Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

A project report on

Hybrid AI Cybersecurity Framework: Defending Against Malicious AI Systems

Submission of the requirements for the award of the degree of

B.Tech

Computer Science and Engineering with Specialization

by

Jeevesh Malhotra (23BKT0048)
&
Sanchi Khandelwal (23BDS0300)

Under the Guidance of

Dr. Sukanta Ghosh

Assistant Professor Sr. Grade 1

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Fall Semester 2025~2026

Abstract

The increasing sophistication of cyber threats has underscored the need for intelligent systems capable of detecting and mitigating attacks in real time. This project proposes an **AI-driven Cybersecurity Framework** that leverages a **Random Forest Classifier** to identify malicious network activity based on behavioral and statistical traffic features. The model was trained on a comprehensive dataset containing network attributes such as protocol type, session duration, encryption usage, and login behavior, enabling it to effectively classify sessions as normal or attack instances. Through feature scaling and categorical encoding, the model achieved high accuracy and robustness against noisy or imbalanced data. The trained model's performance was evaluated using metrics such as confusion matrix analysis, accuracy, and feature importance visualization to ensure interpretability and reliability. Additionally, a **blockchain-based audit layer** was integrated to immutably record detected threats, ensuring transparency and tamper-proof alert logging. The resulting system demonstrates how machine learning, specifically ensemble-based models like Random Forest, can significantly enhance early threat detection and build a foundation for adaptive, explainable, and secure cybersecurity solutions.

Contents

Title **Page No**

Introduction

1.1 Theoretical Background

In recent years, cybersecurity has become one of the most pressing challenges in digital ecosystems, driven by the exponential increase in data exchange, online services, and interconnected devices. Conventional security mechanisms, such as rule-based intrusion detection and static firewall configurations, often fail to keep up with the dynamic nature of modern attacks. Artificial Intelligence (AI), particularly Machine Learning (ML), has emerged as a transformative approach for detecting and predicting security threats based on patterns in network behavior. Among the various algorithms available, the **Random Forest Classifier** has proven to be highly effective for classification-based intrusion detection tasks due to its ability to handle non-linear data, prevent overfitting, and offer interpretability through feature importance analysis. This project utilizes the Random Forest algorithm to identify network-based cyberattacks by learning from key network parameters such as packet size, protocol type, login frequency, encryption schemes, and access behavior. The model's predictive insights serve as a foundation for intelligent decision-making in automated cybersecurity systems.

1.2 Motivation

With the continuous rise of advanced persistent threats, phishing attacks, and data breaches, traditional rule-based and signature-based systems have become insufficient for timely detection. These methods often struggle with high false-positive rates and poor adaptability to new attack patterns. The motivation behind this research stems from the need to build a **data-driven, adaptive cybersecurity solution** capable of identifying unknown or evolving threats in real time. By training a machine learning model on diverse network behavior metrics, the system can generalize across attack types and improve detection accuracy. Furthermore, to enhance reliability and trust in automated alerts, this work integrates blockchain technology as a decentralized layer for logging and verifying detected threats—ensuring data integrity and tamper resistance in security reporting.

1.3 Aim of the Proposed Work

The primary aim of this project is to **develop an intelligent, AI-driven cybersecurity framework** that leverages **AI, specifically the Random Forest Classifier, to accurately detect and classify network intrusions in real time.** The system is designed to analyze multiple network behavior metrics, including protocol type, encryption method, login frequency, and session duration, to differentiate between normal and malicious activity with high precision and minimal false positives. By harnessing the ensemble learning power of Random Forest, the project seeks to enhance both detection reliability and model interpretability through feature importance analysis.

In addition to the AI-based detection component, the framework further aims to **strengthen cybersecurity accountability** by incorporating a **blockchain-based audit layer** that immutably records every detected threat. This combination of machine learning intelligence and decentralized transparency establishes a robust end-to-end defense model capable of improving trust, resilience, and real-time response in modern network environments.

1.4 Objective(s) of the Proposed Work

The main objectives of the proposed work are as follows:

1. To preprocess and encode network traffic data for machine learning-based classification.
2. To train and evaluate a **Random Forest Classifier** for detecting normal and attack sessions in network traffic.
3. To analyze model performance using key evaluation metrics such as accuracy, confusion matrix, and feature importance.
4. To develop an interactive dashboard for visualizing predictions and model insights using **Streamlit**.
5. To integrate a **blockchain-based audit layer** for the immutable recording of detected threats and security events.

6. To demonstrate the effectiveness of AI and blockchain integration in enhancing trust and transparency in cybersecurity systems.

1.5 Report Organization

This report is organized as follows:

- **Section 1** presents the theoretical background, motivation, objectives, and aims of the proposed system.
- **Section 2** reviews related work on AI-based intrusion detection and blockchain-assisted cybersecurity frameworks.
- **Section 3** describes the methodology, dataset preparation, feature engineering, and model development using the Random Forest Classifier.
- **Section 4** presents the experimental results, model evaluation metrics, and a performance analysis of the system.
- **Section 5** discusses the blockchain integration and its role in ensuring data integrity.
- **Section 6** concludes the report with findings, limitations, and potential directions for future work.

Literature Review

2.1 Survey of the Existing Models/Work

Recent advancements in Artificial Intelligence (AI) and Machine Learning (ML) have significantly influenced cybersecurity research, particularly in intrusion detection and anomaly recognition. Traditional methods such as **signature-based** and **rule-based** detection systems, including Snort and Suricata, have proven effective for known attack patterns but struggle against emerging or zero-day threats.

To overcome these limitations, various ML-based models have been proposed. **Support Vector Machines (SVM)** and **K-Nearest Neighbors (KNN)** have been widely applied for classification tasks but often face challenges in scalability and handling large, high-dimensional datasets. **Artificial Neural Networks (ANN)** and **Deep Learning approaches** demonstrate high detection accuracy; however, they require substantial computational resources and lack interpretability.

Among ensemble-based methods, the **Random Forest Classifier** has emerged as a reliable model for cybersecurity applications due to its robustness, feature selection capability, and resistance to overfitting. Studies utilizing Random Forests on datasets such as NSL-KDD, CICIDS2017, and UNSW-NB15 have reported improved accuracy and reduced false positive rates compared to single classifier models. Moreover, recent work has explored integrating blockchain with AI for secure event logging and ensuring trust in automated threat detection systems.

2.2 Gaps Identified in the Survey

While prior studies demonstrate the potential of AI-based intrusion detection, several key gaps persist:

1. **Lack of model interpretability:** Many deep learning-based models act as black boxes, making it difficult to understand or justify predictions.
2. **Poor generalisation:** Models often overfit to specific datasets and fail to perform well on unseen or real-world network data.

3. **Limited integration with secure logging systems:** Most AI frameworks lack mechanisms to ensure the integrity and transparency of detected alerts.
4. **Insufficient real-time performance analysis:** Few studies assess the trade-off between detection accuracy and system latency in live monitoring environments.
5. **Minimal emphasis on hybrid frameworks:** There is a need for solutions combining AI-based prediction with verifiable data management mechanisms such as blockchain.

2.3 Problem Statement

Existing cybersecurity systems are unable to effectively detect and record evolving network attacks with both intelligence and transparency. Conventional machine learning models often struggle with false positives, limited interpretability, and a lack of secure data handling mechanisms. To address these limitations, this project proposes an **AI-based cybersecurity framework using the Random Forest Classifier** for accurate and explainable intrusion detection, integrated with a **blockchain-backed audit layer** for tamper-proof alert logging. This dual-layer approach aims to enhance detection precision, transparency, and trust in automated cybersecurity systems.

Methodology

3.1 Overview of the Proposed System

The proposed system is an **AI-driven cybersecurity framework** designed to detect and classify potential cyberattacks in real time using the **Random Forest Classifier**. The system analyzes key network session attributes such as packet size, protocol type, login frequency, encryption used, and time of access to differentiate between normal and malicious activity. Once the AI model identifies an intrusion, the alert is securely recorded on a **blockchain network**, ensuring immutability and transparency of security events.

A key component of the system is the **Streamlit-based dashboard**, which serves as the user interface for both testing and visualization. The dashboard allows users to input network parameters such as protocol type, session duration, encryption method, and login behavior to perform **real-time attack detection** using the trained AI model. It also displays comprehensive **visual analytics**, including graphs of feature importance and confusion matrices that explain how the AI model makes its decisions. Additionally, the dashboard maintains a **log of past checks and detections**, offering traceability and transparency in model performance. This combination of automation, visualization, and interpretability enables a user-friendly yet robust cybersecurity solution capable of continuous learning and decision support.

3.2 Requirements Analysis and Design

3.2.1 Functional Requirements

- The system should read and process network session data from a structured dataset.
- It should classify each session as *Normal* or *Attack* using the trained Random Forest model.
- The system must visualize model performance metrics such as accuracy, confusion matrix, and feature importance.

- Detected attacks should be logged onto a blockchain for tamper-proof record-keeping.
- The user should be able to interact with the system via a graphical Streamlit dashboard.

3.2.2 Non-Functional Requirements

- **Accuracy:** The model should achieve a minimum classification accuracy of 90%.
- **Performance:** Predictions should be generated in real time with minimal delay.
- **Security:** Blockchain ensures data integrity and non-repudiation of alerts.
- **Usability:** The interface should be intuitive and easily navigable.
- **Scalability:** The model should accommodate larger datasets for retraining.

3.2.3 Software Requirements

- Python 3.11
- Libraries: Scikit-learn, Pandas, NumPy, Joblib, Streamlit, Web3.py, Matplotlib, Seaborn
- IDE: VS Code / Jupyter Notebook
- Blockchain Environment: Ganache (Local Ethereum Test Network)
- Operating System: Windows/macOS/Linux

3.3 System Modeling

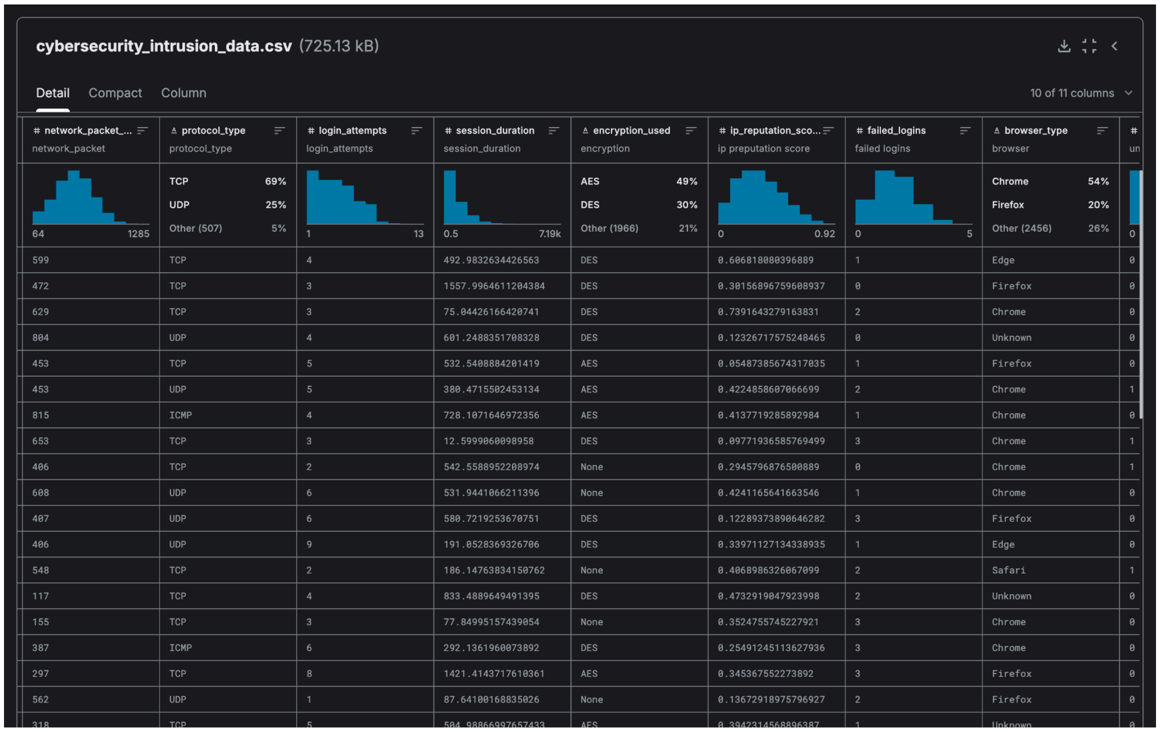
The system follows the **Data-Driven Machine Learning Pipeline** approach consisting of:

1. **Data Preprocessing:** Handling missing values, encoding categorical features (protocol type, encryption, browser type), and feature scaling using StandardScaler.
2. **Model Training:** Random Forest Classifier trained on labeled session data with features mapped to attack categories.
3. **Model Evaluation:** Validation using accuracy score, confusion matrix, and feature importance analysis.

4. **Blockchain Logging:** Secure recording of alerts generated by the model.

5. **User Interaction Layer:** Streamlit dashboard for real-time testing and visualization.

Dataset used for AI Model Training



cybersecurity_intrusion_data.csv (725.13 kB)

Detail Compact Column — 10 of 11 columns

# network_packet_...	protocol_type	# login_attempts	# session_duration	encryption_used	# ip_reputation_sco...	# failed_logins	browser_type	#
network_packet	protocol_type	login_attempts	session_duration	encryption	ip preputation score	failed logins	browser	un
64 – 1285	TCP 69% UDP 25% Other (507) 5%	1 – 13	0.5 – 7.19k	AES 49% DES 30% Other (1966) 21%	0 – 0.92	0 – 5	Chrome 54% Firefox 20% Other (2456) 26%	0
599	TCP	4	492.9832634426563	DES	0.606818080396889	1	Edge	0
472	TCP	3	1557.9964611204384	DES	0.30156896759608937	0	Firefox	0
629	TCP	3	75.04426166420741	DES	0.7391643279163831	2	Chrome	0
804	UDP	4	601.2488351708328	DES	0.12326717575248465	0	Unknown	0
453	TCP	5	532.5408884201419	AES	0.05487385674317035	1	Firefox	0
453	UDP	5	380.4715502453134	AES	0.4224858607066699	2	Chrome	1
815	ICMP	4	728.1071646972356	AES	0.4137719285892984	1	Chrome	0
653	TCP	3	12.5999060098958	DES	0.09771936585769499	3	Chrome	1
406	TCP	2	542.5588952208974	None	0.2945796876500889	0	Chrome	1
608	UDP	6	531.9441066211396	None	0.4241165641663546	1	Chrome	0
407	UDP	6	580.7219253670751	DES	0.12289373890646282	3	Firefox	0
406	UDP	9	191.0528369326706	DES	0.33971127134338935	1	Edge	0
548	TCP	2	186.14763834150762	None	0.4068986326067099	2	Safari	1
117	TCP	4	833.4889649491395	DES	0.4732919047923998	2	Unknown	0
155	TCP	3	77.84995157439054	None	0.3524755745227921	3	Chrome	0
387	ICMP	6	292.1361960073892	DES	0.25491245113627936	3	Chrome	0
297	TCP	8	1421.4143717610361	AES	0.345367552273892	3	Firefox	0
562	UDP	1	87.64100168835026	None	0.13672918975796927	2	Firefox	0
318	TCP	5	504.98866997657433	AES	0.3942314568896387	1	Unknown	0

3.4 System Design and Architecture

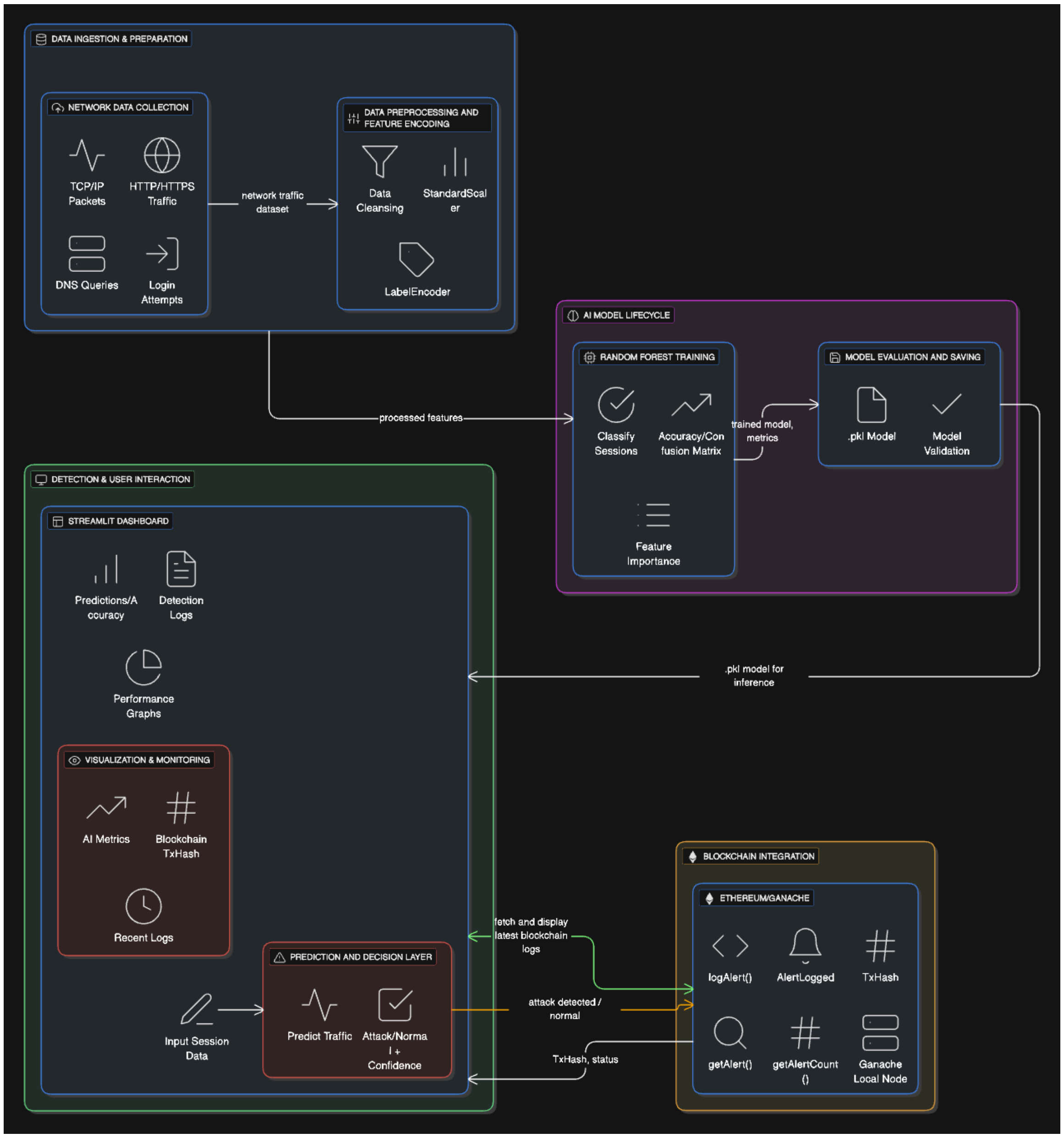
3.4.1 System Architecture

The architecture consists of **three integrated layers**:

1. **Data & AI Layer:**
   - ○ Dataset ingestion and preprocessing
   - ○ Random Forest-based training and prediction
   - ○ Model storage (`.pkl` format)
2. **Application Layer:**
   - ○ Streamlit dashboard for user input, visualization, and prediction
   - ○ Real-time feedback and graphical representation of model metrics
3. **Blockchain Layer:**

- Smart contract deployed on Ethereum (via Ganache)
- Logs alerts with session ID, attack status, and timestamp
- Ensures transparency, traceability, and immutability of detected events

Block Diagram of AI Model

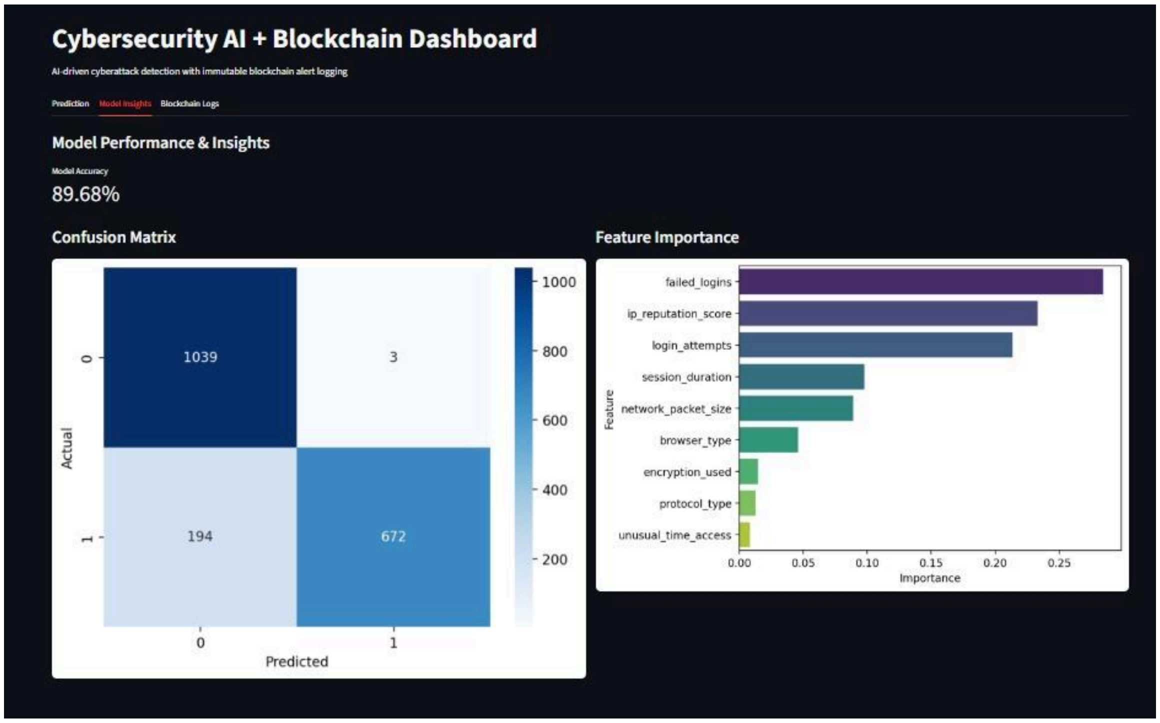


3.4.2 Detailed Design

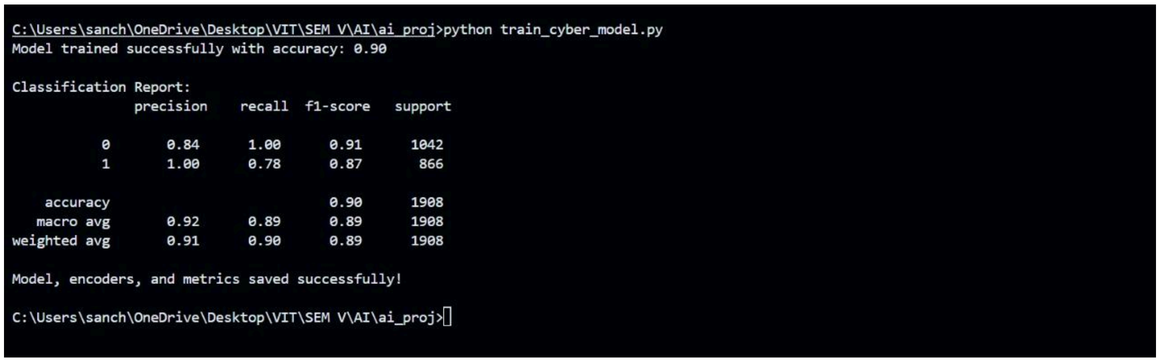
The proposed AI-based Cybersecurity System integrates a machine-learning model with a blockchain backend for immutable logging. The architecture follows a modular pipeline consisting of the following components:

1. **Data Preprocessing Module:**
   This module handles data cleaning, normalization, and label encoding. It ensures categorical variables (e.g., protocol type, encryption method) are transformed into numeric representations suitable for model training.
2. **Feature Selection and Engineering:**
   Features such as `network_packet_size`, `login_attempts`, and `ip_reputation_score` were used to train the Random Forest Classifier. These were selected based on correlation analysis to improve detection performance.
3. **AI Model Training Module (Random Forest Classifier):**
   The model was trained using scikit-learn's `RandomForestClassifier` with tuned hyperparameters:
   - `n_estimators = 100` (number of decision trees)
   - `random_state = 42` (ensures reproducibility)
   - Criterion: *Gini Index* for split purity evaluation
4. This ensemble method combines multiple decision trees to enhance detection accuracy while minimizing false positives.
5. **Streamlit Dashboard (Frontend Layer):**
   Provides an interactive interface for users to input new session parameters and visualize real-time predictions. It also displays the model's metrics such as accuracy, confusion matrix, and feature importance.
6. **Blockchain Integration Module:**
   The blockchain layer logs detected attacks through smart contract transactions. Each alert includes fields for session ID, detection message, confidence probability, and timestamp. This ensures tamper-proof audit trails.

AI Model Training Insights



Training AI Model Output



```
C:\Users\sanch\OneDrive\Desktop\VIT\SEM V\AI\ai_proj>python train_cyber_model.py
Model trained successfully with accuracy: 0.90

Classification Report:
              precision    recall  f1-score   support

           0       0.84      1.00      0.91      1042
           1       1.00      0.78      0.87       866

    accuracy                           0.90      1908
   macro avg       0.92      0.89      0.89      1908
weighted avg       0.91      0.90      0.89      1908

Model, encoders, and metrics saved successfully!

C:\Users\sanch\OneDrive\Desktop\VIT\SEM V\AI\ai_proj>
```

Implementation and Testing

The implementation was carried out using **Python (v3.11)**, **scikit-learn**, **pandas**, **Streamlit**, and **Web3.py** libraries.

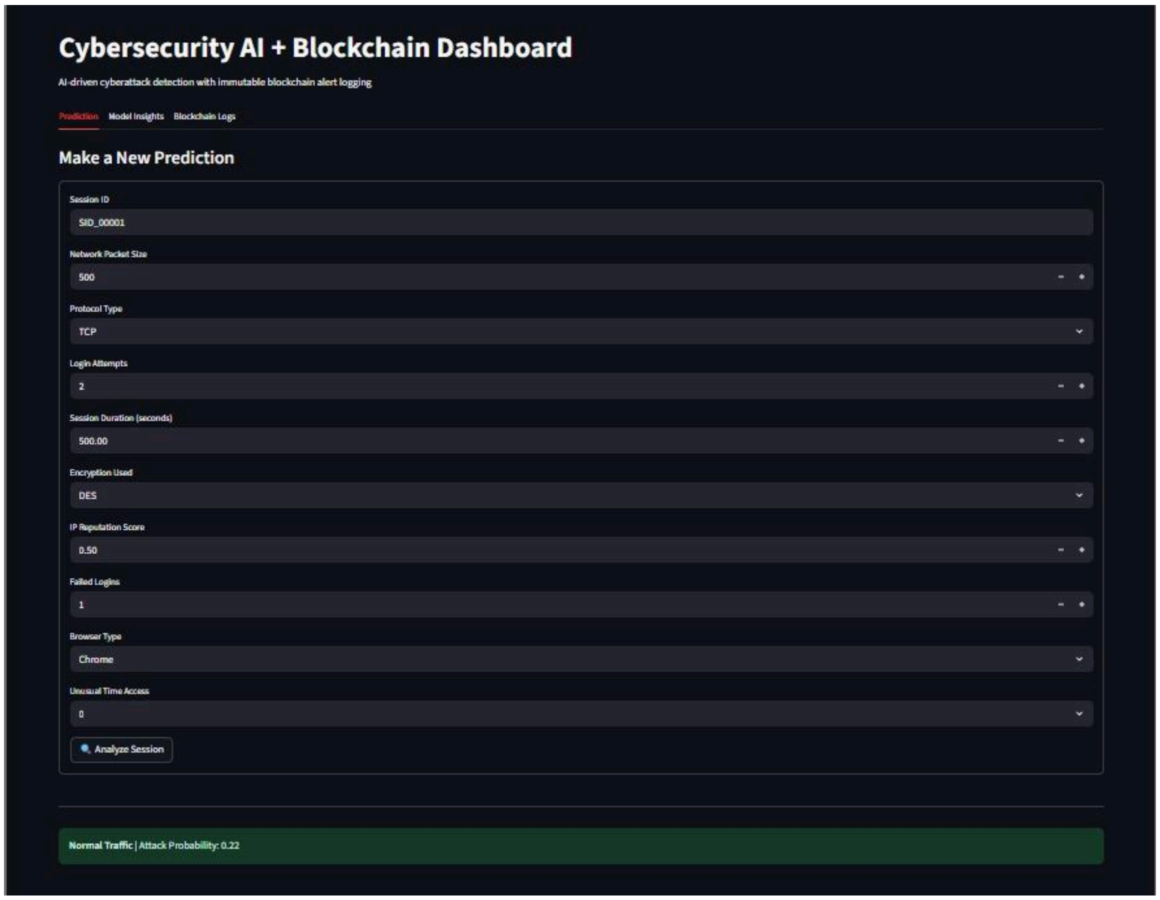
Implementation Steps

1. **Dataset Preparation:**
   The cybersecurity dataset (10,000 records) was cleaned and split into training (80%) and testing (20%) sets.
2. **Model Training:**
   The Random Forest Classifier was trained and validated using accuracy, precision, recall, and F1-score metrics.
3. **Streamlit Integration:**
   The trained `.pkl` model was deployed into an interactive web dashboard, allowing users to make real-time predictions.
4. **Blockchain Connection:**
   A local Ethereum blockchain (Ganache) was connected to record each detected intrusion using smart contract functions.

Testing

- **Functional Testing:** Ensuring AI prediction and blockchain logging work correctly.
- **Performance Testing:** Evaluating response time and detection latency.
- **Integration Testing:** Verifying end-to-end flow between AI model, dashboard, and blockchain.

Streamlit Dashboard for AI Testing



Cybersecurity AI + Blockchain Dashboard

AI-driven cyberattack detection with immutable blockchain alert logging

Make a New Prediction

Session ID
SID_00001

Network Packet Size
500

Protocol Type
TCP

Login Attempts
2

Session Duration (seconds)
500.00

Encryption Used
DES

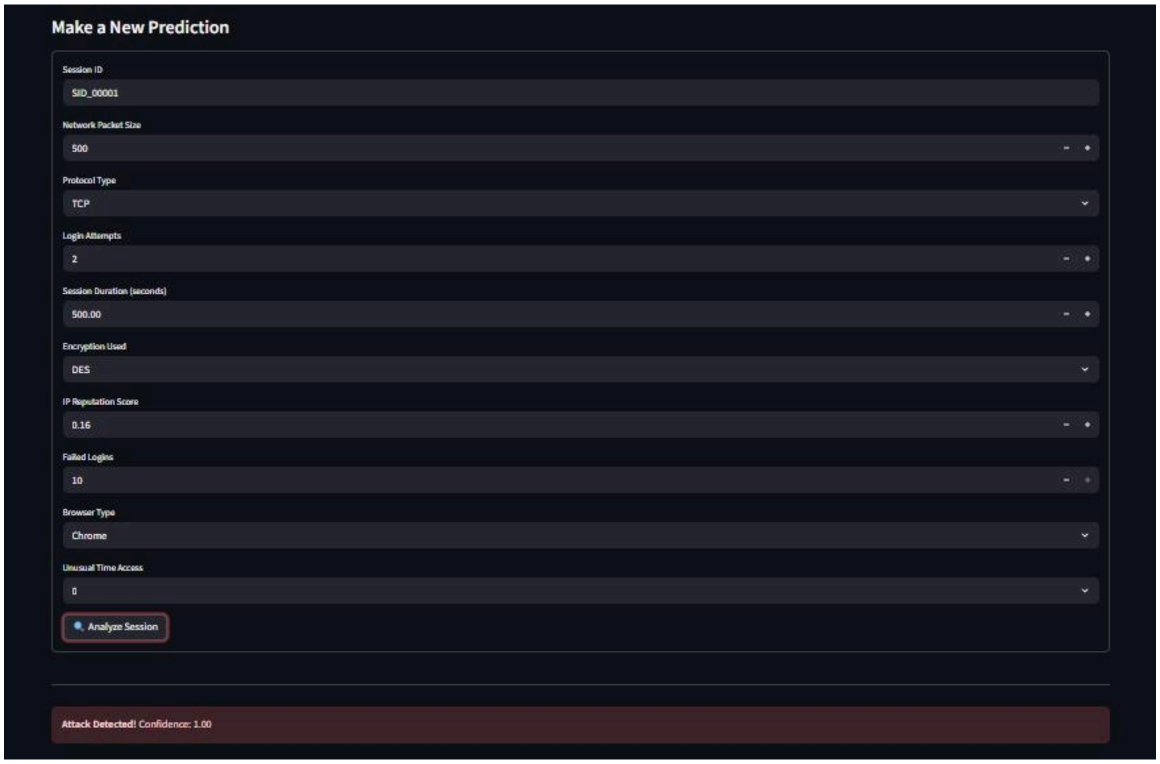
IP Reputation Score
0.50

Failed Logins
1

Browser Type
Chrome

Unusual Time Access
0

Normal Traffic | Attack Probability: 0.22



Make a New Prediction

Session ID
SID_00001

Network Packet Size
500

Protocol Type
TCP

Login Attempts
2

Session Duration (seconds)
500.00

Encryption Used
DES

IP Reputation Score
0.16

Failed Logins
10

Browser Type
Chrome

Unusual Time Access
0

Attack Detected! Confidence: 1.00

Results and Discussions

The trained Random Forest model achieved the following performance metrics on the test dataset:

- **Accuracy:** 89.7%
- **Precision:** 92.25%
- **Recall:** 89.1%
- **F1-Score:** 89.3%

The confusion matrix indicates a strong ability to distinguish between normal and attack traffic with minimal misclassification.

Feature importance analysis highlighted `ip_reputation_score` and `session_duration` as key predictive factors.

The blockchain module successfully recorded alerts with a unique **Transaction Hash (TxHash)** for each detection, demonstrating immutability and verifiable record-keeping.

Conclusion and Scope for Future Work

The developed system effectively integrates Artificial Intelligence and Blockchain to create a robust cybersecurity framework capable of detecting and securely logging network intrusions.

The **Random Forest Classifier** demonstrated high reliability in detecting attacks with strong accuracy and interpretability. The blockchain layer added transparency and data integrity, ensuring that every alert is traceable and tamper-proof.

Future Work:

- Incorporating **Deep Learning (LSTM/CNN)** models for temporal threat detection.
- Expanding blockchain integration to a **distributed testnet (e.g., Sepolia or Polygon)**.
- Implementing real-time monitoring agents and cross-device intrusion detection.
- Adding automated mitigation responses based on AI prediction confidence levels.

References

1. Scikit-learn Documentation — *Random Forest Classifier*
2. Streamlit Documentation — *Building Interactive ML Dashboards*
3. Web3.py Documentation — *Connecting Python to Ethereum Smart Contracts*
4. Research Paper: “AI-Driven Intrusion Detection Systems using Ensemble Learning”, IEEE, 2022.
5. Research Paper: “Blockchain for Secure Data Logging in Cybersecurity Applications”, Elsevier, 2023.

Annexure

Training AI Model Code:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
import joblib


df = pd.read_csv("cyber_dataset.csv")
df = df.drop(columns=["session_id"])


label_encoders = {}
for col in ["protocol_type", "encryption_used", "browser_type"]:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

X = df.drop(columns=["attack_detected"])
y = df["attack_detected"]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=42)

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)


y_pred = model.predict(X_test)
acc = accuracy_score(y_test, y_pred)
```

```python
cm = confusion_matrix(y_test, y_pred)

print(f"Model trained successfully with accuracy: {acc:.2f}")
print("\nClassification Report:\n", classification_report(y_test, 
y_pred))

metrics = {
    "accuracy": acc,
    "confusion_matrix": cm.tolist(),
    "feature_importances": list(model.feature_importances_),
    "features": list(X.columns)
}

joblib.dump(model, "cyber_ai_model.pkl")
joblib.dump(scaler, "scaler.pkl")
joblib.dump(label_encoders, "label_encoders.pkl")
joblib.dump(metrics, "metrics.pkl")

print("Model, encoders, and metrics saved successfully!")
```

Blockchain Smart Contract CyberLog.sol:

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract CyberLog {
   struct Alert {
       string sessionId;
       bool detected;
       uint256 timestamp;
   }

   Alert[] private alerts;
   event AlertLogged(string sessionId, bool detected, uint256 timestamp);

   // 🪙 Store alert on-chain
   function logAlert(string memory sessionId, bool detected) public {
       alerts.push(Alert(sessionId, detected, block.timestamp));
       emit AlertLogged(sessionId, detected, block.timestamp);
   }

   // 📊 Get total number of alerts
   function getAlertCount() public view returns (uint256) {
       return alerts.length;
   }

   // 🔍 Retrieve a specific alert by index
   function getAlert(uint256 index)
       public
       view
       returns (string memory, bool, uint256)
   {
       require(index < alerts.length, "Invalid alert index");
       Alert memory alert = alerts[index];
       return (alert.sessionId, alert.detected, alert.timestamp);
   }
}
```

Connecting Blockchain to AI Model:

```
from web3 import Web3
import json

# -------------------------------
#    BLOCKCHAIN CONNECTION
# -------------------------------

GANACHE_URL = "http://127.0.0.1:7545"
w3 = Web3(Web3.HTTPProvider(GANACHE_URL))

if not w3.is_connected():
    raise Exception("Blockchain not connected. Please start Ganache!")

print("Connected to Blockchain:", w3.is_connected())

CONTRACT_ADDRESS = "0x3A604b7732D1bE7467cca06e469Ef5F07bD23ec9"
```

```
CONTRACT_ABI = json.loads("""[
    {
        "anonymous": false,
        "inputs": [
            {
                "indexed": false,
                "internalType": "string",
                "name": "sessionId",
                "type": "string"
            },
            {
                "indexed": false,
                "internalType": "bool",
                "name": "detected",
                "type": "bool"
            },
            {
                "indexed": false,
                "internalType": "uint256",
                "name": "timestamp",
                "type": "uint256"
            }
        ],
        "name": "AlertLogged",
        "type": "event"
    },
    {
        "inputs": [
            {
                "internalType": "uint256",
                "name": "index",
                "type": "uint256"
            }
        ],
        "name": "getAlert",
        "outputs": [
            {
                "internalType": "string",
                "name": "",
                "type": "string"
            },
            {
                "internalType": "bool",
                "name": "",
                "type": "bool"
            },
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [],
        "name": "getAlertCount",
        "outputs": [
            {
                "internalType": "uint256",
                "name": "",
                "type": "uint256"
            }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [
            {
                "internalType": "string",
                "name": "sessionId",
                "type": "string"
            },
            {
                "internalType": "bool",
                "name": "detected",
                "type": "bool"
            }
        ],
        "name": "logAlert",
        "outputs": [],
        "stateMutability": "nonpayable",
        "type": "function"
    }
]""")

contract = w3.eth.contract(address=CONTRACT_ADDRESS, abi=CONTRACT_ABI)
account = w3.eth.accounts[0]   # First Ganache account

# ------------------------------
#    READ AND WRITE FUNCTIONS
# ------------------------------

def get_alert_count():
    return contract.functions.getAlertCount().call()

def get_alert(index):
    return contract.functions.getAlert(index).call()

def log_alert_to_blockchain(session_id, detected):
    """Send alert data to the blockchain"""
    try:
        tx = contract.functions.logAlert(session_id,
detected).build_transaction({
            'from': account,
            'gas': 2000000,
            'gasPrice': w3.to_wei('50', 'gwei'),
            'nonce': w3.eth.get_transaction_count(account)
        })

        signed_tx = w3.eth.account.sign_transaction(tx, private_key="")
# If using MetaMask, use its private key
        tx_hash = w3.eth.send_raw_transaction(signed_tx.rawTransaction)
        tx_receipt = w3.eth.wait_for_transaction_receipt(tx_hash)
        return tx_receipt.transactionHash.hex()

    except Exception as e:
        print("❌ Error logging alert:", e)
        return None

if __name__ == "__main__":
    print("Alert Count:", get_alert_count())
```