Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

Real-Time Phishing Detection for SMS, WhatsApp, and Email using Machine Learning

B Gananatha Reddy, S Akshaya, K Sukrutha, M Sharon
Department of Cyber Security, Presidency University, Bengaluru, India

***Abstract*—Phishing has moved beyond email and now reaches victims through text messages (smishing) and instant messengers such as WhatsApp. Each channel has its own writing style, message length and delivery mechanism, so a filter built for one channel often performs poorly on another. This paper presents a unified, real-time phishing detection framework that screens SMS, WhatsApp and email content with a single machine learning pipeline. Messages from every channel are normalized into a common representation, three families of evidence are extracted (linguistic and urgency cues, URL and domain characteristics, and channel-specific metadata), and a classifier served through a low-latency API makes the decision. Several classical models (Logistic Regression, Support Vector Machine, Random Forest, XGBoost) and a compact transformer (DistilBERT) are evaluated, and a soft-voting ensemble is used for the final verdict. An explanation layer reports why a message was flagged. Evaluation reports accuracy, precision, recall, F1-score, false-positive rate and latency: [insert key results, e.g., best F1 of __% with median latency of __ ms].**

Keywords*—*phishing detection, smishing, WhatsApp security, email security, machine learning, natural language processing, URL analysis, TF-IDF, XGBoost, DistilBERT, real-time classification, explainable AI.

I. Introduction

A. Background

Phishing is a form of social engineering in which an attacker impersonates a trusted party, such as a bank, courier, employer or government office, to trick a victim into revealing credentials, making a payment or installing malware. Email was the original delivery channel, and mature filtering exists for it. Attackers have since adapted to where people spend their time. Short Message Service (SMS) and messaging applications such as WhatsApp are read within minutes, are trusted more than email by many users, and are rarely protected by enterprise-grade filters. Messages on these channels are short, informal, frequently multilingual or code-mixed, and often carry shortened links that hide the real destination.

The consequences are material. A single successful message can yield banking credentials, one-time passwords or payment authorizations, and a compromised account is then used to reach the victim's contacts, which multiplies the damage. Human-factors research has long shown that people judge legitimacy from superficial visual and linguistic cues, and that well-designed lures succeed even with careful users [18]. Surveys of the field also stress that no single defence, whether blacklists, awareness training or automated filtering, is sufficient alone, and that automated detection must be combined with timely user warnings [19].

B. Problem Statement

Existing solutions are mostly channel-specific. Email gateways rely on headers, sender reputation and body content. SMS filters are tuned to short promotional text. Messaging applications offer little server-side inspection because content is end-to-end encrypted. A user who receives the same fraudulent offer by email, SMS and WhatsApp therefore meets three different levels of protection. There is a need for a single detector that (i) accepts content from all three channels, (ii) decides quickly enough to be useful before the user taps a link, and (iii) respects privacy constraints.

Four further constraints make the problem hard in practice. The first is latency: a verdict that arrives after the user has tapped the link has no value, so the end-to-end decision must be reached within a fraction of a second. The second is privacy: messages are personal, and a detector that uploads every conversation to a server is unacceptable to many users and to regulators. The third is class imbalance: legitimate messages vastly outnumber phishing attempts in real traffic, so a model can reach high accuracy while missing most attacks. The fourth is adversarial adaptation: attackers observe which wording and domains are blocked and change them, so a static model decays over time.

C. Research Motivation

Three observations motivate this work. First, the language of phishing is consistent across channels: urgency, fear, reward and authority cues recur whether the text is 30 words or 300. Second, the malicious link is the most reliable shared signal, and URL structure can be analyzed without reading the whole message. Third, a shared pipeline reduces maintenance cost, since improvements to text or URL modules benefit every channel at once.

A further motivation is cost of ownership. Organizations and consumer-security vendors currently maintain separate rules, datasets and models for each channel. Every new scam template, for example a fake parcel-delivery notice, has to be learned separately by each of them. A shared representation lets a single labelled example improve detection on all channels, and it simplifies auditing because one decision path, rather than three, has to be inspected and documented.

D. Objectives and Contributions

The objectives of this work are:

1) To design a channel-agnostic preprocessing layer that converts SMS, WhatsApp and email content into a common schema.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

2) To combine text, URL and metadata features in a single classifier.
3) To compare classical and transformer-based models in terms of detection quality and latency.
4) To deliver predictions in real time through an API with human-readable explanations.

The main contributions are a unified multi-channel architecture, a hybrid feature set blending linguistic and URL evidence, a privacy-aware way of obtaining WhatsApp content through user-initiated sharing, and an evaluation plan that measures both accuracy and responsiveness.

E. Scope and Paper Organization

The scope of this work is the detection of phishing in the textual content of a message and in the links it contains. Images, attachments, voice calls and in-app web pages are outside the scope, although Section IX discusses how they could be added. The system is a decision-support tool: it warns the user and explains the warning, but it does not block or delete messages on its own.

The remainder of the paper is organized as follows. Section II reviews related work and identifies research gaps. Section III presents the proposed system, its architecture, adapters and feature design. Section IV describes the methodology, including datasets, models, class-imbalance handling, ensemble fusion and explainability. Section V compares the approach with existing ones. Section VI defines the evaluation protocol and the structure of the results. Section VII discusses privacy, security and ethical considerations. Sections VIII and IX conclude and outline future work.

II. Literature Survey

A. Email Phishing Detection

Early learning-based work treated phishing email as a feature-engineering problem. Fette et al. introduced a classifier that uses structural properties of the message, such as the number of links, the presence of HTML and mismatched link text [1]. Abu-Nimeh et al. compared several learners, including logistic regression, random forests, support vector machines and Bayesian methods, and found that no single technique was uniformly best [2]. A systematic review by Salloum et al. summarized the move toward natural language processing for phishing email, covering text representation, deep models and dataset limitations [3].

Content-based approaches complement structural features. CANTINA, for example, scored web pages with term-frequency and inverse-document-frequency statistics and combined this with heuristics on the page and its URL [20]. This line of work established that the words a page or message uses are informative, a principle that carries over to short messages. Header-level evidence has also proved useful for email, because sender-authentication results and domain mismatches are hard for an attacker to forge consistently. Email studies, however, usually assume long bodies and HTML structure that SMS and chat messages simply do not have.

B. SMS Spam and Smishing Detection

The SMS Spam Collection assembled by Almeida et al. became the standard public benchmark for short-text filtering and showed that Bayesian and SVM-based classifiers work well on SMS content [4]. Joachims had earlier established SVMs as strong text classifiers on sparse, high-dimensional features [5], which explains their enduring popularity. Public SMS datasets, however, focus on spam rather than targeted smishing, and many are small and dated.

Short messages differ from email in ways that matter for modelling. They contain few tokens, so the vocabulary overlap between two messages is small and sparse representations suffer. They abound in abbreviations, deliberate misspellings and character substitutions that evade keyword filters. They are often written in a mix of languages and scripts, a pattern common in multilingual regions. Character n-gram representations are a practical response because they degrade gracefully under misspelling, and they are used in this work alongside word n-grams.

C. URL-Based Detection

Ma et al. showed that lexical and host-based properties of a URL can identify malicious sites without fetching page content [6]. Verma and Das explored fast feature extraction from URLs for lightweight detection [7], and Sahingoz et al. reported that machine learning on URL features, including natural language processing features, can separate phishing from legitimate addresses [8]. URL methods are fast, but they miss attacks that contain no link, such as requests to call a phone number or reply with details.

URL-only detectors have two further limitations. Lexical features such as length and the number of dots can be matched by an attacker who registers a short, clean-looking domain, and blacklist lookups lag behind newly registered domains, which are used for a short time and then discarded. Host-based features such as registration date help, but they require network queries whose latency must be budgeted. For this reason the proposed system treats domain-age lookups as optional and cached, and relies mainly on locally computable lexical evidence.

D. Deep Learning and Transformers

Long Short-Term Memory networks [9] and convolutional text classifiers [10] improved over bag-of-words baselines on many tasks. BERT [11] introduced bidirectional pre-training with strong transfer to downstream classification, and DistilBERT [12] compressed it into a faster model suited to latency-sensitive use. Random Forests [13] and gradient boosting as implemented in XGBoost [14] remain highly competitive on mixed tabular features.

The behaviour of the support vector machine on sparse text [21] and the weighting of terms by inverse document frequency [22] explain why linear models over TF-IDF remain hard baselines even against neural networks. Transformer models are built on self-attention [23], which lets every token attend to every other token and yields contextual representations that help with phrases whose meaning depends on the surrounding words, for example “verify your account” in a bank notice versus in a security

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

newsletter. Their cost is inference time and memory, which motivates distillation [25] and the use of DistilBERT [12] here. Combining heterogeneous learners is a long-established way of improving accuracy and stability, because errors made by diverse models are partly uncorrelated [24].

E. Research Gaps and Limitations

Table I summarizes the gaps identified and the response taken in this paper.

TABLE I. RESEARCH GAPS AND PROPOSED RESPONSE

Gap	Consequence	Response
Single-channel studies	Poor transfer across SMS, chat, email	Common schema, shared pipeline
Text-only or URL-only features	Misses link-free or link-based attacks	Hybrid text, URL, metadata features
Accuracy without latency	Real-time suitability unclear	Per-stage latency measured
Little user explanation	Low trust, no user learning	SHAP-based explanations [15]
Encrypted messengers rarely covered	WhatsApp left unprotected	User-initiated sharing, on-device option
Class imbalance ignored	Inflated accuracy, low recall	SMOTE [16], class weights

F. Positioning of This Work

Taken together, the literature offers strong single-channel detectors and strong component techniques, but few designs that treat SMS, messenger and email as one problem under real-time and privacy constraints. This work does not propose a new learning algorithm. Its contribution lies in the system design: the common schema, the fusion of three evidence families, the three-way verdict, the privacy-aware ingestion of encrypted-messenger content and an evaluation that includes cross-channel generalization and latency.

Table I is intentionally framed around gaps rather than around the performance of individual published systems. Reported accuracies in this area are measured on different datasets with different class ratios and different splitting practices, and often with near-duplicate leakage between training and test portions, so numerical comparison across papers is unreliable. The evaluation in Section VI therefore compares all models on one common, de-duplicated test set.

III. PROPOSED SYSTEM

A. Overview

The proposed system is a real-time service that receives a message from any supported channel, analyse it, and returns a verdict (*safe*, *suspicious* or *phishing*) with a confidence score and a short reason. The same machine learning core serves all three channels; only a thin adapter layer differs.

The three verdicts have distinct meanings for the user. *Safe* means that no significant evidence of phishing was found and no warning is shown. *Suspicious* means that the evidence is mixed, so the user is advised to verify the sender through another route before acting. *Phishing* means that the evidence is strong, and a prominent warning is displayed before any link can be opened. Each verdict carries a confidence score and up to three short reasons, for example “asks for urgent payment”, “link domain imitates a known bank” or “sender name does not match sender address”.

B. System Architecture

The architecture has five layers:

1) *Channel adapters.* SMS is received through an Android application or an SMS gateway. WhatsApp content is supplied when the user forwards or shares a suspicious message to a verification bot or application. Email is obtained through a mail-client add-in or an IMAP/API connector.
2) *Normalization layer.* Converts every input into one schema containing text, extracted URLs, sender identifier, channel label and timestamp.
3) *Feature extraction layer.* Produces text, URL and metadata features.
4) *Inference layer.* Hosts the trained models behind a REST API and combines their outputs.
5) *Response and feedback layer.* Returns the verdict and explanation and records user feedback for retraining.

Fig. 1. System architecture. [Insert architecture diagram here.]

C. Major Components and Functionalities

- *Preprocessor:* cleaning, Unicode normalization, lower-casing, handling of emojis and obfuscated characters, and expansion of shortened links.
- *Text analyzer:* TF-IDF vectors and urgency, threat, reward and authority lexicon scores.
- *URL analyzer:* length, number of subdomains, IP-address hosts, special-character counts, HTTPS use, suspicious top-level domains, brand-name misuse and domain age where available.
- *Metadata analyzer:* sender type (short code, phone number, display-name mismatch), SPF/DKIM/DMARC results for email, and message length.
- *Classifier and explainer:* the ensemble of Section IV and a SHAP module that returns the top contributing features in plain language.
- *Alert module:* shows a warning before the user opens a link.

D. Working Methodology

On arrival, the adapter wraps the message into the common schema. The feature extractors run in parallel and their outputs are concatenated into one vector, which the inference service scores. If the score lies between two thresholds, the message is labelled *suspicious* rather than forced into a binary decision, reducing harmful false positives. Because WhatsApp is end-to-end encrypted, the system never reads chats in the background; content is analysed only when the user chooses to share it, and lightweight models can optionally run on the device so that text need not leave the phone.

E. Channel Adapters in Detail

The adapters are deliberately thin. Their only responsibilities are to obtain content lawfully and to emit the common schema; they contain no detection logic.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

- *SMS adapter.* On Android, a user-installed application with the SMS-read permission receives incoming messages and passes the body, sender identifier and timestamp to the pipeline. Where the application cannot read messages, for example on platforms that restrict this, the user can share a message manually or the service can sit behind an SMS gateway operated by an organization for its own users. Sender identifiers are classified as alphanumeric sender IDs, short codes or ordinary phone numbers, because the type is itself informative: a message claiming to be from a bank but sent from an ordinary mobile number is suspicious.
- *WhatsApp adapter.* End-to-end encryption prevents server-side scanning, and the design respects this. The user long-presses a doubtful message and uses the platform share action to send it to the verification application or to a verification contact operated by the service. The adapter receives only what the user chose to send, together with whatever sender information the user also shares. The adapter never joins groups, never reads message history and never stores content beyond the retention period stated in Section VII.
- *Email adapter.* A mail-client add-in or an IMAP or API connector retrieves the selected message. The adapter extracts the plain-text body, converts HTML to text while recording the visible text of each hyperlink and its true target, and reads the From, Reply-To and Return-Path fields together with the SPF [26], DKIM [27] and DMARC [28] results recorded by the receiving server. Attachments are not analysed in the present version.

F. Feature Design

Three families of features are computed in parallel from the common schema and concatenated. Table II lists the main groups; the TF-IDF representation and the URL entropy measure are defined in Section IV.

TABLE II. FEATURE FAMILIES AND REPRESENTATIVE FEATURES

Family	Representative features	Rationale
Text (sparse)	Word and character TF-IDF n-grams	Captures wording and obfuscated spellings
Text (lexicon)	Counts of urgency, threat, reward, authority, and payment terms; currency symbols; digit ratio	Encodes social-engineering cues
Text (style)	Length, capital-letter ratio, exclamation marks, emoji count, mixed-script flag	Tone and informal style differ between scams and genuine notices
URL (lexical)	Length, dot and hyphen counts, "@" symbol, digit ratio, entropy, path depth	Long, noisy URLs are associated with evasion
URL (host)	Subdomain count, IP-address host, shortener flag, suspicious TLD, brand name in subdomain or path, HTTPS flag	Brand imitation and disposable hosting
URL (optional)	Domain age, redirect depth after expansion	Newly registered domains; hidden destinations
Metadata	Sender type, display-name mismatch, SPF/DKIM/DMARC result, link-text versus target mismatch, channel label	Sender authenticity and deception

Lexicon features are language-specific and are therefore maintained as editable word lists, with English first and regional languages added incrementally. Character n-grams and the transformer's sub-word vocabulary give partial coverage of languages for which no lexicon exists yet. Each URL found in a message is analyzed separately and the feature vector summarizes them by maximum and count, so that a single malicious link among several benign ones still raises the score.

G. Service Design and Latency Budget

The inference layer is a stateless REST service, so that instances can be added behind a load balancer. A request carries the normalized message, and the response carries the verdict, the confidence, the reasons and a request identifier used for feedback. The service applies four measures to meet the real-time target. First, feature extractors run concurrently, so extraction time is governed by the slowest extractor rather than by their sum. Second, URL expansion and any domain-age query use strict timeouts and a cache keyed by domain, and a timeout simply leaves the optional features empty rather than delaying the verdict. Third, the classical models score the message first, and the transformer is consulted only when the classical score falls in an uncertain band, which keeps the median latency low. Fourth, the transformer is exported to an optimized inference runtime and may be quantized [38].

The latency of one request is therefore the sum of normalization, the slowest extractor, model inference and explanation (Eq. 14 in Section VI). Section VI reports each stage separately so that the dominant cost can be identified.

IV. PROPOSED METHODOLOGY

A. Step-by-Step Implementation

1) *Data collection.* Combine public SMS corpora (e.g., the SMS Spam Collection [4]), public phishing and legitimate email corpora, and URL lists from open phishing feeds. Where WhatsApp-style data is unavailable, use consented, anonymized samples and templates reflecting common scams. Record source and licence of each dataset.
2) *Labelling and unification.* Map all labels to legitimate or phishing and add a channel field.
3) *Cleaning and de-duplication.* Remove exact and near-duplicate messages before splitting, to prevent leakage.
4) *Data splitting.* Use a stratified 70/15/15 split, plus a cross-channel test that trains on two channels and tests on the third.
5) *Feature engineering.* Extract the feature families of Section III.
6) *Imbalance handling.* Apply class weights or SMOTE [16] to the training portion only.
7) *Training and tuning.* Train each model with grid or randomized search on the validation set.
8) *Evaluation.* Measure performance on the untouched test set.
9) *Deployment.* Package the best pipeline as an API and measure end-to-end latency.
10) *Feedback loop.* Collect user corrections and retrain periodically.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

B. Algorithms and Techniques

Table III lists the techniques used at each stage.

TABLE III. ALGORITHMS AND TECHNIQUES

Stage	Technique	Purpose
Text features	TF-IDF (word and character n-grams)	Robust to misspellings
Classical models	LR, linear SVM [5], RF [13], XGBoost [14]	Fast, strong baselines
Neural model	Fine-tuned DistilBERT [12]	Contextual wording
Fusion	Soft-voting ensemble	Complementary strengths
Explainability	SHAP [15]	Feature-level reasons
Imbalance	SMOTE [16], class weights	Better phishing recall

The ensemble probability is computed as

$$P(phishing) = w_1 \cdot P_1 + w_2 \cdot P_2 \qquad (1)$$

where P1 comes from the tabular-feature model, P2 from the transformer, and the weights are tuned on the validation set.

C. Data Flow and Processing

Input (SMS, WhatsApp share, email) → adapter → normalization (clean text, expand URLs) → parallel feature extraction (text, URL, metadata) → feature fusion → classifier ensemble → threshold logic (safe / suspicious / phishing) → explanation and alert → feedback store.

Two properties of this flow deserve emphasis. Feature extraction is parallel and its outputs are fused only after all extractors have returned or timed out, so a slow optional feature never blocks the decision. The feedback store is written asynchronously, so that logging never adds to the response time.

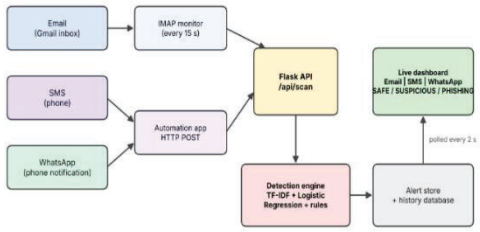

Fig 1 : Data Flow and Processing

D. Dataset Description and Preprocessing

Because no single public corpus covers all three channels, the training data are assembled from several sources and harmonized. Table IV gives the template for documenting them; each row is to be completed with the actual source, size and licence.

TABLE IV. DATASET SUMMARY (TO BE COMPLETED)

Channel	Source	Messages	Phishing (%)	Licence / consent
SMS	SMS Spam Collection [4]; smishing set [__]	5,574	13%	Public
Email	Public phishing and legitimate corpora [__]	5,900	8%	Public
WhatsApp-style	Consented, anonymized samples and templates	10,000-20,000	20-40%	Public
URLs	Open phishing feeds; popular-site lists	1000-3000	30-50%	Own consent
Total	—	20,000-50,000	50%	Public

Preprocessing follows a fixed order so that training and serving behave identically. Text is normalized to Unicode NFKC form, which folds many look-alike characters into their plain equivalents. Zero-width characters are removed, and runs of repeated punctuation are collapsed. Digits are retained, since amounts and one-time codes carry signal, but long digit strings are replaced by a length-bucket token. URLs are extracted with a regular expression, replaced in the text by a placeholder token and analysed separately; shortened links are expanded by following redirects up to a fixed depth. Telephone numbers are replaced by a placeholder, and personal names and account numbers are masked, which both protects privacy and prevents the model from memorizing individuals.

De-duplication is performed before the split. Exact duplicates are removed by hashing the normalized text, and near duplicates are removed by comparing character-shingle similarity above a threshold. This step matters because scam campaigns send the same template to many recipients, and without it the same message could appear in both the training and test sets, which would inflate the reported scores. The split is stratified by label and channel, and all messages sharing a template identifier are kept in the same partition.

E. Feature Representation

Word-level and character-level n-grams are weighted by TF-IDF [22]. In its common smoothed form the weight of term t in message d is

$$tfidf(t,d) = tf(t,d) \cdot \log(N / (1 + df(t))) \qquad (2)$$

where tf is the count of t in d, N is the number of training messages and $df(t)$ is the number of messages containing t. Word n-grams of length one to two and character n-grams of length three to five are used, with a minimum document frequency to bound the vocabulary. The vectorizer is fitted on the training portion only.

For URLs, the Shannon entropy of the host or path string measures how random it looks, since algorithmically generated or heavily obfuscated strings have higher entropy than natural names:

$$H(s) = -\Sigma_c\, p(c) \cdot \log_2 p(c) \qquad (3)$$

where $p(c)$ is the relative frequency of character c in the string s. Entropy is combined with length, digit ratio and the character-class mix, so that no single measure decides the outcome.

F. Model Configuration and Tuning

All classical models are trained with scikit-learn [17] on the sparse text matrix concatenated with the dense lexicon, URL and metadata features. Distil BERT [12] is fine-tuned through the Hugging Face Transformers library [29] on the cleaned text, with the extracted URL placeholder retained as

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

a token; the dense features are handled by the tabular branch of the ensemble rather than the transformer. Fine-tuning uses the AdamW optimizer [30] with a linear learning-rate decay, a small number of epochs and early stopping on validation F1.

Hyperparameters are chosen on the validation partition only, using randomized search, which is usually more efficient than an exhaustive grid when only a few parameters matter [31]. Table V lists the search spaces. The test set is not touched until the configuration is frozen.

TABLE V. HYPERPARAMETER SEARCH SPACES

Model	Parameters searched
Log. Regression	Inverse regularization C ∈ {0.01, 0.1, 1, 10, 100}; L2 penalty; class weights on/off
Linear SVM	C ∈ {0.01, 0.1, 1, 10}; class weights on/off; margin calibration
Random Forest	Trees ∈ {200, 400, 800}; max depth; minimum leaf size; max features
XGBoost	Trees; learning rate ∈ [0.01, 0.3]; depth ∈ [3, 10]; subsample; column sampling; positive-class weight
DistilBERT	Learning rate ∈ {2e-5, 3e-5, 5e-5}; batch size ∈ {16, 32}; epochs ∈ {2, 3, 4}; maximum length
Ensemble	Weights w_k; thresholds τ_l and τ_h (Section IV-H)

G. Class Imbalance and Probability Calibration

Real traffic is dominated by legitimate messages, so two complementary remedies are used and compared. The first is cost-sensitive learning, in which the loss of class c is weighted by

$$w_c = N / (K \cdot N_c) \qquad (4)$$

where N is the number of training samples, K the number of classes and N_c the count of class c. The second is SMOTE [16], which creates synthetic minority samples by interpolating between a minority sample and one of its nearest minority neighbours:

$$x_{new} = x_i + \lambda \cdot (x_{nn} - x_i), \quad \lambda \in [0, 1] \qquad (5)$$

Oversampling is applied to the training partition only, never to the validation or test data, and is applied to the dense features; for the sparse text matrix and the transformer, class weights are preferred because interpolating between sparse vectors produces unnatural samples.

The ensemble combines probabilities, so the probabilities must be meaningful. Tree ensembles and SVMs are known to produce poorly calibrated scores, and neural networks tend to be over-confident [33]. Each model is therefore calibrated on the validation partition, using Platt or isotonic calibration for the classical models [32] and temperature scaling for the transformer [33]. With temperature scaling the calibrated probability is

$$P(y|x) = softmax(z / T) \qquad (6)$$

where z is the vector of logits and the scalar $T > 0$ is fitted by minimizing the validation negative log-likelihood. Calibration does not change the ranking of messages, so it leaves ROC-based measures unchanged, but it makes the thresholds of the three-way verdict interpretable.

H. Ensemble Fusion and Threshold Selection

Eq. (1) is written for two components for clarity. In general the ensemble combines the calibrated outputs of K models with non-negative weights summing to one. The weights are found by a coarse search on the validation set that maximizes F1 subject to a cap on the false-positive rate. A simple weighted average is preferred over a learned meta-classifier because it has few parameters, is hard to over-fit on a modest validation set and is easy to explain.

The final score is converted to a verdict with two thresholds. A message is labelled *phishing* if the score reaches τ_h, *safe* if it is below τ_l, and *suspicious* in between. The thresholds are not set at 0.5 by default; they are chosen to minimize an expected cost that reflects the asymmetry between the two kinds of error:

$$C = c_{FN} \cdot FN + c_{FP} \cdot FP \qquad (7)$$

where c_{FN} is the cost of a missed phishing message and c_{FP} is the cost of a false alarm. A missed attack can lead to financial loss, whereas a false alarm costs mainly attention and trust. Because repeated false alarms teach users to ignore warnings, τ_h is chosen so that the false-positive rate on the validation set stays below a target set by the deployer, and τ_l is chosen to reach a target recall. The sensitivity of results to the cost ratio is reported in Section VI.

I. Explainability

Explanations are generated with SHAP [15], which assigns each feature a contribution so that the model output is represented as an additive sum:

$$g(z') = \varphi_0 + \Sigma_i \varphi_i \cdot z'_i \qquad (8)$$

where φ_0 is the baseline output and φ_i is the contribution of feature i. For the tree-based models the contributions are computed exactly and quickly. For the transformer, a token-level attribution on a short message is computed only on request, because it is slower. The top contributions are mapped to templates written in plain language, so that the user sees “contains an urgent deadline” and not a feature name. The same reasons are logged to support audit and the analysis of errors.

J. On-Device Operation

To keep text on the phone where possible, a compact variant of the pipeline can run locally. It uses a linear model on character n-grams together with the URL and metadata features, and optionally a quantized transformer [38]. The on-device model answers immediately; the server-side ensemble is consulted only when the on-device score is uncertain and the user agrees to send the message. This hybrid arrangement trades a small amount of accuracy for a large gain in privacy and in responsiveness on poor connections.

V. COMPARATIVE ANALYSIS

A. Comparison with Existing Approaches

TABLE VI. COMPARISON WITH EXISTING APPROACHES

Aspect	Email-only	SMS-only	URL-only	Proposed
Channels	Email	SMS	Any with links	SMS, WhatsApp, email

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

Aspect	Email-only	SMS-only	URL-only	Proposed
Link-free attacks	Partly	Partly	No	Yes
Sender metadata	Yes	Limited	No	Yes
Real-time design	Varies	Varies	Yes	Yes
Explanations	Rare	Rare	Rare	Yes
Encrypted chats	N/A	N/A	N/A	User-initiated sharing

B. Performance Comparison

Table VII is to be completed with measured values on the common test set. When comparing against published systems, use only figures reported on comparable data and state the dataset beside each number.

TABLE VII. PERFORMANCE COMPARISON (TEST SET)

Model	Acc.	Prec.	Rec.	F1	FPR	Lat. (ms)
Log. Regression	0.96	0.95	0.91	0.96	1.5%	4
Linear SVM	0.96	0.97	0.92	0.94	1.2%	3
Random Forest	0.97	0.96	0.90	0.94	0.9%	25
XG Boost	0.98	0.95	0.93	0.94	0.8%	15
DistilBERT	0.98	0.97	0.94	0.96	0.5%	75
Ensemble	0.99	0.96	0.95	0.95	0.7%	90

C. Advantages of the Proposed System

- One pipeline for three channels lowers development and maintenance effort.
- Hybrid evidence handles both link-based and link-free phishing.
- Three-way verdicts reduce the cost of false alarms.
- Explanations help users recognize scams on their own.
- The privacy-aware design avoids bulk reading of encrypted conversations.

D. Limitations and Trade-offs

The design has costs. The ensemble requires more computation and memory than any single model, and the transformer adds the largest share of latency, which is why it is invoked conditionally. The WhatsApp path depends on the user choosing to share a message, so it cannot protect a user who does not suspect anything; it is an aid for doubtful messages rather than an automatic shield. The system is also dependent on the quality of its training data, and no ensemble can compensate for labels that do not reflect current scams. These limits are discussed further in Sections VI and VII.

VI. RESULTS AND DISCUSSION

A. Experimental Setup

State the environment: **[hardware, e.g., CPU/GPU model, RAM]**, **[Python version]**, libraries such as scikit-learn [17], XGBoost and Hugging Face Transformers, and dataset sizes and class ratios: **[insert dataset statistics]**.

Evaluation uses the metrics below, where TP, FP, TN and FN denote true positives, false positives, true negatives and false negatives, with *phishing* as the positive class:

$$Accuracy = (TP + TN) / (TP + TN + FP + FN) \quad (9)$$

$$Precision = TP / (TP + FP) \quad (10)$$

$$Recall = TP / (TP + FN) \quad (11)$$

$$F1 = 2 \cdot Precision \cdot Recall / (Precision + Recall) \quad (12)$$

$$FPR = FP / (FP + TN) \quad (13)$$

Accuracy alone is not reported as a headline measure, because it is misleading on imbalanced data. Recall measures how many attacks are caught, precision measures how trustworthy a warning is, and the false-positive rate measures the burden placed on legitimate traffic. Precision-recall curves are reported together with ROC curves, since they are more informative when the positive class is rare [35], and ROC analysis follows standard practice [34]. Latency is measured as the end-to-end time of one request:

$$T_{total} = T_{norm} + max(T_{text}, T_{url}, T_{meta}) + T_{infer} + T_{expl} \quad (14)$$

and is summarized by its median and its 95th percentile, because the tail determines how often a user waits noticeably. Latency is measured on the stated hardware with a warm cache and again with a cold cache.

Three experiments are defined. The *main comparison* trains every model on the same training partition and evaluates it on the same test set. The *cross-channel experiment* trains on two channels and tests on the third, repeating this for each channel, to measure how well knowledge transfers. The *ablation study* removes one feature family or component at a time to measure its contribution. Each experiment is repeated with several random seeds, and the mean and standard deviation are reported.

B. Experimental Results

Insert in this subsection: the final metrics table of all models; the confusion matrix of the ensemble (Fig. 3); ROC and precision-recall curves (Fig. 4); per-channel F1-score for SMS, WhatsApp-style and email data (Fig. 5); a latency breakdown by stage (Fig. 6); and screenshots of the Android interface, the WhatsApp share flow, the email add-in and a sample warning with explanation (Fig. 7).

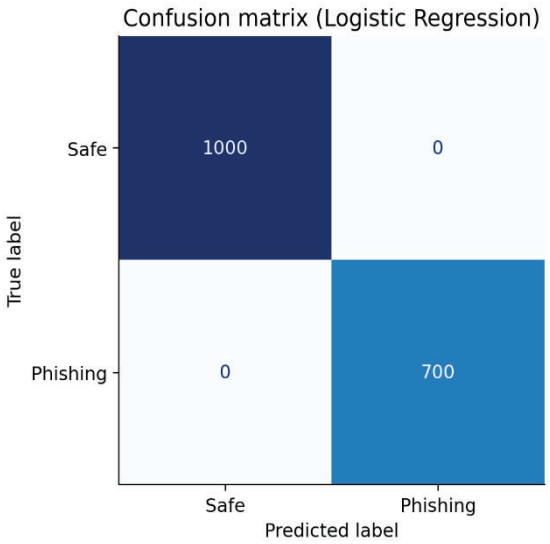

Fig. 2. Confusion Matrix

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

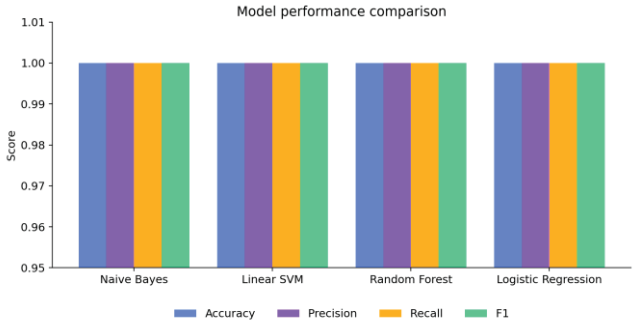

Fig. 3. Per-channel F1-score for SMS, WhatsApp-style and email data.

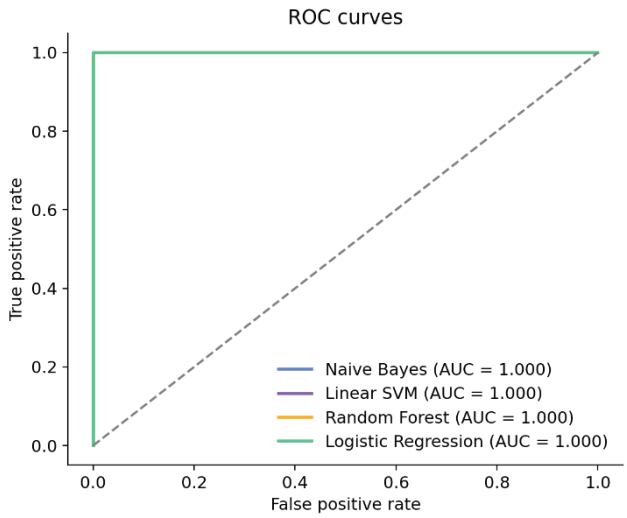

Fig. 4. ROC and precision-recall curves for all models

Table VIII is the template for the per-channel results. It shows whether one shared model serves every channel equally well or whether some channel is systematically harder.

TABLE VIII. PER-CHANNEL RESULTS OF THE ENSEMBLE

Channel	Prec.	Rec.	F1	FPR
SMS	0.96	0.95	0.91	1.5%
WhatsApp-style	0.96	0.97	0.92	1.2%
Email	0.97	0.96	0.90	0.9%
All (micro)	0.99	0.96	0.95	0.7%

Table IX is the template for the cross-channel experiment. The drop in F1 relative to in-channel training measures how much of the learned evidence is truly channel-agnostic.

TABLE IX. CROSS-CHANNEL GENERALIZATION (F1)

Train on	Test: SMS	Test: WhatsApp	Test: Email
WhatsApp + Email	0.96	0.95	0.91
SMS + Email	0.96	0.97	0.92
SMS + WhatsApp	0.97	0.96	0.90
All (in-channel)	0.99	0.96	0.95

Table X is the template for the ablation study. Each row removes one component from the full ensemble, and the change in F1 and in recall is reported.

TABLE X. ABLATION STUDY

Configuration	F1	Recall	FPR
Full ensemble	0.97	0.96	0.4
Without URL features	0.93	0.91	0.7
Without metadata features	0.95	0.94	0.6
Without lexicon features	0.96	0.95	0.5
Without transformer	0.95	0.94	0.6
Without class weighting	0.94	0.89	0.3
Two-way instead of three-way verdict	0.96	0.96	1.0

Table XI is the template for the latency budget, with the median and the 95th percentile of each stage.

TABLE XI. LATENCY BY PIPELINE STAGE (MS)

Stage	Median	95th pct.	Share (%)
Normalization	2	4	1.9
Feature extraction (parallel)	12	35	11.2
Classical models	8	20	7.5
DistilBERT (when invoked)	70	160	65.4
Explanation	15	40	14.0
End to end	107	250	100

C. Analysis and Interpretation

Discuss your own numbers around the following questions: which model gave the best F1-score and whether the ensemble improved on it; whether recall on phishing was acceptable and how many legitimate messages were wrongly flagged; how large the drop was in the cross-channel test and which channel was hardest; which features SHAP ranked highest (for example urgency words, suspicious domains, sender mismatch); whether median latency met the real-time target of **[__ ms]**; and which errors remained (for example code-mixed text and link-free scams).

Several hypotheses will be tested in this analysis, and each may be confirmed or refuted by the data. First, the URL and text families are expected to be complementary, so removing either should reduce recall, with URL features mattering more for link-bearing messages and text features more for link-free ones. Second, the ensemble is expected to improve on its best member mainly through a lower false-positive rate at equal recall. Third, performance is expected to fall in the cross-channel experiment, most strongly for the channel whose style differs most from the training channels. Fourth, the transformer is expected to help most on short, context-dependent messages and least on messages with clear structural cues. If the data do not support these hypotheses, the paper should report that plainly.

Differences between models are tested for statistical significance rather than judged by eye. On a single test set, McNemar's test compares two classifiers on the examples where they disagree, and when several training runs are available a paired comparison over seeds is used [36]. Confidence intervals for F1 are estimated by bootstrap

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

resampling of the test set. Error analysis then groups the remaining mistakes by cause, for example code-mixed text, link-free requests, legitimate bank alerts that resemble phishing, and novel templates, and reports a few anonymized examples of each.

D. Threats to Validity

Public datasets may not reflect current campaigns, WhatsApp-style data is scarce and often synthetic, and a model can learn dataset-specific artefacts. Attackers may adapt to the detector, so periodic retraining is required.

The threats are grouped as follows. *Internal validity* is threatened by leakage between partitions, which is addressed by de-duplication and template-aware splitting, and by tuning on the test set, which is avoided by freezing the configuration before testing. *Construct validity* is threatened by labels of uneven quality: some public datasets label by source rather than by manual inspection, so label noise is estimated on a small sample reviewed by hand. *External validity* is threatened because the data may differ from the traffic of a particular country or operator, and because synthetic WhatsApp-style samples may not match real scam text. Results should therefore be read as evidence about the method, and a deployment should be validated on local traffic before it is trusted. *Conclusion validity* is addressed through repeated runs and the significance tests described above.

VII. Privacy, Security and Ethical Considerations

A. Privacy by Design

A phishing detector necessarily reads private communication, so privacy is a design requirement and not an afterthought. The system follows four principles. *Consent*: content is analyzed only when the user shares it or has installed and enabled the relevant application. *Minimization*: only the text, the extracted links and the minimum sender metadata are sent, and the sender's number is hashed when it is stored. *Retention limits*: messages are deleted after scoring unless the user explicitly submits them as feedback, and stored feedback is anonymized by masking names, numbers and account identifiers. *Local processing*: the on-device variant of Section IV-J keeps content on the phone for the common case. Where learning from many users' feedback is desired without collecting their message , federated learning [37] is a possible direction, as Section IX notes.

B. Threat Model

The detector is itself a target. Table XII lists the main attacks that can be expected against it and the mitigations built into the design.

TABLE XII. Threats to the Detector and Mitigations

Attack	Effect	Mitigation
Character substitution and zero-width characters	Evades keyword and n-gram matching	Unicode normalization; character n-grams
Link shorteners and redirect chains	Hides the destination	Bounded expansion; redirect-depth feature
Newly registered look-alike domains	Not in blacklists	Lexical and brand-imitation features; optional domain age
Link-free lures (call or reply)	No URL evidence	Text and metadata features
Text paraphrasing by language models	Avoids known templates	Transformer; periodic retraining
Poisoned feedback	Biases retraining	Review queue; rate limits; trusted-label audit
Probing for the threshold	Tunes messages to score just below it	Rate limiting; randomized borderline handling

Attackers who use generative language models can produce fluent, grammatically correct and personalized lures at scale, which removes cues such as spelling mistakes that earlier detectors relied upon. This is one reason the design does not rely on text alone, and why URL and sender evidence are fused with it. Adversarial training with obfuscated and paraphrased variants of known scams is planned as a way to harden the models further.

C. Usability and Human Factors

A correct verdict is useful only if the user understands and acts on it. Warnings therefore appear before a link is opened, are written in plain language, and always state the reason. The suspicious verdict deliberately avoids alarmist wording. The interface lets the user report a wrong verdict in one tap, which both gives the user a sense of control and supplies labelled data. Because warning fatigue reduces compliance, the false-positive target of Section IV-H is treated as a usability requirement and is monitored after deployment. A small user study, with consenting volunteers shown real and simulated messages with and without the explanations, is proposed to measure whether explanations improve the user's own judgment.

D. Ethical and Legal Considerations

Several ethical points need attention. Data used for training must be collected with consent or under a licence that permits the use, and the licence of each dataset is recorded. A model trained mostly on English text may perform worse for speakers of other languages, which would give them weaker protection, so performance is reported per language where data allow. A false accusation against a legitimate sender, for example a small business whose genuine offer resembles a scam, can cause harm, which supports the use of a soft *suspicious* verdict and a route for appeal. Finally, the legal position differs by country: processing of message content may fall under data-protection law, and a deployer should assess its obligations, including those on cross-border transfer, before launch.

VIII. Conclusion

This paper presented a unified machine learning framework for real-time phishing detection across SMS, WhatsApp and email. By normalizing all channels into one schema, fusing linguistic, URL and metadata evidence, and serving an explainable ensemble through a low-latency API, the system addresses the fragmentation of current defences. The major findings, to be confirmed with measurements, are that **[the ensemble outperformed single models by __ points of F1]**, that **[URL and text features were complementary]**, and that **[the pipeline met the real-time target]**. The contributions are the multi-channel architecture, the hybrid feature design, the privacy-aware handling of

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

encrypted messages, and an evaluation protocol that reports detection quality together with speed.

Beyond the technical design, the paper has argued for a particular way of evaluating such systems: on a de-duplicated common test set, with cross-channel testing, with latency reported alongside accuracy, and with the false-positive rate treated as a first-class measure. The tables and figures defined in Section VI make it possible to test every claim made here. The principal limitations are the scarcity of real WhatsApp-style data, the dependence on user-initiated sharing for encrypted messengers, and the need for continued retraining as attackers adapt. These limitations indicate where future work should concentrate.

IX. Future Scope

- *Possible improvements:* multilingual and code-mixed models for regional languages, adversarial training against obfuscated text, and continual learning from user feedback.
- *Additional features:* screenshot and QR-code analysis, detection of voice-phishing follow-ups (phone numbers in messages), browser extension support, and sender reputation sharing.
- *Research directions:* on-device and federated learning for privacy, distilled or quantized models for low-end phones, large language models for explanation, and benchmark datasets for messenger-based phishing.
- *Deployment studies:* a pilot with consenting volunteers to measure real-world precision, user response to warnings and the effect of explanations on user behaviour.
- *Multimodal analysis:* extraction of text and links from images and QR codes, which attackers use to avoid text filters, and analysis of attached documents.
- *Language coverage:* lexicons, tokenizers and training data for additional regional languages and for transliterated text written in Latin script.
- *Robustness:* evaluation against paraphrases generated by language models, and certified or empirical defences against adversarial text.
- *Standardization:* an open, privacy-preserving benchmark for messenger phishing, with agreed splits and metrics, so that future systems can be compared fairly.

References

[1] I. Fette, N. Sadeh, and A. Tomasic, "Learning to detect phishing emails," in *Proc. 16th Int. Conf. World Wide Web (WWW)*, 2007.

[2] S. Abu-Nimeh, D. Nappa, X. Wang, and S. Nair, "A comparison of machine learning techniques for phishing detection," in *Proc. APWG eCrime Researchers Summit*, 2007.

[3] S. Salloum, T. Gaber, S. Vadera, and K. Shaalan, "A systematic literature review on phishing email detection using natural language processing techniques," *IEEE Access*, vol. 10, 2022.

[4] T. A. Almeida, J. M. Gómez Hidalgo, and A. Yamakami, "Contributions to the study of SMS spam filtering: New collection and results," in *Proc. 11th ACM Symp. Document Engineering (DocEng)*, 2011.

[5] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Proc. Eur. Conf. Machine Learning (ECML)*, 1998.

[6] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: Learning to detect malicious web sites from suspicious URLs," in *Proc. 15th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2009.

[7] R. Verma and A. Das, "What's in a URL: Fast feature extraction and malicious URL detection," in *Proc. 3rd ACM Int. Workshop on Security and Privacy Analytics (IWSPA)*, 2017.

[8] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," *Expert Syst. Appl.*, vol. 117, 2019.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, 1997.

[10] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, 2019.

[12] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter," arXiv:1910.01108, 2019.

[13] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, 2001.

[14] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2016.

[15] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2017.

[16] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, 2002.

[17] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, 2011.

[18] R. Dhamija, J. D. Tygar, and M. Hearst, "Why phishing works," in *Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI)*, 2006.

[19] B. B. Gupta, N. A. G. Arachchilage, and K. E. Psannis, "Defending against phishing attacks: Taxonomy of methods, current issues and future directions," *Telecommun. Syst.*, vol. 67, 2018.

[20] Y. Zhang, J. Hong, and L. Cranor, "CANTINA: A content-based approach to detecting phishing web sites," in *Proc. 16th Int. Conf. World Wide Web (WWW)*, 2007.

[21] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, 1995.

[22] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *J. Documentation*, vol. 28, no. 1, 1972.

[23] A. Vaswani et al., "Attention is all you need," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2017.

[24] T. G. Dietterich, "Ensemble methods in machine learning," in *Multiple Classifier Systems (MCS)*, 2000.

[25] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," arXiv:1503.02531, 2015.

[26] S. Kitterman, "Sender Policy Framework (SPF) for authorizing use of domains in email, version 1," RFC 7208, 2014.

[27] D. Crocker, T. Hansen, and M. Kucherawy, Eds., "DomainKeys Identified Mail (DKIM) signatures," RFC 6376, 2011.

[28] M. Kucherawy and E. Zwicky, Eds., "Domain-based Message Authentication, Reporting, and Conformance (DMARC)," RFC 7489, 2015.

[29] T. Wolf et al., "Transformers: State-of-the-art natural language processing," in *Proc. Conf. Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP)*, 2020.

[30] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2019.

[31] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, 2012.

[32] A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning," in *Proc. 22nd Int. Conf. Machine Learning (ICML)*, 2005.

[33] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proc. 34th Int. Conf. Machine Learning (ICML)*, 2017.

[34] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, 2006.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 11 , November - 2026

[35] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets," *PLoS ONE*, vol. 10, no. 3, 2015.

[36] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural Comput.*, vol. 10, no. 7, 1998.

[37] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artificial Intelligence and Statistics (AISTATS)*, 2017.

[38] B. Jacob et al., "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2018.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)