A

Project Completion Report

On

**Real-Time Threat Prediction System using Machine**

**Learning**

By using

**“AI/ML FOR NETWORKING”**

At

**INTEL UNNATI TRAINING PROGRAM**

AProject Report

Submitted towards the completion of the

Intel Unnati Training Program.

**Submitted by**

Mohana Priya.R

Jahnavi.V.S.A

Inbaa.S

Sangeetha.S

Mowriya.M

Department Of Computer Science Engineering

J.N.N Institute Of Engineering

**Abstract:**

This project is part of the Intel® Unnati AI/ML training program and focuses on developing a real-time threat prediction system using machine learning. The aim is to build a pipeline that can process incoming data, extract relevant features, use a pre-trained model to predict threat types, and display the results through a Flask-based web application. The system ensures rapid and accurate threat detection, making it useful in cybersecurity scenarios.

**Problem Statement:**

## In today’s cybersecurity landscape, detecting threats in real-time is critical. The goal of this project is to develop a machine learning-based solution that can predict threat labels from live input data and display results in a user-friendly web interface.

**Dataset Description:**

* The dataset consists of labeled threat records collected from system logs and intrusion detection systems.
* Each row represents system activity with features like IP address, protocol type, duration, byte counts, etc.
* The target column is a threat label (e.g., normal, DoS, probe, etc.).

**Architecture Flow:**

The pipeline includes CSV input, feature extraction, loading a saved model and encoder, predicting labels, and displaying the results via a Flask frontend.

**Model Design & Features:**

We used a RandomForestClassifier, with selected features based on importance and correlation. The label encoder is used to decode predictions. The system includes a Flask app for real-time prediction.

* **Model Used:** RandomForestClassifier (trained offline)
* **Feature Selection:** Done using correlation and importance-based filtering
* **Label Encoding:** Handled using label\_encoder.pkl
* **Prediction Module:** predict\_new.py
* **Frontend Interface:** Built using Flask (accepts file, shows results)
* **Visualizations:** Accuracy, confusion matrix, and classification report graphs

**Team Contributions:**

|  |  |
| --- | --- |
| **Name** | **Contribution** |
| **Mohana Priya** | Real-time prediction module, Flask UI, model integration |
| **Inbaa** | Data cleaning, preprocessing, feature engineering |
| **Mowriya** | Model training, hyperparameter tuning |
| **Jahnavi** | Evaluation metrics, graph plotting |
| **Sangeetha** | Documentation, PPT |

**Results and Evaluation:**

* **Accuracy:** 93.7% on test data
* **Visualization Outputs:**
  + Confusion Matrix
  + Accuracy vs. Epochs
  + Prediction vs. Ground Truth Comparison
* **Frontend Output:** Predictions shown instantly after uploading CSV file

**Issues Faced & Fixes:**

|  |  |
| --- | --- |
| **Issue Faced** | **Fix Applied** |
| Model file not found | Ensured model.pkl and label\_encoder.pkl are in root directory |
| Label decoding error | Used .inverse\_transform() method correctly |
| Incorrect CSV format | Validated column names before processing |
| Flask app crash | Wrapped model prediction in try-except block |
| Graphs not rendering | Installed matplotlib and used savefig() |

**Function Descriptions:**

**predict\_new.py**

* load\_model(): Loads the trained classifier from model.pkl
* load\_encoder(): Loads label encoder from label\_encoder.pkl
* extract\_features(df): Extracts the required features from the uploaded CSV
* predict(df): Predicts the threat labels
* decode\_labels(pred): Converts encoded labels back to original

**app.py (Flask)**

* /: Home route to upload CSV
* /predict: Processes CSV, returns predictions as HTML table

**model.py**

 load\_data(filepath)  
Loads dataset from a CSV file and returns a DataFrame.

 preprocess\_data(df)  
Cleans the data, encodes categorical columns, and separates features (X) and labels (y).

 split\_data(X, y)  
Splits the data into training and testing sets.

 train\_model(X\_train, y\_train)  
Trains a Random Forest model on the training data.

 evaluate\_model(model, X\_test, y\_test)  
Evaluates model accuracy or performance on test data.

 encode\_labels(y)  
Converts threat labels into numeric format using LabelEncoder.

 save\_model(model)

 Saves the trained model as a .pkl file.

 save\_encoder(encoder)  
Saves the label encoder to disk for future decoding.

**Log Issue List:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No** | **Issue** | **Type** | **Fixed (Yes/No)** |
| 1 | Model not loading | Code bug | Yes |
| 2 | Prediction mismatch | Logic error | Yes |
| 3 | GUI not displaying output | UI Issue | Yes |
| 4 | File type error | Input format | Yes |
| 5 | LabelEncoder not found | Dependency | Yes |

## **Learnings:**

* Handling real-time data input for ML systems
* Model persistence and deployment via Flask
* Interpreting model performance using visualizations
* Team collaboration and modular development
* Debugging and improving model inference speed

**Conclusion:**

This project successfully demonstrates a real-time threat detection module using machine learning. With a clean interface, high model accuracy, and modular code, it reflects practical deployment of AI in cybersecurity domains.