🛡️ AI-Powered Phishing Email Detector

An end-to-end machine learning system that detects phishing emails using natural language processing and classification algorithms.

Python 3.8+

License MIT

ML Scikit-Learn

📋 Table of Contents

🎯 Overview

Phishing emails remain one of the most common initial attack vectors for cybercriminals. Traditional rule-based systems often fail against new, adaptive phishing techniques. This project implements a machine learning approach to detect phishing emails by analyzing text patterns, linguistic features, and structural characteristics.

PROF

Problem Statement

- Phishing attacks cause billions of dollars in losses annually
- Rule-based detection systems can't adapt to new phishing techniques
- Need for automated, intelligent email classification

Solution

A Logistic Regression classifier trained on TF-IDF features from email text, providing:

- Binary classification (phishing vs. legitimate)
- Probability scores for risk assessment
- Fast, explainable predictions

✨ Features

- **Data Pipeline**: Automated data ingestion and cleaning
- **Text Preprocessing**: URL/email normalization, HTML removal, tokenization
- **Feature Engineering**: TF-IDF vectorization with n-grams
- **Model Training**: Logistic Regression with cross-validation
- **Evaluation**: Comprehensive metrics and visualizations
- **Web Interface**: Interactive Streamlit application for real-time predictions

📁 Project Structure

```
ai-phishing-email-detector/
├── data/
│   ├── raw/                    # Raw email datasets
│   └── processed/              # Cleaned and processed data
├── docs/
│   ├── evaluation_report.txt   # Model evaluation results
│   └── plots/                  # Visualization outputs
├── notebooks/                  # Jupyter notebooks for exploration
├── src/
│   ├── data/
│   │   └── make_dataset.py     # Data ingestion module
│   ├── features/
│   │   └── build_features.py   # Feature engineering module
│   ├── models/
│   │   ├── train.py            # Model training module
│   │   └── evaluate.py         # Model evaluation module
│   ├── ui/
│   │   └── app.py              # Streamlit web application
│   ├── api/                    # REST API (optional)
│   └── utils/                  # Utility functions
├── tests/                      # Unit tests
├── .gitignore
├── README.md
└── requirements.txt
```

PROF

📊 Dataset

Recommended Datasets

Dataset	Source	Description
Enron Spam	Kaggle	Classic email spam dataset
Phishing Email	Kaggle	Labeled phishing emails
CEAS 2008	CEAS	Conference anti-spam corpus

Data Format

The dataset should be a CSV file with at least these columns:

```
email_text,label
"Email content here...",0
"Suspicious email here...",1
```

Where:

- `label = 0` → Legitimate email
- `label = 1` → Phishing email

Data Placement

Place your dataset at: `data/raw/emails.csv`

> ⚠ **Note**: Raw data is excluded from version control. Check dataset licenses before redistribution.

🚀 Installation

Prerequisites

- Python 3.8 or higher
- pip package manager
- Git

Setup

1. **Clone the repository**

   ```
   git clone https://github.com/<your-username>/ai-phishing-email-detector.git
   cd ai-phishing-email-detector
   ```

PROF

2. **Create virtual environment**

   ```
   python -m venv .venv
   source .venv/bin/activate  # Linux/Mac
   # or
   .venv\Scripts\activate     # Windows
   ```

3. **Install dependencies**

   ```
   pip install -U pip
   pip install -r requirements.txt
   ```

4. **Download dataset**
    - Download a phishing email dataset from Kaggle
    - Place it at `data/raw/emails.csv`

💻 Usage

1. Data Ingestion

Clean and preprocess the raw email data:

```
python src/data/make_dataset.py
```

2. Feature Engineering

Extract TF-IDF features from email text:

```
python src/features/build_features.py
```

3. Model Training

Train the Logistic Regression classifier:

```
python src/models/train.py
```

4. Model Evaluation

Generate evaluation metrics and plots:

PROF

```
python src/models/evaluate.py
```

5. Run Web Application

Launch the interactive Streamlit interface:

```
streamlit run src/ui/app.py
```

Then open http://localhost:8501 in your browser.

📈 Model Performance

Classification Report

```
              precision    recall  f1-score   support

  Legitimate       0.XX      0.XX      0.XX      XXXX
    Phishing       0.XX      0.XX      0.XX      XXXX

    accuracy                           0.XX      XXXX
   macro avg       0.XX      0.XX      0.XX      XXXX
weighted avg       0.XX      0.XX      0.XX      XXXX
```

Key Metrics

Metric	Value
Accuracy	XX.XX%
Precision	XX.XX%
Recall	XX.XX%
F1-Score	XX.XX%
ROC-AUC	XX.XX%

Confusion Matrix

```
              Predicted
              Leg    Phi
Actual  Leg   TN     FP
        Phi   FN     TP
```

PROF

> **Note**: Replace XX values with actual metrics after training.

🏗️ Architecture

```
┌─────────────────┐     ┌─────────────────┐     ┌─────────────────┐
│   Raw Emails    │────▶│  Preprocessing  │────▶│    TF-IDF       │
│   (CSV)         │     │  (clean_text)   │     │   Vectorizer    │
└─────────────────┘     └─────────────────┘     └─────────────────┘
                                                        │
                                                        ▼
┌─────────────────┐     ┌─────────────────┐     ┌─────────────────┐
│   Prediction    │◀────│   Logistic      │◀────│    Feature      │
│   (0/1 + prob)  │     │  Regression     │     │   Matrix        │
└─────────────────┘     └─────────────────┘     └─────────────────┘
```

Pipeline Steps

1. **Data Ingestion**: Load and validate raw email data
2. **Preprocessing**: Clean text, normalize URLs/emails, remove HTML
3. **Feature Extraction**: TF-IDF vectorization with unigrams and bigrams
4. **Model Training**: Logistic Regression with balanced class weights
5. **Evaluation**: Classification metrics, confusion matrix, ROC curve
6. **Inference**: Real-time prediction via Streamlit UI

⚖ Ethical Considerations

Privacy

- ✅ Only public datasets are used
- ✅ No real private emails are processed
- ✅ No personal data is stored or transmitted

Bias and Fairness

- Training data may have inherent biases
- Model performance may vary across different email types
- Regular evaluation on diverse datasets is recommended

Misuse Prevention

- This tool is for **educational and research purposes**
- Should not be the sole basis for blocking emails
- False positives can cause legitimate emails to be missed

Responsible AI

- Model decisions are explainable (feature coefficients)
- Probability scores provided for risk assessment
- Users encouraged to verify through official channels

PROF

Limitations

- May not detect zero-day phishing techniques
- Performance depends on training data quality
- Text-only analysis (doesn't check attachments or links)

🤝 Contributing

Contributions are welcome! Please follow these steps:

1. Fork the repository
2. Create a feature branch (`git checkout -b feature/amazing-feature`)
3. Commit your changes (`git commit -m 'Add amazing feature'`)
4. Push to the branch (`git push origin feature/amazing-feature`)

5. Open a Pull Request

Development Setup

```
# Install development dependencies
pip install -r requirements.txt

# Run tests
pytest tests/

# Format code
black src/

# Lint code
flake8 src/
```

📄 License

This project is licensed under the MIT License - see the LICENSE file for details.

🙏 Acknowledgments

- Scikit-learn for ML algorithms
- Streamlit for the web interface
- Kaggle for public datasets
- The cybersecurity community for research and datasets

⚠ **Disclaimer**: This tool is for educational purposes only. Always verify suspicious emails through official channels and consult cybersecurity professionals for critical decisions.