Enhancing SMS Spam Detection using Deep Learning Techniques

MSc Research Project
MSc Data Analytics

Tamil Selvan Giri Moorthy
Student ID: x23189975

School of Computing
National College of Ireland

Supervisor: Mr. Bharat Agarwal

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Tamil Selvan Giri Moorthy		
Student ID:	X23189975		
Programme:	MSc Data Analytics	**Year:**	2024
Module:	MSc Research Project		
Supervisor:	Bharat Agarwal		
Submission Due Date:	12/12/2024		
Project Title:	Enhancing SMS Spam Detection using Deep Learning Techniques		
Word Count:	7711		
Page Count	20		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: TAMIL SELVAN GIRI MOORTHY

Date: 12th December 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	☑
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☑
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☑

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Enhancing SMS Spam Detection using Deep Learning Techniques

Tamil Selvan Giri Moorthy
x23189975

Abstract

SMS spam is a major problem in mobile communication which will cause issues like financial loss and privacy violations. SMS spam detection is used to identify unwanted messages, protecting users from scams. The primary goal of SMS spam detection is to identify the difference between legitimate (ham) and unwanted messages(spam). Many models have been used to detect spam, but due to advancements in spam techniques we needed better detection methods to identify the advanced spam messages. Traditional machine learning models like Random Forest and deep learning models such as RNN, LSTM, BI-LSTM, and GRU have worked well in identifying spam messages, but it fails to understand the deeper meaning of the message. This research focuses on identifying these issues by using advanced Transformer models like Bert to compare whether these models will perform better than the traditional methods and other deep learning models. This research is very important because due to the rise in technology development spam messages are more sophisticated to identify, so we need powerful and accurate detection models to detect them. People use their mobile a lot for communication, transferring information and protecting them from unwanted messages will improve their security and user experience. Transformer models like Bert are good in understanding the deeper meaning of the word, so this research identify whether these models can expect to do a better performance compared to other methods in detecting SMS Spam.

1 Introduction

Spam messages, also known as unwanted or irrelevant texts, are a major issue in mobile communication, causing problems in financial scams and privacy violations. In this modern world mobile devices plays a vital role in communication and information exchange, so it is very important to have system to understand and identify the spam messages to stop these unwanted messages from the users to keep their information safe. Over the years, many techniques have been developed to detect the spam messages. Traditional machine learning models like, Random Forest and deep learning models such as RNN, LSTM, Bi-LSTM, and GRU. These methods are used before to identify the spam messages by understanding the patterns of text, but due to the advancements in spam messages it often changes the pattern of these messages, so these models struggle to detect more advanced spam messages

The main issue in these models is that they don't understand the deeper meaning and context of words in a message. For an instance, some spam messages use tricky language or hidden pattern in which it is difficult for these models to identify the context of message. These limitations reduce their effectiveness in identifying cleverly crafted spam. To overcome these challenges, advanced transformer models like BERT will give a better result in understanding sophisticated patterns and messages. BERT model is good at understanding the meaning of words based on the words around them. It will look the entire sentence to understand how words are related to each other. By reading the entire sentence it will identify the true meaning of the message even if the text is tricky or complicated.

This research focuses on using BERT to improve spam detection. It will compare BERT's performance with older models to see if it does a better performance at identifying spam. By filling this gap in current research, the study aims to offer a stronger and more reliable solution providing new knowledge that could greatly improve spam detection system.

The rise in technology has made spam messages more sophisticated and giving many challenges to detect spam messages with traditional methods. This research is crucial because it focuses on developing more accurate detection models to address this growing challenge. The immense rise of technology almost 80% percentage of peoples are using mobile devices, so stopping unwanted messages can help the users to their privacy information safe.

Research question and Objectives:
Research Question: Can Transformers models like BERT outperform traditional machine learning and deep learning model in detecting SMS spam.
To analyse the performance of traditional machine learning models like Random Forest in spam detection. To evaluate the effectiveness of deep learning models such as RNN, LSTM, Bi-LSTM, and GRU for identifying spam messages. To assess the performance of Bert in detecting SMS spam and compare it with traditional and deep learning models. First, we need to understand the performance of each model. By go thorouging all this model performance we can determine which model is performing better.

Limitations:
This research has some challenges. First, the data used to train the models might have some biases which could affect the results. Also, the findings may not work well for every type of spam messages since spam can vary a lot. Another challenge is that advanced model like BERT need a lot of computer storage power to work, which might make it hard to use them in places with limited resources

Assumption:
The data used for training includes many different types of spam and regular messages, so it will give a good outcome. The result seen when testing BERT will work well on other similar data, not just the specific dataset used in the research.

Structure of the report:

1. **Introduction:** Provides the background, importance, research question and objectives.
2. **Literature Review:** In this section we will discuss about previous research on SMS spam detection and identifies the gaps in the field.
3. **Methodology:** Outlines the datasets, preprocessing steps, and models used including traditional machine learning, deep learning and Transformer models like BERT.

4. **Results and Discussion:** Presents and compares the performance of different models, highlighting the key findings.
5. **Conclusion:** Summarizes the research, discusses implications, and provides suggestion for future work

2 Related Work

2.1 Understanding the classification of Ham and spam messages

In this section we will discuss what we learned from the previous research paper to achieve our research question. First, we learnt about the basics of ham and spam messages. Saab et al. (2024), the author clearly explains ham and spam messages. This paper explains what kind of messages will lead to cause a spam message such as information overload users will receive too many irrelevant messages. Loss of productivity it takes time to sort out useful messages. Then security risks spam messages may contain malware or links to phishing websites. The we learned about what is ham message what type of information it will contain in the message. By analysing these papers, we learned about what is the difference between ham and spam message.

2.2 Understanding the related work in SMS spam Detection

In this section we will discuss the traditional methods which are used to classify the messages. By understanding this paper by Dada et al., 2019, we learned the rule-based filtering in which expert manually create rules to identify spam. For an instance message containing keywords were flagged as spam. In this paper the author explains the drawbacks of these models with lack of adaptability and high false positives. Then we learned about keyword-based filtering in which messages are scanned for specific words or phrases commonly found in spam. For instance, a message which words like ‘buy now’ or ‘exclusive offer’ might be flagged as spam. But these cannot adapt to new or disguised spam messages. Then false positives will legitimate the messages which contain flagged keywords are often misclassified. By reading this paper we learned about how traditional models analyze the text to classify the message.

The field of SMS spam detection has been explored extensively with several approaches using machine learning and deep learning techniques. One such approach is the comparative study by Gandhi et al. (2024) which evaluates the performance of deep learning models such as Long Short-Term Memory (LSTM) and Bidirectional LSTM (Bi-LSTM) in SMS spam detection. The authors found that LSTM and Bi-LSTM models demonstrated a promising performance by achieving an accuracy of 92.98% and 93.98% respectively. The study highlighted the importance of model architecture in handling sequential data. However, one of the limitations of this study was its reliance on a relatively small dataset and the absence of hyperparameter optimization which could improve performance. Similarly, the Venkata Kalyani et al. (2024) study utilized a dataset which contains 5,570 samples and focused on handling class imbalance using oversampling techniques like ADASYN. The results show that hybrid model combining Bi-LSTM and GRU achieved an accuracy of 99% which outperformed the individual models. The paper emphasizes the potential of deep learning architecture in addressing class imbalance, but the study does not explore the computational complexity of these models, which may limit their scalability for large datasets. Additionally, Sarangi et al. (2024) performed a comparative analysis of various deep learning models, specifically focusing on SMS spam detection. Their work compared the performance of deep learning models such as Bi-LSTM and GRU, highlighting the superior performance of hybrid architectures. While the accuracy rates were impressive, the authors did not provide an in-depth analysis of how their models handled varying data distributions or noise, which are critical factors in real- world applications. Now we will discuss about the Bert, Manish et al. (2019) performed a sentiment classification using BERT. The authors used BERT, a state-of-the-art model for natural language processing, to classify sentiments more accurately. Their experiments showed that BERT performs better than many other popular models for this task, even without using complex model designs. This paper suggests that BERT is an excellent choice for doing advanced text classification tasks. Raga and BL (2024) proposed a fine-tuned BERT model for SMS spam detection which outperformed traditional machine learning models like Naïve Bayes and Random Forest. The study showed an AUC of 96.10% and F1 score of 92% emphasizing BERT's ability to capture the contextual meaning of messages. While BERT demonstrated superior performance, the study did not discuss the computational burden of fine-tuning BERT, which can be resource-intensive and may not be suitable for real-time spam detection in resource-constrained environments.

2.3 Summary and Justification for research

The reviewed literature highlights the effectiveness of machine learning and deep learning approaches in SMS and email spam detection, with notable success from models such as Bi-LSTM, GRU and BERT. While these models offer significant accuracy improvements, challenges remain regarding class imbalance, scalability, and computational cost. Traditional machine learning methods like Random Forest and SVM are effective for smaller datasets, and it faces limitations in handling the complexities of evolving spam techniques. Given these finding, there is clear gap in the literature regarding the use of hybrid model that combine content-based and collaborative filtering approaches to enhance spam detection. Moreover, the computational limitations of fine-tuning deep learning models like BERT for real time applications need further exploration. Therefore, this research aims to address the gaps by leveraging a hybrid approach that incorporates both BERT-based content embeddings and collaborative filtering signals, with goal of creating a more efficient and scalable spam detection system.

3 Research Methodology

In this project, we followed a structured methodology to detect SMS spam messages effectively by using our traditional model approaches. The first step involves a collecting a dataset of 5,570 SMS messages from Kaggle. These messages were classified as spam or ham, so it will give a foundation for training and evaluating our models. This dataset contains more legitimate messages than spam messages which provides an imbalanced dataset. To balance this dataset, we used the ADASYN oversampling technique to create a balanced representation. By conducting this step, we can check that the models will equally learn both spam and non- spam messages. After balancing the dataset, we pre-processed the data to make the dataset clean and organizes the raw text data to make easier for the model to analyse and understand the text data. Then we used TF-IDF (Term Frequency- Inverse Document Frequency) Sarangi et al. (2024) to convert text into numbers to help the model focus on the important terms to identify the message is spam or not and to predict the results more accurately. Then we applied our learning models to determine the results. The following process of this analysis are explained below in detail.

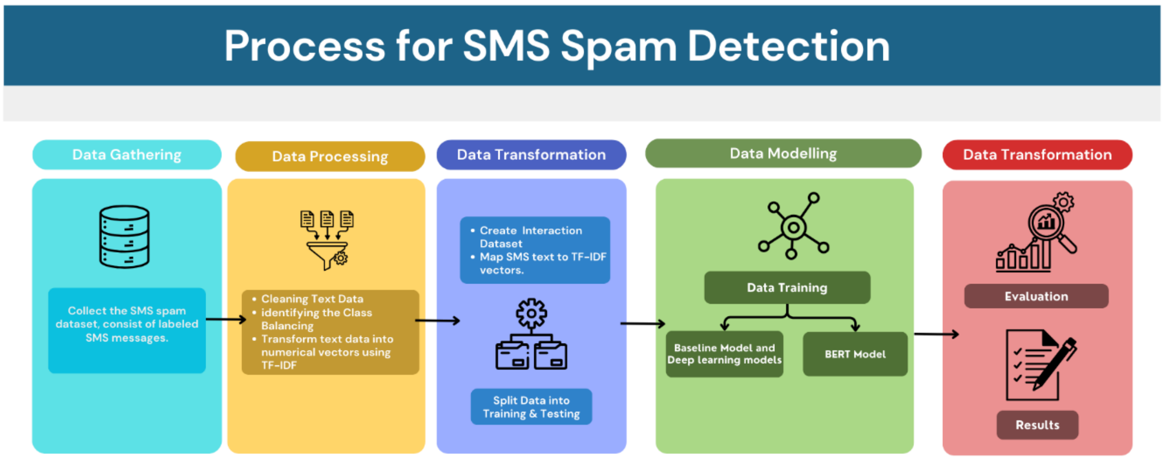

Process for SMS Spam Detection
Data Gathering
Data Processing
Data Transformation
Data Modelling
Data Transformation
Collect the SMS spam dataset, consist of labeled SMS messages.
• Cleaning Text Data
• identifying the Class Balancing
• Transform text data into numerical vectors using TF-IDF
• Create Interaction Dataset
• Map SMS text to TF-IDF vectors.
Split Data into Training & Testing
Data Training
Baseline Model and Deep learning models
BERT Model
Evaluation
Results

Figure 1: SMS Spam Detection workflow

3.1 Dataset Description

In this analysis, we used dataset which is specially designed for SMS spam detection we gathered this dataset from Kaggle website. This dataset contains a total of 5,570 SMS messages which provides a reliable source for analysing and building a spam detection system. Each message is labelled as either spam or non-spam. Spam messages which contain a message that are unwanted, promotional, or fraudulent. It often sent the bulk messages to many people at same time. The non-spam messages which contain legitimate, and it will include regular conversations, personal texts or useful notifications

Dataset Description		
SL.NO	**Label**	**Text category**
1	spam	It describes the message, which is irrelevant, promotional or fraudulent.
2	ham	It describes the message, which is legitimate, regular conversations or useful notifications.

Figure 2: describes the columns in the dataset

3.2 Data Preprocessing

Once we collected our dataset, we conducted a data preprocessing analysis. Firstly, we need to understand the dataset shape and how many columns and data entries are present in the dataset. Once we identified the data entries. Then we checked the number of null values and duplicates values presented in the dataset. We found there are 401 duplicate values present in the dataset. We dropped the duplicate values present in the dataset. After completing these steps, we cleaned the text data.

3.2.1 Cleaning Text Data

By cleaning the unnecessary elements, we can check the data is easier for the model to analyse and learn the text. The following steps will conclude how we clean text data.

- **Remove unnecessary characters:** In this step we removed the unnecessary characters from the text data like exclamation marks and special characters such as @, #, $ and %.

These symbols will not give any meaning to the text and make it difficult for the model to understand the meaning of the words in-depth. By removing this special character from the text data, it will make the message simple and easier for the model to process the words.

- **Convert text to lowercase:** we converted all text in a data to lowercase to check the consistency of the word. For an instance there are two words like "task" and "TASK" both the words will give the same meaning so it should be treated as same words, but casing in both words is different sometimes the model will mistakenly consider them as different words. To stop these mistakes, we lowercase the data and check uniformity across all messages.

- **Tokenization:** In this process we break the text into smaller parts and store it as a token. If we have a sentence in our dataset, it will break the sentence, separate all the words from a sentence and store it as a token. By doing this process it will help the model to focus on individual words and their pattern instead of looking into a whole sentence. This step is very important to analyse the structure and meaning of the text.

These above steps will ensure that the data is in simple, understandable, and organized form making it easier for the learning model to identify the pattern and relationships between words.

3.2.2 Class Balancing

After cleaning our text data, we need to build our model to detect spam messages. When building the model, one major problem will come which is data imbalance. This means there are more non-spam legitimate messages present in the dataset compared to spam messages. If we didn't fix this imbalance, the model might focus too much on the majority class, and it will ignore the minority class. This issue will make the model fail to detect the spam message. The below figure 3 displays the number of ham and spam messages present in the dataset. We can clearly see that the dataset is imbalanced because 90% of messages are spam. So, we need to balance the dataset.

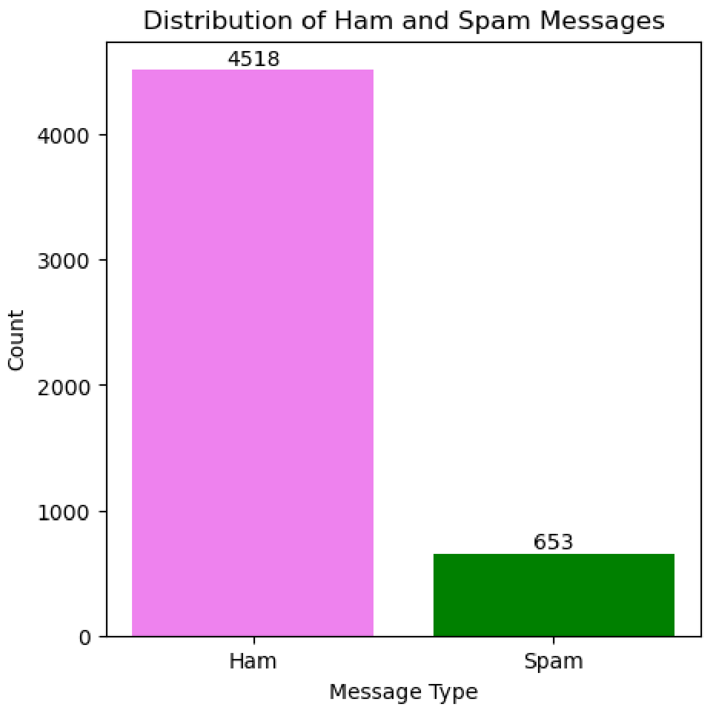


Figure 3: displaying the imbalance between the messages

To balance our dataset, we used Adasyn (Adaptive Synthetic Sampling) technique. It will analyse the smaller group of messages in our dataset which is spam messages, and it will add the required amount of new spam examples to check both the spam and non- spam messages are equally balanced. By balancing the dataset, the model will identify better in spam messages even if the messages are hidden among a lot of non-spam messages. Now we ensure the model is trained in a fair mix of spam and non-spam messages in which it will help to detect spam messages more effectively.

3.2.3 Feature Representation

After balancing our dataset, now we need to convert the words in text messages to numerical format because the learning model will understand only the numerical values. To do this, first we need to split the messages into smaller parts and store them as tokens. A token is a single word which will store a character or a phrase.

In machine learning, this tokenization process breaks the text into words and counts how often each word has appeared. These counts are helpful for doing calculations in TF (Term Frequency) – IDF (Inverse Document Frequency). This IDF will reduce the words which have appeared in all the messages in the dataset. These TF and IDF combine and calculate a final score for each word in a message. In deep learning these tokenization processes will not depend on basic word counts. First, it will break the text into tokens and then assign numerical embeddings to represent the deeper meaning of the words.

For deep learning models tokenization and embedding will help the model to understand the relationships between the words and it will improve their ability to detect the unwanted spam indicators. By understanding the text as meaningful numbers, both approaches allow the model to understand and classify messages accurately.

3.3 Data Transformation

When we are working with text classification dataset, data transformation is essential to make the text into understandable structure for machine learning and deep learning models. This can be achieved by creating an interactive dataset and splitting the data. In this step we will explore how these processes were differed for traditional learning models, deep learning models and Bert.

Creating interactive dataset:
In traditional learning models we used TF-IDF vectorizer to transform each SMS message into a vector of numbers. Each vector represents the message numerically and captures the importance of specific spam- related words. Finally, it will combine these vectors with their corresponding labels (1 for spam, 0 for non-spam). In deep learning models will require the text to be tokenized and converted into sequences of number that will keep the order of words and context. First it will convert each SMS message into sequence of word tokens then it will assign each token into a unique numerical ID or embedding. If the SMS messages are varied in length, pad shorter messages to a mixed length to check the consistent input to the model. For Bert Model, it will work with pre-trained embeddings and special input formats. We used Bert tokenizer to split text into smaller chunks. Then we converted each token to unique ID from Bert and created an attention mask to understand which tokens are actual words and which are padding. By following these steps, we created an interactive dataset for our analysis.

Split Dataset:
Once the dataset is we need to divide it into training and testing sets to evaluate the model properly. The data is divided into train and test. To train our model we used 80% of the data. To test our model, we used 20% of the data to evaluate the model performance. In deep learning model we used validation to monitor the model's performance during training. For Bert model, since it is a pre-trained on large corpus, the focus is to fine-tune the dataset. We split the data into train, test and validation. It requires validation during fine-tuning to avoid overfitting on the training data.

3.4 Data Modelling

In this process we build the model development phase which involves training different models to classify messages which is ham and spam. It starts with a baseline model to set a performance benchmark and progresses to more advanced models for better performance. The below image gives a step in simple to understand the model for our analysis. The figure displays the proposed model for our analysis.

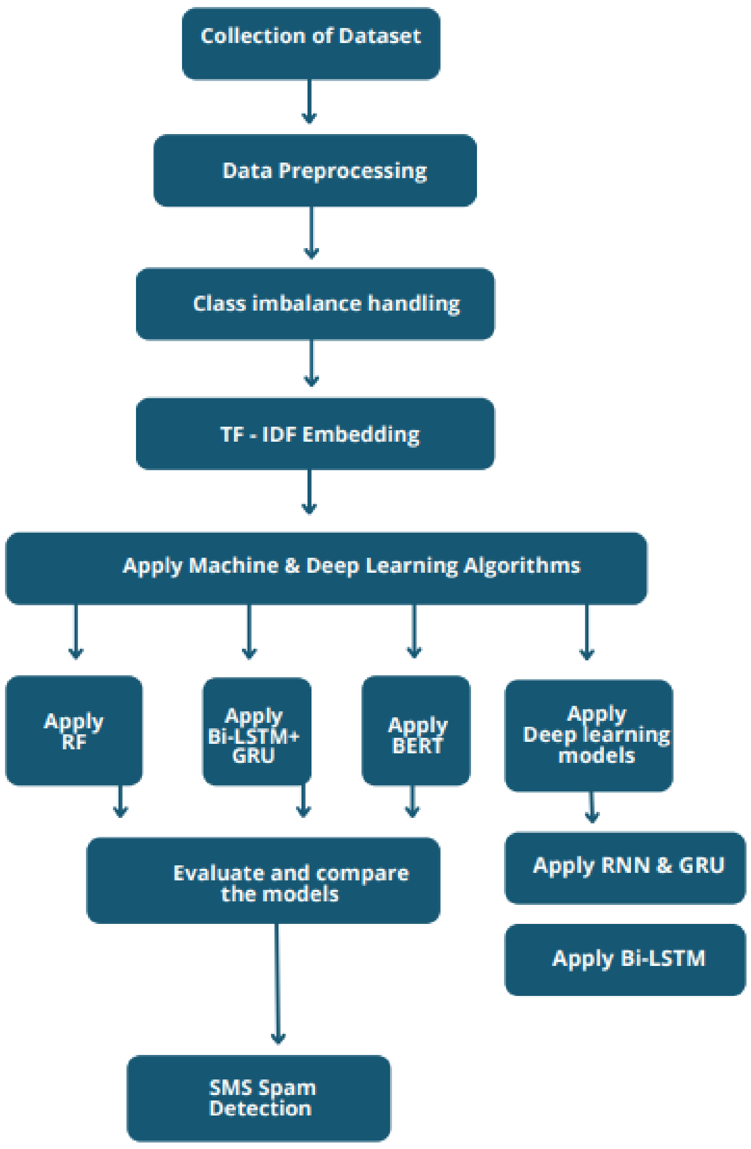


Figure 4: Proposed model for our analysis

3.5 Data Evaluation

The final stage in building an SMS spam detection system is evaluating how well the model is performing. This step will help us to understand which model works best and we will describe why it is giving better performance compared to other models. It involves checking the performance of the models using specific metrics and analysing the results to draw meaningful conclusions.

4 Design Specification

In this section, we discuss the in-depth explanation of the design specifications, technique, tools and framework used in our spam detection analysis. It outlines methodologies implemented, the underlying requirements, and how these were integrated to achieve the goal of classifying the messages accurately.

Overview of the Approach:
In this spam detection project, we used three main approaches: Machine learning (ML), Deep learning (DL) and BERT which is a powerful tool used to understand the context of words effectively. Machine learning is started with simple models to create a starting point of the model to measure the result is good or bad these baseline models are fast and easy to train. In deep learning models we used more advanced techniques to improve the accuracy of spam detection by teaching the model to understand the complex patterns in the text. Finally, we used BERT, which is one the most modern and effective tools for understanding the meaning and context of words in sentences. BERT is especially good at understanding the relationship between the words, which makes it excellent for text analysis tasks like spam detection.

4.1 Techniques and Frameworks Used:

1. **Machine Learning Techniques:**
   We began our analysis with traditional machine learning models because it is very simple and fast by providing a baseline for performance comparison.

   Algorithms Used:
   We selected Random Forest (RF) because it works by combining many smaller decision trees to make accurate predictions. RF can easily handle numerical data, which is very important when working with transformed text data.

   Feature Representation:
   Venkata Kalyani et al. (2024) In traditional models we cannot process the text directly, so we used TF-IDF to convert text into numerical data. Term Frequency (TF), it measures the word which is most commonly present in all the messages. Words that show up more frequently are considered important within that message. Inverse Document Frequency (IDF) it measures the words which are uniquely present in all the messages. By combining TF and IDF we created a numerical vector that highlights words important for spam detection.

Implementation:

Once we converted the text into TF-IDF vector, we can process this numerical data into the Random Forest classifier. The model will analyse these patterns to learn, and it will differentiate the messages between ham and spam.

2. **Deep Learning Techniques:**

   After we finished the baseline models, we moved on to deep learning models to capture more complex relationships within the text. These models are designed to identify complex patterns.

   Architecture Used:

   The architecture used in this project is inspired by Gandhi et al. (2024).

   1. **Simple RNN (Recurrent Neural Network):**
      This model will read the text one word at a time, considering the order of words. It helps capture dependencies between words, but it can struggle when it is analysing longer messages.

   2. **LSTM (Long Short-Term Memory):**
      LSTM improves RNNs by remembering information from earlier in the message. Since it is very good in identifying the earlier words, it is useful for processing longer texts.

   3. **BI-LSTM (Bidirectional LSTM):**
      Bi-LSTM processes the text into two directions forward from start to end and backward from end to start. By reading the model in two directions it will give the model more context to understand the meaning of words based on their surroundings.

   4. **GRU (Gated Recurrent Unit):**
      GRU works like LSTM but is faster and simpler, it is very efficient to train the model particularly when we are working on the large dataset.

   5. **Hybrid Bi-LSTM + GRU:**
      To combine the strength of Bi-LSTM (it will understand the deep context of words) and GRU (it processes them quickly and efficiently) we build a hybrid model. This model balances accuracy with computational efficiency.

Data Preprocessing for Deep learning models:

Before training the models, we prepared the text by:

- **Cleaning:** Removing the unnecessary punctuation and words which don't add give any meaning to the sentence.
- **Tokenizing:** In this process we break the sentence into token to provide an easy process and we converted the tokens into numerical formats using embedding layers in which it captures the meaning of each word into numerical form.

Implementation:

- Each deep learning model was trained on the pre-processed data. During training, the models learned to identify patterns in spam messages (e.g. common phrases like ‘win a prize’ or ‘limited offer’) and non-spam messages (e.g. conversational texts like ‘meeting at 3pm’).
- We evaluated the models based on their accuracy, precision, and recall which measure how well the models are classifying the spam and non-spam messages.

3. **BERT**

Finally, we used Bert, which is a highly advanced tool for understanding text. It takes spam detection to the next level by capturing the meaning and relationship between all the words in a sentence.

Why we use BERT?

The traditional model will read the text one word at a time. But in BERT it looks all the words in a sentence at once both forwards and backward. This allows it to understand the full context of each word.

Pre-trained knowledge: Bert has already been trained on massive datasets so it will understand a lot about language patterns. This will save our time and improve our performance.

Context Understanding: It is excellent at understanding complex sentences, including sarcasm or ambiguous phrases that traditional models might failed to identify.

Implementation:

1. **Tokenization:**
   We used Bert’s tokenizer, which breaks the text into smaller pieces of words to ensure that every part of the message is identified by BERT model. For an instance ‘english’ might be broken into ‘eng’ and ‘lish’.

2. **Training the model:**
   We fine-tuned the pre-trained BERT model on our spam dataset. This means we adjusted BERT to focus specifically on spam detection. During training, this model will learn the unique patterns of spam messages (e.g. promotional language or phrases like ‘click here’) and how they are differing from non-spam messages.

3. **Classification:**
   Bert generated contextual embeddings (numerical representations of words based on their meaning and context). These embeddings were used to classify the spam and non-spam messages.

4.3 Implementation Required:

1. Hardware specifications, dependent libraries and tools Used.

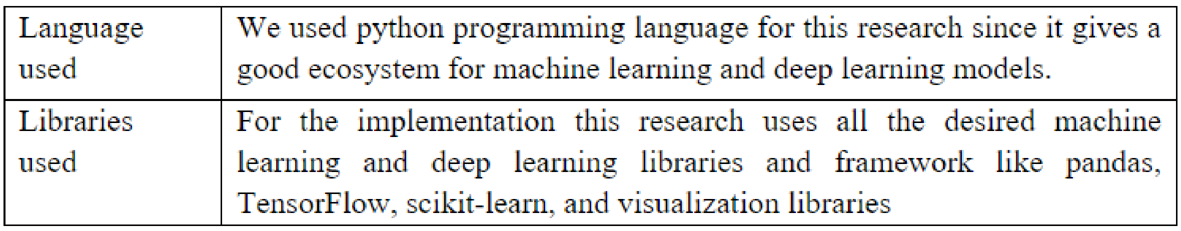
For coding and testing, we used Jupyter Notebook and Google Colab.

Jupyter Notebook:

This tool allowed us to write and test code by step-by-step process. It’s user friendly and help us to visualize the results easily.

Google Colab:

This platform gave us access to run powerful compiler like GPU and TPU which made it faster to train our deep learning and BERT models. By using these tools and technique together, we can ensure that our project is efficient, accurate and well organized. These platforms will help us to handle simple and complex text pattern effectively, by improving our ability to classify messages as spam or not spam.



Language used	We used python programming language for this research since it gives a good ecosystem for machine learning and deep learning models.
Libraries used	For the implementation this research uses all the desired machine learning and deep learning libraries and framework like pandas, TensorFlow, scikit-learn, and visualization libraries

5 Implementation

In this process we explained the final process that we conducted for our analysis.

5.1 Data Transformation and Preprocessing:

Once we collected the dataset, first we focused on understanding the dataset. Then we go thorough the valuable insights of dataset. We pre-processed our dataset by removing the null values and unnecessary details from the dataset. Then we removed the extra characters like punctuation and symbols which doesn’t give the meaningful information. Then we converted all the text to lowercase to keep all the words consistent. Then break the sentence into token to provide an easy process and we converted the tokens into numerical formats using embedding layers in which it captures the meaning of each word into numerical form.

5.2 Exploratory Data Analysis (EDA)

Once we cleaned and transformed our dataset, we conducted an EDA to identify the valuable information of the dataset. In this step we visualized how many emails are spam and how many are non-spam. We identified most the emails are non-spam and only a smaller number of messages are spam. Then we measured the number of characters in emails to find the average length for spam and non-spam emails.

By visualizing this we identified spam messages are longer with an average length of 130 to 180 characters. Non-spam messages are shorter averaging only 70 characters. Then we counted number of words and number of sentences are present in the dataset to find their average words and sentence are present in the dataset. By visualizing this we identified that the spam messages are longer, and it contains more words and sentence when we compared to non-spam messages. Features like length, word count, and sentence count can help to improve our spam detection model. This analysis will give a deeper understanding of the dataset and help us to prepare better features to train our machine and deep learning models.

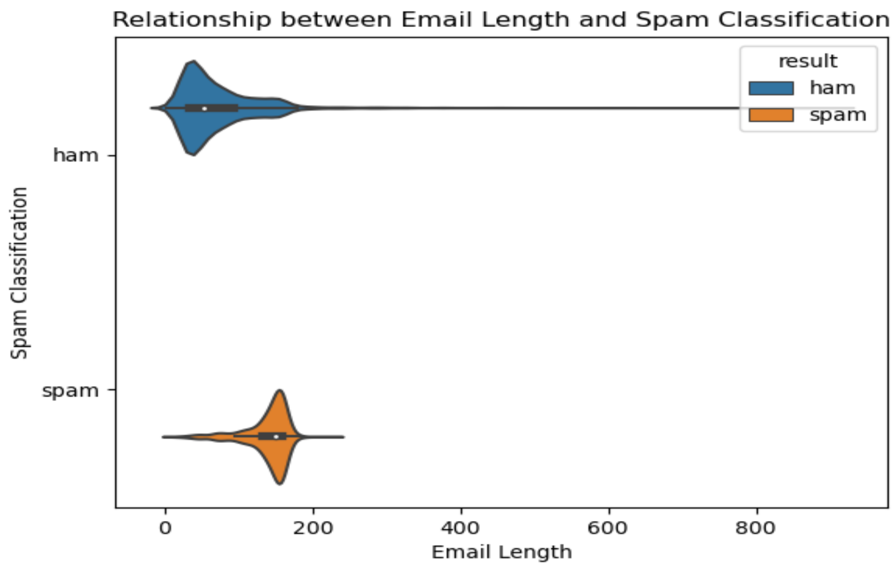


Figure 5: Relationship between Email Length and spam classification

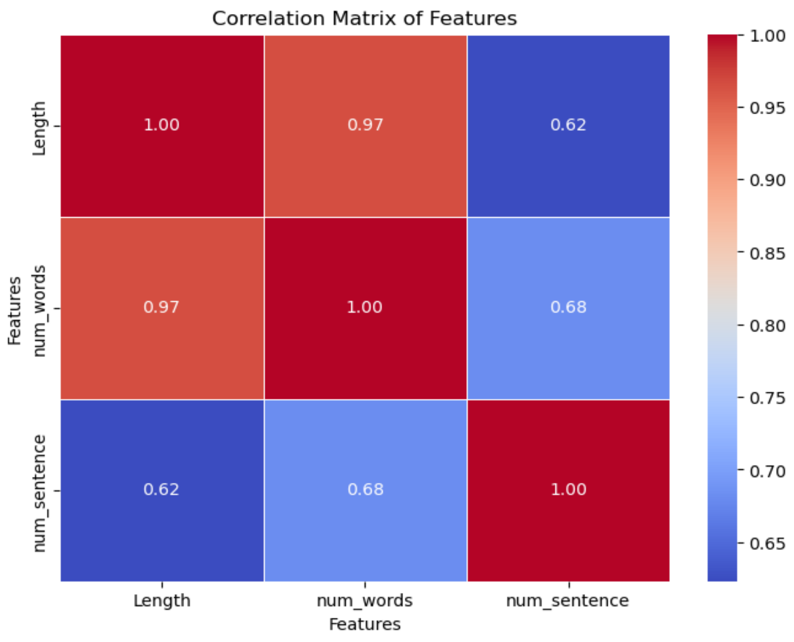
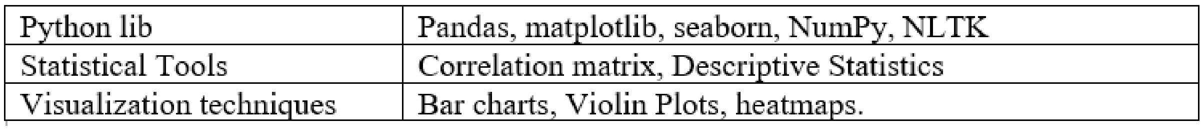


Figure 6: Correlation matrix for ham and Spam

The above correlation matrix shows how features like email length, word count, and sentence count are connected. Email length and word count have a strong relationship (0.97). This means longer emails usually contain more words. Word count and sentence count are also related (0.68) the emails with more words contain more sentences. Email length and sentence count have a weaker connection (0,62). This means that while longer emails might have more sentences, the relationship is not strong as with the number of words.

Tools used:



Python lib	Pandas, matplotlib, seaborn, NumPy, NLTK
Statistical Tools	Correlation matrix, Descriptive Statistics
Visualization techniques	Bar charts, Violin Plots, heatmaps.

5.3 Model Development

Machine Learning Model: In this process we divided the dataset into two parts 80% percentage of the data was used to train the model to teach it how to identify spam and non-spam emails. 20% of data was saved to test how well the model works to check it can identify the spam emails. Then we trained the model with random forest algorithm to classify emails as spam or non-spam. The model was trained using TF-IDF in which we can transfer text into numbers by measuring how important words are in emails. After training our model, we tested the model performance using measurements like Accuracy, Precision and recall.
Tools used: RandomForestClassifier, scikit-learn, TfidVectorizer.

Deep Learning Model: In this process, first we balanced the imbalanced dataset. We changed the text labels ham and spam into numbers and then we mapped ham to 0 and spam is mapped to 1. After this step, we divided the dataset into test and train. We used 80% of data for training and 20% of data for testing the model.

After splitting the dataset, then we pre-processed the text for further analysis. These models can’t understand the text, so we converted into numbers. To convert we tokenize the words and converted into numerical sequences finally we checked all the sequence are of the same length. To preprocess the hyperparameter we set a maximum length of sequence to 50 words. If the sequence has more than 50 words, it will cut off. We set a variable to cut off long sequences. In our code we create a tokenizer object with the defined hyperparameter and use it to process the training data. It will keep only the top 500 common words, and it adds a special token for any word not in the top 500. Then the tokenizer creates a word index where unique values are assigned in a unique integer. There are 4,169 unique words are present in the training data.

After pre-processed the text we understanded few hyperparameter to control how the model works. Each learning model layer has 20 units to process the text data. We set return sequence to true to tell models to give output for every word in the input sequence. To prevent overfitting 20% of nodes will be randomly turned off. Then we build the deep learning models. The models contain embedding layer which converts each word into numerical vector. The we processed the sequence of word embeddings. Each model uses two layers to analyse the sequence of words in the text. The first layer outputs a sequence for the second layer to analyse. The second layer processes the sequence and sends its output to final layer. The final layer uses a sigmoid activation function to give an output between 0 and 1 which indicates the message is ham and spam.

After building the model we trained and compiled the models. Then we evaluated the results for all deep learning models.

BERT Model

Splits the Dataset
For this model we split the dataset into training, validation, and testing sets. First, we split the data for creating training and temporary data. The dataset split into 70% for training and 30% for temp text and temp labels. The labels indicate the spam or non-spam are also split in the same way. The second split is splitting the temporary data 30% from the first split. Validation data 15% of the total dataset. Test data 15% of the total dataset. The validation set is used to tune the model during training. The test set is kept separate to evaluate the model's performance after training.

Import Bert Model and Bert Tokenizer
Bert cannot process raw text, so the tokenizer breaks text into smaller parts called tokens and converts them into numbers that the model can understand. Then Bert Tokenizer downloads and loads the tokenizer corresponding to the Bert-base-uncased model. It checks the text is processed in a way that matches how Bert was trained. The tokenizer ensures the text input is formatted correctly for BERT.

Convert Integer sequence to Tensors
In this step, The input id of training sets from the tokenized training data into PyTorch tensor. These input ids represent the words in training set, converted into numeric format by a tokenizer. Then we converted the attention mask from the tokenized training data into a tensor. An attention mask tells the model which parts of the input text are relevant, usually 1 for real words and 0 for padding words. It helps the model focus on the actual content and ignore padded parts. Then we converted the labels of the training data into a tensor. The labels are correct classifications (spam or ham) for each training message. The same step we followed for validation set and test set.

Create Data loaders
In this step, we prepared the training and validating datasets for use in a PyTorch model. First it defines a batch size then the training data, validation data are combined into tensor dataset which stores the input sequences, attention masks, and labels together. For training set, random sampler is used to shuffle the data during training, ensuring the model learns from varied samples. For validation set, a sequential sampler is used to keep the data in order. Both datasets are loaded into Data loader which helps manage batching during training and validation.

Fine-Tune Bert
In this step, we discuss Bert Fine-tune in this code function trains a model using a training dataset, optimizing its performance over multiple batches. The model is set to training mode total_loss and total_accuracy track the model's performance, and total_preds stores prediction. The training data is processed in chunks every 50 batches are progressed and printed. Each batch is sent to the GPU for faster computation. Batch data is separated into input and target. For training the model previous gradients are cleared, predictions are made and difference from actual values is calculated using a function loss. Finally, predictions are moved to the CPU, reshaped and saved. After fine tune the Bert, we started model training and evaluated the results.

Output:
After following these steps all the learning models are trained and ready to classify the spam messages with high precision and recall.

6 Evaluation

In evaluation we will discuss the efficacy of the models used for spam detection highlighting their performance, strengths and trade-offs.

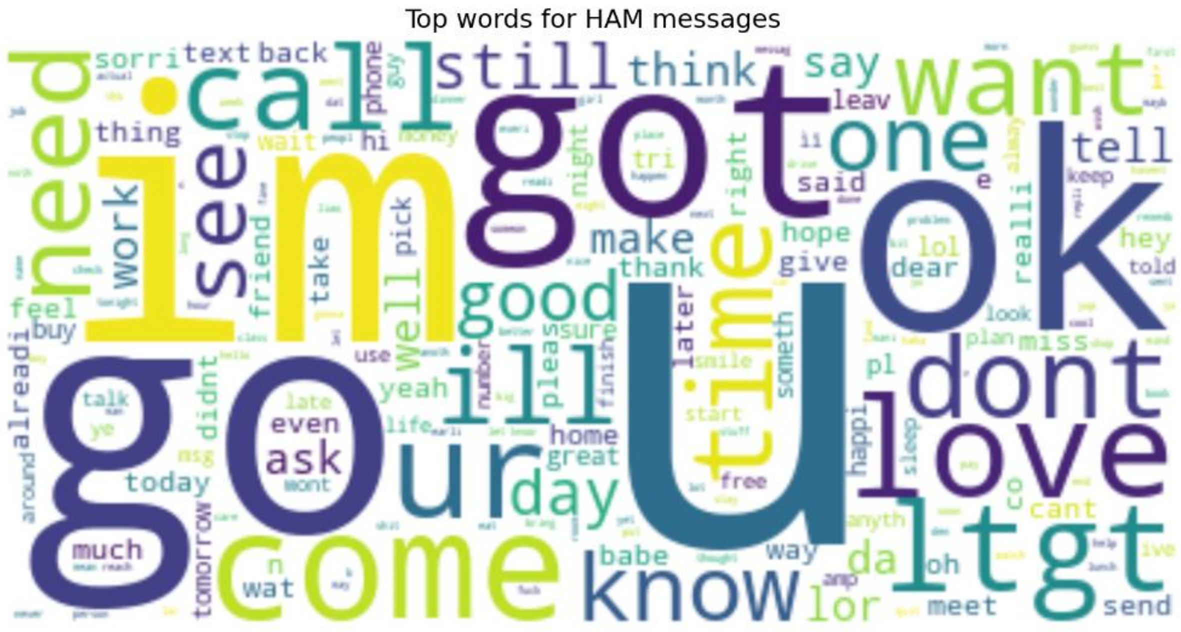


Figure 7: Displays the top 50 ham words present in the dataset

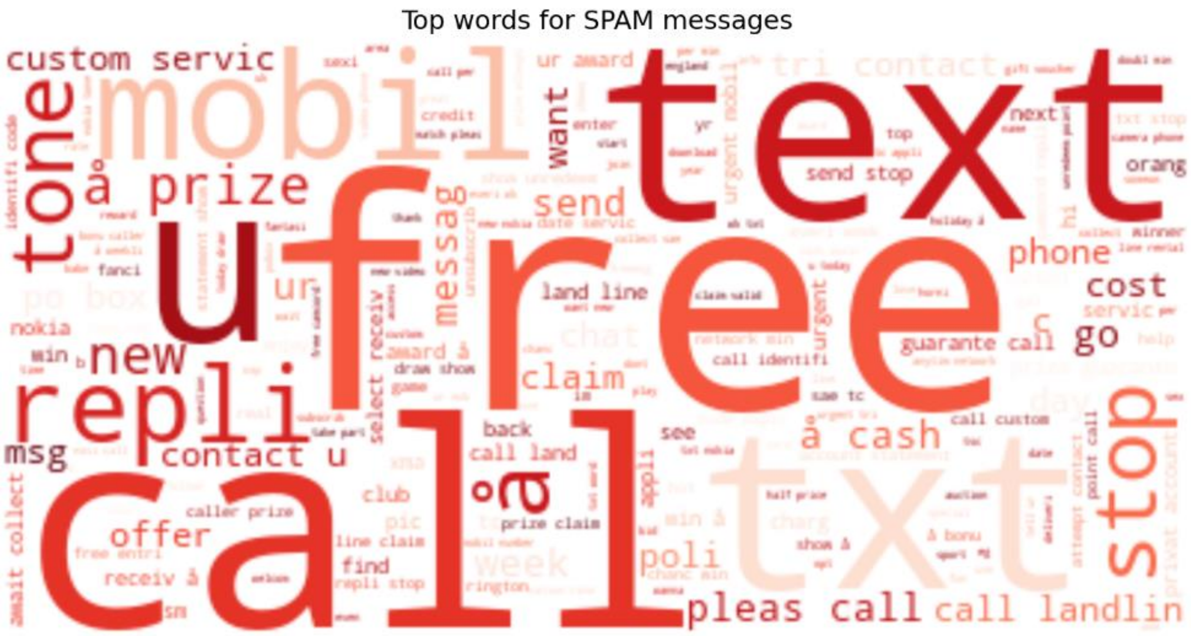


Figure 8: Displays the top 50 ham words present in the dataset

6.1 Key Findings

Comparison of Models

The chart below provides a comparative summary of the model performance

Model	Accuracy	Loss	Precision	Recall	F1-Score
Random Forest	0.89	-	0.58	-	-
Simple RNN	0.51	0.49	0.92	0.66	0.14
LSTM	0.92	0.87	0.97	0.92	0.92
GRU	0.48	0.48	1.00	0.65	0.04
Bi-LSTM	0.94	0.90	0.98	0.94	0.94
Bi-LSTM + GRU	0.95	0.91	0.99	0.95	0.95
BERT	0.98	0.89	0.99	0.93	0.99

1. **Random Forest**
   This model achieved an accuracy of 89%. It means it identifies most of the messages. But the precision for spam detection is 58% means it misclassified many non-spam messages as spam. While it gives a overall good performance but it is less effective than advanced deep learning models in identifying spam messages accurately.

2. **Simple RNN**
   The simple RNN has an overall accuracy of 51% which is very low. It performs well in detecting non-spam messages with a precision of 92% but struggles significantly with spam detection achieving only 49% precision. The recall for non-spam is extremely low at 8% it fails to identify most non-spam messages correctly. This imbalance shows that the model cannot handle complex data patterns effectively.

3. **LSTM**
   The LSTM model achieves an accuracy of 93.98% it correctly classified most of the messages. The low error rate shows it effectively learned complex data patterns. This model is good in detecting longer messages. But it requires more computational resources and took longer to train the model compared to other simpler models. It is highly effective for spam detection task.

4. **GRU**
   The GRU model performed poorly on this dataset, with an accuracy of 48%. While it correctly identified all non-spam messages with precision of 100%. But it struggled to recognize the actual spam messages with recall of 2 %. This indicates a strong imbalance in the results where the model favoured non-spam messages and failed to classify most spam ones.

5. **BI-LSTM**
   The Bi-LSTM model achieved an accuracy of 94% with strong performance in identifying both spam and non-spam messages. It had high precision and recall with F1 scores of 94% showing it was good at classifying both types of messages. Its strength lies in capturing context from both directions in text, making it one of the best models for detecting spam.

6. **BI-LSTM & GRU**
   The Bi-LSTM-GRU hybrid model achieved 95% accuracy, performing even better than Bi-LSTM alone. It provided a better balance in classifying both spam and non-spam messages. By combining the strengths of Bi-LSTM and GRU, it improved the results slightly making it the best-performing model for spam detection.

7. **BERT**
   The Bert model achieved an impressive 98% accuracy in detecting spam messages. It performed exceptionally well at identifying non-spam messages, with almost perfect precision and recall. For spam messages it also performed good by achieving a 93% F1-score. As a transformer model, Bert excels at capturing semantic and contextual information in the text. The most accurate model in the study highlighting its superiority for NLP tasks like spam detection.

6.2 Discussion

My research has some similarities and differences compared to the reference paper. The reference study showed that advanced LSTM-based models and hybrid architectures like Bi-LSTM and GRU achieved excellent results with accuracy up to 99%. Similarly, in my research, LSTM models performed well, with Bi-LSTM achieving 94% accuracy and the Bi-LSTM-GRU hybrid reaching 95%. However, the improvement from combining Bi-LSTM and GRU was minimal, suggesting that the hybrid model could be further optimized for better performance. My GRU model performed poorly, achieving only 48% accuracy, like Simple RNN, which struggled due to their inability to handle complex data patterns. This shows a need for better training techniques and fine-tuning. The Bert model in my research stood out, achieving 98% accuracy which aligns with the reference paper's findings. However, Bert requires significant computational power, making it less practical in some cases.

7 Conclusion and Future Work

Conclusion:

In this project, we aimed to answer the research questions: can Transformer models like BERT outperform traditional machine learning and deep learning models in detecting SMS spam. Through systematic analysis, we compared traditional models such as Random Forest, advanced deep learning models like LSTM and Bi-LSTM and the Bert model. Our key findings show that BERT significantly outperforms other models, achieving an accuracy of 98% and excelling in spam detection due to its superior ability to understand the semantic and contextual meaning of text. while traditional models like Random Forest provided reasonable accuracy of 89% but it struggled with precision in identifying spam. Deep learning models, particularly Bi-LSTM and the Bi-LSTM-GRU hybrid, demonstrated strong performance, with accuracies of 94% and 95% respectively. However, they require optimization to achieve further improvements. The study highlights that spam messages are becoming more advanced and harder to identify. Models like Bert are very powerful because they can read messages like humans do, understanding the meaning behind the text instead of just analysing individual words. By comparing these different models, this project shows that modern approaches like BERT are better suited for detecting spam messages in today's world. These findings can help develop more effective tools to protect people from scams and unwanted messages. In our analysis Bert is the best choice model for detecting spam because it is accurate, smart and handles complex messages.

Future Work:

We can optimize a model Bert is powerful but requires a lot of computing power, which can be expensive and slow. To solve this, we can try light versions of BERT like DistilBert or ALBERT which are smaller and faster but still it maintains high accuracy. We can combine Bert with other deep learning models like LSTM or Bi-LSTM to take advantage of their strengths. This might create an even better system for detecting spam messages by using the best features of both types of models. So far, we worked with dataset of SMS messages. Next, we can test these models on real-world SMS data to see how they perform in practical solutions. This will show if the model work well outside of a controlled environment can handle real-life challenges like different languages or formats. Apart from the words in the messages we can extra features to improve the model. For an instance sentiment analysis to check if the message feels negative or promotional. Turn the research into a real product by creating a tool. This tool could be used by messaging platforms or mobile apps to automatically filter spam messages, improving user experience and security. There are steps we can implement in future to make the model more efficient, smarter, and can be used in real world applications.

References

Saab, S.A., Mitri, N. and Awad, M., 2014, April. Ham or spam? A comparative study for some content-based classification algorithms for email filtering. In *MELECON 2014-2014 17th IEEE Mediterranean Electrotechnical Conference* (pp. 339-343). IEEE.

Saini, H. and Saini, K.S., 2023, April. Hybrid Model for Email Spam Prediction Using Random Forest for Feature Extraction. In *2023 International Conference on Artificial Intelligence and Applications (ICAIA) Alliance Technology Conference (ATCON-1)* (pp. 1-4). IEEE.

Rahman, S.E. and Ullah, S., 2020, June. Email spam detection using bidirectional long short term memory with convolutional neural network. In *2020 IEEE Region 10 Symposium (TENSYMP)* (pp. 1307-1311). IEEE.

Chandra, A. and Khatri, S.K., 2019, November. Spam SMS filtering using recurrent neural network and long short-term memory. In *2019 4th international conference on information systems and computer networks (ISCON)* (pp. 118-122). IEEE.

Gadde, S., Lakshmanarao, A. and Satyanarayana, S., 2021, March. SMS spam detection using machine learning and deep learning techniques. In *2021 7th international conference on advanced computing and communication systems (ICACCS)* (Vol. 1, pp. 358-362). IEEE.
Liu, S., Tao, H. and Feng, S., 2019, November. Text classification research based on bert model and bayesian network. In *2019 Chinese automation congress (CAC)* (pp. 5842-5846). IEEE.

Raga, S.S. and Chaitra, B.L., 2022, December. A bert model for sms and twitter spam ham classification and comparative study of machine learning and deep learning technique. In *2022 IEEE 7th international conference on recent advances and innovations in engineering (ICRAIE)* (Vol. 7, pp. 355-359). IEEE.

Datta, S., Bandyopadhyay, S. and Mondal, B., 2023, November. Classification of Spam and Ham Emails with Machine Learning Techniques for Cyber Security. In *2023 International Conference on Integrated Intelligence and Communication Systems (ICIICS)* (pp. 1-6). IEEE.

Oyeyemi, D.A. and Ojo, A.K., 2024. SMS Spam Detection and Classification to Combat Abuse in Telephone Networks Using Natural Language Processing. *arXiv preprint arXiv:2406.06578.*

Sireesha, S.A., Karthik, S.B., Srena, K., Gopal, S.N. and Reddy, S.K., 2023. SMS Spam Detection Using Machine Learning. *SJIS-P*, *35*(1), pp.749-754.

Ghourabi, A. and Alohaly, M., 2023. Enhancing spam message classification and detection using transformer-based embedding and ensemble learning. *Sensors*, *23*(8), p.3861.

Murynets, I. and Piqueras Jover, R., 2012, November. Crime scene investigation: SMS spam data analysis. In *Proceedings of the 2012 Internet Measurement Conference* (pp. 441-452).

Roy, P.K., Singh, J.P. and Banerjee, S., 2020. Deep learning to filter SMS Spam. *Future Generation Computer Systems*, *102*, pp.524-533.

Jazzar, M., Yousef, R.F. and Eleyan, D., 2021. Evaluation of machine learning techniques for email spam classification. *International Journal Of Education And Management Engineering*, *11*(4), pp.35-42.

Srinivasarao, U. and Sharaff, A., 2023. Machine intelligence based hybrid classifier for spam detection and sentiment analysis of SMS messages. *Multimedia Tools and Applications*, *82*(20), pp.31069-31099.

Guzella, T.S. and Caminhas, W.M., 2009. A review of machine learning approaches to spam filtering. *Expert Systems with Applications*, *36*(7), pp.10206-10222.

Taheri, R. and Javidan, R., 2017, October. Spam filtering in SMS using recurrent neural networks. In *2017 Artificial Intelligence and Signal Processing Conference (AISP)* (pp. 331-336). IEEE.

Chen, H., 2018, September. Spam message filtering recognition system based on tensorflow. In *2018 3rd International Conference on Mechanical, Control and Computer Engineering (ICMCCE)* (pp. 564-567). IEEE.

Jáñez-Martino, F., Alaiz-Rodríguez, R., González-Castro, V., Fidalgo, E. and Alegre, E., 2023. Classifying spam emails using agglomerative hierarchical clustering and a topic-based approach. *Applied Soft Computing*, *139*, p.110226.

Yaseen, Q., 2021. Spam email detection using deep learning techniques. Procedia Computer Science, 184, pp.853-858.