# MACHINE LEARNING APPROACHES FOR INVESTIGATING TECHNIQUES IN SMS SPAM DETECTION

***Mustafa Mohiuddin Amer*[1], *Mohammed Abbas Qureshi*[2], *Sridhar Gummalla*[3]**

*[1] PG Scholar, Department of Computer Science and Engineering, Shadan College of Engineering & Technology, Hyderabad, Telangana, India.* amer.mustafa1121@gmail.com

*[2]Associate Professor, Department of Computer Science and Engineering, Shadan College of Engineering & Technology, Hyderabad, Telangana, India* mohammedabbas09@yahoo.com

*[3]Professor, Department of Computer Science and Engineering Shadan College of Engineering & Technology, Hyderabad, Telangana, India.* Sridhar_gummalla@yahoo.com

**ABSTRACT:**
The permanence of SMS spam still presents serious problems, so it is necessary to create efficient detection systems that can manage the increasingly complex evasion tactics used by spammers. By offering a thorough SMS spam filtering system [14] that makes use of machine learning models—with a particular emphasis on Long Short-Term Memory (LSTM) networks—this study tackles these issues. We provide a new SMS dataset message that is the largest publicly available SMS spam dataset to date, consisting of 39% spam and 61% real (ham) messages. The evolution of spam was analysed longitudinally, and then syntactic and semantic aspects were extracted for assessment. Next, we compared several machine learning techniques, from simple models to sophisticated deep neural networks. Our research shows that conventional anti-spam services and shallow models are susceptible to evasion tactics, which leads to subpar performance. The LSTM model, on the other hand, performed better, classifying SMS messages with 98% accuracy. Even with this high accuracy, some evasion techniques continue to pose problems for identification, indicating areas that require more study. This study offers important insights into the efficacy of deep learning models in preventing SMS spam and promotes on-going improvement in reliable SMS spam filtering systems.

## I. INTRODUCTION

Even after nearly 20 years of research, SMS spam1 detection remains a difficult and significant problem for our contemporary digital societies. With a projected USD $330 million (more than double the 2021 amount) wasted to SMS scammers in the US in 2022, SMS spam has increased alarmingly in recent years[2]. In a similar vein, the Scam Watch division of the Australian Competition and Consumer Commission (ACCC) [3] revealed that yearly losses nearly doubled from 2020 (AUD 175 million) to 2021 (AUD 323 million). Reports of SMS fraud increased from 32,337 in 2020 to 67,180 in 2021. In February 2022 alone, over 8,835 SMS scams were reported, the most of any scam delivery method. In this study, we pinpoint four primary obstacles to preventing SMS spam: Data Availability: The lack of extensive, real-world, annotated datasets is a significant obstacle when developing SMS spam detection programs. Previous research [18] frequently uses old and unbalanced datasets that only contain a few hundred spam messages. To the best of our knowledge, the two most recent datasets on SMS spam are the Spam Hunter Dataset [4] and SMS Spam Collection [18]. Only 747 spam messages are included in the out dated SMS Spam Collection (released in 2012), whereas the Spam Hunter Dataset only has 947 annotated spam messages. Their usefulness for combating SMS spam is called into question due to the lack of recent and limited collection of SMS spam messages. Their usefulness for combating SMS spam is called into question due to the lack of recent and limited collection of SMS spam messages. This constraint raises the possibility of over fitting and restricts the model's capacity to generalise and function effectively on unobserved data [15]. Absence of Dataset Benchmarks: Numerous techniques have been put forth for the identification of SMS spam [5, 6, 9, 10, 11, 12, 13]. Nevertheless, the lack of a standardised benchmark dataset for thorough comparisons [11], [10] has caused research in this field to become fragmented.

## OBJECTIVE

The threat of SMS spam has been addressed by combining evasive strategies with machine learning (ML) and deep learning (DL) approaches. Almeida et al. examined a number of ML classifiers and discovered that NB performed well; however, they did not assess DL models and only used word frequency as a feature [18]. In a similar vein, Gupta et al. only assessed conventional ML and one DL classifier while comparing eight ML classifiers, including NB, and utilising just TF-IDF features [1][9]. For SMS spam identification utilising different model stack topologies, Roy et al. [7] tested LSTM with conventional two-class classifiers and discovered that LSTM outperformed them. However, no cutting-edge transformer-based models were assessed, and they solely employed conventional two-class algorithms. Jain and associates. Utilising LSTM with different word embedding methods, [22] only assessed LSTM using Word2Vec; neither contextualised word embedding nor state-of-the-art transformer-based models were employed. On the one hand, prior research in the field of SMS spam identification has been fragmented due to the use of limited features and models. However, classifiers that use PU and one-class learning methods have not gotten as much attention. To the best of our knowledge, no previous study has assessed such a wide range of machine learning models, which makes our research stand out as unique. Using extensive syntactic and semantic aspects, we evaluate their effectiveness and resilience in thwarting SMS spam.

## II. PROBLEM STATEMENT

The widespread persistence and evolution of SMS spam present significant challenges to communication security, as spammers continuously employ sophisticated evasion techniques to bypass traditional detection systems. Existing spam filters, particularly those relying on shallow machine learning models or rule-based methods, often fail to effectively detect and classify these increasingly complex spam messages. Therefore, there is a critical need for an advanced and adaptive SMS spam filtering system that can accurately identify and mitigate spam, even in the presence of evolving attack strategies. This study aims to address this issue by leveraging deep learning techniques, specifically Long Short-Term Memory (LSTM) networks, to build a robust and high-performance spam detection model using a newly introduced large-scale SMS dataset.

## III. EXISTING SYSTEM

To combat the problem of SMS spam, a number of traditional machine learning (ML) and deep learning (DL) approaches have been put forth. Almeida et al. tested a number of ML classifiers and discovered that SVM performed well; however, they did not assess DL models and only used word frequency as a feature [18], [17]. In a similar vein, Gupta et al. only assessed conventional ML and one DL classifier while comparing eight ML classifiers, including SVM and CNN, using TF-IDF features [9]. For SMS spam identification utilising different model stack structures, Roy et al. [7] compared CNN with conventional two-class classifiers and discovered that CNN outperformed them. However, no cutting-edge transformer-based models were assessed, and they solely employed conventional two-class algorithms. Jain and associates. [22] utilised a variety of word embedding methods, but only Word2Vec was used for evaluation; neither contextualised word embedding nor cutting-edge transformer-based models were employed.

**Disadvantage of Existing System**

- The combination of SVM with CNN may result in more complex model architecture, necessitating more advanced pre-processing, integration, and coordination between the two algorithms.
- This can lead to longer development times and increased maintenance efforts.
- CNNs, in particular, require large datasets and high-performance hardware, while SVMs need significant time for parameter tuning. This makes the combined model more resource-intensive.
- Training Time: The combination of SVM and CNN may result in increased training times. While CNNs typically require more training time due to the complexity of the model and the need for large datasets, SVMs also require time-consuming parameter tuning.

## IV PROPOSED SYSTEM

Prior algorithms have been effective in detecting spam communications, but by using evasive tactics, attackers can drastically impair their effectiveness [8]. In the area of black-box evasive approaches (the subject of this work) on SMS spam filters, a number of tactics have been found [5], [21]: injecting Ham words, spacing spam words, poisoning, changing labels and replacing synonyms. Combining these three methods—TFIDF for feature extraction, LSTM for sequence learning, and Naive Bayes for classification—allows the system to better detect SMS spam by utilising both conventional machine learning approaches and cutting-edge deep learning techniques. The system can efficiently tackle the evasive methods used by spammers and handle the semantic and syntactic intricacies of SMS messages thanks to the hybrid approach. The suggested method is a reliable solution for SMS spam filtering in the real world since it takes advantage of Naive Bayes' efficiency in handling big datasets, which makes it both accurate and scalable.

**Advantages of Proposed System**

- High Accuracy: The combination of LSTM and Naive Bayes helps achieve high accuracy (98%) in classifying SMS messages as spam or ham.
- Effective Feature Extraction: TFIDF efficiently captures the importance of words in messages, improving the system's ability to distinguish between spam and legitimate messages.
- Resilience to Evasive Techniques: The LSTM network is able to adapt to changes in spam strategies like misspellings, word substitutions, and unusual sentence structures, making the system more robust.
- Scalable and Flexible: The system can easily be trained with large datasets and updated to detect new types of spam, ensuring long-term effectiveness.
- Fast and Efficient: Naive Bayes is a simple and fast classifier that works well with the features extracted by TFIDF and LSTM, ensuring quick classification of SMS messages.

## V. RELATED WORKS

To facilitate SMS spam detection research, a number of SMS datasets were made public. These datasets are constrained, though, and only include a small number of spam messages. Table 1 summarises the most noteworthy datasets on SMS spam, ranging from the oldest from 2012 to the most recent from 2022. The SMS Spam Collection [12][18], which was published in 2012, is incredibly unbalanced and out dated—the earliest spam messages in the collection date back before 2010. Out of 5,574 messages in total, only 747 are spam. The defunct Grumble text website4 (a UK forum) and SMS Spam Corpus v.0.1 Big [20] were the sources of the spam messages. There are 67,093 SMS messages in the National University of Singapore (NUS) SMS Corpus [16] (last updated on March 9, 2015). In order to gather and extract SMS spam data from publicly shared SMS images on Twitter, the "Spam Hunter" framework was recently presented [4]. Over a five-year period (2018–2022), 25,889 Twitter messages in various languages were collected and published using Spam Hunter. The SMS spam dataset produced by the Spam Hunter framework has shugenoise, notwithstanding the novelty of the Spam Hunter technique (a considerable number of benign and awareness messages were wrongly crawled and included). The dataset also includes a large number of duplicate messages and OCR errors. If given to the ML model, this noise might seriously mist rain it, necessitating a manual inspection to eliminate errors and noise.

## VI. METHODOLOGY OF APPLICATION

**Module Description:**

**Collect Data:**

The first step is to gather relevant textual data from reliable sources. This data should be representative of the task, such as classification, sentiment analysis, or language modelling. High-quality, labelled data is essential for supervised learning. Data can be collected from APIs, web scraping, or open datasets. The size and diversity of the dataset directly impact model performance. Proper data handling, storage, and permission checks are crucial.

**Augment Data:**

Data augmentation involves creating new training samples from existing data to increase dataset size and diversity. Techniques include synonym replacement, random insertion, and back-translation. This helps the model generalize better and reduces over fitting. Augmentation is especially useful when labelled data is limited. Care must be taken to preserve the original context and meaning. It boosts model robustness across varied inputs.

**Pre-process Text:**

Pre-processing cleans and prepares raw text for analysis. It typically involves removing punctuation, converting text to lowercase, and removing stop words. Lemmatization or stemming may also be applied to reduce words to their base form. This step reduces noise and standardizes the input for better model understanding. Proper pre-processing improves the quality of tokenization and feature extraction. It ensures consistent and structured input for the model.

**Tokenize Text:**

Tokenization splits text into smaller units, usually words or sub words, called tokens. These tokens are then converted into numerical representations for input into neural networks. Common tokenization methods include word-level, character-level, or using libraries like WordPiece or Byte Pair Encoding (BPE). Tokenization helps the model understand semantic structure. Padding and truncation are applied to ensure uniform input length. This step bridges the gap between raw text and numerical modelling.

**Create LSTM Model:**

Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) capable of learning long-term dependencies in sequences. An LSTM model is well-suited for text tasks like sentiment analysis or language generation. It uses memory cells, input, forget, and output gates to control data flow. Layers are stacked with embedding, LSTM, and dense output for classification or regression. Model architecture is tailored based on the complexity of the task. Proper regularization (like dropout) helps prevent over fitting.

**Train Model:**

Training involves feeding the tokenized and pre-processed data into the LSTM model and adjusting weights to minimize loss. The dataset is divided into training and validation sets for performance monitoring. Techniques like early stopping, learning rate scheduling, and dropout may be used. During training, the model learns patterns and relationships in the data. Accuracy and loss metrics are tracked to evaluate progress. Training duration depends on data size and model complexity.

**DeployModel:**

Deployment makes the trained model available for real-world use. This could involve integrating the model into a web app, mobile app, or API service. Deployment ensures that users can input new data and receive predictions in real-time. Tools like Flask, Django, or FastAPI are often used in Python-based deployments. Cloud platforms like AWS, Azure, or Heroku support scalable deployment. Proper testing and monitoring are necessary to ensure reliability.’

**Make Prediction:**

Once deployed, the model can take in new, unseen text, pre-process and tokenize it, and output predictions. For classification, this might mean predicting a category or label (e.g., positive/negative, disease/no disease). The model uses the patterns it learned during training to infer results. Predictions must be interpreted with confidence scores or probabilities. Feedback loops can help improve future model versions. This step delivers the final output that supports decision-making.

## VII. ALGORITHM USED IN APPLICATION

To determine whether an SMS message is spam or real (ham), the suggested method, TFIDF-LSTM with Naive Bayes, uses a number of crucial processes. Prior to being tokenised into individual words, the SMS messages are pre-processed by eliminating stop words and extraneous letters. TFIDF (Term Frequency-Inverse Document Frequency) is used to transform these tokenised words into numerical vectors, which aid in determining the significance of each word in the message according to its frequency. These vectors are then processed using the LSTM (Long Short-Term Memory) model. The system can identify trends in communications, even if they contain small modifications or evasive tactics used by spammers, thanks to LSTM, a form of neural network that excels at comprehending the context and word sequence. Following the LSTM's processing of the message, a Naive Bayes classifier receives the features and determines whether the message is spam or ham. The message is subsequently assigned to the class with the highest probability by the classifier.

## VIII. DATA FLOW DIAGRAM:

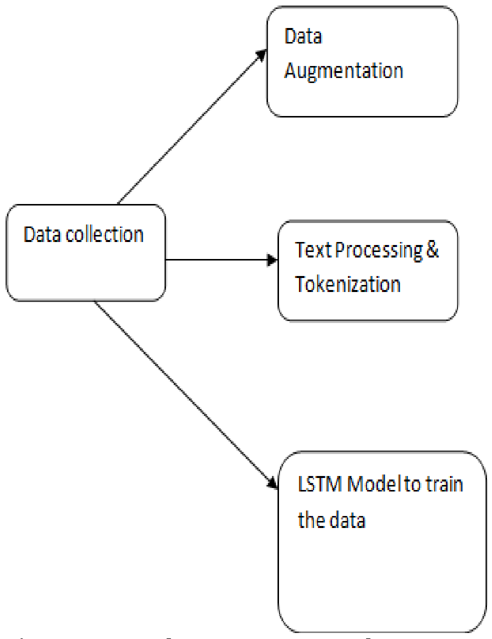


**Fig 8.1:** Data Flow Diagram Level 0

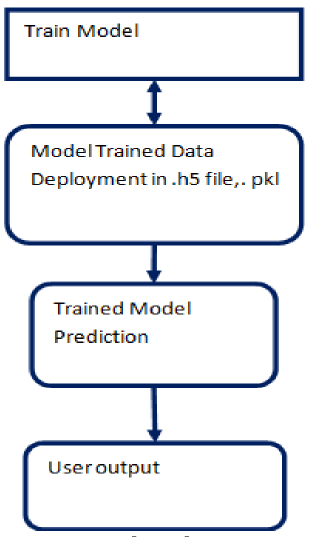


**Fig 8.2**: Data Flow diagram Level 1

**Explanation:** A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. Often, they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

## IX. SYSTEM ARCHITECTURE

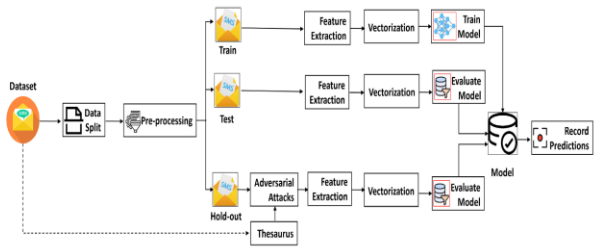


**Fig. 9.1:** System Architecture

**Explanation:** The system begins with dataset collection, followed by splitting the data into training and testing sets. Pre-processing is applied to clean the text. Feature extraction is performed on the training data and passed through a machine learning model for training. The test data is then vectorized and evaluated against predicted outputs.
Finally, to counter adversarial attacks, a hold-out set is processed through a thesaurus to generate altered inputs. These are also feature-extracted, vectorized, and evaluated to verify the model's robustness.

## X. RESULTS:

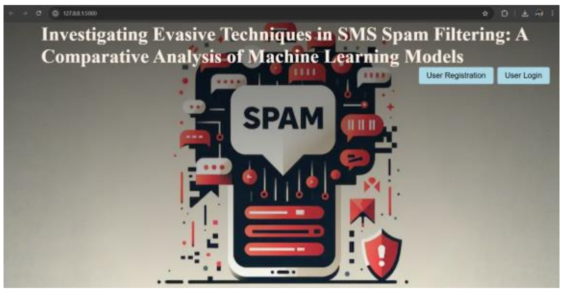

Investigating Evasive Techniques in SMS Spam Filtering: A Comparative Analysis of Machine Learning Models

**Fig 10.1:** Redirected Main Page

**Description:** This is the default page that shows up when the application is opened.

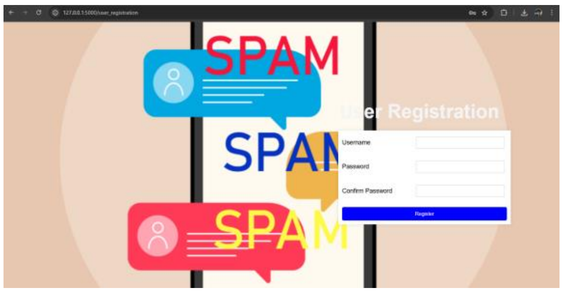


**Fig 10.2:** User registration Page

**Description:** This page allows the user to register. It initiates user onboarding into the spam detection application.

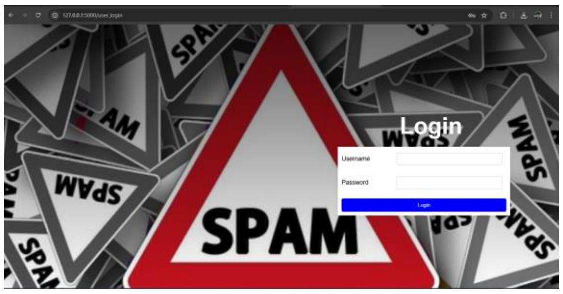


**Fig 10.3:** User Login Page

**Description:** The login page prompts returning users to enter valid credentials to gain access to the spam detection system. This ensures secure access control.

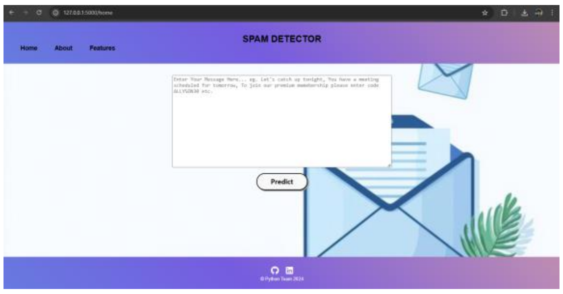
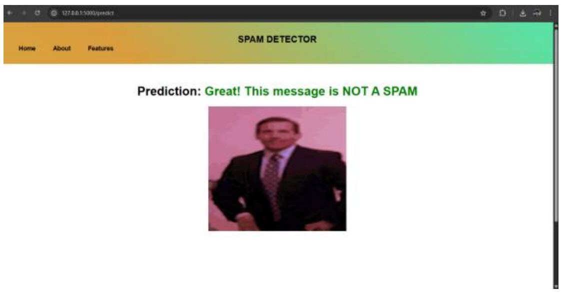


**Fig 10.4:** Application Home Page

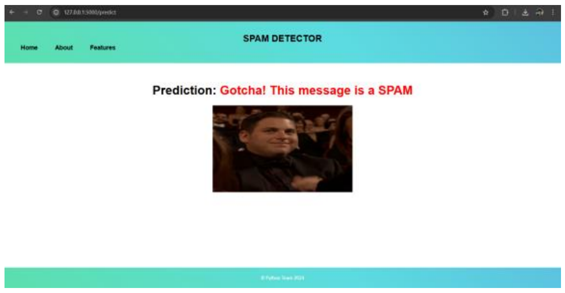
**Description:** This is the main interface of the spam detector tool. Users can input SMS text here for real-time spam analysis using the implemented LSTM-Naive Bayes hybrid model.



Prediction: Great! This message is NOT A SPAM

**Fig 10.7:** Predication when the entered text is not a spam

**Description:** When a legitimate (ham) message is entered, the system predicts the text as not spam, confirming the input is safe and does not exhibit spam characteristics.



Prediction: Gotcha! This message is a SPAM

**Fig 10.8:** Prediction when the entered text is suspected of spam

**Description:** Displays the result when a suspicious or spam-like message is entered. The system flags it accordingly, showing that the message is identified as spam by the model.

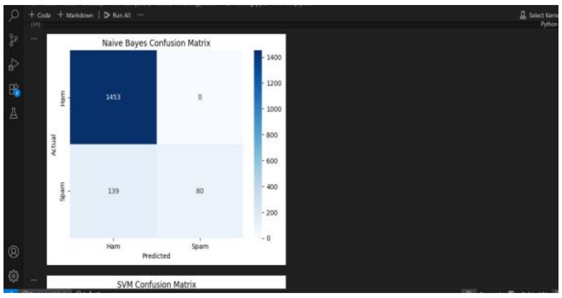


**Fig 10.9:** Naive Bayes Confusion Matrix

**Description:** This matrix visualizes the classification performance of the Naive Bayes model, showing the count of true positives, false positives, true negatives, and false negatives.

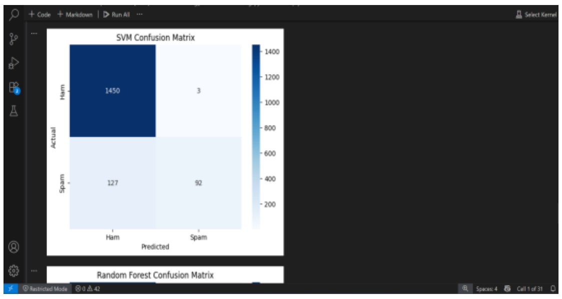


**Fig 10.10:** SVM Confusion Matrix

**Description:** Illustrates the results of the Support Vector Machine model's predictions. It compares predicted labels against actual labels to evaluate accuracy and misclassification rates.

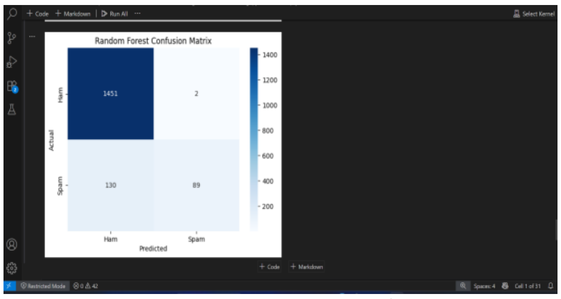
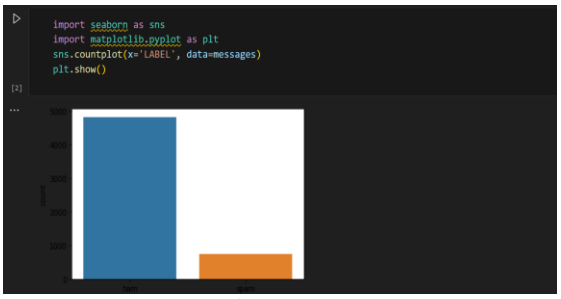


**Fig 10.11:** Random Forrest Confusion Matrix

**Description:** This confusion matrix shows the predictive accuracy of the Random Forest model, indicating how well it distinguishes between spam and non-spam messages.



```
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(x='LABEL', data=messages)
plt.show()
```

**Fig 10.12:** Count of Not Spam (Blue) and Spam (Orange)

**Description:** A bar graph representing the distribution of classified messages. It visually compares the volume of spam versus non-spam (ham) messages detected in the dataset.

## XI. CONCLUSION AND FUTURE ENHANCEMENTS:

While the proposed LSTM-based SMS spam filtering system has demonstrated impressive performance, there remains considerable scope for further improvement and expansion. Future work may explore the integration of hybrid deep learning models, such as combining LSTM with Convolutional Neural Networks (CNN) or Transformer-based architectures like BERT, to capture deeper contextual and semantic nuances. Incorporating real-time adaptive learning mechanisms can help the system stay updated with emerging spam tactics by continually learning from new data. Additionally, expanding the dataset to include multilingual SMS messages and diverse regional slang could enhance the model's global applicability. Another promising direction is the use of explainable AI techniques to improve model transparency, enabling users and developers to understand prediction rationales. Finally, deploying the system in real-world messaging platforms and evaluating its effectiveness in live environments would provide practical insights and guide further refinements to strengthen SMS spam defence strategies.

In Conclusion, we presented and described a sizable new SMS dataset in order to illustrate the evolving features of SMS spam. We benchmarked the effectiveness and resilience of various machine-learning-based models and the anti-spam ecosystem for SMS spam detection using the dataset. The outcomes demonstrated that every machine learning-based model effectively recognised authentic SMS messages (ham SMS). However, only a few number of anti-spam text apps and deep learning models were able to categorise spam messages with a precision score over 90% and 80%, while the remainder fell short of this standard. The limitations of recent anti-spam advancements and possible research avenues are highlighted by our examination of the machine learning model and SMS anti-spam ecosystem. We contend that SMS spam still presents a serious problem, requiring more study to create systems that can successfully counteract the evasion tactics used by spammers and protect the public from SMS spam.

## XII. REFERENCES:

[1] M. Salman, M. Ikram, and M. A. Kaafar, “Investigating Evasive Techniques in SMS Spam Filtering: A Comparative Analysis of Machine Learning Models,” *IEEE Access*, vol. 12, pp. 24306–24322, Feb. 2024, doi:10.1109/ACCESS.2024.3364671.

[2] FCC. (2022). *The Top Text Scams of 2022*. Accessed: Oct. 8, 2023. [Online]. Available: https://www.ftc.gov/news-events/data-visualizations/data-spotlight/2023/06/iykyk-top-text-scams-2022

[3] ACCS. (2022). *Accs Scam Statistics*. [Online]. Available: https://www.scamwatch.gov.au/scam-statistics

[4] S. Tang, X. Mi, Y. Li, X. Wang, and K. Chen, ‘‘Clues in tweets: Twitter-guided discovery and analysis of SMS spam,’’ in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2022, pp. 2751–2764.

[5] S. Rojas-Galeano, ‘‘Using BERT encoding to tackle the mad-lib attack in SMS spam detection,’’ 2021, arXiv:2107.06400.

[6] T. Xia and X. Chen, ‘‘A discrete hidden Markov model for SMS spam detection,’’ *Appl. Sci.*, vol. 10, no. 14, p. 5011, Jul. 2020.

[7] P. K. Roy, J. P. Singh, and S. Banerjee, ‘‘Deep learning to filter SMS spam,’’ *Future Gener. Comput. Syst.*, vol. 102, pp. 524–533, Jan. 2020.

[8] N. H. Imam and V. G. Vassilakis, ‘‘A survey of attacks against Twitter spam detectors in an adversarial environment,’’ *Robotics*, vol. 8, no. 3, p. 50, Jul. 2019.

[9] M. Gupta, A. Bakliwal, S. Agarwal, and P. Mehndiratta, ‘‘A comparative study of spam SMS

detection using machine learning classifiers,'' in *Proc. 11th Int. Conf. Contemp. Comput. (IC3)*, Aug. 2018, pp. 1–7.
[10] J. Li, S. Ji, T. Du, B. Li, and T. Wang, ''TextBugger: Generating adversarial text against real-world applications,'' 2018, arXiv:1812.05271.
[11] S. M. Abdulhamid, M. S. A. Latiff, H. Chiroma, O. Osho, G. Abdul-Salaam, A. I. Abubakar, and T. Herawan, ''A review on mobile SMS spam filtering techniques,'' *IEEE Access*, vol. 5, pp. 15650–15666, 2017.
[12] A. A. Al-Hasan and E.-S.-M. El-Alfy, ''Dendritic cell algorithm for mobile phone spam filtering,'' *Proc. Comput. Sci.*, vol. 52, pp. 244–251, Jan. 2015.
[13] A. Narayan and P. Saxena, ''The curse of 140 characters: Evaluating the efficacy of SMS spam detection on Android,'' in *Proc. 3rd ACM Workshop Secur. Privacy Smartphones Mobile Devices*, Nov. 2013, pp. 33–42.
[14] T. Almeida, J. M. Hidalgo, and T. Silva, ''Towards SMS spam filtering: Results under a new dataset,'' *JiSS*, vol. 2, no. 1, pp. 1–18, 2013.
[15] A. van der Schaaf, C.-J. Xu, P. van Luijk, A. A. van't Veld, J. A. Langendijk, and C. Schilstra, ''Multivariate modeling of complications with data driven variable selection: Guarding against overfitting and effects of data set size,'' *Radiotherapy Oncol.*, vol. 105, no. 1, pp. 115–121, Oct. 2012.
[16] T. Chen and M.-Y. Kan, ''Creating a live, public short message service corpus: The NUS SMS corpus,'' *Lang. Resour. Eval.*, vol. 47, pp. 299–335, Aug. 2012.
[17] K. Mathew and B. Issac, ''Intelligent spam classification for mobile text message,'' in *Proc. Int. Conf. Comput. Sci. Netw. Technol.*, vol. 1, Dec. 2011, pp. 101–105.
[18] T. A. Almeida, J. M. G. Hidalgo, and A. Yamakami, ''Contributions to the study of SMS spam filtering: New collection and results,'' in *Proc. 11th ACM Symp. Document Eng.*, Sep. 2011, pp. 259–262.
[19] L. Duan, N. Li, and L. Huang, ''A new spam short message classification,'' in *Proc. 1st Int. Workshop Educ. Technol. Comput. Sci.*, 2009, pp. 168–171.
[20] J. M. Gómez Hidalgo, G. C. Bringas, E. P. Sánz, and F. C. García, ''Content based SMS spam filtering,'' in *Proc. ACM Symp. Document Eng.*, Oct. 2006, pp. 107–114.
[21] S. Webb, S. Chitti, and C. Pu, ''An experimental evaluation of spam filter performance and robustness against attack,'' in *Proc. Int. Conf. Collaborative Computing, Netw., Appl. Worksharing*, 2005, p. 8.
[22] G. Jain, M. Sharma, and B. Agarwal, ''Optimizing semantic LSTM for spam detection,'' *Int. J. Inf. Technol.*, vol. 11, no. 2, pp. 239–250.