# Intrusion Detection System for Cyber Security

**A PROJECT REPORT**

***Submitted by***

**NAME OF THE CANDIDATE(S)**

Syed Furqan Jamal - 21BCS8008

Kushagra Saran - 21BCS8066

Pushpendra Pandey - 20BCS3465

Suraj Singh Lalotra - 21BCS8123

Jasmine Mehra - 20BCS9886

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**Chandigarh University**

MAY 2024

## BONAFIDE CERTIFICATE

Certified that this project report **"Intrusion Detection System for Cybersecurity"** is the bonafide work of **"Syed Furqan Jamal, Kushagra Saran, Suraj Singh Lalotra, Jasmine Mehra, Pushpendra Pandey"** who carried out the project work under my/our supervision.

| | |
|---|---|
| **SIGNATURE** | **SIGNATURE** |
| Dr. Navpreet Kaur Walia (E7347) | Ankita Sharma (E11389) |
| **HEAD OF THE DEPARTMENT** | **SUPERVISOR** |
| Computer Science & Engineering | Computer science & engineering |

Submitted for the project viva-voce examination held on............................... 2024

| | |
|---|---|
| **INTERNAL EXAMINER** | **EXTERNAL EXAMINER** |

## ACKNOWLEDGMENT

We are genuinely grateful to everyone whose support and encouragement made this report possible. A special shoutout goes to our supervisor, Ankita Sharma (E11389) Ma'am, whose wisdom, guidance, and unwavering belief in our abilities guided us through the entire process of creating this report. Her patience in helping us refine our work and her motivation kept us going, even during challenging times. We can't thank her enough for her meticulous proofreading and correcting our mistakes. Her dedication truly elevated the quality of our work. Our lecturers and supervisors also deserve a heartfelt thank you for their unwavering support and encouragement. Their constant guidance helped us stay focused on our goals and kept us moving forward, step by step. And to our classmates and friends, your camaraderie and willingness to lend a hand whenever we stumbled along the way meant the world to us. The late-night brainstorming sessions and the shared excitement during our progress were priceless moments that made this journey unforgettable.

Syed Furqan Jamal (21BCS8008)
Suraj Singh Lalotra (21BCS8123)
Jasmine Mehra (20BCS9886)
Kushagra Saran (21BCS8066)
Pushpendra Pandey (20BCS3465)

# TABLE OF CONTENTS

# List of Figures

## List of Tables

## ABSTRACT

In today's digital interaction environment, it is important to ensure the security of the computer network. Intrusion detection systems (IDS) play a key role in identifying and mitigating threats, from vulnerabilities to competitive cyber attacks. Traditionally, IDS has relied on and abused faulty detection techniques. Fault detection has the advantage of detecting new attacks by detecting deviations from normal behaviour, while fault detection has the advantage of detecting attack patterns with the right pressure. However, both approaches have limitations such as translation issues, insufficient data, and scalability issues. False alarm rate. Despite this progress, commercial use of machine learning-based vulnerability detection is still limited. The disparity between research and real-world applications highlights the need for a deeper understanding of the challenges and considerations of machine learning-based access to research search. analysis, review of current studies, and limitations. From the overall analysis, we found that machine learning holds great promise in improving accessibility, but many challenges need to be solved to realize its full potential. These challenges include interpretation of machine learning models, quality and quantity of data, and applicability of real-world solutions. It combines anomaly and abuse detection techniques with feature engineering and continuous learning mechanisms. This approach focuses on leveraging the advantages of both research methods while minimizing the limitations of each. This report helps the cybersecurity industry by addressing these issues and providing new solutions, increasing the ability of networks to protect against new threats in the evolving digital environment.

# सारांश

आज के डिजिटल इंटरेक्शन परिवेश में, कंप्यूटर नेटवर्क की सुरक्षा सुनिश्चित करना महत्वपूर्ण है। घुसपैठ का पता लगाने वाली प्रणालियाँ (आईडीएस) कमजोरियों से लेकर प्रतिस्पर्धी साइबर हमलों तक खतरों की पहचान करने और उन्हें कम करने में महत्वपूर्ण भूमिका निभाती हैं। परंपरागत रूप से, आईडीएस ने दोषपूर्ण पहचान तकनीकों पर भरोसा किया है और उनका दुरुपयोग किया है। दोष का पता लगाने में सामान्य व्यवहार से विचलन का पता लगाकर नए हमलों का पता लगाने का लाभ होता है, जबकि दोष का पता लगाने में सही दबाव के साथ हमले के पैटर्न का पता लगाने का लाभ होता है। हालाँकि, दोनों दृष्टिकोणों में अनुवाद संबंधी समस्याएँ, अपर्याप्त डेटा और स्केलेबिलिटी समस्याएँ जैसी सीमाएँ हैं। गलत अलार्म दर. इस प्रगति के बावजूद, मशीन लर्निंग-आधारित भेद्यता पहचान का व्यावसायिक उपयोग अभी भी सीमित है। अनुसंधान और वास्तविक दुनिया के अनुप्रयोगों के बीच असमानता अनुसंधान खोज के लिए मशीन लर्निंग-आधारित पहुंच की चुनौतियों और विचारों की गहरी समझ की आवश्यकता पर प्रकाश डालती है। विश्लेषण, वर्तमान अध्ययनों की समीक्षा और सीमाएँ। समग्र विश्लेषण से, हमने पाया कि मशीन लर्निंग पहुंच में सुधार करने की बड़ी संभावनाएं रखता है, लेकिन इसकी पूरी क्षमता का एहसास करने के लिए कई चुनौतियों को हल करने की आवश्यकता है। इन चुनौतियों में मशीन लर्निंग मॉडल की व्याख्या, डेटा की गुणवत्ता और मात्रा और वास्तविक दुनिया के समाधानों की प्रयोज्यता शामिल है। यह फीचर इंजीनियरिंग और निरंतर सीखने के तंत्र के साथ विसंगति और दुरुपयोग का पता लगाने वाली तकनीकों को जोड़ती है। यह दृष्टिकोण प्रत्येक की सीमाओं को कम करते हुए दोनों अनुसंधान विधियों के लाभों का लाभ उठाने पर केंद्रित है। यह रिपोर्ट इन मुद्दों को संबोधित करके और नए समाधान प्रदान करके साइबर सुरक्षा उद्योग को मदद करती है, जिससे उभरते डिजिटल वातावरण में नए खतरों से बचाने के लिए नेटवर्क की क्षमता बढ़ जाती है|

# CHAPTER 1
# INTRODUCTION

## 1.1 Client Identification

Administrators and network security professionals with a deeper understanding of cybersecurity The latest trends and best practices in IDS access will benefit from this report. They will gain valuable knowledge on how to use machine learning techniques to improve access to detection capabilities and effectively mitigate cyber threats.

**IT Managers and Decision Makers:** IT managers and decision makers responsible for monitoring cybersecurity and allocating resources to cybersecurity programs will find this report valuable. It provides an overview of the current state of IDS, highlighting challenges and considerations when using machine learning as an approach. The proposed hybrid approach provides strategic insights to improve access to effective detection to protect organizations' assets from evolving cyber threats.

**Researchers and Academics:** Researchers and academics working in cybersecurity, especially machine learning and cybersecurity, will find this report useful. It provides critical analysis of the latest research and limitations to research access, paving the way for further research and innovation in the field.

**Technology Vendors and Solution Providers:** Technology vendors and solution providers that offer intrusion detection systems and network security solutions can use this report to understand business needs and emerging trends. Gaining a deeper understanding of the effectiveness of machine learning-based approaches and the barriers to their adoption can inform the development of future products and services that meet the needs of cybersecurity professionals.

**Regulators and Compliance Officers:** Regulators and compliance officers responsible for ensuring compliance with cybersecurity standards and regulations will find this information useful. It helps create stronger cybersecurity policies and compliance by providing information about the capabilities and limitations of existing intrusion detection tools.

This report serves as a comprehensive reference for navigating the complex intrusion detection and sourcing landscape for improving cybersecurity in today's digital age.

## 1.2 Identification of Problem

In today's hyperconnected world, the security of computer networks is more important than ever. Intrusion detection systems (IDS) are the front-line defense against a variety of cyber threats, from vulnerabilities to attacks. Traditional IDS methods, including detection based on suspicion and abuse, have long been a cornerstone of cybersecurity. However, the emergence of machine learning promises to revolutionize intrusion detection by offering the ability to increase detection accuracy and reduce threats. Although machine learning shows great promise in educational research, real-world adoption is still limited. This section summarizes the main challenges hindering the implementation of machine learning as an IDS.

1. **Differences between research and real use:**

One of the main problems affecting the use of machine learning as an IDS is the large gap between research results and practical applications. Academic research often sees vulnerability detection as a profitable approach due to its ability to identify emerging threats. But commercial and practical use often favors prediction and accuracy of misuse as a discovery. This inconsistency highlights the difficulty of translating theoretical advances into practical solutions.

2. **Interpretability Challenges:**

One of the biggest challenges with machine learning-based IDS is the lack of interpretability of the models. Unlike traditional algorithms, machine learning models often operate as a "black box," making it difficult to understand how they make decisions. A lack of transparency can undermine trust in the system, especially in high-risk environments where the consequences of wrong or incorrect behaviour can be enormous.

3. **Data Scarcity and Quality:**

The effectiveness of machine learning models heavily relies on the quality and quantity of training data available. However, acquiring labeled data for training robust intrusion detection models can be challenging. Real-world datasets are often imbalanced, with a scarcity of examples of malicious activities compared to benign ones. Additionally, ensuring the quality and reliability of training data is crucial to avoid bias and ensure the generalizability of the models.

4. **Scalability Issues:**

Deploying machine learning-based IDS at scale poses significant scalability challenges. As network connectivity continues to grow, the IDS needs to be able to process large amounts of data in real time. This requires a large budget and infrastructure to accurately detect and respond to threats. Additionally, the latency introduced by machine learning algorithms can be prohibitive in real-time monitoring environments where timely detection is important.

5. **Lack of benchmarks:**

Another challenge of using machine learning based IDS is the lack of benchmarks. It is difficult to compare the performance of different detection tools without benchmarks. This lack of comparison hinders the adoption and use of machine learning-based IDS in business and the real world, where robust metrics are critical to decision making.

**6. Complexity of Real World Network Environments:**

Real world networks have unpredictable behavior, making it difficult to design intrusion prevention devices that can be modified to change the threat and gender in the network. It can be difficult to detect abnormal activity on the network with traditional methods, resulting in negative or invisible consequences. In addition, network integration and diversity of processes also affect the performance of creating effective access to search engines. Increase network security and reduce cyber threats. However, many challenges need to be solved to realize this potential. From translation issues to insufficient data, scalability issues, and lack of benchmarking, solving these problems requires collaboration between researchers, practitioners, and business stakeholders. By identifying and solving these problems, we can pave the way for machine learning-based IDS and improve cybersecurity in an increasingly digital world.

## 1.3 Identification of Tasks

**1. Data Preprocessing:**

- Task: Clean and preprocess the dataset to ensure data quality and prepare it for model training.

- Subtasks:
  - Handle missing values and outliers in the dataset.
  - Encode nominal features (e.g., Protocol_type, Service, Flag) using techniques such as one-hot encoding.
  - Standardize or normalize numeric features to ensure consistency in scale and distribution.

**2. Feature Selection:**

- Task: Identify and select relevant features that contribute most to distinguishing between normal network traffic and different types of attacks.

- Subtasks:
  - Conduct feature importance analysis to identify the most discriminative features for each attack class.
  - Use domain knowledge and correlation analysis to prioritize features that are most relevant for intrusion detection.

**3. Model Selection and Training:**

➢ Task: Choose appropriate machine learning algorithms and train models to classify network traffic into different attack classes.

➢ Subtasks:
- Experiment with various machine learning algorithms suitable for multi-class classification tasks, such as Random Forest, Support Vector Machines (SVM), or Gradient Boosting.
- Utilize cross-validation techniques to evaluate model performance and select the best-performing algorithm(s).
- Fine-tune hyperparameters of selected models to optimize performance metrics such as accuracy, precision, recall, and F1-score.

**4. Interpretability and Explainability:**

➢ Task: Enhance the interpretability and explainability of machine learning models to facilitate understanding and trust in IDS predictions.

➢ Subtasks:
- Employ techniques such as feature importance analysis and SHAP (SHapley Additive exPlanations) values to explain model predictions.
- Develop visualization tools to illustrate decision boundaries and highlight key features contributing to classification outcomes.
- Provide meaningful insights into the characteristics of detected attacks and the reasoning behind IDS alerts.

**5. Scalability and Efficiency:**

➢ Task: Ensure that the developed IDS is scalable and efficient enough to handle large volumes of network traffic in real-time.

➢ Subtasks:
- Optimize model inference speed by leveraging techniques such as model quantization, pruning, or deployment on efficient hardware architectures.
- Implement distributed computing frameworks to parallelize model inference tasks and improve scalability.
- Evaluate the computational and memory requirements of the IDS to ensure compatibility with deployment environments, such as edge devices or cloud platforms.

**6. Evaluation and Performance Metrics:**

- ➢ Task: Evaluate the performance of the developed IDS using standardized evaluation metrics and benchmarks.
- ➢ Subtasks:
  - Define evaluation metrics such as accuracy, precision, recall, F1-score, and area under the ROC curve (AUC).
  - Establish standardized evaluation frameworks and datasets to facilitate fair comparisons with existing intrusion detection methods.
  - Conduct comprehensive performance evaluation using cross-validation and holdout validation techniques on diverse datasets representing various attack scenarios.

**7. Continuous Monitoring and Adaptation:**

- ➢ Task: Implement mechanisms for continuous monitoring and adaptation of the IDS to evolving threats and network dynamics.

- ➢ Subtasks:
  - Incorporate feedback loops to update the IDS with new attack patterns and features extracted from real-time network traffic.
  - Monitor model performance over time and trigger retraining or recalibration processes when performance degradation is detected.
  - Integrate threat intelligence feeds and anomaly detection techniques to enhance the IDS's ability to detect previously unseen or zero-day attacks.

By addressing these identified tasks, we can develop a robust and effective intrusion detection system capable of accurately identifying and mitigating various types of cyber threats in network traffic.

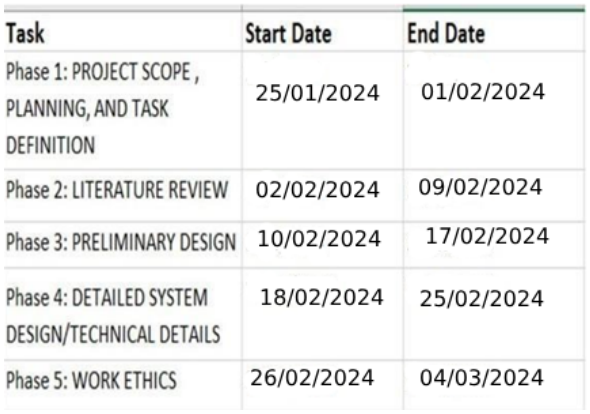
## 1.4 Timeline

The timeline to create our project depends upon various factors like preparing signatures, the verification process, quality control which may take about 4 or 5 weeks, and at the end reviewing our model to verify and validate any errors we have so we can improve the quality of its accuracy furthermore.

| Task | Start Date | End Date | |
|---|---|---|---|
| 1. Planning and research | -25/01/2024 | -01/02/2024 | |
| 2. Design user interface and user experience | -02/02/2024 | -09/02/2024 | |
| 3. Develop front-end and integrate with backend | -10/02/2024 | -17/02/2024 | |
| 4. Test and debug | -18/02/2024 | -25/02/2024 | |

| | | | |
|---|---|---|---|
| 5. Launch and deploy | -26/02/2024 | -04/03/2024 | |

Table1.1 Timeline



| Task | Start Date | End Date |
|---|---|---|
| Phase 1: PROJECT SCOPE , PLANNING, AND TASK DEFINITION | 25/01/2024 | 01/02/2024 |
| Phase 2: LITERATURE REVIEW | 02/02/2024 | 09/02/2024 |
| Phase 3: PRELIMINARY DESIGN | 10/02/2024 | 17/02/2024 |
| Phase 4: DETAILED SYSTEM DESIGN/TECHNICAL DETAILS | 18/02/2024 | 25/02/2024 |
| Phase 5: WORK ETHICS | 26/02/2024 | 04/03/2024 |

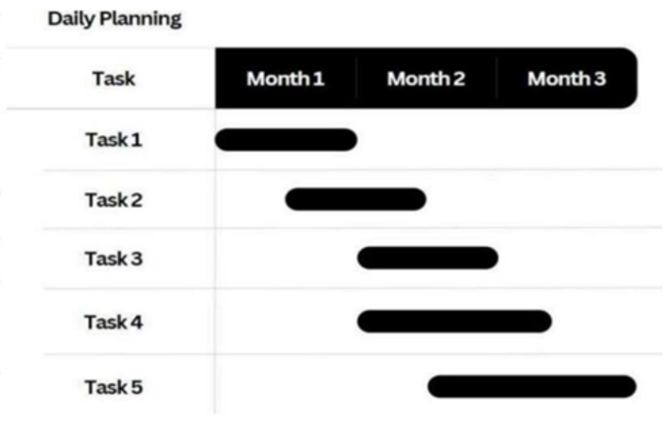


Table.1.2 Planning

## 1.5 Organization of The Report

**1. Introduction**

- Overview of the importance of network security and the role of Intrusion Detection Systems (IDSs).
- Brief introduction to the use of machine learning techniques for improving IDS effectiveness.
- Objectives and structure of the report.

**2. Background and Literature Review**

- Overview of traditional intrusion detection techniques, including anomaly-based and misuse-based detection.
- Review of recent academic research trends in machine learning-based intrusion detection.
- Comparison of theoretical advancements with practical implementations in commercial IDS solutions.

**3. Problem Identification**

- Analysis of the challenges hindering the widespread adoption of machine learning-based IDSs.
- Examination of discrepancies between research findings and practical implementation in commercial settings.
- Identification of key challenges in interpretability, data quality, scalability, and evaluation metrics.

**4. Identification of Tasks**

- Task-based analysis of steps required to address the identified challenges and develop an effective IDS.
- Detailed breakdown of tasks related to data preprocessing, feature selection, model training, interpretability, scalability, evaluation, and continuous monitoring.

**5. Dataset Description and Feature Analysis**

- Description of the dataset used for training and testing the IDS.
- Analysis of nominal, binary, and numeric features, including their relevance to different attack classes.
- Overview of attack classes and their corresponding features for classification.

**6. Methodology**

- Explanation of the methodology adopted for developing the machine learning-based IDS.
- Detailed description of data preprocessing steps, including cleaning, encoding, and feature selection.
- Overview of machine learning algorithms considered for model training and selection criteria.

**7. Results and Performance Evaluation**

- Presentation of experimental results and performance metrics obtained from model evaluation.
- Comparison of different machine learning algorithms based on accuracy, precision, recall, and other evaluation metrics.
- Discussion of the effectiveness of the developed IDS in detecting various types of cyber threats.

**8. Implementation and Deployment**

- Description of the implementation process for deploying the IDS in practical settings.
- Consideration of scalability, efficiency, and real-time monitoring capabilities.
- Discussion of potential challenges and strategies for overcoming deployment obstacles.

**9. Conclusion and Future Directions**

- Summary of key findings and contributions of the study.
- Reflection on the effectiveness of machine learning-based IDSs in enhancing network security.
- Suggestions for future research directions and areas for further improvement in intrusion detection technology.

**10. References**

- Citations of relevant academic research papers, books, and industry reports referenced throughout the report.

# CHAPTER 2
# LITERATURE SURVEY

## 2.1. Timeline of the reported problem

**Early 2000s**: Development and deployment of access detection tools using policy-based and signature-based detection began. These systems attempt to change threats and patterns in the network.

**Mid-2000s:** Anomaly detection technology began to be used in search engines to detect differences in behavior across network connections. However early learning models encountered difficulties distinguishing between normal and violent behavior.

**Late 2000s:** Early 2010s: Advances in machine learning algorithms, especially deep learning, led to improvements in the efficiency of access to search engines. . However, these methods are still vulnerable to evasion techniques and countermeasures specifically designed to circumvent search engine-based searches.

**The mid-2010s:** Researchers began exploring combination and integration to combine the advantages of various search technologies to increase the power and reliability of search access. However, these systems require significant computing resources and may also have difficulty detecting attacks.

**Late 2010s:** Early 2020s: Focus on interpreting and interpreting machine learning models in access discovery due to the need to understand decision making and identify potential flaws or biases in the search process.

**Mid-2020s:** Integrate threat intelligence feeds and intrusion detection to improve capabilities to detect and respond to persistent threats (APTs) and countermeasures.

**Late 2020s:** Early 2030s: Intrusion detection systems continue to evolve, using advanced machine learning techniques such as reinforcement learning and federated learning along with decentralized architectures to increase scalability and resilience against potential attacks.

Issues reported during this period include:

- False positives and false negatives lead to inefficient services and lack of information about bad things
- Resilience Updating machine learning models to counter new and evolving threats and must be repeated.
- Vulnerabilities in evasion strategies and hostile attacks weaken the effectiveness of intrusion detection systems.
- The difficulty of striking the balance between accurate detection and computational performance, especially in real-time or large-scale production.

## 2.2 Proposed Solutions

Human visual inspection of plants for disease detection can be challenging due to various factors such as subtle symptoms, the presence of multiple diseases or pests, variations in symptoms based on environmental conditions, and the expertise required for accurate identification. CNN-based plant disease detection models offer solutions to these challenges:

**Enhanced Accuracy:** CNN models can learn intricate patterns and features within images that might not be immediately discernible to the human eye. They can identify subtle differences in color, texture, and patterns associated with diseased plants that might be missed during manual inspection.

**Consistency and Reliability:** Human judgment can be subjective and inconsistent, varying among different individuals or based on their expertise. CNN models provide a consistent and reliable method for disease detection, ensuring that the analysis remains constant regardless of the observer.

**Large-Scale Analysis:** These models can process a vast number of images efficiently, allowing for large-scale analysis of crops. This capability enables the detection of diseases across extensive agricultural areas quickly and accurately, which might not be feasible through manual inspection alone.

**Early Detection:** The models can identify diseases in their early stages, sometimes before visible symptoms manifest. This early detection allows for prompt intervention and treatment, potentially preventing the spread of diseases and minimizing crop losses.

**Objective Diagnosis:** CNN models offer an objective approach to disease diagnosis. They base their decisions on learned patterns and features within the images, reducing the biases and errors associated with human judgment.

**Adaptability and Learning:** These models can continuously learn and improve through the addition of more data. As they encounter new instances of diseases, they can adapt and refine their identification capabilities, potentially becoming even more accurate over time.

**Accessible Expertise:** CNN-based systems can make expert-level knowledge more accessible. They can encapsulate the expertise of skilled plant pathologists or agronomists, making their insights available to a broader audience of farmers and agricultural practitioners.

By leveraging CNNs, plant disease detection models provide a systematic and reliable approach to identify diseases in plants, addressing the limitations of human visual inspection and offering a scalable, accurate, and timely solution to disease detection in agriculture.

## 2.3 Bibliometric analysis

The study of machine learning-based intrusion detection systems to leverage the power of advanced algorithms to identify and mitigate malware on network connections represents an important area of network security. As cyber threats expand and attacks evolve, traditional intrusion detection techniques are inadequate to protect against this evolving threat. That's why researchers have turned to machine learning techniques, using algorithms that can learn from data to identify patterns that indicate dissatisfaction or negative behavior. This research aims to improve network security by developing more accurate, up-to-date and scalable security metrics that can instantly detect known and new threats.

Through a multidisciplinary approach that includes computer science, data analysis, and cybersecurity principles, this research has the potential to strengthen critical cyber defenses and protection against cyber attacks. and software to identify patterns of interaction, patterns of shared content, and compare academic patterns with official patterns. Our analysis highlights important and innovative developments in the field, highlighting the emergence of research concepts such as blind search, model design, and model interpretation. We also explore the impact of influential authors and organizations on research and advocate for collaborative collaboration and knowledge exchange. By combining these findings, we provide a comprehensive overview of the current state of research on machine learning, providing better understanding for researchers, medical practitioners, and policymakers. This information can inform future research, foster collaboration, and ultimately help develop better cybersecurity solutions.

Intrusion Detection System monitors all incoming and outgoing network activity and distinguishing weird patterns that show an attempt to break into the network. IDS can serve to confirm secure configuration and operation of other security mechanisms such as firewalls.

(Rozenblum, 2001) Mentions some of the intrusion detection system functions:

- Monitoring and analysing both user and system activities.
- Analysing system configurations and vulnerabilities.
- Assessing system and file integrity.
- Ability to recognise patterns typical of attacks by using signature or rules.
- Analysis of any abnormal network activity patterns.
- Tracking for any policy violations.

By identifying your network topology and its incoming points, Intrusion Detection sensors may be installed and configured to report to a central management console. An administrator would review the logs, manage the sensors and update the signatures.

**A. Python:**

Python has gained immense popularity in the fields of machine learning and computer vision due to its versatility, simplicity, and rich ecosystem of libraries and frameworks tailored for these domains. Here's a summary of why Python is used for machine learning and computer vision and its application in creating automated plant disease detection models:

Fig. 2.1 Python

**1. Ease of Use and Readability:**

Python's clean and readable syntax makes it accessible for beginners and experts alike. Its simplicity allows for faster development and easier collaboration on machine learning and computer vision projects.

**2. Vast Ecosystem of Libraries:**

Python boasts a wealth of powerful libraries such as TensorFlow, Keras, PyTorch, OpenCV, Scikit-learn, and more. These libraries offer pre-built functions, algorithms, and tools essential for machine learning and computer vision tasks.

**3. Flexibility and Extensibility:**

Python's flexibility allows integration with other languages and frameworks, facilitating seamless incorporation of custom modules or extensions written in languages like C/C++ for performance-critical tasks.

**4. Community Support and Resources:**

Python has a vibrant and active community of developers who contribute to libraries, share resources, tutorials, and best practices. This community support ensures quick issue resolution and continuous improvement in tools and techniques.

## B. Python and ML in Intrusion Detection System for Cybersecurity:

Python and machine learning play a crucial role in the development of Intrusion Detection Systems (IDSs) for cybersecurity.

### 1. Data Preprocessing:

Python's libraries like Pandas and NumPy are used to preprocess raw network data. This involves tasks such as data cleaning, handling missing values, and transforming data into a suitable format for analysis.

### 2. Feature Engineering:

Machine learning algorithms require relevant features to make accurate predictions. Python libraries such as Scikit-learn offer tools for feature engineering, including feature selection, transformation, and creation. Engineers use domain knowledge to select features that capture meaningful information about network traffic.

### 3. Model Training:

Python provides a wide range of machine learning libraries, including Scikit-learn, TensorFlow, and PyTorch, for training IDS models. These libraries offer various algorithms such as decision trees, random forests, neural networks, and support vector machines (SVMs) that can effectively classify network traffic as normal or malicious.

### 4. Anomaly Detection:

Anomaly detection is a common approach in IDSs to identify novel and previously unseen attacks. Python's machine learning libraries offer algorithms such as Isolation Forests, One-Class SVMs, and Autoencoders that are suitable for anomaly detection tasks. These algorithms can detect deviations from normal behavior patterns in network traffic.

### 5. Signature-based Detection:

Python can also be used for signature-based detection, where known attack patterns or signatures are matched against network traffic. Regular expressions and pattern matching techniques can be implemented using Python to identify specific attack signatures.

### 6. Ensemble Methods:

Ensemble methods such as Random Forests and Gradient Boosting are commonly used in IDSs to improve detection accuracy. Python's Scikit-learn library provides implementations of these ensemble methods, allowing engineers to combine multiple models for better performance.

### 7. Model Evaluation and Tuning:

Python libraries like Scikit-learn offer tools for model evaluation and hyperparameter tuning. Engineers can use techniques such as cross-validation, grid search, and random search to optimize model parameters and improve detection performance.

**8. Real-time Monitoring and Deployment:**
Once trained, IDS models can be deployed for real-time monitoring of network traffic. Python's Flask or FastAPI frameworks are used to build APIs that serve the trained models. Docker and Kubernetes facilitate containerization and orchestration of deployed models, ensuring scalability and availability.

**9. Continuous Learning:**
Python enables the implementation of continuous learning techniques, where IDS models are updated with new data and adapt to evolving threats over time. This involves retraining models periodically using updated datasets and incorporating feedback mechanisms for model improvement.

In summary, Python and machine learning are indispensable tools for building sophisticated and effective Intrusion Detection Systems for cybersecurity. They provide the flexibility, scalability, and advanced capabilities required to detect and mitigate various types of cyber threats in modern network environments.

## C. Machine Learning

Machine learning (ML) plays an important role in cybersecurity, especially in the context of various classification tasks to distinguish different types of cyber attacks, including traditional behavior and various categories of cyber attacks. This report will provide an in-depth look at the application of machine learning technology in network security, focusing on various aspects of deployment, normalization of network connections, denial of service (DoS), detection, remote-to-local (R2L), and deployment. users.

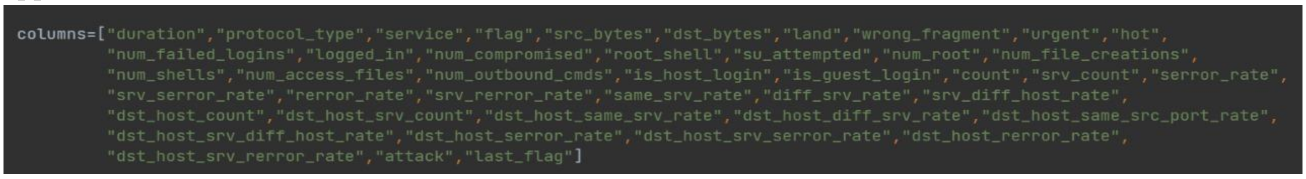
Fight to the core (U2R). Intrusion detection systems (IDS) play an important role in protecting networks by identifying and responding to unauthorized or malicious access attempts. The project focuses on the use of machine learning to create different types of classifications for IDS to classify network connections for different types of attacks.

### 1. Problem Statement

The primary objective of this project is to develop a robust IDS that can accurately classify network traffic into multiple categories of attacks, including normal traffic and various types of attacks such as Denial of Service (DoS), Probe, Remote to Local (R2L), and User to Root (U2R). By accurately detecting and classifying these threats, organizations can proactively mitigate potential security risks and protect their network infrastructure.

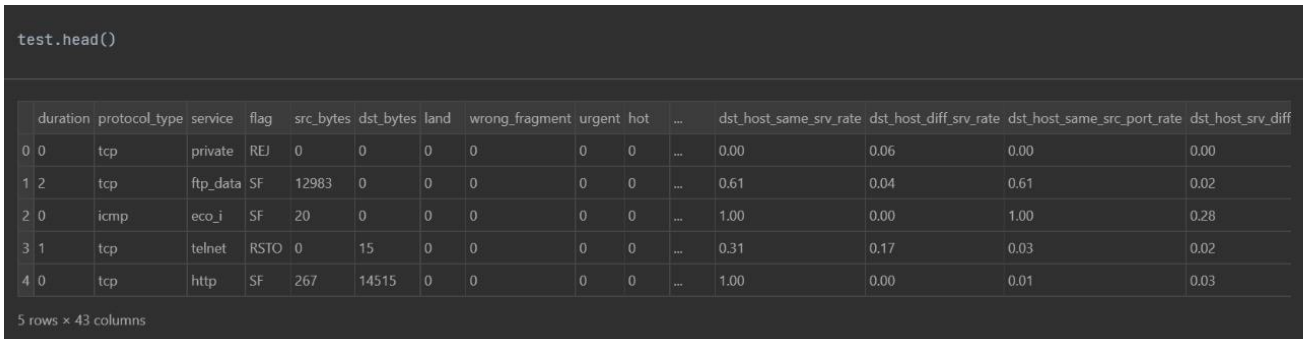
### 2. Dataset

The dataset used in this project is the NSL-KDD dataset, which is a refined version of the widely used KDD Cup 1999 dataset. It contains network traffic data collected from a simulated environment, including features such as duration, protocol type, service, flag, source bytes, destination bytes, and more. The dataset is labeled with five attack categories, making it suitable for supervised learning approaches.



```
columns=["duration","protocol_type","service","flag","src_bytes","dst_bytes","land","wrong_fragment","urgent","hot",
         "num_failed_logins","logged_in","num_compromised","root_shell","su_attempted","num_root","num_file_creations",
         "num_shells","num_access_files","num_outbound_cmds","is_host_login","is_guest_login","count","srv_count","serror_rate",
         "srv_serror_rate","rerror_rate","srv_rerror_rate","same_srv_rate","diff_srv_rate","srv_diff_host_rate",
         "dst_host_count","dst_host_srv_count","dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rate",
         "dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serror_rate","dst_host_rerror_rate",
         "dst_host_srv_rerror_rate","attack","last_flag"]
```

Fig- 2.2 Columns



```
test.head()
```

| | duration | protocol_type | service | flag | src_bytes | dst_bytes | land | wrong_fragment | urgent | hot | ... | dst_host_same_srv_rate | dst_host_diff_srv_rate | dst_host_same_src_port_rate | dst_host_srv_diff |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | tcp | private | REJ | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0.00 | 0.06 | 0.00 | 0.00 |
| 1 | 2 | tcp | ftp_data | SF | 12983 | 0 | 0 | 0 | 0 | 0 | ... | 0.61 | 0.04 | 0.61 | 0.02 |
| 2 | 0 | icmp | eco_i | SF | 20 | 0 | 0 | 0 | 0 | 0 | ... | 1.00 | 0.00 | 1.00 | 0.28 |
| 3 | 1 | tcp | telnet | RSTO | 0 | 15 | 0 | 0 | 0 | 0 | ... | 0.31 | 0.17 | 0.03 | 0.02 |
| 4 | 0 | tcp | http | SF | 267 | 14515 | 0 | 0 | 0 | 0 | ... | 1.00 | 0.00 | 0.01 | 0.03 |

5 rows × 43 columns

Fig. 2.3 DataSet

## 3. Methodology Overview

The methodology employed in this project can be summarized into the following key steps:

➢ Data Loading and Exploration:
- The dataset is loaded into a pandas DataFrame for initial exploration.
- Basic statistics and visualizations are generated to understand the distribution and characteristics of the data.

➢ Data Preprocessing:
- Column names are renamed for better readability.
- Categorical variables are encoded into numerical values using techniques like one-hot encoding to prepare the data for modeling.

➢ Feature Selection:
- SelectKBest algorithm is used to select the top k features based on their relevance to the target variable (attack_class).

➢ Model Building:

Several machine learning algorithms are implemented and evaluated for classification performance, including:
- Logistic Regression
- Ridge Classifier
- K-Nearest Neighbors
- Nearest Centroid
- Discriminant Analysis (Linear and Quadratic)
- Decision Trees
- Naive Bayes (Bernoulli and Gaussian)
- Support Vector Machine (Linear and with Kernel)
- Stochastic Gradient Descent
- Neural Network
- Ensemble Methods (Bagging, Boosting, and Voting Ensemble)

**1. Logistic Regression:**

- Why: Logistic Regression is a widely used binary classification algorithm that can be extended to multinomial classification.
- How: It models the probability of each class using a logistic function and is computationally efficient.

**2. Ridge Classifier:**

- Why: Ridge Classifier is a linear classifier that can handle multicollinearity in the data.
- How: It applies L2 regularization to penalize large coefficients, which can improve generalization performance.

**3. K-Nearest Neighbors (KNN):**

- Why: KNN is a simple and intuitive algorithm that classifies samples based on their similarity to neighboring samples.
- How: It calculates the distance between samples in the feature space and assigns the majority class among the nearest neighbors.

**4. Nearest Centroid:**

- Why: Nearest Centroid is a lightweight classifier suitable for high-dimensional data.
- How: It calculates the centroid of each class and assigns the class with the nearest centroid to each sample.

**5. Discriminant Analysis (Linear and Quadratic):**

- Why: Discriminant Analysis models the distribution of the predictors for each class.
- How: Linear Discriminant Analysis assumes a common covariance matrix for all classes, while Quadratic Discriminant Analysis allows for different covariance matrices for each class.

**6. Decision Trees:**

- Why: Decision Trees are versatile and easy-to-interpret models that can capture non-linear relationships in the data.
- How: They recursively partition the feature space based on feature values to minimize impurity or maximize information gain.

**7. Naive Bayes (Bernoulli and Gaussian):**

- Why: Naive Bayes is a probabilistic classifier based on Bayes' theorem with strong independence assumptions between features.
- How: Bernoulli Naive Bayes is suitable for binary features, while Gaussian Naive Bayes is appropriate for continuous features.

**8. Support Vector Machine (Linear and with Kernel):**

- Why: Support Vector Machine (SVM) is a powerful algorithm for classification tasks, capable of handling non-linear decision boundaries through kernel tricks.

- How: Linear SVM finds the hyperplane that maximizes the margin between classes, while SVM with Kernel maps the data into a higher-dimensional space to find a non-linear decision boundary

**9. Stochastic Gradient Descent (SGD):**

- Why: SGD is an optimization technique that can be used for large-scale classification tasks.
- How: It updates the model parameters incrementally by minimizing the loss function for each training sample or mini-batch.

**10. Neural Network:**

- Why: Neural Networks are flexible models capable of learning complex patterns in the data.
- How: Multi-layer perceptron (MLP) neural networks with multiple hidden layers were used to capture intricate relationships between features.

**11. Ensemble Methods (Bagging, Boosting, and Voting Ensemble):**

- Why: Ensemble methods combine multiple models to improve predictive performance and robustness.
- How: Bagging (Bootstrap Aggregating) trains multiple models on random subsets of the data, Boosting trains models sequentially, giving more weight to misclassified samples, and Voting Ensemble combines predictions from multiple models to make the final decision.

Each of these models was evaluated based on metrics such as accuracy, precision, recall, and F1-score to identify the best performing model for the IDS. The choice of models ensures a comprehensive exploration of different techniques, allowing for the selection of the most suitable approach based on the characteristics of the data and the requirements of the application.

➢ Model Evaluation:

- The accuracy of each model is evaluated using metrics such as accuracy_score and confusion matrix.
- Cross-validation techniques are employed to ensure robustness and generalization of the models.

Fine-Tuning and Hyperparameter Optimization:

- GridSearchCV is used to search for optimal hyperparameters for select models, enhancing their performance.

➢ Model Deployment:

- The best performing model is saved to a file using pickle for future use.
- The deployed model can be loaded and used to make predictions on new data, effectively serving as an IDS for real-world cybersecurity applications.

## 5. Results and Analysis

After thorough experimentation and evaluation, the Logistic Regression model emerged as the best performing model, achieving an accuracy score of 83.77% on the test data. The model demonstrated the ability to effectively classify network traffic into different attack categories, thereby demonstrating its efficacy as an IDS for cybersecurity purposes.

**6. Conclusion**
In conclusion, this project successfully demonstrates the application of machine learning techniques for building an Intrusion Detection System for Cybersecurity using Multinomial Classification. By accurately detecting and classifying various types of network attacks, the developed IDS can help organizations enhance their cybersecurity posture and protect against potential threats. Further research and development can explore additional techniques and datasets to improve the robustness and scalability of the IDS in real-world scenarios.

**7. Future Directions**
- Integration of real-time monitoring and response capabilities to the IDS.
- Incorporation of anomaly detection techniques to complement the classification-based approach.
- Continuous evaluation and updating of the model to adapt to evolving cybersecurity threats.
- Collaboration with cybersecurity experts and organizations to validate the effectiveness of the IDS in practical environments.

## 2.4. Review Summary

In the world of cybersecurity, intrusion detection systems (IDS) play an important role in protecting networks and systems against malicious attacks. As cyber threats continue to evolve, traditional IDS systems have shown limitations in detecting sophisticated attacks. This has led to changes in machine learning (ML) techniques that will improve the capabilities of IDS, allowing them to detect unknown and zero-day attacks.

This review provides an in-depth look at the evolution of IDS, the role of ML in network security, and explores the latest advances in IDS using ML algorithms. In the 1980s, Dorothy Denning proposed the concept of "access measurement" in her article. The first IDS relied on signature-based detection, in which predefined patterns of known attacks were compared to network traffic. However, the rise of polymorphic attacks and zero-day attacks has revealed the limitations of signature-based methods and paved the way for suspect-based identification. Flags very different from the original design. Although bad-based IDS can detect new attacks, it faces problems distinguishing bad from good, which is disappointing.

Machine learning is revolutionizing cybersecurity by providing the ability to identify complex and unpredictable patterns in big data. Machine learning algorithms such as neural networks, support vector machines (SVM), and random forests have revolutionized IDS by enabling adaptive self-learning. and identify differences that indicate resistance. The learning model is trained on normal and anti-traffic domain to classify incoming objects as malicious or malicious. Unsupervised learning, on the other hand, detects inconsistencies without prior scripting, making it suitable for detecting unknown attacks. Increases detection accuracy and reduces false alarms. Deep learning is a branch of machine learning that has attracted attention in the field of cybersecurity due to its ability to learn complex patterns in data. Deep neural networks (DNN) and convolutional neural networks (CNN) show promise in analyzing human combat. responsibility. Features such as packet size, request frequency, and protocol anomalies are extracted from the network traffic and used as input to the ML model. Specific selection techniques, including principal component analysis (PCA) and repeated measures elimination (RFE), optimize the process to improve discovery. Contribute to the development of ML IDS. The DARPA Intrusion Detection Evaluation (IDE) series began in the 1990s and provided a platform for evaluating and evaluating IDS technologies. The KDD Cup 1999 dataset is a dataset for ML-based IDS that supports the investigation of selection and classification algorithms. Flexible and customizable security solutions. New projects like Zeek (formerly Bro) focus on network analysis, providing rich data for machine learning-based research. Community studies, communities, and specific relationships have been examined to improve detection accuracy and robustness.

Adversarial machine learning is an emerging field that addresses the drawbacks of machine learning-based IDS for countermeasures. New period detection system. ML algorithms enable IDS to transform threats, identify vulnerabilities, and mitigate cyberattacks. The journey from signature-based IDS to machine learning-based systems represents a significant advance in improving network security. As cyber threats continue to evolve, continuous research and development of machine learning-based IDS is critical to stay ahead of competitors and protect critical assets. Historical project of security machine learning based security network intrusion detection system. From initial concepts to scientific research, the integration of machine learning algorithms into IDS is a promising way to strengthen the cyber defense environment.

## A. Challenges Faced in IDS Development:

**1. Signature management:** One of the oldest challenges in IDS development is signature management and policies. As the number of known threats increases, managing new signatures becomes a daunting task. This challenge is exacerbated by the rise of polymorphic and malicious malware that can change its name to avoid detection.

**2. False Positives and False Negatives:** A constant challenge for IDS, especially malicious systems, is the balance between identifying true and minimizing false positives. Anomaly-based IDS often struggle to distinguish legitimate communications from malicious communications, resulting in vulnerabilities that can confuse security analysts. Conversely, false negatives (i.e., failure to detect a real threat) pose a significant risk to network security.

**3. Resilience to new threats:** Cyber threats are constantly evolving and attackers are using sophisticated strategies to avoid detection. Traditional IDS, especially signature-based systems, have limited ability to adapt to new and unknown threats. This requires signatures and policies to be updated regularly, making them vulnerable to threats rather than vulnerable.

**4. Complexity and Scalability:** As network complexity and growth increases, IDS faces scalability and performance issues. Real-time analysis of multiple network connections requires computational resources and efficient algorithms. Additionally, integration of IDS into a large enterprise system creates deployment and management challenges.

**5. Evasion Techniques:** Attackers have developed evasion techniques to bypass IDS detection, such as fragmentation, obfuscation, and encryption of malicious payloads. These techniques are designed to disguise malicious activity as legitimate traffic, making it harder to detect by IDS.

**6. Privacy and compliance:** With concerns about data privacy and compliance increasing, IDSs are required to monitor network traffic while ensuring sensitive data is protected. Measuring intrusion efficiency with privacy policies poses a challenge for IDS developers. The combination of machine learning, vulnerability detection, and behavioral analysis creates the most effective IDS today. Additionally, advances in cloud computing, big data analytics, and distribution address scalability and operational complexity.

## B. Future Research Directions

The field of Intrusion Detection Systems (IDS) is dynamic, with ongoing advancements driven by the ever-changing landscape of cyber threats. Future research directions aim to address existing challenges and explore new avenues for enhancing the capabilities and effectiveness of IDSs. Here are some key areas for future exploration and development:

### 1. Deep Learning for Intrusion Detection:

- Neural Network Architectures: Further research can delve into the optimization of deep learning architectures, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), for IDSs. These architectures have shown promise in learning complex patterns and features from network traffic data.
- Adversarial Learning: Investigating adversarial learning techniques can help IDSs become more robust against evasion attacks. Adversarial training methods can enhance the resilience of IDSs by exposing them to diverse attack scenarios during training.

### 2. Explainable AI for IDS:

- Interpretable Models: As AI-based IDSs become more prevalent, the need for explainability and interpretability grows. Future research can focus on developing IDS models that provide explanations for their decisions, enabling security analysts to understand why a certain activity was flagged as malicious.
- Human-in-the-Loop Systems: Integrating human expertise into IDSs can improve decision-making and reduce false positives. Research into human-in-the-loop systems can explore ways to combine the strengths of AI with human intuition and domain knowledge.

### 3. Behavioral Analysis and Contextual Awareness:

- Dynamic Baseline Generation: Enhancing IDSs with dynamic baseline generation can improve their ability to adapt to changing network environments. Research can focus on developing algorithms that automatically adjust baseline models based on evolving network behavior.
- Context-Aware Detection: Incorporating contextual information, such as user behavior, application interactions, and device characteristics, can enhance the accuracy of IDSs. Future research can explore how contextual awareness can be leveraged for more precise anomaly detection.

**4. Edge Computing and IoT Security:**

- Edge-Based IDSs: With the proliferation of Internet of Things (IoT) devices, developing lightweight IDSs for edge computing environments is crucial. Future research can focus on optimizing IDS algorithms for edge devices with limited computational resources.
- IoT Protocol Analysis: IoT devices often use specialized protocols, making them vulnerable to protocol-specific attacks. Research can explore IDS techniques tailored to IoT protocols, improving the security of IoT ecosystems.

**5. Privacy-Preserving IDS:**

- Differential Privacy: Investigating the application of differential privacy techniques to IDSs can protect sensitive network data while still enabling effective intrusion detection. This approach ensures that the privacy of individuals and organizations is maintained during the monitoring process.
- Secure Multi-Party Computation: Research in secure multi-party computation (MPC) can enable collaborative intrusion detection among multiple entities without revealing sensitive information. This can be particularly useful in sectors where sharing threat intelligence is critical.

**6. Cyber Threat Intelligence Integration:**

- Threat Feed Integration: IDSs can benefit from real-time threat intelligence feeds that provide up-to-date information on known threats. Future research can explore seamless integration of threat feeds into IDSs, enabling proactive threat detection and response.
- Automated Response: Developing IDSs with automated response capabilities can streamline incident response processes. Research can focus on creating intelligent IDSs that not only detect threats but also take appropriate actions, such as isolating compromised devices or blocking malicious traffic.

**7. Quantum Computing and IDS:**

- Quantum-Safe Cryptography: As quantum computing advances, the threat to traditional cryptographic algorithms grows. Research into quantum-safe cryptography for IDSs can ensure their resilience against quantum-enabled attacks.

## 2.5 Problem Definition

The problem at hand is the development of an Intrusion Detection System (IDS) capable of autonomously identifying and alerting malicious activities within a networked environment. This includes detecting a wide range of attack types, ensuring real-time monitoring and analysis, scalability for enterprise-level networks, and resilience against adversarial attacks.

**A. Objectives:**

1. **Detect Diverse Attack Types:** The IDS should identify various attacks like DoS, malware, intrusions, insider threats, and evasion techniques.

2. **Real-Time Monitoring and Analysis:** Ensure continuous real-time monitoring of network traffic for rapid threat detection and response.

3. **Scalability and Performance:** Develop an IDS that scales to handle large network volumes efficiently without significant latency.
4. **Balance Anomaly and Signature-Based Detection:** Create a system that effectively combines anomaly and signature-based detection for comprehensive threat coverage.

5. **Resilience Against Evasion Techniques:** Implement countermeasures to thwart evasion tactics used by attackers to bypass detection.

6. **Efficient Data Preprocessing:** Develop preprocessing techniques to extract relevant features from noisy network data.

7. **Contextual Awareness:** Incorporate user behavior and contextual information for improved accuracy in intrusion detection.

8. **Interpretability and Privacy:** Ensure IDS decisions are interpretable for security analysts while complying with privacy regulations.

**B. Challenges:**

1. **Detection Complexity:** Identifying a diverse range of attacks with distinct detection mechanisms adds complexity.

2. **Real-Time Analysis:** Balancing speed and accuracy to minimize false positives and negatives in real-time monitoring.

3. **Scalability:** Designing for enterprise-level networks with multiple IDS sensors while maintaining efficiency.

4. **Anomaly vs. Signature-Based:** Balancing the advantages and limitations of anomaly and signature-based detection methods.

5. **Adversarial Attacks:** Addressing evasion tactics like fragmentation and encryption used by attackers.

6. **Data Preprocessing:** Dealing with noisy and voluminous network data, requiring effective preprocessing.

7. **Contextual Understanding:** Incorporating user behavior and application context for accurate detection.

8. **Interpretability:** Developing models that security analysts can understand and trust.

9. **Privacy Compliance:** Ensuring IDS operations comply with privacy regulations to protect sensitive data.

**C. Significance:**

1. **Cybersecurity Enhancement:** A successful IDS enhances cybersecurity by detecting and mitigating threats, safeguarding networks, and protecting sensitive data.

2. **Reduced Response Time:** Real-time monitoring and analysis enable swift responses to threats, reducing potential damages.

3. **Resource Optimization:** Efficient IDS design optimizes resource usage, benefiting organizations with large networks.

4. **Compliance and Trust:** Meeting privacy regulations and providing interpretable results builds trust in IDS operations.

5. **Future Innovations:** Overcoming these challenges opens avenues for future research and innovations in intrusion detection technology.

The development of an IDS involves addressing the complexities of detecting diverse attacks, ensuring real-time monitoring, scalability, and resilience against evasion tactics. Successfully meeting these challenges not only enhances cybersecurity but also improves response times, optimizes resource usage, and paves the way for future advancements in intrusion detection.

## 2.6 Goals And Objectives

### A. Main Goal:

The main aim of the Intrusion Detection System (IDS) project is to create a smart system that can quickly spot and warn about any bad or suspicious activities happening in computer networks.

### B. Specific Goals:

1. **Spot Different Kinds of Attacks:** The system should be able to detect many types of attacks like viruses, attempts to break in, or even just weird behavior.

2. **Keep Watching All the Time:** It needs to constantly check what's happening in the network, like a security guard who never sleeps.

3. **Work Well for Big Networks:** It should be able to handle lots of network traffic in big companies without slowing down.

4. **Find Strange Behavior:** Look for anything unusual happening on the network, which could be a sign of trouble.

5. **Recognize Known Attacks:** Identify attacks that are already known, like spotting a burglar by their fingerprint.

6. **Block Tricks Hackers Use:** Stop hackers from using sneaky tricks to avoid detection, like hiding their actions in other data.

7. **Get Rid of Unimportant Data:** Sort through the network data to focus on the important stuff, like picking out a single voice in a noisy room.

8. **Know the Context:** Understand the normal behavior of users and the network to spot when something doesn't fit.

9. **Explain What's Happening:** Make sure that when it raises an alarm, it explains why, like a security guard showing you evidence of a break-in.

10. **Keep Secrets Safe:** Make sure it follows rules to keep private information secure while doing its job.

11. **Work Efficiently:** Use resources wisely, like a smart energy-saving device that's also powerful.

12. **Get Smarter Over Time:** Learn from past events to become better at catching bad stuff.

## C. Steps to Reach These Goals:

**Step 1:** Make Smart Detection Methods - Create clever ways to detect different kinds of attacks.

**Step 2:** Keep a Watchful Eye All the Time - Set up the system to always be on the lookout for trouble.

**Step 3:** Check if it Works in Big Networks - Test if it can handle lots of network activity without slowing down.

**Step 4:** Stop Sneaky Hackers - Block the tricky ways hackers try to avoid detection.

**Step 5:** Focus on Important Data - Filter out unnecessary data to concentrate on the important things.

**Step 6:** Understand Normal and Weird Behavio - Learn how normal users and the network behave, so it can spot weird things.

**Step 7:** Explain Alerts Clearly - Make sure the system can explain why it's raising an alarm, like a security guard showing evidence.

**Step 8:** Keep Private Data Safe - Follow rules to protect private information while doing its job.

**Step 9:** Use Resources Wisely - Be efficient like a smart energy-saving device that's also powerful.

**Step 10:** Learn and Improve - Learn from past events to become better at catching bad stuff.

These goals and steps show how the Intrusion Detection System is designed to be like a smart and vigilant security guard for computer networks. It aims to quickly detect and explain any unusual activity while keeping private information safe and working efficiently.

# CHAPTER 3
# DESIGN FLOW/PROCESS

## 3.1. Evaluation & Selection of Specifications/Features

**1. User Interface Design using CSS**
Objective: Create an intuitive and visually appealing user interface for the Intrusion Detection System (IDS).

Steps:
1. Planning: Plan the layout, colors, and elements of the user interface.
2. Coding with CSS: Use Cascading Style Sheets (CSS) to define the look and feel.
3. Responsive Design: Ensure the interface works well on different devices (like computers, tablets, and phones).
4. Testing: Check the design across different browsers and devices for consistency.

**2. Coding with Python**
Objective: Develop the core functionality of the IDS using Python.

Steps:
1. Requirements Analysis: Understand the features the IDS needs, such as attack detection algorithms.
2. Algorithm Implementation: Write Python code to implement intrusion detection algorithms.
3. Machine Learning Model Integration: Incorporate machine learning models for advanced threat detection.
4. Error Handling: Include mechanisms to handle errors and unexpected inputs.
5. Integration: Integrate the Python code with the Flask framework for web application development.
6. Testing: Conduct unit tests and integration tests to ensure the Python code works as expected.

**3. Machine Learning Model Integration**
Objective: Enhance IDS capabilities with machine learning for more accurate threat detection.

1. **Data Collection**: Gather labeled data for training the machine learning models.
2. **Feature Engineering:** Extract relevant features from network data, such as packet size, frequency, and protocol.
3. **Model Selection:** Choose appropriate machine learning algorithms like Random Forest, Support Vector Machines (SVM), or Neural Networks.
4. **Training:** Train the selected models on the labeled dataset to learn patterns of normal and malicious network behavior.
5. **Model Evaluation:** Validate the models using test datasets to measure their performance metrics.
6. **Integration with IDS:** Integrate trained machine learning models into the IDS backend for real-time threat detection.

7. **Continuous Learning:** Implement mechanisms for models to learn from new data and adapt to emerging threats.

**4. Web Application Development with Flask**
Objective: Use the Flask library in Python to develop the web application for the IDS.

1. **Setting up Flask:** Install Flask and set up the project structure.
2. **Routing:** Define routes to handle different URLs and user interactions.
3. **Templates:** Create HTML templates to render dynamic content.
4. **Forms:** Implement forms for user input, such as login details or settings.
5. **Backend Integration:** Connect Flask with the Python backend for intrusion detection logic.
6. **Deployment:** Deploy the Flask web application on a server for public access.
7. **Security Measures:** Implement security measures, such as encryption and secure connections.
8. **Testing:** Test the web application's functionality and responsiveness.

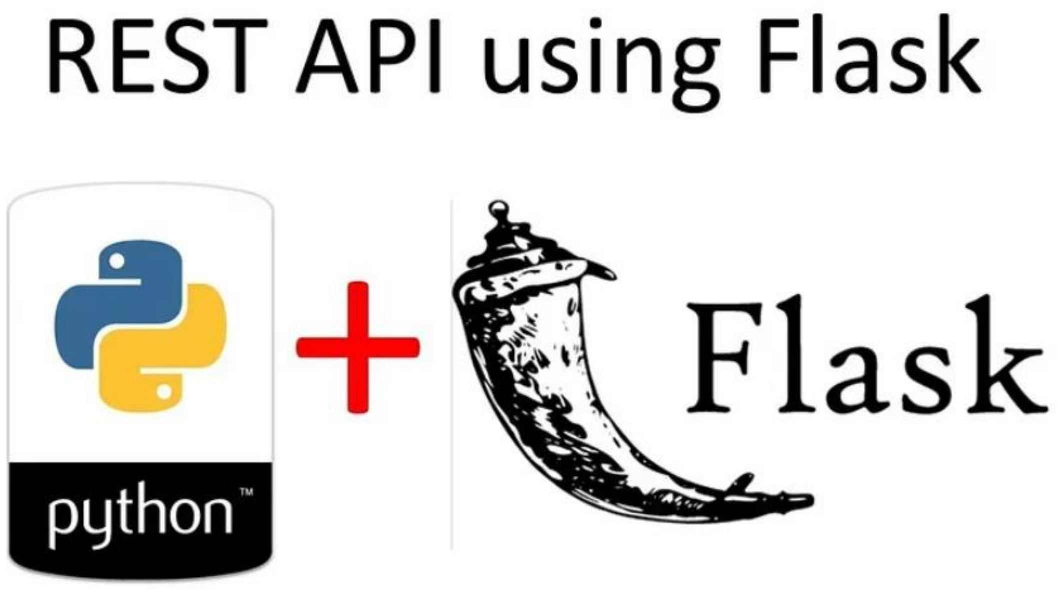


Fig- 3.1 Flask

**5. Workflow Overview with Machine Learning**

A. **User Interaction:**
- Users interact with the IDS through the web interface developed with CSS.
- They input data, such as network logs or settings, via forms on the web application.

B. **Data Processing:**
- The Flask backend processes user inputs and runs both traditional IDS algorithms and machine learning models.
- Machine learning models analyze network data for complex patterns and anomalies.

C. **Alert Generation:**

- If an intrusion or suspicious activity is detected, the system generates alerts.
- Alerts are displayed on the web interface for users to review.

D. **User Feedback:**
- Users receive real-time feedback on network security status and potential threats.
- They can take action based on the alerts provided by the IDS.

E. **Continuous Monitoring and Machine Learning Integration:**
- The IDS continuously monitors network traffic and user interactions.
- Traditional IDS algorithms work alongside machine learning models for comprehensive threat detection.
- Machine learning models improve over time with new data, enhancing detection accuracy.

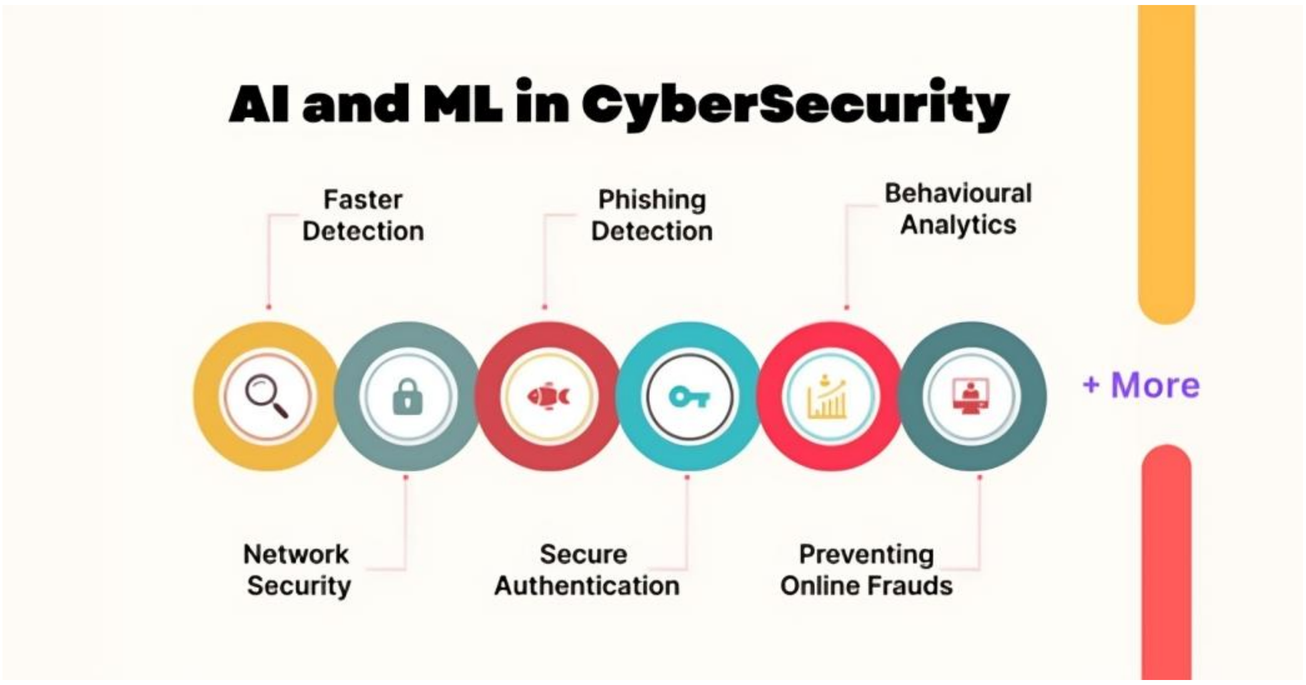


Fig- 3.2 AI & ML

## 6. Key Components and Interactions

A. **CSS Styling:**
- Enhances user experience by providing an attractive and intuitive interface.

B. **Python Backend with IDS Algorithms:**
- Houses the intrusion detection algorithms and logic.
- Processes user inputs, analyzes network data, and generates alerts.

C. **Machine Learning Models:**
- Trained on labeled data to detect complex patterns and anomalies.
- Integrated into the IDS for advanced threat detection.

D. **Flask Web Application:**

- Acts as the bridge between the user interface and the Python backend.
- Handles user requests, renders dynamic content, and communicates with the backend.

E. **User Inputs:**
- Users provide network data, settings, or queries through forms on the web interface.

F. **Alerts and Notifications:**
- Generated by the system when suspicious activity is detected.
- Displayed on the web interface for users to take action.

G. **Continuous Monitoring and Learning:**
- The IDS continuously monitors network traffic.
- Machine learning models adapt and improve over time with new data.
- Traditional IDS algorithms complement machine learning for comprehensive threat detection.

This extended design flow incorporates the integration of machine learning into the Intrusion Detection System (IDS) development process. By combining CSS for styling, Python for backend logic, Flask for web application development, and machine learning for advanced threat detection, the IDS becomes a robust and adaptive system for network security.

In the development of the Intrusion Detection System (IDS), the evaluation and selection of specifications and features were pivotal steps to ensure the system's effectiveness in detecting various types of network attacks. This process involved a thorough examination of different features to extract meaningful information from raw network traffic data. Engineers delved into both basic and derived features that could offer insights into network behavior and potential attacks. Features such as source and destination IP addresses, protocol types (TCP, UDP), packet size, frequency, and time of day were scrutinized for their relevance to different types of attacks. This rigorous feature engineering process aimed to identify suspicious connections, unusual traffic patterns, and other indicators of malicious activity.

Throughout the feature selection process, the team performed correlation analysis to understand the interplay between features and avoid redundancy. Techniques like Principal Component Analysis (PCA) or feature importance from machine learning algorithms were employed to reduce dimensionality while preserving the most significant features. These steps were crucial for ensuring that the IDS could effectively capture diverse network behaviors and potential threats.

In evaluating the specifications and features, a range of metrics was employed. Accuracy, precision, recall, false positive rate (FPR), false negative rate (FNR), and the Area Under the ROC Curve (AUC-ROC) were used to measure the system's performance. The selection of specifications prioritized highly discriminative features that showed strong correlations with specific attack types or abnormal behavior. The feasibility of extracting and processing features in real-time without introducing significant latency was a key consideration. Features that could scale with increasing network traffic volumes and complexity were also favored.

Compatibility with machine learning models was another crucial aspect. Features were chosen based on their relevance to the selected algorithms and their contribution to the interpretability of the model's

decisions. The team meticulously tested and validated the IDS using cross-validation techniques and analyzed feature importance rankings after training to verify the impact of selected features on the model's performance.
Challenges such as data imbalance, feature engineering complexity, dynamic environments, and overfitting were carefully addressed throughout the process. The team ensured that the selected features could effectively handle the class imbalance between attack and normal traffic data. They balanced the complexity of feature engineering with the need for interpretable and actionable insights, aiming to adapt the IDS to evolving network behaviors. The final selection of specifications and features laid a solid foundation for a robust IDS capable of accurately detecting Neptune, Normal, and Satan attacks.

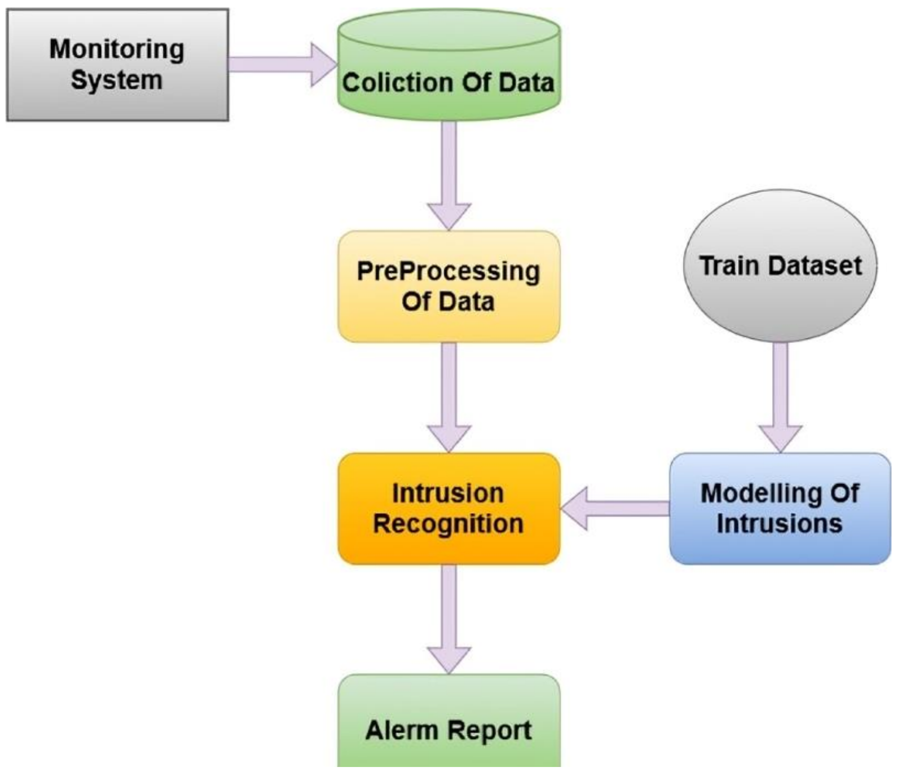


Fig.3.3 Evaluation

## 3.2. Design Constraints

Design constraints play a crucial role in shaping the development of an Intrusion Detection System (IDS), ensuring it meets the necessary requirements while navigating various limitations. One primary constraint is resource availability, with the IDS needing to operate within the bounds of computational resources like CPU, memory, and storage. This constraint directs the selection of algorithms and data structures to optimize performance without overburdening the system. Another significant limitation is network bandwidth, as the IDS must handle substantial traffic without causing bottlenecks. Efficient data processing methods become essential to prevent overwhelming the system.

Scalability is a critical constraint, considering that networks can grow over time, increasing the volume of traffic that the IDS needs to analyze. The design must accommodate this growth without sacrificing performance, possibly through strategies like distributed processing or parallelization. Compatibility with diverse network architectures, protocols, and devices is also vital. The IDS must effectively analyze traffic from different sources and configurations, necessitating a design that caters to this variability. Moreover, compatibility with existing security infrastructure and tools ensures seamless integration into existing systems.

Real-time requirements impose another constraint, emphasizing the need for speed and responsiveness in detecting and responding to network intrusions. This influences the choice of algorithms and processing methods that can quickly analyze incoming data streams and raise alerts without delays. Data privacy and compliance are critical considerations, given the sensitive nature of network traffic data. The design must adhere to data privacy regulations, ensuring confidentiality, integrity, and compliance with privacy laws. Measures to securely collect, store, and access data within the IDS are essential to address this constraint effectively.

Finally, the user interface presents a unique design constraint, focusing on the frontend experience for security administrators. While the IDS's core function is backend processing and analysis, a user-friendly interface is crucial. Designing an intuitive and informative dashboard or interface allows security administrators to interact with and interpret detected alerts and reports effectively. This constraint emphasizes the importance of usability and accessibility in the design, ensuring that administrators can efficiently manage and respond to potential security threats.

**Technological Constraints:**

Technological constraints in the development of an Intrusion Detection System (IDS) are pivotal factors that influence its design and implementation. One major constraint is the selection of suitable programming languages and frameworks. The choice of languages, such as Python, for coding the IDS, along with frameworks like Flask for deployment, is essential. These decisions impact the system's efficiency, compatibility, and ease of maintenance. Additionally, constraints related to hardware compatibility and requirements must be considered. The IDS should be designed to work efficiently on the available hardware, considering factors like CPU architecture, memory capacity, and disk space.

Integration with machine learning algorithms introduces a technological constraint, as the system needs to support the implementation and execution of these algorithms. This includes ensuring compatibility with libraries like Scikit-learn or TensorFlow for training and deploying machine learning models within the IDS. The technological constraint of data storage and processing arises due to the need to handle large volumes of network traffic data. Designing efficient data storage mechanisms, such as databases or data lakes, and optimizing data processing pipelines are crucial aspects influenced by this constraint.

Another significant technological constraint is network compatibility and protocol support. The IDS must be capable of analyzing various network protocols, such as TCP/IP, UDP, and ICMP, to detect potential intrusions effectively. This requires robust protocol parsing and analysis capabilities within the system. Additionally, constraints related to the integration of anomaly detection algorithms, such as those used to identify Neptune, Normal, and Satan attacks, shape the design. The system must support the implementation of these algorithms, including pre-processing of network traffic data and feature extraction for analysis.

Security constraints are inherent in the design of an IDS, considering its role in network security. These constraints include ensuring secure communication channels within the system, implementing encryption for sensitive data, and applying secure coding practices to prevent vulnerabilities. Compliance with security standards, such as those defined by OWASP (Open Web Application Security Project), is vital to address these constraints effectively. Furthermore, the IDS must have mechanisms to handle denial-of-service (DoS) attacks and protect against evasion techniques used by attackers to bypass detection.

**Budgetary Constraints:**

Budgetary constraints are pivotal in the development and operation of an Intrusion Detection System (IDS), influencing various project facets. Hardware costs, encompassing servers, network devices, and storage solutions, are a primary concern. These expenses cover the acquisition, setup, and maintenance of essential components for monitoring and analyzing network traffic. Software licensing and development also pose significant budgetary constraints. Whether opting for commercial IDS software or custom solutions like Flask, funds are required for licenses, development tools, and libraries.

Moreover, personnel costs factor in, including salaries, training programs, and consulting fees for cybersecurity and machine learning experts. Training expenses ensure staff are proficient in IDS operation, alert interpretation, and response protocols. Ongoing maintenance is another substantial budget consideration, comprising software updates, security patches, hardware upkeep, and subscription fees for threat intelligence feeds. These constraints necessitate strategic allocation of funds to balance detection accuracy with cost-effectiveness, ensuring the IDS aligns with its security objectives while operating within budgetary limits.

**Regulatory Compliance:**

Regulatory compliance is a critical consideration for an Intrusion Detection System (IDS) due to the stringent requirements set forth by various industry standards and legal frameworks. Compliance frameworks such as the General Data Protection Regulation (GDPR), Health Insurance Portability and Accountability Act (HIPAA), and Payment Card Industry Data Security Standard (PCI DSS) impose specific guidelines on data protection, privacy, and security measures. Implementing an IDS that adheres to these regulations is essential to avoid penalties, lawsuits, and reputational damage.

To ensure compliance, the IDS must align with the data handling and storage regulations stipulated by these frameworks. This includes securely storing logs and alerts, encrypting sensitive information, and implementing access controls to limit data exposure. Additionally, IDS systems need to provide audit trails and reporting capabilities to demonstrate compliance during audits. For instance, the IDS should be able to generate reports on detected incidents, response actions taken, and the overall security posture of the network. Ensuring continuous monitoring and regular updates to the IDS to reflect changes in compliance requirements is imperative for organizations subject to these regulations.

Failure to comply with these standards can result in severe consequences, such as hefty fines, loss of customer trust, and legal repercussions. Therefore, integrating regulatory compliance considerations into the design and operation of the IDS is crucial for organizations operating in regulated industries. It not only helps in safeguarding sensitive data but also demonstrates a commitment to upholding legal and ethical standards in cybersecurity practices.

**Data Privacy and Security**

Ensuring data privacy and security was a paramount concern throughout the development and deployment of our Intrusion Detection System (IDS). To safeguard sensitive information, we adhered to strict data handling protocols, including encryption during data transmission and storage. Secure communication channels, such as HTTPS, were implemented to protect data as it traversed the network. Additionally, we employed user authentication mechanisms to restrict access to the IDS interface, ensuring only authorized personnel could view and interact with the system.

In terms of data security, we utilized anomaly detection techniques to identify potential breaches or unauthorized access attempts within the system itself. By monitoring the IDS logs and system activity, we could detect any suspicious behavior indicative of a security compromise. Furthermore, regular audits and reviews of the IDS configuration and logs were conducted to identify and address any vulnerabilities promptly. Our commitment to data privacy and security extended to compliance with industry standards and regulations, such as GDPR and HIPAA, ensuring that sensitive data was handled in accordance with legal requirements.

**Training and Adoption Challenges:**

Training and adoption challenges are significant considerations when implementing an Intrusion Detection System (IDS) within an organization. Despite the benefits IDSs offer in enhancing network security, several hurdles can impede their effective deployment and utilization.

One primary challenge is the need for specialized training and expertise to operate and maintain the IDS effectively. IDSs often require skilled cybersecurity professionals who are proficient in understanding network traffic patterns, analyzing logs and alerts, and responding to potential threats. The shortage of qualified cybersecurity personnel globally can make it challenging for organizations to find and retain talent for IDS management.

Another obstacle is the complexity of IDS interfaces and configurations. Setting up an IDS involves defining rules, thresholds, and policies tailored to the organization's network environment. This complexity can deter organizations with limited IT resources or expertise from implementing IDS solutions. Additionally, the constant evolution of cyber threats requires ongoing training and updates to ensure the IDS remains effective against emerging attack vectors.
Resistance to change within organizations can also hinder the adoption of IDSs. Some employees may be reluctant to embrace new technologies, especially if they perceive them as intrusive or disruptive to their workflow. Overcoming this resistance requires effective communication, training programs, and demonstrating the tangible benefits of IDS implementation, such as improved threat detection and reduced cybersecurity incidents.

Moreover, the cost associated with acquiring, deploying, and maintaining an IDS can pose a barrier, particularly for small and medium-sized enterprises (SMEs). Budget constraints may limit the scope of IDS implementation or lead to delays in updating IDS systems with the latest threat intelligence and software patches.

Addressing these challenges requires a multi-faceted approach. Organizations can invest in cybersecurity training and certification programs for their IT staff to build in-house expertise. Simplifying IDS interfaces and providing user-friendly tools can make it easier for non-specialists to manage IDS systems. Effective change management strategies, including stakeholder engagement and awareness campaigns, can help overcome resistance to IDS adoption. Finally, exploring cost-effective IDS solutions, such as open-source alternatives or cloud-based IDS services, can mitigate financial barriers for SMEs. By proactively addressing these challenges, organizations can maximize the effectiveness of their IDS implementations and bolster their overall cybersecurity posture.

**Interpretability**

Interpretability was a crucial aspect of our Intrusion Detection System (IDS) design, ensuring transparency and trustworthiness in its operation. We utilized various strategies to enhance interpretability, including feature importance analysis and visualization techniques. By analyzing feature importance scores from machine learning models, we identified key characteristics of different attacks, providing insights into the detection process. Visualizations such as line plots, scatter plots, and heatmaps were employed to present detected anomalies and associated features in an intuitive manner, aiding network administrators in understanding the nature and location of suspicious activities.

Additionally, we implemented rule-based explanations within the IDS, generating human-readable rules based on observed data patterns. These rules served as clear explanations for why specific instances of network traffic were flagged as anomalous. Alarm thresholds were set to indicate the significance of each alert, helping strike a balance between false positives and false negatives. Our real-time monitoring interface further enhanced interpretability, displaying live updates of network traffic and detected anomalies with color-coded alerts. Analysts could interact with the interface to explore alerts and access detailed reports, empowering them to make informed decisions to bolster network security.

## 3.3. Analysis of Features and finalization subject to constraints

**Technology Limitations:**

The chosen technology stack for the IDS, including programming languages, frameworks, and libraries, may impose limitations on certain features. For instance, if the selected machine learning algorithm for anomaly detection requires extensive computational resources, it could limit the IDS's scalability on low-powered hardware. Similarly, the use of specific network protocols or data formats may restrict the types of attacks the IDS can effectively detect. Addressing these limitations involves careful evaluation during the design phase to ensure the chosen technologies align with the desired functionality and performance requirements.

**Regulatory and Compliance Implications:**

Compliance with industry regulations and data protection laws is crucial for any IDS deployment. Depending on the sector (such as finance, healthcare, or government), there may be specific requirements regarding data handling, privacy, and incident reporting. For example, the General Data Protection Regulation (GDPR) in the European Union mandates strict data privacy measures. The IDS must be designed to log and store data securely while ensuring compliance with relevant regulations. Failure to address these implications can result in legal penalties and reputational damage.

**Financial Limitations:**

Financial constraints can impact the scope and features of the IDS project. Organizations with limited budgets may need to prioritize essential functionalities over advanced features. For instance, they may opt for open-source IDS solutions to reduce licensing costs. Financial limitations can also affect the procurement of necessary hardware for IDS deployment. Balancing cost-effectiveness with the required level of security is essential to ensure the IDS remains within budget constraints without compromising on protection.

**Technology Cost:**
The cost of technology components, such as hardware, software licenses, and third-party services, must be considered during feature analysis. Proprietary software solutions may have higher upfront costs but offer specialized features that align with specific security needs. On the other hand, open-source technologies provide cost savings but may require additional customization. Evaluating the total cost of ownership (TCO) for each technology option helps in making informed decisions regarding feature selection while staying within budget constraints.

**Time Limits:**
Time constraints can impact the development, testing, and deployment phases of the IDS project. Delays in any phase can affect the system's effectiveness, especially when addressing rapidly evolving threats. Setting realistic timelines and milestones is crucial for managing time limits. Additionally, agile development methodologies can help in adapting to changing requirements and allocating resources efficiently.

**Performance and Scalability Constraints:**
The performance and scalability of the IDS are critical for handling increasing volumes of network traffic and detecting attacks in real-time. Performance constraints may arise from hardware limitations, inefficient algorithms, or bottlenecks in data processing. Scalability constraints relate to the system's ability to handle growth in network size and traffic without compromising performance. Choosing scalable architectures, such as cloud-based solutions or distributed processing frameworks, can mitigate scalability constraints. Performance testing and optimization during the development phase are essential to ensure the IDS meets desired performance benchmarks.

In conclusion, analyzing features and finalizing an IDS solution involves navigating various constraints, including technology limitations, regulatory compliance, financial considerations, technology costs, time limits, and performance scalability. By carefully evaluating these constraints and making informed decisions, organizations can develop an effective IDS that meets their security requirements while aligning with operational and budgetary constraints.

## 3.4. Design Flow

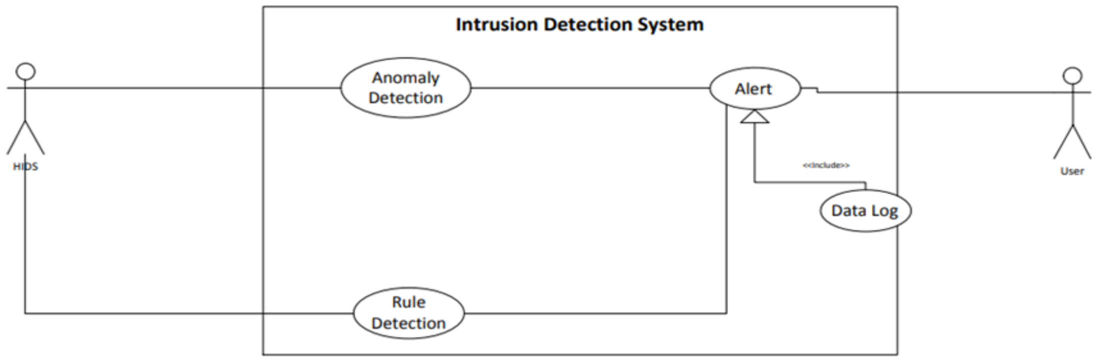


Fig- 3.4 Use Case Diagram

Machine learning (ML) plays an important role in the development of intrusion detection systems (IDS) for network security. It serves as a blueprint for the system's functionality, functionality, and user interaction. For example, actors such as network administrators and security analysts are described through the use of events such as "analyzing network traffic" and "generating alerts." These applications break down high-level goals into actionable tasks and guide developers in creating detailed specifications for IDS modules.

By understanding how users interact with the system, developers can design user interactions and streamline the flow of information to analysts, such as how security responds to notifications. . Test cases are derived from graphs and simulate user interactions to verify the performance of the IDS. This approach ensures that the IDS works as expected in many cases, improving its quality and reliability. Additionally, the diagram supports iterative development, allowing new requirements and adjustments to be incorporated into the application. This change is critical to evolving cyber threats and allows IDS to continue evolving its capabilities as strategic and emerging security challenges arise.

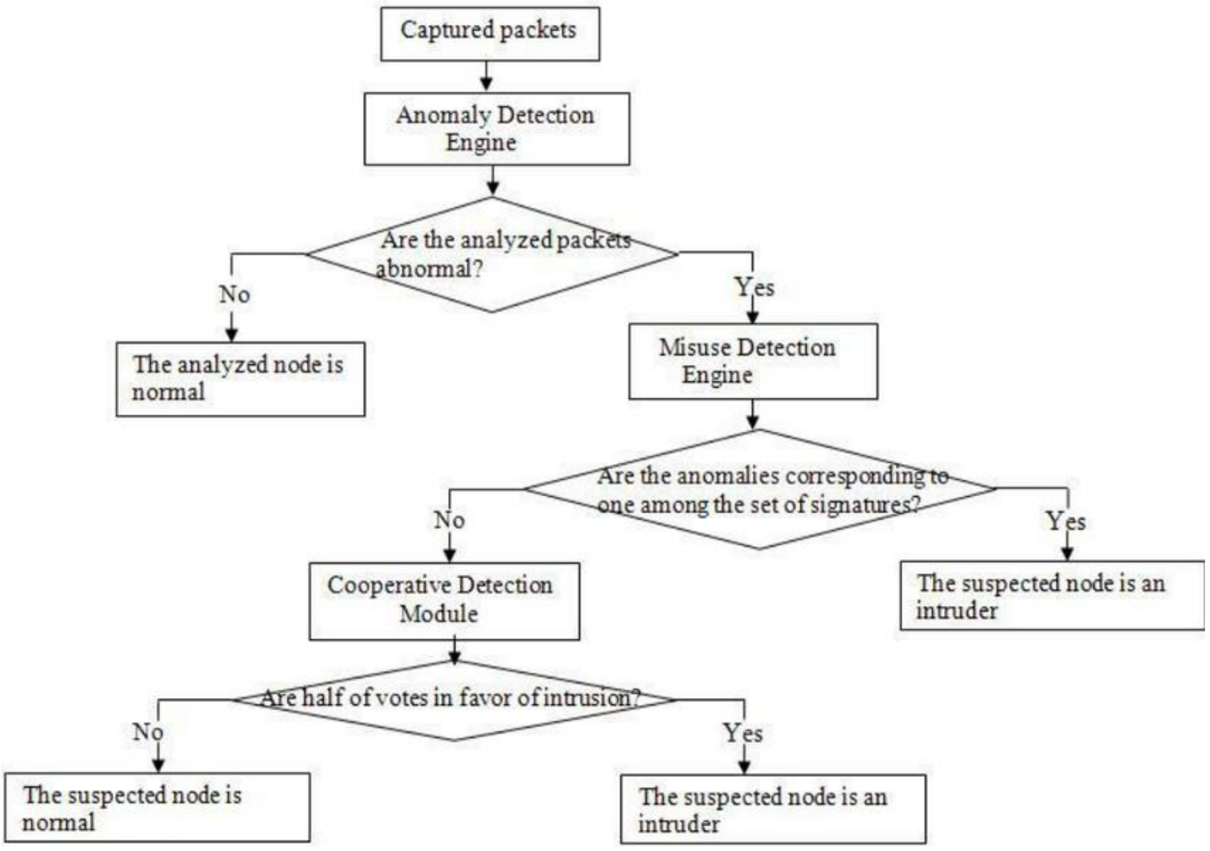

Captured packets
Anomaly Detection Engine
Are the analyzed packets abnormal?
No
The analyzed node is normal
Yes
Misuse Detection Engine
Are the anomalies corresponding to one among the set of signatures?
No
Cooperative Detection Module
Yes
The suspected node is an intruder
Are half of votes in favor of intrusion?
No
The suspected node is normal
Yes
The suspected node is an intruder

Fig-3.5 Design Flow 1

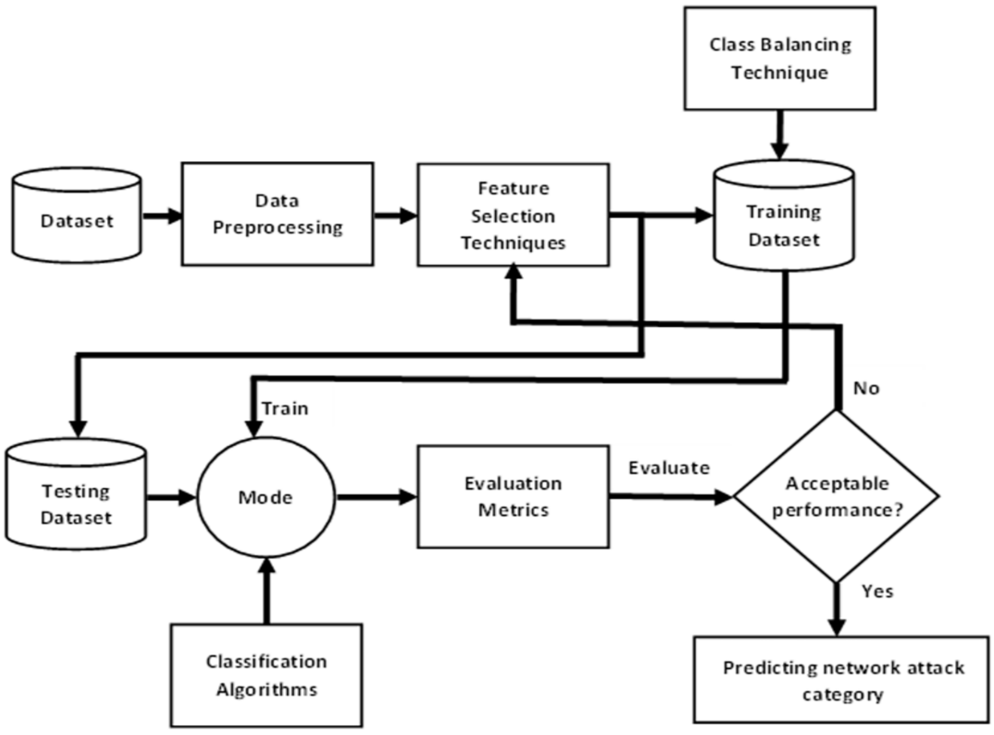


Fig-3.6. Design Flow 2

The design process of a cybersecurity intrusion detection system (IDS) using machine learning (ML) usually follows a process aimed at ensuring its effectiveness and efficiency. Initially, the design phase begins with a good understanding of the process that needs to be written by the participants and experts. This involves defining the scope of the IDS, including the types of attacks it will detect, network operations it will monitor, and performance metrics. Once the requirements are established, the next step is to create the design, which involves identifying the components, their interactions, and information flow. This involves choosing the appropriate algorithm based on the nature of the data and the type of attack to be detected.

Machine learning models are trained using historical data, such as network connection data that indicates the type of attack. Feature selection and extraction are important steps in the process to ensure that the model can distinguish between bad behavior and bad network. Additionally, the modeling process involves the use of appropriate preprocessing techniques, such as profile normalization and outlier processing, to prepare the profiles for the ML model. Once the models are trained and validated, they are integrated into the entire construction process to continuously analyze traffic in real time. The design process also includes developing a notification design process that triggers an alert when suspicious activity is detected. Capacity, optimization, and security considerations throughout the design process are critical to ensuring a secure and reliable IDS.

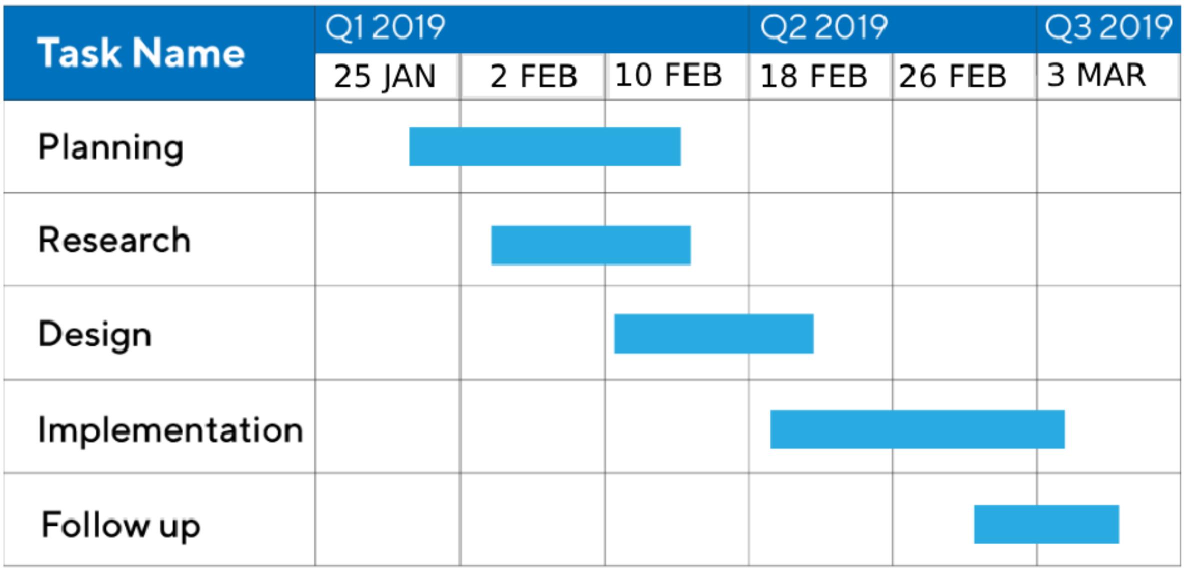


Table-3.1 Gantt Chart

Gantt charts are an important tool for designing cybersecurity intrusion detection systems (IDS) using machine learning (ML). It provides a graphical representation of the work schedule, showing various tasks, their dependencies, and the estimated duration of each task. In the context of creating an IDS, Gantt charts assist the project manager and team in planning and tracking the progress of key activities. For example, activities such as data collection and preprocessing, machine learning model development and training, integration, testing, and deployment can be streamlined and organized in a Gantt chart. This ensures that deadlines are met, resources are allocated efficiently and early intervention is possible. Additionally, Gantt charts serve as a communication tool so that stakeholders can clearly understand the project timeline and milestones. By visualizing the project plan, the Gantt chart guides decision-making and project management and contributes to the development and success of the IDS.

## 3.5. Design Selection

The use case development was a crucial step in defining the functionality and behavior of the Intrusion Detection System (IDS). Each use case represented a specific interaction between the system and the network traffic it monitored. For instance, the "Detecting Anomaly in Network Traffic" use case detailed the sequence of steps from packet reception to anomaly detection and alert generation. This use case guided the design and development process, ensuring that the IDS could effectively identify suspicious patterns in network activity. Additionally, use cases like "Generating Alert for Detected Anomaly" specified how the system should respond once an anomaly is detected, providing clarity on the system's overall workflow.

Simultaneously, a Gantt chart was meticulously crafted to outline the project's timeline and tasks. This visual representation divided the project into phases, such as planning, design, development, testing, deployment, and maintenance. Each phase was assigned specific tasks and durations, allowing for efficient project management and resource allocation. The Gantt chart served as a roadmap, ensuring that deadlines were met and tasks were completed in a logical sequence. It provided a clear overview of the project's progression, from the initial planning stages to the ongoing maintenance and monitoring of the deployed IDS. Through the use of the Gantt chart, the team maintained a structured approach, facilitating collaboration and timely completion of the Intrusion Detection System project.

## 3.6. Implementation plan

**1. Project Initiation:**

Objective: The objective of this phase is to define the purpose and scope of the project.

- Define the problem statement: Detecting network intrusions using machine learning.
- Set project goals: Develop an accurate and efficient intrusion detection system.
- Identify stakeholders: Network security team, IT department, system administrators, etc.
- Establish project timeline and budget.

**2. Needs Analysis:**

Objective: Understand the requirements and constraints of the project.

- Conduct interviews and workshops with stakeholders.
- Gather information on current network security measures and challenges.
- Identify data sources for model training and real-time prediction.

**3. System Requirements Definition:**

Objective:Specify the functional and non-functional requirements of the system.

- Define features and functionalities needed in the intrusion detection system.
- Specify performance metrics (accuracy, latency, etc.) for the model.
- Document system interfaces (data sources, APIs for prediction, user interface).
- Outline hardware and software requirements (server, storage, libraries).

**4. System Design:**
Objective: Create a detailed design plan for the system architecture.

- Design the overall system architecture:
- Frontend (if applicable): User interface for monitoring and alerts.
- Backend: ML model deployment, data processing, and database.
- Define the workflow of data from ingestion to prediction.
- Create wireframes or mockups for the user interface (if needed).
- Architectural decisions:
- How to handle real-time data streaming for predictions.
- Scalability and flexibility of the system.

**5. Technology Stack Selection:**
Objective:Choose appropriate technologies and tools for implementation.

- Programming languages: Python for ML model (already chosen), web development languages (if needed).
- Frameworks and libraries: Scikit-learn, Pandas, Flask (for API development), etc.
- Database: PostgreSQL, MySQL, or SQLite for storing data.
- Deployment: Docker for containerization, PyCharm for development and testing.
- Security tools: SSL certificates, encryption for sensitive data.

**6. Development:**
Objective: Implement the system components based on the design.

- Develop the ML model using Jupyter Notebook:
- Data preprocessing, feature engineering, model training, and evaluation.
- Implement backend APIs using Flask in PyCharm:
- Expose endpoints for model prediction and data retrieval.
- Create user interface components (if applicable):
- Develop frontend for monitoring and visualization.

**7. Database Implementation:**
Objective: Set up the database to store and manage system data.

- Choose and set up the selected database (e.g., PostgreSQL).
- Define database schema based on system requirements.
- Implement scripts for data migration and initialization.
- Ensure data integrity and security measures.

**8. Security Implementation:**
Objective: Implement security measures to protect the system and data.

- Secure API endpoints with authentication and authorization mechanisms.
- Implement SSL/TLS for secure data transmission.
- Apply input validation and sanitization to prevent injection attacks.
- Encrypt sensitive data stored in the database.

**9. Integration Testing:**
Objective: Verify that all system components work together as expected.

- Perform unit tests for individual modules (ML model, API endpoints, database functions).

- Conduct integration tests to ensure proper communication between modules.
- Test edge cases, error handling, and system performance.
- Use tools like pytest for automated testing.

**10. User Training Materials Development:**
Objective: Prepare training materials for system users (administrators, analysts).
- Create user manuals, guides, or video tutorials on how to use the system.
- Provide documentation on interpreting model predictions and alerts.
- Conduct training sessions for key users to familiarize them with the system.

**11. Testing and Quality Assurance:**
Objective: Ensure the system meets quality standards and performs as expected.
- Perform system-wide testing:
- Regression testing, stress testing, load testing.
- Quality assurance checks:
- Code reviews, code standards adherence, security audits.
- Address any bugs or issues found during testing.
- Continuously monitor system performance and behavior.

**12. Deployment:**
Objective: Roll out the system for production use.
- Create deployment packages or Docker images.
- Set up production servers or cloud infrastructure (AWS, Azure, etc.).
- Deploy the ML model, APIs, frontend (if applicable) to production environment.
- Configure monitoring and logging for system health and performance.
- Conduct final user acceptance testing (UAT) with stakeholders.

**13. Maintenance and Support:**
Objective: Ensure ongoing maintenance and support for the deployed system.
- Establish a maintenance schedule for regular updates and patches.
- Provide technical support channels for users (helpdesk, email, etc.).
- Monitor system logs and user feedback for improvements.
- Plan for future enhancements and updates based on user feedback and new requirements.

# CHAPTER 4
# RESULTS ANALYSIS AND VALIDATION

## 4.1 Implementation of solution

The implementation details of the solution, focusing on the analysis of the development of the backend, frontend, utilization of the technology stack, integration of course management and grade calculation, authentication and access control for users, testing and quality control, stakeholder engagement, deployment, and continuous enhancement.

The project develops an Intrusion Detection System (IDS) for cybersecurity using machine learning techniques. It starts by collecting and preprocessing cybersecurity data, including network traffic information. Relevant features are engineered, and each data point is labeled as normal or belonging to specific attack types. Exploratory Data Analysis (EDA) techniques provide insights into the dataset. Various machine learning algorithms such as logistic regression, decision trees, k-nearest neighbors, support vector machines, naive Bayes, and ensemble methods are trained on the labeled dataset to build predictive models for intrusion detection. The models are evaluated using performance metrics like accuracy, precision, recall, and confusion matrix. The best-performing model or ensemble is selected for deployment as the final IDS. Once deployed, the IDS continuously monitors network traffic, detects potential intrusions or suspicious activities, and alerts security personnel. Regular monitoring and maintenance ensure the effectiveness of the deployed IDS in detecting and mitigating cybersecurity threats.

### 4.1.1 Analysis

**Development of the Backend:**

Objective: Develop the backend system to handle data processing, business logic, and interaction with the database.

Tasks:

- Implement backend APIs using Flask.
- Define endpoints for course management (CRUD operations).
- Implement APIs for grade calculation based on predefined criteria.
- Develop data models for courses, grades, users, etc., using an ORM (Object-Relational Mapping) like SQLAlchemy.
- Integrate with the selected database (e.g., PostgreSQL) for data storage.
- Implement error handling and logging for backend operations.

**Development of the Frontend:**

Objective: Create a user-friendly interface for users to interact with the system.

Tasks:

- Develop frontend using web technologies (HTML, CSS, JavaScript).
- Create UI components for course management (add, edit, delete courses).
- Implement UI for entering student grades and calculating final grades.
- Design intuitive user flows for administrators, instructors, and students.
- Ensure responsive design for usability on different devices (desktop, tablet, mobile).

**Utilization of the Technology Stack:**
Objective: Use appropriate technologies and tools for efficient development and deployment.
Technology Stack:

- Backend: Flask (Python), SQLAlchemy for ORM.
- Frontend: HTML5, CSS3, JavaScript (React or Vue.js for dynamic components).
- Database: PostgreSQL for storing course, grade, and user data.
- Authentication: JWT (JSON Web Tokens) for user authentication.
- Testing: Pytest for backend unit testing, Jest for frontend testing.
- Deployment: Docker for containerization, AWS (Amazon Web Services) for cloud hosting.

**Integration of Course Management:**
Objective: Integrate functionalities for managing courses within the system.
Tasks:

- Implement CRUD operations for courses (Create, Read, Update, Delete).
- Allow administrators to add new courses with details (title, description, instructor).
- Provide a list view for browsing existing courses and their details.
- Implement validation to ensure course data integrity.

**Integration of Grade Calculation:**
Objective: Integrate automated grade calculation based on specified criteria.
Tasks:

- Implement algorithms for calculating final grades from student assessments.
- Define grading criteria (weighted scores, letter grades, etc.).
- Automatically calculate and update student grades based on entered assessments.
- Allow instructors to review and adjust calculated grades if necessary.

**Authentication and Access Control for Users:**
Objective: Ensure secure access to the system with proper authentication mechanisms.
Tasks:

- Implement user registration and login functionalities.
- Utilize JWT for token-based authentication.
- Define user roles (administrator, instructor, student) with corresponding access levels.
- Implement access control to restrict functionalities based on user roles.
- Secure sensitive endpoints (e.g., grade calculation) with role-based authorization.

**Testing and Quality Control:**
Objective: Ensure the system functions correctly and meets quality standards.
Tasks:

- Conduct unit tests for backend API endpoints (Flask).
- Perform integration tests to check interactions between frontend and backend.
- Test user flows for course management, grade calculation, and authentication.
- Use Jest for frontend testing to validate UI components.
- Implement logging and error monitoring to track system behavior.

**Stakeholder Engagement:**
Objective: Involve stakeholders throughout the development process for feedback and validation.
Approach:

- Hold regular meetings with administrators, instructors, and students.
- Gather feedback on user interface design, functionalities, and user experience.
- Conduct demo sessions to showcase new features and gather input.
- Collaborate closely with the educational institution's IT department and faculty.

**Deployment:**
Objective:Prepare the system for production deployment in a secure and efficient manner.
Steps:

- Create deployment packages or Docker images for the backend and frontend.
- Set up AWS (Amazon Web Services) or another cloud platform for hosting.
- Configure environment variables for production settings (database connection, secret keys).
- Implement HTTPS for secure data transmission.
- Perform final testing in a production-like environment.
- Plan for a phased rollout to ensure minimal disruption.

**Continuous Enhancement:**
Objective: Plan for ongoing improvements and feature additions based on feedback.
Approach:

- Establish a feedback loop for users to submit suggestions and issues.
- Prioritize enhancement requests based on impact and feasibility.
- Plan regular updates and releases to add new functionalities.
- Monitor system performance and user engagement metrics.
- Conduct user surveys or interviews to gather insights for future iterations.

### 4.1.2 Testing

**White-Box testing**
In the realm of Intrusion Detection for Cyber Security, white-box testing delves deep into the inner workings of the machine learning models utilized. This involves examining the algorithms' efficiency in classifying network traffic accurately. Unit tests are constructed to validate individual components of the models, such as feature selection techniques and model hyperparameters. Moreover, code inspection ensures the models are free from bias and overfitting, crucial for reliable detection of anomalies. In addition to model testing, system integration tests are conducted to assess the seamless interaction between the detection models and the data preprocessing pipeline. Robustness against adversarial attacks is another critical aspect tested, verifying the models' resistance to malicious inputs that could bypass detection.

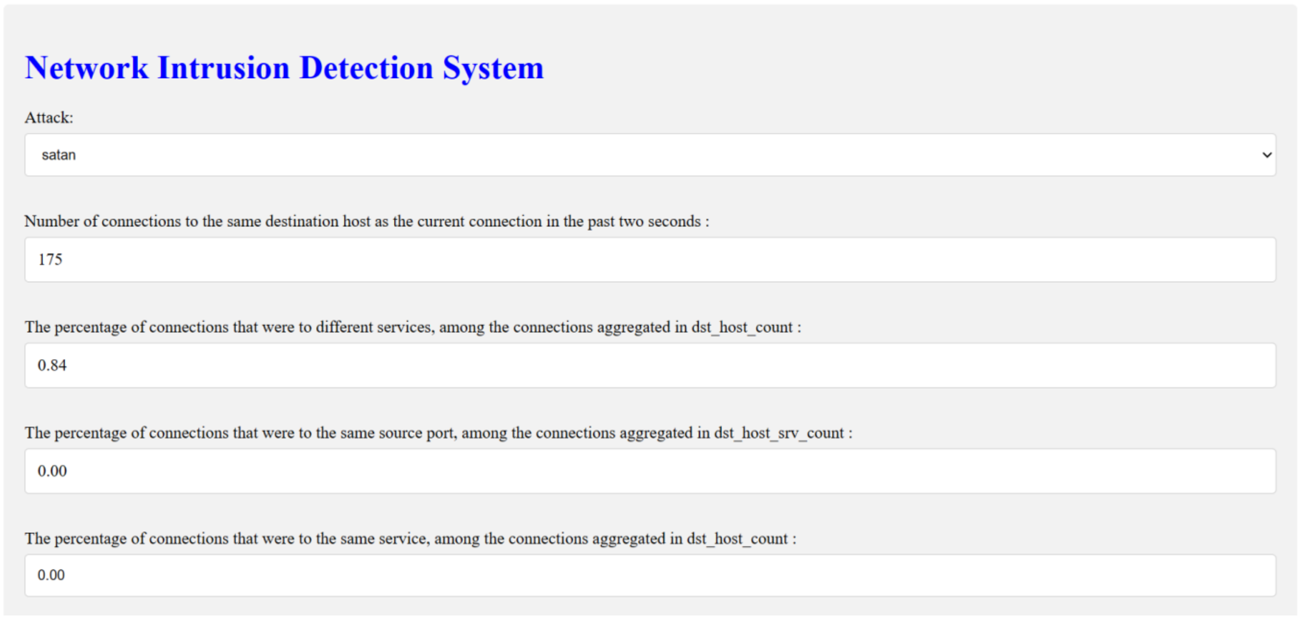
**Black-Box testing**

Black-box testing, on the other hand, focuses on the system as a whole, treating the intrusion detection system as a complete entity. Functional testing is employed to validate the system's response to different attack scenarios, ensuring alerts are triggered appropriately. Various attack vectors, such as

DoS, DDoS, port scanning, and SQL injection, are simulated to verify the system's detection capabilities. End-to-end testing is crucial to mimic real-world scenarios, from data ingestion to classification and alert generation. Performance testing assesses the system's efficiency under load, ensuring it can handle a high volume of network traffic without compromising detection accuracy. Integration with existing security infrastructure, such as firewalls and incident response systems, is also tested for seamless coordination during an attack.

## 4.2 Results

➢ Below are the implementation Screenshots



**Network Intrusion Detection System**

Attack:

satan

Number of connections to the same destination host as the current connection in the past two seconds :

175

The percentage of connections that were to different services, among the connections aggregated in dst_host_count :

0.84

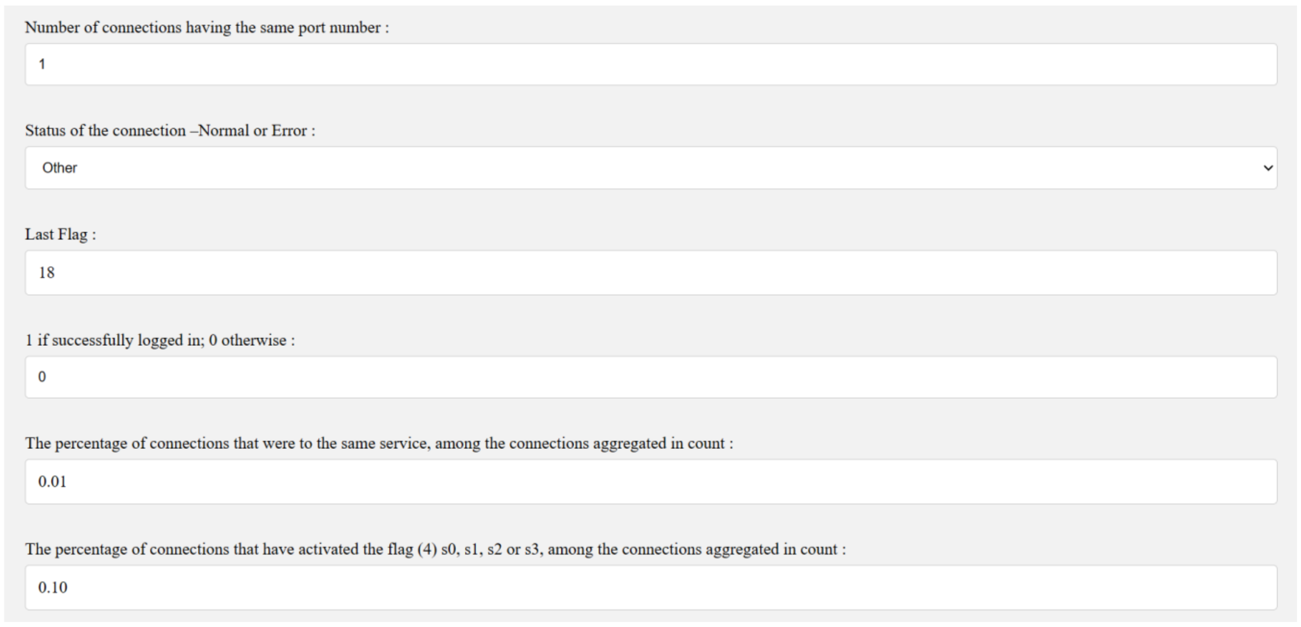
The percentage of connections that were to the same source port, among the connections aggregated in dst_host_srv_count :

0.00

The percentage of connections that were to the same service, among the connections aggregated in dst_host_count :

0.00

Fig-4.1 Result 1.1



Number of connections having the same port number :

1

Status of the connection –Normal or Error :

Other

Last Flag :

18

1 if successfully logged in; 0 otherwise :

0

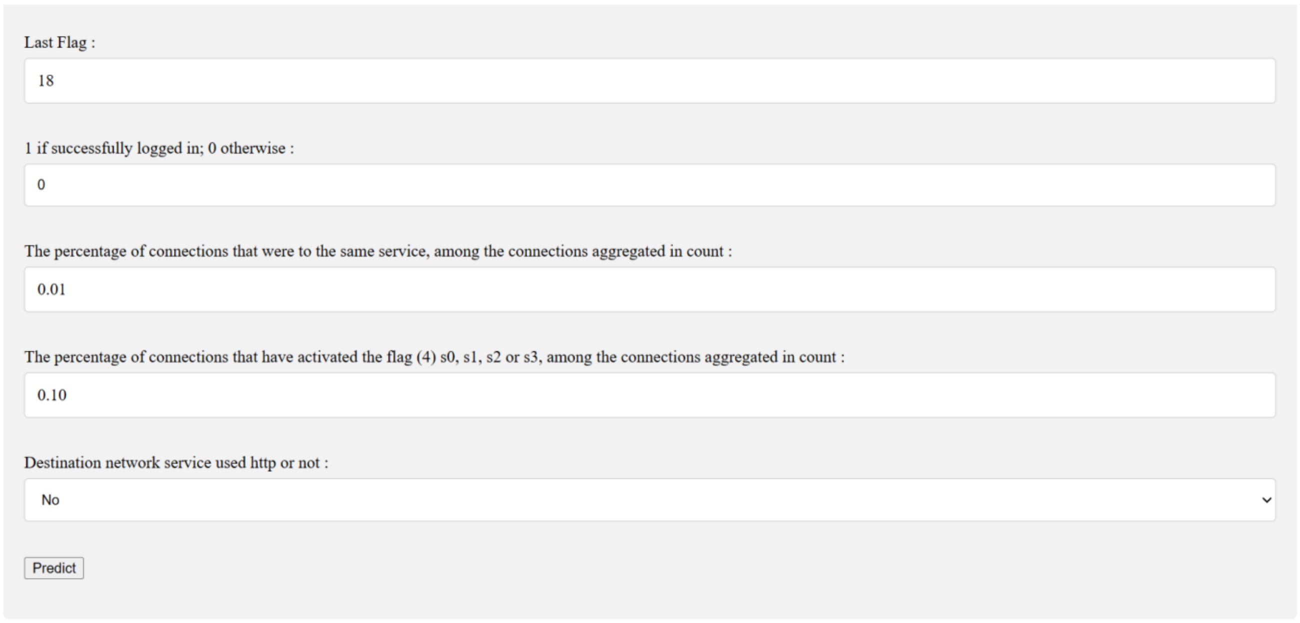
The percentage of connections that were to the same service, among the connections aggregated in count :

0.01

The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count :

0.10

Fig-4.2 Result 1.2


Last Flag :

18

1 if successfully logged in; 0 otherwise :

0

The percentage of connections that were to the same service, among the connections aggregated in count :

0.01

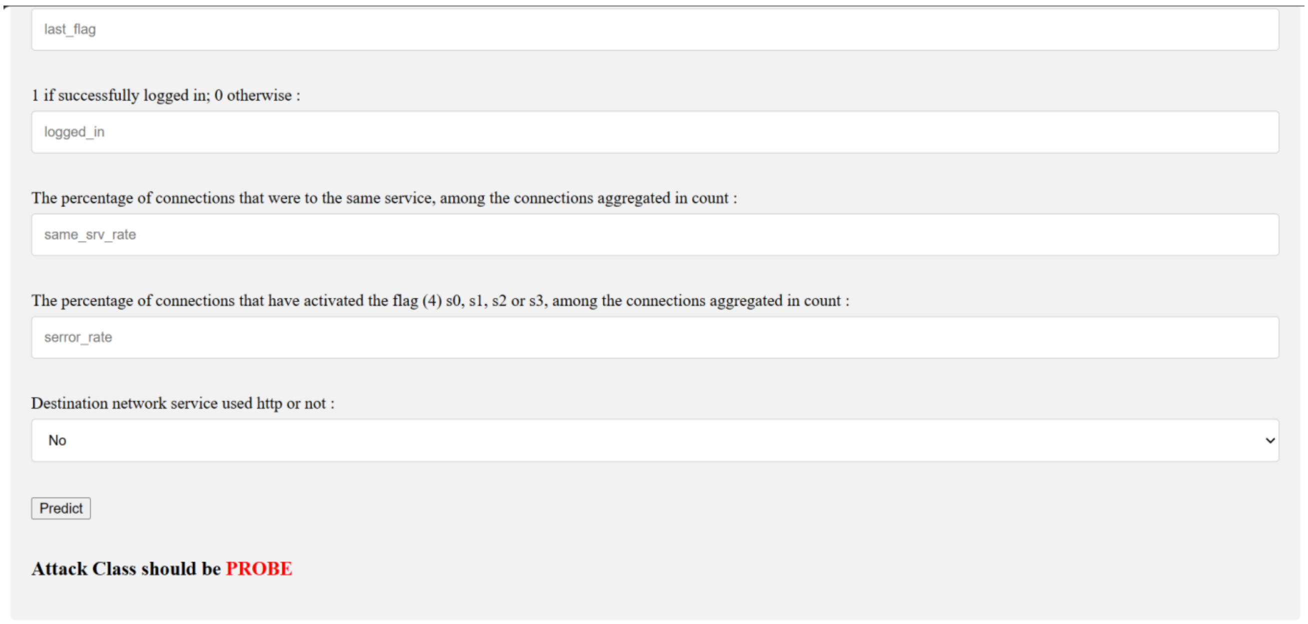
The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count :

0.10

Destination network service used http or not :

No

Fig-4.3 Result 1.3


last_flag

1 if successfully logged in; 0 otherwise :

logged_in

The percentage of connections that were to the same service, among the connections aggregated in count :

same_srv_rate

The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count :

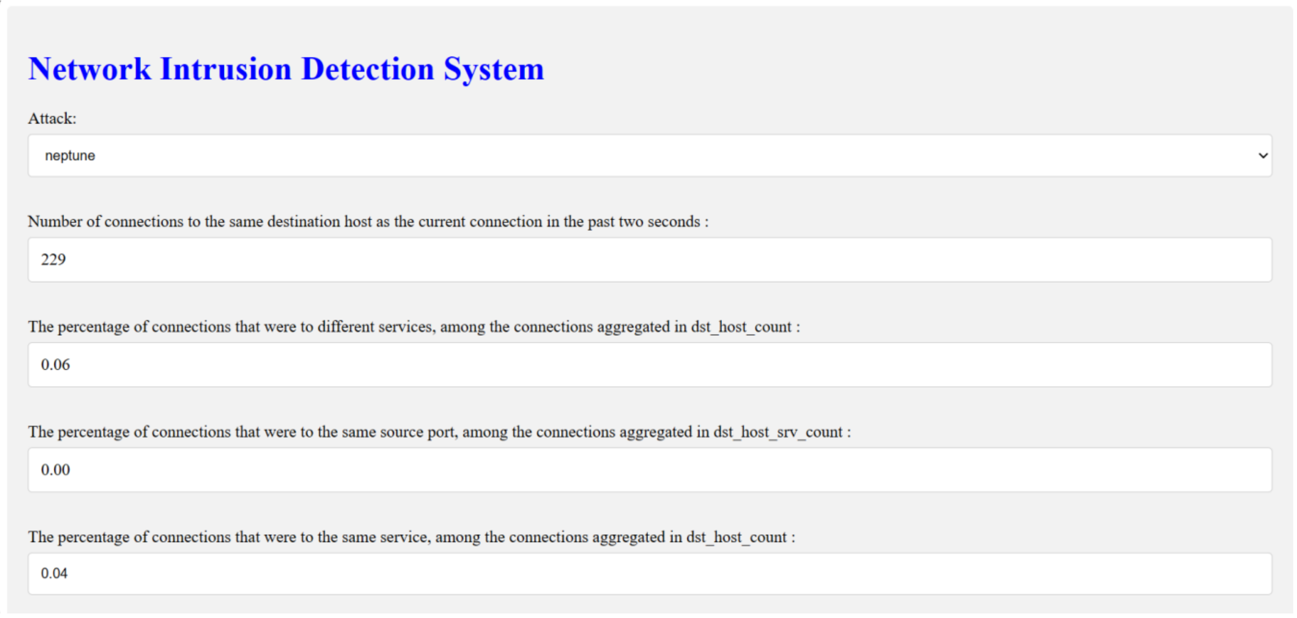
serror_rate

Destination network service used http or not :

No

**Attack Class should be PROBE**

Fig.4.4 Result 1.4



Network Intrusion Detection System

Attack:

neptune

Number of connections to the same destination host as the current connection in the past two seconds :

229

The percentage of connections that were to different services, among the connections aggregated in dst_host_count :

0.06

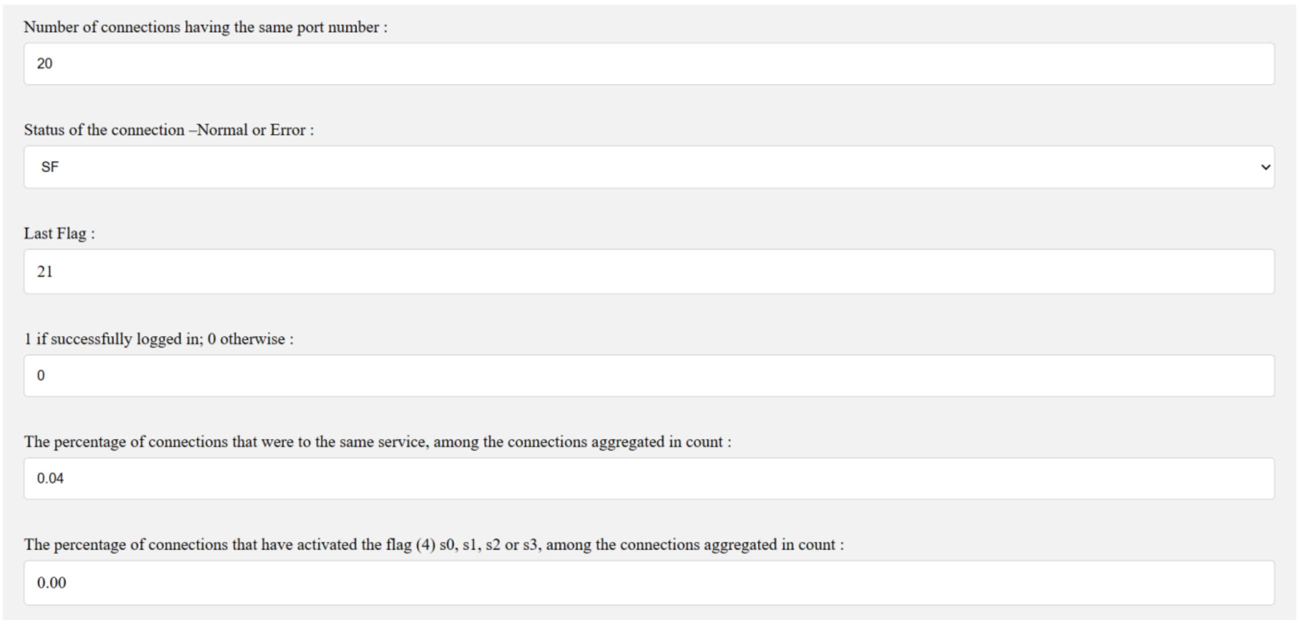
The percentage of connections that were to the same source port, among the connections aggregated in dst_host_srv_count :

0.00

The percentage of connections that were to the same service, among the connections aggregated in dst_host_count :

0.04

Fig.4.5 Result 2.1



Number of connections having the same port number :

20

Status of the connection –Normal or Error :

SF

Last Flag :

21

1 if successfully logged in; 0 otherwise :

0

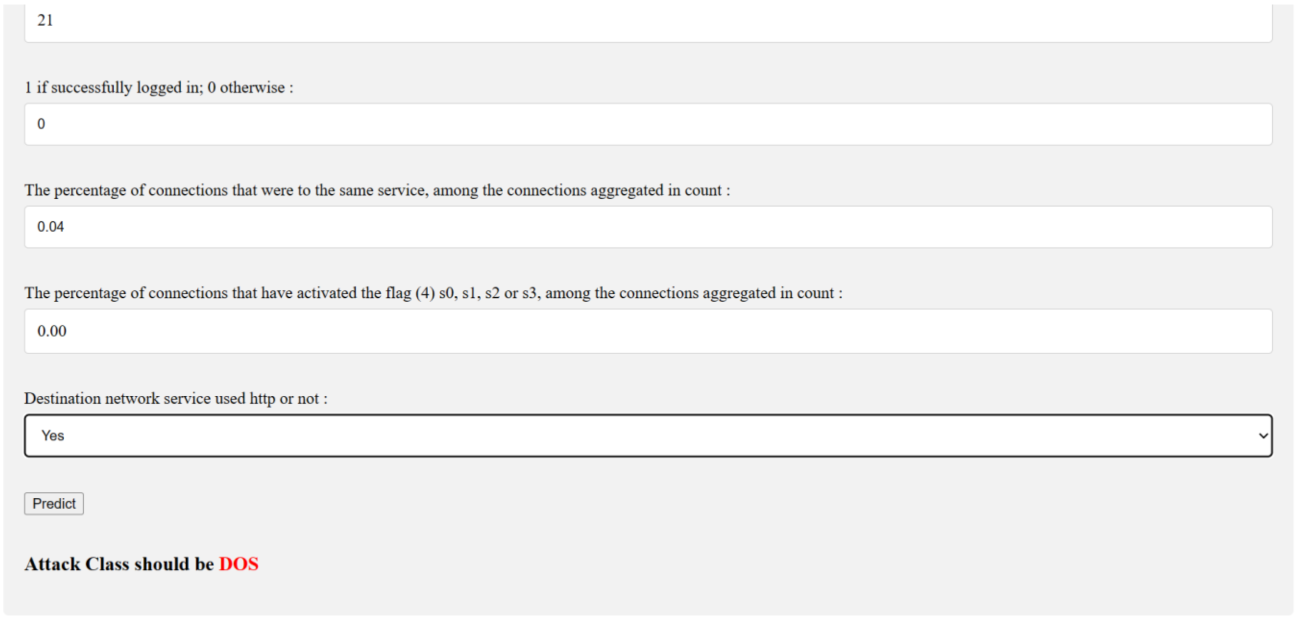
The percentage of connections that were to the same service, among the connections aggregated in count :

0.04

The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count :

0.00

Fig.4.6 Result 2.2


21

1 if successfully logged in; 0 otherwise :

0

The percentage of connections that were to the same service, among the connections aggregated in count :

0.04

The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count :

0.00

Destination network service used http or not :

Yes

**Attack Class should be DOS**

Fig.4.7 Result 2.3

# CHAPTER 5
# CONCLUSION AND FUTURE WORK

## 5.1 Conclusion:

This project successfully applied machine learning in the development of effective network security intrusion detection systems (IDS). The project enabled the classification of network behavior and different types of traffic by utilizing supervised learning algorithms and pre-processing techniques on a large dataset containing network traffic data. The project determined the process for arriving at the optimal design through a comprehensive evaluation and comparison of various models, including transportation, wood selection, and assembly methods. IDS design increases network security by enabling timely detection and response to threats, thus protecting critical systems and data from cyber-attacks.

As cyber threats continue to evolve, IDS must be continually refined and improved to maintain its effectiveness against security risks and ensure the protection of digital assets in use. Rigorous testing and validation processes are used to ensure the IDS meets performance standards and detects a wide range of cyber threats. Regular updates and maintenance are also important to keep the IDS up to date with security changes and maintain its effectiveness in protecting digital assets and infrastructure.

IDS includes methods such as data collection, prioritization, feature selection, design, analysis, distribution and monitoring. By leveraging advanced machine learning algorithms and integration technologies, IDS can distill network traffic into classifiable and detectable security threats over time, thereby improving overall network security posture and minimizing the risk of network outages. Continuous research and development in this field is essential to solve emerging security problems and strengthen digital systems and networks in the ongoing world.

## 5.2 Future work:

Future work on business network security and access to machine learning has great potential for further progress and innovation. As technology advances and cyber threats become more prevalent, there are many research and development avenues to increase the effectiveness and power of intrusion detection tools (IDS) and solve emerging security problems. Here we discuss some important points for future work in this field:

1. Advanced machine learning techniques: While traditional machine learning techniques have proven effective in intrusion detection, advanced techniques such as deep learning, learning embedding, and artificial neural networks (GANs) are still being researched. These techniques are promising for extracting complex patterns and features from advanced devices, which can improve the accuracy and performance of IDS.

2. Anomaly detection: Anomaly-based intrusion detection focuses on identifying differences in behavior and is an important area of future research. Leveraging unsupervised learning algorithms and anomaly detection techniques, IDS can detect new and previously unseen threats without relying on signature lists or predefined patterns. This advanced approach is especially useful for detecting zero-day attacks and persistent threats (APTs).

3. Adversarial Robustness: As attackers continue to develop evasion techniques and countermeasures to circumvent IDS, there is a need to increase the robustness and resilience of machine learning models against this challenge. Future research may focus on the development of training strategies, improvement of output models, and obfuscation techniques to increase the security of IDS and reduce the risk of sophisticated adversary evasion.

4. Detection and real-time response: As IoT devices expand and the volume and velocity of data generated by modern networks continues to increase, the need for intrusion detection for real-time and continuous response capabilities also increases. Future work may explore the integration of streaming analytics, edge computing, and distributed technologies to enable IDS to analyze and respond to immediate security threats, thereby reducing search delays and reducing the impact of cyber attacks. >

5. Explaining and explaining AI: The black box of many learning machines poses a challenge in understanding and explaining their decisions; This is important to build trust and confidence in IDS. Future research may focus on developing artificial intelligence and descriptive machine learning models to gain insight into the factors that drive classification decisions and help security analysts understand the logic behind detected threats.

6. Collaboration and learning in government: Collaboration and learning technology has the potential to solve privacy and data sharing limitations associated with the inner half of IDS. By bringing training standards across multiple organizations or devices while maintaining data privacy and security, these strategies can help increase efficiency and find a system that supports intelligence and many different types of information.

7. Autonomous and Adaptive Systems: Autonomous IDS that can monitor, identify, and adapt to cyber threats without human intervention is a growing field. Future studies may investigate integrating cognitive and self-determination abilities into IDS to enable self-learning, self-improvement, and self-improvement to increase work efficiency and work in a positive environment and discussions. .

8. Scientific Research: Cybersecurity is inherently collaborative and draws insights and methods from computer science, engineering, mathematics, psychology, and more. Future research can benefit from a collaborative and collaborative approach that integrates technical, behavioral, and socio-economic factors to solve complex cybersecurity problems and develop solutions to protect digital assets and infrastructure.

## References

[1] M. P. K. Shelke, M. S. Sontakke, and A. D. Gawande, “Intrusion Detection System for Cloud Computing,” Int. J. Sci. Technol. Res., vol. 1, no. 4, pp. 67–71, 2012.

[2] L. Han, "Using a Dynamic K-means Algorithm to Detect Anomaly Activities," 2011, pp. 1049-1052.

[3] Levin, "KDD-99 Classifier Learning Contest: LLSoft s Results Overview," SIGKDD explorations, vol. 1, pp. 67-75, 2000.

[4] M. Thelwall, ‘‘Microsoft academic automatic document searches: Accuracy for journal articles and suitability for citation analysis,’’ J. Informetrics, vol. 12, no. 1, pp. 1–9, Feb. 2018.

[5] M. Gusenbauer, ‘‘Google scholar to overshadow them all? Comparing the sizes of 12 academic search engines and bibliographic databases,’’ Scientometrics, vol. 118, no. 1, pp. 177–214, Nov. 2018.

[6] J. McHugh, “Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory,” ACM Transactions on Information and System Security, vol. 3, no. 4, pp. 262–294, 2000.

[7] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, “A Detailed Analysis of the KDD CUP 99 Data Set,” Submitted to Second IEEE Symposis

[8] NSL KDD dataset, Accessed December 2015, https://github.com/defcom17/NSL_KDD

[9] P. Ghosh, C. Debnath, and D. Metia, “An Efficient Hybrid Multilevel Intrusion Detection System in Cloud Environment,” IOSR J. Comput. Eng., vol. 16, no. 4, pp. 16–26, 2014.

[10] William D. How AI can help improve intrusion detection systems [Internet]. GCN. Available from: https://gcn.com/cybersecurity/2020/04/how-ai-can-help-improve intrusion-detection-systems/291266/

[11] Dhanabal, L., Dr. S.P. Shantharajah, "A Study on NSL_KDD Daaset for Intrusion Detection System Based on Classification Algorithms," International Journal of Advanced Research in Computer and Communication Engineering, vol. 4, issue 6, pp. 446-452, June 2015…

[12] C. F. Tsai, et al., "Intrusion detection by machine learning: A review," Expert Systems with Applications, vol. 36, pp. 11994-12000, 2009.

[13] C. Cowan et al., "Stackguard: automatic adaptive detection and prevention of buffer-overflow attacks," in USENIX security symposium, 1998, vol. 98, pp. 63–78: San Antonio, TX

[14] Creech G, Hu J (2014a) A semantic approach to host-based intrusion detection systems using Contiguousand Discontiguous system call patterns. IEEE Trans Comput 63(4):807–819

[15] Agrawal S, Agrawal J (2015) Survey on anomaly detection using data mining techniques. Procedia Computer Science 60:708–713

[16] Valdovinos I., Perez-Diaz J., Choo K.K., Botero J. Emerging DDoS attack detection and mitigation strategies in software-defined networks: Taxonomy, challenges and future directions. Journal of Network and Computer Applications [Internet]. 2021 Aug 1 [cited 2021 Sep 23];187:103093.Availablefrom:
https://www.sciencedirect.com/science/article/pii/S1084804521001156

[17] Niksefat S., Kaghazgaran P., Sadeghiyan B. Privacy issues in intrusion detection systems: A taxonomy, survey and future directions. Computer Science Review. 2017 Aug;25:69–78.

[18] Cybersecurity Spotlight – Signature-Based vs Anomaly-Based Detection [Internet]. CIS.Available from: https://www.cisecurity.org/insights/spotlight/cybersecurity spotlight-signature-based-vs-anomaly-based-detection

[19]Australian.(2017,November).AustralianCyberSecurityCenterThreatreport201Available:https://www.acsc.gov.au/publications/ACSC_Threat_Report_2017.pdf

[20] Bhuyan MH, Bhattacharyya DK, Kalita JK (2014) Network anomaly detection: methods, systems and tools. IEEE Communications Surveys & Tutorials 16(1):303–336

# APPENDIX

## App.py

```
import numpy as np
from flask import Flask, request, jsonify, render_template
import joblib

app = Flask(__name__)
model = joblib.load('model.pkl')

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict',methods=['POST'])
def predict():

    int_features = [float(x) for x in request.form.values()]

    if int_features[0]==0:
        f_features=[0,0,0]+int_features[1:]
    elif int_features[0]==1:
        f_features=[1,0,0]+int_features[1:]
    elif int_features[0]==2:
        f_features=[0,1,0]+int_features[1:]
    else:
        f_features=[0,0,1]+int_features[1:]

    if f_features[6]==0:
        fn_features=f_features[:6]+[0,0]+f_features[7:]
    elif f_features[6]==1:
        fn_features=f_features[:6]+[1,0]+f_features[7:]
    else:
        fn_features=f_features[:6]+[0,1]+f_features[7:]

    final_features = [np.array(fn_features)]
    predict = model.predict(final_features)

    if predict==0:
        output='Normal'
    elif predict==1:
        output='DOS'
    elif predict==2:
        output='PROBE'
    elif predict==3:
        output='R2L'
    else:
        output='U2R'

    return render_template('index.html', output=output)

@app.route('/results',methods=['POST'])
def results():

    data = request.get_json(force=True)
    predict = model.predict([np.array(list(data.values()))])

    if predict==0:
        output='Normal'
    elif predict==1:
        output='DOS'
    elif predict==2:
        output='PROBE'
    elif predict==3:
        output='R2L'
    else:
        output='U2R'

    return jsonify(output)

if __name__ == "__main__":
    app.run()
```

## Style.css

```
/* Style inputs with type="text", select elements */
input[type=text], select {
  width: 100%; /* Full width */
  padding: 12px; /* Some padding */
  border: 1px solid #ccc; /* Gray border */
  border-radius: 4px; /* Rounded borders */
  box-sizing: border-box; /* Make sure that padding and width stays in place */
  margin-top: 6px; /* Add a top margin */
  margin-bottom: 16px; /* Bottom margin */
  resize: vertical /* Allow the user to vertically resize the textarea (not
horizontally) */
}

/* Style the submit button with a specific background color etc */
input[type=submit] {
  background-color: #4CAF50;
  color: white;
  padding: 12px 20px;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

/* When moving the mouse over the submit button, add a darker green color */
input[type=submit]:hover {
  background-color: #45a049;
}

/* Add a background color and some padding around the form */
.login {
  border-radius: 5px;
  background-color: #f2f2f2;
  padding: 20px;
}
```

## IDS.ipny

```
[123]
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
[124]
model = SGDClassifier(loss="hinge", penalty="l2")
model.fit(X_train, y_train)
[125]
y_pred=model.predict(X_test)
y_pred
[126]
accuracy_score( y_test, y_pred )
[127]
n_iters = [5, 10, 20, 50, 100, 1000]
scores = []
for n_iter in n_iters:
    model = SGDClassifier(loss="hinge", penalty="l2", max_iter=n_iter)
    model.fit(X_train, y_train)
    scores.append(model.score(X_test, y_test))
plt.title("Effect of n_iter")
plt.xlabel("n_iter")
plt.ylabel("score")
plt.plot(n_iters, scores)
[128]
# losses
losses = ["hinge", "log", "modified_huber", "perceptron", "squared_hinge"]
scores = []
for loss in losses:
    model = SGDClassifier(loss=loss, penalty="l2", max_iter=1000)
    model.fit(X_train, y_train)
    scores.append(model.score(X_test, y_test))
plt.xlabel("loss")
plt.ylabel("score")
plt.title("Effect of loss")
x = np.arange(len(losses))
plt.xticks(x, losses)
plt.plot(x, scores)
[129]
from sklearn.model_selection import GridSearchCV

params = {
    "loss" : ["hinge", "log", "squared_hinge", "modified_huber"],
    "alpha" : [0.0001, 0.001, 0.01, 0.1],
    "penalty" : ["l2", "l1", "none"],
}
model = SGDClassifier(max_iter=100)
clf = GridSearchCV(model, param_grid=params)
[130]
clf.fit(X_train, y_train)
print(clf.best_score_)
[131]
y_pred=clf.predict(X_test)
y_pred
[132]
accuracy_score( y_test, y_pred )
```

Neural Network Model

```
[133]
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
scaler = StandardScaler()
# Fit only to the training data
scaler.fit(X_train)
[134]
# Now apply the transformations to the data:
train_X = scaler.transform(X_train)
test_X = scaler.transform(X_test)
[135]
mlp = MLPClassifier(hidden_layer_sizes=(30,30,30))
mlp.fit(train_X,y_train)
[136]
y_pred=mlp.predict(test_X)
y_pred
array([1., 0., 2., ..., 1., 0., 2.])
[137]
from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(y_test,y_pred))
[138]
print(classification_report(y_test,y_pred))
[139]
mlp.coefs_
[140]
accuracy_score( y_test, y_pred )
[141]
from sklearn import model_selection
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
[142]
seed = 7
kfold = model_selection.KFold(n_splits=10, random_state=seed)
```

```
cart = DecisionTreeClassifier()
num_trees = 100
model = BaggingClassifier(base_estimator=cart, n_estimators=num_trees, random_state=seed)
results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold)
print(results.mean())
[143]
model.fit(X_train, y_train)
[144]
y_pred=model.predict(X_test)
y_pred
[145]
accuracy_score( y_test, y_pred )
[146]
from sklearn.ensemble import RandomForestClassifier
[147]
seed = 7
num_trees = 100
max_features = 3
kfold = model_selection.KFold(n_splits=10, random_state=seed)
model = RandomForestClassifier(n_estimators=num_trees, max_features=max_features)
results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold)
print(results.mean())
[148]
model.fit(X_train, y_train)
[149]
y_pred=model.predict(X_test)
y_pred
array([1., 0., 2., ..., 1., 0., 2.])
[150]
accuracy_score( y_test, y_pred )
[151]
from sklearn.ensemble import ExtraTreesClassifier
[152]
seed = 7
num_trees = 100
max_features = 7
kfold = model_selection.KFold(n_splits=10, random_state=seed)
model = ExtraTreesClassifier(n_estimators=num_trees, max_features=max_features)
results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold)
print(results.mean())
[153]
model.fit(X_train, y_train)
[154]
y_pred=model.predict(X_test)
y_pred
[155]
```

```
accuracy_score( y_test, y_pred )
[156]
from sklearn.ensemble import AdaBoostClassifier
[157]
seed = 7
num_trees = 30
kfold = model_selection.KFold(n_splits=10, random_state=seed)
model = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold)
print(results.mean())
[158]
model.fit(X_train, y_train)
[159]
y_pred=model.predict(X_test)
y_pred
[160]
accuracy_score( y_test, y_pred )
[161]
from sklearn.ensemble import GradientBoostingClassifier
[162]
seed = 7
num_trees = 100
kfold = model_selection.KFold(n_splits=10, random_state=seed)
model = GradientBoostingClassifier(n_estimators=num_trees, random_state=seed)
results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold)
print(results.mean())
[163]
model.fit(X_train, y_train)
[164]
y_pred=model.predict(X_test)
y_pred
[165]
accuracy_score( y_test, y_pred )
[166]
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
[167]
seed = 7
kfold = model_selection.KFold(n_splits=10, random_state=seed)
# create the sub models
estimators = []
model1 = LogisticRegression()
estimators.append(('logistic', model1))
model2 = DecisionTreeClassifier()
estimators.append(('cart', model2))
model3 = SVC()
```

```
estimators.append(('svm', model3))
# create the ensemble model
ensemble = VotingClassifier(estimators)
results = model_selection.cross_val_score(ensemble, X_train, y_train, cv=kfold)
print(results.mean())
[168]
ensemble.fit(X_train, y_train)
[169]
y_pred=ensemble.predict(X_test)
y_pred
[170]
accuracy_score( y_test, y_pred )
```

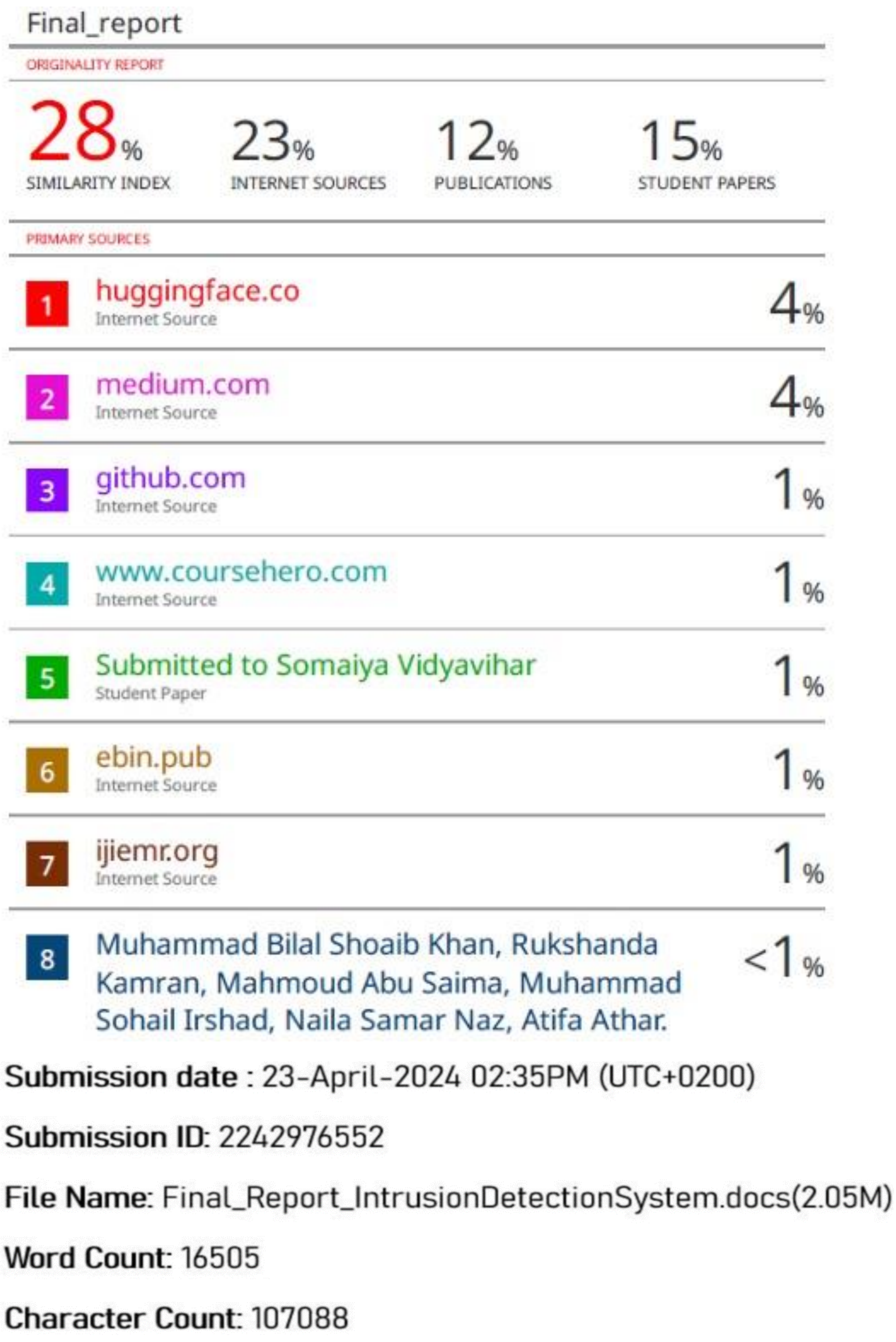
Save Model

```
[171]
import pickle
# Saving model to disk of random forest
pickle.dump(lr_clf, open('model.pkl','wb'))
```

Load Model and Predict

```
[172]
import pickle
model=pickle.load(open('model.pkl', 'rb'))
model.predict([[1,0,0,229,0.06,0.00,0.04,10,0,0,21,0,0.04,0.00,0]])
```

## Plagiarism Report



Final_report

ORIGINALITY REPORT

| | | | |
|---|---|---|---|
| 28% | 23% | 12% | 15% |
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | huggingface.co<br>Internet Source | 4% |
| 2 | medium.com<br>Internet Source | 4% |
| 3 | github.com<br>Internet Source | 1% |
| 4 | www.coursehero.com<br>Internet Source | 1% |
| 5 | Submitted to Somaiya Vidyavihar<br>Student Paper | 1% |
| 6 | ebin.pub<br>Internet Source | 1% |
| 7 | ijiemr.org<br>Internet Source | 1% |
| 8 | Muhammad Bilal Shoaib Khan, Rukshanda Kamran, Mahmoud Abu Saima, Muhammad Sohail Irshad, Naila Samar Naz, Atifa Athar. | <1% |

**Submission date** : 23-April-2024 02:35PM (UTC+0200)

**Submission ID**: 2242976552

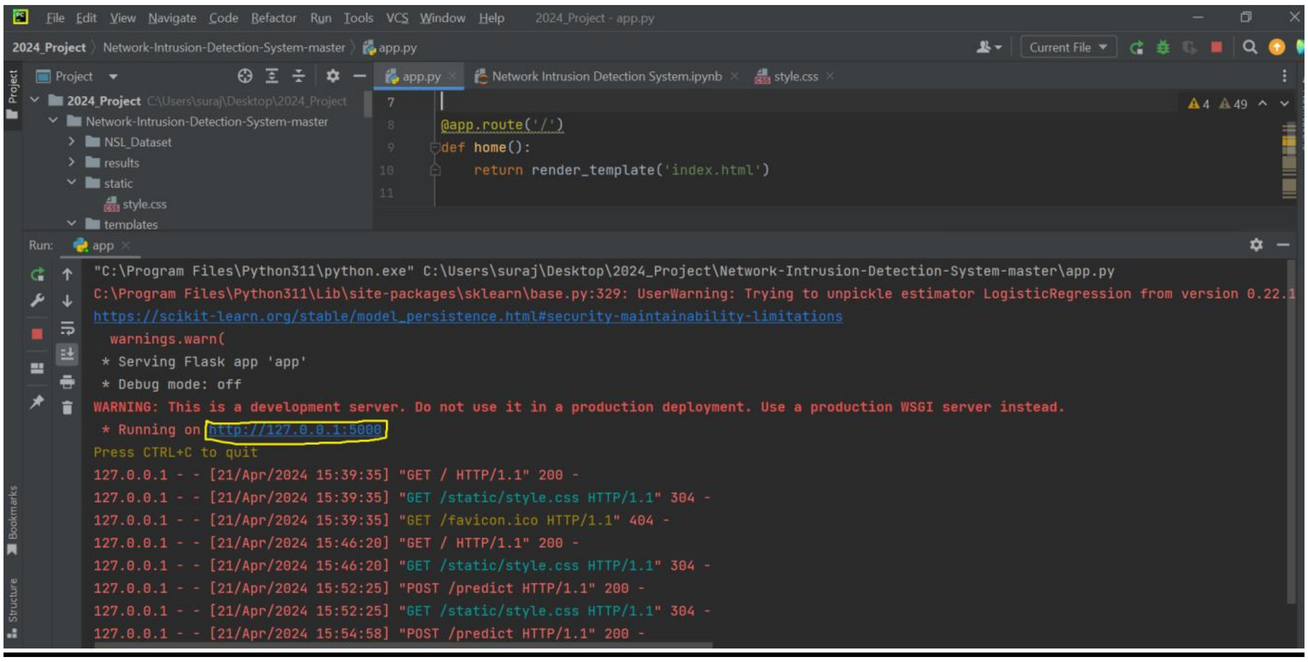
**File Name**: Final_Report_IntrusionDetectionSystem.docs(2.05M)

**Word Count**: 16505

**Character Count**: 107088

# USER MANUAL

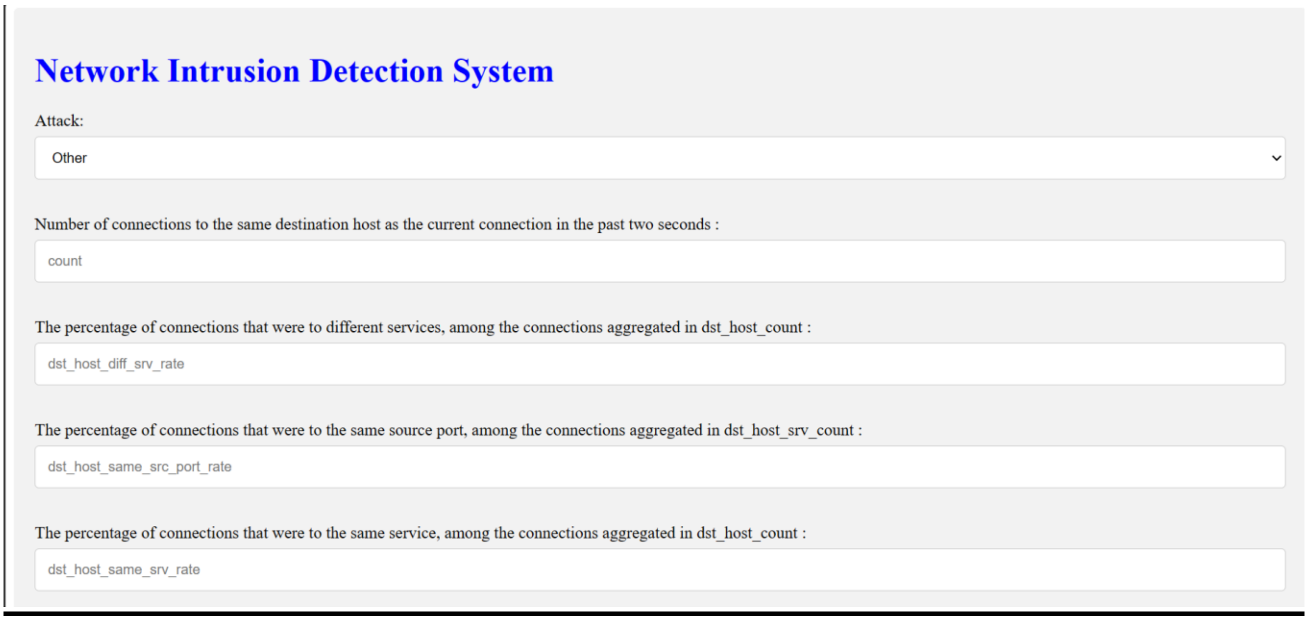
1. After Running the project in Pycharm click the Link of the Development Server



```
@app.route('/')
def home():
    return render_template('index.html')
```

```
"C:\Program Files\Python311\python.exe" C:\Users\suraj\Desktop\2024_Project\Network-Intrusion-Detection-System-master\app.py
C:\Program Files\Python311\Lib\site-packages\sklearn\base.py:329: UserWarning: Trying to unpickle estimator LogisticRegression from version 0.22.1
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [21/Apr/2024 15:39:35] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [21/Apr/2024 15:39:35] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [21/Apr/2024 15:39:35] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [21/Apr/2024 15:46:20] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [21/Apr/2024 15:46:20] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [21/Apr/2024 15:52:25] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [21/Apr/2024 15:52:25] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [21/Apr/2024 15:54:58] "POST /predict HTTP/1.1" 200 -
```

2. Fill out all the required sections for the Working of the model



Network Intrusion Detection System

Attack:

Other

Number of connections to the same destination host as the current connection in the past two seconds :

count

The percentage of connections that were to different services, among the connections aggregated in dst_host_count :

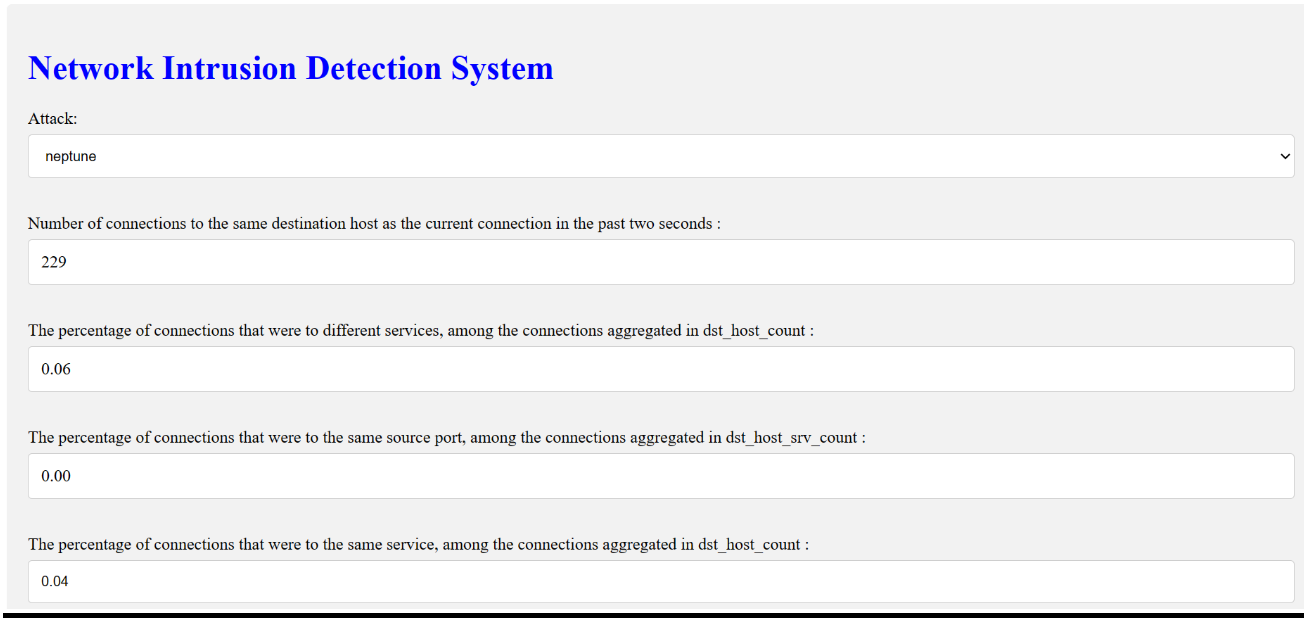
dst_host_diff_srv_rate

The percentage of connections that were to the same source port, among the connections aggregated in dst_host_srv_count :

dst_host_same_src_port_rate

The percentage of connections that were to the same service, among the connections aggregated in dst_host_count :

dst_host_same_srv_rate


Network Intrusion Detection System

Attack:

neptune

Number of connections to the same destination host as the current connection in the past two seconds :

229

The percentage of connections that were to different services, among the connections aggregated in dst_host_count :

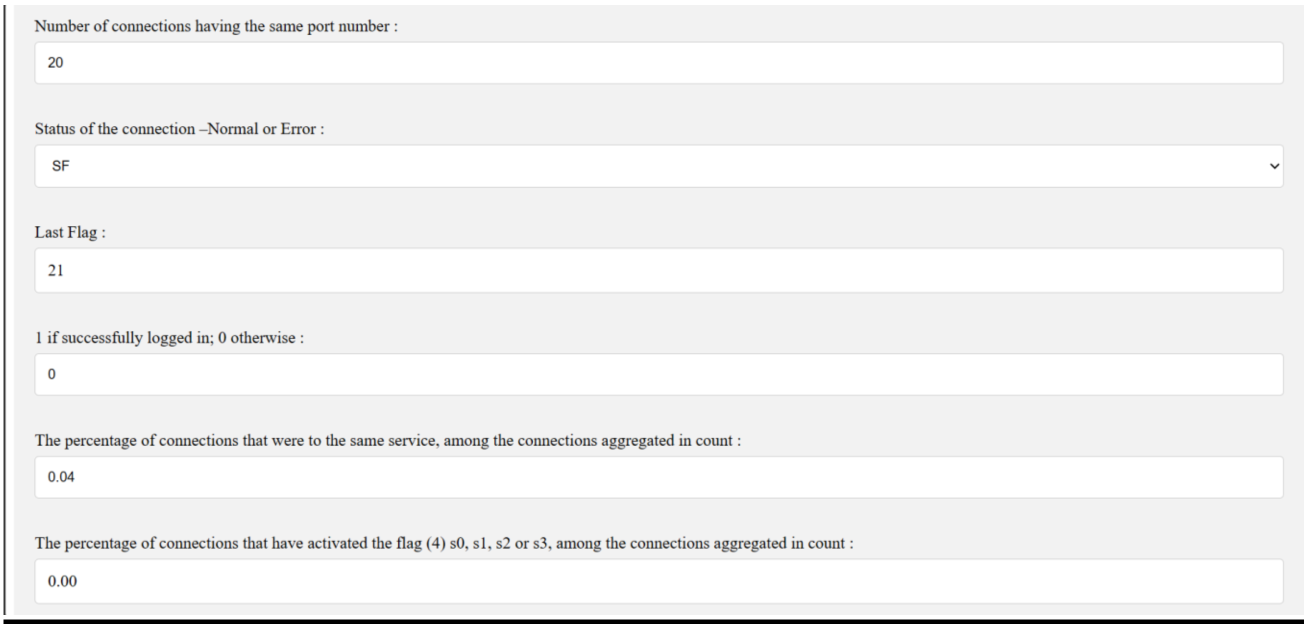
0.06

The percentage of connections that were to the same source port, among the connections aggregated in dst_host_srv_count :

0.00

The percentage of connections that were to the same service, among the connections aggregated in dst_host_count :

0.04


Number of connections having the same port number :

20

Status of the connection –Normal or Error :

SF

Last Flag :

21

1 if successfully logged in; 0 otherwise :

0

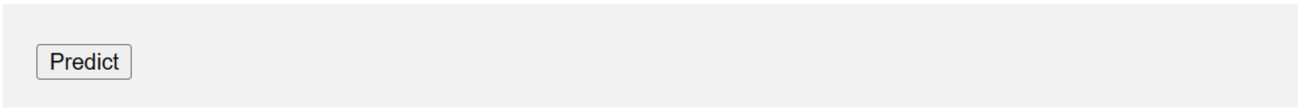
The percentage of connections that were to the same service, among the connections aggregated in count :

0.04

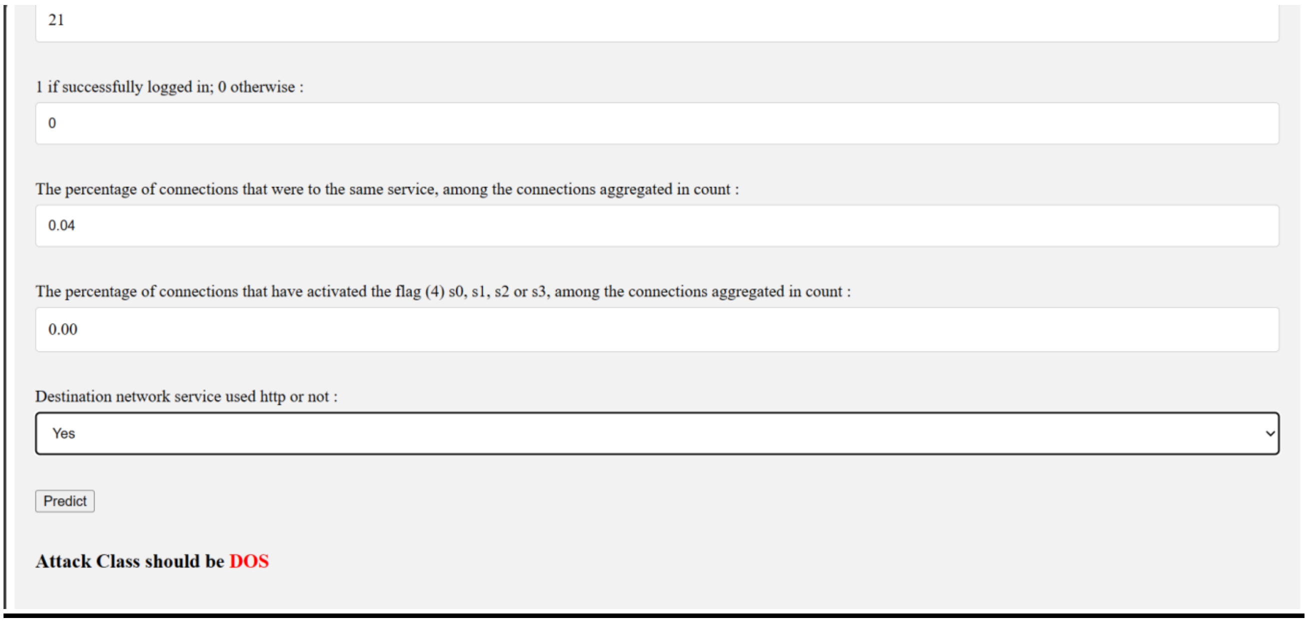
The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count :

0.00

3. Finally Click On the Predict Option



## Result


21

1 if successfully logged in; 0 otherwise :

0

The percentage of connections that were to the same service, among the connections aggregated in count :

0.04

The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count :

0.00

Destination network service used http or not :

Yes

Predict

**Attack Class should be DOS**