Content Based Approach for Detecting Smishing Messages in Mobile Phones Using an Improved Convolutional Neural Networks Model

Rose Mueni Mbevi[1]
John Kamau[2]
Faith Mueni Musyoka[3]

[1]rosembevi6@gmail.com
[2]jkamau@mku.ac.ke
[3]mueni.faith@embuni.ac.ke

[1, 2]Mount Kenya University, [3]University of Embu, [1,2,3]Kenya

https://doi.org/10.51867/ajernet.6.2.17

ABSTRACT

SMS stands for Short Message Service (SMS). Short messaging service is a text messaging service where a user can send short messages via a mobile device. Short message service has evolved and become very popular as a communication medium in the last decade. It has become a more effective mode of communication compared to email. Unfortunately, smishing (SMS phishing) has emerged as the most common type of spam because traditional detection methods have difficulty understanding the informal nature of these messages. An improved class of CNN-based models targeted at accurate detection of smishing on mobile devices was developed. Deep learning theory was used in this work. (UCI) refers to University of California, Irvine (UCI). The UCI Machine Learning Repository contains datasets, domain theories, and data generators used by the machine learning community to empirically study machine learning algorithms. In this study, a research design was carried out and samples of the UCI Machine Learning Repository were used to build an experimental model. The analyzed dataset was a set of 5, 574 SMS messages from both spam and non-spam messages. The performance metrics used were Precision, Recall, F1-Score and Accuracy. The CNN model used for evaluation, had a bigger number of hidden layers for better detection. A higher accuracy of 99. 95% was achieved, indicating good performance and better detection of the SMS spams (SMS phishing). In the analysis mentioned in the text, the text preprocessing greatly contributes to improved detection accuracy and CNN outperforms the traditional detection methods. It shows that the sophisticated nature of smishing attacks make it necessary for advanced detection mechanisms to be applied to prevent future SMS threats. Although some authorities for implementation should allocate resources to implement these solutions, other authorities must define roles for different detection systems in order to realize the ideal and continuous performance of the detection tools. However, various authorities should also continuously enhance detection mechanisms by feature enhancement, data augmentation, and regular performance evaluation. The training for the staff and establishment of the performance benchmarks needs to be implemented. The users should also contribute with their comments, reports suspicious message, and raising awareness for each other, while all other stakeholders should make available their expertise and resources to support this work.

Keywords: CNN, Content Based Approach, Legitimate Messages, Smishing Detection, Smishing Messages

I. INTRODUCTION

The inexorable advancement of Information Technology (IT) has rendered mobile phones an indispensable facet of our daily lives. Smartphones have gained predominance among users due to their multifaceted functionalities. Their extended battery life, compact form, and portability have contributed to their widespread popularity. Recent years have witnessed an upsurge in smartphone adoption, attributable to the availability of budget-friendly models, cost-effective Wi-Fi connections, affordable data plans, and the proliferation of app marketplaces. Mobile devices are embraced by individuals across various economic strata and age groups. Alongside this surge in mobile usage, the security risks targeting mobile smartphones have also escalated (Mishra & Soni, 2019a).

Smishing, a blend of "SMS" and "phishing," is a type of cyber fraud where fraudulent text messages are used to deceive individuals into disclosing sensitive information such as banking details, credit card numbers, and login credentials (Shweta & Main, 2023). In the course of a Smishing attack, malevolent actors dispatch insidious text messages to victims' smartphones. These messages frequently incorporate links to malicious websites or phishing mechanisms that solicit the user's sensitive information via a user interface. Once activated, malware infiltrates the user's device, subsequently instigating malicious activities. Given the propensity of smartphone users to frequently input their user credentials, confidential financial information such as credit or debit details, user IDs, and passwords, are unwittingly exposed via the interface provided by these attackers(Mishra & Soni, 2019a).

Smishing attacks executed through SMS targeting mobile users, are experiencing a daily surge. Attackers increasingly favor SMS over email as their medium of choice for deceiving users. With approximately 7 billion mobile subscriptions globally compared to just 2.5 billion email users, this contrast highlights the dominant preference for SMS(Morreale, 2017). Prior research endeavours aimed at filtering SMS Spam predominantly relied on manually identified features. Notably, neural network models have demonstrated exceptional capabilities in sentence and document modelling. Among these models, the Convolutional Neural Network (CNN) stands out as the mainstream architecture for such modelling tasks, adopting distinct approaches to deciphering natural language. This endeavour aims to leverage the benefits of a content-based strategy and introduces an innovative, integrated model referred to as CNN for sentence representation and text classification.

1.1 Statement of the Problem

Smishing (SMS phishing) attacks are on the rise as mobile devices become more common in our daily life. They use malicious text messages to gain personal information or download dangerous software to users. Due to their short and informal nature the SMS message makes it difficult for both the user and traditional detection systems to recognize malicious contents. Spammers continuously improve their malicious tactics so that detecting them using static keyword-based methods or basic filtering mechanisms becomes challenging (Jain et al., 2022). According to recent studies, SMS spam has a negative impact on 68% of mobile phone users with the presence of harmful activities including Smishing (a form of phishing) in SMS spam (Alexander, 2024).

SMS spam is likely to grow rapidly due to the availability of bulk pre-paid SMS packages for low cost. Also, the mass communication media have higher responses rate because of its personal and trusted nature. Criminal organizations have turned to the mobile communication medium based on SMS for various reasons like ease of use, speed, reliability, and affordability (Delany et al., 2012). Though there were several approaches to detection of SMS spam before, its problem is that feature extraction technique was not efficient. Instead, the method of finding out a suitable feature for exact classification was based on knowledge acquired in prior stage as well as domain knowledge and features are always switched off and on depending on changes in data acquisition stage (Gomaa, 2020).

In its report, the study sought to address the urgent SMS spam problem by looking at smishing tactics designed to trap users into divulging sensitive information. As the widely adopted standard for communications, SMS messaging has become an enormous target for cybercriminals. In fact, 98% of mobile users who check their SMS messages on the same day deal with what they receive. Yet as unsolicited potentially malicious messages present serious security threats and nuisance to both people and organizations, they are also a large source of annoyance.

A set of key issues in mobile spam detection is tackled by the present study through the use of state-of-the-art machine learning techniques that not only contribute to the advancement of cybersecurity research but also yield practical solutions for SMS detection. The results show that improving detection methods to suit SMS communication context leads to enhanced security for users against cyber threats.

To address the unsuitable feature engineering problem, Deep neural networks offer a promising solution. Deep learning is a special subfield in machine learning that has multiple layers of memory for automatic feature recognition and non-supervised feature extraction. In order to address this research gap, the research paper used a method of deep neural networks to detect spam SMS messages. More precisely, Convolutional Neural Network (CNN) was used.

1.2 Research Objectives

i. To examine existing techniques for detecting smishing messages in mobile phones.
ii. To develop an improved CNN based model for Smishing messages detection.
iii. To test the proposed CNN based model for Smishing messages detection.

II. LITERATURE REVIEW

2.1 Theoretical Review

Deep Learning Theory is the theory of the structure and training methods of deep neural networks (e. g., Convolutional Neural Networks) which focuses on how multilayered models of representation can automatically learn hierarchical representation of data. In the context of detection of smishing using CNNs, these deep learning concepts facilitate the design of CNN architectures that automatically obtain sophisticated features from raw SMS messages and thus improve detection performance (LeCun et al., 2015).

In deep learning based models the feature extraction stage typically involves manual intervention (based on domain knowledge). The accuracy of the classifier depends heavily on the features that are manually extracted from the data. Deep learning subsequently changes this paradigm and automatically extracts the features of hidden data eliminating the need for manual extraction. Some examples of such implementations of the above approach include CNN. Convolutional Neural Networks have been shown to be highly successful on this feature extraction task in deep

learning as it has been proven to be very effective on tasks with much higher complexity when solving pattern recognition problems. The experimental analysis report of the Convolutional Neural Networks based model on a dataset of randomly sampled data which is run under 10-fold cross-validation showed the effectiveness of the model. The model achieved an excellent 99. 44% accuracy in identifying spam text messages from legitimate text messages (Roy et al., 2020)**.**

The CNN-based model has three layers and operates in three main stages: *Word Matrix Creation:* In this stage, a matrix of word vectors is created. *Hidden Feature Identification:* The model finds out hidden features in the text data. *Classification:* The model classifies the data into predefined classes. The data before the creation of the word matrices are preprocessed where stop words (common words in English which don't really have any meaning) are deleted. It's done with stemming so that variations of words are treated as the same word which makes it easier to attach numerical values. This CNN-based model has proved to be effective in text classification tasks, equivalent to the functionality of the visual cortex in the animal brain. The difference is that this model works on matrices of word vectors. Therefore it can be used to solve the task of Smishing detection (Amin & Nadeem, 2019).

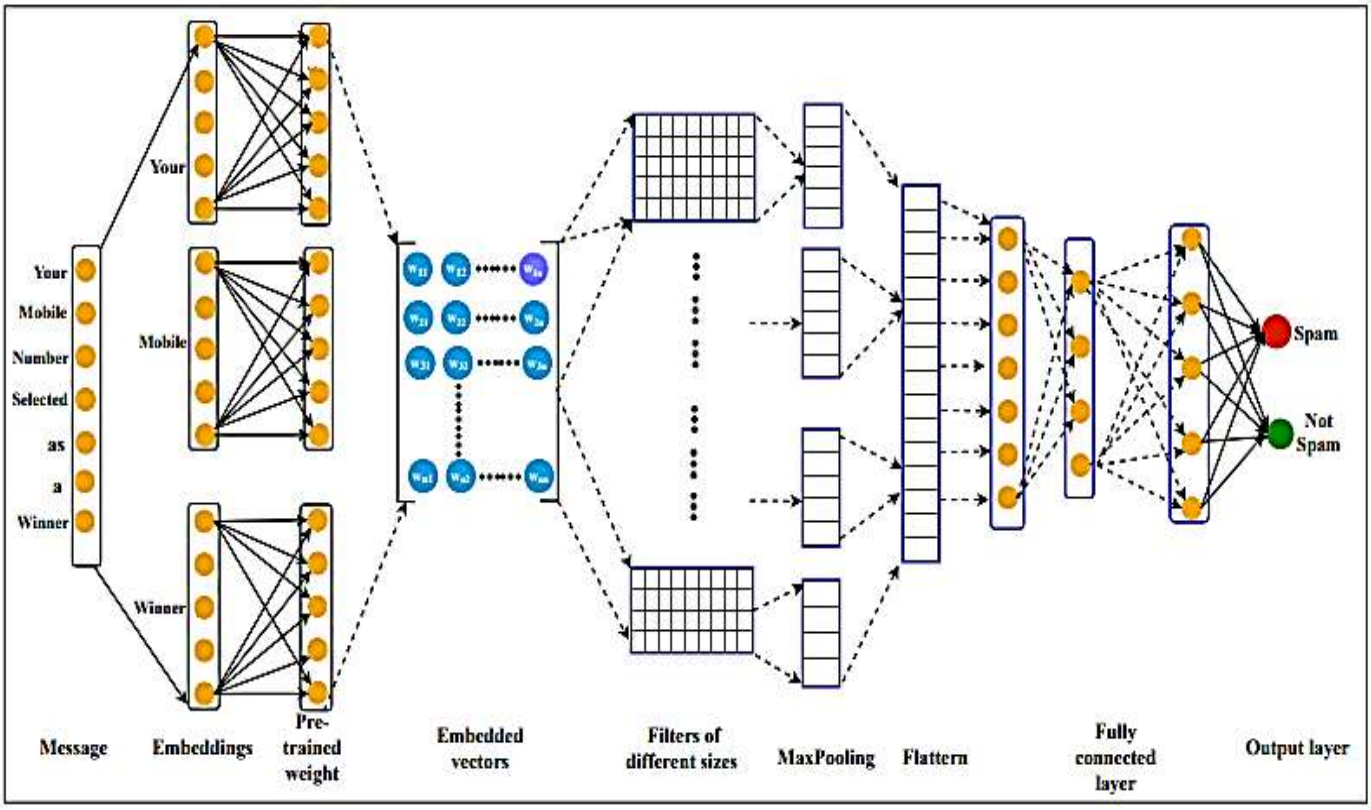

Figure 1
A Framework of Convolutional Neural Network (Roy et al., 2020)

The CNN architecture depicted in Figure 1 arrives at a culmination of the various layers that make up a CNN. The input layer utilizes data, which are the word embeddings that will be encoded as word matrices. The model now formulates its patterns, starting in the convolutional layer, where the output is shared in the pooling layers. The data are then transformed into a single vector to deliver multiple pieces of information for the fully connected layers to process together this is referred to as flattening. The fully connected layers then make inferences, based on the patterns developed and from other layers, and lastly classify Smishing attacks (Roy et al., 2020).

2.2 Empirical Review

2.2.1 Smishing Detection Techniques by Other Researchers

In the research paper titled 'SMS Phishing and Mitigation Approaches', the authors provide an extensive overview of strategies, techniques, and approaches for detecting Smishing. This valuable research aims to embellish the knowledge of analysts, movable consumers, and students about the significance of countering Smishing attacks(Mishra & Soni, 2019b).Tanbhir et al. (2025) developed a hybrid model integrating BERT (Bidirectional Encoder

Representations from Transformers)embeddings with character-level CNNs for detecting Bangla smishing texts, achieving an accuracy of 98.47%(Tanbhir et al., 2025).

Goel et al. (2024) introduced a framework that normalizes informal language in SMS messages and employs a Naive Bayesian classifier, achieving a detection accuracy of 96.2%(Goel et al., 2024). Mishra and Soni (2022) implemented a smishing detection model using ANN(Artificial Neural Network), which outperformed other machine learning algorithms with a final accuracy of 97.40% (Mishra & Soni, 2022).

Mishra and Soni (2020) presents a Smishing detection model with four distinct stages, achieving an impressive accuracy rate of 96.2%. The basic objective concerning this research is to judge the legitimacy of URLs embedded in SMS messages(Mishra & Soni, 2020). Remmide et al. (2025) introduced a decentralized smishing detection method using federated learning, employing LSTM (Long Short Term Memory) and Bi-LSTM (Bidirectional LSTM) models, and achieving an accuracy of 88.78%(Remmide et al., 2025). Mahmud et al. (2024) proposed a hybrid deep learning approach combining Bidirectional Gated Recurrent Units (Bi-GRUs) and Convolutional Neural Networks (CNNs) for smishing detection, addressing the limitations of traditional methods(Mahmud et al., 2024).

Another content-based approach is adopted in the study by Mishra and Soni (2019a), this method involves scrutinizing the occupancy of form tags in the source code and the downloading of malicious files through links found in SMS messages to identify Smishing messages(Mishra & Soni, 2019a).

2.2.2 Deep Learning Models for Smishing Detection

This study examined multiple deep learning models for unsolicited call classification using Tiago's dataset. The preprocessing phase involved text normalization, tokenization, lemmatization, and the removal of numbers, punctuation, stop words, and irrelevant content. Two advanced deep learning architectures were implemented: A Convolutional Neural Network (CNN) and a hybrid CNN-Long Short-Term Memory (LSTM) model. To improve accuracy, the study incorporated two-word embedding techniques—the BUNOW (Bi-Unique Number of Word) model and a security-based method. While commonly used in sentiment analysis, these techniques were effectively applied to text classification. After 15 iterations with a 70%-30% train-test split, the highest recorded accuracy was 98.44%, achieved using a combination of CNN, LSTM, and BUNOW models (Maqsood et al., 2023).

Deep learning models are often criticized for their lack of transparency, as they function as black-box systems. Implementing techniques such as attention mechanisms or saliency maps can improve interpretability by offering insights into the model's decision-making process. This, in turn, enhances trust and usability in real-world applications (Testas, 2023). Sheikhi et al. (2020) introduced Averaged Neural Network with one hidden layer which is used for the classification of spam messages and reported an accuracy of 98.8%(Sheikhi et al., 2020).

Jain et al. (2020) introduced backpropagation algorithm which is used for the classification of spam messages. 14 spam features of the messages were selected for the identification of spam messages and finally, their classification showed an accuracy of 95.81%(Jain et al., 2020). Nivaashini et al. (2018) introduced Deep Neural Network which is used for segregation of spam messages and Restricted Boltzmann Machine is used for extraction of relevant features. Neural Network with three hidden layers provided the best accuracy for their system(Nivaashini et al., 2018). Jain and Gupta (2019) proposed segregation of spam messages with the help of ten features selected using Neural Network. The proposed approach was also proved effective for the zero-hour attack. The accuracy shown by the system is 98.74%(Jain & Gupta, 2019).

Roy et al. (2020) developed a deep learning model to classify spam and legitimate text messages using the SMS Spam Collection dataset, achieving an outstanding accuracy of 99.44% with CNN and LSTM models. The study focused on classification rather than analyzing the characteristics of the extracted embeddings. Achieving a precision rate of 99.44% highlights the effectiveness of deep learning techniques in distinguishing spam from legitimate messages—an essential aspect of security applications where reducing false positives and negatives is crucial. The researchers conducted a comparative analysis of CNN and LSTM models, providing valuable insights into their strengths and weaknesses for SMS classification. This comparison helps in understanding the advantages of each approach and guides future research in selecting the most suitable models for specific tasks(Roy et al., 2020).

III. METHODOLOGY

3.1 Research Design

The study introduces a content-based approach for Smishing message detection based on a convolutional neural network (CNN) model on mobile devices. Both explanatory and experimental research design experiments are used as both of them offer distinct but complementary approaches for investigating the phenomenon and to elucidate the hypotheses related to Smishing detection.

3.1.1 Experimental Steps

This will describe technical approaches used in the work. Here the aim of the study was to apply CNN model that is a robust model capable to learn and identify patterns from different SMS messages. The experimental model was implemented using Tensorflow and conducted in the Google Colab environment. Figure 2 is a flowchart depicting the experimental workflow used for the study. Some of the steps are: Data collection from the UCI repository, data preprocessing which included text cleaning, tokenization, lemmatization, and special characters' removal, feature extraction which included use of TF-IDF, data splitting into training and testing set and classification using the CNN model to either predict the SMS messages as either spam or nor spam.

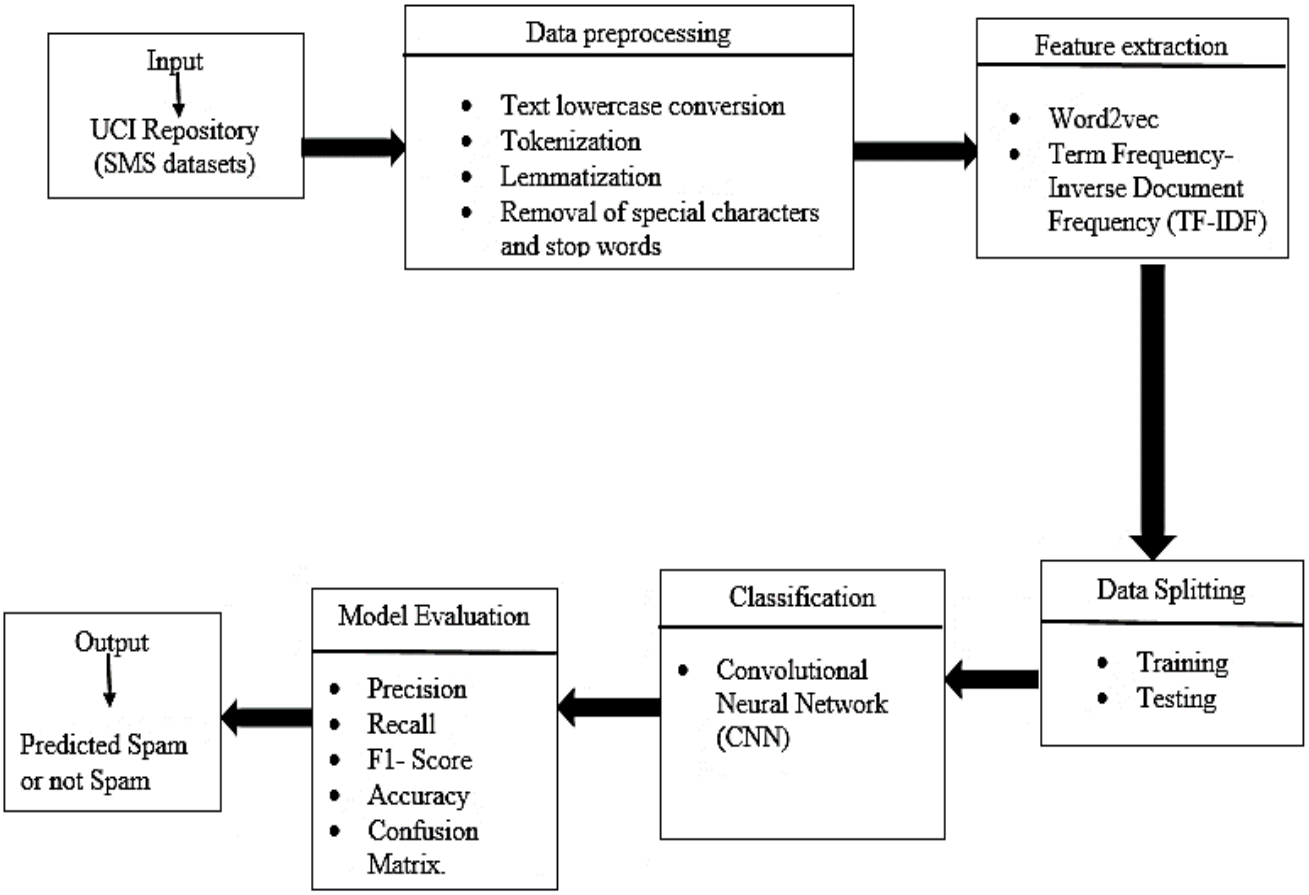

Figure 2

Experimental Workflow (Architecture of the Study)

During laboratory technique stage, the experiment plan was implemented accurately using Python programming language and applying Tensorflow as machine learning library and Keras as a framework for model development. Using Python's extensive library of resources made the project easier in many ways, from data pre-processing to model evaluation. Also, the availability of computing resources (system RAM, GPU RAM, disk storage) to carry out deep learning experiments and trains in a scalable and performant way have increased responsible for enabling large datasets and neural network architectures. This was key in the iterative model refinement and performance optimization phase during experimentation.

3.2 Target Population

It consisted of SMS text messages dataset from the SMS Spam Collection dataset found in the UCI Machine Learning Repository (Almeida & Hidalgo, 2011). This included a set of 5,574 messages grouped into a spam (747 messages) and ham (4,827 messages) combination. These messages were collected from different sources including mobile phone users in England and have been annotated for classifying a spam.

3.3 Sample Size Determination and Sampling Procedures

In conclusion, determining the appropriate number of samples for detecting smishing messages on mobile devices with the use of secondary data, brought in many factors, including the characteristics of the secondary data, the statistical approaches to follow, the factors of power analysis, precision requirements, limited feasibilities, and ethical

implications. The study utilized a sampling method known as simple random sampling, also known as chance sampling. The purpose of simple random sampling is to provide an equal opportunity for each message to be selected. Random selection minimizes bias in the selection of the sample and facilitates generalizing the findings to the sampling population (Emerson, 2015).

Random sampling includes the use of a random number generator or systematic procedures to randomly select messages, thus providing a sample of messages that are appropriate to examine the detection of smishing messages. The method of random sampling aims to include all spectrums of mobile device messages and provides sufficient scrutiny of how well the Convolutional Neural Networks infer smishing messages. Additionally, random sampling provides further reliability, and credibility, and enhances the experimental design of the study.

3.4 Data Collection Methods

This study used secondary data sources to provide insightful comment regarding alignment with recent studies. Sources in data acquisition were on web-based platforms and data banks and thus the set of data was accessible from the UCI Machine Learning Repository. Text message labeled data set is publicly accessible to use via the UCI Machine Learning Repository, and its first data curator was Almeida in 2011(Almeida & Hidalgo, 2011). This corpus consists of 5,574 English SMS messages, which have been divided into two distinct groups: legitimate messages, or "ham," and spam messages. The corpus was gathered with caution from different sources, including the UK-based Grumbletext website, the NUS SMS Corpus (NSC), and the SMS Spam Corpus version 0.1 Big. The corpus sources are explained in Table 1.

Table 1

Source of the Dataset

Spam Dataset	**Total**
Grumbletext website	425 (spam)
NUS SMS Corpus (NSC)	3,375 (ham)
Caroline Tag's PhD Thesis	450 (ham)
Corpus v.0.1 Big	1,002 (ham), 322 (spam)

Source*:* Tiago Ameida (2011).

As per the information given in Table 1, the dataset utilized by this research consists of a total of 5,574 labelled messages. Out of these, 4,827 messages are labelled as 'ham' and the remaining 747 are labelled as 'spam.' Interestingly, the dataset consists of two highly disparate columns: one showing message labels ('ham' or 'spam') and the other containing text message content.

3.5 Data Analysis Techniques

This study utilized descriptive and inferential analysis approach to comprehensively examine and interpret spam text data. Descriptive analysis aims to address the question, "What occurred?" It served the purpose of identifying areas of strength and areas requiring improvement. Furthermore, it established the groundwork for more advanced data analysis procedures(Erdelyi, 2020). The analysis of data encompassed descriptive statistics, including the presentation of data in tables, frequencies, percentages, and graphical representations. Inferential statistics, in contrast, were used for making predictions, inferences, or for making inferences about a broader population based on a data sample. In the context of spam text analysis, inferential statistics applied regression analysis to explore relationships between various features (e.g., message length, time of day) and the likelihood of a message being spam**.**

The data was analyzed using Python, and the results illustrated through tables, charts, and graphs. These analyses were instrumental in determining whether there is a necessity to formulate additional information security measures to enhance the secure provision of services to mobile phone users. The use of both descriptive and inferential analysis techniques aligns with the study's objective to comprehensively asses and analyze spam text data. These techniques afford a robust framework or understanding patterns assessing correlations and divulging conclusions crucial for enhancing information security measures for mobile phones users who are susceptible to malicious Smishing attacks.

3.6 Ethical Considerations

Ethical considerations in research are underpinned by a set of principles that provide a guiding framework for research design and its execution. Scientists and researchers are duty-bound to uphold a specific code of conduct when gathering data from individuals. These guiding principles play a crucial role in protecting research participants' rights while strengthening the study's credibility, and upholding the integrity of scientific and academic pursuits, in line with the insights of(Bhandari, 2021).

To facilitate the study, an introductory letter from Mount Kenya University (MKU) was obtained. This letter served as an attachment to the essential documents needed to secure a research authorization from the National Commission for Science, Technology, and Innovation (NACOSTI). Furthermore, the research endeavour encompassed the inclusion of a consent letter for study group participants, signifying their voluntary involvement. It is essential to underscore that the data gathered was exclusively employed for this research and managed with strict adherence to the tenets of research ethics.

IV. FINDINGS & DISCUSSION

4.1 Data Importation

Data for this study was collected from secondary sources, which offer valuable insights into how the research aligns with existing studies. The sources in the data collection were from online platforms and data repositories hence the data was sourced from the UCI Machine Learning Repository (*https://archive.ics.uci.edu/dataset/228/sms+spam+collection*)(Almeida & Hidalgo, 2011).

```
[ ] import numpy as np
    from google.colab import drive
    drive.mount('/content/drive')

    #Specify the delimiter as this is a text file which is not tabular in nature
    df = pd.read_csv('/content/SMSSpamCollection', delimiter='\t', quoting=3)
    df.head()
```

Figure 3
Data Importation

To load the data from the google drive, mounting the google drive was essential. After mounting the Google Drive, the dataset was located and the link copied and attached to the read_csv function as a string. UCI Datasets was downloaded and saved into the Colab Environment. Figure 3 which is the provided code snippet demonstrates how to interact with Google Drive using the Google Colab (Colaboratory) environment. Overall, this code snippet demonstrates the basic steps involved in accessing and reading data from a user's Google Drive within the Google Colab environment, which can be a useful tool for data-driven projects and collaborations.

4.2 Data Preprocessing

The above process was done to make data ready for text classification. A unified function was created to incorporate all preprocessing steps, enabling seamless application to the text column (Figure 4). The code snippet (Figure 4) focuses on the preprocessing of text data, which is a crucial step in natural language processing (NLP) tasks, the process starts by defining a set of stop words using the Natural Language Toolkit (NLTK), which includes common English words (e.g., 'and,' 'the,' 'is') that usually carry little semantic significance. This is essential as eliminating these words reduces data noise and enhances the efficiency of subsequent analyses.

```
# Remove all stop words
stop_words = set(stopwords.words('english'))
def preprocess_text(text):
  """
  Function to preprocess the text by lowercasing the text, removing special characters, tokenizing, and lemmatizing.
  """
  # Converting the text to lower case
  text = text.lower()
  # Remove all punctuations in the text
  text = re.sub(r'\W', ' ', text)
  # Tokenizing the text
  tokens = word_tokenize(text)
  lemmatized_tokens = []
  # A for loop to loop through all tokens and lemmatize them
  for token in tokens:
      if token not in stop_words:
          doc = nlp(token)
          lemmatized_tokens.append(doc[0].lemma_)

  return ' '.join(lemmatized_tokens)

df['text'] = df['text'].apply(preprocess_text)
```

Figure 4
Data Preprocessing

The preprocess_text function embodies several key preprocessing steps. First, it converts the input text to lowercase, ensuring uniformity and helping to avoid discrepancies caused by case sensitivity. Next, it removes punctuation using a regular expression (REGEX), which is essential for simplifying the text and focusing on the actual words. Tokenization was performed using the NLTK library, breaking the cleaned text into individual words or tokens. This step is crucial for detailed text analysis at a granular level**.** The function then initializes an empty list for storing lemmatized tokens. Lemmatization is the process of converting words to their base or root form (e.g., 'running' to **'run')** which is beneficial for standardizing the data and enhancing the model's understanding. The for loop iterates through the tokens, checking if each token is not a stop word before proceeding to lemmatize it.

4.3 Feature Extraction

Feature extraction involved the label encoding and the utilization of n-gram along with word vectors created using TF-IDF. This method calculates the vector distances between words, resulting in the extraction of relevant features for each message. After doing the text pre-processing, the label encorder was used to encode the target variable into numerical formart, this is recommended to ensure the data is in a format the machine learning model can understand.

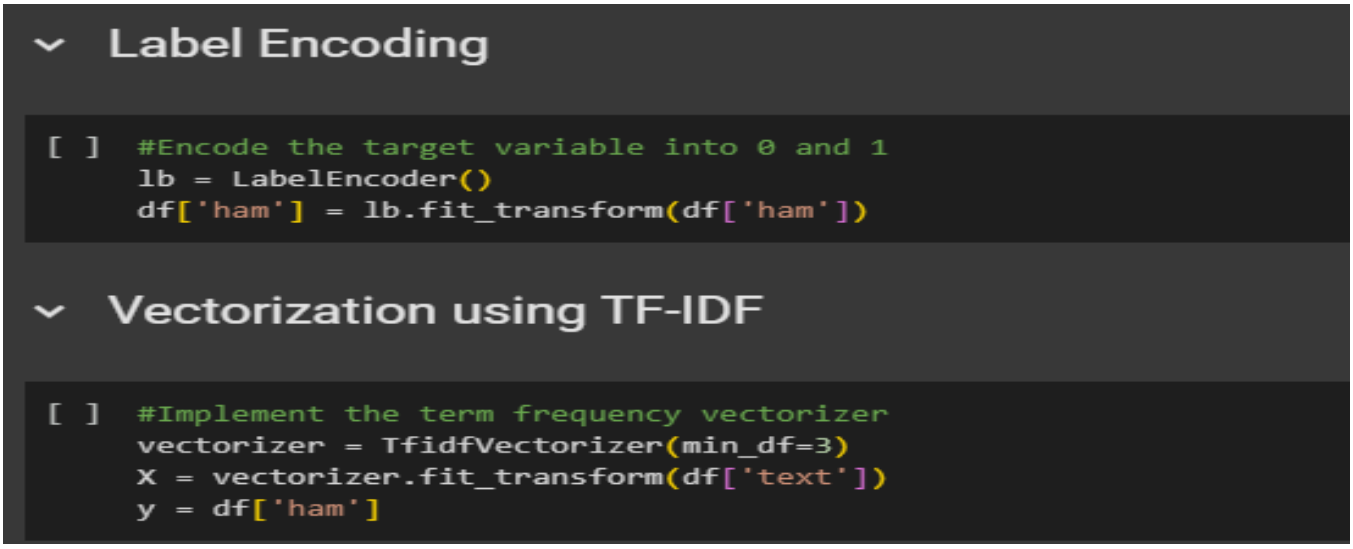

Figure 5
Label Encoding and Vectorization Using TF-IDF

Figure 5 shows the code snippet which highlights some important actions to perform on a dataset before feeding it to a machine learning model, with focus on text classification. First, the target variable; it is the feature indicating 'ham' or 'spam' message classification is transformed using the LabelEncoder to convert its values to binary form, in this case, 0 and 1. Such a change is necessary because most machine learning algorithms take numerical inputs. Since, the "ham" column has been encoded, the model is well capable of distinguishing the two classes without much confusion.

4.4 Data Splitting

Data splitting involved the training and testing of the data. The dataset is divided into two parts: a training set to train the model and a test set to evaluate its performance. A common ratio is 80/20 or 70/30. This study used 70/30 ratio whereby, the option test_size=0.3 was used to define the percentage of the data: 30% will be used for test data, while the rest will be used for training; the random_state =42 parameter allows achieving the same split provided the code is run again.

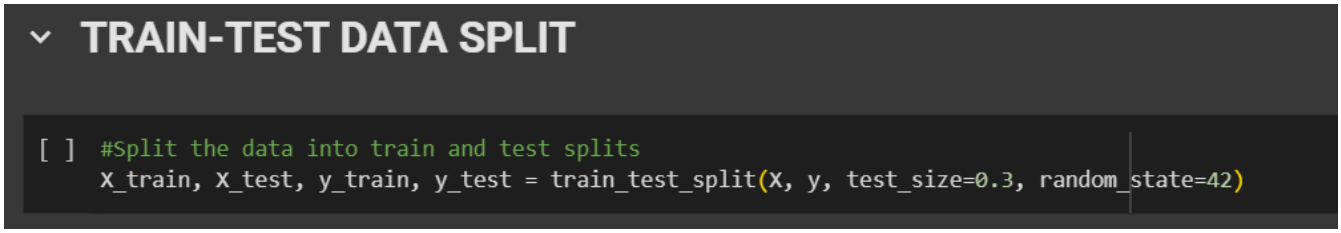

Figure 6
Validation of the Model using Train-Test Split

Figure 6 outlines the code snippet which helps to illustrate the preprocessing involved in processing text data for machine learning so that the data models are properly encoded, vectorized, and divided for modelling and evaluation.

4.5 Classification

It involves development and training of the Model. After splitting the data, the model was initialized using additional two layers with different activation functions from the output layer. Of importance is the last output layer which has the Softmax activation function. The model has three tiers, or what in this model are called layers. The first two, a dense neural layers with 500 neurons in each are active layers that use the ReLu activation function. The last layer is one more dense layer containing 2 neurons that correspond to two classes, namely ham and spam. This uses the softmax activation function which scales the output of the network by converting it into probabilistic measure of the classes. This is especially important for scenarios where model needs to classify an image into one of several categories such as multi-class classification because it provides a numeric value for each class for ease of making the final decision based on the maximum value.

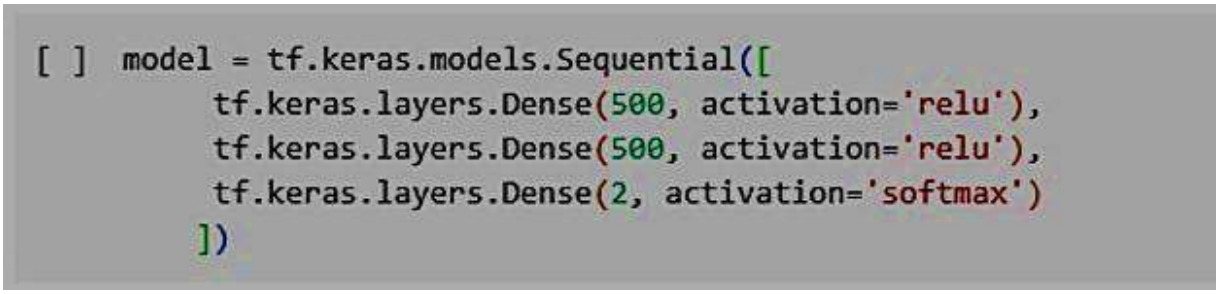

Figure 7
Model Initialization

Figure 7 of the code snippet communicates the choice elements that fit into building a neural network for classification with emphasis on those choices that refine the performance of the model. By grounding itself into the practices All the model's parameters for feature extraction from the preprocessed text data are ready to be learned with the help of the activation function.

4.5.1 Model Parameters

A total of 500 Neurons were chosen in this process. *Balance Capacity and Efficiency:* Provided substantial learning capacity without being overly large or small**.** The neuron size was optimized to be sufficient for learning diverse features while preventing overfitting to the training data. *Use of ReLU Activation Function:* In this case, ReLU Activation Function was used for learning Layers. This allowed the model to capture intricate patterns and relationships in the data while also accelerating the training process. The computation process helped avoid the vanishing gradient problem, ensuring that neurons remained active and contributed to learning. The activation mechanism converted all negative inputs to zero, leaving positive inputs unchanged.

Use of Softmax Activation Function: The output layer used Softmax Activation Function, converting the raw prediction scores (logits) into probabilities. In this case, the probabilistic interpretation ensured the output probabilities sum to 100%, allowing clear probabilistic interpretation of predictions. Moreover, multi-class classification tasks was used providing the likelihood of each class. In this model, the output layer function outputs two probabilities corresponding to the two classes (ham and smishing).

4.5.2 Converting the Data to Tensors

Tensors, which are multi-dimensional arrays, were utilized to represent various data types such as text, numerical values, and image (Figure 8). Due to their ability to facilitate effective data manipulation and processing, they are a basic idea in many machine learning and deep learning techniques.

```
[ ]  #Convert the train and test sets into a formart tensorflow can ingest and train.
     y_train = tf.convert_to_tensor(y_train)
     X_train = tf.convert_to_tensor(X_train.toarray())

     X_test = tf.convert_to_tensor(X_test.toarray())
     y_test = tf.convert_to_tensor(y_test)
```

Figure 8
Converting the Data to Tensors

Overall, this snippet (Figure 8) represents a crucial stage in the machine learning process. As the name suggests, this stage changes the input datasets into tensor format so that the TensorFlow model can take them in and train or evaluate them. This is in line with best practice when doing deep learning since the format of data determines the performance and accuracy of the outcomes.

4.5.3 Model Training

The model was compiled using the adam optimizer because it offers a quick and fast convergence during the model training. It also offers an automated learning rate setting parameter which ensures training optimization. The Cross-entropy loss was used because it was efficient for multiclass classification.

```
[ ]  model.compile(optimizer='adam',
                   loss='sparse_categorical_crossentropy',
                   metrics=['accuracy'])
```

Figure 9
Model Training

Figure 9 of the code snippet is related to compilation of a neural network model with TensorFlow's Keras API which is the pre-requisite step to the training of the model. This step includes defining an optimizer, a loss function, and, of course, an evaluation metric, which are essential aspects for any model and affect its training. *Model training (Fitting*): The model was trained using 10 epochs calculating loss and accuracy as training progresses. Figure 10 shows the results of the improved CNN model.

```
[ ]  #Train the model on 100 epochs
     model.fit(X_train, y_train, epochs=10)

     Epoch 1/10
     122/122 [==============================] - 3s 3ms/step - loss: 0.2232 - accuracy: 0.9039
     Epoch 2/10
     122/122 [==============================] - 0s 3ms/step - loss: 0.0372 - accuracy: 0.9895
     Epoch 3/10
     122/122 [==============================] - 0s 3ms/step - loss: 0.0078 - accuracy: 0.9982
     Epoch 4/10
     122/122 [==============================] - 0s 3ms/step - loss: 0.0031 - accuracy: 0.9992
     Epoch 5/10
     122/122 [==============================] - 0s 3ms/step - loss: 0.0031 - accuracy: 0.9992
     Epoch 6/10
     122/122 [==============================] - 0s 3ms/step - loss: 0.0025 - accuracy: 0.9995
     Epoch 7/10
     122/122 [==============================] - 0s 4ms/step - loss: 0.0024 - accuracy: 0.9995
     Epoch 8/10
     122/122 [==============================] - 1s 4ms/step - loss: 0.0022 - accuracy: 0.9992
     Epoch 9/10
     122/122 [==============================] - 1s 4ms/step - loss: 0.0021 - accuracy: 0.9995
     Epoch 10/10
     122/122 [==============================] - 1s 4ms/step - loss: 0.0023 - accuracy: 0.9995
     <keras.src.callbacks.History at 0x7f1780203220>
```

Figure 10
Model Training (Fitting)

Figure 10 represents the training process of a deep learning model over 10 epochs**,** showing how the loss decreases and accuracy increases as training progresses. The loss starts at 0.2232 in the first epoch and drops to 0.0023 by the 10th epoch, while accuracy rises from 98.39% to 99.95%**.** This suggests that the model is learning effectively, but there is a risk of overfitting due to excessively low loss values.

To prevent overfitting, the research incorporated learning rate scheduling as a technique to stop overfitting and achieve better generalization with new data. The model sustained effective convergence through training rate adjustments enabled by learning rate scheduling. The implemented learning rate used an adaptive mechanism that began with a high rate for swift learning during early training phases. The learning rate decreased step by step while the training process continued. This method refined the model weights effectively while avoiding excessive optimization and improved its pattern learning capabilities that are independent of training data specificity.

4.6 Performance Evaluation of the Developed Model

Looking at the reported test loss of 0. 0704, the findings show that the predicted values are close to the actual labels hence a good fit of the model. As in

, the optimizer that is used during training as well as the quality of the descriptors that are obtained before feeding the data into our network.

```
[ ]  #Evaluate the models performance on the test sets
     loss, accuracy = model.evaluate(X_test, y_test)
     print(f'Test Loss: {loss}, Test Accuracy: {accuracy}')

     53/53 [==============================] - 0s 4ms/step - loss: 0.0704 - accuracy: 0.9886
     Test Loss: 0.07035035640001297, Test Accuracy: 0.9886363744735718
```

Figure 11
Model Performance Evaluation

Figure 11 indicates the model performance evaluation which calculates both the loss (the test loss) and accuracy (the test accuracy) of the model. The findings show the model is a good fit for detecting smishing messages in mobile phones.

4.6.1 Model Predictions (using Evaluation Metrics)

In the model average accuracy of all classes reaches the highest mark and classification report demonstrates respectable values, especially for class 0 that possesses the precision of 0. 99 and the recall of 1. 00. This is also a sign that the model performs well in the identification of true positives of this class with very small false positives meaning

that the predictions from this model are very correct. This is further confirmed by the F1-score of 0. 99 which is a balanced measure of precision and recall of the automated extraction.

On the other hand, the performance of class 1 indicates 99% precision and 92% of recall value. This could be interpreted as the fact that while the model is accurate in detecting true positives of class 1, there are cases of missed true positives, thus a higher value of false negatives. Having an F1-score of 0. 95 for this class mean that though on the right path, the model could still do better in recognize all the instances belonging to this class. Here the accuracy is 0. 99, which means the current model was very accurate on the entire dataset and the performance is also high as given by the macro and weight averages. From the above table, the macro average of 0. 96 shows that the model performance across classes is average and the weighted average of 0. 99 clarifies that class imbalance does not affect the overall performance of the model.

```
[ ]  #Generate predictions using the model
     y_preds_prob = model.predict(X_test)
     y_preds = tf.argmax(y_preds_prob, axis=1)

     53/53 [==============================] - 0s 2ms/step

[ ]  report = classification_report(y_test, y_preds)
     print(report)

                   precision    recall  f1-score   support

                0       0.99      1.00      0.99      1450
                1       0.99      0.92      0.96       222

         accuracy                           0.99      1672
        macro avg       0.99      0.96      0.97      1672
     weighted avg       0.99      0.99      0.99      1672
```

Figure 12(a)
Model Prediction

Figure 12(a) shows that the model handles the class imbalance quite efficiently as seen from the precision and recall scores above. The class 0 which stands for the ham class had a pretty good score whereas the target variable which is of the minority class containing the spam texts was also good. The model was efficient in capturing and identifying the target of interest which had spam text. Misclassifying non-spam text is detrimental and the model handles this efficiently.

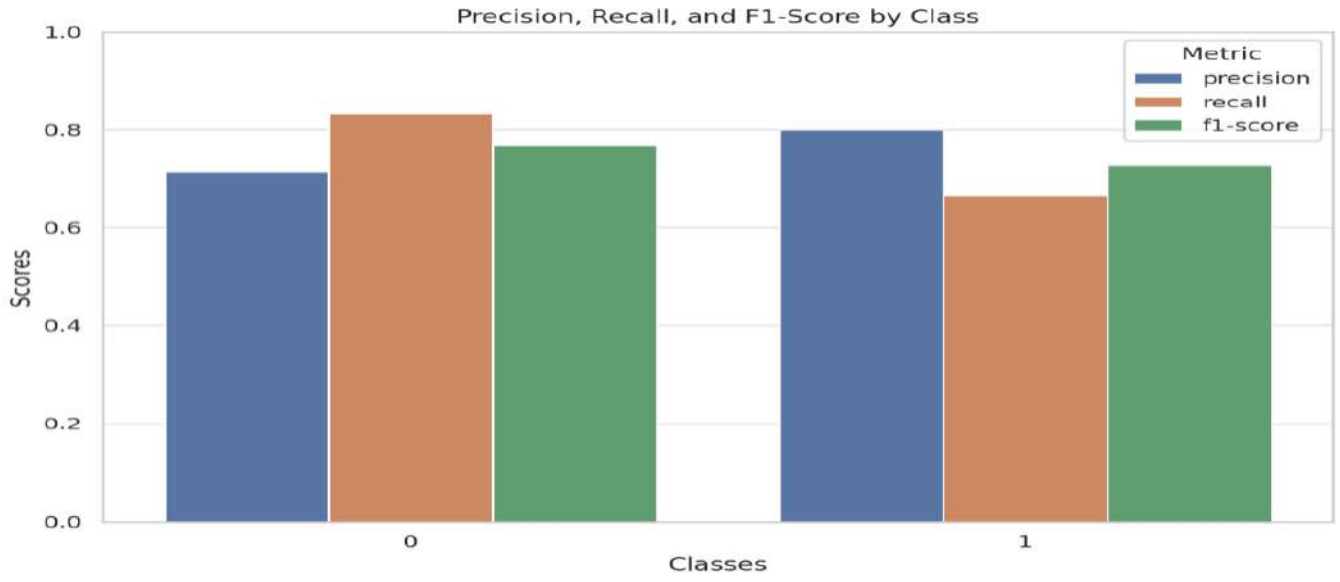

Figure 12(b)
Model Prediction using Bar Chart

Figure 12(b) presents a bar chart comparing Precision, Recall, and F1-Score for two classes (0 and 1). Class 0 has a higher recall, meaning the model effectively identifies most instances of this class, while Class 1 has lower recall, indicating some misclassified instances. Precision for Class 1 is slightly higher than recall, suggesting that when the model predicts Class 1, it is often correct, but it may still miss some actual positives. The F1-scores for both classes are relatively balanced, showing an overall fair trade-off between precision and recall. However, improvements such as addressing class imbalance, adjusting the classification threshold, or using ensemble methods could enhance recall for Class 1 and improve overall model performance.

4.6.2 Confusion Matrix

The confusion matrix (Figure 13) provided a thorough analysis of the model's functionality, enabling evaluation of its advantages, disadvantages, and possible growth areas. The model effectively performed the task of distinguishing between text categories into those that exhibit spam and those that do not, based on the high true positive and true negative rates (Whereby: 0= Ham, 1= Smishing).

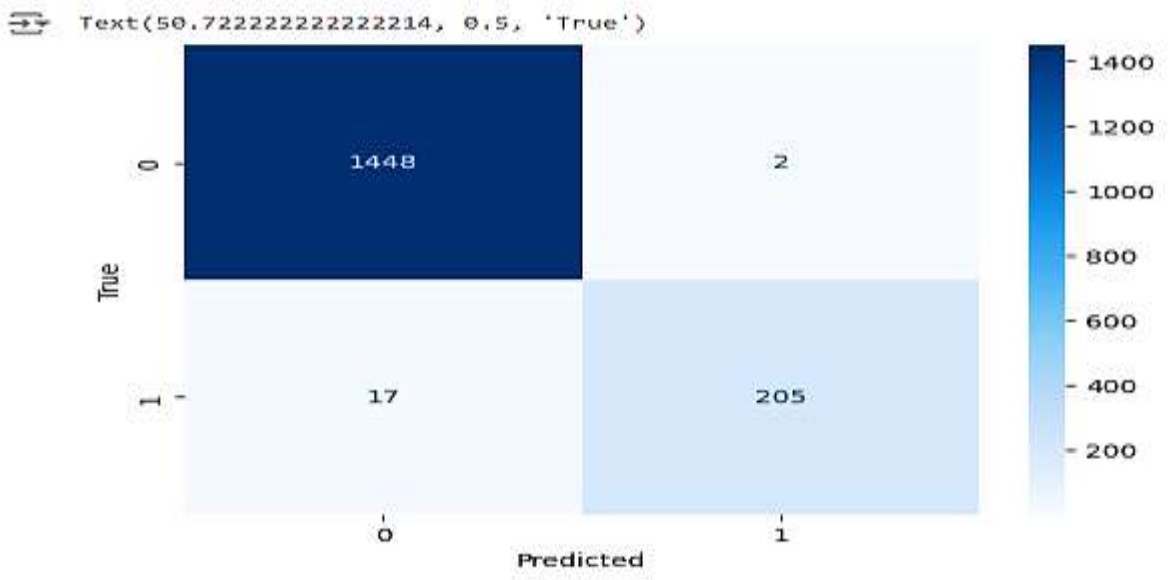

Figure 13
Confusion Matrix showing the Results of CNN

Figure 13 shows that the model successfully recognised 205 out of 222 spam texts (True Positives) based on the data presented in the confusion matrix. 17 texts containing spams were mistakenly classified as non-spam by the model (False Negatives). The model properly recognised 1448 out of 1450 non-spam texts (True Negatives) for non-spam texts. In addition, the model mislabeled 2 non-spam as (false positives). When recognising texts that exhibit spam or non-spam, the model exhibits a high degree of accuracy. The model performs well overall, as seen by its low rate of false positives (2 instances) and false negatives (17 cases).

4.7 Model Output

The output generated from the model. summary() function gives the user information on the design of a sequential neural network model. The fact is that the particular model has three dense layers and increasing the denses' number enhances complexity and capacity. The total parameter of the present model is 1,446,502 and all the parameters are trainable.

```
[ ]  model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 500)               1195000

 dense_1 (Dense)             (None, 500)               250500

 dense_2 (Dense)             (None, 2)                 1002

=================================================================
Total params: 1446502 (5.52 MB)
Trainable params: 1446502 (5.52 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 14
Model Summary

Figure 14 outlines the model summary which shows the analysis of the model, based on the different type of layers used, the arrangement made on those layers, the shape of their output and the number of parameters used.

4.8 Discussion

The way the major obstacles in creating a smishing detection system—from text preprocessing to feature extraction and model development—were methodically addressed is highlighted in the explanation of the individual objective outcomes.

4.8.1 To Examine Existing Techniques for Detecting Smishing Messages in Mobile Phones

The objective of examining existing techniques for detecting deceptive text messages on mobile phones is critical in light of the increasing sophistication of these attacks, as highlighted in the empirical literature. Researchers have explored various methodologies to enhance detection capabilities, underscoring the need for robust systems. For example, Goel and Jain (2018) emphasizes the development of tools tailored to identify SMS phishing threats, illustrating a proactive approach to mobile security(Goel & Jain, 2018).

The discussion on various techniques of detection brought forward the need for a layered approach towards fighting Smishing. URL validation combined with machine learning algorithms and user behaviour analysis showed a broader coverage in the usage of techniques to detect Smishing effectively. The real-time analysis and user education, according to the study, are those two important elements that need to be considered for a major reduction in Smishing risk. Therefore, the findings of the research cumulatively provide a foreground where embedding technologically advanced preprocessing techniques, innovative model development techniques, and deeper understanding of detection methods are essential in effectively countering the emerging menace of Smishing in mobile communications.

In summary, existing techniques for smishing detection show promise, but there is still work to be done in improving real-time capabilities, reducing false positives, and adapting to evolving attack methods.

4.8.2 To Implement an Improved CNN Based Model for Smishing Messages Detection

The enhanced CNN-based model successfully captured all patterns for smishing detection by utilising the intrinsic structure of text data. By adjusting the hyperparameters, experimenting with several CNN architectures, adding more feature types, and adding the number of hidden layers the model's performance was increased. According to the results of the study, by optimizing the CNN based model with additional hidden layers, it achieved a remarkable 99.95% accuracy. The following table 2 presents a comparative analysis of various smishing detection models from recent literature, highlighting their methodologies, datasets, and achieved accuracies:

Table 2
A Comparative Analysis of various Smishing Detection models from Recent Literature

Model	Methodology	Dataset	Accuracy	Reference
CNN	Content based approach	T.A Almeida	99.95%	Rose et al
CNN-BiGRU	Hybrid Deep Learning (CNN + BiGRU)	T.A. Almeida	99.82%	(Mahmud et al., 2024)
CNN-LSTM	Hybrid Deep Learning (CNN + LSTM)	T.A. Almeida	99.44%	(Roy et al., 2020)
DsmishSMS	Backpropagation Algorithm	T.A. Almeida & Pinterest.com	97.93%	(Mishra & Soni, 2023)
Smishing Detector	Neural Network	Real-time smishing messages, Almeida, Pinterest.com	97.40%	(Mishra & Soni, 2022)
Hybrid Machine Learning Model	Heuristic & Content-based Feature Extraction	Not specified	96.00%	(Mahmood & Hameed, 2023)
Random Forest (RF)	Supervised Machine Learning	Not specified	90.10%	(Mahmood & Hameed, 2023)
CNN-LSTM	Deep Learning	Various datasets	99.74%	(Mehmood et al., 2024)

According to the comparative analysis of various smishing detection models from recent literature in table 2, CNN-based models have outperformed conventional machine learning algorithms and deep learning algorithms in the identification of SMS spam and smishing. When it comes to distinguishing between smishing and non-smishing messages, CNN-based models demonstrated excellent accuracy. Further improvements in smishing detection model performance was achieved by combining CNN with additional deep learning methods like Long Short-Term Memory (LSTM). Notably, effective text preparation is essential for enhancing the CNN-based model's performance. This includes feature engineering, tokenization, and data cleaning. The model's performance can be improved through hyperparameter tuning, which involves adjusting the number of convolutional filters, filter dimensions, and network depth.

4.8.3 To Test the Improved CNN Based Model for Smishing Messages Detection

Testing the improved CNN-based model for Smishing message detection aligns with the findings from empirical literature that have established the potency of the model to further improve security against such an attack. For instance, it was established by Roy et al. (2020) that CNN can be effective in classifying between spam and legitimate messages with a high accuracy rate of 99.44%(Roy et al., 2020). This shows that automatically, the salient features will be detected by the model itself, and less human intervention is required; hence, this is a strong improvement from previous methods.

Amin et al. (2019) describe a conceptual framework where several factors interplay in influencing the test outcome, such as model architecture and preprocessing. From their insights, it can be concluded that architecture-based robust performance in Smishing detection can be achieved in CNNs together with efficient handling of data(Amin & Nadeem, 2019). These findings offer a thorough summary of how well CNN-based models identify smishing messages. The primary results indicated that CNNs are a good option for smishing message detection since they are well-suited for text classification tasks. The suggested CNN-based models detected smishing messages with great recall, accuracy, precision, and F1-score.

V. CONCLUSION & RECOMMENDATIONS

5.1 Conclusion

This study analyzes smishing attacks and evaluates detection methods, focusing on a CNN-based model. Findings show the increasing sophistication of smishing and the need for advanced safeguards. A multi-faceted approach, including real-time analysis and user education, is essential. Text preprocessing significantly enhances detection accuracy, and the CNN model outperforms traditional methods. Testing confirms its high accuracy in detecting smishing messages, emphasizing the role of machine learning in mobile security. The study underscores the need for advanced detection systems and ongoing improvements to combat evolving mobile phishing threat.

5.2 Recommendations

Authorities should secure adequate resources, define roles, and continuously enhance the smishing detection system through new features, data augmentation, and regular assessments. They must also allocate funds for staff training and set benchmarks to measure effectiveness. Beneficiaries should provide feedback, raise awareness, and report suspicious messages to authorities. Other stakeholders should contribute by sharing best practices and supporting

awareness efforts with resources and expertise. Future research should explore advanced deep learning architectures, integrate diverse data sources, and address emerging smishing tactics. Efforts should focus on developing adaptive machine learning algorithms, reducing false positives, and minimizing personal data exposure. Enhancing the system's ability to detect complex smishing attacks will further strengthen user security and mitigate cyber threats.

REFERENCES

Alexander, G. (2024, September 23). *We're a nonprofit text message solution | Tatango [The premier text fundraising service built for nonprofits]*. Tatango—SMS Marketing Software. https://www.tatango.com/blog/text_message_content_gateway/

Almeida, T. A., & Hidalgo, J. M. G. (2011). *SMS Spam Collection (Version V.1) [Dataset, plain text file]*. UCI Machine Learning Repository. https://doi.org/10.24432/C5CC84

Amin, M. Z., & Nadeem, N. (2019, October 6). *Convolutional neural network: Text classification model for open domain question answering system*. arXiv. https://doi.org/10.48550/arXiv.1809.02479

Bhandari, P. (2021, October 18). *Ethical considerations in research | Types & examples*. Scribbr. https://www.scribbr.com/methodology/research-ethics/

Delany, S. J., Buckley, M., & Greene, D. (2012). SMS spam filtering: Methods and data. *Expert Systems with Applications, 39*(10), 9899–9908. https://doi.org/10.1016/j.eswa.2012.02.053

Emerson, R. W. (2015). Convenience sampling, random sampling, and snowball sampling: How does sampling affect the validity of research? *Journal of Visual Impairment & Blindness, 109*(2), 164–168. https://doi.org/10.1177/0145482X1510900215

Erdelyi, L. (2020, March 16). *The five stages of the data analysis process*. Lighthouse Labs. https://www.lighthouselabs.ca/en/blog/the-five-stages-of-data-analysis

Goel, D., & Jain, A. K. (2018). Smishing-classifier: A novel framework for detection of smishing attack in mobile environment. In P. Bhattacharyya, H. G. Sastry, V. Marriboyina, & R. Sharma (Eds.), *Smart and innovative trends in next generation computing technologies* (pp. 502–512). Springer. https://doi.org/10.1007/978-981-10-8660-1_38

Goel, D., Ahmad, H., Jain, A. K., & Goel, N. K. (2024, December 9). *Machine learning driven smishing detection framework for mobile security*. arXiv. https://doi.org/10.48550/arXiv.2412.09641

Gomaa, W. H. (2020). The impact of deep learning techniques on SMS spam filtering. *International Journal of Advanced Computer Science and Applications, 11*(1), 544–549. https://doi.org/10.14569/IJACSA.2020.0110167

Jain, A. K., & Gupta, B. B. (2019). Feature based approach for detection of smishing messages in the mobile environment. *Journal of Information Technology Research, 12*(2), 17–35. https://doi.org/10.4018/JITR.2019040102

Jain, A. K., Goel, D., Agarwal, S., Singh, Y., & Bajaj, G. (2020). Predicting spam messages using back propagation neural network. *Wireless Personal Communications, 110*(1), 403–422. https://doi.org/10.1007/s11277-019-06734-y

Jain, A. K., Gupta, B. B., Kaur, K., Bhutani, P., Alhalabi, W., & Almomani, A. (2022). A content and URL analysis-based efficient approach to detect smishing SMS in intelligent systems. *International Journal of Intelligent Systems, 37*(12), 11117–11141. https://doi.org/10.1002/int.23035

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444. https://doi.org/10.1038/nature14539

Mahmood, A. R., & Hameed, S. M. (2023). Review of smishing detection via machine learning. *Iraqi Journal of Science, 64*(8), 4244–4259. https://doi.org/10.24996/ijs.2023.64.8.42

Mahmud, T., Prince, M. A. H., Ali, M. H., Hossain, M. S., & Andersson, K. (2024). Enhancing cybersecurity: Hybrid deep learning approaches to smishing attack detection. *Systems, 12*(11), 490. https://doi.org/10.3390/systems12110490

Maqsood, U., Ur Rehman, S., Ali, T., Mahmood, K., Alsaedi, T., & Kundi, M. (2023). An intelligent framework based on deep learning for SMS and e-mail spam detection. *Applied Computational Intelligence and Soft Computing, 2023*, 1–16. https://doi.org/10.1155/2023/6648970

Mehmood, M. K., Arshad, H., Alawida, M., & Mehmood, A. (2024). Enhancing smishing detection: A deep learning approach for improved accuracy and reduced false positives. *IEEE Access, 12*, 137176–137193. https://doi.org/10.1109/ACCESS.2024.3463871

Mishra, S., & Soni, D. (2019a). A content-based approach for detecting smishing in mobile environment. *SSRN Electronic Journal*, 986–993. https://doi.org/10.2139/ssrn.3356256

Mishra, S., & Soni, D. (2019b). SMS phishing and mitigation approaches. In *2019 Twelfth International Conference on Contemporary Computing (IC3)* (pp. 1–5). IEEE. https://doi.org/10.1109/IC3.2019.8844920

Mishra, S., & Soni, D. (2020). Smishing detector: A security model to detect smishing through SMS content analysis and URL behavior analysis. *Future Generation Computer Systems, 108*, 803–815. https://doi.org/10.1016/j.future.2020.03.021

Mishra, S., & Soni, D. (2022). Implementation of 'Smishing Detector': An efficient model for smishing detection using neural network. *SN Computer Science, 3*(3), 189. https://doi.org/10.1007/s42979-022-01078-0

Mishra, S., & Soni, D. (2023). DSmishSMS—A system to detect smishing SMS. *Neural Computing and Applications, 35*(7), 4975–4992. https://doi.org/10.1007/s00521-021-06305-y

Morreale, M. (2017, March 6). *Daily SMS mobile usage statistics | SMSEagle*. SMSEagle. https://www.smseagle.eu/2017/03/06/daily-sms-mobile-statistics/

Nivaashini, M., R.S. Soundariya, A. Kodieswari, & P. Thangaraj. (2018). SMS spam detection using deep neural network. *International Journal of Pure and Applied Mathematics, 119*(18), 2425–2436.

Remmide, M. A., Boumahdi, F., Ilhem, B., & Boustia, N. (2025). A privacy-preserving approach for detecting smishing attacks using federated deep learning. *International Journal of Information Technology, 17*(1), 547–553. https://doi.org/10.1007/s41870-024-02144-x

Roy, P. K., Singh, J. P., & Banerjee, S. (2020). Deep learning to filter SMS spam. *Future Generation Computer Systems, 102*, 524–533. https://doi.org/10.1016/j.future.2019.09.001

Sheikhi, S., Kheirabadi, M. T., & Bazzazi, A. (2020). An effective model for SMS spam detection using content-based features and averaged neural network. *International Journal of Engineering, 33*(2), 221–228. https://doi.org/10.5829/ije.2020.33.02b.06

Shweta, & Main, K. (2023, July 17). *What is smishing? Definition, examples & protection*. Forbes Advisor. https://www.forbes.com/advisor/business/what-is-smishing/

Tanbhir, G., Shahriyar, M. F., Shahed, K., Chy, A. M. R., & Adnan, M. A. (2025, February 3). *Hybrid machine learning model for detecting Bangla smishing text using BERT and character-level CNN*. arXiv. https://doi.org/10.48550/arXiv.2502.01518

Testas, A. (2023). *Distributed machine learning with PySpark: Migrating effortlessly from pandas and Scikit-learn* (1st ed.). Apress. https://doi.org/10.1007/978-1-4842-9751-3