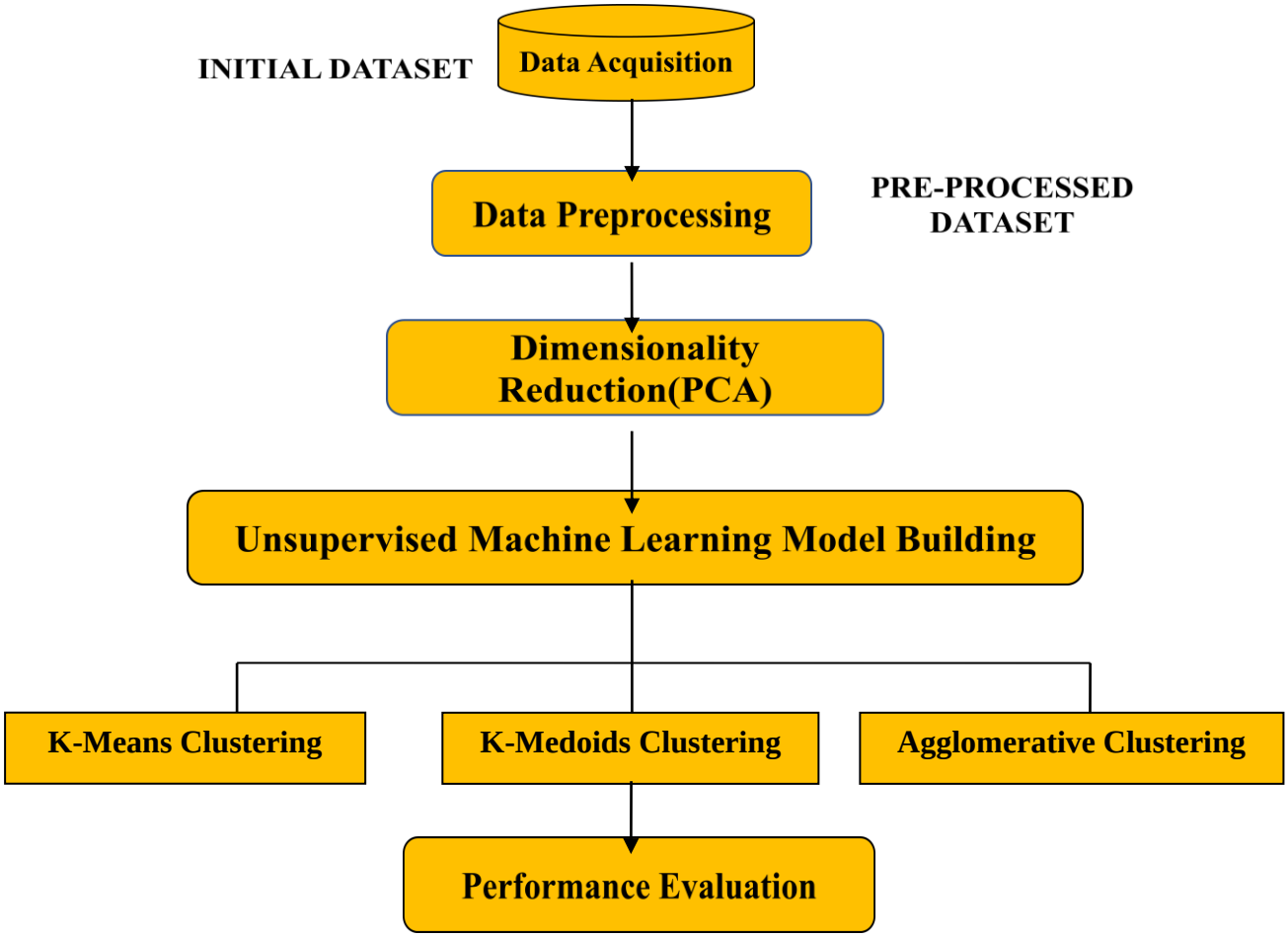


Figure 4.2: Methodology

Tasks performed:

Task 1: Load a dataset from file.

Task 2: Pre-Process a dataset by removing the inessential data and label encoding it.

Task 3:Reduce the elements of the information utilizing Principal Component Analysis (PCA).

Task 4: Applying Clustering calculations like K-Means, K-Medoids and Agglomerative Hierarchical Clustering.

Task 5: Finally, assess its exhibition utilizing Silhouette coefficient and Davies-Bouldin Index.

Why CVE

CVE was sent off as a local area exertion in 1999 when most digital protection instruments involved their own data sets with their own names for security weaknesses. There was no significant assortment among items at that point, and there was no direct technique to tell whether a few data sets were alluding to a similar issue. Besides, each device merchant announced the quantity of weaknesses or openings identified utilizing various standards, intending that there was no normal reason for looking at the items.

The CVE has turned into the business standard for naming weaknesses. CVE Identifiers act as information trade focuses so that network safety items and administrations can speak with each other. They also work as a benchmark for looking at gadget and transporter inclusion, allowing clients to become mindful of which gear is the most appropriate for their requirements. To put it plainly, CVE-consistent administrations and items consider higher inclusion, less convoluted interoperability, and further developed security.

How CVE Works

The most common way of making a CVE Identifier starts with the disclosure and detailing of potential security weakness. The data is then allocated a CVE Identifier by a CVE Numbering Authority (CNA) and posted on the CVE List on the CVE site by the CVE Editor. As a feature of its administration of CVE, the Miter Corporation capacities as Editor and Primary CNA. The CVE Editorial Board manages this cycle.

NVD:

The National Vulnerability Database (NVD) is the US repository for security professionals, researchers, and vendors to report Common Vulnerabilities and Exposures (CVEs). Security teams throughout the world use it to stay aware of newly found security vulnerabilities.

Common Vulnerabilities and Exposures (CVE)

In 1999, the CVE was made to address this terminology befuddle. MITRE began it as a collective task. It's an assortment or word reference of generally involved terms for freely known data security defects and openings. The CVE store can be downloaded in Comma-

isolated design from the CVE site. Every passage in the CVE storehouse has highlights like Name, Description, and References, with Name filling in as the section's essential key.

Using a common name simplifies data sharing between disparate databases and Vulnerability scanners that were previously difficult to combine. As a result, CVE is critical for disseminating security vulnerability information. Although CVE is not technically a vulnerability database, it is referenced in practically all major Vulnerability Databases (VDs) and vulnerability scanners.

It is a better repository for clustering than any other vulnerability database since it provides a universally understood name convention for common vulnerabilities.

The dataset utilized in this paper comes from the National Vulnerability Database (NVD). The dataset has 89660 records and 13 elements. The depiction of component factors can be seen as in **Table 4.1**. For the NVD, announced weaknesses are investigated and included a normalized design. In particular, a dataset passage contains the accompanying:

1) A Common Vulnerability Exposure (CVE) ID number that extraordinarily distinguishes the weakness.
2) The weakness passage's distribution date.
3) The weakness passage's adjusted date.
4) The weakness type/classification, as grouped by the Common Weakness Enumeration (CWE) code and the name related with the CWE code.
5) The name of the vulnerability in the CWE name is related to the CWE code.
6) The CVSS Score is specifically designed for assessing the severity of vulnerability.
7) The summary provides a description of the vulnerability.

ATTRIBUTE	DTYPE	DESCRIPTION
mod_date	Datetime	The date the section was last adjusted
pub_date	Datetime	The date the passage was distributed

Cvss	Float	Normal Vulnerability Scoring System (CVSS) score, a proportion of the seriousness of a weakness
cwe_code	Categorical	Common Weakness Enumeration (CWE) code, identifying the type of weakness
cwe_name	Categorical	The name associated with the CWE code
Summary	Str	A text summary of the vulnerability
access_authentication	Categorical	This measurement measures the times an assailant should verify to an objective to take advantage of weakness.
access_complexity	Categorical	This measurement mirrors the intricacy of the assault expected to take advantage of the weakness.
access_vector	Categorical	This metric reflects on how the vulnerability can be exploiting. The more remote vulnerability can be exploited the higher the rating.
impact_availability	Categorical	Accessibility alludes to on how data assets can be access. This measurement reflects accessibility of an effectively taken advantage of weakness.

impact_confidentiality	Categorical	This metric reflects information confidentiality of a successfully exploited vulnerability.
impact_integrity	Categorical	Integrity refers to the trustworthiness of the information. This metrics reflects integrity of a successfully exploited vulnerability.

Table 4.1: Attributes and its description

The CVSS Score is explicitly intended for surveying weakness to decide the seriousness of the weakness. Score goes from 1 to 10, with the issues having a score of 10 being the most extreme and the ones having a score of 1 being the most un-serious. **Table 4.2** shows the CVSS seriousness level edges.

Label	**Score**
Low	0.0–3.9
Medium	4.0–6.9
High	7.0–8.9
Critical	9.0–10.0

Table 4.2: Common Vulnerability Scoring System (CVSS)

NVD involves the CVSS standard for rating seriousness. The Common Vulnerability Scoring System (CVSS) is a free and open industry standard for evaluating the seriousness of PC

framework security weaknesses. CVSS endeavors to appoint weakness seriousness scores, permitting responders to focus on reactions and assets in view of danger. In PC security practice, this score can be utilized to rate security weaknesses and demonstrate how genuine the weaknesses are. Each CVSS is made out of three measurement gatherings: Base, Temporal, and Environmental, each comprising of a bunch of measurements.

The Base Metrics

The Base Metrics are made up of variety of components. **Table 4.3** lists these components and their descriptions.

S.No	Base Metrics Component	Explanation	Metric Value & Description
1	**Access Vector (AV)**	This measurement considers how the weakness can take advantage of. The more far off weakness can be taken advantage of the greater the rating.	**Network**: Can be exploit from a distance. **Adjacent**: Can be take advantage of remotely yet restricted to the equivalent physical or legitimate organization. **Local**: Not expect organization to take advantage of the weakness. It additionally can be taking advantage of by actually admittance to weakness.
2	**Access Complexity (AC)**	This measurement mirrors the intricacy of the assault expected to take advantage of the weakness.	**Low**: Specialized condition doesn't exist and an aggressor can anticipate repeatable accomplishment against the weak part. **Medium**: Specialized condition to some degree exist and an assailant should spend a measure of work to

			exploit the weak part. **High**: Specialized that's what condition exist make aggressor should spend a measure of work to exploit the weak part.
3	**Access Authentication (Au)**	This measurement estimates the times an aggressor should validate to an objective to take advantage of weakness.	**None**: Authentication does not expected to take advantage of weakness. **Single**: Authentication expect to take advantage of weakness. **Multiple**: Authentication required at least two in request to take advantage of weakness.
4	**Impact Confidentiality (IC)**	This measurement reflects data classification of an effectively taken advantage of weakness.	**None**: There is no deficiency of data privacy on the framework. **Partial**: There is some deficiency of data classification. **High**: There is all out loss of data classification.
5	**Impact Integrity (II)**	Honesty alludes to the dependability of the data. This measurement reflects honesty of an effectively taken advantage of weakness.	**None**: Modification of framework records is incomprehensible. **Partial**: Modification of some framework records is conceivable. **High**: Modification of whole framework records is conceivable.

6	**Impact Availability (IA)**	Accessibility alludes to on how data assets can be access. This measurement reflects accessibility of an effectively taken advantage of weakness.	**None**: Performance and assets doesn't influence. **Partial**: Modification of some system records is possible. **Complete**: Performance and assets can be control effectively by assailant.

Table 4.3: Base Metric Components

4.2.2 Data Pre-processing

Reviews of AI designers and information researchers show that the information assortment and readiness steps can take up to 80% of an AI task's time. As the adage goes, "garbage in, garbage out." The work invested prepping and purifying data is definitely worth it because machine learning models must learn from it. Data pre-processing techniques are used to improve the quality of the data so as to improve the result and efficiency of the clustering process. There are various forms of data pre-processing:

Exploratory Data Analysis

Exploratory Data Analysis (EDA) in Python is the initial phase in your information investigation process created by John Turkey during the 1970s. In insights, exploratory information investigation is a way to deal with breaking down informational collections to sum up their primary attributes, frequently with visual techniques. The fundamental objective of exploratory information examination is to get sufficient trust in your information to have the option to utilize an AI strategy. Exploratory Data Analysis is an important stage prior to continuing on toward AI or information demonstrating. This permits you to decide if they chose qualities are reasonable for demonstrating, whether each of the elements are essential, and whether any connections exist, permitting you to get back to the Data Pre-handling stage or continue to displaying.

4.2.3 Dimensionality Reduction (PCA)

Principal Component Analysis is a prominent dimensionality reduction approach (PCA). PCA is a multivariate analysis technique developed by Karl Pearson in 1901 and further improved by Harold Hotelling in 1933. The Dimensionality Reduction technique was utilised to break down the large problem into multiple smaller problems, which were then solved. It is difficult to extract value from three-dimensional data since it is unbalanced.

The issue of finding the result can be simplified by lowering the data dimensions. The goal of PCA is to minimise the dimensionality of a data set with a large number of linked features while keeping as much of the data set's variation as possible. These principal components are uncorrelated and linearly independent, with the first principal component retaining the majority of the variation in all of the original qualities. This outcome in the main head part being the hugest while the last being the most un-critical.

PCA processes the Eigen esteem deterioration of an information covariance lattice gauge and uses the essential Eigen vectors to project the component space to a lower-layered space. PCA is used for data exploration and predictive modelling. It is a technique for detecting hidden patterns in a dataset by reducing variances. It employs a feature extraction technique. It is a statistical approach that is frequently utilised in many different disciplines.

It entails conserving variability in order to discover novel variables that are linear functions of the original dataset's data points. The variance of each feature is determined through PCA. The high variance feature demonstrates a good split between the classes and hence decreases dimensionality. Image processing and movie recommendation systems both employ PCA. PCA considers the most essential traits and ignores the less important ones.

The pseudo code of PCA is given as follows:

Algorithm 2 Pseudo-Code of Principal Component Analysis (PCA)

```
1: Arrange the data into a structural form
2: Carry out the normalization of the data
3: Calculate the Covariance of Z
4: Determine the Eigen-Vectors and Eigen-Values
5: Carry out the sorting of the calculated Eigen-Vectors
6: Assess the Principal Components or new features
7: Remove the insignificant features from new dataset
```

4.2.4 Unsupervised Machine Learning Model Building

Utilizing an assortment of procedures and calculations, train the model to gain from the excellent information gathered. The three significant classes of AI are supervised learning, unsupervised learning, and reinforcement learning. The model is prepared on named information prior to being utilized to make expectations on unlabeled information in Supervised Learning. Unsupervised learning is the point at which a model is prepared on unlabeled information and afterward gains from it independently by removing highlights and examples. An agent is taught on the environment through Reinforcement Learning, which allows the agent to identify the best answer and achieve a goal in a complicated circumstance.

Unsupervised Machine Learning

Unsupervised learning refers to algorithms that find patterns in data sets with no labels or classifications. As a result, the algorithms are able to categorise, label, and organise the data points inside the data sets without requiring any external assistance. The model does not require user supervision. In unsupervised learning, an AI system will categorise unsorted data based on similarities and differences even if no categories are given. The basic goal of unsupervised learning is to uncover hidden and interesting patterns in unlabelled data. The general architecture of unsupervised learning is depicted in the diagram below.

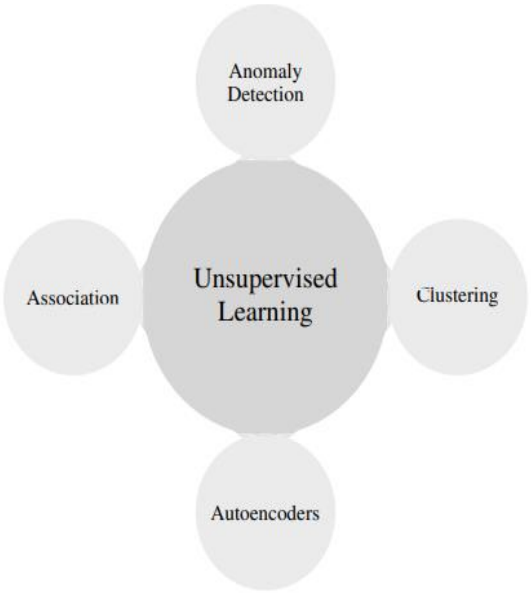


Figure 4.3: Sorts of Unsupervised Learning

Clustering

Clustering is a central idea in unsupervised learning. Its primary objective is to find a design or example in a lot of uncategorized information. Unsupervised Learning Clustering calculations will handle your information and track down regular clusters (gatherings) in the event that they exist in the information. Clustering is the main unsupervised learning task since it involves perceiving a design in a bunch of unlabeled information. It is the most common way of lumping information into bunches with hubs connected somehow or another. As outlined in **Figure 4.4**, a cluster is an assortment of items that are comparable among themselves yet "unique" to objects from different groups.

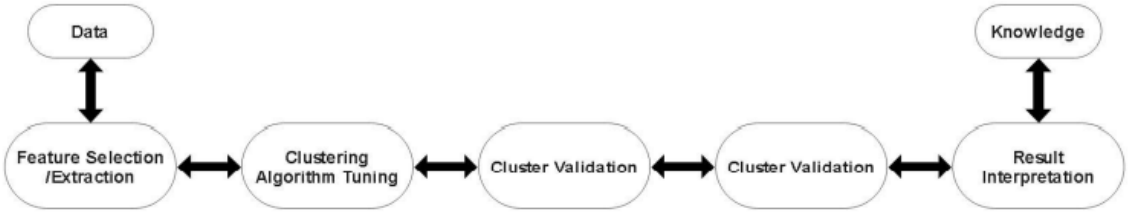


Figure 4.4: Workflow of clustering in unsupervised ML

The groups are defined based on their similarities.

Clustering can be divided into several types, including partitioning, hierarchical, overlapping, and probabilistic. The various types of clustering are depicted in **Figure 4.5**. Dividing clusters information with the goal that every information point can have a place with one group. It is also referred to as exclusive clustering. Partitioning clustering is demonstrated by K-means. Each piece of information is a cluster in hierarchical clustering. Data is clustered using overlapping fuzzy sets. Each point might be an individual from at least two bunches with shifting levels of enrolment. Information will be related with a fitting participation esteem, like Fuzzy C-Means, for this situation. Finally, the probabilistic creates clusters using probability distribution.

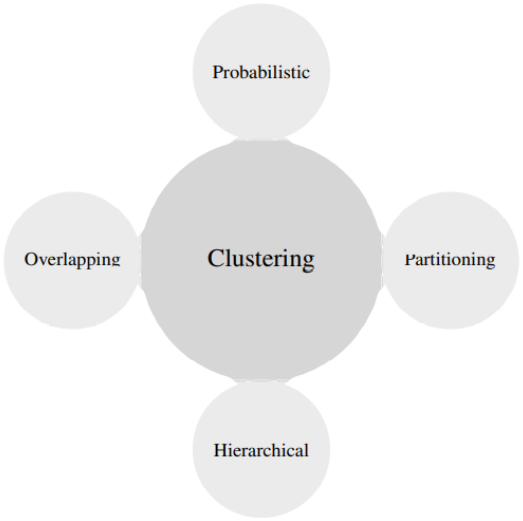


Figure 4.5: Types of Clustering

Association

Because the data is not labelled, the algorithm attempts to learn without the assistance of a teacher using the association rule. The rule is descriptive, not predictive, as is commonly used to uncover exciting relationships hidden in large datasets. The relationship is typically represented by rules or frequent item sets. Association rules mining is used to discover new and exciting connections between objects in a set, frequent pattern in transactional data or any relational database.

Anomaly Detection

Anomaly detection refers to any process that identifies outliers in a dataset. These anomalies could indicate unusual network traffic, a faulty sensor, or data that needs to be cleaned before analysis. When data patterns go beyond or deviate from normal patterns, this is referred to as an anomaly. For instance, an unusual network traffic pattern indicates that the compromised system is sending sensitive data to an unauthorised host. Anomalies are detected by identifying or forecasting data points that deviate from the normal pattern.

Autoencoders

Autoencoders are a solo learning procedure that use the abilities of brain networks for the undertaking of portrayal learning. Autoencoders are a specific type of feed forward neural

network where the input is the same as the output. They compress the input into a lower-dimensional code and then reconstruct the output from this representation. An auto encoder is made up of three parts: encoder, code, and decoder. The encoder compresses the input and generates the code, which the decoder then uses to reconstruct the input. An auto encoder requires three components: an encoding method, a decoding method, and a loss function to compare the output to the target. Autoencoders are dimensionality reduction (or compression) algorithms at their core.

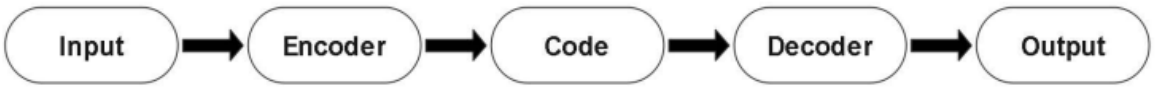


Figure 4.6: Autoencoder Components

This project includes a broad overview of unsupervised learning methodologies and algorithms, as well as the metrics used to evaluate each unsupervised learning model's performance and a comparison of the models' accuracy.

- As we have a unsupervised issue, clustering is more useful than different calculations, so utilizing this is more reasonable in finding regular gatherings in the component space of info information.
- Grouping is utilized as an information investigation method for finding fascinating examples with regards to information.
- The Scikit-learn library gives a set-up of various clustering issues to look over.
- It is hugely important as it determines the intrinsic grouping among the unlabelled data present.

Formal impediments of clustering

- According to a natural perspective, the clustering issue has an extremely clear objective; appropriately bunching a bunch of unlabeled information.
- Notwithstanding its undeniable allure, the idea of "cluster " is hard to depict, which makes sense of the huge variety of clustering strategies advertised.

K-means Clustering

K-Means Clustering is a calculation for unsupervised learning. The unlabeled information is isolated into various bunches, with the quantity of gatherings indicated by "K". K can be any number; for example, K=2 will result in two clusters, while K=5 will result in five clusters. It's a time-consuming algorithm that divides an unlabelled dataset into K clusters. Each dataset is part of a single group with similar qualities. It facilitates the data to be broken down into different groups. K-means is a notable grouping procedure that has been being used since it was first distributed autonomously by a few scholastics. Its popularity stems from the algorithm's simplicity, effectiveness, and ease of implementation.

The following is the pseudo-code for K-mean clustering:

Algorithm 1 Pseudo Code of K-means Clustering

1: Select the number "K" to define the number of clusters
2: Choose random K points or centroids
3: Assign all data points to their nearest centroid to create predetermined K clusters
4: Calculate the variance and enter a new centroid of each cluster
5: Repeat the 3rd step and keep reassigning each data point to the closest centroid of latest cluster
6: **If:** "any reassignment happens, then perform Step 4"
7: **Else:** "End"

K-Medoids Clustering:

- K-medoids come under partitioning-based clustering.
- It is also called PAM (Partitioning Around Medoids).
- In this method, the algorithms provide partitioning of data with the objective of minimising the distance between the point to be partitioned and the medoids.
- Unlike k-means, k-medoids select actual observations from the data as the centre of the cluster.

The Benefits of Using K-Medoids Clustering:

- It is more adaptable and can be used with any similarity measure, whereas K-means requires Euclidean distance to produce efficient results.

- It minimizes the sum of pair-wise dissimilarities rather than the total of squared Euclidean distances.
- Despite k-means, it is less prone to noise and outliers.

Limitations

- It is not suitable for clustering non-spherical shaped groups of observations as they use compactness as a clustering criterion.
- It cannot be used to scale up for large datasets due to the high computational overhead.
- One has to choose the value of the number of clusters in advance as an input to the algorithm.

Hierarchical Clustering

Rather than delivering a level parcel of information, it is as often as possible helpful to create an order of ideas by creating a cluster of settled groups that can be sorted out to shape a tree structure. While parceled grouping approaches stand out enough to be noticed in late exploration, various levelled bunching calculations have for some time been the prevailing strategy for record bunching as text assortments as a rule contain expansive subjects that can be naturally partitioned into additional pertinent issues. Hierarchical calculations are for the most part coordinated into two unmistakable classes:

Agglomerative:

Begin with each article being allotted to a singleton cluster. Apply a base up technique, combining the most practically identical sets of groups at each stage.

Divisive:

Begin with a solitary group that contains the things in general. Apply a hierarchical strategy wherein a chose group is parted into two sub-clusters at each stage. Regardless, the subsequent ordered progression can be pictured utilizing a dendrogram, a treelike design that has nodes for each cluster made by the clustering technique, as well as cluster relations that show the merge or split operations conducted throughout the clustering process.

It merits seeing that the similitude between the picked sets of clusters lessens with each union cycle. Not at all like other apportioned calculations, which require a client to give an incentive for the quantity of clusters k ahead of time, various levelled techniques permit a client to physically choose k by breaking down the produced dendrogram and deciding a suitable endpoint. For example, we can obtain a clustering of the data for k = 2 from the two leaf nodes at that level by cutting the tree in Figure 1.2 at the level specified.

Agglomerative Hierarchical Clustering

The most common way of developing a tree of bunches from the base up is known as agglomerative hierarchical grouping (AHC). However, we concentrate on the conventional formulation, which is extensively used in a variety of fields and goes as follows:

- Each object should be assigned to a singleton cluster.
- The pair wise inter-cluster similarity matrix should be updated.
- Find the most comparable clusters and merge them.
- Rehash Step 2 until only one bunch remains or the predefined end rule is met.

Whenever the quantity of clusters k is assessed ahead of time, the system might be finished when the dendrogram contains the expected number of leaf hubs. There are a few linkage calculations for figuring out which sets of groups ought to be converged out of the multitude of potential outcomes. While these strategies are typically characterized as far as distances, they can promptly be changed over completely to utilize similitude values delivered by the cosine measure.

Single linkage:
Cluster likeness is characterized as the greatest closeness between an item given to Ca and an article allotted to Cb in the most prevalent strategy, known colloquially as the closest neighbor technique:

$$sim(C_a, C_b) = \max_{x_i \in C_a, x_j \in C_b} S_{ij} \longrightarrow (1)$$

While this method is widely used, it is vulnerable to "chaining," which occurs when singletons are repeatedly merged with an existing cluster, resulting in a single gigantic, extended cluster with substantially separate objects at each end.

Complete linkage:

Minimal likeness between an article allotted to Ca and an item relegated to Cb is characterized as the similitude between two groups (Ca, Cb):

$$sim(C_a, C_b) = \min_{x_i \in C_a, x_j \in C_b} S_{ij} \longrightarrow (2)$$

This technique favors clusters that are densely packed and firmly connected, what's more, it is much of the time delicate to the presence of exceptions.

Average linkage:

The comparability between a couple of groups (Ca, Cb) is determined as the mean similitude between objects allotted to Ca and items allocated to Cb:

$$sim(C_a, C_b) = \frac{\sum_{x_i \in C_a} \sum_{x_j \in C_b} S_{ij}}{|C_a|\,|C_b|} \longrightarrow (3)$$

Since normalizing by cluster size gives indistinguishable loads to objects allotted to bunches of fluctuating sizes, this system is frequently alluded to as the unweighted pair group method using arithmetic averages (UPGMA). Obviously, the coupling approach picked significantly affects the design of the clusters made by AHC. Accordingly, choosing a decent system ahead of time for a given dataset might be a non-unimportant boundary determination challenge. Practically speaking, a client could build various ordered progressions utilizing different techniques and afterward physically survey the outcomes to track down the most appropriate response.

Agglomerative is a type of hierarchal clustering in which each data point is considered a single cluster. Afterward, these clusters are successively united or agglomerated. It is a bottom-up approach. A dendrogram represents the hierarchy of the clusters. This method starts with each sample being a different cluster and then merges them by the ones closest to each other until there is only one cluster. It can be further divided into a single linkage and a complete linkage.

The pseudo code of Agglomerative is as follows:

Algorithm 3 Pseudo-Code of Agglomerative

1: Take each data point as a single cluster, such as K clusters as the number of the data points is K at the start
2: Merge two closest data points to make a big cluster (K-1 clusters will be formed)
3: Afterwards, in order to make more clusters, merge two closest clusters (K-2 clusters will be formed)
4: Continue creating big cluster unless no data point is left for joining or K becomes zero
5: Finally, after creating a single big cluster, the dendrograms are distributed into several clusters

Limitations of Agglomerative Algorithms

The precision of the last arrangement can be firmly affected by misguided decisions made from the get-go in the grouping system, which is a significant disadvantage of typical agglomerative methods. Several potential mergers may appear to be equally viable in the absence of the application of a global objective function.

Once a merging decision has been taken, there is no way to go back and correct an incorrect decision. On the contrary, as the clustering process progresses, the negative consequences of these actions are frequently exaggerated. Beside grouping precision hardships, progressive bunching calculations are frequently undeniably more computationally costly than their partitional partners, with a period intricacy of O (n 3).

4.2.5 Performance Evaluation

According to our survey, researchers utilise a variety of evaluation measures to assess the efficacy of various clustering and dimensionality reduction techniques.

A selection of commonly used evaluation metrics is presented in **Table 4.4**.

Evaluation Metric	Definition	Formula
Silhouette Coefficient	The Silhouette Coefficient is the most widely recognized method for consolidating the measurements of Cohesion and division in a solitary measure.	$S = \frac{1}{N}\sum_{i=1}^{N} s(i)$ where: **N**: Number of data points in the same cluster, **S(i):** Data point in the cluster, i = 1,2, 3…. n,
Davies-Bouldin Index	Davies-Bouldin list is determined as the normal likeness of each bunch with a group generally like it.	$DB \equiv \frac{1}{N}\sum_{i=1}^{N} D_i$ Where: **N**: Total number of clusters **Di**: Similarity measure of each cluster.

Table 4.4: Performance Evaluation Metrics

A crucial element of the clustering data process is evaluating the outcomes of a clustering algorithm. The assessment of the subsequent characterization model is an urgent piece of the managed educational experience, and there are an assortment of assessment strategies and methodology to browse. Cluster evaluation, otherwise called cluster validation, is less evolved in unsupervised learning because of its temperament. Surveying the nature of a grouping calculation in bunching situations is troublesome.

This brings about various assessment methods. Frequently, the assessment cycle incorporates an infamous distinction: how the estimation is performed relies upon the calculation used to acquire the clustering results. While investigating clustering results, a few angles should be considered for the approval of the calculation results:

- deciding the clustering propensity in the information.
- deciding the right number of clusters.
- evaluating the nature of the clustering results without outer data.
- contrasting the outcomes with outer information; and
- Figuring out which of two arrangements of clusters is prevalent.

Silhouette coefficient

The silhouette coefficient is the most commonly used method for combining cohesion and separation statistics into a single statistic. The typical distance a(i) between every model and everything different occurrences in a similar cluster is determined:

$$a(i) = \frac{1}{|C_a|} \sum_{j \in C_a, i \neq j} d(i,j) \longrightarrow (4)$$

where:

a(i): It is the average distance between i and all the other data points in the cluster to which i belongs. For every model, the base typical distance b(i) between the model and the models contained in each bunch not containing the investigated model:

$$b(i) = \min_{C_b \neq C_a} \frac{1}{|C_b|} \sum_{j \in C_b} d(i,j) \longrightarrow (5)$$

where:

b(i): It is the average distance from i to all clusters to which i does not belong. For every model, the outline not entirely set in stone by the accompanying articulation:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \longrightarrow (6)$$

where:

s(i): It is the silhouette coefficient of the data point i. For every model in our informational collection, the outline coefficient is characterized in the stretch [-1, 1]. The worldwide outline coefficient is basically the normal of the outline coefficients for every individual model:

$$S = \frac{1}{n}\sum_{i=1}^{n} s(i) \longrightarrow (7)$$

Rather than other consolidated measures, the outline coefficient gives a clear structure to capability. Positive qualities show that groups are very much isolated. Negative qualities show that the groups are combined as one. At the point when the outline coefficient is zero, it shows that the information is conveyed consistently across Euclidean space. Tragically, the outline coefficient's high computational intricacy, O (dn2), makes it unreasonable while managing enormous informational indexes.

Davies-Bouldin index

A connected inward approval method was proposed in that considers the proportion of intra-cluster dissipate to between bunch detachability across all k gatherings in a clustering. Officially, the DB file is characterized as a component of each group's vicinity to its closest neighbor:

$$DB(\mathcal{C}) = \frac{1}{k}\sum_{i=1}^{k}\max_{i\neq j}\left\{\frac{\Delta(C_i)+\Delta(C_j)}{\delta(C_i,C_j)}\right\} \longrightarrow (8)$$

As clusters become more conservative and particular, this worth will diminish, making more modest qualities for this list attractive.

The DB record has a critical detriment in that it comes up short on fixed range, with a result esteem simply restricted to being non-negative, making understanding troublesome. Moreover, observational proof proposes that while endeavoring to choose k, this file will in general misjudge the quantity of gatherings, particularly for feebly grouped information.

RESULTS AND DISCUSSIONS

CHAPTER 5

RESULTS AND DISCUSSIONS

This undertaking depends on the weaknesses gathered from the National Vulnerability Database (NVD), which is a data set of weaknesses. NVD stores weaknesses with CVE values, which incorporate weakness, shortcoming, and openness definitions. The uniqueness of the issues is safeguarded by utilizing one-of-a-kind identifiers (CVE numbers) from the CVE list. CVSS is used to calculate the severity of each vulnerability mentioned in NVD (Common Vulnerability Scoring System). The performance of this project can be done in five phases:

Phase 1: Data Acquisition

The most obvious and first thing we need to comprehend is data, to make sense of, perform any analysis, and to interpret it. Data acquisition is the foundation for the project and most significant step. This is used to collect data from relevant sources before it can be stored, cleaned, pre-processed, and used for further mechanisms. While gathering data, make sure to have enough features to properly train learning model. The more information you have the better the outcomes, so make a point to accompany an adequate number of columns. Once acquired the appropriate data, it’s time to prepare it. Filtering, manipulation, sampling, scaling and reducing are multiple transformation operations used further to make the best.

The accuracy of your model is determined by the quality of the data you provide the machine. If your data is faulty or old, you will get inaccurate results or predictions that are irrelevant. The data in this study comes from the National Vulnerability Database (NVD), which is kept up with by NIST. Make sure you use data from a reputable source, as it will have a direct impact on the model's conclusion. Good data is meaningful, contains few missing and duplicated values, and accurately represents the many subcategories and groups.

- The vulnerabilities detected in software products are represented by the samples in the dataset.
- So, in machine learning terms, we're dealing with an unsupervised learning problem.
- There are numerous approaches to creating such a model, but we will focus on clustering techniques.

Phase 2: Data Pre-Processing

The data set must be inspected, with data removed and altered as needed. The information is then completely dissected to furnish us with understanding into the informational collection and permit us to decipher the information accurately.

Following that, we use PCA to perform dimensionality reduction. We bunch our information utilizing k-means, K-medoids, and the AHC calculation with a lower aspect. We put our method to the test with clusters by using a silhouette coefficient and the Davies-Bouldin Score. At long last, we will examine our discoveries, expect future turns of events, and think about our work.

- The second phase of this project will involve preparing the data for vulnerability analysis.
- The first step is to make the data compatible with the subsequent phases.
- Rows are frequently referred to as samples, and columns as features.
- After generating the dataset, it must be cleaned up, i.e., the irrelevant information must be operated.
- Originally, all null (NA) rows were eliminated.
- Data integration was not necessary because the CVE repository is a single data file. Because the CVE repository is small, no data reduction was performed.
- Categorical variables are present in the dataset. So, I'd like to convert this to numerical form because the machine cannot process categorical variables to produce results.
- To convert categorical variables to numeric form, label encoding was utilised.
- As we can see, the imputed data corresponds well to the values.
- The EDA (Exploratory Data Analysis) method was then used to examine the dataset at a high level.
- Visualization is a fantastic tool for identifying patterns and trends in data, which leads to a better understanding and the discovery of key insights.
- The graphical illustration of data visualisation also aids in faster decision making.
- Visualize the data to see how it's organised, the links between the different variables and classes, and any connections between the different attributes we discovered.
- The severity distribution of vulnerabilities was investigated using the CVSS score.
- To gain a complete grasp of the data, each attribute value is reviewed in depth.

- Analysing the data is another critical step before developing the model. This is a crucial stage in understanding how to interpret data, or how to understand what the data means and how to interpret our final outcomes.
- The informational collection is shown and the highlights are analysed in this review.

Phase 3: Dimensionality Reduction (PCA)

K-means, K-medoids, and Agglomerative Hierarchical Clustering algorithms were used to cluster unsupervised data. Cluster formation requires a minimum of two samples (PCA1, PCA2). K-means was applied to all of the Principal Component Analysis projections, yielding a most extreme difference with a dimensionality decrease of 2.

As a result, the Clusters created utilizing the K-means Algorithm with the Principal Components aspect brought down to 2 were considered awesome. The K-means calculation isolates a dataset into k foreordained sets, with each guide having a place toward one of them. This calculation expects to keep data of interest in the bunch as near one another as conceivable by keeping the number of squared distances between the point and the group's centroid as little as could be expected.

The cluster is more homogeneous on the off chance that the fluctuation is less between the focuses in a similar group. The amount of all distances between useful pieces of information and centroids is then processed by the K-means calculation, and every information point is allocated to the closest centroid. To determine the best potential cluster, K-means method uses the Expectation-Maximization approach.

We created clusters based on the cve name and CVSS using K-means. The Primary Components Analysis (PCA) technique was then used to extract principal components. Clusters of head parts were framed utilizing an AI calculation like k-means. Novel variations are developed using principal component analysis and k-means clustering, and data from the National Vulnerability Database (NVD) is utilised.

Due of the corresponded segments, a dimensionality decrease strategy ought to be utilized. We chose to utilize Principal Component Analysis (PCA) to investigate our information because it has the added benefit of suppressing noise. In related research, this has been shown to produce superior results in vulnerability detection experiments. The data must first be scaled before PCA can be applied to it. Since the greatness of the information in every section

shift, this scaling is required while utilizing PCA. PCA limits dimensionality while protecting however much fluctuation as could reasonably be expected to guarantee insignificant data misfortune. A segment with a higher greatness has a greater difference, bringing about a bother some conduct where PCA is one-sided toward the section with the higher extent, i.e., a higher weight.

While settling on the quantity of primary parts, PCA utilized a 95 percent least fluctuation as a measure. To decrease the commotion, 5% of the variety is forgotten about. Out of the 10 segments in the cleaned informational index, the PCA procedure yielded two head parts. The change dissemination is displayed in **figure 5.1**, and the intensity map grid of the vital parts coefficients, i.e., the connection between the primary parts and the cleaned informational index segments, is displayed in **figure 5.2.**

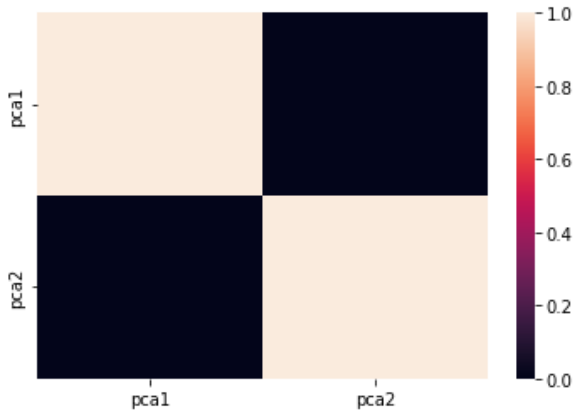


Figure 5.1: Heat map of principal components

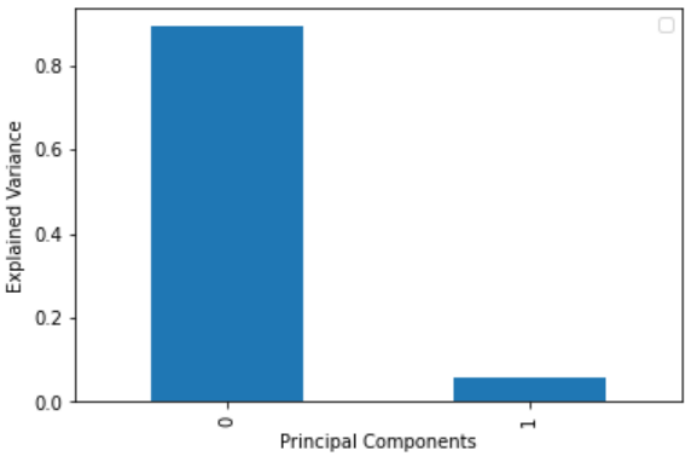


Figure 5.2: Variance in each principal component

Phase 4: Unsupervised Machine Learning Model Building

- In this phase, the data is ready for applying algorithms.
- As it is an unlabelled data, it doesn't have a target variable so, clustering algorithms is more suitable than other algorithm techniques.
- As we have a unsupervised issue, clustering is utilized in finding normal gatherings in the component space of information.
- Clustering works on dataset in which there is no target variable i.e., Unlabelled data.
- It is more helpful than others as our aim is for analysis purpose, in order to learn more about the problem domain so called pattern discovery or knowledge discovery.
- Scikit-learn library gives a set-up of various clustering calculations to browse.
- Among all clustering algorithms, k-means, k-medoids and agglomerative hierarchical clustering were used.

Application of the k-means algorithm

With dataset, the model can be created by initializing the k-means algorithm. But first, an appropriate k needs to be chosen as the k-means algorithm needs this parameter to initialize.

The elbow method is a good start for finding the proper k. This elbow method could normally also be used for choosing the number of principal components, In figure 4.3, there is an elbow to choose the number of principal components.

Elbow method

In any unsupervised technique, determining the optimal number of clusters into which the data can be divided is critical. The Elbow Method is one of the most well-known criteria for determining the ideal value of k.

The elbow approach is a strategy for finding the best parameter in algorithms like k-means and PCA. The objective is to utilize K-Means to sort out the ideal number of clusters for different group sizes.

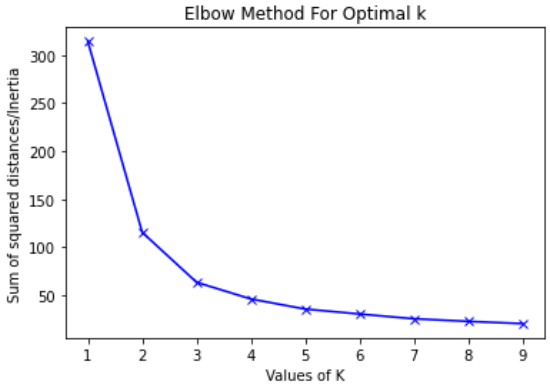


Figure5.3: Elbow method

Getting going with a cleaned informational collection, we initially scaled informational index by normalization so there won't be any predisposition while applying PCA. From that point forward, we applied PCA to decrease the aspects in our informational index. PCA additionally diminished some noise when it leaved out 5% change out of the informational collection. To find the sensible explanation for the groups, we checked out at the base worth, most extreme worth and normal incentive for every section of each cluster and derived the thinking behind the clusters.

K-Means Clustering:

- The k-means calculation produces bunches. These clusters are grouped by similarities in terms of values for each feature of each data point.
- This calculation can deliver various results, as this algorithm requires randomization during initialization. Accordingly, the similitudes on which the groups are based might be different for every execution of the calculation.
- Since the calculation doesn't have any idea what each component implies, how groups are bunched in the way that they are might be wanted or undesirable, expected or unforeseen.
- It is up to them to determine yet if the clusters are plausible for our given problem, which may necessitate running the grouping calculation on numerous occasions. Accordingly, it is basic to approve the groups created prior to continuing to the subsequent stage. The calculation's means are as per the following:

1. Choose various k clusters to segment n data of interest into.
2. Initialize k cluster centroids arbitrarily by choosing focuses in the space as the data of interest.

3. For every data of interest, process the blunder for each cluster centroid. Appoint the information highlight the group of the cluster centroid with the base mistake.
4. For each cluster, figure the mean of the multitude of useful pieces of information in the group. This implies this is the new bunch centroid.
5. Repeat stages 3 and 4 until the new cluster centroids don't change.

Officially talking, the k-means calculations limit the number of squared mistakes.

Algorithm: K means Algorithm

Result: Find k clusters using K-means

- X← {x1, x2, x3...xn} V ← {v1, v2, v3...vk} (a set of centroids chosen at random)
- Choose k centroids at random.
- Ascertain the distance between every piece of information while information focuses are reassigned. Appoint information focuses to the centroid with a base distance of
- Recalculate the new group utilizing: vi = 1 ki Pki j = 1 xi (where ki addresses the quantity of data of interest in the ith cluster)
- Recalculate the distance between every data of interest and the new centroid end.
- It shows the clusters acquired utilizing K-means. Examination of our outcomes, shows the conveyance of seriousness in various spaces, similar to Memory and Buffer Overflow weaknesses, and Network and Authentication weaknesses.

K-Medoids clustering:

- K-Medoids is a grouping calculation that works similarly that K-Means does.
- The manner in which it picks group focuses shifts fiercely from the K-Means calculation.
- The former takes the average of a cluster's points as its centre, whereas the latter always chooses the actual data points from the clusters as their centres.
- Thus, the K-medoids calculation is more commotion lenient than the K-means calculation.

Algorithm

Step1: Initialize k bunches in the given information space D.

Step2: Randomly pick k items from n objects in information and appoint k items to k groups

with the end goal that each article is allotted to one and only one bunch. Subsequently, it turns into an underlying medoids for each group.

Step3: For all leftover non-medoid objects, figure the Cost (distance as registered by means of Euclidean, Manhattan, or Chebyshev strategies) from all medoids.

Step4: Now, assign each leftover non-medoid object to that bunch whose medoid distance to that item is least when contrasted with different groups medoid.

Step5: Compute the complete expense for example it is the complete amount of all the non-medoid objects distance from its bunch medoid and appoint it to dj.

Step6: Randomly select a non-medoid object i.

Step7: Now, impermanent trade the article I with medoid j and Repeat Step5 to recalculate complete expense and allocate it to di.

Step8: If di<dj then make the transitory trade in Step7 long-lasting to shape the new arrangement of k medoid. Else fix the brief trade done in Step 7.

Step9: Repeat Step 4, Step 5, Step 6, Step 7, Step 8. Until no change;

Agglomerative Hierarchical Clustering:

In information mining and measurements, progressive grouping investigation is a strategy for bunch examination that looks to fabricate a pecking order of groups i.e., tree-type structure in light of the pecking order.

The agglomerative bunching is the most well-known kind of various leveled bunching used to bunch objects in groups in light of their closeness. It's otherwise called AGNES (Agglomerative Nesting). The calculation begins by regarding each item as a singleton bunch. Then, sets of bunches are progressively converged until all groups have been converged into one major group containing all articles. The outcome is a tree-based portrayal of the articles, named dendrogram.

Agglomerative Clustering otherwise called granular perspective or various levelled agglomerative bunching (HAC). A design that is more enlightening than the unstructured arrangement of groups returned by level bunching. This grouping calculation doesn't expect us to prespecify the quantity of bunches.

How it works:

1. The procedure begins with determining the dissimilarity between the N objects.
2. Next, two objects that, when clustered together, minimise a specific agglomeration criterion are clustered together, resulting in the creation of a class that includes these two objects.
3. The agglomeration criterion is then used to compute the dissimilarity between this class and the N-2 other objects.
4. The two things or classes of objects whose grouping reduces the agglomeration criterion are subsequently grouped together.
5. Repeat step 4 until all of the objects have been grouped.

Phase 5: Performance Evaluation

In this phase, evaluation was done based on how it performed on applying the algorithms and how well the clusters formed basis. For evaluation, we picked the two most popular methods such as silhouette coefficient, which is most effective for performance evaluation compared to other methods. Another one is Davies-Bouldin index.

Silhouette Coefficient:

This method is used because the ground truth labels are not known, evaluation must be performed using the model itself. This is an example for such evaluation, where a higher silhouette score relates to a model with better clusters.

This methos is performed with the module Sklearn.metrics.Silhouette_Score

This technique is characterized for each example in the cluster and it is made out of two scores. They are,

1. The mean distance between an example and different focuses in a similar cluster.
2. The mean distance between an example and any remaining focuses in the following closest cluster.

The silhouette coefficient for a bunch of tests is given as the mean of the outline for each example.

The silhouette score indicates:

- -1 refers, the clusters are not formed well or it is the indication of incorrect clustering.
- +1 refers the clusters formed well and it indicates highly dense clustering.
- 0 refers to the clusters are overlapped.

Through silhouette coefficient, the evaluation results we obtained through k-means is 0.98. which approximately equals to 1. And through k-medoids the score is 0.54. And, through agglomerative clustering we got a score of 0.30.

	Model	silhouette_score
0	KMeans	0.980797
1	KMedoids	0.545023
2	AgglomerativeClustering	0.303567

Figure 5.4: Results Obtained through silhouette coefficient

By comparing the three algorithms performance, K-means topped among other two algorithms with a score of 0.98 which indicates it formed highly densed clusters and provides a formation of good clustering. Through the evaluation, we came to a conclusion that, k-means is more suitable algorithm and produce a good result among other two algorithms performed.

Davies-Bouldin Index:

As the ground truth names are not known, DB record can be utilized to assess the model, where lower score connects with a model with better partition between the groups.

This means the typical comparability between groups, where likeness is an action that contrasts the distance among bunch and the size of bunches themselves.

0 is the least conceivable score, esteems more like zero demonstrates a superior parcel. Through DB index, the evaluation results we obtained through k-means is 0.064. And through k-medoids the score is 1.55. And, through agglomerative clustering we got a score of 1.37.

	Model	davies_bouldin_score
0	KMeans	0.064826
1	KMedoids	1.552862
2	AgglomerativeClustering	1.379946

Figure 5.5: Results Obtained through Davies-Bouldin Index

By comparing the three algorithms performance, in this also K-means topped among other two algorithms with a score of 0.06 which has a least possible score that indicates the better cluster formation. Through the evaluation, we came to a conclusion that, k-means is more suitable algorithm and produce a good result among other two algorithms performed.

CONCLUSION AND SCOPE FOR FUTURE ENHANCEMENT

CHAPTER 6

CONCLUSION AND SCOPE FOR FUTURE ENHANCEMENT

The main intention of this project is to check the quality of the clusters formed through clustering algorithms and it is not an easy problem to solve. The conclusion was that the k-means technique performed better than other methods. The escalating cyber danger was the driving force behind this effort. In this project, we examined the security risks that machine learning may potentially solve, as well as how machine learning with a data-driven model can fight against security attacks.

The aim here is to work on cyber security data science, which has shown some good results compared to traditional systems. Although we cannot recommend a single algorithm as the best because the algorithms outperformed each other in different aspects of security and incidents. We need to tailor the algorithm according to our needs. The main intention behind this project is to provide an understanding, through conceptualizing and modelling, of how data science could be potentially used in the cyber security domain.

We pre-processed and label encoded the dataset in this project. Then, to analyse vulnerabilities in source code, we used a few unsupervised clustering techniques. With extensive experiments, it is shown that K-means clustering achieve impressive results in analysing vulnerabilities in code. By learning about vulnerabilities in the dataset, this project provides an ideal solution to organisations with the goal of enhancing vulnerability analysis efficacy.

The proposed model gives improved performance evaluation scores, according to the empirical observations. The most accurate method was K-means, which was considered and used. The system concludes that the K-Means clustering technique is more accurate than the processes after doing the comparison. As a result, the K-Means algorithm looks to be the most efficient method for analysing vulnerabilities exactly. According to the study, the majority of the researchers employed deep learning and natural language processing approaches. There are numerous ways to incorporate this experiment into future work.

This project's future enhancements could include a strategy for integrating more and more technologies that will be integrated with new domains in the future to perform automated vulnerability analysis on a large-scale and cross-architecture basis.

REFERENCES

CHAPTER 7

REFERENCES

- Armerding, Taylor. “What Is CVE, Its Definition and Purpose?” CSO Online, CSO, 10 July 2017, www.csoonline.com/article/3204884
- “About CWE”, Common Weakness Enumeration, September 26, 2007, Available: http://cwe.mitre.org
- Common Vulnerabilities and Exposures. Available online: https://cve.mitre.org
- “CWE- Common Weakness Enumeration”, National Vulnerability Database, Available: http://nvd.nist.gov/cwe.cfm
- “CVSS- A complete Guide to the Common Vulnerability Scoring System Version 2.0”, FIRST: Forum of Incident Response and Security Teams, Available: http://www.first.org/cvss
- Ghaffarian, S.M.; Shahriari, H.R. Software vulnerability analysis and discovery using machine-learning and data-mining techniques: A survey. ACM Comput. Surv. 2017, 50, 56. dx.doi.org
- MITRE, Common Weakness Enumeration. https://cwe.mitre.org
- National Vulnerability Database. Available online: https://nvd.nist.gov
- “NVD Common Vulnerability Scoring System Support v2”, National Vulnerability Database, June 20, 2007, Available: http:// nvd.nist.gov
- “NVD Data Feed and Product Integration”, National Vulnerability Database, Available: http://nvd.nist.gov/download.cfm
- “OSVDB: The Open Source Vulnerability Database”, OSVDB, Available: http://osvdb.org/
- S Christey and R. A. Martin, “Vulnerability Type Distributions in CVE”, Common Weakness Enumeration- A Community-Developed Dictionary of Software Weakness Types, May 22, 2007, Available: http://cwe.mitre.org/documents/vuln-trends
- Y. Y. Chang, P. Zavarsky, R. Ruhl and D Lindskog, “Trend Analysis of Common CVE Vulnerability Types”, Concordia University College of Alberta, May 2011.

ANNEXURE

CHAPTER 9

ANNEXURE

9.1 Coding and Screenshots

```
from mpl_toolkits. mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt # plotting
import numpy as np
import os
import pandas as pd
import seaborn as sns
from glob import glob
```

```
df['access_authentication'].value_counts()
sns.countplot(data=df, x='access_authentication', palette='summer');
```

```
sns.countplot(data=df, x='access_authentication', palette='summer');
sns.countplot(data= df, x='access_complexity', palette='rocket', hue='access_complexity');
sns.countplot(data= df, x='access_complexity', palette='rocket', hue='access_complexity');
df['access_vector'].value_counts()
```

```
import scipy.optimize as opt
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from datetime import timedelta
```

```
import scipy.stats as sps
from glob import glob
df.pub_date = pd.to_datetime(df.pub_date)

X = df.pub_date.dt.to_period('Q').sort_index().value_counts()
X.index = X.index.to_timestamp()
X = X.sort_index()
X[X.index] = np.cumsum(X.values)

rolling = df.pub_date.dt.to_period('M').sort_index().value_counts()
rolling.index = rolling.index.to_timestamp()
rolling = rolling.sort_index()
rolling[rolling.index] = np.cumsum(rolling.values)
rolling = rolling.rolling(12, axis=0).sum().pct_change(axis=0)

fig = make_subplots (rows=1, cols=2)

fig.add_trace(
go.Bar(
        x=X.index[:-20],
        y=X.values[:-20],
marker_color="#f4829f",
    ),
    row=1,
    col=1
)

fig.add_trace(
```

```
go.Bar(
        x=X.index[-20:],
        y=X.values[-20:],
marker_color="#ae45f1"
    ),
    row=1,
    col=1
)

fig.add_trace(
go.Bar(
        x=X.index.map(lambda x: "{}-{}".format(x-1,x)),
        y=X.values/np.sum(X.values)*100,
marker_color=['#708090', '#708090', '#708090', '#DC143C', '#DC143C', '#DC143C',
'#DC143C', '#708090', '#708090', '#708090'],
        text=np.vectorize(lambda x: str(x) + "%")(np.round((X.values/np.sum(X.values) *
100),1)),
textposition='outside'
))

fig.update_layout(
    title=dict(
        text="Severity Distribution",
xref="paper",
        x=0., y=1.
    ),
    font=dict(
        family="Times New Roman",
        size=14,
```

```
color="#000607"
    ),
xaxis=dict(
showgrid=False,
    ),
yaxis=dict(
showgrid=False,
showticklabels=False
    ),
    annotations=[
dict(
xref='paper',
yref='paper',
          x=0., y=1.2,
showarrow=False,
          text ="CVSS scores reflect severity of vulnerability. Over 75 percent of scores fall in FIRST Medium (4.0-6.9) vulnerable category<br>",
valign='top',
          align='left'
        ),
    ],
paper_bgcolor='rgba(0,0,0,0)',
plot_bgcolor='rgba(0,0,0,0)',
bargap=0
)

fig.show()
```

```
single =
pd.get_dummies(df.access_authentication.dropna()).groupby(df.pub_date.dt.to_period('Y')).me
an().SINGLE
none =
pd.get_dummies(df.access_authentication.dropna()).groupby(df.pub_date.dt.to_period('Y')).me
an().NONE

traces = [
    [single, none],
    [low, med, high],
    [net, loc, adj],
    [part, no_ia, comp],
    [part_ic, no_ic, comp_ic],
    [part_ii, no_ii, comp_ii],
]

texts = [
    ['Single', 'None'],
    ['Low', 'Medium', 'High'],
    ['Network', 'Local', 'Adj. Network'],
    ['Partial', 'None', 'Complete'],
    ['Partial', 'None', 'Complete'],
    ['Partial', 'None', 'Complete'],
]

colors = ['#2aa198', '#268bd2', '#dc322f']

fig = make_subplots (
    rows=3,
```

```
    cols=2,
subplot titles= [
        'Authentication required',
        'Impact on availability',
        'Access required',
        'Impact on confidentiality',
        'Attack vector',
        'Impact on integrity'
])

for i, (t, txt) in enumerate (zip (traces, texts)):
    for ndx, trace in enumerate(t):
fig.add_trace(
go.Scatter(
                x = [trace.index.to_timestamp()[-1]],
                y = [trace [-1]],
                name = "",
                text = " {} % {}”. format(np.round(trace[-1] * 100, 1), txt[ndx]),
textposition='middle right',
                mode='text',
cliponaxis=False
            ),
            row=i%3 + 1,
            col=i//3 + 1,
        )

fig.add_trace(
go.Scatter(
```

```
            x = trace.index.to_timestamp(),
            y = trace,
            name = "",
            line=dict(color=colors[ndx])
        ),
        row=i%3 + 1,
        col=i//3 + 1
    )

fig.add_trace(
go.Scatter(
            x = [trace.index.to_timestamp()[0]],
            y = [trace [0]],
            name = "",
            text = "{} %“. format(np.round(trace[0] * 100), 1),
textposition='middle left',
            mode='text',
cliponaxis=False
        ),
        row=i%3 + 1,
        col=i//3 + 1
    )

fig.update_layout(
showlegend=False,
   height=1200,
   title=dict(
      text="Access and impact",
```

```
    ),
    font=dict(
        family="Times New Roman",
        size=12,
color="#000607"
    ),
xaxis=dict(
showgrid=False,
    ),
yaxis=dict(
showgrid=False,
showticklabels=False
    ),
    yaxis2=dict(
showgrid=False,
showticklabels=False
    ),
    yaxis3=dict(
showgrid=False,
showticklabels=False
    ),
    yaxis4=dict(
showgrid=False,
showticklabels=False
    ),
    yaxis5=dict(
showgrid=False,
showticklabels=False
```

```
    ),
    yaxis6=dict(
showgrid=False,
showticklabels=False
    ),
paper_bgcolor='rgba(0,0,0,0)',
plot_bgcolor='rgba(0,0,0,0)'
)

fig.show()

from sklearn import preprocessing
# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'access_authentication'.
df['access_authentication’] =label_encoder.fit_transform(df['access_authentication'])
df['access_authentication']. unique ()
df['impact_availability’] =label_encoder.fit_transform(df['impact_availability'])
df['impact_availability']. unique ()
df['impact_confidentiality’] =label_encoder.fit_transform(df['impact_confidentiality'])
df['impact_integrity’] =label_encoder.fit_transform(df['impact_integrity'])
df['cve_id’] =label_encoder.fit_transform(df['cve_id'])
df["cwe_name"] = label_encoder.fit_transform(df["cwe_name"])

import matplotlib.pyplot as plt
%Matplotlib inline
from sklearn.decomposition import PCA
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import StandardScaler, normalize

scaler = StandardScaler()
scaled = scaler.fit_transform(df)
normalizedfull = normalize(df)
nfull = pd.DataFrame(normalizedfull)

from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

#ELBOW Method
Sum_of_squared_distances = []
K = range (1,10)
for num_clusters in K:
kmeans = KMeans(n_clusters=num_clusters)
kmeans.fit(nfull)
Sum_of_squared_distances.append(kmeans.inertia_)
plt.plot(K,Sum_of_squared_distances,'bx-')
plt.xlabel('Values of K')
plt.ylabel('Sum of squared distances/Inertia')
plt.title('Elbow Method For Optimal k')
plt.show()

from sklearn.decomposition import PCA
pca = PCA(n_components=2)
x = pca.fit_transform(nfull)
print ("original shape:   ", df.shape)
```

```
print ("transformed shape:", x.shape)


plt.figure(figsize =(10,10))
plt.scatter(x[:,0],x[:,1])
plt.xlabel('pca1')
plt.ylabel('pca2')


SSE = []
for cluster in range (1,20):
kmeans = KMeans(n_clusters = cluster, init='k-means++')
kmeans.fit(data_scaled)
SSE.append(kmeans.inertia_)

# Converting the results into a dataframe and plotting them
frame = pd.DataFrame({'Cluster':range(1,20), 'SSE':SSE})
plt.figure(figsize=(12,6))
plt.plot(frame['Cluster'], frame['SSE'], marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')


K = 500
M = 100
L = 10
thresh = .01


cwes = pd.get_dummies(cve.cwe_name)
top_K = products.vulnerable_product.value_counts()[:K]
counts = top_K.values
```

```
X = products [products.vulnerable_product.isin(
top_K.index
    )]. join (
cwes
). groupby(
        'vulnerable_product'
). sum ()

Z = KernelPCA(n_components=2, kernel="cosine").fit_transform(X)

H = 50
top_H = top_K = products.vulnerable_product.value_counts()[:H]
fig = ff.create_dendrogram(
    X[X.index.isin(top_H.index)].sort_index(),
    labels=np.vectorize(lambda x: " ".join(map(lambda x: x.title() if len(x) > 2 else x.upper(),
x.split("_"))))(top_H.sort_index().index),
    orientation='left')

labels = kmeans.labels_
silhouette_score(final_data, labels, metric='euclidean')
#Calculating the silhouette score:
print (f'Silhouette Score(n=2): {silhouette_score(final_data, label)}')

from sklearn.metrics import silhouette_score
kMedoids = KMedoids(n_clusters = 2, random_state = 0)
kMedoids.fit(data)
y_kmed = kMedoids.fit_predict(data)
```

```
silhouette_score = silhouette_score(data, y_kmed)
print(silhouette_score)

results = {}
for i in range (2,10):
kmeans = KMeans(n_clusters=i, random_state=30)
    labels = kmeans.fit_predict(final_data)
    db_index1 = davies_bouldin_score(final_data, labels)
results.update({i: db_index1})

plt.plot(list(results.keys()), list(results.values()))
plt.xlabel("Number of clusters")
plt.ylabel("Davies-Boulding Index")
plt.show()

models = [('KMeans', kmeans_s),#, kmeans_d),
          ('KMedoids', kmedoids_s),#, kmedoid_d),
          ('AgglomerativeClustering', agglo_s)#, Agglo_d)
```

Screenshots

```
df = pd.read_csv(r'C:\Users\Admin\Documents\VulniD\cve.csv')
df.head(2)
```

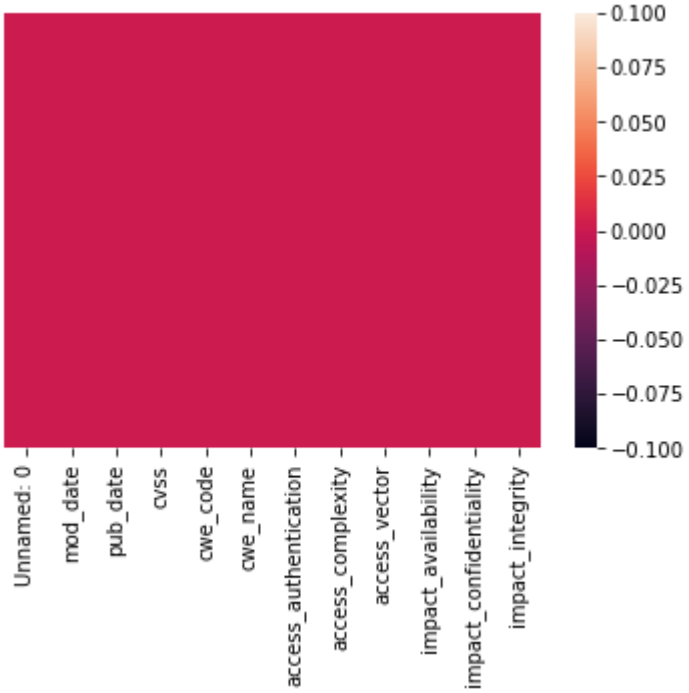
	Unnamed: 0	mod_date	pub_date	cvss	cwe_code	cwe_name	summary	access_authentication	access_complexity	access_vector	impact_availability
0	CVE-2019-16548	21-11-2019 15:15	21-11-2019 15:15	6.8	352	Cross-Site Request Forgery (CSRF)	A cross-site request forgery vulnerability in ...	NaN	NaN	NaN	NaN
1	CVE-2019-16547	21-11-2019 15:15	21-11-2019 15:15	4.0	732	Incorrect Permission Assignment for Critical ...	Missing permission checks in various API endpo...	NaN	NaN	NaN	NaN

Loading Dataset

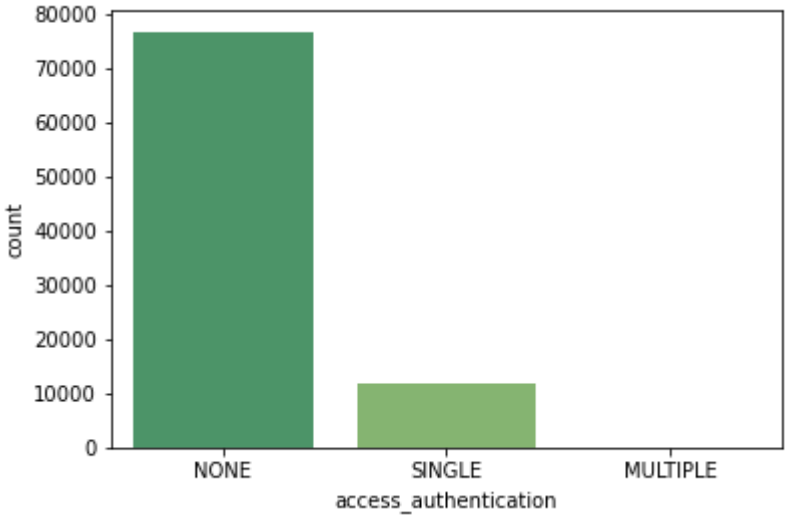
```
#basic information about data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 89660 entries, 0 to 89659
Data columns (total 13 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Unnamed: 0              89660 non-null  object
 1   mod_date                89660 non-null  object
 2   pub_date                89660 non-null  object
 3   cvss                    89660 non-null  float64
 4   cwe_code                89660 non-null  int64
 5   cwe_name                89660 non-null  object
 6   summary                 89660 non-null  object
 7   access_authentication   88776 non-null  object
 8   access_complexity       88776 non-null  object
 9   access_vector           88776 non-null  object
 10  impact_availability     88776 non-null  object
 11  impact_confidentiality  88776 non-null  object
 12  impact_integrity        88776 non-null  object
dtypes: float64(1), int64(1), object(11)
memory usage: 8.9+ MB
```

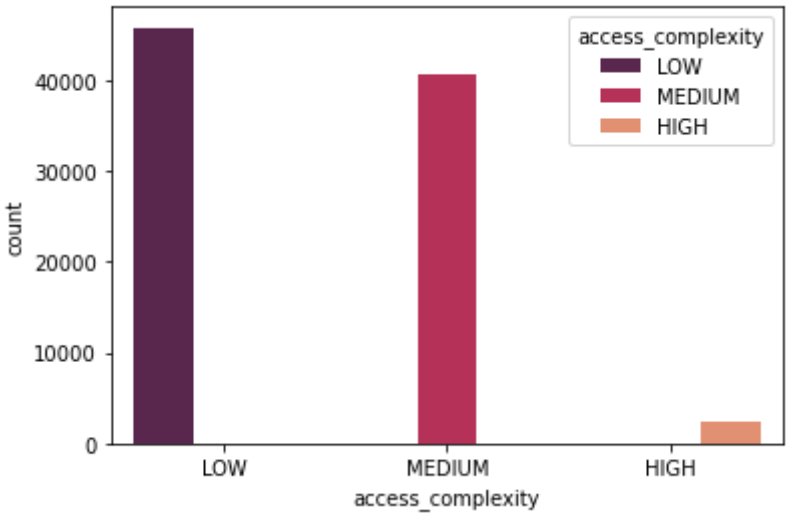
Attributes Information



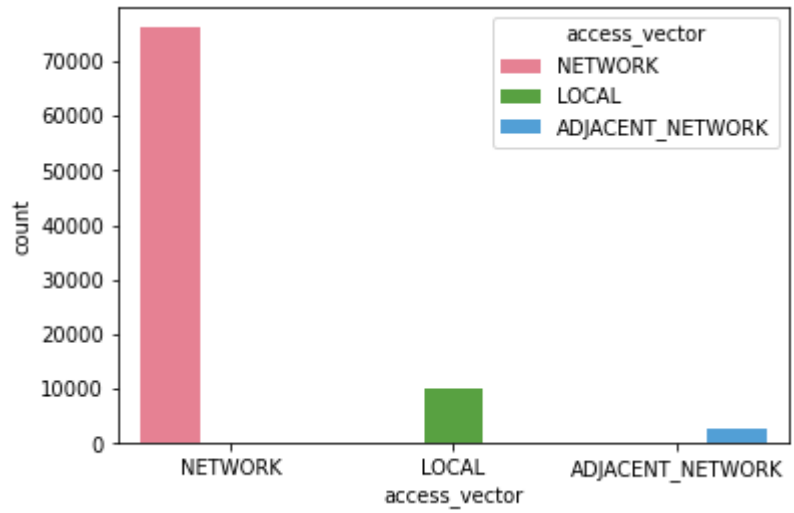
Heatmap



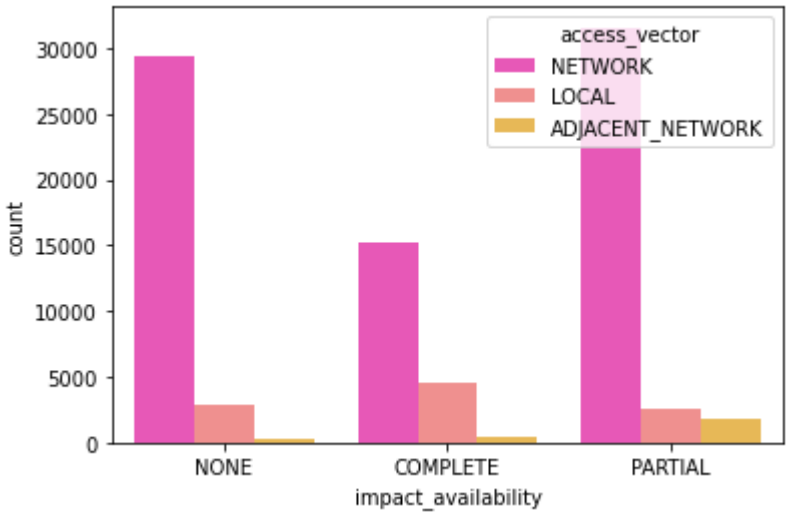
Count Plot for Access_Authentication



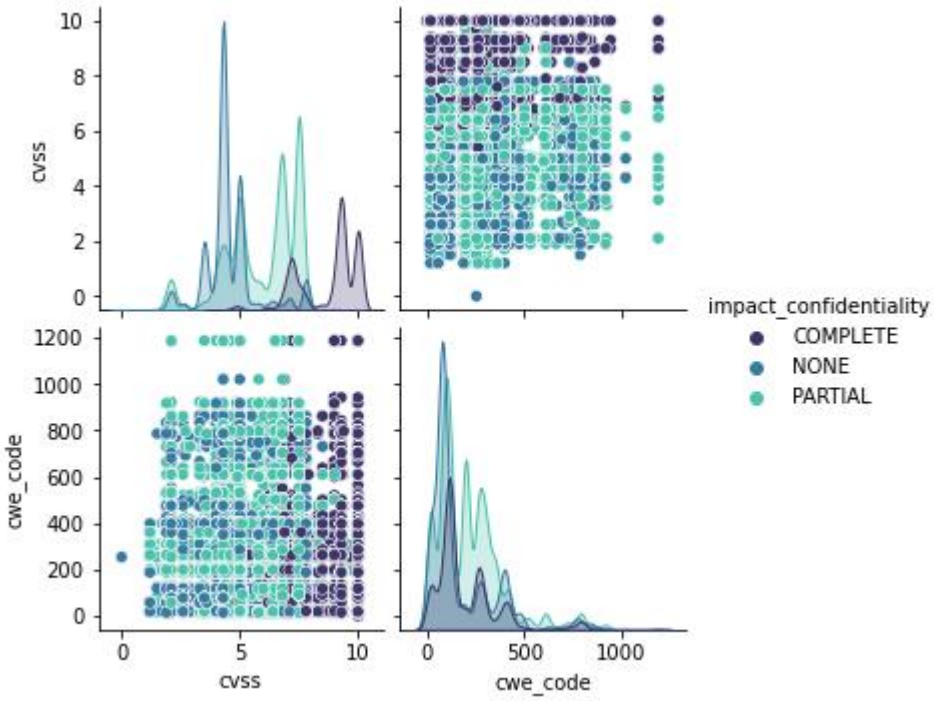
Count Plot for Access_Complexity



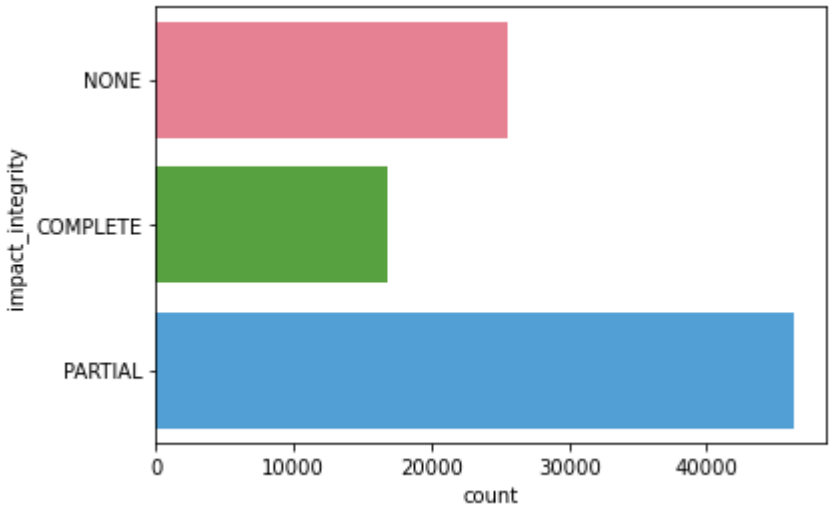
Count Plot for Access Vector



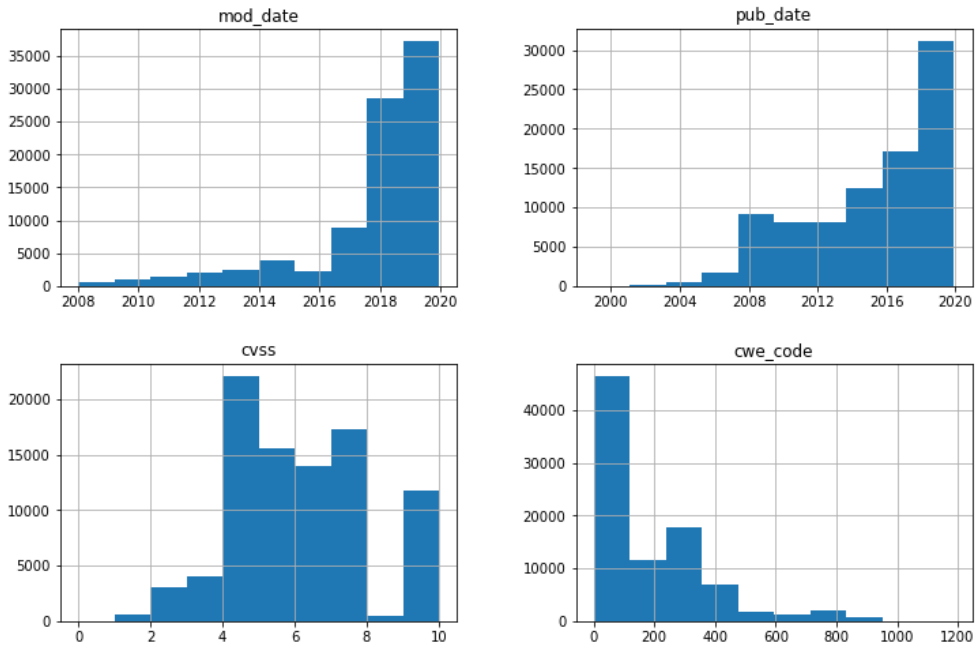
Count Plot for Impact_Availability



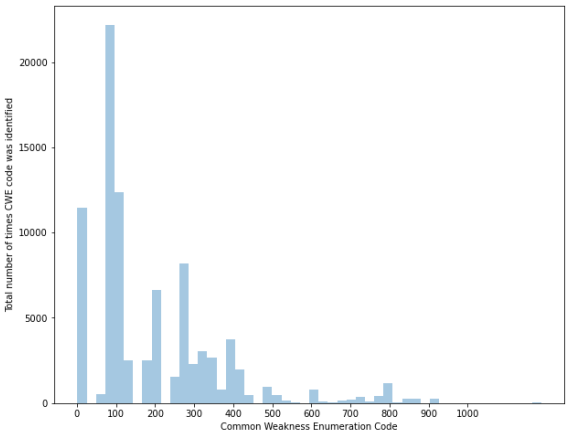
Count Plot for Impact Confidentiality



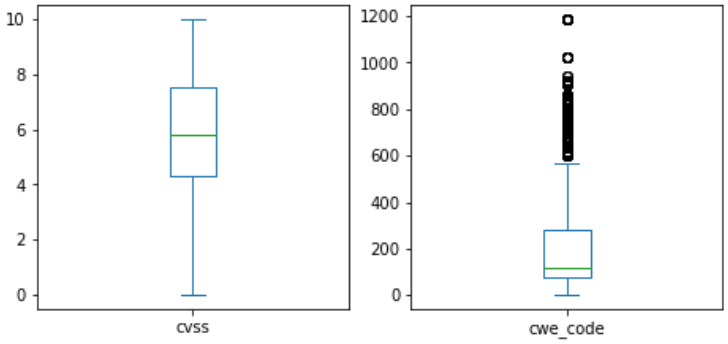
Count Plot for Impact Integrity



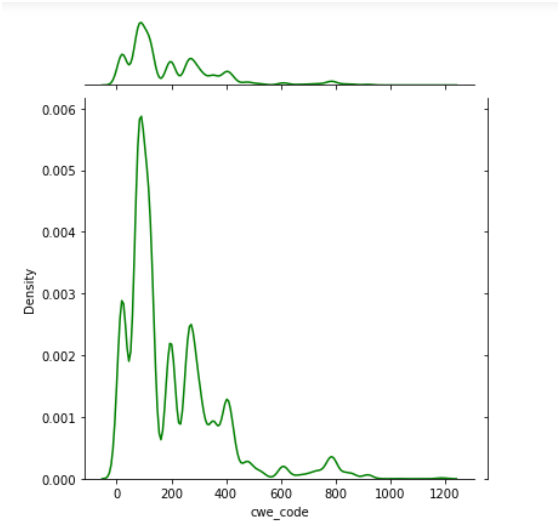
Histogram



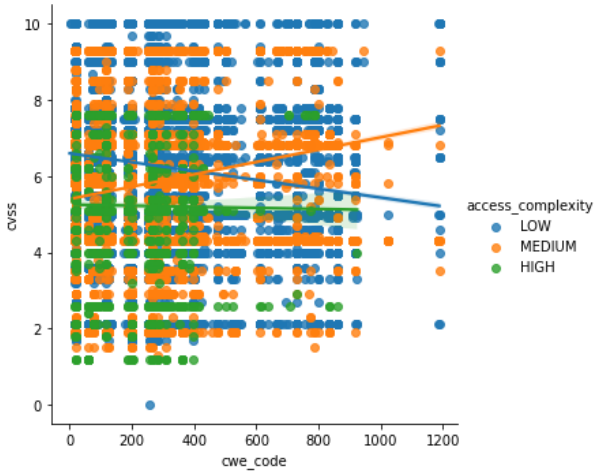
Common Weakness Enumeration Code



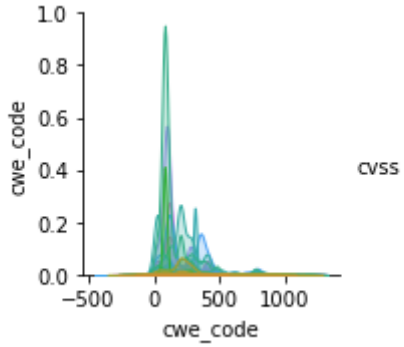
Box Plot



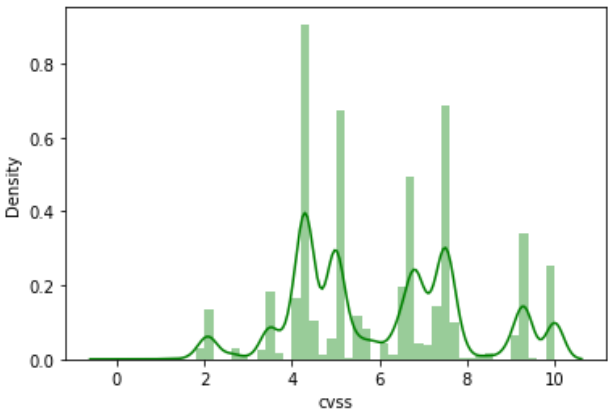
Joint Plot



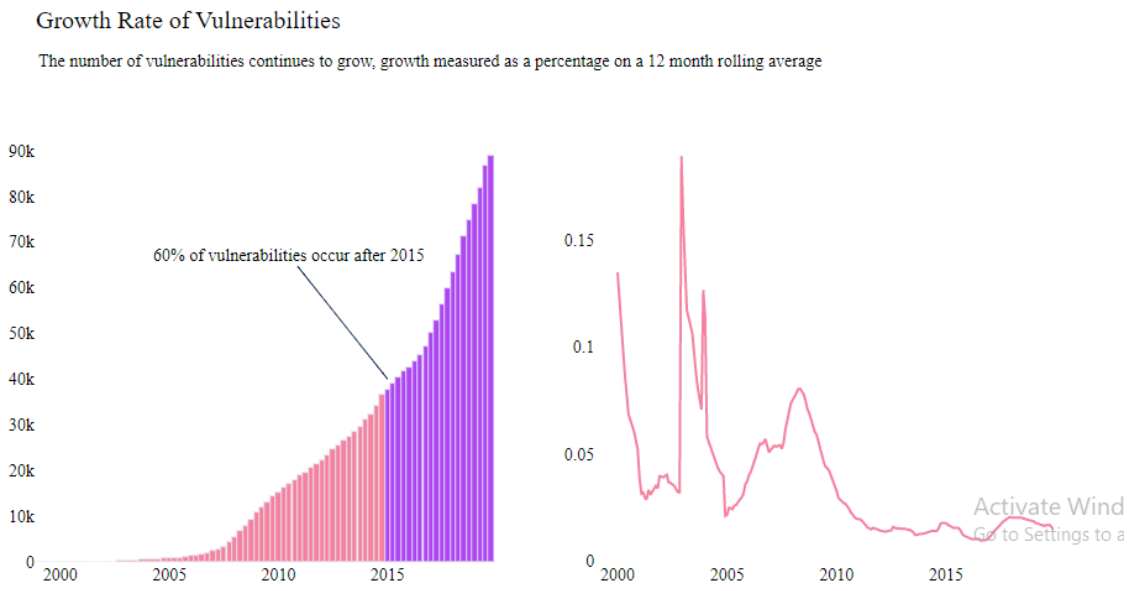
Count plot of cwe_code and cvss



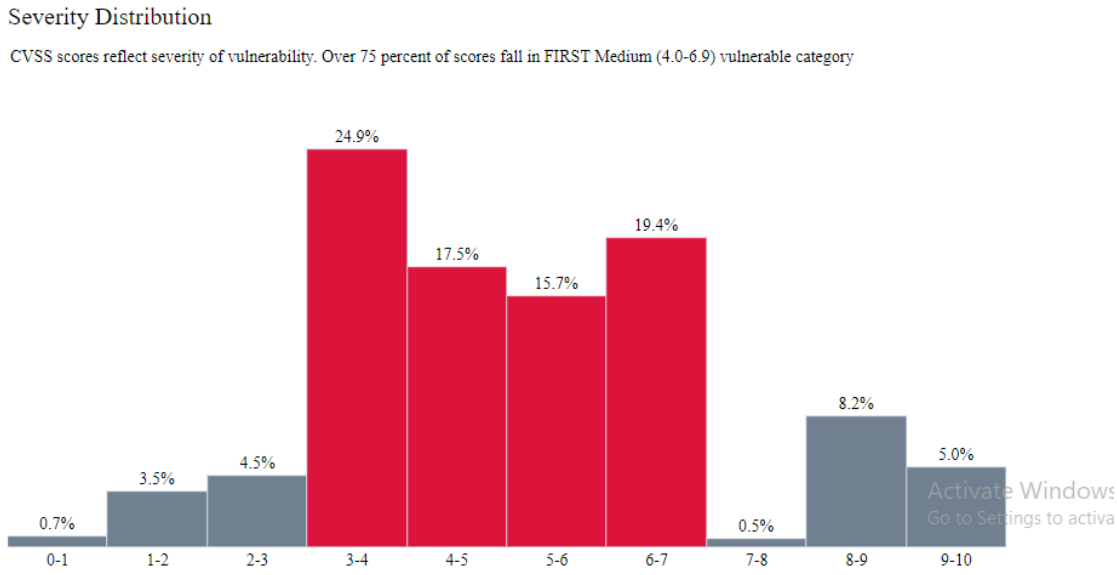
Pair Plot of cvss



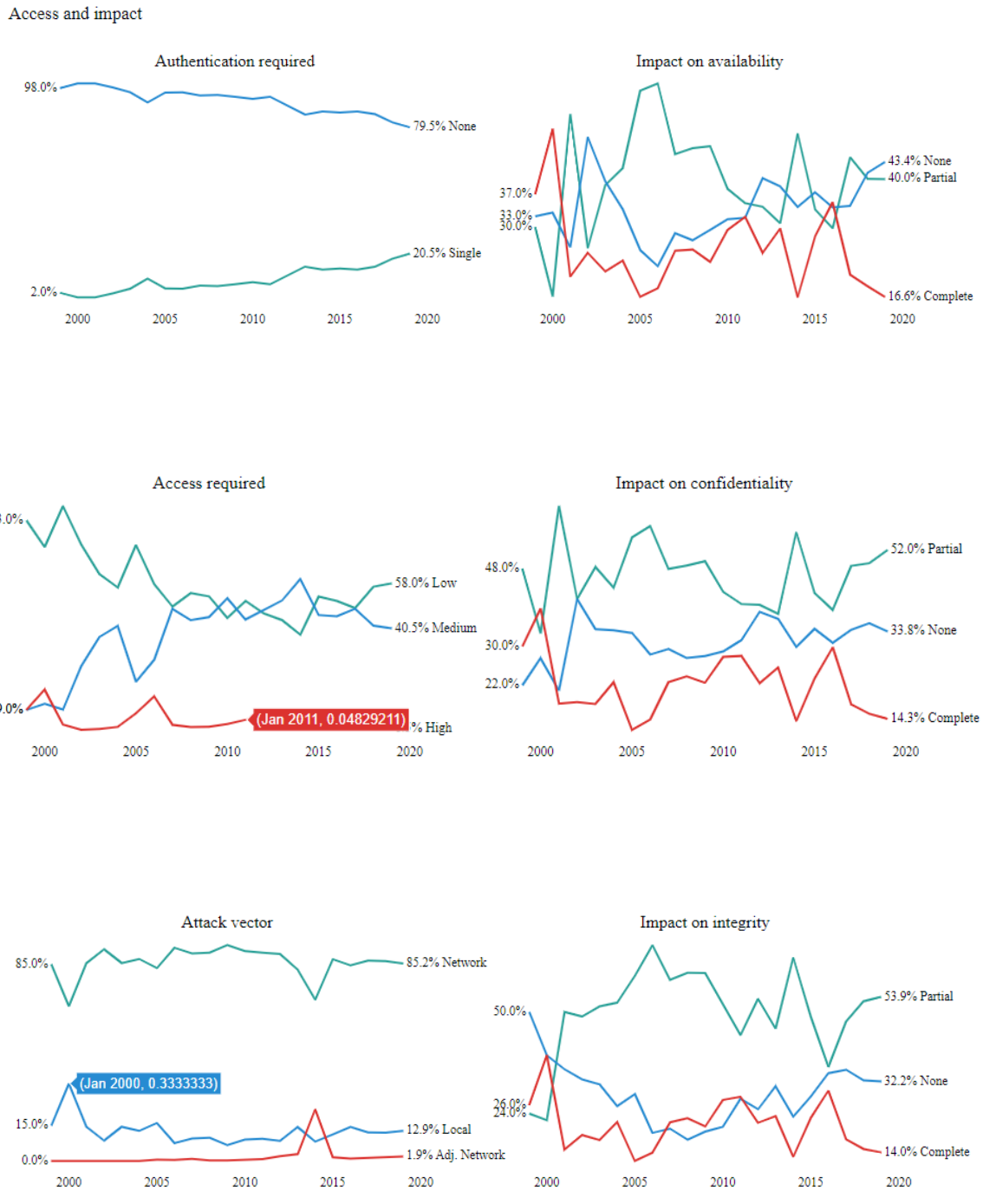
Dist Plot of cvss



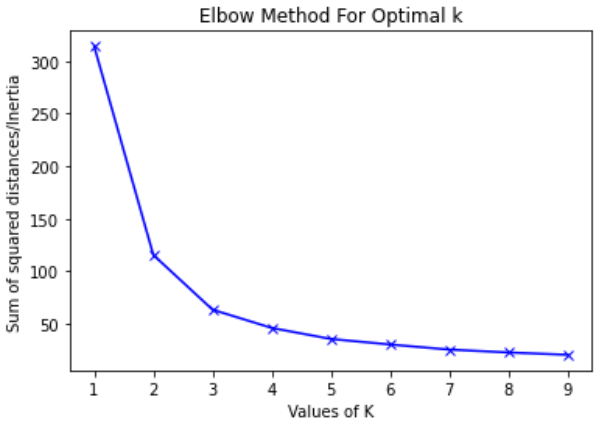
Growth Rate of Vulnerabilities



Severity Distribution of Vulnerabilities



Value Distribution

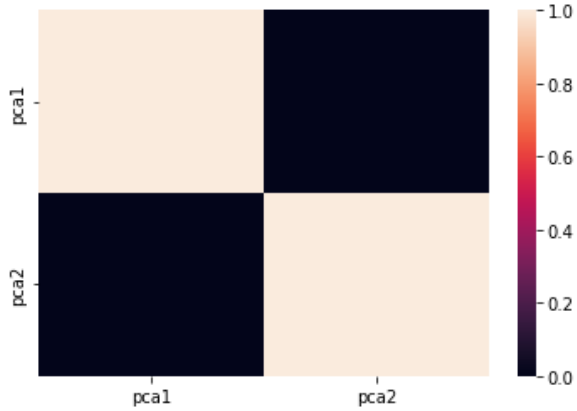


Elbow Method

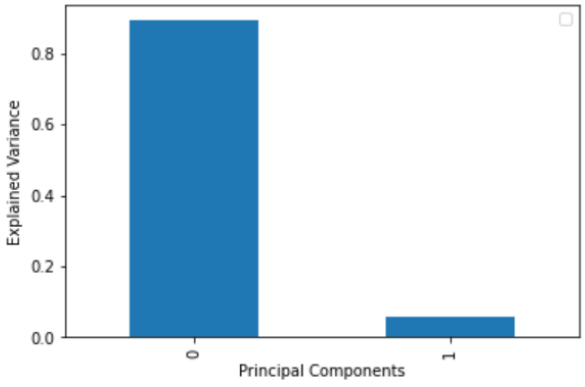
	pca1	pca2
0	-0.011420	-0.000821
1	-0.010211	-0.000889
2	-0.007931	-0.001183
3	-0.009720	-0.001511
4	-0.011535	-0.000837
...	...	...
88771	0.350962	-0.094255
88772	0.482640	-0.188127
88773	0.537764	0.133628
88774	0.592489	0.031163
88775	0.972410	-0.008092

88776 rows × 2 columns

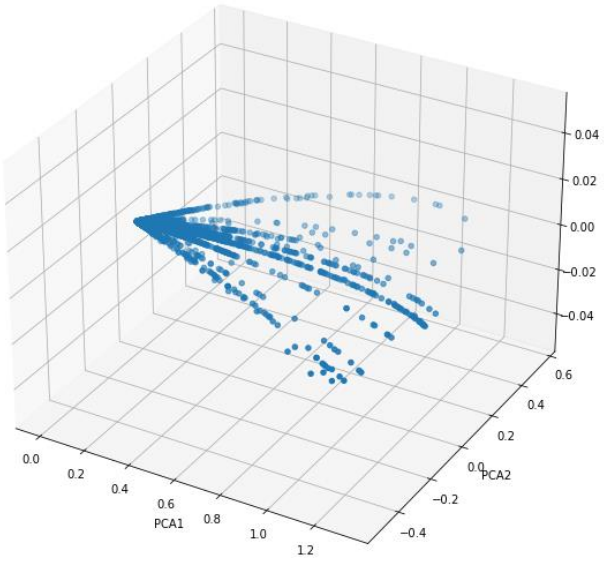
Principal Components



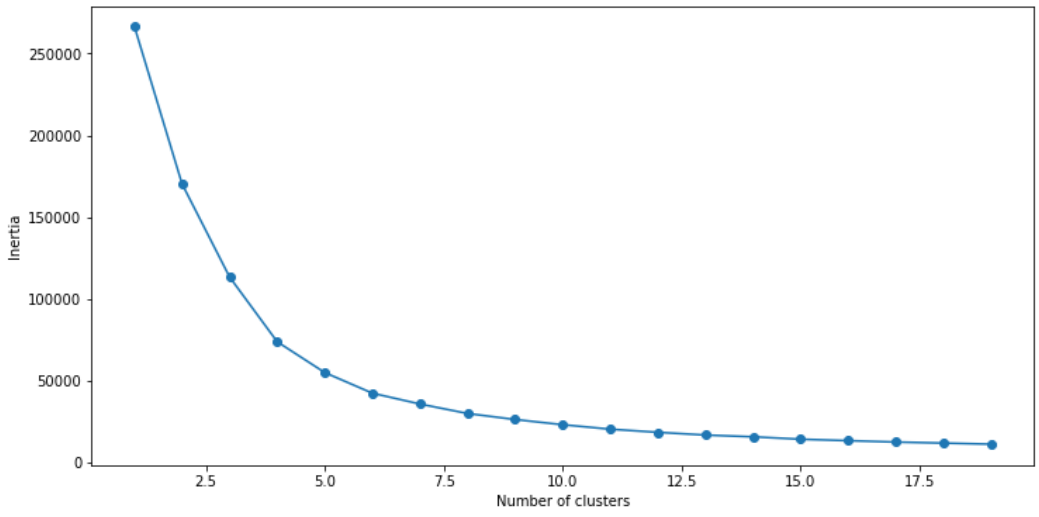
Heatmap after applying PCA

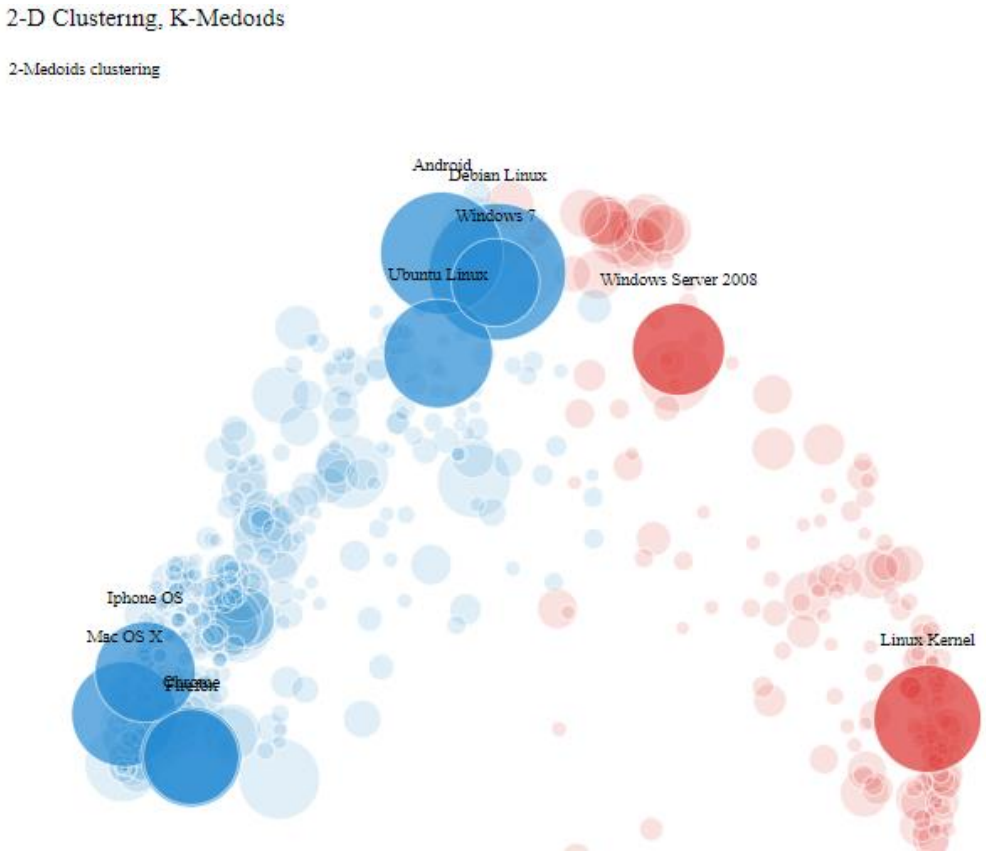


Variance of principal Components

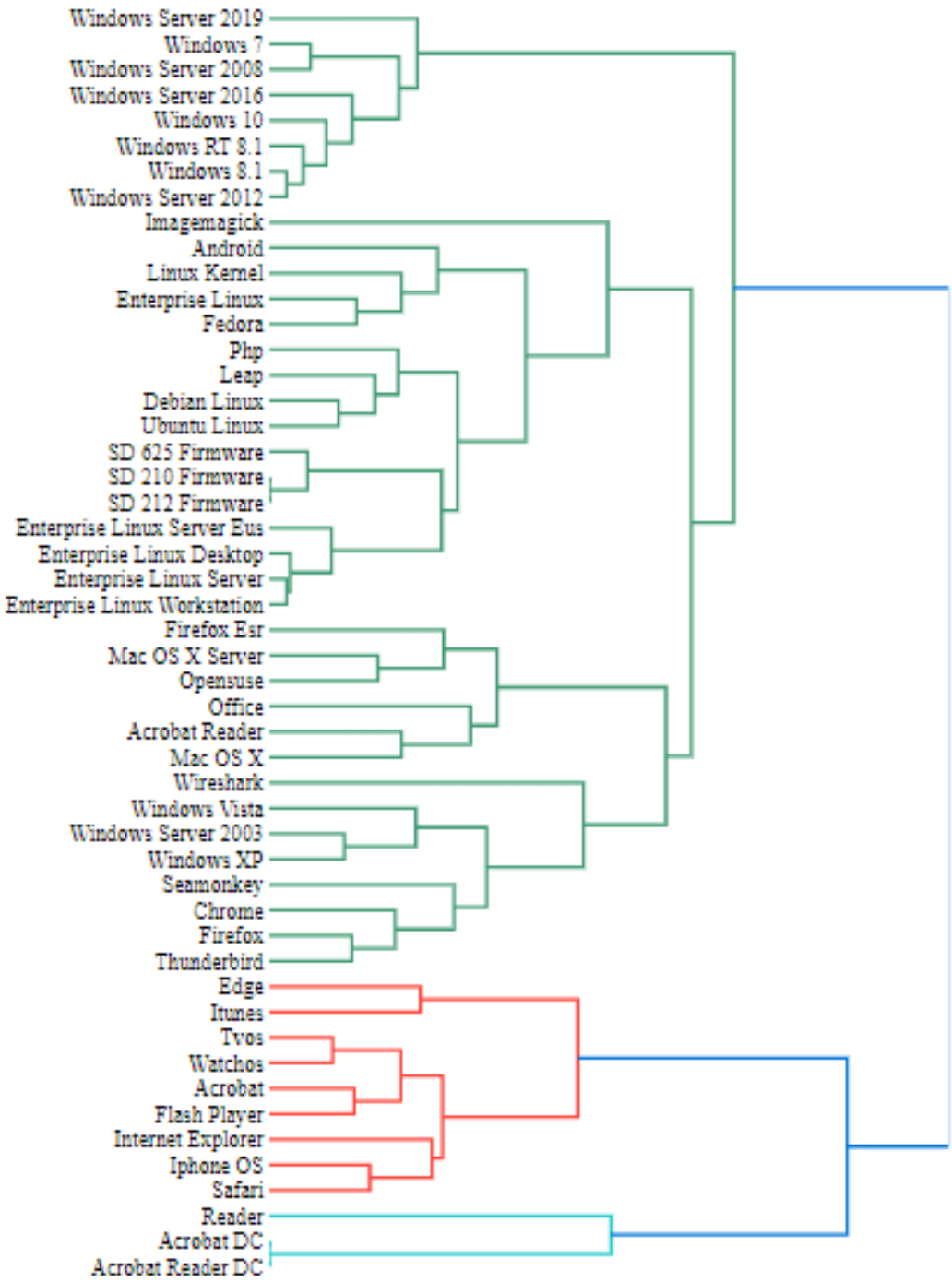



```
Text(0, 0.5, 'Inertia')
```

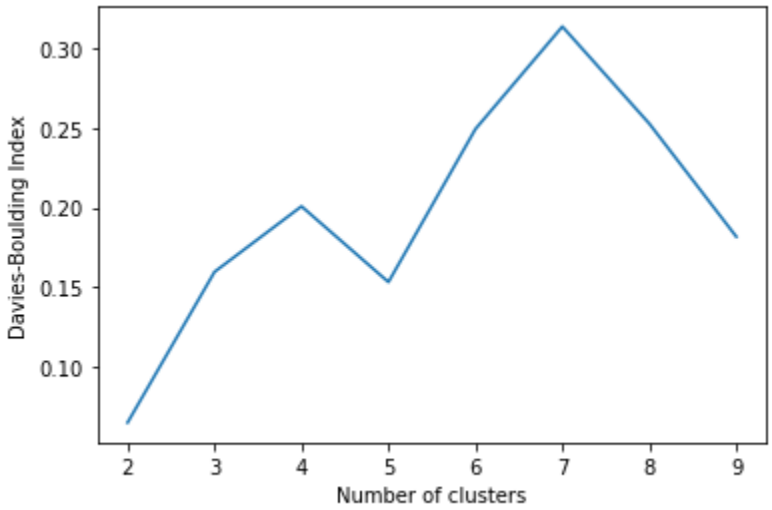


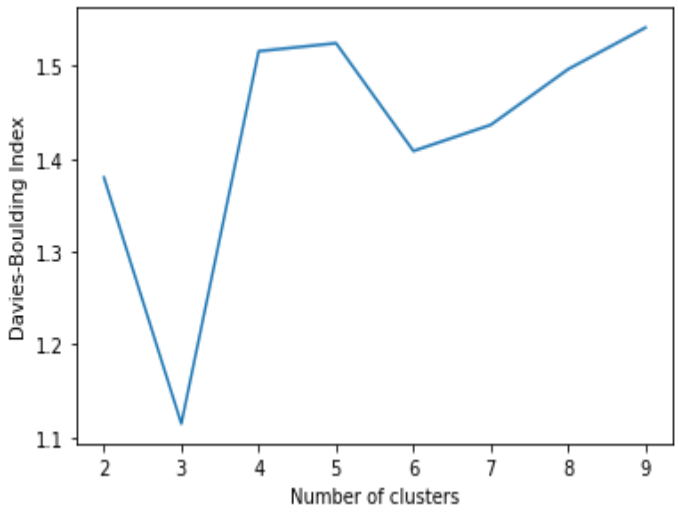


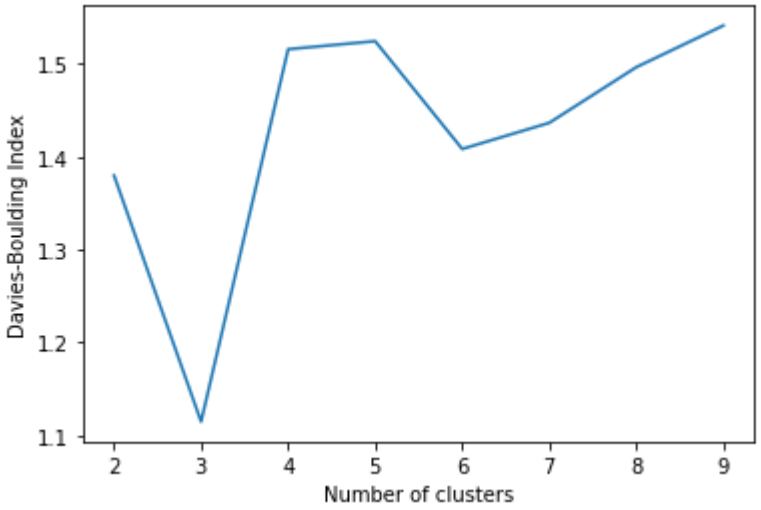
K-Medoid Clustering



Hierarchical Agglomerative Clustering


Davies-Boulding Index
0.30
0.25
0.20
0.15
0.10
2
3
4
5
6
7
8
9
Number of clusters


Davies-Boulding Index
1.5
1.4
1.3
1.2
1.1
2
3
4
5
6
7
8
9
Number of clusters

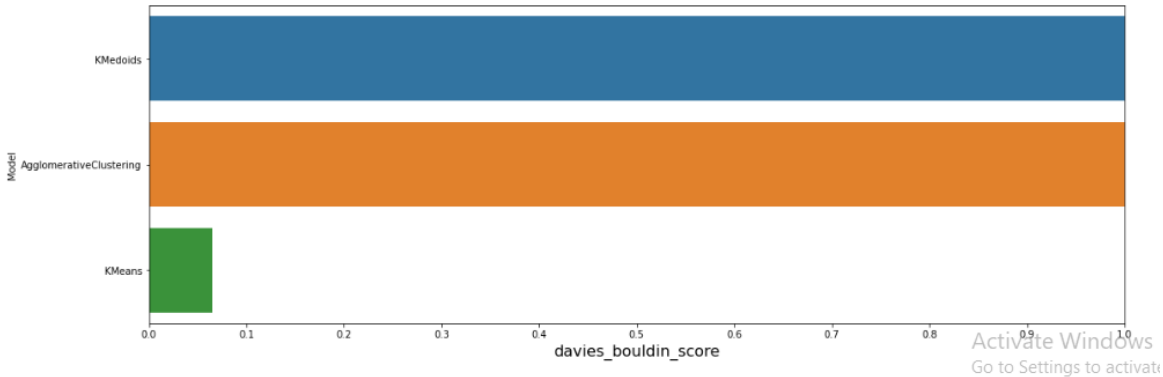

Davies-Boulding Index
1.5
1.4
1.3
1.2
1.1
2
3
4
5
6
7
8
9
Number of clusters

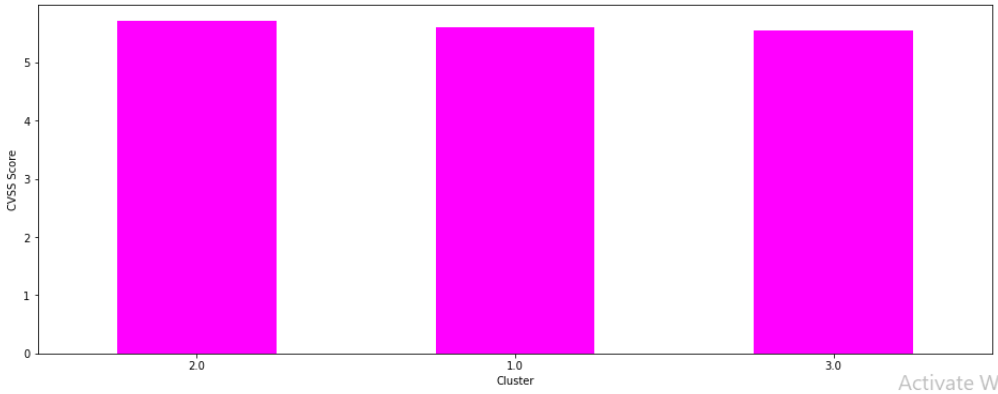
	Model	silhouette_score
0	KMeans	0.980797
1	KMedoids	0.545023
2	AgglomerativeClustering	0.303567

Results obtained through Silhouette Scores

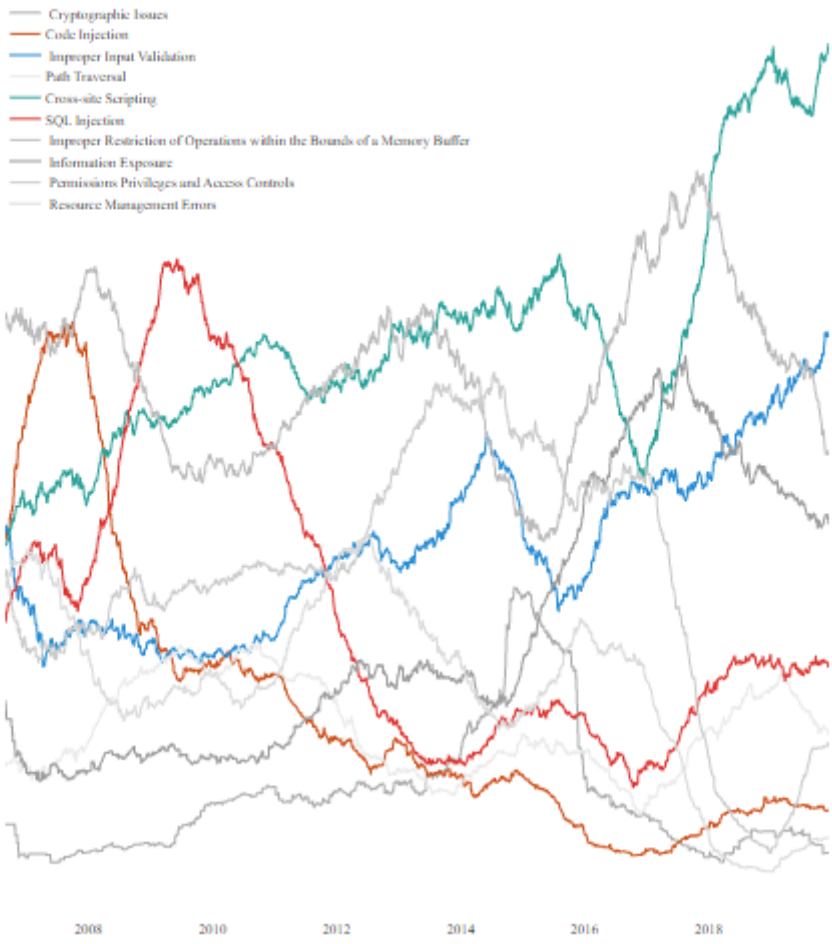
	Model	davies_bouldin_score
0	KMeans	0.064826
1	KMedoids	1.552862
2	AgglomerativeClustering	1.379946

Results Obtained through Davies Bouldin Index

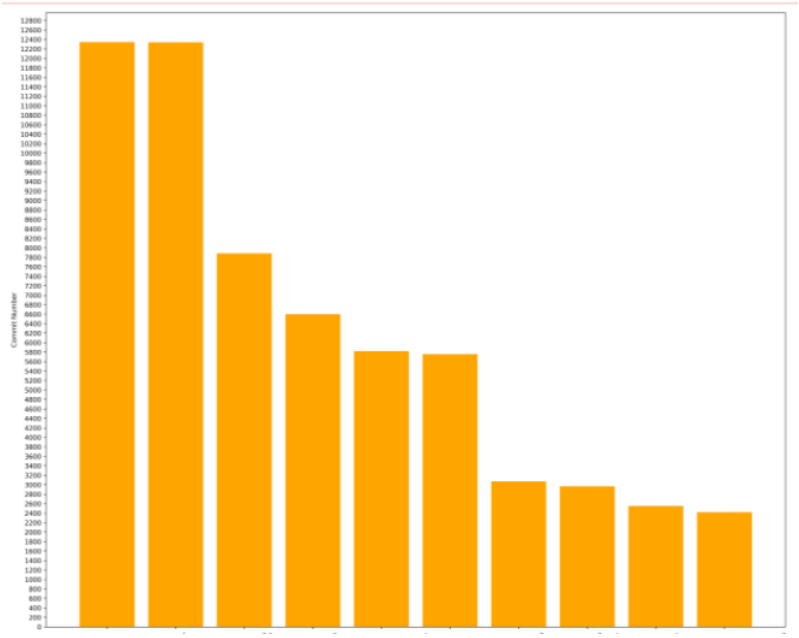




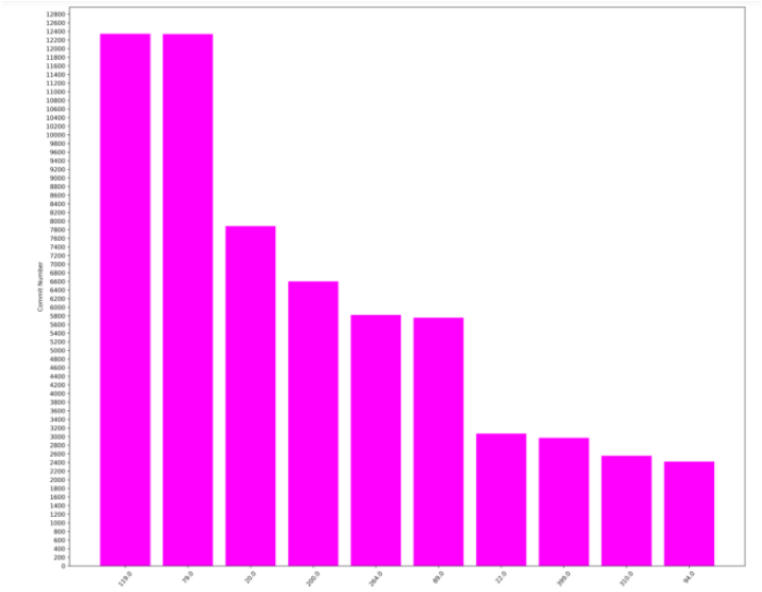
Clusters



Vulnerabilities Over Time



Top Ten Vulnerabilities



Top CWE Codes

Exploited Products