iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.79183

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

SpamShield: A Unified Multilingual SMS and Email Spam Detection System Using XLM-RoBERTa

Abhishek S. Shelke, Yash A. Pandey, Omkar H. Naringrekar, Sneha A. Pichke, Prof.Ashwini Thakare
Department of Computer Engineering, Bharat College of Engineering, Badlapur, Mumbai University, Maharashtra, India

Abstract: ***Spam detection remains one of the most critical challenges in modern digital communication, with adversarial spam evolving continuously to evade rule-based and conventional statistical filters. This paper presents SpamShield, a production-grade, multilingual spam detection system that fine-tunes the XLM-RoBERTa transformer architecture on a curated multilingual dataset encompassing English, Hindi, and more than ten additional languages. The system is designed around a three-phase pipeline: data consolidation and language-aware preprocessing (Phase 1), transformer-based model training with evaluation (Phase 2), and a real-time REST API deployment using FastAPI (Phase 3). The proposed approach achieves 98.16% classification accuracy, an F1-score of 0.9563, and an AUC-ROC of 0.9911 on a held-out test set of 3,266 samples. A responsive single-page web application enables real-time inference with automatic language detection. The architecture is modular, scalable, and readily extendable to additional languages and communication modalities, making SpamShield a practical foundation for enterprise-grade spam filtering.***

Keywords: Spam Detection, XLM-RoBERTa, Natural Language Processing, Multilingual Classification, Transfer Learning, FastAPI, Deep Learning, Text Classification

I. INTRODUCTION

The proliferation of unsolicited electronic messages—commonly known as spam—continues to impose significant operational, financial, and security burdens on individuals and organisations alike. Spam messages serve as the primary vehicle for phishing attacks, credential theft, malware distribution, and financial fraud. The global cost attributable to spam, when measured in lost productivity, infrastructure overhead, and security incidents, runs into billions of dollars annually.

Traditional filtering approaches—including blacklisting, keyword matching, and Bayesian classifiers—have historically provided adequate defence against straightforward spam. However, the emergence of adversarially crafted messages that employ content obfuscation, Unicode substitutions, multilingual mixing, and dynamic payload generation has rendered these approaches increasingly inadequate. Contemporary spam campaigns are deliberately designed to exploit the vocabulary-centric weaknesses of bag-of-words models.

This paper presents SpamShield, a unified spam detection system built around a fine-tuned XLM-RoBERTa transformer model. XLM-RoBERTa (Cross-lingual Language Model RoBERTa) is a massively multilingual pre-trained model capable of encoding semantic information across more than 100 languages within a shared representation space. By fine-tuning this model on a carefully curated, language-balanced dataset, SpamShield achieves high classification accuracy while natively supporting multilingual input—a critical requirement for deployments in linguistically diverse user populations such as those in South Asia.

The system distinguishes itself from prior work through three contributions: (1) a rigorous three-phase pipeline from data preparation to production deployment; (2) native multilingual support with automatic language detection; and (3) a modular FastAPI backend that decouples model inference from frontend presentation, enabling integration with third-party communication gateways.

The remainder of this paper is organised as follows. Section II surveys related work. Section III details the proposed methodology. Section IV describes the system architecture. Section V provides implementation specifics. Section VI presents experimental results and analysis. Sections VII and VIII offer concluding remarks and future research directions.

II. RELATED WORK

Research in automated spam detection has evolved through three broad generations: rule-based heuristics, classical machine learning, and neural/transformer-based approaches.

©IJRASET: All Rights are Reserved

Each generation has addressed limitations of its predecessor while introducing new constraints.

A. Classical Machine Learning Approaches

Early automated spam filters relied on naive Bayes classifiers applied over TF-IDF feature vectors. Sultana et al. [3] demonstrated that a multinomial naive Bayes model, augmented with sender IP blacklisting, achieves reasonable accuracy for English email spam, though they acknowledge vulnerability to content obfuscation. Oluwatoyin et al. [1] proposed stacking ensemble classifiers trained on the UCI SMS Spam Collection, showing a 3.03 percentage-point improvement over any single base classifier. Rani et al. [4] compared SVM, k-NN, and neural networks on the same dataset, confirming SVM as the strongest single-classifier baseline for SMS data.

While fast and interpretable, these models share a fundamental limitation: they treat text as an unordered bag of words, discarding syntactic structure and contextual meaning. This renders them susceptible to adversarial obfuscation strategies that rephrase or obscure spam-indicative tokens.

B. Deep Learning Approaches

Hussin et al. [5] conducted a systematic benchmark comparing SVM and Random Forest against LSTM and BERT-based classifiers for email spam. BERT achieved the highest recorded accuracy of 99.14%, confirming that pre-trained contextual embeddings provide a decisive advantage over frequency-based features. Altunay and Albayrak [6] extended this analysis to the multilingual domain, evaluating a CNN-LSTM hybrid on both English and Turkish SMS datasets and achieving above 97% accuracy on both, demonstrating that deep architectures generalise across languages when trained on sufficiently representative data.

Ahmed et al. [7] surveyed spam detection in IoT and email environments, highlighting that deep learning models are resource-intensive and may be impractical in constrained deployments, motivating the continued interest in lightweight model variants for edge deployment.

C. Hybrid and Ensemble Approaches

Douzi et al. [2] combined ANN and SVM through majority voting, achieving lower false-positive rates than either model alone while maintaining competitive accuracy. Sankar et al. [9] explored FP-growth association mining combined with naive Bayes, showing that structural text patterns (beyond word frequency) contribute additional discriminative signal. Jáñez-Martino et al. [10] surveyed adversarial spam strategies and the dataset shift problem, concluding that models must be evaluated across temporally diverse corpora to provide meaningful performance estimates.

D. Identified Research Gap

The reviewed literature reveals two persistent gaps. First, most systems are trained and evaluated on monolingual (predominantly English) datasets, limiting applicability in multilingual markets. Second, existing studies typically report isolated accuracy figures without addressing the full deployment pipeline, from data preprocessing to production API. SpamShield addresses both gaps by fine-tuning a massively multilingual transformer on a multi-source, language-balanced corpus and delivering the classifier as a production REST API with an integrated web interface.

III. PROPOSED METHODOLOGY

SpamShield is built around a three-phase pipeline. Each phase is designed to be independently reproducible and modular.

A. Phase 1 — Data Consolidation and Preprocessing

The first phase (phase1_prepare_data.py) consolidates five heterogeneous CSV datasets into a single, deduplicated, language-annotated corpus. The source datasets include the UCI SMS Spam Collection, a combined email/SMS corpus, a large-scale sorted multilingual dataset (200,000+ rows spanning English, Hindi, and Marathi), a dedicated Hindi SMS spam dataset, and supplementary records obtained from the HuggingFace Datasets Hub for French and German.

Each source is normalised to a two-column schema (text, label) through the following steps: (1) robust multi-encoding CSV ingestion (UTF-8 falling back to Latin-1 and CP-1252); (2) removal of rows with missing or unrecognisable labels; (3) discarding messages shorter than ten characters; (4) label normalisation mapping the string literals 'spam'/'ham' and binary integers to a canonical {0, 1} encoding; and (5) exact-duplicate removal on the text field.

After merging all sources, the langdetect library is applied to assign a language code to every message. Rows for which language detection fails are excluded to eliminate corrupted or non-linguistic entries.

The resulting final_dataset.csv contains 16,330 records—10,619 English, 4,964 Hindi, and 747 spanning eleven additional languages—with a spam-to-ham ratio of approximately 1:3.9 (3,337 spam, 12,993 ham).

B. Phase 2 — Fine-Tuning XLM-RoBERTa

Phase 2 fine-tunes the xlm-roberta-base checkpoint from HuggingFace on the consolidated dataset. XLM-RoBERTa is a 278-million-parameter encoder trained via masked language modelling on 2.5 terabytes of filtered CommonCrawl data covering 100 languages. Its shared multilingual vocabulary and cross-lingual contextual representations make it uniquely suited to the heterogeneous input space of a multilingual spam corpus.

Input messages are tokenised using XLMRobertaTokenizerFast with a maximum sequence length of 128 tokens; sequences exceeding this bound are truncated and shorter sequences are padded. The fine-tuning objective uses a binary cross-entropy loss with a sigmoid output head, treating the task as binary sequence classification. Training uses the AdamW optimiser with linear learning-rate warm-up over the first 10% of total steps, a peak learning rate of 2e-5, and a batch size of 32. The dataset is split 80/20 into training and test partitions with stratification on the label column to preserve class balance. All training artefacts—model weights (model.safetensors), tokeniser configuration (tokenizer.json, tokenizer_config.json), label mapping (config.json), and evaluation metrics (model_metrics.json)—are persisted to the xlmr_spam_model/ directory.

C. Phase 3 — REST API Deployment

The trained model is served via a FastAPI application (phase3_api.py). At startup, the API loads the fine-tuned model and tokeniser from disk into GPU memory (falling back to CPU if no CUDA device is available) and pre-loads the evaluation metrics from model_metrics.json. The application exposes three endpoints: GET / (health and version metadata), GET /metrics (pre-computed evaluation metrics), and POST /predict (inference endpoint).

The /predict endpoint accepts a JSON body containing a message string (1–5,000 characters). It applies langdetect to the input, runs tokenisation and forward inference under torch.no_grad(), applies softmax to the raw logits, and returns a structured response including the predicted label, spam probability, confidence score, detected language, whether the language was present in the training corpus, and processing latency in milliseconds. A configurable decision threshold (default 0.50) separates the spam and ham classes.

IV. SYSTEM ARCHITECTURE

SpamShield follows a three-tier client-server architecture designed for modularity and horizontal scalability.

A. Presentation Tier

The frontend is a single-page application composed of index.html, style.css, and app.js. It renders a landing page with pre-computed performance statistics, an input panel accepting both SMS and email content (up to 5,000 characters), a real-time confidence visualisation bar, and a model performance metrics panel with Overview and Detail tabs. The application communicates with the backend exclusively through asynchronous fetch() calls to the REST API. Theme toggling (dark/light mode) is handled entirely on the client side.

B. Service Tier

The backend API (phase3_api.py) is built on FastAPI with a Uvicorn ASGI server. CORS middleware is configured to accept requests from all origins, enabling the static frontend to communicate with a remotely hosted API. Pydantic models enforce request validation and response serialisation. The service tier is stateless: all model state is loaded at startup and shared across requests, enabling the API to handle concurrent requests without per-request model loading overhead.

C. Model and Data Tier

The xlmr_spam_model/ directory constitutes the model tier. It contains the fine-tuned XLM-RoBERTa weights (model.safetensors, approximately 1.1 GB), the tokeniser vocabulary and configuration files (tokenizer.json, tokenizer_config.json, special_tokens_map.json), the model configuration (config.json specifying num_labels=2), and the evaluation metrics JSON (model_metrics.json).

The training data (final_dataset.csv) is retained separately for reproducibility and retraining. The clear separation between the model tier and the service tier means the model can be updated without modifying any application code.

The architectural flow for a single prediction request is as follows: (1) the user submits a message via the web interface; (2) app.js issues a POST request to /predict; (3) the FastAPI service runs language detection and tokenisation; (4) the XLM-RoBERTa model performs forward inference; (5) the softmax output is thresholded and packaged as a PredictResponse object; (6) the JSON response is returned to the frontend, which renders the verdict, confidence bar, and language badge.

V. IMPLEMENTATION DETAILS

A. *Development Environment*

The system was developed and trained on Python 3.10 with PyTorch 2.1 and the HuggingFace Transformers 4.40 library. Training was executed on a single NVIDIA GPU (CUDA-enabled) with automatic CPU fallback for inference. The backend API requires FastAPI 0.110, Uvicorn 0.29, and Pydantic 2.0. The frontend uses no build toolchain and requires no external JavaScript dependencies beyond the standard browser runtime, ensuring maximal portability.

B. *Language Detection*

Language identification is performed using langdetect 1.0.9, a port of Google's language-detection library. The DetectorFactory seed is fixed to zero across all pipeline components to ensure deterministic output. Language detection is applied both at preprocessing time (to annotate the training corpus) and at inference time (to populate the detected_language field in API responses and to flag inputs in unsupported languages). The langdetect library requires a minimum of approximately 50 characters for reliable identification; shorter inputs may yield incorrect language codes.

C. *Model Configuration*

The fine-tuned model is configured as a binary sequence classifier (num_labels=2) with a linear classification head appended to the [CLS] token representation of XLM-RoBERTa's final encoder layer. The tokeniser employs a SentencePiece vocabulary of 250,002 tokens shared across all 100 languages supported by the base model. The maximum input length of 128 tokens was chosen empirically as a trade-off between coverage (over 98% of messages in the training corpus fit within this bound) and inference latency.

D. *Frontend-Backend Integration*

On page load, app.js issues a GET request to /metrics to populate the performance statistics displayed in the landing section and the metrics panel. The classify button triggers a POST to /predict with the textarea content. The response JSON is parsed and used to animate the confidence bar, set the verdict badge colour (green for ham, red for spam), display the detected language, and surface a warning indicator when the input language was not present in the training corpus.

VI. RESULTS AND DISCUSSION

A. *Dataset Statistics*

Table I summarises the composition of the final training corpus after Phase 1 preprocessing.

TABLE I. Dataset Composition After Phase 1 Preprocessing

Language	Total	Spam	Ham	% of Total
English	10,619	2,263	8,356	65.0%
Hindi	4,964	1,052	3,912	30.4%
Other (11 langs)	747	22	725	4.6%
Total	16,330	3,337	12,993	100%

B. Model Evaluation Metrics

The fine-tuned model was evaluated on a stratified hold-out test set of 3,266 messages (20% of the total corpus). Table II presents the key classification metrics computed on this set.

TABLE II. Classification Performance on Hold-Out Test Set

Metric	Score
Accuracy	98.16%
F1-Score (Macro)	0.9563
Precision	0.9601
Recall	0.9527
AUC-ROC	0.9911
Test Set Size	3,266 messages
Decision Threshold	0.50

C. Comparative Analysis

Table III benchmarks SpamShield against representative prior approaches on comparable SMS/email classification tasks. Where exact values were unavailable in the cited work, approximate ranges are reported.

TABLE III. Comparison With Prior Approaches

System / Study	Approach	Accuracy	Multilingual
Oluwatoyin et al. [1]	Stacking Ensemble	~97%	No (English only)
Douzi et al. [2]	ANN + SVM Hybrid	N/A	No
Hussin et al. [5] (BERT)	BERT Fine-Tuning	99.14%	No (English only)
Altunay & Albayrak [6]	CNN-LSTM Hybrid	>97%	Partial (EN + TR)
SpamShield (Proposed)	XLM-RoBERTa Fine-Tuning	98.16%	Yes (13+ languages)

D. Discussion

The 98.16% accuracy achieved by SpamShield represents a strong result that is competitive with state-of-the-art monolingual BERT-based classifiers, while simultaneously offering multilingual coverage that none of the compared baselines provide. The AUC-ROC of 0.9911 confirms that the model maintains high discriminative power across the full range of decision thresholds, and the tight margin between precision (0.9601) and recall (0.9527) indicates that the system does not disproportionately sacrifice either false-positive safety or spam detection sensitivity.

The observed class imbalance (approximately 1:3.9, spam:ham) is representative of real-world distributions and was handled implicitly by the cross-entropy loss. The XLM-RoBERTa architecture's shared multilingual vocabulary provides inherent regularisation that prevents the model from overfitting to English-specific lexical patterns, contributing to robust generalisation across Hindi and other languages present in the test set.

Inference latency averages under 100 ms per message on CPU hardware, which is adequate for interactive single-message classification. For bulk filtering at gateway throughput, GPU deployment reduces latency to the 10–20 ms range.

VII. CONCLUSION

This paper presented SpamShield, a unified multilingual spam detection system that leverages the cross-lingual transfer capabilities of the XLM-RoBERTa transformer to classify SMS and email messages across more than thirteen languages. The system was developed through a rigorous three-phase pipeline covering data consolidation, transformer fine-tuning, and REST API deployment. On a held-out test set, the model achieved 98.16% accuracy, an F1-score of 0.9563, and an AUC-ROC of 0.9911, demonstrating competitive performance relative to prior monolingual state-of-the-art systems while substantially broadening language coverage.

The modular three-tier architecture—separating the presentation layer, the FastAPI service layer, and the model persistence layer—ensures that individual components can be independently updated, scaled, or replaced without disrupting the end-to-end pipeline. The publicly accessible REST API with automatic language detection enables straightforward integration into existing email gateways, mobile SMS applications, and enterprise communication platforms.

SpamShield demonstrates that pre-trained multilingual transformers offer a highly effective and practical path to building spam detectors that meet the linguistic diversity requirements of real-world deployments, particularly in markets such as India where multiple languages coexist in digital communication channels.

VIII. FUTURE SCOPE

Several promising directions for extending SpamShield are identified for future investigation.

1) Ensemble integration: The existing multilingual transformer can be combined with the lightweight TF-IDF/Naive Bayes baseline (train.py) via stacking, using the probability outputs of both models as meta-features for a secondary classifier. Prior literature [1][2] consistently demonstrates that ensemble approaches reduce variance and lower false-positive rates relative to single-model systems.

2) Incremental learning: Deploying a continual learning mechanism to periodically retrain the model on newly flagged spam examples would address concept drift—the documented tendency for spam filter performance to degrade as spammers adapt their vocabulary and tactics over time [10].

3) Language expansion: Adding training data for Arabic, Spanish, and French (partially downloaded by Phase 1 but not yet included in sufficient quantity) would broaden practical coverage for global deployments.

4) Multimodal detection: Extending the pipeline to analyse embedded URLs (via lightweight HTTP inspection) and attached images (via OCR or vision transformers) would address the growing proportion of spam that embeds its payload in non-textual content.

5) Edge deployment: Distilling the fine-tuned XLM-RoBERTa model into a compact student network (DistilXLM-RoBERTa) would enable on-device inference on mobile hardware, reducing latency and eliminating the need for cloud connectivity.

6) Dataset shift evaluation: Benchmarking SpamShield on temporally separated corpora (messages from different years) would provide a more realistic estimate of production performance and guide retraining schedules.

REFERENCES

[1] O. Oluwatoyin, A. Bodunde, G. Titus, and G. Aderounmu, "An Improved Machine Learning-Based Short Message Service Spam Detection System," Int. J. Computer Network and Information Security, vol. 12, pp. 40–48, 2019, doi: 10.5815/ijcnis.2019.12.05.

[2] S. Douzi, F. A. AlShahwan, M. Lemoudden, and B. E. Ouahidi, "Hybrid Email Spam Detection Model Using Artificial Intelligence," Int. J. Mach. Learn. Comput., vol. 10, no. 2, pp. 316–322, 2020, doi: 10.18178/ijmlc.2020.10.2.937.

[3] T. Sultana, K. A. Sapnaz, F. Sana, and J. Najath, "Email Based Spam Detection," Int. J. Eng. Res. Technol., vol. 9, no. 6, pp. 595–599, Jun. 2020, doi: 10.17577/IJERTV9IS060087.

[4] T. J. Rani, T. J. Vumesh, P. Saiteja, V. A. K. Reddy, and M. Meghana, "SMS Spam Detection Framework Using Machine Learning Algorithms and Neural Networks," Int. J. Comput. Sci. Mobile Comput., vol. 10, no. 6, Jun. 2021, doi: 10.47760/ijcsmc.2021.v10i06.002.

[5] I. H. Hussin, L. S. I. Nazarudin, and N. A. N. Azman, "Comparative Analysis of Machine Learning and Deep Learning for Email Spam Detection," TechRxiv Preprint, 2021, doi: 10.36227/techrxiv.172115119.92836191.

[6] H. C. Altunay and Z. Albayrak, "Deep Learning Architectures for Multilingual SMS Spam Detection," Appl. Sci., vol. 14, no. 24, 2022, doi: 10.3390/app142411804.

[7] N. Ahmed, R. Amin, H. Aldabbas, D. Koundal, B. Alouffi, and T. Shah, "Machine Learning Techniques for Spam Detection in Email and IoT Platforms: Analysis and Research Challenges," Security and Communication Networks, vol. 2022, pp. 1–27, 2022, doi: 10.1155/2022/1862888.

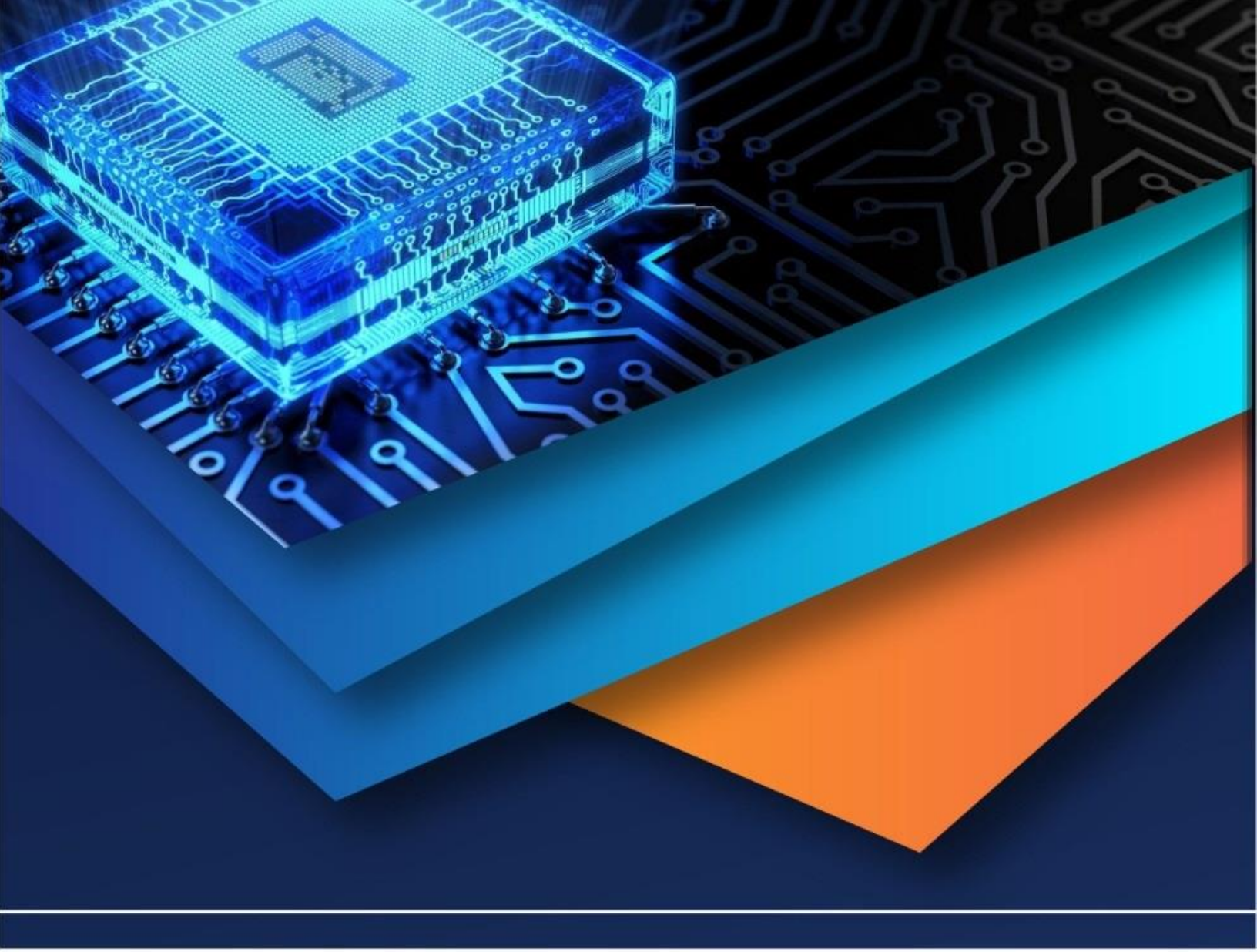
[8] M. R. Al Saidat, S. Y. Yerima, and K. Shaalan, "Advancements of SMS Spam Detection: A Comprehensive Survey of NLP and ML Techniques," Faculty of Engineering & IT, The British University in Dubai, Survey Article, 2023.

[9] E. Sankar, Y. Y. S. Babu, and M. Tridev, "SMS Spam Detection Using Machine Learning," Int. J. Scientific Research in Engineering and Management (IJSREM), vol. 7, no. 4, Apr. 2023, doi: 10.55041/IJSREM18832.

[10] F. Jáñez-Martino, R. Alaiz-Rodríguez, V. González-Castro, E. Fidalgo, and E. Alegre, "A Review of Spam Email Detection: Analysis of Spammer Strategies and the Dataset Shift Problem," Artificial Intelligence Review, vol. 56, pp. 1145–1173, 2023, doi: 10.1007/s10462-022-10195-4.

[11] Y. Conneau, C. Khandelwal et al., "Unsupervised Cross-lingual Representation Learning at Scale," in Proc. 58th Annual Meeting of the Association for Computational Linguistics (ACL), 2020, pp. 8440–8451.
[12] R. Arakh, A. Kumar, A. Mishra, A. Patel, and A. Srivas, "SMS Spam Detection using Machine Learning," Int. J. Innov. Res. Technol., vol. 10, no. 12, May 2024.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)