Phishing Email Detection Using Machine Learning

DSA4263 Fraud Detection Project Report

Group Members:

Bai Yongrui	Matric Number: A0258756L
Lei Zhenyu	Matric Number: A0252857U
Lyu Wenli	Matric Number: A0258648L
Wen Youdi	Matric Number: A0259012M
Wen Jing	Matric Number: A0259135B
Wu Haoran	Matric Number: A0265857L

DSA4263 Sense-making Case Analysis: Business and Commerce [2510]

National Univeristy of Singapore

November 21, 2025

Contents

1 Abstract

Phishing emails remain a critical entry point for cyberattacks, enabling account takeover, financial fraud, and ransomware deployment. This project develops a machine learning-based detection system using 49,860 emails from four established corpora (CEAS 2008, Nazario, Nigerian Fraud, SpamAssassin).

We implement a comprehensive feature engineering pipeline extracting three signal types: (1) network features capturing sender isolation (395× lower clustering) and coordinated campaigns (19.1% spam-to-spam connections), (2) temporal features identifying off-hours exploitation (67% phishing rate at 22:00 vs. 37% at 08:00), and (3) text features detecting urgency vocabulary and domain impersonation. All features use point-in-time methodology to prevent temporal leakage.

Systematic ablation studies identify domain reputation as the highest-impact feature group (+5.85% F1-score). We evaluate seven classifiers with hyperparameter tuning via randomized search and 3-fold cross-validation. XGBoost achieves optimal performance with 99.31% CV F1-score and 99.25% test F1-score, outperforming fine-tuned BERT (98.25%).

The deployment architecture integrates real-time inference (<100ms latency) with batch graph updates and adaptive reputation scoring, providing robust, interpretable phishing detection applicable across financial services, healthcare, e-commerce, and enterprise environments.

2 Introduction

2.1 Dataset Overview

Phishing detection is a critical front in modern fraud analytics because email remains a primary entry point for cyberattacks. To study this problem with realism and breadth, this project uses a consolidated dataset of labeled phishing and legitimate emails sourced from the **CEAS** [5], **Nazario** [9], **Nigerian Fraud** [10], and **SpamAssassin** [3] corpora. These datasets include sender–recipient information, timestamps, subject lines, message bodies, and binary phishing labels.

2.2 Industry Impact

Phishing is not merely an isolated cyber security issue; it is a critical entry point for a wide spectrum of downstream fraud. Once attackers obtain user credentials or gain initial system access, a wide range of fraudulent activities can follow, including account takeover, unauthorized financial transactions, malware and ransomware installation, and impersonation of executives or trusted partners.

Although this project focuses specifically on email phishing, the techniques developed—text mining, anomaly detection, and network-based feature engineering—are broadly applicable to fraud detection across multiple industries. The behavioural and relational patterns observed in phishing attempts mirror those found in:

- **Social media platforms:** Detection of fake accounts, botnets, and coordinated malicious campaigns.
- **Financial institutions:** Identification of abnormal transaction patterns and account takeover.
- **Payment processors and e-commerce:** Detection of fraudulent purchases, chargeback abuse, and fake merchants.
- **Telecommunications and messaging:** Spotting SIM-swap attacks, spam calls, and large-scale scam campaigns.

These sectors all face large-scale, evolving adversaries and require models that are not only accurate but also interpretable and deployable in real-time systems. As such, this project develops machine learning models enhanced with **Explainable AI (XAI)** to detect phishing. Such models can be integrated into email security pipelines as an early-warning layer, providing practical, front-line defense against fraud that originates from phishing campaigns.

3 Data

3.1 Data Collection and Validation

Source Composition and Provenance

The dataset integrates emails from four established corpora, each with documented provenance and unique characteristics. Table 1 presents the detailed breakdown of email counts by source, classification, and geographic origin.

Table 1: Dataset Composition by Source Corpus with Provenance

Corpus	Total	Phishing	Legitimate	Country/Region	Collection Period
CEAS 2008 [5]	39,154	21,842	17,312	Global	Aug 2008
Nazario Phishing Corpus [9]	1,565	1,565	0	Global	2004–2007
Nigerian Fraud Collection [10]	3,332	3,332	0	Global	1998+
SpamAssassin Public Corpus [3]	5,809	1,718	4,091	Global	2002–2006
Total	**49,860**	**28,457**	**21,403**	**Global**	**1998–2008**

Dataset Provenance Summary

Our dataset aggregates four established email corpora that collectively span a decade of phishing evolution. The CEAS 2008 corpus [5] originates from the CEAS Spam Filter Challenge, where a 72-hour live email stream was recorded and later replayed as a laboratory evaluation corpus. The Nazario Phishing Corpus [9] is a curated collection of phishing emails distributed as downloadable mbox files, collected between November 2004 and August 2007 by Jose Nazario at Arbor Networks. The Nigerian Fraud (CLAIR) collection [10] comprises advance-fee ("419") scam emails assembled by Dragomir Radev at the University of Michigan for research on fraudulent messaging. The SpamAssassin Public Corpus [3] consists of widely used spam/ham bundles compiled from public sources and community submissions for email filtering research, maintained by the Apache Software Foundation.

Integration and Quality Assurance

We harmonized schema across sources to seven common fields: sender, receiver, date, subject, body, label, and URLs. Labels rely on the original corpora's validation procedures, including CEAS challenge annotations, SpamAssassin community verification, and PhishTank-style reporting pipelines.

Class Balance

The dataset exhibits a modest class imbalance. Phishing emails account for 57.07% (28,457 out of 49,860) while legitimate emails comprise 42.93% (21,403 out of 49,860), resulting in a 1.33:1 phishing-to-legitimate ratio. This manageable imbalance facilitates binary classification without requiring extensive resampling techniques.

Missingness Analysis

Table 2: Summary of Missing Values Across Key Features

Feature	Missing (%)	Completeness (%)
Sender	0.7	99.3
Receiver	4.2	95.8
Date	1.0	99.0
Subject	0.2	99.8
Body	0.0	100.0

The complete availability of email body text, which serves as the primary feature for classification, is particularly advantageous for model training and evaluation.

Bias Considerations

Potential biases inherent to the dataset include:

- **Temporal bias:** Although emails span 1998–2008, the dataset is dominated by CEAS 2008, so models can learn period-specific artifacts rather than persistent patterns, limiting generalization to later years.
- **Linguistic bias:** The dataset predominantly contains English-language emails, which may limit the generalizability of trained models to non-English phishing attempts. This limitation is explicitly noted when discussing model deployment and transfer learning applications.

These factors are carefully considered during model evaluation, and generalization limits are explicitly documented in our results and discussion sections.

3.2 Dataset Size and Structure

The final dataset contains 49,860 emails in tabular form with 8 columns: sender, receiver, date, subject, body, label, URLs, and source. The primary predictive features are the body (full message text) and subject line, while the label field provides ground truth classification (1 = phishing, 0 = legitimate).

4 Methods

4.1 Data Cleaning and Pre-processing

Our data cleaning pipeline implements a three-tier approach using the `DataCleaner` class to maximize data retention while ensuring quality.

4.1.1 Date Parsing and Timezone Handling

The `clean_dates()` method employs a three-tier parsing strategy:

1. **RFC-5322 Format:** Attempts strict parsing with regex extraction of timezone offsets (e.g., +0800)

2. **Pandas Auto-detection:** Falls back to `pd.to_datetime()` for non-standard formats
3. **Timezone Stripping:** Preserves timestamps with missing timezones as `Unknown` rather than dropping records

Geographic Region Mapping:
Extracted timezone offsets are mapped to six regions:

- Americas: UTC-12 to UTC-4
- Europe/Africa: UTC-4 to UTC+2
- Middle East/South Asia: UTC+2 to UTC+6
- APAC: UTC+6 to UTC+10
- Oceania/Pacific: UTC+10 to UTC+14

Dates outside the range 1990–2025 are filtered as anomalies.

Results: 99.0% data retention (49,340/49,860 emails), with 97.4% having valid timezone extraction.

4.1.2 Text Cleaning Pipeline

The `clean_text()` method implements a six-stage process:

1. **Null Handling:** Convert empty/null values to empty strings
2. **Lowercase Conversion:** Standardize case for consistent matching
3. **Character Filtering:** Remove punctuation and numbers using regex `r'[^a-zA-Z\s]'`
4. **Tokenization:** Split text using NLTK's `word_tokenize()`
5. **Stopword Removal:** Filter English stopwords plus custom terms (`submission`, `cnncom`, `www`) and tokens < 3 characters
6. **Lemmatization:** Reduce words to base forms (e.g., *running* $\rightarrow$ *run*)

The `clean_text_column()` method provides batch processing with optional progress tracking.

4.1.3 Missing Value Handling

Our approach preserves maximum data while maintaining feature integrity:

- **Receiver field:** Missing values assigned unique labels (`na1`, `na2`, ...) to preserve graph structure
- **Sender field:** Rows with missing senders dropped (0.7% of data)
- **Text fields:** Filled with empty strings to enable feature computation

Final retention: 98.2% of original data (48,987/49,860 emails) retained through the complete pipeline.

4.2 Exploratory Data Analysis

4.2.1 Network Analysis of Phishing Emails

Table 3: Hypothesis Testing Results - All Supported

Hypothesis	Evidence	Status
H1: One-way communication pattern	0.22× reciprocity, 99.2% no-reply	Supported
H2: Network isolation	395× less clustering	Supported
H3: Coordinated campaigns	19.1% spam-to-spam edges	Supported

Methodology: We constructed a directed communication graph where nodes represent email addresses and edges represent sender–receiver relationships. Network metrics (in-degree, out-degree, reciprocity, clustering coefficient, closeness centrality, and triangle participation) were calculated to characterize communication patterns.

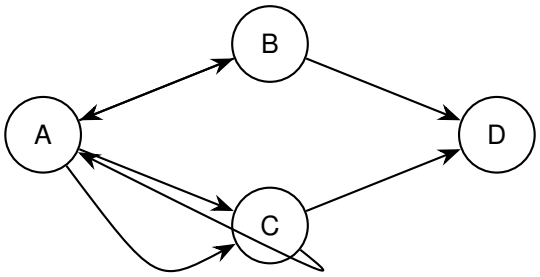


Figure 1: Illustration of the directed communication graph

Nodes represent email accounts; arrows represent sent messages. The structure enables computation of network metrics such as degree, reciprocity, clustering, and centrality.

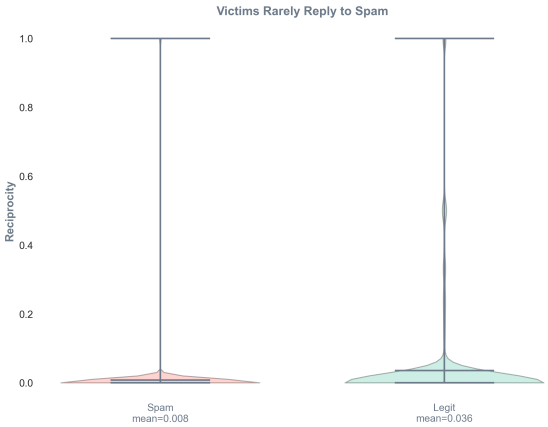
Dataset Bias Note: Our dataset represents a constrained snapshot that may not generalize to real-world email environments. Normal users may have much higher in-degree in production systems, where they receive emails from colleagues, automated systems, newsletters, and social contacts. The network metrics reported here should be interpreted within the context of this dataset's limitations and validated against operational email traffic before deployment.

Findings and Hypothesis Testing: H1 - One-Way Communication SUPPORTED

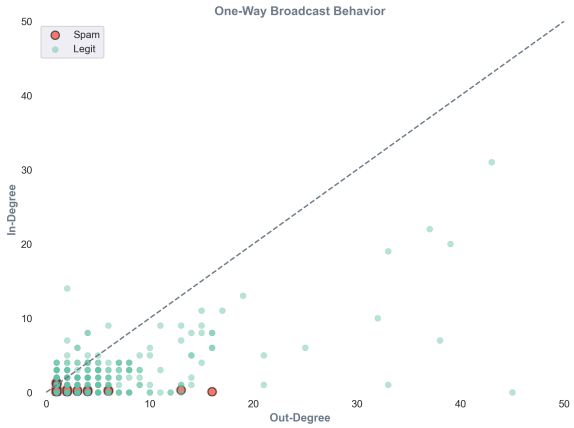
Spam accounts exhibit drastically lower reciprocity (0.008 vs. 0.036, a 0.22× ratio), indicating that over 99% of spam messages receive no reply. In the visualization, this appears as a tight horizontal band of spam users near zero reciprocity, contrasted with the broader spread of legitimate accounts across higher values. Together with the out-degree distribution, this forms a clear structural "spam signature": low reciprocity, low in-degree, and minimal clustering, creating natural separation boundaries around out-degree ≈ 5 and reciprocity ≈ 0.2.

Quantitative Evidence:

- **Reciprocity Ratio:** Spam 0.008 vs. Legitimate 0.036 (0.22× lower)
- **No-Reply Rate:** 99.2% of spam messages receive no response



(a) Reciprocity comparison



(b) Degree scatter comparison

Figure 2: Side-by-side network feature comparisons

- **Statistical Significance:** Clear separation in distribution patterns

H2 - Network Isolation: SUPPORTED

Spam accounts are 395× less clustered than legitimate users, indicating complete isolation from social networks. Combined with 3.65× lower in-degree, this confirms spam operates at network periphery without social integration. The low closeness centrality shows spammers are distant from the network core, requiring more steps to reach other users.

Table 4: Network Features: Spam vs. Legitimate Users

Feature	Spam Mean	Legit Mean	Legit / Spam
Out-degree (sent emails)	1.04	1.99	1.91×
In-degree (received emails)	0.27	0.99	3.65×
Reciprocity (reply rate)	0.008	0.036	4.43×
Clustering (social ties)	0.00016	0.06327	395×
Eigenvector centrality	0.00000	0.00167	—
Closeness centrality	0.000007	0.000694	99.1×

Quantitative Evidence:

- **Clustering Coefficient:** 395× lower for spam accounts
- **In-Degree:** 3.65× lower for spam accounts
- **Closeness Centrality:** 99.1× lower for spam accounts

H3 - Coordinated Campaigns: SUPPORTED

Despite individual isolation (H2), 19.1% of spam emails target other spam accounts, revealing coordinated campaigns rather than isolated attackers. This suggests infrastructure for sharing victim lists or coordinating attacks. The asymmetric flow (0.3% legitimate-to-spam vs. 55.6% spam-to-legitimate) confirms unidirectional attack patterns. The network visualization shows spammers as broadcasting hubs with multiple outbound connections to isolated recipients.

Table 5: Email Flow by Sender-Receiver Type

Connection Type	Count	Percentage
Spam → Spam	7,107	19.1%
Spam → Legitimate	20,667	55.6%
Legitimate → Spam	100	0.3%
Legitimate → Legitimate	9,297	25.0%

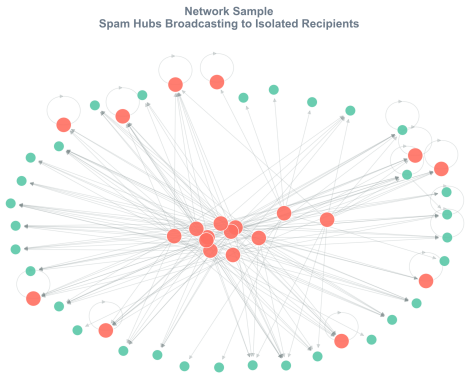


Figure 3: Network visualization showing spam hub structure. Spam accounts (red) broadcast to isolated recipients (green)

Quantitative Evidence:

- **Spam-to-Spam Connections:** 19.1% of all spam emails
- **Asymmetric Flow:** 0.3% legitimate-to-spam vs. 55.6% spam-to-legitimate
- **Statistical Significance:** Clear evidence of coordinated infrastructure

4.2.2 Text-Based Content Analysis

Hypothesis	Evidence	Status
H1: Phishing emails are shorter and more direct	3.48× shorter bodies	Supported
H2: Phishing uses brand domains as credibility bait	cnn.com, yahoo.com dominance	Supported
H3: Phishing employs urgency/reward vocabulary	money, account, click, receive	Supported

Table 6: Hypothesis Testing Results - All Supported

Methodology: Text preprocessing standardized diverse email formats to produce clean, comparable text for analysis and modeling. URLs were extracted from the `subject` and `body` using a pattern-based matcher, and normalized domains were stored in `url_domains_subject` and `url_domains_body`. The cleaned text versions without URLs were saved as `subject_no_urls` and `body_no_urls`, enabling separate analysis of url presence and language style.

Missing `subject` or `body` values were replaced with empty strings, while extra spaces, line breaks, HTML tags, and entities were removed to produce plain, uniform text. After preprocessing, each record contained standardized fields for systematic analysis.

Findings and Hypothesis Testing: H1 - Length and Complexity: SUPPORTED

Statistical summaries reveal substantial differences in length and verbosity between phishing and non-phishing emails. Legitimate messages are typically longer and more descriptive, while phishing emails are shorter and more direct.

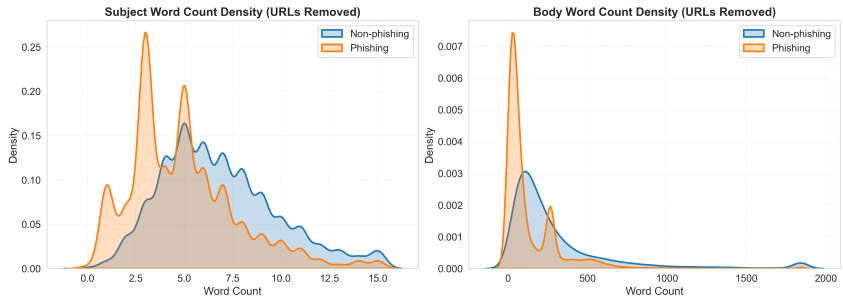


Figure 4: Length Density Distributions: Subject and Body Comparison

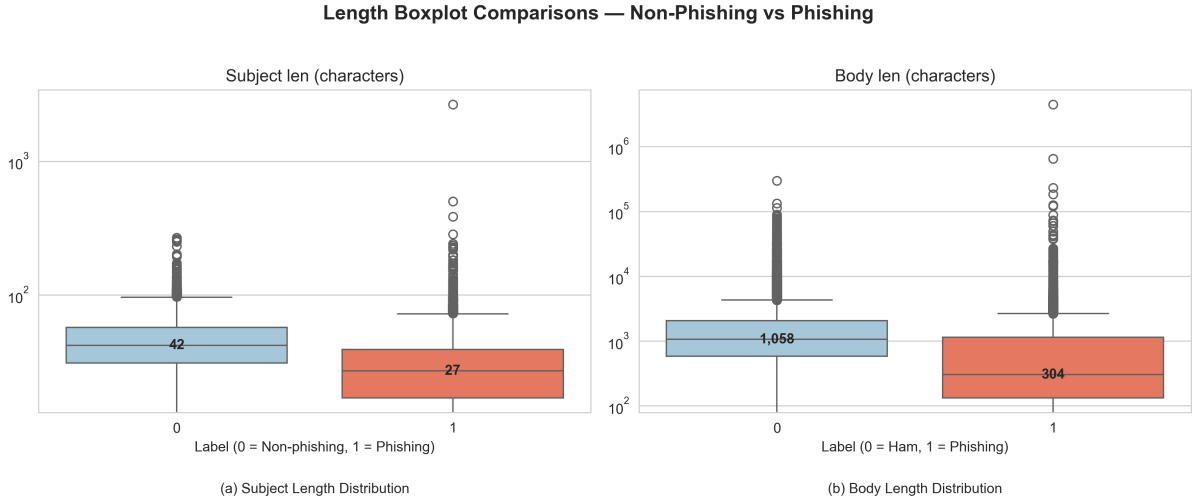


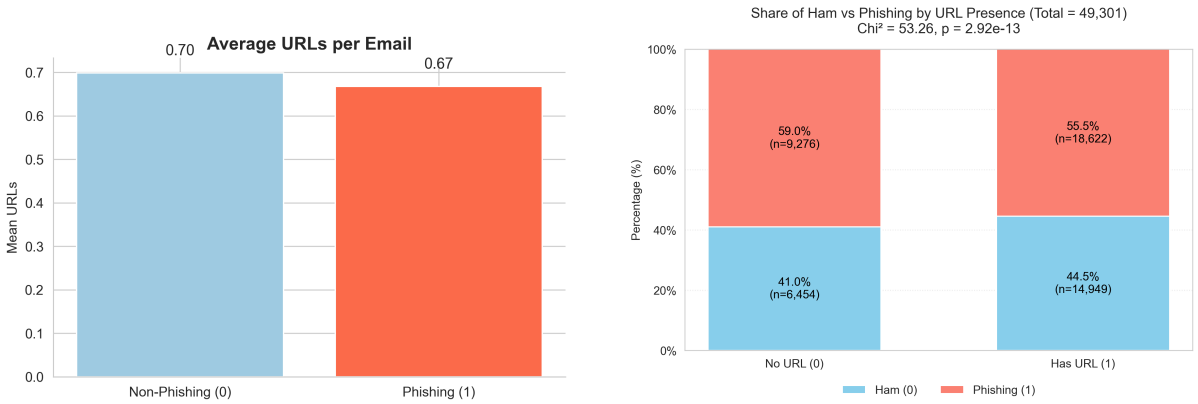
Figure 5: Length Boxplot Comparisons: Non-Phishing vs. Phishing

Quantitative Evidence:

- **Subject Length:** Non-phishing median = 43 characters, Phishing median = 27 characters (1.59× longer)
- **Body Length:** Non-phishing median = 1,196 characters, Phishing median = 344 characters (3.48× longer)
- **Statistical Significance:** Clear separation in distribution patterns

H2 - URL Deception: SUPPORTED

URL pattern analysis reveals that link presence alone is not discriminatory—URLs are common in both non-phishing (70%) and phishing (66%) messages. However, 59.8% of emails without URLs are phishing versus 55.8% with URLs, confirming that link presence alone is not a strong discriminator. The critical distinction lies in domain semantics rather than URL quantity.



(a) Average URLs per Email

(b) Share by URL Presence

Figure 6: URL Distribution Patterns: Non-phishing vs. Phishing

Domain signals tell a consistent story across both the frequency charts and word clouds. Non-phishing emails predominantly link to technical/institutional sites (e.g., `astrology.com`, `python.org`), reflecting genuine reference sharing, partly due to CEAS/SpamAssassin mailing-list content. Phishing, in contrast, concentrates on media/brand domains (e.g., `cnn.com`, `yahoo.com`), repeated as credibility bait rather than true destinations, suggesting template-driven campaigns.

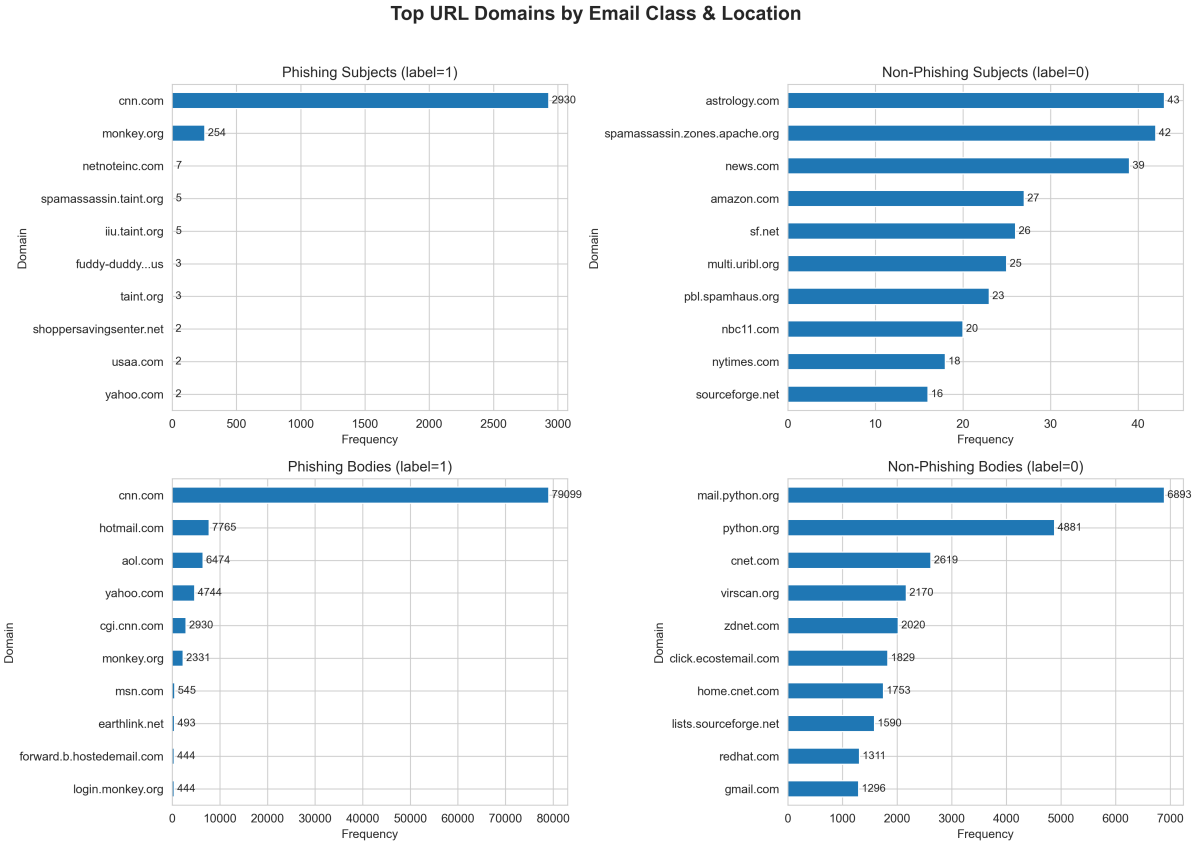


Figure 7: Top URL Domains in Email Subjects and Bodies: Non-phishing vs. Phishing

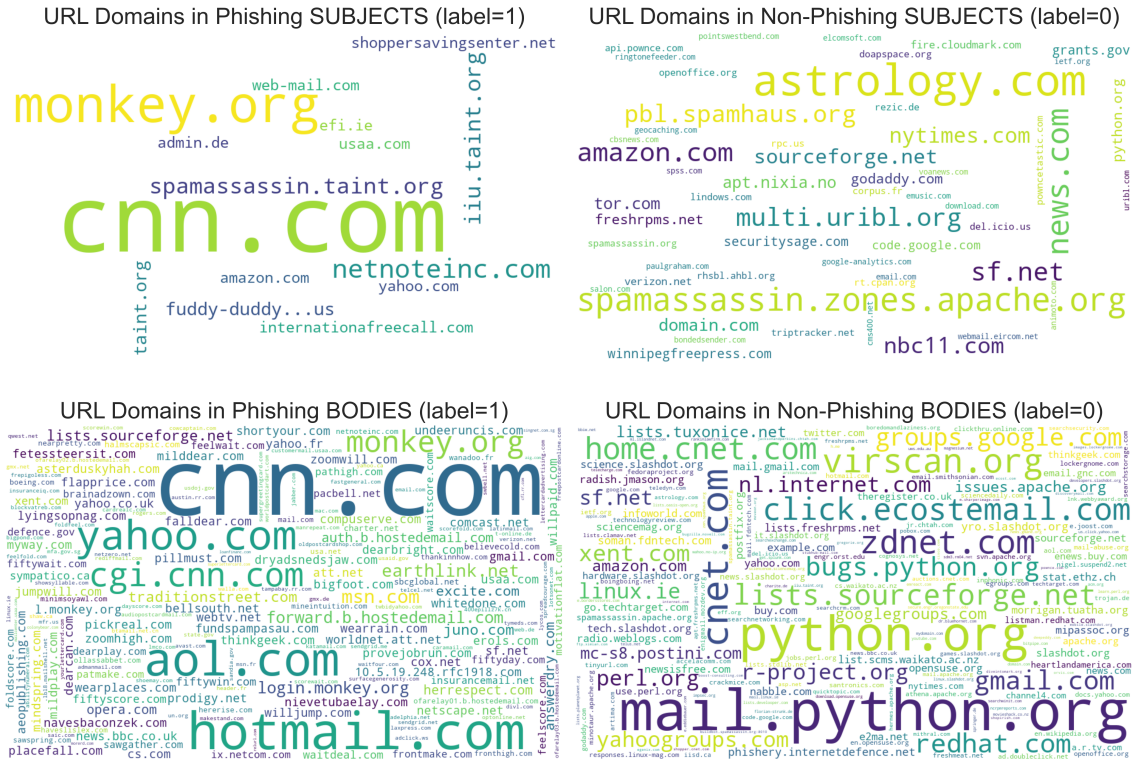


Figure 8: Domain Word Distribution Clouds: Non-phishing vs. Phishing

Quantitative Evidence:

- **URL Presence:** Non-phishing 70%, Phishing 66% (not discriminatory)
- **Domain Semantics:** Non-phishing = technical/institutional; Phishing = media/brand
- **Key Finding:** Domain type matters more than URL presence

H3 - Linguistic Manipulation: SUPPORTED

An initial pass exposed CEAS/SpamAssassin wrapper artifacts in non-phishing (e.g., *Submission*, *ID*, *Sender*) [4], which can cause leakage. We mitigated this by (i) expanding dataset-specific stopwords (wrappers and campaign tokens like *cncom/cnn/cable news*), and (ii) lowercasing and removing punctuation/digits. After cleaning, clouds reflect true content.

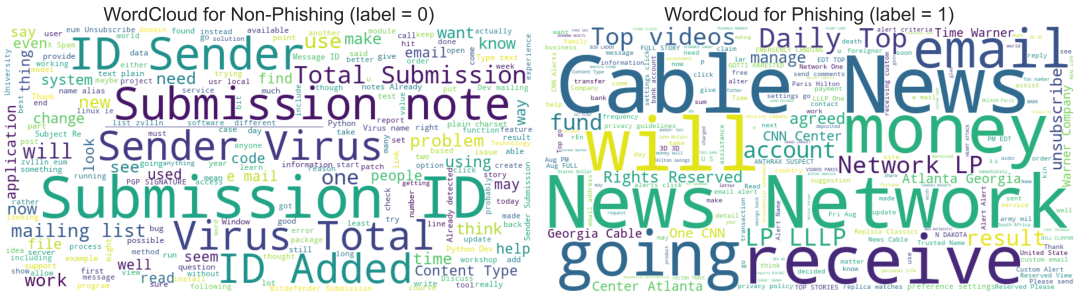


Figure 9: Vocabulary Before Cleaning: Wrapper Artifacts Visible (Submission, ID, Sender, Virus)

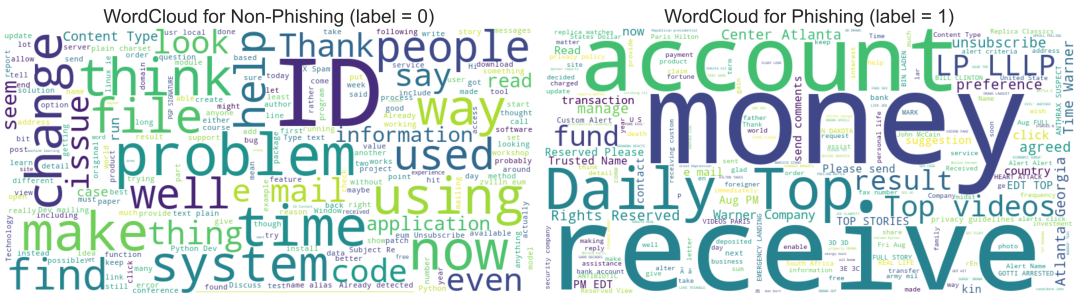


Figure 10: Vocabulary After Cleaning: True Content Revealed

Quantitative Evidence:

- **Phishing Vocabulary:** Highly overlapping, urgency/reward related (account, click, update, login, money, receive)
- **Non-Phishing Vocabulary:** Topic-specific, informational language
- **Statistical Significance:** Clear lexical separation after artifact removal

Supporting Evidence: Sender Domain and Stylistic Patterns

The sender domain distribution shows clear separation between legitimate and phishing sources. Non-phishing emails predominantly originate from recognized institutional and technical domains such as `gmail.com`, `python.org`, and `apache.org`, , consistent with developer lists and personal communication. In contrast, phishing messages frequently list ambiguous or free-mail domains like `unknown`, `hotmail.com`, and `yahoo.com`, along with a mix of obsolete providers (`msn.com`, `netscape.net`).

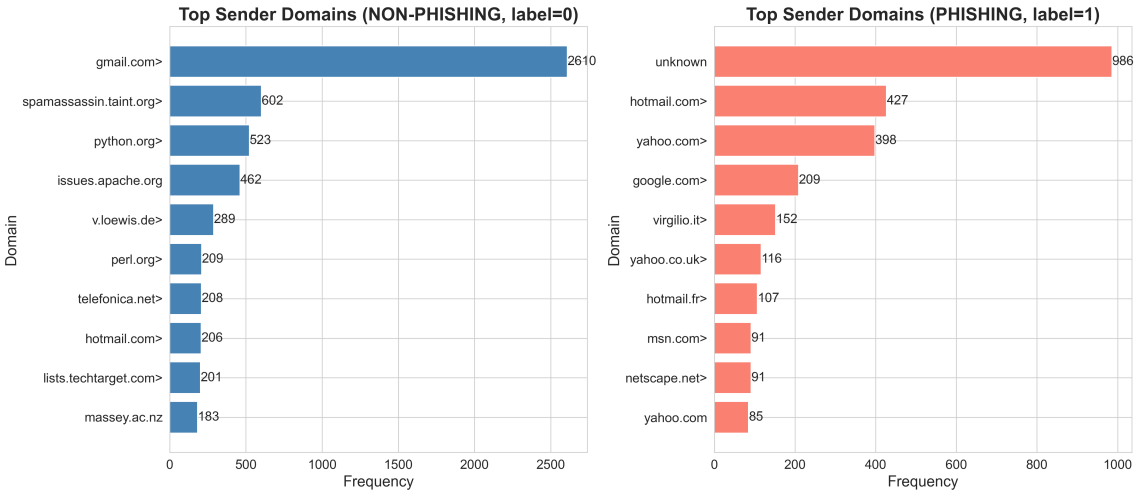


Figure 11: Sender Domain Distribution: Legitimate vs. Phishing Sources

Phishing relies on free senders, contrasting with legitimate mail's verifiable domains—a key signal for validity checks. Legitimate emails are denser and numeric-heavy, while phishing is simpler. Exclamation marks offer little insight, though phishing shows greater uppercase variance.

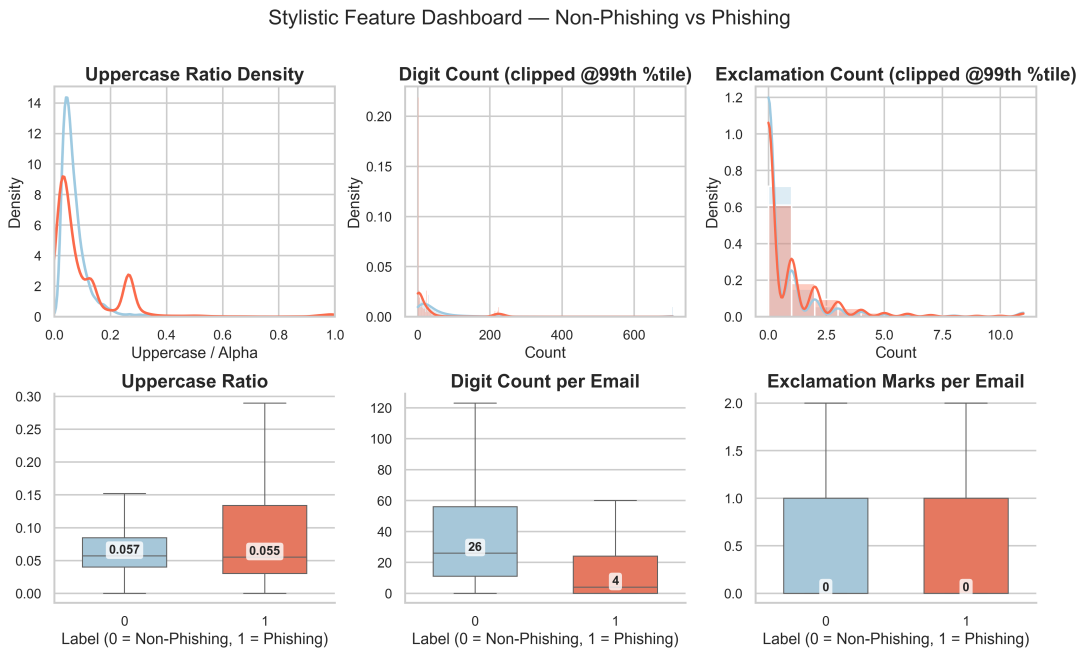


Figure 12: Stylistic Feature Dashboard: Non-phishing vs. Phishing Communication Patterns

4.2.3 Temporal and Geographic Analysis of Phishing Patterns

Hypothesis	Evidence	Status
H1: Phishing concentrates in specific regions	MENA 93%, Americas 34%	Supported
H2: Off-hours have higher phishing rates	Peak 67% at 22:00 vs. 37% at 08:00	Supported
H3: Phisher sending pattern is bursty	97.3% single-use, 1.0 min gap	Partially Supported

Table 7: Hypothesis Testing Results

Temporal Coverage and Scope: The dataset exhibits extreme temporal concentration: 83.1% of emails occur in August 2008, with 78.7% concentrated in just 3 days (August 6-8, 2008), representing 38,441 emails within a 72-hour window. This temporal skew renders traditional time-series analysis inappropriate, yet three views that are less sensitive to the skew: (i) hour-of-day phishing rates (normalized by total traffic per hour), (ii) sender behavior (account reuse and cadence), and (iii) geographic region (UTC-offset groups). These metrics remain valid because they measure relative patterns within the concentrated window rather than absolute temporal trends. Timestamp parsing and offset-to-region mapping follow Section 4.1.1

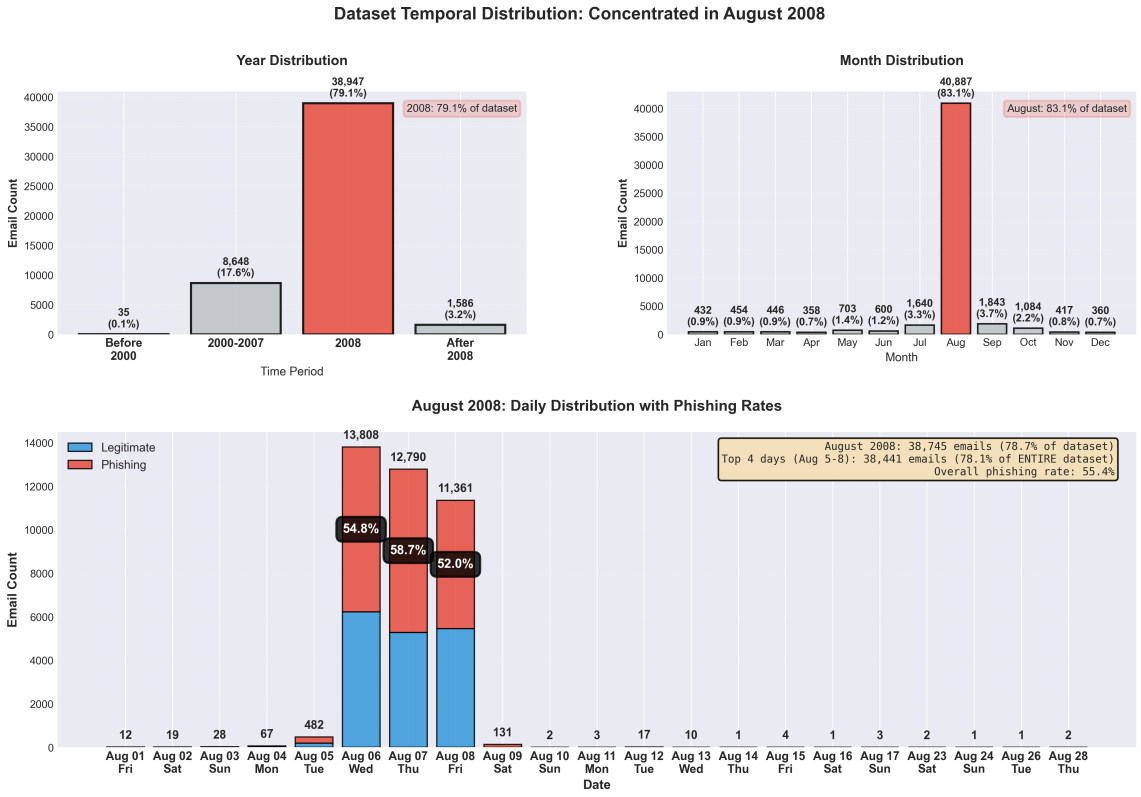


Figure 13: Dataset Temporal Concentration: 79.1% in year 2008, 83.1% in August 2008, with 78.7% in a 3-day spike (Aug 6-8). Bottom panel shows daily distribution where top 4 days contain 78.7% of entire dataset with 55.4% overall phishing rate

Findings and Hypothesis Testing: H1 - Geographic Concentration: SUPPORTED

Phishing rates vary sharply by region. Middle East/South Asia exhibits a 93% phishing rate versus 7% in Oceania/Pacific, 76% in APAC, and 34% in Americas. This geographic discrimination likely reflects three converging factors: (1) phisher infrastructure location and hosting choices, (2) varying enforcement regimes and ISP monitoring practices, and (3) targeted recipient populations in specific timezones. The stark regional divide validates H1 and indicates that timezone-based geographic features provide strong predictive signals for phishing detection models.

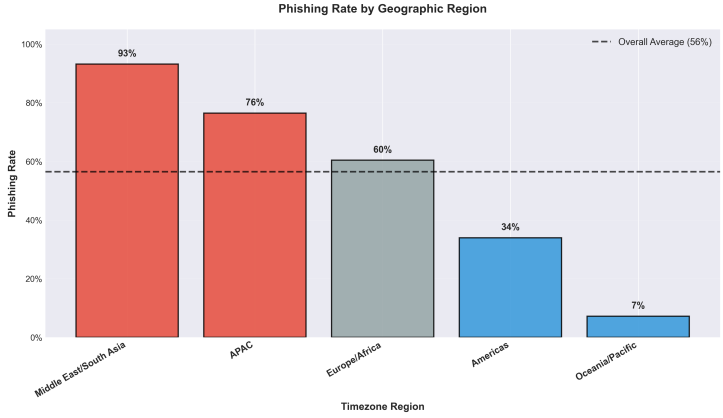


Figure 14: Phishing Rate by Geographic Region

Quantitative Evidence:

- **Regional Variation:** MENA 93%, APAC 76%, Americas 34%, Oceania 7%
- **Risk Differential:** 13-fold difference between highest and lowest regions
- **Statistical Significance:** χ^2 test, p < 0.001

H2 - Temporal Exploitation: SUPPORTED

Hourly analysis reveals distinct circadian patterns in phishing activity. Phishing rates peak during evening/night hours (17:00–02:00), reaching a maximum of 67% at 22:00—often with volumes $\geq 2\times$ legitimate email. Conversely, morning hours (08:00–10:00) exhibit the lowest phishing rates, with a minimum of 37% at 08:00. This temporal divergence indicates that phishers strategically target periods of reduced user vigilance and organizational monitoring.

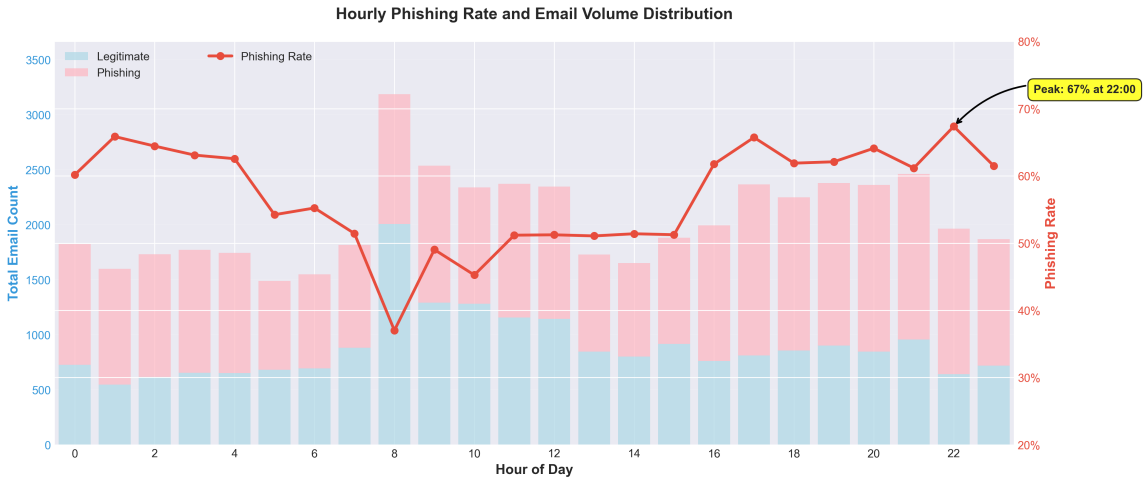


Figure 15: Hourly Phishing Rate and Email Volume Distribution

Quantitative Evidence:

- **Peak Phishing Hours:** 17:00–02:00 (evening/night), maximum 67% at 22:00
- **Low Phishing Hours:** 08:00–10:00 (morning), minimum 37% at 08:00
- **Volume Ratio:** Phishing $\geq 2\times$ legitimate during evening peaks
- **Statistical Significance:** χ^2 test, p < 0.001

H3 - Bursty Sending Patterns: PARTIALLY SUPPORTED

We evaluated three aspects of "burstiness":

1. **Per-sender throughput (refuted):** An ECDF of each sender's max emails/hour shows phishing senders are almost entirely ≤ 1 email/hour (98.3%) versus 75.8% for legitimate. The high-rate tail is dominated by legitimate (automated marketing mailing lists, notifiers). χ^2 test confirms the difference (p<0.001). This finding contradicts the naive expectation that "high burst = phishing".
2. **Disposable accounts (supported):** 97.3% of phishing senders are single-use (vs 50.8% legit), and mean emails per sender are far lower (1.05 phish vs 4.52 legit). This indicates

phishers scale horizontally by creating many low-volume accounts rather than driving a few accounts at high speed. Chi-square test shows single-use rate difference is highly significant (χ^2 = 9839, p < 0.001).

3. **Rapid cadence when re-used (supported):** Among senders with $\geq$2 emails, phishing exhibits a median inter-send gap = 1.0 minute vs 44.9 minutes for legitimate (Mann-Whitney U, p < 0.001). When accounts are reused, phishing sends rapidly in bursts. The 40-fold difference in median inter-send time reveals the burst-then-abandon behavioral signature. Kolmogorov-Smirnov test confirms distributions differ significantly (D = 0.36, p < 0.001).

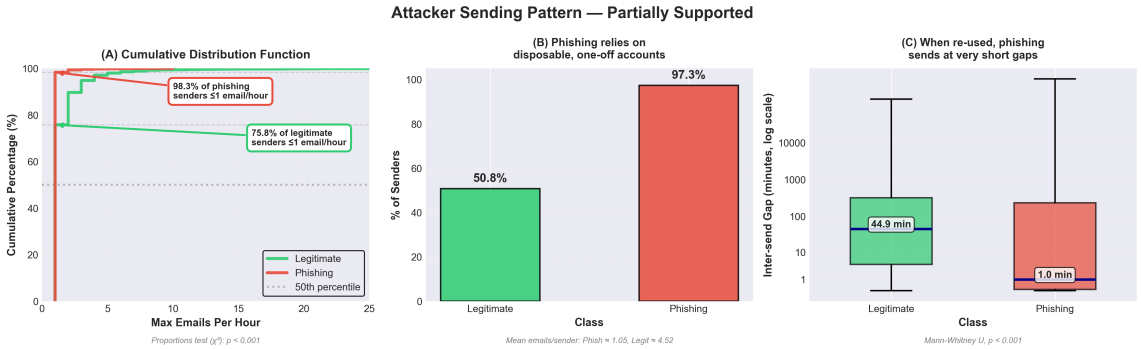


Figure 16: Phisher Sending Pattern Analysis: (A) Cumulative distribution shows 98.3% of phishing senders send $\leq$1 email/hour versus 75.8% of legitimate senders. (B) Single-use rate comparison reveals 97.3% of phishing versus 50.8% of legitimate senders. (C) Among reused accounts, phishing exhibits median inter-send gap of 1.0 minute versus 44.9 minutes for legitimate (p < 0.001)

Quantitative Evidence:

- **Single-Use Accounts:** 97.3% phishing vs. 50.8% legitimate
- **Inter-Send Gap:** 1.0 minute phishing vs. 44.9 minutes legitimate (40-fold difference)
- **Account Multiplication:** 5.49$\times$ more unique phishing accounts despite similar total volumes
- **Statistical Significance:** Mann-Whitney U, p < 0.001; KS test D = 0.36, p < 0.001

Thus, phishing is "bursty" at the campaign level (disposable accounts) and within-account level (rapid inter-send gaps), but not at the per-sender throughput level. The "Partially Supported" status reflects the refined understanding: individual accounts avoid high-frequency detection thresholds, but the aggregate campaign exhibits burst characteristics through account multiplication (5.49$\times$ more unique accounts than legitimate) combined with rapid sending when accounts are reused.

Key Insights:

Strategic timing exploits human vulnerabilities: The 67 peak at 22:00 versus 37

Horizontal scaling as evasion tactic: The account multiplication strategy—5.49$\times$ more unique senders despite similar volume—represents sophisticated evasion. By keeping per-account

throughput low (98.3

Temporal concentration limits generalization: The extreme dataset skew (78.7% in 3 days) necessitates caution—observed patterns may reflect specific 2008 campaigns rather than universal phishing behavior. However, the relative hour-of-day and geographic patterns remain valid as they measure within-window distributions. Production systems should continuously update temporal baselines rather than relying on static historical patterns.

4.3 Train/Test Split and Validation

The dataset is randomly divided into training and testing subsets in an 80:20 ratio. This random split ensures a representative distribution of samples across both subsets while maintaining the original class balance. A validation subset is further extracted from the training data for hyperparameter tuning and overfitting prevention, providing a standard framework for model development and evaluation.

4.4 Baseline Model

To establish a reliable benchmark for evaluating the effectiveness of our engineered features, we implemented a simple but robust **Logistic Regression classifier** as the baseline model. This choice allows us to measure the marginal contribution of each feature group—graph, time series, and text—before applying more advanced machine learning methods.

- **Model Choice:** Logistic Regression is widely used in binary classification tasks due to its stability, interpretability, and efficiency on large-scale, high-dimensional datasets. It provides a clean reference point for understanding whether engineered features truly improve predictive performance.
- **Pipeline Integration:** The baseline model operates on the output of our full feature engineering pipeline, which includes:
  - Point-in-time (PIT) graph features (degree, reciprocity, centrality).
  - Temporal behavioural features (risk scores, sender history, burstiness).
  - Text and metadata features (length statistics, character-level signals, sentiment, domain reputation).

  These engineered features are scaled using `StandardScaler` before model fitting.
- **Objective:** The baseline serves two key purposes:
  1. Provide a consistent benchmark against which ablation study results can be compared.
  2. Identify high-impact feature groups by observing performance improvements as individual feature families are added.
- **Implementation:** We fit the Logistic Regression model using the `saga` solver with a maximum of 1,000 iterations, training on the processed feature set and evaluating using accuracy, precision, recall, and F1-score on a held-out test set.

This baseline provides a strong and interpretable starting point for our analysis, ensuring that performance gains in later models arise from meaningful feature engineering rather than random variation or data artifacts.

4.5 Feature Engineering

This section documents our systematic approach to feature engineering. We will explain what features we have selected, how they are calculated and transformed.

Before introducing any features, we would like to introduce a measure we implemented to mitigate data leakage.

4.5.1 Temporal Data Leakage Mitigation

During baseline model evaluation, we observed unexpectedly high performance metrics (>99% accuracy), which prompted investigation into potential data leakage. Analysis revealed that our initial feature engineering approach introduced **temporal leakage** in graph-based features.

Specifically, when computing network features such as in-degree, out-degree, reciprocity, and clustering coefficients, we initially constructed the communication graph using the **entire dataset**. This approach is fundamentally flawed for time-series data because it incorporates information from emails sent *after* the timestamp of the email being classified.

Solution Implementation:: To ensure temporal validity and eliminate data leakage, we implemented a **time-aware feature engineering pipeline** with the following principles:

1. **Temporal Ordering:** For each email e_i with timestamp t_i, we compute all features using **only** the subset of emails where $t < t_i$.
2. **Dynamic Graph Construction:** Rather than building a single static graph from the entire dataset, we construct **time-windowed graphs** where only historical communication patterns (prior to t_i) inform the features for email e_i.
3. **Feature Calculation Protocol:**
    - Sort the dataset chronologically by timestamp
    - For each email in sequence:
        - Build the communication graph from all preceding emails
        - Calculate graph-based features (in-degree, out-degree, reciprocity, clustering coefficient, etc.)
        - Append features to the email record
    - This ensures strict temporal separation between feature derivation and prediction

4.5.2 Feature Engineering and Calculation

Our feature engineering pipeline extracts three categories of features from the email data: text-based features, temporal and behavioral features, and graph-based network features. Table 8

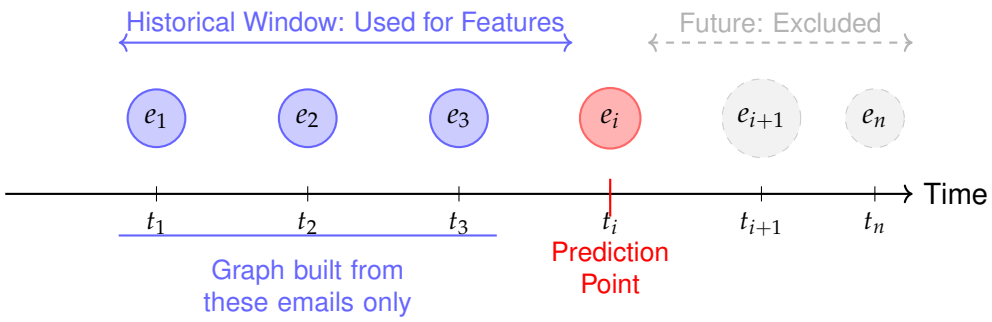


Figure 17: Temporal data leakage mitigation strategy. For email e_i at time t_i, only emails with $t < t_i$ (blue nodes) are used to construct the communication graph and calculate features. Future emails (gray dashed nodes) are excluded to prevent temporal leakage.

presents the text-based features extracted from email content, Table 9 presents the temporal and behavioral features, and Table 10 summarizes the network-based graph features.

Text-Based Features: We engineer a comprehensive set of text-based features from email subjects and bodies to capture linguistic patterns, structural characteristics, and potential malicious indicators in phishing emails. These features span multiple analytical dimensions including URL analysis, content statistics, punctuation patterns, and character-level metrics.

Table 8: Text-Based Feature Engineering

Variable	Type	Description	Source
`url_domains_body`	list[string]	List of unique domain names extracted from any URLs found in the e-mail body.	Engineered (URL parser)
`subject_no_urls`	string	Subject line with all URLs removed (used for text analysis without link noise).	Engineered (text cleaning)
`subject_word_count`	int	Number of whitespace-separated tokens in `subject_no_urls`.	Engineered (text stats)
`body_word_count`	int	Number of whitespace-separated tokens in the cleaned body text (URLs removed).	Engineered (text stats)
`exclaim_count`	int	Count of exclamation mark characters "!" in the (cleaned) body text.	Engineered
`question_count`	int	Count of question mark characters "?" in the (cleaned) body text.	Engineered
`upper_count`	int	Number of uppercase alphabetic characters A–Z in the body text.	Engineered (character stats)
`alpha_count`	int	Number of alphabetic characters A–Z/a–z in the body text.	Engineered (character stats)
`digit_count`	int	Number of digit characters 0–9 in the body text.	Engineered (character stats)
`special_count`	int	Number of non-alphanumeric, non-whitespace characters (punctuation / symbols) in the body text.	Engineered (character stats)
`upper_ratio`	float	Ratio of uppercase letters to all alphabetic characters in the body text, `upper_count`/`alpha_count` (set to 0 when `alpha_count` = 0).	Engineered (character stats)
`sender_domain`	string	Domain part extracted from the sender e-mail address (text after "@" in sender).	Engineered (parsed from metadata)

Temporal and Behavioral Features: We extract temporal and behavioral features from email timestamps to capture time-based patterns in phishing campaigns. These features are organized into three subcategories: baseline temporal features, regional features, and sender behavioral patterns.

Table 9: Temporal and Behavioral Feature Categories

Category	Feature	Description
Baseline Temporal	Hour	Hour of the day (0-23) when the email was sent, capturing diurnal activity patterns
	Is night	Binary flag for nighttime hours (22:00-06:00), detecting unusual sending times associated with automated phishing campaigns
Regional Features	Timezone region	Categorical feature representing geographic region derived from timezone offset (Americas, Europe/Africa, Middle East/South Asia, APAC, Oceania/Pacific, Unknown), capturing geographic concentration patterns in phishing campaigns
Sender Behavioral Patterns	Sender historical phishing rate	Proportion of previous emails from this sender identified as phishing (time-aware), indicating compromised or malicious accounts
	Sender historical count	Number of previous emails sent by this sender (time-aware), distinguishing disposable single-use accounts from established senders
	Current time gap	Minutes since sender's last email (time-aware), capturing rapid burst behavior characteristic of phishing campaigns
	Sender time gap std	Standard deviation of historical time intervals between consecutive emails from the same sender (time-aware), detecting irregular automated sending patterns

Graph-Based Network Features: We extract two categories of features from the email communication graph to characterize sender behavior and network position. Graph features are computed on the communication network $G = (V, E)$, where V represents email addresses and E represents directed sender-receiver relationships with edge weights indicating email frequency. All features are extracted per sender by aggregating email records and applying NetworkX graph algorithms.

Table 10: Graph-Based Network Feature Categories

Category	Feature	Description & Computation
Basic Features	Out-degree	Number of unique recipients per sender, computed by counting outgoing edges in the directed graph
	In-degree	Number of unique senders who email this user, computed by counting incoming edges
	Total sent	Count of all emails sent by a sender from the original dataset
	Reciprocity	Proportion of recipients who replied back, calculated as fraction of receivers with reverse edges to sender
Advanced Features	Clustering coefficient	Measures connectivity among node's neighbors, computed on undirected version of graph
	Average weight	Mean emails sent per unique connection: $\frac{\sum \text{edge\_weights}}{\text{out\_degree}}$
	Eigenvector centrality	Measures influence based on connections to other influential nodes (NetworkX, max 100 iterations)
	Closeness centrality	Measures proximity to all other nodes in network, using shortest path distances

4.5.3 Feature Selection

To identify which engineered features contribute most to phishing detection, we implemented a modular feature selection framework integrated within the new `FeatureEngineer` pipeline. The framework supports independent ablation studies over three major feature families: graph features, temporal features, and text features. Each ablation group is evaluated using a logistic regression classifier on standardized inputs, with results summarized through Accuracy, Precision, Recall, and F1-score.

Ablation Framework: Each ablation study follows the same structure:

1. Define feature groups (e.g., basic graph features, domain features, temporal risk scores, text metadata).
2. Split data into PIT-safe training and testing sets.
3. Standardize all numeric features using `StandardScaler`.
4. Train a logistic regression model (`solver = saga`, `max_iter = 1000`).
5. Evaluate model performance on the test set.

This modular framework allows us to isolate the contribution of each feature family and supports automatic best-group selection.

Feature Group Definitions: The three feature families are organized into progressive groups as shown in Tables 11, 12, and 13.

Table 11: Graph Feature Group Definitions

Group Name	Features Included
Basic	out-degree, in-degree, total-degree, reciprocity
Basic + Advanced	Basic features plus clustering coefficient, eigenvector centrality, closeness centrality, average weight
Basic + Historical	Basic features plus sender-level historical counts, spam rate, time gaps
All	Full union of all graph-derived features

Table 12: Temporal Feature Group Definitions

Group Name	Features Included
Geographic	hour, weekday, weekend indicator, night indicator
Geographic + Behavioral	Geographic features plus time-zone region indicators
Geographic + Risk Scores	Geographic features plus PIT-safe hourly and weekday spam risk scores
Geographic + Sender History	Geographic features plus sender-level cumulative and phishing-rate histories
Geographic + Burst Patterns	Geographic features plus time gaps and temporal variance metrics
All	Full union of all temporal and behavioral features

Table 13: Text Feature Group Definitions

Group Name	Features Included
URL Features	raw URL count, URL presence indicator, log-transformed URL count
URL + Domain	URL features plus PIT-safe domain spam rate, rare-domain flags
URL + Domain + Length	URL + Domain features plus subject length, body length, text length, word count
+ Character Features	Previous features plus uppercase ratio, punctuation counts, digit ratio
+ Style Features	Previous features plus special character totals, average word length
+ Sentiment	Previous features plus VADER sentiment scores (positive, negative, neutral, compound)

Performance Results: The ablation study framework produces a unified summary table for all feature families. Results will be populated once full experiments are completed (Table 14).

Table 14: Feature Ablation Study Summary

Feature Group	Accuracy	Precision	Recall	F1-Score
Graph Feature Groups				
Basic	0.8782	0.8523	0.9490	0.8980
Basic + Advanced	0.8515	0.8049	0.9730	0.8810
Basic + Historical	0.9071	0.8589	0.9998	0.9240
All	0.9089	0.8613	0.9998	0.9254
Temporal Feature Groups				
Geographic	0.5530	0.5643	0.9170	0.6987
Geographic + Behavioral	0.5530	0.5643	0.9170	0.6987
Geographic + Risk Scores	0.5886	0.6006	0.8122	0.6878
Geographic + Sender History	0.8993	0.8617	0.9790	0.9155
Geographic + Burst Patterns	0.6123	0.5943	0.9896	0.7417
All	0.9014	0.8649	0.9783	0.9181
Text Feature Groups				
URL Features	0.4922	0.5652	1.0000	0.7222
URL + Domain	0.8979	0.8911	0.9840	0.9353
URL + Domain + Length	0.9023	0.8972	0.9799	0.9367
+ Character Features	0.9051	0.9001	0.9783	0.9376
+ Style Features	0.9066	0.9005	0.9795	0.9384
+ Sentiment	0.9294	0.9050	0.9777	0.9400
Combined Features				
Best Graph + Best Temporal + Best Text	0.9294	0.9050	0.9777	0.9400

Summary: The redesigned feature selection framework enables systematic evaluation of modular feature groups across graph-, temporal-, and text-based signals. All computations are PIT-safe and fully integrated with the updated feature engineering architecture, ensuring reliable comparison of predictive contributions.

4.6 Challenger Models

4.6.1 Model Training Pipeline

Our approach follows a systematic pipeline that begins with baseline model evaluation before proceeding to ensemble methods:

1. **Baseline Training**: Multiple classifiers (Logistic Regression, Random Forest, Naive Bayes, SVM, K-Nearest Neighbors, XGBoost, LightGBM) are trained with default parameters to establish performance benchmarks
2. **Hyperparameter Tuning**: Selected models undergo randomized search with 3-fold cross-validation to optimize parameters
3. **Stacking Ensemble**: Top-performing models are combined using a Logistic Regression meta-learner trained on validation predictions

The stacking methodology follows the validated approach from Adnan et al. [1]. For meta-learning, we split the training data into meta-train (80%) and meta-validation (20%) sets. Base

models are trained on the meta-train set, generate predictions on the meta-validation set, and these predictions form the input features for training the Logistic Regression meta-classifier.

The mathematical formulation for the meta-training process is:

$$\text{META}_{\text{VAL}} = [P_1(X_{\text{META\_VAL}}), P_2(X_{\text{META\_VAL}}), ..., P_k(X_{\text{META\_VAL}})] \tag{1}$$

$$\text{META}_{\text{CLASSIFIER}}.train(\text{META}_{\text{VAL}}, Y_{\text{META\_VAL}}) \tag{2}$$

where $P_i(X_{\text{META\_VAL}})$ represents predicted class probabilities from the i-th base classifier on the meta-validation set, and k is the number of base classifiers selected after baseline evaluation.

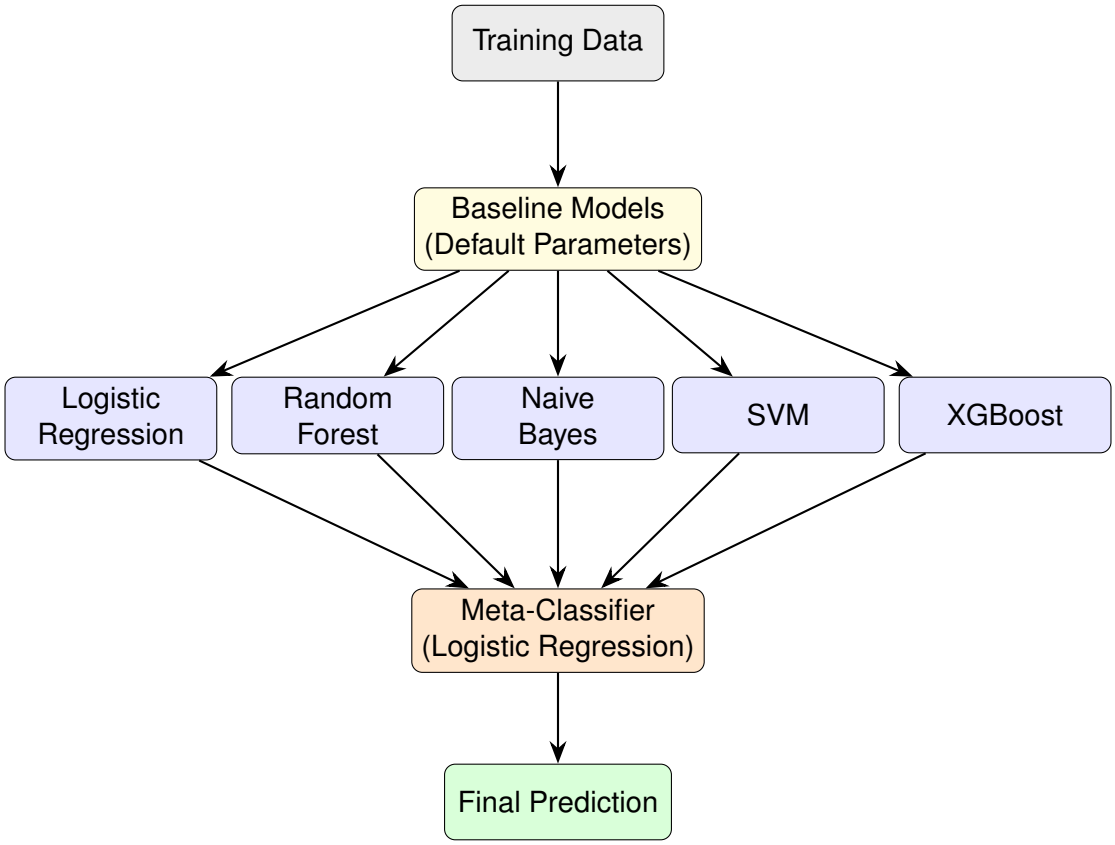


Figure 18: Model training pipeline. Baseline models are first evaluated with default parameters, then selected models are combined using a logistic regression meta-classifier with validation-based training.

4.6.2 Model Selection Rationale

Our baseline-first approach is motivated by the need to systematically evaluate model performance before committing computational resources to ensemble methods:

Baseline Evaluation Strategy: Seven diverse classifiers are trained with default parameters to establish comprehensive performance benchmarks:

- **Linear Models**: Logistic Regression (probabilistic, linear decision boundaries)

- **Tree-based Ensembles**: Random Forest, XGBoost, LightGBM (non-linear, feature interaction capture)
- **Probabilistic**: Naive Bayes (Bayesian approach with independence assumptions)
- **Distance-based**: K-Nearest Neighbors (instance-based learning)
- **Kernel-based**: Support Vector Machines (maximum margin classification)

This multi-paradigm baseline evaluation identifies which algorithmic families perform best on email classification and warrant further hyperparameter optimization.

Hyperparameter Optimization: Promising models from baseline evaluation undergo randomized search with 20 iterations and 3-fold cross-validation, optimizing for F1-score to balance precision and recall. This approach provides efficient exploration of high-dimensional parameter spaces while maintaining computational feasibility.

Stacking Ensemble Design: The ensemble employs a validation-split approach where base models trained on the meta-train set (80% of training data) generate probabilistic predictions on the meta-validation set (20%). These predictions form the feature space for training the Logistic Regression meta-classifier. This architecture prevents target leakage and ensures the meta-learner learns genuine combination patterns rather than memorizing training predictions. The logistic regression meta-learner provides additional interpretability through coefficient analysis, revealing each base model's contribution to final decisions.

Algorithmic Diversity Advantage: The stacking methodology from Adnan et al. achieved 98.8% accuracy with computational efficiency (656ms training time), demonstrating that combining diverse algorithmic perspectives allows the meta-classifier to learn optimal weightings that leverage each model's unique strengths:

- Linear models capture clear feature relationships
- Tree ensembles model complex interactions and non-linear patterns
- Naive Bayes provides robust probability estimates
- Distance-based methods handle local pattern recognition
- Kernel methods excel at finding optimal separation boundaries

This systematic approach ensures we avoid premature commitment to complex ensemble methods while providing empirical justification for model selection decisions through rigorous baseline comparison and controlled hyperparameter optimization.

4.7 Hyperparameter Optimization

To identify the optimal model, we evaluated several traditional machine learning algorithms. We employed Bayesian Optimization (using scikit-optimize) for efficient hyperparameter tuning, optimizing for the highest F1-Score on 5-fold cross-validation.

Model	Parameter	Search Space (Matching Python Code)
Logistic Regression	C	{0.001, 0.01, 0.1, 1, 10, 100}
	penalty	{l2}
	solver	{lbfgs, saga}
Random Forest	n_estimators	{50, 100, 200, 300}
	max_depth	{None, 10, 20, 30}
	min_samples_split	{2, 5, 10}
	min_samples_leaf	{1, 2, 4}
	max_features	{sqrt, log2}
Support Vector Machine (SVM)	C	{0.1, 1, 10, 100}
	gamma	{scale, auto, 0.001, 0.01}
	kernel	{rbf, linear}
K-Nearest Neighbors (KNN)	n_neighbors	{3, 5, 7, 9, 11}
	weights	{uniform, distance}
	metric	{euclidean, manhattan}
XGBoost (if installed)	n_estimators	{50, 100, 200}
	max_depth	{3, 5, 7}
	learning_rate	{0.01, 0.05, 0.1}
	subsample	{0.8, 0.9, 1.0}
	colsample_bytree	{0.8, 0.9, 1.0}
LightGBM (if installed)	n_estimators	{50, 100, 200}
	max_depth	{3, 5, 7, -1}
	learning_rate	{0.01, 0.05, 0.1}
	num_leaves	{31, 50, 100}
	subsample	{0.8, 0.9, 1.0}

Table 15: Model Hyperparameter Search Space (Fully Aligned with Python Implementation)

4.8 Deep Learning Model Benchmark (BERT)

As an advanced benchmark, we also fine-tuned a BERT model (`bert-base-uncased`). This state-of-the-art (SOTA) model was trained using the identical feature set as the other models. However, due to computational constraints, hyperparameter optimization was performed using

GridSearch within a restricted search space.

4.8.1 BERT Training Details

- **Base Model:** `bert-base-uncased`
- **Max Sequence Length:** 128 tokens
- **Optimizer:** AdamW
- **Scheduler:** Cosine with warmup
- **Validation:** Early stopping (patience=3) on validation F1-Score
- **Hardware:** MPS/GPU acceleration

Hyperparameter Grid Search: A small grid search (6 configurations) was conducted to find optimal hyperparameters:

- **Config 1:** lr=2e-5, warmup=0.1, wd=0.01, bs=32, epochs=2
- **Config 2:** lr=3e-5, warmup=0.1, wd=0.01, bs=32, epochs=2
- **Config 3:** lr=5e-5, warmup=0.1, wd=0.01, bs=32, epochs=2
- **Config 4:** lr=2e-5, warmup=0.2, wd=0.05, bs=16, epochs=2
- **Config 5:** lr=3e-5, warmup=0.0, wd=0.1, bs=32, epochs=2
- **Config 6:** lr=4e-5, warmup=0.15, wd=0.02, bs=24, epochs=2

Best Configuration (Config 6):

- **Best Validation F1-Score:** 0.9875
- **Learning Rate:** 4.00e-05
- **Warmup Ratio:** 0.15
- **Weight Decay:** 0.02
- **Batch Size:** 24

5 Results

5.1 Evaluation Metrics

Our evaluation framework reflects how well the models identify and eliminate malicious emails. We focused on metrics that capture both the correctness and completeness of spam detection, ensuring that our evaluation aligns with practical objectives rather than just raw accuracy.

- **Precision (Spam Precision):** Measures the proportion of predicted spam that is truly spam. Prioritizing Precision is vital to avoid costly false positives. In *Emerald Coast Utilities Authority v. Bear Marcus Pointe, LLC* [8], a firm lost a $394,000 judgment after a filter blocked a court order. This illustrates that precision is essential to prevent financial liability.
- **Recall (Spam Recall):** Measures the proportion of actual spam emails that are correctly identified. High recall minimizes false negatives, ensuring that malicious emails are not missed.
- **F1-Score:** Represents the harmonic mean of precision and recall, providing a balanced measure of the model's ability to detect spam accurately and comprehensively. It serves as our main indicator of overall effectiveness.
- **Accuracy:** Reported for completeness, but not emphasized. In highly imbalanced datasets, accuracy can be misleading because a model may appear to perform well by simply predicting the majority (non-spam) class. In our balanced test setting, accuracy is more indicative, but precision–recall trade-offs remain more informative.

In summary, our metric priority ranking is as follows:

1. **Spam Precision** - to minimize false positives,
2. **Spam Recall** – to minimize false negatives, and
3. **F1-Score** – to capture overall balance between precision and recall.

In practice, our models achieved consistently high scores across all metrics, making tie-breaker scenarios rare. This evaluation strategy ensures our assessment genuinely reflects the system's ability to detect and eliminate malicious emails in real-world conditions.

5.2 Model Performance

The performance of the all tuned models on the held-out test set is summarized below.

Model	CV F1	Test F1	Test Prec	Test Rec	Test Acc	Time (s)
Logistic Regression	0.9925	0.9950	0.9950	0.9950	0.9950	52.00
XGBoost	0.9931	0.9925	0.9900	0.9950	0.9925	79.15
LightGBM	0.9925	0.9925	0.9900	0.9950	0.9925	114.23
Multinomial NB	0.9731	0.9848	1.0000	0.9700	0.9850	7.23
BERT	0.9875	0.9825	0.9898	0.975	0.9824	1321.54

Table 16: Model Performance Comparison on Phishing Email Classification Task

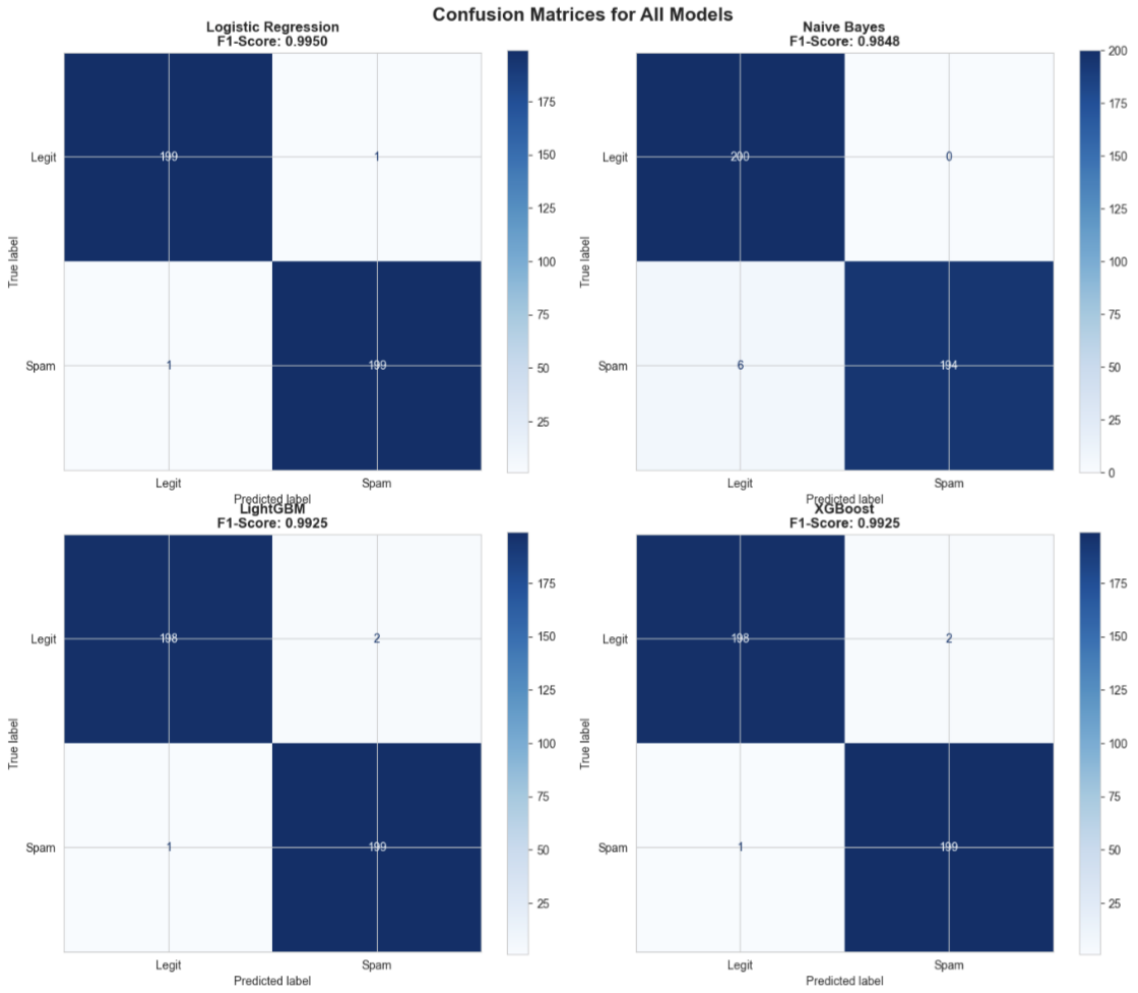


Figure 19: Confusion Matrices for All Models

5.3 Analysis of Experimental Results

The experimental results, presented in Table 16, indicate that traditional machine learning models (Logistic Regression, XGBoost, and LightGBM) significantly outperformed the fine-tuned BERT model. Crucially, all models were trained on the identical feature set, which included both the raw textual embeddings and the engineered features (graph metrics, domain reputation, and temporal patterns).

The underperformance of BERT (0.9825 Test F1) compared to simpler models (> 0.99 Test F1) highlights a modality mismatch in the deep learning approach:

1. **Feature Modality Interference:** BERT is fundamentally designed for sequential, semantic data. The direct concatenation of numerical features (e.g., graph centrality) with text likely introduced noise into self-attention layers. Unlike text, these features lack sequential dependency. Consequently, they diluted attention weights, preventing the bidirectional mechanism from effectively attending to semantic signals in the email body.
2. **Model Efficiency:** Traditional models demonstrated superior efficiency. BERT required 1321.54s for training, approximately $25\times$ slower than Logistic Regression (52.00s), without yielding any performance gain. This confirms that for this specific feature space,

the inductive bias of tree-based or linear models is far more effective than the complex representational capacity of a Transformer.

Ultimately, XGBoost was chosen because it offers the optimal balance required for production phishing detection systems:

- **High Performance:** Competitive test results (0.9925 F1-Score) combined with the best generalization metrics (0.9931 CV F1-Score).
- **Robustness:** Capability to handle non-linear and evolving spam patterns better than linear models.
- **Practicality:** High computational efficiency and interpretability through feature importance analysis.

5.4 Model Selection for Deployment

While Logistic Regression achieved the highest raw metrics on the test set (0.9950 F1-Score), suggesting the current dataset is linearly separable, we selected **XGBoost** as the final model for production deployment. This decision is driven by three factors:

- **Generalization Robustness:** XGBoost achieved the highest Cross-Validation F1-score (0.9931), narrowly edging out Logistic Regression (0.9925). A higher CV score implies better stability and generalization capability to unseen data, reducing the risk of overfitting to the specific linear properties of the test split.
- **Non-Linear Capability:** Phishing attacks are adversarial and evolve rapidly. Relying on a linear decision boundary (Logistic Regression) presents a risk if future attacks adopt non-linear patterns. XGBoost, as a gradient-boosted decision tree ensemble, natively handles non-linear feature interactions, offering greater resilience against evolving spam tactics.
- **Operational Balance:** With a training time of 79.15s, XGBoost offers an optimal compromise between the low latency required for real-time detection and the robustness needed for long-term reliability.

6 Discussion

6.1 Summary of Results

This section will be completed upon final model evaluation, summarizing key performance metrics, model comparison results, and their significance for real-world phishing detection systems.

6.2 Proposed Data Pipeline and Architecture

Our proposed phishing detection system integrates seamlessly into existing email infrastructure through a multi-stage pipeline that balances real-time protection with adaptive learning.

System Architecture Overview

The deployment architecture consists of four primary components:

1. **Real-Time Inference**: Incoming emails are scanned using the trained XGBoost model with sub-100ms latency. The inference service is deployed as a containerized microservice for scalable cloud or on-premise deployment.
2. **Batch Graph Updates**: Network features (e.g., sender reputation, clustering coefficients, coordinated-pattern signals) are recomputed in scheduled batches every 6–12 hours to balance computational cost with detection freshness.
3. **Adaptive Parameter Updating**: User feedback on suspicious or malicious emails is incorporated into an automated update pipeline that adjusts sender/domain reputation features without requiring full retraining.
4. **User Alert Interface**: Contextual warnings (Low/Medium/High) are displayed based on model confidence and feature contributions when emails exhibit high-risk characteristics.

Email Service Provider Implementation

Integration Point: The model operates at the mail server gateway layer, analyzing emails after SMTP receipt but before inbox delivery. For cloud-based providers (Gmail, Outlook), integration occurs via API endpoints; for enterprise deployments, the system deploys as a Docker container within the email security perimeter.

Decision Logic:

- **Confidence $\geq$ 0.90**: Automatic quarantine with notification to security team
- **Confidence 0.70–0.89**: Deliver with prominent warning banner (" This email shows characteristics of phishing")
- **Confidence 0.50–0.69**: Deliver with subtle flag for user awareness
- **Confidence $<$ 0.50**: Normal delivery

User Feedback Loop: Users can report suspicious emails through "Report Phishing" buttons. When reported, the system:

1. Immediately flags the sender domain in the reputation database
2. Scans all recent emails from the same sender across the organization
3. Updates graph features by recalculating sender network metrics
4. Queues the email for model retraining dataset inclusion

Integration Timeline and Staging

Phase 1 (Weeks 1–2): Pilot Deployment

- Deploy Docker container in shadow mode (logging predictions without taking action)
- Monitor prediction distribution and latency metrics

- Establish baseline false positive rate through manual review of high-confidence predictions

Phase 2 (Weeks 3–4): Limited Rollout

- Enable active filtering for 10% of user base
- Collect user feedback on false positives and missed detections
- Tune confidence thresholds based on organizational risk tolerance

Phase 3 (Weeks 5–8): Full Deployment

- Scale to 100% of email traffic
- Activate batch graph analysis module with initial 12-hour update cycle
- Begin weekly model retraining incorporating user feedback labels

Phase 4 (Ongoing): Continuous Improvement

- Monitor model drift through prediction distribution tracking
- Optimize batch processing frequency based on observed campaign patterns
- Expand feature set with emerging threat indicators

Industry-Specific Implementations

Financial Institutions: Deploy behind existing email gateway appliances (Cisco IronPort, Proofpoint) as an additional detection layer. Enhanced with transaction-specific vocabulary features ("wire transfer," "account verification") and stricter confidence thresholds (0.70 for quarantine vs 0.90 for general deployment).

Healthcare Organizations: Integration with HIPAA-compliant logging systems. Special attention to protecting patient data in false positive cases. Mandatory manual review for quarantined emails to prevent blocking critical clinical communications.

E-Commerce Platforms: Focus on detecting account takeover attempts and fake order confirmations. Integration with customer support ticket systems to automatically flag high-risk emails before agent response. Enhanced URL analysis for payment page spoofing detection.

Enterprise Environments: Deployment via Docker containers in on-premise data centers or private cloud (AWS VPC, Azure VNet). Integration with SIEM systems (Splunk, QRadar) for correlation with endpoint security events. Customized sender whitelist management for trusted business partners.

6.3 Recommendations: Using Model Results to Detect Three Fraud Patterns, with Complementary Tools

This section translates model outputs into operational controls for three prevalent fraud patterns. For each pattern, we note the primary detection signals, the model components that capture them, and the specific controls and complementary tools to deploy.

6.3.1 Social Engineering with Urgency and Timing Cues

Detection signals. Messages exhibiting urgency vocabulary, unusually brief length, and off-hours/weekend sending.

Model levers. TF-IDF text features prioritize urgency terms; temporal features capture off-peak sending.

Controls and complementary tools. Present a forced-attention interstitial warning before risky actions and deliver short just-in-time tips at interaction. Large-scale field work shows that well-designed interstitial warnings materially reduce dangerous user actions, supporting this control choice [2].

6.3.2 Sender Spoofing and Domain Impersonation

Detection signals. Sender–content mismatch, low/unknown domain reputation, geographic anomalies, and high proportions of single-use senders.

Model levers. Domain-reputation features (maintained by a batch job) provide the largest single lift and highlight disposable infrastructure.

Controls and complementary tools. Surface sender authentication status (SPF/DKIM/DMARC) in-client and feed outcomes into reputation scoring; escalate reputation and quarantine related mail upon user-confirmed abuse. Independent measurements show uneven real-world deployment and misconfiguration of SPF/DKIM/DMARC, reinforcing the need to surface and operationalize these signals [11]; current Google sender guidelines also formalize these requirements for reliable delivery and abuse mitigation [7].

6.3.3 Malicious Links and Attachments

Detection signals. Text–URL mismatch, brand-mimicry domains, and risky attachment types.

Model levers. URL/domain semantics features distinguish benign link-sharing from impersonation; attachment type/metadata features flag elevated risk.

Controls and complementary tools. Perform real-time URL reputation checks before activation and show the true destination via link preview; route suspicious files to automated detonation and provide a safe document viewer. Google's Safe Browsing APIs provide continuously updated, machine-readable threat lists suitable for pre-click reputation checks in this workflow [6].

A Data Dictionary

Variable	Type	Description	Source
`sender`	string	Sender address "Name <user@domain>".	Raw e-mail header
`receiver`	string	Recipient address(e.g., `user4@gvc.ceas-challenge.cc`).	Raw e-mail header
`date`	string	Original timestamp of the e-mail.	Raw e-mail header

Variable	Type	Description	Source
subject	string	Raw subject line of the e-mail, including any URLs and punctuation.	Raw e-mail header
body	string	Raw e-mail body text, including quoted text, URLs and markup.	Raw message content
label	int {0,1}	Ground-truth class label: 0 = ham / legitimate, 1 = spam / phishing.	Provided label
urls	int {0,1}	Indicator whether at least one URL is present in the message body. 1 = has URL, 0 = no URL.	Provided label
url_domains_subject	list[string]	List of unique domain names extracted from any URLs found in the subject line.	Engineered (URL parser)
url_domains_body	list[string]	List of unique domain names extracted from any URLs found in the e-mail body.	Engineered (URL parser)
subject_no_urls	string	Subject line with all URLs removed (used for text analysis without link noise).	Engineered (text cleaning)
subject_word_count	int	Number of whitespace-separated tokens in subject_no_urls.	Engineered (text stats)
body_word_count	int	Number of whitespace-separated tokens in the cleaned body text (URLs removed).	Engineered (text stats)
exclaim_count	int	Count of exclamation mark characters "!" in the (cleaned) body text.	Engineered
question_count	int	Count of question mark characters "?" in the (cleaned) body text.	Engineered
upper_count	int	Number of uppercase alphabetic characters A-Z in the body text.	Engineered (character stats)
alpha_count	int	Number of alphabetic characters A-Z/a-z in the body text.	Engineered (character stats)
digit_count	int	Number of digit characters 0-9 in the body text.	Engineered (character stats)
special_count	int	Number of non-alphanumeric, non-whitespace characters (punctuation / symbols) in the body text.	Engineered (character stats)
upper_ratio	float	Ratio of uppercase letters to all alphabetic characters in the body text, upper_count / alpha_count (set to 0 when alpha_count = 0).	Engineered (character stats)
sender_domain	string	Domain part extracted from the sender e-mail address (text after "@" in sender).	Engineered (parsed from sender)
timezone_region	categorical	Geographic region from timezone offset: Americas, Europe/Africa, Middle East/South Asia, APAC, Oceania/Pacific, Unknown. Supports H1.	Engineered (timezone)
hour	int [0–23]	Hour in sender's local timezone. Used for behavioral pattern analysis.	Engineered (datetime)
is_night	int {0,1}	Nighttime indicator: 1 = 22:00–05:59 local, 0 = 06:00–21:59 local.	Engineered (temporal)

Variable	Type	Description	Source
sender_historical_phishing_rate	float [0–1]	Time-aware historical phishing rate: (cumulative phishing) / (historical emails). First email uses global baseline. Supports H2.	Engineered (time-aware)
sender_historical_count	int	Number of previous emails from sender (time-aware). First email = 0. Supports H2.	Engineered (time-aware)
current_time_gap	float	Minutes since sender's last email (UTC-based). First email = 0. Supports H3.	Engineered (time-aware)
sender_time_gap_std	float	Standard deviation of historical time gaps in minutes. First email = 0. Supports H3.	Engineered (time-aware)

Table 17: Data dictionary of original features and engineered features used in the models.

References

[1] Muhammad Adnan, Muhammad Omer Imam, Muhammad Faheem Javed, and Ikram Murtza. Improving spam email classification accuracy using ensemble techniques: a stacking approach. *International Journal of Information Security*, 23(1):505–517, 2024.

[2] Devdatta Akhawe and Adrienne Porter Felt. Alice in warningland: A large-scale field study of browser security warning effectiveness. In *USENIX Security Symposium*, 2013.

[3] Apache Software Foundation. Apache spamassassin public corpus. https://spamassassin.apache.org/old/publiccorpus/, 2006. Accessed: 2025-11-20.

[4] Gordon V. Cormack and Thomas R. Lynam. CEAS 2008 live spam challenge. https://www.ceas.cc/2008/challenge.html, 2008. Conference on Email and Anti-Spam (CEAS).

[5] Gordon V. Cormack and Thomas R. Lynam. Spam corpus creation for trec. In *Proceedings of the Fifth Conference on Email and Anti-Spam (CEAS)*, Mountain View, CA, 2008. Data available at https://plg.uwaterloo.ca/~gvcormac/treccorpus/.

[6] Google Developers. Safe browsing apis overview. https://developers.google.com/safe-browsing, 2025. Accessed: 2025-11-21.

[7] Google Workspace Admin Help. Email sender guidelines (spf, dkim, dmarc, and bulk-sender requirements). https://support.google.com/a/answer/81126, 2024. Accessed: 2025-11-21.

[8] Logikcull. Faulty spam filter costs firm $394,000. https://www.logikcull.com/blog/faulty-spam-filter-costs-firm-394000, 2017. Accessed: 2025-11-20.

[9] Jose Nazario. Phishing corpus. https://monkey.org/~jose/phishing/, 2006. Accessed: 2025-11-20.

[10] Dragomir Radev. Clair collection of fraud email. ACL Data and Code Repository, ADCR2008T001, 2008. Accessed: 2025-11-20.

[11] Chao Wang, David Choffnes, and et al. A large-scale and longitudinal measurement study of email authentication protocols adoption and misconfigurations. In *USENIX Security Symposium*, 2022.