End-to-End AI Spam Detection System

A robust, automated solution for real-time email spam classification.

Our Project Team

Yeswanth Kosuri

AP23110010553

A,V. Ram Sathvik

AP23110010281

P. Vijay Yaswanth

AP23110010511

D. Lekith Kishore

AP23110011591

Project Purpose & Problem Statement

Spam emails pose significant threats, from phishing to malware. Our project aims to construct a highly accurate, rapid, and transparent spam detection engine.

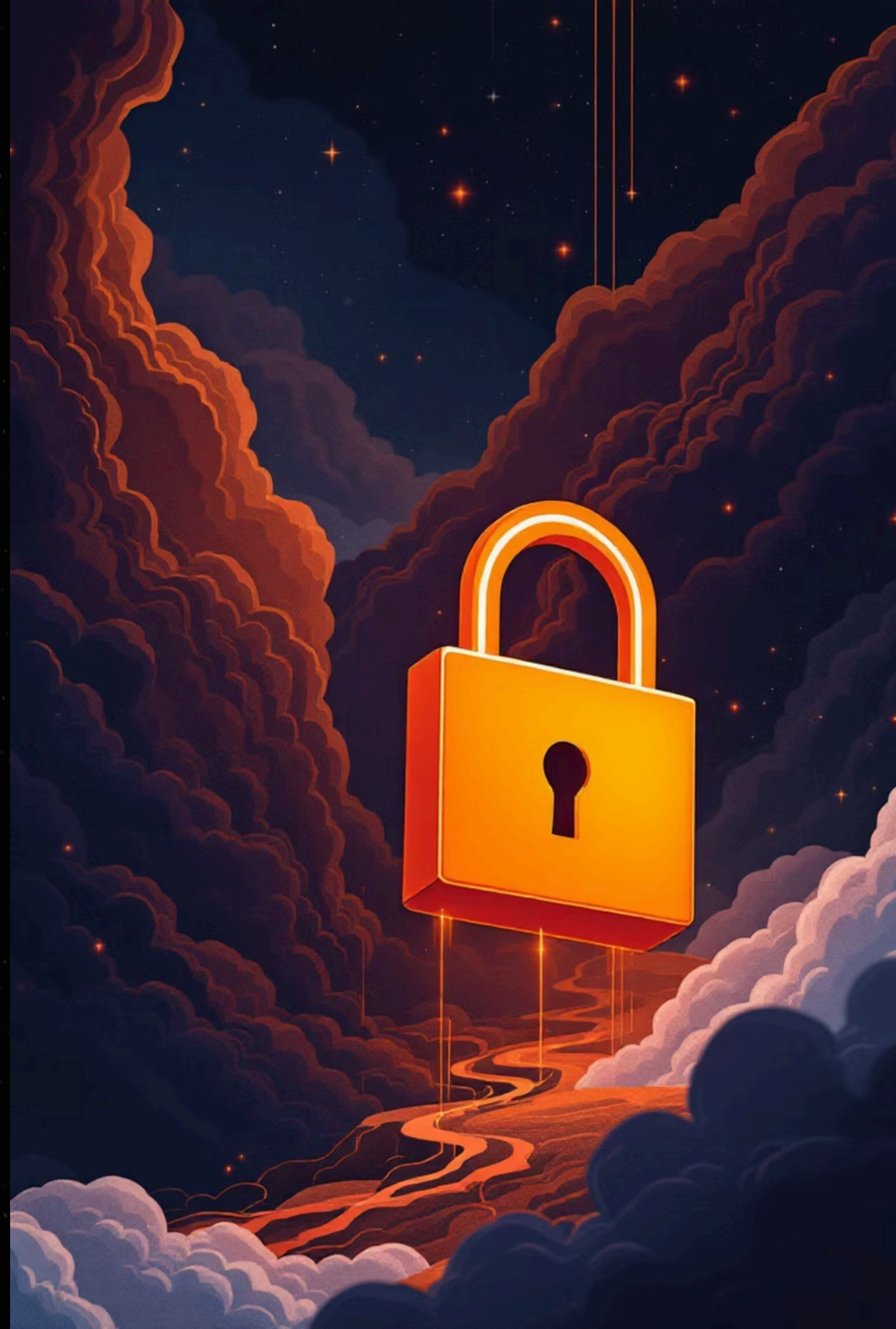
The Threat of Spam

Malicious links, phishing attempts, and scams hidden within unsolicited messages.

Our Solution

An explainable AI system combining advanced text processing and multiple ML models.

This system covers the full ML lifecycle, from data ingestion to real-time deployment.



End-to-End Workflow: Training Phase

The training phase meticulously prepares data and models for optimal spam detection.

01

Data Loading & Cleaning

CSV reading, null value removal, label mapping (ham→0, spam→1), and initial text cleaning.

02

Text Preprocessing

Standardization including lowercasing, extra space removal, and text normalization for consistency.

03

Train/Test Split

80% for training, 20% for testing, with stratification to maintain the original spam/ham ratio.

04

TF-IDF Vectorization

Transforms text into numerical vectors, identifying spam patterns like "win money" or "urgent click here."

05

Model Training

Four distinct ML models (Naive Bayes, SVM, Random Forest, Logistic Regression) are trained.

06

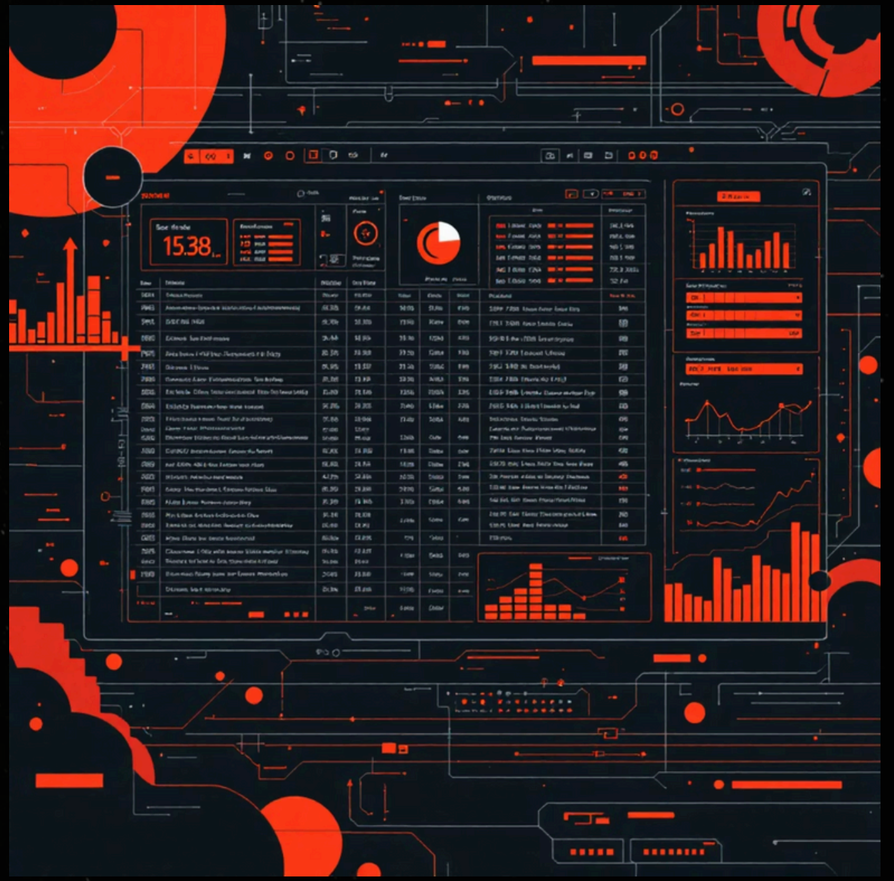
Evaluation & Saving

Models are evaluated on Accuracy, Precision, Recall, F1 Score, and then saved (.pkl files).

Dataset & Methodology Overview

Our system was developed using a comprehensive dataset of email messages, processed through a multi-stage methodology.

The **spam mail.csv** dataset contains 5572 messages, with a distribution of 4825 legitimate (ham) and 747 spam entries. Each message is labeled by category and contains the raw text content.

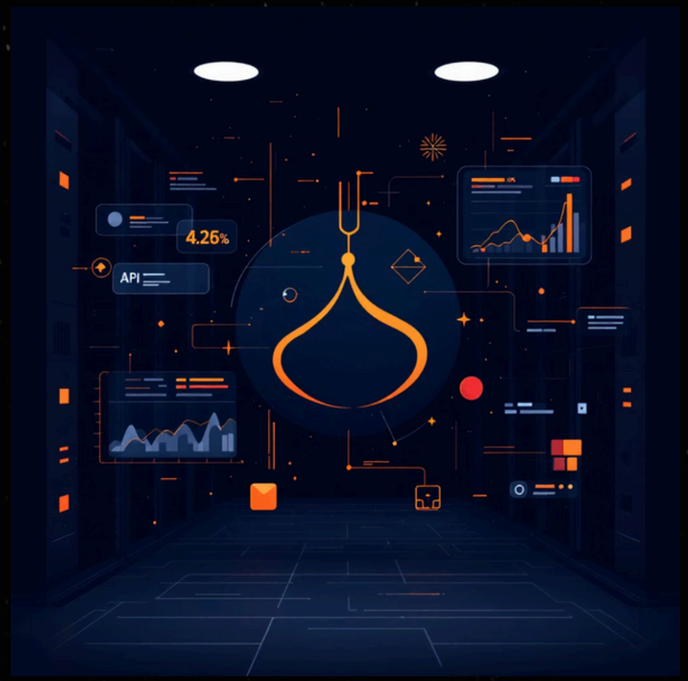


Key Methodological Components:

- **Text Preprocessing:** Lowercasing, space removal, and text normalization.
- **Feature Extraction:** TF-IDF with N-grams (max_features=3000, ngram_range=(1,3), stop_words='english') to capture meaningful text patterns.
- **Multiple ML Models:** Multinomial Naive Bayes, Linear SVM, Random Forest, and Logistic Regression.
- **Ensemble Prediction:** A weighted combination of individual model outputs for enhanced accuracy and robustness.

End-to-End Workflow: Prediction & Deployment

Our Flask-based API enables real-time spam detection with comprehensive results.



Flask Backend (app.py)

- Loads all trained models and the vectorizer at startup.
- Exposes a /api/predict endpoint.
- Accepts JSON input containing email subject and content.

Spam Prediction (predict_all())

1. Combines subject + content into a single text block.
1. Applies the same preprocessing as the training phase.
1. Utilizes regex for detecting known spam patterns ("account suspended", etc.).
1. Vectorizes the text using the trained TF-IDF model.

1. Generates probabilities from all four ML models.

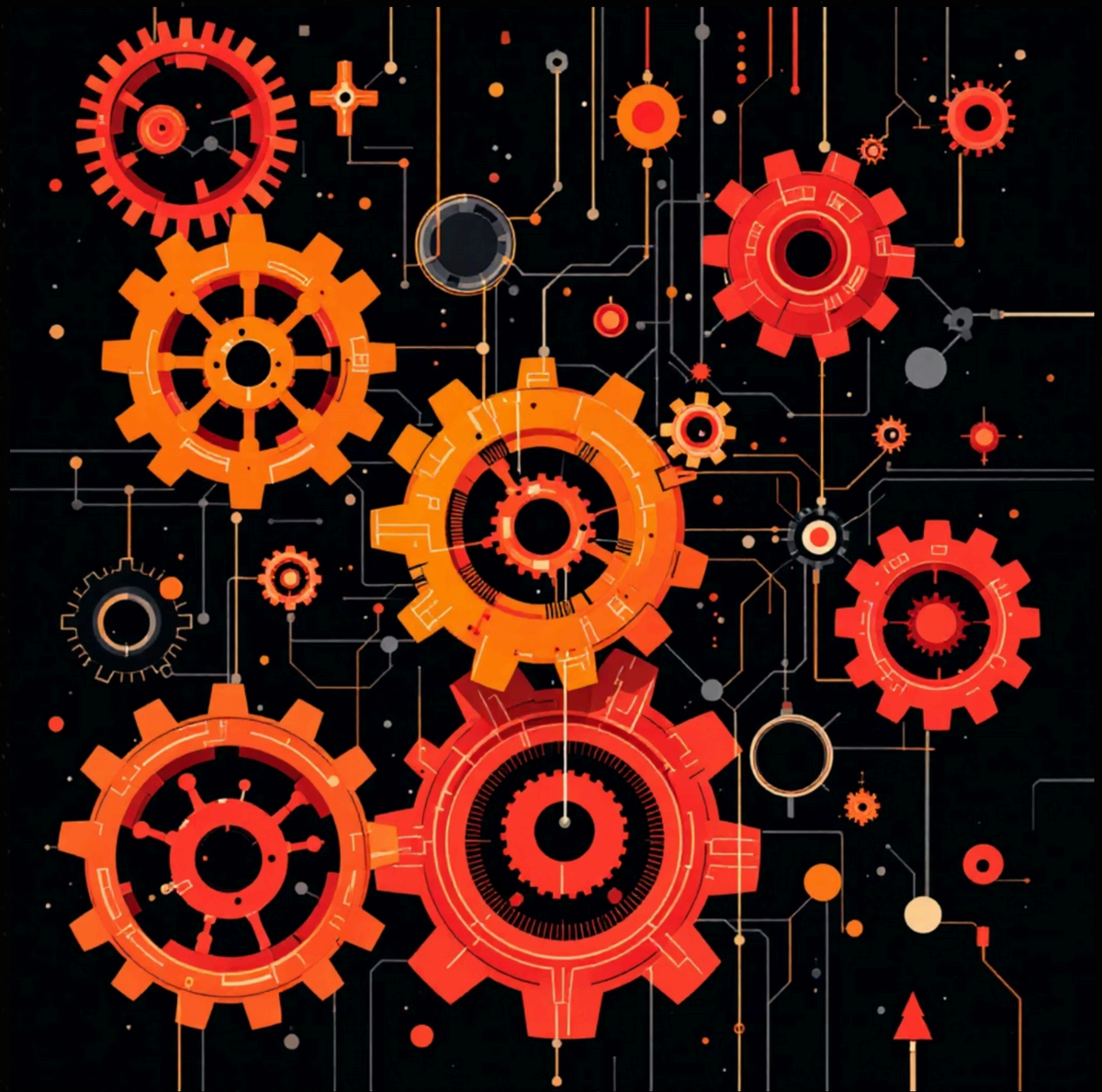
Ensemble Prediction: The Heart of Accuracy

The ensemble method combines the strengths of individual models for a more robust and stable final prediction.

Weighted Sum Approach

Each model contributes to the final spam score based on its proven performance and reliability.

Random Forest	35%
Support Vector Machine	25%
Multinomial Naive Bayes	20%
Logistic Regression	20%



Conclusion & Future Scope

This project delivers a complete, explainable, and scalable ML system for spam detection, embodying the full ML lifecycle.

Key Achievements

- Effective text cleaning and preprocessing.
- Robust TF-IDF and N-gram feature engineering.
- Four diverse ML models for comprehensive analysis.
- Advanced ensemble weighting for superior accuracy.
- Real-time Flask API for practical application.
- Detailed, explainable output with patterns and probabilities.

Future Enhancements

- Integrate deep learning models (e.g., Transformers).
- Expand regex patterns for emerging spam techniques.
- Implement active learning for continuous model improvement.
- Develop a user-friendly frontend for wider access.
- Explore multi-language spam detection.