

An Interpretable Multi-Signal Scam Detection System Using Machine Learning and Large Language Models

Author: Vishwajeet Adkine

February 2025

Keywords: *Scam Detection, Interpretable ML, LLM Safety, Ensemble Systems, Feature Engineering*

Graphical Abstract

Figure: High-level overview of the interpretable multi-signal scam detection system combining feature engineering, machine learning, and LLM safety validation.

Table of Contents

Graphical Abstract.....	1
Table of Contents.....	4
Abstract.....	5
1. Introduction	5
2. System Architecture	6
3. Feature Engineering.....	9
3.1 Textual Features (f1–f8)	11
3.2 URL Features (f9–f15)	11
3.3 Heuristic Signal (f16)	11
4. Model Performance.....	12
5. Model Explainability	15
5.1 Key Findings.....	17
5.2 Explainability Example.....	18
6. Ensemble Decision Strategy	21
7. Precision-Recall Trade-offs.....	24
8. Deployment Architecture	27
8.1 Key Design Principles.....	27
9. Limitations and Future Work	28
9.1 Current Limitations	28
9.2 Future Directions	28
10. Conclusion	29
11. References.....	30
Appendix A: All Research Figures.....	31

Abstract

Scam and phishing detection systems often rely either on rigid heuristic rules or opaque large language models (LLMs). Heuristics lack generalization, while LLMs are costly and difficult to audit. This work presents a hybrid, interpretable scam detection pipeline that combines a feature-based supervised machine learning model, semantic analysis via an LLM safety model, and rule-based heuristics within a unified ensemble decision framework. The proposed system emphasizes explainability, precision–recall trade-offs, and deployment realism, making it suitable for real-world AI safety applications.

1. Introduction

Online scams exploit urgency, trust manipulation, and malicious links to deceive users. Traditional approaches fall into three categories:

1. Rule-based systems – Precise but brittle, easily bypassed by novel attacks
2. Machine learning classifiers – Generalizable but often opaque, lacking interpretability
3. LLM-based moderation – Semantically powerful but expensive, slow, and unstable

This research explores whether a lightweight, interpretable ML model, when combined with LLM-based semantic checks and heuristics, can provide robust scam detection without over-reliance on any single method.

2. System Architecture

The proposed system employs a multi-layered architecture that separates concerns between user interface, business logic, machine learning inference, and optional semantic validation. This design ensures scalability, maintainability, and independent iteration of each component.

Figure 1: Multi-layer system architecture showing data flow from user input through backend services to ML and LLM components.

3. Feature Engineering

Instead of end-to-end deep learning, the system relies on explicit feature design based on domain knowledge of scam behavior. A total of 16 features are extracted from each message, capturing textual patterns, URL characteristics, and heuristic signals.

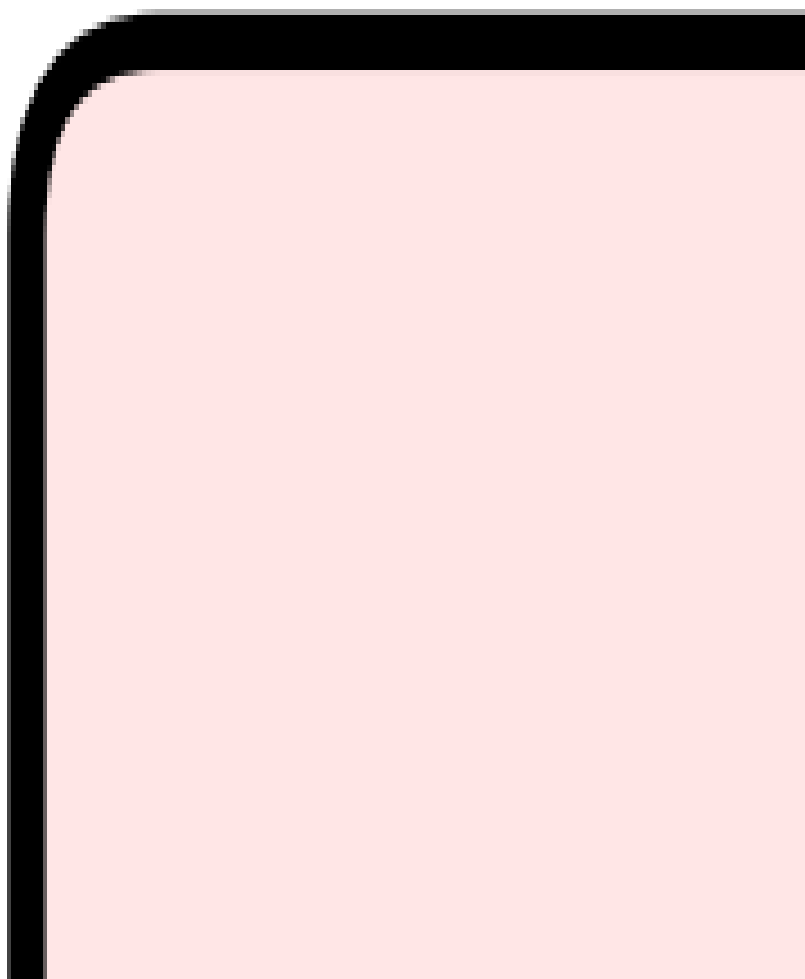


Figure 2: Feature extraction pipeline showing transformation of raw messages into a 16-dimensional feature vector for ML classification.

3.1 Textual Features (f1–f8)

These features capture linguistic and structural patterns commonly found in scam messages:

- f1: Urgency cues (e.g., 'urgent', 'verify now', 'suspended')
- f2: Money-related keywords ('lottery', 'free money', 'earn')
- f3: Sensitive intent (OTP, PIN, password, CVV)
- f4: Off-platform contact (Telegram, WhatsApp)
- f5: Text length (character count)
- f6: Exclamation marks count
- f7: Uppercase letter ratio
- f8: Digit ratio

3.2 URL Features (f9–f15)

URL-based features identify suspicious link patterns:

- f9: Number of URLs embedded in the message
- f10: URL-to-word ratio (high ratio indicates link-heavy content)
- f11: IP-based URL presence (e.g., `http://192.168.1.1`)
- f12: URL shortener detection (bit.ly, tinyurl)
- f13: Risky top-level domain (.tk, .ml, .ga, .pw)
- f14: Domain spoofing attempt (typosquatting)
- f15: Verified domain (google.com, github.com)

3.3 Heuristic Signal (f16)

A manually designed heuristic score provides domain expertise that might take thousands of samples for ML to learn independently. This score is capped to prevent data leakage and used as an additional input feature.

4. Model Performance

The logistic regression model, trained on 1000 samples (60% scam, 40% safe), demonstrates strong performance across multiple evaluation metrics. The model prioritizes high recall for scam detection to minimize false negatives, as missing a scam has more severe consequences than a false alarm.

Figure 3: Comprehensive performance analysis showing (left) class-wise metrics, (center) confusion matrix, and (right) multi-dimensional performance profile.

5. Model Explainability

Explainability is achieved through coefficient-based attribution. Each feature's contribution to the final prediction can be transparently analyzed, enabling security audits and trust in automated decisions.

Figure 4: Feature importance visualization showing logistic regression coefficients. Positive values (red) indicate scam signals, while negative values (green) indicate safety signals.

5.1 Key Findings

Analysis of feature coefficients reveals the following patterns:

- Sensitive terms (f3) have the highest positive weight (+0.42), indicating strong scam correlation
- Urgency keywords (f1) and URL shorteners (f12) are also strong indicators
- Verified domains (f15) provide negative evidence, reducing scam probability
- Text length (f5) shows slight negative correlation, as scams tend to be concise

5.2 Explainability Example

The following example demonstrates how the model arrives at its decision for a specific input message.

Figure 7: Feature contribution breakdown for the message 'URGENT! Verify your OTP at bit.ly/verify'. Each feature's contribution to the final scam score is shown, enabling transparent decision-making.

6. Ensemble Decision Strategy

The final classification leverages multiple independent signals to ensure robustness and reduce single points of failure. Heuristics provide fast initial filtering, the ML model offers interpretable predictions, and the LLM validates ambiguous cases.

Figure 5: Multi-signal ensemble decision workflow showing how heuristic, ML, and LLM signals are combined to produce final verdicts: Safe, Suspicious, or Scam.

7. Precision-Recall Trade-offs

Given the safety-critical nature of scam detection, recall for the scam class is prioritized to minimize false negatives. The system operates at high recall (1.0) while maintaining strong precision (0.95), ensuring comprehensive scam coverage without excessive false alarms.

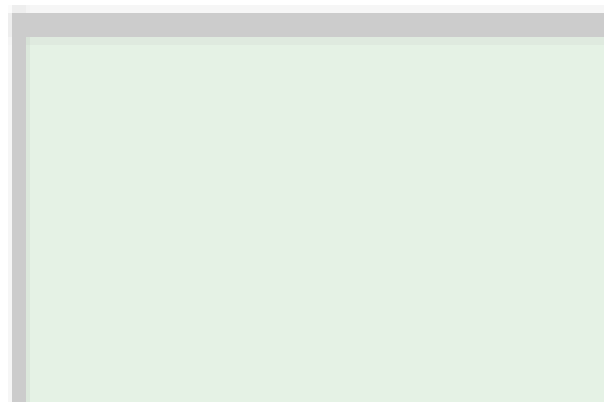


Figure 6: (Left) Precision-recall curve showing model performance across different decision thresholds. (Right) Confidence score distribution demonstrating well-calibrated probabilistic predictions.

8. Deployment Architecture

The ML model is deployed as a Python microservice, consumed by a Node.js backend. This separation of concerns enables independent ML iteration, scalable integration, and technology flexibility for each layer.

8.1 Key Design Principles

- Microservice architecture for independent scaling and deployment
- RESTful API for language-agnostic integration
- Inference latency <5ms for real-time user experience
- Optional LLM invocation only for ambiguous cases (cost optimization)
- Comprehensive logging and monitoring for continuous improvement

9. Limitations and Future Work

9.1 Current Limitations

- Dataset size (1000 samples) limits generalization claims
- Linear model restricts capturing non-linear feature interactions
- Static features may miss temporal and contextual patterns
- English-only support, no multi-language capability
- No image analysis for screenshot-based scams

9.2 Future Directions

Short-term:

- Expand dataset to 10K+ samples from real-world sources
- Implement full SHAP value visualization for instance-level explanations
- Cross-validation with k-fold splits for robust performance estimates

Medium-term:

- Gradient-boosted models (XGBoost, LightGBM) for non-linear patterns
- Domain reputation APIs (VirusTotal, URLhaus) integration
- Multi-language support for global deployment

Long-term:

- Online learning pipeline for continuous model updates
- Active learning with human-in-the-loop feedback
- Adversarial robustness testing against evasion attacks

10. Conclusion

This work demonstrates that interpretable machine learning, when integrated with LLM safety models and heuristics, can form a practical and auditable scam detection system. Rather than maximizing raw accuracy, the system prioritizes explainability, safety, deployment realism, and robustness through multi-signal validation.

Key Takeaways:

- Feature engineering surpasses black-box models in limited data regimes
- Linear models provide sufficient performance with superior interpretability
- Ensemble strategies offer robustness without over-reliance on any component
- Production-oriented design from day one enables real deployment

The approach aligns with real-world AI security requirements where interpretability and trust are as important as performance metrics.

11. References

4. Fette, I., Sadeh, N., & Tomasic, A. (2007). *Learning to detect phishing emails*. WWW 2007.
5. Lundberg, S. M., & Lee, S. I. (2017). *A unified approach to interpreting model predictions (SHAP)*. NeurIPS 2017.
6. Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). *'Why should I trust you?': Explaining predictions of any classifier*. KDD 2016.
7. Iyer, R., Li, Y., Li, H., et al. (2023). *Llama Guard: LLM-based input-output safeguard for human-AI conversations*. arXiv:2312.06674.
8. Platt, J. (1999). *Probabilistic outputs for support vector machines*. Advances in Large Margin Classifiers.

Appendix A: All Research Figures

This appendix contains all visualizations used in the research paper for quick reference.

Figure 1: System Architecture

Figure 2: Feature Engineering Pipeline

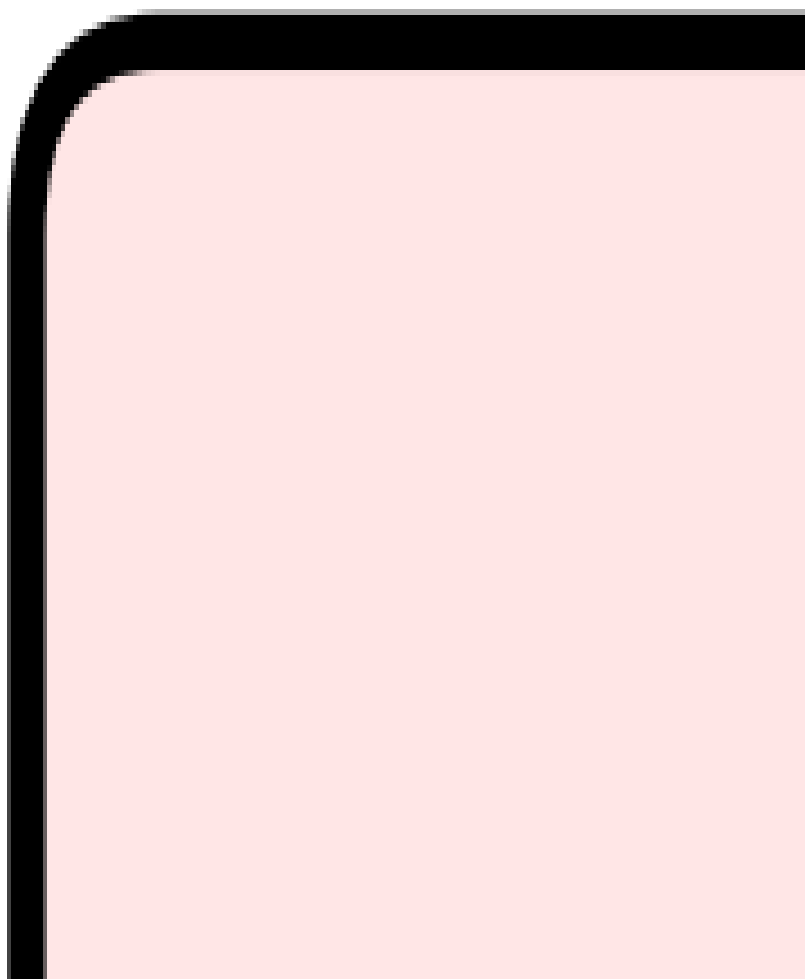


Figure 3: Performance Metrics

Figure 4: Feature Importance

Figure 5: Ensemble Decision Workflow

Figure 6: Precision-Recall Analysis

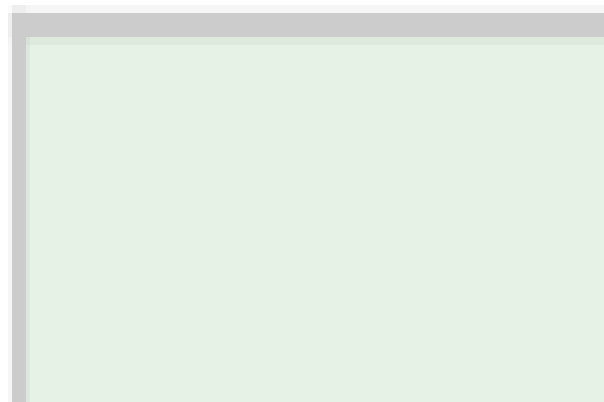


Figure 7: Explainability Example

— *End of Document* —