Phishing Detection System using Machine Learning

This project implements a Machine Learning-based cybersecurity system designed to detect malicious URLs (phishing) without analyzing page content. It utilizes a supervised learning approach with an XGBoost classifier trained on 30 handcrafted lexical and host-based features.

Project Overview

The system provides a complete pipeline from data collection to deployment, offering:

- **Real-time Analysis:** Instant classification of URLs via a web interface or REST API.
- **Feature Extraction:** Automated extraction of lexical, structural, and host-based features.
- **Model Explainability:** Visualization of extracted features and confidence scores.
- **Scalability:** Lightweight architecture suitable for integration into existing security workflows.

Features

- Real-time URL scanning and classification.
- REST API integration for automated queries.
- Extraction of 30 distinct URL features (e.g., length, entropy, IP usage).
- High-performance XGBoost model optimized for binary classification.
- Confidence probability scoring.
- Web interface for manual testing and demonstration.

Installation

Prerequisites

- Python 3.10 or higher
- Conda (recommended) or pip
- Git

Setup Instructions

1. **Clone the repository**

```
git clone <your-repo-url>
cd phishing_project
```

2. **Create and activate the environment** Using Conda:

```
conda env create -f environment.yaml
conda activate phishing-ml-env
```

Using pip:

```
pip install flask pandas scikit-learn xgboost joblib tldextract
```

3. **Verify Model Availability** Ensure the trained model exists in the correct directory:

```
ls models/modelo_xgb_url_final_full.pkl
```

If the model is missing, refer to the "Model Training" section below.

Usage

Web Application

To launch the graphical interface:

```
cd app
python 4_aplicacion_mlops.py
```

Access the application at: `http://localhost:5000`

REST API

The system exposes endpoints for programmatic access.

1. Prediction Endpoint

- **URL:** `/api/predict`
- **Method:** `POST`
- **Header:** `Content-Type: application/json`
- **Body:** `{"url": "https://www.example.com"}`

Example Request:

```
curl -X POST http://localhost:5000/api/predict \
  -H "Content-Type: application/json" \
  -d '{"url": "[https://www.example.com](https://www.example.com)"}'
```

Example Response:

```
{
  "success": true,
  "url": "[https://www.example.com](https://www.example.com)",
  "prediction": "Legitimate",
  "is_phishing": false,
```

```
  "confidence": 95.23,
  "features_extracted": 30,
  "features": {
    "UseIP": 0,
    "URLLength": 23,
    "HTTPS": 1
  }
}
```

2. Health Check

- **URL:** `/health`
- **Method:** `GET`

Model Training

To reproduce the results or retrain the model with new data:

1. **Dataset Construction:**

   ```
   python src/build_url_dataset_full.py
   ```

2. **Feature Extraction:**

   ```
   python src/extract_url_features_full.py
   ```

3. **Model Training:**

   ```
   python src/train_final_url_model_full.py
   ```

   The artifacts will be saved to the `models/` directory.

Project Structure

```
phishing_project/
├── app/
│   ├── 4_aplicacion_mlops.py       # Flask API and Web App
│   ├── templates/                  # HTML Templates
│   └── static/                     # Static assets (CSS, Images)
├── src/
│   ├── url_features.py             # Feature extraction logic
│   ├── build_url_dataset_full.py   # Dataset compilation script
│   ├── extract_url_features_full.py # Batch feature processing
│   └── train_final_url_model_full.py # Model training script
├── data/
```

```
│   ├── raw/                        # Original CSV datasets
│   └── processed/                  # Feature-engineered datasets
├── models/
│   └── modelo_xgb_url_final_full.pkl # Serialized XGBoost model
├── notebooks/                      # Exploratory Data Analysis (EDA)
├── environment.yaml                # Environment dependencies
└── README.md                       # Project documentation
```

Technical Details

Extracted Features

The model relies on 30 handcrafted features, including:

- **Lexical:** URL length, digit count, special character density (@, -, .).
- **Host-based:** IP address usage, TLD analysis, subdomain count.
- **Protocol:** HTTPS usage, redirect analysis.

Model Configuration

The core classifier is an XGBoost model configured with:

- **Estimators:** 600
- **Max Depth:** 8
- **Learning Rate:** 0.05
- **Objective:** Binary Logistic
- **Class Balancing:** Enabled via scale_pos_weight

License

This project is part of an Applied Artificial Intelligence Master's .