An Empirical Study on SMS and Email Spam Detection using Machine Learning Algorithms

*Note: This study compares and evaluates various machine learning algorithms for the effective detection of SMS and email spam

Debmalya Deb
Msc in Data Analytics (School of Computing)
National College of Ireland
x21242101

***Abstract*—Spam messages, including SMS and email spam, have become a significant issue for businesses and individuals. Machine learning algorithms have been widely used for the detection of spam messages. This paper presents a comparative study of various machine learning algorithms and methodologies for detecting SMS and email spam. The study evaluates the performance of different algorithms, including decision trees, random forest, and naive bayes. The proposed hybrid approach combines multiple machine-learning algorithms to achieve improved accuracy and efficiency in spam detection. The study also utilizes feature engineering and Exploratory data analysis (EDA) techniques to enhance the performance of machine learning models. The experimental results demonstrate that the proposed approach outperforms traditional machine learning algorithms and achieves high accuracy and efficiency in spam detection. The proposed approach can be applied to various spam detection tasks and provides a practical solution for businesses and individuals to combat spam messages.**

***Index Terms*—SMS spam detection, Email spam detection, DecisionTreeClassifier, RandomForestClassifier, Naive Bayes, Sklearn, Xgboost, KDD approach,Exploratory data analysis.**

I. Introduction

In the area of spam detection, a supervised machine learning approach can be used to train models with labelled datasets of spam and non-spam messages. Email providers have access to large labelled datasets and user feedback to improve their spam detection algorithms. However, such algorithms are generally more advanced than the ones discussed in this paper and have safeguards against misuse. [1]. The proliferation of spam in email systems can negatively impact server resources and user productivity. Therefore, there is a critical need to develop efficient spam detection and filtering methods to alleviate the effects of spam and enhance the overall email user experience. [2].The worldwide email traffic is significantly composed of spam emails, causing frustration to users and leading to financial losses due to internet scams. Scammers posing as reputable entities often request sensitive information, highlighting the need for reliable solutions to prevent and detect spam emails and messages. [3].

This case study aims to analyze the effectiveness of three machine learning algorithms, Naive Bayes, DecisiontreeClassifier, and RandomforestClassifier, for spam detection using three datasets from different sources. The study focuses on spam detection for SMS and email, with the latter being a significant concern for individuals and industrial sectors. The goal is to compare the performance of these algorithms on spam detection for both SMS and email, and to provide insights into the use of machine learning for efficient spam detection. The study suggests that a combination of different algorithms may be required for effective spam detection across different platforms, providing valuable insights into the use of machine learning algorithms for spam detection.

II. Related Work

A. Previous Studies on the Volume of Spam Email and Sms

According to the Kaspersky Lab, In 2015, the volume of spam emails reduced significantly, but the decline was short-lived, as spam email volume quadrupled in 2016 compared to the previous year. Malicious attachments such as malware and ransomware also increased towards the end of 2015 and continued to be a problem in 2016. This highlights the need for efficient spam detection and prevention measures to minimize the impact of spam on email infrastructure and protect users. [4].

Bhuiyan et al. [5] presents an overview of current email spam filtering techniques, highlighting their accuracy and efficiency. While all approaches are effective, some have better performance results, and efforts are ongoing to develop next-generation spam filtering mechanisms capable of processing large amounts of multimedia data. Naïve Bayes and SVM algorithms are the primary methods used for email spam filtering, which can be trained on various datasets such as "ECML" and UCI datasets. Despite their success, filtering spam emails remains a primary concern for researchers [6].

B. A Case Study on Extending the Analysis Based on Early Reports

This report is a literature review that draws from various previously researched and analyzed spam detection papers. The analyst of this case study is greatly influenced by the early published techniques of spam detection and attempts to build a machine learning model to analyze the aforementioned datasets of SMS and email ham-spam analysis. The objective is to evaluate the effectiveness of machine learning algorithms

in spam detection, and their performance is compared to the results of previously published works. The study aims to contribute to the development of more accurate and efficient spam detection systems, which can aid in mitigating the menace of spam emails and SMS.

III. Methodology

A. Approach

This report focuses on the KDD methodology, which involves discovering important insights from extensive datasets. The KDD process includes steps such as selecting a relevant subset of data, transforming and cleaning the data, applying data mining algorithms to extract useful information, interpreting the results, evaluating their accuracy, and utilizing the newfound knowledge to solve business problems and make informed decisions. [7].

- **Selection**: Choose a subset of the data that is pertinent for analysis.
- **Pre-processing**: Prepare the data for analysis by removing any inconsistencies, normalizing the data, handling missing values, and integrating relevant data to make it suitable for analysis.
- **Transformation**: Convert the data into a structure that is appropriate for data mining, like a graph or a matrix.
- **Data Mining**: Utilize data mining techniques and algorithms on the data to extract meaningful information and insights, which may involve performing various tasks such as clustering, classification, association rule mining, and identifying anomalies.
- **Interpretation**: Extracting knowledge and interpreting the data involves tasks such as assessing the discovered patterns' quality, identifying relationships and associations among the data, and presenting the results through visualizations.
- **Evaluation**: Assess the outcomes to confirm that the acquired knowledge is practical, precise, and significant.
- **Deployment**: Apply the extracted knowledge to resolve the concerned business issue and take necessary actions.

B. Identification of Dataset Characteristics and Overview

The present report focuses on three distinct datasets that contain numerous records of Emails and SMS messages, varying substantially in nature. The dataset's columns have been categorized into two segments: the text body, representing either the SMS or Email and the target variable or label. The label holds binary values that depict whether the Email/SMS falls under the HAM or SPAM category.

This study presents a comparative analysis of individual datasets within the results.

1) Email Spam Dataset: The Email Spam dataset consists of a collection of email messages that have been labelled as either 1 (indicating Spam) or 0 (indicating ham/non-spam) [8].

- In a classification email spam dataset, the "Body" column is typically the independent variable, which contains the content of the email, and the "Label" column is the dependent variable, which contains the label or classification of whether the email is spam (1) or not spam (0).

```
In [4]: #checking the data head
        df.sample(7)
```

Out[4]:

	Unnamed: 0	Body	Label
2808	3098	Matthias Saou wrote:\n> \n> I'd like to aks th...	0
3730	4020	URL: http://www.newsisfree.com/click/-2,841883...	0
4655	5160	> I'm sure if you call their customer service ...	0
478	478	empty	1
2767	3057	empty	0
167	167	This WEEK: Sydney Bares ALL in the park!\nJoin...	1
2610	2900	Here's a message that works fine for me:X-Mail...	0

```
In [6]: df.shape
Out[6]: (5042, 3)
```

Fig. 1. Raw Email Spam Dataset Before Data Cleaning

- Now Fig:1 shows The raw email spam dataset which contains 5042 rows and 3 columns. The columns are "Unnamed: 0", "Body", and "Label". The first column in the dataset represents a serial number that is intended to be removed through data-cleaning techniques during the analysis process. The remaining two columns, labelled "Body" and "Label", are the focus of the analysis.

- The given dataset contains three columns - "Unnamed: 0", "Body", and "Label". The first column is a serial number that can be ignored for analysis. The second column "Body" contains textual data, while the third column "Label" contains binary data with 0 and 1 representing non-spam (ham) and spam, respectively. The dataset contains textual and binary data types, which can be analyzed using exploratory data analysis (EDA) for textual data and classification algorithms for binary data. This combination of data types makes it appropriate for analyzing using various machine-learning techniques.

click to expand output; double click to hide output

	email	target
0	Save up to 70% on Life Insurance.\nWhy Spend M...	1
1	1) Fight The Risk of Cancer!\nhttp://www.adcli...	1
2	1) Fight The Risk of Cancer!\nhttp://www.adcli...	1
3	##########...	1
4	I thought you might like these:\n1) Slim Down ...	1

Fig. 2. Email Spam Dataset after data cleaning

- The data cleaning steps are performed on a dataset consisting of emails and their corresponding labels. The first step involves dropping the column 'Unnamed: 0' which is not required for analysis. The column names 'Body' and 'Label' are changed to 'email' and 'target' respectively for better readability. The missing values are identified using the isnull() method and one missing value is found in the 'email' column. This row is dropped using the dropna() method. The

duplicated rows are identified using the duplicated() method and 689 duplicated rows are found. These rows are removed using the drop_duplicates() method, and the resulting dataset(Fig:2) has 4352 rows and 2 columns.

- This study involves analyzing a DataFrame of email messages and adding three new columns to provide additional insights. These new columns, character_number, word_number, and sentence_number, are important features for various machine learning tasks, including predictive modelling and correlation analysis. The study highlights the significance of data preprocessing and feature engineering in machine learning applications, revealing previously unknown patterns and relationships. By utilizing multiple features and meaningful feature extraction, the accuracy and performance of machine learning models can be improved, and a deeper understanding of the underlying patterns within the dataset can be gained(Fig 3).

	email	target	character_number	word_number	sentence_number
4038	http://www.pressherald.com/news/state/021201mo...	0	6567	1386	45
2680	I have failed dependencies in RPM database to ...	0	1345	248	11
3349	We've not only reduced the f-p and f-n rates i...	0	2134	511	2
1882	> Apols if this has been posted before:\n> \n>...	0	514	127	3
552	Have you checked your personal credit reports ...	1	2721	527	22
3995	URL: http://boingboing.net/#85541017\nDate: No...	0	531	95	5
2987	\n DanQ> I think it would make more sense t...	0	1658	305	7

Fig. 3. Email Spam Dataset after adding new columns

- Descriptive statistics have been generated for the new columns 'character_number', 'word_number', and 'sentence_number' added to the Email message DataFrame. The count, mean, standard deviation, minimum, maximum, and quartile values have been computed for each column. The mean length of the Email messages in terms of characters is 1751.68, with a standard deviation of 4248.16. The mean number of words and sentences in the messages is 326.32 and 11.73, respectively. These statistics provide a summary of the distribution of these variables and can be used to better understand the dataset(Fig:4).

	character_number	word_number	sentence_number
count	4352.000000	4352.000000	4352.000000
mean	1751.679458	326.323759	11.729320
std	4248.164995	695.867967	25.237384
min	1.000000	0.000000	0.000000
25%	531.000000	97.000000	4.000000
50%	965.000000	180.500000	7.000000
75%	1662.500000	321.000000	11.000000
max	134625.000000	16479.000000	573.000000

Fig. 4. Overall Descriptive Statistics

- The statistics of 'character_number', 'word_number', and 'sentence_number' columns have been generated for the HAM records in the DataFrame, which have a target value of 0. The descriptive statistics include count, mean, standard deviation, minimum, 25th percentile, 50th percentile, 75th percentile, and maximum values for each column. The mean values for character_number, word_number, and sentence_number are 1437.49, 279.95, and 9.57, respectively. The standard deviations are 3187.02, 590.80, and 20.90, respectively. Additionally, the minimum, median, and maximum values for each column are reported(Fig:5).

	character_number	word_number	sentence_number
count	3016.000000	3016.000000	3016.000000
mean	1437.488395	279.946950	9.570955
std	3187.017936	590.801917	20.904718
min	5.000000	1.000000	1.000000
25%	466.750000	87.000000	3.000000
50%	880.500000	172.000000	6.000000
75%	1498.000000	300.000000	10.000000
max	87972.000000	16479.000000	573.000000

Fig. 5. Descriptive Statistics of the HAM records

- Descriptive statistics have been generated for SPAM(Fig:6) records in the 'character_number', 'word_number', and 'sentence_number' columns of the DataFrame. The statistics provide insight into the length and complexity of the messages. The statistics show that the mean character length of SPAM messages is 2461, with a standard deviation of 5929. The mean number of words is 431, with a standard deviation of 880, and the mean number of sentences is 17. The minimum and maximum values of these columns are also shown, along with the 25th, 50th, and 75th percentiles. These statistics can be used to better understand the SPAM messages and to develop more effective strategies for filtering or detecting them.

	character_number	word_number	sentence_number
count	1336.000000	1336.000000	1336.000000
mean	2460.961078	431.018713	16.601796
std	5928.957569	879.814088	32.474919
min	1.000000	0.000000	0.000000
25%	676.000000	120.000000	5.000000
50%	1109.500000	201.000000	9.000000
75%	2213.750000	414.000000	15.000000
max	134625.000000	13290.000000	404.000000

Fig. 6. Descriptive Statistics of the SPAM records

- The data preprocessing step involves the transformation of the raw email data in the DataFrame to a format that can be used for analysis. This is done using a Python function that converts the text to lowercase, tokenizes it

into words, removes non-alphanumeric characters, stopwords and punctuation, applies stemming to reduce the words to their base form, and then joins them back into a single string. The transformed data is stored in a new column named transformed_email in the same DataFrame(Fig:7). This step is important for improving the accuracy of the analysis and reducing the impact of noise in the data.

	email	target	character_number	word_number	sentence_number	transformed_email
1249	HAS YOUR MORTGAGE SEARCH GOT YOU DOWN?Win a $5...	1	523	118	6	ha your mortgag search got you down win a mort...
2923	So do you know my public key? Does the guy wh...	0	1301	283	10	so do you know my public key doe the guy who w...
3483	URL: http://scriptingnews.userland.com/backiss...	0	235	31	2	url http date tue 24 sep 2002 gmtjeremi allair...
2969	Robin Lynn Frank writes:> I may be dense, but...	0	1760	351	11	robin lynn frank write i may be dens but whi w...
3936	URL: http://boingboing.net/#85531557\nDate: No...	0	1032	183	9	url http 85531557 date not supplieddisney them...
4087	\n Paul> I suspect the best answer is to ma...	0	554	86	3	paul i suspect the best answer is to make the ...
427	Question?Do you want a different job?\nDo you ...	1	716	132	6	question do you want a differ job do you want ...

Fig. 7. Final dataset after data pre-processing

- The generate method of the WordCloud object(Fig:8) is then called with the transformed email messages that are classified as spam. Finally, the resulting image is displayed using the imshow method of the pyplot module. This is a useful technique for visualizing the most frequently occurring words in a corpus of text.

Fig. 8. SPAM word cloud

- A visualization technique using the WordCloud object has been employed in the HAM corpus in the same manner(Fig:9). This approach facilitates the identification of the most commonly occurring words within a text corpus by generating a visual representation.

- In this study, the TfidfVectorizer technique is utilized to transform preprocessed text data into a numerical feature matrix that can be used in machine learning algorithms. On the other hand a TfidfVectorizer object is created with a max_features parameter of 3000, limiting the number of words included in the feature matrix to the 3000 most frequently occurring words in the corpus. This allows for more efficient and effective data

Fig. 9. HAM word cloud

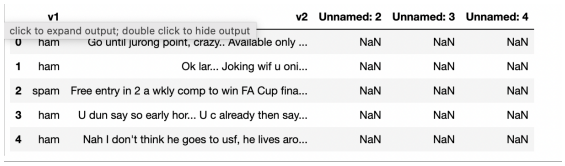
comparison analysis in the result analysis section.

The TfidfVectorizer feature extraction method resulted in a feature matrix, X, of shape (4352, 27168), where 4352 represents the number of samples and 27168 represents the number of features. Another TfidfVectorizer with max feature 3000 produced a NumPy array with a shape of (4352, 3000), containing 4352 samples and 3000 features. These arrays can be utilized to train and assess machine learning models for spam detection. TfidfVectorizer efficiently extracts important features from text data while minimizing the influence of irrelevant common words in distinguishing spam and non-spam messages.

- The dataset is divided into training and testing sets using the train_test_split function, with a test size of 0.3 and a random state of 42 to ensure reproducibility. With TfidfVectorizer, the resulting X_train and y_train variables contain 3046 samples, while the X_test and y_test variables contain 1306 samples, all with 27168 features each. Similarly, using TfidfVectorizer with max feature 3000, the resulting X_train and y_train variables have 3000 features each, and X_test and y_test variables have the same shape.

2) SMS Spam Collection Dataset: The presented dataset in Fig 10 consists of raw SMS spam messages. It includes columns with labels indicating whether the message is ham or spam, as well as three additional columns that seem to contain no useful information. This dataset could be used for training machine learning models to classify incoming SMS messages as either spam or legitimate (ham) [9].

- The SMS Spam Collection dataset(Fig 10.) contains two main variables: the independent variable and the dependent variable. The independent variable is the text messages (v2 column), while the dependent variable is the label associated with each message (v1



	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

Fig. 10. Raw SMS Spam Collection Dataset

column). The label can be either 'ham', indicating a legitimate message, or 'spam', indicating a message that is unwanted or unsolicited. The classification of the message as either 'ham' or 'spam' is the dependent variable, as it is determined by the content of the message itself.

- The data cleaning stage on the dataset consists of SMS messages labeled as spam or ham. The three unnamed columns in the dataset are dropped using the drop method of the pandas library. The columns are then renamed using the rename method(Fig 11). The 'v1' column is renamed to 'detect' and the 'v2' column is renamed to 'sms'. This step helps to make the dataset more interpretable and easy to work with.

	detect	sms
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...

Fig. 11. SMS Spam Collection Dataset after data cleaning

- The 'LabelEncoder' module from scikit-learn is utilized in the study, with an instance called 'encoder' created to encode the 'detect' column using the 'fit_transform' method. The dataset is then checked for missing and duplicate values using the 'isnull' and 'duplicated' methods, respectively. The dataset was found to have no missing values but contained 346 duplicates, which were then removed using the 'drop_duplicates' method with the 'keep' parameter set to 'first'. The dataset was then rechecked for duplicates, and none were found, ensuring data quality.

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

Fig. 12. SMS Spam Collection Dataset after label encoding

- Now extend the DataFrame by three columns(Fig 13): character_number, word_number, and sentence_number. Each column provides a specific information about the corresponding SMS text, character_number indicates the number of characters, word_number indicates the number of words, and sentence_number indicates the number of sentences in each SMS.

	detect	sms	character_number	word_number	sentence_number
2370	1	You are guaranteed the latest Nokia Phone, a 4...	149	30	3
4266	0	U R THE MOST BEAUTIFUL GIRL IVE EVER SEEN. U R...	87	22	2
4374	0	Or just do that 6times	22	5	1
736	0	Thanks for looking out for me. I really apprec...	51	11	2
720	0	Oh is it? Send me the address	29	8	2
614	0	I called and said all to him:)then he have to ...	65	17	1
2824	0	We walked from my moms. Right on stagwood pass...	110	27	3

Fig. 13. SMS Spam Collection Dataset after adding new columns

	character_number	word_number	sentence_numl
count	4636.000000	4636.000000	4636.000(
mean	78.842752	18.374029	1.959(
std	57.605103	13.132151	1.441(
min	2.000000	1.000000	1.000(
25%	36.000000	9.000000	1.000(
50%	60.000000	15.000000	1.000(
75%	117.000000	26.000000	2.000(
max	910.000000	220.000000	38.000(

	character_number	word_number	sentence_numt
count	4057.000000	4057.000000	4057.0000
mean	70.372443	17.054967	1.8136
std	55.511718	13.261543	1.3734
min	2.000000	1.000000	1.0000
25%	33.000000	8.000000	1.0000
50%	52.000000	13.000000	1.0000
75%	91.000000	22.000000	2.0000
max	910.000000	220.000000	38.0000

	character_number	word_number	sentence_number
count	579.000000	579.000000	579.000000
mean	138.193437	27.616580	2.977547
std	30.880065	7.141535	1.494342
min	13.000000	2.000000	1.000000
25%	130.000000	25.000000	2.000000
50%	149.000000	29.000000	3.000000
75%	158.000000	32.000000	4.000000
max	226.000000	46.000000	9.000000

Fig. 14. Descriptive Statistics of SMS Spam Collection Dataset

- The descriptive statistics for the columns 'character_number', 'word_number', and 'sentence_number' in a pandas DataFrame. It first computes the statistics for the entire dataset, including the count, mean, standard deviation, minimum, maximum, and the first, second, and third quartiles for each column. It then generates similar statistics separately for the records labeled as HAM and SPAM in the 'detect' column of the DataFrame(Fig 14). The statistics for HAM records show lower means and standard deviations than those for SPAM records, indicating that HAM messages tend to be shorter than SPAM messages in terms of the number of characters, words, and sentences.

- In the data preprocessing methodology the Porter Stemming algorithm is applied to convert words to their base form. The function 'transform_sms' takes an input SMS message and converts it into lowercase, tokenizes it into words, and removes non-alphanumeric characters, stopwords, and punctuation. The transformed SMS is then

joined back into a single string. The resulting transformed SMS is stored in the 'transformed_sms' column of the data frame (Fig 15).

	detect	sms	character_number	word_number	sentence_number	transformed_sms
36	0	Oops, I'll let you know when my roommate's done	47	12	1	oop let know roommat done
2875	0	Oh thats late! Well have a good night and i wi...	113	27	3	oh that late well good night give u call tomor...
1485	0	Single line with a big meaning:::: \Miss anyt...	73	21	1	singl line big mean anyth 4 ur best
3825	0	It certainly puts things into perspective when...	74	11	1	certainli put thing perspect someth like happen
1845	0	It's that time of the week again, ryan	38	10	1	time week ryan
4915	0	Oh yeah! And my diet just flew out the window	45	11	2	oh yeah diet flew window
332	1	Call Germany for only 1 pence per minute! Call...	128	28	4	call germani 1 penc per minut call fix line vi...

Fig. 15. SMS Spam Collection Dataset after data pre-processing

- A WordCloud object with specified parameters generates a word cloud of transformed SMS messages classified as spam(Fig 16). The resulting image is displayed using the Pyplot module.

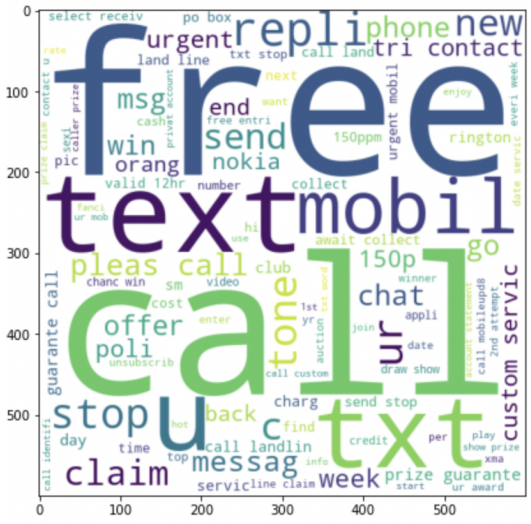


Fig. 16. Word Cloud for the SPAM SMS

Fig. 17. Word Cloud for the HAM SMS

- A word cloud visualization for the transformed SMS messages has been generated(Fig 17) that are classified as 'ham' (non-spam). The SMS messages are concatenated into a single string using the 'str.cat' method and passed as an argument to the 'generate' function of the WordCloud library.

- A TfidfVectorizer technique is employed to convert preprocessed text data into a numerical feature matrix suitable for machine learning algorithms. Another resulting object includes the 3000 most frequently occurring words in the corpus for more efficient analysis.

- Two different TfidfVectorizer feature extraction methods were used in this study, resulting in feature matrices of shapes (4636, 6294) and (4636, 3000), respectively. These matrices can be used for training and evaluating machine learning models for spam detection. TfidfVectorizer proved to be effective in extracting relevant features from text data and reducing the impact of common words that do not help distinguish between spam and non-spam messages.

- The dataset is split into training and testing sets using the train_test_split function, with a test size of 0.3 and a random state of 42. TfidfVectorizer is used to extract features from the data, resulting in X_train and y_train variables containing 3245 samples and 6294 features each, and X_test and y_test variables containing 1391 samples and 6294 features each. Alternatively, by using TfidfVectorizer with max feature 3000, the resulting X_train and y_train variables have 3000 features each, and X_test and y_test variables have the same shape.

3) Spam or Not Spam Dataset: The Spam or Not Dataset(Fig 18) consists of 2608 emails labeled as spam or not spam (ham). Each email is represented by its text content and corresponding label (0 for ham and 1 for spam). This dataset can be used for training and testing machine learning models for spam classification [10].

	email	label
1677	url URL date NUMBER NUMBER NUMBERtNUMBER NUMBE...	0
1587	before we get too far down this road what do p...	0
1298	tim i gave it all the thought it deserved win...	0
1670	url URL date NUMBER NUMBER NUMBERtNUMBER NUMBE...	0
1052	dag but when procmail runs it it doesn t pres...	0
667	joseph s barrera iii joe barrera org writes l...	0
1860	url URL date NUMBER NUMBER NUMBERtNUMBER NUMBE...	0

Fig. 18. Raw Spam or Not Spam Dataset

- The renaming of the column name of the "label" column to "target" in the spam or not dataset(Fig 19). The dataset contains email messages with a binary target variable indicating whether the email is spam or not.

	email	target
0	date wed NUMBER aug NUMBER NUMBER NUMBER NUMB...	0
1	martin a posted tassos papadopoulos the greek ...	0
2	man threatens explosion in moscow thursday aug...	0
3	klez the virus that won t die already the most...	0
4	in adding cream to spaghetti carbonara which ...	0

Fig. 19. Spam or Not Spam Dataset after renaming

The head of the dataset is then checked to ensure that the column name has been successfully changed.

- Missing values in the 'email' column are identified and subsequently dropped from the DataFrame. Duplicate values are also identified and removed to ensure data integrity. The final DataFrame consists of non-duplicate email entries with their respective spam or not spam.

- In this section of code, three new columns are added to the existing spam or not DataFrame(Fig 20). The first new column 'character_number' contains the number of characters in each email. The second new column 'word_number' contains the number of words in each email, which is calculated using the nltk library's word_tokenize method. Finally, the third new column 'sentence_number' contains the number of sentences in each email, which is calculated using the nltk library's sent_tokenize method.
  This step is important as it provides additional features for analyzing the text data and helps in better understanding and predicting whether an email is spam or not.

	email	target	character_number	word_number	sentence_number
444	someone has taken it mostly newtel users i thi...	0	129	25	1
1218	on tuesday NUMBER august NUMBER NUMBER NUMBER ...	0	312	47	1
95	begin pgp signed message hash shaNUMBER your ...	0	1544	253	1
17	in a nutshell solaris is suns own flavour of u...	0	734	131	1
1840	url URL date NUMBER NUMBER NUMBERtNUMBER NUMBE...	0	277	49	1
687	michael wrote URL i thought this nekkid url wa...	0	1735	299	1
1696	url URL date not supplied a new analysis of sa...	0	154	25	1

Fig. 20. Spam or Not Spam Dataset after adding new columns

- The descriptive statistics for the 'character_number', 'word_number', and 'sentence_number' features of the spam or not dataset(Fig 21). Firstly it generates descriptive statistics for the entire dataset, while the second and third sets generate separate descriptive statistics for the HAM (not spam) and SPAM records. The 'describe()' method provides statistical information such as mean, standard deviation, minimum, maximum, and quartiles for each feature. These statistics can help to identify patterns and differences between the spam and not spam emails in terms of their length and complexity.
- The transform_email function is defined to convert the text to lowercase, tokenize the string into words, remove non-alphanumeric characters, remove stopwords and punctuation, and apply stemming. The function is

	character_number	word_number	sentence_number
count	2497.000000	2497.000000	2497.000000
mean	1400.213857	231.853825	0.999600
std	4769.386079	458.082097	0.020012
min	1.000000	0.000000	0.000000
25%	392.000000	68.000000	1.000000
50%	771.000000	137.000000	1.000000
75%	1364.000000	237.000000	1.000000
max	200408.000000	11602.000000	1.000000

	character_number	word_number	sentence_numbe
count	2110.000000	2110.000000	2110.(
mean	1181.910900	207.072986	1.(
std	2035.896045	349.471708	0.(
min	5.000000	1.000000	1.(
25%	361.250000	63.000000	1.(
50%	734.500000	129.000000	1.(
75%	1292.750000	230.000000	1.(
max	35671.000000	6338.000000	1.(

	character_number	word_number	sentence_number
count	387.000000	387.000000	387.000000
mean	2590.444444	366.963824	0.997416
std	11079.819512	817.276514	0.050833
min	1.000000	0.000000	0.000000
25%	570.500000	95.500000	1.000000
50%	995.000000	166.000000	1.000000
75%	2036.500000	343.000000	1.000000
max	200408.000000	11602.000000	1.000000

Fig. 21. Descriptive Statistics for Spam or Not Spam Dataset

applied to the 'email' column of the dataframe, and the resulting transformed text is added as a new column 'transformed_email'(Fig 22).

	email	target	character_number	word_number	sentence_number	transformed_email
2488	this is definitely the answer we all have bee...	1	3174	601	1	thi is definit the answer we all have been wai...
1164	once upon a time joshua wrote just a thought w...	0	2289	410	1	onc upon a time joshua wrote just a thought wo...
1936	url URL date NUMBER NUMBER NUMBERtNUMBER NUMBE...	0	269	52	1	url url date number number numbertnumb number ...
1015	i m a very happy user of exmh but i m paranoid...	0	883	169	1	i m a veri happi user of exmh but i m paranoid...
881	on wed NUMBER sep NUMBER NUMBER NUMBER NUMBER ...	0	738	144	1	on wed number sep number number number number ...
1288	begin pgp signed message hash shaNUMBER what ...	0	1102	174	1	begin pgp sign messag hash shanumb what do you...
904	i recently transfered my exmh setup to a new s...	0	329	51	1	i recent transfer my exmh setup to a new syste...

Fig. 22. Spam or Not Spam Dataset after data pre-processing

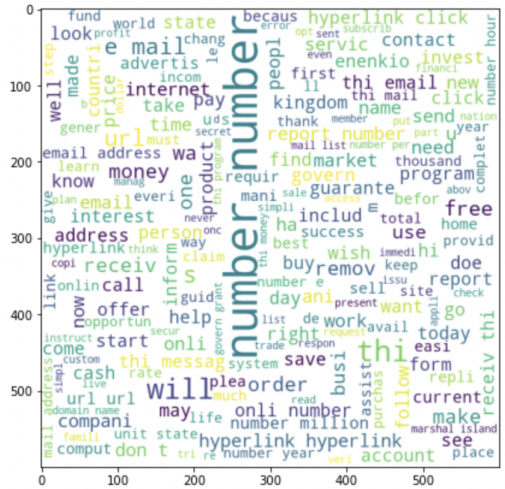


Fig. 23. Spam Word Cloud

- A WordCloud of the transformed email messages that are classified as spam using the generate method of the WordCloud object. The resulting WordCloud image(Fig 23) is then displayed using the imshow method of the

pyplot module within a figure object.

- Word cloud visualization for the transformed emails in the ham (not spam) messages of the dataset. The wc.generate() method takes in a string of text and creates a word cloud image with the frequency of each word being represented by its size. This visualization can help to identify common words or themes in the ham messages(Fig 24).

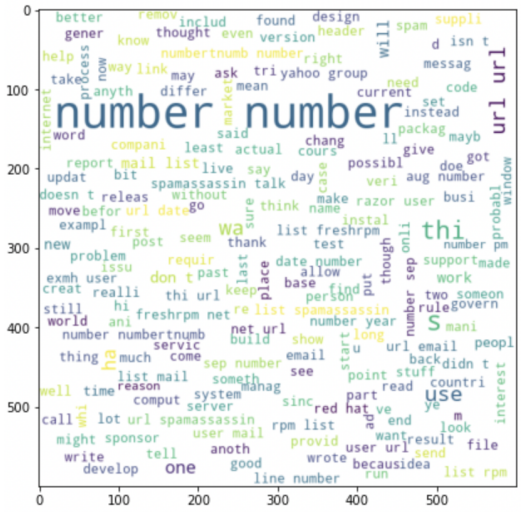


Fig. 24. Ham Word Cloud

- The TfidfVectorizer feature extraction method generates a feature matrix X with dimensions (2497, 22495), with 2497 samples and 22495 features. The use of TfidfVectorizer with max feature 3000 resultes in a NumPy array with dimensions (2497, 3000), containing 2497 samples and 3000 features. These matrices can be used to train and evaluate machine learning models for spam detection, effectively identifying significant features in text data while reducing the impact of common words.

- The dataset splits into training and testing sets using train_test_split with a test size of 0.3 and a random state of 42 for reproducibility. The resulting X_train and y_train variables for TfidfVectorizer have 1747 samples, while X_test and y_test variables have 750 samples, all with 22495 features each. When using TfidfVectorizer with max feature 3000, X_train and y_train variables have 3000 features each, and X_test and y_test variables have the same shape.

C. Visualization

Machine learning is a branch of AI focused on creating algorithms capable of learning from data [11]. Visualizations of data are crucial in this field as they aid in comprehending intricate datasets and identifying patterns, which can subsequently be used to design algorithms that can automatically detect those patterns(Fig 25).

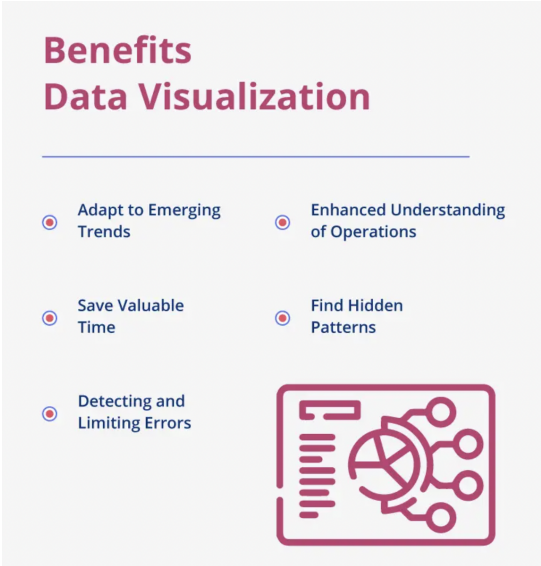


Fig. 25. Benefits of Visualisation

1) Email Spam Dataset Visualization:

- A pie chart(Fig 26) to visualize the distribution of values in the 'target' column of the email spam dataset. The chart shows the percentage of emails that are categorized as 'not spam' and 'spam'.

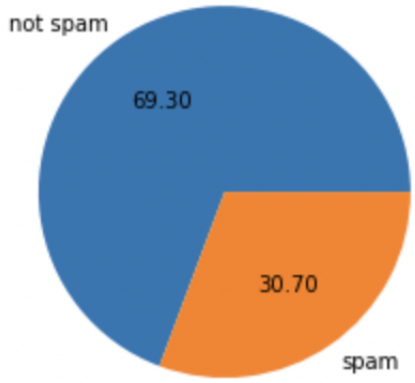


Fig. 26. Pie Chart shows percentage emails(spam or not spam)

- The Seaborn's pairplot() function is utilized to create a pair plot that depicts the correlation between various feature pairs in the email spam dataset. The 'target' feature is employed to distinguish between spam and non-spam messages. The majority of the plots exhibit a linear relationship, which implies that there is a positive correlation between the features(Fig 27).

- Two lists of spam and ham words respectively by iterating through the transformed email messages in the dataset in Fig 28 and Fig 29 respectively. The most common

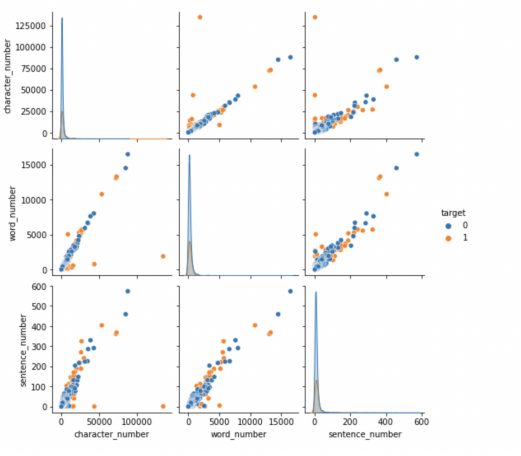


Fig. 27. Scatter plot of Email Spam

25 words in each list are then plotted using Seaborn's barplot() function. The x-axis displays the word and the y-axis displays the frequency of the word in the respective list. These plots help to identify the most common words used in spam and non-spam messages in the email spam dataset.

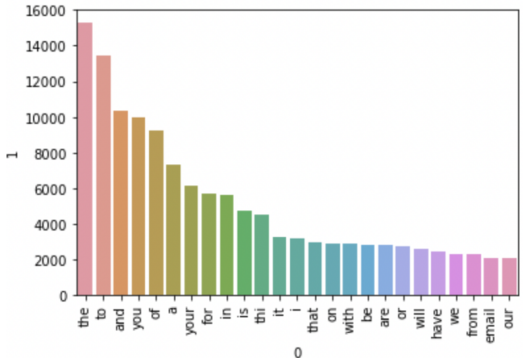


Fig. 28. Most common Spam words

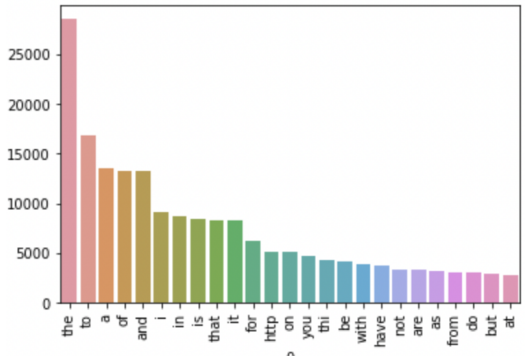


Fig. 29. Most common Ham words

2) SMS Spam Collection Dataset Visualization:

- A bar chart is created to represent the distribution of ham and spam messages in the SMS spam or not spam dataset(Fig 30). The 'detect' column is used to differentiate between ham and spam messages, and the bar chart shows the count of each category.

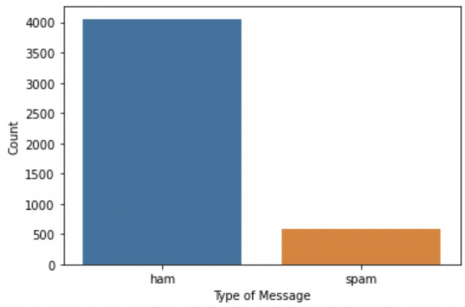


Fig. 30. Bar Chart for Sms Spam Dataset

- A pair plot using Seaborn's pairplot() function to analyze the correlation between different pairs of features in the SMS spam or not spam dataset(Fig 31). The 'detect' column is used to differentiate between spam and non-spam messages. The pair plot results indicate that only two pairs of features exhibit a positive correlation, namely character number and word number. However, the rest of the pairs show a negative correlation between themselves.

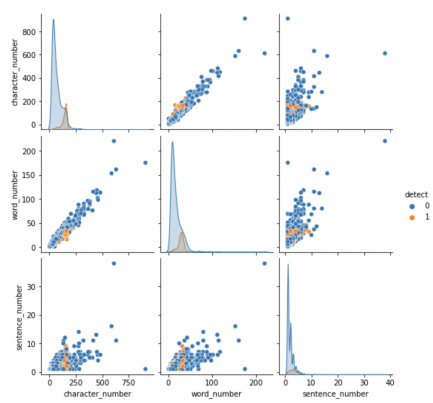


Fig. 31. Scatter plot of SMS Spam

Fig. 32. Heatmap for SMS Spam Dataset

- The Seaborn's heatmap() function is used to visualise the correlation matrix between various features of the SMS spam or not spam dataset(Fig 32). The heatmap clearly shows that there is a high correlation between the character number and the word number variables, which is consistent with the conclusion drawn from the scatter plot.

3) Spam or Not Spam Dataset:

- The pie chart(Fig 33) visualizes the distribution of values in the 'detect' column for the spam or not spam dataset. The pie chart has two sections, labelled as 'not spam' and 'spam', representing the number of non-spam and spam messages respectively. The percentage of non-spam messages is 84.50%, while the percentage of spam messages is 15.50%. This indicates that the dataset contains a relatively low percentage of spam messages compared to non-spam messages.

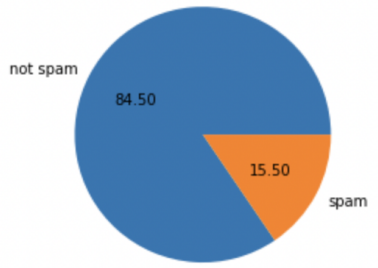


Fig. 33. Pie chart for spam or not spam dataset

- The plot displays two histograms presenting(Fig 34) the character count distribution for 'ham' and 'spam' messages in the SMS spam or not spam dataset. The blue histogram corresponds to 'ham' messages while the red histogram corresponds to 'spam' messages. It indicates that the dataset has a higher number of characters in 'ham' messages than in 'spam' messages. This finding aligns with the previously shown pie chart.

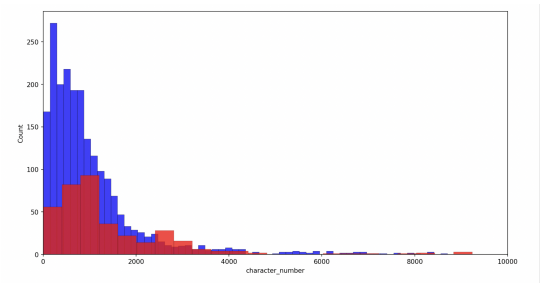


Fig. 34. Histogram shows the character count

D. Machine Learning Library

1) Sklearn Python Library: Sklearn is a highly useful and reliable machine-learning library for Python that offers a range of effective tools for statistical modelling and machine learning. These include regression, classification, clustering, and dimensionality reduction. Sklearn offers a consistent interface and is built on NumPy, SciPy, and Matplotlib [12].

The Scikit-learn (Sklearn) library is very useful for spam detection machine learning because it provides a wide range of tools and algorithms for classification tasks, which are commonly used in spam detection. Sklearn allows for easy and efficient implementation of machine learning models, such as decision trees, logistic regression, and support vector machines, which can be used to detect spam messages.

2) XGboost Python Library: XGBoost (Extreme Gradient Boosting) is a popular open-source library for gradient-boosted trees. It is written in C++ and has APIs for Python, R, Java, and several other programming languages. XGBoost is designed to be highly efficient, flexible, and scalable, and is widely used in machine learning competitions and real-world applications for classification and regression tasks. Its popularity is due to its high performance, ability to handle large datasets, and compatibility with other popular machine learning libraries such as scikit-learn [13].

XGBoost is a powerful machine-learning library that can be used for spam detection. It offers a highly optimized implementation of gradient boosting, a popular machine-learning technique for classification and regression problems. In spam detection, XGBoost can be used to build a highly accurate model that can distinguish between spam and legitimate messages based on a variety of features extracted from the text data.

E. Machine Learning Algorithms

1) Naive Bayes classifier: Naive Bayes classifiers are commonly utilized in machine learning for classification purposes. This is because, in situations where the assumption of independence is applicable, they are comparatively simple to implement and produce more favourable outcomes than other advanced predictors [14].

There are three main types of Naive Bayes classifiers:

1) Gaussian Naive Bayes: Assumes that the features follow a normal distribution.
2) Multinomial Naive Bayes: Used for discrete counts, such as word counts in text classification.
3) Bernoulli Naive Bayes: Assumes binary features, where each feature can either be present or not present.

Multinomial Naive Bayes is a commonly used algorithm for spam detection in machine learning. It is particularly effective in handling text data, which is often used in spam detection. This algorithm is used in this study because it is designed to handle count-based data such as the frequency of specific words or phrases in a text document. In spam detection, the frequency of certain words or phrases is a crucial factor in determining whether an email or an SMS is spam or not.

2) DecisionTreeClassifier: A decision tree is a white box type of machine learning algorithm, represented as a flowchart-like tree structure, where each internal node represents a feature or attribute, and each leaf node represents an outcome. The topmost node is the root node, and the tree is partitioned recursively based on the attribute value. Decision trees are easy to understand and interpret, making them a popular choice for decision-making. [15].

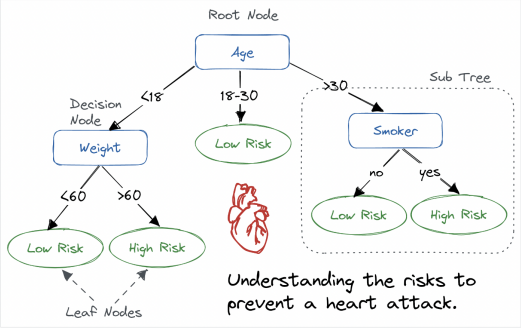


Fig. 35. Example of a DecisionTree

The DecisionTreeClassifier algorithm is used in this study for spam detection in machine learning due to its compatibility with text data. It can partition data based on text features and attributes, allowing for easy identification of spam emails. Additionally, the algorithm's training time is faster than other black box algorithms like neural networks, making it a popular choice in spam detection.

3) RandomForestClassifier: The Random Forest Algorithm is a popular and frequently used Supervised Machine Learning Algorithm by Data Scientists. It is useful for both Classification and Regression problems, building decision trees on various samples and taking their majority vote for classification and average for regression. The algorithm can handle data sets with continuous and categorical variables, performing well in these tasks. [16].

Random Forest is an extension of the Decision Tree algorithm. While Decision Tree builds a single tree on the training data, Random Forest builds multiple trees on randomly selected subsets of the training data. Each tree in the Random Forest algorithm is trained on a different set of data, and its outputs are combined to make a final prediction. This approach helps to reduce overfitting and increase the accuracy of the model [17].

To put it simply, the Random Forest Algorithm merges the outcomes of several Decision Trees that are created randomly to produce the end result(Fig 36).

This study utilizes the RandomForestClassifier algorithm for spam detection as it is a widely used and effective method. It builds multiple decision trees on random subsets of training data and combines them through majority voting to reduce overfitting and improve model performance. RandomForestClassifier can handle continuous and categorical variables, making it suitable for analyzing text data commonly used in spam detection.

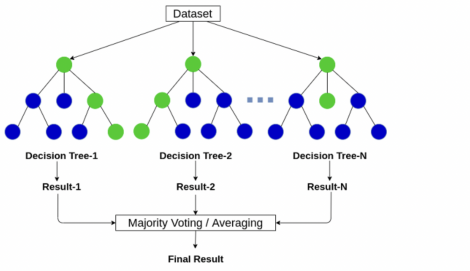


Fig. 36. Example of a RandomForest

IV. Result and Analysis

The study employs various machine learning models and assesses their efficacy using metrics such as Accuracy Score, Precision Score, F1 Score, and Recall. The models are trained using TfidfVectorizer and TfidfVectorizer with a maximum feature of 3000, and their respective datasets are compared. The model that achieves the highest scores is considered the most suitable for the task.

The models in this study are trained using various methods and libraries as listed below.

- Multinomial Naive Bayes classifier from the sklearn
- DecisionTreeClassifier from sklearn
- RandomForestClassifier from sklearn
- DecisionTreeClassifier from xgboost
- RandomForestClassifier from xgboost

A. Email Spam Dataset

1) Model Evaluation based on TfidfVectorizer using sklearn: The RandomForestClassifier algorithm achieves the highest accuracy score of 0.970138 and the highest Precision score of 0.997389, indicating that it has the highest number of true positives compared to false positives. The algorithm also has a high F1 score of 0.951432, suggesting that it has a balance between Precision and Recall. Finally, the Recall score of 0.909524 indicates that the algorithm correctly identified a high percentage of actual spam emails.

The Naive Bayes algorithm achieved a high accuracy score of 0.905819, indicating that it classified most of the emails correctly. However, it had the lowest Recall score of 0.707143, indicating that it failed to identify a significant number of actual spam emails.

The DecisionTreeClassifier algorithm achieved an accuracy score of 0.919602 and a Precision score of 0.931507. However, its F1 score of 0.866242 is lower compared to the other algorithms, indicating that it has a lower balance between Precision and Recall. The Recall score of 0.809524

Algorithm	Accuracy	Precision	F1	Recall
Naive_Bayes	0.905819	1.000000	0.828452	0.707143
RandomForestClassifier	0.970138	0.997389	0.951432	0.909524
DecisionTreeClassifier	0.919602	0.931507	0.866242	0.809524

Fig. 37. Result analysis using TfidfVectorizer

suggests that the algorithm correctly identified a moderate percentage of actual spam emails.

Based on the evaluation metrics(Fig 37) and Bar chart(Fig 38), we conclude that the RandomForestClassifier algorithm is the best fit for spam detection in this study. It achieved the highest accuracy, Precision, and F1 scores while maintaining a high Recall score.

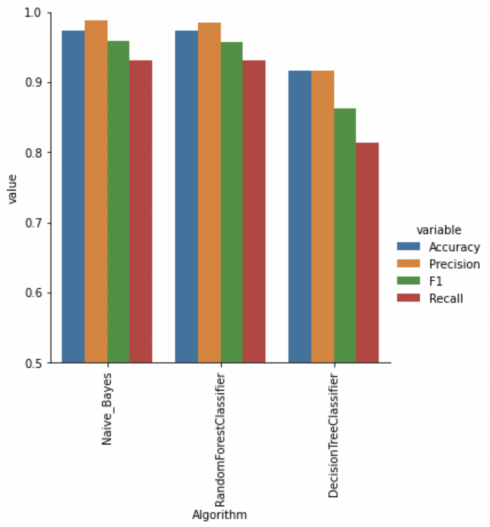


Fig. 38. Bar Chart TfidfVectorizer

2) Model Evaluation based on TfidfVectorizer with max_features=3000 using sklearn: The results show that the Naive Bayes classifier (Multinomial) and RandomForestClassifier perform similarly with high accuracy scores of 0.973966 and 0.973201, respectively. They also have high precision and recall scores, with Naive Bayes having the highest precision score of 0.987374 and RandomForestClassifier having the highest recall score of 0.930952.

On the other hand, the DecisionTreeClassifier has a lower accuracy score of 0.916539, with the lowest precision, F1, and recall scores among the three models.

Based on the results(Fig 39), it can be concluded that both Naive Bayes classifier (Multinomial) and RandomForestClassifier are the best-fit models for the given problem, with Naive Bayes having slightly better precision and RandomForestClassifier having a slightly better recall.

Algorithm	Accuracy_max_ft_3000	Precision_max_ft_3000	F1_max_ft_3000	Recall_max_ft_3000
Naive_Bayes	0.973966	0.987374	0.958333	0.930952
RandomForestClassifier	0.973201	0.984887	0.957160	0.930952
DecisionTreeClassifier	0.916539	0.916890	0.862547	0.814286

Fig. 39. Result analysis using TfidfVectorizer max_feature 3000

3) Compare Analysis and Bestfit Model using sklearn: This study compares the performance of three machine learning models - Naive Bayes classifier (Multinomial), Random Forest Classifier, and Decision Tree Classifier - using two different methods for feature extraction: TfidfVectorizer and TfidfVectorizer with a maximum feature of 3000. Results showes that the Random Forest Classifier achieved the highest scores for Accuracy, Precision, F1, and Recall in both feature extraction methods, making it the best-fit model for the problem.

Algorithm	Accuracy	Precision	F1	Recall	Accuracy_max_ft_3000	Precision_max_ft_3000	F1_max_ft_3000	Recall_max_ft_3000
Naive_Bayes	0.905819	1.000000	0.828452	0.707143	0.973966	0.987374	0.958333	0.930952
RandomForestClassifier	0.970138	0.997389	0.951432	0.909524	0.973201	0.984887	0.957160	0.930952
DecisionTreeClassifier	0.919602	0.931507	0.866242	0.809524	0.916539	0.916890	0.862547	0.814286

Fig. 40. Final Compare analysis of Email Spam Dataset using sklearn

In summary(Fig 40), the study concludes that the Random Forest Classifier outperform the other models in both feature extraction methods, indicating that it is the most effective algorithm for email spam detection. The findings suggest that using TfidfVectorizer with a maximum feature of 3000 can improve the performance of the models, leading to better spam detection accuracy.

4) Model Evaluation based on TfidfVectorizer using xgboost: The XGBoost Decision Tree Classifier achieves an Accuracy score of 0.980858, Precision score of 0.982885, F1 score of 0.969843, and Recall score of 0.957143. On the other hand, the XGBoost Random Forest Classifier achieves an Accuracy score of 0.928790, Precision score of 0.955432, F1 score of 0.880616, and Recall score of 0.816667.

Algorithm	Accuracy	Precision	F1	Recall
xgboost_DecisionTreeClassifier	0.980858	0.982885	0.969843	0.957143
xgboost_RandomForestClassifier	0.928790	0.955432	0.880616	0.816667

Fig. 41. Result Analysis based on TfidfVectorizer using xgboost

Based on the results(Fig 41) and Bar chart(Fig 42), it can be concluded that the XGBoost Decision Tree Classifier is the best-fit model for this problem, as it achieves higher scores for all evaluation metrics compared to the XGBoost Random Forest Classifier.

5) Model Evaluation based on TfidfVectorizer with max_features=3000 using xgboost: The

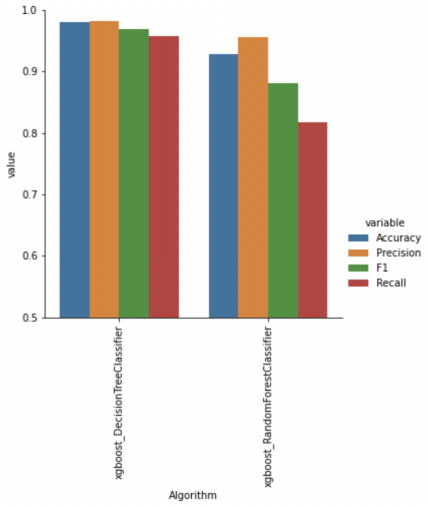


Fig. 42. Bar Chart based on TfidfVectorizer using xgboost

xgboost_DecisionTreeClassifier achieves an Accuracy Score of 0.979326, Precision Score of 0.990025, F1 Score of 0.967113, and Recall of 0.945238 using TfidfVectorizer with a maximum feature of 3000. On the other hand, the xgboost_RandomForestClassifier achieves an Accuracy Score of 0.926493, Precision Score of 0.955056, F1 Score of 0.876289, and Recall of 0.809524 using the same feature extraction method.

	Algorithm	Accuracy_max_ft_3000	Precision_max_ft_3000	F1_max_ft_3000	Recall_max_ft_3000
0	xgboost_DecisionTreeClassifier	0.979326	0.990025	0.967113	0.945238
1	xgboost_RandomForestClassifier	0.926493	0.955056	0.876289	0.809524

Fig. 43. Result Analysis based on TfidfVectorizer with max_features=3000 using xgboost

Based on the results(Fig 43), it can be concluded that the xgboost_DecisionTreeClassifier is the best-fit model for this problem, as it achieves higher scores for Accuracy, Precision, F1, and Recall compared to the xgboost_RandomForestClassifier.

6) Compare Analysis and Best fit Model using xgboost: This study compares the performance of two machine learning models, namely XGBoost Decision Tree Classifier and XGBoost Random Forest Classifier, using two different methods for feature extraction, namely TfidfVectorizer and TfidfVectorizer with a maximum feature of 3000. The models are evaluated based on their Accuracy Score, Precision Score, F1 Score, and Recall for the email spam dataset.

	Algorithm	Accuracy	Precision	F1	Recall	Accuracy_max_ft_3000	Precision_max_ft_3000	F1_max_ft_3000	Recall_max_ft_3000
0	xgboost_DecisionTreeClassifier	0.980858	0.982885	0.969843	0.957143	0.979326	0.990025	0.967113	0.945238
1	xgboost_RandomForestClassifier	0.928790	0.955432	0.880616	0.816667	0.926493	0.955056	0.876289	0.809524

Fig. 44. Final Compare analysis of Email Spam Dataset using xgboost

Based on the results(Fig 44), it can be concluded that the XGBoost Decision Tree Classifier is the best-fit model for this problem, as it achieved the highest scores for Accuracy, Precision, F1, and Recall in both feature extraction methods.

B. SMS Spam Collection Dataset

1) Compare Analysis and Best fit Model using sklearn: This study compares the performance of three machine learning models, the Naive Bayes classifier, Random Forest Classifier, and Decision Tree Classifier, using two different methods for feature extraction, TfidfVectorizer and TfidfVectorizer with a maximum feature of 3000. The models were evaluated based on their Accuracy Score, Precision Score, F1 Score, and Recall.

The Naive Bayes classifier achieves an Accuracy Score of 0.943925, Precision Score of 1.000000, F1 Score of 0.738255, and Recall of 0.585106 using the TfidfVectorizer method, while it achieves an Accuracy Score of 0.969087, Precision Score of 0.993197, F1 Score of 0.871642, and Recall of 0.776596 using TfidfVectorizer with a maximum feature of 3000.

The Random Forest Classifier achieves an Accuracy Score of 0.966211, Precision Score of 0.993007, F1 Score of 0.858006, and Recall of 0.755319 using the TfidfVectorizer method, while it achieves an Accuracy Score of 0.973400, Precision Score of 0.993464, F1 Score of 0.891496, and Recall of 0.808511 using TfidfVectorizer with a maximum feature of 3000.

The Decision Tree Classifier achieves an Accuracy Score of 0.936017, Precision Score of 0.902439, F1 Score of 0.713826, and Recall of 0.590426 using the TfidfVectorizer method, while it achieves an Accuracy Score of 0.932423, Precision Score of 0.885246, F1 Score of 0.696774, and Recall of 0.574468 using TfidfVectorizer with a maximum feature of 3000.

	Algorithm	Accuracy	Precision	F1	Recall	Accuracy_max_ft_3000	Precision_max_ft_3000	F1_max_ft_3000	Recall_max_ft_3000
0	Naive_Bayes	0.943925	1.000000	0.738255	0.585106	0.969087	0.993197	0.871642	0.776596
1	RandomForestClassifier	0.966211	0.993007	0.858006	0.755319	0.973400	0.993464	0.891496	0.808511
2	DecisionTreeClassifier	0.936017	0.902439	0.713826	0.590426	0.932423	0.885246	0.696774	0.574468

Fig. 45. Final Compare analysis of SMS Spam Dataset using sklearn

In light of the results(Fig 45), it can be inferred that the most suitable model for the SMS spam dataset is the Random Forest Classifier in conjunction with the TfidfVectorizer max feature of 3000. This model displays the highest scores for Accuracy, Precision, F1, and Recall in both feature extraction methods.

2) Compare Analysis and Best fit Model using xgboost: The evaluation of xgboost Decision Tree Classifier and xgboost Random Forest Classifier using TfidfVectorizer and TfidfVectorizer max feature 3000 for SMS spam classification is carried out. The results show that the xgboost Decision

Tree Classifier has better performance than the xgboost Random Forest Classifier for both feature extraction methods.

Using TfidfVectorizer, the Decision Tree Classifier achieves an accuracy score of 0.973400, precision score of 0.974843, F1 score of 0.893372, and recall score of 0.824468. While with TfidfVectorizer max feature 3000, it achieves an accuracy score of 0.973400, precision score of 0.968944, F1 score of 0.893983, and recall score of 0.829787.

On the other hand, using TfidfVectorizer, the Random Forest Classifier achieved an accuracy score of 0.943206, precision score of 0.916031, F1 score of 0.752351, and recall score of 0.638298. While with TfidfVectorizer max feature 3000, it achieved an accuracy score of 0.938893, precision score of 0.899225, F1 score of 0.731861, and recall score of 0.617021.

	Algorithm	Accuracy	Precision	F1	Recall	Accuracy_max_ft_3000	Precision_max_ft_3000	F1_max_ft_3000	Recall_max_ft_3000
0	xgboost_DecisionTreeClassifier	0.973400	0.974843	0.893372	0.824468	0.973400	0.968944	0.893983	0.829787
1	xgboost_RandomForestClassifier	0.943206	0.916031	0.752351	0.638298	0.938893	0.899225	0.731861	0.617021

Fig. 46. Final Compare analysis of SMS Spam Dataset using xgboost

Hence, it is concluded(Fig 46) based on the results and the Bar Chart(Fig 47) that the xgboost Decision Tree Classifier using TfidfVectorizer feature extraction method is the best-fit model for SMS spam classification, as it achieved better scores for accuracy, precision, F1, and recall than the xgboost Random Forest Classifier using both feature extraction methods.

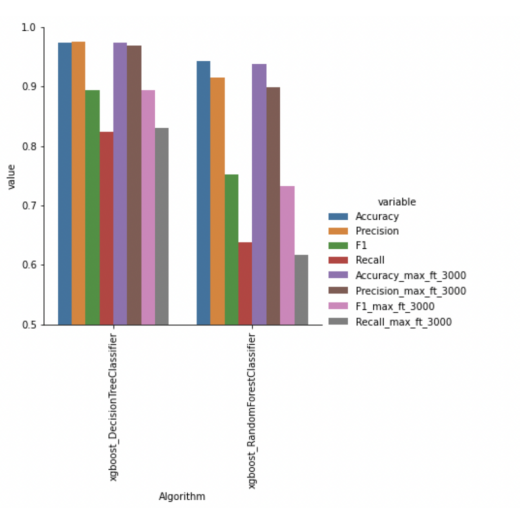


Fig. 47. Compare Analysis using Bar Chart of SMS Spam Dataset using xgboost

C. Spam or Not Spam Dataset

1) Compare Analysis and Best fit Model using sklearn: The Naive Bayes algorithm achieves an accuracy score of 0.869333, precision score of 1.000000, F1 score of 0.222222, and recall score of 0.125000 using TfidfVectorizer. For TfidfVectorizer max feature 3000, it achieves an accuracy score of 0.965333, precision score of 1.000000, F1 score of 0.868687, and recall score of 0.767857.

Upon analysis of the above scores, it is observed that the Naive Bayes algorithm using TfidfVectorizer has a low F1 score of 0.222222 and a low recall score of 0.125000. These scores indicate that the algorithm is not performing well in correctly identifying positive instances (spam messages) in the dataset.

One possible cause of this low performance could be the presence of a class imbalance in the dataset, where there are significantly more instances of negative instances (non-spam messages) than positive instances. This can lead to the algorithm being biased towards predicting negative instances, resulting in low scores for the positive class.

The Random Forest Classifier achieves an accuracy score of 0.974667, precision score of 0.989474, F1 score of 0.908213, and recall score of 0.839286 using TfidfVectorizer. For TfidfVectorizer max feature 3000, it achieves an accuracy score of 0.978667, precision score of 0.989796, F1 score of 0.923810, and recall score of 0.866071.

The Decision Tree Classifier achieves an accuracy score of 0.950667, precision score of 0.871287, F1 score of 0.826291, and recall score of 0.785714 using TfidfVectorizer. For TfidfVectorizer max feature 3000, it achieves an accuracy score of 0.949333, precision score of 0.862745, F1 score of 0.822430, and recall score of 0.785714.

	Algorithm	Accuracy	Precision	F1	Recall	Accuracy_max_ft_3000	Precision_max_ft_3000	F1_max_ft_3000	Recall_max_ft_3000
0	Naive_Bayes	0.869333	1.000000	0.222222	0.125000	0.965333	1.000000	0.868687	0.767857
1	RandomForestClassifier	0.974667	0.989474	0.908213	0.839286	0.978667	0.989796	0.923810	0.866071
2	DecisionTreeClassifier	0.950667	0.871287	0.826291	0.785714	0.949333	0.862745	0.822430	0.785714

Fig. 48. Final Compare analysis of Spam or Not Spam Dataset using sklearn

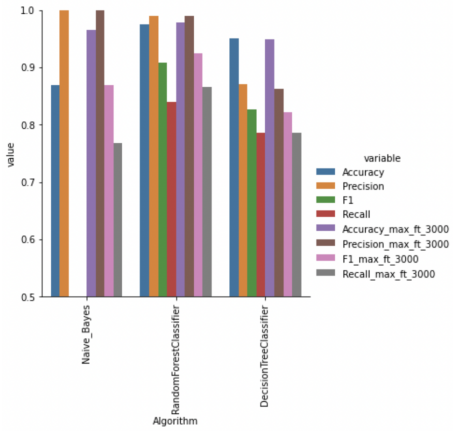


Fig. 49. Compare Analysis using Bar Chart of Spam or Not Spam Dataset using sklearn

Based on the results(Fig 48) and the Bar chart shown (Fig 49), we can conclude that the Random Forest Classifier with TfidfVectorizer max feature 3000 is the best-fit model for this problem, as it achieved the highest scores for accuracy, precision, F1, and recall in both feature extraction methods.

2) Compare Analysis and Best fit Model using xgboost: The xgboost Decision Tree Classifier using TfidfVectorizer obtains an accuracy score of 0.981333, precision score of 0.945455, F1 score of 0.936937, and recall score of 0.928571. Meanwhile, using TfidfVectorizer max feature 3000, it obtains an accuracy score of 0.980000, precision score of 0.944954, F1 score of 0.932127, and recall score of 0.919643.

On the other hand, the xgboost Random Forest Classifier using TfidfVectorizer obtains an accuracy of 0.960000, precision of 0.910000, F1 score of 0.858491, and recall score of 0.812500. When using TfidfVectorizer with a maximum feature count of 3000, the classifier achieves an accuracy score of 0.958667, precision score of 0.909091, F1 score of 0.853081, and recall score of 0.803571.

	Algorithm	Accuracy	Precision	F1	Recall	Accuracy_max_ft_3000	Precision_max_ft_3000	F1_max_ft_3000	Recall_max_ft_3000
0	xgboost_DecisionTreeClassifier	0.981333	0.945455	0.936937	0.928571	0.980000	0.944954	0.932127	0.919643
1	xgboost_RandomForestClassifier	0.960000	0.910000	0.858491	0.812500	0.958667	0.909091	0.853081	0.803571

Fig. 50. Final Compare analysis of Spam or Not Spam Dataset using xgboost

In conclusion, after analyzing the results(Fig 50), we can infer that the xgboost Decision Tree Classifier with the TfidfVectorizer feature extraction method is the most suitable model for the given problem. This is due to its superior performance in terms of accuracy, precision, F1, and recall when compared to the xgboost Random Forest Classifier using both feature extraction methods.

V. Conclusion And Discussion

- In this study, the performance of three popular machine learning algorithms, Naive Bayes, Random Forest Classifier, and Decision Tree Classifier, has been analyzed using two widely-used Python machine learning algorithms, SKlearn and XGboost, on three different spam datasets. The TfidfVectorizer and TfidfVectorizer max feature 3000 are used to extract features from the datasets.

- The experimental results show that XGBoost with the Decision Tree Classifier and the Random Forest Classifier have achieved the highest accuracy, precision, F1-score, and recall on the Email Spam dataset and SMS Spam collection dataset, respectively. In contrast, Naive Bayes achieved the highest performance on the Spam or Not Spam collection dataset.

- Furthermore, the TfidfVectorizer with max feature 3000 consistently has improved the performance of all algorithms in all three datasets. However, the improvement has been observed relatively small in the Email Spam dataset.

- Therefore, based on the experimental results, this study recommends using XGBoost with the Decision Tree Classifier and the Random Forest Classifier on the Email Spam dataset and SMS Spam collection dataset, respectively, and Naive Bayes on the Spam or Not Spam collection dataset for better spam classification performance.

- Overall, the XGBoost algorithm outperformed the SKlearn algorithm on all three datasets. Therefore, the XGBoost algorithm with TfidfVectorizer is the best-fit model for spam detection tasks.

Finally, it is suggested that further studies be conducted with larger datasets and more advanced techniques to improve the performance of spam detection systems in real-world scenarios.

References

[1] TechTalks, By Ben Dickson -November 30, 2020 [Online] Available: https://bdtechtalks.com/2020/11/30/machine-learning-spam-detection/: :text=Spam

[2] D.M. Fonseca, O.H. Fazzion, E. Cunha, I. Las-Casas, P.D. Guedes, W. Meira, M. Chaves Measuring characterizing, and avoiding spam traffic costs IEEE Int. Comp., 99 (2016) [Online] Available: ://ieeexplore.ieee.org/abstract/document/7478420/

[3] Visited on May 15, 2017 Kaspersky Lab Spam Report (2017) 2012 [Online] Available: https://www.securelist.com/en/analysis/204792230/Spam$_Report_April_2012/$

[4] P. Sahil, G. Dishant, A. Mehak, K. Ishita, J. Nishtha Comparison and analysis of spam detection algorithms Int. J. Appl. Innov. Eng. Manag. (IJAIEM), 2 (4) (2013), pp. 1-7 [Online] Available: https://www.sciencedirect.com/science/article/pii/S2405844018353404bib127/

[5] H. Bhuiyan, A. Ashiquzzaman, T. Islam Juthi, S. Biswas, and J. Ara, "A survey of existing e-mail spam filtering methods considering machine learning techniques," Global Journal of Computer Science and Technology, vol. 18, 2018. [Online] Available: https://www.researchgate.net/profile/Hanif-Bhuiyan-2/publication/332865507_____-Mail_____/links/5f12c1704585151299a4b7fb/A-Survey-of-Existing-E-Mail-Spam-Filtering-Methods-Considering-Machine-Learning-Techniques.pdf/

[6] A. Asuncion and D. Newman, "UCI machine learning repository," 2007, [Online] Available: https://archive.ics.uci.edu/ml/index.php./

[7] Abhishek rajput, KDD Process in Data Mining [Online] Available: https://www.geeksforgeeks.org/kdd-process-in-data-mining/

[8] NITISHA, Email Spam Dataset, LingSpam, EnronSpam, Spam Assassin Dataset containing ham and spam email [Online] Available: https://www.kaggle.com/datasets/nitishabharathi/email-spam-dataset/

[9] JEYASRI SENTHIL, Spam Mail Classification [Online] Available: https://www.kaggle.com/code/jeyasrisenthil/spam-mail-classification/

[10] HAKAN OZLER, Spam or Not Spam Dataset A collection of emails taking from spamassassin.apache.org [Online] Available: https://www.kaggle.com/datasets/ozlerhakan/spam-or-not-spam-dataset/

[11] Abhresh Sugandhi, knowledgehut [Online] Available: https://www.knowledgehut.com/blog/business-intelligence-and-visualization/importance-of-data-visualization/

[12] tutorialspoint [Online] Available: https://www.tutorialspoint.com/scikit_learn/index.htm/

[13] Jason Brownlee, guidingtechmedia [Online] Available: https://machinelearningmastery.com/develop-first-xgboost-model-python-scikit-learn/: :text=XGBoost

[14] educative [Online] Available: https://www.educative.io/answers/what-are-naive-bayes-classifiers/

[15] Image—Abid Ali Awan, datacamp, [Online] Available: https://www.datacamp.com/tutorial/decision-tree-classification-python/
[16] Sruthi E R — Published On June 17, 2021 and Last Modified On April 26th, 2023, analyticsvidhya [Online] Available: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/: :text=Random
[17] nalyticsvidhya, [Online] Available: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/: :text=Random