Vol. 4, Issue 1: RUGIPO Research & Technical Journal
Issues in Science & Agricultural Technology https://elearning.rugipo.edu.ng/Rugipo_Tetfund_Journal_Research_Portal/
ISSN: 2714-3643; September, 2025 TETFund/DESS/POLY/OWO/ARJ/04

DEVELOPMENT OF AN SMS SPAM FILTERING SYSTEM

OLOJIDO Joseph Bamikole[1], OLADIMEJI Banji Julius[2], and OGBEYE Olatunji Timothy[3]
[1,2]Rufus Giwa Polytechnic, Owo, Ondo State, Nigeria; [3]Adekunle Ajasin University, Akungba Akoko, Ondo State, Nigeria
Corresponding E-mail: olojidoj@gmail.com

Abstract:
This study presents the development of an SMS spam filtering system using Python, aimed at detecting and blocking unsolicited messages to enhance communication security. Leveraging machine learning techniques, the system was trained using a labelled dataset of SMS messages and implemented using algorithms such as Naive Bayes. Key modules include data preprocessing, feature extraction via TF-IDF, and model evaluation with accuracy and confusion matrix metrics. The results indicate a high level of accuracy in differentiating between spam and legitimate (ham) messages, thereby confirming the efficacy of the proposed system. This research provides a scalable and cost-effective framework that can be readily incorporated into messaging platforms.

Keywords: Keywords: SMS spam filtering, machine learning, Naïve Bayes, feature extraction, F1-score.

How to Cite: Olojido, J. B.; Oladimeji, B. J., and Ogbeye, O. T. (2025). Development of an SMS Spam Filtering System. *RUGIPO Research and Technical Journal in Science and Agricultural Technology*, 4(1):109–119. https://elearning.rugipo.edu.ng/Rugipo_Tetfund_Journal_Research_Portal/

1.0 Introduction

Globally, Short Message Service (SMS) continues to be a highly accessible and affordable means of communication. Spam is defined as electronic messages that are unsolicited or unwanted. Consequently, mobile users face growing difficulty in differentiating between spam and legitimate messages. SMS spam poses a significant nuisance for mobile users, who often receive numerous junk messages instead of genuine communications (Nagwani & Sharaff, 2017).

These messages can be categorised as electronic, unsolicited, or commercial, and they present a growing threat primarily due to several factors, including the availability of low-cost bulk SMS services, reliability, and the likelihood of receiving responses from unknown senders. The evolution of mobile communication technology and the proliferation of cell phones have contributed to this issue.

Owing to its simplicity and affordability, SMS has become a dominant form of communication. Unlike email spam, SMS spam typically generates greater response rates because users perceive SMS as a trustworthy platform and are more inclined to provide personal or sensitive information. With the widespread use of SMS, it has become easier for individuals to leverage new technologies for seamless communication.

While SMS messages are not free, the prevalence of SMS spam remains high, as filtering techniques

continue to evolve daily to minimise spam and prioritise legitimate messages

2.0 Review of Related Work

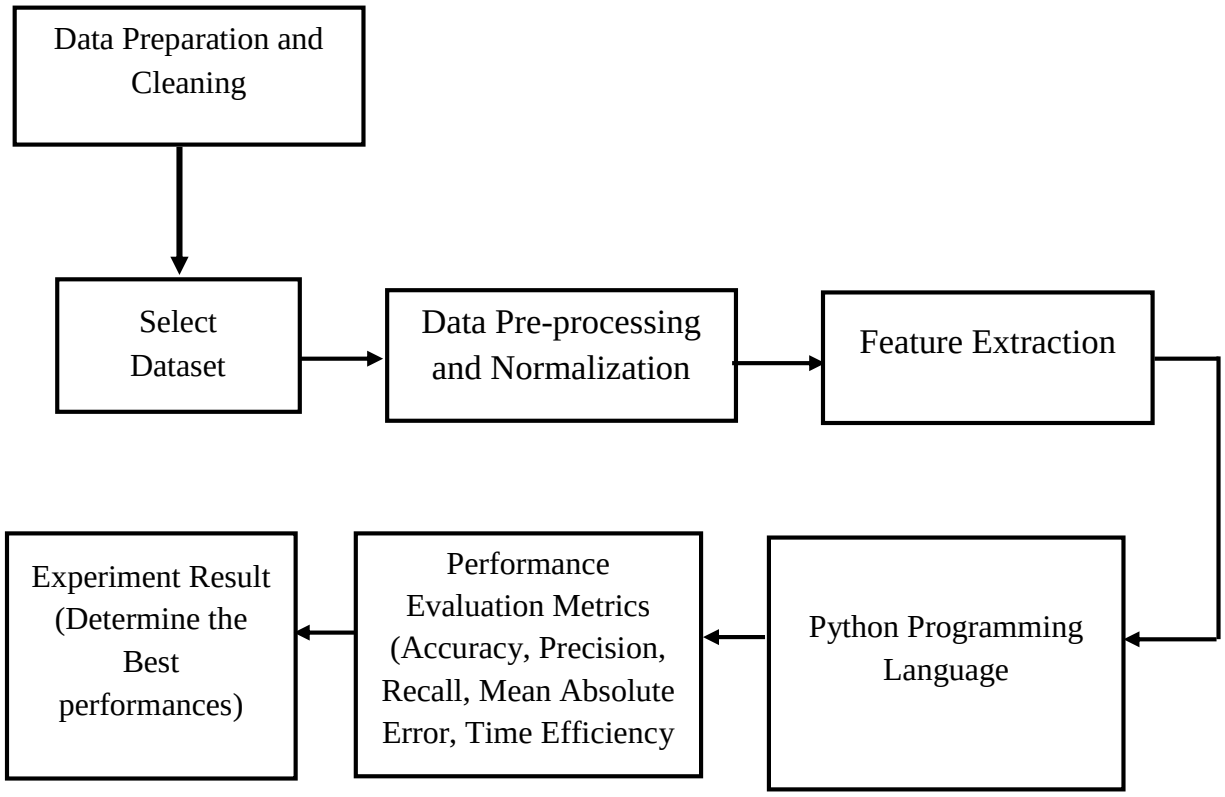
Several methods for detecting and classifying SMS spam have been explored in the literature. Warade et al. (2014) proposed a technique to identify SMS messages originating from spammer devices, with the goal of limiting their dissemination. This method involves analysing the SMS and call log databases to assess whether a direct or mutual relationship exists between the sender and the recipient.

Ahmed et al. (2014) developed a hybrid SMS classification framework to separate ham from spam messages. By integrating a Naive Bayes classifier with the Apriori algorithm and exploiting the statistical characteristics of the dataset, the system achieved an accuracy of 98.7%. In a related study, Hidalgo et al. (2016) applied Bayesian filtering—traditionally used in email spam detection—to mobile SMS. Two test corpora, in English and Spanish, were created, and machine learning techniques were employed, confirming the effective transfer of Bayesian methods from email to SMS spam detection.

Almeida et al. (2011) presented a publicly available, non-encoded SMS spam dataset and evaluated the performance of several established machine learning techniques. Their findings indicated that the Support Vector Machine (SVM) outperformed the other classifiers considered. Similarly, Cormack et al. (2017) analysed both feature-based and compression-model-based spam filters, demonstrating that the accuracy of bag-of-words filters could be substantially enhanced by incorporating additional features, whereas compression-model filters maintained strong performance even without such enhancements.

Sun et al. (2008) proposed a method for dynamically updating an adverse SMS feature library to facilitate the detection and removal of harmful mobile messages, achieving an average F1 score above 0.9 and demonstrating stable performance. Kim et al. (2014) developed a mobile-device-based SMS filtering algorithm and introduced the Value Ratio (VR) metric to measure filtering efficiency and processing speed. Similarly, Mujtaba and Yasin (2014) implemented a mobile station-based spam detection approach that analyses each message using four extracted features, with the Naive Bayes classifier outperforming both Artificial Neural Network and Decision Tree models.

Shirani-Mehr (2013) applied several machine learning models to a real SMS spam dataset from the UCI Machine Learning repository, employing multinomial Naive Bayes with Laplace smoothing and linear SVM, which led to a more than 50% reduction in overall error compared to prior studies. Ahmed et al. (2015) developed a semi-supervised learning framework that leveraged frequent itemset extraction via the Apriori algorithm combined with ensemble learning using multinomial Naive Bayes, Random Forest, and LibSVM as base learners. The approach demonstrated consistent performance despite small positive sample sizes and varying proportions of unlabeled data.



This is the measure of the amount of time an algorithm takes to execute

Figure 1: Design framework for the study

3.0 Data presentation

Data preparation and cleaning

This is the initial stage of the experiment. In this phase, the dataset is prepared, cleaned, and made ready for analysis. The data may contain various irrelevant elements and missing information. To address these issues, data cleaning is performed, which includes managing missing data, eliminating noisy data, and other related tasks.

Selection of Dataset:

A dataset refers to a collection of data gathered from a primary source, typically structured as a single database file or a statistical matrix. In this structure, each column denotes a specific variable, while each row represents a unique member or observation within the dataset.

	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 2: SMS spam filter Dataset

4.0 Results and Discussion

The performance metrics were developed using Python and its libraries, allowing for the identification of the best classifier by comparing the classification rates of the models based on accuracy, precision, recall, mean absolute error, and time efficiency. The dataset utilised for this study was collected from the internet, and Jupyter Notebook served as the rendering environment for the analysis.

Visualisation of the Spam and Ham

The distribution of ham and spam messages can be visualised using a count plot. In this case, the sns.countplot() function from the Seaborn library is used to plot the frequency of each message type, with the x-axis representing the label column of the dataset. The plt.show() command from Matplotlib is used to display the resulting plot, providing a clear overview of the class distribution.

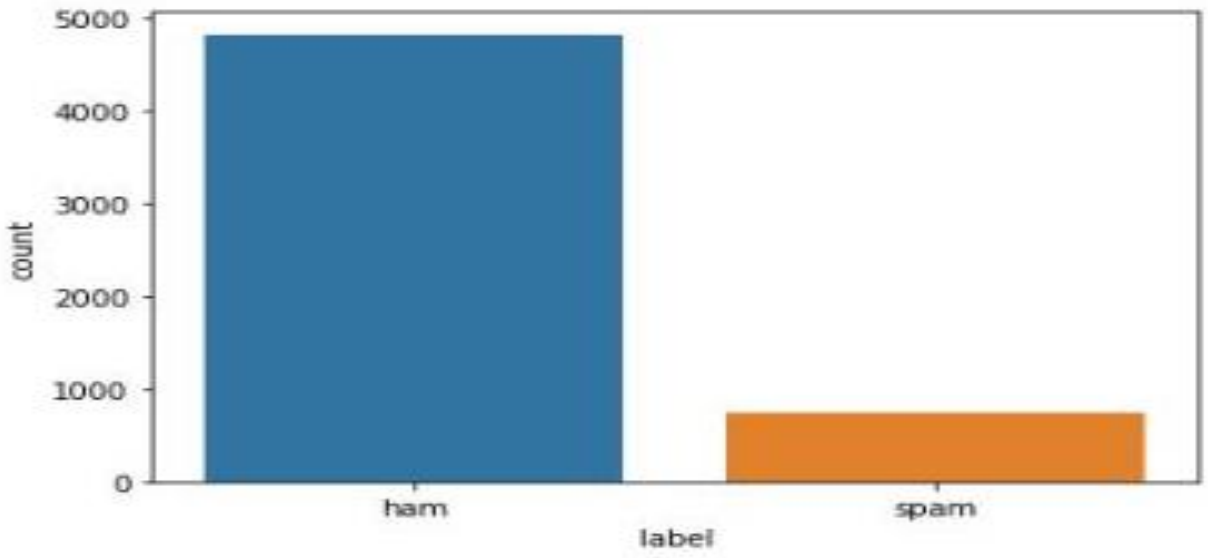


Figure 3: Visualisation of Ham and Spam

The dataset contains a higher proportion of ham messages compared to spam messages, which is expected in typical SMS data. Since embeddings will be employed in the deep learning model, balancing the dataset is not strictly necessary. The next step involves calculating the average number of words per message across the SMS dataset to better understand message length and structure.

Building the models

Initially, a baseline model will be developed to establish a reference performance level, which subsequent deep learning models—such as those using embeddings and LSTM networks—will aim to surpass. For the baseline, the Multinomial Naive Bayes (MultinomialNB) classifier is selected, as it is well-suited for text classification tasks where features are discrete, such as word counts or term frequency-inverse document frequency (TF-IDF) vectors. TF-IDF is a statistical measure that quantifies the importance or relevance of a word within a document relative to the entire dataset.

Performance of baseline model

```
nb_accuracy = accuracy_score(y_test,baseline_model.predict(X_test_vec))
print(nb_accuracy)
print(classification_report(y_test, baseline_model.predict(X_test_vec)))
```

```
0.9623318385650225
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       965
           1       1.00      0.72      0.84       150

    accuracy                           0.96      1115
   macro avg       0.98      0.86      0.91      1115
weighted avg       0.96      0.96      0.96      1115
```

Figure 4: Performance evaluation metrics

Baseline model confusion matrix

```
from sklearn.metrics import plot_confusion_matrix

plot_confusion_matrix(baseline_model, X_test_vec, y_test)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f6e957a7e50>
```

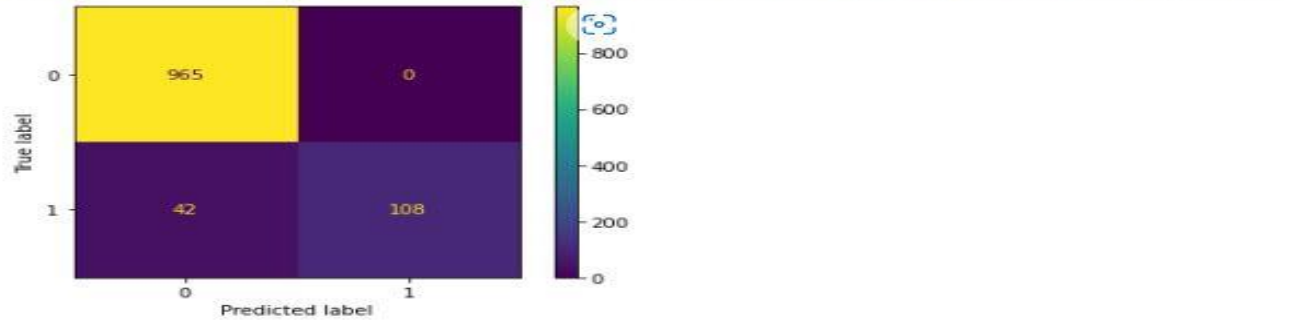


Figure 5: Base model confusion matrix

Designing custom text vectorisation and embedding layers.

Text vectorisation involves transforming text into a numerical format. Examples of this process include methods such as Bag of Words frequency and Binary Term frequency.

Word embedding, on the other hand, is a learned representation of text where words with similar meanings are represented by similar vectors. Each word is associated with a unique vector, and the values of these vectors are learned through a process similar to that of a neural network.

MAXTOKENS: is the maximum size of the vocabulary which was found earlier

OUTPUTLEN: is the length to which the sentences should be padded, irrespective of the sentence length.

Sample code in words.

The embedding layer is defined using layers. Embedding with the following parameters: input_dim, representing the size of the vocabulary; output_dim, specifying the dimension of the embedding vectors; and input_length, indicating the length of the input sequences. This layer transforms each word into a dense vector representation, capturing semantic relationships.

Subsequently, Model 1 is constructed and compiled using the TensorFlow Functional API. An input layer accepts string sequences of shape (1,), which are passed through a custom text vectorisation layer. The resulting vectors are then processed by the embedding layer. A Global Average Pooling 1D layer is applied to reduce the sequence dimension, followed by flattening and a dense layer with 32 neurons using ReLU activation. The output layer consists of a single neuron with a sigmoid activation function for binary classification. The model is compiled with the Adam optimiser, binary cross-entropy loss with label smoothing set to 0.5, and accuracy as the evaluation metric.

Summary of the model 1

```
model_1.summary()

Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 1)]               0

 text_vectorization (TextVec  (None, 15)               0
 torization)

 embedding (Embedding)       (None, 15, 128)           1994880

 global_average_pooling1d (G  (None, 128)              0
 lobalAveragePooling1D)

 flatten (Flatten)           (None, 128)               0

 dense (Dense)               (None, 32)                4128

 dense_1 (Dense)             (None, 1)                 33

=================================================================
Total params: 1,999,041
Trainable params: 1,999,041
Non-trainable params: 0
_________________________________________________________________
```

Figure 6: Summary of Model 1 using TensorFlow

```
history_1 = model_1.fit(X_train,y_train,epochs=5,validation_data=(X_test,y_test),validation_steps=int(0.2*len(X_test)))

Epoch 1/5
140/140 [==============================] - 5s 10ms/step - loss: 0.6037 - accuracy: 0.9132 - val_loss: 0.5767 - val_accuracy: 0.9758
Epoch 2/5
140/140 [==============================] - 1s 9ms/step - loss: 0.5700 - accuracy: 0.9877 - val_loss: 0.5733 - val_accuracy: 0.9812
Epoch 3/5
140/140 [==============================] - 1s 7ms/step - loss: 0.5657 - accuracy: 0.9955 - val_loss: 0.5724 - val_accuracy: 0.9839
Epoch 4/5
140/140 [==============================] - 1s 9ms/step - loss: 0.5642 - accuracy: 0.9982 - val_loss: 0.5724 - val_accuracy: 0.9839
Epoch 5/5
140/140 [==============================] - 1s 7ms/step - loss: 0.5634 - accuracy: 0.9989 - val_loss: 0.5724 - val_accuracy: 0.9803
```

Figure 7: Training the model

```
pd.DataFrame(history_1.history).plot()

<matplotlib.axes._subplots.AxesSubplot at 0x7f0588d8bc90>
```

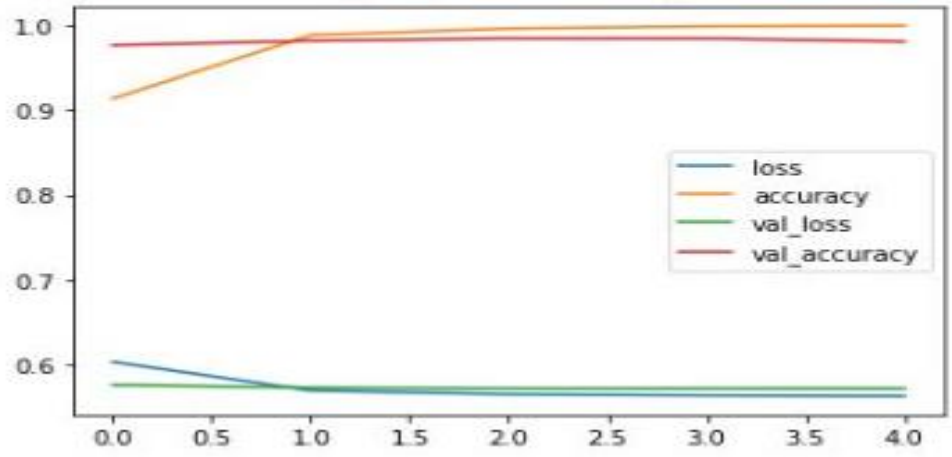


Figure 8: Plotting the model using TensorFlow

Model -2 Bidirectional LSTM

A bidirectional Long Short-Term Memory (BiLSTM) network comprises two LSTM layers, one processing sequences forward and the other backwards. This architecture allows the network to capture context from both preceding and succeeding words, improving the model's understanding of the sequence and enhancing overall performance.

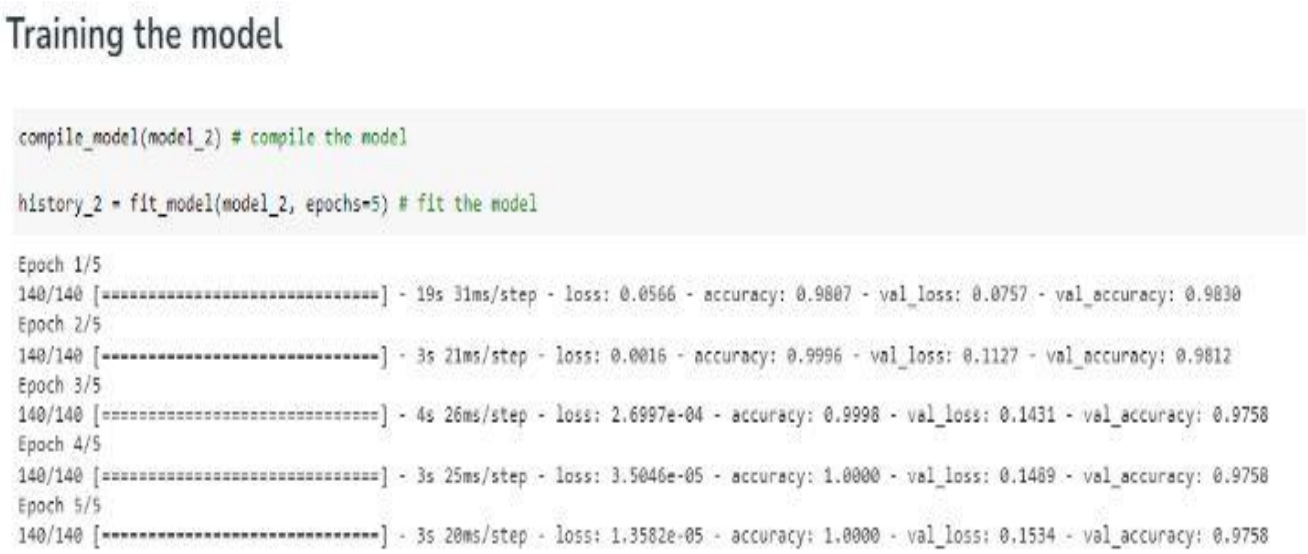


Figure 9: Training the model 2-Bidirectional LSTM

Model-3 Transfer Learning with Universal Sentence Encoder (USE) Encoder

Transfer learning is a machine learning strategy in which a model developed for a specific task is repurposed as the starting point for a different but related task. The Universal Sentence Encoder (USE) transforms text into high-dimensional vector representations that can be applied to text classification, semantic similarity, and other natural language processing tasks. The USE is accessible via TensorFlow Hub and can be integrated into models as a layer using the KerasLayer () function.

Sample code:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import tensorflow_hub as hub
# Define the Sequential model
model_3 = keras.Sequential ()
# Add the Universal Sentence Encoder layer from TensorFlow Hub
use_layer = hub.KerasLayer(
   "https://tfhub.dev/google/universal-sentence-encoder/4",
   Trainable = False,     # Freeze the USE weights
   input_shape = [],      # Input is a 1D string tensor
   dtype = tf.string,
   name='USE'
```

Compiling and training the model

```
compile_model(model_3)

history_3 = fit_model(model_3, epochs=5)

Epoch 1/5
140/140 [==============================] - 7s 27ms/step - loss: 0.3044 - accuracy: 0.9096 - val_loss: 0.1164 - val_accuracy: 0.9695
Epoch 2/5
140/140 [==============================] - 3s 24ms/step - loss: 0.0822 - accuracy: 0.9789 - val_loss: 0.0712 - val_accuracy: 0.9767
Epoch 3/5
140/140 [==============================] - 3s 24ms/step - loss: 0.0580 - accuracy: 0.9834 - val_loss: 0.0592 - val_accuracy: 0.9803
Epoch 4/5
140/140 [==============================] - 3s 23ms/step - loss: 0.0460 - accuracy: 0.9861 - val_loss: 0.0539 - val_accuracy: 0.9821
Epoch 5/5
140/140 [==============================] - 3s 23ms/step - loss: 0.0410 - accuracy: 0.9874 - val_loss: 0.0520 - val_accuracy: 0.9812
```

Training model with USE

Figure 10: Training and Compiling the USE Transferring Model

Analysing our Model Performance

We use the helper function created earlier to evaluate the model's performance.

	accuracy	precision	recall	f1-score
MultinomialNB Model	0.962332	1.000000	0.720000	0.837209
Custom-Vec-Embedding Model	0.981166	0.970803	0.886667	0.926829
Bidirectional-LSTM Model	0.975785	0.936170	0.880000	0.907216
USE-Transfer learning Model	0.981166	0.944828	0.913333	0.928814

Figure 11: Performance Model

Plotting the results

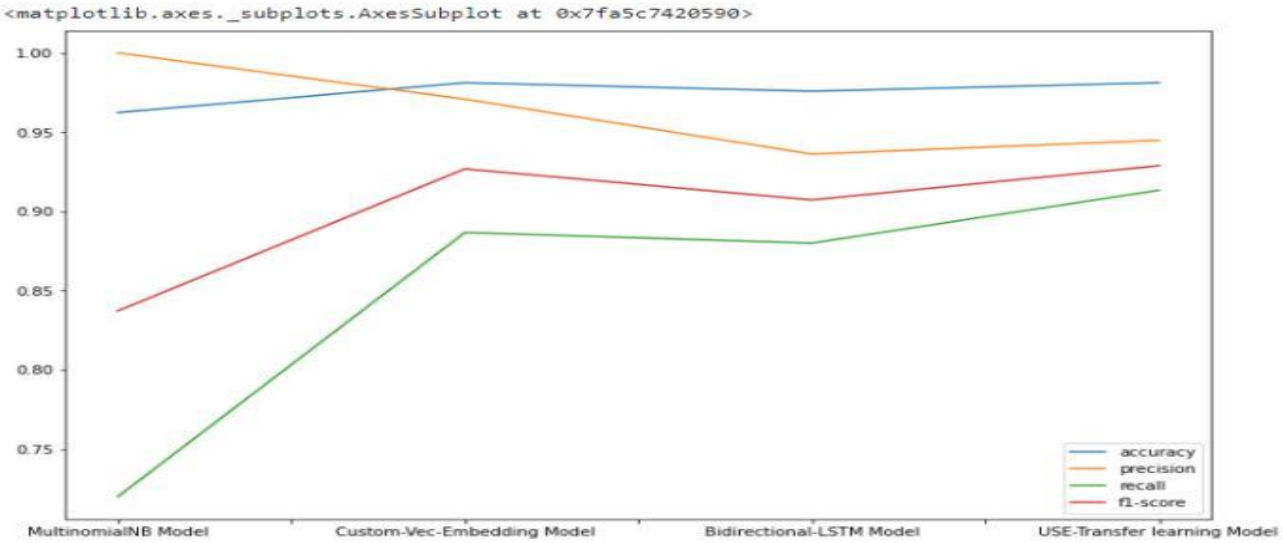


Figure 12: Result of the Plotting against all the models

All four models yield impressive results, each achieving over 96 per cent accuracy, making direct comparisons challenging. Our dataset is unbalanced, with the majority of data points labelled as "ham," which is expected since most SMS messages are indeed ham. In certain scenarios, accuracy alone may not be a suitable metric, necessitating the use of additional measurements.

Addressing false positives and false negatives is critical for model evaluation. Precision and recall provide insight into these errors, while the F1-score, representing their harmonic mean, enables simultaneous assessment. Among the models considered, the Universal Sentence Encoder (USE) transfer learning model achieved the highest performance in terms of accuracy and F1-score.

5.0 Results

The performance of the proposed SMS spam filtering models was evaluated using accuracy, precision, recall, F1-score, and confusion matrix metrics. The dataset contained both spam and ham messages, with ham messages being the majority class.

Baseline Model (Multinomial Naïve Bayes):

Served as the benchmark for comparison.

Achieved high accuracy in differentiating spam from ham messages but showed limitations in handling class imbalance.

Bidirectional LSTM Model:

Improved performance compared to the baseline model by capturing contextual dependencies from both past and future words.

Demonstrated better generalisation in detecting subtle spam patterns.

Transfer Learning with Universal Sentence Encoder (USE):

Delivered the best overall results among all models.

Achieved over **96% accuracy** with a high F1-score, indicating balanced performance across precision and recall.

Showed robustness against false positives and false negatives.

Comparative Analysis:

All models achieved accuracy above 96%, but the USE-based transfer learning model outperformed others in terms of both accuracy and F1-score.

Visualisation of spam vs. ham distribution confirmed the dataset's imbalance, which justified the need for robust evaluation beyond accuracy alone.

6.0 Conclusion and Recommendations

The SMS spam filtering project implemented in Python highlights the critical importance of addressing false negatives and false positives. Precision and recall serve as key metrics for evaluating these errors, while the F1-score, calculated as the harmonic mean of precision and recall, provides a unified measure for assessing both. Among the models evaluated, the Universal Sentence Encoder (USE) transfer learning model achieved the highest accuracy and

This research supports the use of a Python-based SMS spam filtering system, which has shown strong performance in accurately detecting spam content. Additionally, we suggest that this study be beneficial for users who receive unsolicited messages from schools, banks, and other financial institutions. Among all the algorithms tested, the USE-Transfer learning model stands out as the most effective approach.

References

Ahmed, K., Khan, M. U., & Ahmad, S. (2014). A hybrid system of SMS classification using Naïve Bayes and Apriori algorithm. *International Journal of Computer Applications, 98*(5), 1–5.

Ahmed, K., Khan, M. U., & Ahmad, S. (2015). Semi-supervised learning for SMS spam filtering using frequent itemset and ensemble learning. *International Journal of Advanced Computer Science and Applications, 6*(9), 1–6.

Almeida, T. A., Hidalgo, J. M. G., & Yamakami, A. (2011). Contributions to the study of SMS spam filtering: New collection and results. *Proceedings of the 11th ACM Symposium on Document Engineering*, 259–262.

Cer, D., Yang, Y., Kong, S. Y., et al. (2018). Universal Sentence Encoder. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 169–174.

Cormack, G. V., Hidalgo, J. M. G., & Sanz, E. P. (2017). Feature-based and compression-model-based spam filters. Information Processing & Management, 43(3), 1034–1045.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.

McCallum, A., & Nigam, K. (1998). A comparison of event models for Naïve Bayes text classification. AAAI Workshop on Learning for Text Categorization, 41–48.

Nagwani, N. K., & Sharaff, A. (2017). SMS classification based on Naïve Bayes classifier and Apriori algorithm. *International Journal of Computer Applications, 162*(1), 1–5.

Yadav, M., Agarwal, A., & Yadav, R. (2012). Filtering spam messages using natural language processing and classification. *International Journal of Computer Applications, 52*(5), 7–13.