SMS Spam Detection Using Machine Learning Approach

Patrick Ozoh*, Musibau Ibrahim, Ridwan Ojo, Gbotosho, A. Sunmade and Tosin Oyetayo

Faculty of Computing and Information Technology, Osun State University, Nigeria
*corresponding Author: patrick.ozoh@uniosun.edu.ng
Received: 15 April 2025; Accepted: 18 June 2025

Abstract
Currently, as the popularity of mobile phones has increased, Short Message Service (SMS) has grown tremendously. The minimal cost of messaging services has increased spam or unsolicited messages sent to mobile phones. There are differences between spam filtering for text messages and emails. Emails have a set of big datasets, while the actual databases for SMS spam are very limited. Because of the small size of text messages, the features used for classification are smaller than the equivalent number in emails. Text messages consist of abbreviations and have less formal language than that of emails. Short Message Services (SMS) spam has become a pressing issue in mobile communication, disrupting user experiences and posing privacy threats. This study develops a useful system for identifying spam messages in SMS communications. It presents a machine learning-based framework for detecting SMS spam, utilizing a Multi-Layer classifier. This is aimed at tackling the problem of spam messages in SMS communications through the development of a robust and efficient spam detection system. This entails a data preprocessing procedure to prepare the raw SMS dataset. The TF-IDF technique was used to handle feature extraction to represent the text data numerically. This enables the model to capture relevant characteristics distinguishing spam from non-spam messages. The model was trained using the preprocessed data and evaluated through cross-validation. The results highlight the scalability and reliability of this approach, providing a practical solution for enhancing SMS spam detection systems and improving user security in mobile communication, employing the multi-layer classifier for an effective spam detection system, ensuring the models' optimal performance while preventing overfitting to deliver a comprehensive solution to the persistent issue of SMS spam.

Keywords: Spam messages, SMS, Preprocessing, TF-IDF, Feature extraction, Evaluation

Introduction

The growth rate of the mobile phone market has increased over the years, as shown in the literature reviewed. Over the past few years, there has been a significant increase in the mobile phone market. In the second quarter of 2013, there was a 6.0% year-over-year increase, with 432.1 million mobile phones shipped (Biglia *et al*., 2013). SMS is text communication using mobile phones to communicate using short text messages. It is an extensively used application worldwide. There has been an increase in the volume of unsolicited advertisements sent to mobile phones. SMS Spam is more disturbing than email, due to costs accrued to the receiver. With the widespread use of mobile phones, this study is now a multi-billion-dollar commercial industry (Gadde *et al*., 2021). SMS is a text communication platform that allows users to communicate with fewer than 160 seven-bit characters. It is estimated that approximately 3.5 billion people use SMS (Salman *et al*., 2024).

As the platform's popularity has grown, so has text messaging risen. In Africa, including Nigeria, SMS spam is also becoming more common due to the increasing popularity of the platform among young people, as well as the declining cost of text messaging over the years. While SMS spam is not yet as prevalent as email spam, there is evidence of a growing trend. For instance, a 2018 report found that in South Africa, 15% of all text messages sent were spam (Oyeyemi & Ojo, 2024). Ejirika & Omotehinwa (2024) revealed that SMS spam accounts for 21% of all text messages received globally. Some of these messages are unsolicited advertisements for various goods and services (Mambina *et al*., 2024). These spam messages can be particularly frustrating for Nigerian mobile phone users. In some cases, users have reported receiving as many as 50 spam messages (Kalyani *et al*., 2024).

The situation is similar in other African countries as well. In Kenya, for example, SMS spam is estimated to account for about 10% of all text messages received (Kalolo & Mbelwa, 2023). This has led to increased demand for mobile phone spam-filtering software, which is still not widely available in the region. Furthermore, with SMS spam, customers can be charged a fee for receiving such messages. As a result, detecting spam messages in SMS has become an interesting research problem (Zimba *et al*., 2024). Several differences exist between spam filtering. Spam messages are very limited (Srinivasarao & Sharaff, 2023). Additionally, due to the small length of text messages, the number of features that can be used for classification is far smaller than the corresponding number in emails. Here, no header exists as well. The text messages are full of abbreviations and have much less formal language than expected from emails. These factors may result in severe degradation in the performance of email spam filtering algorithms applied to short text messages. To address this problem, researchers and developers have turned to machine-learning approaches to detect and filter out spam messages (Airlangga, 2024). The study aims to build a machine-learning model that can accurately classify SMS messages as spam or not spam. This involves extracting relevant features and training a classifier to distinguish between genuine and spam messages. By developing a robust spam detection system, we can improve the user experience and reduce the impact of SMS spam on individuals and businesses.

The increasing number of SMS spam messages has become a significant problem for individuals and businesses worldwide, causing inconvenience, irritation, and sometimes even financial loss. Traditional rule-based methods for detecting SMS spam have proven inadequate, as spammers find ways to bypass these filters. Machine learning approaches have shown promise in detecting and filtering SMS spam messages, but there is a need to improve the accuracy and efficiency of these models. Therefore, this study addresses how a robust and accurate machine learning-based SMS spam detection system can effectively differentiate between spam and legitimate SMS messages. The classifiers used by the SMS spam detection model can be slightly better than previous techniques and will still improve the final model. To improve performance of the final model, a classifier can be used in conjunction with other methods to produce an ensemble model.

There are 5 sections. Section 1 contains the background study, introduction, and contributions to knowledge. The section begins with an overview of the SMS spam problem and its impact on individuals and organizations. The literature review and related works are embedded in Section 2. The review focuses on machine learning approaches to SMS spam detection and their application to SMS spam detection. It also discusses different feature extraction methods used in SMS spam detection. The methods used are contained in Section 3. This includes data collection methods, algorithms, and the evaluation of results. The results and discussion are discussed in Section 4. This section summarizes the key findings of this study, including its performance. Section 5 includes the summary and conclusion of the study. The section also highlights the development of more sophisticated feature extraction methods and the integration of different machine-learning techniques. The contributions to knowledge are as follows:

1. The need to improve the accuracy of the spam detection model.
2. It will enable individuals and businesses to filter out unwanted SMS spam messages.
3. It will evaluate the developed SMS spam detection system using appropriate evaluation metrics.

Literature Review

This section provides an overview of their strengths and weaknesses and any gaps in the literature. A study by Ahmed *et al*. (2022) used Naive Bayes and decision trees to classify SMS messages as spam. The paper achieved an accuracy of 99.28% using Naive Bayes and 98.12% using decision trees. The strength of this study is that it used a large dataset and achieved high accuracy. However, the study did not evaluate the performance of other machine learning algorithms, which could have led to better results. Qazi *et al*. (2024) used a combination of Naive Bayes and k-Nearest Neighbor (k-NN) algorithms to classify SMS messages as spam. The authors achieved an accuracy of 97.6%. Saeed (2023) investigates the usefulness of machine learning algorithms in differentiating spam from non-spam messages. The techniques considered are the J48, K-Nearest Neighbors (KNN), and Decision Tree (DT). It used a simple and effective combination of algorithms. The proposed method is evaluated dependent on accuracy, recall, and precision. The outcome indicates that the Decision Tree method got higher accuracy than other machine learning classifiers. However, the study did not evaluate the performance of other machine learning algorithms or consider the impact of feature selection on the results.

Recent trends include deep learning techniques, such as convolutional neural networks and recurrent neural networks used for feature extraction and classification. According to Saeed (2023), SMS spam has become a significant problem for mobile phone users worldwide enabled by adaptive spam detection algorithms. Al-Kabbi *et al.* (2023) present a method for an SMS spam detection model to collect text message features using a combined deep learning model to enhance feature representation. The model is evaluated using the UCI dataset, compared with deep learning algorithms, RF, and BERT using cross-validation to test for accuracy and robustness. The proposed model shows superior performance, achieving a good accuracy of 99.56% than other methods. The efficiency of this model in SMS spam detection shows its impact on real-world implementation. About the spread of SMS spam messages, De Goma *et al.* (2024) enhance existing SMS spam detection models using deep learning models and word embedding techniques to show the importance of combating SMS spam and spam filtering. The methods used include LSTM and BiLSTM models. The methods use the K-Fold cross-validation, with the results evaluating the efficacy of TF-IDF vectorization in boosting SMS spam detection. The study provides mobile phone users with enhanced defenses against the harmful threat of SMS spam.

Machine learning techniques have great potential in detecting SMS (Hadi & Baawi, 2024). Al-shanableh *et al*. (2024) provide an overview of various machine-learning approaches for SMS spam detection, including decision trees, random forests, support vector machines, and deep-learning techniques. Investigating the performance of SMS spam detection models, the evaluation metrics used include precision, recall, F1 score, and accuracy. These metrics investigate and guide the optimization of the model. The use of various feature extraction techniques and classification algorithms in detecting spam messages within SMS data is investigated by Ahmadi *et al*. (2025). the Naive Bayes, K-Nearest Neighbors, Support Vector Machines, Linear Discriminant Analysis, Decision Trees, and Deep Neural Networks were analyzed using two feature extraction methods: bag-of-words and TF-IDF. The results show that the TF-IDF method outperforms the bag-of-words approach over all six classifiers. The study provides a basis for improving SMS spam detection. Johari *et al.* (2025) examine the performance of SMS spam detection datasets. The study utilizes the Decision Tree and Multinomial Naïve Bayes models and uses two removal groups: one in English and the other using non-English languages. The results proposed a dataset for future SMS spam detection, showing a high qualitative assessment score of 3.8 out of 5.0. The study introduces robust and adaptable spam detection models that can handle errors in models. Senthilkumar *et. al.* (2025) uses machine learning techniques to examine an SMS Spam Detection system. The methods employed include using natural language processing (NLP) techniques and machine learning algorithms to select and classify SMS messages as spam or non-spam. The results indicate that the Naive Bayes show a high level of accuracy in differentiating spam from non-spam messages. The study helps to improve the efficiency and reliability of SMS spam detection systems.

Methodology

An overview of the methodology used to build an SMS spam detection model is provided that includes data collection process, feature selection and extraction, model selection, model training, and model evaluation.

Data Collection

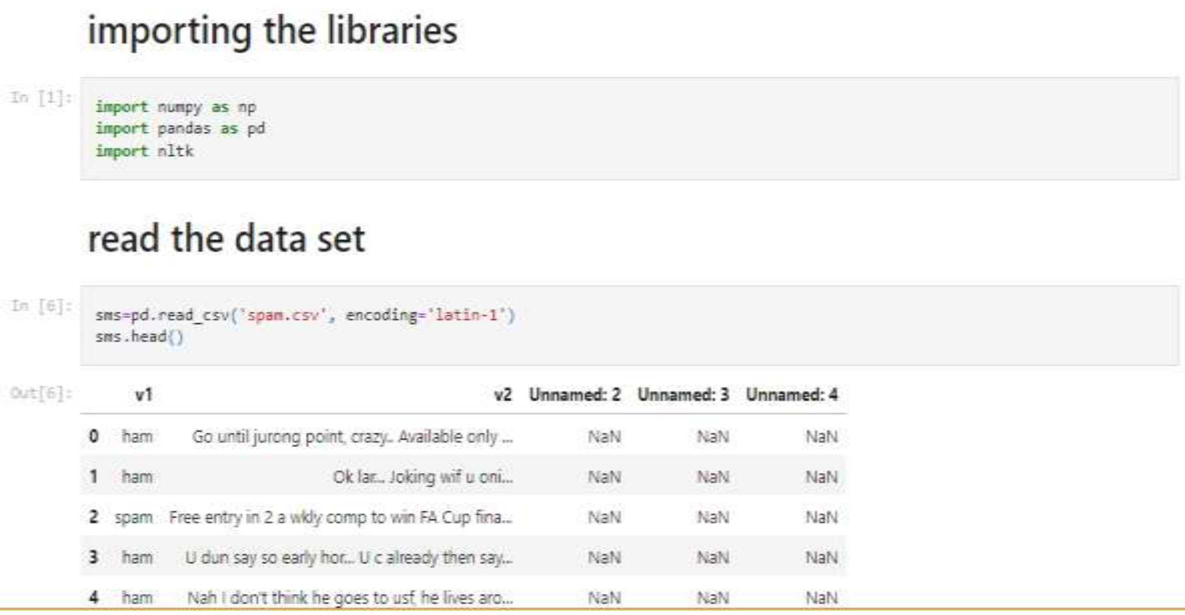
The dataset used is obtained from publicly available sources such as Kaggle.

Data Pre-processing

The data preprocessing highlights include:

Data Loading

A dataset containing SMS messages labeled as spam or non-spam (ham) is collected and loaded into a machine-learning environment. The data should be in a format that can be easily processed, such as a CSV or text file. This is shown as follows (Figure 1):



importing the libraries

```
In [1]: import numpy as np
        import pandas as pd
        import nltk
```

read the data set

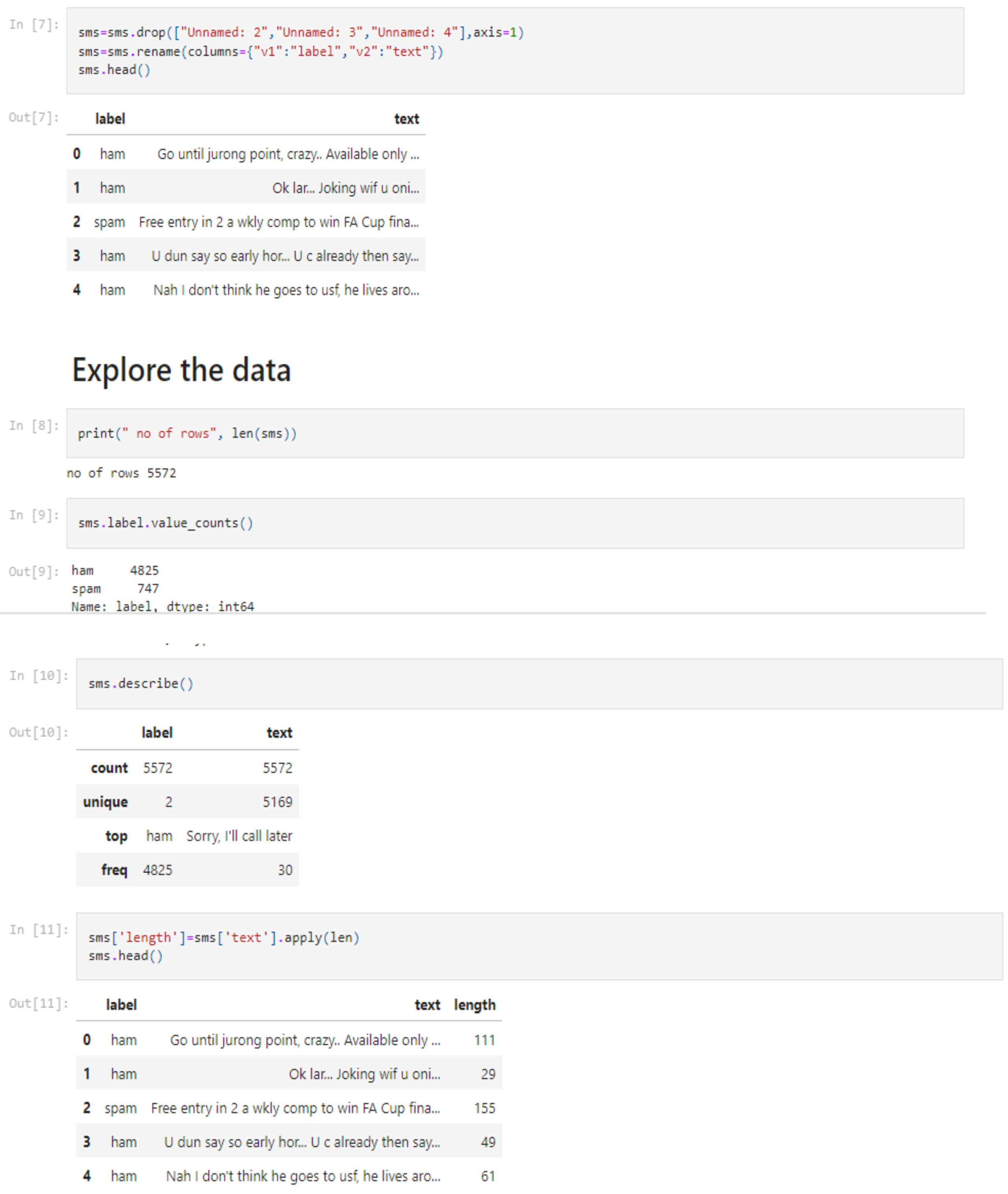
```
In [6]: sms=pd.read_csv('spam.csv', encoding='latin-1')
        sms.head()
```

```
Out[6]:
     v1                                             v2  Unnamed: 2  Unnamed: 3  Unnamed: 4
0   ham    Go until jurong point, crazy.. Available only ...        NaN         NaN         NaN
1   ham                        Ok lar... Joking wif u oni...        NaN         NaN         NaN
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...        NaN         NaN         NaN
3   ham   U dun say so early hor... U c already then say...        NaN         NaN         NaN
4   ham   Nah I don't think he goes to usf, he lives aro...        NaN         NaN         NaN
```

Figure 1: Data loading

Data Exploration

Explore the dataset to understand its structure and characteristics. Check for the distribution of spam and non-spam messages to identify if the data is imbalanced. Analyze the length of SMS messages to get insights into the average and maximum. This is shown below (Figure 2):



In [7]:
```
sms=sms.drop(["Unnamed: 2","Unnamed: 3","Unnamed: 4"],axis=1)
sms=sms.rename(columns={"v1":"label","v2":"text"})
sms.head()
```

Out[7]:

	label	text
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

Explore the data

In [8]:
```
print(" no of rows", len(sms))
```

```
no of rows 5572
```

In [9]:
```
sms.label.value_counts()
```

Out[9]:
```
ham     4825
spam     747
Name: label, dtype: int64
```

In [10]:
```
sms.describe()
```

Out[10]:

	label	text
count	5572	5572
unique	2	5169
top	ham	Sorry, I'll call later
freq	4825	30

In [11]:
```
sms['length']=sms['text'].apply(len)
sms.head()
```

Out[11]:

	label	text	length
0	ham	Go until jurong point, crazy.. Available only ...	111
1	ham	Ok lar... Joking wif u oni...	29
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	155
3	ham	U dun say so early hor... U c already then say...	49
4	ham	Nah I don't think he goes to usf, he lives aro...	61

Figure 2: Data exploration

Data Cleaning
Check for and eliminate any duplicate SMS messages in the dataset to avoid bias in the model training. Examine if there are any missing values in the dataset. Decide how to handle missing values. This is done by imputing with appropriate values or removing the rows with missing data. Convert all SMS messages to lowercase to ensure uniformity in text.

Tokenization
Split each SMS message into words (tokens) to process them individually. Remove common and irrelevant words (stopwords) that don't contribute much to the classification task, such as "a," "the," and "is." Eliminate any non-alphanumeric characters, punctuation marks, and symbols that are not relevant for spam detection. Remove digits or numbers from the messages, as they may not be crucial for spam detection, and consider using a spell checker to correct common spelling errors in the text. If the dataset is imbalanced, use techniques such as oversampling (e.g., SMOTE - Synthetic Minority Over-sampling Technique) or under sampling to balance the classes. This ensures that the model is not biased towards the majority class.

Feature Selection and Extraction
This is a crucial step in building a robust SMS spam detection model. It involves transforming the raw data into informative features that can effectively represent the underlying patterns in the text data. For example, Text Length: Create a feature representing the length of each SMS message (e.g., the number of characters or words). Longer or shorter messages might have different characteristics for spam and non-spam messages.

Number of Digits: Count the number of digits in each SMS message. Spam messages might contain more numbers, such as phone numbers or random digits. Several Special Characters: Count the number of special characters (e.g., "$", "%", "#") in each SMS message. Spam messages might use more special characters to grab attention.

Capital Letters Ratio: Calculate the ratio of capital letters to the total number of letters in each SMS. Spam messages might use more uppercase characters to be eye-catching.
Number of Exclamation Marks: Count the number of exclamation marks in each SMS. Spam messages might use excessive exclamation marks for emphasis.

Presence of URLs or Phone Numbers: Create binary features to indicate the presence of URLs or phone numbers in each SMS message, as spam messages may contain such information.

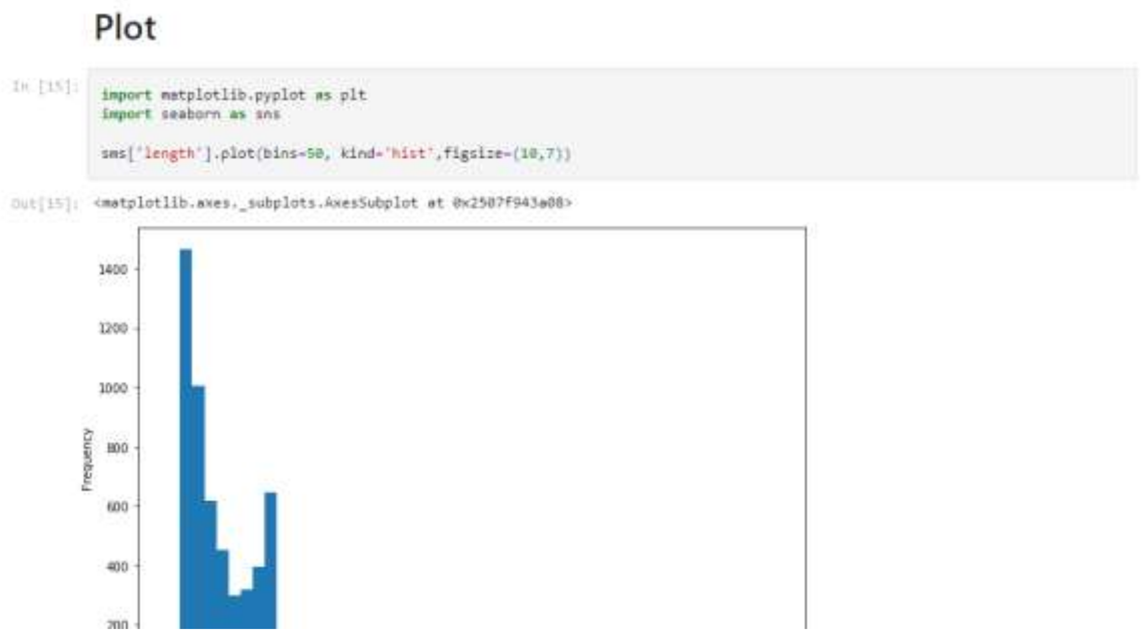
Presence of Keywords: Create binary features based on the presence of specific keywords or phrases that are commonly associated with spam messages.

Text Sentiment Analysis (optional): Perform sentiment analysis on the SMS messages to see if spam messages have a different sentiment compared to non-spam messages, *N-grams:* Extract word n-grams (e.g., bi-grams, tri-grams) to capture local word patterns that might be spam.

The features used to train the SMS spam detection model are chosen based on their ability to distinguish between spam and harmful messages. The features used to train the SMS spam detection model are chosen based on their ability to relate between spam and ham messages. The features used in SMS spam detection include word frequency, character frequency, and n-grams. The features are extracted using the bag-of-words model, TF-IDF, and word embedding techniques.

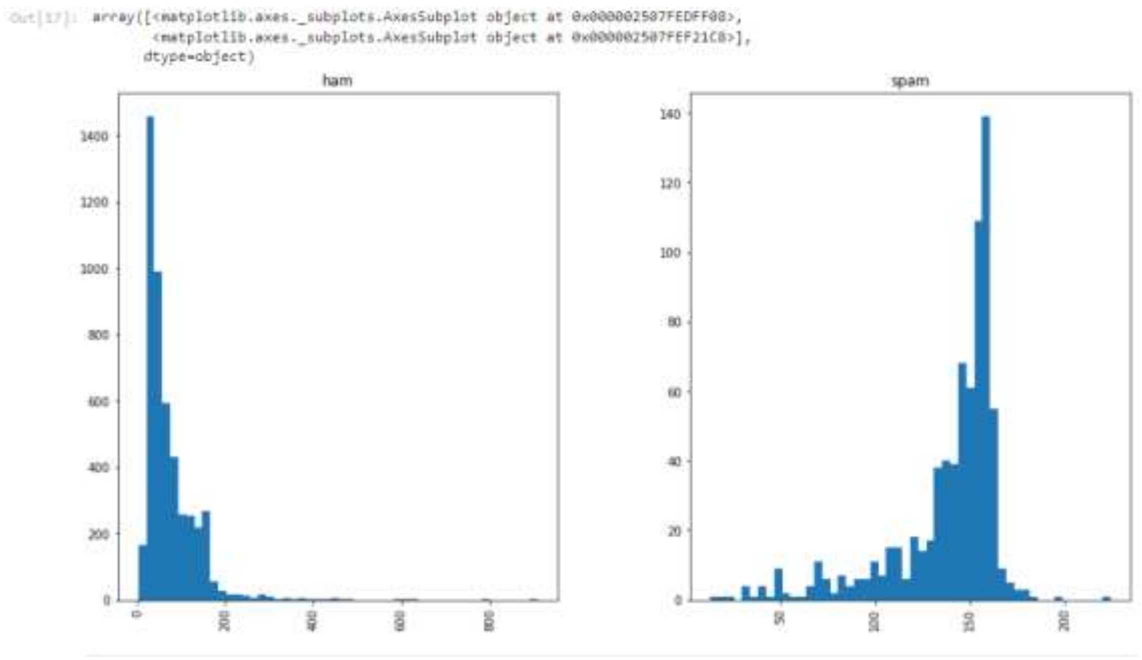
The features are selected based on their ability to capture the semantic and syntactic characteristics of the messages. By including features that represent the context and meaning of the text, the accuracy of the model will be improved.

Model Training
The preprocessed dataset is divided into training and testing sets-70% of the data for training and the remaining 30% for testing the model's performance. The spam and non-spam labels are converted into numerical values (e.g., 0 for non-spam and 1 for spam). Figure 3 shows the training data.



Plot

```
In [15]:  import matplotlib.pyplot as plt
          import seaborn as sns

          sms['length'].plot(bins=50, kind='hist',figsize=(10,7))

Out[15]:  <matplotlib.axes._subplots.AxesSubplot at 0x2507f943a08>
```



```
Out[17]: array([<matplotlib.axes._subplots.AxesSubplot object at 0x000002587FEDFF08>,
               <matplotlib.axes._subplots.AxesSubplot object at 0x000002587FEF21C8>],
              dtype=object)
```

Figure 3: Training data

An Array of Data Descriptions
This section includes the algorithms for the models.

Naive Bayes Algorithm
This uses Bayesian conditional probability theory. The algorithm is used to construct classifier models that assign the labels to instances in question given in feature value vectors. The algorithm trains the model while treating features, as discussed by the Bayesian conditional probability theory.
It is given as follows.
Step 1: Select a training set of positive and negative elements.
Step 2: Train the classification algorithm using the training data.
Step 3: Use a classification algorithm to classify sample data.
Step 4: Compare the results against the known results set.
The method used Andy Bromberg's method of sentiment extraction.

Naive Bayes Algorithm
The Bayes theorem given by Bharadiya (2023) is represented in Figure 4.

$$P(c\backslash x) = \frac{P(x\backslash c)P(c)}{P(x)}$$

Figure 4: Bayes theorem

Where:

$P(x \backslash c)$ =Likelihood function

The notation P (c | x) means "the probability of c given x." To compute sentiment analysis on structured data using the naïve Bayes classifier, the probability of a given category gives a string of words in a document.

$P(c)$ =Class Prior Probability function

$P(x)$ =Predictor Prior Probability function

Maximum Entropy Algorithm
The algorithm is the empirical distribution p, with x number of iterations derived in Equation (1) (Patel *et al*., 2016).

$$\log_{\vec{p}}(p) = log \prod_{x \in X} p(x)^{\overrightarrow{p(x)}} = \sum_{\mathrm{x} \in \mathrm{X}} p(x) \log p(x) \tag{1}$$

Support Vector Machine
It is represented as in Equation 2 (Land *et al*., 2020).

$$\begin{aligned} TS_i &= \max\left\{\left|\frac{\overline{x}_{ik} - \overline{x}_i}{m_k s_i}\right|, \quad k = 1, 2, \ldots K\right\} \\ \overline{x}_{ik} &= \sum_{j \in C_k} \overline{x}_{ij} / n_k \\ \overline{x}_i &= \sum_{j=1}^{n} x_{ij} / n \\ s_i^2 &= \frac{1}{n-K} \sum_{k} \sum_{j \in C_k} (x_{ij} - \overline{x}_{ik})^2 \\ m_k &= \sqrt{1/n_k + 1/n} \end{aligned} \tag{2}$$

There are k classes, max {yk, k=1,2,3,…K} is the maximum of all yk. k is the number of iterations. The number of samples is given as n. si is the pooled within-class standard deviation.

K-Means Clustering
K-means clustering is a type of unsupervised learning that puts objects into classes. Given a dataset X=[x1,…,xn], xnεRd. The dataset is partitioned into M disjoint subsets C1,…Cm. The algorithm is as in Equation 3 (Sinaga & Yang, 2020).

$$\frac{1}{m_k}\sum_{i=1}^{m_k}\left\|x^i - \mu_c k\right\|^2 \quad (3)$$

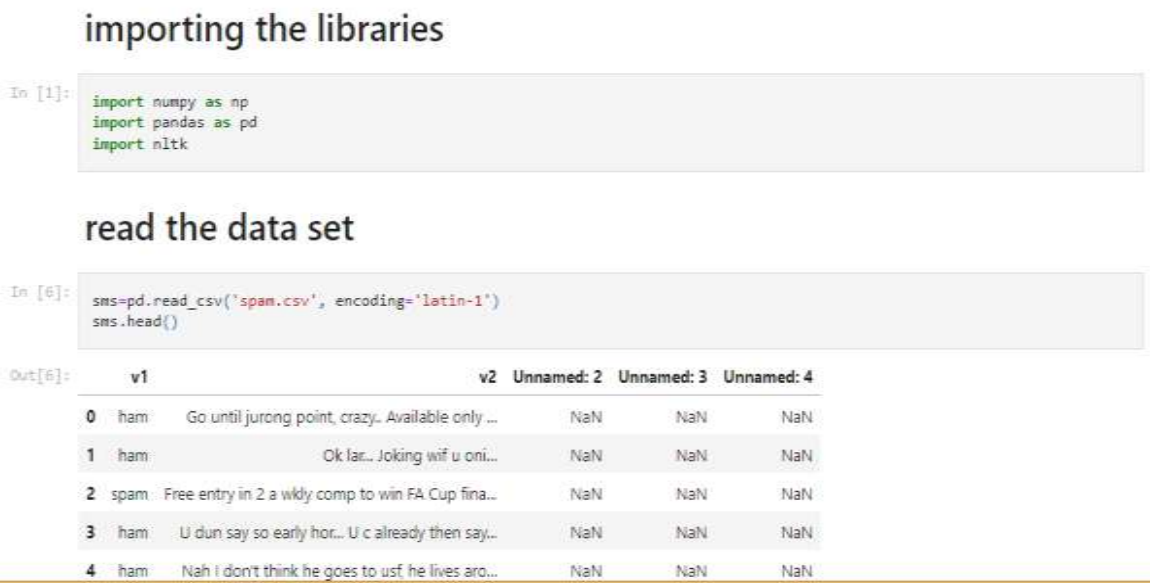
The TFIDF Technique
This technique is used to determine the relevance of a comment to a given context. To achieve this an information retrieval method is employed. TFIDF is a statistical model that infers its importance in a given textual context. It is used in modern search engines to model a content query from text to be passed to a search engine to retrieve similar documents. This method involves assigning weights to every term in a document based on term frequency (Naeem *et al*. 2022). The weights are assigned as in Equation 4.

$$W_{i,j} = TF_{t,d}\left(\frac{N}{D_t}\right) \quad (4)$$

Dt is the value of D at a specific time t.
N is the total number of observations.

BERT Model
The bidirectional encoder representations from transformers (BERT) are a language model representing text as a sequence of vectors using self-supervised learning. BERT significantly improves the state-of-the-art for large language models. BERT is a uniquely used natural language processing (NLP) modeling technique. The code for the BERT model is given in Appendix A.

Results and Discussion
Results include obtaining a dataset containing SMS messages labeled as spam or non-spam. The data is loaded into the machine learning environment. The data is loaded into the machine learning environment (Figure 5).



importing the libraries

```
In [1]: import numpy as np
        import pandas as pd
        import nltk
```

read the data set

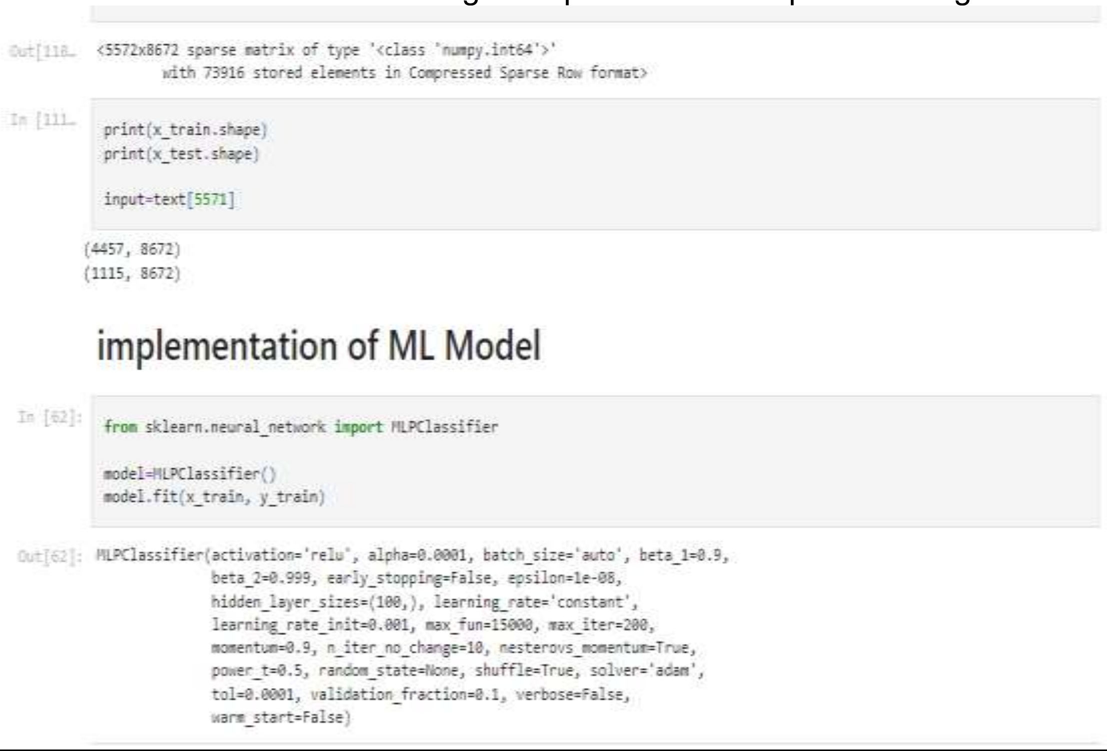
```
In [6]: sms=pd.read_csv('spam.csv', encoding='latin-1')
        sms.head()
```

Out[6]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

Figure 5: Reading the data

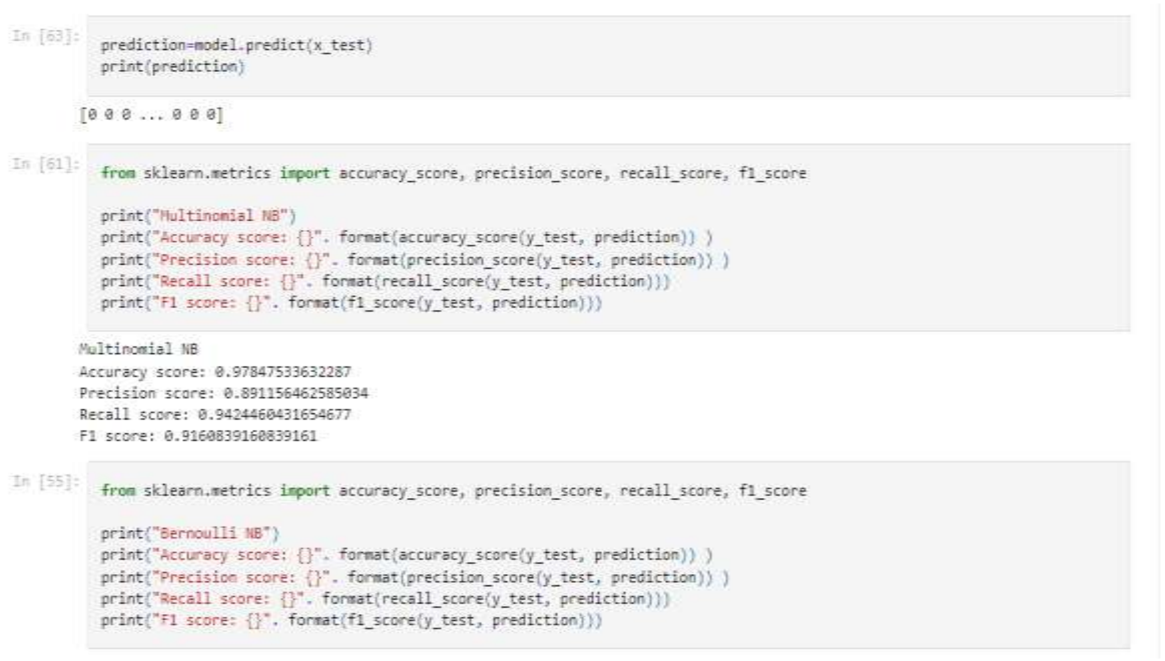
Figure 6 shows the code for implementing the TFIDF technique. This figure gains insights into what characteristics distinguish spam from non-spam messages.



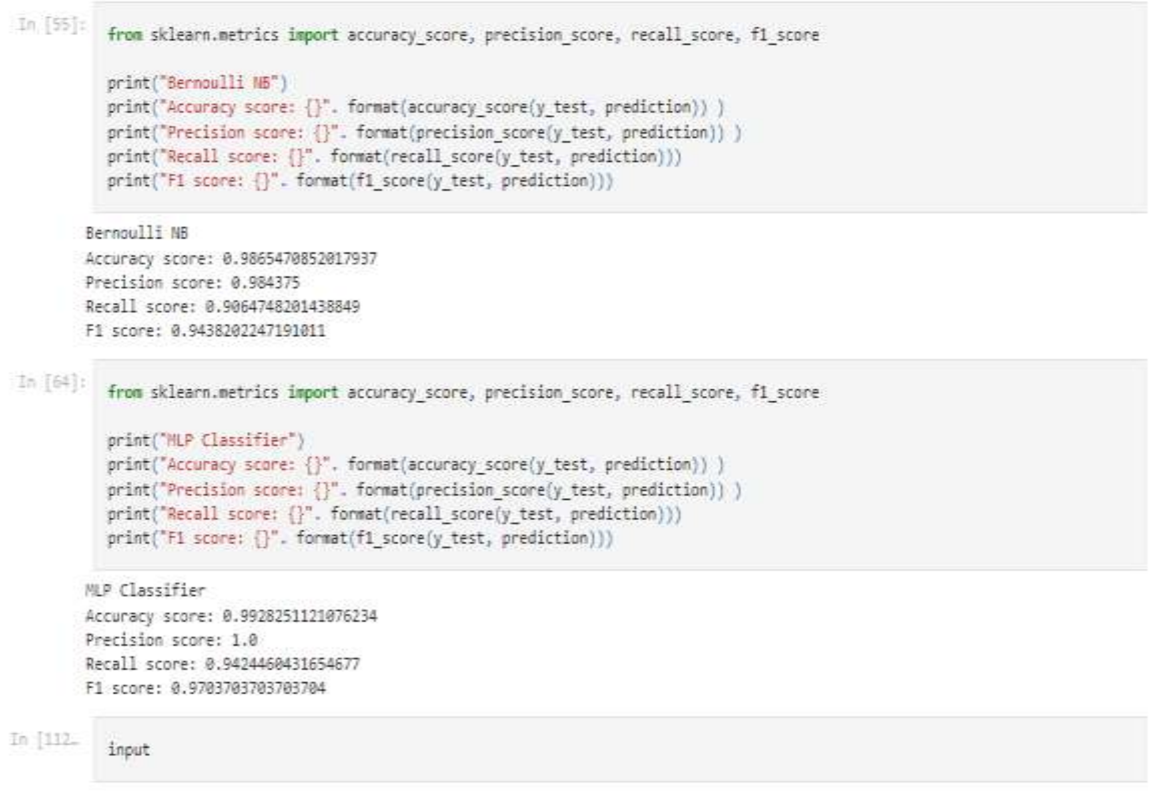
```
Out[110… <5572x8672 sparse matrix of type '<class 'numpy.int64'>'
                with 73916 stored elements in Compressed Sparse Row format>

In [111… print(x_train.shape)
         print(x_test.shape)

         input=text[5571]

(4457, 8672)
(1115, 8672)
```

implementation of ML Model

```
In [62]: from sklearn.neural_network import MLPClassifier

         model=MLPClassifier()
         model.fit(x_train, y_train)

Out[62]: MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
                       beta_2=0.999, early_stopping=False, epsilon=1e-08,
                       hidden_layer_sizes=(100,), learning_rate='constant',
                       learning_rate_init=0.001, max_fun=15000, max_iter=200,
                       momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
                       power_t=0.5, random_state=None, shuffle=True, solver='adam',
                       tol=0.0001, validation_fraction=0.1, verbose=False,
                       warm_start=False)
```



```
In [63]:  prediction=model.predict(x_test)
          print(prediction)

[0 0 0 ... 0 0 0]

In [61]:  from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

          print("Multinomial NB")
          print("Accuracy score: {}". format(accuracy_score(y_test, prediction)) )
          print("Precision score: {}". format(precision_score(y_test, prediction)) )
          print("Recall score: {}". format(recall_score(y_test, prediction)))
          print("F1 score: {}". format(f1_score(y_test, prediction)))

Multinomial NB
Accuracy score: 0.97847533632287
Precision score: 0.891156462585034
Recall score: 0.9424460431654677
F1 score: 0.9160839160839161

In [55]:  from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

          print("Bernoulli NB")
          print("Accuracy score: {}". format(accuracy_score(y_test, prediction)) )
          print("Precision score: {}". format(precision_score(y_test, prediction)) )
          print("Recall score: {}". format(recall_score(y_test, prediction)))
          print("F1 score: {}". format(f1_score(y_test, prediction)))
```



```
In [55]:  from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

          print("Bernoulli NB")
          print("Accuracy score: {}". format(accuracy_score(y_test, prediction)) )
          print("Precision score: {}". format(precision_score(y_test, prediction)) )
          print("Recall score: {}". format(recall_score(y_test, prediction)))
          print("F1 score: {}". format(f1_score(y_test, prediction)))

Bernoulli NB
Accuracy score: 0.9865470852017937
Precision score: 0.984375
Recall score: 0.9064748201438849
F1 score: 0.9438202247191011

In [64]:  from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

          print("MLP Classifier")
          print("Accuracy score: {}". format(accuracy_score(y_test, prediction)) )
          print("Precision score: {}". format(precision_score(y_test, prediction)) )
          print("Recall score: {}". format(recall_score(y_test, prediction)))
          print("F1 score: {}". format(f1_score(y_test, prediction)))

MLP Classifier
Accuracy score: 0.9928251121076234
Precision score: 1.0
Recall score: 0.9424460431654677
F1 score: 0.9703703703703704

In [112…  input
```

Figure 6: Implementation of the TFIDF technique

Figure 7 is the encoder (BERT representation). These are used for modeling and analyzing social media comments. Figure 6 is the predicting model of the raw text.

What is BERT?

[illegible]

Masked Language Modeling (Masked LM)

[illegible]

Next Sentence Prediction (NSP)

[illegible]

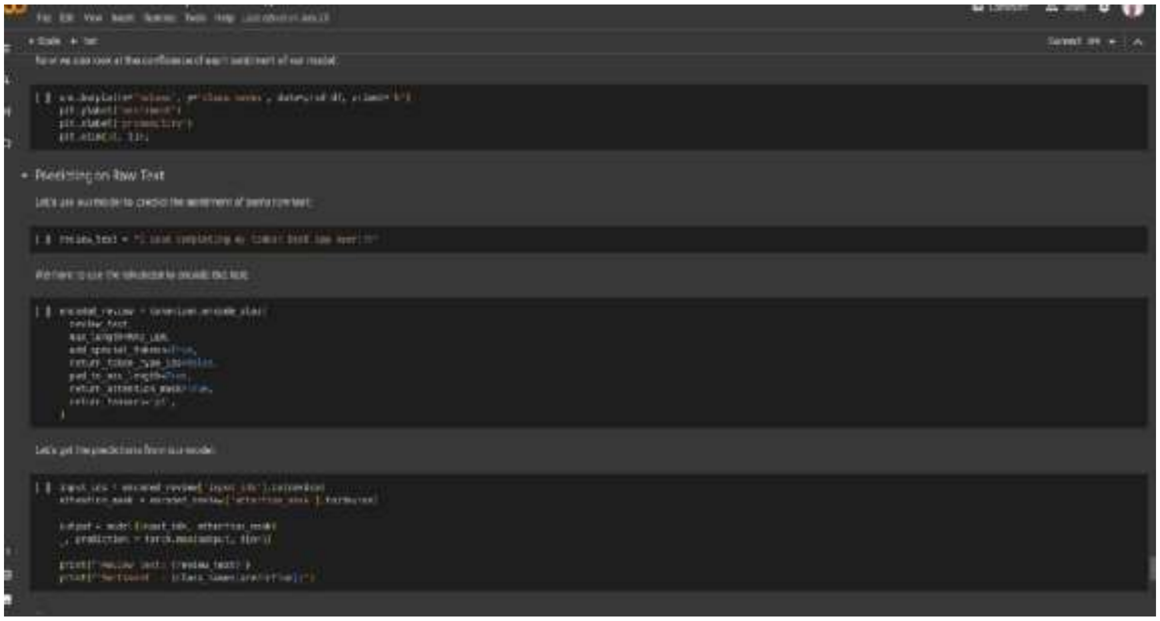
Label = IsNext

[illegible]

Label = NotNext

[illegible]

Figure 7: BERT representations



Now we can look at the confidence of each sentiment of our model.

```
[ ] sns.barplot(x='values', y='class_names', data=pred_df, orient='h')
    plt.ylabel('sentiment')
    plt.xlabel('probability')
    plt.xlim([0, 1]);
```

Predicting on Raw Text

Let's use our model to predict the sentiment of some raw text:

```
[ ] review_text = "I love completing my todos! Best app ever!!!"
```

We have to use the tokenizer to encode the text:

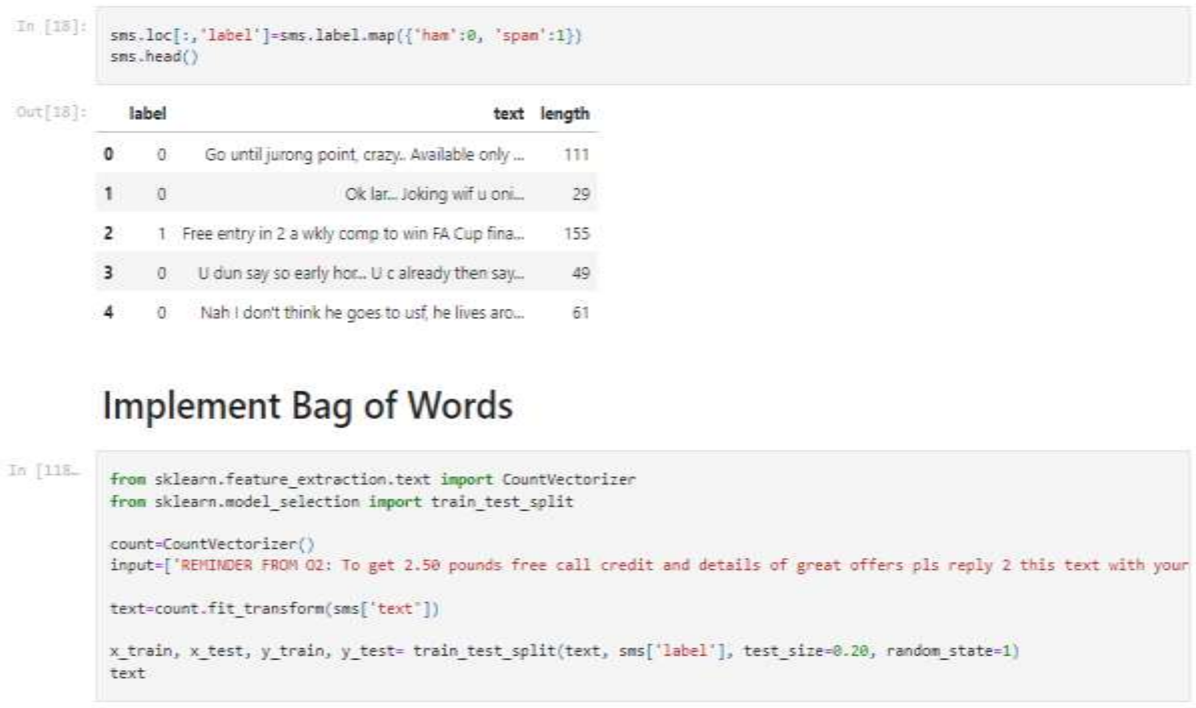
```
[ ] encoded_review = tokenizer.encode_plus(
      review_text,
      max_length=MAX_LEN,
      add_special_tokens=True,
      return_token_type_ids=False,
      pad_to_max_length=True,
      return_attention_mask=True,
      return_tensors='pt',
    )
```

Let's get the predictions from our model:

```
[ ] input_ids = encoded_review['input_ids'].to(device)
    attention_mask = encoded_review['attention_mask'].to(device)

    output = model(input_ids, attention_mask)
    _, prediction = torch.max(output, dim=1)

    print(f'Review text: {review_text}')
    print(f'Sentiment  : {class_names[prediction]}')
```

Figure 8: Predicting of raw text

The features of the SMS spam model with the TF-IDF vectors were obtained during data preprocessing to create a comprehensive feature set for the model. The identified most relevant spam detection features improve the model's accuracy and interpretability (Figure 6 and Figure 8). Figure 9 shows the results from the implementation of the bag of words.



```
In [18]:
sms.loc[:,'label']=sms.label.map({'ham':0, 'spam':1})
sms.head()
```

Out[18]:

	label	text	length
0	0	Go until jurong point, crazy.. Available only ...	111
1	0	Ok lar... Joking wif u oni...	29
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155
3	0	U dun say so early hor... U c already then say...	49
4	0	Nah I don't think he goes to usf, he lives aro...	61

Implement Bag of Words

```
In [118…
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split

count=CountVectorizer()
input=['REMINDER FROM O2: To get 2.50 pounds free call credit and details of great offers pls reply 2 this text with your

text=count.fit_transform(sms['text'])

x_train, x_test, y_train, y_test= train_test_split(text, sms['label'], test_size=0.20, random_state=1)
text
```

Figure 9: Implementation of a bag of words

This study introduces the study's focus on SMS spam detection using the Multi-Layer classifier technique. The significance of spam filtering for enhancing communication security is discussed. The objectives and scope are outlined, along with the data source and the model evaluation metrics. This covers steps such as data collection, cleaning, methods utilized, and the results. It also covers the evaluation of the trained SMS spam detection model. SMS spam detection using a machine learning approach has provided valuable insights and techniques to tackle the issue of spam messages in the mobile messaging system. The combination of model creation and evaluation has resulted in a robust spam detection system. This study's implications extend beyond SMS spam detection and can impact various text classification tasks and communication systems. Continuous research and implementation are essential in ensuring a safer, more productive, and enjoyable mobile messaging experience for users worldwide.

Conclusion

In conclusion, this study has achieved its goals by delivering an effective spam detection system. The study's outcomes contribute to advancing the field of text classification and communication security, creating a positive impact on users and businesses worldwide. The knowledge gained can be a foundation for future research and development in combating spam messages and ensuring a secure and enjoyable mobile messaging experience.

References

Ahmadi, M., Khajavi, M., Varmaghani, A., Ala, A., Danesh, K. and Javaheri, D. (2025). Leveraging large language models for cybersecurity: enhancing sms spam detection with robust and context-aware text classification. arXiv preprint arXiv:2502.11014.

Ahmed, N., Amin, R., Aldabbas, H., Koundal, D., Alouffi, B. and Shah, T. (2022). Machine learning techniques for spam detection in email and IoT platforms: analysis and research challenges. *Security and Communication Networks,* 2022(1), 1862888.

Airlangga, G. (2024). Optimizing SMS spam detection using machine learning: A comparative analysis of ensemble and traditional classifiers. *Journal of Computer Networks, Architecture and High Performance Computing*, 6(4).

Al-Kabbi, H. A., Feizi-Derakhshi, M. R. and Pashazadeh, S. (2023). Multi-type feature extraction and early fusion framework for sms spam detection. *IEEE Access*, 11, 123756-123765.

Al-shanableh, N., Alzyoud, M. S. and Nashnush, E. (2024). Enhancing email spam detection through ensemble machine learning: A comprehensive evaluation of model integration and performance. *Communications of the IIMA*, 22(1), 2.

Biglia, N., Maggiorotto, F., Liberale, V., Bounous, V. E., Sgro, L. G., Pecchio, S. and Ponzone, R. (2013). Clinical-pathologic features, long term-outcome and surgical treatment in a large series of patients with invasive lobular carcinoma (ILC) and invasive ductal carcinoma (IDC). *European Journal of Surgical Oncology (EJSO),* 39(5), 455-460.

Bharadiya, J. P. (2023). Artificial intelligence in transportation systems a critical review. American Journal of Computing and Engineering, 6(1), 35-45.

De Goma, J., Bravo, J. A., Prudente, S. and Rondilla, R. F. (2024). Detection of SMS Spam Messages Using TF-IDF Vectorizer and Deep Learning Models. In *Proceedings of the 2024 9th International Conference on Intelligent Information Technology*, 245-249.

Ejirika, E. R. and Omotehinwa, T. O. (2024). Analysis of Machine Learning Models for Spam Email Detection and Real-Time Integration. In *2024 International Conference on Science, Engineering and Business for Driving Sustainable Development Goals (SEB4SDG)*, 1-10, IEEE.

Gadde, S., Lakshmanarao, A. and Satyanarayana, S. (2021). SMS spam detection using machine learning and deep learning techniques. In *2021 7th international conference on advanced computing and communication systems (ICACCS),* Vol. 1, 358-362, IEEE.

Johari, M. F., Chiew, K. L., Hosen, A. R., Yong, K. S., Khan, A. S., Abbasi, I. A. and Grzonka, D. (2025). Key insights into recommended SMS spam detection datasets. *Scientific Reports*, 15 (1), 8162.

Hadi, M. T. and Baawi, S. S. (2024). Email Spam Detection by Machine Learning Approaches: A Review. In *International Conference on Forthcoming Networks and Sustainability in the AIoT Era*, 186-204. Cham: Springer Nature Switzerland.

Kalolo, C. and Mbelwa, J. (2023). Comparative Analysis of Machine Learning Models for Detecting Mobile Messaging Spam In Swahili SMS. In *2023 First International Conference on the Advancements of Artificial Intelligence in African Context (AAIAC)*,1-7, IEEE.

Kalyani, V. V., Sundari, M. R., Neelima, S., Prasad, P. S. S., Mohan, P. P. and Lakshmanarao, A. (2024). SMS Spam Detection using NLP and Deep Learning Recurrent Neural Network Variants. In *2024 International Conference on Cognitive Robotics and Intelligent Systems (ICC-ROBINS),* 92-96, IEEE.

Land Jr, W. H., Schaffer, J. D., Land, W. H. and Schaffer, J. D. (2020). The support vector machine. *The Art and Science of Machine Intelligence: With An Innovative Application for Alzheimer's Detection from Speech*, 45-76.

Mambina, I. S., Ndibwile, J. D., Uwimpuhwe, D. and Michael, K. F. (2024). Uncovering SMS spam in Swahili text using deep learning approaches, *12*, 25164-25175, IEEE Access.

Naeem, M. Z., Rustam, F., Mehmood, A., Ashraf, I. and Choi, G. S. (2022). Classification of movie reviews using term frequency-inverse document frequency and optimized machine learning algorithms. *PeerJ Computer Science*, *8*, 914.

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*.

Patel, D., Saxena, S., Verma, T. and Student, P. G. (2016). Sentiment analysis using maximum entropy algorithm in big data. *International Journal of Innovative Research in Science, Engineering and Technology*, *5*(5): 8355-8361.

Qazi, A., Hasan, N., Mao, R., Abo, M. E. M., Dey, S. K. and Hardaker, G. (2024). Machine Learning-Based Opinion Spam Detection: A Systematic Literature Review. *IEEE Access*. DOI 10.1109/ACCESS.2024.33992.

Saeed, V. A. (2023). A Method for SMS Spam Message Detection Using Machine Learning. *Artif. Intell. Robot. Dev. J*, *3*, 214-228.

Salman, M., Ikram, M. and Kaafar, M. A. (2024). Investigating evasive techniques in sms spam filtering: A comparative analysis of machine learning models, *12*, 24306-24324, IEEE Access.

Sinaga, K. P. and Yang, M. S. (2020). Unsupervised K-means clustering algorithm, *8*, 80716-80727, IEEE Access.

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), 31069-31099.

Senthilkumar, R., Subhalakshmi, R.T., Ramasamy, S. and Devendran. (2025). SMS Spam Detection using Machine Learning. *Journal of Science Technology and Research*, 6(1), 1-19.

Zimba, A., Phiri, K. O., Kashale, C. and Phiri, M. N. (2024). A machine learning and natural language processing-based smishing detection model for mobile money transactions. *International Journal on Information Technologies & Security*, *16*(3).