Empowering SMS Spam Filtering: Neural Networks and Big Data Storage & Processing

*Omprakash Vootla*

*SBS23020*

[*Sbs23020@student.cct.ie*](mailto:Sbs23020@student.cct.ie)

*Master of Science in Data Analytics*

*CCT College, Dublin, Ireland*

***Abstract*:**

***The connection between deep learning and big data opens new possibilities and challenges across various fields. Through exploratory research, breakthroughs like autoencoders, recurrent neural networks, and generative adversarial networks have emerged. These algorithms help in discovering features without supervision, modeling sequences, and recognizing images, respectively, especially when dealing with large datasets. Notably, advancements in both hardware, software, and distributed computing have addressed scalability issues in deep learning models, allowing them to process larger volumes of binary data seamlessly.***

***Short Message Service (SMS) spam is a serious problem across the globe now, because of the availability of very cheap prepaid SMS packages makes it easy for spammers to target mobile users. There are some systems to detect and filter spam messages for English, most of which use machine learning techniques to analyze the content of messages and classify them. There is some research on spam email filtering but very few focused on SMS.***

***In this work we are focused on comparing the performance of machine learning and deep learning models in classifying the SMS Spam messages. Machine learning model Multinomial NB achieved an accuracy of 97.58% and a precision of 99.02%. Deep learning Model RNN with LSTM demonstrated a promising trend in accuracy of 97.37%. RNN BiLSTM model demonstrated even more impressive performance with accuracy of 99.32%.***

***In summary, three models exhibited strong performance in terms of accuracy, precision, and loss, with the RNN BiLSTM model showing slightly superior results, achieving higher accuracy and lower loss compared to the Multinomial NB and RNN LSTM models.***

Keywords—Bigdata, PySpark, Python, RNN, Text Classification

# Introduction

In recent times, the widespread adoption of mobile networks, both in Ireland and globally, has led to a surge in the use of Short Message Service (SMS). Unfortunately, this increase has also fueled the growth of SMS spam, driven by the affordability and high response rates of these messages.

SMS spam has become a significant concern worldwide, largely due to the availability of cheap prepaid SMS packages. Reports indicate that there's a rising trend of fake 'Eflow' sites targeting unsuspecting Irish consumers, with up to 10 new scam sites popping up daily, according to the Irish Times. This wave of fake text messages has also affected customers of Bank of Ireland.

The proliferation of spam SMS messages brings various issues for mobile users. Apart from the annoyance, there's a risk of financial loss if users fall for these scams. Responding to these messages may lead to unintended calls to premium rate numbers or unwitting subscriptions to costly services. Additionally, there's a danger of users exposing themselves to harmful websites or downloading malware.

Mobile network operators also face financial repercussions from spam SMS, as they may lose customers or incur additional expenses on spam prevention measures. To combat this problem, several methods and systems have been developed to filter spam messages in English on mobile networks. Many of these systems rely on machine learning techniques to analyze message content and other relevant information for spam detection.

One of the pioneering systems in this field was proposed by Graham, which used a Naive Bayes classifier to achieve commendable results. Currently, the most advanced systems for detecting English spam SMS boast an impressive accuracy rate of 97.5%, with a minimal false positive rate of just 0.2%.

This paper compared the performance of machine learning and deep learning models on text classification. Machine learning model Multinomial NB achieved an accuracy of 97.58% and a precision of 99.02%. Deep learning Model RNN with LSTM demonstrated a promising trend in accuracy of 97.37%. RNN BiLSTM model demonstrated even more impressive performance with accuracy of 99.32%.

GitHub:

<https://github.com/OmVootla/Sem2-BDP-AD-CA1>

or

<https://github.com/OmVootla/MScSem2-BDP_AD-CA1>

## Deep Learning and Bigdata

Deep learning, a subset of machine learning, represents the latest advancement in AI technology. It enables computers to learn independently from vast amounts of data, providing them with the capability to extract valuable insights, identify patterns, and make forecasts. When combined with big data, deep learning algorithms become even more powerful, allowing for the extraction of meaningful insights from large and complex datasets.

The intersection of deep learning and big data signifies a significant development in the fields of AI and data science. It represents a crucial point where cutting-edge deep learning algorithms and the extensive capabilities of big data analytics converge, revolutionizing how organizations utilize data to innovate, gain competitive advantages, and tackle complex challenges across various domains.

Deep learning algorithms, inspired by the hierarchical structure of the human brain, excel in tasks such as image recognition, natural language processing, and speech synthesis. These algorithms leverage dense layers of neural networks to process and understand complex data. However, the effectiveness of deep learning models is heavily reliant on the availability of large-scale datasets for training, which is where big data plays a vital role.

Big data, characterized by its volume, velocity, and variety, provides ample opportunities to train deep neural networks. It encompasses diverse sources of data, including sensor data from modern devices like IoT devices, social media interactions, transaction records, and medical imaging scans. By leveraging big data infrastructure, organizations can gather, process, and analyze massive datasets, laying the foundation for AI and deep learning models capable of addressing complex real-world problems effectively.

# OBJECTIVES

## Technical Objectives

The main goal of this project is to define the progress and technologies of deep learning using big data. By examining the effectiveness of Recurrent Neural Networks (RNNs) employing LSTM, and BiLSTM methods in classifying SMS spam messages.

This examination occurs within the context of Big Data Storage and Processing utilizing Pyspark.

## Research Question

Compare the RNN LSTM and BiLSTM model performances on SMS spam classification with traditional Machine learning Multinomial Naïve Bayse Model performance.

How do the machine learning model and deep learning RNNs perform on text classification.

## State of Art

The current state of the art in spam detection and sentiment analysis reveals a diverse range of approaches. Ensemble Learning methods, combining multiple machine learning models, have shown remarkable effectiveness, achieving a top accuracy of 99.91% in classifying SMS spam. Similarly, Kernel Extreme Learning Machines (KELM) demonstrated high accuracy across different datasets, reaching an AUC of up to 0.9699. In Twitter spam detection, models leveraging deep learning techniques such as LSTM achieved accuracies exceeding 97%. For email spam identification, deep learning approaches like RNN and BLSTM, coupled with regularization techniques, proved superior. Additionally, hybrid models like CNN-LSTM displayed excellent performance, outperforming other methods with high precision and recall rates. The exploration of correlation algorithms, particularly AdaBoost, has shown promise in phishing SMS detection. Furthermore, deep learning models like BiLSTM exhibited impressive accuracies of 93.4% to 98.6% across different datasets. These advancements underscore the growing effectiveness and versatility of machine learning and deep learning techniques in tackling the evolving challenges of spam detection and sentiment analysis.

# TECHNOLOGIES USED

## Reasoning Behind Choosing PySpark:

We are using PySpark as our platform because of its impressive ability to handle vast amounts of text data effortlessly. PySpark stands out in this regard by leveraging parallel processing and distributed computing, allowing it to handle large datasets with ease. Additionally, its seamless integration with Python, comprehensive built-in machine learning tools, efficient resource management, and robust community support make it ideal for our project. With PySpark, we're confident that our analysis of SMS messages will scale smoothly and be manageable.

## Reasoning Behind Choosing Recurrent Neural Networks (RNNs):

We chosen recurrent neural networks (RNNs) because they excel at capturing the sequential structure of text. Unlike standard neural networks, which treat each input in isolation, RNNs possess memory that aids in understanding the context and sentiment progression within a sentence. This memory enables them to retain information from previous words while processing next one. This feature is particularly valuable for tasks such as analyzing SMS messages, where comprehending the conversation flow and identifying patterns over time are crucial for identifying spam in lengthy or context-dependent messages.

# RESEARCH METHODOLOGY

To study deeper level between Deep Learning employing Big Data, researchers followed a systematic way, therefore, they compiled a collation of recently published research papers from well-known conferences and journals. Applying the galore of libraries like TensorFlow, Poarch, and Keras for Python, a known powerhouse for its flexibility and robustness allowed for experimentation, model implementations and prototyping. This methodology made sure that the research process included a multidisciplinary approach, which involved exploration of the latest findings, methods, and key concepts in data science. The Python language has made deep inroads in the field of data science and artificial intelligence because it is easy to learn, has an extensive list of libraries and enjoys the support of a large cohort of contributors. TensorFlow is perhaps the most widely used deep learning framework, and it is particularly attributed to its wonderful flexibility, tolerance for scalability, and ease with which it is deployed across various platforms including mobile, web, and server. The company has created the most detailed and convenient API to allow researchers to rapidly build and train deep learning models in these distributed environments, leveraging large scale computing capabilities to process their data.

Other than Keras which was developed in Python and has become popular as it is a highly accessible deep learning framework that other researchers can easily use, PyTorch, an open-source deep learning framework that is developed and maintained by Facebook's AI Research Lab (FAIR) has also emerged as another deep learning framework that is used widely. The above machine learning framework PyTorch possesses an imperative programming style which leads to more flexibility opportunities into the inventive model development and experimentation contexts, as such, this framework is most useful for researchers wishing to disclose new architectures and algorithms, concerning Deep Learning using massive data sets. Keras is a high-level API, which has been created for building and training neural networks in Python and then integrating them to TensorFlow and other deep learning technologies. Keras' interface is user-friendly, which makes it easy to use it. The simplicity and modularity of it make it the perfect instrument for quick prototyping and trial and error experimentations, clinching researchers to adjust models and hyperparameter in a timely manner.

The utilization of Python and its associated libraries facilitated various stages of the research process: The utilization of Python and its associated libraries facilitated various stages of the research process:

Data Collection: Investigators conducted giant studies by finding articles from top journals and conferences using Python for analysis, data pulling and organizing. The Python library enabled me to retrieve articles from sources selected automatically through a search using search criteria I already defined earlier, being also eligible for the scope and depth of the set I am going to analyze.

Data Analysis: Many libraries of Python are remarkable like Pandas, NumPy and Matplotlib for managing data and it was really the main tool while inspecting and analyzing texts from the collected papers. Researchers performed data analysis to start with, which will help them to discover the general trends, patterns, and key themes in the existing literature. This information will enable them to explore state-of-the-art deep learning using big data.

Model Implementation: Researchers implemented deep learning processes via TensorFlow, PyTorch and Keras to achieve replication or development of more generalized methodologies existing in the reviewed literature. Thanks to the fact that working with Python is a much simpler matter than other common language choices as well as the vast ecosystem of pre-trained models, developers are not restrained by traditional time limits and are free to experiment with their own ideas which can all be tried out reliably on big data.

Experimentation: Python's flexibility and scalability prompted us to run all kinds of experiments with the various model topologies, the parameter sets, and the optimizing approaches. The work utilized the parts of the deep learning system in TensorFlow that enabled distributed, high-performance computing for training deep learning models on large-scale dataset in a quick and efficient manner, using GPUs or TPUs so as to accelerate the computation.

Researchers were able to conduct and wide-ranging exploration of the model deployment and experimentation which encompasses various stages of the machine learning pipeline such as data collection and analysis as well as model implementation phase that relied on the robustness and flexibility of Python. Such a method was adopted to provide reliability and replicability in the research process whereby researchers can prove the worthiness of their findings and can add to knowledge in this fast-expanding academic area.

# LITERATURE REVIEW

In a study by [1], a method employing Ensemble Learning was utilized, amalgamating four machine learning models into a single cohesive unit. This integrated approach significantly outperformed the individual models, demonstrating its effectiveness in spam message classification and detection. Using the SMS Spam Collection Dataset, the model achieved a state-of-the-art accuracy of 99.91%.

Another study by [2] focused on sentiment classification of SMS messages using a Kernel Extreme Learning Machine (KELM) classifier. The evaluation criteria included accuracy, recall, f-measure, precision, RMSE, and MAE, with experiments conducted on SMS, Email, and spam-assassin datasets. The proposed architecture achieved an AUC of 0.9699 (SMS dataset), 0.958 (Email dataset), and 0.95(spam assassin).

In the realm of Twitter spam detection and sentiment analysis, [3] proposed a model capable of extracting information from tweets, identifying spam, and assigning sentiment. Utilizing vectorizers like TF-IDF and the Bag of Words model, features were extracted and passed through classifiers, including multinomial Naïve Bayes and various deep learning models such as LSTM. The classification accuracy reached 97.78% with Naïve Bayes and 98.74% with LSTM.

[4] introduced a method employing deep learning approaches, including RNN, LSTM, and BLSTM, for email spam identification. Regularization and dropout layers were utilized to mitigate overfitting, with the Bi-LSTM model demonstrating superior accuracy compared to existing systems.

Using TensorFlow and Keres frameworks, [5] constructed a Deep Machine Learning Algorithm for SMS spam classification, achieving an accuracy of 99.82%. The model consisted of three dense layers with 8672 inputs and utilized a batch size of 32 and 50 epochs.

A hybrid CNN-LSTM model proposed by [6] for SMS spam detection integrated various classifiers such as SVM, KNN, and Naïve Bayes. The CNN-LSTM model outperformed other techniques with an accuracy of 98.37%, precision of 95.39%, recall of 87.87%, F1-Score of 91.48%, and AUC of 93.7%.

[7] explored the detection of phishing SMS using multiple correlation algorithms, with AdaBoost classifier yielding the best accuracy. The Kendall rank correlation algorithm proved superior in feature reduction, achieving 61.53% dimensionality reduction with an accuracy of 98.40%.

Lastly, [8] presented a deep learning model based on BiLSTM, achieving 93.4% accuracy on the ExAIS\_SMS dataset and 98.6% on UCI datasets. Cross-validation was employed for evaluation, demonstrating consistent performance across seven classifiers, with fine-tuning improving accuracy from 90.8% to 93.4%.

# RESEARCH Gap

For the above literature review we found valuable insights of spam detection, particularly in SMS, email, and social media platforms like Twitter. These studies have produced higher spam detection accuracy, there are still some gaps in the research that warrant explored.

One gap is real-time spam detection. Most studies have focused on analysing static datasets, where the entire dataset is available for model training and testing. But in real-world scenarios spam messages generated dynamically necessitating systems that can adapt and learn real time. Research in this area could drive into new techniques for online learning, creating new systems to keep track of spam messages.

Another gap is lack of emphasis on “cultural and linguistic” diversity of spam messages. Many studies have primarily focused on English-language datasets, overlooking the challenges posed by multilingual environments and diverse cultural communication norms. Exploring methods to develop language-based spam detection models or adapting existing models to different linguistic contexts could significantly enhance the efficacy of spam detection systems globally.

Existing studies have primarily focused on binary classification tasks like spam vs. non-spam. There is room for exploring more classification tasks. example identifying different types of spam like phishing, advertising, malware within spam messages could provide deeper insights into spamming tactics. Developing models capable of handling such classifications could greatly improve the precision and accuracy of spam detection systems.

Literature reviews show that several deep learning models have been covered, such as RNN, LSTM, BLSTM, and CNN models. But these deep learning models achieved high accuracy, but there is lack of interpretation of those model results. This making it challenging to understand and hard on decision-making process.

Key Papers Reviewed:

|  |  |
| --- | --- |
| Author(s) | Title |
| Abdallah Ghourabi, Manar Alohaly | Enhancing Spam Message Classification and Detection Using Transformer-Based Embedding and Ensemble Learning. |
| Ulligaddala Srinivasarao, Aakanksha Sharaff | SMS sentiment classification using an evolutionary optimization based fuzzy recurrent neural network. |
| Anisha P Rodrigues, Roshan Fernandes, Aakash A, Abhishek B, Adarsh Shetty, Atul K, Kuruva Lakshmanna, R. Mahammad Shafi | Real-Time Twitter Spam Detection and Sentiment Analysis using Machine Learning and Deep Learning Techniques. |
| Anshuman Mishra, Vigneshwaran Pandia | Classifications of E-MAIL SPAM Using Deep Learning Approaches. |
| J. Palimote, V.I.E Anireh, N.D Nwiabu | A Model for Filtering Spam SMS Using Deep Machine Learning Technique. |
| Abdallah Ghourabi, Mahmood A. Mahmood, Qusay M. Alzubi | A Hybrid CNN-LSTM Model for SMS Spam Detection in Arabic and English Messages. |
| Guni khan Sonowal | Detecting Phishing SMS Based on Multiple Correlation Algorithms. |
| Olusola Abayomi-Alli, Sanjay Misra, Adebayo Abayomi-Alli, | A deep learning method for automatic SMS spam classification: Performance of learning algorithms on indigenous dataset. |

# CRITICAL EVALUATION

## Dataset:

This dataset is a collection of SMS messages from various sources publicly available on UCI Machine learning repositories. Dataset has 5574 instances with 5 columns of spam and ham messages.

[https://archive.ics.uci.edu/dataset/228/sms+spam+collection]

## *Data understanding using pyspark.*

Firstly, we import Spark Session from pyspark.sql and create a Spark Session named “SMS Spam Classification”. We load the csv file into the Data frame (df) enabling spark to infer the schema from the data. Figure 1 shows 5574 records in data frame.

In the Data cleaning process, we dropped unwanted columns and rename remaining two columns “v1” as “target” and “v2” as “text”. categorical labels in target columns where “ham” is replaced with 0 and “spam” replaced with 1.

Dropped 403 duplicates and 1 null value from the data frame.

## *EDA on Pandas Data frame*

To perform Exploratory data analysis, we convert Py Spark Data frame to Pandas Data frame.

As a first task in EDA, we found the 4516 “ham” and 655 “spam” messages in our data frame. We have visualized this using a pie chart as shown below with percentage. Figure 1 shows nearly 88% of “ham” messages and 12% “spam” messages. This indicates that data is imbalanced.

A blue and orange pie chart

Description automatically generated

Figure 1: Percentages of ham and spam messages in text column.

Using “nltk” library we created three new columns “num\_characters”, “num\_words”, and “num\_sentenses” from “text” column.

* to get the number of characters we applied “len” function on text column.
* To get the number of words we applied “nltk. word\_tokenize” function to break SMS messages into words.
* To get the number of sentences we applied “nltk. sent\_tokenize” function to break SMS messages into words.

Describe function illustrating “ham” SMS messages with maximum of 910 characters, 220 words and 38 sentences. And “spam” SMS messages with 223 characters, 46 words, 9 sentences. “ham” messages average of 70 characters, 17 words and 2 sentences. “spam” messages average of 138 characters, 27 words and 3 sentences and average of “spam” message.

A screen shot of a computer

Description automatically generated

Figure 2: Histogram showing number of characters in “ham” and “spam” messages.

A screen shot of a graph

Description automatically generated

Figure 3: Histogram showing number of words in “ham” and “spam” messages.

Figure 2 and 3 clearly shows, that “spam” messages have larger number of characters, words.

A screenshot of a computer screen

Description automatically generated

Figure 4: pair plot showing co linearity between the data.

Figure 4 displays, number of characters and number of sentences are roughly linear, with increase of characters sentences are increasing. There are some outliers in the data.

A screenshot of a computer

Description automatically generated

Figure 5: Heatmap showing correlation between the data.

Figure 5 displays the number of characters is 0.38 % correlated to target which means, with increasing characters there is tendency to become “spam” message. Same as with number of words 0.26 and number of sentences 0.27. If we see the correlation between the values, number of sentences are 0.63 percent correlated to number of characters, and number of words 0.97 percent correlated with number of characters. The number of characters has the most correlation with target.

## *Data Preprocessing*

We need to preprocess the text data for future modelling purposes, for this we apply the steps below.

* Lower Case

With the use of a function “text. Lower” we converted the “text” messages into lower case.

* Tokenization

With “nltk. word\_tokenize” method we break the sentences into words.

* Removing special characters

With “isalnum” method we removed any special characters in text messages.

* Removing stop words and punctuation

With “stop words” and “spring punctuation” we removed stop words and punctuation.

* Stemming

With “Porter Stemmer” library we stem all the “text” messages.

We have created a new column “transformed\_text” applying above function.

### *Spam Word cloud.*

A screenshot of a computer screen

Description automatically generated

Figure 6: Word cloud display of words from “spam” messages.

### *ham word cloud.*

A screenshot of a computer screen

Description automatically generated

Figure 7: Word cloud display of words from “ham” messages.

Figures 6,7 display the top 50 words form “spam” and “ham” messages.

### *Top 30 words in “spam” and “ham” messages:*

For this we have created a list called “spam\_corpus”. Spam corpus has 9952 words. Ham corpus has 35345 words accordingly. Using counter library form collections, we can find the top 30 words and display them using bar plots shown below.

A screenshot of a graph

Description automatically generated

Figure 8: Top 30 words from spam messages.

## *Machine Learning Model Building*

Before going to machine learning model building, we need to convert the text in to vectors or vectorize the text. We can do this in three ways,

* Bag of words, where you create a column with frequent words to find the number of times these frequent words are in each text message.
* TF-IDF converts text documents into numerical vectors while considering the importance of words.
* Word to Vec method transforms words into high-dimensional vectors, capturing semantic similarities and relationships.

For our Modelling we initialize a TF-IDF (Term Frequency-Inverse Document Frequency) Vectorizer, which converts text documents into numerical vectors while considering the importance of words. TF-IDF applied limiting the number of words to 3000.

Now, we fit TF-IDF on X variable “transformed text” from data frame. This will convert the sparse matrix to a dense array. Shape of X is (5170, 3000) where each row represents a message, and each column represents a TF-IDF score for a specific word.

A screenshot of a computer program

Description automatically generated

Figure 9: Multinomial NB model with results.

A blue and green squares

Description automatically generated

Figure 10: Multinomial NB model with results

The data is split into training and testing sets using the train\_test\_split function from scikit-learn. 80% of the data is used for training and 20% for testing.

Initialise Multinomial Naïve Bayes classifier and fit the model and trained on tarin data set. The trained classifier is used to make predictions on the test data set.

For evaluation of the model accuracy, confusion matrix and precision score are computed. Multinomial Naïve Bayes classifier Model achieved accuracy of 97.58% and 99% of precision score. From confusion matrix we can see that this model only gives 1 false positive score. We can find this model as the best model based on the precision score; accuracy is not best, but this doesn’t matter as the data is imbalanced in this dataset.

## *Deep Learning Model Building*

1. We have applied the Deep learning Recurrent Neural Network (RNN) model to classify the SMS spam messages. We have taken “transformed text” as X and “target” variable as y from the pre-processed Data set. Target variables are encoded using label encode to transform from categorical labels into numerical format. Data set is split into train and test sets with 80-20 split.

A tokenizer is utilized to convert text data into sequences of integers, with a specified vocabulary limit of 3000 words and each input sequence padded to a consistent length of 150 to ensure uniformity. The RNN model is constructed using a sequential layering approach, starting with an embedding layer to create dense vector representations of words. It then incorporates an LSTM layer with 64 units to capture temporal dependencies within the text. Following this, a dense layer with 128 units and ReLU activation introduces non-linearity, coupled with a dropout layer to reduce overfitting. The output layer consists of a single neuron with a sigmoid activation function, suitable for binary classification tasks.

The model is compiled using the RMSprop optimizer and binary cross entropy as the loss function, focusing on accuracy as the performance metric. It is trained over 10 epochs with a batch size of 128, incorporating early stopping based on validation loss to prevent overfitting. Post-training, the model's performance is evaluated on the test set, Model achieved 98.94% accuracy, 0.043 loss, on train data and 98.10% accuracy, 0.066 loss on test data sets.

2. We have applied Bidirectional Long Short-Term Memory (BiLSTM) model to learn from the sequence data in both forward and backward directions. BiLSTM model architecture is established starting with an embedding layer that transforms words into dense vectors of fixed size. Followed by a Bidirectional LSTM layer with 64 units, The model includes a dense layer with 256 neurons and employs the ReLU activation function to introduce non-linearity, alongside a dropout layer to mitigate the risk of overfitting.

For the output, a dense layer with a single neuron activated by a sigmoid function and compiled with the binary cross entropy loss function and RMSprop optimizer, emphasizing accuracy as the primary metric.

Trained the model with 10 epochs with 128 batch size. Model achieved 99.7% accuracy with 0.014 loss on train data and 98.4$ accuracy with 0.066 loss on test data.

A graph with a line graph

Description automatically generated with medium confidence

Figure 11: RNN LSTM and RNN BiLSTM Models Accuracy plot

A graph of loss over epops

Description automatically generated

Figure 12: RNN LSTM and RNN BiLSTM Models Loss plot.

## *Implementing Text Classification with PySpark*

To perform text classification with pyspark we have imported required pyspark libraries as shown in Figure 13.

A screenshot of a computer

Description automatically generated

Figure 13: importing pyspark libraries.

We then import “Spam.csv” file as spark data frame.

A screenshot of a computer program

Description automatically generated

Figure 14: sample data frame.

After dropping unnecessary columns and renaming the remaining ones for clarity, it filters out rows with null values in the "text" column and tokenizes the text, splitting it into individual words. We converted labels to binary values with 0 for "ham" and 1 for "spam".

As part of data preprocessing, we train Word2Vec model to convert the words into vectors with vectorize 200 and minCount = 6. On input column words and output column wordVectors. We Transform the Data Frame to add word vectors. We Use Vector Assembler to assemble word vectors into a single feature vector.

A screenshot of a computer program

Description automatically generated

Figure 15: Display of data frame after preprocessing

The dataset is split into training and testing sets, with 80% of the data used for training and 20% for testing. A Multilayer Perceptron (MLP) classifier is trained on the training data to predict whether a message is spam or ham based on its features.

Due to dimension mismatch problem in Multilayer Perceptron (MLP) classification model not able to evaluate the model accuracy score.

## Critical Evaluation of Key Findings

### a). Implications of Key Findings

Machine Learning vs. Deep Learning Performance:

The script executes the tasks for both Multinomial Naive Bayes, a transaction machine learning model, and Recurrent Neural Networks (RNNs) and Bidirectional Long Short-Term Memory (LSTMs) for SMS spam classification. With the implication that these models, the deep learning ones, may offer higher level of accuracy and more complex representations, nevertheless, the traditional machine learning algorithms, due to their simplicity at most, still have the capacity to deliver acceptable outcomes and lower computational requirement could be applied.

Text Preprocessing Impact: Text preprocessing which consists of tokenization, special characters’ removal and stemming are critical components in model performance. This process is for standardizing the text data and eliminating noise. By doing this the model will better understand any unseen data.

Model Interpretability: Deep neural networks provide more accurate results; their transparency is an issue which becomes an obstacle for future understanding. While on one hand, ML algorithms such as Multinomial Naive Bayes are transparent in their operations, letting users know the decision path through the model.

### b). Limitations and Contradicting Viewpoints

Imbalanced Dataset: No matter how dedicated the script is to reducing class imbalance, it might not explicitly indicate such a thing. Although the accuracy of a query may be justified by the statistics, they may be misleading as the distribution between spam and non-spam messages is not homogeneous. Ignoring the issue of class imbalance might result in poor model evaluation that can misdirect the analysis and the interpretation of the results.

Evaluation Metrics: Assessing personalities with accuracy could not be the best way for datasets that are prone to imbalances. The metrics of precision, recall, and F1, which are assumed to be more informative especially in the cases with different costs of false positive and false negatives, are more reliable options in this scenario. The types of rating the evaluating metrics may distort the perceived effectiveness of the models.

Generalizability: The script does not address the question of how the generated models perform when applied on data sets that are very different or at which the models are not trained at. Although the models may execute the SMS spam dataset well, their practicality to see the distinction in real-world scenarios with different data distributions and SMS characteristics, remains unknown. Such disparity points out the necessity of experiencing and state of the art artificial intelligence across various datasets.

## *Conclusion*

From the above Machine learning and Deep learning model performances it is observed that,

Multinomial NB model achieved an accuracy of 97.58% and a precision of 99.02%. In the confusion matrix, it correctly classified 908 instances of one class and 101 instances of the other, with only 1 misclassification in the former and 24 in the latter. These results indicate a strong performance of the model in both overall accuracy and the ability to correctly identify positive instances.

RNN LSTM model demonstrated a promising trend in accuracy, starting at approximately 79.45% and steadily increasing to around 97.37% by the end of training. Similarly, the validation accuracy began at 87.92% and reached approximately 96.26% by the final epoch. In terms of loss, the RNN LSTM model exhibited a decreasing trend from 0.4876 to 0.0992 for training loss and from 0.2692 to 0.1189 for validation loss.

RNN BiLSTM model demonstrated even more impressive performance. Starting with an accuracy of approximately 97.84%, it soared to around 99.32% by the fourth epoch. Validation accuracy followed a similar trajectory, starting at 97.95% and reaching approximately 98.55%. Loss for the RNN BiLSTM model decreased substantially from 0.0807 to 0.0364 for training loss and from 0.0744 to 0.0592 for validation loss.

As a final step in our modelling process, we also used pyspark to perform some preprocessing of data using Word2Vec model to convert word to vectors. We build a model with Multilayer Perceptron Classifier trained on train data but unfortunately not able evaluate the model accuracy due to dimension mismatch error.

In summary, three models exhibited strong performance in terms of accuracy, precision, and loss, with the RNN BiLSTM model showing slightly superior results, achieving higher accuracy and lower loss compared to the MultinomialNB and RNN LSTM models.

# References

1. [Abdallah Ghourabi](https://pubmed.ncbi.nlm.nih.gov/?term=Ghourabi+A&cauthor_id=37112202), [Manar Alohaly](https://pubmed.ncbi.nlm.nih.gov/?term=Alohaly+M&cauthor_id=37112202) Enhancing Spam Message Classification and Detection Using Transformer-Based Embedding and Ensemble Learning. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10146782/>
2. [Ulligaddala Srinivasarao](https://pubmed.ncbi.nlm.nih.gov/?term=Srinivasarao+U&cauthor_id=37362691), [Aakanksha Sharaff](https://pubmed.ncbi.nlm.nih.gov/?term=Sharaff+A&cauthor_id=37362691) SMS sentiment classification using an evolutionary optimization based fuzzy recurrent neural network.

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10107590/>

1. [Anisha P Rodrigues](https://pubmed.ncbi.nlm.nih.gov/?term=Rodrigues%20AP%5BAuthor%5D), [Roshan Fernandes](https://pubmed.ncbi.nlm.nih.gov/?term=Fernandes%20R%5BAuthor%5D), [Aakash A](https://pubmed.ncbi.nlm.nih.gov/?term=A%20A%5BAuthor%5D), [Abhishek B](https://pubmed.ncbi.nlm.nih.gov/?term=B%20A%5BAuthor%5D), [Adarsh Shetty](https://pubmed.ncbi.nlm.nih.gov/?term=Shetty%20A%5BAuthor%5D), [Atul K](https://pubmed.ncbi.nlm.nih.gov/?term=K%20A%5BAuthor%5D), [Kuruva Lakshmanna](https://pubmed.ncbi.nlm.nih.gov/?term=Lakshmanna%20K%5BAuthor%5D), and [R. Mahammad Shafi](https://pubmed.ncbi.nlm.nih.gov/?term=Shafi%20RM%5BAuthor%5D) Real-Time Twitter Spam Detection and Sentiment Analysis using Machine Learning and Deep Learning Techniques

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9033328/>

1. Anshuman Mishra, Vigneshwaran Pandia Classifications of E-MAIL SPAM Using Deep Learning Approaches. <https://www.researchgate.net/publication/366137901>
2. J. Palimote, V.I.E Anireh, N.D Nwiabu. A Model for Filtering Spam SMS Using Deep Machine Learning Technique.

<https://www.researchgate.net/publication/350877959>

1. Abdallah Ghourabi, Mahmood A. Mahmood, and Qusay M. Alzubi. A Hybrid CNN-LSTM Model for SMS Spam Detection in Arabic and English Messages.

<https://www.mdpi.com/1999-5903/12/9/156>

1. [Guni khan Sonowal](https://pubmed.ncbi.nlm.nih.gov/?term=Sonowal%20G%5BAuthor%5D), Detecting Phishing SMS Based on Multiple Correlation Algorithms.

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7604914/>

1. Olusola Abayomi-Alli, Sanjay Misra, Adebayo Abayomi-Alli, A deep learning method for automatic SMS spam classification: Performance of learning algorithms on indigenous dataset. <https://www.researchgate.net/publication/359673970>
2. Thai-Hoang Pham, Phuong Le-Hong Content-based Approach for Vietnamese Spam SMS Filtering.

<https://arxiv.org/pdf/1705.04003v1.pdf>

1. Amanullah, M.A., Habeeb, R.A.A., Nasaruddin, F.H., Gani, A., Ahmed, E., Nainar, A.S.M., Akim, N.M. and Imran, M., 2020 Deep learning and big data technologies for IoT security.

<https://doi.org/10.1016/j.comcom.2020.01.016>

1. Emmert-Streib, F., Yang, Z., Feng, H., Tripathi, S. and Dehmer, M., 2020. An introductory review of deep learning for prediction models with big data. Frontiers in Artificial Intelligence, 3, p.4.

<https://doi.org/10.3389/frai.2020.00004>

1. Atitallah, S.B., Driss, M., Boulila, W. and Ghézala, H.B., 2020. Leveraging Deep Learning and IoT big data analytics to support the smart cities development: Review and future directions. Computer Science Review, 38, p.100303.

<https://doi.org/10.1016/j.cosrev.2020.100303>

1. Hassan, M.M., Gumaei, A., Alsanad, A., Alrubaian, M. and Fortino, G., 2020. A hybrid deep learning model for efficient intrusion detection in big data environment. Information Sciences, 513, pp.386-396.

<https://doi.org/10.1016/j.ins.2019.10.069>

1. Waheed, H., Hassan, S.U., Aljohani, N.R., Hardman, J., Alelyani, S. and Nawaz, R., 2020. Predicting academic performance of students from VLE big data using deep learning models. Computers in Human behavior, 104, p.106189.

<https://doi.org/10.1016/j.chb.2019.106189>

1. Neethirajan, S., 2020. The role of sensors, big data, and machine learning in modern animal farming. Sensing and Bio-Sensing Research, 29, p.100367.

<https://doi.org/10.1016/j.sbsr.2020.100367>

1. Liu, J., Li, T., Xie, P., Du, S., Teng, F. and Yang, X., 2020. Urban big data fusion based on deep learning: An overview. Information Fusion, 53, pp.123-133.

<https://doi.org/10.1016/j.inffus.2019.06.016>

1. Mai A. Shaaban∗1, Yasser F. Hassan2, and Shawkat K. Guirguis3 Deep convolutional forest: a dynamic deep ensemble approach for spam detection in text.
2. Ning Ding∗, Yujia Qin∗, Guang Yang, Fuchao Wei, Zonghan Yang Delta Tuning: A Comprehensive Study of Parameter Efficient Methods for Pre-trained Language Models.
3. Anastasiia Sedova, Andreas Stephan, Marina Speranskaya, Benjamin Roth Knodle: Modular Weakly Supervised Learning with PyTorch.
4. [Ayush Maheshwari](https://paperswithcode.com/author/ayush-maheshwari), [Oishik Chatterjee](https://paperswithcode.com/author/oishik-chatterjee), [Krishna Teja Killamsetty](https://paperswithcode.com/author/krishnateja-killamsetty), [Ganesh Ramakrishnan](https://paperswithcode.com/author/ganesh-ramakrishnan), [Rishabh Iyer](https://paperswithcode.com/author/rishabh-iyer) Semi-Supervised Data Programming with Subset Selection.
5. Anastasiia Sedova, Lena Zellinger, and Benjamin Roth Learning with Noisy Labels by Adaptive Gradient-Based Outlier Removal