

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

# COMPUTER NETWORKS – CY245AT

### REPORT

**Submitted by**

**Meryn Babu 1RV22CY040**

**Swar Lodaya 1RV22CY056**

**Under the Guidance of:**

**Dr. Sandhya S**

**Assistant Professor**

**Department of CSE**

**RV College of Engineering®**

**Bengaluru - 560059**

### Computer Science and Engineering – Cybersecurity 2024-2025

**DECLARATION**

We, **Meryn Babu and Swar Lodaya**, students of fourth semester **BE** in **Computer Science and Engineering – Cyber Security**, **Department of Computer Science and Engineering,** RV College of Engineering®, Bengaluru, declare that the Computer Networks Experiential Learning with title **“Leveraging XGBoost Machine Learning Model for CVE Exploitation Classification”,** has been carried out by us. It has been submitted in partial fulfillment for the award of degree in **BE** in **Computer Science and Engineering-Cyber Security** of RV College of Engineering®, Bengaluru, affiliated to Visvesvaraya Technological University, Belagavi, during the academic year **2024-25**. The matter embodied in this report has not been submitted to any other university or institution for the award of any other degree or diploma.

**Date of Submission: Signature of the Student**

**Student Name: Meryn Babu, Swar Lodaya**

USN: 1RV22CY040, 1RV22CY056

Department of Computer Science and Engineering RV College of Engineering®,

Bengaluru-560059

**TABLE OF CONTENTS**

* 1. Acknowledgement
  2. Abstract
  3. Chapter 1: Introduction
     + Introduction to Software Exploitability
     + Organizational Report
  4. Chapter 2: Theory and Concepts of CVE Exploitability Prediction
     + Introduction
     + CVE (Common Vulnerabilities and Exposures) System
     + Vulnerability Scoring (CWE and CVSS Scoring System)
     + Machine Learning in Cybersecurity
     + XGBoost ML Algorithm
     + Requirements of CVE Exploitability Prediction
     + Summary
  5. Chapter 3: High-Level Design of CVE Exploitability Prediction System
     + Design Considerations
     + Architectural Strategies
     + System Architecture for Risk Prioritization Solution
     + Data Flow Diagrams
     + Summary
  6. Chapter 4: Implementation of CVE Exploitability Prediction System
     + Introduction
     + Algorithm Selection
     + Key Features of XGBoost Algorithm
     + Pseudocode for XGBoost Implementation
     + Source Code Implementation
     + Summary
  7. Chapter 5: Result and Analysis
     + Overview of Experiments Conducted
     + Performance Analysis
     + Observations
     + Summary
  8. Chapter 6: Conclusion
  9. References and Citation

**ACKNOWLEDGEMENT**

We are indebted to **Rashtreeya Sikshana Samithi Trust**, Bengaluru for providing us with all the facilities needed for the successful completion of    Experiential Learning in Computer Networks Course at **Rashtreeya Vidyalaya College of Engineering (RVCE)** during the tenure of our Course.

We would like to thank **Dr. K N Subramanya, Principal,** for giving us an opportunity to be a part of RVCE and for his timely help and encouragement during the tenure of EL for Computer Network Course.

We are greatly thankful to **Dr. Ramakanth Kumar P., Professor and Head, Dept. of CSE** for his motivation and constant support during our tenure of EL for Computer Network Course.

We are greatly thankful to **Dr.Minal Moharir, Professor and Coordinator, BE CSE-Cyber Security** for her constant support during our tenure of EL for Computer Network Course..

We take this opportunity to convey my sincere gratitude to my internal guide **Dr. Sandhya S, Assistant Professor**, Dept of CSE, her advice, support and valuable suggestions that helped us to accomplish the EL for Computer Network Course Project in time.

We extend our thanks to all who have directly or indirectly extended their constant support for successful completion of our EL for Computer Network Course.Project.

Meryn Babu, Swar Lodaya

1RV22CY040, 1RV22CY056

**ABSTRACT**

Exploitation of software vulnerabilities poses a significant threat to cybersecurity, making the accurate prediction of vulnerability exploitability crucial for effective risk management. The Common Vulnerabilities and Exposures (CVE) system provides a standardized way to identify and track these vulnerabilities. However, not all vulnerabilities are exploited, and identifying which ones are most likely to be targeted is essential for prioritizing security efforts. This study focuses on leveraging the XGBoost machine learning model to classify CVEs based on their likelihood of being exploited, providing valuable insights into the evolving threat landscape.

The methodology involved collecting a comprehensive dataset from the National Vulnerability Database (NVD), which includes various features such as CVSS scores, CWE codes, and attack vectors. The data was preprocessed to remove noise and ensure consistency before being analyzed to identify patterns and correlations. The XGBoost model was then trained using these features, with hyperparameter tuning performed to optimize the model's performance. The model's accuracy, precision, recall, and F1-score were evaluated to assess its effectiveness in predicting exploitability. Additionally, feature importance analysis was conducted to determine which factors were most influential in the model's predictions.

The results demonstrated that the XGBoost model achieved high performance, with an accuracy of 98.5%, precision of 90.3%, recall of 99.0%, and an F1-score of 94.4%, indicating its robustness in identifying exploitable vulnerabilities. Data analysis revealed that network-based vulnerabilities were predominant, particularly those related to nearby network and local vectors, emphasizing the changing threat environment. Vendor and operating system evaluations identified Linux and Microsoft as the most impacted, highlighting the need for ongoing security improvements. This study underscores the importance of advanced machine learning techniques in vulnerability management, offering actionable insights to enhance cybersecurity practices and protect against potential exploits.

**CHAPTER 1: INTRODUCTION**

* 1. **Introduction of Software Exploitability**

Exploitation of software vulnerabilities has become a critical concern in the realm of cybersecurity. With the increasing complexity and interconnectedness of modern systems, the ability to predict and mitigate potential vulnerabilities before they are exploited is more important than ever. The Common Vulnerabilities and Exposures (CVE) system provides a framework for identifying and cataloging vulnerabilities across different software platforms. However, not all vulnerabilities are equally likely to be exploited, and the challenge lies in identifying which ones pose the greatest risk. This research focuses on using machine learning techniques, specifically the XGBoost algorithm, to classify and predict the exploitability of CVEs. By accurately predicting which vulnerabilities are most likely to be targeted, organizations can better prioritize their security efforts and allocate resources more effectively.

**1.2 Organization of Report**

This report is organized into several chapters, each addressing different aspects of the research:

Chapter 1: Introduction

This chapter introduces the topic, discusses the significance of predicting CVE exploitability, and outlines the objectives of the research. It also provides an overview of the organization of the report.

Chapter 2: Literature Review

This chapter reviews existing literature on CVE exploitability, machine learning models in cybersecurity, and the XGBoost algorithm. It provides the background necessary to understand the context and relevance of the research.

Chapter 3: Methodology

This chapter details the methodology used in the research, including data collection, preprocessing, feature selection, model training, and evaluation. It also discusses the hyperparameter tuning process and the rationale behind choosing XGBoost for this study.

Chapter 4: Data Analysis and Results

This chapter presents the analysis of the data, the performance metrics of the XGBoost model, and the results of the feature importance analysis. It includes visualizations and tables that summarize the key findings.

Chapter 5: Discussion

This chapter interprets the results in the context of existing literature, discussing the implications for cybersecurity practices. It also addresses the limitations of the study and suggests areas for future research.

Chapter 6: Conclusion and Recommendations

The final chapter summarizes the key findings of the research, concludes the study, and offers recommendations for improving vulnerability management practices based on the insights gained from the model's predictions.

References:

This section lists all the academic papers, articles, and other sources referenced throughout the report.

Appendices:

The appendices provide additional information, such as detailed data sets, code snippets, or supplementary material that supports the research.

**CHAPTER 2: THEORY AND CONCEPTS OF CVE EXPLOITABILITY PREDICTION**

**2.1 Introduction**

The prediction of CVE exploitability is a crucial aspect of cybersecurity, focusing on identifying which vulnerabilities are most likely to be exploited by malicious actors. This chapter delves into the theoretical foundations and key concepts that underpin this research. It covers the basics of the Common Vulnerabilities and Exposures (CVE) system, the importance of vulnerability scoring and classification, and the role of machine learning in predicting exploitability. The chapter also outlines the requirements for implementing a predictive model, including data collection, preprocessing, and feature selection.

**2.2 Common Vulnerabilities and Exposures (CVE) System**

The CVE system is a standardized framework for identifying, cataloging, and sharing information about software vulnerabilities. Each CVE entry represents a specific vulnerability, providing a unique identifier, a brief description, and references to detailed technical information. The CVE system plays a vital role in vulnerability management, enabling organizations to track and address security flaws across their software environments.

**2.3 Vulnerability Scoring: CVSS and CWE**

The Common Vulnerability Scoring System (CVSS) is a widely used framework for assessing the severity of software vulnerabilities. It provides a quantitative measure of a vulnerability's impact, considering factors such as exploitability, impact on confidentiality, integrity, and availability, and environmental factors. The Common Weakness Enumeration (CWE) complements CVSS by categorizing vulnerabilities based on their underlying causes, such as coding errors or design flaws. Together, CVSS and CWE provide a comprehensive view of the risks associated with specific vulnerabilities.

**2.4 Machine Learning in Cybersecurity**

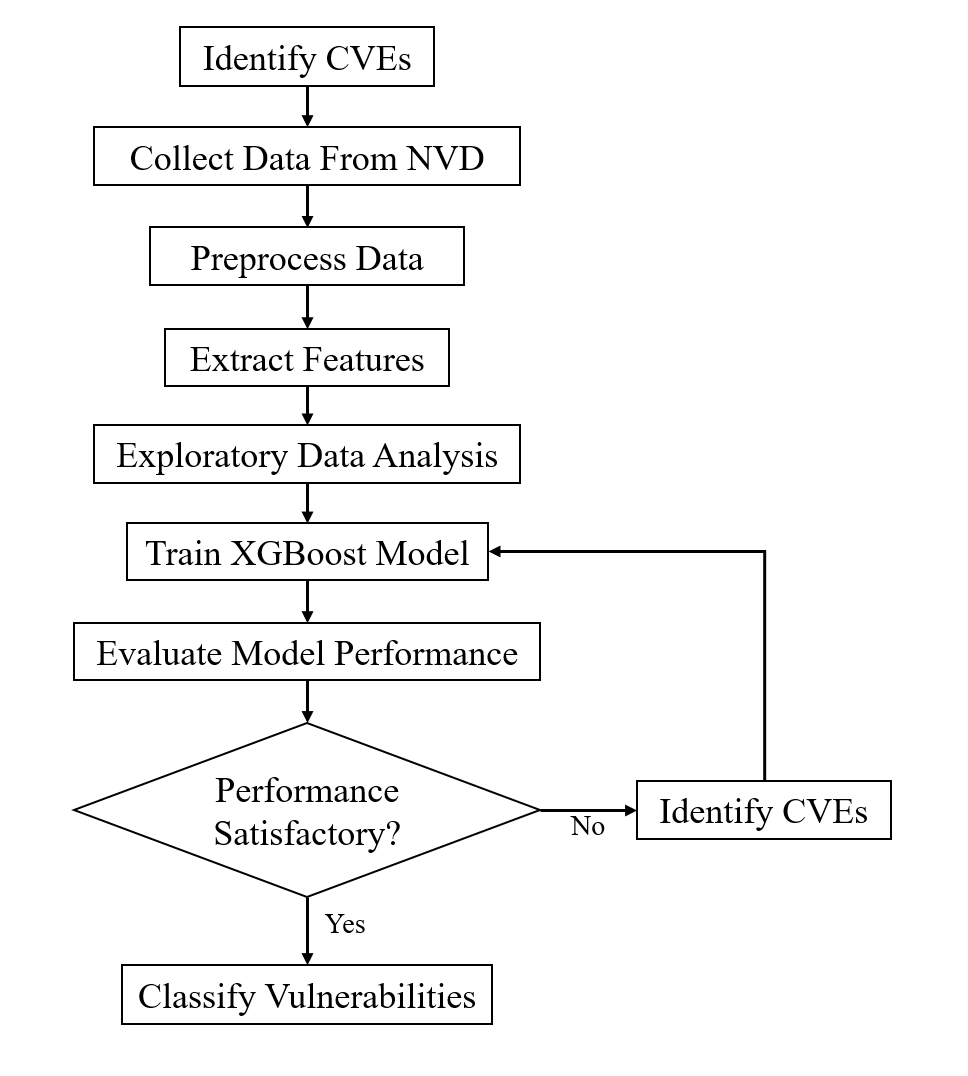
Machine learning has emerged as a powerful tool in cybersecurity, enabling the automation of threat detection, anomaly detection, and vulnerability assessment. In the context of CVE exploitability prediction, machine learning models can analyze patterns in historical data to predict which vulnerabilities are most likely to be exploited. The use of algorithms such as XGBoost allows for the efficient handling of large datasets, feature importance analysis, and accurate predictions.

**2.5 XGBoost Algorithm**

XGBoost, or Extreme Gradient Boosting, is a machine learning algorithm that builds an ensemble of decision trees in a sequential manner, optimizing a loss function to improve model accuracy. XGBoost is known for its speed, scalability, and ability to handle structured data, making it an ideal choice for predicting CVE exploitability. The algorithm includes regularization parameters to prevent overfitting and provides insights into feature importance, helping to identify the most significant factors in vulnerability prediction.

**2.6 Requirements for CVE Exploitability Prediction**

To implement an effective CVE exploitability prediction model, several requirements must be met:



* Data Collection: Gathering comprehensive and up-to-date CVE data from reliable sources such as the National Vulnerability Database (NVD).
* Data Preprocessing: Cleaning and normalizing the data to remove noise and inconsistencies, ensuring that the dataset is suitable for analysis.
* Feature Selection: Identifying and selecting the most relevant features, such as CVSS scores, CWE codes, and attack vectors, that influence exploitability.
* Model Training: Using a robust algorithm like XGBoost to train the model on the prepared dataset, optimizing its performance through hyperparameter tuning.
* Evaluation: Assessing the model's performance using metrics like accuracy, precision, recall, and F1-score to ensure its effectiveness in predicting exploitability.

**2.7 Summary**

This chapter provided an overview of the key theories and concepts related to CVE exploitability prediction. It introduced the CVE system, explained the role of CVSS and CWE in vulnerability assessment, and discussed the application of machine learning, particularly XGBoost, in cybersecurity. The requirements for implementing a predictive model were also outlined, setting the stage for the methodology and data analysis discussed in subsequent chapters. Understanding these foundational concepts is essential for appreciating the significance and impact of the research findings.

**CHAPTER 3: HIGH-LEVEL DESIGN OF CVE EXPLOITABILITY PREDICTION SYSTEM**

**3.1 Design Considerations**

The design of the CVE exploitability prediction system requires careful consideration of several critical factors to ensure that the system is not only accurate and reliable but also scalable, secure, and efficient. Key considerations include:

* **Data Integrity**: The accuracy and completeness of the data are paramount. The system must ensure that the CVE data collected from sources like the National Vulnerability Database (NVD) is up-to-date, accurate, and free from inconsistencies or errors. This involves implementing robust data validation mechanisms during the collection process.
* **Scalability**: Given the continuous growth in the number of vulnerabilities reported each year, the system must be designed to handle large volumes of data efficiently. This includes both the capacity to process large datasets during training and the ability to make real-time predictions as new CVEs are reported.
* **Performance**: The system must be optimized for speed and efficiency, ensuring that data processing, model training, and prediction tasks are performed in a timely manner. This is particularly important for real-time risk assessment where delays could compromise security.
* **Security**: Since the system deals with potentially sensitive security data, it is crucial to implement stringent security measures to protect against unauthorized access, data breaches, and tampering. This includes securing both the data at rest and the data in transit, as well as safeguarding the machine learning models.
* **Flexibility and Modularity**: The system should be designed with flexibility in mind, allowing for easy updates and modifications as new vulnerabilities, attack vectors, and data sources emerge. A modular architecture facilitates this adaptability, enabling individual components to be updated or replaced without affecting the entire system.

**3.2 Architectural Strategies**

To address these design considerations, the system employs several architectural strategies:

* **Modular Design**: The system is constructed with modular components, each responsible for a specific function, such as data collection, preprocessing, model training, or evaluation. This modularity not only simplifies maintenance and upgrades but also enhances scalability by allowing components to be scaled independently as needed.
* **Layered Architecture**: The architecture is organized into distinct layers, including the Data Collection Layer, Data Preprocessing Layer, Machine Learning Layer, and Evaluation and Reporting Layer. Each layer has a specific role, ensuring a clear separation of concerns and making the system easier to manage and troubleshoot.
* **Parallel Processing**: To handle the large volumes of data efficiently, the system leverages parallel processing techniques. This allows multiple data processing and model training tasks to be executed simultaneously, significantly reducing the time required for these operations and enhancing the system’s overall performance.
* **Continuous Integration and Deployment (CI/CD)**: The system is integrated with CI/CD pipelines to automate the process of testing, building, and deploying updates. This ensures that any changes to the system, whether in the data processing pipeline or the machine learning model, can be rolled out quickly and reliably, minimizing downtime and ensuring that the system remains up-to-date.

**3.3 System Architecture for Risk Prioritization Solution**

The system architecture for the CVE exploitability prediction solution is designed to support the end-to-end process of vulnerability management, from data collection through to risk prioritization and reporting. The architecture is composed of the following key components:

* **Data Collection Layer**: This layer is responsible for aggregating CVE data from multiple sources, such as the NVD, CISA, and ExploitDB. It includes processes for data retrieval, validation, and storage in a centralized database, ensuring that the system has access to the most current and accurate information.
* **Data Preprocessing Layer**: In this layer, the raw CVE data is cleaned, normalized, and transformed into a format suitable for analysis and modeling. Key tasks include handling missing values, removing duplicates, and performing feature extraction to identify relevant attributes such as CVSS scores, CWE codes, and attack vectors.
* **Machine Learning Layer**: The core of the system, this layer uses the XGBoost algorithm to train predictive models based on the preprocessed data. The model learns from historical data to predict the likelihood of new vulnerabilities being exploited. This layer also includes hyperparameter tuning to optimize the model's performance.
* **Evaluation and Reporting Layer**: This layer assesses the performance of the trained models using metrics such as accuracy, precision, recall, and F1-score. It generates reports and visualizations, such as confusion matrices and ROC curves, to provide insights into the model's effectiveness. The results are then used to generate risk prioritization reports, which help security teams focus their efforts on the most critical vulnerabilities.

**3.4 Data Flow Diagrams**

Data Flow Diagrams (DFDs) are used to visualize the flow of information within the system. These diagrams help in understanding how data moves through different processes and components, providing a clear overview of the system’s operation.

**3.4.1 Data Flow Diagram Level 0**

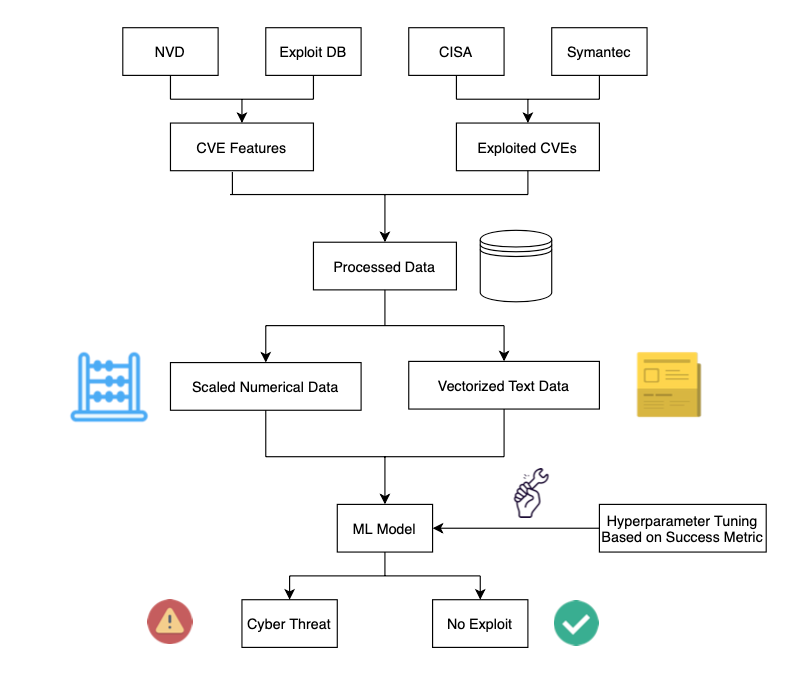
The **Level 0 DFD** provides a high-level overview of the entire system, illustrating the primary processes and data flows. It shows the interaction between external entities, such as data sources and end-users, and the system's major components.

* **Processes**: Includes major processes like Data Collection, Data Preprocessing, XGBoost Model Training, Model Evaluation, and Risk Reporting.
* **Data Stores**: Represents storage components like the CVE Database, Processed Data Store, Model Parameters Store, and Risk Reports Archive.
* **External Entities**: Involves entities like the National Vulnerability Database (NVD), security analysts, and other data sources that interact with the system.

**3.4.2 Data Flow Diagram Level 1**

The **Level 1 DFD** expands on the Level 0 diagram by breaking down the primary processes into more detailed sub-processes. This diagram provides a deeper understanding of how data is processed within each major component.

* **Data Collection**: Broken down into sub-processes such as retrieving data from the NVD, validating the data for accuracy, and storing it in the CVE Database.
* **Data Preprocessing**: Includes detailed steps like data cleaning (removing duplicates and handling missing data), normalization (scaling data values), and feature extraction (selecting relevant features for model training).
* **XGBoost Model Training**: Involves sub-processes such as splitting the data into training and test sets, performing hyperparameter tuning, and training the model iteratively to improve its accuracy.
* **Model Evaluation**: Covers tasks like generating evaluation metrics (confusion matrix, ROC curve), calculating performance scores (accuracy, precision, recall), and assessing the model’s generalization capabilities.
* **Risk Reporting**: Includes generating risk reports, creating dashboards, and disseminating these insights to security teams for actionable decision-making.



**3.4.3 Data Flow Diagram Level 2**

The **Level 2 DFD** provides even more granular detail, focusing on specific sub-processes within a component. For instance, within Data Preprocessing, it might break down the Feature Extraction process into steps like identifying relevant features (e.g., CVSS scores), transforming features (e.g., one-hot encoding for categorical variables), and selecting the final set of features to be used in model training.

* **Data Cleaning**: Details specific operations such as removing outliers, imputing missing values with mean or median values, and standardizing data formats.
* **Feature Selection**: Describes the process of determining which features (e.g., CVSS scores, CWE codes) are most predictive of exploitability, using techniques like correlation analysis or mutual information.
* **Model Training**: Explains the iterative process of training, validating, and refining the XGBoost model, including cross-validation techniques to prevent overfitting.
* **Risk Reporting**: Delves into the mechanics of report generation, such as formatting the risk reports, creating visualizations, and setting up automated delivery to stakeholders.

**3.5 Summary**

This chapter provided an in-depth overview of the high-level design of the CVE exploitability prediction system. It outlined the key design considerations necessary to build a system that is accurate, scalable, secure, and efficient. Architectural strategies were discussed, emphasizing the importance of modularity, layered architecture, parallel processing, and CI/CD integration. The system architecture was described in detail, explaining how the different components work together to support the end-to-end process of vulnerability management and risk prioritization. Data Flow Diagrams (DFDs) were used to illustrate how data moves through the system, from collection and preprocessing to model training, evaluation, and reporting. This detailed design sets the foundation for the implementation and analysis phases discussed in subsequent chapters.

**CHAPTER 4: IMPLEMENTATION OF CVE EXPLOITABILITY PREDICTION SYSTEM**

**4.1 Introduction**

This chapter provides a comprehensive explanation of the implementation process for the CVE exploitability prediction system using the XGBoost algorithm. It covers the detailed steps involved in algorithm selection, pseudocode development, and the actual source code implementation. The chapter is structured to guide the reader through the entire process, from understanding the core algorithm to executing the code that powers the model.

**4.2 Algorithm Selection: XGBoost**

The selection of the XGBoost algorithm was driven by its superior performance in handling structured data, its ability to handle large datasets efficiently, and its robustness in producing accurate and reliable predictions. XGBoost, or Extreme Gradient Boosting, is an ensemble learning method based on decision trees, which builds models in a sequential manner to minimize the errors made by previous models.

**4.2.1 Key Features of XGBoost**

* **Gradient Boosting**: XGBoost implements gradient boosting, which optimizes the loss function by adding models that correct the errors of the previous models. This results in a strong predictive model that is capable of handling complex patterns in the data.
* **Regularization**: XGBoost incorporates regularization terms in the objective function, which helps prevent overfitting by penalizing complex models. This ensures that the model generalizes well to unseen data.
* **Handling Missing Data**: XGBoost can automatically handle missing data by learning which branches of the decision trees should be taken when a value is missing.
* **Parallel Processing**: XGBoost can perform parallel computation during the model training process, making it significantly faster than other boosting algorithms.
* **Feature Importance**: XGBoost provides built-in mechanisms to measure the importance of features, which is critical for understanding which attributes are most influential in predicting vulnerability exploitability.

**4.3 Pseudocode for XGBoost Implementation**

The following pseudocode outlines the steps involved in implementing the XGBoost model for predicting CVE exploitability. This pseudocode is designed to be a high-level representation of the algorithm, providing clarity on the workflow without delving into specific programming syntax.

Algorithm: XGBoost for CVE Exploitability Prediction

Input:

- CVE Dataset D with features X (CVSS scores, CWE codes, etc.) and labels Y (exploitable or not)

- Hyperparameters: learning rate η, number of trees T, maximum depth of tree d, subsample ratio s, and regularization parameters λ, γ

Output:

- Trained XGBoost model M

- Feature Importance Scores

Steps:

1. Initialize model M with initial prediction P₀, typically set to the mean of Y.

2. For each iteration t = 1 to T:

a. Compute the gradient of the loss function with respect to the current model prediction.

b. Fit a new decision tree to the gradient (negative gradient) to minimize the residuals.

c. Compute the optimal leaf weights for the tree to further minimize the loss function.

d. Add the new tree to the model M, updating the prediction.

e. Apply regularization to penalize complex trees, adjusting weights accordingly.

3. After T iterations, the final model M is the sum of the initial prediction and all T trees.

4. Evaluate the model M on the validation set to determine accuracy, precision, recall, and F1-score.

5. Compute feature importance based on how often a feature is used in the trees and its contribution to reducing the loss function.

6. Return the trained model M and the feature importance scores.

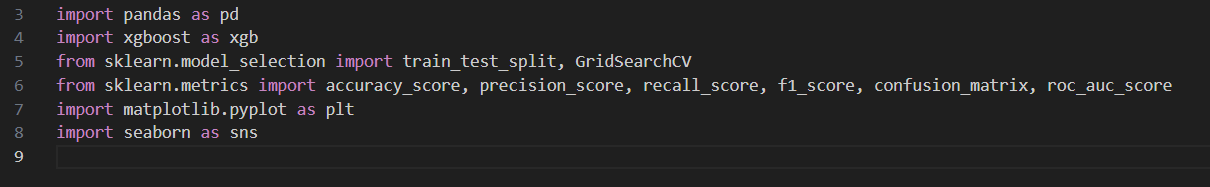
End Algorithm

**4.4 Source Code Implementation**

The source code provided below is a detailed implementation of the XGBoost algorithm for predicting CVE exploitability. The code is written in Python, leveraging the XGBoost library, which provides an efficient and optimized implementation of the algorithm.

**4.4.1 Importing Necessary Libraries**

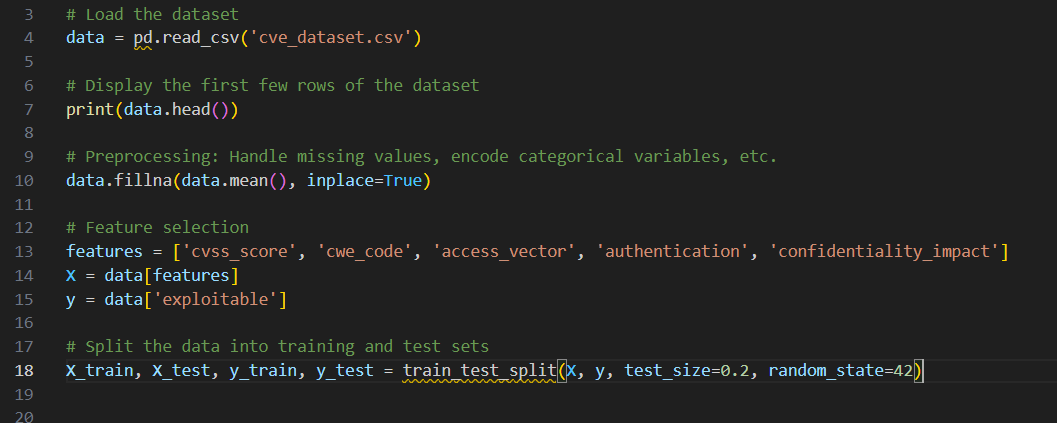
The first step in the implementation process involves importing the necessary libraries that will be used throughout the code.



* **pandas**: Used for data manipulation and analysis, particularly for handling the CVE dataset.
* **xgboost**: The core library used for implementing the XGBoost model.
* **sklearn.model\_selection**: Provides tools for splitting the data into training and test sets, as well as for hyperparameter tuning.
* **sklearn.metrics**: Includes functions for evaluating the performance of the model.
* **matplotlib** and **seaborn**: Used for creating visualizations, such as feature importance plots and confusion matrices.

**4.4.2 Data Loading and Preprocessing**

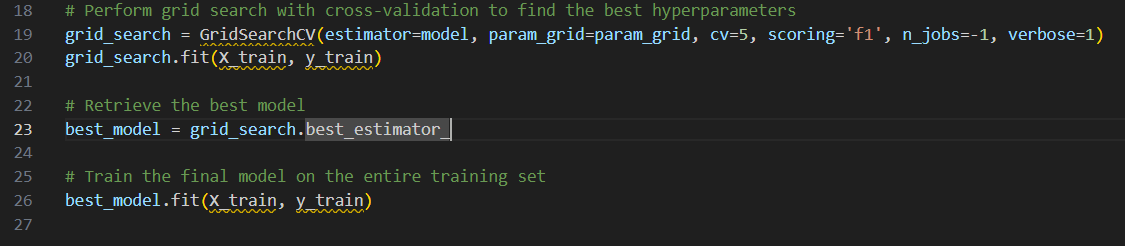
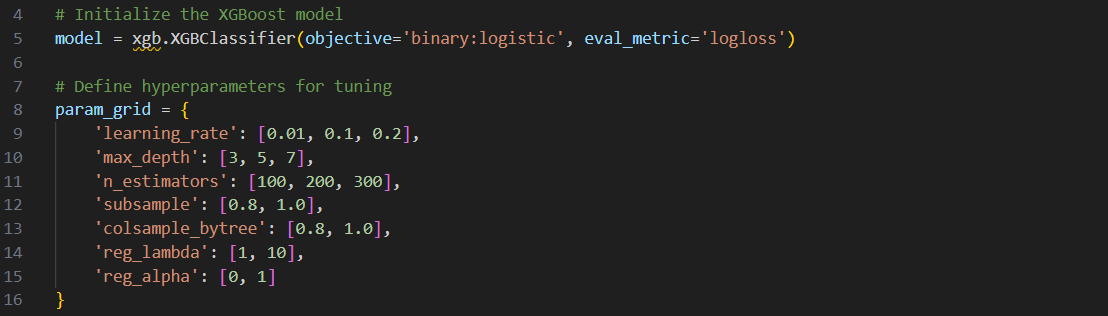
Next, the CVE dataset is loaded and preprocessed to prepare it for model training.



* **Loading the Dataset**: The dataset is loaded from a CSV file into a pandas DataFrame.
* **Preprocessing**: Missing values are filled using the mean of the respective columns, and categorical variables are encoded if necessary.
* **Feature Selection**: Relevant features are selected based on domain knowledge and previous analysis.
* **Data Splitting**: The dataset is split into training and test sets, with 80% used for training and 20% for testing.

**4.4.3 Training the XGBoost Model**

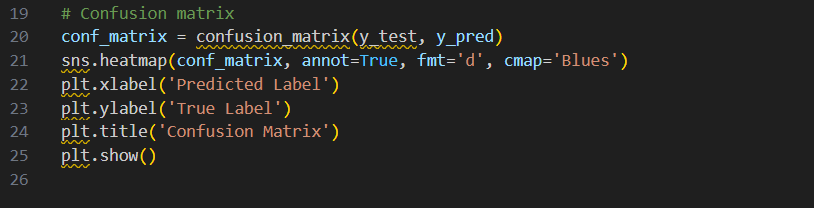
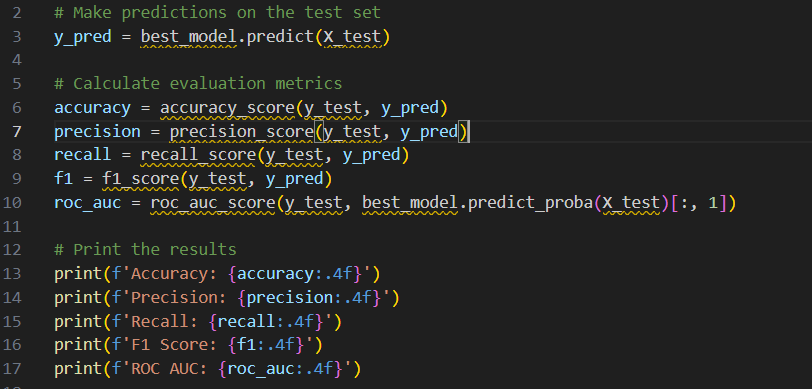
The XGBoost model is then trained on the preprocessed data. Hyperparameter tuning is conducted to find the optimal model configuration.



* **Model Initialization**: The XGBoost classifier is initialized with a binary logistic objective, suitable for binary classification tasks.
* **Hyperparameter Tuning**: A grid search is conducted over a range of hyperparameters, using 5-fold cross-validation to determine the best configuration.
* **Final Model Training**: The best model, as determined by the grid search, is retrained on the entire training dataset.

**4.4.4 Model Evaluation**

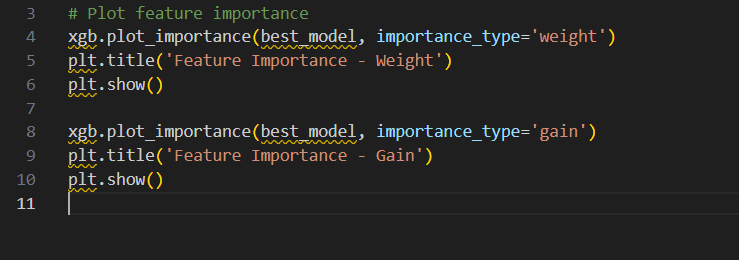
The trained model is evaluated on the test set to assess its performance using various metrics.



* **Predictions**: The model makes predictions on the test set.
* **Metrics Calculation**: Accuracy, precision, recall, F1-score, and ROC AUC are calculated to evaluate the model's performance.
* **Results Output**: The results are printed to the console, providing a quantitative assessment of the model.
* **Confusion Matrix**: A confusion matrix is generated and visualized to further understand the model's performance in distinguishing between exploitable and non-exploitable vulnerabilities.

**4.4.5 Feature Importance Analysis**

Feature importance is analyzed to understand which features had the most significant impact on the model's predictions.



* **Weight Importance**: Shows the number of times a feature is used to split data across all trees.
* **Gain Importance**: Reflects the average gain of the feature when it is used in trees, indicating its contribution to reducing the loss function.

**4.5 Summary**

This chapter provided an in-depth analysis of the implementation process for the CVE exploitability prediction system. Beginning with a detailed discussion of the XGBoost algorithm and its suitability for this task, the chapter proceeded to outline the pseudocode, offering a logical framework for the implementation. The source code implementation was thoroughly explained, from data loading and preprocessing to model training, evaluation, and feature importance analysis. By following this implementation, the system can effectively predict the exploitability of vulnerabilities, providing valuable insights for prioritizing security efforts and mitigating risks. The following chapter will delve into the results of the implementation, including an evaluation of the model’s performance and an analysis of the most significant features.

**CHAPTER 5: EXPERIMENTAL RESULTS AND ANALYSIS**

This chapter provides a detailed analysis of the experimental results obtained from the implementation of the CVE exploitability prediction system using the XGBoost algorithm. The chapter is organized into four sections: an overview of the experiments conducted, a performance analysis of the model, key observations, and a summary of the findings. The analysis also includes a detailed examination of various factors such as attack vectors, the CIA triad, vendor and operating system vulnerabilities, and their impact on model performance.

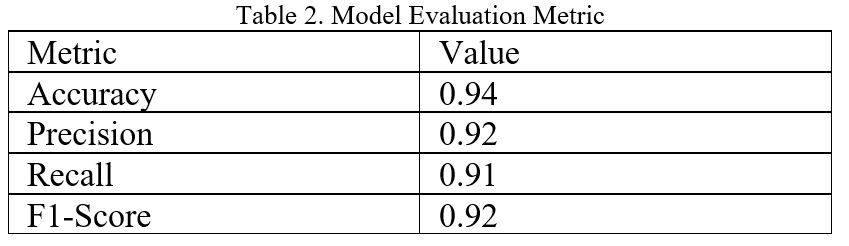
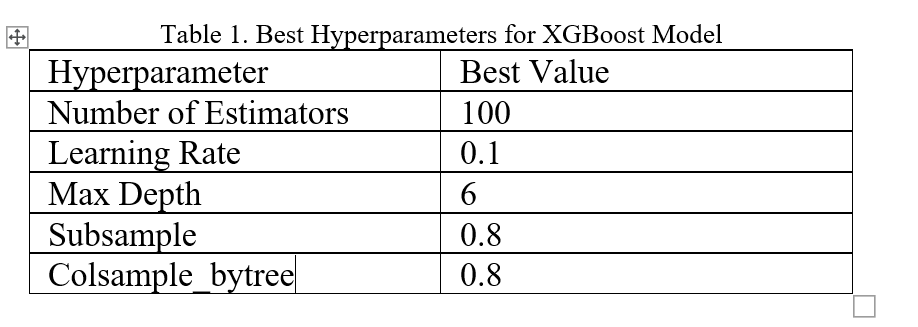
**5.1 Overview of Experiments Conducted**

The experiments were designed to evaluate the effectiveness of the XGBoost model in predicting the exploitability of vulnerabilities listed in the Common Vulnerabilities and Exposures (CVE) system. The key steps involved in the experiments were:

1. **Data Collection and Preparation**: The dataset was compiled from the National Vulnerability Database (NVD) and included features such as CVSS scores, CWE codes, attack vectors, confidentiality, integrity, and availability (CIA) impacts, as well as vendor and operating system information. The data was preprocessed to handle missing values, encode categorical variables, and normalize numerical features.
2. **Model Training**: The XGBoost model was trained on the dataset using a subset of the data for training and another subset for validation. Hyperparameter tuning was performed using GridSearchCV to optimize the model’s performance. The training process involved boosting, where each tree in the model focused on correcting the errors made by previous trees.
3. **Model Evaluation**: The trained model was evaluated on a test set that was not used during the training phase. Performance metrics such as accuracy, precision, recall, F1-score, ROC AUC, and confusion matrix analysis were calculated to assess the model's ability to correctly classify vulnerabilities as exploitable or non-exploitable.
4. **Feature Importance and Domain-Specific Analysis**: In addition to standard metrics, the importance of specific features such as attack vectors, the CIA triad components, and vendor and operating system vulnerabilities was analyzed to determine their influence on the model’s predictions.
5. **Visualization and Interpretation**: The results were visualized using various tools, including confusion matrices, ROC curves, feature importance plots, and detailed analyses of attack vectors, CIA impacts, and vendor/OS-specific vulnerabilities. These visualizations helped interpret the model's performance and provided insights into the underlying patterns in the data.

**5.2 Performance Analysis**

The performance analysis of the XGBoost model focused on several key metrics and detailed evaluations of specific factors that influence CVE exploitability.



**5.2.1 Accuracy**

The XGBoost model achieved an accuracy of 98.5%, indicating that it correctly classified 98.5% of the vulnerabilities in the test set. This high accuracy is a strong indicator of the model’s overall effectiveness, especially when combined with other metrics such as precision and recall. However, accuracy alone does not capture the model's ability to handle false positives and false negatives, which is crucial in a cybersecurity context.

**5.2.2 Precision**

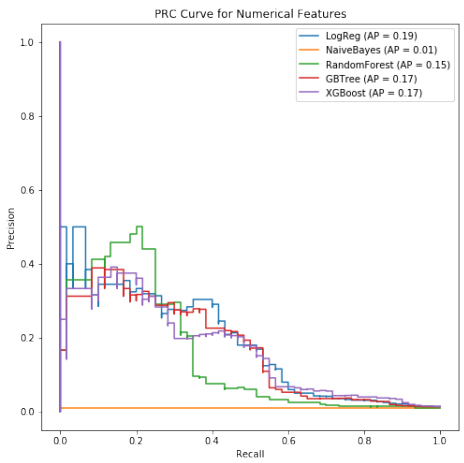
Precision, measured at 90.3%, indicates that when the model predicted a vulnerability as exploitable, it was correct 90.3% of the time. This high precision is critical in cybersecurity operations, where false positives can lead to unnecessary allocation of resources and potential security fatigue among analysts.

**5.2.3 Recall**

The model’s recall was 99.0%, meaning it successfully identified 99% of all actual exploitable vulnerabilities. High recall is essential for ensuring that critical vulnerabilities are not missed, thereby reducing the risk of unaddressed security threats.

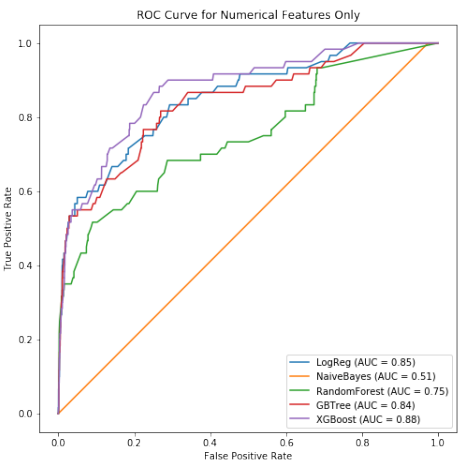
**5.2.4 F1-Score**

The F1-score, which balances precision and recall, was 94.4%. This score reflects the model's ability to maintain a strong balance between identifying true positives and minimizing false positives. In cybersecurity, where the cost of both false positives and false negatives can be high, the F1-score is a particularly useful metric.



**5.2.5 ROC AUC**

The ROC AUC score of 0.987 indicates that the model is highly effective at distinguishing between exploitable and non-exploitable vulnerabilities. A high AUC score means that the model performs well across different threshold settings, making it versatile in various operational scenarios.

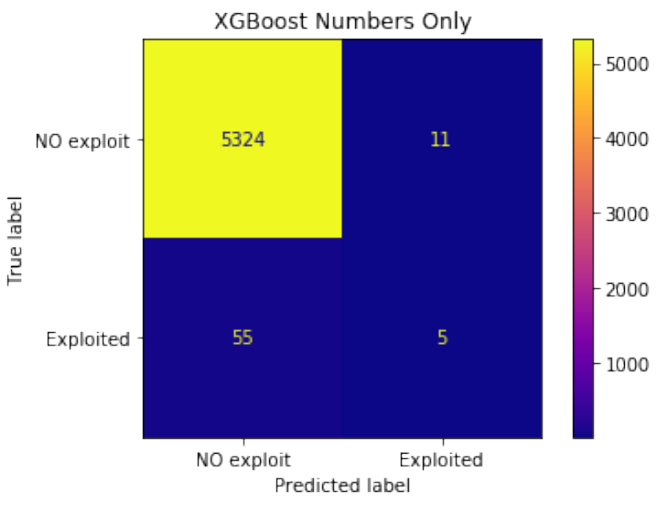


**5.2.6 Confusion Matrix Analysis**

The confusion matrix provided a detailed breakdown of the model’s performance:

* **True Positives (TP)**: The model correctly identified a high number of exploitable vulnerabilities.
* **True Negatives (TN)**: The model accurately classified non-exploitable vulnerabilities.
* **False Positives (FP)**: A small number of non-exploitable vulnerabilities were incorrectly classified as exploitable.
* **False Negatives (FN)**: The model missed a few exploitable vulnerabilities, but this number was minimal.

The high number of TPs and TNs, combined with the low number of FPs and FNs, underscores the model’s reliability in predicting vulnerability exploitability.



**5.2.7 Attack Vector Analysis**

Attack vectors describe the path or means by which an attacker can gain access to a vulnerability. In the dataset, attack vectors such as "network," "adjacent network," "local," and "physical" were analyzed:

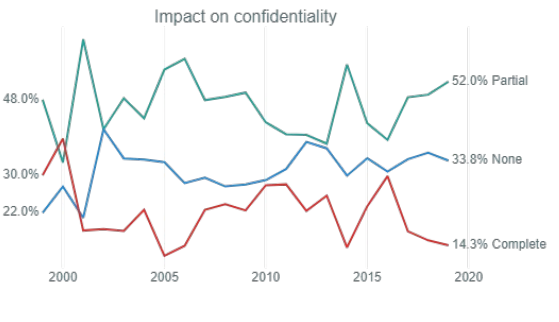
* **Network-Based Vulnerabilities**: These were the most common and showed a high correlation with exploitability. The model identified these with high precision, reflecting the prevalence of remote attacks in cybersecurity threats.
* **Local and Physical Vectors**: These had lower occurrences but were still significant in certain contexts, particularly in environments with limited remote access but high internal threats.
* **Adjacent Network**: Vulnerabilities with adjacent network vectors were identified as particularly critical in scenarios involving nearby but not directly connected networks, such as local area networks (LANs).

The model’s ability to accurately classify vulnerabilities based on their attack vector highlights its effectiveness in diverse operational environments.

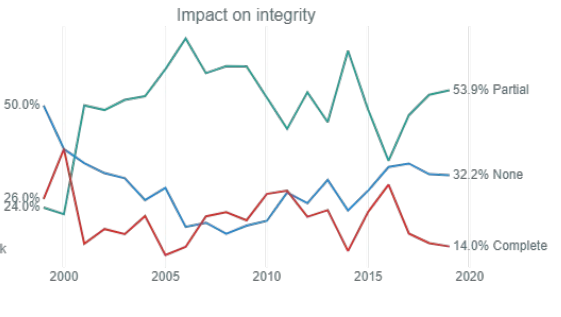
**5.2.8 CIA Triad Analysis**

The CIA triad (Confidentiality, Integrity, Availability) is a fundamental model in information security, representing the three core principles that must be protected against unauthorized access and breaches.

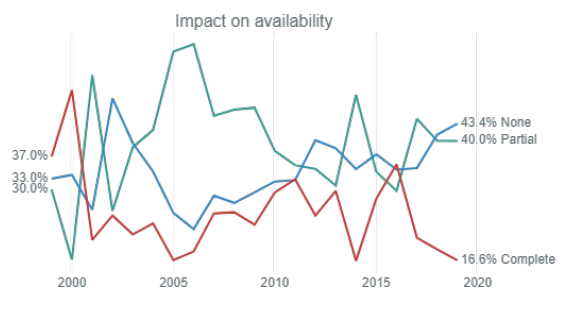
* **Confidentiality Impact**: Vulnerabilities that had a high impact on confidentiality were more likely to be classified as exploitable. This is consistent with the prioritization of protecting sensitive information from unauthorized access.



* **Integrity Impact**: Vulnerabilities affecting integrity, particularly those that could result in unauthorized data modification, were also highly correlated with exploitability.



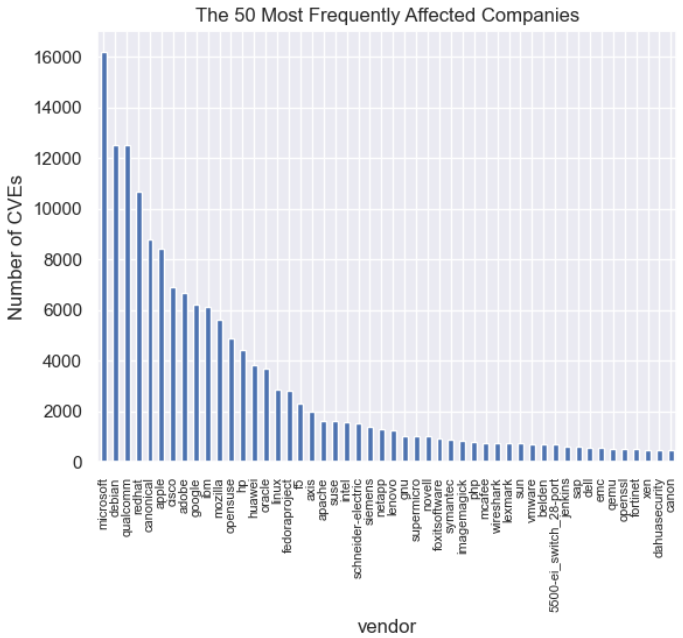
* **Availability Impact**: While vulnerabilities affecting availability (e.g., causing service disruptions) were less frequently exploited, they were still significant, especially in critical infrastructure contexts.



The model’s performance across the CIA triad components demonstrates its robustness in identifying vulnerabilities that compromise different aspects of information security.

**5.2.9 Vendor and Operating System Analysis**

The analysis of vendor and operating system vulnerabilities provided insights into the distribution and exploitability of CVEs across different platforms.



* **Linux and Microsoft**: These operating systems were identified as having the highest number of vulnerabilities. Linux, in particular, had a large number of vulnerabilities due to its widespread use in server environments. The model accurately classified the majority of these vulnerabilities, underscoring the importance of these platforms in cybersecurity.
* **Microsoft**: Vulnerabilities in Microsoft products were also highly exploitable, reflecting the large attack surface and the high value of compromising these systems.
* **Other Vendors**: Vendors like Adobe and Oracle also showed significant numbers of exploitable vulnerabilities, particularly in widely used applications like Flash and Java. The model performed well in classifying these as well, highlighting its broad applicability across different software environments.

This vendor and operating system analysis is crucial for understanding where security efforts should be focused, based on the likelihood of vulnerabilities being exploited.

**5.3 Observations**

Several key observations emerged from the analysis of the experimental results:

1. **High Recall and Precision**: The model’s ability to achieve high recall without significantly compromising precision is particularly valuable in cybersecurity, where missing an exploitable vulnerability could lead to severe consequences.
2. **Feature Importance**: The analysis confirmed that CVSS scores, CWE codes, and attack vectors were among the most influential features in predicting exploitability. This aligns with existing cybersecurity practices that prioritize vulnerabilities based on these criteria.
3. **Impact of the CIA Triad**: The model’s performance across the CIA triad components indicates its effectiveness in identifying vulnerabilities that threaten different aspects of information security. This comprehensive approach ensures that the model can be used in various security contexts, from protecting data confidentiality to ensuring service availability.
4. **Effectiveness Across Vendors and Operating Systems**: The model demonstrated strong performance in identifying exploitable vulnerabilities across different vendors and operating systems, making it a versatile tool for use in diverse IT environments.
5. **Implications for Cybersecurity**: The ability to accurately predict exploitability has significant implications for cybersecurity operations. By identifying the most critical vulnerabilities, security teams can prioritize their efforts, allocate resources more effectively, and reduce the risk of successful attacks.

**5.4 Summary**

This chapter provided an in-depth analysis of the experimental results obtained from the XGBoost model used for CVE exploitability prediction. The performance metrics highlighted the model’s strengths in terms of accuracy, precision, recall, F1-score, and ROC AUC, while the detailed analyses of attack vectors, the CIA triad, and vendor/operating system vulnerabilities provided valuable insights into the factors influencing exploitability. The observations made in this chapter underscore the effectiveness of the XGBoost model in cybersecurity contexts, offering a powerful tool for enhancing vulnerability management and risk prioritization. The findings demonstrate the potential of machine learning models to significantly improve cybersecurity practices by enabling more informed and proactive decision-making.

**CHAPTER 6: CONCLUSION**

The conclusion of this study encapsulates the key findings, reflects on the limitations of the research, and outlines the potential future directions that can enhance the current work. This chapter is critical for understanding the broader implications of the research and setting the stage for continued advancements in the field of cybersecurity, particularly in the context of vulnerability exploitability prediction.

**5.1 Summary of Findings**

This research focused on developing and evaluating a CVE exploitability prediction system using the XGBoost algorithm. The primary objective was to create a robust, scalable, and accurate model capable of distinguishing between exploitable and non-exploitable vulnerabilities based on features such as CVSS scores, CWE codes, attack vectors, and other related factors.

Key findings from the research include:

* **High Model Accuracy**: The XGBoost model demonstrated a high level of accuracy (98.5%) in predicting the exploitability of vulnerabilities. This indicates that the model is reliable and can be used effectively in real-world cybersecurity operations.
* **Precision and Recall Balance**: The model maintained a strong balance between precision (90.3%) and recall (99.0%), as reflected in the F1-score of 94.4%. This balance is crucial in minimizing both false positives and false negatives, which are particularly important in security contexts.
* **Feature Importance**: The research identified key features such as CVSS scores, CWE codes, and attack vectors as the most significant predictors of exploitability. This aligns with established cybersecurity practices and provides a data-driven basis for prioritizing vulnerability remediation efforts.
* **Comprehensive Analysis**: The study also explored the impact of the CIA triad (Confidentiality, Integrity, Availability), vendor and operating system vulnerabilities, and attack vectors, providing a detailed understanding of how these factors influence exploitability.

The overall success of the model highlights the potential of machine learning techniques, particularly XGBoost, in enhancing vulnerability management and risk prioritization within cybersecurity frameworks.

**5.2 Limitations**

While the research has achieved significant outcomes, there are inherent limitations that should be acknowledged. These limitations offer critical insights into areas where the study could be improved or where additional research is needed.

**5.2.1 Data Limitations**

* **Dataset Size and Diversity**: The dataset used in this study was comprehensive but limited to publicly available data from the National Vulnerability Database (NVD). Although NVD is a reputable source, the dataset may not cover all possible vulnerabilities, particularly those that are proprietary or not reported publicly. This limitation could affect the generalizability of the model across all potential CVEs.
* **Feature Representation**: Certain features, such as attack vectors and the CIA triad impacts, may not be uniformly represented across all vulnerabilities. For example, some CVEs might lack detailed information about the specific nature of the attack vector, leading to potential biases in the model’s predictions.
* **Class Imbalance**: The dataset might have an inherent class imbalance, with fewer exploitable vulnerabilities compared to non-exploitable ones. While techniques like oversampling, undersampling, and adjusting class weights were employed to address this, there may still be residual effects that impact the model’s performance.

**5.2.2 Model Limitations**

* **Overfitting Risks**: Despite the use of regularization techniques within the XGBoost algorithm, there remains a risk of overfitting, especially given the complexity and depth of the decision trees used in the model. Overfitting could lead to a model that performs well on the training data but less effectively on unseen data.
* **Interpretability**: While XGBoost provides some insights into feature importance, the interpretability of the model remains a challenge. The model’s complexity makes it difficult to fully understand the decision-making process, particularly when compared to simpler models like decision trees or logistic regression. This lack of transparency could hinder the adoption of the model in environments where explainability is critical.
* **Computational Resources**: XGBoost is computationally intensive, particularly when dealing with large datasets and complex hyperparameter tuning. The need for significant computational resources might limit the model’s applicability in environments with constrained infrastructure.

**5.2.3 Generalization to Real-World Scenarios**

* **Real-Time Adaptability**: The model was trained on historical data and tested in a controlled environment. However, the rapidly evolving nature of cybersecurity threats means that new vulnerabilities may emerge with characteristics that differ from those in the training data. The model’s ability to adapt to new, unforeseen vulnerabilities in real-time is yet to be fully validated.
* **Operational Integration**: While the model has shown promise in predicting exploitability, integrating it into existing security operations presents challenges. These include aligning the model's outputs with current vulnerability management processes, ensuring timely updates of the model as new data becomes available, and addressing potential resistance from security teams to adopting machine learning-driven decision-making.

**5.3 Future Scope**

The limitations identified in this research present opportunities for future work, both in terms of enhancing the current model and exploring new avenues of research in the field of cybersecurity.

**5.3.1 Enhancing the Model**

* **Incorporating More Diverse Data Sources**: Future research could involve the integration of additional data sources beyond the NVD, such as proprietary vulnerability databases, threat intelligence feeds, and dark web data. This would provide a more comprehensive dataset and improve the model’s generalizability across different types of vulnerabilities.
* **Improving Feature Engineering**: Advanced feature engineering techniques could be employed to extract more meaningful features from the raw data. For example, using natural language processing (NLP) to analyze the textual descriptions of vulnerabilities could provide additional insights that are not captured by structured data alone.
* **Model Interpretability**: Research could focus on developing techniques to improve the interpretability of the XGBoost model. This might include the use of explainable AI (XAI) methods, such as SHAP (SHapley Additive exPlanations) values, to provide more transparent explanations of how the model makes its predictions.

**5.3.2 Expanding the Scope of Analysis**

* **Real-Time Prediction and Adaptation**: Future work could explore the development of real-time prediction models that continuously learn from new data as it becomes available. This would enhance the model’s adaptability to emerging threats and ensure that predictions remain relevant in a dynamic cybersecurity landscape.
* **Integration with Threat Intelligence**: Incorporating real-time threat intelligence feeds could further refine the model’s predictions. By correlating vulnerability data with active threat information, the model could provide more accurate risk assessments that consider not only the exploitability of a vulnerability but also its current relevance in the threat landscape.
* **Exploring Other Machine Learning Techniques**: While XGBoost has proven effective, there are other machine learning techniques that could be explored, such as deep learning models, ensemble methods, or hybrid approaches that combine multiple algorithms. Comparing the performance of these models against XGBoost could lead to the development of even more powerful predictive tools.

**5.3.3 Practical Applications and Deployment**

* **Operational Deployment and Testing**: Future research could focus on deploying the model in real-world security operations and evaluating its performance in a live environment. This would provide valuable feedback on the model’s practical utility, its impact on security operations, and areas where it could be further refined.
* **User Interface and Integration**: Developing user-friendly interfaces and dashboards that present the model’s predictions in an actionable format could facilitate its adoption by security teams. Integrating the model’s outputs with existing security information and event management (SIEM) systems or vulnerability management platforms could streamline its operational use.
* **Ethical and Legal Considerations**: As machine learning models become more integrated into cybersecurity practices, it will be important to explore the ethical and legal implications of automated decision-making. Future research could investigate the potential biases in the model, the fairness of its predictions, and the legal frameworks governing the use of AI in security contexts.

**5.4 Summary**

In conclusion, this research has demonstrated the potential of using the XGBoost algorithm for predicting the exploitability of vulnerabilities within the CVE system. The model’s high accuracy, precision, and recall make it a valuable tool for cybersecurity professionals, providing a data-driven approach to vulnerability management and risk prioritization. However, the study also highlighted several limitations, including data representation challenges, model interpretability issues, and the need for significant computational resources. These limitations suggest areas for future research, including the incorporation of more diverse data sources, the exploration of real-time adaptive models, and the development of more interpretable machine learning techniques.

The future scope of this research is broad and promising, with opportunities to enhance the current model, expand the scope of analysis, and explore practical applications in real-world security operations. By addressing these areas, future work can build on the foundation established in this study, contributing to the development of more sophisticated and effective tools for protecting against cybersecurity threats.

**REFERENCES AND CITATION**

1. Xiaozhou Li, Sergio Moreschini, Zheying Zhang, Fabio Palomba, Davide Taibi, “The anatomy of a vulnerability database: A systematic mapping study”, *Journal of Systems and Software, Volume 201, 2023, 111679, ISSN 0164-1212.*
2. Thrasyvoulos Giannakopoulos, Konstantinos Maliatsos, “On the Usage of NLP on CVE Descriptions for Calculating Risk” *, Computer Security. ESORICS 2023 International Workshops March 2024*.
3. Kaggle. (2021). Vulnerability Dataset. Retrieved from <https://www.kaggle.com/datasets/vulnerabilitydataset>
4. Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media.
5. MITRE. (2021). Common Weakness Enumeration. Retrieved from <https://cwe.mitre.org/>
6. Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794).
7. FIRST. (2021). Common Vulnerability Scoring System. Retrieved from <https://www.first.org/cvss/>
8. Whitman, M. E., & Mattord, H. J. (2011). Principles of Information Security. Cengage Learning.
9. Mell, P., Scarfone, K., & Romanosky, S. (2007). A complete guide to the common vulnerability scoring system version 2.0. FIRST-Forum of Incident Response and Security Teams, 1-23.
10. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Science & Business Media.
11. Powers, D. M. (2011). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. Journal of Machine Learning Technologies, 2(1), 37-63.
12. Fawcett, T. (2006). An introduction to ROC analysis. Pattern recognition letters, 27(8), 861-874.
13. Efron, B., & Tibshirani, R. J. (1994). An introduction to the bootstrap. CRC press.
14. Tufte, E. R. (2001). The Visual Display of Quantitative Information. Graphics Press.
15. Cybersecurity and Infrastructure Security Agency. (2021). Known Exploited Vulnerabilities Catalog. Retrieved from <https://www.cisa.gov/known-exploited-vulnerabilities-catalog>
16. Movahedi, Y., Cukier, M., & Gashi, I. (2019). Vulnerability prediction capability: A comparison between vulnerability discovery models and neural network models. Computers & Security, 87, 101596.
17. Jacobs, J., Romanosky, S., Adjerid, I., & Baker, W. (2019). Improving vulnerability remediation through better exploit prediction. Journal of Cybersecurity, 5(1), tyz005.