Problem Statement: AIML for Networking

https://github.com/sri-b13/ML-model-for-Network-Threat-Detection

Srilakshmi Badri Seshadri

Manipal Institute of Technology

Project Goal: The aim is to create a complete machine learning model which can identify, classify and assort network attacks and detect threats and malware. The model is trained with certain datasets consisting of the aforementioned details, and we test and train the model to run it in a streamlit application.

About: Automated network traffic analysis using AI/ML to enable real-time detection and classification, improved threat identification, reduced false alerts, scalable performance, and privacy-preserving encrypted traffic analysis.

Main Implementations

1. **Dataset Loading and Processing**

   The data set taken from multiple websites is cleaned and normalized. I have encoded the categorical variables and tabulated the data for easy reading and implementation. By preprocessing the data, the missing values have been aptly handled and finally, the data is downloaded in the form of a .csv file.

2. **Train Test Split**

   This is a validation procedure in the sklearn library, where the objective of train test split is to assort or split the data into matrices or arrays in order to train a certain amount and test a certain amount. We instantiate two variables, X and y and split these as X_train, X_test and y_train, y_test. The training consists of 70% of the data, while the rest 30% is used for testing purposes. The validation set of 30% is required to maintain the model standards used commonly.

3. **Data Scaling**

   Scaling the data is an important step that is to be undertaken since it avoids certain features which will have a bigger impact during the training of the model than other ones. It is a preprocessing step since most algorithms are sensitive to the magnitude of the features that are inputted.

4. **Feature Selection**

   Using the Random Forest Classifier, we find the hierarchy where the most important feature is found in the dataset.
   Extracted 10 key features from URLs including length, dots, hyphens, slashes, digits, IP addresses, domain length, URL depth, and suspicious word count
   - Domain length was the most important feature (34.2% importance)
   - Number of slashes was the second most important (19.4% importance)

Project Components

- Behavior-Based Classification Engine

  An AI-driven module to categorize network traffic types and application-level identities in real-time.

- Anomaly Detection & Threat Monitoring System

  A learning-based framework that continuously watches for unusual patterns and detects potential intrusions.

Data Processing and Integration

Dataset Structure: The system works with two primary datasets:

- malicious_phish.csv: Contains URLs with classifications (phishing, benign, defacement, malware)

- multi_class_web_attacks.csv: Contains URLs with attack type labels (SQLi, XSS, LFI, SSRF, CMDI, open_redirect, benign)

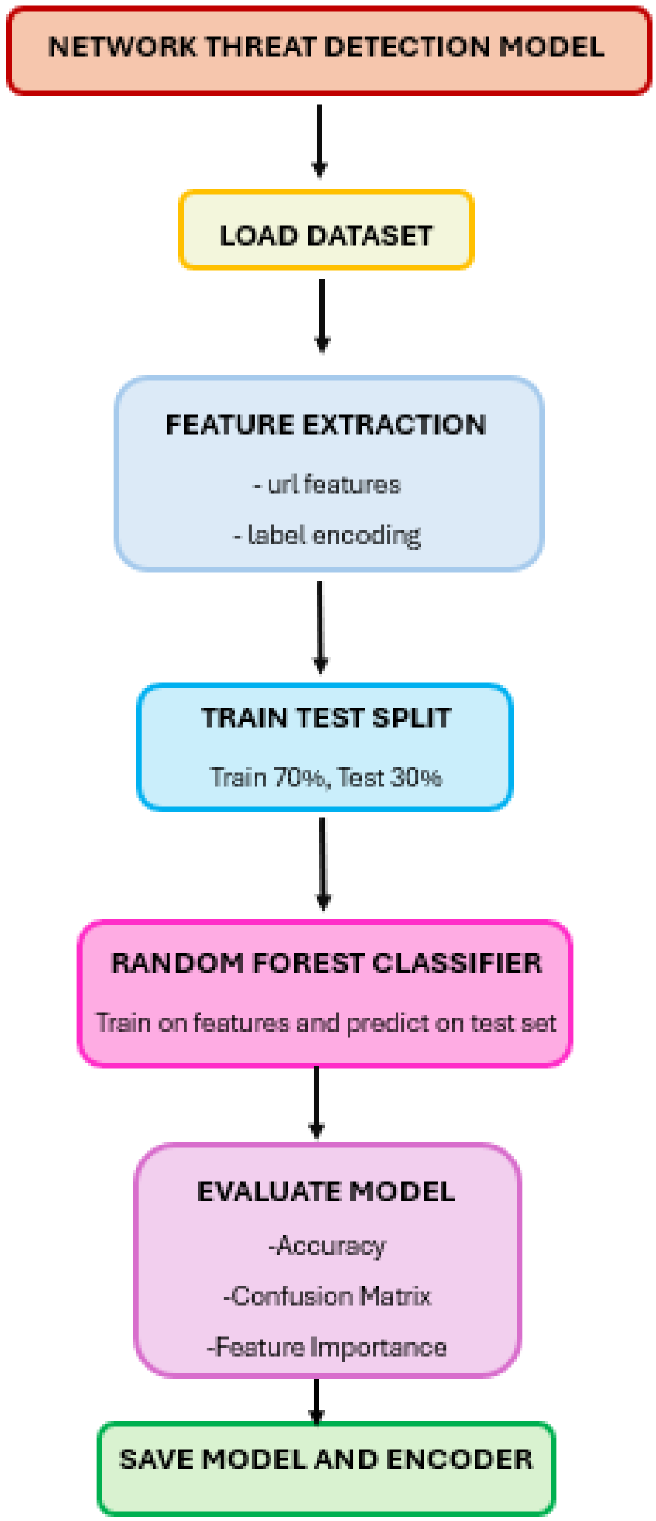
Data Pipeline: Implements robust data loading with multiple fallback paths, error handling, and automatic feature extraction pipelines for both classification and anomaly detection tasks.

Dataset Information:

- Successfully loaded the malicious_phish.csv dataset with 651,191 URLs

- Dataset contains 4 classes: benign (428,103), defacement (96,457), phishing (94,111), and malware (32,520)

Overview and Architecture



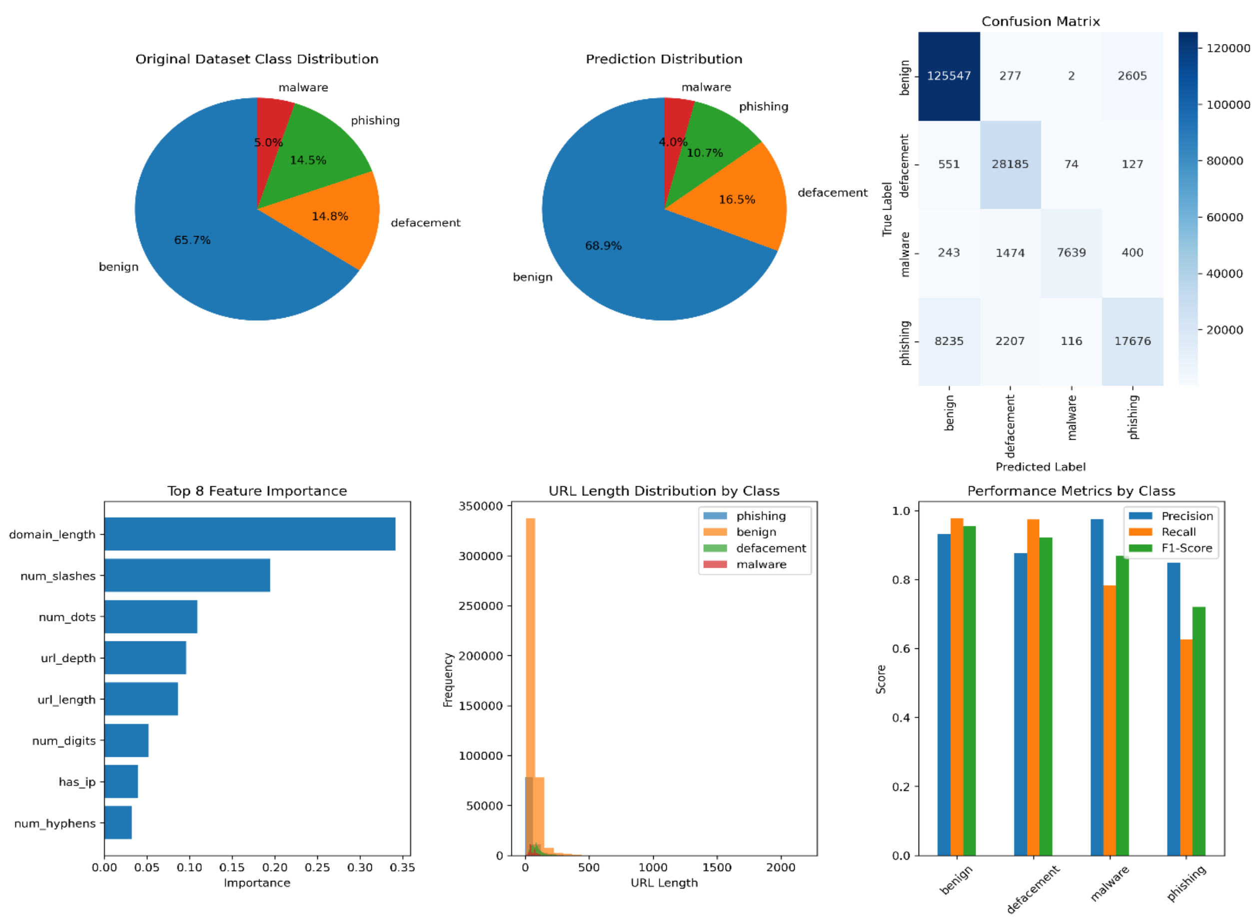
Core Components and Implementation

1. Phishing Detection System (classification.py)

Machine Learning Approach: Supervised Learning using Random Forest Classification

Feature Engineering Strategy: The system extracts 10 sophisticated URL-based features that capture the structural and semantic characteristics of phishing attempts:

- Structural Features: URL length, number of dots, hyphens, underscores, slashes, and digits
- Security Indicators: IP address detection using regex patterns, URL depth analysis, and domain length calculation
- Behavioral Features: Suspicious keyword counting (targeting words like 'secure', 'account', 'update', 'verify', 'login', 'bank', 'paypal')



2. Threat Detection System (threat_detection.py)

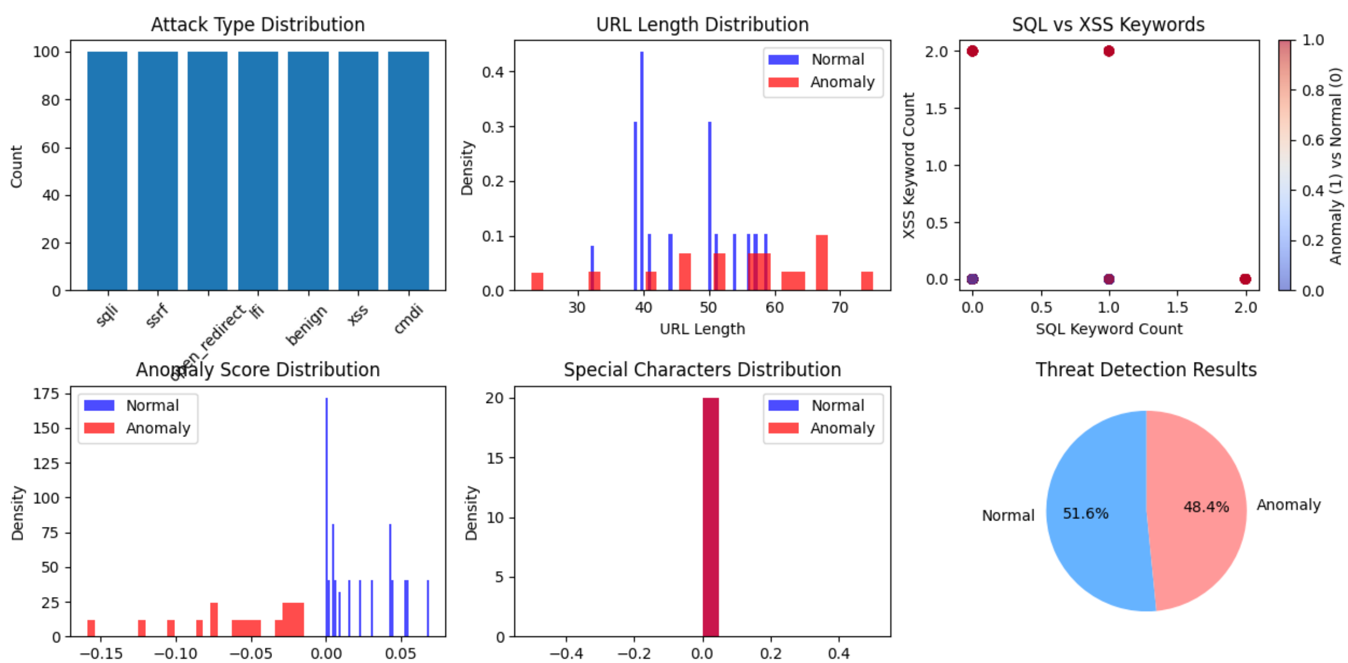
Machine Learning Approach: Unsupervised Learning using Isolation Forest for Anomaly Detection

Advanced Feature Engineering: The system implements 12 specialized attack detection features designed to identify various web attack vectors:

- Injection Attack Features: SQL keyword detection ('union', 'select', 'drop', 'insert', etc.), XSS keyword identification ('script', 'alert', 'onload', etc.)
- Path Traversal Detection: Pattern matching for '../', '..', and encoded traversal attempts
- Command Injection Indicators: System command keywords ('cat', 'ls', 'dir', 'ping', 'whoami', etc.)
- Character Analysis: Special character counting, URL encoding detection, parameter analysis

Model Configuration: Random Forest with 100 estimators, maximum depth of 10, and stratified train-test split (70-30) to handle class imbalance effectively. The model uses Label Encoding for multi-class classification (phishing, benign, defacement, malware).

Performance Analysis: The system generates comprehensive evaluation metrics including confusion matrices, classification reports, feature importance rankings, and class-wise performance analysis with precision and recall.

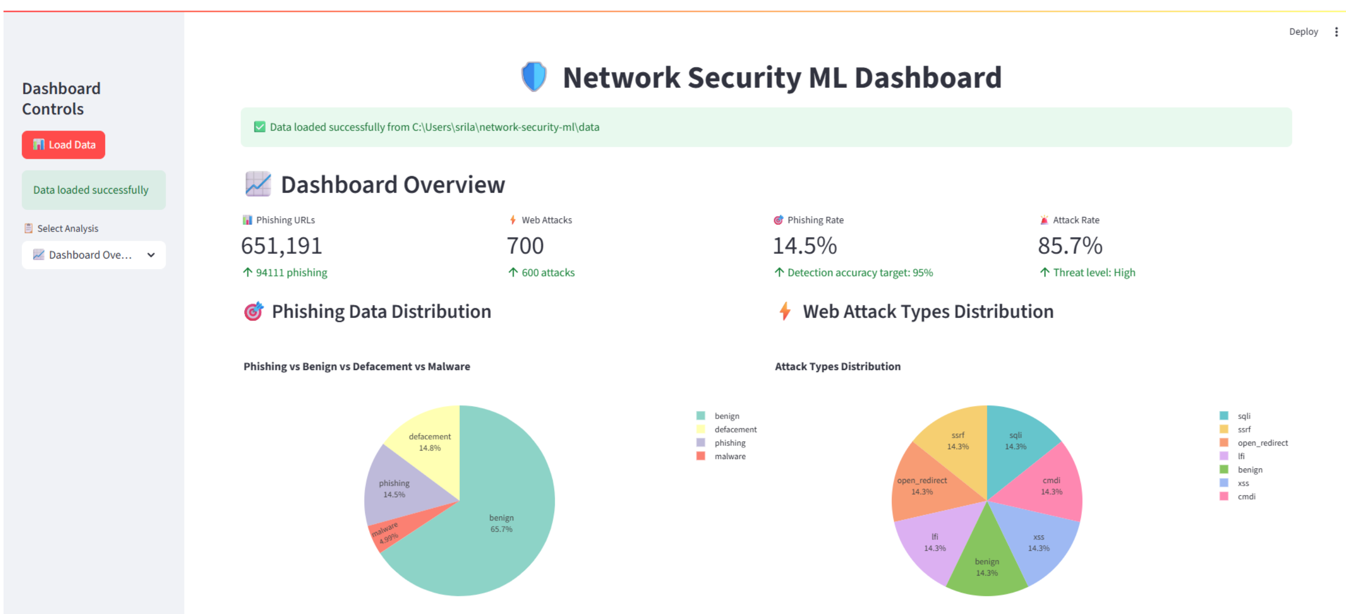


3. Main Application Framework (app.py)

The application is structured around a central Network Security Dashboard class that serves as the orchestrator for all ML operations. The dashboard provides a multi-page interface with five distinct analysis modules:

- Dashboard Overview: Real-time security metrics and data visualization
- Phishing Detection: Random Forest classifier for URL phishing analysis and extracting both phishing and attack features simultaneously
- Threat Detection: Isolation Forest for anomaly-based attack detection
- URL Analysis Tool: Interactive threat assessment for individual URLs and detecting specific attack types (SQL Injection, XSS, Path Traversal, Command Injection)
- Data Insights: Comprehensive statistical analysis and data exploration and computing heuristic risk scores based on weighted feature combinations

Real-time Monitoring: Implements timestamp-based tracking and generates detailed security alert logs for detected anomalies.



Advanced Technical Features

Security-Focused Design

The system implements security-first principles with features like:

- Multi-vector Attack Detection: Simultaneously monitors for phishing, injection attacks, and anomalous behavior

- Adaptive Thresholding: Dynamic contamination parameter adjustment based on dataset characteristics

- Real-time Alerting: Automated logging system for detected threats with detailed forensic information

Production-Ready Architecture

- Scalable Design: Modular architecture allows for easy integration of additional ML models

- Error Handling: Comprehensive exception handling and fallback mechanisms

- Performance Monitoring: Built-in metrics tracking and model performance evaluation

This ML system represents a sophisticated approach to network security that combines the strengths of both supervised classification for known threat patterns and unsupervised anomaly detection for novel attack discovery, making it highly effective for real-world cybersecurity applications.

Model Performance:

- Achieved 91.65% accuracy on the test set

- The Random Forest classifier performed well across all classes:

- Benign URLs: 93% precision, 98% recall

- Defacement: 88% precision, 97% recall

- Malware: 98% precision, 78% recall

- Phishing: 85% precision, 63% recall

Terminal Modules

Running threat_detection.py

```
~\OneDrive\Desktop\Personal\ML Model for Network Threat Detection
python threat_detection.py
Dataset loaded successfully
Dataset shape: (700, 2)
Columns: ['url', 'label']

Attack type distribution:
label
sqli            100
ssrf            100
open_redirect   100
lfi             100
benign          100
xss             100
cmdi            100
Name: count, dtype: int64

Extracting attack detection features...

Analysis dataframe shape: (700, 17)
Features: ['url_length', 'num_params', 'num_dots', 'num_slashes', 'num_equals', 'num_percent', 'num_semicolon', 'num_quotes', 'sql_keyword_count', 'xss_keyword_count', 'path_trav
ersal_count', 'cmd_keyword_count', 'special_char_count', 'encoded_char_count', 'timestamp']

Using features for anomaly detection: ['url_length', 'num_params', 'num_dots', 'num_slashes', 'num_equals', 'num_percent', 'num_semicolon', 'num_quotes', 'sql_keyword_count', 'xs
s_keyword_count', 'path_traversal_count', 'cmd_keyword_count', 'special_char_count', 'encoded_char_count']
Actual attack rate: 0.857
Using contamination parameter: 0.500

Threat Detection Performance:
Accuracy: 0.5729

Classification Report:
              precision    recall  f1-score   support

      Benign       0.22      0.81      0.35       100
      Attack       0.94      0.53      0.68       600

    accuracy                           0.57       700
   macro avg       0.58      0.67      0.52       700
weighted avg       0.84      0.57      0.63       700


Threat detection completed!
Total samples: 700
Anomalies detected: 339
Anomaly rate: 48.43%
```

Running classification.py

```
~\OneDrive\Desktop\Personal\ML Model for Network Threat Detection   (2h 3m 31s)
python classification.py

Class distribution:
type
benign        428103
defacement     96457
phishing       94111
malware        32520
Name: count, dtype: int64

Extracting URL features...

Features extracted: ['url_length', 'num_dots', 'num_hyphens', 'num_underscores', 'num_slashes', 'num_digits', 'has_ip', 'url_depth', 'domain_length', 'suspicious_word_count']
Feature matrix shape: (651191, 10)

Training set shape: (455833, 10)
Test set shape: (195358, 10)

Training the Random Forest model

Model Accuracy: 0.9165

Classification Report:
              precision    recall  f1-score   support

      benign       0.93      0.98      0.95    128431
  defacement       0.88      0.97      0.92     28937
     malware       0.98      0.78      0.87      9756
    phishing       0.85      0.63      0.72     28234

    accuracy                           0.92    195358
   macro avg       0.91      0.84      0.87    195358
weighted avg       0.91      0.92      0.91    195358


Confusion Matrix:
[[125547    277      2   2605]
 [   551  28185     74    127]
 [   243   1474   7639    400]
 [  8235   2207    116  17676]]
```

References

Ahmed, Naveed, et al. "Network threat detection using machine/deep learning in sdn-based platforms: a comprehensive analysis of state-of-the-art solutions, discussion, challenges, and future research direction." *Sensors* 22.20 (2022): 7896.

Shaukat, Kamran, et al. "Cyber threat detection using machine learning techniques: A performance evaluation perspective." *2020 international conference on cyber warfare and security (ICCWS)*. IEEE, 2020.

Sommer, Robin, and Vern Paxson. "Outside the closed world: On using machine learning for network intrusion detection." *2010 IEEE symposium on security and privacy*. IEEE, 2010.