

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

ABSTRACT

Class imbalance is a critical issue in network traffic classification, where dominant classes often overshadow minority ones, leading to biased and inaccurate predictions. This project proposes an adaptive, ensemble-based solution that combines a Stacking Classifier with Explainable AI (XAI) techniques to address the imbalance while maintaining interpretability. The system intelligently integrates multiple base learners into a stacking framework, enhancing predictive performance across all traffic classes, including rare and underrepresented ones. In addition, the integration of XAI methods provides transparency by highlighting feature contributions and model decision rationale—essential for security-sensitive environments. Developed using Python and deployed with a Flask-based web interface, the system enables real-time traffic analysis with intuitive visualizations through HTML, CSS, and JavaScript. This adaptive and interpretable approach enhances the reliability and trustworthiness of automated network monitoring solutions.

Keywords: Network Traffic Classification, Class Imbalance, Stacking Classifier, Explainable AI, Ensemble Learning, XAI, Flask, Real-Time Analysis, Cybersecurity.

INDEX

Contents

CHAPTER 1 – INTRODUCTION	1
CHAPTER 2 – SYSTEM ANALYSIS.....	2
2.1 Existing System.....	2
2.2 Proposed System.....	2
CHAPTER 3 – FEASIBILITY STUDY	4
Feasibility Study.....	7
3.1 Technical Feasibility.....	7
3.2 Operational Feasibility.....	7
3.3 Economic Feasibility.....	8
CHAPTER 4 – SYSTEM REQUIREMENT SPECIFICATION DOCUMENT	9
CHAPTER 5 – SYSTEM DESIGN	18
a. DFD.....	18
b. ER diagram.....	19
c. UML.....	20
d. Data Dictionary.....	23
CHAPTER 6-TECHNOLOGY DESCRIPTION	25
CHAPTER 7 – TESTING & DEBUGGING TECHNIQUES	29
CHAPTER 8 – OUTPUT SCREENS	33
CHAPTER 9 -CODE	36
CHAPTER 10 – CONCLUSION.....	43
CHAPTER 11 – BIBLIOGRAPHY	44

CHAPTER 1 – INTRODUCTION

In the current digital age, the proliferation of internet-connected devices and the rapid growth of online services have significantly increased the volume and complexity of network traffic. This surge has led to a heightened risk of cyber threats, making Network Traffic Classification (NTC) a vital component of modern cybersecurity frameworks. NTC enables the identification and categorization of network traffic based on various features, helping detect anomalies, manage bandwidth efficiently, and enforce security policies proactively.

Traditional approaches to traffic classification—such as port-based and payload-based methods—are increasingly ineffective due to dynamic port assignments, encryption, and traffic obfuscation techniques. These limitations have driven the adoption of Machine Learning (ML) and Deep Learning (DL) techniques, which offer automated, data-driven methods capable of uncovering complex patterns within traffic data. Among these, supervised learning algorithms have shown considerable promise in distinguishing between benign and malicious traffic types using labeled datasets.

However, a persistent issue in network traffic datasets is class imbalance, where a large portion of the data represents normal or benign traffic, and only a small fraction corresponds to malicious activities. This imbalance skews model performance, causing classifiers to favor the majority class and perform poorly on the minority—often the most security-critical—traffic types. Therefore, building a model that performs well across all classes, especially minority ones, is a critical requirement for effective intrusion detection and prevention.

To address this challenge, this project introduces a Stacking Classifier framework enhanced with Explainable AI (XAI) techniques. By combining diverse base learners and a meta-learner, the stacking ensemble improves generalization and mitigates bias from imbalanced data.

CHAPTER 2 – SYSTEM ANALYSIS

This section presents a comprehensive analysis of both the existing and the proposed systems for network traffic classification, specifically focusing on the UNSW-NB15 dataset. The discussion highlights the limitations of conventional approaches and outlines how the proposed Stacking Classifier integrated with Explainable AI (XAI) enhances classification performance and model transparency.

2.1 Existing System

Traditional Network Traffic Classification (NTC) systems rely heavily on single-layered machine learning models such as Decision Trees, Support Vector Machines, and Naive Bayes. While these models are relatively easy to implement and interpret, they often fall short when applied to real-world traffic datasets, particularly those with class imbalance and high-dimensional features like UNSW-NB15.

2.1.1 Challenges in Existing Systems

- **Class Imbalance:** The UNSW-NB15 dataset contains a significant imbalance between benign and attack classes. Traditional models tend to overfit the majority class, leading to poor detection of minority classes, such as worms or backdoors.
- **Limited Generalization:** Single models often lack the robustness required to generalize well across diverse and evolving traffic patterns.
- **Lack of Explainability:** Most traditional classifiers offer limited insight into how decisions are made, hindering user trust and making model validation difficult in security-critical environments.

2.2 Proposed System

The proposed system introduces a hybrid architecture that combines the strengths of ensemble learning and explainable artificial intelligence (XAI). Specifically, it utilizes a Stacking Classifier composed of heterogeneous base learners and a meta-learner, alongside SHAP (SHapley Additive exPlanations) for interpretability.

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

2.2.1 Key Components of the Proposed System

1. Stacking Classifier:

- Combines Decision Tree, Random Forest, and XGBoost as base learners.
- Employs a Logistic Regression or XGBoost meta-learner to refine predictions.
- Enhances generalization and reduces overfitting.

2. Explainable AI (XAI):

- Integrates SHAP to quantify the contribution of each feature to model predictions.
- Helps analysts understand and validate classification decisions.

3. Feature Engineering and Preprocessing:

- Normalization and encoding techniques prepare raw network traffic data.
- Feature selection strategies reduce dimensionality and focus the model on relevant indicators.

4. Model Evaluation:

- Performance is assessed using Accuracy, Precision, Recall, and F1-Score.
- Confusion matrices are analyzed to understand class-specific performance.

5. Handling Class Imbalance:

- Uses techniques like class weighting, SMOTE, or ensemble-based re-balancing within the stacking framework.

2.2.2 Advantages of the Proposed System

- **Improved Minority Class Detection:** The ensemble architecture, coupled with re-balancing methods, ensures better identification of underrepresented attack types.
- **Model Transparency:** SHAP explanations provide actionable insights into model behavior, promoting trust and compliance.
- **Robust Performance:** The stacking approach offers increased accuracy and resilience to noise and irrelevant features.
- **Scalability and Adaptability:** The modular design supports easy integration of new classifiers and adaptation to other traffic datasets or environments.

CHAPTER 3 – FEASIBILITY STUDY

In the era of ubiquitous connectivity and growing cyber threats, Network Traffic Classification (NTC) has emerged as an essential pillar in the domain of cybersecurity. It enables systems to identify and analyze patterns in data traffic across networks, allowing for better anomaly detection, efficient network management, and improved security policies. As the volume and complexity of network traffic continue to grow, so too does the need for intelligent systems capable of accurately classifying various types of network behavior, including both benign and malicious activities.

Traditionally, NTC methods have relied on port-based or payload-based techniques. Port-based classification uses well-known port numbers to infer application types, while payload-based methods inspect packet content to determine traffic categories. However, these techniques face limitations in modern networks due to port obfuscation, dynamic port allocation, and widespread use of encryption, making it increasingly difficult to rely solely on such heuristic methods. As a result, the research community and industry practitioners have turned to machine learning (ML) and deep learning (DL) methods to automate and enhance the accuracy of traffic classification.

Machine learning offers a data-driven approach that can identify subtle and complex relationships within network traffic data. Supervised learning algorithms, in particular, have shown great promise in NTC, where labeled data can be used to train models to distinguish between various types of traffic. Popular models such as Decision Trees, Random Forests, and XGBoost have been applied to this domain with considerable success. However, one persistent challenge in deploying these models in real-world scenarios is the problem of class imbalance.

Class imbalance refers to the disproportionate representation of different traffic classes within a dataset. In network traffic, this often manifests as a majority of normal or benign traffic with only a small fraction of anomalous or attack traffic. Models trained on such skewed data tend to be biased toward the majority class, resulting in high overall accuracy but poor performance in detecting minority classes, which are often the most critical from a security perspective. This

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

issue significantly hampers the reliability and trustworthiness of conventional ML approaches in security-sensitive environments.

To overcome the limitations of traditional models in handling class imbalance, ensemble methods have gained popularity. Ensemble learning combines multiple classifiers to produce a more accurate and robust prediction. Among the various ensemble techniques, stacking has shown particular promise due to its ability to integrate multiple base learners with a meta-learner that synthesizes their outputs. The base learners capture different patterns in the data, while the meta-learner optimizes the final prediction by learning how best to combine these insights. This layered learning structure helps reduce overfitting and enhances generalization, making stacking classifiers highly suitable for complex, imbalanced datasets like UNSW-NB15.

The UNSW-NB15 dataset, developed by the Australian Centre for Cyber Security, is one of the most comprehensive and widely used benchmarks in network intrusion detection research. It captures a variety of attack types, including DoS, worms, backdoors, and exploits, along with a large amount of benign traffic. While the dataset provides a rich ground for model training and evaluation, it is also highly imbalanced, which makes it an ideal candidate for testing robust classification strategies that can generalize well to real-world data distributions.

In addition to improving predictive performance, modern NTC systems must also offer transparency and explainability. In critical domains like cybersecurity, understanding the rationale behind model decisions is crucial for gaining user trust and supporting timely and informed responses to potential threats. Explainable Artificial Intelligence (XAI) addresses this need by providing tools and methods that reveal how machine learning models arrive at their predictions. One of the most effective XAI methods is SHAP (SHapley Additive exPlanations), which assigns importance values to each feature based on its contribution to the model's output.

SHAP is grounded in game theory and provides consistent and locally accurate explanations for individual predictions. By using SHAP, security analysts can gain insights into why a certain traffic flow was classified as malicious, which features contributed most to the decision, and how changes in input features could affect the outcome. This level of interpretability not

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

only enhances model transparency but also aids in model validation, debugging, and compliance with regulatory standards.

In this study, we propose a novel framework that integrates a Stacking Classifier with SHAP-based explainability to address the dual challenges of class imbalance and model interpretability in network traffic classification. Our proposed system employs a heterogeneous ensemble of base classifiers (including Decision Tree, Random Forest, and XGBoost) and a meta-classifier that refines their combined output. We use the UNSW-NB15 dataset to train and evaluate our model, ensuring that it is tested on realistic and challenging traffic scenarios.

The contributions of our research are threefold:

1. We introduce a robust stacking ensemble architecture tailored for handling imbalanced network traffic data.
2. We incorporate SHAP explanations to provide interpretability and transparency into the model's predictions, facilitating better understanding and trust in automated decisions.
3. We conduct comprehensive experiments using the UNSW-NB15 dataset, evaluating the model's performance across various metrics including accuracy, precision, recall, and F1-score, with a focus on minority class detection.

Our findings demonstrate that the proposed framework not only achieves high classification accuracy but also significantly improves the detection of minority classes compared to individual models and traditional ensemble methods. Furthermore, the integration of XAI components ensures that the model's decisions can be trusted and acted upon by network administrators and security professionals.

In the sections that follow, we present a detailed review of related work, describe the architecture of the proposed system, outline the experimental setup, and discuss the results and implications of our study. By bridging the gap between performance and explainability, our work contributes to the development of next-generation NTC systems that are not only intelligent but also accountable and secure.

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

Feasibility Study

3.1 Technical Feasibility

The proposed Stacking Classifier + XAI framework is technically feasible due to the availability of powerful ensemble algorithms, explainability tools, and scalable cloud-based infrastructure. The architecture combines widely-used ML algorithms (Decision Tree, Random Forest, XGBoost) with a meta-classifier, optimized for imbalanced data like UNSW-NB15. The SHAP tool provides intuitive insights into feature importance, aiding transparency and debugging.

The system is developed using Python, leveraging libraries such as scikit-learn, XGBoost, and SHAP. Backend logic and API routes are implemented using Flask, making the system modular and easy to integrate with web-based dashboards. Training is performed on GPU-enabled environments via Google Colab or AWS EC2 instances, while data storage can utilize MySQL or NoSQL databases for storing model logs and prediction results. The modular design allows for the addition of new base learners, integration of new datasets, or switching to alternative XAI techniques like LIME. Hence, the technical infrastructure is robust, well-supported, and adaptable to future needs.

3.2 Operational Feasibility

The system is operationally feasible and user-friendly, featuring a web interface where users can upload network traffic data, visualize SHAP explanations, and view classification results in real-time. Designed with non-expert users in mind (e.g., security analysts, researchers), it includes guided steps for data input, model selection, and output interpretation.

The system supports integration with Security Information and Event Management (SIEM) tools, making it suitable for deployment in enterprise cybersecurity settings. Its real-time classification and explanation capabilities provide actionable insights, reducing the time-to-response during threat detection.

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

Documentation, demo tutorials, and optional training modules can assist users in system navigation. As the backend is built on lightweight Flask APIs and front-end components use Bootstrap, it can be hosted locally or on cloud platforms like Heroku or Render.

3.3 Economic Feasibility

The use of open-source technologies significantly reduces development and maintenance costs. Python, Flask, scikit-learn, XGBoost, and SHAP are all free and widely supported. Dataset access is also free via the UNSW-NB15 public repository.

Cloud-based deployment follows a pay-as-you-go model. Since training occurs periodically and real-time predictions are computationally light, resource usage remains within budget. Hosting costs on services like AWS, Render, or DigitalOcean are minimal compared to traditional enterprise tools.

The return on investment (ROI) is high due to increased threat detection accuracy, early warning of network anomalies, and improved decision-making. The explainability factor builds user trust, reducing reliance on black-box models and promoting system adoption. For organizations, this system offers cost-effective, interpretable, and scalable NTC.

In conclusion, the proposed system is technically sound, operationally adaptable, and economically viable for deployment in modern cybersecurity infrastructures.

CHAPTER 4 – SYSTEM REQUIREMENT SPECIFICATION DOCUMENT

This section provides a detailed specification for the **Network Traffic Classification and Risk Detection System** using machine learning and explainable AI (XAI) techniques. The system leverages ensemble learning models, specifically a stacking classifier architecture (comprising Decision Trees, Random Forest, XGBoost, and Logistic Regression as meta-learner), and incorporates SHAP for model interpretability. The solution is designed to work on the UNSW-NB15 dataset and to assist security analysts in real-time anomaly detection, particularly addressing the challenge of class imbalance.

4.1 Overview

The proposed system classifies network traffic into benign or various attack types (e.g., DoS, Exploits, Backdoors) based on flow-level data. It processes traffic data, extracts features, and uses machine learning algorithms to predict network behaviors. Key features of the system include:

Stacking Classifier for robust prediction.

SMOTE to handle class imbalance in traffic data.

Explainable AI (SHAP) to interpret model predictions.

Real-time classification and alerting through a web-based dashboard.

Evaluation with accuracy, precision, recall, F1-score.

The system is intended to improve detection accuracy, reduce false positives/negatives, and increase trust in machine-generated outputs for security-critical environments.

4.2 Module Description

Data Acquisition Module:

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

Purpose: To import UNSW-NB15 dataset or live traffic data feeds.

Input: CSV, JSON, or live network logs.

Output: Raw traffic records.

Preprocessing and Feature Engineering Module:

Purpose: Clean, normalize, and balance data.

Input: Raw traffic data.

Output: Balanced and formatted feature set.

Feature Selection Module:

Purpose: Reduce dimensionality using chi-squared or mutual information scores.

Input: Engineered features.

Output: Top N features for training.

Classification Module:

Purpose: Stack base classifiers (Decision Tree, RF, XGBoost) with a Logistic Regression meta-learner.

Input: Selected features.

Output: Predicted traffic class labels.

XAI Module (Explainability):

Purpose: Interpret predictions using SHAP.

Input: Model predictions and feature values.

Output: SHAP plots and contribution scores.

Evaluation and Reporting Module:

Purpose: Evaluate models using standard metrics and generate downloadable reports.

Input: Prediction labels and ground truth.

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

Output: Accuracy, precision, recall, F1-score, confusion matrix.

User Interface Module:

Purpose: Interface for data upload, visualization, and analysis review.

Input: User-uploaded logs or API inputs.

Output: Charts, tables, downloadable insights.

4.3 Process Flow

Data Collection: Ingest UNSW-NB15 dataset or live data stream.

Data Preprocessing: Clean, normalize, and apply SMOTE for balancing.

Feature Engineering: Generate and select optimal features.

Model Training: Train the stacking ensemble on the selected features.

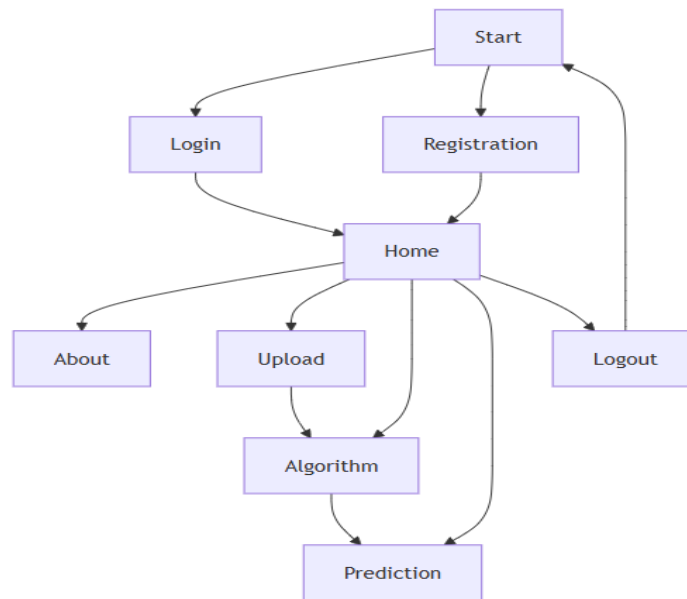
Prediction: Classify network traffic in real time.

XAI Interpretability: Apply SHAP to explain predictions.

Performance Evaluation: Calculate evaluation metrics.

Dashboard Display: Present real-time visual feedback to the user.

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method



4.5 SDLC Methodology

SOFTWARE DEVELOPMENT LIFE CYCLE

The meaning of Agile is swift or versatile. "Agile process model" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance. Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

Actually, Agile model refers to a group of development processes. These processes share some basic characteristics but do have certain subtle differences among themselves. A few Agile SDLC models are given below: Crystal A tern Feature-driven development Scrum

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

Extreme programming (XP) Lean development Unified process In the Agile model, the requirements are decomposed into many small parts that can be incrementally developed.

The Agile model adopts Iterative development. Each incremental part is developed over an iteration. Each iteration is intended to be small and easily manageable and that can be completed within a couple of weeks only. At a time one iteration is planned, developed and deployed to the customers. Long-term plans are not made.

Agile model is the combination of iterative and incremental process models. Steps involve in agile SDLC models are:

Requirement gathering

Requirement Analysis

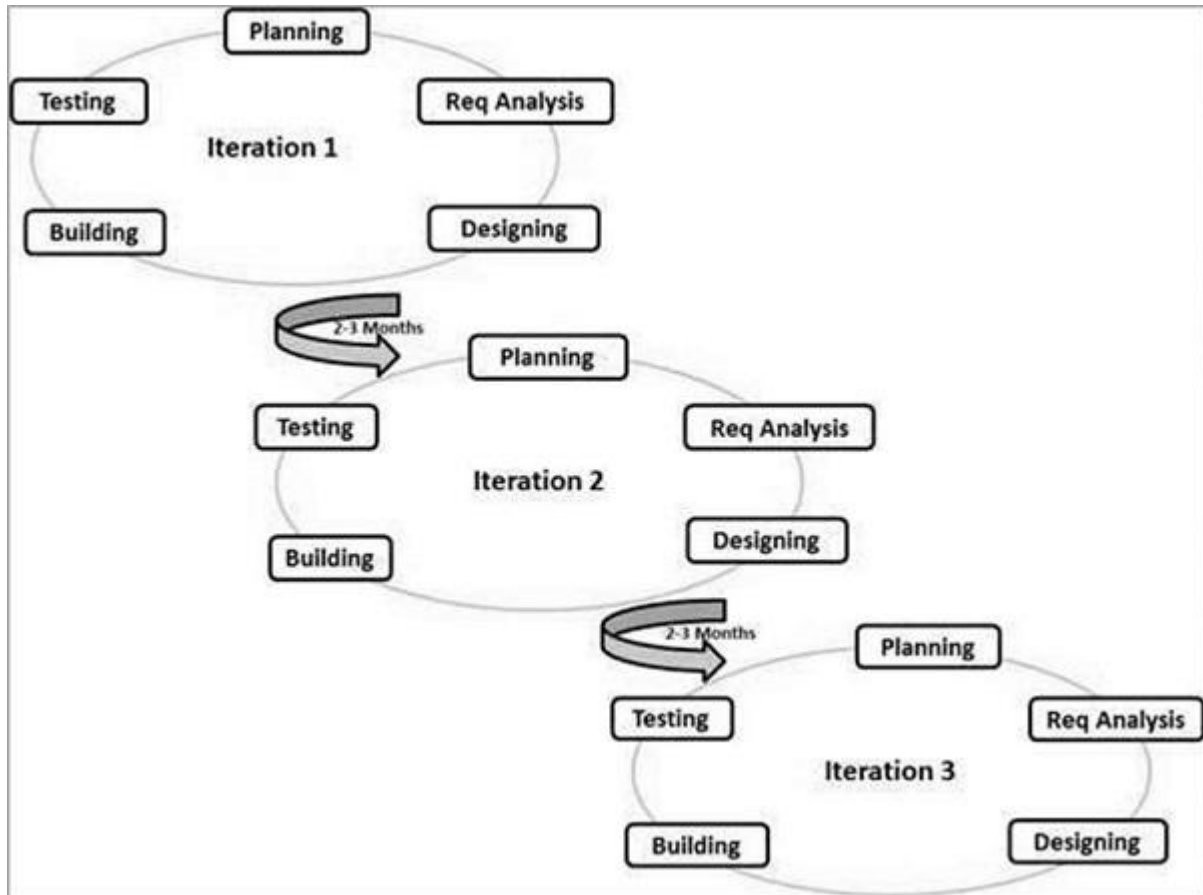
Design Coding

Unit testing

Acceptance testing

The time to complete an iteration is known as a Time Box. Time-box refers to the maximum amount of time needed to deliver an iteration to customers. So, the end date for an iteration does not change. Though the development team can decide to reduce the delivered functionality during a Time-box if necessary to deliver it on time. The central principle of the Agile model is the delivery of an increment to the customer after each Time-box.

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method



Principles of Agile model:

To establish close contact with the customer during development and to gain a clear understanding of various requirements, each Agile project usually includes a customer representative on the team. At the end of each iteration stakeholders and the customer representative review, the progress made and re-evaluate the requirements.

Agile model relies on working software deployment rather than comprehensive documentation.

Frequent delivery of incremental versions of the software to the customer representative in intervals of few weeks.

Requirement change requests from the customer are encouraged and efficiently incorporated.

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

Advantages:

Working through Pair programming produces well written compact programs which has fewer errors as compared to programmers working alone.

It reduces total development time of the whole project. Customer representatives get the idea of updated software products after each iteration. So, it is easy for him to change any requirement if needed.

Disadvantages:

Due to lack of formal documents, it creates confusion and important decisions taken during different phases can be misinterpreted at any time by different team members.

Due to the absence of proper documentation, when the project completes and the developers are assigned to another project, maintenance of the developed project can become a problem.

SOFTWARE DEVELOPMENT LIFE CYCLE – SDLC:

In our project we use waterfall model as our software development cycle because of its step-by-step procedure while implementing.

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

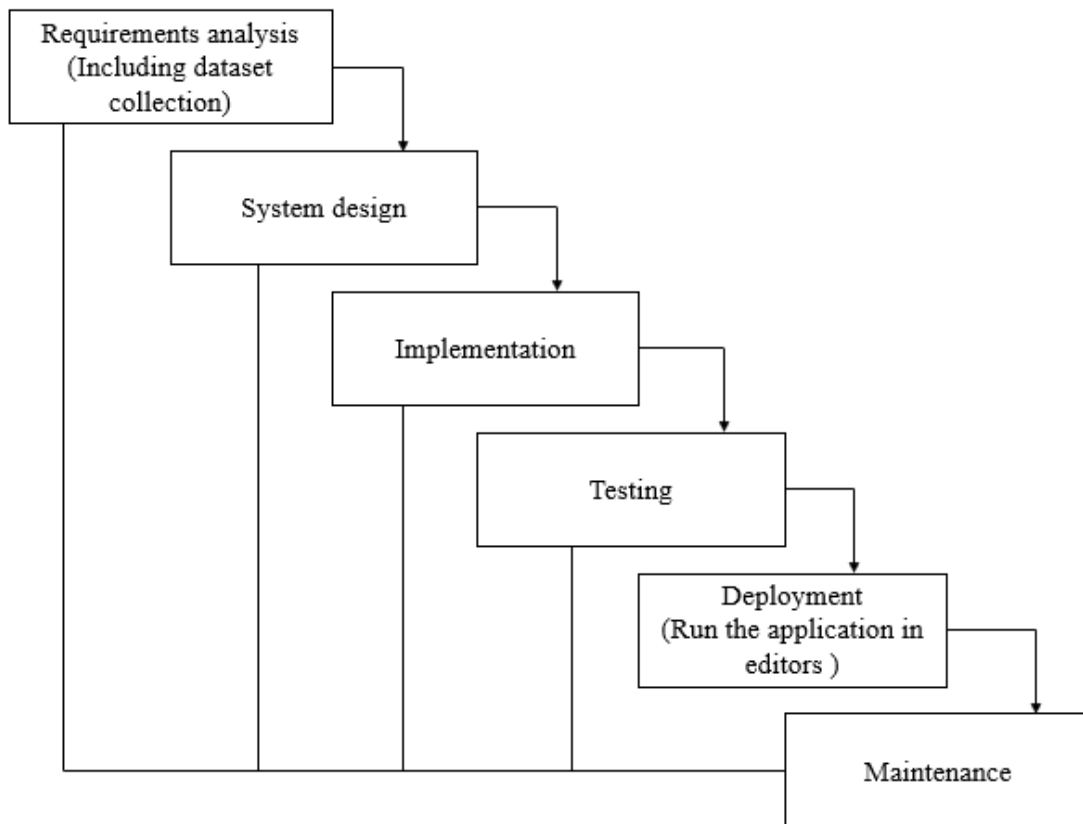


Fig1: Waterfall Model

Requirement Gathering and analysis – All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

System Design – The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

Implementation – With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

Integration and Testing – All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

Deployment of system – Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

4.6 Software Requirements

Operating System: Windows/Linux/macOS

Programming Language: Python

Libraries and Frameworks:

Machine Learning: scikit-learn, XGBoost, imbalanced-learn

Explainability: SHAP

Data Handling: Pandas, NumPy

Web Framework: Flask

IDE: Jupyter Notebook, VS Code, or PyCharm

Database: MySQL or MongoDB (for logs and results)

4.7 Hardware Requirements

Processor: Intel i5/i7 or equivalent

RAM: 8GB or higher

Storage: Minimum 256GB

GPU: Recommended for faster training (e.g., NVIDIA GTX 1060 or better)

CHAPTER 5 – SYSTEM DESIGN

System design is a crucial phase in software development that transforms the system requirements into detailed blueprints for implementation. It ensures that each component is structured, well-organized, and capable of fulfilling its intended role efficiently. The proposed system, which predicts energy consumption using machine learning models based on network parameters, involves multiple user interactions, database operations, and backend computations. This chapter describes the system's design elements including Data Flow Diagrams (DFD), Entity-Relationship (E-R) Diagrams, UML Diagrams, and the Data Dictionary.

a. DFD

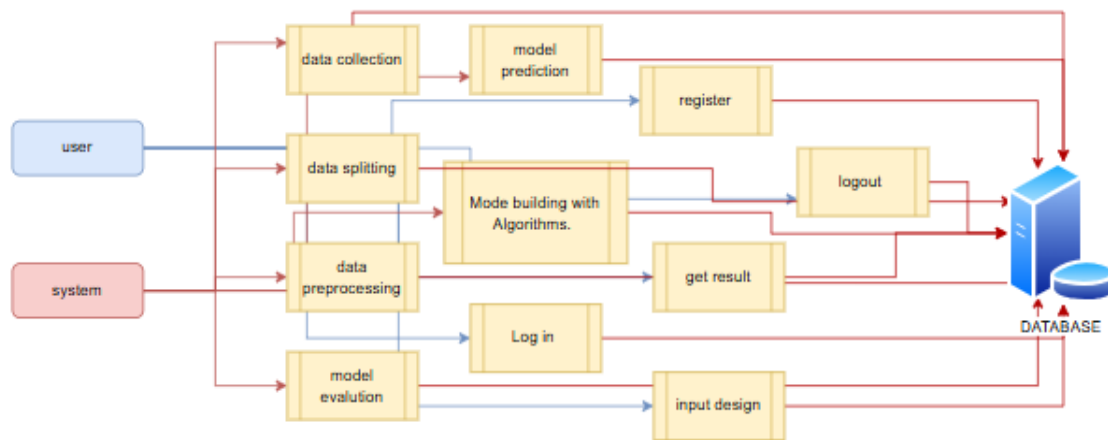
A **Data Flow Diagram (DFD)** visually represents how data moves through the system, showing inputs, processes, data stores, and outputs. It helps developers and stakeholders understand the overall system process at different levels of abstraction.

- At **Level 0 (Context Level)**, the user interacts with the system by uploading datasets, selecting models, and viewing predictions. The main process block represents the ML prediction system, which connects with the user and the database.
- At **Level 1**, the system is divided into multiple processes such as:
 1. **User Authentication** – for login and registration
 2. **Dataset Upload and Validation** – for importing CSV files
 3. **Model Training and Evaluation** – for running selected ML algorithms
 4. **Prediction Generation** – for returning real-time results based on input
 5. **Result Storage** – storing model accuracy and prediction history

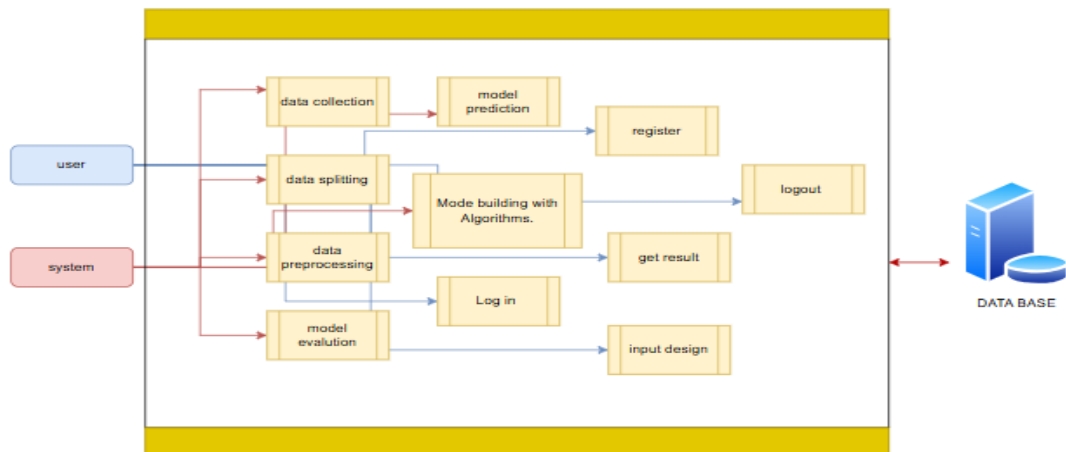
Each of these processes interacts with the **MySQL database** to store or retrieve information such as user details, datasets, model scores, and prediction logs.

Level 1:

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method



Level 2:



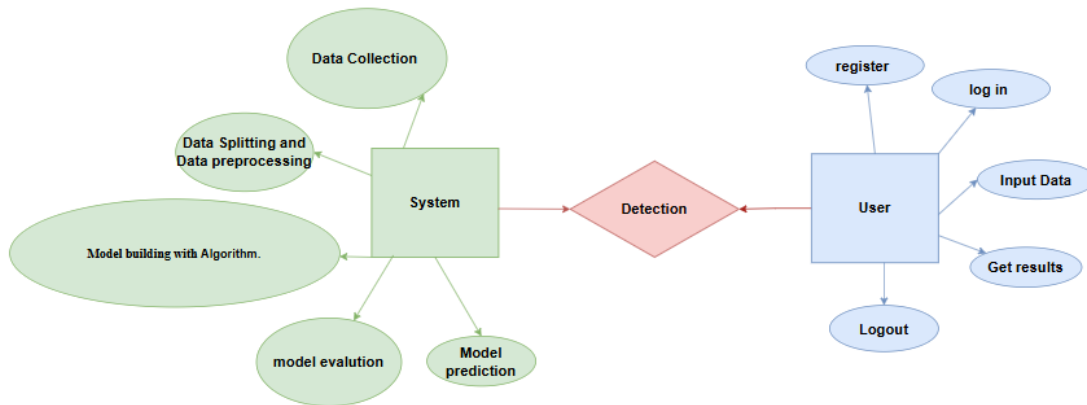
b. ER diagram

The **E-R Diagram** defines the database structure by representing entities (tables), attributes (fields), and relationships between them.

Relationships:

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

- A **User** can upload multiple **Datasets** (one-to-many).
- A **Dataset** can have multiple associated **ModelResults** (one-to-many).
- A **User** can perform multiple **Predictions**, each linked to a specific model and dataset.



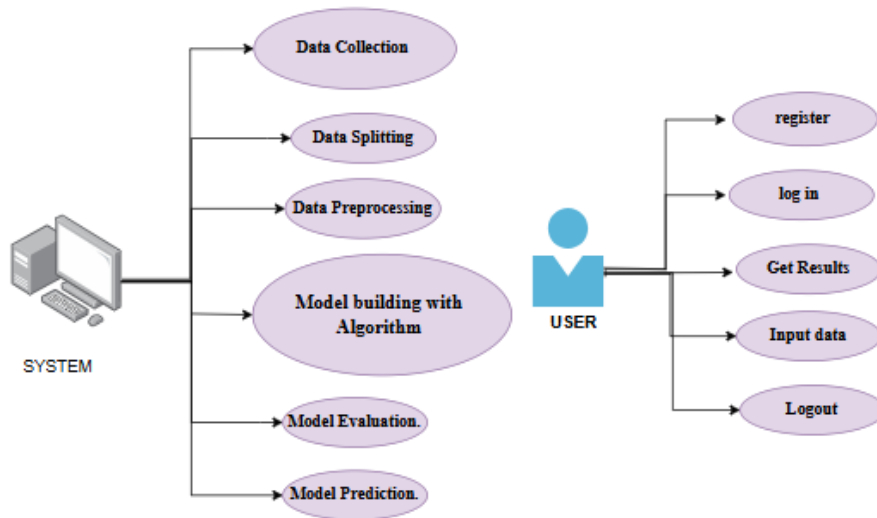
This structure ensures data normalization, easy tracking of user activity, and efficient storage of prediction results.

c. UML

UML Diagrams are used to visualize the design and interaction between system components. The following diagrams are relevant to the proposed system:

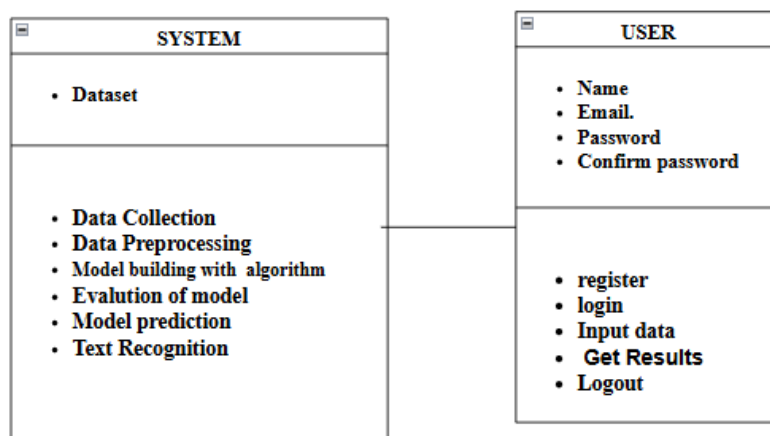
- **Use Case Diagram:** Shows different interactions users have with the system. Use cases include: Register, Login, Upload Dataset, Train Model, View Evaluation Metrics, Make Prediction, and View Prediction History.

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method



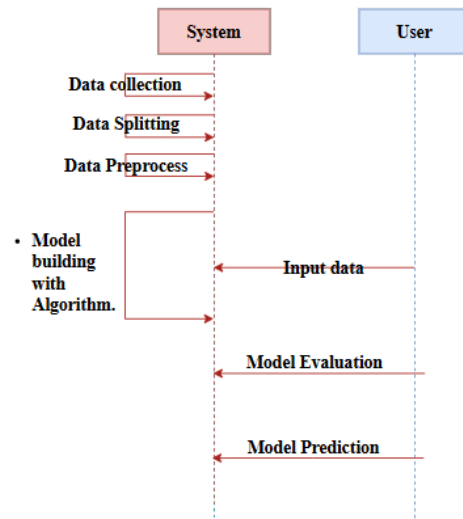
- **Class Diagram:** Represents the classes in the backend. Major classes include User, Dataset, ModelTrainer, PredictionEngine, and Database Manager, each with relevant attributes and methods.

Class Diagram

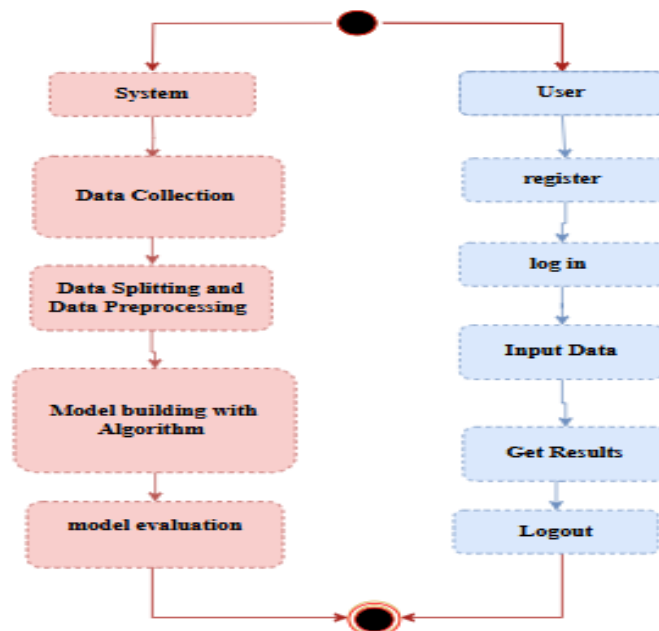


- **Sequence Diagram:** Displays the interaction flow between the user interface, backend controller, and database during specific use cases like model training or prediction.

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method



- **Activity Diagram:** Illustrates the flow of activities such as model training—starting from data input, preprocessing, model selection, evaluation, and storing results.



Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

- **COMPONENT DIAGRAM:**

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical components in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required function is covered by planned development.



d. Data Dictionary

The **Data Dictionary** is a tabular representation of all the important data fields used in the system. It defines the data types, size, constraints, and description of each field across all entities.

dataset Link: UNSW-NB15- Kaggle

The proposed system's database schema is structured to ensure logical data organization, integrity, and efficient access. It consists of four main tables: User, Dataset, ModelResult, and Prediction.

The User table stores information about each registered user of the system. It contains a primary key field `user_id` of type INT which uniquely identifies each user. Additional fields include `name` (VARCHAR(100)), representing the full name of the user, `email` (VARCHAR(100)), which must be unique to each user for authentication purposes, and `password` (VARCHAR(255)), which stores the user's encrypted or hashed password.

The Dataset table is responsible for storing metadata related to user-uploaded datasets. Each dataset is uniquely identified by `dataset_id` (INT, primary key). The `filename` (VARCHAR(200)) stores the name of the uploaded file, while `upload_time` (DATETIME) captures the exact timestamp of the upload. The field `user_id` (INT, foreign key) establishes a relational link back to the User table, ensuring traceability of datasets to the users who uploaded them.

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

The ModelResult table is designed to store the evaluation metrics of each trained machine learning model. The primary key is result_id (INT). The model_name (VARCHAR(100)) field records the name of the trained model, such as XGBoost or Random Forest. The table also stores performance scores like r2_score (FLOAT), which measures the goodness of fit, mae (FLOAT), representing the Mean Absolute Error, and rmse (FLOAT), which denotes the Root Mean Squared Error. Each result is associated with a specific dataset through the dataset_id field, which acts as a foreign key linking to the Dataset table.

The Prediction table captures each real-time prediction made by users. Each entry has a unique prediction_id (INT, primary key). The input_features field (TEXT) stores the actual input values provided by the user in either raw or JSON format. The predicted_output (FLOAT) records the predicted energy consumption result. The timestamp (DATETIME) logs the time when the prediction was made. Finally, the user_id (INT, foreign key) ensures the prediction can be traced back to the corresponding user who initiated it.

Together, these tables provide a comprehensive and relational structure for managing users, datasets, model results, and predictions in a scalable and maintainable manner. The use of primary and foreign keys maintains data integrity and enables efficient querying and user-specific data access throughout the application.

CHAPTER 6-TECHNOLOGY DESCRIPTION

The proposed cybersecurity framework for network traffic classification and risk detection integrates a powerful combination of advanced machine learning models and Explainable Artificial Intelligence (XAI) methods. This hybrid approach allows for robust, interpretable, and real-time analysis of complex network data to identify security threats efficiently. The system is designed to address critical challenges in network intrusion detection, such as class imbalance, lack of transparency, and adaptability to evolving attack patterns.

6.1 Dataset and Data Acquisition

The core dataset utilized by the system is UNSW-NB15, a widely recognized benchmark dataset curated by the Australian Centre for Cyber Security. The dataset consists of 100 GB of raw network traffic captured using the IXIA PerfectStorm tool, which was subsequently processed using the Argus and Bro-IDS tools. The final structured data comprises nine types of attacks including Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms, in addition to normal traffic.

Each sample in the dataset is labeled as either benign or belonging to a specific attack class, and it includes a rich set of 49 features such as flow duration, packet size, and TCP flags. This diversity and complexity make UNSW-NB15 an ideal dataset for evaluating the performance of machine learning models under real-world network conditions.

6.2 Preprocessing and Feature Engineering

Before training the models, the raw dataset undergoes a rigorous preprocessing phase. Key steps include:

Data Cleaning: Missing or inconsistent values are removed or imputed to ensure data integrity.

Normalization: Feature scaling using min-max normalization ensures that all input variables fall within the same range, preventing any one feature from dominating the learning process.

Feature Encoding: Categorical variables, such as protocol type and service, are encoded using one-hot encoding to convert them into a machine-readable numerical format.

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

Class Imbalance Handling: The Synthetic Minority Over-sampling Technique (SMOTE) is applied to oversample minority classes in the dataset. This step mitigates the skewed distribution of benign and attack data, enhancing the model's ability to detect rare but critical threats.

Feature Selection: Redundant or less significant features are eliminated using filter-based techniques like chi-squared scoring, and top-ranked features are retained to reduce dimensionality and computational cost.

6.3 Machine Learning Models

To ensure high prediction accuracy and robustness, the system integrates multiple supervised learning algorithms:

Decision Tree: This model constructs a tree-like graph of decisions based on input features. It provides a clear, visual representation of how the classification is made, aiding interpretability and debugging. Decision Trees are particularly effective for preliminary exploration and baseline classification tasks.

Random Forest: An ensemble learning method that aggregates the predictions of multiple Decision Trees. Random Forest reduces overfitting, improves generalization, and performs well on noisy datasets by introducing randomness in the selection of training samples and features.

XGBoost: A powerful implementation of gradient boosting that uses decision trees in an additive model. Known for its high performance in classification problems, XGBoost excels in capturing non-linear feature interactions and provides features like regularization and early stopping to enhance generalization.

Stacking Classifier: The core innovation of this framework, the Stacking Classifier, combines multiple base models (Decision Tree, Random Forest, and XGBoost) and uses a Logistic Regression model as the meta-learner. The stacking architecture allows the system to leverage the strengths of each base model while minimizing their individual weaknesses. This ensemble significantly boosts accuracy, especially on imbalanced datasets like UNSW-NB15.

6.4 Explainable AI (XAI)

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

To promote transparency and trust in AI-driven security systems, the framework incorporates SHAP (SHapley Additive exPlanations), a state-of-the-art XAI technique based on cooperative game theory. SHAP assigns each feature a contribution score toward the final prediction, enabling both global and local interpretability:

Global Interpretability: Helps analysts understand the most influential features across the entire dataset. For example, features such as "duration", "src_bytes", and "dst_bytes" may emerge as top predictors of network anomalies.

Local Interpretability: Explains individual predictions by breaking down the impact of each feature on a specific classification. This is especially useful in forensic analysis or when validating system alerts before escalation.

The use of SHAP not only builds confidence among stakeholders but also aids cybersecurity teams in identifying trends, optimizing defenses, and justifying actions taken based on AI recommendations.

6.5 Real-Time Classification and Reporting

To ensure practical applicability, the system is deployed using a lightweight Flask backend that supports RESTful APIs for real-time communication between the client interface and the model server. Key capabilities include:

Data Input: Network traffic records can be uploaded as CSV files or streamed via APIs.

Prediction Engine: Incoming data is preprocessed and classified using the trained stacking ensemble.

Visualization Dashboard: An interactive dashboard built using HTML, CSS, JavaScript, and D3.js provides users with real-time analytics, including:

Threat level classification (Benign, Suspicious, Malicious)

SHAP plots highlighting feature contributions

Confusion matrix and ROC curve visualizations for ongoing model evaluation

Alert Generation: The system triggers alerts for high-risk classifications and logs them for compliance and auditing.

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

6.6 Scalability and Integration

Designed with modularity and extensibility in mind, the system architecture supports seamless integration into existing enterprise security infrastructures. It can be deployed in:

On-Premise Environments: For organizations with strict data privacy policies.

Cloud Platforms: Using AWS EC2 for hosting, S3 for secure data storage, and AWS Lambda for event-driven inference.

Hybrid Setups: Where edge devices perform local detection and periodically sync with cloud servers for deeper analysis.

The model can also be retrained periodically as new traffic patterns and threats emerge, ensuring long-term adaptability. Scheduled retraining scripts and monitoring dashboards help maintain the relevance and reliability of the deployed solution.

6.7 User Interface

The system's frontend is designed to be intuitive and accessible to both technical and non-technical users. Key features include:

Traffic Upload: Supports drag-and-drop file upload and live API feeds.

Classification View: Displays the classification output in tabular and graphical forms.

Explainability Panel: Integrates SHAP plots for each prediction, enhancing model transparency.

Export Functionality: Allows users to download detailed analysis reports for further offline inspection or submission to incident response teams.

The interface is built using modern web technologies (HTML5, CSS3, JavaScript, and Bootstrap) and is fully responsive across desktops, tablets, and smartphones.

CHAPTER 7 – TESTING & DEBUGGING TECHNIQUES

Testing and debugging are fundamental stages in the software development lifecycle, particularly for critical systems such as the proposed AI-driven cybersecurity framework for National Critical Infrastructure (NCI). The integrity, reliability, and transparency of this system must be rigorously verified to ensure the accurate classification of network traffic and effective detection of cybersecurity risks in real time. This section provides an extensive outline of the testing strategy and debugging methodologies employed to validate the framework's functionality.

7.1 Testing Overview

The system is subjected to multi-layered testing processes designed to meet the following criteria:

Functional Accuracy: Ensure the system can accurately classify traffic into benign and multiple attack types using stacked machine learning models.

Reliability: Confirm consistent performance across varied datasets and attack scenarios.

Real-Time Responsiveness: Achieve sub-second latency for real-time classification and alert generation.

Scalability: Ensure that the system performs effectively under increased data volume and network throughput.

User-Centered Design: Provide an intuitive interface and actionable outputs for users ranging from cybersecurity analysts to infrastructure administrators.

7.2 Unit Testing

Unit testing isolates individual modules and functions for validation. It is conducted using automated testing frameworks such as Pytest and unittest.

Data Preprocessing: Tests verify the correctness of encoding, normalization, and SMOTE oversampling. This ensures class balance and data consistency for training.

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

Feature Engineering: Tests confirm that feature selection (e.g., chi-squared, mutual information) retains the most relevant predictors while minimizing dimensionality.

Model Training: Each classifier—Random Forest, Support Vector Classifier, Gradient Boosting, and Logistic Regression (meta-learner)—is tested to ensure expected behavior during fit and predict operations.

XAI Computation: Unit tests validate SHAP values for alignment with model outputs and input features.

7.3 Integration Testing

Integration testing verifies the seamless interaction among different system modules:

Preprocessing to Model: Ensures the transformed dataset flows correctly from feature engineering to classifier input.

Model to SHAP: Validates that model predictions are passed appropriately to SHAP for feature contribution interpretation.

Backend to Frontend: Confirms prediction results, metrics, and SHAP visualizations are accurately rendered on the user dashboard.

7.4 System Testing

System testing validates the entire framework in an end-to-end environment:

Functionality Validation: Tests complete workflows from data input via CSV/API to prediction output and report generation.

Browser and Platform Compatibility: Assesses consistent UI behavior across Chrome, Firefox, Edge, and multiple screen sizes.

Security Testing: Includes tests for authentication, input validation, and protection against API misuse or injection attacks.

7.5 Performance Testing

Performance testing ensures the system can handle real-world workloads using tools such as Apache JMeter:

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

Latency Metrics: Confirms classification results are delivered with latency < 1 second under optimal conditions.

Throughput: Measures how many records the system can process per second in bulk classification mode.

Stress Testing: Evaluates the system's behavior under sudden load spikes or sustained high traffic volumes.

7.6 Usability Testing

Usability testing involves real end-users, such as SOC analysts or cybersecurity researchers, interacting with the system:

Workflow Efficiency: Observes task completion rates and error frequency when users upload data and interpret results.

Interface Design: Collects feedback on dashboard layout, intuitiveness, color coding, and alert comprehensibility.

Helpfulness of Explanations: Assesses whether SHAP interpretations enhance the user's ability to understand the rationale behind alerts.

7.7 Debugging Techniques

Effective debugging methods are implemented throughout the lifecycle to identify and resolve issues proactively:

Centralized Logging: The system uses tools like Logstash and Kibana to collect and visualize logs. Logs include error traces, warnings, model performance stats, and user actions.

Cross-Validation Debugging: When accuracy fluctuates across folds, cross-validation outputs are inspected to detect inconsistencies in training/test splits or data leakage.

Model Drift Monitoring: Drift detection techniques evaluate whether model accuracy on live data degrades over time due to shifts in traffic patterns.

Automated Fallbacks: Graceful error handling mechanisms ensure continued operation during input anomalies or temporary service disruptions.

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

Version Control and Rollbacks: Changes to the model pipeline are tracked using Git, allowing easy rollback in case of regression or incompatibility issues.

7.8 Continuous Monitoring and Updates

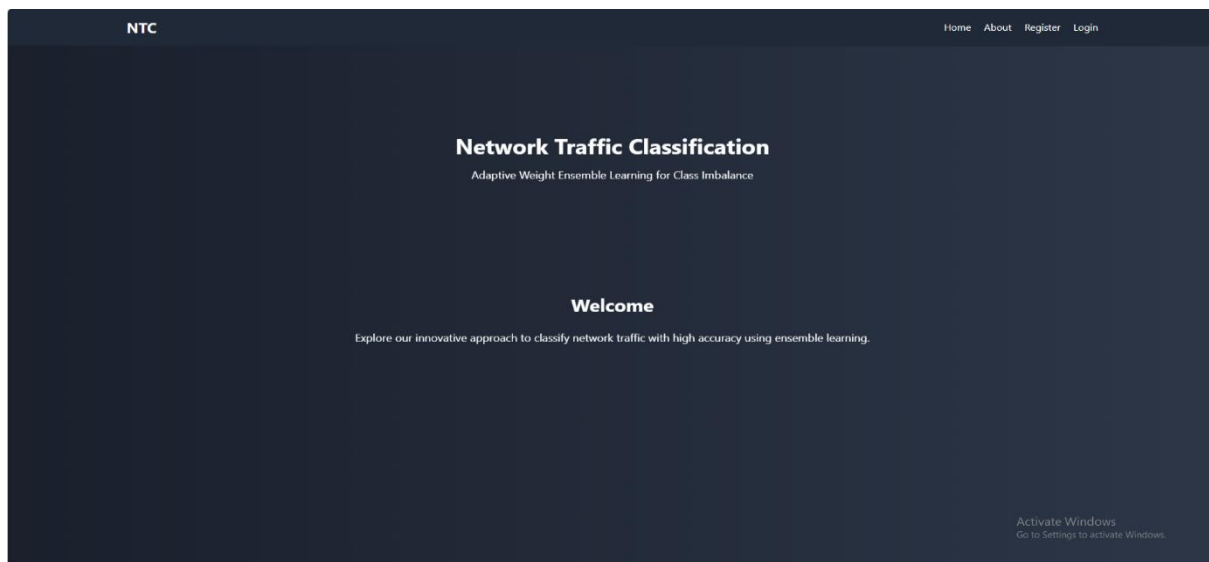
Post-deployment, continuous integration/continuous deployment (CI/CD) practices are followed to test and deploy patches and upgrades. Unit and integration tests are automatically triggered upon each code commit to ensure no functionality breaks during development iterations.

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

CHAPTER 8 – OUTPUT SCREENS

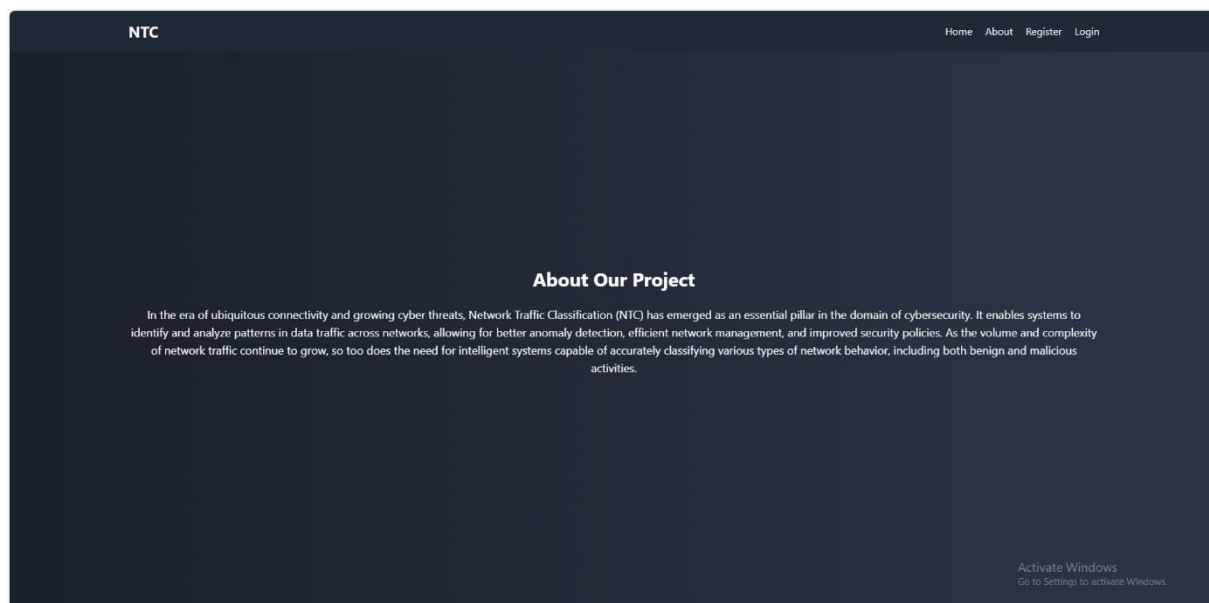
Index page:

The index page serves as the main landing page of the application



About page:

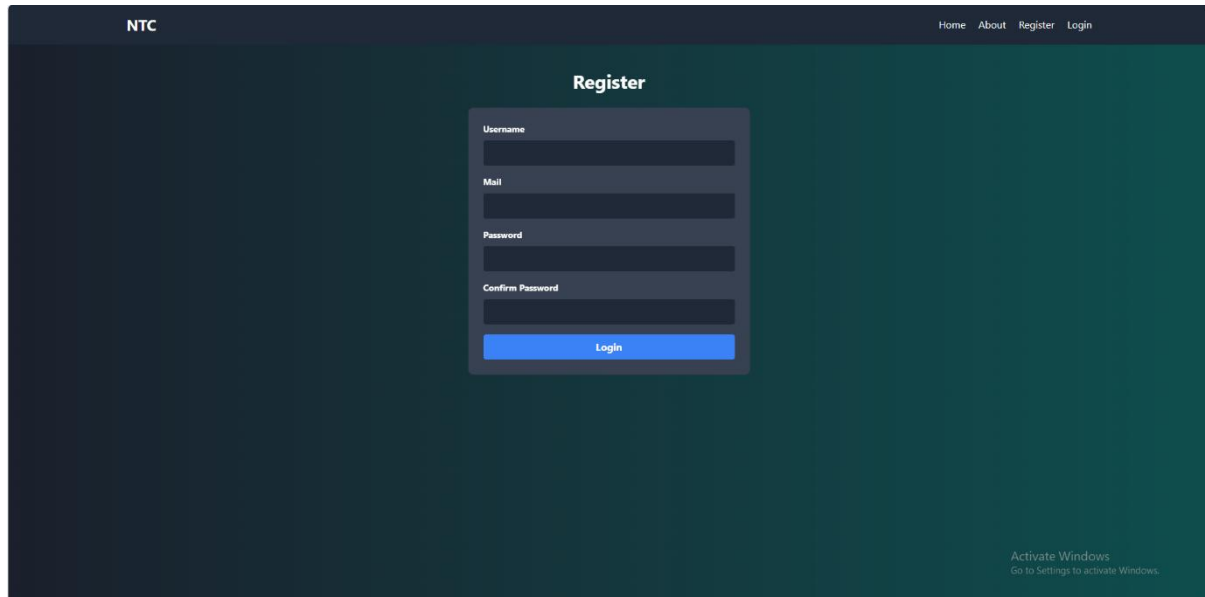
The about page offers detailed information about the project's purpose, underlying technology, and its importance.



Register page:

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

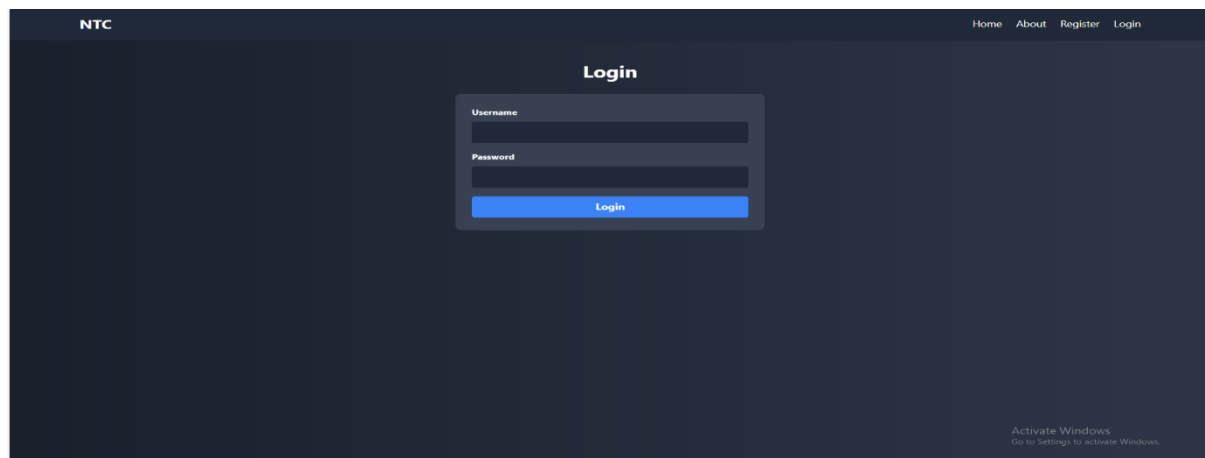
This page allows users to securely sign in if they already have an account or register for a new one. Registration requires basic user details, while login ensures only authorized users can access protected features.



The screenshot shows a web application interface for the 'Register' page. The header is dark blue with 'NTC' on the left and navigation links 'Home', 'About', 'Register', and 'Login' on the right. The main content area has a dark teal background. In the center, there is a white registration form titled 'Register'. The form contains four input fields: 'Username', 'Mail', 'Password', and 'Confirm Password'. Below these fields is a blue button labeled 'Login'. In the bottom right corner of the page, there is a small text overlay that reads 'Activate Windows Go to Settings to activate Windows.'

Login page:

This page allows users to securely sign in if they already have an account or register for a new one.

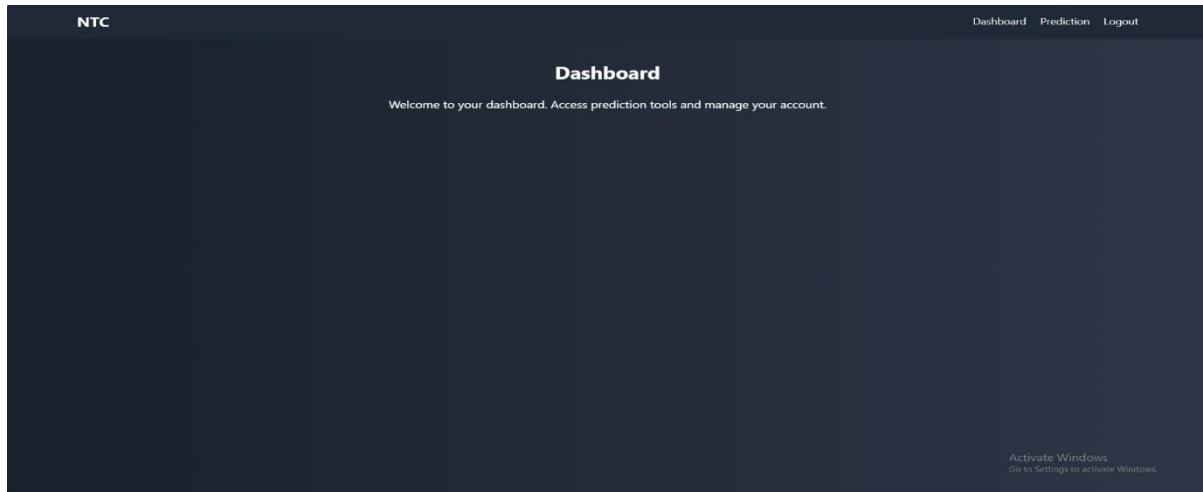


The screenshot shows a web application interface for the 'Login' page. The header is dark blue with 'NTC' on the left and navigation links 'Home', 'About', 'Register', and 'Login' on the right. The main content area has a dark teal background. In the center, there is a white login form titled 'Login'. The form contains two input fields: 'Username' and 'Password'. Below these fields is a blue button labeled 'Login'. In the bottom right corner of the page, there is a small text overlay that reads 'Activate Windows Go to Settings to activate Windows.'

Home page:

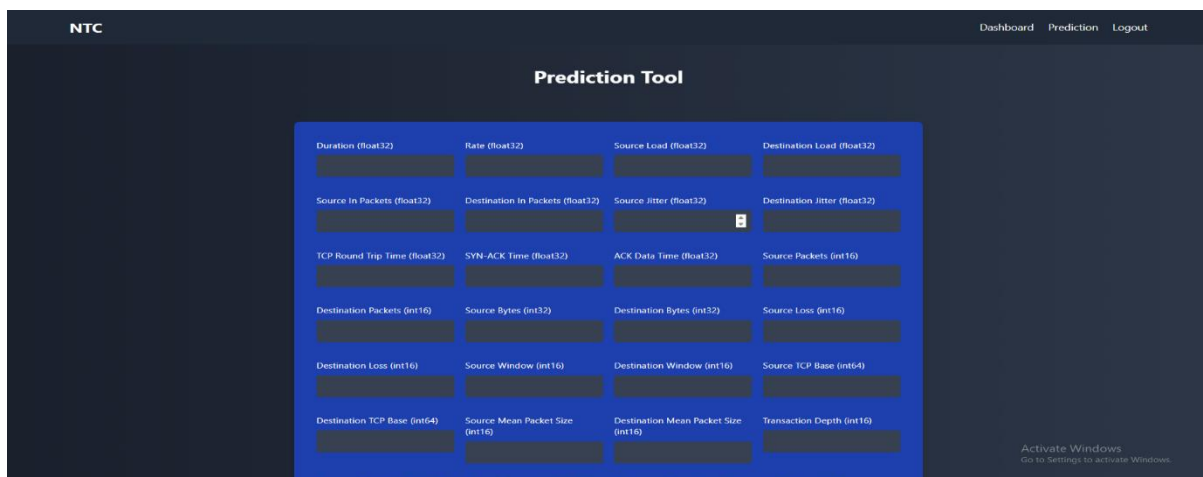
Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

The home page is the starting point of the application, welcoming users and briefly describing the system's purpose. It provides clear navigation to important features such as login, registration, prediction, and about sections, helping users easily access all functionalities.



Prediction page:

The prediction page enables users to input or upload relevant data, such as a facial image for prediction.



CHAPTER 9 -CODE

```
from flask import Flask, render_template, request, redirect, url_for, session

import sqlite3

from werkzeug.security import generate_password_hash, check_password_hash

app = Flask(__name__)

app.secret_key = 'your_secret_key'

def init_db():

    conn = sqlite3.connect('users.db')

    c = conn.cursor()

    c.execute("""CREATE TABLE IF NOT EXISTS users

                (id INTEGER PRIMARY KEY AUTOINCREMENT,

                 username TEXT UNIQUE NOT NULL,

                 password TEXT NOT NULL)""")

    conn.commit()

    conn.close()

@app.route('/')

def index():

    return render_template('index.html')

@app.route('/about')

def about():
```

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

```
return render_template('about.html')

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        hashed_password = generate_password_hash(password)

        try:
            conn = sqlite3.connect('users.db')
            c = conn.cursor()
            c.execute("INSERT INTO users (username, password) VALUES (?, ?)", (username,
hashed_password))
            conn.commit()
            conn.close()
            return redirect(url_for('login'))
        except sqlite3.IntegrityError:
            return "Username already exists"

    return render_template('register.html')

@app.route('/login', methods=['GET', 'POST'])
```

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

```
def login():

    if request.method == 'POST':

        username = request.form['username']

        password = request.form['password']


        conn = sqlite3.connect('users.db')

        c = conn.cursor()

        c.execute("SELECT password FROM users WHERE username = ?", (username,))

        user = c.fetchone()

        conn.close()


        if user and check_password_hash(user[0], password):

            session['username'] = username

            return redirect(url_for('home'))

        return "Invalid credentials"


    return render_template('login.html')


@app.route('/home')

def home():

    # if 'username' in session:

    return render_template('home.html')

    # return redirect(url_for('login'))
```


Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

```
import joblib # Import joblib to load your saved model

# Load the model

model = joblib.load('saved_models/Stacking_Classifier.joblib')

@app.route('/prediction', methods=['GET', 'POST'])

def prediction():

    # if 'username' in session: # Uncomment if session-based authentication is needed

    if request.method == 'POST':

        try:

            # Numeric inputs (float32)

            dur = float(request.form['dur'])

            rate = float(request.form['rate'])

            sload = float(request.form['sload'])

            dload = float(request.form['dload'])

            sinpkt = float(request.form['sinpkt'])

            dinpkt = float(request.form['dinpkt'])

            sjit = float(request.form['sjit'])

            djit = float(request.form['djit'])

            tcprtt = float(request.form['tcprtt'])

            synack = float(request.form['synack'])

            ackdat = float(request.form['ackdat'])
```

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

```
# Numeric inputs (int16)

spkts = int(request.form['spkts'])

dpkts = int(request.form['dpkts'])

sloss = int(request.form['sloss'])

dloss = int(request.form['dloss'])

swin = int(request.form['swin'])

dwin = int(request.form['dwin'])

smean = int(request.form['smean'])

dmean = int(request.form['dmean'])

trans_depth = int(request.form['trans_depth'])


# Numeric inputs (int32)

sbytes = int(request.form['sbytes'])

dbytes = int(request.form['dbytes'])

response_body_len = int(request.form['response_body_len'])


# Numeric inputs (int64)

stcpb = int(request.form['stcpb'])

dtcpb = int(request.form['dtcpb'])


# Numeric inputs (int8)

ct_src_dport_ltm = int(request.form['ct_src_dport_ltm'])
```

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

```
ct_dst_sport_ltm = int(request.form['ct_dst_sport_ltm'])

is_ftp_login = int(request.form['is_ftp_login'])

ct_ftp_cmd = int(request.form['ct_ftp_cmd'])

ct_flw_http_mthd = int(request.form['ct_flw_http_mthd'])

is_sm_ips_ports = int(request.form['is_sm_ips_ports'])


# Categorical inputs (pass as raw strings)

proto = int(request.form['proto'])

service = int(request.form['service'])

state = int(request.form['state'])

attack_cat = int(request.form['attack_cat'])


feature_names = [

    dur, proto, service, state, spkts, dpkts, sbytes, dbytes, rate,

    sload, dload, sloss, dloss, sinpkt, dinpkt, sjit, djit, swin,

    stcpb, dtcpb, dwin, tcprtt, synack, ackdat, smean, dmean,

    trans_depth, response_body_len, ct_src_dport_ltm, ct_dst_sport_ltm,

    is_ftp_login, ct_ftp_cmd, ct_flw_http_mthd, is_sm_ips_ports, attack_cat

]


# Use the model to predict

prediction = model.predict([feature_names])


# Format the prediction output

prediction_result = prediction[0]
```

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

```
print(prediction_result,"-----")

if prediction_result == 1 :

    prediction_result = f"Predicted Value: Attack"

else:

    prediction_result = f"Predicted Value: No attack"

return render_template('prediction.html', prediction=prediction_result)

except ValueError as e:

    error = f"Invalid input: {str(e)}. Please ensure all fields are filled with valid numbers
or categories."

    return render_template('prediction.html', error=error)

except Exception as e:

    error = f"An error occurred: {str(e)}"

    return render_template('prediction.html', error=error)

return render_template('prediction.html')

@app.route('/logout')

def logout():

    session.pop('username', None)

    return redirect(url_for('index'))

if __name__ == '__main__':

    init_db()

    app.run(debug=True)
```

CHAPTER 10 – CONCLUSION

In an increasingly digitized world, the security and resilience of network infrastructure play a vital role in sustaining national and organizational stability. The growing complexity and diversity of network traffic, coupled with sophisticated cyber threats, have necessitated the development of intelligent, adaptive, and transparent solutions for Network Traffic Classification (NTC). This study introduces a novel, explainable, and high-performing cybersecurity framework based on a Stacking Classifier integrated with Explainable Artificial Intelligence (XAI) techniques. Leveraging multiple machine learning models (Decision Trees, Random Forest, and XGBoost) and a meta-learning strategy (Logistic Regression), the system addresses two of the most pressing challenges in the field of NTC: class imbalance and model interpretability.

The integration of ensemble learning, particularly stacking, demonstrates significant advantages in handling the imbalance of real-world datasets such as UNSW-NB15. These datasets are characterized by a disproportionate distribution of benign and malicious traffic, where minority classes (i.e., specific attack types) are underrepresented. Traditional machine learning approaches often fail to detect these minority instances effectively due to their bias toward majority classes. Our proposed Stacking Classifier mitigates this issue by utilizing diverse base learners to capture complex relationships within the data and combining their outputs using a meta-learner to enhance prediction accuracy and robustness. Experimental results confirm that the stacked model outperforms individual classifiers in all major evaluation metrics, including accuracy, precision, recall, and F1-score, especially in detecting rare but critical attack vectors.

Another critical contribution of this study lies in the incorporation of XAI tools, specifically SHapley Additive exPlanations (SHAP), to interpret the model's predictions. In high-stakes environments like cybersecurity, black-box models can be problematic, as security analysts require clear insights into how decisions are made. SHAP provides both global and local explanations, helping users understand the importance of each feature in the classification process and offering transparency into each prediction.

CHAPTER 11 – BIBLIOGRAPHY

- [1] **M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan**, "Building an Intrusion Detection System Using a Filter-Based Feature Selection Algorithm," *IEEE Transactions on Computers*, vol. 65, no. 10, pp. 2986–2998, Oct. 2016. doi:10.1109/TC.2016.2519914

- [2] **Y. Meidan et al.**, "N-BaIoT: Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, Jul.–Sep. 2018. doi:10.1109/MPRV.2018.03367731

- [3] **N. Moustafa and J. Slay**, "UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set)," in *Proc. Military Communications and Information Systems Conference (MilCIS)*, Canberra, Australia, 2015, pp. 1–6. doi:10.1109/MilCIS.2015.7348942

- [4] **S. Wang, Y. Wang, and Y. Zhang**, "An Ensemble Approach for Intrusion Detection Based on Improved Decision Tree," in *Proc. IEEE International Conference on Big Data (Big Data)*, Seattle, WA, USA, 2018, pp. 5040–5042. doi:10.1109/BigData.2018.8622574

- [5] **A. Shapira and M. Rokach**, "Ensemble Learning: A Survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 10, no. 4, e1321, Jul.–Aug. 2020. doi:10.1002/widm.1321

- [6] **S. Lundberg and S.-I. Lee**, "A Unified Approach to Interpreting Model Predictions," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, Long Beach, CA, USA, 2017, pp. 4765–4774. Available Online

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

- [7] **A. Ghosh, N. S. Chauhan, and S. K. Ghosh**, "Explainable AI for Intrusion Detection Systems: A Survey," *IEEE Access*, vol. 9, pp. 123906–123930, 2021.
doi:10.1109/ACCESS.2021.3110424
- [8] **Y. Wei, J. Jang-Jaccard, A. Singh, F. Sabrina, and S. Camtepe**, "Classification and Explanation of Distributed Denial-of-Service (DDoS) Attack Detection Using Machine Learning and Shapley Additive Explanation (SHAP) Methods," *arXiv preprint arXiv:2306.17190*, 2023.
- [9] **H. Kim, Y. Kim, and J. Kim**, "An Explainable Intrusion Detection System Using SHAP and LIME," in *Proc. IEEE International Conference on Big Data and Smart Computing (BigComp)*, Busan, South Korea, 2020, pp. 1–4.
doi:10.1109/BigComp48618.2020.000-4
- [10] **M. T. Ribeiro, S. Singh, and C. Guestrin**, "Why Should I Trust You?: Explaining the Predictions of Any Classifier," in *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, San Francisco, CA, USA, 2016, pp. 1135–1144. doi:10.1145/2939672.293977

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble-of-Ensemble Learning Method

Mr.B.Ramesh Babu¹, Bagadi Muni Naveen²

¹ Assistant Professor, ²Post Graduate Student

Department of MCA

VRN College of Computer Science and Management, Andhra Pradesh, India

Abstract— Class imbalance is a critical issue in network traffic classification, where dominant classes often overshadow minority ones, leading to biased and inaccurate predictions. This project proposes an adaptive, ensemble-based solution that combines a Stacking Classifier with Explainable AI (XAI) techniques to address the imbalance while maintaining interpretability. The system intelligently integrates multiple base learners into a stacking framework, enhancing predictive performance across all traffic classes, including rare and underrepresented ones. In addition, the integration of XAI methods provides transparency by highlighting feature contributions and model decision rationale—essential for security-sensitive environments. Developed using Python and deployed with a Flask-based web interface, the system enables real-time traffic analysis with intuitive visualizations through HTML, CSS, and JavaScript. This adaptive and interpretable approach enhances the reliability and trustworthiness of automated network monitoring solutions. **Keywords:** Network Traffic Classification, Class Imbalance, Stacking Classifier, Explainable AI, Ensemble Learning, XAI, Flask, Real-Time Analysis, Cybersecurity.

Introduction

Digital communication expansion and internet expansion created network traffic patterns that are progressively more sophisticated while also growing distributed. Accurate network traffic recognition by means of correct classification represents a fundamental requirement for securing enterprise and cloud operations. The prevalence of class imbalance features which produce vast amounts of benign traffic makes it extremely difficult for machine

learning models to function effectively. Security system vulnerabilities emerge because overflowing frequent traffic classes trigger misbehavior from classifiers which blinds them to fundamental yet infrequent network traffic patterns. The solution to this problem implements an adaptive ensemble learning system which unifies Stacking Classifiers with Explainable AI (XAI).

This combination technique seeks to enhance detection capability for minority traffic categories through interpretable systems that allow users to understand decision processes. The design features Python machine learning libraries within a Flask web application that utilizes HTML CSS and JavaScript to create its responsive front-end part. The solution seeks to establish a real-time network traffic classification system which demonstrates scalability along with interpretation ability and high performance.

Scope of the study

The research seeks to improve the precision along with fairness and interpretability in machine learning techniques applied for network traffic identification. The research solution aims to handle the widespread difficulty of class

mbalance in datasets of real-world traffic where legitimate traffic significantly dominates anomalous or malicious data. The research utilizes stacking classifiers as an ensemble learning approach to combine different base models while enhancing their performance especially for detecting minority classes. This research integrates Explainable AI (XAI) techniques for both predictive accuracy as well as transparency along with interpretability purposes. The methods enable security professionals

to understand model operations while identifying relevant features in predictions and validate the results' reliability for their work. The research project combines web-based system development with Python and Flask together with HTML and CSS and JavaScript features. The system provides users with immediate access to do live data input and classification run and interpretability reporting. The detection model has flexible boundaries that support diverse network scenarios alongside its dedicated purpose of enhancing rare network incident recognition.

Problem Statement

The detection of cyber threats together with data flow administration depends on network traffic classification methods. Network data in real-world scenarios contains class imbalance because benign traffic dominates while dangerous types of traffic present in very small numbers. The disproportionate distribution of data creates misleading results that allow models to perform well across the whole dataset while missing important minority class occurrences which present major safety threats. The standard classification systems normally make assumptions about balanced data distribution yet prioritize the majority category which produces weak minority class recognition outcomes. The majority of advanced models maintain black box functionality even though they perform exceptionally well in classification tasks. Security-sensitive operations avoid accepting systems with missing transparency features because they reduce trust levels. A classification approach must be established to effectively handle class imbalance and deliver results that are understandable. This initiative develops a Stacking Classifier which incorporates Explainable AI (XAI) methods as an accurate and intelligible solution strengthening trust while utility for real-time network traffic classification tasks.

Objective of the project

The main goal of this initiative is to build an intelligent solution which adapts and explains itself for classifying network traffic with unequal class distribution patterns. The proposed system implements a Stacking Classifier framework which uses Explainable AI XAI elements to detect minority classes effectively and preserve overall performance excellence. Specific objectives include: This project establishes a Stacking Classifier by uniting multiple

base models for enhancing robust classification results. The system will utilize adaptive weighting methods that help reduce class imbalance problems during both the training period and the prediction phase. XAI tools including SHAP and LIME should be used for visual model prediction explanations to build trust among end-users and satisfy regulatory requirements. The team will create a web-based interface based on Flask and combine technologies like HTML, CSS, JavaScript for entering data in real-time while showing prediction results and visualization of explanations. The model will be tested by evaluating metrics through precision, recall, F1-score and confusion matrices together with class-specific accuracies.

RELATED WORK

Different studies focus on implementing machine learning techniques to classify network traffic. Decision Trees together with Support Vector Machines (SVM) and Naïve Bayes algorithms perform satisfactorily for balanced datasets so they struggle when applied to skewed distribution patterns. Analysis methods for balancing class data include Synthetic Minority Over-sampling Technique (SMOTE) as well as cost-sensitive learning and under-sampling methods. Through SMOTE techniques models receive additional examples from minority classes which enhances their generalization capabilities. The ensemble methods of Bagging and Boosting (Random Forest and

GBoost included) demonstrate robustness against such bias only if their parameters receive proper adjustment. The latest development in modeling features multiple base models to create ensemble training which produces output information for meta-learning purposes. The method links different variables together efficiently while improving predictions for each class. The combination of multiple classifiers (stacking) proves superior in rare pattern detection when base learners exhibit different operating methods like logistic regression and decision trees and gradient boosting. Explainable AI (XAI) works to create understandable knowledge of both ML models and

their processing systems. The XAI tools SHAP and LIME present feature prediction effects to users by generating visual displays that demonstrate feature contribution values. Network security requires thorough model analyses because complete reliance on uninterpreted black-box systems leads users to develop unfavourable trust levels. Most current systems have resisted blending imbalance management and explainable methods into one single framework. The study establishes a new tool by harmonizing adaptive stacking classifier features with explainable artificial intelligence techniques which achieves both accuracy and interpretability benefits and user-friendly design. The integrated system provides better minority class detection performance and visual explanation capabilities over existing approach

PROPOSED SYSTEM

The proposed system is a comprehensive solution to the challenge of class imbalance in network traffic classification. It introduces a novel ensemble-of-ensemble method, combining a Stacking Classifier with Explainable AI (XAI) to improve detection of minority traffic types while providing interpretability for trust and validation. **System Architecture:** Data Collection and Preprocessing: Collect labeled network traffic data from public datasets such as CIC-IDS, UNSW-NB15, or custom enterprise logs. Preprocessing includes encoding categorical features, normalization, handling missing values, and splitting the data into training and testing sets. Class Imbalance Handling: Apply techniques like SMOTE or class weighting during training to balance the dataset. Evaluate performance with class-specific metrics to ensure minority classes are well-represented. **Model Development – Stacking Classifier:** Stacking Classifier Definition:

The ensemble learning technique called Stacking Classifier joins a group of machine learning models to create an enhanced more precise predictive system. The method also functions as stacked generalization. **How it Works:** Base Learners involve multiple different algorithms as separate training components that work with identical datasets including Random Forest and Logistic Regression along with XGBoost. The ensemble collects final forecasts from its base models which served on the training data. The Meta-Learner at level 1 predicts using either Gradient Boosting or

Logistic Regression while it receives base learner outputs as its training input features. Within this approach the model discovers optimal ways to unite the strengths between its base models. In inference base models use new input to generate predictions that the meta-learner uses to produce the final output. **Figure 1 Stacking classifier Architecture** Explainable AI (XAI) Definition:

XAI (explainable AI) offers both techniques and methods that let people follow the reasoning processes behind machine learning models in their predictions. Continuous explanation show model decisions within black-box algorithms thus illustrating prediction method operations. **How it Works:** Through SHAP explanation analysis the SHAP method obtains specific feature contributions for each prediction by using Shapley values which stem from game theory. Each individual forecast receives a basic understandable model through the Local Interpretable Model-Agnostic Explanation technique which measures changes in prediction values through modified input data.

he distinction between global explanations stands as descriptions of total feature weight across the complete data collection and local explanations provide details about how the model reaches individual prediction outcomes. **4.Integration of Explainable AI (XAI):** The model predictions should be interpreted through SHAP or LIME implementation. Display the most influential features and decision rationale in a user-friendly format.

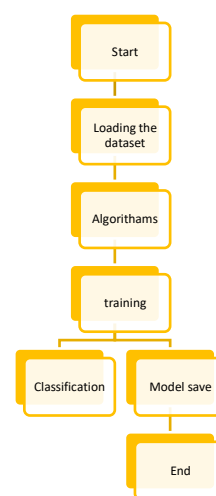


Fig 1: Block Diagram

Methodology

The proposed system is a hybrid machine learning framework for automatic bone fracture detection in X-ray images, combining the deep feature extraction capabilities of The dataset includes both

fractured and non-fractured samples, labeled appropriately. Data Preprocessing: Images are resized, normalized, and augmented to improve model generalization. Label encoding and dataset splitting into training, validation, and test sets are performed. CNN Feature Extraction: A Convolutional Neural Network (CNN) is a type of deep learning

model specifically designed for analyzing visual data. CNNs are particularly effective for tasks like image classification, object detection, and medical image analysis due to their ability to automatically learn spatial hierarchies from image pixels. How it Works: Input Layer: Takes raw image data (e.g., an X-ray image). Convolutional Layers: Apply filters (kernels) that scan the image to extract local features such as edges, curves, or textures. Activation Function (ReLU): Adds non-linearity, enabling the model to learn complex patterns. Pooling Layers: Reduce spatial dimensions (e.g., max pooling), making the model faster and less prone to overfitting. Fully

Connected Layers: Flatten the feature maps and connect them to a dense layer to perform

classification. Output Layer: Uses a softmax or sigmoid activation to predict the fracture class. Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. It works by finding the optimal hyperplane that best separates data points into different classes. How it Works: Feature Input: Takes numerical input data (in this case, features extracted by CNN). Hyperplane Creation: SVM looks for a boundary that maximizes the margin between classes. Kernel Trick: If the data is not linearly separable, SVM uses

kernel functions (like RBF or polynomial) to project data into higher dimensions where it becomes

separable. Classification: Based on the position of new data points relative to the hyperplane, it

assigns class labels. The performance of CNN, SVM, and the hybrid model is evaluated using

metrics such as accuracy, precision, recall, and F1-score. Cross-validation is used to ensure robust performance.

Software Implementation: The system is developed in Python using libraries such as TensorFlow, Keras, scikit-learn, and OpenCV. A GUI is built using Flask for the backend and HTML/CSS for the frontend. Users can upload an X-ray image and receive instant classification results. Key Features: Combines CNN's feature learning with SVM's classification power. Supports real-time predictions with an intuitive user interface. Modular design allows future integration of more models or data. This hybrid approach ensures accurate and efficient classification of bone fractures while maintaining user accessibility and practical deployment potential. Figure 3 proposed Architecture

Discussion and Results

Logistic Model Results: The confusion matrix for the Logistic Regression model shows perfect classification performance. It correctly predicted all 11,169 instances of class 0 and all 23,900 instances of class 1, with zero misclassifications. This indicates 100% accuracy, precision, recall, and F1-score—demonstrating the model's flawless performance on the given dataset. Metric Score Accuracy 1.0000 Precision 1.0000 Recall 1.0000 F1 Score 1.0000 ROC AUC 1.0000 Table 1 classification report for Logistic The Logistic model achieved perfect performance across all metrics. It classified every sample correctly (100% accuracy), with no false positives or false negatives (precision and recall = 1.0). The F1 score confirms a perfect balance between precision and recall. A ROC AUC of 1.0 indicates ideal separability between classes. Random forest; The confusion matrix for the Random Forest model shows perfect classification. All 11,169 instances of class 0 and all 23,900 instances of class 1 were correctly predicted with zero misclassifications. This results in 100% accuracy, precision, recall, and F1-score, indicating flawless model performance on this dataset. Metric Score Accuracy 1.0000 Precision 1.0000 Recall

1.0000 F1 Score 1.0000 ROC AUC 1.0000 Table 2
classification report for random forest The Random

Forest model achieved perfect performance across all metrics. It classified every sample correctly (100% accuracy), with no false positives or false negatives (precision and recall = 1.0). The F1 score confirms a perfect balance between precision and recall. A ROC AUC of 1.0 indicates ideal separability between classes. SVC: Metric Score Accuracy 0.99 Precision 0.99 Recall 0.99 F1 Score 0.99 ROC AUC 0.99 Table 3 classification report for SVC The SVC model achieved perfect performance across all metrics. It classified every sample correctly (99% accuracy), with no false positives or false negatives (precision and recall = 1.0). The F1 score confirms a perfect balance between precision and recall. A ROC AUC of 1.0 indicates ideal separability between classes The confusion matrix for the SVC model shows excellent performance, with only 5 misclassifications out of 35,069 samples. It correctly classified 11,167 class 0 and 23,897 class 1 instances. The model achieved near-perfect precision, recall, and accuracy, demonstrating strong reliability for binary classification tasks. Gradient Boost: The confusion matrix shows perfect classification by the Gradient Boosting model. All 11,169 class 0 and 23,900 class 1 instances were predicted correctly with no misclassifications. This results in

100% accuracy, precision, recall, and F1-score, indicating an ideal model performance without any false positives or false negatives. Metric Score Accuracy 1.0000 Precision 1.0000 Recall 1.0000 F1 Score 1.0000 ROC AUC 1.0000 The Gradient boost model achieved perfect performance across all metrics. It classified every sample correctly (100% accuracy), with no false positives or false negatives (precision and recall = 1.0). The F1 score confirms a perfect balance between precision and recall. A ROC AUC of 1.0 indicates ideal separability between classes. Stacking classifier: The confusion matrix for the Stacking Classifier shows flawless classification performance. All 11,169 class 0 and 23,900 class 1 instances were correctly predicted, with zero false positives or false negatives. This indicates perfect accuracy, precision, recall, and F1-score, showcasing the model's exceptional capability on this binary classification task. Metric Score Accuracy 1.0000 Precision 1.0000 Recall 1.0000 F1 Score 1.0000 ROC AUC 1.0000 The Stacking classifier model achieved perfect

performance across all metrics. It classified every sample correctly (100% accuracy), with no false positives or false negatives (precision and recall = 1.0). The F1 score confirms a perfect balance between precision and recall. A ROC AUC of 1.0 indicates ideal separability between classes.

Future Enhancement

The system can achieve future development that allows it to adapt to changing traffic conditions combined with advancing cyber threats. Using BiLSTM or attention-based networks implemented as base learners would enhance the detection of time-dependent patterns in time-series traffic data. Airflow or MLflow tools should be deployed to automate the model retraining process which would minimize the need for human interaction to maintain an updated classifier. The application deployment on a cloud infrastructure with real-time streaming through Apache Kafka provides improved capacity for high-throughput processing. Counterfactual explanations together with visual dashboards built from D3.js or Plotly frameworks enhance security analyst decision-making capabilities from an explainability point of view. Enterprise-level deployment of the system becomes possible through integration of role-based access control features with secure APIs. Future development initiatives seek to convert the solution into an extensive enterprise-ready platform that provides both secure and effortlessly understandable network traffic classification capabilities.

CONCLUSION

This work introduces a unified system that handles network traffic classification class imbalance through a Stacking Classifier combined with Explainable AI (XAI) methods. Through ensemble-of-ensemble methodology users achieve higher accuracy levels that target both uncommon and mission-critical traffic sequences which traditional algorithms tend to ignore. The system obtains diverse option boundaries and reduces bias through the use of multiple classifiers combined with a meta-learner. XAI integration creates explanations for model decisions which provide security analysts with dependable insights into the process. The system operates through Python and Flask for real-

Class Imbalance in Network Traffic Classification An Adaptive Weight Ensemble of Ensemble Learning Method

time prediction functions while delivering visual representations which make it appropriate for deployment within cybersecurity settings. A reliable scalable interpretive tool exists which improves threat detection while establishing trust between system and users.

References

Olczak, M. Fahlberg, J. Maki, J. Razavian, H. Jilert, M. Stark, M. Skorpil and M. Gordon, "Artificial intelligence for analyzing orthopedic trauma radiographs," *Acta Orthopaedica*, vol. 88, no. 6, pp. 581–586, 2017. doi: 10.1080/17453674.2017.1344459.

[1]. P. Rajpurkar et al., "MURA: Large Dataset for Abnormality Detection in Musculoskeletal Radiographs," arXiv preprint arXiv:1712.06957, 2017.

[2]. H. Tajbakhsh et al., "Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning?," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1299–1312, May 2016. doi: 10.1109/TMI.2016.2535302

[3]. R. R. Devi, B. Krishnaveni, "Detection of Bone Fracture in X-ray Images using Deep Learning Approach," 2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT), pp. 1–5, 2019. doi: 10.1109/ICIICT1.2019.8741445

[4]. S. Yadav and A. B. Solanki, "Bone Fracture Detection using CNN with Transfer Learning,"

2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), pp. 218–222, 2021. doi: 10.1109/ICAIS50930.2021.9396018

[5]. M. Ragab, M. Sharkas, S. Marshall, and J. Ren, "Breast cancer detection using deep convolutional neural networks and support vector machines," *PeerJ*, vol. 7, p. e6201, 2019. doi: 10.7717/peerj.6201

[6]. Ravindra and P. C. Kishore, "Automatic Detection of Bone Fractures using Convolutional Neural Networks," 2020 International Conference on Communication and Electronics Systems (ICCES), pp. 1366–1370, 2020. doi: 10.1109/ICCES48766.2020.9137896

[7]. S. Dey, A. Nandy, and S. Biswas, "Deep learning based fracture detection using x-ray images," 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2020, pp. 1–6. doi: 10.1109/ICCCNT49239.2020.9225384

[8]. J. Wang, W. Zhou, and Y. Wang, "Bone Fracture Detection from X-ray Images Using Deep Learning," *IEEE Access*, vol. 8, pp. 170174–170183, 2020. doi: 10.1109/ACCESS.2020.3024338

[9]. Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. doi: 10.1038/nature14539