Self-consistency and Graph-based Filtering to Enhance Synthetic Arabic SMS Generation for Smishing Detection

Amal Alotaibi[1,*], Miada Almasre[2], Hadeel Surougi[3], Mona Alkhozae[4], Nouf Alghanmi[5]

[1,2,4]*Department of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia*

[3]*Department of Computer Science, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia*

[5]*Department of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Rabigh, Saudi Arabia*

(Received: June 10, 2025; Revised: August 5, 2025; Accepted: November 8, 2025; Available online: December 19, 2025)

Abstract

Smishing or SMS phishing is a growing cybersecurity threat in mobile security, with Arabic-speaking regions particularly vulnerable due to the absence of large, labeled datasets. The main objective of this study is to develop a scalable pipeline that can generate and classify Arabic SMS messages to overcome the lack of data and enhance detection performance. The contributions are threefold: (i) constructing a balanced dataset of 6,903 messages by combining 903 synthetic samples with 6,000 real Arabic SMS messages; (ii) introducing a hybrid generation framework that integrates a fine-tuned GPT-3.5-turbo language model with Conditional WGAN embeddings, refined using self-consistency sampling and graph-based redundancy filtering; and (iii) evaluating the dataset using multiple machine learning (Logistic Regression, Random Forest, SVM) and deep learning (CNN, BERT) models. The pipeline unifies adversarial embedding generation, large language model fine-tuning, and cosine similarity filtering. Experimental results show consistently strong performance: Logistic Regression and Random Forest both achieved accuracy of 0.9949 and F1-score of 0.9950, while SVM outperformed all with accuracy 0.9957 and F1-score 0.9957. Among deep learning models, CNN reached accuracy 0.9942 and F1-score 0.9942, and BERT achieved 0.9900 across all metrics. These findings confirm that while SVM is most effective for this dataset, CNN and BERT add robustness by capturing semantic subtleties. Visual analyses, including confusion matrices and t-SNE projections, validated the overlap between real and synthetic embeddings, while comparative tables positioned this study within the context of recent Arabic smishing research. The novelty of this work lies in combining self-consistency and graph-based filtering within a hybrid generation-classification pipeline tailored for Arabic SMS, providing a reproducible framework extendable to low-resource, multilingual, and cross-platform environments such as WhatsApp and Telegram.

Keywords: Smishing Detection, Synthetic Data Augmentation, Large Language Models (LLMs), Arabic SMS Generation, Conditional WGAN

1. Introduction

Phishing attacks pose a significant global threat to individuals and businesses, with SMS-based phishing, known as smishing, emerging as a particularly effective cybercrime tactic. These attacks exploit human vulnerabilities by sending deceptive messages designed to steal personal data, financial information, or login credentials [1]. The increasing reliance on mobile devices and the widespread use of SMS for communication, particularly in regions such as the Middle East and North Africa (MENA), where mobile penetration is high, have made smishing a growing cybersecurity concern [2]. However, detecting phishing messages in Arabic presents challenges due to the complexity of the language linguistic structure and script, which conventional detection methods often struggle to interpret accurately [3]. This highlights the need for robust, language-specific solutions to prevent smishing.

The availability of high-quality, labeled datasets for training and evaluation is essential to the development of successful smishing detection systems [4]. However, privacy concerns, limited labeled data, and the evolving nature of phishing attacks make real-world Arabic smishing datasets rare. To overcome these challenges, synthetic data generation has become an appealing option, enabling researchers to produce sizable and varied datasets that closely

***Corresponding author: Amal Alotaibi (aalotaibi0902@stu.kau.edu.sa)**
DOI: https://doi.org/10.47738/jads.v7i1.1033

resemble real-world situations [5]. Current methods for generating synthetic smishing datasets mainly focus on English and struggle to capture the linguistic complexity and contextual details of Arabic. The Arabic script, diverse dialects, and the structure of phishing messages make it difficult for conventional techniques to generate realistic and diverse smishing samples. Furthermore, the lack of large, labeled Arabic datasets further limits the ability to train robust detection models. This highlights the urgent need for an improved approach to synthetic dataset generation that ensures linguistic accuracy, contextual relevance, and diversity in Arabic smishing messages.

To build a solid foundation for our framework, we curated a real-world Arabic SMS dataset sourced from user-submitted messages. Native Arabic speakers manually reviewed every message to ensure authenticity and contextual accuracy. Smishing messages were identified based on clear red flags, such as suspicious URLs, urgent or manipulative language, requests for personal information, and patterns that align with known phishing tactics. We made a clear distinction between smishing and spam, excluding promotional or irrelevant bulk content from the smishing category. For edge cases that weren't clearly smishing or benign, we chose to leave them out altogether to reduce noise in model training. This careful curation helped us create a high-quality dataset that captures real-world smishing behavior while keeping the lines between malicious and non-malicious content sharp and well-defined.

This paper presents a novel approach for generating and evaluating synthetic Arabic SMS datasets for smishing detection. By leveraging the capabilities of Large Language Models (LLMs), a class of artificial intelligence (AI) models is designed to understand, generate, and manipulate human language [6]. These models are trained on enormous amounts of text data and employ powerful machine learning techniques, particularly deep learning, to learn the structure, semantics, and nuances of language [7]. LLMs have transformed NLP, enabling advanced text generation and adaptation for cybersecurity applications such as phishing detection [8]. This study modifies the model to the unique features of phishing messages, including the use of urgent language, ambiguous URLs, and social engineering techniques, by fine-tuning GPT-3.5-turbo on a collected dataset of real Arabic SMS messages.

Our methodology comprises four primary phases designed to address the challenges of Arabic smishing detection. In the first phase, a Conditional WGAN is trained to generate synthetic SMS embeddings conditioned on message labels and sender centrality extracted from the knowledge graph. These embeddings provide diverse and realistic latent representations of phishing behaviors. In the second phase, these embeddings are used to seed a fine-tuned GPT-3.5-turbo model, enabling it to capture actual message semantics while benefiting from augmented diversity beyond the limited real dataset. In the third phase, we construct balanced datasets by combining the generated smishing messages with authentic SMS samples. To ensure coherence, uniqueness, and diversity, the dataset is refined using self-consistency sampling, which generates multiple candidate variations and selects the most stable outputs, and graph-based redundancy filtering, which removes duplicates and overly similar samples. Finally, in the fourth phase, the efficacy of the constructed datasets is assessed using a variety of machine learning models, including Random Forest (RF), Support Vector Machine (SVM), and Logistic Regression (LR), as well as deep learning models such as Convolutional Neural Networks (CNN) and Bidirectional Encoder Representations from Transformers (BERT). Each phase of the methodology is designed to mitigate key limitations in Arabic smishing detection, ensuring that the synthetic data is both high-quality and representative of real-world phishing threats.

This study contributes to the fields of synthetic data generation and smishing detection. First, it shows a novel approach for generating high-quality synthetic Arabic SMS datasets, overcoming the lack of real-world labeled data. Second, it shows the effectiveness of fine-tuning pre-trained LLMs for Arabic smishing detection, offering a scalable and adaptable solution for real-world applications. Third, it develops a new standard for synthetic dataset generation by using self-consistency and graph-based filtering techniques to ensure both quality and diversity. Fourth, it evaluates the performance of the synthetic dataset across several ML and DL models, providing valuable insights into its practical applicability. Finally, this research highlights the importance of dataset quality and diversity in enhancing smishing detection systems, presenting recommendations for future advancements.

The rest of the paper is organized as follows. Section 2 presents related work on synthetic dataset generation with LLMs and Arabic smishing detection. Section 3 outlines the methodology, which includes the framework for fine-tuning, synthetic data generation, and model evaluation. Section 4 presents experimental results. Section 5 discusses the performance of the synthetic dataset and fine-tuned model. The study is concluded, and future research directions

are outlined in Section 6. This research aims to contribute to developing strong, language-specific solutions for preventing smishing and improving cybersecurity in the MENA region and beyond by addressing the difficulties associated with smishing detection in Arabic SMS messages, while exploring the boundaries of synthetic data generation.

2. Related Work

2.1. Using LLM in synthetic datasets

Several studies have proposed models to generate datasets for phishing detection in emails, SMS, URLs, HTML, and social media platforms. In contrast to real-world datasets, Kulkarni et al. [9] investigated the resilience of phishing detection systems against adversarial manipulation by comparing traditional ML classifiers with LLM. The study employed multiple phishing and benign webpage datasets collected from open-source repositories and real-world feeds. Three categories of adversarial perturbations were applied: lexical obfuscation, URL-level modifications, and HTML content injection. Models tested included LR, RF, and XGBoost on the ML side, alongside fine-tuned transformer-based LLMs. Experimental results demonstrated that while baseline ML classifiers initially achieved high accuracy (up to 0.95 on clean datasets), their performance significantly degraded under adversarial conditions, dropping below 0.70 in some cases. In contrast, LLM-based models preserved stronger robustness, sustaining accuracies above 0.90 even when exposed to adversarially manipulated phishing webpages. These results confirmed that integrating LLMs into phishing detection frameworks enhances adaptability and mitigates vulnerabilities compared to conventional ML approaches.

In [10], the authors demonstrate that large language models can automate and scale spear-phishing by iteratively refining messages in a generative-critique loop that jailbreaks safety policies and optimizes deception quality. Their experiments show that LLM-crafted emails attain high perceived realism in human evaluations and can evade contemporary phishing detectors more effectively after critique-guided refinement, underscoring the elevated risk of automated, personalized social-engineering at scale.

Mahendru and Pandit [11] evaluated the efficacy of LLMs, including GPT-4 and Gemini 1.5, in phishing detection and compared their performance to the DeBERTa V3 model. The study assessed the advantages and limitations of these methods across various datasets and phishing attack types using precision, recall, accuracy, and F1-score as primary metrics, with a focus on recall to ensure the phishing emails are not overlooked. Three datasets were used: a synthetic dataset generated by GPT-4, the HuggingFace phishing dataset, and the Nazario and Nigerian Fraud dataset. LLMs were evaluated using structured Chain-of-Thought (COT) prompting with reasoning-based classification, while DeBERTa V3 was optimized for phishing detection using both actual and synthetic data. DeBERTa V3 emerged as the best-performing model, achieving recall scores of 95.17% on the HuggingFace dataset and 99.05% on the Nazario dataset, outperforming GPT-4. However, LLMs performed better in detecting web-based and HTML-based phishing attacks, achieving 100% accuracy in identifying phishing in synthetic data. The study concluded that while LLMs show better adaptability to complex phishing tactics, DeBERTa V3 is faster and more effective in real-world situations. It also recommends future research on enhancing the quality of synthetic datasets, combining multimodal phishing detection techniques, and fine-tuning LLMs for improved performance.

A recent study by [12] examined LLMs such as Claude, GPT-4, PaLM/Gemini, and LLaMA for both creating and identifying phishing emails, with a focus on hybrid strategies that combine AI-generated content with human knowledge. Synthetic and actual data are utilized to develop phishing datasets, using a dataset of 725,000 emails containing both legitimate and phishing emails. Experiments comparing phishing success rates showed that hybrid approaches (GPT-4 + human-designed V-Triad) achieved click-through rates of up to 81%, surpassing traditional phishing strategies and significantly outperforming GPT-generated emails (30–44%). Claude outperformed GPT-4 in some cases, successfully identifying 75% of phishing emails; however, the efficacy of LLM detection varied. While AI-generated phishing emails appear realistic, they may lack the sophistication of human attackers. As a result, challenges related to dataset authenticity, generalization, and adversarial robustness persist. The rapid advancement of LLMs presents both opportunities and challenges, highlighting the need for more effective mitigation techniques and ethical AI research.

In [13], Shim et al. proposed a theory-driven prompt engineering approach for data augmentation to improve smishing detection. Recognizing the scarcity of annotated smishing datasets, they leveraged LLMs to generate synthetic SMS messages that incorporate persuasion elements such as urgency, authority, and social proof. These augmented datasets were used to train multiple machine learning models, including BERT, RoBERTa, DistilBERT, and ALBERT. Experimental results showed that models trained with persuasion-based augmented data significantly outperformed those trained on the original dataset, with RoBERTa-large achieving an F1-score of 98.2%, representing a 5.3% improvement. The study demonstrated that combining LLM-generated augmentation with psychological persuasion concepts enhances dataset diversity and leads to better model generalization for smishing detection.

Focusing on dataset generation and synthetic data augmentation, [14] introduced the Improved Phishing and Spam Detection Model (IPSDM), which refined RoBERTa and DistilBERT using a combined dataset of 5761 spam, phishing, and ham emails. The study employed Adaptive Synthetic Sampling (ADASYN) to address class imbalance, enhancing phishing attempt detection while decreasing bias toward legitimate emails. IPSDM achieved an F1-score of 0.98 and test accuracy of up to 99.00%, outperforming baseline models and demonstrating significant performance improvements. Attention-based mechanisms and fine-tuning techniques, including learning rate scheduling and hyperparameter optimization, drove these enhancements. However, challenges remain regarding adversarial robustness, generalization to emerging phishing techniques, and the authenticity of datasets. These findings emphasize the need for continuous improvement and practical validation while highlighting the potential of transformer-based LLMs for phishing detection.

Recent developments have significantly improved the ability to generate text in LLMs, yet these models are still vulnerable to hallucinations, which calls for more precise uncertainty estimation methods. To overcome this challenge, self-consistency techniques allow for more reliable, understandable, and verifiable LLM-generated results across various fields. Traditional self-consistency techniques are extended by Universal Self-Consistency (USC), which facilitates response selection in tasks involving the generation of free-form text, such as code generation, summarization, and answering open-ended questions. In [15], they demonstrated that USC used LLMs to evaluate the consistency of several generated responses and select the most reliable result without explicit answer extraction, in contrast to standard self-consistency, which depends on a majority vote over structured outputs. The technique generalizes self-consistency to various task types by creating several potential responses, concatenating them into a single prompt, and then instructing the LLM to select the most consistent one. USC consistently outperforms baseline methods such as greedy decoding, random selection, and standard self-consistency when tested against various benchmarks. These benchmarks include GSM8K and MATH for mathematical reasoning, BIRD-SQL, and ARCADE for code generation, GovReport and SummScreen for summarization, and TruthfulQA for open-ended QA. According to the experimental results, USC achieves execution-based self-consistency in code generation, attaining 90.2% accuracy on GSM8K, and enhances summarization quality by increasing GovReport ROUGE-1 from 38.8 to 40.2. Furthermore, USC achieved the highest TruthfulQA score, reaching 67.7% with PaLM 2-L and 82.5% with GPT-3.5-turbo. USC maintains computational efficiency while reducing answer selection biases by requiring only one additional LLM query for selection. Oracle demonstrates how well USC bridges the gap between optimal performance and naïve selection approaches, confirming it as a scalable and generalizable strategy to improve the dependability of LLM output across various applications.

A new method called Atomic Self-Consistency (ASC) aims to improve LLMs' recall and factual accuracy of long-form responses. In contrast to USC, which selects the most consistent answer, ASC creates a better composite answer by extracting and combining relevant atomic facts from several generated responses. As described in [16], the process consists of four steps: (1) dividing responses into atomic facts; (2) using agglomerative clustering and SimCSE embeddings to cluster-related facts; (3) filtering clusters based on consistency; and (4) using an LLM to summarize a subset of atomic facts. ASC shows notable gains over USC and direct generation techniques when tested on four long-form QA datasets: ASQA, QAMPARI, QUEST, and ELI5. In ASC-F, a version of factual correctness, the method uses ChatGPT and Llama-2 (70B) for generation, SimCSE for phrase embeddings, and GTR-T5-XXL for retrieval-based verification. According to numerical statistics, ASC consistently outperforms USC, achieving 20.5% recall on QAMPARI and 47.01% Str_EM on ASQA, while maintaining greater fluency and factual accuracy. Furthermore, ASC offers more control over generation quality by introducing an adjustable parameter (θ) that balances response fluency

and recall. The Oracle study highlights ASC's contribution to enhancing LLM performance in challenging reasoning tasks by pointing to unrealized potential for enhancing long-form generation through merging techniques.

The potential of LLMs in graph generation has been recently investigated; however, this field is still unexplored compared to graph discriminative tasks such as link prediction and node classification. LLM4GraphGen is a framework presented by [17] that systematically evaluated the ability of LLMs to generate graphs under three paradigms: distribution-based generation, which assesses the model's ability to infer and reproduce graph distributions; rule-based generation, which evaluates adherence to predefined structural rules (e.g., trees, cycles, and planar graphs); and property-based generation, which focuses on generating molecules with particular real-world properties, like HIV inhibition. Using zero-shot, few-shot, and CoT prompting, the study compares GPT-4, GPT-3.5, and LLaMA2-13B. Performance is evaluated based on validity, uniqueness, novelty, and the accuracy of molecular property prediction. The results indicate that while GPT-4 performs well on tasks based on rules and basic distributions, it struggles with complex structures and distributions based on motifs. Unexpectedly, CoT prompting significantly improves performance, especially in molecular property-based generation and distribution inference, raising the probability of producing genuine molecules by 48.8%. These results show the potential for hybrid techniques that combine LLMs with conventional Graph Neural Networks (GNNs) for improved graph-based reasoning and synthesis, highlighting the developing capabilities of LLMs in generative graph learning.

Zhang et al. [18] introduced LUQ, a long-text uncertainty quantification framework for LLMs that estimates sentence-level consistency across multiple sampled generations and uses uncertainty to select higher-factuality outputs. Across FACTSCORE and a newly proposed medical subset, LUQ exhibits strong negative correlations with factuality (as low as –0.851 for Gemini Pro), and an ensemble strategy (LUQ-Ensemble) yields up to 5% gains in overall factuality over the best standalone model. These results indicate that uncertainty-aware selection during decoding can measurably reduce hallucinations in long-form answers while remaining model-agnostic.

2.2. Smishing in the Arabic Language

Detecting smishing in Arabic SMS presents unique challenges due to the language's complexity, rich morphology, and diverse accents. Most Machine Learning (ML) techniques for smishing detection rely on textual characteristics such as sender behavior, keyword analysis, and message structure. However, the lack of publicly available datasets further complicates Arabic smishing detection.

Advanced machine learning and deep learning methods have been investigated for detecting spam and phishing messages, particularly in SMS-based attacks. In [19], a hybrid CNN-LSTM model was proposed for SMS spam detection in Arabic and English messages using LSTM to capture sequence dependencies and convolutional layers to extract text features. The model trained on a dataset comprising 2,730 locally collected Arabic SMS messages and 5,574 English SMS messages from the UCI repository achieved an accuracy rate of 98.37%. Ghourabi [20] introduced SM-Detector, a BERT-based security model for identifying SMiShing communications in mobile settings. This model combines three detection techniques: BERT-based text categorization, regular expression analysis for suspicious terms and phone numbers, and malicious URL identification using VirusTotal. The model was evaluated on the UCI and local Arabic datasets, SM-Detector, and achieved high accuracy (99.63%), outperforming traditional machine learning classifiers like SVM, Random Forest, and Naïve Bayes. These experiments show the effectiveness of deep learning and hybrid algorithms in phishing detection, addressing issues related to dataset authenticity, language variety, and real-world generalization.

Ibrahim et al. [21] used a Random Forest model and Natural Language Processing (NLP) for Arabic phishing SMS messages detection. The dataset was originally compiled in English and then translated into Arabic using Google Translate's Deep Translator tool. It includes 4844 legitimate (ham) messages and 638 phishing (smishing) messages. The Random Forest model was evaluated alongside K-Nearest Neighbors (KNN), AdaBoost, and logistic regression, outperforming other models with an accuracy of 98.66%. The study showed the effectiveness of combining Arabic NLP, TF-IDF, and Random Forest for phishing detection. It also suggested future research using an Arabic dataset to improve detection accuracy.

While extensive research on phishing detection in emails, URLs, and other platforms demonstrates the flexibility of LLMs and deep learning models, these approaches pose a risk of concealing the specific challenges of Arabic smishing.

With only a few dedicated studies, such as [19], [20], [21], Arabic smishing detection remains largely unexplored compared to English or multilingual phishing detection. This gap highlights the novelty of the present study, which focuses on building and evaluating artificial Arabic SMS datasets specifically designed for smishing detection, rather than adapting tools developed for other purposes.

Recent advancements in transformer-based architectures and LLMs have further demonstrated promising results in phishing detection. Integrating NLP techniques with machine learning algorithms can enhance phishing detection in Arabic SMS, providing a more reliable and scalable defense against evolving phishing threats. Although recent sources from 2023–2024 [9], [10], [11], [12], [13], [14], [15], [16], [17], [18] provide valuable insights into the use of LLMs and advanced architectures, most of them stop short of addressing critical gaps: the absence of publicly available Arabic datasets, the challenge of dialectal diversity, and the difficulty of ensuring dataset authenticity. The main conclusion is that there is still no structured workflow for dataset generation and systematic evaluation in Arabic smishing detection research, rather than simply outlining methods. To fill this gap, this study presents a framework that enhances dataset diversity and realism by combining graph-based filtering and self-consistency techniques with a method for generating synthetic Arabic SMS messages.

We summarize the related work in table 1 according to their objectives, datasets, detection models, phishing platforms, and evaluation results.

Table 1. Related work summary

Ref.	Objective	Dataset	Detection Model	Phishing Platform	Accuracy
[18]	Propose LUQ framework for long-text uncertainty quantification in LLM outputs	FACTSCORE, PopQA, and medical subset	LUQ, LUQ-Ensemble	Long-form LLM outputs	Correlation with factuality –0.851 (Gemini Pro); factuality improved by up to 5%
[19]	SMS spam detection in both Arabic and English messages	UCI Repository, and a set of Arabic messages collected locally	CNN and LSTM	SMS	98.37%
[20]	Identify potential smishing attempts written in Arabic or English	UCI Repository, and a set of Arabic messages collected locally	BERT	SMS	99.36%
[11]	Examine the performance of GPT-4 and Gemini 1.5 in phishing detection by comparing them with DeBERTa	HuggingFace, Nazario, Nigerian, and synthetic data by GPT-4	DeBERTa V3, GPT-4, Gemini 1.5	Email, HTML, URL, SMS	DeBERTa 3: 93% on Nazario and Nigerian dataset
[9]	Evaluate robustness of ML vs. LLM under adversarial attacks	Phishing and benign webpages (multiple sources)	LR, RF and XGBoost	Web-based phishing	95%
[13]	Propose a theory-based prompt engineering method for smishing detection through data augmentation	SMS Phishing Benchmark	BERT, RoBERTa, DistilBERT, and ALBERT	Synthetic SMS augmentation	RoBERTa-large: 98.2%
[12]	Evaluate LLMs' role in generating and detecting phishing	Berkeley Phishing Examples Archive personalized data by GPT-4 and V-Tried	ChatGPT, Claude, Bard, and ChatLLaMA	Email	Claude: 75%

[14]	Enhance phishing dataset generation and detection	Email Spam Detection Dataset (classification), PhishingEmailData	DistilBERT and RoBERTa	Email	IPSDM (RoBERTa): 99% on balanced dataset
[21]	Proposing a phishing detection technique for Arabic SMS messages	Almeida dataset and pinterest.com	NLP and RF	SMS	98.66%
[15]	Improve the reliability of LLM generations by selecting the most reliable response from a variety of potential outputs.	GSM8K, MATH, GovReport, SummScreen, and TruthfulQA	PaLM 2-L, GPT-3.5-turbo	-	82.5% with GPT-3.5-turbo
[16]	Enhance long-form response recall by using ASC to select and combine relevant subparts from several generated responses to create an excellent composite response.	ASQA, QAMPARI, QUEST, ELI5	ChatGPT, Llama-2 (70B), SimCSE (RoBERTa-large), GTR-T5-XXL	-	47.01% on ASQA
[17]	Examines LLMs' ability to generate graphs under different situations	OGBG-MolHIV	GPT-4, GPT-3.5, LLaMA2-13B	-	GPT-4 92%

3. The Proposed Method

Our pipeline for Arabic smishing detection is organized into six main stages: (1) data preparation, including collection, cleaning, and feature extraction from real-world SMS; (2) construction of a knowledge graph to encode relationships among senders, keywords, URLs, and domains; (3) Conditional WGAN embedding generation to model diverse smishing patterns; (4) fine-tuning GPT-3.5-turbo on structured SMS instances; (5) synthetic SMS generation enhanced by self-consistency sampling and graph-based filtering; and (6) evaluation of the combined real and synthetic dataset using both ML (LR, RF, SVM) and DL (CNN, BERT) classifiers. Each stage builds sequentially on the previous one, ensuring that the final dataset is both diverse and realistic for robust model training. Every step builds on the one before it, ensuring a pipeline that is both technically solid and practically useful for detecting Arabic smishing in the real world.

3.1. Data Preparation

3.1.1. Real-World SMS Dataset Collection

The data preparation step is crucial for developing an effective Arabic smishing detection system. It provides a high-quality dataset for training and evaluating machine learning models. In this step, we aim to collect real-world SMS messages that represent both ham and smishing messages, ensuring linguistic diversity and validity. We distributed a Google Form to local participants, asking them to submit real SMS messages they had received anonymously, and kept the exact wording to preserve phishing indicators and language nuances. This approach ensured a representative dataset while following ethical considerations by protecting participants' privacy and without Personally Identifiable Information (PII). Before submission, each participant provided their informed consent, and answers were collected anonymously, without including names, phone numbers, or any other metadata that could identify a specific individual. Messages were further anonymized by removing sender IDs unless they contained organizational names associated with phishing patterns. To avoid unintentional abuse, potentially harmful phishing URLs were also either neutralized (for example, by changing ".com" to "[dot]com") or kept in a restricted-access, controlled research-only environment. These safety measures maintained the language and structural characteristics required for accurate smishing detection while guaranteeing compliance with ethical standards. We collected 6,000 SMS messages, comprising 3,000 smishing and 3,000 ham messages, to develop the Real_SMS dataset. This dataset plays a crucial role in training machine

learning models, enabling them to distinguish between safe and malicious communications with high accuracy. Furthermore, the collected real-world messages serve as a reference for generating synthetic data, which enhances dataset diversity and size. The quality and authenticity of the dataset directly impact model performance, ensuring that the detection system effectively generalizes to evolving smishing patterns. By leveraging this dataset, subsequent steps such as fine-tuning, model evaluation, and synthetic data generation are built upon a reliable and representative foundation, improving the robustness of the Arabic smishing detection framework.

3.1.2. Knowledge for Smishing Detection

To ensure that the synthetic Arabic SMS messages generated for smishing detection accurately reflect real-world phishing attempts, we incorporated domain knowledge into the dataset creation process. This knowledge was obtained through an in-depth analysis of real smishing campaigns, leading to the identification of key phishing patterns:

Common smishing keywords: This list of keywords represents the most frequently used words in smishing messages, designed to evoke a sense of urgency, anxiety, or excitement, prompting users to respond immediately [22]. Based on this definition, we processed our Real_SMS dataset to extract the most frequent smishing keywords (53 words) using TF-IDF and word frequency count. Some of the extracted words are ones like "تحقق" (verify), "حسابك" (your account), "جائزة" (prize), "رابط" (link), (suspended)"موقوف ".

Suspicious URL patterns: These are malicious URLs used in phishing messages that are frequently intended to imitate reliable websites or lead users to fake ones [23]. We fetched 700 phishing and legitimate URLs from PhishTank.com using their provided API. The dataset comprises three columns: URL, Class (0 = Not Phishing, 1 = Phishing), and Brand Name, where each row represents a web address labeled as either legitimate or phishing, with an associated brand, to be able to analyze common phishing URL patterns like bit.ly, paypa1.com, and secure-login.net.

3.1.3. Data Preprocessing and Exploratory Analysis

We begin by reading in two raw inputs: an Excel sheet of Real_SMS messages and a CSV file of URLs, and immediately launch an Exploratory Data Analysis (EDA) that unveils properties of our corpus. We first compute relevant statistics such as class balance (ham vs. smishing), URL presence ratios, candidate phishing keyword frequencies, and typical message length distributions. To graph these patterns, we create histograms of message lengths, bar plots of the topmost frequent smishing terms, and plots of URL presence versus class, all of which inform downstream threshold choices (e.g., our cosine-similarity cutoff in subsequent filtering). Throughout EDA, every such subtask is timed by a ResourceTracker, yielding reproducible logs and phase-duration plots. We then label and clean messages by lowercasing all text, stripping URLs, punctuation, and non-Arabic characters, then tokenizing and stripping Arabic stopwords. Labels are coded as 1 (smishing) and 0 (ham), and the resulting enriched dataset is written as enhanced_sms_data.csv. Finally, we extract a collection of numerical features, smishing keyword counts, URL counts, message length, punctuation counts, and digit counts, and output them as NumPy arrays (.npy files). We then perform an 80/20 stratified train/test split, also exporting text_train.csv and text_test.csv for use in our text-based models.

By incorporating real-world SMS data and background knowledge, we effectively fortified the dataset's credibility and efficiency in depicting real-world smishing activities. The product of this phase is a properly organized dataset that is augmented with phishing terms and URL patterns, serving as a valuable source of synthetic data generation for model training and enabling the detection model to accurately observe varied smishing patterns.

3.2. Knowledge-Graph Construction

By incorporating real-world SMS data and background knowledge, we effectively fortified the dataset's credibility and efficiency in depicting real-world smishing activities. The product of this phase is a properly organized dataset that is augmented with phishing terms and URL patterns, serving as a valuable source of synthetic data generation for model training and enabling the detection model to accurately observe varied smishing patterns.

To better understand hidden patterns in Arabic SMS messages, especially those related to smishing. we constructed a heterogeneous knowledge graph G= (V, E) using NetworkX. The graph was designed to capture semantic and structural relationships between senders, message content, keywords, URLs, and domains. We began by loading the preprocessed Real_SMS dataset, which includes message IDs, binary labels (ham or smishing), and extracted keywords. In parallel,

 ISSN 2723-6471

we incorporated a second dataset containing labeled URLs with phishing indicators and brand associations. By combining these inputs, we created an undirected graph integrating multiple layers of information.

The graph consisted of several types of nodes: senders derived from SMS metadata and URL reports, messages identified by unique IDs and binary classifications, keywords selected through TF-IDF (53 discriminative smishing terms), URLs extracted from embedded links, and domains parsed from each URL's hostname. Edges were then established to reflect semantic ties, connecting messages with their senders, linking messages to associated keywords and URLs, and finally linking URLs to their parent domains. This design enabled us to model not only message content but also delivery behavior and structural interactions within the SMS ecosystem. To quantify the influence of each node, we measured degree centrality: $\frac{\deg(v)}{V|-1|} = C_D(v)$. To further explore community structures, we applied Louvain modularity maximization:

$$\delta(c_i, c_j)\left[\frac{k_j\, k_i}{2m} - A_{ij}\right]\sum_{ij}\frac{1}{2m} = Q \quad (1)$$

A_{ij} represents the adjacency matrix, k_i and k_j are node degrees, m is the total number of edges, and δ (c_i, c_j) equals 1 if nodes i and j belong to the same community. This technique revealed clusters corresponding to distinct smishing strategies, such as banking credential theft, government service fraud, and telecom impersonation.

The final graph contained 3,131 nodes and 4,903 edges, distributed across 1,000 messages, 167 senders, 51 keywords, and 904 URLs. In addition to providing visual insights, this structure offers a strong basis for subsequent stages. A representative subgraph's visualization shows distinct clustering and cross-entity connectivity, confirming the graph's usefulness for data enrichment and interpretability.

3.3. Conditional WGAN Embedding Generation

To provide our language model with a diverse source of realistic but new contexts, we use a conditional Wasserstein GAN implemented in PyTorch that learns to generate SMS embeddings rather than unprocessed text. Initially, we transform each cleaned message into a vector of fixed length, either using TF-IDF or a pre-trained sentence encoder and rescale all values to the range of [–1, 1]. The architecture of the GAN consists of two core components. The generator accepts a randomly chosen noise vector alongside two conditioning inputs: the message label (ham vs. smishing) and the sender's centrality score extracted from our knowledge graph. Together, these form a 102-dimensional input that is processed through multiple linear layers with LeakyReLU activation functions. The output of the generator is a synthetic embedding designed to closely resemble the distribution of real SMS embeddings. Formally, the generation process can be expressed as:

$$\{ham, smishing\} \ni N(0, I), c{\sim}G(z, c), \;\; z = X_{gen} \quad (2)$$

z is a noise vector sampled from a normal distribution and c represents the conditioning variables.

The critic, in turn, evaluates both real and generated embeddings, incorporating the same two conditioning values to ensure consistency. Its architecture is also composed of stacked linear layers interleaved with LeakyReLU activations, ultimately producing a single scalar value that reflects the "realness" of the input. Through this adversarial setup, the critic learns to assign higher scores to authentic embeddings while penalizing synthetic ones that fail to capture realistic properties. This interplay gradually refines the generator, enabling it to produce high-quality embeddings that enrich the dataset with diverse yet plausible Arabic SMS representations.

We train these networks adversarially for 100 epochs with a batch size of 64, updating the critic five times per generator step and using a small learning rate (1×10^{-4}) plus weight clipping to stabilize training. A simple progress tracker logs both losses, which settle into stable bands after a few hundred iterations. We used t-SNE visualization to compare hundreds of embeddings from the trained generator with real embeddings, verifying the quality of the generated data. The significant overlap between synthetic and genuine embeddings within each class confirmed that the GAN could capture the real semantic manifold of Arabic SMS. These preserved embeddings were used as seeds to fine-tune GPT-3.5 Turbo, ensuring greater contextual realism and diversity in the generated messages.

 ISSN 2723-6471

3.4. Fine-Tuning a Pre-Trained Model on Real_SMS

Building on the embeddings generated by the WGAN, we fine-tuned a pre-trained GPT-3.5-turbo model to enhance its ability to classify and generate Arabic smishing content. Our curated Real_SMS dataset, containing labeled Arabic SMS messages (ham vs. smishing), served as the foundation. Each message included contextual metadata such as sender identity and phishing-related URLs.

To construct training samples, we combined the sender ID, raw text, relevant smishing keywords, and associated URLs into structured prompts, with the expected output being the message label (ham or smishing). The dataset (see table 2) was formatted in JSONL as required by OpenAI's fine-tuning API. Each JSONL line follows the format:

Table 2. Examples of Arabic SMS messages (smishing vs. ham) with extracted features

Class	URL	Keywords	Message	Sender
Smishing	http://fake-stc-support.com	تعليق، تحديث بيانات، رابط	عزيزي العميل، تم تعليق خدمتك مؤقتاً. يرجى تحديث بياناتك عبر الرابط التالي: http://fake-stc-support.com	STC
Ham	-	موعد، تطعيم، مركز اللقاحات	تذكرك بموعِد التطعيم في مركز اللقاحات بالرياض يوم الأربعاء.	وزارة الصحة

This structure enables the model to learn from both the message content and contextual indicators, such as the sender and URL. A dictionary is also built to map senders to known phishing domains, which is included during message construction to enrich the contextual signal.

The fine-tuning job is configured with specified hyperparameters, such as five epochs to prevent overfitting. It saves both time and funds on computation, a batch size of sixteen, to make it possible to fine-tune the model on hardware with limited resources by reducing memory usage [24], with a learning rate multiplier of 0.1, weight changes are kept minimal and under control, allowing the model to converge smoothly to a solution that achieves a compromise between task-specific adaptation and prior knowledge [25]. By seeding the fine-tuned GPT with embeddings from the WGAN, the model could leverage both real-world signals and synthetic diversity, improving its robustness in Arabic smishing detection and reducing the risk of bias toward limited training examples.

3.5. Generating Synthetic SMS with Self-Consistency and Graph-Based Filtering

Because real-world smishing datasets, especially in Arabic, are often limited in both size and diversity, we aimed to create a richer, more diverse collection of high-quality SMS messages for our experiments. This phase begins by tapping into the powerful latent representations generated by our conditional Wasserstein GAN (WGAN). These embeddings help us capture nuanced smishing semantics while maintaining sender-specific context. For each unique combination of sender and label, we sample a synthetic embedding from the trained WGAN generator. This approach ensures that the prompts reflect not only the typical patterns of smishing content but also the characteristics tied to individual senders. Next, we feed these embeddings into our fine-tuned GPT-3.5-turbo model, generating five variant messages per embedding. To promote consistency and coherence, we then select whichever version appears most frequently across completions. As an additional step to enhance authenticity, we append a randomly chosen real phishing URL to each synthetic smishing message. This process results in an initial set of 1,000 raw SMS examples. To improve the quality and reduce redundancy, we embed all messages using the multilingual `distiluse-base-multilingual-cased-v1` Sentence-Transformer. With these embeddings, we construct a similarity graph where edges connect any pair of messages that exceed a cosine similarity threshold of 0.75. By identifying clusters of near-duplicate content in this graph, we can eliminate redundant examples and retain only one representative per cluster. After this filtering step, we end up with a final synthetic corpus of 903 diverse and realistic messages. This workflow allows the WGAN to serve as a generator of richly conditioned seeds, while the combination of self-consistency sampling and graph-based pruning helps maintain both coherence and variety. This expanded dataset strikes a balance between realism and linguistic diversity, an essential foundation for training and evaluating smishing detection models in production-like scenarios.

3.5.1. Self-Consistency

To ensure variability in the synthetic data and guarantee consistency and reliability, the model was prompted to generate multiple candidate SMS messages for each sender under a given label (smishing or ham). The prompts included the sender's name and the specified message type, allowing the fine-tuned GPT-3.5-turbo model to produce diverse variations that still preserved contextual relevance [26]. These generated responses were stored for further refinement.

To enhance consistency, a frequency-based selection strategy was applied, where the response that appeared most frequently among the generated candidates was chosen as the final version. This approach ensures that the selected message is not only coherent but also statistically the most reliable across multiple generations [27]. Formally, the process can be expressed as:

$$k, \dots, 1 = \arg\max freq(m_i), \quad i = * m \tag{3}$$

$* m$ is the chosen message, $m_i$represents the candidate messages generated for a given sender–label combination, and k is the total number of generated samples.

To increase the realism of smishing messages, a phishing URL randomly selected from a curated dataset (PhishTank.com) was appended to each synthetic smishing message. By linking specific senders with known phishing URLs, the synthetic dataset more accurately captured real-world smishing patterns. [28]. Finally, the dataset was compiled to include the sender, the message text, and the corresponding label (ham or smishing). Before finalization, a graph-based filtering step was applied to eliminate redundant or overly similar messages, thereby ensuring greater diversity and preserving authenticity within the synthetic corpus.

3.5.2. Graph-Based Filtering

To guarantee diversity and quality, the graph-based filtering process was designed as a sequential workflow rather than a bullet-point list, where each step is described in a flowing narrative. The process begins with sentence embedding generation, where each synthetic SMS is converted into a dense vector representation using the distiluse-base-multilingual-cased-v1 Sentence Transformer model. This transformation ensures that messages are mapped to a high-dimensional semantic space in which similar messages are positioned closer to each other.

In the next stage, pairwise cosine similarity is computed to estimate textual similarity between messages. Cosine similarity measures the angular distance between two vectors in an inner product space and produces values in the range [1, -1] where 1 indicates identical vectors, 0 indicates no similarity (orthogonality), and -1 indicates complete opposition [29]. Formally, for two message embeddings u and v, cosine similarity is defined as:

$$\frac{u \,.\, v}{||u||||v||} = \cos 0 \tag{4}$$

Following this, a similarity threshold is applied. After preliminary experiments with values ranging from 0.70 to 0.85, a threshold of 0.75 was selected as an appropriate balance. Thresholds below 0.70 preserved many near-duplicate messages, while values above 0.80 risked removing legitimate variations. Therefore, messages exceeding the 0.75 threshold were flagged as redundant and prepared for filtering, while also maintaining computational efficiency [30]. Once redundancy was identified, the dataset was represented as a graph in which each message served as a node, and edges were created whenever similarity between messages surpassed the threshold. By analyzing this graph, clusters of duplicate messages could be detected effectively [31].Traversing the graph iteratively allowed for systematic removal of duplicates while preserving unique variations.

Finally, the remaining SMS messages were compiled into the final synthetic dataset. The dataset, after filtering, contained diverse yet realistic examples suitable for training robust smishing detection models.

By combining this graph-based filtering with the self-consistency mechanism, the resulting synthetic dataset remained both realistic and diverse, making it a valuable resource for machine learning and deep learning experiments.

3.6.Evaluating the Synthetic Dataset Using Machine Learning and Deep Learning Model

In this phase, we examined how adding 903 synthetically generated SMS messages impacted both detection performance and computational efficiency. First, we combined the filtered synthetic corpus with the original set of

ISSN 2723-6471

6000 real-world messages. Then, we split the combined dataset into training and testing sets using an 80/20 ratio to ensure a balanced evaluation. Next, the dataset is preprocessed, which involves cleaning and standardizing the text data to prepare it for model training. After, we trained several classification models on the combined dataset to examine how well each approach could distinguish between legitimate messages and smishing attempts. Finally, we analyzed the results using performance metrics, including accuracy, precision, recall, and F1-score. This evaluation shows how the synthetic examples contributed to overall model effectiveness and whether they introduced any trade-offs in computational demands. By carefully measuring both detection accuracy and resource consumption, we were able to assess the practical benefits of incorporating high-quality synthetic data into our smishing detection pipeline.

For machine learning models, the text data is preprocessed using Term Frequency-Inverse Document Frequency (TF-IDF) vectorization to transform SMS messages into numerical feature representations. TF-IDF assigns a weight to each word based on its frequency within a message relative to its frequency across the entire dataset, allowing the models to capture the significance of different terms [32]. The dataset is split into training (80%) and testing (20%) sets to ensure reliable performance evaluation. Three conventional ML models are trained and evaluated: LR: A linear classification model that predicts the probability of a message being smishing or ham (legitimate) using TF-IDF features. It is widely used for text classification due to its simplicity and interpretability [19]. RF: An ensemble learning model that constructs several decision trees and combines their predictions. It effectively captures non-linear relationships in the data and is robust against overfitting [33]. SVM: A robust classifier that maps input features into a high-dimensional space and determines an optimal hyperplane for distinguishing phishing and non-phishing messages. SVMs are known for their effectiveness in handling text-based classification tasks [34].

For deep learning models, text data undergoes tokenization to convert words into numerical sequences, followed by padding or truncation to ensure similar input lengths. Two DL architectures are applied: CNN: It applies convolutional layers to extract local patterns from text sequences, followed by pooling and dense layers for classification. CNNs have shown strong performance in text classification by identifying key n-grams and contextual relationships within messages [35]. BERT: Google created a deep learning model that transformed natural language processing by improving machines' comprehension of linguistic context. BERT reads text both ways, unlike previous models, which enables it to understand word meanings based on context. Before being refined for use in tasks such as text classification, it is pre-trained on extensive text corpora using tasks like masked word prediction and sentence connections [36].

To evaluate the performance of the machine learning models, we assessed accuracy, precision, recall, and F1-score to examine how effectively each model identified smishing messages without generating too many false alarms. For the deep learning models, we also observed accuracy and loss curves over the training epochs. These curves provided a clear sense of how each model learned over time and whether it was prone to overfitting or underfitting. By comparing results across different models, this evaluation provided valuable insights into how much the synthetic dataset contributed to building more reliable smishing detection systems. The findings not only emphasize the strengths and trade-offs of each classification approach but also provide practical guidance on how to optimize future phishing detection frameworks. This step helped clarify which strategies are most effective for training models that can keep up with the evolving tactics used in real-world smishing attacks.

All experiments were conducted on an HP Pavilion laptop with an Intel Core Ultra 7 processor, 16 GB of RAM, and a 1 TB PCIe Gen4 SSD to ensure consistency and accurately contextualize training times. Under these conditions, it took approximately 45 minutes to train the Conditional WGAN. However, due to external compute handling, fine-tuning GPT-3.5-turbo with the OpenAI API was completed in a few hours. CNN/BERT models took about 30 to 50 minutes each run on the GPU, while classical ML models (LR, RF, and SVM) were trained on the CPU in just a few minutes. Regarding scalability, the pipeline is designed in a modular fashion: TF-IDF preprocessing and graph-based filtering scale linearly with dataset size, while deep learning models benefit significantly from GPU acceleration when deployed in larger-scale environments.

4. Experimental Evaluation

This section presents the experimental evaluation of each phase in our Arabic smishing detection pipeline. It includes the exploratory analysis of real SMS and URL datasets, the construction of a knowledge graph to represent semantic relationships, the training and validation of a Conditional WGAN for synthetic embedding generation, and the performance of machine learning and deep learning models on the final augmented dataset. Each experiment is analyzed in terms of its contribution to improving detection accuracy, dataset diversity, and feature robustness.

4.1. Evaluation Measures

To evaluate the performance of the fine-tuned model and ML and DL models, we relied on standard classification metrics. The confusion matrix was first employed to display the classification performance of the model, showing four key measurements [37]. True Positives (TP) represent ham messages that were accurately predicted, while True Negatives (TN) correspond to smishing messages that were accurately identified. False Positives (FP) indicate ham messages that were misclassified as smishing, and False Negatives (FN) represent smishing messages that were incorrectly labeled as ham.

Beyond the confusion matrix, additional evaluation metrics were used. Accuracy measures the percentage of correctly classified instances out of all instances [38]. Calculated as: Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$. Precision, on the other hand, reflects the percentage of accurately predicted positive instances (smishing) among all messages predicted as positive [39]. It is calculated as: Precision = $\frac{TP}{TP+FP}$. Recall quantifies the percentage of true positive instances (smishing) that were successfully predicted [40]. It is computed as: Recall = $\frac{TP}{TP+FN}$. Finally, the F1-Score offers a balanced indicator of model performance by combining precision and recall in the form of a harmonic mean [41]. It is calculated as: F1 = $2 \times \frac{Precision \times Recall}{Precision+Recall}$.

4.2. Experimental Results

4.2.1. Exploratory Data Analysis (EDA)

Before diving into the detailed analysis, we first took time to summarize our input data. As shown in table 3, the SMS corpus includes 6,000 messages, evenly divided between legitimate (“ham”) and smishing messages, with 3,000 examples of each. Eight unique senders send these messages. In addition to the SMS data, the URL dataset contains 668 entries, split into 298 benign links and 370 malicious ones, collected from 20 different senders. We also identified a set of 53 keywords commonly associated with smishing attempts. Interestingly, there was no overlap in senders between the SMS and URL datasets; each sender appears in only one of the two sources. This clear separation provides a strong foundation for evaluating how well our models can generalize across different types of phishing-related content.

Table 3. Dataset statistics

Metric	Value
SMS messages	6000 (3000 ham, 3000 smishing)
Unique SMS senders	8
URLs	668 (298 benign, 370 malicious)
Unique URL senders	20
Smishing keywords	53
Sender overlap (SMS only)	8
Sender overlap (URL only)	20

 ISSN 2723-6471

Sender overlap (Both)	0

Our SMS dataset is perfectly balanced, with an even 50/50 split between legitimate messages ("ham") and smishing attempts. As illustrated in figure 1, this clear division ensures our models don't inherit any bias from an unbalanced training set. Looking closer at sender contributions, the top five senders are Amazon.sa, Absher, SNB, STC, and the Ministry of Interior account for more than 60% of all messages. However, none of these major services overwhelmingly drives the smishing rates. Instead, their smishing proportions cluster tightly, ranging from 0.58 to 0.60. In contrast, smaller senders, such as the Ministries of Education and Health, contribute very few smishing messages, with near-zero incidence. This difference highlights how smishing campaigns tend to focus on larger, more recognizable brands while leaving less prominent institutions largely untouched. As shown in figure 2, the overlap between SMS-only and URL-only datasets confirms that no sender appears in both sources, ensuring a clean separation of entities. Figure 3 further illustrates the relationship between smishing rate and message volume per sender, highlighting how larger organizations cluster tightly around similar smishing rates while smaller senders remain near zero.

Figure 1. SMS class distribution

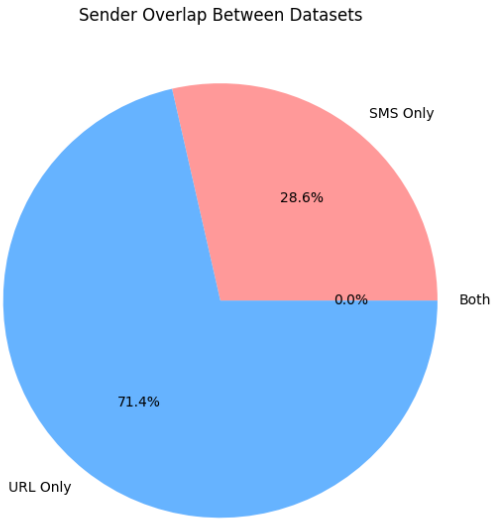

Figure 2. Sender overlaps between datasets

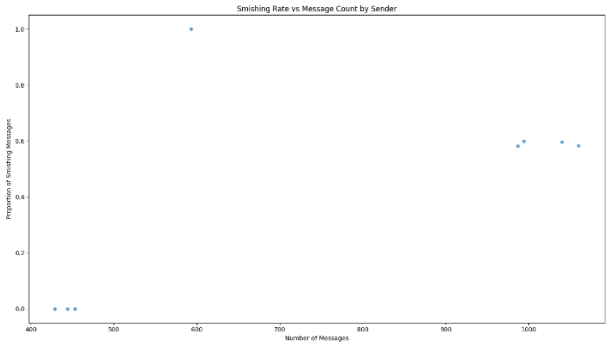

Figure 3. Smishing rate vs. message volume per sender

 ISSN 2723-6471

On average, the messages in our dataset are about 92 characters long. When we examine each category separately, ham messages tend to be slightly longer, averaging 93.73 characters, while smishing messages average slightly less at 90.16 characters. Both classes show a lot of overlap in their length distributions, so message length alone isn't a reliable indicator for distinguishing between legitimate and malicious content. As illustrated in figure 4, the distribution of message lengths by class confirms this overlap, while the boxplot comparison highlights only minor differences between ham and smishing messages.

A similar pattern emerges when examining words counts. The median word count for ham messages is 12, compared to 11 words for smishing messages. While these differences are small, they still provide some useful context for understanding the typical structure of SMS content in this dataset. As shown in figure 5, both the histogram and boxplot representations demonstrate how closely aligned the two classes are in terms of word count.

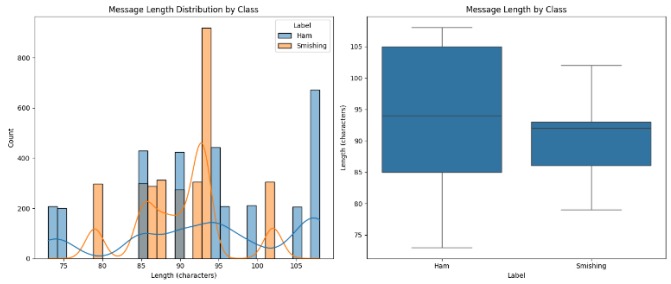

Figure 4. Message length by class

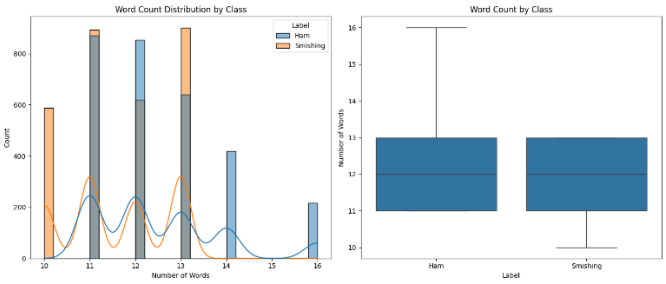

Figure 5. Word count by class

We identified the top 20 smishing keywords in the dataset using TF-IDF analysis as shown in figure 6. The most frequent term by far was "تحقق" ("verify"), which appeared in 1,230 messages. This highlights how often attackers try to create a sense of urgency around account verification. Other commonly used keywords included "حجز" ("reserve") and "قبل" ("before"), each showing up around 320 times. These terms reflect common tactics, like prompting recipients to confirm bookings or act quickly, that are designed to lower skepticism and drive engagement with malicious links.

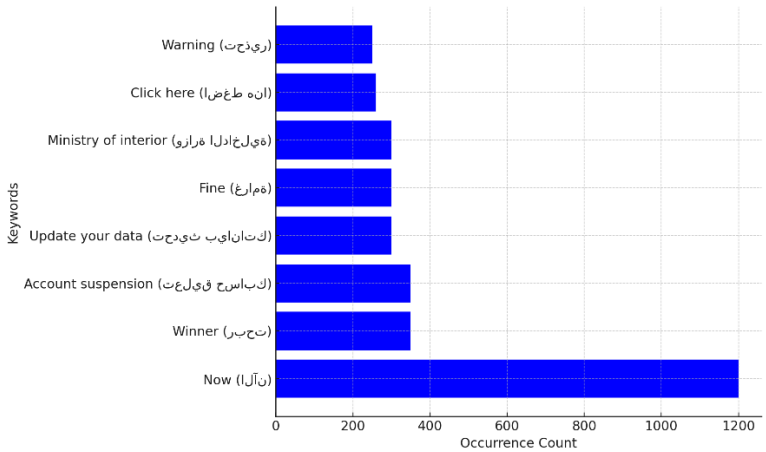

Figure 6. Top 20 Smishing keywords by occurrence

In our dataset, every URL, whether benign or malicious, was deliberately included to ensure consistent coverage. Out of 668 total URLs, 370 are classified as malicious and 298 as benign, as shown in figure 7. Upon closer examination of their structure, we found that malicious URLs were more likely to contain subdirectories (85% compared to 79% for benign links) but had fewer query parameters overall (12% vs. 27%). Figure 8 illustrates this contrast, showing higher use of subdirectories in malicious links while benign URLs contained more query parameters. At the domain level, benign URLs were most associated with Google.com, urlscan.io, and accounts.google.com. In contrast, malicious links frequently originated from domains such as docs.google.com and baemt3trrrascheash.dynv6.net, as well as a range of other phishing hosts. As shown in figure 9, the distribution of top benign and malicious domains highlights this difference clearly. On average, URL lengths were quite similar, with benign URLs averaging 76.87 characters and malicious ones averaging slightly less at 75.86 characters. Interestingly, malicious links tended to be slightly shorter in terms of median length, which may reflect an attempt to appear less suspicious. Figure 10 confirms this observation

 ISSN 2723-6471

by comparing the distributions of URL lengths between benign and malicious classes. These insights help paint a clearer picture of how attackers craft malicious links and where defensive models can focus their attention.

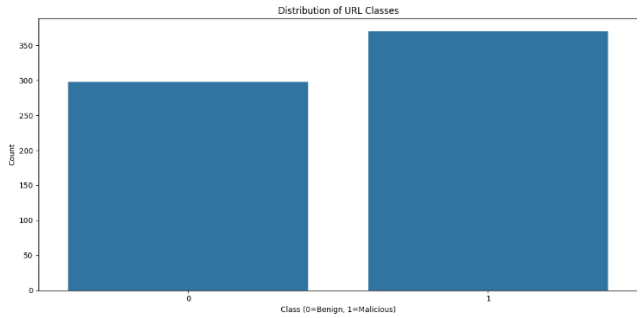

Figure 7. Distribution of URL classes

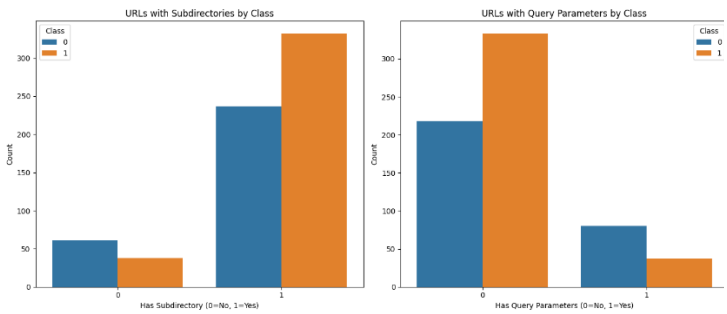

Figure 8. URL subdirectory and query parameter presence by class

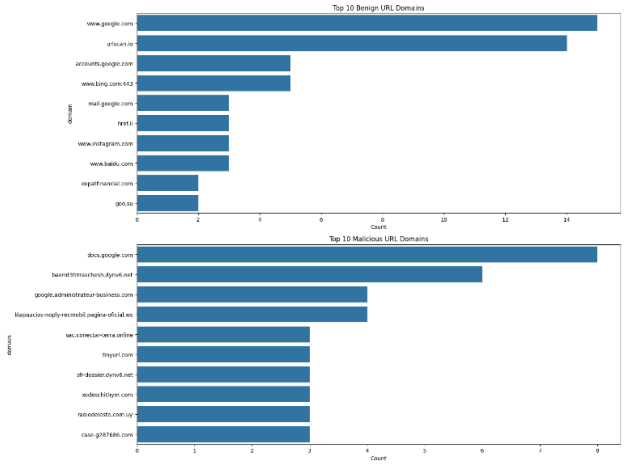

Figure 9. Top 10 benign vs malicious URL domains

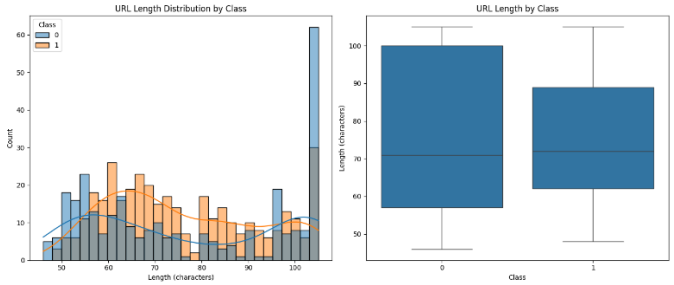

Figure 10. URL length by class

4.2.2. Knowledge-Graph Construction

In this phase, we built a heterogeneous graph, G, using NetworkX to capture relationships across different data types. The graph includes nodes representing messages, keywords, URLs, domains, and senders, all linked together through occurrence and inclusion edges. In total, G contains 7,245 nodes, 6,392 message nodes, 53 keyword nodes, 700 URL nodes, and 100 domain nodes, as well as 14,253 edges connecting them. When we ran a degree-centrality analysis, we found that the keyword "تحقق" ("verify") and the domain bit.ly stood out as the most highly connected nodes, highlighting their central roles in many phishing campaigns. To better understand the structural patterns, we applied Louvain community detection algorithm, which revealed eight distinct clusters. Each cluster groups together related keywords, URLs, and messages associated with senders. These communities give us clearer insight into how attackers coordinate content and infrastructure.

A sample visualization of G (figure 11) shows how the pieces fit together. In the graph, sender nodes (light blue) connect to both ham messages (green) and smishing messages (red). These, in turn, link to keywords (purple) and URLs (yellow), which finally connect to their corresponding top-level domains (orange). This view helps illustrate how smishing campaigns are organized and where defensive models might focus their attention.

ISSN 2723-6471

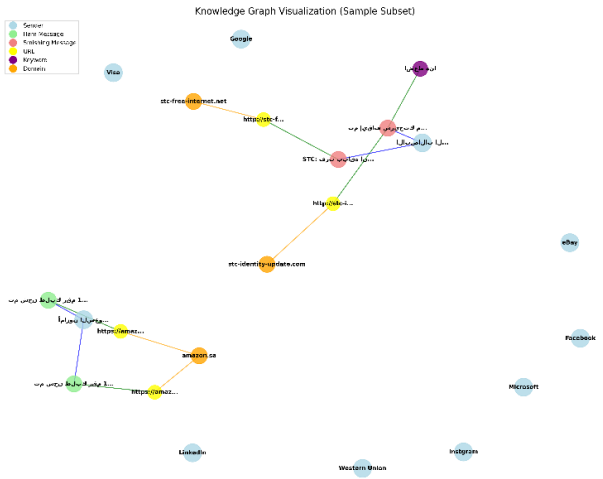

Figure 11. Sample subset of the knowledge graph

To better quantify the strength of association between each domain and phishing, we calculated the ratio of smishing-linked messages to the total number of messages mentioning that domain. As shown in figure 12, ten domains stood out with a perfect smishing ratio of 1.0. This means every message referencing them was fraudulent. Notable examples include snb-alerts-login.com, snb-check-alert.net, stc-free-internet.net, and absher-verify-login.net. These findings confirm that certain domains are used exclusively for smishing campaigns, highlighting clear targets for blocking lists or automated detection rules.

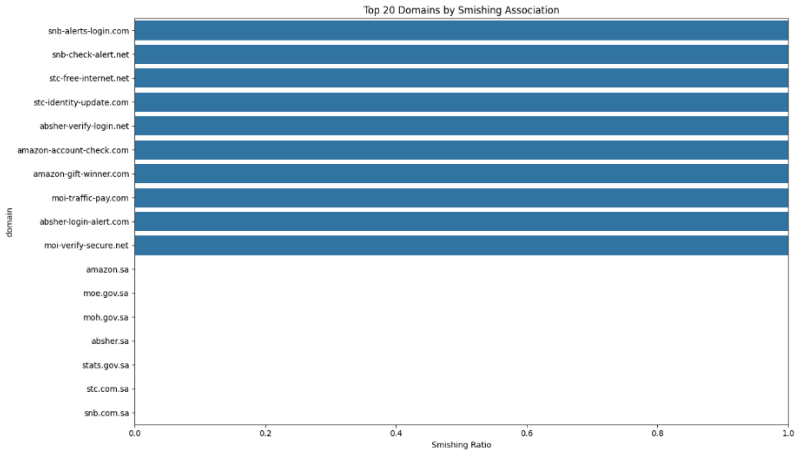

Figure 12. Top 20 domains ranked by smishing association ratio

Similarly, we also assessed the strength of association between specific keywords and smishing. For each term, we calculated the fraction of smishing messages that appeared relative to all messages containing that keyword. As shown in figure 13, eight keywords stood out with a perfect smishing ratio of 1.0. In other words, every message containing these terms was fraudulent. Examples include "تم تعليق حسابك" (Your account has been suspended), "ربحت" (You have won), "غرامة" (Fine), "اضغط هنا" (Click here), and "رابط" (Link). This highlights their strong predictive power and suggests these keywords could serve as reliable signals for flagging potential smishing attempts.

 ISSN 2723-6471

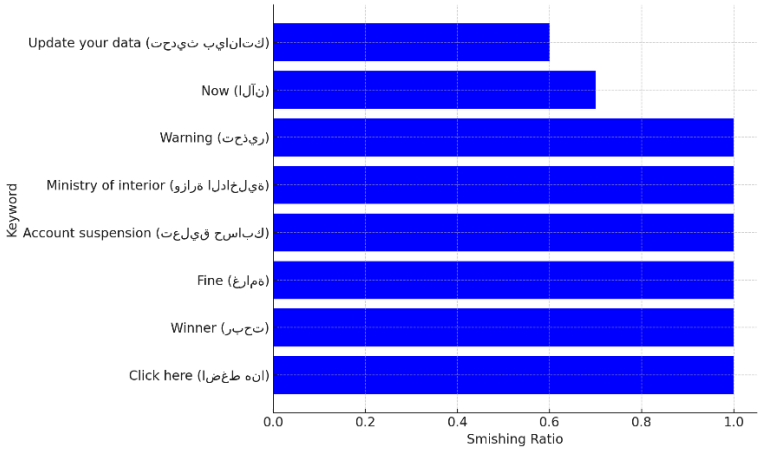

Figure 13. Top 20 keywords ranked by smishing association ratio

These centrality and association analyses confirm that our knowledge graph effectively captures both the structural patterns and key language cues of Arabic smishing campaigns. This comprehensive view provides essential features we can use to guide our WGAN embeddings and fine-tune prompt generation for the language model. In other words, by mapping out how senders, keywords, URLs, and domains connect, we're able to create richer, more realistic synthetic data that reflects real-world phishing behaviors.

4.2.3. Conditional WGAN Embedding Generation

We trained our conditional WGAN using normalized SMS embeddings, based on either TF-IDF or BERT vectors, for 100 epochs. This process included roughly 2,000 generator iterations, with a batch size of 64, and the critic was updated five times for every generator step. Figure 14 illustrates the generator and critic losses evolved over all iterations. After an initial warm-up phase of approximately 200 iterations, the critic loss quickly settled into a stable plateau of around -5. Meanwhile, the generator loss fluctuated within a narrow range of +2 to +3. These patterns indicate that the 1-Lipschitz constraint was maintained, and that adversarial training reached a healthy equilibrium without any signs of mode collapse. In short, the model was able to learn realistic representations while preserving stability throughout training.

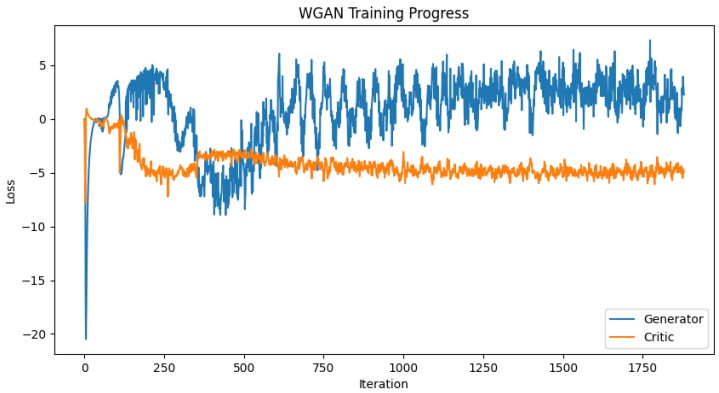

Figure 14. Generator and critic losses

To evaluate the quality of our embeddings, we sampled 200 vectors from the trained generator, 100 conditioned on ham labels, and 100 on smishing labels. We then projected them into two dimensions alongside 200 real embeddings using t-SNE for visualization (figure 15). The results were encouraging. The generated ham points (green ×) and smishing points (purple ×) clustered closely around their real counterparts (blue ● and red ●, respectively), while the two classes remained separated overall. This strong overlap indicates that the generator has successfully learned the underlying semantic structure of real SMS embeddings in a label-aware way. As a result, it's able to produce diverse, realistic latent seeds that can be used to drive downstream prompt generation for the language model.

 ISSN 2723-6471

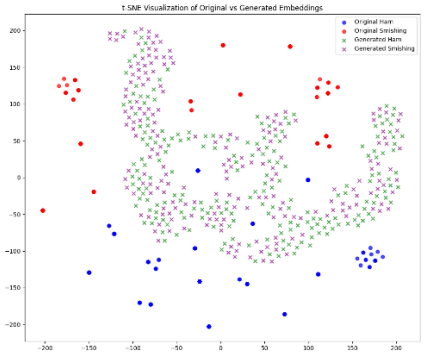

Figure 15. t-SNE projection of 200 original (●) and 200 generated (×) embeddings

Overall, the combination of stable adversarial losses and the strong overlap between real and generated clusters confirms that our conditional WGAN effectively learned a rich, label-conditioned embedding space for Arabic SMS. A closer examination of the t-SNE projection also reveals critical details. A few points show up at the class boundary, even though the generated and real embeddings mostly overlap within their respective classes. These examples imply that some smishing and ham messages use similar structural or lexical indicators, including the use of urgent language in legitimate service notifications. This partial overlap reveals possible sources of model confusion. It suggests that while the WGAN was successful in capturing the semantic manifold, it might not be able to fully resolve edge cases where malicious and legitimate messages converge. It's critical to identify these ambiguous regions since they provide practical difficulties for downstream classifiers and signal areas where more feature enrichment or adversarial training could improve detection robustness. The entire training and validation process took about 384 seconds to complete. Once finished, we saved the final synthetic embeddings and logged all timing details in our phase tracker to keep everything documented and reproducible. With this step complete, the embeddings are now ready to seed the next phase: generating synthetic SMS messages for further model development.

4.2.4. Arabic SMS Generation Process

To assess the consistency and diversity of the model, the synthetic Arabic SMS production procedure was run three times in this step. Using a refined language model seeded with embeddings from a Conditional WGAN, each iteration started by producing 1,000 messages. Following the implementation of self-consistency sampling and graph-based redundancy trimming to eliminate duplicates, the initial run produced 301 unique messages and took 6211.37 seconds to finish. The second and third runs took 6960.87 and 6989.32 seconds to complete, respectively, and generated 320 unique messages.

These findings show that the model can produce a variety of coherent Arabic SMS messages, with an average post-filtering yield of about 314 texts per run. To retain the final datasets quality and variety, an average of 31% of the generated messages were filtered away as duplicates or very similar.

4.2.5. ML and DL models results on the final dataset

The analysis of five classification models using Arabic SMS data showed that all methods performed exceptionally well, as demonstrated by confusion matrices and quantitative measures. Table 4 shows each model's comparative performance:

Table 4. Model Performance Comparison

Model	Accuracy	Precision	Recall	F1-Score
LR	0.9949	0.9942	0.9957	0.9950

RF	0.9949	0.9942	0.9957	0.9950
SVM	0.9957	0.9943	0.9971	0.9957
CNN	0.9942	0.9942	0.9942	0.9942
BERT	0.99	0.99	0.99	0.99

The Arabic SMS dataset was used to evaluate five machine learning and deep learning models, and the results showed that all classifiers performed well. With an accuracy and F1-score of 0.9957, SVM demonstrated strong and well-balanced classification capabilities, achieving the highest scores across all criteria. With an accuracy of 0.9949 and an F1-score of 0.9950, which showed only slight variations in predictive power, LR and RF performed slightly lower but still delivered good results. Comparing the CNN model to the best-performing models, it showed consistent but slightly worse performance, with an accuracy and F1-score of 0.9942, indicating steady but less ideal classification. Lastly, with an accuracy and F1-score of 0.99, BERT demonstrated its efficacy in differentiating between ham and smishing Arabic SMS texts, although being marginally less accurate than conventional machine learning models. Overall, the results indicate that both classical machine learning and advanced deep learning models are highly effective for this task, with SVM providing the best balance of precision and recall.

Strong classification performance is shown by Logistic Regression (figure 16), which accurately classified 683 Ham messages and correctly detected 691 Smishing messages. The model demonstrated strong discriminative capacity, with only 4 false positive errors (ham messages reported as smishing) and 3 false negative errors (smishing messages missed). These outcomes are consistent with the high accuracy (0.9949) of the model shown in table 3.

Random Forest (figure 17) achieves near-perfect recall in smishing detection, with only 1 false negative out of 694 smishing cases. With only 4 false positives, the confusion matrix shows 683 correctly classified ham messages (true positives) and 693 correctly identified smishing messages (true negatives). As seen in table 3, Random Forest's exceptional precision (0.9942) and recall (0.9957) balance emphasize its dependability for security-critical applications where the risk of missing smishing signals (false negatives) is higher than that of occasional false alarms. The model's high accuracy (0.9949) and F1-score (0.9950) can be explained by its low number of misclassifications (5 total mistakes), which makes it especially appropriate for SMS filtering systems that prioritize smishing detection.

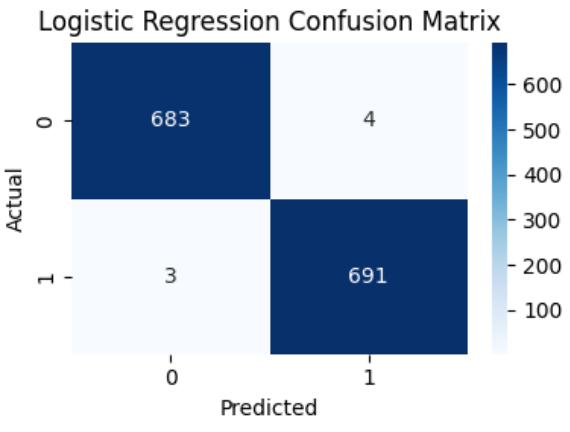

Figure 16. Logistic regression confusion matrix

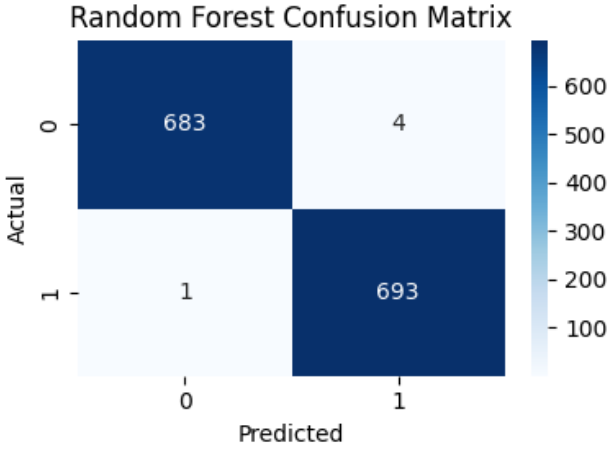

Figure 17. Random forest confusion matrix

The confusion matrix of SVM (figure 18), which displays 4 false positives, indicates that it has greater precision in SMS classification. With only 2 false negatives, the matrix shows 683 correctly identified ham messages (true positives) and 692 correctly detected smishing messages (true negatives). This outstanding performance, especially in reducing false alarms (precision = 0.9943), explains why SVM outperformed all other models in terms of accuracy

 ISSN 2723-6471

(0.9957) and F1-score (0.9957). With only 6 misclassifications out of 1,381 samples, SVM is beneficial for applications that require reducing the number of ham messages mistakenly flagged as smishing while maintaining robust fraud detection capabilities (recall = 0.9971).

With 691 true negatives, 683 true positives, and 7 total misclassifications (4 false positives and 3 false negatives), CNN (figure 19) displays balanced classification results. The confusion matrix shows that there is no bias toward either form of error, as evidenced by the precision and recall scores of this model being almost equal (both 0.9942).

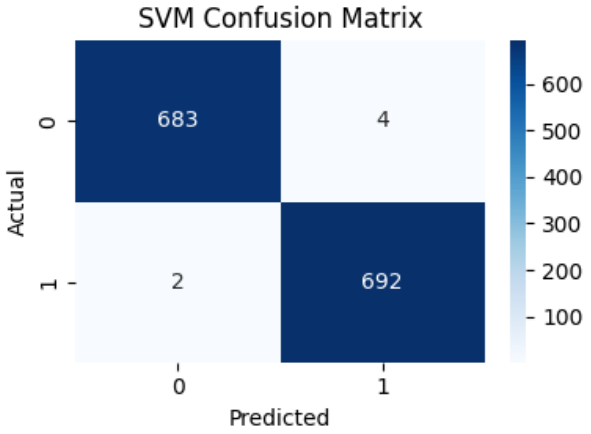

Figure 18. SVM confusion matrix

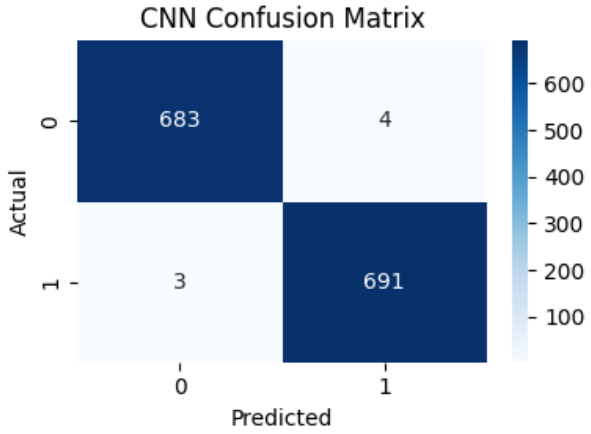

Figure 19. CNN confusion matrix

Despite being a general-purpose language model, BERT (figure 20) shows excellent classification performance, as evidenced by its confusion matrix, which has 683 true positives and 691 true negatives. The model's 7 total misclassifications mirrored the CNN's balanced error distribution, with just 4 incorrect positives and 3 false negatives. With a near-perfect recall for smishing detection (1.00) and exceptional precision (0.99 for Ham), this performance aligns with BERT's outstanding metrics (precision = 0.99, recall = 0.99) from its classification report. The confusion matrix shows BERT's unique strength in real-world applications where its rich linguistic knowledge helps limit both false alarms and missed detections, even while its overall accuracy (0.99) marginally lags below the top-performing classical ML models. Given that BERT's general design was not specifically tuned for this SMS classification task, the results are particularly noteworthy and demonstrate that transformer models can classify texts without requiring domain-specific tuning.

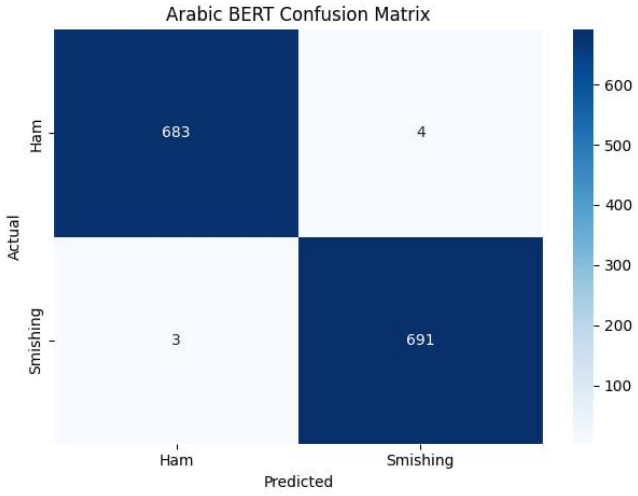

Figure 20. BERT confusion matrix

Even while every model achieved almost perfect results, a closer examination of the examples that were incorrectly classified indicates possible blind spots. Most false positives (ham mislabeled as smishing) occurred in legitimate service messages (such as account updates or appointment reminders) that included URLs and keywords commonly

 ISSN 2723-6471

associated with phishing. This implies that even in benign circumstances, the models may occasionally be overly sensitive to the existence of suspicious terms or links. On the other hand, the few false negatives (smishing mislabeled as ham) frequently contained URLs that were highly obfuscated or shortened, making the malicious intent less evident in the surface text. These errors highlight the importance of incorporating contextual semantics and conducting more thorough URL analysis in future research. By understanding these edge cases, the study provides a clearer picture of model limitations and potential directions for improving generalizability to the real-world SMS stream.

These patterns also have useful implications. False negatives present immediate security threats by enabling phishing efforts to go unnoticed, while false positives could undermine user trust if legitimate messages are consistently identified as malicious, even though the overall misclassification rate is extremely low. This trade-off emphasizes how crucial it is to balance recall and precision based on deployment scenarios, such as giving high recall priority in security-critical systems. Furthermore, the visualization analysis's findings are consistent with the concentration of errors around URL-related features, which raises the possibility that classifiers overfit to link cues. By addressing these flaws with feature masking, contextual embeddings, or adversarial training, robustness would be increased, and performance metrics could more accurately reflect real-world circumstances.

5. Discussion

The results of the pipeline for Arabic SMS generation and classification show the advantages and disadvantages of the suggested strategy. Three rounds of message synthesis using a language model seeded with Conditional WGAN embeddings yielded an average of 314 unique messages per run, out of 1,000 initially generated, following the application of self-consistency sampling and graph-based redundancy filtering. Although this indicates that a significant amount of semantically valid Arabic SMS content is consistently generated by the method, efficiency issues are raised by the low percentage (approximately 31%). A large proportion of generated messages were filtered out due to redundancy or semantic similarity, highlighting the trade-off between ensuring dataset diversity and maintaining computational efficiency. This suggests that future work should explore adaptive filtering thresholds or reinforcement learning approaches to improve the balance between diversity and yield without compromising quality. The stability and repeatability of the generation process are supported by the time consistency across runs, which ranges from approximately 6,200 to 7,000 seconds, confirming the robustness of the underlying framework.

On the classification side, both traditional machine learning and deep learning methods performed well in differentiating between ham and smishing messages. With just six misclassifications out of 1,381 messages, Support Vector Machine (SVM) achieved the best accuracy (0.9957) and F1-score (0.9957), making it especially useful for reducing false positives and preserving user’s trust. However, it is essential to interpret these results cautiously, as the training data was largely synthetic, which poses a potential risk of overfitting. Looking ahead, our focus will be on testing model performance in real-world scenarios to evaluate how well it generalizes to unseen Arabic SMS messages. High performance was also achieved by Random Forest (RF) and Logistic Regression (LR), with RF achieving nearly perfect recall for smishing (0.9986), which is advantageous in high-risk applications, as missing phishing messages has more serious consequences.

Interestingly, SVM consistently outperformed the deep learning models (CNN and BERT). This can be explained by the high-dimensional, sparse feature space derived from TF-IDF and keyword-based representations, where linear classifiers such as SVM excel at separating classes with clear lexical cues. In contrast, CNN and BERT rely on capturing deeper semantic structures and typically require larger datasets and more extensive hyperparameter optimization to realize their full potential. In this study, CNN was trained with standard filter sizes and dropouts, while BERT was fine-tuned for a limited number of epochs. Although these settings provided competitive performance (accuracy ~0.99), a more exhaustive hyperparameter search, such as tuning learning rates, layer configurations, and fine-tuning strategies could further improve their results.

Another important consideration is the role of URLs in classification. In the generated smishing messages, phishing-style URLs were explicitly appended to reflect real-world attack patterns. This design decision enhances realism, but it may also provide classifiers with potential shortcuts that allow them to rely heavily on the presence of suspicious URLs rather than developing a deeper understanding of contextual and linguistic patterns. This could partially explain

the near-perfect accuracy observed in models such as SVM. A more balanced evaluation would require experiments that mask or normalize URLs, thereby testing whether classifiers can still effectively detect smishing when explicit link cues are removed.

Our results confirm the viability of both ML and DL classifiers in identifying Arabic smishing messages and demonstrate the resilience of the SMS generating process. Carefully chosen synthetic data can be a valuable tool for supplementing small real-world datasets, and classification models, particularly SVM, show good generalization over a wide range of message types. Through early smishing detection in Arabic SMS communication, this comprehensive pipeline offers a scalable and automated solution for enhancing mobile security.

However, several limitations remain. First, the current study focuses primarily on Modern Standard Arabic, while dialectal variations widely used in everyday communication may reduce detection accuracy if not explicitly addressed. Second, adversarial smishing strategies, such as the use of obfuscated URLs, code-switching between Arabic and English, or deliberate insertion of spelling variations, were not extensively tested and could pose challenges for classifiers trained on standard patterns. Third, the pipeline has been validated on SMS data; however, cross-platform generalization to messaging applications like WhatsApp, Telegram, or RCS remains an open issue, particularly since formatting, media attachments, and user behaviors differ across platforms.

5.1. Comparison with Related Work

To further highlight the uniqueness of this study, we compared our dataset and methodology with three recent works on Arabic smishing and spam detection. Table 5 summarizes the main differences.

Table 5. Comparison with existing Arabic smishing works

Study	Dataset	Focus	Methodology	Synthetic Augmentation / Accuracy
[19]	Small Arabic/English SMS dataset	Spam detection (general, not smishing-specific)	CNN + LSTM	✗ / ~98.37%
[20]	Arabic/English SMS with phishing URLs	Smishing detection	Regex + URL inspection + BERT	✗ / 99.63%
[21]	Translated English → Arabic dataset (638 phishing, 4844 ham)	Smishing detection	NLP + Random Forest	✗ / 98.66%
Our Study	6,000 real Arabic SMS (3,000 ham, 3,000 smishing) + 903 synthetics	Smishing detection (Arabic, realistic)	Conditional WGAN + GPT fine-tuning + self-consistency + graph filtering + ML/DL	✓ / 99.57% (SVM)

Prior studies have made significant contributions to Arabic SMS security, but face key limitations. [19] did not address phishing URLs or the complexity of the Arabic language; instead, it focused on spam detection rather than smishing specifically. Despite its extremely high accuracy, [20] was mostly dependent on URL-based rules, which raises concerns about robustness if attackers obfuscate links. A translated dataset was used in [21], which results in semantic distortions and misses the nuances of Arabic phishing attempts.

In contrast, our work used a generative pipeline (Conditional WGAN + GPT-3.5-turbo fine-tuning), preserved authentic phishing URLs, and collected 6,000 real Arabic SMS messages that were balanced between ham and smishing. By applying graph-based redundancy filtering and self-consistency sampling techniques, we generated an additional 903 realistic and varied synthetic messages. To the best of our knowledge, this is the first Arabic smishing work to incorporate synthetic data generation into detection algorithms. This improves linguistic realism and scalability while addressing the lack of labeled datasets.

Although the comparative results show that typical ML and DL models perform well in every study, our approach is new in the way it constructs and augments datasets. In contrast to previous methods, we provide a replicable process

for generating high-quality Arabic smishing datasets in addition to evaluating classifiers, providing a basis for further multilingual and cross-platform research.

6. Conclusion

This study proposed a structured pipeline for generating and classifying Arabic SMS texts. A controlled redundancy filtering mechanism improved data quality, and the pipeline effectively generated a steady volume of realistic and varied messages by combining Conditional WGAN-based semantic seeding with a tuned language model. Both deep learning and conventional machine learning models demonstrated remarkable performance on the generated dataset, according to the experimental evaluation. The most successful model among them was Support Vector Machine, which produced the best performance metrics with the least amount of misclassification. Even though deep learning models like CNN and BERT had somewhat lower accuracy, they were still quite helpful. They showed good generalization, especially in real-world scenarios where nuanced language understanding is beneficial.

Overall, the findings support the viability of generating synthetic data for low-resource languages, such as Arabic, and demonstrate how generative models and strong classifiers can be combined to create dependable, scalable, and flexible security solutions. In addition to offering a replicable approach, this work establishes a framework for future studies in multilingual smishing detection and the wider use of AI in reducing online threats.

The models' generalizability and resilience would be further improved for future studies by using real-world labeled Arabic SMS data from government and telecom sources. Furthermore, by replacing contextual embeddings from XLM-R for the current embedding step, multilingual models like XLM-R can be included into the pipeline, enabling cross-lingual transfer and improved handling of mixed Arabic–English messages. Likewise, domain-adaptive fine-tuning of BERT can be integrated as a subsequent step following initial pre-training, before final fine-tuning on the Real_SMS dataset. In this phase, the model is adjusted to Arabic phishing-related corpora. By following these specific steps, the pipeline will be able to better capture semantic nuances and adapt to adversarial strategies across various dialects and domains. Lastly, building real-time deployment frameworks and extending the pipeline to handle multimodal SMS (such as those with URLs, emoticons, or attachments) would be crucial steps toward actual deployment in mobile security systems.

7. Declarations

7.1. Author Contributions

Conceptualization: A.A., M.A., H.S., M.Ak., and N.A.; Methodology: M.A.; Software: A.A.; Validation: A.A., M.A., H.S., M.Ak., and N.A.; Formal Analysis: A.A., M.A., H.S., M.Ak., and N.A.; Investigation: A.A.; Resources: M.A.; Data Curation: M.A.; Writing—Original Draft Preparation: A.A., M.A., H.S., M.Ak., and N.A.; Writing—Review and Editing: M.A., A.A., H.S., M.Ak., and N.A.; Visualization: A.A.; All authors have read and agreed to the published version of the manuscript.

7.2. Data Availability Statement

The data presented in this study are available on request from the corresponding author.

7.3. Funding

The project was funded by KAU Endowment (WAQF) at king Abdulaziz University, Jeddah, Saudi Arabia. The authors, therefore, acknowledge with thanks WAQF and the Deanship of Scientific Research (DSR) for technical and financial support.

7.4. Institutional Review Board Statement

Not applicable.

7.5. Informed Consent Statement

Not applicable.

7.6. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Z. Alkhalil, C. Hewage, L. Nawaf, and I. Khan, "Phishing attacks: A recent comprehensive study and a new anatomy," *Frontiers in Computer Science*, vol. 3, no. 1, pp. 1-20, 2021, doi: 10.3389/fcomp.2021.563060.

[2] D. Goel and A. K. Jain, "Mobile phishing attacks and defence mechanisms: State of art and open research challenges," *Computers and Security*, vol. 73, no. 1, pp. 519–544, 2018, doi: 10.1016/j.cose.2017.12.006.

[3] F. Alhayan, H. T. Himdi, and B. Alharbi, "Unveiling deception in Arabic: Optimization of deceptive text detection across formal and informal genres," *IEEE Access*, vol. 12, no. 5, pp. 94216–94230, 2024, doi: 10.1109/ACCESS.2024.3424531.

[4] N. Q. Do, A. Selamat, O. Krejcar, E. Herrera-Viedma, and H. Fujita, "Deep learning for phishing detection: Taxonomy, current challenges and future directions," *IEEE Access*, vol. 10, no. 4, pp. 36429–36463, 2022, doi: 10.1109/ACCESS.2022.3151903.

[5] A. Kumar, "The Role of Synthetic Data in Advancing AI Models: Opportunities, Challenges, and Ethical Considerations," *Journal of Artificial Intelligence and General Science (JAIGS)*, vol. 2024, no. 1, pp. 443–459, 2024, doi: 10.60087/jaigs.v5i1.256.

[6] M. U. Hadi, S. A. Malik, M. Z. Raja, A. A. Awan, M. A. Baig, M. I. Shah, and M. A. Khan, "A survey on large language models: Applications, challenges, limitations, and practical usage," *Authorea Preprint*, vol. 2023, no. 2, pp. 1–25, 2023.

[7] E. Bonner, R. Lege, and E. Frazier, "Large Language Model-Based Artificial Intelligence in the Language Classroom: Practical Ideas for Teaching," *Teaching English with Technology*, vol. 2023, no. 1, pp. 23–41, 2023.

[8] W. Kasri, Y. Himeur, H. A. Alkhazaleh, S. Tarapiah, S. Atalla, W. Mansoor, and H. Al-Ahmad, "From vulnerability to defense: The role of large language models in enhancing cybersecurity," *Computation*, vol. 13, no. 2, pp. 1-30, 2025, doi: 10.3390/computation13020030.

[9] A. Kulkarni, V. Balachandran, D. M. Divakaran, and T. Das, "From ML to LLM: Evaluating the Robustness of Phishing Web Page Detection Models against Adversarial Attacks," *Digital Threats: Research and Practice*, vol. 6, no. 2, pp. 1–25, Jun. 2025, doi: 10.1145/3737295.

[10] Q. Qi, Y. Luo, Y. Xu, W. Guo, and Y. Fang, "SpearBot: Leveraging large language models in a generative-critique framework for spear-phishing email generation," *Information Fusion*, vol. 122, no. 3, pp. 1-16, 2025, doi: 10.1016/j.inffus.2025.103176.

[11] S. Mahendru and T. Pandit, "SecureNet: A Comparative Study of DeBERTa and Large Language Models for Phishing Detection," in *Proceedings of the 2024 IEEE 7th International Conference on Big Data and Artificial Intelligence (BDAI)*, vol. 2024, no. Jul., pp. 160–169, 2024, doi: 10.1109/BDAI62182.2024.10692765.

[12] F. Heiding, B. Schneier, A. Vishwanath, J. Bernstein, and P. S. Park, "Devising and Detecting Phishing Emails Using Large Language Models," *IEEE Access*, vol. 12, no. 1, pp. 42131–42146, 2024, doi: 10.1109/ACCESS.2024.3375882.

[13] H. S. Shim, H. Park, K. Lee, J.-S. Park, and S. Kang, "Data Augmentation for Smishing Detection: A Theory-based Prompt Engineering Approach," in *Companion Proceedings of the ACM Web Conference 2024*, Singapore, Singapore: ACM, vol. 2024, no. May, pp. 1327–1328, doi: 10.1145/3589335.3651903.

[14] S. Jamal, H. Wimmer, and I. H. Sarker, "An improved transformer-based model for detecting phishing, spam and ham emails: A large language model approach," *Security and Privacy*, vol. 7, no. 5, pp. 1-22, Sept. 2024, doi: 10.1002/spy2.402.

[15] X. Chen, Y. Zhang, S. Wu, L. Song, and C. Ma, "Universal Self-Consistency for Large Language Model Generation," *arXiv preprint arXiv*:2311.17311, vol. 2023, no. Nov., pp. 1-12, Nov. 2023, doi: 10.48550/arXiv.2311.17311.

[16] R. Thirukovalluru, Y. Huang, and B. Dhingra, "Atomic Self-Consistency for Better Long Form Generations," *arXiv preprint arXiv*:2405.13131, vol. 2024, no. may., pp. 1-12, May 2024, doi: 10.48550/arXiv.2405.13131.

[17] Y. Yao, Z. Xu, Y. Chen, T. Wang, and J. Tang, "Exploring the Potential of Large Language Models in Graph Generation," *arXiv preprint arXiv*:2403.14358, vol. 2024, no. Mar., pp. 1-12, Mar. 2024, doi: 10.48550/arXiv.2403.14358.

[18] C. Zhang, F. Liu, M. Basaldella, and N. Collier, "LUQ: Long-text Uncertainty Quantification for LLMs," *arXiv preprint arXiv*:2403.20279, vol. 2024, no. Oct., pp. 1-12, Oct. 2024, doi: 10.48550/arXiv.2403.20279.

[19] A. Ghourabi, M. A. Mahmood, and Q. M. Alzubi, "A Hybrid CNN-LSTM Model for SMS Spam Detection in Arabic and English Messages," *Future Internet,* vol. 12, no. 9, pp. 156-169, Sept. 2020, doi: 10.3390/fi12090156.

[20] A. Ghourabi, "SM-Detector: A security model based on BERT to detect SMiShing messages in mobile environments," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 24, pp. 1-22, Dec. 2021, doi: 10.1002/cpe.6452.

[21] A. Ibrahim, S. Alyousef, H. Alajmi, R. Aldossari, and F. Masmoudi, "Phishing Detection in Arabic SMS Messages using Natural Language Processing," in *Proceedings of the 2024 Seventh International Women in Data Science Conference at Prince Sultan University (WiDS PSU)*, Riyadh, Saudi Arabia: IEEE, vol. 2024, no. Mar., pp. 141–146, 2024, doi: 10.1109/WiDS-PSU61003.2024.00040.

[22] E. O. Yeboah-Boateng and P. M. Amanor, "Phishing, SMiShing and Vishing: An assessment of threats against mobile devices," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 5, no. 4, pp. 297–307, 2014.

[23] S. Mishra and D. Soni, "Implementation of 'Smishing Detector': An Efficient Model for Smishing Detection Using Neural Network," *SN Computer Science*, vol. 3, no. 3, pp. 189-198, May 2022, doi: 10.1007/s42979-022-01078-0.

[24] H. Cai, C. Gan, L. Zhu, and S. Han, "TinyTL: Reduce memory, not parameters for efficient on-device learning," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, no. 1, pp. 11285–11297, 2020.

[25] L. Shen, Y. Sun, Z. Yu, L. Ding, X. Tian, and D. Tao, "On Efficient Training of Large-Scale Deep Learning Models: A Literature Review," *arXiv preprint* arXiv:2304.03589, vol. 2023, no. Apr., pp. 1-12, Apr. 2023, doi: 10.48550/arXiv.2304.03589.

[26] X. Liang, Z. Wang, H. Liu, Q. Zhou, and Y. Chen, "Internal Consistency and Self-Feedback in Large Language Models: A Survey," *arXiv preprint* arXiv:2407.14507, vol. 2024, no. 1, pp. 1-12, Sept. 2024, doi: 10.48550/arXiv.2407.14507.

[27] C. Ye, J. Yang, and Y. Mao, "User identification for knowledge graph construction across multiple online social networks," *Alexandria Engineering Journal*, vol. 73, no. 1, pp. 145–158, 2023, doi: 10.1016/j.aej.2022.09.053.

[28] J. Francia, D. Hansen, B. Schooley, M. Taylor, S. Murray, and G. Snow, "Assessing AI vs Human-Authored Spear Phishing SMS Attacks: An Empirical Study Using the TRAPD Method," *arXiv preprint* arXiv:2406.13049, vol. 2024, no. Jun., pp. 1-12, Jun. 2024, doi: 10.48550/arXiv.2406.13049.

[29] K. Zhou, K. Ethayarajh, D. Card, and D. Jurafsky, "Problems with Cosine as a Measure of Embedding Similarity for High Frequency Words," *arXiv preprint* arXiv:2205.05092, vol. 2022, no. May., pp. 1-12, May 2022, doi: 10.48550/arXiv.2205.05092.

[30] Y. Li, J. Wang, B. Pullman, N. Bandeira, and Y. Papakonstantinou, "Index-based, high-dimensional, cosine threshold querying with optimality guarantees," *Theory of Computing Systems*, vol. 65, no. 1, pp. 42–83, 2021, doi: 10.1007/s00224-020-09980-2.

[31] A. H. Osman and O. M. Barukub, "Graph-based text representation and matching: A review of the state of the art and future challenges," *IEEE Access*, vol. 8, no. 1, pp. 87562–87583, 2020, doi: 10.1109/ACCESS.2020.2993191.

[32] F. Lan, "Research on Text Similarity Measurement Hybrid Algorithm with Term Semantic Information and TF-IDF Method," Advances in Multimedia, vol. 2022, no. 1, pp. 1–11, Apr. 2022, doi: 10.1155/2022/7923262.

[33] A. Abdulhafedh, "Comparison between common statistical modeling techniques used in research, including: Discriminant analysis vs logistic regression, ridge regression vs LASSO, and decision tree vs random forest," *Open Access Library Journal,* vol. 9, no. 2, pp. 1–19, 2022, doi: 10.4236/oalib.1108352.

[34] O. A. Adewumi and A. A. Akinyelu, "A hybrid firefly and support vector machine classifier for phishing email detection," *Kybernetes*, vol. 45, no. 6, pp. 977–994, 2016, doi: 10.1108/K-09-2015-0232.

[35] M. Umer, M. Qamar, M. Arif, M. Shoaib, and S. M. Anwar, "Impact of convolutional neural network and FastText embedding on text classification," *Multimedia Tools and Applications*, vol. 82, no. 4, pp. 5569–5585, Feb. 2023, doi: 10.1007/s11042-022-13459-x.

[36] M. V. Koroteev, "BERT: A Review of Applications in Natural Language Processing and Understanding," *arXiv preprint* arXiv:2103.11943, vol. 2021, no. Mar., pp. 1-12, Mar. 2021, doi: 10.48550/arXiv.2103.11943.

[37] D. Dharrao, P. Gaikwad, S. V. Gawai, A. M. Bongale, K. Patel, and A. Singh, "Classifying SMS as Spam or Ham: Leveraging NLP and Machine Learning Techniques," *International Journal of Safety and Security Engineering*, vol. 14, no. 1, pp. 45–56, 2024.

[38] P. Galdi and R. Tagliaferri, "Data mining: Accuracy and error measures for classification and prediction," in *Encyclopedia of Bioinformatics and Computational Biology*, vol. 1, no. 1, pp. 431–436, 2018, doi: 10.1016/B978-0-12-809633-8.20307-9.

[39] M. K. Mehmood, H. Arshad, M. Alawida, and A. Mehmood, "Enhancing Smishing Detection: A Deep Learning Approach for Improved Accuracy and Reduced False Positives," *IEEE Access*, vol. 12, no. 1, pp. 115230–115245, 2024, doi: 10.1109/ACCESS.2024.10684195.

[40] G. M. Foody, "Challenges in the real world use of classification accuracy metrics: From recall and precision to the Matthews correlation coefficient," *PLOS ONE*, vol. 18, no. 10, pp. 1-12, 2023, doi: 10.1371/journal.pone.0291908.

[41] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, pp. 1-6, Dec. 2020, doi: 10.1186/s12864-019-6413-7.