A Project Report

On

SMS Spam Detection Using Machine Learning

Submitted in partial fulfillment of the

requirement for the award of the degree of

BACHELOR OF COMPUTER APPLICATION

Session 2023-24

in

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

By

Aman Pal
21SCSE1430003
Ashish Kumar Jha
21SCSE1430009
Rajeev
21SCSE1480009

Under the guidance of

Ms. Aishwarya
Assistant Professor

SCHOOL OF COMPUTER APPLICATION AND TECHNOLOGY
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
April,2024

SCHOOL OF COMPUTER APPLICATION AND TECHNOLOGY

GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the project, entitled **"SMS SPAM DETECTION USING MACHINE LEARNING"** in partial fulfillment of the requirements for the award of the BCA (Bachelor of Computer Application) submitted in the School of Computer Application and Technology of Galgotias University, Greater Noida, is an original work carried out during the period of January, 2024 to May and 2024, under the supervision of **Ms. Aishwarya Assistant Professor,** Department of School of Computer Application and Technology, Galgotias University, Greater Noida.

The matter presented in the project has not been submitted by us for the award of any other degree of this or any other places.

Aman Pal, 21SCSE1430003

Ashish Kumar Jha, 21SCSE1430009

Rajeev,21SCSE1480009

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Ms. Aishwarya

Assistant Professor

CERTIFICATE

This is to certify that Project Report entitled **"SMS SPAM DETECTION USING MACHINE LEARNING"** which is submitted by Aman Pal(21SCSE1430003), Ashish Kumar Jha(21SCSE1430009), Rajeev(21SCSE1480009) in partial fulfillment of the requirement for the award of degree BCA. in Department of **SCAT** of School of Computer Application and Technology, Galgotias University, Greater Noida, India is a record of the candidate own work carried out by him/them under my supervision. The matter embodied in this thesis is original and has not been submitted for the award of any other degree.

Signature of Examiner(s) **Signature of Supervisor(s)**

Date: April, 2024

Place: Greater Noida

ABSTRACT

In today's digital era, where digitization is pervasive, SMS has emerged as a crucial form of communication. Unlike platforms such as Facebook and WhatsApp, SMS doesn't depend on an active internet connection. However, the prevalence of spam SMS poses a significant threat as it can deceive mobile users into divulging confidential information, leading to severe consequences. Recognizing the gravity of this issue, there is a pressing need to develop an effective spam filtration solution. The model would be trained to identify spam messages based on a variety of features, such as the keywords in the messages was sent. Machine learning algorithms can be used to train a model to identify spam messages, such as Naive Bayes classifiers serve as straightforward and resilient probabilistic classifiers, proving especially valuable in tasks related to text classification. The algorithm operates on the premise of assuming conditional independence among features given a class, providing a practical initial approximation for real-world scenarios. The problem of SMS spam detection is a machine learning model that can accurately identify spam messages in real time. The model is a prediction of whether a given SMS message is spam or ham. The prediction can be used to block the message, or to take other actions, such as warning the user or moving the message to a spam folder. In conclusion, SMS spam detection plays a pivotal role in safeguarding user privacy, enhancing user experience, and ensuring regulatory compliance. It leverages techniques like rule-based filtering, machine learning, and user feedback to effectively identify and filter out spam messages.

TABLE OF CONTENTS

LIST OF FIGURE

LIST OF SYMBOL

SMS	Short Message Service
IDE	Integrated Development Environment
NumPy	Numerical Python
EDA	Exploratory Data Analysis
pd	Pandas
sklearn	Scikit learn
Matplotlib	Mathematical Plotting Library
nltk	Natural Language Toolkit
SVM	Support Vector Machine
KNN	K- Nearest Neighbor
TF-IDF	Term Frequency-Inverse Document Frequency
ROC	Receiver Operating Characteristics
AUC	Area Under Curve
NB	Naïve Bayes
DT	Decision Tree
RF	Random Forest

CHAPTER-1

INTRODUCTION

1.1 Introduction

In today's interconnected world, Short Message Service (SMS) is an integral part of our daily communication. Whether it's sharing updates with friends or receiving important alerts from businesses, SMS is a ubiquitous and convenient means of staying connected. However, this ease of communication has also opened the door to SMS spam, which inundates our inboxes with unwanted and often malicious messages. SMS spam not only disrupts our personal and professional lives but also poses security and privacy risks. The primary objective of SMS spam detection is to automatically identify and filter out these unwanted messages, ensuring that users receive only legitimate and relevant texts while protecting them from potential security threats. This process involves the application of various techniques, including rule-based filtering, machine learning, and user feedback. SMS spam detection is not only essential for user privacy and security but also for regulatory compliance, user experience improvement, and maintaining an efficient and trusted messaging system. As the landscape of mobile communication evolves, the importance of effective SMS spam detection continues to grow, making it a vital area of research and development in information technology and telecommunications.

1.2 Formulation of Problem

SMS spam is a growing problem, with billions of spam messages sent each day. Spam messages can be annoying and inconvenient, but they can also be dangerous, as they can be used to spread malware, phishing attacks, and other scams. This project aims to develop a SMS Spam Detection using the Naïve Bayes algorithm, leveraging a comprehensive dataset that includes a wide array of features, such as Message Text, Message Length, Message Type, Message Structure, Emoticons and Symbols' and more.

In the context of SMS spam detection, the Naive Bayes algorithm uses Bayes' theorem to calculate the probability of a message being spam, given the presence of certain keywords or phrases in the message. To train the Naive Bayes algorithm, it is given a set of labelled SMS messages, where

each message is labelled as either spam or ham. The algorithm then calculates the probabilities of each keyword or phrase occurring in spam and ham messages.

A GUI, or graphical user interface, is a way for users to interact with a computer program using visual elements such as buttons, menus, and text boxes. GUIs are commonly used in SMS spam detection applications to allow users to easily label messages as spam or ham, train the spam detection model, and predict whether new messages are spam or ham.

1.2.1 Tools and Technology Used

In this project, the primary technology utilized is the Naive Bayes algorithm is rooted in Bayes' theorem from machine learning. According to this theorem, the probability of an event occurring, given that another event has already occurred, is determined by multiplying the probability of the first event by the probability of the second event happening given that the first event has occurred. This product is then divided by the probability of the second event occurring.

In the context of SMS spam detection, Naive Bayes can be used to calculate the probability of a message being spam, given the presence of certain keywords or phrases in the message. The algorithm works by first training on a set of labelled SMS messages, where each message is labelled as either spam or ham. The algorithm then calculates the probabilities of each keyword or phrase occurring in spam and ham messages.

For example, let's say that the Naive Bayes algorithm has been trained on a set of labeled SMS messages. The algorithm has computed that the likelihood of the word "free" appearing in a spam message is 0.8, while the probability of encountering the word "free" in a ham message is 0.2.

Now, let's say that the algorithm receives a new SMS message that contains the word "free". The algorithm will calculate the probability of the message being spam as follows:

$$P(spam \mid free) = P(free \mid spam) * P(spam) / P(free)$$

where:

- P(spam | free) is the probability of the message being spam, given that it contains the word "free".
- P(free | spam) is the probability of the word "free" occurring in a spam message.
- P(spam) is the probability of a message being spam.
- P(free) is the probability of the word "free" occurring in any message.

If the probability of the message being spam exceeds 0.5, it will be categorized as spam. Otherwise, if the probability is equal to or less than 0.5, the message will be classified as ham.

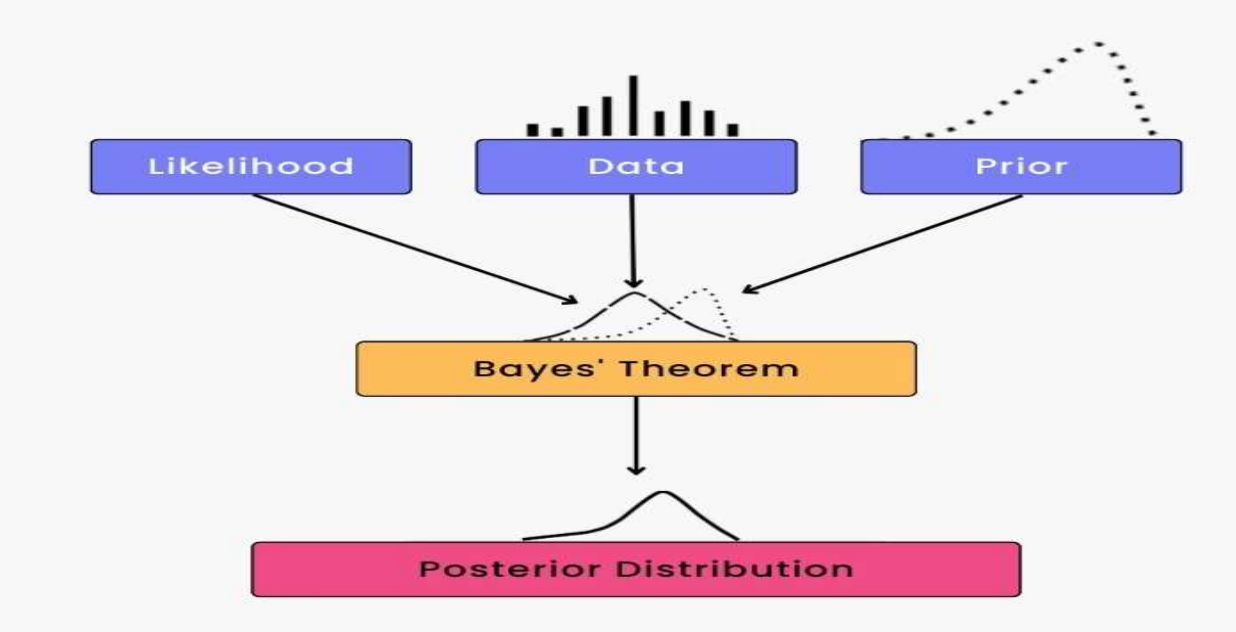


Figure 1: Working of the Naïve Bayes Algorithm

Here are some of the key tools and technologies employed in the project:

1.2.2 Python:

Python stands out as a leading programming language in the realm of machine learning, thanks to its extensive array of libraries and tools. These resources streamline the process of developing, training, and deploying machine learning models. Python's readability, community support, and flexibility contribute to its popularity in the field of machine learning.

1.2.3 Jupyter Notebook IDE:

Jupyter Notebook is a popular integrated development environment (IDE) for machine learning projects. It provides an interactive platform for coding in languages like Python, combining code with documentation and data visualization.

1.2.4 NumPy:

NumPy is a fundamental library in machine learning due to its efficient array operations. It enables data manipulation and numerical computations, making it easier to work with datasets. NumPy's

array structure allows for vectorized operations, improving the speed and efficiency of machine learning algorithms.

1.2.5 Pandas:

Pandas plays a crucial role in the field of machine learning as a key library for data manipulation and analysis. Through its versatile data structures, such as DataFrames, Pandas facilitates the organization and manipulation of structured data. This functionality is particularly vital for tasks like data cleaning. feature extraction, and exploratory data analysis (EDA). Pandas simplifies data handling, making it easier to prepare datasets for machine learning algorithms and gain insights into the data's characteristics and distribution.

1.2.6 Scikit-learn (Sklearn):

Scikit-learn stands out as a freely available, open-source machine learning library designed for Python. It encompasses an extensive array of algorithms, covering classification, regression, clustering, and dimensionality reduction, making it a versatile tool for various machine learning tasks. Scikit-learn is built on top of NumPy, SciPy, and Matplotlib, and is known for its simplicity, efficiency, and extensibility.

1.2.7 Matplotlib:

Matplotlib is a widely used Python library for creating data visualizations, which is essential in machine learning. It offers a range of functions for generating various types of plots and charts, aiding in data exploration, feature visualization, and model performance assessment.

1.2.8 Seaborn:

Seaborn, a Python library for statistical data visualization, is constructed atop Matplotlib. It furnishes a high-level interface, enabling the creation of visually appealing and informative statistical plots with ease. Seaborn is widely used in machine learning to visualize training and testing data, model performance, and predictions.

1.2.9 WordCloud:

The wordcloud library can be used in machine learning to visualize and understand the distribution of words in a dataset. This can be helpful for identifying important keywords, trends, and outliers. Word clouds can also be used to generate creative visualizations of machine learning models and predictions.

1.2.10 Nltk:

The Natural Language Toolkit (NLTK) is a Python library for machine learning that offers essential tools for text processing and natural language understanding. It is widely used for tasks like text classification, sentiment analysis, and language modeling.

CHAPTER-2

LITERATURE REVIEW/ PROJECT DESIGN

2.1 Literature Reviews:

Over the past few years, numerous research endeavors have surfaced in the domain of SMS spam detection and classification. These studies have explored various machine learning techniques, including Support Vector Machine (SVM), Decision Tree, K-Nearest Neighbor (KNN), Random Forest, among others.

- In their paper [4] Gupta, Suparna Das, Soumyabrata Saha, showed that machine learning algorithm can be utilized to identify spam SMS messages, achieving a 95% accuracy rate. They employed the TF-IDF vectorization algorithm for data preprocessing using the UCI Machine Learning Repository dataset.
- Research [5] describes that the data The data underwent preprocessing, involving the removal of stop words, and feature extraction was carried out using Wordnet Lemmatizer for tokenization and Count Vectorizer for converting words to vectors. The research employed various machine learning algorithms, including naive Bayes, logistic regression, random forest, and decision tree classifiers. Notably, the random forest classifier demonstrated superior effectiveness, achieving an accuracy of 98.13% in the detection of SMS spam messages.
- The article [6] discusses the growing threat of spam SMS and how the exploratory analysis was conducted on a sample dataset from UCI to find an effective solution. The analysis showed that the length feature was important in distinguishing between ham and spam, but there was a high imbalance in the dataset between the two classes. The dataset underwent pre-processing and cleaning to mitigate noise, with features extracted through the implementation of the Bag of Words model. Subsequently, machine learning models, including XGBoost, decision tree, and Random Forest, were employed on the dataset, revealing XGBoost as the most effective model among the options.
- In their paper [7] Housemand Shirani-Mehr, In the original paper referencing this dataset, the most effective classifier employs SVM as the learning algorithm, achieving an impressive overall accuracy of 97.64%. Following closely is the boosted naive Bayes

classifier with an overall accuracy of 97.50%. When compared to the outcomes of prior research, our classifier substantially diminishes the overall error, cutting it by more than half.

- In their paper [8] M. Rubin Julis, S. Alagesan, the paper investigates various machine learning classifiers using a large corpus of SMS messages for individuals. The Accuracy Comparison results indicate that Support Vector Machines achieve the highest accuracy score at 98%, surpassing all other algorithms. In terms of the Training Set Comparison, Support Vector Machines with the RBF kernel are observed to undergo more frequent training sessions than any other algorithms.
- In their paper [9] Mr. E.Sankar, Y Y S Shekhar Babu, M.Tridev, B.E, The collaborative approach of Naive Bayes and FP-Growth yields the highest average accuracy at 98.506%, surpassing the accuracy achieved without utilizing FP-Growth for the SMS Spam Collection v. 1 dataset by 0.025%. This improvement contributes to an enhanced perfection score, establishing the combined result as more accurate.
- [10] Despite the proliferation of various communication mediums through messenger applications on mobile phones, SMS (Short Message Service) continues to remain the primary choice for communication.
- [11] In today's digitized world, where technology is omnipresent, SMS has emerged as a crucial means of communication. This sets it apart from other chat-based messaging systems such as Facebook and WhatsApp.
- [12] While mobile messaging channels are currently perceived as "clean" and reliable in many parts of the world, recent reports starkly reveal a significant daily increase in the volume of smartphone spam.
- [13] Electronic mail is undergoing a vital revolution, supplanting traditional communication systems due to its convenient, economical, fast, and user-friendly nature.

2.2 Problem Definition:

The SMS spam detection problem revolves around the need to identify and mitigate the ever-present issue of unsolicited and potentially harmful SMS messages, commonly referred to as spam, in the world of mobile communication. The primary challenge is to create a system capable of

distinguishing between legitimate SMS messages that users want to receive and unwanted or malicious ones that can range from annoying advertisements and scams to phishing attempts and fraudulent schemes. The core objective of SMS spam detection is to automatically classify incoming text messages as either spam or legitimate, allowing users to have cleaner and more secure SMS inboxes while ensuring their privacy and user experience. This problem is multifaceted, involving data collection, feature engineering, machine learning model development, and real-time message processing. By addressing this problem, organizations can enhance user satisfaction, protect user privacy, maintain regulatory compliance, and uphold the trustworthiness and security of SMS communication.

2.2.1 Key Components of SMS Spam Detection:

1. **Message Content Analysis:** One of the primary methods for detecting SMS spam involves analyzing the content of the message. This analysis may involve text classification techniques, where machine learning models are trained to differentiate between legitimate and spam messages based on keywords, message structure, or other text features. Certain words or phrases associated with spam are used as indicators.
2. **Sender Reputation:** Another aspect of SMS spam detection is examining the sender's reputation. Legitimate organizations and individuals typically have established reputations, while spammers often use anonymous or temporary phone numbers. Systems can flag messages from unknown or suspicious senders for further inspection.
3. **User Feedback:** Many SMS applications allow users to report spam messages. User feedback is valuable in improving detection systems. Messages that receive multiple user reports are often marked as spam.
4. **Machine Learning:** Machine learning algorithms are widely used for SMS spam detection. These algorithms are trained on labeled datasets, with examples of both spam and legitimate messages. They learn to recognize patterns and features associated with each category and can subsequently classify new messages.

2.3 Data Flow Diagram:

A Data Flow Diagram (DFD) is a graphical representation that visually depicts the flow of data within a system. Employing symbols to represent processes, data stores, data flows, and external

entities, DFDs offer a comprehensive illustration of how information traverses a system. This visualization facilitates a clear understanding and design of information processes. Widely utilized in system analysis and design, DFDs serve as effective tools for modeling and documenting data interactions.

2.3.1 Front End Model Diagram

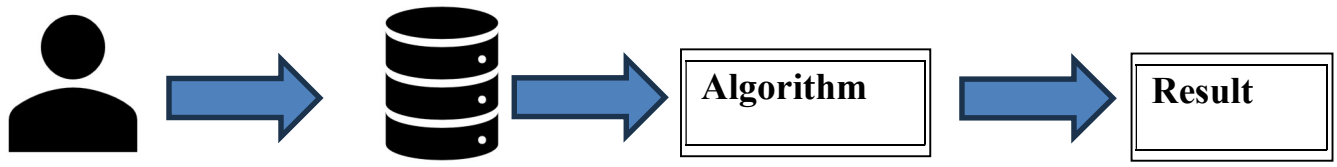


Figure 2: Front end Diagram

2.3.2 Back End Module Diagram

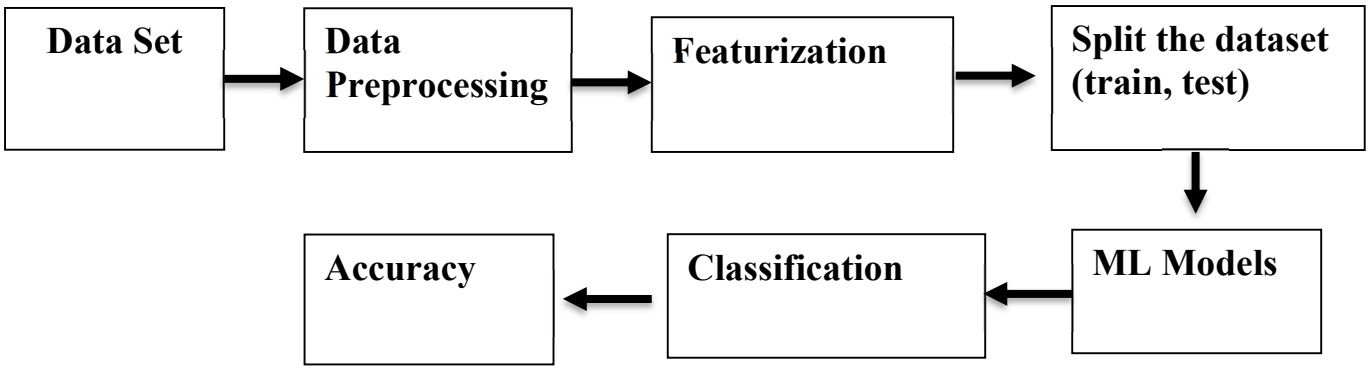


Figure 3: Backend Diagram

2.4 System Design

System design involves delineating the architecture, components, modules, interfaces, and data of a system to meet predefined requirements. It involves translating system requirements into a detailed design that serves as a blueprint for the actual implementation. Key considerations include

hardware and software specifications, data organization, and overall system structure. The goal is to create a robust and efficient solution that meets the intended purpose.

2.4.1 Architecture Design

Architecture design involves creating a high-level structure for a system, defining its components, modules, and their relationships. It focuses on achieving system objectives while addressing key architectural concerns such as scalability, performance, and maintainability. The outcome is a blueprint that guides the implementation and ensures the system's coherence and effectiveness.

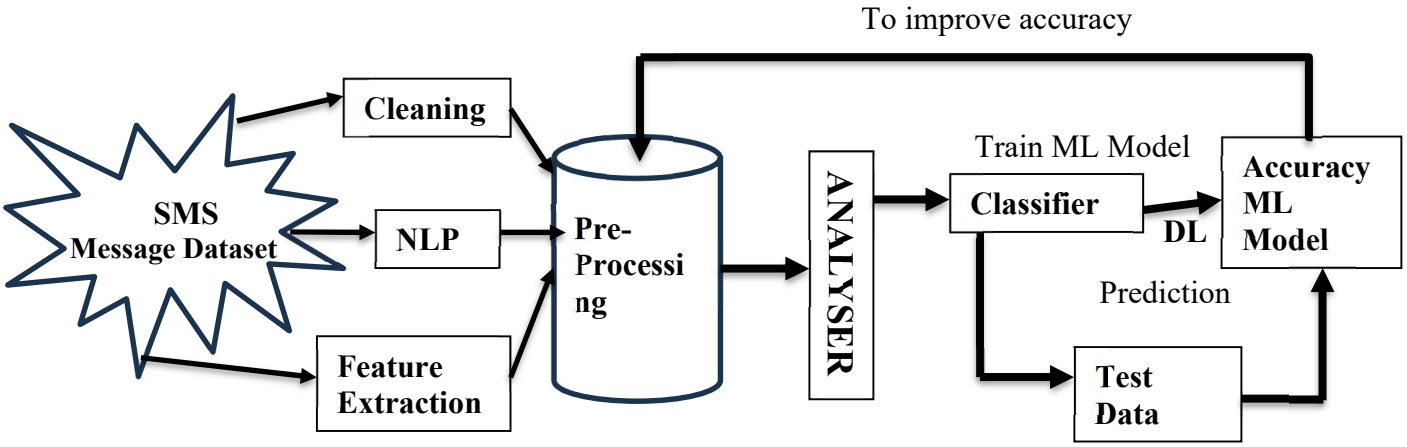


Figure 4: Architecture Diagram

2.4.2 Use Case Diagram

Use case design is a process within system development that identifies, clarifies, and models the interactions between users and a system. It involves creating detailed scenarios or use cases to illustrate how users interact with the system to accomplish specific goals. Use case diagrams help depict these interactions visually, outlining the sequence of events and system responses. Use case

design aids in understanding user requirements, guiding system development, and ensuring the alignment of features with user needs for effective software or system functionality.

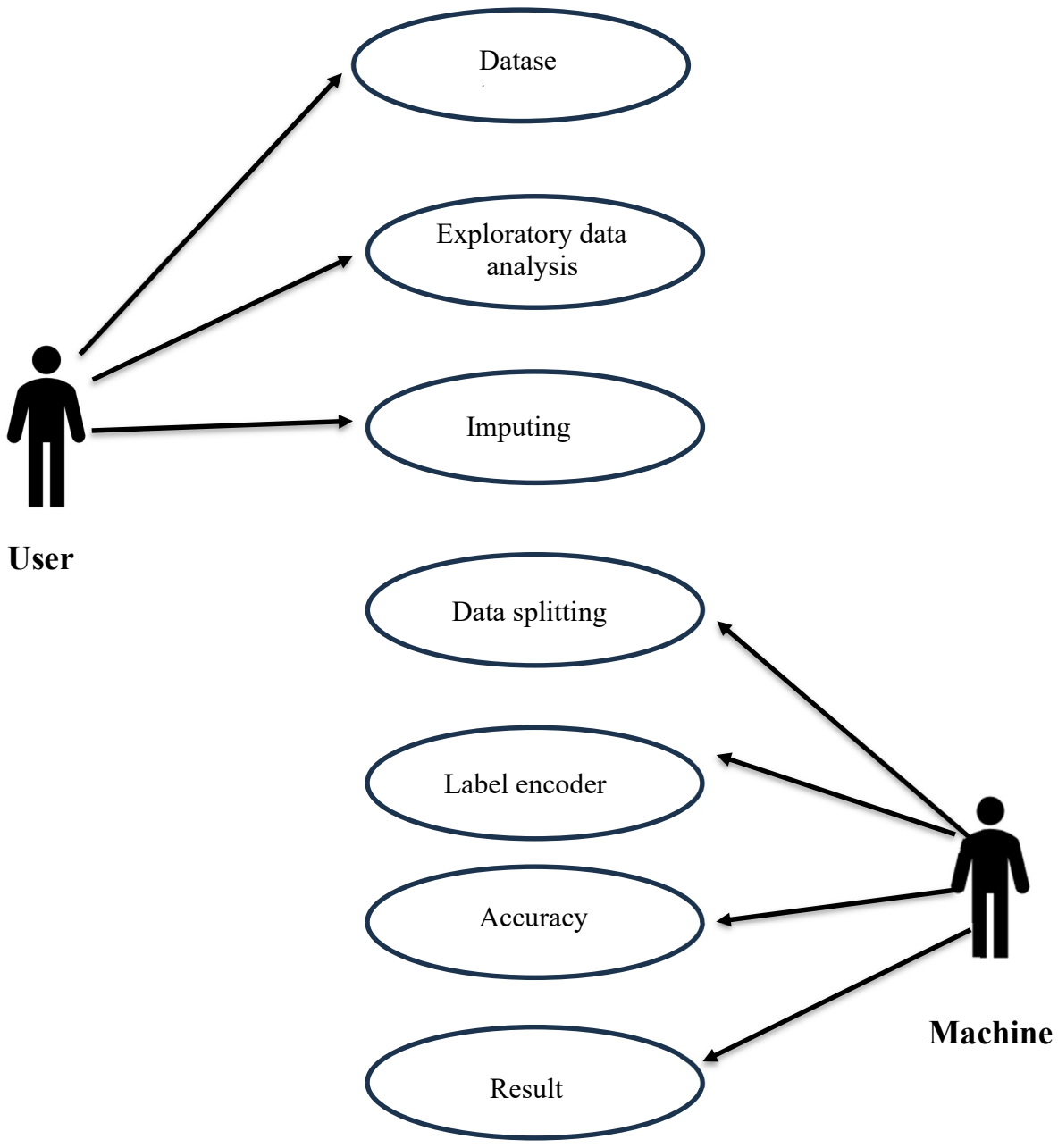


Figure 5: Use Case Diagram

2.4.3 Sequence diagram

Sequence design is a phase in software development that focuses on detailing the interactions and order of execution among objects or components within a system. It involves creating sequence diagrams to illustrate the flow of messages and actions between these entities during specific scenarios or processes. This design phase aids developers in understanding and implementing the dynamic aspects of a system's functionality.

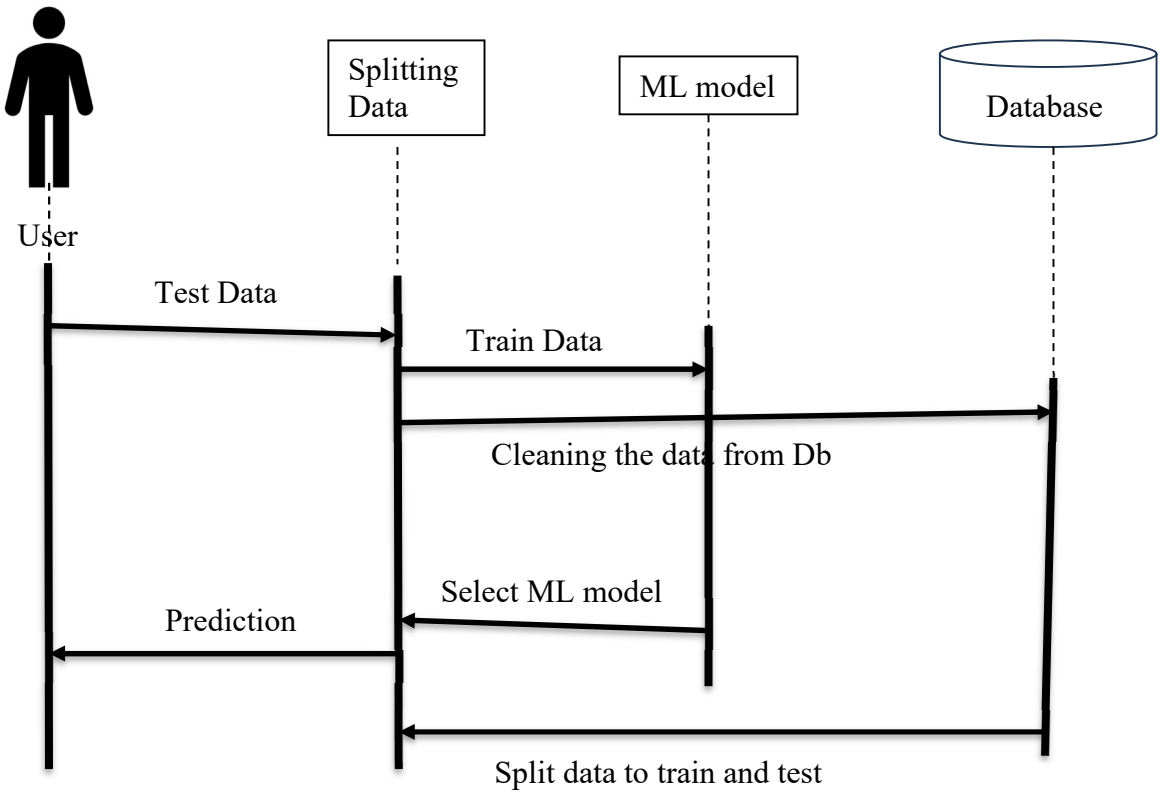


Figure 6: Sequence diagram

2.5 Methodology:

2.5.1 Data Collection:

We utilize a database comprising 5574 text messages sourced from the UCI Machine Learning repository, compiled in 2012. This database encompasses a set of 425 SMS spam messages that were manually extracted from the Grumble text website (e.g., spam.csv).

2.5.2 Data cleaning:

Within a machine learning project, data cleaning encompasses the identification and correction of errors, inconsistencies, and missing values in the dataset. This process is crucial for ensuring that the data used for model training and analysis is of high quality, reliable, and accurate. It includes tasks like handling missing data, removing duplicates, addressing outliers, and correcting inconsistent or erroneous values. Data cleaning is a crucial step to prevent bias and errors in machine learning models, ultimately leading to better predictive performance.

2.5.3 Exploratory Data Analysis (EDA):

Exploratory Data Analysis (EDA) in a machine learning project involves data profiling, visualization, and statistical analysis to understand the dataset's characteristics, patterns, and relationships. It helps identify missing data, outliers, and opportunities for feature engineering, improving data quality and model performance.

Here are some common EDA steps that are used in SMS spam detection:

- **Imbalance Assessment:** EDA also helps in assessing the class distribution. Detecting imbalanced data (where spam messages may be significantly less frequent than non-spam) can guide later steps in the project, like sampling techniques or model evaluation strategies.
- **Pattern Identification:** EDA aids in identifying patterns in spam messages, such as common keywords, phrases, or characteristics that distinguish them from legitimate messages. This knowledge can guide feature selection and model building.
- **Statistical Analysis:** EDA can involve statistical analysis to calculate summary statistics for both spam and non-spam categories. This helps in identifying differences in message length, word frequency, and other relevant attributes.

2.5.4 Data Preprocessing:

Data preprocessing stands as a crucial step in every machine learning endeavor, including SMS spam detection. This step entails the cleaning and transformation of data to enhance its suitability for the machine learning algorithm.

Here are some common data preprocessing steps that are used in SMS spam detection:

- **Remove stop words**: Stop words, which include common terms like "the," "is," and "of," contribute minimal meaning to the text. Eliminating stop words proves beneficial in reducing the dataset size and enhancing the accuracy of the machine learning model.

- **Stem or lemmatize words:** Stemming and lemmatization are two techniques for reducing words to their root forms. Enhancing the accuracy of the machine learning model is achievable by enabling the model to recognize related words, even when they exhibit different spellings.
- **Convert to lowercase:** Lowercasing all words in the dataset can enhance the accuracy of the machine learning model by diminishing the count of unique words within the dataset.
- **Handle missing values:** Missing values are data points that are not present in the data set. There are a variety of ways to handle missing values, such as removing the data points, imputing values, and using machine learning algorithms to predict the missing values.
- **Feature engineering:** Feature engineering involves crafting new features from the available data. This practice is employed to enhance the accuracy of the machine learning model or to tailor the dataset to better align with the requirements of the specific machine learning algorithm in use.

2.5.5 Model Selection:

Naive Bayes stands out as a straightforward yet powerful machine learning algorithm designed for classification tasks. Its foundation lies in Bayes' theorem, a mathematical formula used to calculate the probability of an event happening when another event has already occurred.

In the realm of SMS spam detection, Naive Bayes can be applied to determine the likelihood of a new SMS message being classified as either spam or ham (legitimate). This determination is based on the probability of each word in the message occurring in both spam and ham messages.

For example, let's say we have the following two SMS messages:

Spam message: I love you, baby. Please send me $100.

Ham message: I am going to the store. Do you need anything?

Leveraging Naive Bayes allows us to compute the probability of each word in these messages appearing in either spam or ham messages. For instance, the word "love" is more inclined to occur in a spam message than in a ham message, while the word "store" is more likely to be found in a ham message than in a spam message.

After determining the probability of each word in the new message occurring in both spam and ham messages using Naive Bayes, we can then utilize the algorithm to calculate the overall

probability of the new message being categorized as spam or ham. The classification is determined by comparing the probability of the message being spam to that of it being ham. If the probability of the message being spam is higher, the message is classified as spam.

Here's a simplified example illustrating how Naive Bayes functions for SMS spam detection:

New message: "I love you, baby. Please send me $100."

Probability of the message being spam:

P(spam) * P(I love you) * P(baby) * P(please send me) * P($100)

Probability of the message being ham:

P(ham) * P(I love you) * P(baby) * P(please send me) * P($100)

If the likelihood of the message being spam surpasses the likelihood of it being ham, the message is categorized as spam.

2.5.6 Model Training:

Partition the data into training and test sets. The training set is employed to train the model, enabling it to learn patterns and relationships between the selected features and the labels indicating spam or legitimacy. The test set, on the other hand, is utilized to assess the model's performance.

2.5.7 Model Evaluation:

1.Testing Dataset:

The testing dataset is a separate portion of your data that the model hasn't seen during the training phase. It serves as an independent sample for evaluating the model's performance.

2. Performance Metrics:

Use various performance metrics to assess how well the model classifies SMS messages. Common metrics include:

- **Accuracy:** The proportion of correctly classified messages (both spam and legitimate).
- **Precision:** The true positive rate represents the proportion of correctly classified spam messages to the total number of predicted spam messages. This metric gauges the model's capacity to minimize false positives.

- **Recall:** The true positive rate is the ratio of correctly predicted spam messages to the overall number of actual spam messages. This metric reflects the model's proficiency in identifying all spam messages, thereby reducing instances of false negatives.
- **F1-Score:** The F1 score, a harmonic mean of precision and recall, offers a balanced assessment of a model's performance.
- **Receiver Operating Characteristic (ROC) Curve and Area Under the Curve (AUC):** In the context of binary classification models, the ROC curve and AUC evaluate the model's effectiveness in discerning between spam and legitimate messages across varying thresholds.

3. Confusion Matrix:

Create a confusion matrix to visualize the model's performance. It includes true positives, true negatives, false positives, and false negatives, providing a clear summary of the model's classifications.

Let's consider a binary classification task where we aim to classify SMS messages as either "spam" or "legitimate (ham)."

- **True Positives (TP):** These instances denote the model's accurate predictions of "spam." In the context of SMS spam detection, TP signifies spam messages that the model correctly identified.
- **True Negatives (TN):** These instances represent the model's accurate predictions of "legitimate." In the SMS context, TN denotes legitimate (ham) messages that the model correctly identified as not being spam.
- **False Positives (FP):** These instances signify situations where the model inaccurately predicted "spam" when the actual category was "legitimate." In SMS spam detection, FP stands for legitimate messages that were erroneously classified as spam. False positives are alternatively referred to as Type I errors.
- **False Negatives (FN):** These instances highlight situations where the model inaccurately predicted "legitimate" when the actual category was "spam." In SMS spam detection, FN stands for spam messages that were not accurately identified as spam. False negatives are alternatively termed Type II errors.

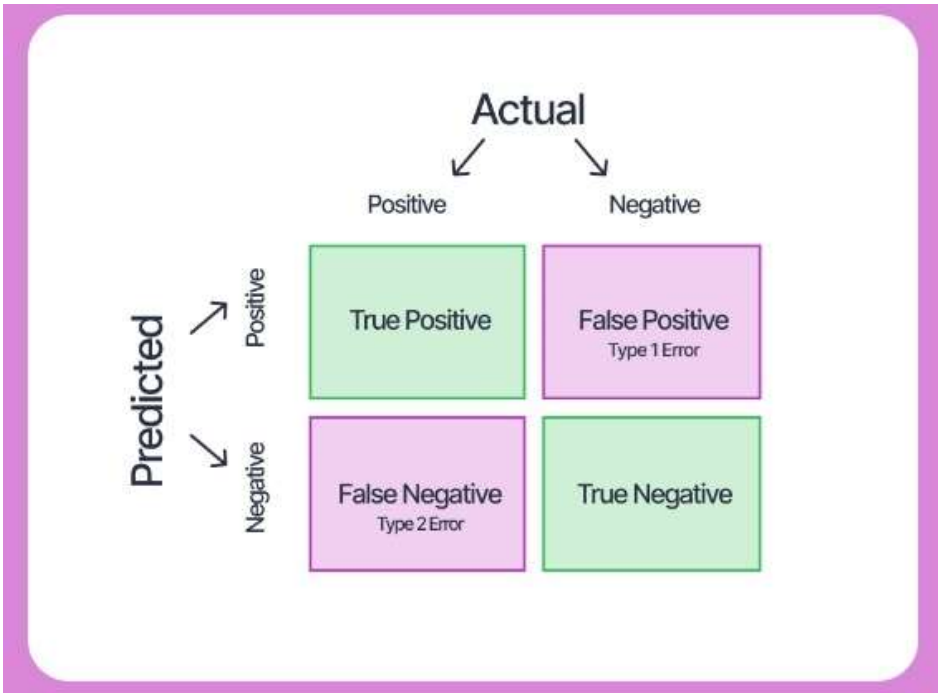


Figure 7: Confusion Matrix

2.5.8 Graphical User Interface

We have used Streamlit which is a Python library for creating web applications and interactive dashboards with simplicity and speed. With Streamlit, we can turn our data scripts into shareable web apps by adding a few lines of code. It offers widgets for user interaction, seamless integration with data science libraries, and automatic updates, making it an ideal choice for building web-based data visualizations and tools with minimal development effort.

CHAPTER-3

WORKING OF PROJECT

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

In [2]:
```
df = pd.read_csv('spam.csv')
```

In [3]:
```
df.sample(10)
```

Out[3]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
3114	ham	I wait 4 Ì_ inside da car park...	NaN	NaN	NaN
3498	ham	I hope you arnt pissed off but id would really...	NaN	NaN	NaN
4102	spam	U have a secret admirer who is looking 2 make ...	NaN	NaN	NaN
1685	ham	Cramps stopped. Going back to sleep	NaN	NaN	NaN
1092	ham	Where r we meeting?	NaN	NaN	NaN
1224	spam	You are a winner U have been specially selecte...	NaN	NaN	NaN
5098	spam	TheMob>Hit the link to get a premium Pink Pant...	NaN	NaN	NaN
49	ham	U don't know how stubborn I am. I didn't even ...	NaN	NaN	NaN
1862	ham	The last thing i ever wanted to do was hurt yo...	NaN	NaN	NaN
3864	ham	THATåÕS ALRITE GIRL, U KNOW GAIL IS NEVA WRONG...	NaN	NaN	NaN

In [4]:
```
df.shape
```

Out[4]:
```
(5572, 5)
```

5.1 Data Cleaning

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   v1          5572 non-null   object
 1   v2          5572 non-null   object
 2   Unnamed: 2  50 non-null     object
 3   Unnamed: 3  12 non-null     object
 4   Unnamed: 4  6 non-null      object
dtypes: object(5)
memory usage: 217.8+ KB
```

In [6]:
```
# drop last 3 columns
df.drop(columns=['Unnamed: 2','Unnamed: 3','Unnamed: 4'],inplace=True)
```

In [7]:
```
df.sample(10)
```

Out[7]:

	v1	v2
4901	spam	* FREE* POLYPHONIC RINGTONE Text SUPER to 8713...
4408	spam	For your chance to WIN a FREE Bluetooth Headse...
1519	ham	Check wid corect speling i.e. Sarcasm
4638	ham	Of course. I guess god's just got me on hold r...
5460	spam	December only! Had your mobile 11mths+? You ar...
3293	ham	Beautiful tomorrow never comes.. When it comes...
300	ham	Awesome, I remember the last time we got someb...
3321	ham	Ok darlin i supose it was ok i just worry too ...
4962	ham	I want to see your pretty pussy...
2117	ham	Wish u many many returns of the day.. Happy bi...

In [8]:
```
# renaming the columns
df.rename(columns={'v1': 'target','v2': 'text'}, inplace= True)
df.sample(5)
```

Out[8]:

	target	text
4471	spam	3. You have received your mobile content. Enjoy
2924	ham	Are you coming to day for class.
1609	ham	You all ready for * big day tomorrow?
4324	ham	Am only searching for good dual sim mobile pa.
2083	ham	you are sweet as well, princess. Please tell m...

In [9]:
```
# To convert categorical values (target) into 0 and 1
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
```

In [10]:
```
df['target'] = encoder.fit_transform(df['target'])
```

In [11]:
```
df.head()
```

Out[11]:

	target	text
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...

In [12]:
```
# missing values
df.isnull().sum()
```

Out[12]:
```
target    0
text      0
dtype: int64
```

In [13]:
```
# duplicate values
df.duplicated().sum()
```

Out[13]:
```
403
```

In [14]:
```
# remove duplicate values
df= df.drop_duplicates(keep='first')
```

In [15]:
```
# again check duplicate values
df.duplicated().sum()
```

Out[15]:
```
0
```

In [16]:
```
df.shape
```

Out[16]:
```
(5169, 2)
```

5.2 Exploratory Data Analysis (EDA)

```
In [17]: df.head()
```

Out[17]:

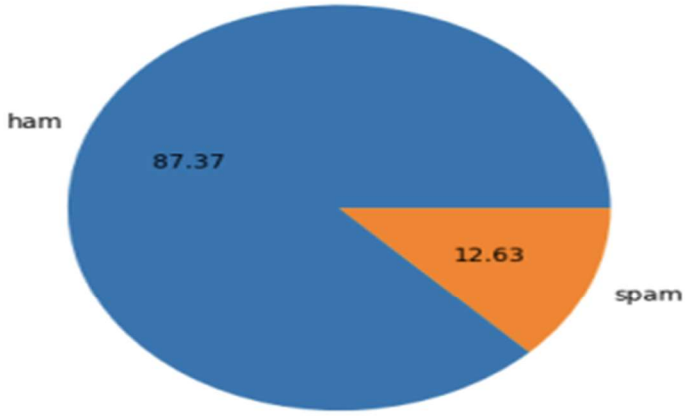
	target	text
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...

```
In [18]: df['target'].value_counts()
```

```
Out[18]: 0    4516
         1     653
         Name: target, dtype: int64
```

```
In [19]: import matplotlib.pyplot as plt
         plt.pie(df['target'].value_counts(), labels=['ham','spam'], autopct="%0.2f"
```

```
Out[19]: ([<matplotlib.patches.Wedge at 0x23b7fbd1e50>,
           <matplotlib.patches.Wedge at 0x23b7fbdc310>],
          [Text(-1.0144997251399075, 0.4251944351600247, 'ham'),
           Text(1.014499764949479, -0.4251943401757036, 'spam')],
          [Text(-0.5533634864399495, 0.23192423736001344, '87.37'),
           Text(0.5533635081542612, -0.23192418555038377, '12.63')])
```




```
In [20]: # Data is imbalanced
```

```
In [21]: nltk.download('punkt')
```

In [22]:
```
# Natural Language toolkit
import nltk
```

In [23]:
```
#finding number of characters
df.loc[:,"num_characters"] = df['text'].apply(len)
```

In [24]:
```
df.head()
```

Out[24]:

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

In [25]:
```
#finding number of words
df.loc[:,'num_words']=df['text'].apply(lambda x:len(nltk.word_tokenize(x)))
```

In [26]:
```
df.head()
```

Out[26]:

	target	text	num_characters	num_words
0	0	Go until jurong point, crazy.. Available only ...	111	24
1	0	Ok lar... Joking wif u oni...	29	8
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37
3	0	U dun say so early hor... U c already then say...	49	13
4	0	Nah I don't think he goes to usf, he lives aro...	61	15

In [27]:
```
#finding number of sentences
df.loc[:,'num_sentences']=df['text'].apply(lambda x:len(nltk.sent_tokenize(
```

In [28]:
```
df.head()
```

Out[28]:

	target	text	num_characters	num_words	num_sentences
0	0	Go until jurong point, crazy.. Available only ...	111	24	2
1	0	Ok lar... Joking wif u oni...	29	8	2
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2
3	0	U dun say so early hor... U c already then say...	49	13	1
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1

In [29]:
```
df[['num_characters','num_words','num_sentences']].describe()
```

Out[29]:

	num_characters	num_words	num_sentences
count	5169.000000	5169.000000	5169.000000
mean	78.977945	18.455794	1.965564
std	58.236293	13.324758	1.448541
min	2.000000	1.000000	1.000000
25%	36.000000	9.000000	1.000000
50%	60.000000	15.000000	1.000000
75%	117.000000	26.000000	2.000000
max	910.000000	220.000000	38.000000

In [30]:
```
#not spam
df[df['target']==0][['num_characters','num_words','num_sentences']].describ
```

Out[30]:

	num_characters	num_words	num_sentences
count	4516.000000	4516.000000	4516.000000
mean	70.459256	17.123782	1.820195
std	56.358207	13.493970	1.383657
min	2.000000	1.000000	1.000000
25%	34.000000	8.000000	1.000000
50%	52.000000	13.000000	1.000000
75%	90.000000	22.000000	2.000000
max	910.000000	220.000000	38.000000

In [31]:
```
#not spam
df[df['target']==1][['num_characters','num_words','num_sentences']].describ
```

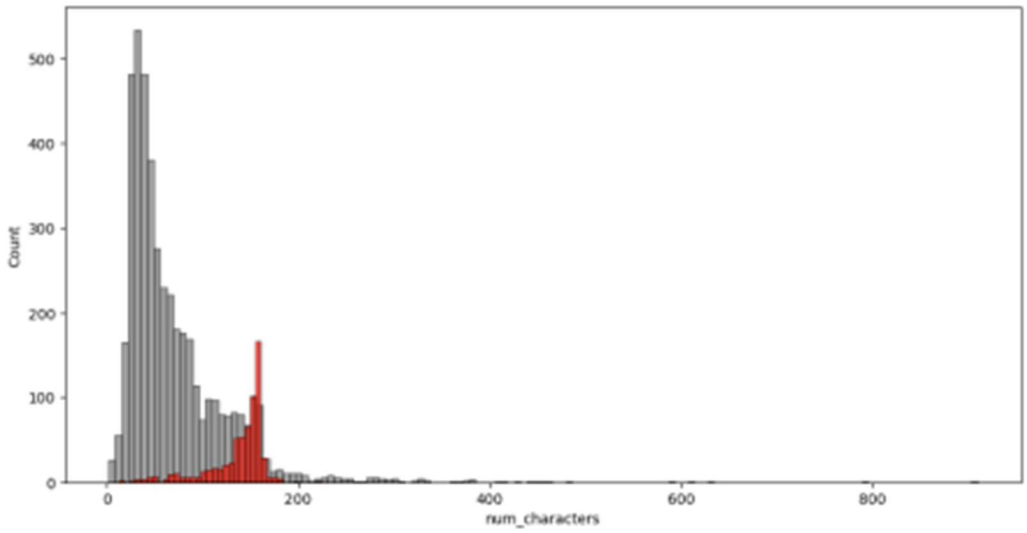
Out[31]:

	num_characters	num_words	num_sentences
count	653.000000	653.000000	653.000000
mean	137.891271	27.667688	2.970904
std	30.137753	7.008418	1.488425
min	13.000000	2.000000	1.000000
25%	132.000000	25.000000	2.000000
50%	149.000000	29.000000	3.000000
75%	157.000000	32.000000	4.000000
max	224.000000	46.000000	9.000000

In [32]:
```
import seaborn as sns
```

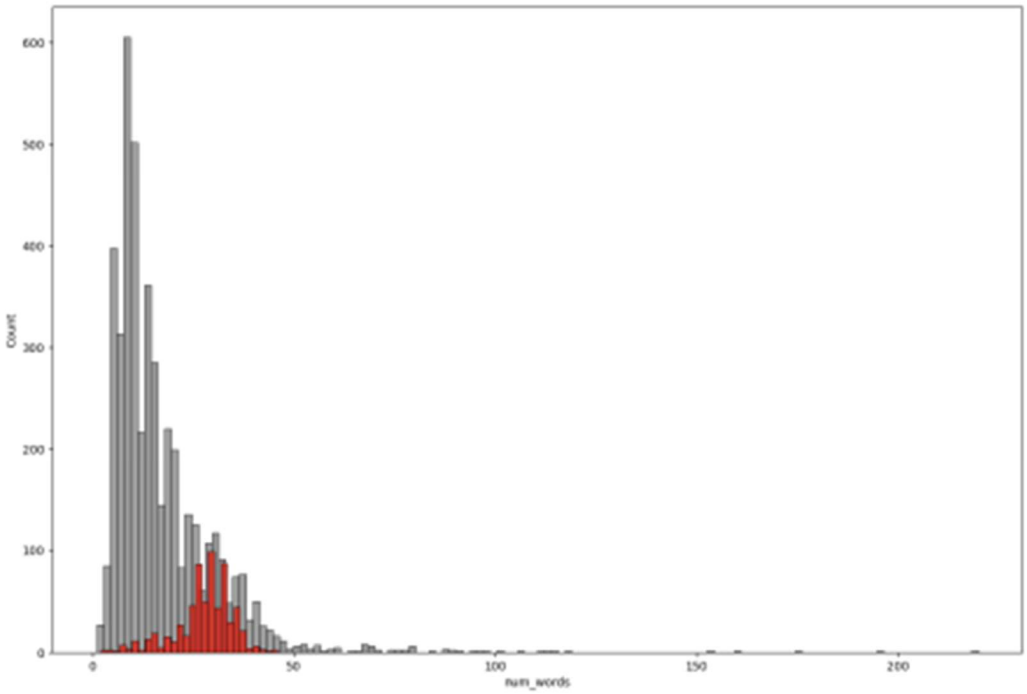
In [33]:
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target']==0]['num_characters'],color='gray')
sns.histplot(df[df['target']==1]['num_characters'],color='red')
```

Out[33]: <Axes: xlabel='num_characters', ylabel='Count'>



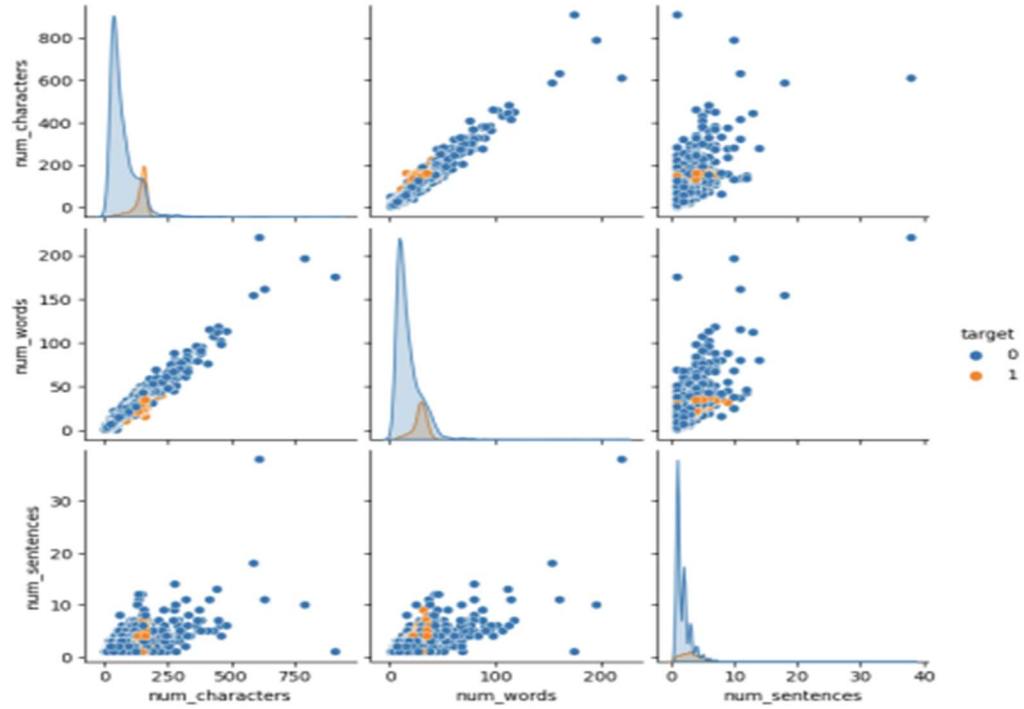
In [34]:
```
plt.figure(figsize=(15,10))
sns.histplot(df[df['target']==0]['num_words'],color='gray')
sns.histplot(df[df['target']==1]['num_words'],color='red')
```

Out[34]: <Axes: xlabel='num_words', ylabel='Count'>



In [35]:
```
sns.pairplot(df,hue='target')
```

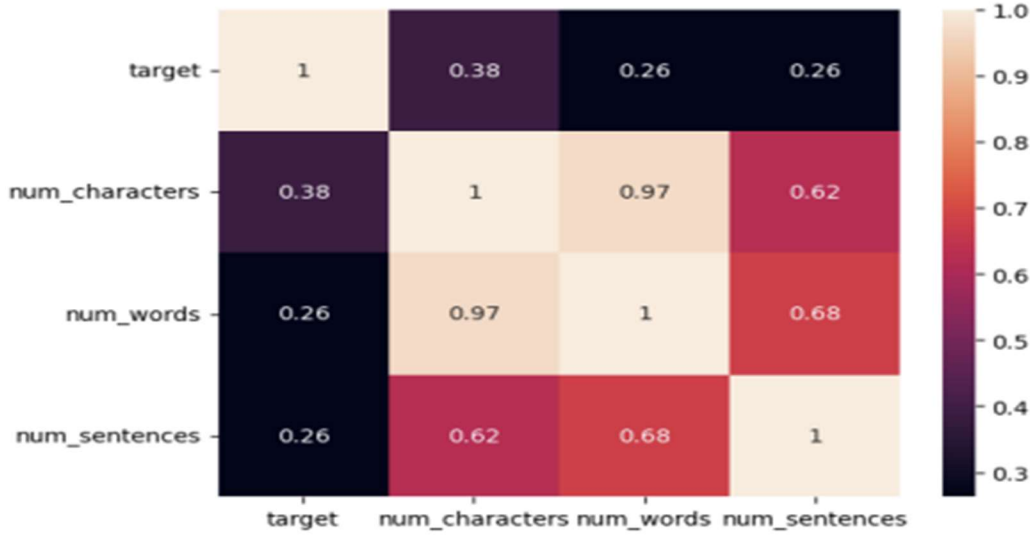
Out[35]: <seaborn.axisgrid.PairGrid at 0x23b03aa9390>



In [36]:
```
sns.heatmap(df.corr(),annot=True)
```

```
C:\Users\Ashish Jha\AppData\Local\Temp\ipykernel_27140\4277794465.py:1: Fu
tureWarning: The default value of numeric_only in DataFrame.corr is deprec
ated. In a future version, it will default to False. Select only valid col
umns or specify the value of numeric_only to silence this warning.
  sns.heatmap(df.corr(),annot=True)
```

Out[36]: <Axes: >



5.3 Data Preprocessing

In [38]:
```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
```

In [39]:
```
def transform_text(text):
    text= text.lower()
    text= nltk.word_tokenize(text)

    y=[]
    for i in text:
        if i.isalnum():
            y.append(i)

    text=y[:]
    y.clear()
    for i in text:
        if i not in stopwords.words('english') and i not in string.punctuat
            y.append(i)
    text=y[:]
    y.clear()
    for i in text:
        y.append(ps.stem(i))
    return " ".join(y)
```

In [40]:
```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to C:\Users\Ashish
[nltk_data]     Jha\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[40]:
```
True
```

In [41]:
```
from nltk.corpus import stopwords
```

In [42]:
```
transform_text("I'm gonna be home soon and i don't want to talk about this
```

Out[42]:
```
'gon na home soon want talk stuff anymor tonight k cri enough today'
```

In [43]:
```
df['text'][10]
```

Out[43]:
```
"I'm gonna be home soon and i don't want to talk about this stuff anymore
tonight, k? I've cried enough today."
```

In [44]:
```
ps.stem('dancing')
```

Out[44]:
```
'danc'
```

In [45]:
```
df.loc[:,'transformed_text']=df['text'].apply(transform_text)
```

In [46]:
```
df.head()
```

Out[46]:

	target	text	num_characters	num_words	num_sentences	transformed_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail bugi n great world...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2	free entri 2 wkli comp win fa cup final tkt 21...
3	0	U dun say so early hor... U c already then say...	49	13	1	u dun say earli hor u c alreadi say
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1	nah think goe usf live around though

In [47]:
```
#pip install wordcloud
```

In [48]:
```
from wordcloud import WordCloud
wc = WordCloud(width=500,height=500,min_font_size=10,background_color='whit
```

In [49]:
```
spam_wc=wc.generate(df[df['target']==1]['transformed_text'].str.cat(sep=""))
```

`

In [50]:
```
plt.figure(figsize=(15,6))
plt.imshow(spam_wc)
```

Out[50]:
```
<matplotlib.image.AxesImage at 0x23b062f0d10>
```

In [51]:
```
ham_wc=wc.generate(df[df['target']==0]['transformed_text'].str.cat(sep=""))
```

In [52]:
```
plt.figure(figsize=(15,6))
plt.imshow(spam_wc)
```

Out[52]:
```
<matplotlib.image.AxesImage at 0x23b06378d10>
```

In [53]:
```
df.head()
```

Out[53]:

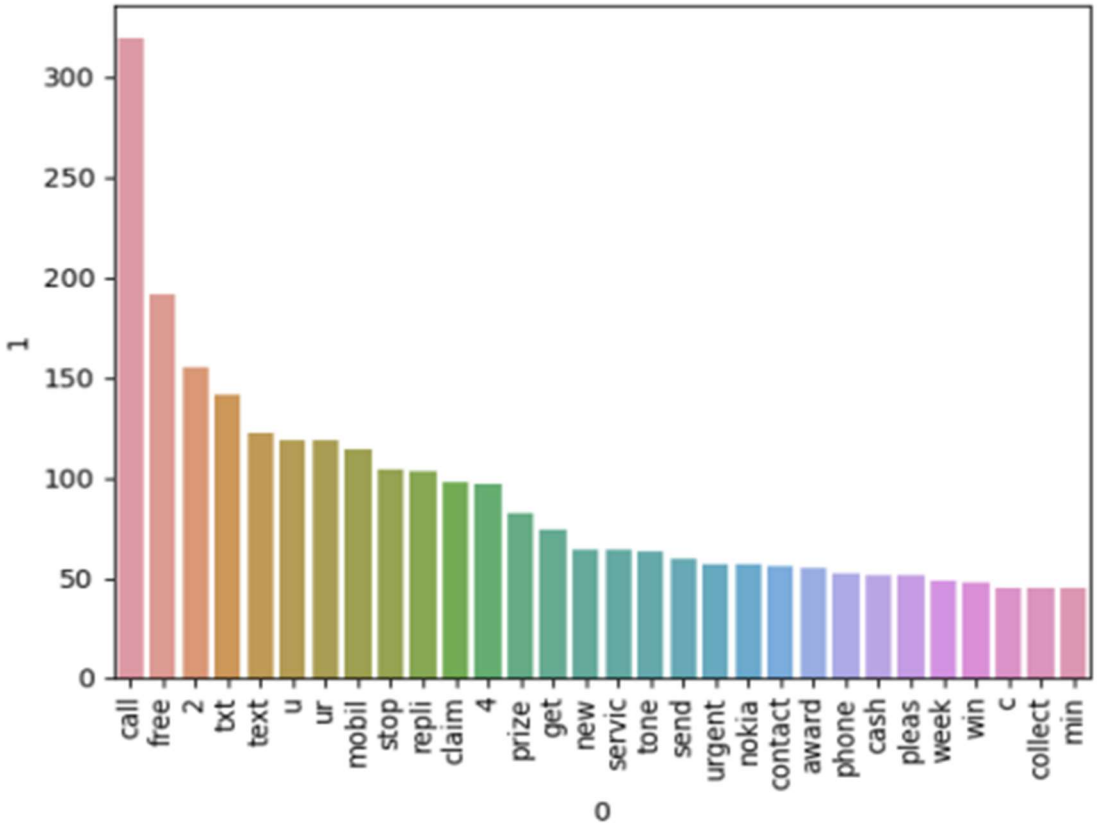
	target	text	num_characters	num_words	num_sentences	transformed_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail bugi n great world...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2	free entri 2 wkli comp win fa cup final tkt 21...
3	0	U dun say so early hor... U c already then say...	49	13	1	u dun say earli hor u c alreadi say
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1	nah think goe usf live around though

In [54]:
```
spam_corpus=[]
for msg in df[df['target']==1]['transformed_text'].tolist():
    for word in msg.split():
        spam_corpus.append(word)
```

In [55]:
```
len(spam_corpus)
```

Out[55]: 9939

In [56]:
```
from collections import Counter
sns.barplot(x=pd.DataFrame(Counter(spam_corpus).most_common(30))[0],y=pd.Da
plt.xticks(rotation='vertical')
plt.show()
```

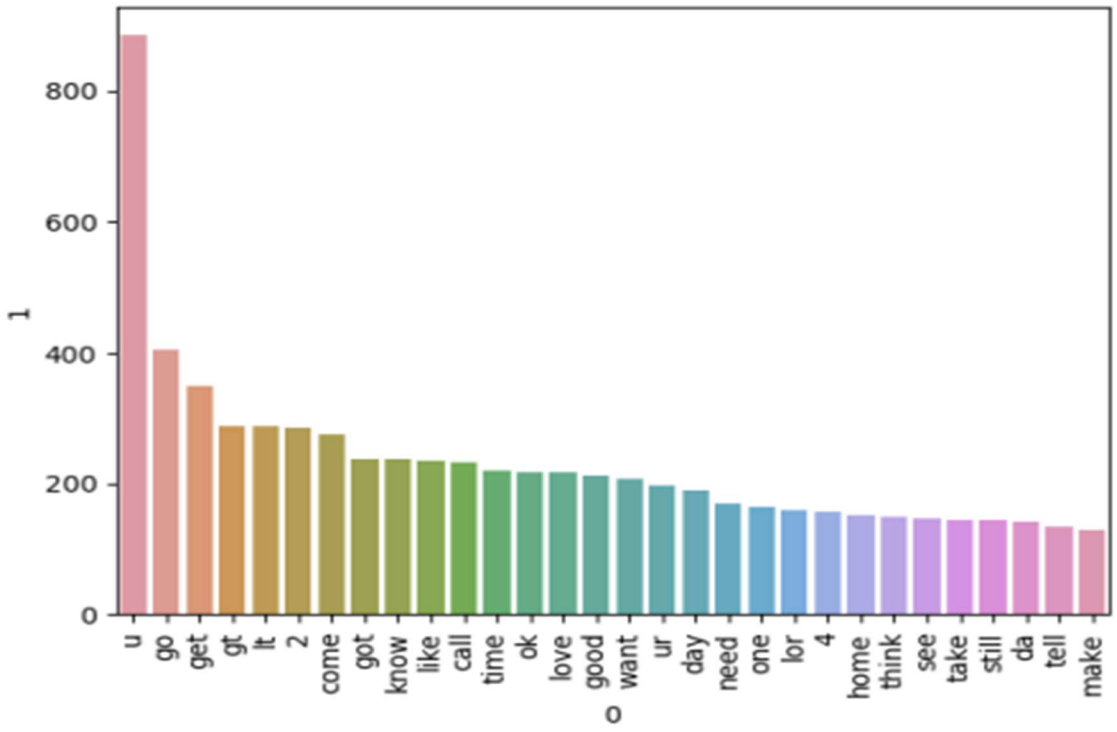


In [57]:
```
ham_corpus=[]
for msg in df[df['target']==0]['transformed_text'].tolist():
    for word in msg.split():
        ham_corpus.append(word)
```

In [58]:
```
len(ham_corpus)
```

Out[58]: 35404

In [59]:
```
from collections import Counter
sns.barplot(x=pd.DataFrame(Counter(ham_corpus).most_common(30))[0],y=pd.Dat
plt.xticks(rotation='vertical')
plt.show()
```



In [60]:
```
# text vectorization
# using  Bag of words
df.head()
```

Out[60]:

	target	text	num_characters	num_words	num_sentences	transformed_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail bugi n great world...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2	free entri 2 wkli comp win fa cup final tkt 21...
3	0	U dun say so early hor... U c already then say...	49	13	1	u dun say earli hor u c alreadi say
4	0	Nah I don't think he goes to usf, he lives aro...	61	15	1	nah think goe usf live around though

5.4 Model Buildings

In [61]:
```
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
cv=CountVectorizer()
tfidf=TfidfVectorizer(max_features=3000)
```

In [62]:
```
X = tfidf.fit_transform(df['transformed_text']).toarray()
```

In [63]:
```
X.shape
```

Out[63]:
```
(5169, 3000)
```

In [64]:
```
y = df['target'].values
```

In [65]:
```
y
```

Out[65]:
```
array([0, 0, 1, ..., 0, 0, 0])
```

In [66]:
```
from sklearn.model_selection import train_test_split
```

In [67]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
```

In [68]:
```
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
from sklearn.metrics import accuracy_score, confusion_matrix, precision_sco
```

In [69]:
```
gnb= GaussianNB()
mnb= MultinomialNB()
bnb=BernoulliNB()
```

In [70]:
```
gnb.fit(X_train, y_train)
y_pred1=gnb.predict(X_test)
print(accuracy_score(y_test,y_pred1))
print(confusion_matrix(y_test,y_pred1))
print(precision_score(y_test,y_pred1))
```

```
0.8694390715667312
[[788 108]
 [ 27 111]]
0.5068493150684932
```

In [71]:
```
mnb.fit(X_train, y_train)
y_pred2=mnb.predict(X_test)
print(accuracy_score(y_test,y_pred2))
print(confusion_matrix(y_test,y_pred2))
print(precision_score(y_test,y_pred2))
```

```
0.9709864603481625
[[896   0]
 [ 30 108]]
1.0
```

In [72]:
```
bnb.fit(X_train, y_train)
y_pred3=bnb.predict(X_test)
print(accuracy_score(y_test,y_pred3))
print(confusion_matrix(y_test,y_pred3))
print(precision_score(y_test,y_pred3))
```

```
0.9835589941972921
[[895   1]
 [ 16 122]]
0.991869918699187
```

In [73]:
```
# tfidf --> mnb
```

In [74]:
```
#pip install xgboost
```

In [75]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
```

In [76]:
```
svc = SVC (kernel='sigmoid', gamma=1.0)
knc =KNeighborsClassifier()
mnb =MultinomialNB()
dtc = DecisionTreeClassifier(max_depth=5)
lrc =LogisticRegression (solver='liblinear', penalty='l1')
rfc = RandomForestClassifier (n_estimators=50, random_state=2)
abc = AdaBoostClassifier(n_estimators=50, random_state=2)
bc = BaggingClassifier (n_estimators=50, random_state=2)
etc =ExtraTreesClassifier (n_estimators=50, random_state=2)
gbdt =GradientBoostingClassifier (n_estimators=50, random_state=2)
xgb =XGBClassifier (n_estimators=50, random_state=2)
```

In [77]:
```
clfs = {
    'SVC': svc,
    'KN' : knc,
    'NB': mnb,
    'DT': dtc,
    'LR': lrc,
    'RF': rfc,
    'AdaBoost': abc,
    'ETC': etc
}
```

In [78]:
```
def train_classifier(clf,X_train,y_train,X_test,y_test):
    clf.fit(X_train,y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test,y_pred)
    precision = precision_score(y_test,y_pred)
    return accuracy,precision
```

In [79]:
```
current_accuracy, current_precision=train_classifier(svc,X_train,y_train,X_
print(current_accuracy)
print(current_precision)
```

```
0.9758220502901354
0.9747899159663865
```

In [80]:
```
accuracy_scores = []
precision_scores = []

for name, clf in clfs.items():
    current_accuracy, current_precision= train_classifier(clf, X_train,y_tr
    print("For ",name)
    print("Accuracy - ",current_accuracy)
    print("Precision - ", current_precision)
    accuracy_scores.append(current_accuracy)
    precision_scores. append(current_precision)
```

```
For  SVC
Accuracy -  0.9758220502901354
Precision -  0.9747899159663865
For  KN
Accuracy -  0.9052224371373307
Precision -  1.0
For  NB
Accuracy -  0.9709864603481625
Precision -  1.0
For  DT
Accuracy -  0.9303675048355899
Precision -  0.8367346938775511
For  LR
Accuracy -  0.9584139264990329
Precision -  0.9702970297029703
For  RF
Accuracy -  0.9758220502901354
Precision -  0.9829059829059829
For  AdaBoost
Accuracy -  0.960348162475822
Precision -  0.9292035398230089
For  ETC
Accuracy -  0.9748549323017408
Precision -  0.9745762711864406
```

In [81]:
```
performance_df=pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy':accuracy_sc
```

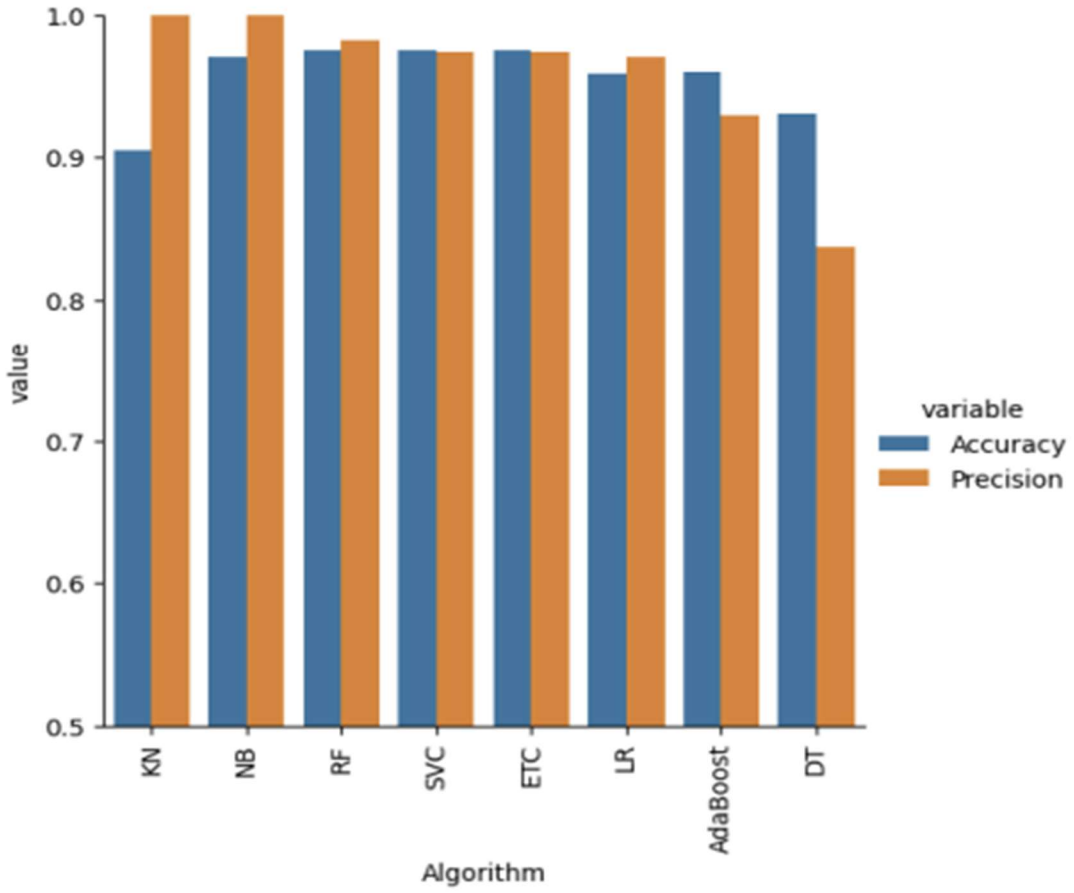
In [82]:
```
performance_df
```

Out[82]:

	Algorithm	Accuracy	Precision
1	KN	0.905222	1.000000
2	NB	0.970986	1.000000
5	RF	0.975822	0.982906
0	SVC	0.975822	0.974790
7	ETC	0.974855	0.974576
4	LR	0.958414	0.970297
6	AdaBoost	0.960348	0.929204
3	DT	0.930368	0.836735

In [83]:
```
performance_df1=pd.melt(performance_df,id_vars="Algorithm")
```

In [84]:
```
sns.catplot(x='Algorithm', y='value', hue='variable',data=performance_df1,k
plt.ylim(0.5,1.0)
plt.xticks(rotation='vertical')
plt.show()
```



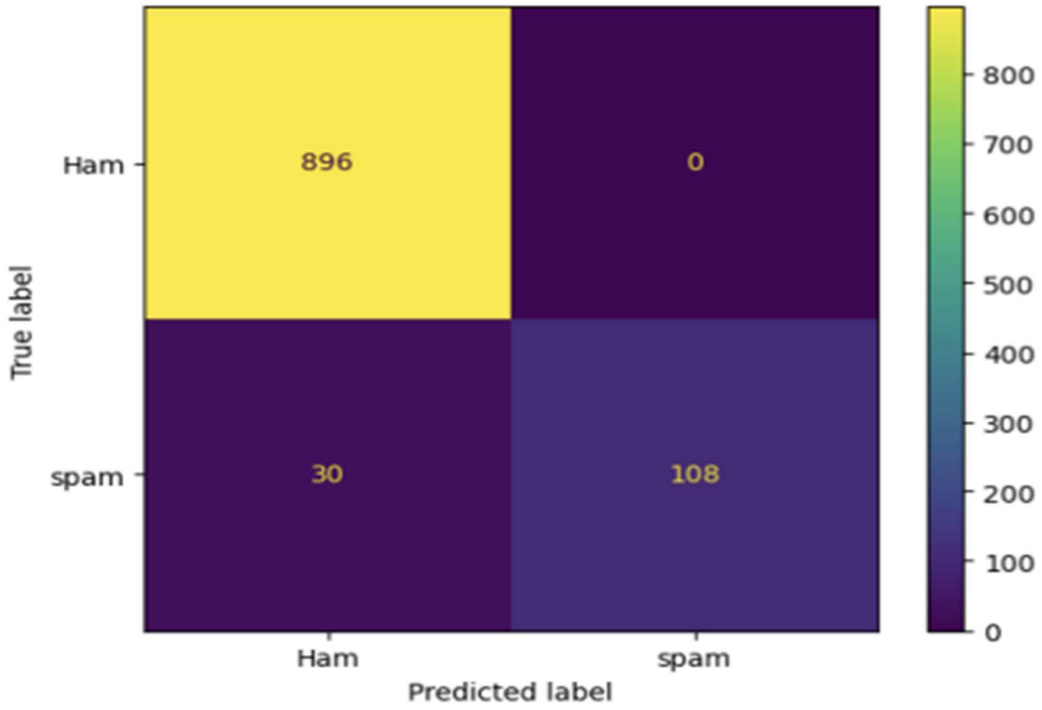
5.5 Naïve Bayes

In [88]:
```
import pickle
pickle.dump(tfidf,open('vectorizer.pkl','wb'))
pickle.dump(mnb,open('model.pkl','wb'))
```

In [86]:
```
from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
mnb =MultinomialNB()
mnb.fit(X_train, y_train)
y_pred2=mnb.predict(X_test)
score_nv=round(accuracy_score(y_test,y_pred2)*100,2)
print("The accuracty score achieved using Naive Bayes : "+str(score_nv))

print("The precision score Achieved using Naive Bayes :" +str(round(precisi
print("The confusion matrix is :")

cm=confusion_matrix(y_test,y_pred2)

cmd = ConfusionMatrixDisplay(cm, display_labels=['Ham','spam'])
cmd.plot()
```

```
The accuracty score achieved using Naive Bayes : 97.1
The precision score Achieved using Naive Bayes :100.0
The confusion matrix is :
```

Out[86]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x23b065
a5ed0>
```



5.6 Graphical User Interface:

app.py × ipykernel_launcher.py

```
import string
import streamlit as st
import pickle
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer

ps = PorterStemmer()


1 usage
def transform_text(text):
    text = text.lower()
    text = nltk.word_tokenize(text)

    y = []
    for i in text:
        if i.isalnum():
            y.append(i)

    text = y[:]
    y.clear()
    for i in text:
        if i not in stopwords.words('english') and i not in string.punctuation:
            y.append(i)
    text = y[:]
    y.clear()
    for i in text:
        y.append(ps.stem(i))
    return " ".join(y)
```

```
tfidf = pickle.load(open('vectorizer.pkl', 'rb'))
model = pickle.load(open('model.pkl', 'rb'))
st.title("Email/SMS Spam Classifier")

input_sms = st.text_area("Enter the message")

if st.button('Predict'):


    # 1. preprocess
    transformed_sms = transform_text(input_sms)
    # 2. vectorize
    vector_input = tfidf.transform([transformed_sms])
    # 3. Predict
    result = model.predict(vector_input)[0]

    # 4. Display
    if result == 1:
        st.header("Spam")
    else:
        st.header("Not Spam")
```

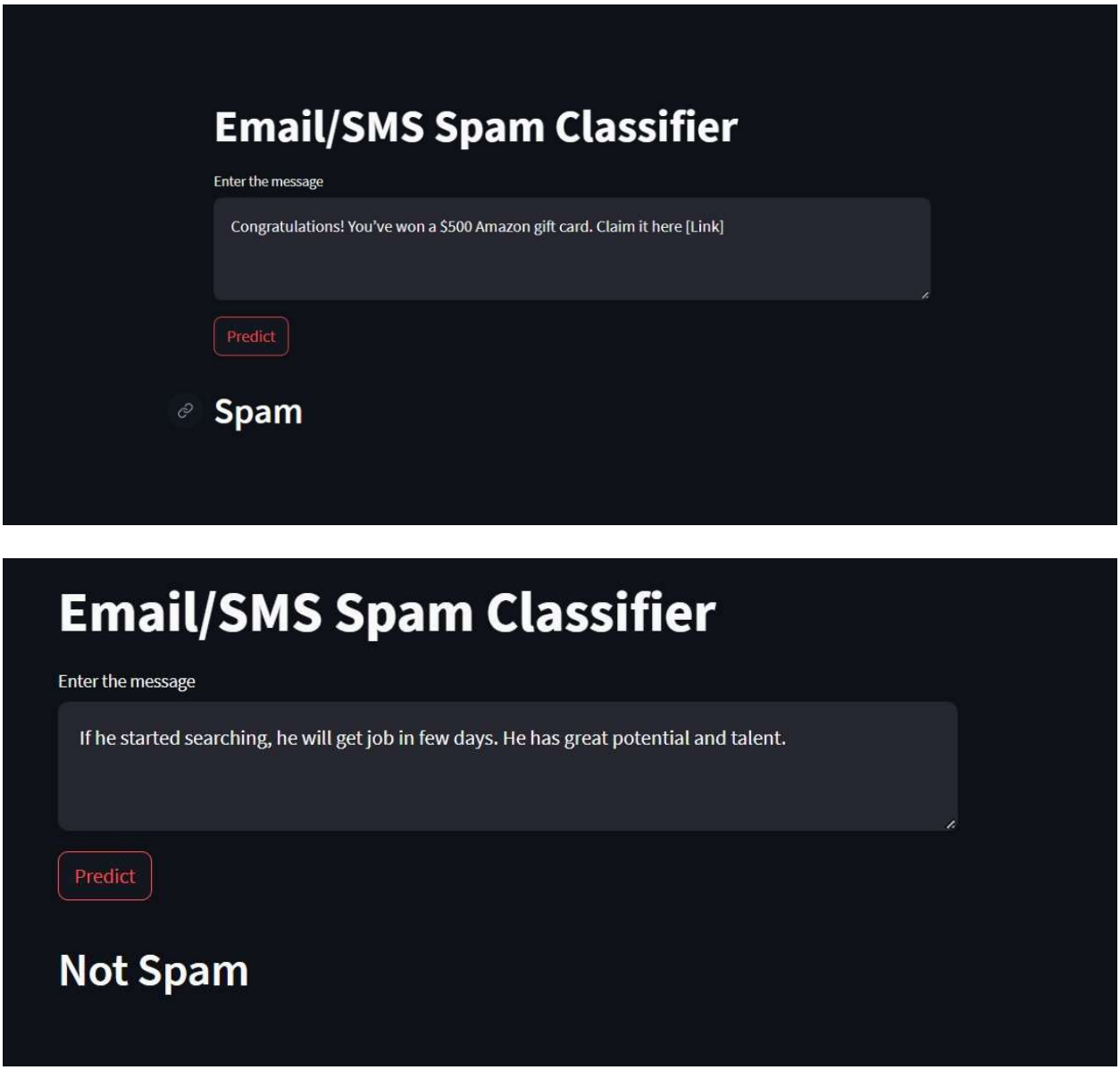


Figure 8: Graphical User Interface

CHAPTER-4

RESULT AND DISCUSSION

This project involved the application of various machine learning algorithms to determine whether an SMS is spam or not. Logistic Regression, Naive Bayes, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Decision Tree, and Random Forest were assessed for their accuracy and precision scores. These accuracy percentages and precision score are a measure of how well each algorithm performed in correctly classifying whether a SMS is spam or not. Naive Bayes demonstrated effectiveness in classifying SMS messages as either "spam" or "ham" (non-spam) with notable success in this study. The Naive Bayes model achieved an accuracy rate of around 97.1%, underscoring its capability to accurately identify a substantial portion of SMS messages. Notably, it exhibited a robust precision score of around 100%, denoting that the majority of messages classified as "spam" were indeed spam, and a recall score of approximately 97%, emphasizing its ability to identify actual spam messages among all true spam messages. The balanced F1-score, hovering around 94%, demonstrated the model's ability to strike a favorable balance by minimizing both false positives and false negatives. Nevertheless, the project identified instances of false positives and false negatives, highlighting the need for continuous refinement to enhance the overall user experience. Our SMS spam prediction project, driven by the Naive Bayes algorithm, has delivered an efficient and reliable solution for identifying unwanted SMS messages. Regular user engagement, model updates, and fairness considerations will be instrumental in its continued success in combatting SMS spam.

```
In [90]: clfs = {
             'SVC': svc,
             'KN' : knc,
             'NB': mnb,
             'DT': dtc,
             'LR': lrc,
             'RF': rfc,
             'AdaBoost': abc,
             'ETC': etc
         }
```

```
In [91]: def train_classifier(clf,X_train,y_train,X_test,y_test):
             clf.fit(X_train,y_train)
             y_pred = clf.predict(X_test)
             accuracy = accuracy_score(y_test,y_pred)
             precision = precision_score(y_test,y_pred)
             return accuracy,precision
```

```
In [98]: accuracy_scores = []
         precision_scores = []

         for name, clf in clfs.items():
             current_accuracy, current_precision= train_classifier(clf, X_train,y_tr
             print("For ",name)
             print("Accuracy - ",current_accuracy)
             print("Precision - ", current_precision)
             accuracy_scores.append(current_accuracy)
             precision_scores. append(current_precision)
```

```
In [102]: sns.catplot(x='Algorithm', y='value', hue='variable',data=performance_df1,k
          plt.ylim(0.5,1.0)
          plt.xticks(rotation='vertical')
          plt.show()
```

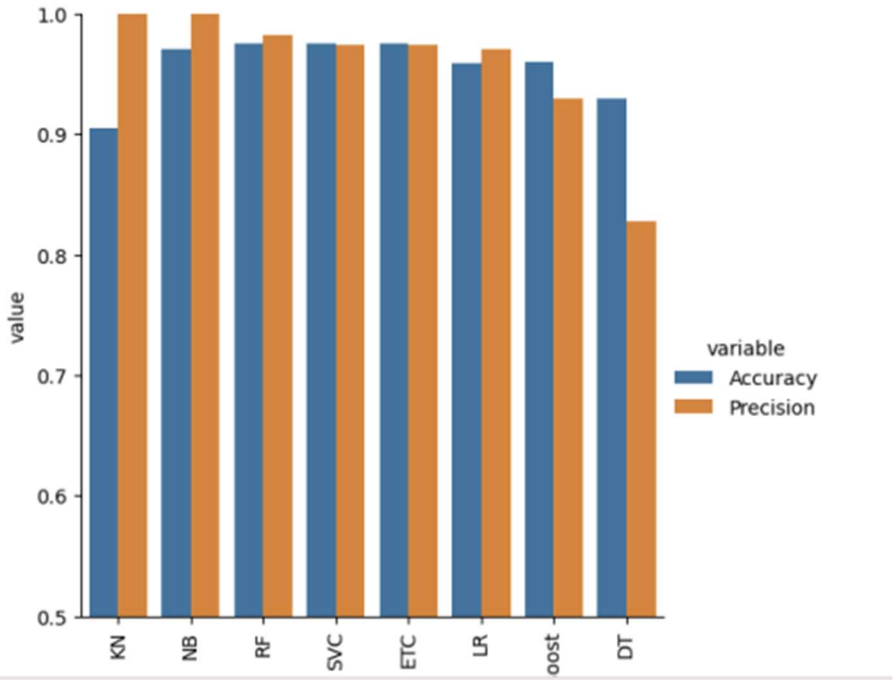


Figure 9: Comparison With Different Algorithm

CHAPTER-5

CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

In conclusion, the application of the Naïve Bayes algorithm for SMS spam detection has proven to be a robust and efficient approach. Naïve Bayes, known for its simplicity and effectiveness in text classification tasks, demonstrates noteworthy performance in distinguishing between spam and legitimate messages. Through the utilization of probabilistic calculations based on Bayes' theorem, the algorithm assigns probabilities to different classes, making it well-suited for the inherently probabilistic nature of language.

To address this classification problem, we employed the Naive Bayes Algorithm, specifically opting for the Multinomial Naive Bayes algorithm due to its superior precision score, signifying minimal occurrences of False Positives. For the vectorization technique, we utilized TFIDF.

"TF-IDF is an information retrieval technique that weighs a term's frequency (TF) and its inverse document frequency (IDF). Each word or term that occurs in the text has its respective TF and IDF score."

The key advantages of the Naïve Bayes algorithm in SMS spam detection include its quick training time, low computational complexity, and effectiveness even with relatively small datasets. These factors make it particularly suitable for real-time processing of incoming SMS messages, providing users with prompt and accurate spam classification.

Although Naïve Bayes performs well in numerous scenarios, it operates under the assumption of independence between features, a condition that may not always align with the interdependence present in real-world text data. Despite this simplifying assumption, Naïve Bayes consistently demonstrates competitive performance and is often a preferred choice in scenarios where interpretability and computational efficiency are crucial.

As we navigate the evolving landscape of SMS communication and spam tactics, the Naïve Bayes algorithm serves as a reliable foundation. However, to stay ahead of emerging challenges and enhance accuracy, ongoing research and exploration of advanced machine learning models and techniques, including deep learning architectures and behavioral analysis, present exciting avenues for future development in SMS spam detection. The adaptability and effectiveness of Naïve Bayes,

combined with the continuous integration of cutting-edge methodologies, contribute to the overall success of SMS spam detection systems in ensuring a secure and seamless messaging experience for users.

5.2 Future Scope:

For future development, additional machine learning models can be explored and incorporated into the hybrid system. Moreover, enhancing results could involve introducing additional preprocessing steps, such as assigning more weight to the "$" sign in spam classification. Lastly, there is potential for developing a real-time application to implement and evaluate the suggested hybrid model in a real-time performance scenario.

The future of SMS spam detection holds several promising developments and opportunities, driven by advancements in technology, artificial intelligence, and changing patterns in spam tactics. Here are some future scopes and potential areas of growth for SMS spam detection:

1. **Advanced Machine Learning Models:**
    - Continued exploration and deployment of advanced machine learning models, including the utilization of deep learning architectures like recurrent neural networks (RNNs) and transformer-based models, aim to improve the accuracy and efficiency of spam detection.
2. **Hybrid Models:**
    - Integration of multiple models or a combination of rule-based and machine learning approaches to create hybrid systems that leverage the strengths of different techniques, providing more robust spam detection.
3. **Explainable AI:**
    - Implementation of explainable AI (XAI) techniques to make the decision-making process of spam detection models more transparent and understandable. This is especially important for regulatory compliance and user trust.
4. **Behavioral Analysis:**
    - Adoption of behavioral analysis to detect spam based on user behavior patterns, considering factors such as the frequency and timing of messages, user interactions, and feedback.

REFERENCES

[1] https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset

[2] https://iopscience.iop.org/article/10.1088/1742-6596/1797/1/012017/pdf

[3] Gupta, Suparna Das, Soumyabrata Saha, and Suman Kumar Das. "SMS Spam Detection Using Machine Learning." Journal of Physics: Conference Series. Vol. 1797. No. 1. IOP Publishing, 2021

[4] N Krishnamoorthy, Suresh Muthusamy, Seyedali Mirjalili, M Vignesh, R Vishnuhari, and SK Sanjeev Raja. Comparative study on sms spam message detection with different machine learning methods for safety communication. In Artificial Intelligence for Internet of Things, pages 65–73. CRC Press, 2022.

[5] Sadoga Abdallah Yousef Mohamedali. Spam Detection SMS Using Machine Learning. Ph.D. thesis, Al-Neelain University, 2022.

[6] https://www.researchgate.net/publication/347260266_Hybrid_SMS_Spam_Filtering_System Using_Machine_Learning_Techniques

[7] SMS Spam Detection using Machine Learning Approach using Support vector machine (SVM) BY Housemand Shirani-Mehr IN 2013.

[8] Spam Detection In Sms Using Machine Learning Through Text Mining using Support vector machine (SVM) BY M. Rubin Julis, S. Alagesan IN 2020.

[9] SMS SPAM DETECTION USING MACHINE LEARNING using and Naive Bayes Classifier BY Mr. E.Sankar, Y Y S Shekhar Babu, M.Tridev, B.E, IN 2023.

[10] Enhancing Spam Detection on Mobile Phone Short Message Service (SMS) Performance using FP-Growth and Naive Bayes Classifier BY Dea Delvia, Arifin, Shaufiah Moch. Arif Bijaksana IN 2016.

[11] SMS Spam Detection Using Machine Learning BY Suparna Das gupta , Soumyabrata Saha, Suman Kumar Das IN 2020.

[12] Spam Detection In Sms Using Machine Learning Through Text Mining BY M.Rubin Julis, S.Alagesan in 2020

[13] MACHINE LEARNING BASED SPAM DETECTION SYSTEM BY Saurabh Masurkar, Arjunsingh Rajput, Anish Angane, Simran Madaan, Shaveta Malik in 2020

FP

ORIGINALITY REPORT

16%	12%	8%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Fred Nwanganga, Mike Chapple. "Practical Machine Learning in R", Wiley, 2020 Publication	2%
2	www.ijraset.com Internet Source	2%
3	www.coursehero.com Internet Source	1%
4	www.researchgate.net Internet Source	1%
5	Nahid Ebrahimi Majd, Mandar Shivaji Hanchate. "Spam SMS Classification Using Machine Learning", 2023 32nd International Conference on Computer Communications and Networks (ICCCN), 2023 Publication	1%
6	www.analyticsvidhya.com Internet Source	1%
7	www.mdpi.com Internet Source	1%

8	www.slideshare.net Internet Source	1 %
9	Edward Wijaya, Gracella Noveliora, Kharisma Dwi Utami, Rojali, Ghinaa Zain Nabiilah. "Spam Detection in Short Message Service (SMS) Using Naïve Bayes, SVM, LSTM, and CNN", 2023 10th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), 2023 Publication	<1 %
10	Linesh Patil, Jainish Sakhidas, Divya Jain, Sahil Darji, Karuna Borhade. "Chapter 16 A Comparative Study of Spam SMS Detection Techniques for English Content Using Supervised Machine Learning Algorithms", Springer Science and Business Media LLC, 2023 Publication	<1 %
11	www.grin.com Internet Source	<1 %
12	sist.sathyabama.ac.in Internet Source	<1 %
13	www.ijstr.org Internet Source	<1 %
14	nanopdf.com Internet Source	<1 %

15	downloads.hindawi.com Internet Source	<1 %
16	journalofbigdata.springeropen.com Internet Source	<1 %
17	thesai.org Internet Source	<1 %
18	bspace.buid.ac.ae Internet Source	<1 %
19	www.research.manchester.ac.uk Internet Source	<1 %
20	www.researchsquare.com Internet Source	<1 %
21	Akbari, Fatemeh, and Hedieh Sajedi. "SMS spam detection using selected text features and Boosting Classifiers", 2015 7th Conference on Information and Knowledge Technology (IKT), 2015. Publication	<1 %
22	Neha Kandula, Ram Kumar. "A Deep Dive into Academic Excellence: Using Deep Learning to Evaluate and Improve Engineering Students' Performance", Research Square Platform LLC, 2023 Publication	<1 %
23	etheses.dur.ac.uk Internet Source	<1 %