Journal of Theoretical and Applied Information Technology
31st July 2025. Vol.103. No.14

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

MESSAGE CONVERSATION BASED SOCIAL ENGINEERING ATTACK DETECTION USING MACHINE LEARNING

SEAH NI MIN[1], NOR FAZLIDA MOHD SANI[2]
[1,2]Department of Computer Science, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang 43400, Selangor Malaysia
E-mail: [1]gs67602@student.upm.edu.my, [2]fazlida@upm.edu.my

ABSTRACT

Social engineering attacks present a major threat in today's interconnected world, exploiting the intricacies of human communication to deceive individuals and extract sensitive information. With the increasing reliance on messaging platforms, communication has become highly informal, often involving colloquial language, abbreviations, and dynamically evolving linguistic styles. These characteristics obscure user intent and make it difficult to identify malicious or deceptive behaviour. Detecting such threats requires a deep understanding of conversational context, which is often lacking in current approaches thereby leaving people vulnerable to subtle social engineering attacks embedded within everyday messages. This study addresses this gap by fine-tuning DistilBERT, a state-of-the-art natural language processing (NLP) model, to detect social engineering attacks in message conversations. Leveraging its ability to understand contextual semantics while maintaining computational efficiency, the model was trained and evaluated using the SMS Spam Collection dataset. The proposed approach achieved a high detection accuracy of 99.46%, outperforming previous models such as SOCIALBERT. While the results demonstrate strong classification performance, limitations include the use of a single text-based dataset and the exclusion of multimodal content such as images and links. Future work should explore more diverse and multilingual datasets, incorporate multimodal detection, and optimise the model further for deployment in resource-constrained environments.

Keywords: *Messages, Social Engineering Attack, NLP, Machine Learning, DistilBERT.*

1. INTRODUCTION

In today's digitally connected world, cybersecurity threats are no longer limited to technical vulnerabilities but increasingly exploit human behaviour. Among these, social engineering attacks are particularly dangerous, leveraging psychological manipulation rather than technical exploits to deceive individuals into compromising security. These attacks are prevalent across critical sectors such as financial services, healthcare, and government, where disclosing sensitive information can lead to devastating consequences [1][2]. Despite advancements in cybersecurity technologies, humans remain the weakest link. According to the 2024 Verizon Data Breach Investigations Report, 68% of security breaches involved human factors, such as being deceived by social engineering attacks or unintentional errors. This alarming statistic highlights the urgent need to strengthen defences not just at the system level, but also in detecting socially engineered manipulation at the communication level.

The rise of digital messaging platforms has further complicated the landscape. While these tools have made communication faster and more convenient [3], they also introduce new vulnerabilities [4]. Modern digital communications often contain informal language, abbreviations, and ambiguous phrasing, which attackers exploit to mask their true intent [5]. Unlike traditional phishing emails or messages that may explicitly request credentials, modern social engineering messages use subtle tactics such as persuasion, authority, or urgency to manipulate users without directly asking for sensitive data [6]. However, existing detection methods often do not incorporate a deep contextual understanding of language, focusing instead on surface-level features such as keywords or syntactic patterns. This highlights a research gap in developing models that can capture the nuanced meanings and implicit cues commonly found in social engineering messages.

Therefore, this study proposes a machine learning-based approach using DistilBERT, a lighter and faster variant of the Bidirectional Encoder

Representations from Transformers (BERT) language model to enhance the detection accuracy of social engineering attacks in message conversations. DistilBERT retains 97% of BERT's language understanding capability while being 40% smaller and 60% faster during inference, making it suitable for real-time or resource-constrained environments [7]. This efficiency is achieved using only six transformer layers compared to the 12 layers in the BERT base model [7]. BERT-based models are known for their bidirectional understanding of language, allowing them to capture subtle cues and deeper semantic context [8][9]. The model will be trained and evaluated using the SMS Spam Collection dataset, which includes a range of malicious and legitimate messages. To assess performance, standard metrics such as accuracy, precision, recall, and F1-score will be used. This study is significant as it explores the intersection of natural language processing and cybersecurity, aiming to enhance the detection of social engineering attacks in digital messages. By leveraging state-of-the-art NLP models, the study seeks to contribute to more robust and intelligent security mechanisms.

The following section presents the research background, providing context for the study. This will be followed by the methodology section, which focuses on data preprocessing, model training, and evaluation. The next section covers the results and discussion, analysing the model's performance and key findings. Finally, the concluding section summarises the project, highlighting its limitations and potential directions for future work.

2. RESEARCH BACKGROUND

A social engineering attack is a cyberattack strategy that leverages human vulnerabilities to undermine the security of various cyberspace elements, including infrastructure, data, resources, users, and operations [10]. In contrast to traditional cyberattacks, which target technical flaws, social engineering attacks deceive individuals into disclosing sensitive information or circumventing security measures. Attackers frequently use psychological tactics to induce a sense of urgency or fear, prompting victims to act quickly without critically assessing the situation. According to a survey conducted by [3], social engineering is recognised as one of the most significant cybersecurity threats, as it can bypass even the most robust security measures, such as firewalls and intrusion detection systems. In the early efforts to counter social engineering, researchers and practitioners focused primarily on user education and security policies [11]. Training programs aimed to raise awareness of social engineering techniques and teach users how to recognise and respond to suspicious activities. While these efforts helped reduce user susceptibility to known tactics, they were not sufficient to keep pace with the increasing sophistication and evolving nature of attacks.

Subsequent detection systems relied on rule-based and signature-based approaches, which flagged messages containing specific keywords, phrases, or structural patterns often associated with smishing attempts [12]. For instance, [13] proposed an approach that uses a pre-defined Topic Blacklist (TBL) to verify the discussion topics in text lines generated by potential attackers. Similarly, [14] proposed integrating topic blacklists with NLP to detect question-command patterns that suggest malicious intent. Their study focused on identifying question-command patterns in text conversations using NLP, extracting topics from these question-commands, and comparing them against pre-defined TBL [14]. Although effective in identifying well-defined attack patterns, these static methods lacked the flexibility to detect novel and subtle tactics that evade pre-defined rules [3].

To overcome these limitations, some researchers have conducted numerous studies on detecting social engineering attacks using NLP, machine learning, and deep learning algorithms. The approach for social engineering attack detection, as presented by [15], incorporates case-based reasoning (CBR) systems for malicious URL detection and convolutional neural networks (CNNs) to determine if a conversation suggests a social engineering attack. Besides, the SEADer model used NLP to detect grammatical inconsistencies and classify conversations using artificial neural networks (ANN) [16], while SEADer++ v2 further enhances this process by adding three additional columns to the classification dataset, creating a total of seven columns, to improve output quality. This model leverages NLP for text processing and classification with Random Forest (RF), Multi-Layer Perceptron (MLP), and k-nearest neighbours (KNN) algorithms. Consequently, the proposed method in their study achieved slightly better accuracy (80.1%) and average curve results (89.2%) compared to SEADer++ [17]. Furthermore, a detection model that relies solely on text input for identifying social engineering -

Journal of Theoretical and Applied Information Technology
31st July 2025. Vol.103. No.14

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

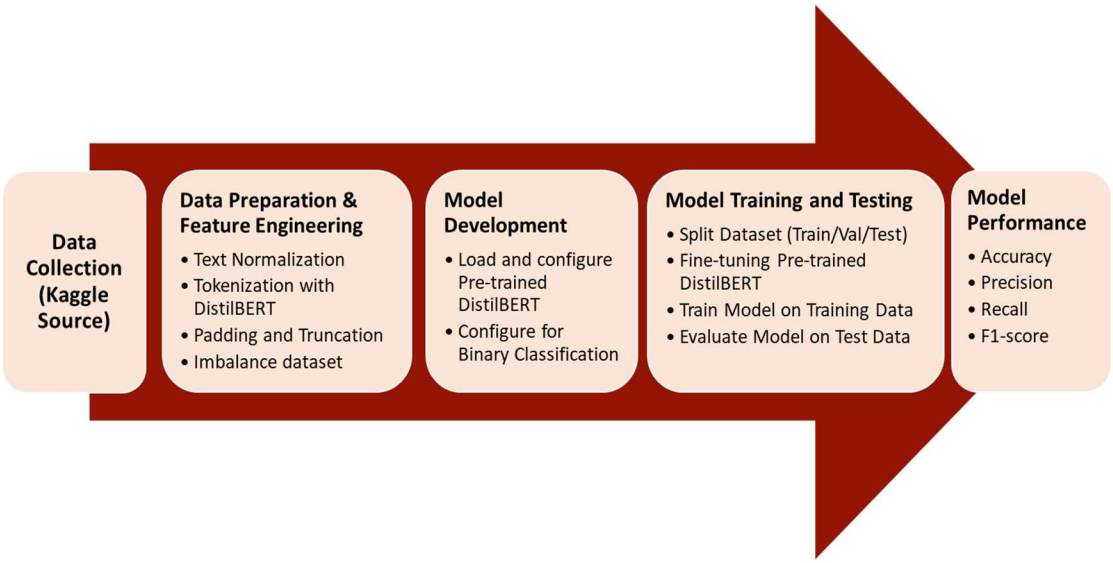

Figure 1: An overview of research design

attacks has achieved over 80% classification accuracy, with classifiers like Neural Network, RF, and Support Vector Machine (SVM), where SVM demonstrated the overall best performance [18]. In social engineering attack detection using an attention-based Bidirectional Long Short-Term Memory (Bi-LSTM) and CNN, a dataset consisting of user information and chat dialogue is pre-processed before input into the module for final analysis. The attention-based Bi-LSTM is used to capture contextual semantics from the dialogue text, while the CNN integrates user characteristics and content features for classification and judgement [6]. This model focuses on detecting social engineering attacks in conversational form by analysing both chat history and user information. Additionally, the SOCIALBERT model, built on DistilBERT, was fine-tuned for a downstream task to detect various social engineering tactics in text messages, achieving a high accuracy of 97.55% [19].

Although recent studies have incorporated advanced NLP, machine learning, and deep learning models into social engineering attack detection, a significant gap remains underexplored in terms of contextual understanding. While [6] employed an attention-based Bi-LSTM combined with CNN to capture contextual semantics from chat dialogue, the study did not report standard performance metrics such as accuracy, precision, recall, or F1-score, limiting the assessment of its practical effectiveness. In another study, the SOCIALBERT model [19], which is based on the DistilBERT algorithm known for its contextual understanding capabilities. However, the study primarily focused on categorising the different types of social engineering tactics presented in messages and evaluated their performances.

3. METHODOLOGY

The methodology employed to detect social engineering attacks using machine learning, with a specific focus on the fine-tuning of the pre-trained DistilBERT model. It also includes the description of the dataset, preprocessing steps, model development, training, and evaluation techniques. The goal is to create a robust classification model that differentiates the text messages between social engineering attack and non-attack, thereby enhancing the detection performance. Figure 1 provides an overview of the research design process.

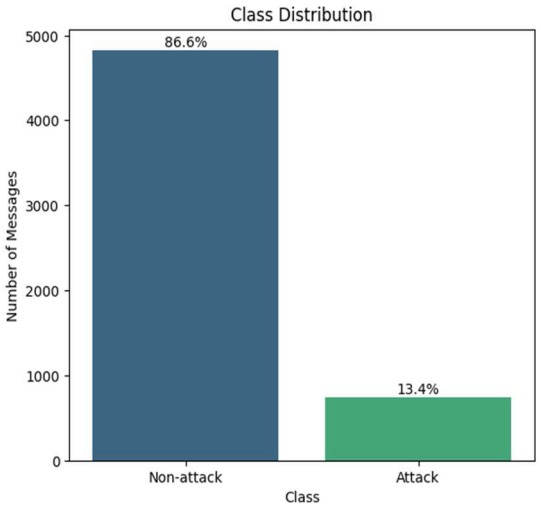

Figure 2: The class distribution of dataset

3.1 Dataset

This study utilises a publicly available, real-world dataset: the SMS Spam Collection Dataset, which was downloaded from Kaggle. An overview of the dataset reveals a total of 5572 rows, all in English and primarily derived from message conversations. These messages represent scenarios that may involve either an attempted social engineering attack or a benign (non-attack) interaction. Among these, 86.6% (4825 rows) are labelled as benign or legitimate, while 13.4% (747 rows) are labelled as malicious, as illustrated in Figure 2. Furthermore, the dataset is organized into five columns: v1, v2, and three unnamed columns. The v1 column serves as a class label, indicating whether the message is legitimate or not, while the v2 column contains a string representing a message or a segment of a conversation. This string serves as the raw input to the model.

3.2 Data Preprocessing

Data preprocessing is essential for preparing the dataset for effective machine learning training and evaluation. It cleans and standardises raw textual data, addressing inconsistencies, noise, and missing elements that can impact model performance. In this study, three unnamed columns with over 99% missing data were manually removed, and the column names v1 and v2 were renamed to label and text, respectively, for clarity. The classification labels were also updated to "attack" and "non-attack" to align with the project's scope. Subsequent preprocessing steps included text normalisation, tokenisation with DistilBERT, applying padding and truncation to ensure uniform input sizes for the DistilBERT model, and handling the imbalanced dataset.

3.2.1 Text normalization

To ensure uniformity and minimises variations caused by inconsistent formatting, all text input is converted to lowercase using the distilbert-base-uncased parameter from the Hugging Face Transformers library during tokenisation, as DistilBERT is a case-sensitive model [20]. The term “uncased” indicates that the model is designed to handle text without sensitivity to letter casing. In other words, it treats uppercase and lowercase letters equivalently, meaning that “Attack” and “attack” are considered identical during both tokenisation and processing.

3.2.2 Tokenisation

Tokenisation involves converting raw text into a structured format that a machine learning model can process. It employs WordPiece Tokenisation to break down text into smaller components, sub words or morphemes called tokens. DistilBERT uses a pre-trained tokeniser that maps tokens to numerical identifiers, known as token IDs, based on the model's pre-defined vocabulary. Meanwhile, Special tokens are added to structure the input sequence for the DistilBERT model. The [CLS] token is inserted at the beginning to represent the entire sequence and serve as a classification token, while the [SEP] token is appended at the end to indicate sequence boundaries.

3.2.3 Padding and truncation

Padding involves appending special padding tokens [PAD] to the end of a sequence to ensure it matches the required maximum sequence length when text sequences are shorter than the model's expected input length. After adding, an attention mask is generated to help the model distinguish between meaningful tokens and padding tokens. This mask assigns a value of 1 to actual tokens and 0 to padding tokens which effectively instructs the model to ignore the padding during computations. Moreover, truncation is applied to handle sequences that exceed the maximum length permitted by the model. It involves discarding tokens beyond the maximum allowed length by setting the true parameter to the truncation during tokenisation.

3.2.4 Imbalance Dataset

This imbalance makes it difficult for models to effectively learn and classify minority classes, often resulting in biased predictions and poor performance on malicious messages. To address this issue, sample based class weight (SBCW) is used to adjust the contribution of each class to the loss function during training. Class weights are calculated based on the frequency of each class in the dataset. According to [21], the formula is:

$$w_c = \frac{n}{k \cdot n_c}$$

Where:

w_c: Weight for class c
n: Total number of samples in the dataset
k: Number of unique classes
n_c: Number of samples in class c

The computed weights for the two classes are 0.5775 for the non-attack class and 3.7266 for the attack

class. These weights are then converted into a PyTorch tensor and integrated into the loss function to enhance learning for the minority class.

3.3 Model Development

The development of the social engineering attack detection model involves leveraging DistilBERT, a transformer-based NLP model known for its efficiency and contextual understanding capabilities. The process begins by loading a pre-trained DistilBERT model, specifically DistilBERTForSequenceClassification, from the Hugging Face Transformers library to handle the classification task. The model is configured for binary classification by setting the num_labels parameter to 2, ensuring it outputs two logits corresponding to "attack" and "non-attack" messages. Dropout mechanisms are applied to enhance generalisation and prevent overfitting, thereby improving the model's accuracy on unseen data. Specifically, attention_probs_dropout_prob = 0.5 introduces dropout in the self-attention mechanism, randomly zeroing out 50% of attention scores during training to reduce over-reliance on specific words or phrases. Similarly, hidden_dropout_prob = 0.5 applies dropout in the fully connected layers, deactivating 50% of neurons to encourage learning of broader patterns rather than memorising the training data.

3.4 Model Training and Testing

The dataset is initially divided into training and test sets using an 80:20 ratio. Additionally, the training set is further split into training and validation sets using another 80:20 ratio, ensuring the validation set is available to monitor the model's performance during training. This validation set is essential for hyperparameter tuning, allowing the selection of the best parameters for the final training and evaluation process. A random seed is applied to ensure reproducibility by generating the same sequence of random numbers across different runs, resulting in a consistent dataset split every time the code is executed.

3.4.1 Hyperparameter Tuning

Hyperparameter tuning is a step in optimising machine learning models, as it determines the configuration of parameters that most effectively enhance model performance. Optuna, an open-source hyperparameter optimisation framework is applied to automate the process of discovering the optimal hyperparameters for machine learning models. The Optuna workflow for hyperparameter tuning consists of four main steps: defining the objective function, specifying the search space, creating a study, and running the optimisation. First, an objective function is defined to encapsulate the model training and evaluation process. This function returns a validation loss that Optuna seeks to minimise. Next, the search space is defined to represent the range and types of hyperparameters to explore, including learning rate, batch size, number of epochs, and weight decay. The learning rate is set to 2e-5, 3e-5, or 5e-5, the batch size is either 16 or 32, the number of epochs ranges from 2 to 4, and the weight decay ranges from 0.01 to 0.05. After defining the objective function and search space, an Optuna study is created to manage the hyperparameter optimisation process. The optimisation process is then initiated, iteratively evaluating the objective function with various hyperparameter combinations and navigating the search space to identify configurations that provide the best hyperparameters.

3.4.2 Final Training

The final training process involves combining the training and validation sets to maximise data utilisation, especially when working with small datasets. By merging these sets, the model can leverage a larger and more diverse dataset, enabling it to learn richer representations and improve performance on unseen data. This approach is particularly beneficial for small datasets, as it maximises the data used for training while reserving the test set for unbiased evaluation. The model is then trained on the aggregated dataset using the best hyperparameters obtained from the tuning process. These optimal hyperparameters—such as learning rate, batch size, number of epochs, and weight decay—are detailed in Table 1. After training, its performance is rigorously evaluated using a test set to determine its ability to generalise to unseen data.

Table 1: The best parameter after hyperparameter tuning

Hyperparameter	Value
Learning rate	5e-05
Batch size	16
Number of epochs	2
Weight decay	0.01892612832422345

3.5 Performance Metrics

The effectiveness of the machine learning model for detecting social engineering attacks is

Journal of Theoretical and Applied Information Technology
31st July 2025. Vol.103. No.14

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

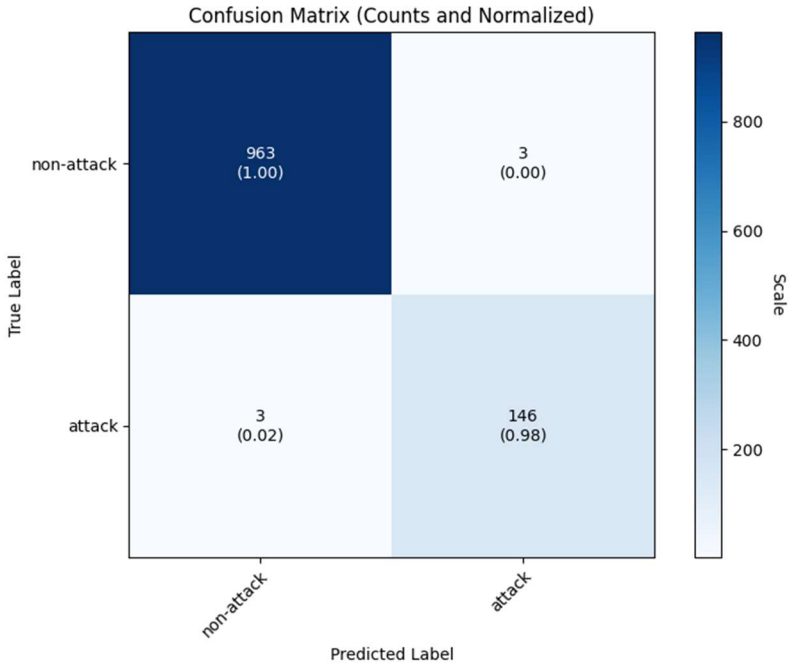

Figure 3: Result of confusion matrix

evaluated using key metrics, including accuracy, precision, recall, and F1-score. These metrics are computed from the confusion matrix, which encapsulates the model's performance in terms of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

Accuracy: It measures the overall correctness of the model by calculating the ratio of correctly classified instances (both positive and negative) to the total instances. It offers a general assessment of the model's performance across all classes, ensuring that the predicted outcomes align closely with the actual labels.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision: Precision assesses the accuracy of the model's positive predictions. A high precision score indicates that the model is likely correct when it predicts an attack. This metric is particularly important in situations where minimising false positives is critical to reducing false alarms.

$$Precision = \frac{TP}{TP + FP}$$

Recall: Recall measures the model's ability to correctly identify all actual positive instances. High recall ensures that the model detects most attacks, which is vital in applications where missing an attack (false negatives) has significant consequences.

$$Recall = \frac{TP}{TP + FN}$$

F1-score: The F1-score combines precision and recall into a single metric by calculating their harmonic mean. It provides a balance between precision and recall, especially useful in situations where the dataset is imbalanced and both metrics are critical for evaluating performance.

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

4. RESULT AND DISCUSSION

Figure 3 depicts the confusion matrix, which provides a detailed breakdown of the model's classification performance by comparing predicted labels with true labels. The results indicate that the model correctly classified 146 instances of malicious messages as "attack" (true positives) and 963 instances of benign messages as "non-attack" (true negatives). Additionally, the model misclassified only 3 benign messages as "attack" (false positives) and 3 malicious messages as "non-attack" (false negatives). These results suggest a strong classification performance with minimal misclassification errors.

Table 2: Result of performance metrics

Metrics	Value (%)
Accuracy	99.46
Precision	99.46
Recall	99.46
F1-Score	99.46

The evaluation results, as shown in Table 2, demonstrate the model's exceptional performance in detecting social engineering attacks, achieving an accuracy of 99.46%. This indicates that the model correctly classified nearly all test samples, highlighting its reliability and strong generalisation to unseen data. Additionally, precision, recall, and F1-score are all equally high at 99.46%, underscoring the model's balanced capability to minimise both false positives and false negatives. These metrics confirm the model's effectiveness in accurately identifying both attack and non-attack messages. Overall, the consistently high performance across all evaluation metrics suggests that the model successfully learns the distinguishing patterns of attack messages.

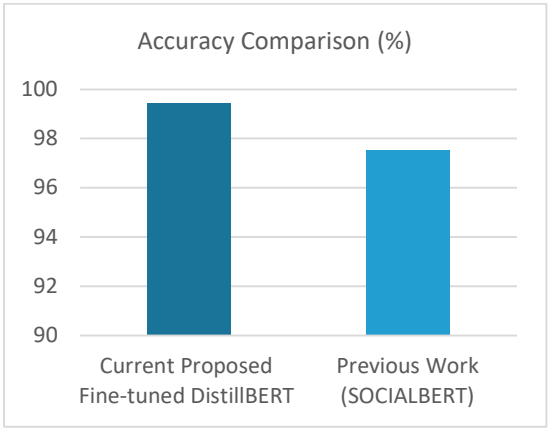

Figure 4: Comparison of accuracy between current and previous work

Figure 4 provides a comparative analysis in terms of accuracy between the proposed model and the SOCIALBERT model previously introduced by [19]. The analysis highlights that the fine-tuned DistilBERT model achieves an accuracy of approximately 99.5%, surpassing the 97.5% accuracy of SOCIALBERT. While the 2% difference may seem minor, it is significant in cybersecurity contexts, where even small improvements can greatly reduce the risk of undetected social engineering attacks. Although the proposed model demonstrates higher accuracy, it is important to note that the SOCIALBERT model also achieves a high level of accuracy. The difference in results primarily stems from the models' distinct objectives and configurations. SOCIALBERT was designed for multi-class classification to distinguish between multiple types of social engineering tactics, such as pretexting, baiting, urgency, consensus, authority, and familiarity. This broader classification task increases complexity, as it involves distinguishing subtle features across multiple categories.

In contrast, the present study focused on binary classification (attack vs. non-attack), allowing for more focused optimisation. The proposed model incorporates strategies such as applying class weights to handle class imbalance, ensuring that the minority class receives adequate emphasis during training. This tailored approach enhances the model's ability to accurately detect attacks and contributes to its improved performance, with the variance in results underscoring the importance of aligning model architecture and training strategies with specific detection objectives. In summary, the experimental results demonstrate that the proposed fine-tuned DistilBERT model effectively meets the research objectives by delivering high accuracy compared to the prior model.

5. CONCLUSION

In conclusion, this study presents a significant contribution to social engineering attack detection by introducing a fine-tuned DistilBERT model, optimised for both performance and efficiency. With an accuracy of 99.46%, the model demonstrates a substantial improvement over previous approaches, such as SOCIALBERT, which achieved 97.5%. This work contributes to the body of knowledge by leveraging DistilBERT's contextual understanding capabilities to enhance detection accuracy in binary classification tasks, addressing the critical challenge of identifying social engineering attacks with high precision. The evaluation metrics, including precision, recall, and F1-score, further emphasise the model's robustness and potential to minimise false positives and negatives in real-world applications. While the study is limited to text-based data, excluding multimodal threats involving images, videos, and links, it provides a foundational approach for text-based social engineering detection, setting the stage for future research. The lack of diverse publicly available datasets presents a challenge for generalising the model across different communication formats and languages, but it also points to an opportunity for future work to expand datasets and explore multilingual models. Furthermore, while the DistilBERT is computational efficiency outperforms BERT, there remains room for further optimisation in resource-constrained

environments. Future studies should focus on multimodal detection, dataset diversification, and computational efficiency improvements through techniques such as model quantisation and pruning. These advancements will push the boundaries of social engineering attack detection and pave the way for scalable, efficient, and comprehensive solutions to counter evolving cyber threats.

REFERENCES

[1] Mishra, M. K., & Pandey, K. D. (2024). Social Engineering Attacks and Counter Measures: A Comprehensive Analysis. *International Journal of Advanced Research in Science, Communication and Technology*, 167–171. https://doi.org/10.48175/IJARSCT-17826

[2] Naz, A., Sarwar, M., Kaleem, M., Mushtaq, M. A., & Rashid, S. (2024). A comprehensive survey on social engineering-based attacks on social networks. International Journal of ADVANCED AND APPLIED SCIENCES, 11(4), 139–154. https://doi.org/10.21833/ijaas.2024.04.016

[3] Salahdine, F., & Kaabouch, N. (2019). Social Engineering Attacks: A Survey. Future Internet, 11(4), 89. https://doi.org/10.3390/fi11040089

[4] Aslan, Ö., Aktuğ, S. S., Ozkan-Okay, M., Yilmaz, A. A., & Akin, E. (2023). A Comprehensive Review of Cyber Security Vulnerabilities, Threats, Attacks, and Solutions. *Electronics*, *12*(6), 1333. https://doi.org/10.3390/electronics1206133

[5] Zerkina, N., Kostina, N., & Pitina, S. A. (2015). Abbreviation Semantics. Procedia - Social and Behavioral Sciences, 199, 137–142. https://doi.org/10.1016/j.sbspro.2015.07.497

[6] Lan, Y. (2021). Chat-Oriented Social Engineering Attack Detection Using Attention-based Bi-LSTM and CNN. 2021 2nd International Conference on Computing and Data Science (CDS), 483–487. https://doi.org/10.1109/CDS52072.2021.00089

[7] Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2020). DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter (No. arXiv:1910.01108). arXiv. https://doi.org/10.48550/arXiv.1910.01108

[8] Moon, K. Z., Salma, U., & Uddin, M. S. (2024). BERT-Based Personalized Course Recommendation System from Online Learning Platform. 2024 6th International Conference on Electrical Engineering and Information & Communication Technology (ICEEICT), 980–985. https://doi.org/10.1109/ICEEICT62016.2024.10534490

[9] Renuka, O., & Radhakrishnan, N. (2024). BERT for Twitter Sentiment Analysis: Achieving High Accuracy and Balanced Performance. Journal of Trends in Computer Science and Smart Technology, 6(1), 37–50. https://doi.org/10.36548/jtcsst.2024.1.003

[10] Wang, Z., Zhu, H., & Sun, L. (2021). Social Engineering in Cybersecurity: Effect Mechanisms, Human Vulnerabilities and Attack Methods. IEEE Access, 9, 11895–11910. https://doi.org/10.1109/ACCESS.2021.3051633

[11] Hoeschele, M., & Rogers, M. (2006). Detecting Social Engineering. In M. Pollitt & S. Shenoi (Eds.), Advances in Digital Forensics (Vol. 194, pp. 67–77). Kluwer Academic Publishers. https://doi.org/10.1007/0-387-31163-7_6

[12] Jain, A. K., & Gupta, B. B. (2018). Rule-Based Framework for Detection of Smishing Messages in Mobile Environment. Procedia Computer Science, 125, 617–623. https://doi.org/10.1016/j.procs.2017.12.079

[13] Bhakta, R., & Harris, I. G. (2015). Semantic analysis of dialogs to detect social engineering attacks. Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015), 424–427. https://doi.org/10.1109/ICOSC.2015.7050843

[14] Sawa, Y., Bhakta, R., Harris, I. G., & Hadnagy, C. (2016). Detection of Social Engineering Attacks Through Natural Language Processing of Conversations. 2016 IEEE Tenth International Conference on Semantic Computing (ICSC), 262–265. https://doi.org/10.1109/ICSC.2016.95

[15] Lansley, M., Polatidis, N., Kapetanakis, S., Amin, K., Samakovitis, G., & Petridis, M. (2019). Seen the villains: Detecting Social Engineering Attacks using Case-based Reasoning and Deep Learning. ICCBR Workshops. https://api.semanticscholar.org/CorpusID:207967464

[16] Lansley, M., Polatidis, N., & Kapetanakis, S. (2019). SEADer: A Social Engineering Attack Detection Method Based on Natural Language Processing and Artificial Neural Networks. In N. T. Nguyen, R. Chbeir, E. Exposito, P. Aniorté, & B. Trawiński (Eds.), Computational Collective Intelligence (Vol. 11683, pp. 686–

696). Springer International Publishing. https://doi.org/10.1007/978-3-030-28377-3_57

[17] Lansley, M., Kapetanakis, S., & Polatidis, N. (2020). SEADer++ v2: Detecting Social Engineering Attacks using Natural Language Processing and Machine Learning. 2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA), 1–6. https://doi.org/10.1109/INISTA49547.2020.9194623

[18] Lopez, J. C., & Camargo, J. E. (2022). Social Engineering Detection Using Natural Language Processing and Machine Learning. 2022 5th International Conference on Information and Computer Technologies (ICICT), 177–181. https://doi.org/10.1109/ICICT55905.2022.00038

[19] Abobor, M., & Josyula, D. P. (2023). SOCIALBERT a Transformer based Model Used for Detection of Social Engineering Characteristics. 2023 International Conference on Computational Science and Computational Intelligence (CSCI), 174–178. https://doi.org/10.1109/CSCI62032.2023.00033

[20] Distilbert/distilbert-base-uncased · Hugging Face. (2024, March 11). Hugging Face. https://huggingface.co/distilbert/distilbert-base-uncased

[21] Bakirarar, B., & Elhan, A. H. (2023). Class Weighting Technique to Deal with Imbalanced Class Problem in Machine Learning: Methodological Research. *Turkiye Klinikleri Journal of Biostatistics*, *15*(1), 19–29. https://doi.org/10.5336/biostatic.2022-93961