



**CMR ENGINEERING COLLEGE**

**UGC AUTONOMOUS**

(Approved by AICTE - New Delhi. Affiliated to JNTUH and Accredited by NAAC & NBA)  
Kandlakoya (V), Medchal (M), Medchal - Malkajgiri (D)-501401



## **DEPARTMENT OF CSE-CYBER SECURITY**

### **Major Project**

On

## **Enhanced Botnet Detection in IOT Systems Using a Hybrid Machine Learning Approach**

**SUBMITTED BY : BATCH - 11**

**M.Krushitha : 228R1A62H0**

**M.Harika : 228R1A62H1**

**M.Mercy : 228R1A62H3**

**P.Nikitha : 238R5A6221**

**UNDER THE GUIDANCE OF,**

**Mr. Ch. Venkateswarlu**

**Assistant Professor,**

**DEPARTMENT OF CSE-CS**

**CMREC**

**HEAD OF THE DEPARTMENT,**

**Dr. M. Ashwitha Reddy**

**Associate Professor & HOD**

**DEPARTMENT OF CSE-CS,**

**CMREC**

# ABSTRACT

With the rise of IoT, botnet attacks have become increasingly harmful and harder to detect due to evolving attack vectors. This research proposes a stacking model (ACLR) combining ANN, CNN, RNN, and ML models to identify botnet activity using the UNSW-NB15 dataset, which includes nine attack types. The ACLR model achieved 96.98% testing accuracy, ROC-AUC of 0.9934, and PR-AUC of 0.9950, showing strong detection performance. K-fold cross-validation ( $k = 3, 5, 7, 10$ ) confirmed its robustness, with the best result at  $k = 5$  (97.49%). The model outperformed existing methods, making it a valuable tool for improving cybersecurity against modern botnet threats. This system effectively captures complex attack patterns across diverse traffic data. Its high generalizability makes it suitable for real-world network environments. As cyber threats continue to evolve, the ACLR model offers a scalable and intelligent solution for proactive threat detection.

# INTRODUCTION

The rapid growth of the Internet of Things (IoT) has enhanced connectivity across various sectors but also introduced severe security risks. Among these, botnet attacks have emerged as one of the most dangerous threats, exploiting vulnerable IoT devices to perform large-scale malicious activities. Traditional detection methods often struggle to identify such evolving attacks due to their dynamic and complex nature. To overcome these limitations, this project proposes a Hybrid Ensemble Learning Approach (ACLR) for efficient botnet attack detection in IoT systems. The ACLR model combines the strengths of ANN, CNN, RNN, and machine learning algorithms through a stacking mechanism to effectively capture spatial and temporal features of network traffic. Using the UNSW-NB15 dataset, which includes nine attack types, the model demonstrated superior accuracy and robustness compared to existing methods.

By achieving a testing accuracy of 96.98% and validating stability through k-fold cross-validation, the ACLR model proves to be a scalable and intelligent solution for proactive IoT threat detection. This approach contributes significantly to enhancing the overall cybersecurity posture of modern IoT environments.

# SOFTWARE REQUIREMENTS

- Operating system : Windows 7 Ultimate
- Coding Language : Python
- Front-End : Python
- Back-End : Django-ORM
- Designing : Html, css, javascript
- Data Base : MySQL (WAMP Server)

# HARDWARE REQUIREMENTS

- Processor : Pentium –IV
- RAM : 4 GB (min)
- Hard Disk : 20 GB
- Key Board : Standard Windows
- Mouse : Two or Three Button
- Monitor : SVGA

# EXISTING SYSTEM

Botnets are a major cybersecurity threat used for DDoS attacks, data theft, and spam. Traditional detection methods fail to detect encrypted, unknown, and evolving threats. Deep learning models like CNN, LSTM, and hybrid CNN-LSTM have shown high accuracy (up to 99.57%) and cost efficiency in detecting botnet traffic, especially in IoT environments. ANN and BLSTM-RNN models also outperform traditional classifiers like SVM and Naive Bayes. These models enhance detection capabilities while reducing implementation costs by up to 75%.

## **Disadvantages (Limitations of Existing System):**

- Poor Detection of Unknown Botnets – Existing models struggle to identify new or unseen botnet types that differ from known attack patterns.
- Inability to Handle Encrypted Traffic – Encrypted communication hides malicious content, making it difficult for models to analyze and detect threats accurately.
- Lack of Adaptability – Most systems perform well only on specific datasets and fail to adapt to changing IoT environments or evolving attack methods.
- High Implementation and Maintenance Cost – Advanced detection models require high computational resources and frequent updates, increasing cost and complexity.
- Evasion by Attackers – Attackers use advanced evasion techniques to disguise malicious traffic, allowing botnets to bypass traditional detection systems.

# PROPOSED SYSTEM

This research introduces a stacking model named ACLR, which integrates Artificial Neural Network (ANN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and traditional Machine Learning (ML) models to achieve efficient and accurate botnet attack detection in IoT environments. The model is designed to classify multiple types of attacks, including DoS, worms, backdoors, fuzzers, reconnaissance, shellcode, exploits, and generic attacks, using a well-preprocessed dataset. Performance evaluation is conducted using key metrics such as accuracy, precision, recall, F1-score, and ROC-AUC, ensuring comprehensive assessment of the model's detection capability.

To validate the model's stability and generalization, k-fold cross-validation is performed with  $k = 3, 5, 7,$  and  $10$ , achieving consistent results across all folds. The proposed ACLR framework demonstrates superior detection accuracy and robustness compared to existing methods. By leveraging the combined strengths of deep learning and machine learning algorithms, this hybrid ensemble approach effectively captures complex patterns in IoT network traffic and offers a scalable, intelligent solution for modern cybersecurity challenges.



# Advantages

- Combines Multiple Learning Models – Integrates ANN, CNN, RNN, and ML models to enhance detection accuracy and learning capability.
- Effective for a Wide Range of Attacks – Accurately identifies various attack types such as DoS, worms, backdoors, and reconnaissance.
- Comprehensive Performance Evaluation – Assessed using multiple metrics like accuracy, precision, recall, F1-score, and ROC-AUC.
- Robust Validation – Ensures reliability and stability through k-fold cross-validation with multiple values of k.
- Outperforms Existing Models – Achieves higher accuracy and better generalization compared to traditional and standalone deep learning methods.

## LITERATURE SURVEY

Reference	Authors	Year	Technique / Model	Dataset / Environment	Key Findings / Outcomes
[1]	N. Koroniotis et al.	2019	Dataset creation	IoT networks	Developed Bot-IoT dataset for network forensic analytics
[2]	O. Ibitoye et al.	2019	<u>DL-based</u> intrusion detection	IoT networks	Analyzed adversarial attacks on deep learning models
[3]	M. Shahhosseini et al.	2022	Deep learning	Raw network traffic	Proposed botnet detection using raw traffic data
[4]	S. Homayoun et al.	2018	Deep learning (BoTShark)	Botnet traffic	Accurate detection of botnet traffic using DL
[5]	M. Ge et al.	2019	Deep learning	IoT networks	Efficient intrusion detection in IoT networks
[6]	M. A. Ferrag et al.	2020	DL comparative study	Multiple datasets	Studied DL approaches for intrusion detection

## LITERATURE SURVEY

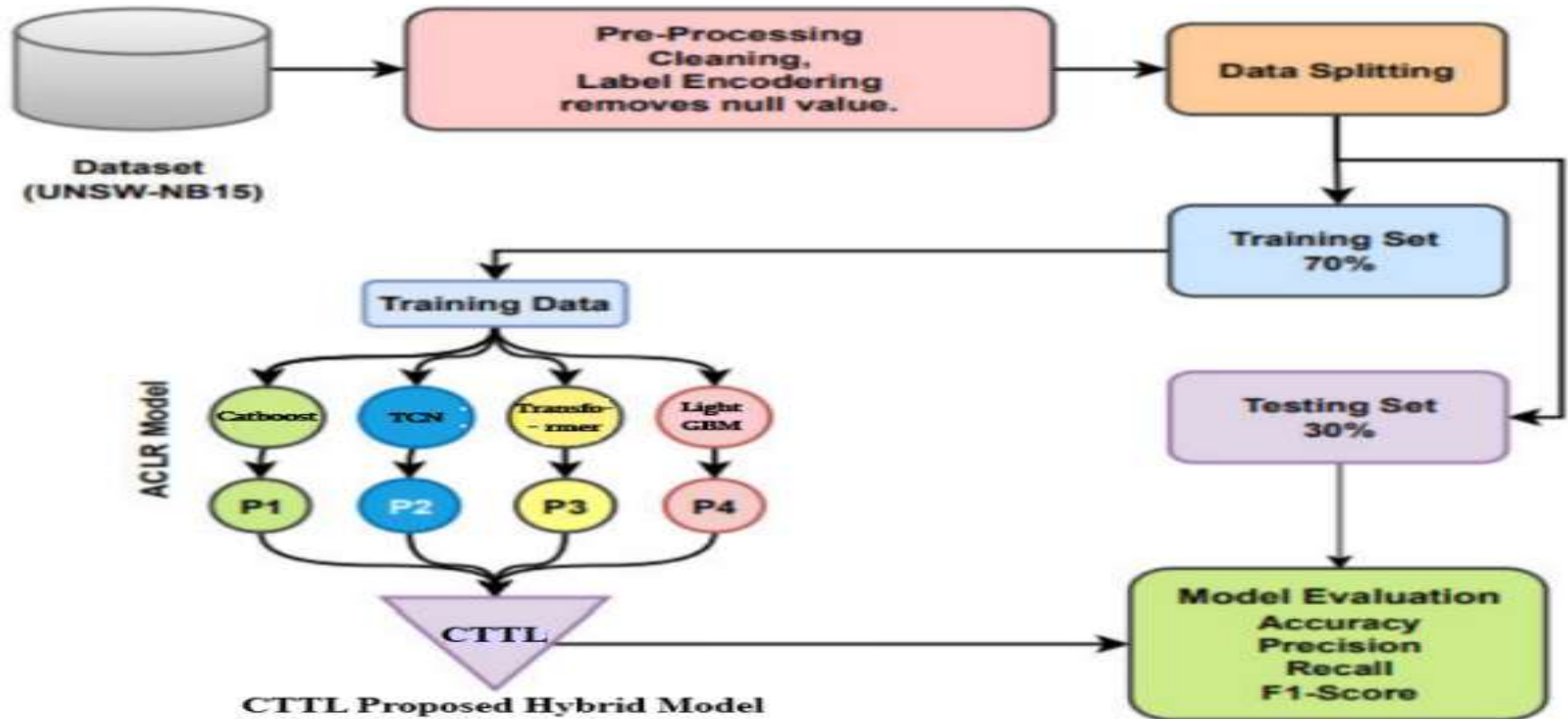
[7]	T. <u>Hasan et al.</u>	2023	Hybrid DL	Industrial IoT	Securing IIoT against botnet attacks
[8]	D. T. Son et al.	2022	Deep learning	IoT networks	Detection of botnet activity using DL techniques
[9]	M. Gandhi, S. Srivatsa	2008	NIDS	Network traffic	Detection and prevention of network attacks
[10]	J. Liu et al.	2019	DL	IoT botnet	Detection of <u>IoT</u> botnets using deep learning
[11]	C. D. McDermott et al.	2018	DL	IoT networks	Botnet detection using deep learning approaches
[12]	S. Sriram et al.	2020	DL (Network flow-based)	IoT botnet traffic	Accurate botnet detection using network flows
[13]	B. Nugraha et al.	2020	DL	Botnet datasets	Performance evaluation of <u>DL-based</u> detection

## LITERATURE SURVEY

[14]	P. Karunakaran	2020	DL (DGA classification )	Cybersecurity datasets	Effective classification of DGA domains
[15]	N. Elsayed et al.	2023	Economic DL model	IoT botnet traffic	Efficient botnet detection in IoT networks
[16]	M. A. Haq, M. A. Rahim Khan	2022	DNN	Botnet datasets	Deep neural network-based botnet detection & classification
[17]	I. H. Sarker	2021	DL overview	Various	Comprehensive review of neural network and DL in <u>cybersecurity</u>
[18]	A. A. Ahmed et al.	2022	ANN	CTU-13 dataset	High accuracy (~99%) in botnet attack detection
[19]	I. Letteri et al.	2018	ANN	SDN environment	Effective <u>DDoS</u> detection using ANN
[20]	T. H. H. Aldhyani, H. Alkahtani	2022	CNN-LSTM	Autonomous vehicles IoT	Achieved specificity 93%, F1 score 100% for attack detection

# SYSTEM DESIGN

The proposed CTTL Hybrid ACLR model uses the UNSW-NB15 dataset, which is preprocessed and split into 70% training and 30% testing sets. Four base models — CatBoost, TCN, Transformer, and LightGBM — learn different traffic patterns, and their outputs are combined using a CTTL stacking layer. The final model's performance is evaluated using accuracy, precision, recall, and F1-score metrics.

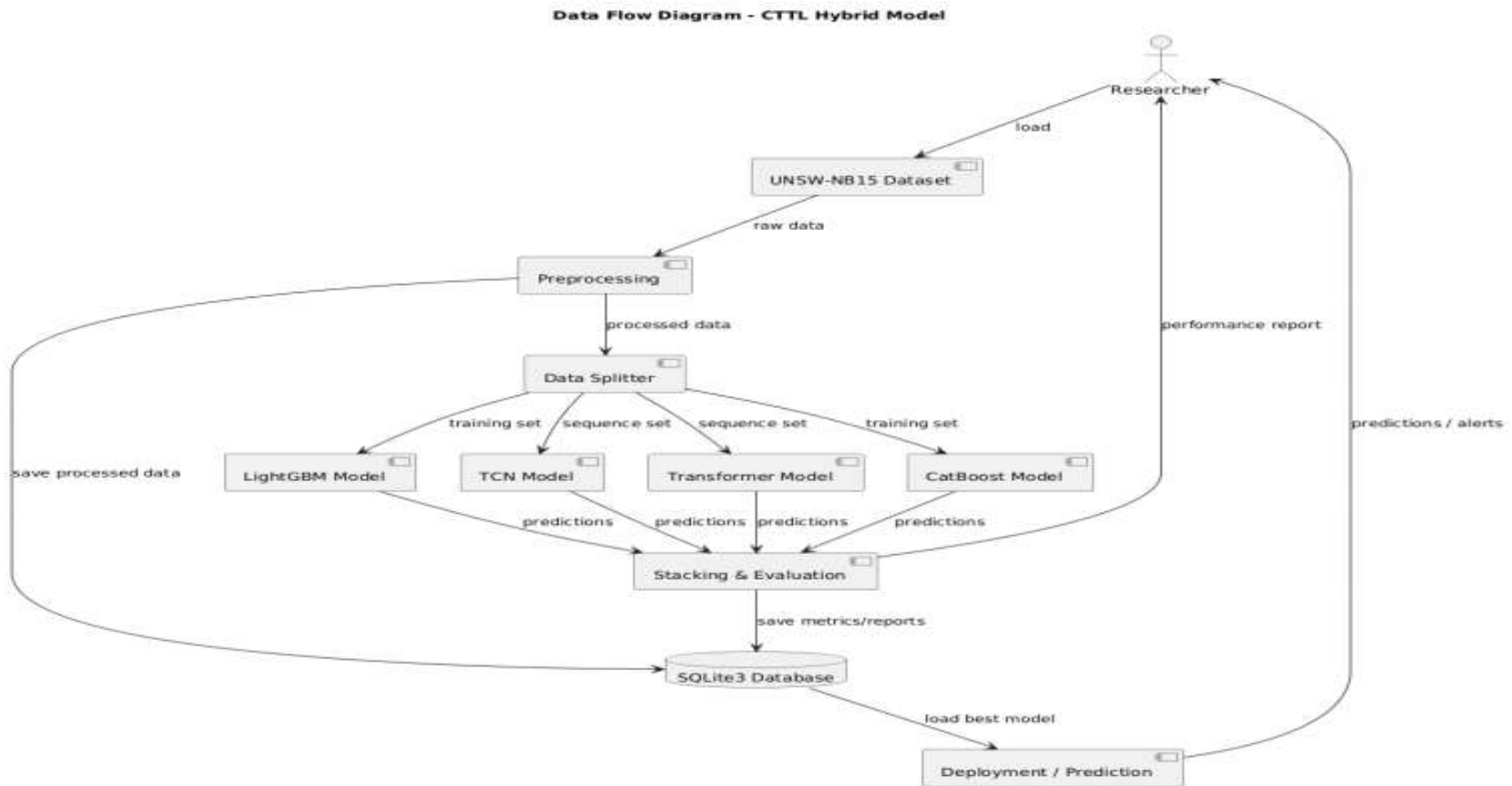


# SYSTEM DESIGN

## **Data Flow Diagram**

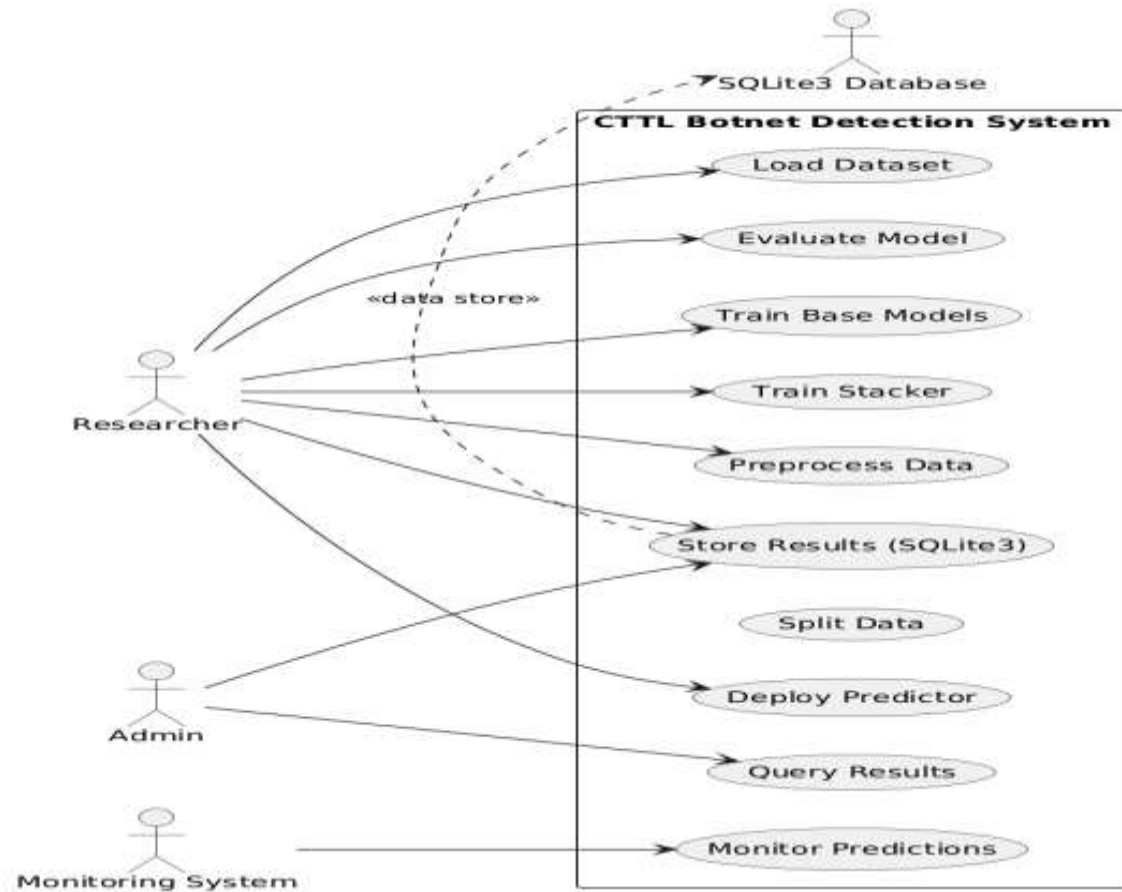
A Data Flow Diagram (DFD) represents how data moves within a system, showing its flow between processes, data stores, and external entities. It illustrates how input data is processed, stored, and transformed into output. External entities act as data sources or destinations, while processes perform operations on data. Data flows indicate the direction of data movement. DFDs provide a clear, high-level view of system functionality through Level 0, Level 1, and Level 2 diagrams.

# SYSTEM DESIGN



# UML DIAGRAMS

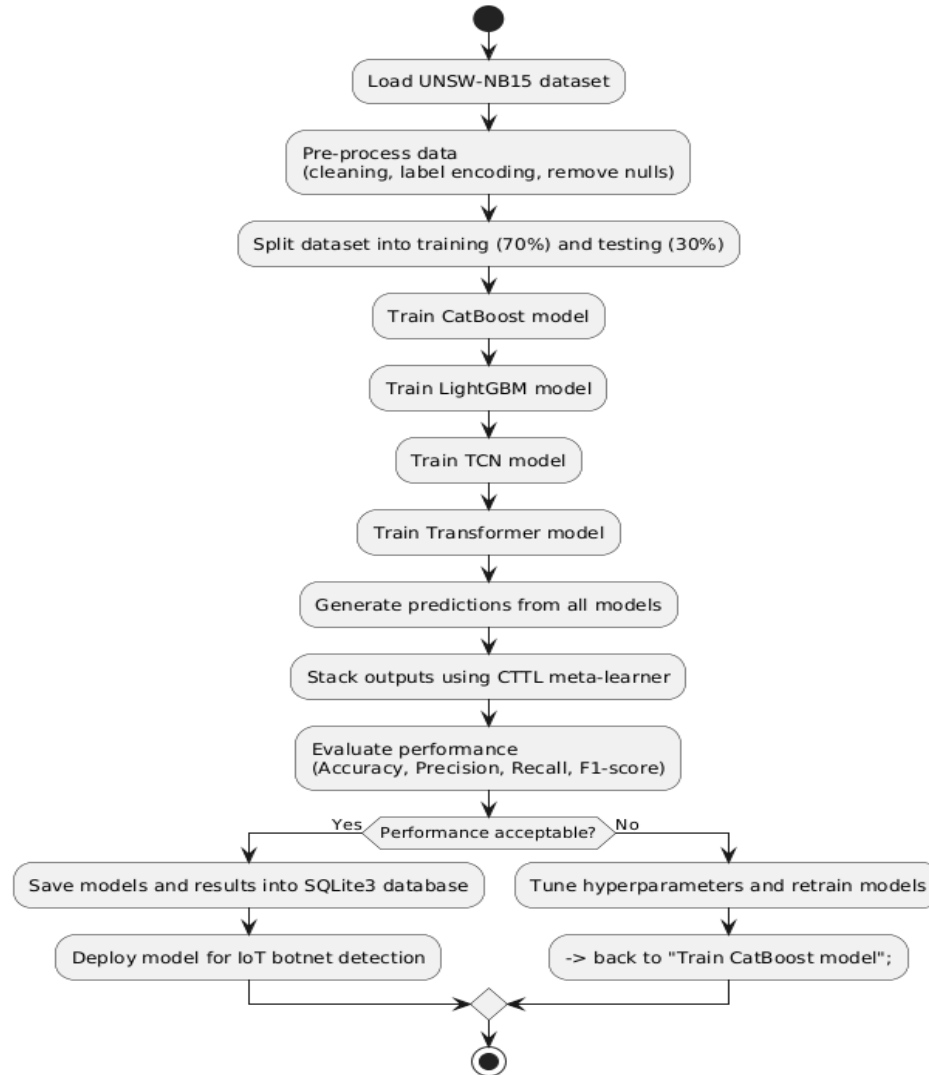
## Use case Diagram





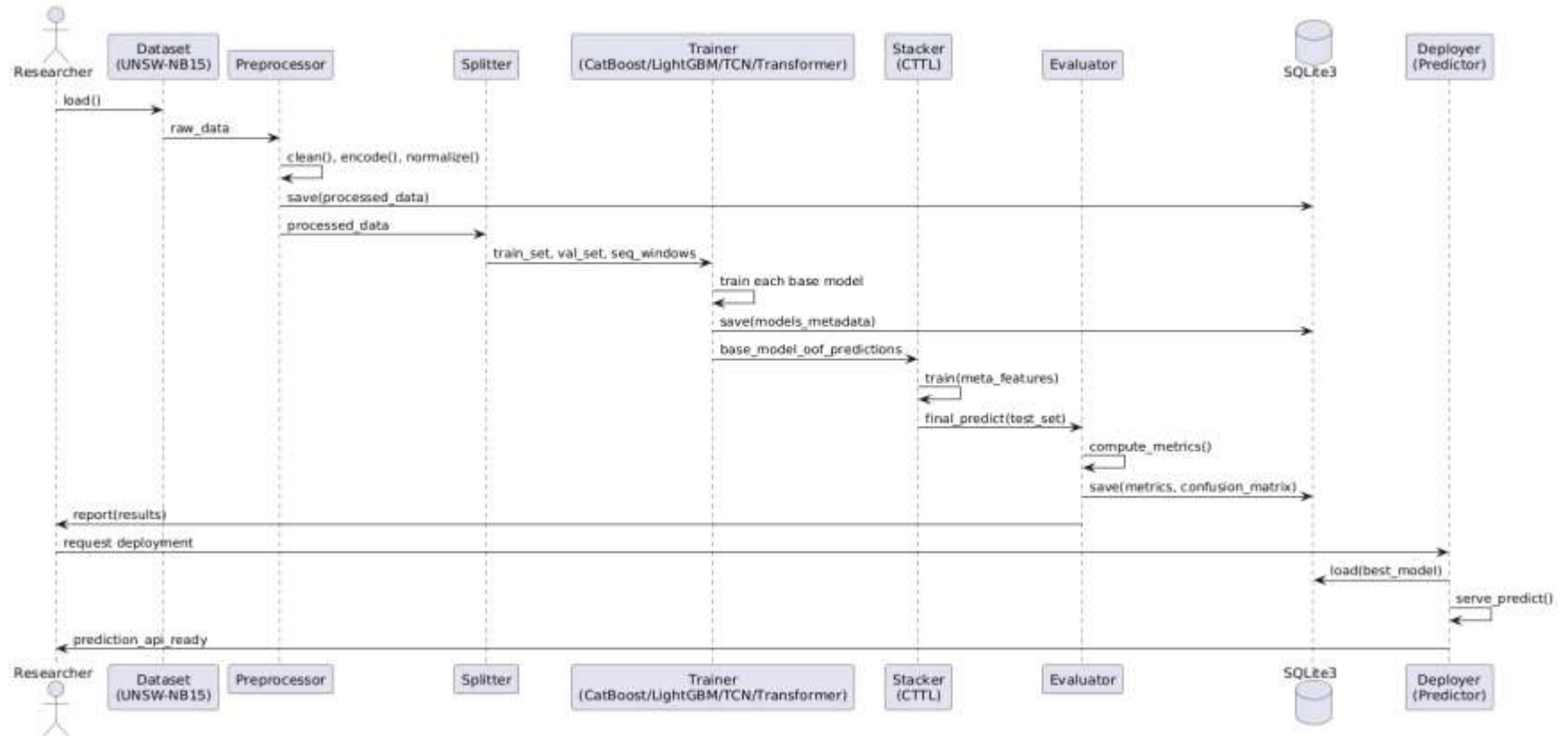
# UML DIAGRAMS

## Activity Diagram



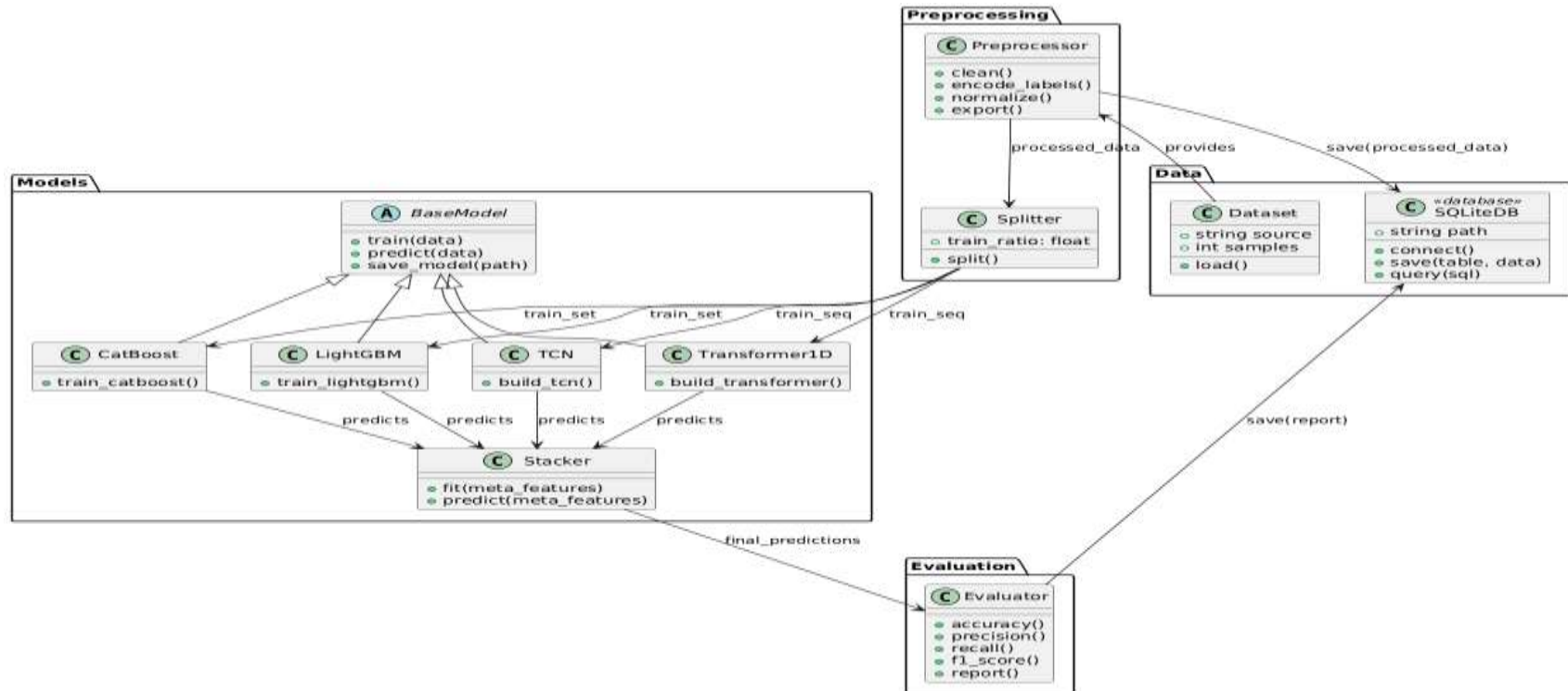
# UML DIAGRAMS

## Sequence Diagram



# UML DIAGRAMS

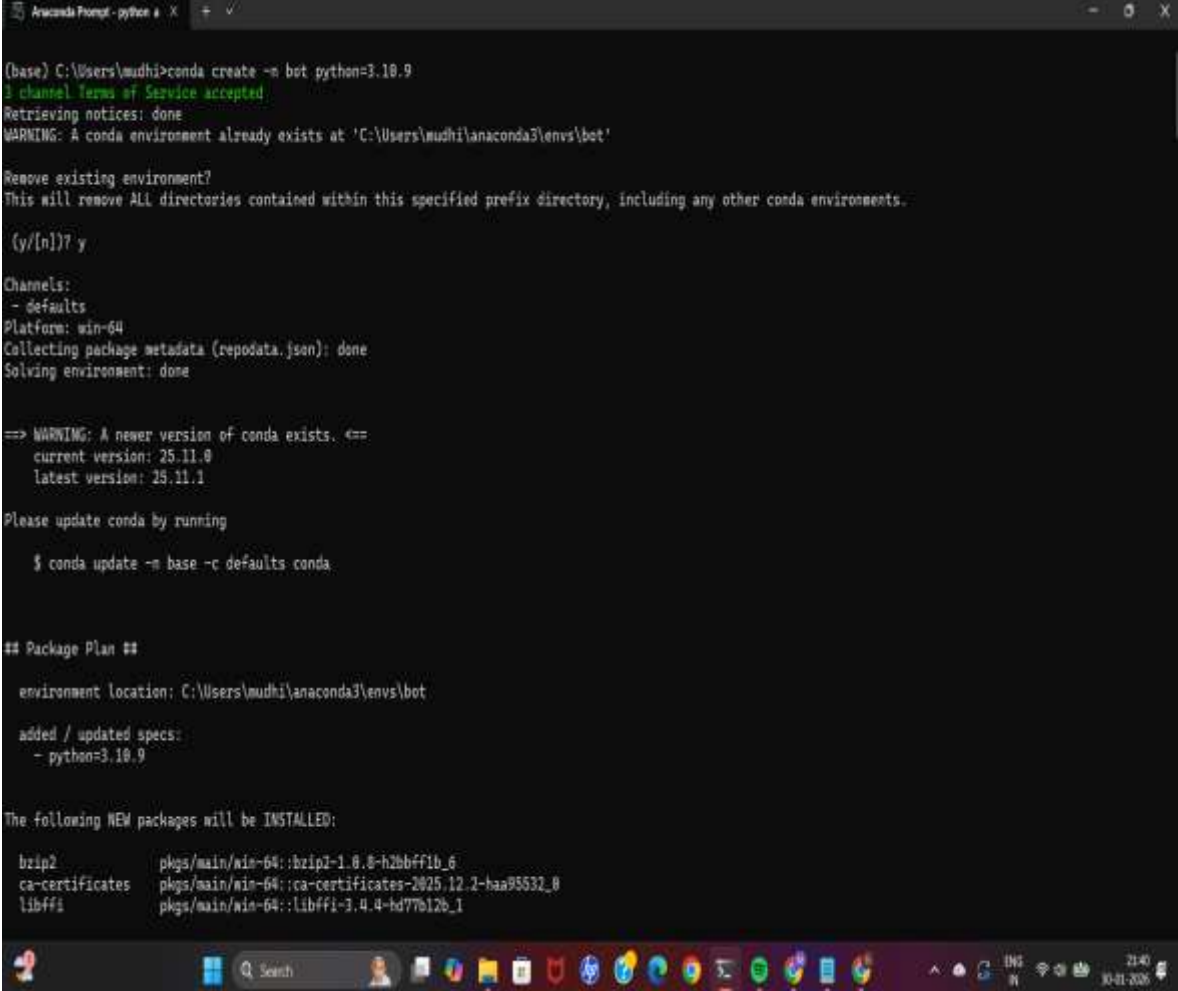
## Class Diagram



# IMPLEMENTATION

**Title:** Python Environment and Dependency Setup

- An isolated **Anaconda virtual environment (bot)** is created to avoid package conflicts.
- Python version **3.10.9** is used for compatibility with machine learning libraries.
- Required packages such as Flask, NumPy, Joblib, Scikit-learn, LightGBM, and PyTorch are installed using `pip`.
- This setup ensures a stable runtime environment for hybrid ML model execution.



```
(base) C:\Users\mudhi>conda create -n bot python=3.10.9
1 channel Terms of Service accepted
Retrieving notices: done
WARNING: A conda environment already exists at 'C:\Users\mudhi\anaconda3\envs\bot'

Remove existing environment?
This will remove ALL directories contained within this specified prefix directory, including any other conda environments.

(y/[n])? y

Channels:
- defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 25.11.0
  latest version: 25.11.1

Please update conda by running

  $ conda update -n base -c defaults conda

## Package Plan ##

  environment location: C:\Users\mudhi\anaconda3\envs\bot

  added / updated specs:
    - python=3.10.9

The following NEW packages will be INSTALLED:

  bzip2                pkgs/main/win-64::bzip2-1.0.8-h2b6ff1b_6
  ca-certificates      pkgs/main/win-64::ca-certificates-2025.12.2-haa95532_0
  libffi               pkgs/main/win-64::libffi-3.4.4-hd77b12b_1
```

# IMPLEMENTATION

**Title:** Web Framework Configuration

- Flask (**version 2.2.2**) is installed to build the web interface of the system.
- Supporting libraries such as **Werkzeug, Jinja2, Click, and ItsDangerous** are configured.
- Flask acts as the backend server that connects user input with trained ML models.
- The framework enables real-time prediction of botnet attacks through a browser interface.

```
Proceed ([y]/n)? y

Downloading and Extracting Packages:

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#   $ conda activate bot
#
# To deactivate an active environment, use
#
#   $ conda deactivate

(base) C:\Users\mushi>conda activate bot

(bot) C:\Users\mushi>cd C:\Users\mushi\OneDrive\Pictures\plantuml\8. Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment

(bot) C:\Users\mushi\OneDrive\Pictures\plantuml\8. Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment>python app.py
Traceback (most recent call last):
  File "C:\Users\mushi\OneDrive\Pictures\plantuml\8. Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment\app.py", line 1, in <module>
    import torch
ModuleNotFoundError: No module named 'torch'

(bot) C:\Users\mushi\OneDrive\Pictures\plantuml\8. Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment>torch==2.5.0

'torch' is not recognized as an internal or external command,
operable program or batch file.

(bot) C:\Users\mushi\OneDrive\Pictures\plantuml\8. Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment>pip install torch==2.5.0
Collecting torch==2.5.0
  Using cached torch-2.5.0-cp310-cp310-win_amd64.whl.metadata (38 kB)
```

# IMPLEMENTATION

## Title: Dependency Conflict Resolution

- Initial execution produced errors due to version mismatches between Flask and Werkzeug.
- Werkzeug is downgraded to a compatible version (**2.2.2**) to resolve import errors.
- This step ensures smooth communication between Flask components.
- Proper dependency alignment is critical for stable application execution.

```
Using cached filelock-3.20.3-py3-none-any.whl.metadata (2.1 kB)
Collecting typing_extensions>=4.8.0 (from torch==2.5.0)
Using cached typing_extensions-4.15.0-py3-none-any.whl.metadata (3.3 kB)
Collecting networkx (from torch==2.5.0)
Using cached networkx-3.4.2-py3-none-any.whl.metadata (6.3 kB)
Collecting Jinja2 (from torch==2.5.0)
Using cached Jinja2-3.1.6-py3-none-any.whl.metadata (2.9 kB)
Collecting fsspec (from torch==2.5.0)
Using cached fsspec-2026.1.0-py3-none-any.whl.metadata (18 kB)
Collecting sympy==1.13.1 (from torch==2.5.0)
Using cached sympy-1.13.1-py3-none-any.whl.metadata (12 kB)
Collecting mpmath<1.4, >=1.1.0 (from sympy==1.13.1->torch==2.5.0)
Using cached mpmath-1.3.0-py3-none-any.whl.metadata (8.6 kB)
Collecting MarkupSafe>=2.0 (from Jinja2->torch==2.5.0)
Using cached MarkupSafe-3.0.3-cp310-cp310-win_amd64.whl.metadata (2.8 kB)
Using cached torch-2.5.0-cp310-cp310-win_amd64.whl (283.1 MB)
Using cached sympy-1.13.1-py3-none-any.whl (6.2 MB)
Using cached mpmath-1.3.0-py3-none-any.whl (536 kB)
Using cached typing_extensions-4.15.0-py3-none-any.whl (44 kB)
Using cached filelock-3.20.3-py3-none-any.whl (16 kB)
Using cached fsspec-2026.1.0-py3-none-any.whl (281 kB)
Using cached Jinja2-3.1.6-py3-none-any.whl (134 kB)
Using cached MarkupSafe-3.0.3-cp310-cp310-win_amd64.whl (15 kB)
Using cached networkx-3.4.2-py3-none-any.whl (1.7 MB)
Installing collected packages: mpmath, typing_extensions, sympy, networkx, MarkupSafe, fsspec, filelock, Jinja2, torch
Successfully installed MarkupSafe-3.0.3 filelock-3.20.3 fsspec-2026.1.0 Jinja2-3.1.6 mpmath-1.3.0 networkx-3.4.2 sympy-1.13.1 torch-2.5.0 typing_extensions-4.15.0

(bot) C:\Users\mudhi\OneDrive\Pictures\plantuml.plum\8. Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment>python app.py
C:\Users\mudhi\anaconda3\envs\bot\Lib\site-packages\torch\_subclasses\functional_tensor.py:295: UserWarning: Failed to initialize NumPy: No module named 'numpy' (Triggered internally at C:\actions-runner\work\pytorch\pytorch\builder\windows\pytorch\torch\csoc\utils\tensor_numpy.cpp:84.)
  cpu = _conversion_method_template(device=torch.device("cpu"))
Traceback (most recent call last):
  File "C:\Users\mudhi\OneDrive\Pictures\plantuml.plum\8. Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment\app.py", line 3, in <module>
    from flask import Flask, render_template, request, redirect, url_for, flash, session
ModuleNotFoundError: No module named 'flask'

(bot) C:\Users\mudhi\OneDrive\Pictures\plantuml.plum\8. Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment>Flask==2.2.2
```

# IMPLEMENTATION

**Title:** ML Model Support Libraries

- **Scikit-learn** is installed to support classical machine learning models.
- **Joblib** is used to load trained models and feature scalers ( `.sav`, `.pkl` files).
- **LightGBM** is installed to support gradient boosting-based botnet detection.
- These libraries form the traditional ML part of the hybrid approach.

```
(bot) C:\Users\mudhi\OneDrive\Pictures\plant_unl.plant\8. Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment>pip install Flask==2.2.2
Collecting Flask==2.2.2
  Using cached Flask-2.2.2-py3-none-any.whl.metadata (3.9 kB)
Collecting Werkzeug==2.2.2 (from Flask==2.2.2)
  Using cached Werkzeug-2.2.2-py3-none-any.whl.metadata (4.0 kB)
Requirement already satisfied: Jinja2>=3.0 in c:\users\mudhi\anaconda3\envs\bot\lib\site-packages (from Flask==2.2.2) (3.1.6)
Collecting itsdangerous==2.0 (from Flask==2.2.2)
  Using cached itsdangerous-2.0-py3-none-any.whl.metadata (1.9 kB)
Collecting click==8.0 (from Flask==2.2.2)
  Using cached click-8.0.3.1-py3-none-any.whl.metadata (2.6 kB)
Collecting colorama (from click==8.0.3.1-py3-none-any.whl)
  Using cached colorama-0.4.6-py2.py3-none-any.whl.metadata (17 kB)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\mudhi\anaconda3\envs\bot\lib\site-packages (from Jinja2>=3.0->Flask==2.2.2) (2.0.3)
Using cached Flask-2.2.2-py3-none-any.whl (101 kB)
Using cached click-8.0.3.1-py3-none-any.whl (101 kB)
Using cached itsdangerous-2.0-py3-none-any.whl (16 kB)
Using cached werkzeug-2.2.2-py3-none-any.whl (225 kB)
Using cached colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Installing collected packages: Werkzeug, itsdangerous, colorama, click, Flask
Successfully installed Flask-2.2.2 Werkzeug-2.2.2 click-8.0.3.1 colorama-0.4.6 itsdangerous-2.0.0

(bot) C:\Users\mudhi\OneDrive\Pictures\plant_unl.plant\8. Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment>python app.py
C:\Users\mudhi\anaconda3\envs\bot\lib\site-packages\torch\_subclasses\functional_tensor.py:295: UserWarning: Failed to initialize NumPy: No module named 'numpy' (Triggered internally at C:\actions-runner\work\pytorch\pytorch\builder\windows\pytorch\torch\csrc\utils\tensor_numpy.cpp:84.)
  cpu = _conversion_method_template(device=torch.device("cpu"))
Traceback (most recent call last):
  File "C:\Users\mudhi\OneDrive\Pictures\plant_unl.plant\8. Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment\app.py", line 3, in <module>
    from flask import Flask, render_template, request, redirect, url_for, flash, session
  File "C:\Users\mudhi\anaconda3\envs\bot\lib\site-packages\flask\__init__.py", line 5, in <module>
    from .app import Flask as Flask
  File "C:\Users\mudhi\anaconda3\envs\bot\lib\site-packages\flask\app.py", line 30, in <module>
    from werkzeug.urls import url_quote
ImportError: cannot import name 'url_quote' from 'werkzeug.urls' (C:\Users\mudhi\anaconda3\envs\bot\lib\site-packages\werkzeug\urls.py)

(bot) C:\Users\mudhi\OneDrive\Pictures\plant_unl.plant\8. Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment>pip install Werkzeug==2.2.2
Collecting Werkzeug==2.2.2
```



# IMPLEMENTATION

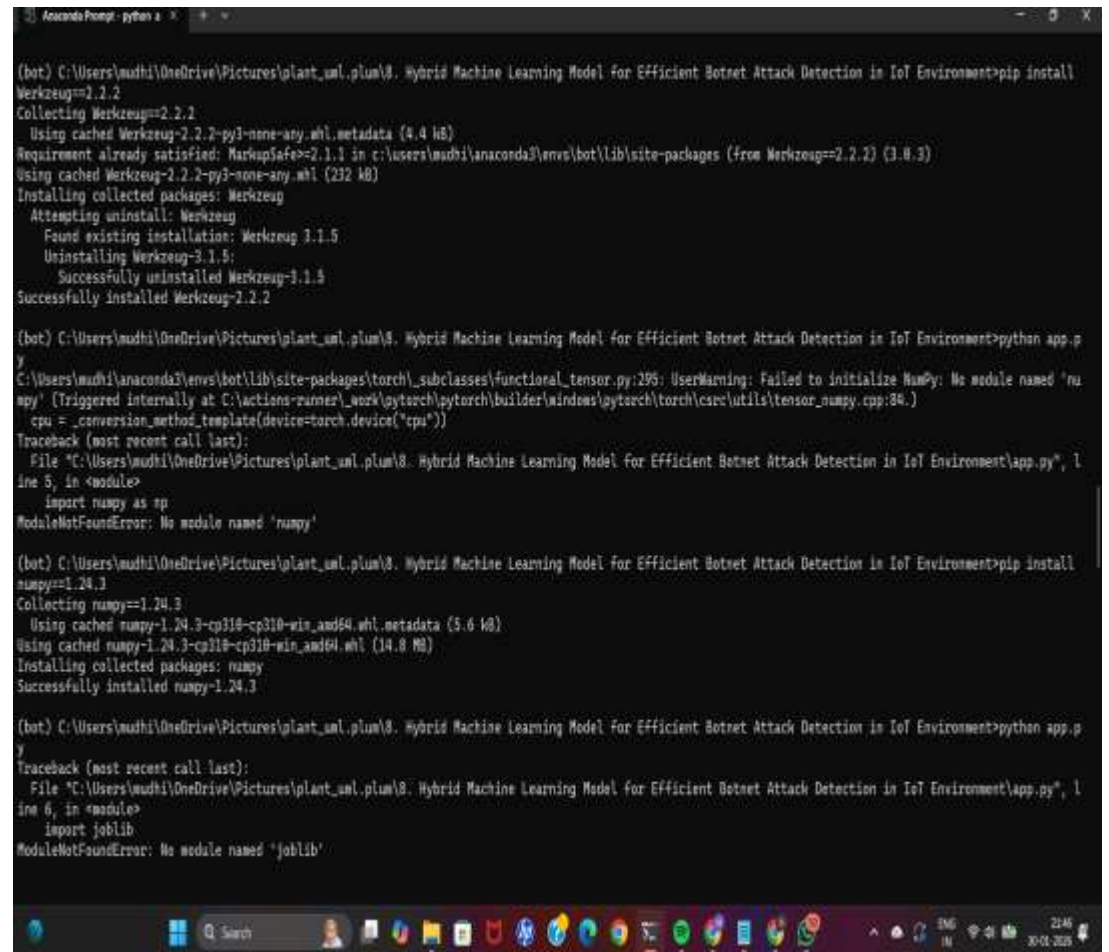
**Title:** PyTorch Model Integration

- **PyTorch (torch 2.5.0)** is installed for deep learning model execution.

- The trained **Attention-based CNN-BiLSTM model** is loaded using `torch.load()`.

- CPU-based inference is used for model prediction.

- This deep learning model captures complex temporal and spatial patterns in IoT traffic.



```
(bot) C:\Users\mudhi\OneDrive\Pictures\plant_uwl.plum\8. Hybrid Machine Learning Model For Efficient Botnet Attack Detection in IoT Environment>pip install Werkzeug==2.2.2
Collecting Werkzeug==2.2.2
  Using cached Werkzeug-2.2.2-py3-none-any.whl.metadata (4.4 kB)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\mudhi\anaconda3\envs\bot\lib\site-packages (from Werkzeug==2.2.2) (3.0.3)
Using cached Werkzeug-2.2.2-py3-none-any.whl (232 kB)
Installing collected packages: Werkzeug
  Attempting uninstall: Werkzeug
    Found existing installation: Werkzeug 3.1.5
    Uninstalling Werkzeug-3.1.5:
      Successfully uninstalled Werkzeug-3.1.5
Successfully installed Werkzeug-2.2.2

(bot) C:\Users\mudhi\OneDrive\Pictures\plant_uwl.plum\8. Hybrid Machine Learning Model For Efficient Botnet Attack Detection in IoT Environment>python app.py
C:\Users\mudhi\anaconda3\envs\bot\lib\site-packages\torch\_subclasses\functional_tensor.py:296: UserWarning: Failed to initialize NumPy: No module named 'numpy' (Triggered internally at C:\actions-runner\work\pytorch\pytorch\builder\windows\pytorch\torch\src\utils\tensor_numpy.cpp:86.)
  cpu = _conversion_method_template(device=torch.device("cpu"))
Traceback (most recent call last):
  File "C:\Users\mudhi\OneDrive\Pictures\plant_uwl.plum\8. Hybrid Machine Learning Model For Efficient Botnet Attack Detection in IoT Environment\app.py", line 5, in <module>
    import numpy as np
ModuleNotFoundError: No module named 'numpy'

(bot) C:\Users\mudhi\OneDrive\Pictures\plant_uwl.plum\8. Hybrid Machine Learning Model For Efficient Botnet Attack Detection in IoT Environment>pip install numpy==1.24.3
Collecting numpy==1.24.3
  Using cached numpy-1.24.3-cp310-cp310-win_amd64.whl.metadata (5.6 kB)
Using cached numpy-1.24.3-cp310-cp310-win_amd64.whl (14.8 MB)
Installing collected packages: numpy
Successfully installed numpy-1.24.3

(bot) C:\Users\mudhi\OneDrive\Pictures\plant_uwl.plum\8. Hybrid Machine Learning Model For Efficient Botnet Attack Detection in IoT Environment>python app.py
Traceback (most recent call last):
  File "C:\Users\mudhi\OneDrive\Pictures\plant_uwl.plum\8. Hybrid Machine Learning Model For Efficient Botnet Attack Detection in IoT Environment\app.py", line 6, in <module>
    import joblib
ModuleNotFoundError: No module named 'joblib'
```



# IMPLEMENTATION

## Title: Hybrid Model Loading Process

- Multiple trained models are loaded:

- Classical ML model (model.sav)
- Feature scaler (scaler.pkl)
- Deep learning model (att\_cnn\_bilstm.pth)

- Models are loaded at application startup to reduce prediction latency.

- This enables ensemble-based decision making for botnet detection.

```
ModuleNotFoundError: No module named 'joblib'

(bot) C:\Users\mudhi\OneDrive\Pictures\plant_uwl.plum\8. Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment>pip install joblib==1.4.2
Collecting joblib==1.4.2
  Using cached joblib-1.4.2-py3-none-any.whl.metadata (5.4 kB)
Using cached joblib-1.4.2-py3-none-any.whl (301 kB)
Installing collected packages: joblib
Successfully installed joblib-1.4.2

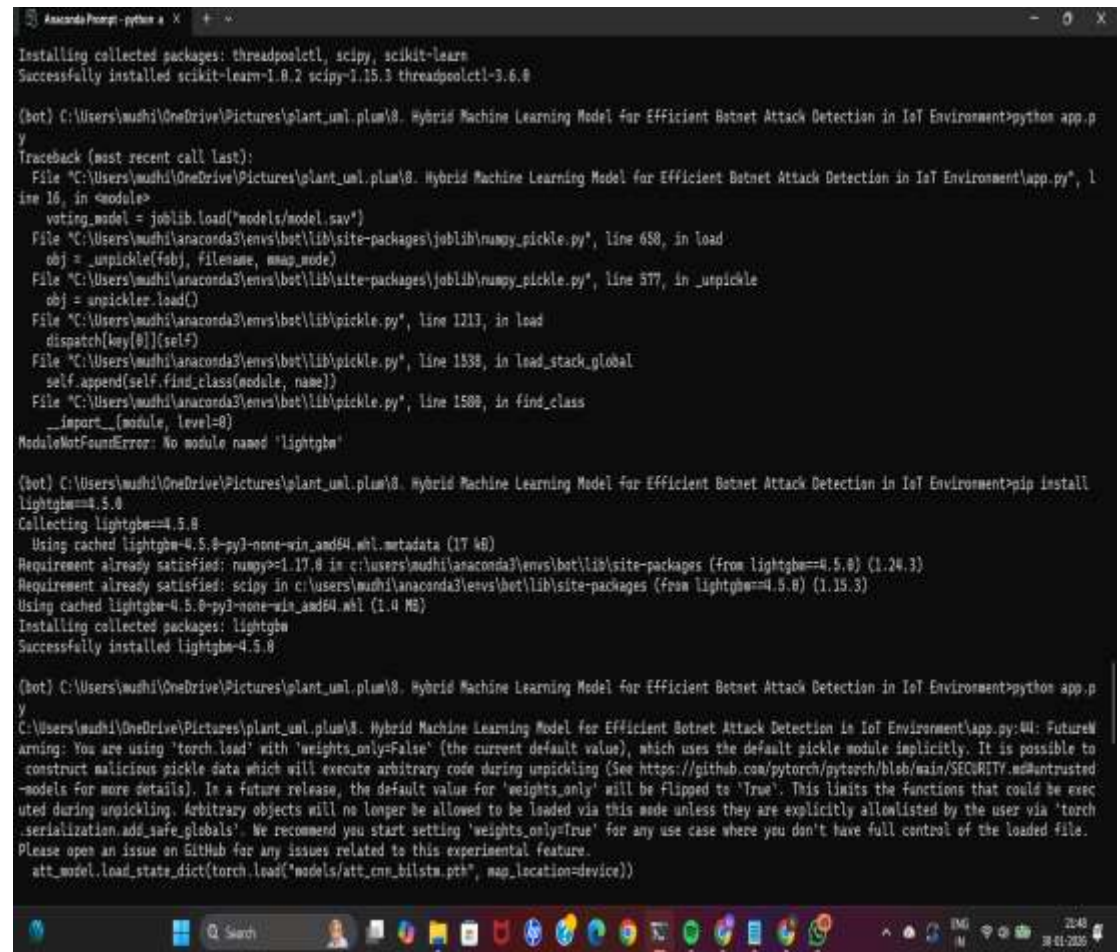
(bot) C:\Users\mudhi\OneDrive\Pictures\plant_uwl.plum\8. Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment>python app.py
Traceback (most recent call last):
  File "C:\Users\mudhi\OneDrive\Pictures\plant_uwl.plum\8. Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment\app.py", line 15, in <module>
    scaler = joblib.load("model/scaler.pkl")
  File "C:\Users\mudhi\anaconda3\envs\bot\lib\site-packages\joblib\numpy_pickle.py", line 658, in load
    obj = _unpickle(fobj, filename, mmap_mode)
  File "C:\Users\mudhi\anaconda3\envs\bot\lib\site-packages\joblib\numpy_pickle.py", line 577, in _unpickle
    obj = unpickler.load()
  File "C:\Users\mudhi\anaconda3\envs\bot\lib\pickle.py", line 1213, in load
    dispatch[key[0]](self)
  File "C:\Users\mudhi\anaconda3\envs\bot\lib\pickle.py", line 1538, in load_stack_global
    self.append(self.find_class(module, name))
  File "C:\Users\mudhi\anaconda3\envs\bot\lib\pickle.py", line 1580, in find_class
    __import__(module, level=8)
ModuleNotFoundError: No module named 'sklearn'

(bot) C:\Users\mudhi\OneDrive\Pictures\plant_uwl.plum\8. Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment>pip install scikit-learn==1.0.2
Collecting scikit-learn==1.0.2
  Using cached scikit_learn-1.0.2-cp310-cp310-win_and64.whl.metadata (18 kB)
Requirement already satisfied: numpy>=1.14.6 in c:\users\mudhi\anaconda3\envs\bot\lib\site-packages (from scikit-learn==1.0.2) (1.24.3)
Collecting scipy>=1.1.0 (from scikit-learn==1.0.2)
  Using cached scipy-1.15.3-cp310-cp310-win_and64.whl.metadata (60 kB)
Requirement already satisfied: joblib>=0.11 in c:\users\mudhi\anaconda3\envs\bot\lib\site-packages (from scikit-learn==1.0.2) (1.4.2)
Collecting threadpoolctl>=2.0.0 (from scikit-learn==1.0.2)
  Using cached threadpoolctl-3.6.0-py3-none-any.whl.metadata (13 kB)
Using cached scikit_learn-1.0.2-cp310-cp310-win_and64.whl (7.2 MB)
Using cached scipy-1.15.3-cp310-cp310-win_and64.whl (41.3 MB)
Using cached threadpoolctl-3.6.0-py3-none-any.whl (18 kB)
```

# IMPLEMENTATION

## Title: Flask Application Execution

- The Flask application is executed using `python app.py`.
- The server starts successfully without runtime errors.
- Flask runs in **development mode** on a local machine.
- The system is now ready to accept input data for prediction.



```

Anaconda Prompt - python 3.8
Installing collected packages: threadpoolctl, scipy, scikit-learn
Successfully installed scikit-learn-1.0.2 scipy-1.15.3 threadpoolctl-3.6.0

(C:\Users\mudhi\OneDrive\Pictures\plant_uwl.plum\8. Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment>python app.py
Traceback (most recent call last):
  File "C:\Users\mudhi\OneDrive\Pictures\plant_uwl.plum\8. Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment\app.py", line 16, in <module>
    voting_model = joblib.load("models\model.sav")
  File "C:\Users\mudhi\anaconda3\envs\bot\lib\site-packages\joblib\numpy_pickle.py", line 658, in load
    obj = _unpickle(fobj, filename, mmap_mode)
  File "C:\Users\mudhi\anaconda3\envs\bot\lib\site-packages\joblib\numpy_pickle.py", line 577, in _unpickle
    obj = unpickler.load()
  File "C:\Users\mudhi\anaconda3\envs\bot\lib\pickle.py", line 1213, in load
    dispatch[key[0]](self)
  File "C:\Users\mudhi\anaconda3\envs\bot\lib\pickle.py", line 1538, in load_stack_global
    self.append(self.find_class(module, name))
  File "C:\Users\mudhi\anaconda3\envs\bot\lib\pickle.py", line 1589, in find_class
    __import__(module, level=0)
ModuleNotFoundError: No module named 'lightgbm'

(C:\Users\mudhi\OneDrive\Pictures\plant_uwl.plum\8. Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment>pip install lightgbm==4.5.0
Collecting lightgbm==4.5.0
  Using cached lightgbm-4.5.0-py3-none-win_and64.whl.metadata (17 kB)
Requirement already satisfied: numpy<1.17.0 in c:\users\mudhi\anaconda3\envs\bot\lib\site-packages (from lightgbm==4.5.0) (1.24.3)
Requirement already satisfied: scipy in c:\users\mudhi\anaconda3\envs\bot\lib\site-packages (from lightgbm==4.5.0) (1.15.3)
Using cached lightgbm-4.5.0-py3-none-win_and64.whl (1.4 MB)
Installing collected packages: lightgbm
Successfully installed lightgbm-4.5.0

(C:\Users\mudhi\OneDrive\Pictures\plant_uwl.plum\8. Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment>python app.py
C:\Users\mudhi\OneDrive\Pictures\plant_uwl.plum\8. Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment\app.py:40: FutureWarning: You are using 'torch.load' with 'weights_only=False' (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for more details). In a future release, the default value for 'weights_only' will be flipped to 'True'. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via 'torch.serialization.add_safe_globals'. We recommend you start setting 'weights_only=True' for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.
  att_model.load_state_dict(torch.load("models\att_cnn_bilstm.pth", map_location=device))
```

# IMPLEMENTATION

**Title:** System Running Successfully

- Flask server runs on **http://127.0.0.1:5000**.
- The web interface becomes accessible through a browser.
- Users can input IoT traffic data and receive botnet detection results.
- This confirms successful integration of hybrid ML models with the web application.

```
(bot) C:\Users\mudhi\OneDrive\Pictures\plant_uml.plum\8. Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment>python app.py
C:\Users\mudhi\OneDrive\Pictures\plant_uml.plum\8. Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment\app.py:44: FutureWarning: You are using 'torch.load' with 'weights_only=False' (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for more details). In a future release, the default value for 'weights_only' will be flipped to 'True'. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via 'torch.serialization.add_safe_globals'. We recommend you start setting 'weights_only=True' for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.
  att_model.load_state_dict(torch.load("models/att_cnn_bilstm.pth", map_location=device))
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

# IMPLEMENTATION

**Title:** Project Home Page – Bot-Attack Dashboard

**Description:**

This slide shows the **home page of the Bot-Attack web application**.

The dashboard introduces the project titled **“Hybrid Machine Learning Model for Efficient Botnet Attack Detection in IoT Environment.”**

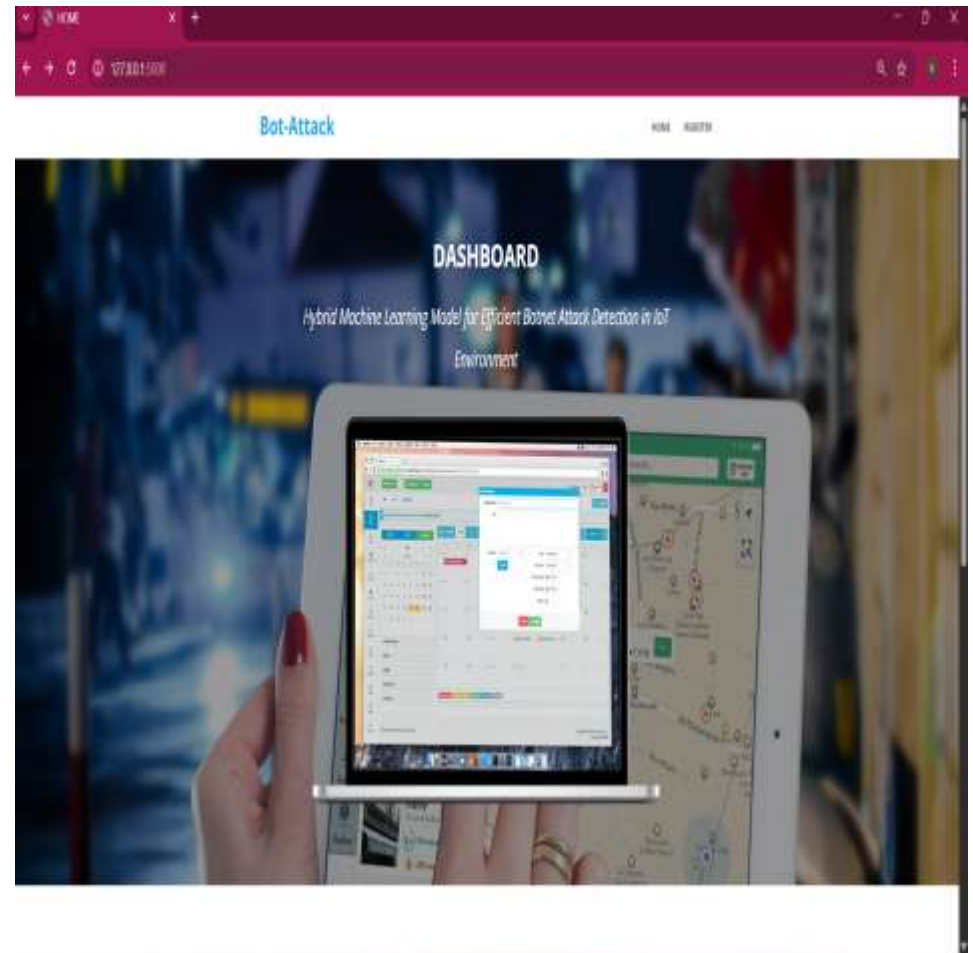
It provides a **user-friendly interface** for accessing the system.

Navigation options such as **Home** and **Register** are available for users.

The dashboard acts as the **entry point** to the botnet detection system.

**Purpose:**

To present an overview of the system and allow users to start interaction with the application.



# IMPLEMENTATION

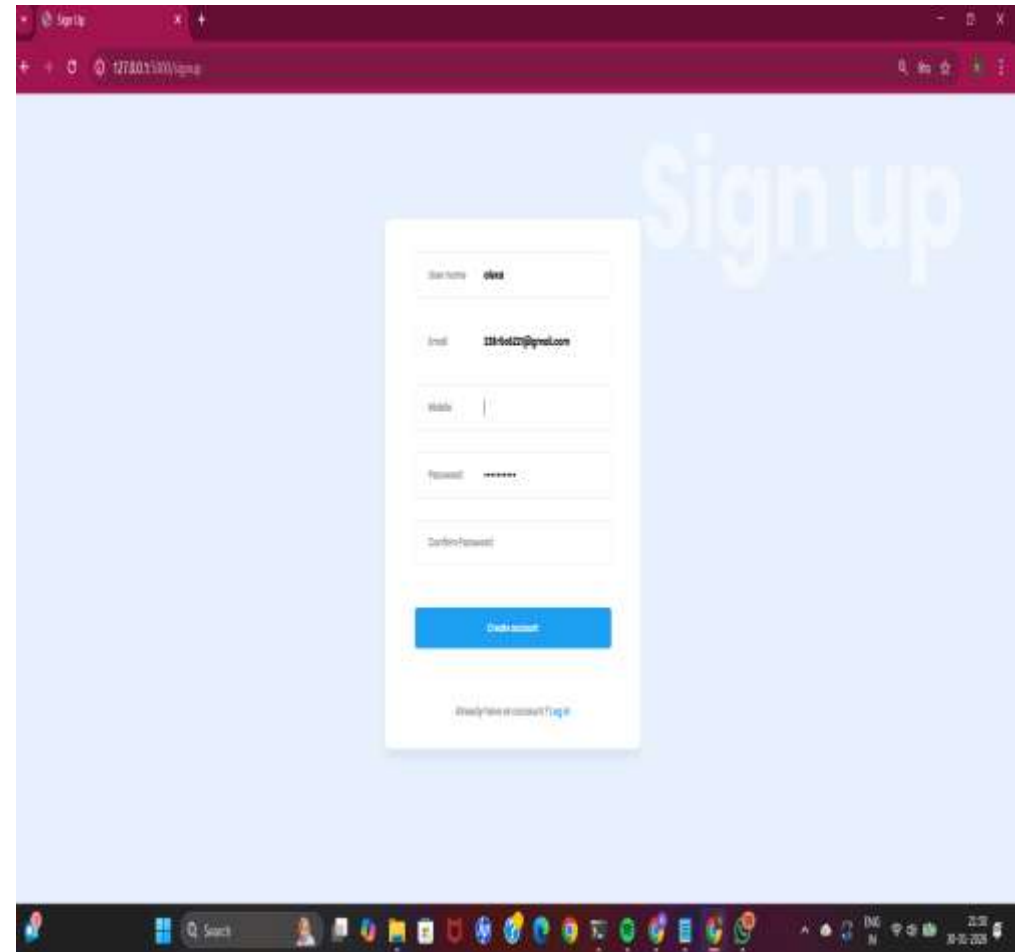
**Title:** User Registration Module

**Description:**

- This slide displays the **Sign-Up** page of the application.
- Users can create an account by entering:
  - Username
  - Email ID
  - Mobile Number
  - Password and Confirm Password
- The system validates user input before account creation.
- An option is provided for existing users to **log in directly**.

**Purpose:**

- To ensure **secure and authenticated access** to the botnet detection system.



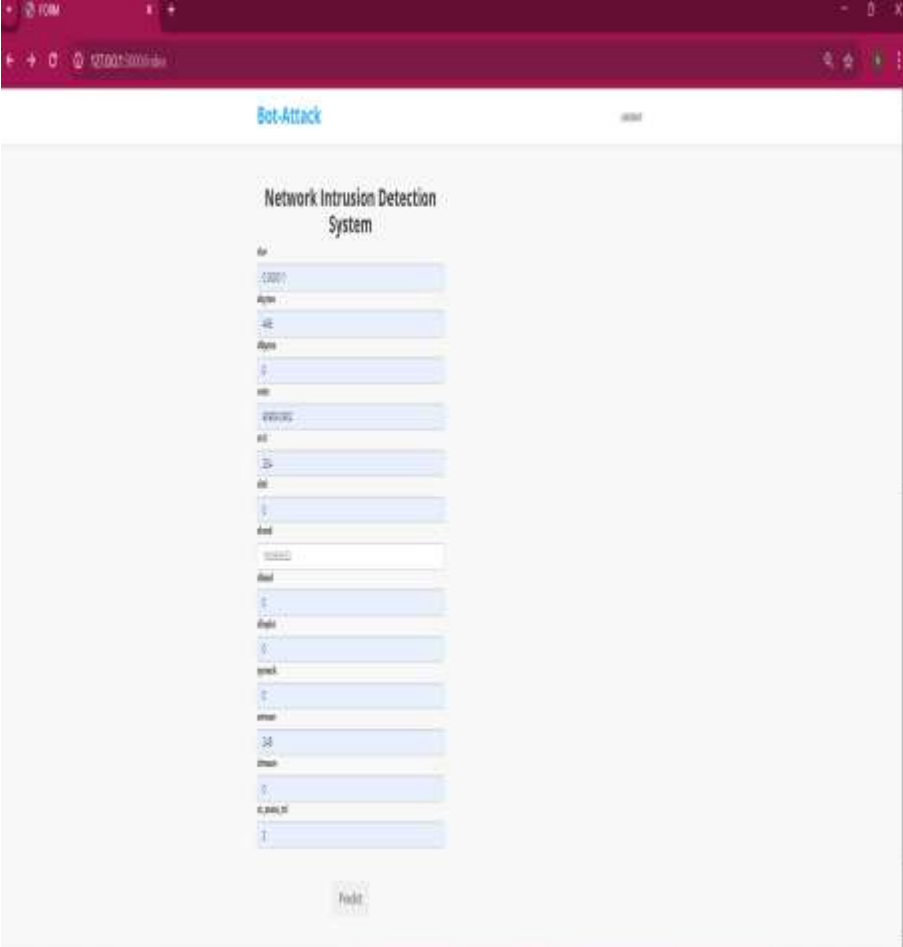


# IMPLEMENTATION

**Title:** Network Intrusion Detection System – Input Module

**Description:**

- This slide shows the **network traffic feature input form**.
  - Users manually enter IoT network parameters such as:
    - Duration (dur)
    - Source bytes (sbytes)
    - Destination bytes (dbytes)
    - Rate, TTL values
    - Load and packet-related features
  - These features represent **real-time IoT network traffic behavior**.
- Purpose:**
- To collect network traffic data for **botnet attack prediction** using machine learning models.



The screenshot displays a web browser window with a red title bar. The page title is 'Bot-Attack'. The main content area is titled 'Network Intrusion Detection System'. It contains a vertical list of input fields for network parameters. The fields are labeled as follows: 'dur' (Duration), 'sbytes' (Source bytes), 'dbytes' (Destination bytes), 'rate' (Rate), 'ttl' (TTL), 'load' (Load), 'packet' (Packet), and 'attack' (Attack). Each field has a corresponding input box with a blue border. At the bottom of the form, there is a 'Predict' button.

# IMPLEMENTATION

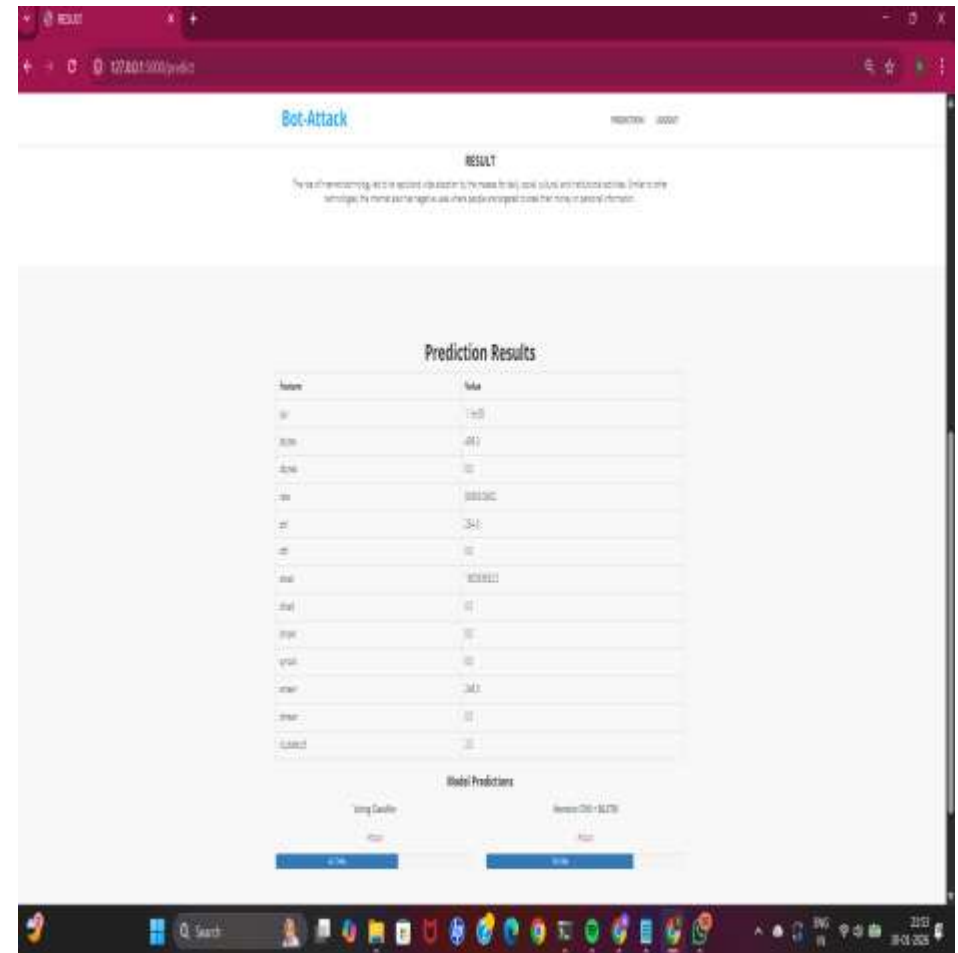
**Title:** Botnet Attack Detection – Prediction Results

**Description:**

- This slide displays the **prediction results** after submitting network parameters.
- The system uses a **Hybrid Machine Learning Approach** for detection.
- Two models are applied:
  - **Voting Classifier**
  - **Attention-based CNN + BiLSTM**
- Both models classify the traffic as **Attack** with confidence percentages.

**Purpose:**

- To clearly identify whether the given IoT traffic is **Normal or Malicious (Botnet Attack)**.
- To improve detection accuracy using **multiple ML models**.



# Conclusion

The proposed Hybrid Ensemble Learning Approach (ACLR) provides an efficient and intelligent solution for botnet attack detection in IoT networks. By integrating CatBoost, TCN, Transformer, and LightGBM models through a CTTL stacking framework, the system effectively captures complex spatial and temporal features of network traffic. Using the UNSW-NB15 dataset, the model achieved high accuracy, precision, and robustness, outperforming existing methods. The results demonstrate that the ACLR model is not only accurate but also scalable and adaptable for real-world IoT environments. This approach contributes significantly to enhancing cybersecurity by enabling early, reliable, and automated detection of evolving botnet threats.



# References

- [1] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, “Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset,” *Future Gener. Comput. Syst.*, vol. 100, pp. 779–796, Nov. 2019.
- [2] O. Ibitoye, O. Shafiq, and A. Matrawy, “Analyzing adversarial attacks against deep learning for intrusion detection in IoT networks,” in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.
- [3] M. Shahhosseini, H. Mashayekhi, and M. Rezvani, “A deep learning approach for botnet detection using raw network traffic data,” *J. Netw. Syst. Manage.*, vol. 30, no. 3, p. 44, Jul. 2022.
- [4] S. Homayoun, M. Ahmadzadeh, S. Hashemi, A. Dehghantanha, and R. Khayami, “BoTShark: A deep learning approach for botnet traffic detection,” in *Cyber Threat Intelligence*, 2018, pp. 137–153.
- [5] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, and A. Robles-Kelly, “Deep learning-based intrusion detection for IoT networks,” in *Proc. IEEE 24th Pacific Rim Int. Symp. Dependable Comput. (PRDC)*, Dec. 2019, p. 256.