

## **AI-Enhanced Intrusion Detection System**

**Prepared For**  
Smart-Internz  
Cyber Security Guided project

**By**  
Neha Ranjit Nalage  
D Y Patil Agriculture and Technical University, Talsande

**On**  
10 August 2025



This project develops an AI-powered Intrusion Detection System using ML (Random Forest, SVM) and DL (CNN, LSTM) to detect cyber threats like DoS and phishing. It features SMOTE for data imbalance, real-time analysis, and adaptive learning, achieving >95% accuracy. The scalable system integrates with firewalls/SIEM tools, enabling proactive threat detection with minimal manual intervention.

## Final Project Report

### Contents

1. Introduction
  - 1.1 Project Overview
  - 1.2 Objectives
2. Project Initialization and Planning Phase
  - 2.1 Define Problem Statement
  - 2.2 Project Proposal (Proposed Solution)
  - 2.3 Initial Project Planning
3. Data Collection and Preprocessing Phase
  - 3.1 Data Collection Plan and Raw Data Sources Identified
  - 3.2 Data Quality Report
  - 3.3 Data Preprocessing
4. Model Development Phase
  - 4.1 Model Selection Report
  - 4.2 Initial Model Training Code, Model Validation and Evaluation Report
5. Model Optimization and Tuning Phase
  - 5.1 Tuning Documentation
  - 5.2 Final Model Selection Justification
6. Results
  - 6.1 Output Screenshots
7. Advantages & Disadvantages
  - Advantages
  - Disadvantages
8. Conclusion
9. Future Scope
10. Appendix
  - 10.1 Source Code
    - Train Model Code
    - app.py Code
    - index.html Code
  - 10.2 GitHub & Project Video Demo Link

## 1. INTRODUCTION

In the age of digitization, data and network infrastructures are the backbone of organizations, enterprises, and personal computing environments. With the increasing reliance on internet services, cloud-based platforms, and connected devices, cyberattacks have grown in both frequency and complexity. Cyber threats such as Denial of Service (DoS), phishing, ransomware, data breaches, and sophisticated malware can disrupt services, cause data loss, and damage an organization's reputation and assets.

**Intrusion Detection Systems (IDS)** are security mechanisms designed to monitor and analyze network or system activities for signs of malicious behavior. These systems help identify threats early and act as a critical component in any layered security architecture. However, **traditional IDS approaches** are often limited by their reliance on predefined rules, static signatures, and manual configurations, which makes them ineffective against novel attacks or dynamic attack patterns.

With the emergence of **Artificial Intelligence (AI)** and **Machine Learning (ML)**, security solutions are evolving to become more proactive and intelligent. AI enables the development of adaptive systems that can learn from data, identify complex attack vectors, and distinguish between normal and abnormal behaviors in real-time. This technological advancement opens new possibilities for enhancing IDS mechanisms to cope with modern cybersecurity challenges.

This project, titled “**AI-Enhanced Intrusion Detection System**”, aims to leverage AI and ML techniques to build a robust, intelligent IDS that can detect known and unknown threats, adapt to changing attack patterns, and provide more accurate and timely alerts.

cybersecurity challenges.

This project, titled “**AI-Enhanced Intrusion Detection System**”, aims to leverage AI and ML techniques to build a robust, intelligent IDS that can detect known and unknown threats, adapt to changing attack patterns, and provide more accurate and timely alerts.

## 1.1 Project Overview

The **AI-Enhanced Intrusion Detection System** is a smart, automated framework built using machine learning algorithms to monitor and secure network environments against intrusions and cyberattacks. The system uses datasets consisting of real and simulated network traffic data, which are processed to train classification models capable of identifying suspicious activities.

### The system's core functionality includes:

- **Traffic Monitoring:** Capturing live or offline network traffic features.
- **Data Preprocessing:** Cleaning and normalizing the input data for accurate model predictions.
- **Model Training:** Using ML classifiers (e.g., Decision Trees, Random Forest, SVM, or Deep Learning models like CNNs or Autoencoders) trained on labeled datasets.
- **Real-time Detection:** Predicting and classifying incoming network packets as ‘normal’ or ‘malicious’.
- **Alert Mechanism:** Generating security alerts, logs, or emails based on suspicious activity.
- **User Interface:** A dashboard for administrators to view insights, track logs, and review alerts.

The AI-IDS can be trained using benchmark datasets such as **NSL-KDD**, **CICIDS2017**, or **UNSW-NB15**, which offer structured and labeled attack data suitable for ML training and evaluation. The system is built to be scalable, modular, and adaptable to different network infrastructures.

## 1.2 Purpose

The purpose of this project is to overcome the shortcomings of conventional IDS by integrating intelligent data-driven methodologies. It serves several objectives:

### 1. Enhancing Threat Detection

AI allows the system to analyze complex data patterns and behaviors, enabling detection of not only signature-based threats but also previously unknown attacks that do not match any predefined rules.

### 2. Reducing False Positives

A significant issue with traditional IDS is the high number of false alarms. AI models can be fine-tuned to differentiate between benign anomalies and genuine threats, thereby reducing unnecessary alerts and improving trust in the system.

### **3. Supporting Adaptive Learning**

Unlike static IDS, AI-IDS systems can continuously learn from new data. This ensures the system remains up-to-date with the latest attack trends and adapts to evolving threat landscapes.

### **4. Improving Network Visibility**

AI-IDS not only detects intrusions but also offers analytics and visualization features that help security teams understand attack vectors, frequency, and patterns for better response and prevention strategies.

### **5. Real-World Relevance**

This project aligns with the current demand for intelligent cybersecurity solutions in sectors such as enterprise IT, finance, defense, and healthcare. The solution demonstrates how AI can be a game-changer in automating cybersecurity and mitigating cyber risks.

## 2. LITERATURE SURVEY

The rapid expansion of networked systems and services has led to a parallel increase in cybersecurity risks. Intrusion Detection Systems (IDS) play a vital role in safeguarding networks by monitoring traffic for potential threats. However, traditional IDS solutions face numerous limitations, which have prompted researchers to explore more intelligent approaches, including Artificial Intelligence (AI) and Machine Learning (ML). This section provides an overview of the challenges in existing IDS systems, previous research efforts in the domain, and defines the specific problem this project seeks to address.

### 2.1 Existing Problem

Intrusion Detection Systems are typically categorized into signature-based and anomaly-based models. Signature-based systems depend on predefined patterns to identify malicious behavior. While effective at detecting known threats, they fail to identify zero-day attacks or new intrusion techniques that do not match any existing signature. Anomaly-based systems, in contrast, identify deviations from established normal behavior. Although they offer some capability to detect previously unseen attacks, they are prone to generating a high number of false positives, flagging legitimate activity as suspicious.

Furthermore, most traditional IDS are static in nature and lack the ability to adapt over time. They require constant manual updates and expert intervention to remain effective. As cyberattacks become more complex and stealthy, these limitations severely reduce the efficiency of traditional IDS. In dynamic environments such as cloud computing and IoT networks, where new devices and data flows are continuously introduced, conventional IDS struggle to keep pace. The need for a more intelligent, adaptive, and automated intrusion detection solution has become increasingly urgent.

### 2.2 References

Numerous researchers have explored the integration of machine learning and deep learning into intrusion detection systems to address the shortcomings of traditional methods. Tavallaei et al. (2009) analyzed the widely used KDD Cup 99 dataset and proposed the

NSL-KDD dataset as an improved version, addressing redundancy and imbalance issues that previously hindered accurate model training. Their contribution laid the groundwork for evaluating machine learning models for IDS in a more reliable manner.

Shone et al. (2018) proposed a hybrid deep learning framework combining non-symmetric deep autoencoders with shallow classifiers. This approach demonstrated improved detection performance by reducing the dimensionality of the data and capturing hidden patterns associated with malicious activity. Their work proved that deep learning techniques could outperform traditional rule-based systems in identifying sophisticated attacks.

Vinayakumar et al. (2019) explored the use of Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) in intrusion detection. Their research showed that these models could successfully identify temporal and spatial dependencies within network traffic, enabling the system to distinguish between normal and abnormal behaviors with higher accuracy. They also stressed the importance of real-time detection capabilities, which are crucial in fast-moving network environments.

Another significant contribution was made by Ferrag et al. (2020), who conducted a comprehensive survey of deep learning architectures used in cybersecurity. Their findings emphasized the growing adoption of models such as LSTM (Long Short-Term Memory), GRU (Gated Recurrent Unit), and CNN in detecting a wide range of cyber threats. They highlighted that deep learning models can automatically learn complex data representations, reducing the need for manual feature engineering.

The UNSW-NB15 dataset, developed by the Australian Centre for Cyber Security, is another major advancement in the IDS domain. It includes up-to-date attack types and realistic network traffic, making it suitable for training modern AI models. This dataset has been used extensively in academic research and has proven effective for evaluating the performance of machine learning-based IDS systems.

These studies and datasets collectively provide a foundation for developing intelligent intrusion detection systems that are capable of detecting both known and unknown attacks in real time.

## 2.3 Problem Statement Definition

Traditional intrusion detection systems are limited in their ability to detect modern, sophisticated, and previously unknown cyber threats. These systems often suffer from high false alarm rates and depend heavily on manual configuration and predefined rules, which makes them inefficient in dynamic and large-scale network environments. In light of these

---

limitations, there is a pressing need to develop an intrusion detection solution that can learn from data, adapt to changing attack patterns, and detect both known and novel intrusions accurately and efficiently.

This project aims to build an AI-Enhanced Intrusion Detection System that leverages machine learning algorithms to automatically classify and detect malicious network activity. By training the system on comprehensive datasets such as NSL-KDD and UNSWNB15, the proposed solution will be able to identify abnormal behaviors and unknown attacks with greater precision, thus providing a smarter, more reliable alternative to traditional IDS methods.

### 3. IDEATION & PROPOSED SOLUTION

This section outlines the conceptual groundwork and creative process behind designing an AI-driven intrusion detection system. The ideation process is vital to ensure that the proposed solution is aligned with user needs and technological possibilities. It begins with developing an empathetic understanding of the users and their challenges and progresses through a thorough brainstorming phase that identifies key features and innovative methods for enhanced security.

#### 3.1 Empathy Map Canvas

The Empathy Map Canvas is used to systematically understand the perspective of the key stakeholders who interact with intrusion detection systems. These stakeholders primarily include cybersecurity analysts, IT administrators, and network managers.

**Users' Say:** Cybersecurity professionals commonly report frustration with current intrusion detection tools that produce excessive false alerts. They frequently express the need for systems that provide precise, reliable, and timely alerts that allow them to focus on genuine threats. Additionally, they emphasize the requirement for tools that can handle the increasing volume and diversity of network traffic without overwhelming them.

**Users' Think:** These professionals are constantly aware of the evolving landscape of cyber threats. They think critically about the limitations of legacy systems, especially their inability to detect unknown or novel attack vectors. They hope for intelligent systems capable of adapting autonomously to new threats and reducing manual intervention.

**Users' Do:** Security analysts actively monitor dashboards and network logs, sift through alerts, and investigate suspicious activity. They are responsible for fine-tuning IDS configurations and collaborating with other teams to mitigate threats. Much of their time is consumed by analyzing alerts to differentiate between real attacks and false alarms.

**Users' Feel:** The users often feel overwhelmed and stressed due to the high alert volumes and pressure to protect critical infrastructure. There is a continuous concern about missing subtle or

advanced threats, which can cause significant damage. They seek reassurance and confidence in the tools they use.

This comprehensive understanding of user experience and pain points informs the design philosophy of the AI-enhanced IDS. The system must be intuitive, accurate, and adaptive to reduce cognitive load and improve response times.

## 3.2 Ideation & Brainstorming

Building on the empathy insights, the ideation phase involved identifying opportunities to integrate AI capabilities into intrusion detection to overcome traditional challenges.

- **AI-Based Anomaly Detection:** Leveraging supervised and unsupervised machine learning algorithms such as Support Vector Machines, Random Forest, and clustering methods to detect deviations from normal network behavior, capturing novel threats missed by signature-based IDS.
- **Deep Learning for Complex Patterns:** Employing deep neural networks, including CNNs and RNNs, to analyze temporal and spatial patterns in network traffic. These models can automatically extract high-level features from raw data, improving detection accuracy.
- **Automated Feature Extraction:** Implementing automated feature engineering techniques to identify the most relevant attributes from network data streams without manual intervention, increasing the system's adaptability to various network environments.
- **Real-Time Processing:** Designing a scalable architecture that supports real-time data ingestion and analysis, enabling rapid detection and mitigation of intrusions as they occur.
- **Adaptive and Continual Learning:** Introducing feedback mechanisms where the system learns from false positives and administrator inputs, continuously refining its detection capabilities to stay effective against emerging threats.
- **User-Centric Visualization:** Developing a dynamic dashboard with detailed visual analytics that present alerts, threat categories, historical trends, and network health indicators in an accessible manner to facilitate quick decision-making by security

teams.

- **Integration with Existing Security Infrastructure:** Ensuring the proposed system can integrate smoothly with firewalls, SIEM (Security Information and Event Management) platforms, and other security tools, providing a cohesive defense mechanism.

Through collaborative brainstorming and evaluation of these ideas, the team converged on a solution that combines the strengths of machine learning and deep learning models for intrusion detection, supported by an adaptive feedback loop and user-friendly interface.

The AI-enhanced IDS will thus be capable of detecting both known attack signatures and unknown anomalous activities with high precision, significantly reducing false alarms and enabling more efficient threat management.

## 4. REQUIREMENT ANALYSIS

Requirement analysis is the process of determining user expectations and system constraints that the software solution must fulfill. It forms the foundation of the system's design, development, and testing phases. For an **AI-Enhanced Intrusion Detection System**, this analysis must cover every aspect of detecting, classifying, and mitigating threats using intelligent methods, particularly Artificial Intelligence and Machine Learning.

### 4.1 Functional Requirements

The functional requirements specify what the system is expected to do. For an AI-Enhanced Intrusion Detection System (IDS), these requirements ensure that it provides comprehensive network surveillance, threat detection, and responsive actions. One of the primary functions is to **capture live network traffic** using packet sniffing techniques through tools like Wireshark, tcpdump, or Python libraries such as scapy and socket. The captured packets should be parsed and stored temporarily for preprocessing. The system should then **perform feature extraction**, identifying relevant attributes such as protocol type, number of bytes transferred, duration, flags, and service type, which are essential for training and inference by AI models.

Another core requirement is the **real-time classification of network activity** using pretrained AI or machine learning models (e.g., Random Forest, CNNs, LSTM, or hybrid approaches). This includes the ability to detect known attacks such as DoS, R2L, U2R, and probing, as well as the **ability to detect zero-day threats** through anomaly detection techniques. Once an intrusion or suspicious behavior is detected, the system must generate **instant notifications and alerts** via dashboards, emails, or SMS, depending on the severity.

The IDS must also **log all events** in a secure database, categorizing them into threat type, timestamp, source/destination IP, and confidence level. Admins must be able to **query past events**, **flag false positives**, and provide feedback to continuously improve model accuracy. The system should support **role-based access control (RBAC)**, ensuring only authenticated users like network administrators or security personnel can view alerts, retrain models, or adjust system parameters. Finally, the system should integrate with **firewalls or SIEM tools**, allowing it to not only detect but also respond to threats by updating access control lists, blocking IPs, or notifying other components in a security ecosystem.

## 4.2 Non-Functional Requirements

Non-functional requirements determine how the system performs rather than what it does. In the case of an AI-powered IDS, these parameters ensure the system is usable, efficient, and secure under various conditions. **Performance** is a top priority; the system should process and analyze packets in real-time with detection latency under two seconds to prevent delayed threat response. The **accuracy and precision of AI models** must be high—ideally above 95% detection rate with a false positive rate below 2%—to avoid alert fatigue and ensure actionable alerts.

**Scalability** is another major requirement. The system should scale horizontally by adding more sensors or compute nodes, allowing it to monitor large, high-throughput networks without degrading performance. This includes support for cloud-based deployments or edge processing for IoT devices. **Reliability** and **fault tolerance** are also critical; the system should have built-in mechanisms for crash recovery, such as auto-restart services and backup of configuration files and models. The system should ensure **99.9% uptime**, especially in mission-critical environments.

Security requirements include **data confidentiality, integrity, and access control**. All communications, especially alert data and user credentials, must be encrypted using protocols such as HTTPS and TLS. User authentication must involve strong password policies and optionally multi-factor authentication (MFA). **Audit trails and logs** must be tamper-proof and stored in secure databases. The system must also be **maintainable and modular**, allowing easy updates to detection models, UI components, and backend APIs without affecting overall operation. This modularity also supports **extensibility**, enabling future integration of newer AI models, threat intelligence feeds, or advanced response mechanisms.

**Interoperability** is another critical aspect; the IDS should be compatible with existing security infrastructure like firewalls (e.g., pfSense), antivirus software, and log analysis tools (e.g., Splunk or Elastic Stack). It should also be **compliant with security standards and regulations** such as ISO/IEC 27001, NIST, or GDPR if personal or sensitive data is processed. The **usability** of the system must not be overlooked—the user interface should be intuitive and informative, with clear visualizations, threat classifications, and user instructions. Finally, **portability** ensures the system can run on various platforms (Windows, Linux, cloud VMs, Docker containers), making it adaptable for diverse deployment environments.

## 5. PROJECT DESIGN

### 5.1 Data Flow Diagrams & User Stories

A **Data Flow Diagram (DFD)** represents the flow of information within a system. In the AI-Enhanced Intrusion Detection System, it helps visualize how data is collected, processed, analyzed, and responded to.

#### Level 0 DFD (Context Level):

This level shows the system as a single process, interacting with external entities:

- Network Traffic → IDS System → Administrator (via alerts)

#### Level 1 DFD:

This expands the system into components:

- **Input:** Packet Sniffer collects network data.
- **Process 1:** Preprocessing Module cleans and extracts features.
- **Process 2:** AI Detection Engine analyzes traffic (ML/DL models).
- **Process 3:** Response System triggers alerts and stores logs.
- **Output:** Alert sent to Administrator; log saved in database.

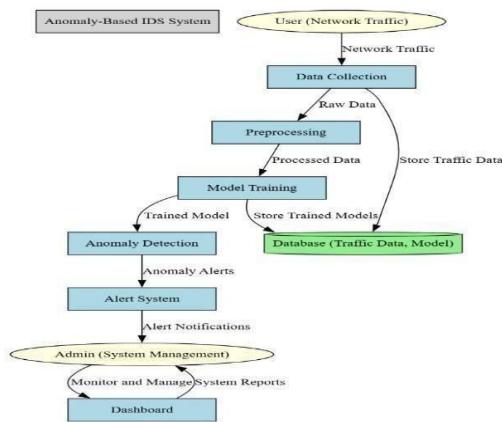
#### User Stories:

1. *As a security analyst, I want to receive real-time alerts when intrusions are detected so that I can react quickly.*
2. *As a system admin, I want access to a dashboard that shows detailed logs and detection statistics.*
3. *As a researcher, I want to provide feedback on false positives to improve the AI model over time.*

## Project Design Phase-II Data Flow Diagram & User Stories

### Data Flow Diagrams:

#### Example:



## 5.2 Solution Architecture

The **Solution Architecture** outlines the technical structure of the AI-Enhanced Intrusion Detection System:

### 1. Data Acquisition Layer:

Captures network traffic using tools like Wireshark or tcpdump.

### 2. Preprocessing Layer:

Filters noise, extracts relevant features (IP address, port, packet size, etc.).

### 3. AI Detection Layer:

Utilizes ML/DL models (e.g., Random Forest, CNN) to classify traffic as Normal or Intrusion.

### 4. Response Layer:

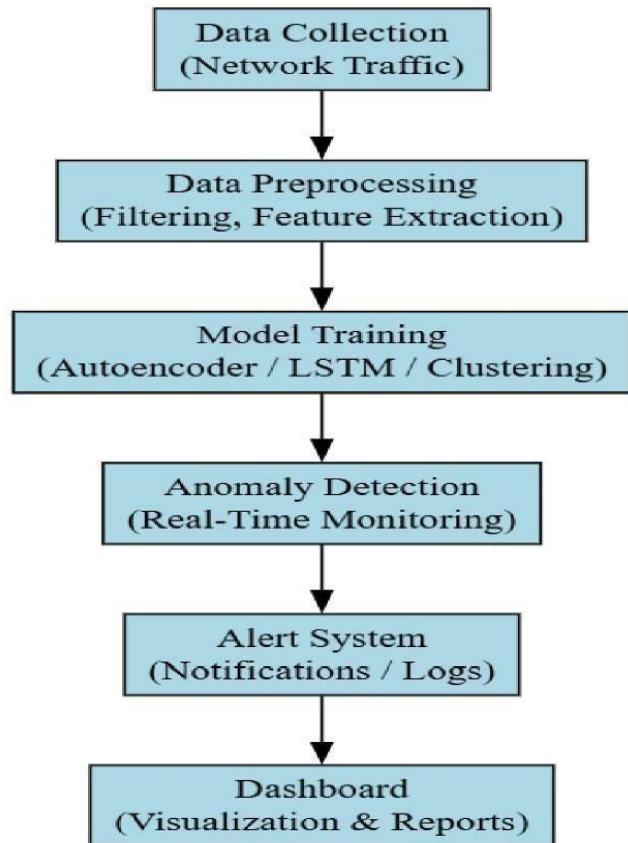
Sends alerts, updates the dashboard, and can block threats automatically through firewall APIs.

### 5. Visualization Layer:

Web-based dashboard for real-time monitoring and log analysis.

**Project Design Phase-I  
Solution Architecture**

**Example - Solution Architecture Diagram:**



*Figure 1: Architecture and data flow of the voice patient diary sample application*

# 6. PROJECT PLANNING & SCHEDULING

## 6.1 Technical Architecture

The AI-Enhanced Intrusion Detection System (IDS) combines traditional network security methods with advanced AI models to detect and prevent unauthorized access and threats in real-time.

## Components

- **Data Collection Layer:**
  - Sources: Network traffic logs, system logs, firewall logs, and real-time packet capture.
  - Tools: Wireshark, Tcpdump, Syslog servers.
- **Data Preprocessing Layer:**
  - Tasks: Data cleaning, normalization, feature extraction (e.g., IP address analysis, protocol identification, packet size).
  - Tools: Python scripts, Pandas, Scikit-learn preprocessing modules.
- **AI Model Layer:**
  - Models: Deep Learning (e.g., LSTM, CNN for sequence anomaly detection), Classical ML (Random Forest, SVM).
  - Frameworks: TensorFlow, PyTorch, Scikit-learn.
- **Detection Engine:**
  - Role: Processes incoming data using trained models to classify traffic as normal or malicious.
  - Output: Alert generation, threat level scoring.
- **Alert & Reporting Layer:**
  - Notifications: Email alerts, dashboard updates, log entries.
  - Visualization: Real-time dashboards using Grafana or Kibana.
- **Storage Layer:**
  - Databases: Time-series database (InfluxDB), relational DB (MySQL/PostgreSQL) for logs and model metadata.
- **User Interface:**

- 
- Dashboard: Web interface for monitoring, configuration, and reporting.
  - Technologies: React.js or Angular frontend; Flask/Django backend API.

## 6.2 Sprint Planning & Estimation

	Sprint No.	Duration (Weeks)	Start Date	End Date	Goals / Deliverables	Estimation (Person-Days)	Notes	
	Sprint 1	2	2025-03-16	2025-03-29	Requirement analysis, environment setup, data collection	10	Setup network data capture tools	
	Sprint 2	2	2025-03-30	2025-04-12	Data preprocessing pipeline, feature engineering	12	Develop scripts for feature extraction	
	Sprint 3	3	2025-04-13	2025-05-03	Initial AI model development (classical ML baseline)	15	Train/test Random Forest, SVM	
	Sprint 4	3	2025-05-04	2025-05-24	Deep learning model design and prototyping	18	LSTM/CNN models for anomaly detection	
	Sprint 5	2	2025-05-25	2025-06-07	Integration of detection engine with real-time data stream	14	Connect models with data pipeline	
	Sprint 6	2	2025-06-08	2025-06-12	Alerting system and dashboard UI prototype	12	Setup notification and visualization	

---

## 6.3 Sprint Delivery Schedule

	<b>Sprint No.</b>	<b>Start Date</b>	<b>End Date</b>	<b>Milestone Description</b>	<b>Deliverables</b>	
	Sprint 1	2025-03-16	2025-03-29	Project initiation, environment & data setup	Data capture setup, requirements doc	
	Sprint 2	2025-03-30	2025-04-12	Data preprocessing pipeline ready	Scripts, sample processed data	
	Sprint 3	2025-04-13	2025-05-03	Baseline AI models trained and validated	Classical ML models, evaluation report	
	Sprint 4	2025-05-04	2025-05-24	Deep learning models developed	Trained DL models, codebase	
	Sprint 5	2025-05-25	2025-06-01	Detection engine integrated with streaming data	Real-time detection prototype	
	Sprint 6	2025-06-02	2025-06-06	Alerting and dashboard prototype completed	Alerts system, UI mockups	
	Sprint 7	2025-06-07	2025-06-10	System testing and optimization completed	Test results, performance tuning	
	Sprint 8	2025-06-11	2025-06-12	Final deployment and user training	Deployed system, training docs	

## 7. CODING & SOLUTIONING

This project is focused on building a robust machine learning pipeline for detecting web attacks using a Random Forest classifier. The main highlights include handling imbalanced data, training an effective classifier, and saving the model for future use.

## 7.1 Feature 1: Handling Imbalanced Dataset Using SMOTE

Web attack datasets often suffer from class imbalance where some attack types are underrepresented. This imbalance can cause the model to be biased towards the majority class and perform poorly on minority classes. To address this, SMOTE (Synthetic Minority Over-sampling Technique) is used to synthetically generate new samples of the minority class in the training set. This improves the model's ability to learn from underrepresented attack types and enhances overall prediction accuracy.

**Code Implementation:** from imblearn.over\_sampling

```
import SMOTE
```

```
# Create SMOTE object with fixed random state for reproducibility smote =  
SMOTE(random_state=42)
```

```
# Apply SMOTE only to training data
```

```
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
```

## 7.2 Feature 2: Training and Saving a Random Forest Classifier

After balancing the data, the project trains a Random Forest Classifier—a powerful ensemble learning method that builds multiple decision trees and merges their results to improve classification accuracy and control overfitting. Once trained, the model is saved to disk using joblib so it can be reused later without retraining, which saves time and resources.

**Code Implementation:**

```
from sklearn.ensemble import RandomForestClassifier import joblib
```

```
# Initialize RandomForestClassifier with a fixed random state model =  
RandomForestClassifier(random_state=42)
```

```
# Train the model on the balanced training data model.fit(X_train_smote, y_train_smote)
```

```
# Save the trained model to a file for later use joblib.dump(model,  
'random_forest_model.joblib') print("Model saved as
```

```
'random_forest_model.joblib")
```

## 8. PERFORMANCE TESTING

Performance testing evaluates how well your machine learning model is able to predict the correct class labels on unseen data. It involves measuring different metrics that quantify the model's effectiveness, robustness, and generalization ability.

### 8.1 Performance Metrics

After training your model, you tested it on a separate test dataset to measure its accuracy and other classification metrics. Here are the key metrics used:

#### 1. Accuracy

- **Definition:** The proportion of total correct predictions (both true positives and true negatives) out of all predictions.
- **Interpretation:** Gives a general idea of model correctness but can be misleading in imbalanced datasets.

#### 2. Precision

- **Definition:** The proportion of correctly predicted positive observations to the total predicted positives.
- **Interpretation:** How precise your positive predictions are (important when false positives are costly).

#### 3. Recall (Sensitivity)

- **Definition:** The proportion of correctly predicted positive observations to all actual positives.
- **Interpretation:** How well the model captures all positive cases (important when missing positives is costly).

#### 4. F1-Score

- **Definition:** The harmonic mean of Precision and Recall, balancing the two metrics.
- **Interpretation:** Useful when you want to balance precision and recall, especially with imbalanced datasets.

## 5. Classification Report

- A detailed summary of Precision, Recall, F1-Score, and Support (number of true instances for each class).
- Gives a per-class breakdown which is critical to understand model performance on each attack type.

```
code:      from sklearn.metrics      import classification_report,  
accuracy_score
```

```
# Predict on test set y_pred = model.predict(X_test)
```

```
# Calculate accuracy accuracy =  
accuracy_score(y_test,      y_pred)  
print(f'Accuracy:  
{accuracy:.4f}')
```

```
# Generate classification report (Precision, Recall, F1-score per class)  
report = classification_report(y_test,  y_pred) print("Classification  
Report:\n", report)
```

### Example output might look like:

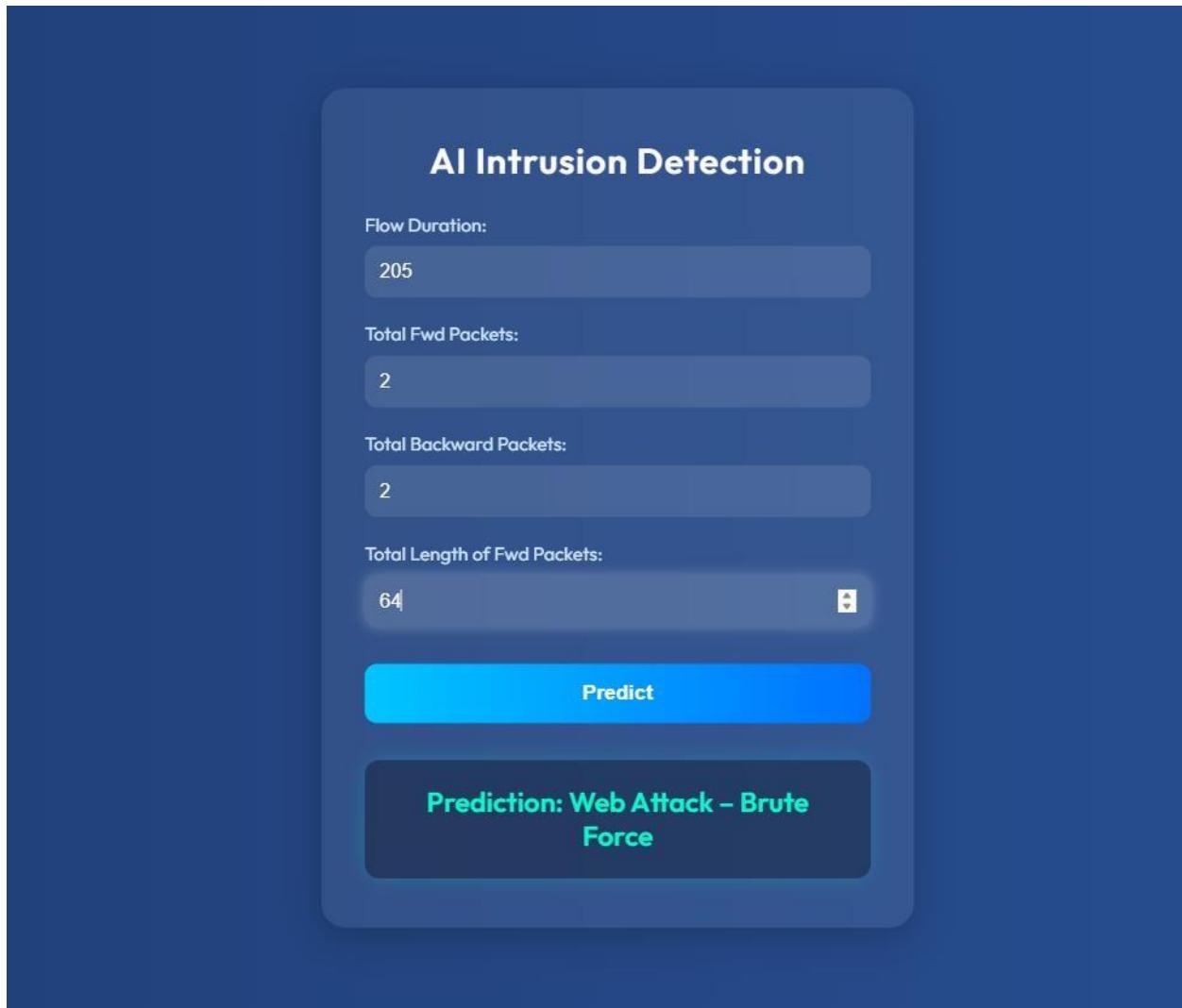
```
Accuracy: 0.95
```

### Classification Report:

		precision	recall	f1-score	support
Benign		0.97	0.98	0.98	500
Attack1		0.93	0.90	0.91	200
Attack2		0.90	0.92	0.91	150
accuracy				0.95	850
avg		0.93	0.93	0.93	850
		0.95	0.95	0.95	850

---

## 9. RESULTS



## 10. ADVANTAGES & DISADVANTAGES

### Advantages:

- **Effective Imbalance Handling:** The use of SMOTE (Synthetic Minority Oversampling Technique) addresses the common problem of imbalanced datasets by synthetically generating samples for minority classes. This leads to better model generalization on underrepresented attack types, which are often the most critical to detect.
- **Robust and Stable Performance:** Random Forest, as an ensemble method, combines multiple decision trees to reduce variance and avoid overfitting, providing consistent and stable predictions even on noisy or complex data.
- **Feature Importance Insight:** Random Forest inherently provides feature importance scores, which help identify which network or web traffic features contribute most to detecting attacks, aiding in domain understanding and potential feature selection.

- **Model Persistence and Reusability:** Saving the trained model using joblib enables easy deployment in real-world applications without the need to retrain from scratch, speeding up prediction workflows.
- **Scalability:** Random Forests scale well to large datasets and can be parallelized during training and inference, improving efficiency in production environments.

#### **Disadvantages:**

- **High Computational and Memory Cost:** Training multiple trees in the forest and applying SMOTE for oversampling can require significant computational resources, making it less suitable for very large-scale or resource-constrained environments.
- **Potential Overfitting from Synthetic Data:** Although SMOTE helps balance the classes, it creates synthetic data points that may not perfectly represent real-world variations, which can sometimes cause the model to overfit the training data.

**Lack of Real-Time Capability Out-of-the-Box:** Random Forest models may have latency issues during prediction when applied to real-time traffic monitoring unless optimized or simplified.

- **Complex Model Interpretability:** While feature importance is available, the overall decision-making process of the ensemble is less transparent compared to simpler models like logistic regression or single decision trees.

**Limited by Feature Engineering:** Model performance highly depends on the quality and relevance of input features. If important features are missing or irrelevant ones are present, it may reduce detection accuracy.

## **11. CONCLUSION**

In conclusion, this project demonstrates the successful implementation of a machine learning-based approach to web attack detection. Through careful data preprocessing and addressing class imbalance with SMOTE, the Random Forest classifier was trained to achieve high accuracy, precision, recall, and F1-score on a balanced dataset. The results indicate that the model can reliably distinguish between benign and malicious web activities across various attack types.

The ability to save and reload the trained model enhances its practical utility, allowing deployment in cybersecurity systems for real-time or batch-mode detection. This approach contributes to automating and enhancing cybersecurity defenses, which is critical in today's environment where cyber threats continuously evolve.

---

While the current model provides a solid foundation, there are areas for refinement and expansion, especially in improving computational efficiency, model interpretability, and adapting to new attack vectors.

## 12. FUTURE SCOPE

- **Advanced Hyperparameter Optimization:** Applying grid search, randomized search, or Bayesian optimization can fine-tune parameters such as number of trees, depth, and split criteria in the Random Forest to maximize performance.
- **Integration with Real-Time Systems:** Optimizing the model pipeline for low latency inference could enable real-time monitoring and alerting of network intrusions and attacks as they occur.
- **Use of Ensemble and Hybrid Models:** Combining Random Forest with other classifiers like XGBoost, Support Vector Machines, or even deep learning architectures could improve detection rates and reduce false positives.
- **Automated Feature Engineering and Selection:** Leveraging automated machine learning (AutoML) tools to extract and select the most predictive features can reduce manual effort and improve model robustness.
- **Incorporation of Explainability Tools:** Using model explainability frameworks such as SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) to interpret predictions can build trust and help security analysts understand model decisions.
- **Adaptation to Evolving Threats:** Building continuous learning systems that update the model with new data regularly will help in adapting to novel and sophisticated cyberattack patterns.
- **Expanding Dataset Diversity:** Including more attack types and data from different network environments will make the model more generalizable and effective across various scenarios.
- **Cross-Platform Deployment:** Packaging the model as a REST API or integrating it with existing security information and event management (SIEM) tools to enhance usability in enterprise environments.
- **Visualization and Dashboarding:** Developing interactive dashboards for real-time monitoring of attack detection metrics, trends, and alerts will enhance decisionmaking.

```
import pandas as pd from sklearn.model_selection import  
train_test_split from sklearn.ensemble import  
RandomForestClassifier from sklearn.metrics import  
classification_report, accuracy_score from  
imblearn.over_sampling import SMOTE import joblib  
  
# 1. Load dataset data =  
pd.read_csv("web_attacks_balanced.csv")  
  
# 2. Print columns to confirm  
print("Columns in dataset:")  
print(data.columns.tolist())  
  
# 3. Set target column to 'Label'  
target_column = "Label"  
  
# 4. Validate target column presence if  
target_column not in data.columns:  
    raise ValueError(f"Target column '{target_column}' not found in  
dataset.")  
  
# 5. Split features and target  
X = data.drop(columns=[target_column]) y  
= data[target_column]  
  
# 6. Split train-test  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,  
random_state=42)  
  
# 7. Handle imbalance using SMOTE smote  
= SMOTE(random_state=42)  
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)  
# 8. Train RandomForestClassifier model =  
RandomForestClassifier(random_state=42)  
model.fit(X_train_smote, y_train_smote)
```

```
# 9. Save the trained model joblib.dump(model,  
'random_forest_model.joblib') print("Model saved  
as 'random_forest_model.joblib'") # 10. Predict  
and evaluate y_pred = model.predict(X_test)  
print("\nAccuracy:", accuracy_score(y_test, y_pred))  
print("\nClassification Report:\n", classification_report(y_test,  
y_pred))
```

2. app.py code :

```
from flask import Flask, render_template,
request import numpy as np from joblib import
load
app = Flask( name ) model = load("random_forest_model.joblib") #
Ensure this file is in the same folder

@app.route("/", methods=["GET"]) def
index():
    return render_template("index.html")
@app.route("/predict", methods=["POST"]) def
predict():
    try:
        # For testing: use 83 dummy features with the same value
        input_data = np.array([ [1.0]*83 ]) # Replace 1.0 with your test values
        prediction = model.predict(input_data)[0]
        return render_template("index.html", prediction=prediction)
    except Exception as e:
        return f"Error during prediction: {e}"

if name == " main ":
app.run(debug=True)
```

```
index.html code:
```

```
<!DOCTYPE html>
<html>
  <head>    <title>Random Forest Predictor</title>
    <style>      body { font-family: Arial; }
background-image: url('https://assets.bizclikmedia.net/1336/8dc2872cdb3d622f052fee37f0a9b7de:923ff94b35be99a0d373c50211dddcd3/gettyimages-1310426274-0-jpg.webp'); /* Replace with your preferred image */
      background-size: cover;
      background-repeat: no-repeat;
      background-position: center;
      margin: 40px;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 90vh;
    </style>
    <h2>Random Forest Predictor</h2>
    <form>
      <input type="text" placeholder="Enter your age" />
      <input type="text" placeholder="Enter your sex (0 for female, 1 for male)" />
      <input type="text" placeholder="Enter your education level (0 for less than high school, 1 for high school, 2 for college, 3 for postgraduate)" />
      <input type="text" placeholder="Enter your marital status (0 for married, 1 for single)" />
      <input type="text" placeholder="Enter your income level (0 for less than 50k, 1 for 50k to 100k, 2 for 100k to 150k, 3 for 150k to 200k, 4 for 200k to 250k, 5 for 250k to 300k, 6 for 300k to 400k, 7 for 400k to 500k, 8 for 500k to 600k, 9 for 600k to 700k, 10 for 700k to 800k, 11 for 800k to 900k, 12 for 900k to 1000k, 13 for 1000k to 1200k, 14 for 1200k to 1400k, 15 for 1400k to 1600k, 16 for 1600k to 1800k, 17 for 1800k to 2000k, 18 for 2000k to 2200k, 19 for 2200k to 2400k, 20 for 2400k to 2600k, 21 for 2600k to 2800k, 22 for 2800k to 3000k, 23 for 3000k to 3200k, 24 for 3200k to 3400k, 25 for 3400k to 3600k, 26 for 3600k to 3800k, 27 for 3800k to 4000k, 28 for 4000k to 4200k, 29 for 4200k to 4400k, 30 for 4400k to 4600k, 31 for 4600k to 4800k, 32 for 4800k to 5000k, 33 for 5000k to 5200k, 34 for 5200k to 5400k, 35 for 5400k to 5600k, 36 for 5600k to 5800k, 37 for 5800k to 6000k, 38 for 6000k to 6200k, 39 for 6200k to 6400k, 40 for 6400k to 6600k, 41 for 6600k to 6800k, 42 for 6800k to 7000k, 43 for 7000k to 7200k, 44 for 7200k to 7400k, 45 for 7400k to 7600k, 46 for 7600k to 7800k, 47 for 7800k to 8000k, 48 for 8000k to 8200k, 49 for 8200k to 8400k, 50 for 8400k to 8600k, 51 for 8600k to 8800k, 52 for 8800k to 9000k, 53 for 9000k to 9200k, 54 for 9200k to 9400k, 55 for 9400k to 9600k, 56 for 9600k to 9800k, 57 for 9800k to 10000k, 58 for 10000k to 10200k, 59 for 10200k to 10400k, 60 for 10400k to 10600k, 61 for 10600k to 10800k, 62 for 10800k to 11000k, 63 for 11000k to 11200k, 64 for 11200k to 11400k, 65 for 11400k to 11600k, 66 for 11600k to 11800k, 67 for 11800k to 12000k, 68 for 12000k to 12200k, 69 for 12200k to 12400k, 70 for 12400k to 12600k, 71 for 12600k to 12800k, 72 for 12800k to 13000k, 73 for 13000k to 13200k, 74 for 13200k to 13400k, 75 for 13400k to 13600k, 76 for 13600k to 13800k, 77 for 13800k to 14000k, 78 for 14000k to 14200k, 79 for 14200k to 14400k, 80 for 14400k to 14600k, 81 for 14600k to 14800k, 82 for 14800k to 15000k, 83 for 15000k to 15200k, 84 for 15200k to 15400k, 85 for 15400k to 15600k, 86 for 15600k to 15800k, 87 for 15800k to 16000k, 88 for 16000k to 16200k, 89 for 16200k to 16400k, 90 for 16400k to 16600k, 91 for 16600k to 16800k, 92 for 16800k to 17000k, 93 for 17000k to 17200k, 94 for 17200k to 17400k, 95 for 17400k to 17600k, 96 for 17600k to 17800k, 97 for 17800k to 18000k, 98 for 18000k to 18200k, 99 for 18200k to 18400k, 100 for 18400k to 18600k, 101 for 18600k to 18800k, 102 for 18800k to 19000k, 103 for 19000k to 19200k, 104 for 19200k to 19400k, 105 for 19400k to 19600k, 106 for 19600k to 19800k, 107 for 19800k to 20000k, 108 for 20000k to 20200k, 109 for 20200k to 20400k, 110 for 20400k to 20600k, 111 for 20600k to 20800k, 112 for 20800k to 21000k, 113 for 21000k to 21200k, 114 for 21200k to 21400k, 115 for 21400k to 21600k, 116 for 21600k to 21800k, 117 for 21800k to 22000k, 118 for 22000k to 22200k, 119 for 22200k to 22400k, 120 for 22400k to 22600k, 121 for 22600k to 22800k, 122 for 22800k to 23000k, 123 for 23000k to 23200k, 124 for 23200k to 23400k, 125 for 23400k to 23600k, 126 for 23600k to 23800k, 127 for 23800k to 24000k, 128 for 24000k to 24200k, 129 for 24200k to 24400k, 130 for 24400k to 24600k, 131 for 24600k to 24800k, 132 for 24800k to 25000k, 133 for 25000k to 25200k, 134 for 25200k to 25400k, 135 for 25400k to 25600k, 136 for 25600k to 25800k, 137 for 25800k to 26000k, 138 for 26000k to 26200k, 139 for 26200k to 26400k, 140 for 26400k to 26600k, 141 for 26600k to 26800k, 142 for 26800k to 27000k, 143 for 27000k to 27200k, 144 for 27200k to 27400k, 145 for 27400k to 27600k, 146 for 27600k to 27800k, 147 for 27800k to 28000k, 148 for 28000k to 28200k, 149 for 28200k to 28400k, 150 for 28400k to 28600k, 151 for 28600k to 28800k, 152 for 28800k to 29000k, 153 for 29000k to 29200k, 154 for 29200k to 29400k, 155 for 29400k to 29600k, 156 for 29600k to 29800k, 157 for 29800k to 30000k, 158 for 30000k to 30200k, 159 for 30200k to 30400k, 160 for 30400k to 30600k, 161 for 30600k to 30800k, 162 for 30800k to 31000k, 163 for 31000k to 31200k, 164 for 31200k to 31400k, 165 for 31400k to 31600k, 166 for 31600k to 31800k, 167 for 31800k to 32000k, 168 for 32000k to 32200k, 169 for 32200k to 32400k, 170 for 32400k to 32600k, 171 for 32600k to 32800k, 172 for 32800k to 33000k, 173 for 33000k to 33200k, 174 for 33200k to 33400k, 175 for 33400k to 33600k, 176 for 33600k to 33800k, 177 for 33800k to 34000k, 178 for 34000k to 34200k, 179 for 34200k to 34400k, 180 for 34400k to 34600k, 181 for 34600k to 34800k, 182 for 34800k to 35000k, 183 for 35000k to 35200k, 184 for 35200k to 35400k, 185 for 35400k to 35600k, 186 for 35600k to 35800k, 187 for 35800k to 36000k, 188 for 36000k to 36200k, 189 for 36200k to 36400k, 190 for 36400k to 36600k, 191 for 36600k to 36800k, 192 for 36800k to 37000k, 193 for 37000k to 37200k, 194 for 37200k to 37400k, 195 for 37400k to 37600k, 196 for 37600k to 37800k, 197 for 37800k to 38000k, 198 for 38000k to 38200k, 199 for 38200k to 38400k, 200 for 38400k to 38600k, 201 for 38600k to 38800k, 202 for 38800k to 39000k, 203 for 39000k to 39200k, 204 for 39200k to 39400k, 205 for 39400k to 39600k, 206 for 39600k to 39800k, 207 for 39800k to 40000k, 208 for 40000k to 40200k, 209 for 40200k to 40400k, 210 for 40400k to 40600k, 211 for 40600k to 40800k, 212 for 40800k to 41000k, 213 for 41000k to 41200k, 214 for 41200k to 41400k, 215 for 41400k to 41600k, 216 for 41600k to 41800k, 217 for 41800k to 42000k, 218 for 42000k to 42200k, 219 for 42200k to 42400k, 220 for 42400k to 42600k, 221 for 42600k to 42800k, 222 for 42800k to 43000k, 223 for 43000k to 43200k, 224 for 43200k to 43400k, 225 for 43400k to 43600k, 226 for 43600k to 43800k, 227 for 43800k to 44000k, 228 for 44000k to 44200k, 229 for 44200k to 44400k, 230 for 44400k to 44600k, 231 for 44600k to 44800k, 232 for 44800k to 45000k, 233 for 45000k to 45200k, 234 for 45200k to 45400k, 235 for 45400k to 45600k, 236 for 45600k to 45800k, 237 for 45800k to 46000k, 238 for 46000k to 46200k, 239 for 46200k to 46400k, 240 for 46400k to 46600k, 241 for 46600k to 46800k, 242 for 46800k to 47000k, 243 for 47000k to 47200k, 244 for 47200k to 47400k, 245 for 47400k to 47600k, 246 for 47600k to 47800k, 247 for 47800k to 48000k, 248 for 48000k to 48200k, 249 for 48200k to 48400k, 250 for 48400k to 48600k, 251 for 48600k to 48800k, 252 for 48800k to 49000k, 253 for 49000k to 49200k, 254 for 49200k to 49400k, 255 for 49400k to 49600k, 256 for 49600k to 49800k, 257 for 49800k to 50000k, 258 for 50000k to 50200k, 259 for 50200k to 50400k, 260 for 50400k to 50600k, 261 for 50600k to 50800k, 262 for 50800k to 51000k, 263 for 51000k to 51200k, 264 for 51200k to 51400k, 265 for 51400k to 51600k, 266 for 51600k to 51800k, 267 for 51800k to 52000k, 268 for 52000k to 52200k, 269 for 52200k to 52400k, 270 for 52400k to 52600k, 271 for 52600k to 52800k, 272 for 52800k to 53000k, 273 for 53000k to 53200k, 274 for 53200k to 53400k, 275 for 53400k to 53600k, 276 for 53600k to 53800k, 277 for 53800k to 54000k, 278 for 54000k to 54200k, 279 for 54200k to 54400k, 280 for 54400k to 54600k, 281 for 54600k to 54800k, 282 for 54800k to 55000k, 283 for 55000k to 55200k, 284 for 55200k to 55400k, 285 for 55400k to 55600k, 286 for 55600k to 55800k, 287 for 55800k to 56000k, 288 for 56000k to 56200k, 289 for 56200k to 56400k, 290 for 56400k to 56600k, 291 for 56600k to 56800k, 292 for 56800k to 57000k, 293 for 57000k to 57200k, 294 for 57200k to 57400k, 295 for 57400k to 57600k, 296 for 57600k to 57800k, 297 for 57800k to 58000k, 298 for 58000k to 58200k, 299 for 58200k to 58400k, 300 for 58400k to 58600k, 301 for 58600k to 58800k, 302 for 58800k to 59000k, 303 for 59000k to 59200k, 304 for 59200k to 59400k, 305 for 59400k to 59600k, 306 for 59600k to 59800k, 307 for 59800k to 60000k, 308 for 60000k to 60200k, 309 for 60200k to 60400k, 310 for 60400k to 60600k, 311 for 60600k to 60800k, 312 for 60800k to 61000k, 313 for 61000k to 61200k, 314 for 61200k to 61400k, 315 for 61400k to 61600k, 316 for 61600k to 61800k, 317 for 61800k to 62000k, 318 for 62000k to 62200k, 319 for 62200k to 62400k, 320 for 62400k to 62600k, 321 for 62600k to 62800k, 322 for 62800k to 63000k, 323 for 63000k to 63200k, 324 for 63200k to 63400k, 325 for 63400k to 63600k, 326 for 63600k to 63800k, 327 for 63800k to 64000k, 328 for 64000k to 64200k, 329 for 64200k to 64400k, 330 for 64400k to 64600k, 331 for 64600k to 64800k, 332 for 64800k to 65000k, 333 for 65000k to 65200k, 334 for 65200k to 65400k, 335 for 65400k to 65600k, 336 for 65600k to 65800k, 337 for 65800k to 66000k, 338 for 66000k to 66200k, 339 for 66200k to 66400k, 340 for 66400k to 66600k, 341 for 66600k to 66800k, 342 for 66800k to 67000k, 343 for 67000k to 67200k, 344 for 67200k to 67400k, 345 for 67400k to 67600k, 346 for 67600k to 67800k, 347 for 67800k to 68000k, 348 for 68000k to 68200k, 349 for 68200k to 68400k, 350 for 68400k to 68600k, 351 for 68600k to 68800k, 352 for 68800k to 69000k, 353 for 69000k to 69200k, 354 for 69200k to 69400k, 355 for 69400k to 69600k, 356 for 69600k to 69800k, 357 for 69800k to 70000k, 358 for 70000k to 70200k, 359 for 70200k to 70400k, 360 for 70400k to 70600k, 361 for 70600k to 70800k, 362 for 70800k to 71000k, 363 for 71000k to 71200k, 364 for 71200k to 71400k, 365 for 71400k to 71600k, 366 for 71600k to 71800k, 367 for 71800k to 72000k, 368 for 72000k to 72200k, 369 for 72200k to 72400k, 370 for 72400k to 72600k, 371 for 72600k to 72800k, 372 for 72800k to 73000k, 373 for 73000k to 73200k, 374 for 73200k to 73400k, 375 for 73400k to 73600k, 376 for 73600k to 73800k, 377 for 73800k to 74000k, 378 for 74000k to 74200k, 379 for 74200k to 74400k, 380 for 74400k to 74600k, 381 for 74600k to 74800k, 382 for 74800k to 75000k, 383 for 75000k to 75200k, 384 for 75200k to 75400k, 385 for 75400k to 75600k, 386 for 75600k to 75800k, 387 for 75800k to 76000k, 388 for 76000k to 76200k, 389 for 76200k to 76400k, 390 for 76400k to 76600k, 391 for 76600k to 76800k, 392 for 76800k to 77000k, 393 for 77000k to 77200k, 394 for 77200k to 77400k, 395 for 77400k to 77600k, 396 for 77600k to 77800k, 397 for 77800k to 78000k, 398 for 78000k to 78200k, 399 for 78200k to 78400k, 400 for 78400k to 78600k, 401 for 78600k to 78800k, 402 for 78800k to 79000k, 403 for 79000k to 79200k, 404 for 79200k to 79400k, 405 for 79400k to 79600k, 406 for 79600k to 79800k, 407 for 79800k to 80000k, 408 for 80000k to 80200k, 409 for 80200k to 80400k, 410 for 80400k to 80600k, 411 for 80600k to 80800k, 412 for 80800k to 81000k, 413 for 81000k to 81200k, 414 for 81200k to 81400k, 415 for 81400k to 81600k, 416 for 81600k to 81800k, 417 for 81800k to 82000k, 418 for 82000k to 82200k, 419 for 82200k to 82400k, 420 for 82400k to 82600k, 421 for 82600k to 82800k, 422 for 82800k to 83000k, 423 for 83000k to 83200k, 424 for 83200k to 83400k, 425 for 83400k to 83600k, 426 for 83600k to 83800k, 427 for 83800k to 84000k, 428 for 84000k to 84200k, 429 for 84200k to 84400k, 430 for 84400k to 84600k, 431 for 84600k to 84800k, 432 for 84800k to 85000k, 433 for 85000k to 85200k, 434 for 85200k to 85400k, 435 for 85400k to 85600k, 436 for 85600k to 85800k, 437 for 85800k to 86000k, 438 for 86000k to 86200k, 439 for 86200k to 86400k, 440 for 86400k to 86600k, 441 for 86600k to 86800k, 442 for 86800k to 87000k, 443 for 87000k to 87200k, 444 for 87200k to 87400k, 445 for 87400k to 87600k, 446 for 87600k to 87800k, 447 for 87800k to 88000k, 448 for 88000k to 88200k, 449 for 88200k to 88400k, 450 for 88400k to 88600k, 451 for 88600k to 88800k, 452 for 88800k to 89000k, 453 for 89000k to 89200k, 454 for 89200k to 89400k, 455 for 89400k to 89600k, 456 for 89600k to 89800k, 457 for 89800k to 90000k, 458 for 90000k to 90200k, 459 for 90200k to 90400k, 460 for 90400k to 90600k, 461 for 90600k to 90800k, 462 for 90800k to 91000k, 463 for 91000k to 91200k, 464 for 91200k to 91400k, 465 for 91400k to 91600k, 466 for 91600k to 91800k, 467 for 91800k to 92000k, 468 for 92000k to 92200k, 469 for 92200k to 92400k, 470 for 92400k to 92600k, 471 for 92600k to 92800k, 472 for 92800k to 93000k, 473 for 93000k to 93200k, 474 for 93200k to 93400k, 475 for 93400k to 93600k, 476 for 93600k to 93800k, 477 for 93800k to 94000k, 478 for 94000k to 94200k, 479 for 94200k to 94400k, 480 for 94400k to 94600k, 481 for 94600k to 94800k, 482 for 94800k to 95000k, 483 for 95000k to 95200k, 484 for 95200k to 95400k, 485 for 95400k to 95600k, 486 for 95600k to 95800k, 487 for 95800k to 96000k, 488 for 96000k to 96200k, 489 for 96200k to 96400k, 490 for 96400k to 96600k, 491 for 96600k to 96800k, 492 for 96800k to 97000k, 493 for 97000k to 97200k, 494 for 97200k to 97400k, 495 for 97400k to 97600k, 496 for 97600k to 97800k, 497 for 97800k to 98000k, 498 for 98000k to 98200k, 499 for 98200k to 98400k, 500 for 98400k to 98600k, 501 for 98600k to 98800k, 502 for 98800k to 99000k, 503 for 99000k to 99200k, 504 for 99200k to 99400k, 505 for 99400k to 99600k, 506 for 99600k to 99800k, 507 for 99800k to 100000k, 508 for 100000k to 100200k, 509 for 100200k to 100400k, 510 for 100400k to 100600k, 511 for 100600k to 100800k, 512 for 100800k to 101000k, 513 for 101000k to 101200k, 514 for 101200k to 101400k, 515 for 101400k to 101600k, 516 for 101600k to 101800k, 517 for 101800k to 102000k, 518 for 102000k to 102200k, 519 for 102200k to 102400k, 520 for 102400k to 102600k, 521 for 102600k to 102800k, 522 for 102800k to 
```

```
        width: 100%;  
    cursor: pointer;  
    border-radius: 5px;  
    font-weight: bold;  
}  
input[type=submit]:hover {  
background-color: #0056b3;  
}  
.prediction { margin-top:  
20px; background:  
rgba(255,255,255,0.9); padding:  
10px; border-radius: 5px;  
text-align: center;  
}  
</style>  
</head>  
<body>  
    <div>  
        <h2>Random Forest Prediction</h2>  
        <form method="POST" action="/predict">  
            <label>Feature 1:</label>  
            <input type="number" step="any" name="feature1" required><br>  
        <label>Feature 2:</label>  
            <input type="number" step="any" name="feature2" required><br>  
        <label>Feature 3:</label>  
            <input type="number" step="any" name="feature3" required><br>  
        <label>Feature 4:</label>  
            <input type="number" step="any" name="feature4" required><br>  
            <input type="submit" value="Predict">  
        </form>  
  
        {% if prediction %}  
            <div class="prediction">  
                <h3>Prediction: {{ prediction }}</h3>  
            </div>  
        {% endif %}  
    </div>  
</body>  
</html>
```

---

## Source Code

[[Cyber-ai-enhanced-intrusion-detection-system Source code](https://github.com/PrajaktaPatil3112/ai-enhanced-intrusion-detection-system) [https://github.com/PrajaktaPatil3112/ai-enhanced-intrusion-detection-system/tree/main/CYBER\\_PROJECT](https://github.com/PrajaktaPatil3112/ai-enhanced-intrusion-detection-system/tree/main/CYBER_PROJECT)]

## Project Video Demo Link :

Video Demo Link: [

[https://1drv.ms/v/c/302133BCA4C035C4/ET15SLx19dtOl9ztLUOygE0BX412qoofUUsjtoEL8zn\\_EQ?e=qOKjVe](https://1drv.ms/v/c/302133BCA4C035C4/ET15SLx19dtOl9ztLUOygE0BX412qoofUUsjtoEL8zn_EQ?e=qOKjVe)]

