AEGIS-SECURE
Machine Learning Model

CONTENTS

1 Introduction **3**

1.1 The Problem Landscape 3

1.2 Problem Statement 3

2 System Architecture **4**

2.1 High-Level Architecture 4

2.2 Activity Workflow 6

2.3 Lifecycle Sequence 7

2.4 Backend Modularity 8

3 Level 1: Statistical ML **9**

3.1 Strategy: Speed & Feature Extraction 9

3.2 Support Vector Machines (SVM) 9

3.3 Logistic Regression 10

3.4 XGBoost (Extreme Gradient Boosting) 10

4 Level 2: Deep Learning **12**

4.1 Attention-Based Bi-LSTM 12

4.2 Hierarchical Attention 12

4.3 RCNN Model 13

5 Level 3: Transformers **14**

5.1 BERT Architecture 14

5.2 Self-Attention Mechanism 14

5.3 DeBERTa: Decoding-enhanced BERT 14

6 Level 4: The Forensic Judge **16**

6.1 Llama 4 Maverick (LLM) 16

6.2 Architectural Innovations 16

6.2.1 1. RMSNorm (Root Mean Square Norm) 16

6.2.2 2. SwiGLU Activation 16

6.2.3 3. Rotary Positional Embeddings (RoPE) 17

6.2.4 4. Grouped-Query Attention (GQA) 17

7 Experimental Results **18**

7.1 Confusion Matrix 18

7.2 Accuracy Comparison 18

7.3 Probability Distribution 18

7.4 Live System Output 19

8 Conclusion **19**

8.1 Summary: Aegis Secure Achievements 20

8.2 Future Directions 20

1 INTRODUCTION

1.1 The Problem Landscape

The Evolving Threat: Why Traditional Security Fails

Cybercrime is no longer manual. It is automated, AI-driven, and highly scalable.

> **THE CURRENT STATE**
>
> Cybercrime is no longer manual. It is automated, AI-driven, and highly scalable.

Key Drivers:

- **Generative AI** creates perfect lures.
- **Zero-Day** attacks bypass blacklists.
- **Speed** of attacks exceeds human response.

> **THE DEFICIT**
>
> Existing tools lack **Context** and **Explainability**. Users are alerted, but not educated.

1.2 Problem Statement

The core challenge addressed by Aegis Secure is identified as follows:

> **THE CRITICAL GAP**
>
> "How do we detect semantically novel phishing attacks that use legitimate infrastructure, while providing forensic explanations to the user?"

Key Constraints:

- Must process in < 2 seconds.
- Must explain *why* a URL is malicious.
- Must handle high throughput.

2 SYSTEM ARCHITECTURE

2.1 High-Level Architecture

The system employs a multi-modal pipeline combining Statistical ML, Deep Learning, and LLMs. Below is the left section of the architecture.

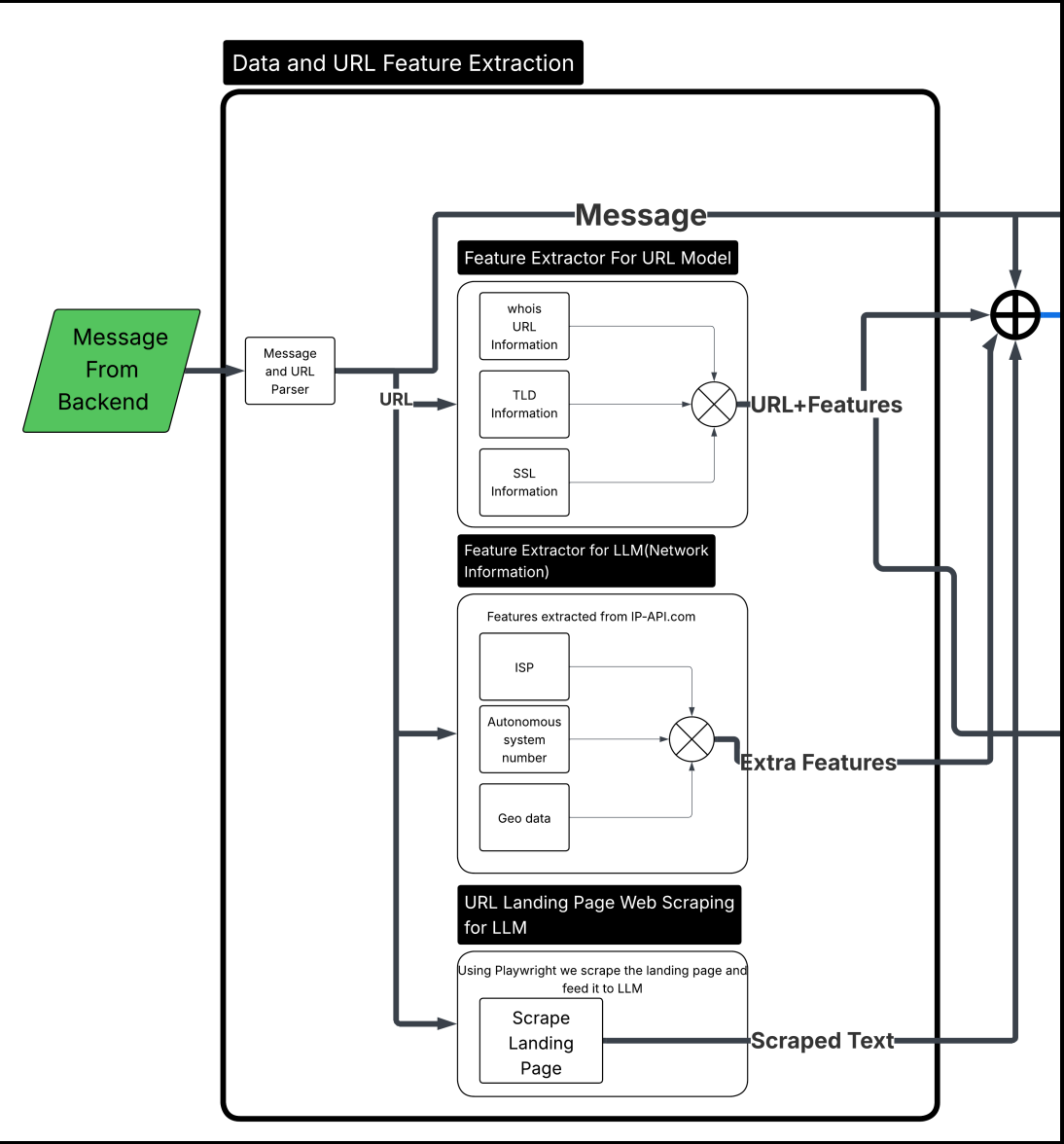

Figure 1: Main System Architecture Diagram (Part 1/2)

ble

High-Level Architecture (Continued)

Below is the right section of the architecture pipeline.

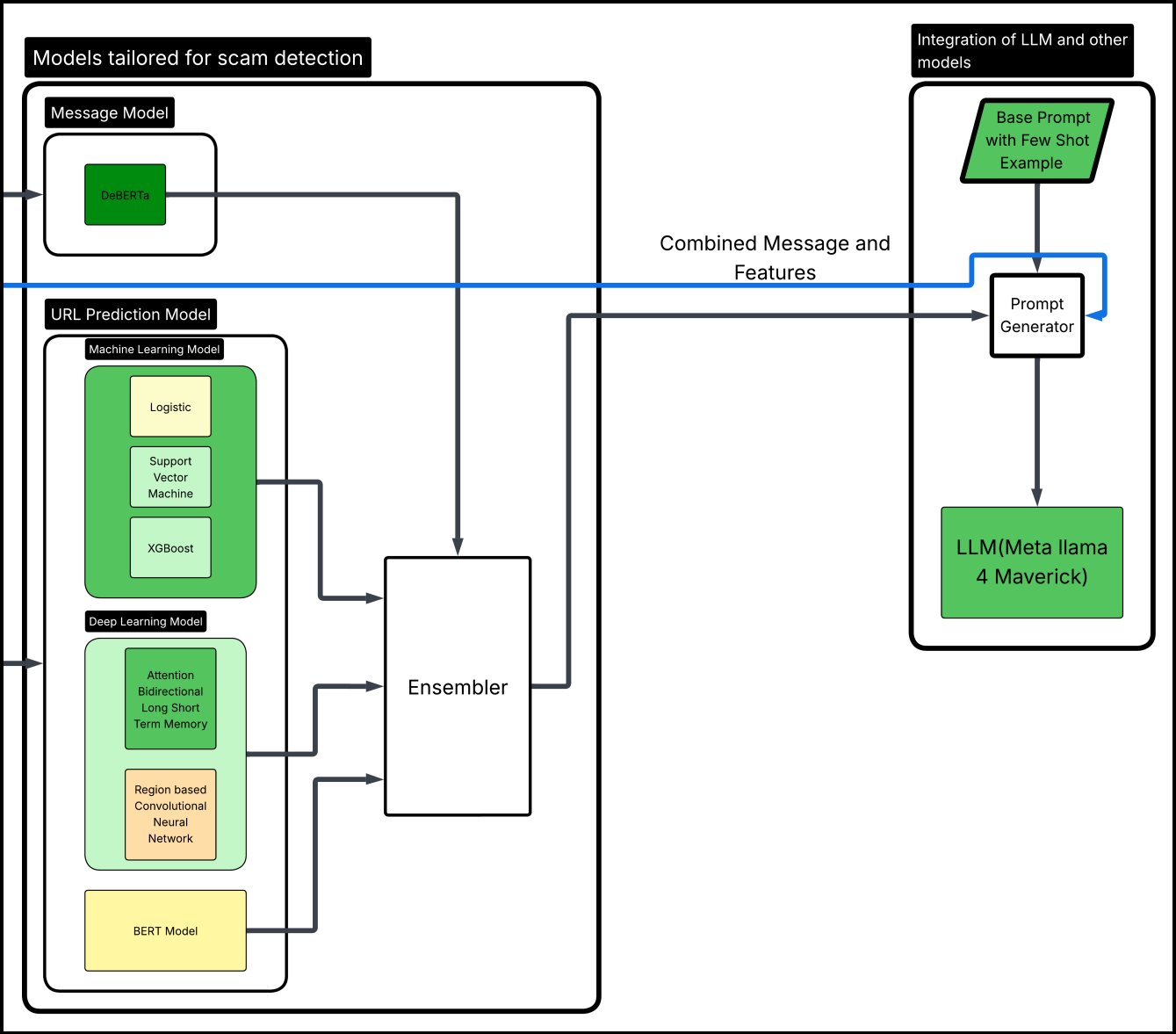

Figure 2: Main System Architecture Diagram (Part 2/2)

2.2 Activity Workflow

The asynchronous processing flow ensures non-blocking analysis.

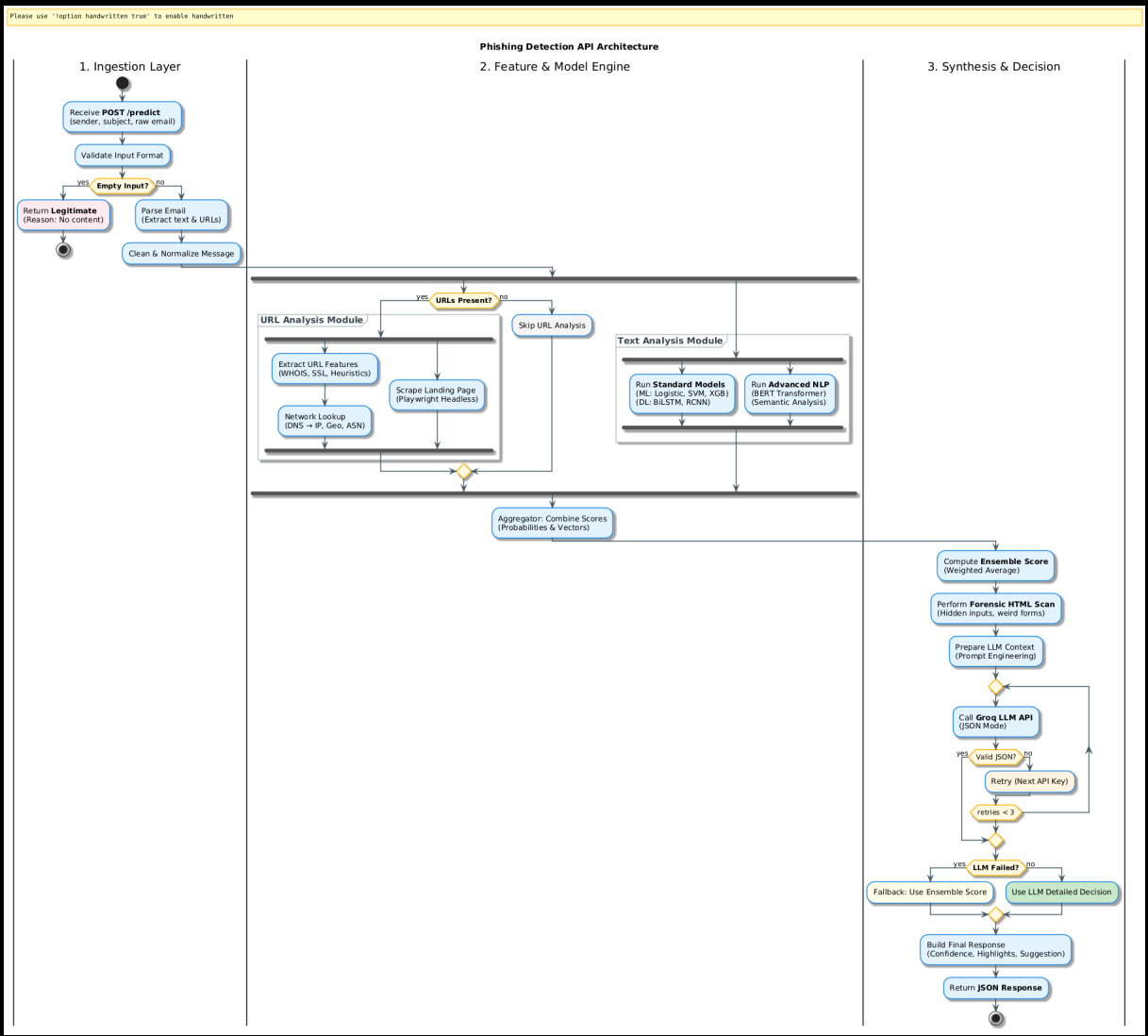

Figure 3: Activity Diagram (Async Flow)

2.3 Lifecycle Sequence

The Request/Response cycle details the interaction between the client and the analysis engine.

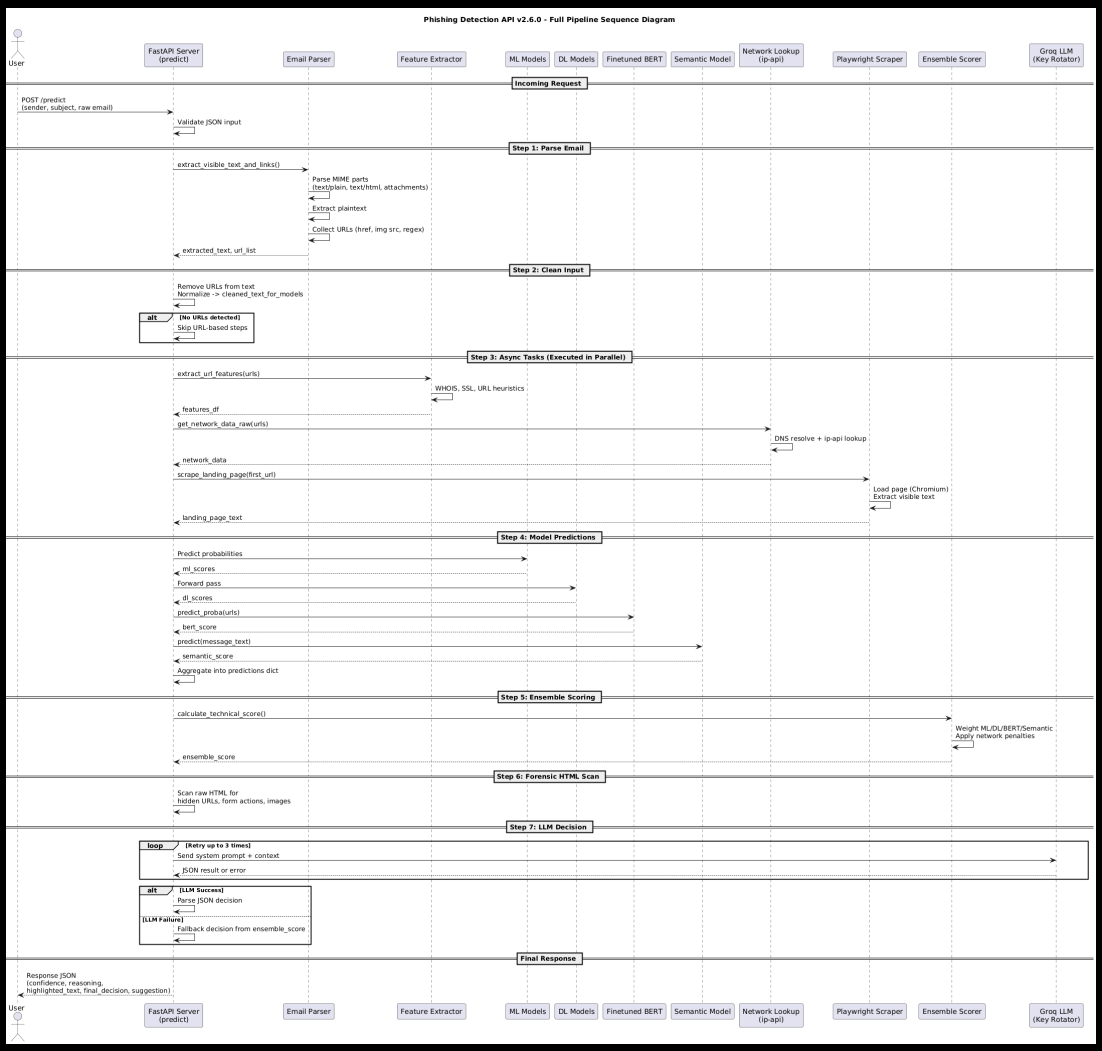

Figure 4: Sequence Diagram

2.4 Backend Modularity

The Python class structure is designed for modularity and scalability.

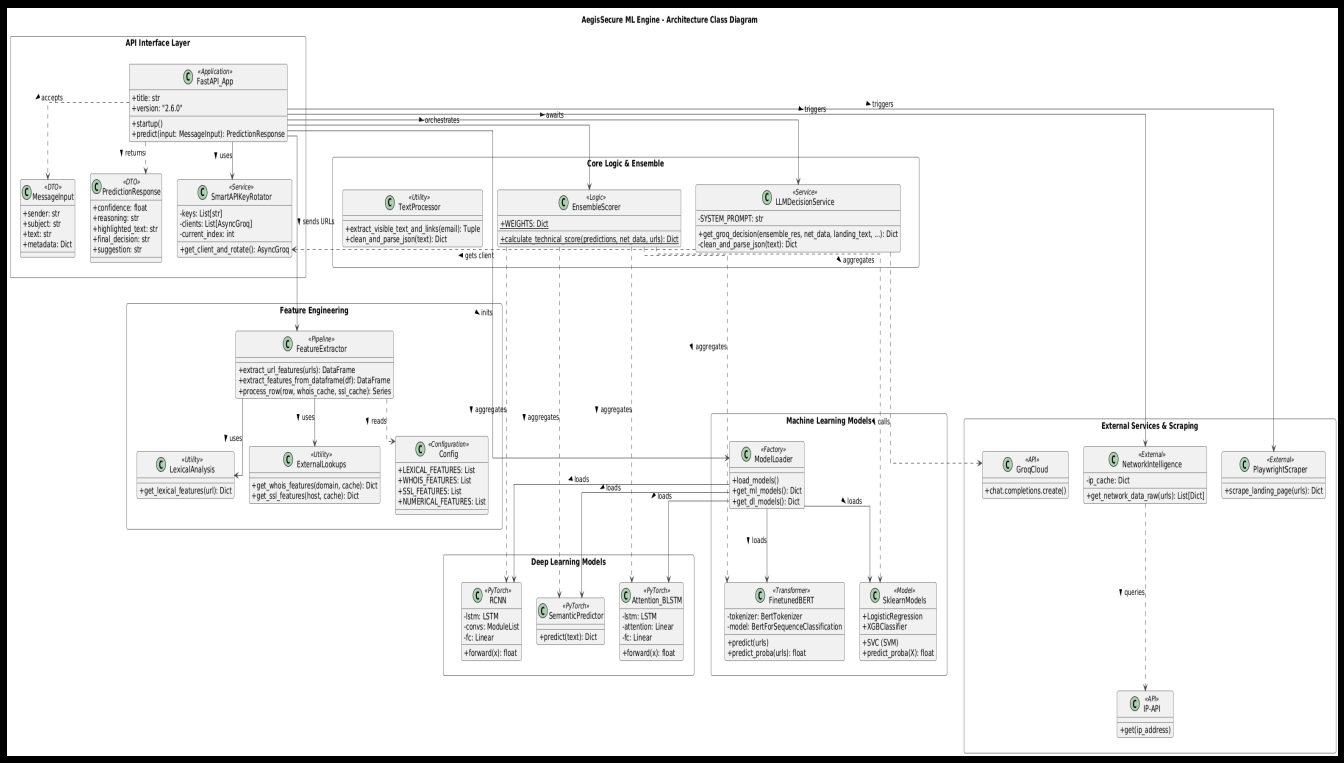

Figure 5: Class Diagram

3 LEVEL 1: STATISTICAL ML

3.1 Strategy: Speed & Feature Extraction

We begin with lightweight models to capture lexical and host-based features.

- **SVM**: Geometric separation of high-dimensional vectors.
- **Logistic Regression**: Baseline probability estimation.
- **XGBoost**: Handling non-linear relationships in tabular data.

3.2 Support Vector Machines (SVM)

Geometric Intuition: We seek a plane that maximizes the margin γ between malicious ($+1$) and safe (-1) data points.

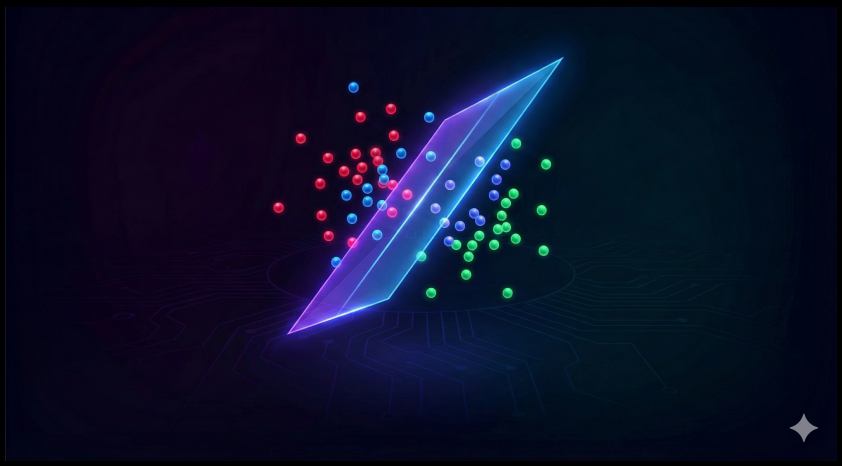

Figure 6: SVM Hyperplane

Primal Optimization: To maximize the margin $\frac{2}{||w||}$, we minimize the norm of weights w.

Primal Objective

$$\min_{w,b,\xi} \mathcal{J}(w,\xi) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{N}\xi_i$$

Subject to: $y_i(w^T\phi(x_i)+b) \geq 1-\xi_i$, where ξ_i are slack variables allowing for some misclassification (Soft Margin).

The Kernel Trick: URLs are not linearly separable in 2D. We project them into infinite dimensions using the **RBF Kernel**.

Radial Basis Function

$$K(x_i,x_j) = \exp\left(-\gamma||x_i-x_j||^2\right)$$

3.3 Logistic Regression

Logistic regression estimates the probability $P(y = 1|x)$ utilizing the sigmoid curve.

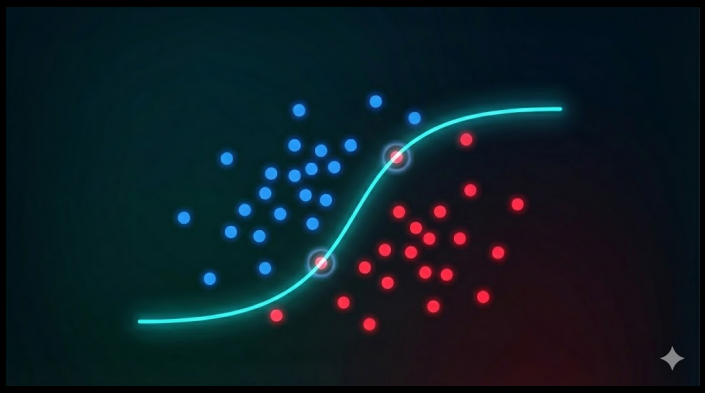

Figure 7: Sigmoid Curve

Hypothesis

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Log-Loss Minimization: We optimize parameters θ by minimizing Cross-Entropy. The λ term (Ridge) prevents overfitting.

Cost Function with Regularization

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(h_\theta) + (1 - y^{(i)})\log(1 - h_\theta)\right] + \frac{\lambda}{2m}||\theta||^2$$

3.4 XGBoost (Extreme Gradient Boosting)

An ensemble of weak decision trees f_k where $\hat{y}_i = \sum_{k=1}^{K} f_k(x_i)$.

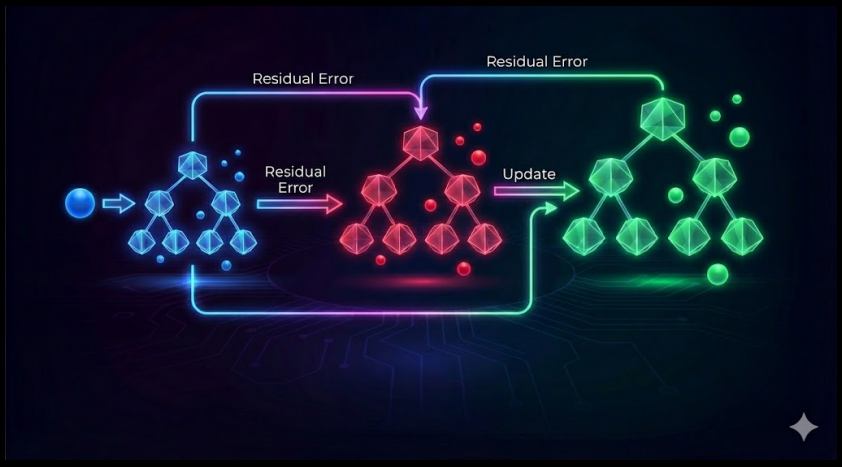

Figure 8: Decision Tree Ensemble

Regularized Objective: At step t, we add a new tree f_t to minimize the loss, where $\Omega(f_t)$ penalizes tree complexity.

Objective

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t)$$

Taylor Expansion: XGBoost uses the Second-Order Taylor Approximation.

Approximate Objective

$$\tilde{\mathcal{L}}^{(t)} \approx \sum_{i=1}^{n} [l(y_i, \hat{y}^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t)$$

4 LEVEL 2: DEEP LEARNING

4.1 Attention-Based Bi-LSTM

Standard models fail to understand the order of characters. LSTM (Long Short-Term Memory) solves this by capturing sequence information.

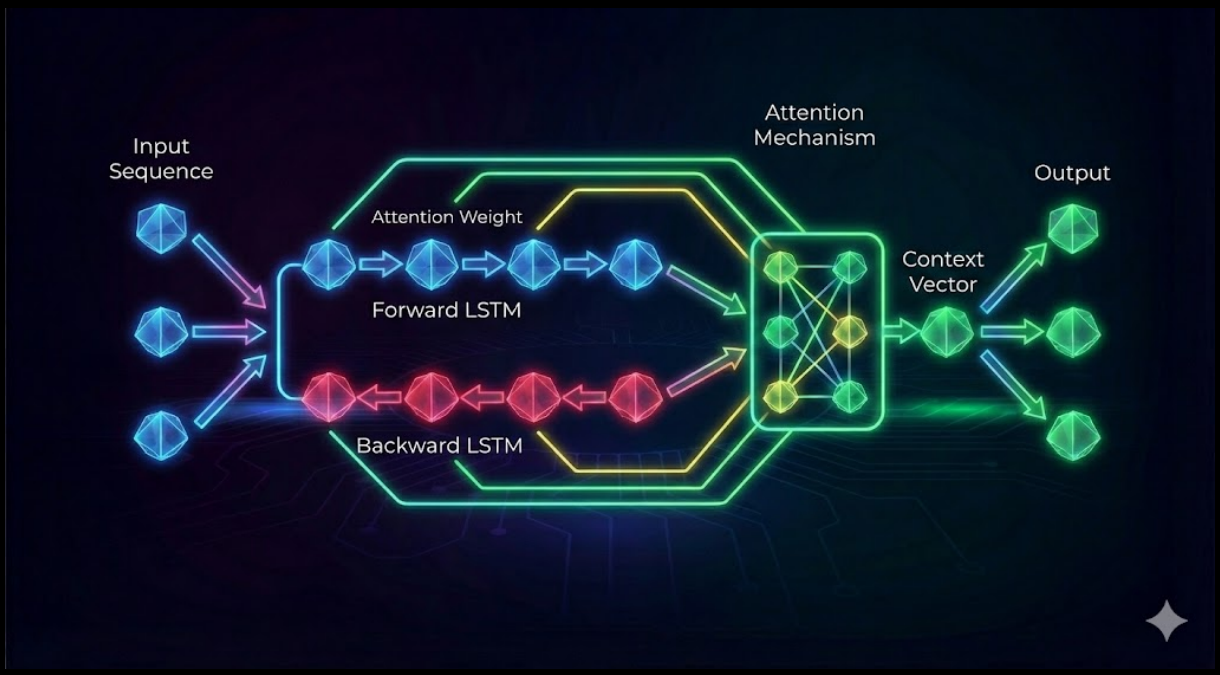

Figure 9: Bi-LSTM Architecture

LSTM Dynamics: The Gates

- **Input Gate (i_t)**: What new info to store?
- **Forget Gate (f_t)**: What info to delete?

Cell State Update

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(\text{NewInfo})$$

Hidden State Output

$$h_t = o_t \odot \tanh(c_t)$$

4.2 Hierarchical Attention

Not all parts of a URL are equal (e.g., "bank-verify" vs "index.html"). The final representation v is a weighted sum: $v = \sum \alpha_t h_t$.

Attention Weight Calculation

$$u_t = \tanh(W_w h_t + b_w)$$
$$\alpha_t = \frac{\exp(u_t^T u_w)}{\sum_t \exp(u_t^T u_w)}$$

4.3 RCNN Model

Recurrent Convolutional Neural Networks combine RNNs for context and CNNs for max-pooling significant features.

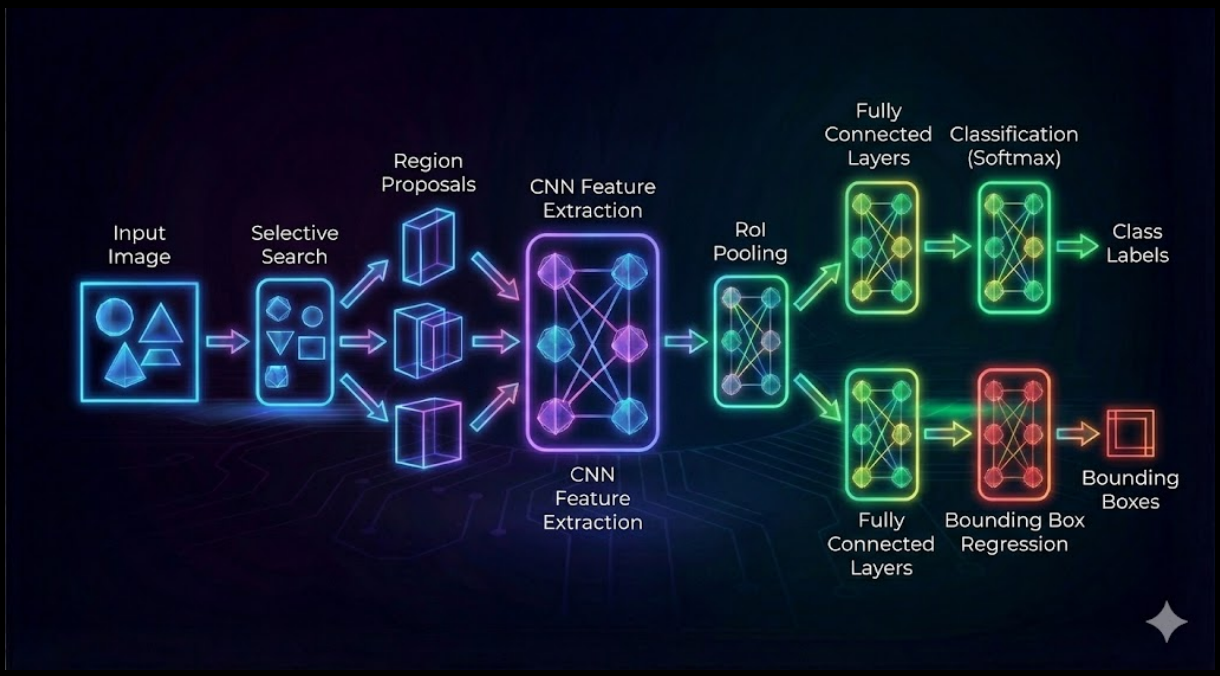

Figure 10: RCNN Diagram

5 LEVEL 3: TRANSFORMERS

5.1 BERT Architecture

Bidirectional Encoder Representations from Transformers (BERT) allows parallel processing of the entire sequence for deep semantic understanding.

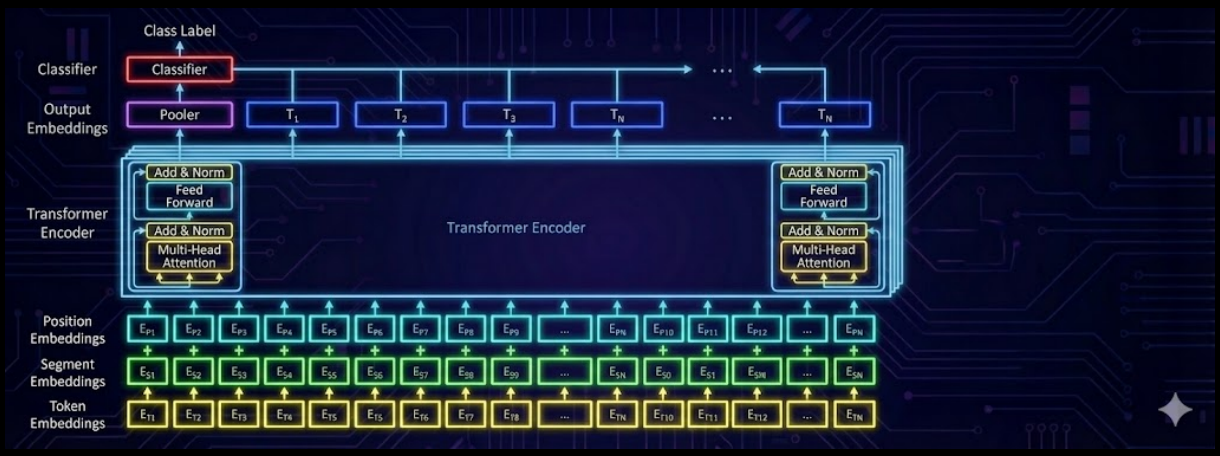

Figure 11: BERT Encoder Stack

5.2 Self-Attention Mechanism

Scaled Dot-Product Attention

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Multi-Head Attention: We run attention h times in parallel to capture different types of relationships (e.g., syntactic vs. semantic).

5.3 DeBERTa: Decoding-enhanced BERT

DeBERTa improves upon BERT by separating Content embeddings from Position embeddings.

Figure 12: DeBERTa Disentangled Attention

Disentangled Attention Math: Standard BERT adds Content + Position. DeBERTa calculates interactions separately.

Score Decomposition

$$A_{i,j} = \underbrace{H_i^T H_j}_{\text{Content-Content}} + \underbrace{H_i^T P_{j|i}}_{\text{Content-Pos}} + \underbrace{P_{i|j}^T H_j}_{\text{Pos-Content}}$$

6 LEVEL 4: THE FORENSIC JUDGE

6.1 Llama 4 Maverick (LLM)

The Concept: Explainable AI (XAI) Classifiers give a score (e.g., 0.98), but humans need a reason (e.g., "This URL mimics PayPal"). Llama 4 acts as the forensic judge.

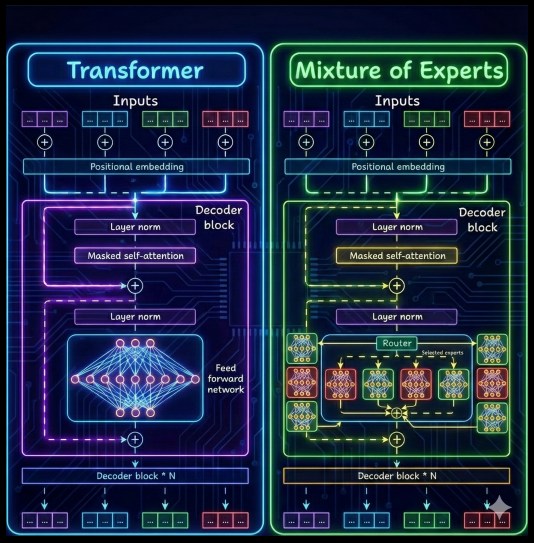

Figure 13: Llama Model Architecture

6.2 Architectural Innovations

Llama 4 utilizes three key mathematical modifications for stability and speed:

1. **RMSNorm** (Normalization)
2. **SwiGLU** (Activation)
3. **RoPE** (Positional Encoding)

6.2.1 1. RMSNorm (Root Mean Square Norm)

LayerNorm subtracts mean and divides by variance. RMSNorm only scales invariance.

RMSNorm Formula

$$\bar{a}_i = \frac{a_i}{\text{RMS}(a)} g_i, \quad \text{where RMS}(a) = \sqrt{\frac{1}{n}\sum a_i^2}$$

6.2.2 2. SwiGLU Activation

Replaces the standard ReLU in the Feed Forward Network (FFN). This gating mechanism controls information flow more fluidly.

SwiGLU FFN

$$\text{FFN}(x) = (\text{Swish}_\beta(xW) \otimes (xV))W_2$$

Where $\text{Swish}(z) = z \cdot \sigma(\beta z)$.

6.2.3 3. Rotary Positional Embeddings (RoPE)

Instead of adding a static position vector, we **rotate** the embedding vector in the complex plane.

> **THE INTUITION**
>
> Absolute position doesn't matter. Relative distance between tokens matters. Rotation encodes relative distance naturally.

For a token at position m, we apply a rotation matrix Θ. This ensures that the inner product $q_m^T k_n$ depends only on $m - n$.

Rotation Operation

$$\begin{pmatrix} q_0 \\ q_1 \end{pmatrix} \rightarrow \begin{pmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{pmatrix} \begin{pmatrix} q_0 \\ q_1 \end{pmatrix}$$

6.2.4 4. Grouped-Query Attention (GQA)

To run a 17B model fast, we reduce the memory bandwidth of the KV-Cache.

$$\text{KV-Heads} = \frac{H_Q}{G}$$

7 EXPERIMENTAL RESULTS

7.1 Confusion Matrix

The system focuses on minimizing False Negatives.

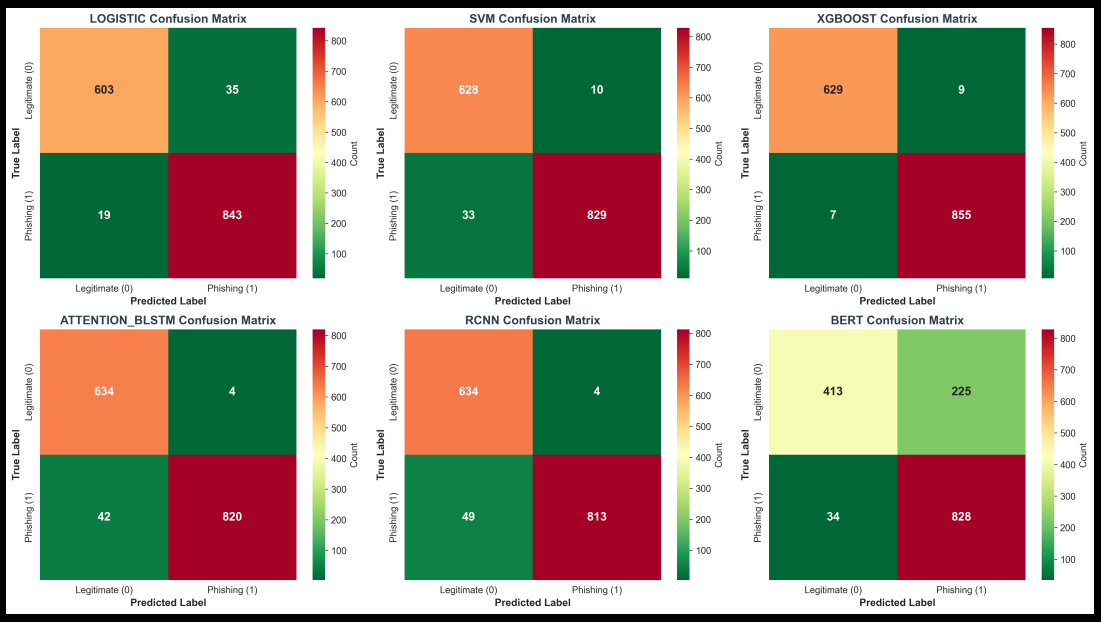

Figure 14: Confusion Matrix

7.2 Accuracy Comparison

Result: The Ensemble achieves 98.56% accuracy, outperforming isolated models.

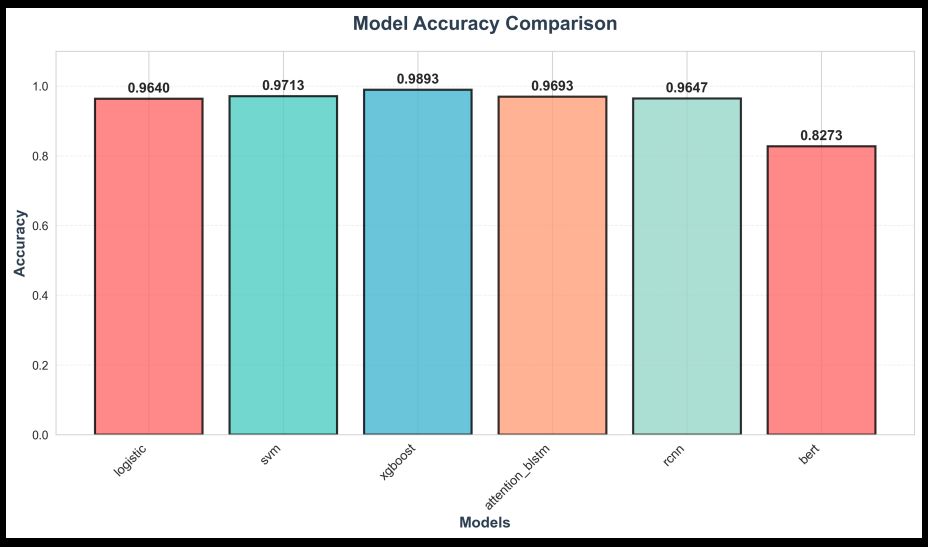

Figure 15: Bar Chart Accuracy

7.3 Probability Distribution

Visualizing the separation between benign and malicious entities.

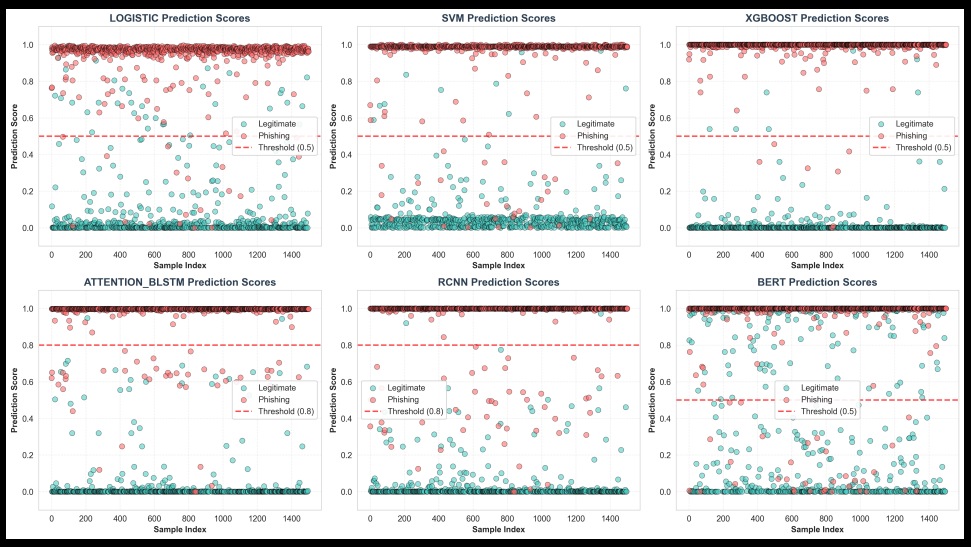

Figure 16: Scatter Plot

7.4 Live System Output

Below represents a sample JSON response from the API.

```
NEW REQUEST: LinkedIn Job Alerts <jobalerts-noreply@linkedin.com>
Parsing Email MIME Structure
Parsed Content. Extracted 6 unique URLs.
Proceeding with 6 URLs
Initiating Parallel Async Tasks
Extracting Features for 6 URLs
Geo-Locating Hosts: ['www.linkedin.com']
Live Playwright Scraping: https://www.linkedin.com/comm/jobs/view/...
huggingface/tokenizers: The current process just got forked, after parallelism has already been used. Disabling parallelism to avoid deadlocks...
To disable this warning, you can either:
    - Avoid using `tokenizers` before the fork if possible
    - Explicitly set the environment variable TOKENIZERS_PARALLELISM=(true | false)
    ├─ Resolved www.linkedin.com: Cloudflare, Inc. [Canada]
Scraped 593 chars successfully.
    ├─ Feature Extraction: Complete
Running Machine Learning Inference
    ├─ ML: logistic   : 0.9520
    ├─ ML: svm        : 0.0750
    ├─ ML: xgboost    : 0.1290
    ├─ DL: attention_blstm: 0.0074
    ├─ DL: rcnn       : 0.0000
Running Semantic Text Analysis
    ├─ Semantic Prob: 0.9990
Calculating Ensemble Weights
    ├─ ML Models Consensus: 0.3854 (Weight: 0.3)
    ├─ Deep Learning Consensus: 0.0037 (Weight: 0.2)
    ├─ Semantic Analysis: 0.9990 (Weight: 0.1)
    ├─ Network Risk Calculated: 40.00 (Weight: 0.2)
    ├─ Ensemble Technical Score: 37.03/100
Starting Forensic HTML Scan
    ├─ Forensic Scan: Found 1 potential indicators
Sending LLM Request (Attempt 1/3)
    ├─ LLM Response Received: Length: 2604 chars
LLM Response Parsed Successfully

REQUEST COMPLETE

    Execution Time: 13.95s
    Technical Score: 37
    FINAL VERDICT: LEGITIMATE
INFO:     10.16.18.114:38731 - "POST /predict HTTP/1.1" 200 OK
INFO:     10.16.8.187:14605 - "GET /robots.txt HTTP/1.1" 404 Not Found
```

Figure 17: JSON Response from API

8 CONCLUSION

8.1 Summary: Aegis Secure Achievements

SUCCESS METRICS

- **High Accuracy**: Hybrid Ensemble reduces error rates.
- **Deep Understanding**: Transformers capture semantic lures.
- **Human-Centric**: Llama 4 provides actionable forensic reports.

8.2 Future Directions

1. **Browser Extension**: Real-time client-side blocking.
2. **Quantization**: Running Llama on edge devices (phones).
3. **Few-Shot Learning**: Adapting to new campaigns instantly.

APPENDIX: TECH STACK

- **Backend**: FastAPI, Python 3.10
- **ML Ops**: Docker, Kubernetes
- **Training**: v-5 TPU (Google-Colab) , NVIDIA RTX 4060