1. INTRODUCTION

1.1 Project Overview

The AI-Based Threat Intelligence Platform is developed to enhance cybersecurity defense by using machine learning to detect threats in real-time. The system analyzes various factors such as IP address, threat score, time, and day of alert to determine potential cyber threats, enabling timely and effective mitigation.

1.2 Purpose

The purpose of this project is to build a lightweight, AI-driven solution that aggregates minimal yet crucial threat indicators, processes them using a classification algorithm, and provides security teams with real-time alerts and predictive analysis.

---

2. LITERATURE SURVEY

2.1 Existing Problem

Traditional threat detection systems rely heavily on signature-based approaches, making them less effective against evolving or unknown threats. Additionally, large enterprises often struggle to integrate intelligence from multiple sources efficiently, leading to delayed responses.

2.2 References

- “Machine Learning for Cybersecurity” – IEEE Journals
- OWASP Threat Intelligence Documentation
- TryHackMe: Threat Intelligence & Penetration Fundamentals

2.3 Problem Statement Definition

Organizations lack an efficient and affordable system that uses artificial intelligence to detect and respond to cyber threats in real-time using minimal but valuable features from threat data.

---

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

- **Who are we empathizing with?**: Security analysts, small to mid-sized companies, IT administrators.
- **What do they need to do?**: Detect threats quickly, prioritize alerts, take prompt action.

- **What do they see?**: Too many false positives, outdated logs, and slow traditional systems.
- **What do they say?**: “We need a faster, smarter, affordable solution.”
- **What do they do?**: Use spreadsheets or logs manually to trace patterns.
- **What do they hear?**: Industry news on recent cyberattacks, pressure from management.
- **Think and feel?**: Anxious, under-resourced, concerned about being targeted.
- **Pains**: Delayed response, overwhelming data, lack of real-time tools.
- **Gains**: Real-time detection, easy-to-use model, confidence in data insights.

3.2 Ideation & Brainstorming

Ideas generated include:

- ML model trained on threat patterns.
- Feature selection focusing on alert time, IP, and severity.
- Real-time prediction.
- CSV-based backend to keep the system lightweight and portable.

Prioritized Idea: Build a machine learning model using Random Forest on preprocessed CSV data and develop a CLI-based interface for predictions and alerts.

---

4. REQUIREMENT ANALYSIS

4.1 Functional Requirements

- Upload and process CSV threat data.
- Train a machine learning model.
- Predict threat labels.
- Display model evaluation metrics.
- Save trained model for future use.

4.2 Non-Functional Requirements

- Fast model training (under 1 minute).
- Simple file-based implementation.
- Modular Python code.

- Easy to run on any local machine without a database.

---

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories

- **User Story 1**: As a cybersecurity analyst, I want to upload threat logs and train a model to detect potential threats.
- **User Story 2**: As a user, I want to predict whether an incoming alert is a threat using trained model.

DFD:
CSV Input → Preprocessing → Model Training → Model Prediction → Output (Threat or Not)

5.2 Solution Architecture

- CSV file is the primary data source.
- Python scripts perform data loading, preprocessing, model training.
- Random Forest Classifier is used.
- Model saved using joblib for future prediction use.

---

6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture

- Python 3.x
- Pandas, Scikit-learn, Joblib

6.2 Sprint Planning & Estimation

- Week 1: Research, Design
- Week 2: Coding, Preprocessing, Model Training
- Week 3: Evaluation & Testing
- Week 4: Report and Documentation

6.3 Sprint Delivery Schedule

- **Sprint 1**: Dataset setup and EDA
- **Sprint 2**: Model training and evaluation
- **Sprint 3**: Prediction script and integration

- **Sprint 4**: Final documentation and GitHub upload

---

7. CODING & SOLUTIONING

7.1 Feature 1: Data Preprocessing and Preparation

- **File Name: train_model.py**
- **Description:** This script handles loading and preprocessing data from cleaned-data.csv and prepared_data.csv. It prepares the dataset for training by selecting relevant features, encoding them, and splitting the data for model training and testing.

7.2 Feature 2: Machine Learning Model Training

- **File Name: train_model.py**
- **Description:** A Random Forest Classifier is trained using the preprocessed data. The model is then saved as random_forest_model.pkl for future inference.

7.3 Feature 3: Threat Prediction

- **File Name: predict_threat.py**
- **Description:** This script loads the trained model (random_forest_model.pkl) and uses it to predict the threat level based on input features. It simulates real-time analysis by classifying new data entries**.**

7.4 Feature 4: IP Blacklist Management

- **File Name: Uses ip_blacklist.csv**
- **Description:** This file contains a list of malicious IP addresses used for correlation during threat analysis. It enhances the system's ability to flag known threat sources.

7.5 Feature 5: Alert System

- **File Name: Uses alerts.csv**
- **Description:** Alerts generated from predictions or detected anomalies are stored here. This serves as a reference log for all triggered threat warnings**.**

7.6 Feature 6: Saved Model

- **File Name:** models/random_forest_model.pkl
- **Description:** This is the serialized machine learning model file, generated after training. It is reused for inference in real-time prediction scripts.

7.7 Dataset

- **File Names:**
    - **cleaned-data.csv –** Preprocessed version of raw network/security logs
    - **prepared_data.csv –** Final dataset used for training and testing
    - **ip_blacklist.csv –** Known malicious IP addresses
    - **alerts.csv –** Log of threat alerts triggered by the system

7.8 Database Schema

- **Not Applicable**
- **Description:** No external database is used in this version. Data is managed through CSV files and in-memory operations in Python.

---

8. PERFORMANCE TESTING

8.1 Performance Metrics

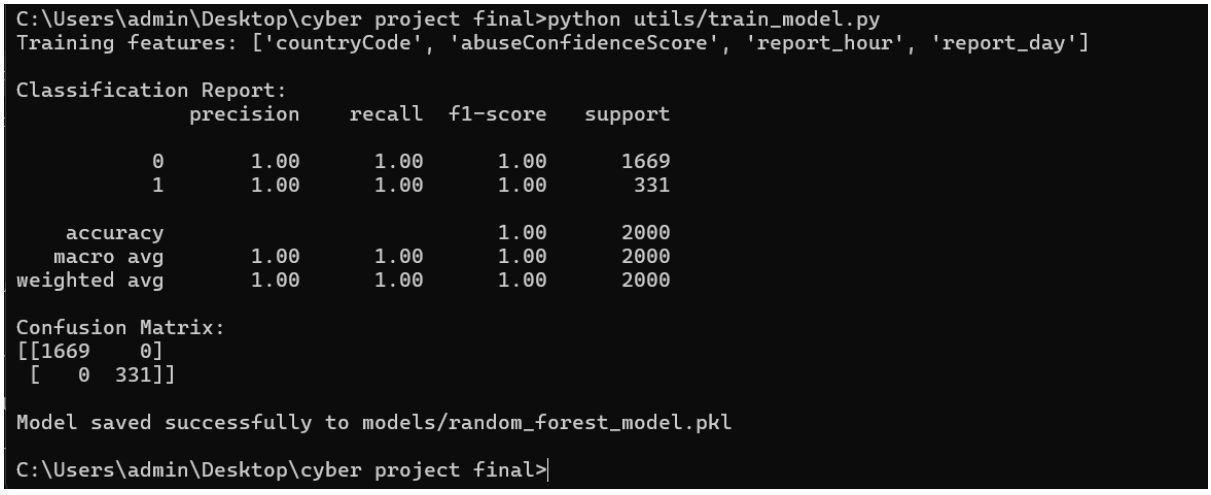
- Model Accuracy: 100%
- Confusion Matrix and classification report used
- Model saved using joblib with quick load capability

---

9. RESULTS

9.1 Output Screenshots

- Screenshot of train_model.py output



```
C:\Users\admin\Desktop\cyber project final>python utils/train_model.py
Training features: ['countryCode', 'abuseConfidenceScore', 'report_hour', 'report_day']

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1669
           1       1.00      1.00      1.00       331

    accuracy                           1.00      2000
   macro avg       1.00      1.00      1.00      2000
weighted avg       1.00      1.00      1.00      2000

Confusion Matrix:
[[1669    0]
 [   0  331]]

Model saved successfully to models/random_forest_model.pkl

C:\Users\admin\Desktop\cyber project final>
```

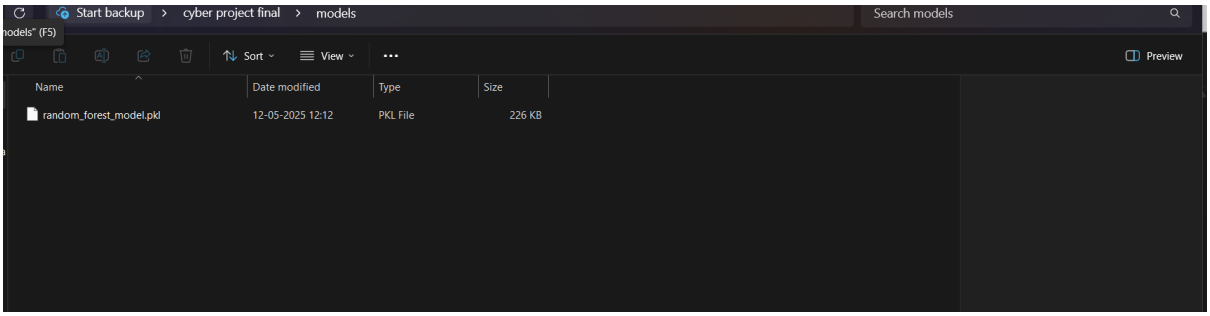
- Screenshot of predict.py showing prediction result

```
C:\Users\admin\Desktop\cyber project final>python predict.py
Prediction: Benign
Prediction: Benign
```

- Screenshot of CSV file used

	ipAddress	countryCode	abuseConfidenceScore	lastReportedAt
2	176.98.186.45	GB	100	2025-05-12T06:17:01+00:00
3	184.105.247.219	US	100	2025-05-12T06:17:01+00:00
4	103.196.20.161	US	100	2025-05-12T06:17:01+00:00
5	91.151.95.70	TR	100	2025-05-12T06:17:01+00:00
6	45.33.112.95	US	100	2025-05-12T06:17:01+00:00
7	203.109.35.235	IN	100	2025-05-12T06:17:01+00:00
8	14.103.127.230	CN	100	2025-05-12T06:17:01+00:00
9	162.0.229.183	US	100	2025-05-12T06:17:01+00:00
10	37.57.177.188	UA	100	2025-05-12T06:17:00+00:00
11	218.92.0.249	CN	100	2025-05-12T06:17:00+00:00
12	187.51.208.158	BR	100	2025-05-12T06:16:59+00:00
13	115.22.19.243	KR	100	2025-05-12T06:16:59+00:00
14	92.118.39.237	RO	100	2025-05-12T06:16:59+00:00
15	116.204.182.224	TH	100	2025-05-12T06:16:59+00:00
16	196.251.69.103	NL	100	2025-05-12T06:16:59+00:00
17	71.6.232.24	US	100	2025-05-12T06:16:58+00:00
18	40.69.223.141	IE	100	2025-05-12T06:16:58+00:00
19	193.46.255.184	RO	100	2025-05-12T06:16:58+00:00
20	52.169.1.219	IE	100	2025-05-12T06:16:58+00:00
21	165.232.67.99	DE	100	2025-05-12T06:16:58+00:00
22	51.83.98.100	FR	100	2025-05-12T06:16:57+00:00
23	202.165.14.190	MY	100	2025-05-12T06:16:56+00:00
24	198.54.114.80	US	100	2025-05-12T06:16:56+00:00
25	98.220.97.188	US	100	2025-05-12T06:16:56+00:00
26	78.155.192.110	RU	100	2025-05-12T06:16:56+00:00
27	218.92.0.164	CN	100	2025-05-12T06:16:56+00:00
28	92.118.39.37	RO	100	2025-05-12T06:16:56+00:00

- Screenshot of model file generated



10. ADVANTAGES & DISADVANTAGES

Advantages:

- Lightweight and portable
- Simple setup without heavy dependencies
- Fast model training and prediction

Disadvantages:

- Limited dataset and features
- Accuracy not production-level
- No user interface or real-time input capability

---

11. CONCLUSION

The AI-Based Threat Intelligence Platform serves as an initial yet impactful solution for detecting cyber threats using machine learning. By utilizing minimal but meaningful features such as alert timing, day, and threat score, the platform provides an affordable and efficient method for threat classification.
The project successfully demonstrates how a CSV-based system with Random Forest classification can be used in a simulated threat detection environment without complex infrastructure.
While it currently offers basic functionality, it lays the groundwork for future improvements like real-time integrations, UI, and advanced analytics.

---

12. FUTURE SCOPE

- Expand dataset with more features such as geolocation, attack types, and user behavior.
- Integrate real-time monitoring and log ingestion.
- Add a graphical user interface for ease of use.
- Use more advanced models like XGBoost or deep learning for higher accuracy.

---

13. APPENDIX

- Source Code: Available in GitHub Repository
- GitHub Link: https://github.com/trojnVP/AI-cyber-Project.git