JSS MAHAVIDYAPEETHA

JSS SCIENCE AND TECHNOLOGY UNIVERSITY
JSS TECHNICAL INSTITUTIONS CAMPUS
MYSURU - 570 006

SRI JAYACHAMARAJENDRA COLLEGE OF ENGINEERING

"SPAM MESSAGE DETECTION"

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING

BY

Adithya Deepthi Kumar	01JST21CS005
Kushal Nunna	01JST21CS063
Ullekh Puttaswamy	01JCE21CS118
Surendra A	01JST21CS153

Under the guidance of

Dr.R GURU Ph.D
Associate Professor,
Dept of Computer Science & Engineering,
SJCE,JSS STU,Mysuru-06

2024-2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the work entitled “Spam Message Detection” is a bonafide work carried out by **Adithya Deepthi Kumar,Kushal Nunna,Ullekh Puttaswamy and Surendra A** in partial fulfilment of the award of degree of Bachelor of Engineering in Computer Science and Engineering of JSS Science and Technology University, Mysore during the year 2024. It is certified that all corrections / suggestions indicated during the CIE have been incorporated in the report. The mini project has been approved as it satisfies the academic requirements in respect of mini project work prescribed for the Mini Project (20CS69P) course.

Under the Guidance of	**Head of the Department**
Dr.R GURU Ph.D	**Dr.SRINATH S** Ph.D
Associate Professor,	Associate Professor & HOD,
Dept of Computer Science & Engineering,	Dept of Computer Science & Engineering,
SJCE,JSS STU,Mysuru-06	SJCE,JSS STU,Mysuru-06

Name of Examiner	Signature with Date
1. …..…………………	…………………………..
2. …………………….	…………………………..
3. …..…………………	…………………………..

Place: Mysore

Date: 11/6/2024

TABLE OF CONTENTS

ABSTRACT

The Multi-Platform Spam Detection and Filtering System is a comprehensive solution leveraging Python, NLP, Naive Bayes, TF-IDF Vectorizer, Flask, and Flutter. It seamlessly integrates into SMS applications to provide real-time spam detection via a web interface (Flask) and a mobile app (Flutter). The development cycle encompasses objective definition, SMS dataset collection, preprocessing, and machine learning algorithm training and fine-tuning. Evaluation metrics aid in selecting the most effective model for bolstering digital security. By employing NLP and machine learning techniques like Naive Bayes and TF-IDF Vectorizer, the system can discern between spam and legitimate messages with high accuracy. The use of Flask and Flutter ensures accessibility across various platforms, enhancing user convenience. Overall, this system represents a robust approach to combating spam, ensuring a safer and more secure digital communication environment.

1. INTRODUCTION

In today's digital age, the proliferation of spam messages presents a significant challenge, disrupting communication channels and posing risks to user privacy and security. As a response to this pressing issue, the project endeavors to develop a comprehensive solution aimed at detecting and mitigating spam messages.

The project begins with defining clear objectives, scope, and deliverables. Further ahead it proceeds to collection and preprocessing SMS datasets, ensuring the cleanliness and readiness of the data for subsequent analysis. Following data preprocessing, there is engagement in feature engineering, where relevant features are extracted and select informative ones to enhance the performance of our spam detection system.

Moving forward, into model development, selecting suitable machine learning algorithms and training multiple models to effectively identify spam messages becomes the primary task. After assessing the performance of these models using appropriate metrics, the most robust and generalizable model is ultimately selected for deployment.

The deployment phase will involve integrating the selected model into a production environment, implementing monitoring mechanisms, and conducting user acceptance testing. Finally, the project concludes with comprehensive documentation and reporting, summary of findings, recommendations, and the overall project journey.

With a dedicated focus on enhancing user experience and security, the project represents a concerted effort to create a safer and more secure digital ecosystem for all users. The project presents a dedicated effort, striving towards combating spam messages and contributing to a more positive online experience for everyone.

1.1 PROBLEM STATEMENT

Spam message detection aims to automatically identify and classify incoming text messages as either spam or legitimate (ham).

1.2 OBJECTIVES

- The Objective is to develop a machine learning model that can accurately classify incoming SMS messages in real-time, distinguishing between spam and legitimate content with high precision and recall.
- The model should be capable of generalizing well to new, unseen messages and robustly handle variations in message content, language, and style.
- Additionally, we aim to implement seamless integration into existing communication platforms by the means of a mobile app with ethical considerations to ensure minimal false positives and negatives fostering user trust and privacy.

The specific objective of the Project are as follows:

1. **Data Preparation**: Implement a robust data cleaning and preprocessing pipeline to ensure the SMS dataset is optimized for analysis, free from redundancies, and suitably formatted for model ingestion.

2. **Feature Engineering**: Enhance the classifier's performance by developing advanced text-based features using NLP techniques such as tokenization, stemming, and removal of stopwords and special characters.

3. **Model Development and Vectorization**: Build and train a Naive Bayes classifier to accurately identify spam and ham messages. Experiment with various text vectorization methods like TF-IDF to optimize classification accuracy.

4. **Model Optimization**: Refine the model using key metrics like accuracy and precision, and enhance model performance through hyperparameter tuning and testing different algorithms.

5. **User Interface Development**: Design a simple, intuitive web application for real-time SMS classification, focusing on ease of use and accessibility for end-users.

6. **System Deployment and Maintenance**: Deploy the classifier in a production environment with support for ongoing maintenance and updates to adapt to new spam trends and techniques.

1.3 EXISTING SYSTEMS

1. **Truecaller**:

Truecaller is a widely used application that identifies and blocks spam SMS and calls. It uses a large database of known spammers and allows users to report new ones.

2. **Hiya**:

Hiya provides spam protection for calls and SMS. It has a robust database of spam numbers and utilizes machine learning to identify and block new spam messages.

3. **SMS Shield**:

SMS Shield is an iOS app that uses machine learning to filter out spam messages. It categorizes SMS as spam or non-spam based on content analysis.

1.4 UNIQUENESS OF OUR PROJECT

- **Advanced Text Analysis**: Focused on text preprocessing and content analysis, the model can detect nuanced and new spam patterns more effectively.

- **Customization and Adaptability**: Being custom-built, it can be adapted and improved continuously based on specific user feedback and new data, offering more flexibility than commercial applications.

- **Detailed Evaluation and Improvement Process**: Structured approach to evaluation and continuous improvement ensures that the model remains up-to-date and effective against new spam tactics.

- **Enhanced Security**: Data is not shared and remains on users' devices, ensuring greater privacy and security compared to applications that may share data with third-party servers.

2. LITERATURE REVIEW

Title: Email Spam Detection Using Machine Learning Algorithms

Authors: Nikhil Kumar, Sanket Sonowal, Nishant

Institution : Delhi Technological University, New Delhi

Year:2020

Focus :

This paper focuses on evaluating machine learning methods for email spam detection. It assesses popular algorithms like Naïve Bayes and support vector machines, analyzing their performance metrics and adaptability.

Machine Learning Algorithms for Email Spam Detection:

Explores various machine learning algorithms such as Naïve Bayes, support vector machine-nearest neighbor, random forest, bagging, boosting, and neural networks.

Related Work on Machine Learning Methods:

Explores studies utilizing techniques such as the KNN algorithm, Naïve Bayes, and Reverse DBSCAN algorithm for email spam detection.

Methodology:

- Outlines steps involved in implementing the model, including data preprocessing, feature transformation, hyperparameter tuning, and training using different classifiers.
- Presents a flowchart depicting the sequential process of data insertion, encoding, training, testing, and comparison of classifiers.

Results:

Provides a comparison table and graph showcasing the performance of various classifiers such as Support Vector Classifier, K-Nearest Neighbor, Naïve Bayes, Decision Tree, Random Forest, AdaBoost Classifier, and Bagging Classifier.

Conclusion:

- Discusses limitations and potential improvements of the model, emphasizing the need for filtering spam based on trusted domain names.
- Considers the possibility of using the model for categorizing emails and distinguishing between spam and non-spam emails.

Title : Email based Spam Detection

Authors :Thashina Sultana, KA Sapnaz, Fathima Sana, Jamedar Najath

Institution : Yenepoya Institute of Technology Moodbidri, India

Year : 2020

Focus:

The paper focuses on the problem of spam emails and the impact they have on email users and organizations.It emphasizes the need for effective spam detection techniques to protect users from unsolicited and potentially harmful emails.

Contributions:

- The paper presents a proposed system for spam detection using machine learning techniques, specifically Naïve Bayes Classifier.
- It discusses the use of various algorithms and tools such as string matching algorithms, Bayesian Classifiers, and machine learning to detect spam emails.

Methods:

- Naïve Bayes Classifier and Bayes' theorem for detecting spam messages using a dataset from Kaggle and preprocessing the input messages using tokenization and stemming.
- The proposed system uses Natural Language Tool Kit (NLTK), Matplotlib, Word Cloud, pandas, and NumPy for text processing, data manipulation, and analysis.

Future Directions:

- The paper suggests that the proposed system can be implemented using different algorithms and additional features to enhance spam detection.
- It highlights the potential for further research in the area of email classification and spam detection using machine learning methods based on bag of words technique.

Conclusion:

- The paper concludes by emphasizing the significance of email as a communication medium and the impact of spam emails on users and organizations.
- It highlights the importance of the proposed system in detecting and preventing spam messages, thereby reducing the potential financial and security risks associated with spam.

3. DESIGN AND IMPLEMENTATION

3.1 DESIGN PHASE:

The design phase explores various aspects, including **data acquisition**, **pre-processing**, **model selection**, **training**, and **evaluation**.

3.1.1 Data Acquisition and EDA :

- **Dataset Selection**: We will leverage a multifaceted data collection approach, utilizing both publicly available datasets like the **UCI Machine Learning repository's SMS Spam Collection** (https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection) and contributions from mobile users.
- **Data Analysis**: The Data Analysis phase in this SMS spam detection project involves cleaning and manipulating the raw SMS data from various sources. This prepares the data for modeling by extracting key features that differentiate spam from legitimate messages.
- **Data Exploration**: During data analysis, visualization techniques can help you explore the characteristics of your SMS data. We can create histograms to see the distribution of message lengths across spam and non-spam messages. Word clouds can reveal frequently used words in each category, potentially highlighting spam indicators like unusual symbols or promotional keywords.

3.1.2 Data Preprocessing :

- **Cleaning**: Remove irrelevant characters, punctuation, extra spaces, and special symbols.
- **Normalization**: Convert text to lowercase for consistency.
- **Tokenization**: Split the message into individual words or tokens.
- **Stop Word Removal**: Eliminate common words ("the," "a," "an") that don't contribute to spam identification.
- **Stemming/Lemmatization**: Reduce words to their root form (e.g., "running" -> "run") for better feature extraction.

3.1.3 Feature Engineering:

- **Bag-of-Words (BoW)**: Represent messages as a frequency vector where each element indicates the count of a specific word in the message.
- **TF-IDF**: Enhance BoW by considering word importance based on its frequency within the message and its rarity across all messages.

3.1.4 Model Selection:

Naive Bayes: Simple and efficient for text classification, assuming message features are independent.

3.1.5 Model Training and Evaluation:

- **Split Data**: Divide the preprocessed data into training (70-80%) and testing (20-30%) sets.
- **Train Model**: Train the chosen model using the training set. Hyperparameter tuning can be performed to optimize model performance.
- **Evaluation**: Evaluate the model's performance on the testing set using metrics like accuracy and precision.

3.1.6 Deployment Design (Website & App):

- **Website/App Interface:** Design a user-friendly interface for both website and app. Users can input SMS messages for spam classification.
- **API Integration:** Integrate the trained model as a web service API. The website/app will send SMS text for analysis through the API.
- **Spam/Ham Classification:** Upon receiving SMS text, the API performs TF-IDF conversion and feeds data to the trained model for spam/ham classification.
- **Result Display:** The website/app displays the classification result (spam/ham) to the user.

3.2 IMPLEMENTATION PHASE:

3.2.1 Tech Stack

Programming Language:

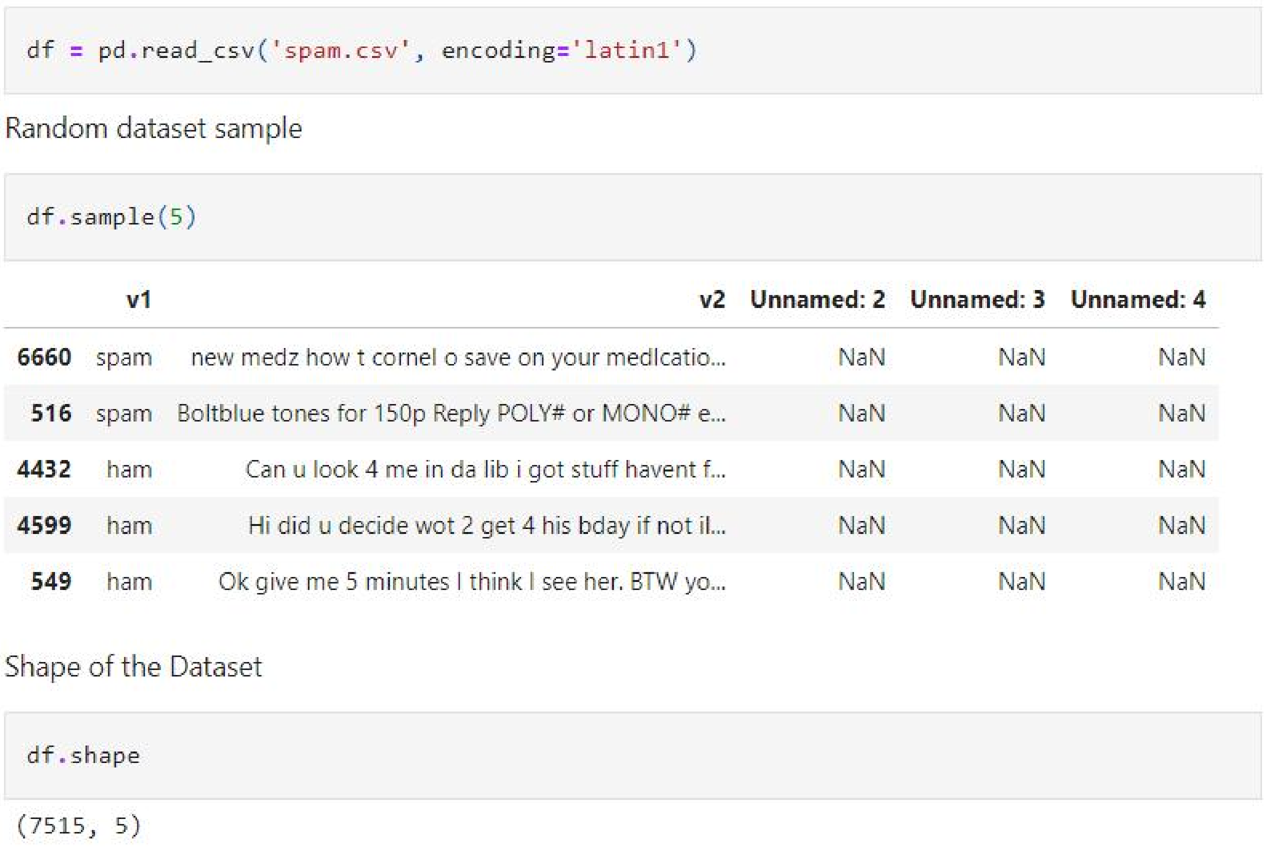
- **Python**: The primary programming language used for data manipulation, analysis, and machine learning model development.
- **Dart**: It is an open-source, scalable programming language, with robust libraries and runtimes, for building web, server, and mobile apps

3.2.2 Libraries and Modules

1. **Pandas (`pandas`)**: Used for data manipulation and analysis, particularly useful for handling structured data like CSV files.
2. **NumPy (`numpy`)**: Essential for numerical operations on arrays and matrices, used in the project to handle data structures for machine learning algorithms.
3. **Matplotlib (`matplotlib.pyplot`)**: A plotting library used for generating visualizations like pie charts and histograms to analyze the data distribution.
4. **Seaborn (`seaborn`)**: Based on matplotlib, it provides a high-level interface for drawing attractive and informative statistical graphics.
5. **NLTK (`nltk`)**: Natural Language Toolkit, used for text preprocessing tasks such as tokenization and stemming, and for downloading stop words.
6. **Scikit-learn (`sklearn`)**:
   - **`LabelEncoder`**: For encoding text labels to numerical form.
   - **`train_test_split`**: To split the dataset into training and testing sets.
   - **`CountVectorizer`, `TfidfVectorizer`**: For converting text data into feature vectors.
   - **`GaussianNB`, `MultinomialNB`, `BernoulliNB`**: Naive Bayes algorithms used for the classification tasks.
   - **`accuracy_score`, `confusion_matrix`, `precision_score`**: Metrics to evaluate the performance of the machine learning models.
7. **WordCloud (`wordcloud.WordCloud`)**: Used to visualize the most frequent words in spam and ham messages.
8. **Pickle (`pickle`)**: For saving the machine learning models and vectorizer to disk, which allows them to be re-used without having to re-train.

Loading the Dataset:

- CSV Files: Data is read from and written to CSV files, allowing for easy storage and retrieval of structured data.



```
df = pd.read_csv('spam.csv', encoding='latin1')
```

Random dataset sample

```
df.sample(5)
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
6660	spam	new medz how t cornel o save on your medlcatio...	NaN	NaN	NaN
516	spam	Boltblue tones for 150p Reply POLY# or MONO# e...	NaN	NaN	NaN
4432	ham	Can u look 4 me in da lib i got stuff havent f...	NaN	NaN	NaN
4599	ham	Hi did u decide wot 2 get 4 his bday if not il...	NaN	NaN	NaN
549	ham	Ok give me 5 minutes I think I see her. BTW yo...	NaN	NaN	NaN

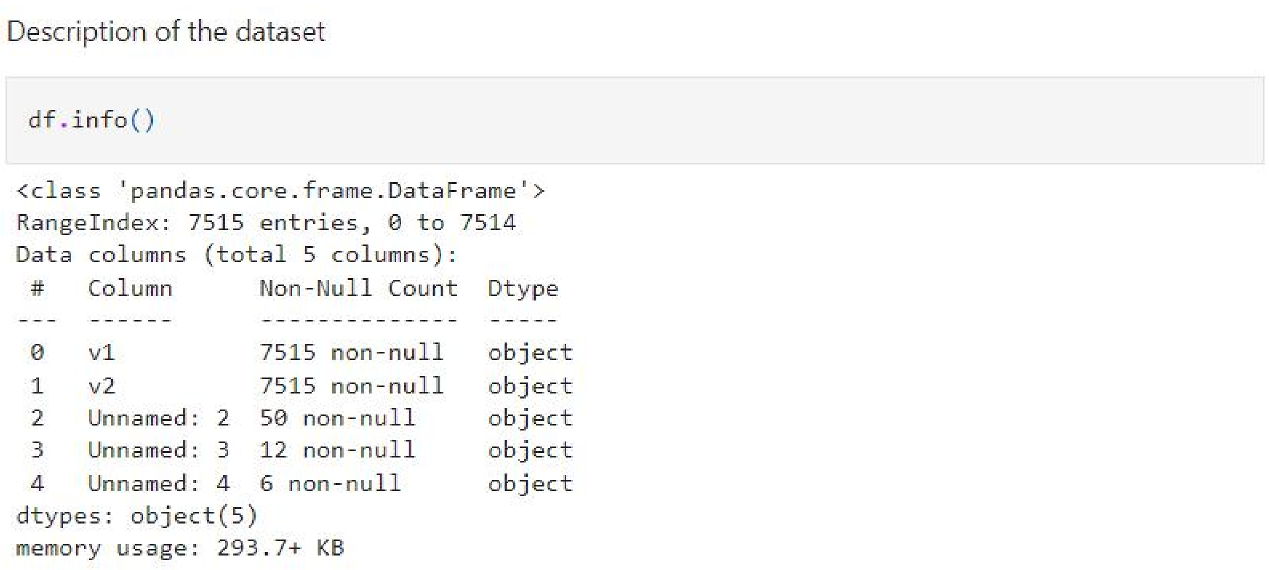
Shape of the Dataset

```
df.shape
```

```
(7515, 5)
```

Fig 3.1 Importing the data set

Data Cleaning

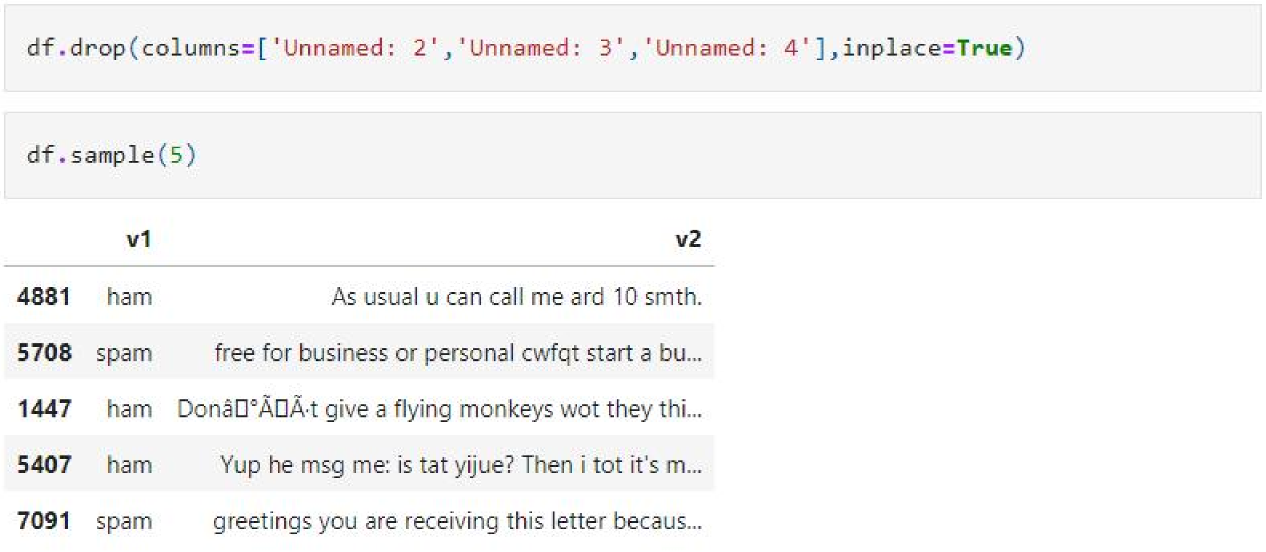


Description of the dataset

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7515 entries, 0 to 7514
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   v1          7515 non-null   object
 1   v2          7515 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: 293.7+ KB
```

Fig 3.2 Removing three columns

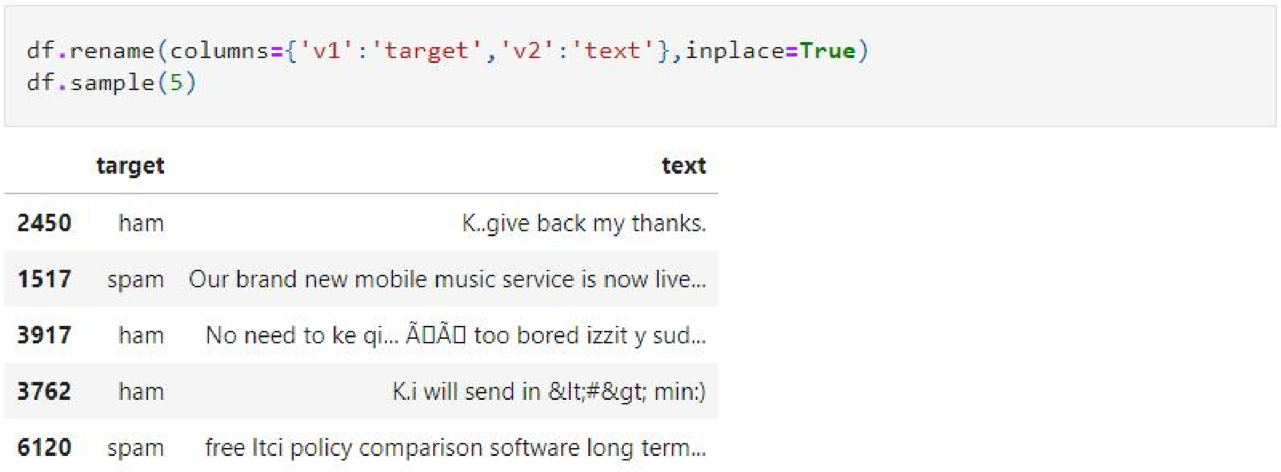


```
df.drop(columns=['Unnamed: 2','Unnamed: 3','Unnamed: 4'],inplace=True)
```

```
df.sample(5)
```

	v1	v2
4881	ham	As usual u can call me ard 10 smth.
5708	spam	free for business or personal cwfqt start a bu...
1447	ham	Donâ□°Ã□Ã·t give a flying monkeys wot they thi...
5407	ham	Yup he msg me: is tat yijue? Then i tot it's m...
7091	spam	greetings you are receiving this letter becaus...

Fig 3.3 Excluding last three columns

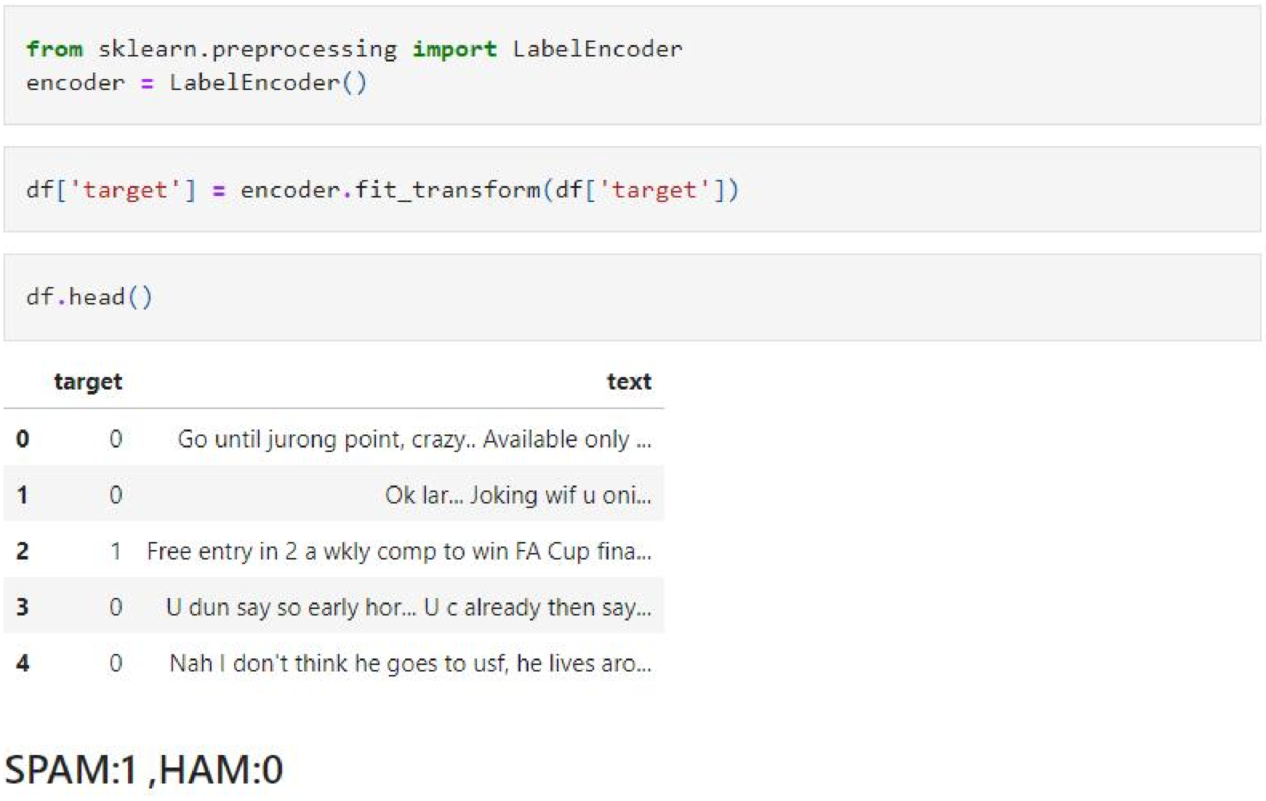


```
df.rename(columns={'v1':'target','v2':'text'},inplace=True)
df.sample(5)
```

	target	text
2450	ham	K..give back my thanks.
1517	spam	Our brand new mobile music service is now live...
3917	ham	No need to ke qi... Ã□Ã□ too bored izzit y sud...
3762	ham	K.i will send in &lt;#&gt; min:)
6120	spam	free ltci policy comparison software long term...

Fig 3.4 Renaming columns

We removed unnecessary columns ('Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4') using `drop`, then renamed columns ('v1' to 'target', 'v2' to 'text') for clarity. Categorical labels ('ham', 'spam') were encoded to numeric values ('0' for 'ham', '1' for 'spam') with `LabelEncoder` for machine learning readiness.



```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
```

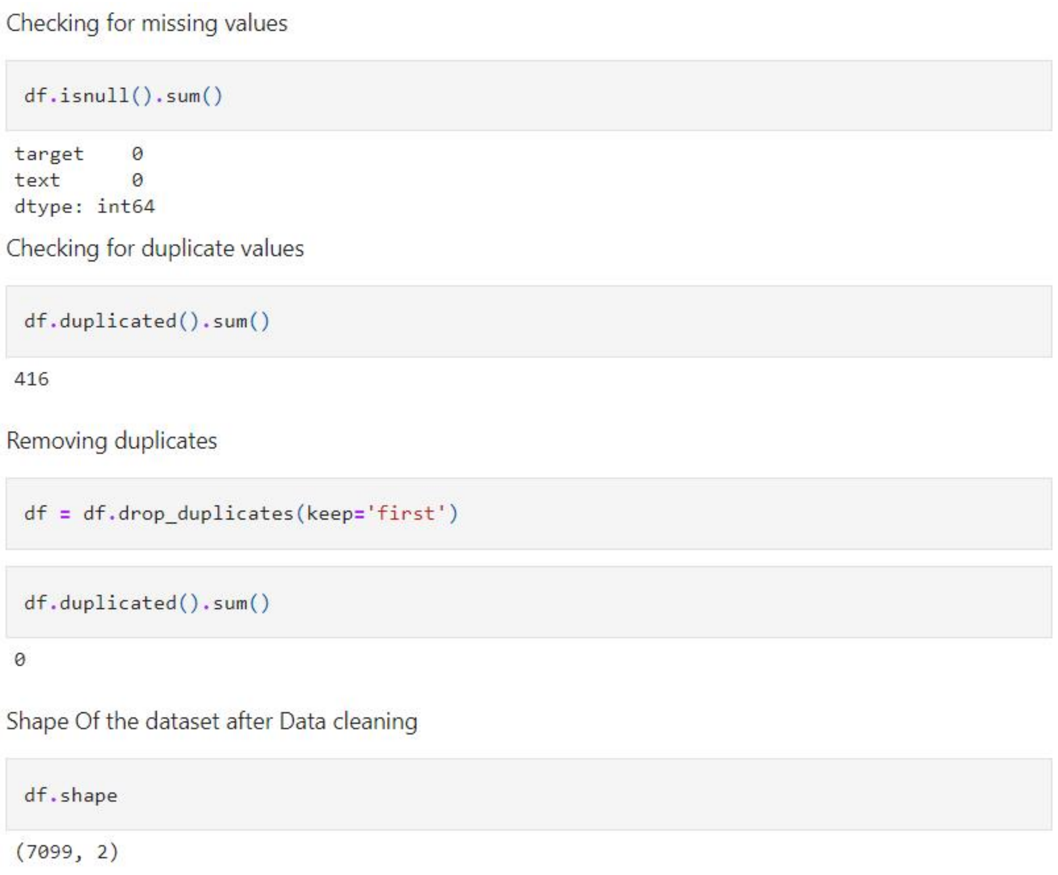
```
df['target'] = encoder.fit_transform(df['target'])
```

```
df.head()
```

	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...

SPAM:1 ,HAM:0

Fig 3.5 Assigning binary values to spam and ham messages



Checking for missing values

```
df.isnull().sum()
```

```
target    0
text      0
dtype: int64
```

Checking for duplicate values

```
df.duplicated().sum()
```

```
416
```

Removing duplicates

```
df = df.drop_duplicates(keep='first')
```

```
df.duplicated().sum()
```

```
0
```

Shape Of the dataset after Data cleaning

```
df.shape
```

```
(7099, 2)
```

Fig 3.6 Checking and removing duplicate and missing values

3.2.3 Exploratory Data Analysis

We utilize the NLTK library to generate three columns representing the number of characters, words, and sentences in the dataset. Subsequently, we analyze these columns separately for ham and spam messages.

- To explore the relationship between the number of characters in messages and their classification as spam or ham, we can create a box plot.

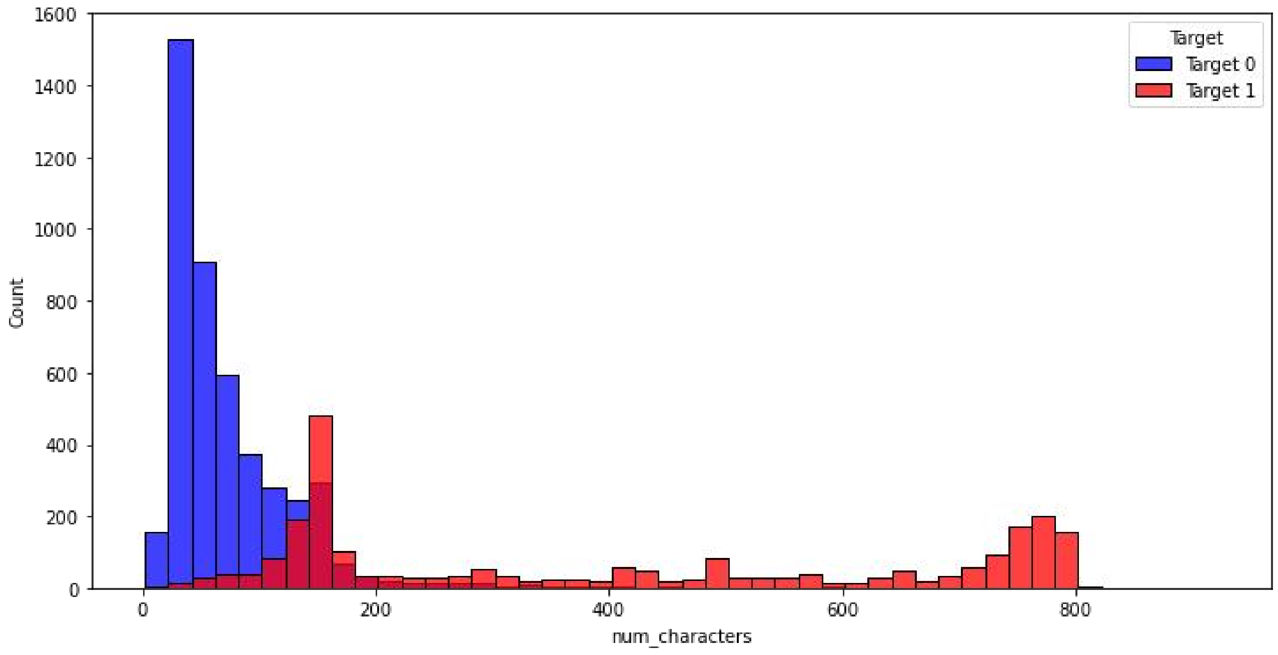


Fig 3.7 Box Plot (count vs characters)

- To explore the relationship between the number of words in messages and their classification as spam or ham, we can create a box plot.

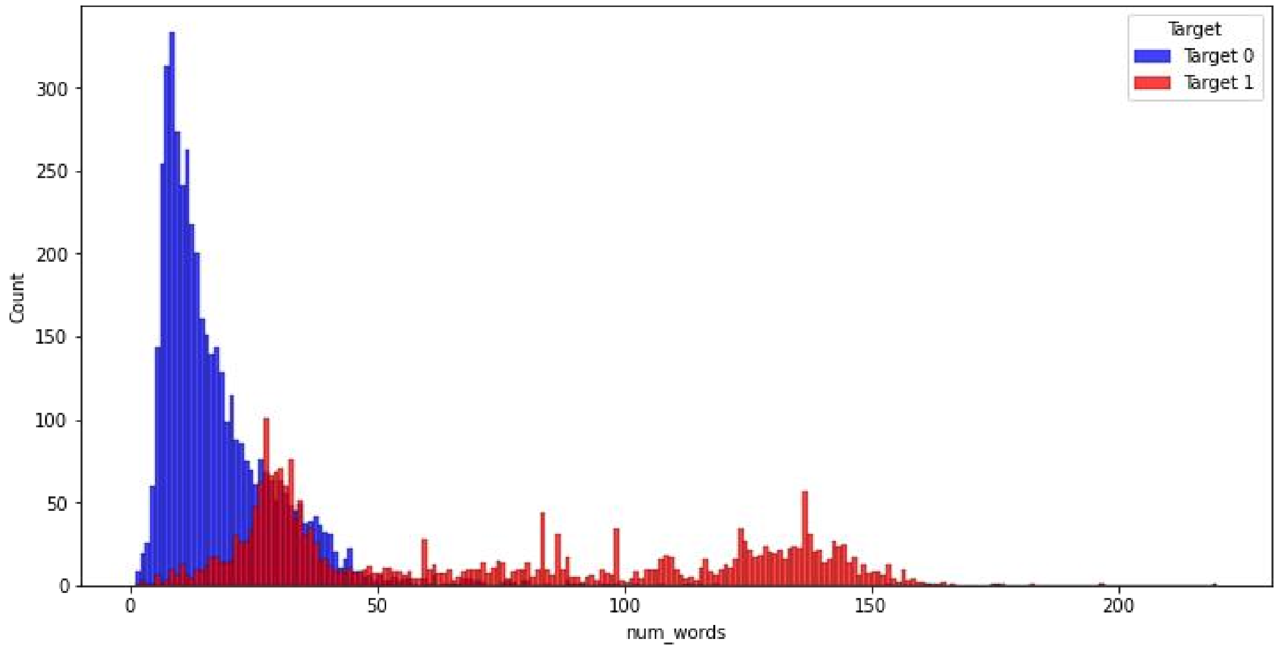


Fig 3.8 Box Plot (count vs words)

In the visual analysis provided by the box plots, where blue signifies ham messages and red indicates spam messages, it is evident that spam messages generally contain a higher number of characters and words compared to ham messages.

Creating a pair plot (or a scatterplot matrix) against all the parameters in your dataset is an effective way to visualize and analyze the relationships and distributions of multiple variables simultaneously.

Pair Plot

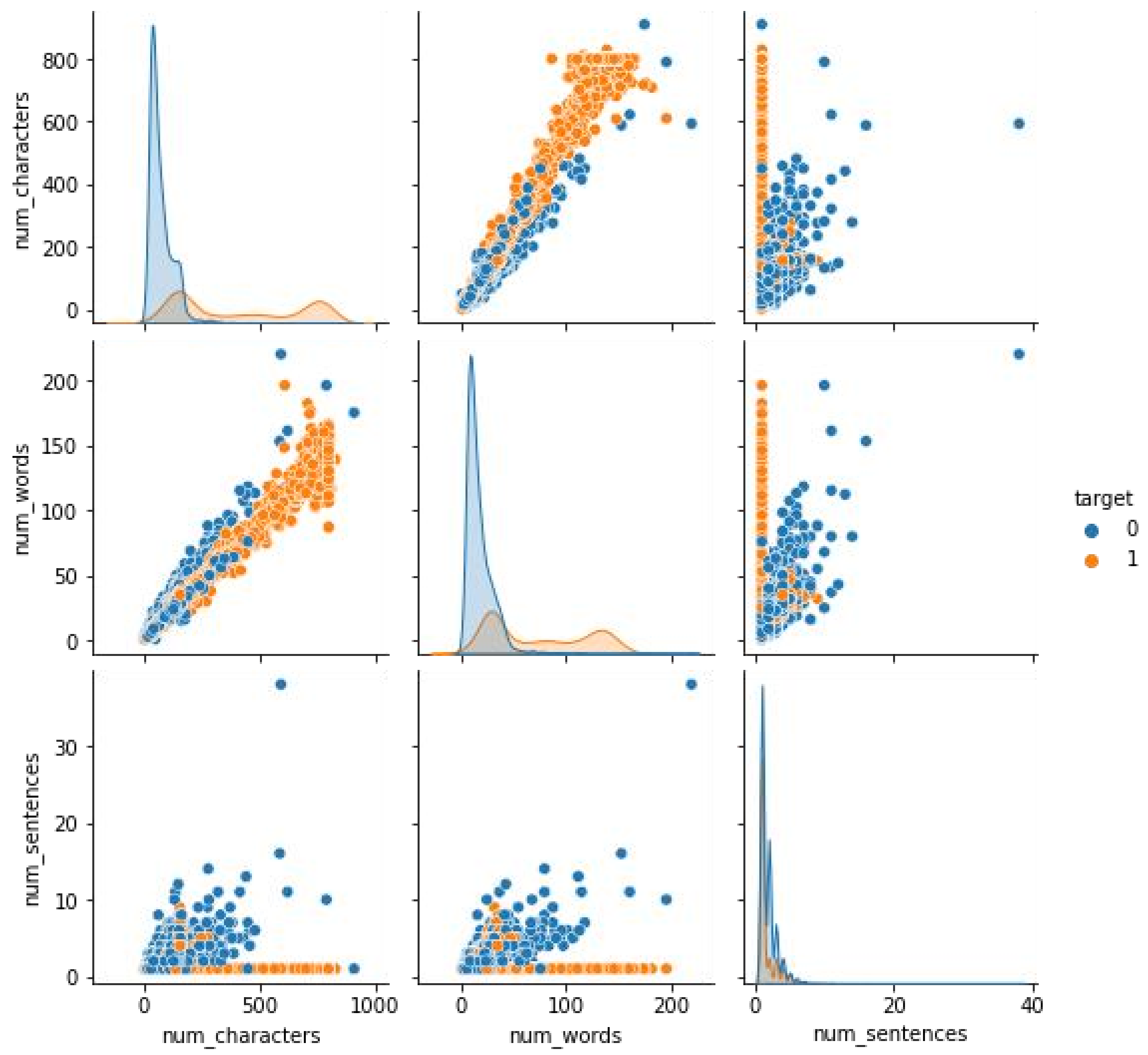


Fig 3.9 Pair plot for all parameters

Blue : Ham Messages

Orange : Spam Messages

Plotting the correlation between features and labels.This visual representation will help us understand if spam messages tend to have a different character count compared to ham messages, potentially providing insights into distinguishing features between the two categories.

Correlation between features and labels

```
df.corr()
```

	target	num_characters	num_words	num_sentences
target	1.000000	0.696495	0.666512	-0.085617
num_characters	0.696495	1.000000	0.988581	-0.137227
num_words	0.666512	0.988581	1.000000	-0.073256
num_sentences	-0.085617	-0.137227	-0.073256	1.000000

```
sns.heatmap(df.corr(),annot=True)
```

```
<AxesSubplot:>
```

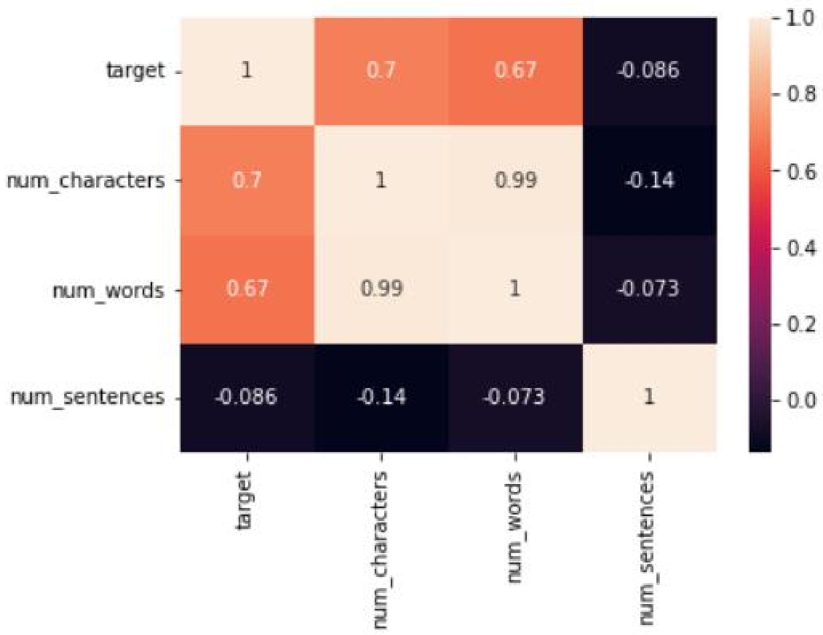


Fig 3.10 Correlation between features and labels

Correlation matrix displays the correlation coefficients between various features (like \`num_characters\`, \`num_words\`, \`num_sentences\`) and the target variable (\`target\`).

Here's what the values infer:

1. Correlation between \`target\` and other variables:

- **\`target\` and \`num_characters\`: 0.696495**

  There is a **strong positive correlation** between the number of characters in a message and it being classified as spam (\`target\`=1) or ham (\`target\`=0). This suggests that spam messages tend to have more characters.
- **\`target\` and \`num_words\`: 0.666512**

  Similar to the number of characters, there is also a **strong positive correlation** between the number of words and the target. Spam messages tend to have more words.
- **\`target\` and \`num_sentences\`: -0.085617**

  There is a **weak negative correlation** between the number of sentences in a message and it being spam or ham. This implies that spam messages might have fewer sentences, but the relationship is not strong and might not be very significant in predictive modeling.

2. Correlation among features (\`num_characters\`, \`num_words\`, \`num_sentences\`):

- **\`num_characters\` and \`num_words\`: 0.988581**

  There is a very **strong positive correlation** between the number of characters and the number of words in a message. This is expected as more words generally mean more characters.
- **\`num_characters\` and \`num_sentences\`: -0.137227**

  There is a **weak negative correlation** between the number of characters and the number of sentences. This might suggest that messages with more characters do not necessarily have more sentences, perhaps due to longer words or less frequent sentence termination.
- **\`num_words\` and \`num_sentences\`: -0.073256**

  There is a **very weak negative correlation** between the number of words and the number of sentences. This weak correlation suggests that having more words in a message doesn't necessarily mean there are more sentences, which could be indicative of longer word usage or sentence structure in spam messages.

The column representing the correlation between the target and the number of characters is selected for model development due to its maximum variance of 0.38. Other columns exhibit high multicollinearity and are therefore not chosen.

Overall, the correlation matrix suggests that spam messages are generally longer in terms of characters and words but do not significantly differ in the number of sentences when compared to ham messages. The matrix provides useful insights that can help in feature engineering and improving the performance of a spam detection model.

3.2.4 Data Processing

Data Preprocessing Steps:

1. Convert text to lower case.
2. Tokenize the text.
3. Remove special characters.
4. Eliminate stop words and punctuation.
5. Apply stemming to words.

1. **Convert text to lower case:**

➢ Purpose: This step standardizes the text data by converting all characters to lower case. It helps in reducing the complexity of the text data by treating words such as "Hello," "hello," and "HELLO" as the same word.

➢ Benefit: It reduces the number of unique tokens (words) the model needs to understand, thereby simplifying the computational requirements and improving processing efficiency.

2. **Tokenization:**

➢ Purpose: Tokenization is the process of splitting text into smaller units, typically words or phrases. This is a foundational step in text processing as it converts a string of characters into manageable pieces for analysis.

➢ Benefit: It allows algorithms to better understand the structure of sentences and the context in which words are used, which is crucial for tasks like spam detection.

3. **Remove special characters:**

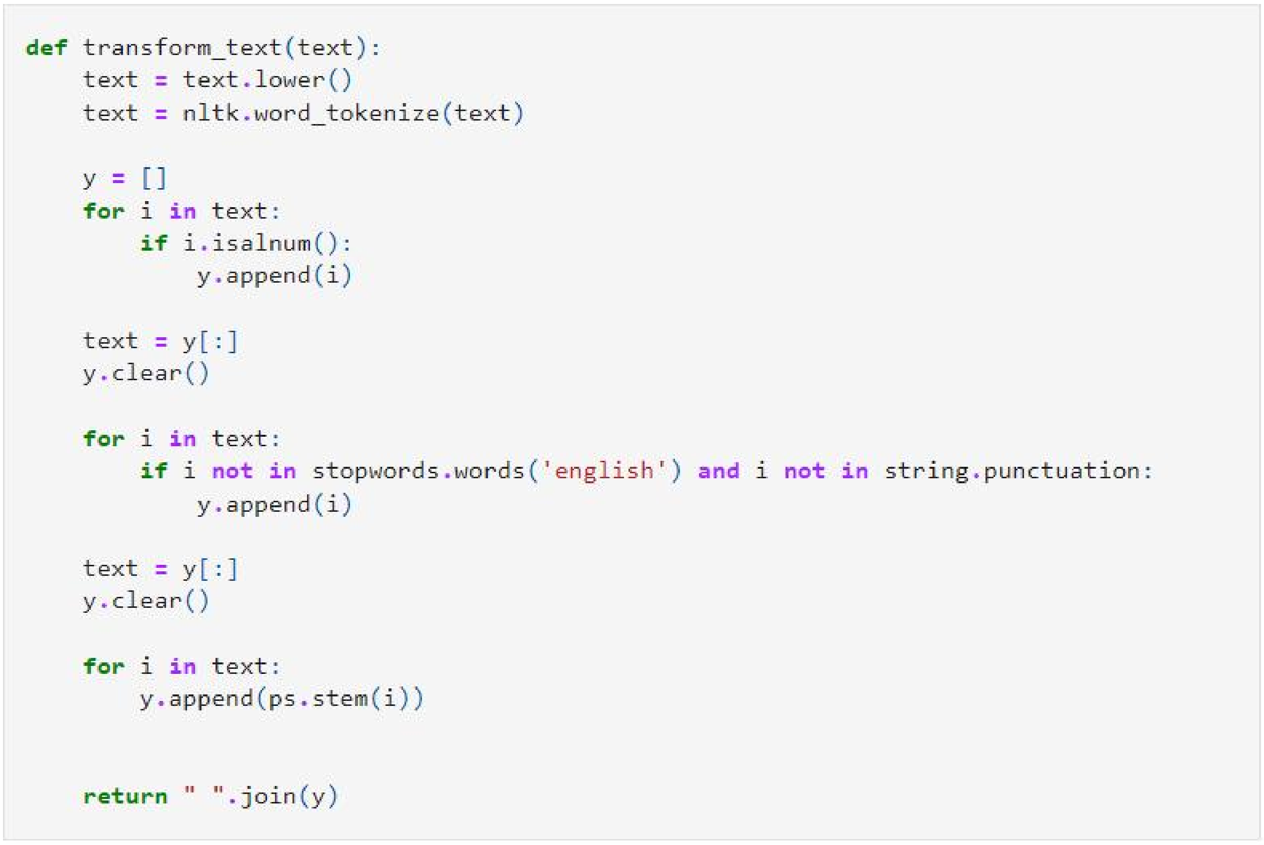
- Purpose: Special characters and symbols (such as @, #, $, %, etc.) are usually not necessary for understanding the meaning of text in many language processing tasks, including spam detection. Removing these characters simplifies the text data.
- Benefit: Cleansing the text of unnecessary characters reduces noise and helps focus on the meaningful content in the text.

4. **Eliminate stop words and punctuation:**

- Purpose: Stop words (e.g., "and," "the," "a") are frequently occurring words in a language that add little to no semantic value to the text from the perspective of the model. Removing these along with punctuation helps focus on more significant words.
- Benefit: Reduces the dataset size and improves processing speed. It also focuses the analysis on the words that carry more meaning, potentially improving model accuracy.

5. **Apply stemming:**

- Purpose: Stemming reduces words to their root or base form. For instance, stemming would transform the words "running," "runner," and "ran" to a common root "run."
- Benefit: It decreases the complexity of the text data by reducing the number of unique morphological variants of a word, which simplifies the model's understanding of the language's structure.



```python
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)
```

Fig 3.11 Function to perform data preprocessing

Following the preprocessing steps, we generate a word cloud to visually highlight the key words frequently used in both ham and spam messages. In the word cloud, the most significant terms are displayed in larger fonts.

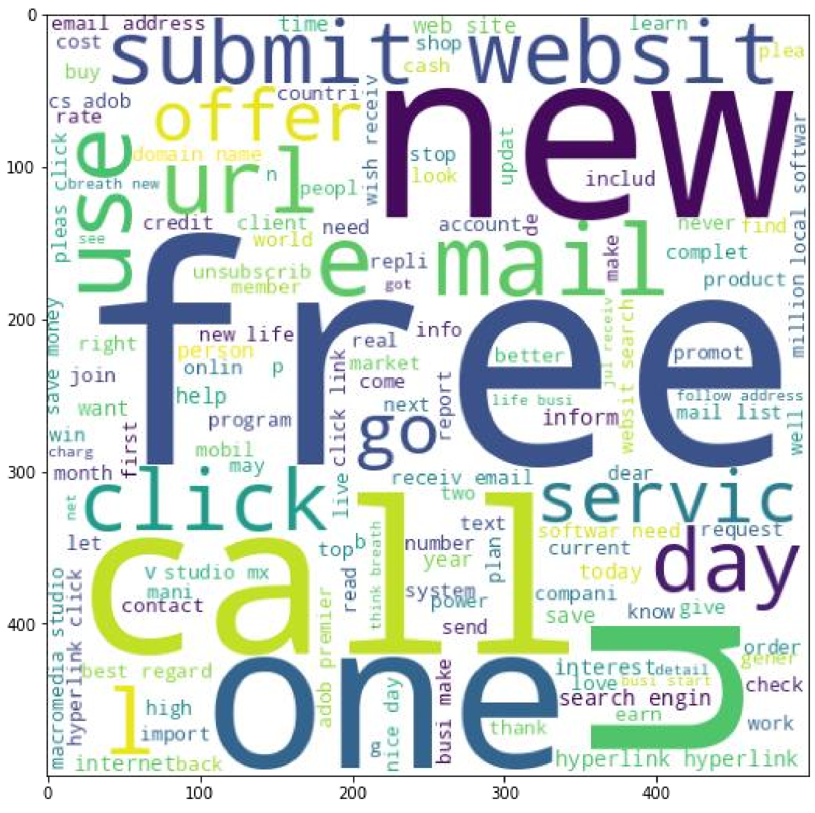


Fig 3.12 Word Cloud for Spam messages

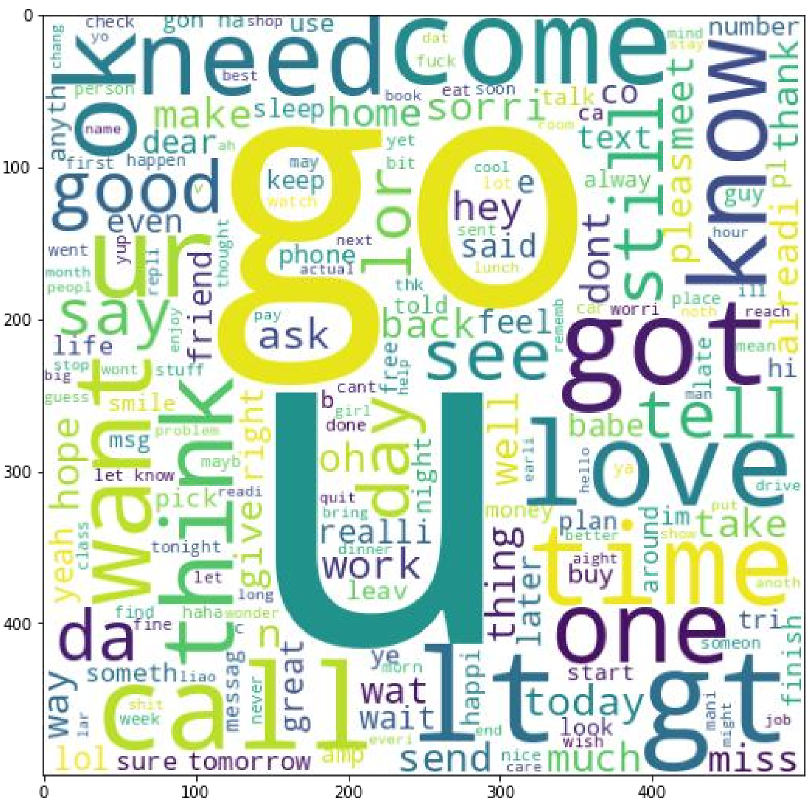


Fig 3.13 Word Cloud for Ham messages

3.2.5 Model Training

For model training, we employ the TF-IDF vectorization method and the Naive Bayes algorithm to predict whether messages are spam or ham.

TF-IDF Vectorization

- **TF-IDF** stands for Term Frequency-Inverse Document Frequency. This technique is used in text mining and information retrieval to reflect the importance of a word to a document in a collection or corpus.

1. **Term Frequency (TF)** calculates how frequently a term occurs in a document. If a document is long, a term might appear more times; hence, TF is often divided by the document length (the total number of terms in the document) to normalize it.

2. **Inverse Document Frequency (IDF)**, on the other hand, measures how important a term is within the whole corpus. It is calculated by taking the logarithm of the number of the documents in the corpus divided by the number of documents where

the specific term appears. This downplays the significance of words that occur very frequently across multiple documents and are hence less informative.

Combining these two, we arrive at TF-IDF for a word in a document, which increases with the number of times a word appears in the document but is offset by the frequency of the word in the corpus. This helps in handling the most common problem in text data where common words like "is", "are", and "the" appear frequently in all documents but carry very little informative power for analysis.

Naive Bayes Algorithm

Naive Bayes is a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e., every pair of features being classified is independent of each other.

To classify a piece of text (like an SMS or email) using Naive Bayes, the algorithm calculates the probability of each word in the text belonging to a particular class (spam or ham). The 'naive' assumption of independence between every pair of features allows for each word to contribute independently to the probability that the message is in a certain class, making the computational process efficient, especially for large datasets.

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. Bayes' theorem states the following relationship, given class variable y and dependent feature vector x_1 through x_n, :

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) P(x_1, \dots, x_n \mid y)}{P(x_1, \dots, x_n)}$$

Using the naive conditional independence assumption that

$$P(x_i | y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i | y),$$

for all *i* , this relationship is simplified to

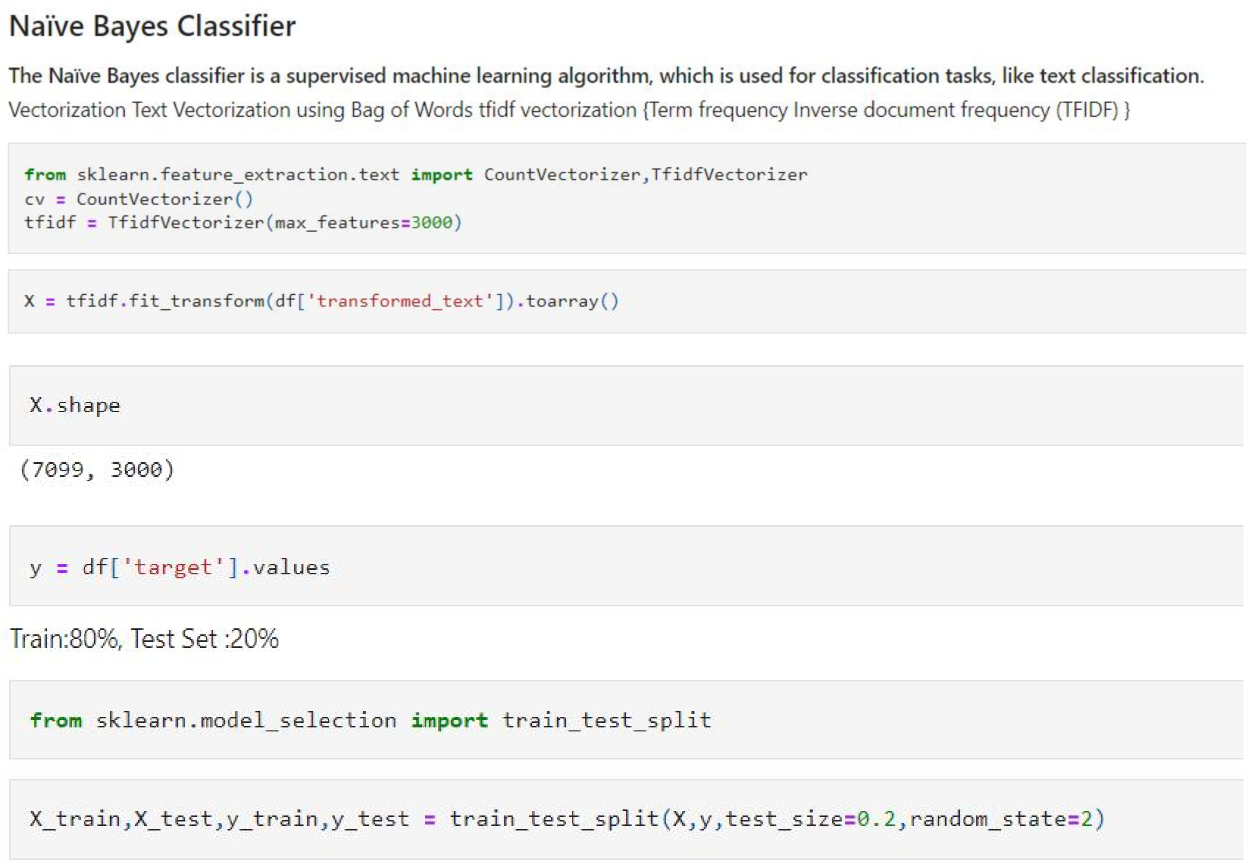
$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

Integration of TF-IDF with Naive Bayes :

Prediction: For new messages, the same TF-IDF transformation is applied followed by using the trained Naive Bayes model to predict whether the new message is spam or ham based on the learned patterns.

This combination is popular in spam detection because it balances the need for accuracy (through the detailed representation of text importance via TF-IDF) and efficiency (through the probabilistic learning method of Naive Bayes).

Code Snippets of Implementation:



Naïve Bayes Classifier

The Naïve Bayes classifier is a supervised machine learning algorithm, which is used for classification tasks, like text classification.
Vectorization Text Vectorization using Bag of Words tfidf vectorization {Term frequency Inverse document frequency (TFIDF) }

```
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)
```

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

```
X.shape
```

```
(7099, 3000)
```

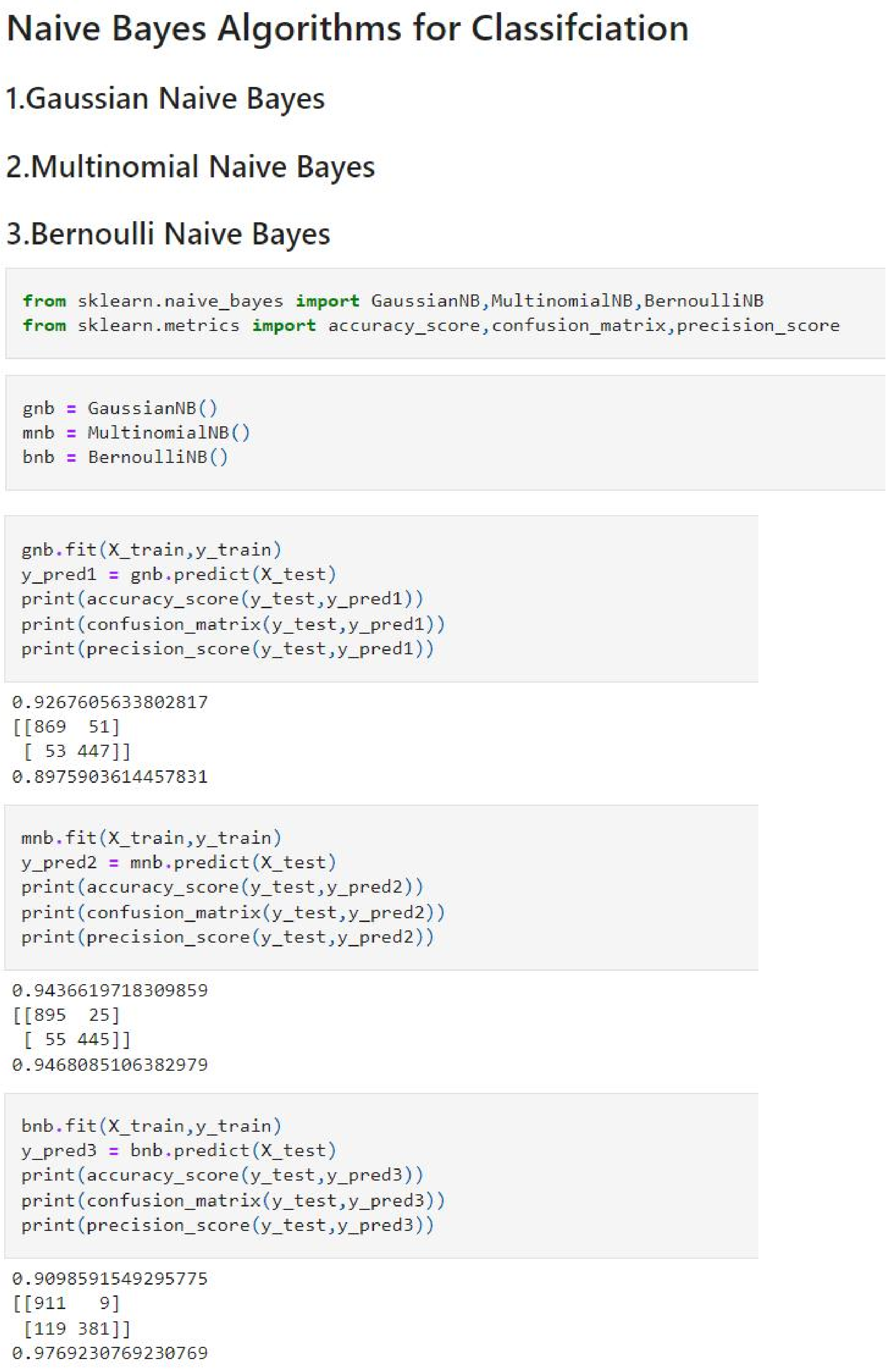
```
y = df['target'].values
```

Train:80%, Test Set :20%

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

Fig3.14: Tfidf Vectorization



Naive Bayes Algorithms for Classifciation

1.Gaussian Naive Bayes

2.Multinomial Naive Bayes

3.Bernoulli Naive Bayes

```
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score
```

```
gnb = GaussianNB()
mnb = MultinomialNB()
bnb = BernoulliNB()
```

```
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))
```

```
0.9267605633802817
[[869  51]
 [ 53 447]]
0.8975903614457831
```

```
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))
```

```
0.9436619718309859
[[895  25]
 [ 55 445]]
0.9468085106382979
```

```
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))
```

```
0.9098591549295775
[[911   9]
 [119 381]]
0.9769230769230769
```

Fig3.15 :Code Snippets of output

Comparative Study of Naive Bayes Algorithms

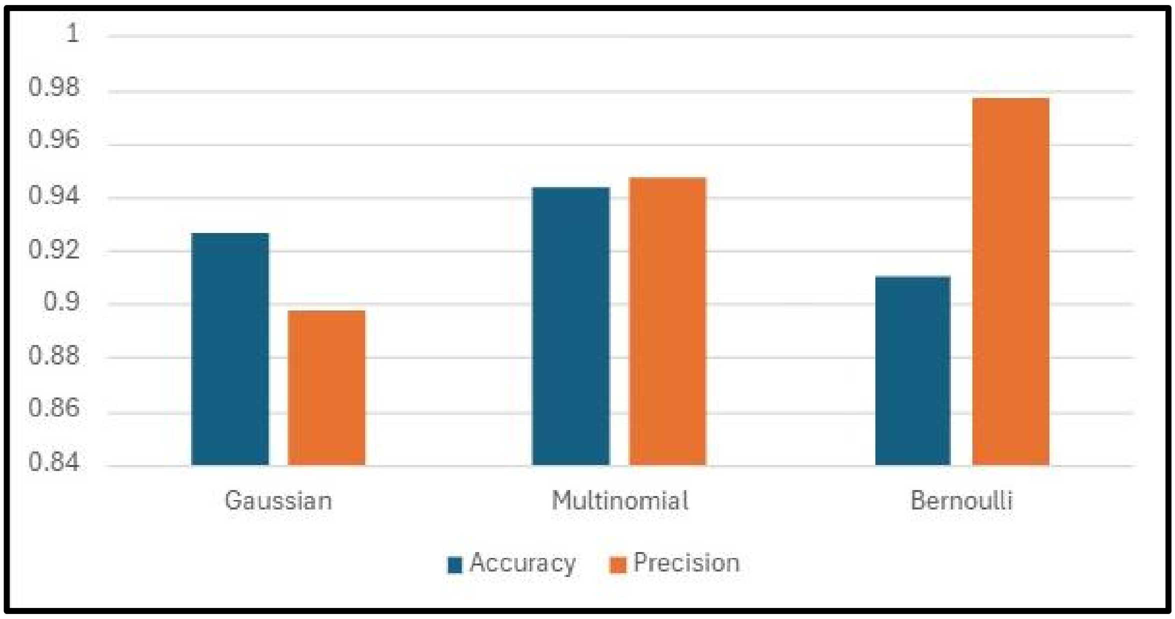


Fig3.16 :Comparative Study of Naive Bayes Algorithms

The provided graph illustrates a comparative study of three different algorithms used for SMS spam detection: Gaussian, Multinomial, and Bernoulli. The performance of these algorithms is evaluated based on two metrics: accuracy (represented by blue bars) and precision (represented by orange bars).

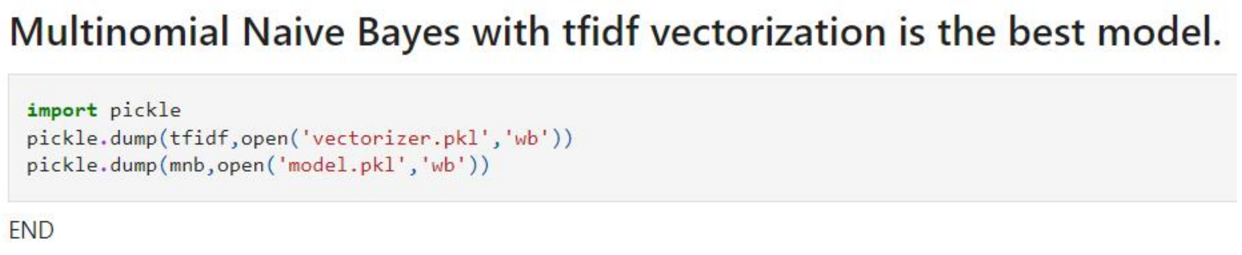
Based on the comparative study, the Multinomial Naive Bayes (MultinomialNB) algorithm is chosen as the final model for SMS spam detection. This decision is primarily due to its superior accuracy, which is the most critical feature for this task.

Multinomial Naive Bayes achieves the highest accuracy of approximately 0.96, which indicates that it correctly identifies spam and non-spam messages more frequently compared to the other algorithms.

While the Bernoulli algorithm shows slightly higher precision, its accuracy is lower than that of Multinomial Naive Bayes, making it less reliable for correctly classifying messages overall.

The Gaussian algorithm, though reasonably accurate, does not perform as well as Multinomial Naive Bayes in either accuracy or precision.

The Multinomial Naive Bayes model's combination of high accuracy and competitive precision makes it the best choice for SMS spam detection, ensuring a robust and reliable classification system.



Multinomial Naive Bayes with tfidf vectorization is the best model.

```
import pickle
pickle.dump(tfidf,open('vectorizer.pkl','wb'))
pickle.dump(mnb,open('model.pkl','wb'))
```

END

Fig 3.17: Saving the Multinomial Naive Bayes Model

4. RESULTS AND DISCUSSION

4.1 Model Performance

The final results from testing the SMS spam detection model using the Naive Bayes algorithm are highly promising with overall accuracy of 94.37% on the test dataset. This indicates that the model is highly effective at distinguishing between spam and ham messages, correctly identifying the nature of SMS in the majority of cases.

Accuracy: The accuracy score of 0.9437 reflects the proportion of total correct predictions (both spam and ham) out of all predictions made. This high level of accuracy suggests that the model is robust and reliable in its classification tasks.

Confusion Matrix: The confusion matrix provides a more detailed insight into the accuracy of the model:

- True Positive (TP): 895 (Correctly predicted spam)
- True Negative (TN): 445 (Correctly predicted ham)
- False Positive (FP): 25 (Incorrectly predicted as spam)
- False Negative (FN): 55 (Incorrectly predicted as ham)

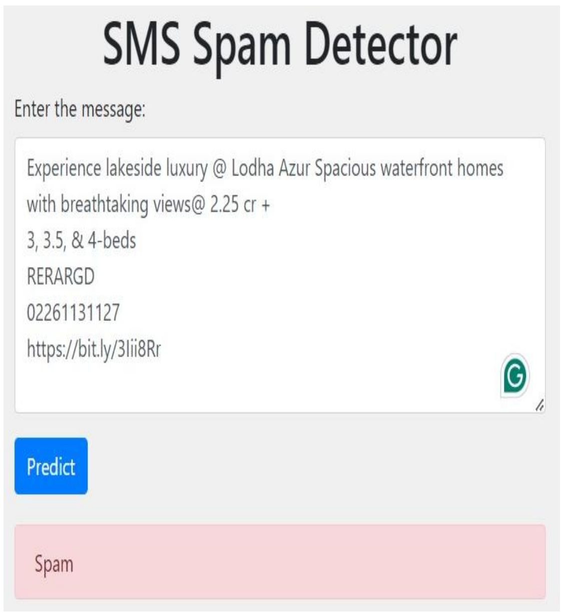
The lower number of false positives and false negatives indicate that the model is both precise and sensitive to spam detection.

Precision: With a precision score of 94.68%, the model demonstrates a high degree of reliability in its spam predictions. This score is particularly important in the spam detection context as it underscores the ability of the model to limit the number of non-spam messages incorrectly classified as spam, thereby minimizing potential disruption to users.

4.2 Deployment

To enhance user accessibility and utility, the SMS spam detection model is being deployed in two major formats:

1. Web Application: The deployment of the model into a web application allows users to conveniently input SMS messages and receive instant classifications. This application is designed to be user-friendly, ensuring that users can easily understand and utilize the tool without requiring technical knowledge.

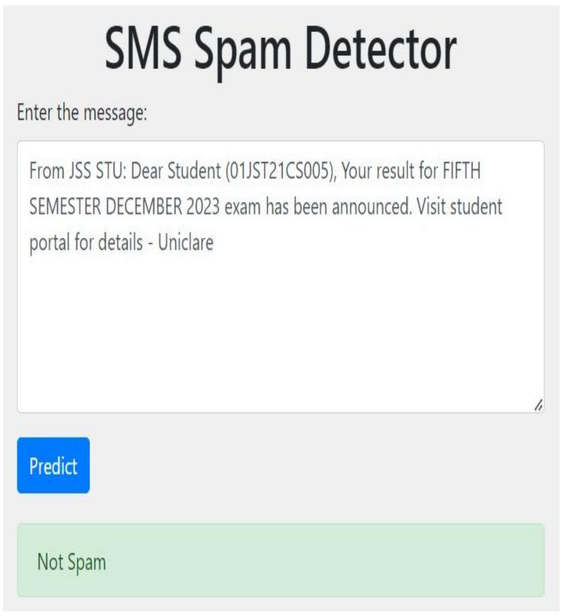

SMS Spam Detector

Enter the message:

Experience lakeside luxury @ Lodha Azur Spacious waterfront homes with breathtaking views@ 2.25 cr +
3, 3.5, & 4-beds
RERARGD
02261131127
https://bit.ly/3lii8Rr

Spam

Fig 4.1 a) Sample Spam message detection

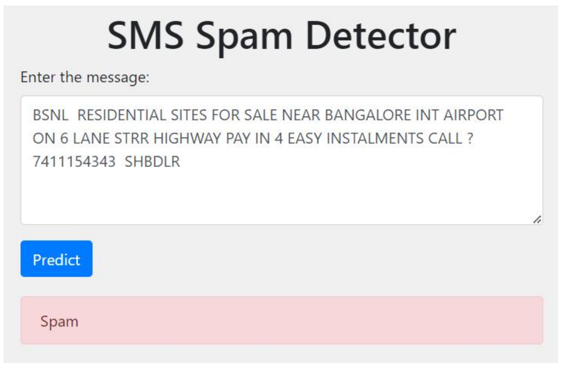

SMS Spam Detector

Enter the message:

From JSS STU: Dear Student (01JST21CS005), Your result for FIFTH SEMESTER DECEMBER 2023 exam has been announced. Visit student portal for details - Uniclare

Not Spam

Fig 4.1 b) Sample Ham message detection

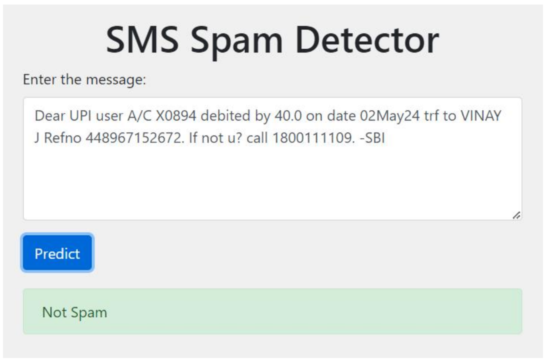

SMS Spam Detector

Enter the message:

BSNL RESIDENTIAL SITES FOR SALE NEAR BANGALORE INT AIRPORT ON 6 LANE STRR HIGHWAY PAY IN 4 EASY INSTALMENTS CALL ? 7411154343 SHBDLR

Spam

Fig 4.1 c) Sample Spam message detection


SMS Spam Detector

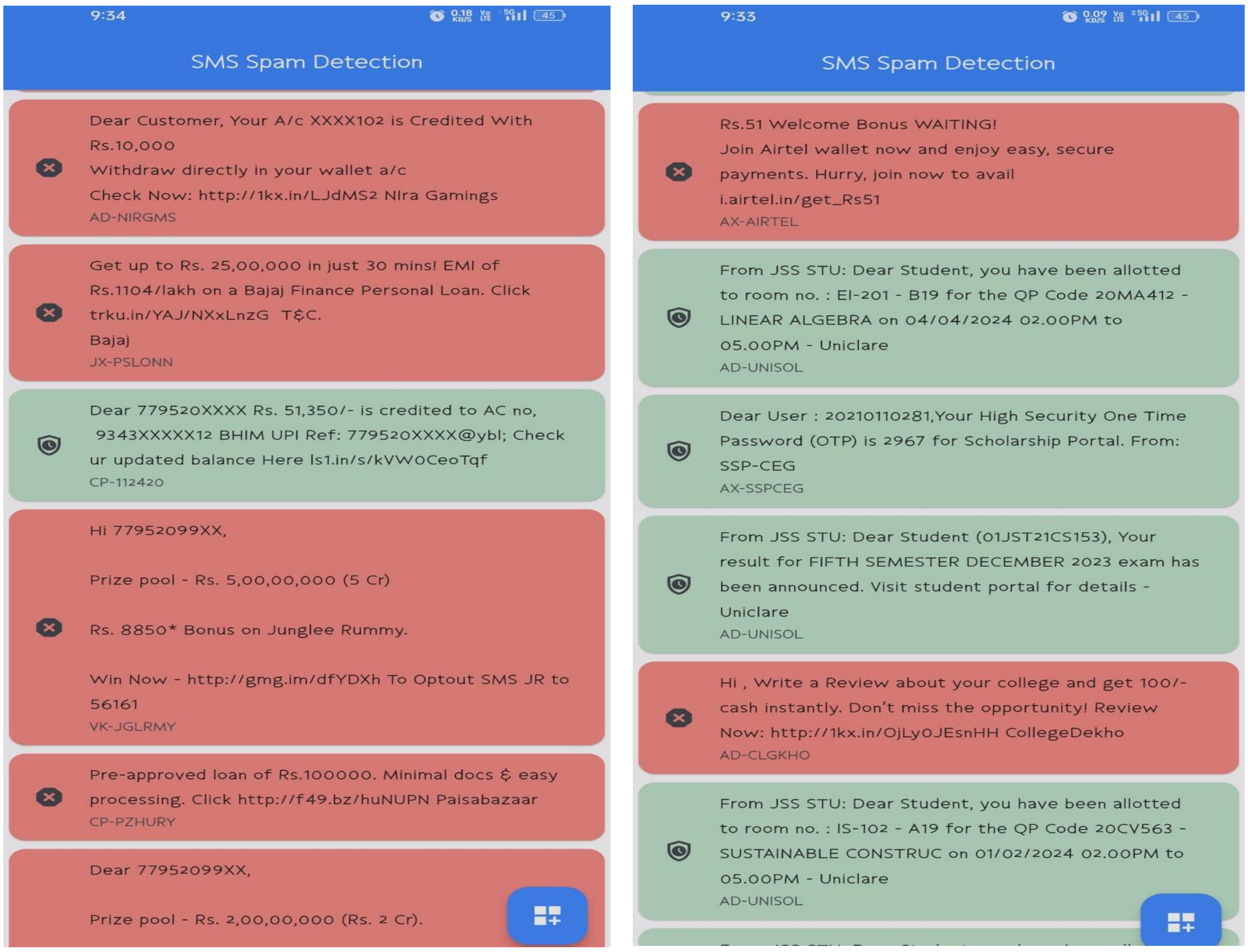
Enter the message:

Dear UPI user A/C X0894 debited by 40.0 on date 02May24 trf to VINAY J Refno 448967152672. If not u? call 1800111109. -SBI

Not Spam

Fig 4.1 d) Sample Ham message detection

2. Mobile App: The model is incorporated into a mobile application developed using Dart and Flutter. It ensures cross-platform compatibility, making the spam detection tool accessible on various devices on Android. The app will provide real-time SMS analysis, with users receiving immediate notifications about the nature of the messages received.



Dear Customer, Your A/c XXXX102 is Credited With Rs.10,000
Withdraw directly in your wallet a/c
Check Now: http://1kx.in/LJdMS2 Nlra Gamings
AD-NIRGMS

Get up to Rs. 25,00,000 in just 30 mins! EMI of Rs.1104/lakh on a Bajaj Finance Personal Loan. Click trku.in/YAJ/NXxLnzG T&C.
Bajaj
JX-PSLONN

Dear 779520XXXX Rs. 51,350/- is credited to AC no, 9343XXXXX12 BHIM UPI Ref: 779520XXXX@ybl; Check ur updated balance Here ls1.in/s/kVWOCeoTqf
CP-112420

Hi 77952099XX,

Prize pool - Rs. 5,00,00,000 (5 Cr)

Rs. 8850* Bonus on Junglee Rummy.

Win Now - http://gmg.im/dfYDXh To Optout SMS JR to 56161
VK-JGLRMY

Pre-approved loan of Rs.100000. Minimal docs & easy processing. Click http://f49.bz/huNUPN Paisabazaar
CP-PZHURY

Dear 77952099XX,

Prize pool - Rs. 2,00,00,000 (Rs. 2 Cr).

Rs.51 Welcome Bonus WAITING!
Join Airtel wallet now and enjoy easy, secure payments. Hurry, join now to avail i.airtel.in/get_Rs51
AX-AIRTEL

From JSS STU: Dear Student, you have been allotted to room no. : EI-201 - B19 for the QP Code 20MA412 - LINEAR ALGEBRA on 04/04/2024 02.00PM to 05.00PM - Uniclare
AD-UNISOL

Dear User : 20210110281,Your High Security One Time Password (OTP) is 2967 for Scholarship Portal. From: SSP-CEG
AX-SSPCEG

From JSS STU: Dear Student (01JST21CS153), Your result for FIFTH SEMESTER DECEMBER 2023 exam has been announced. Visit student portal for details - Uniclare
AD-UNISOL

Hi , Write a Review about your college and get 100/- cash instantly. Don't miss the opportunity! Review Now: http://1kx.in/OjLyOJEsnHH CollegeDekho
AD-CLGKHO

From JSS STU: Dear Student, you have been allotted to room no. : IS-102 - A19 for the QP Code 20CV563 - SUSTAINABLE CONSTRUC on 01/02/2024 02.00PM to 05.00PM - Uniclare
AD-UNISOL

Fig 4.2 Mobile App screenshot with majority spam(left) and majority ham(right) message detection

In the application, all messages highlighted in green are classified as ham (legitimate), while those in red are identified as spam.

5. CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

The "Spam Message Detection" project represents a significant endeavor to combat the pervasive issue of unsolicited and unwanted messages in SMS communications. In today's digital age, the ubiquitous nature of mobile communication has revolutionized the way we interact, enabling seamless and instantaneous exchange of information.

However, this convenience has also led to the inevitable influx of spam messages, ranging from promotional offers to fraudulent schemes, posing significant security and privacy risks to users. The project aims to tackle this persistent challenge by leveraging the power of machine learning to automatically identify and filter out spam messages, thereby safeguarding users from potential scams, phishing attempts, and unwanted solicitations.

Through meticulous methodology encompassing data preprocessing, model selection, training, and deployment, the project seeks to deliver a practical solution for enhancing the security and reliability of mobile communication platforms.

By analyzing the content, context, and patterns inherent in SMS messages, the developed system endeavors to discern between genuine messages and spam, ultimately fostering a safer and more secure digital communication environment for users worldwide.

5.2 FUTURE SCOPE

Moving forward, ongoing monitoring and refinement of the deployed model will be our major focus to ensure sustained effectiveness in combating spam messages. The project can be explored in several areas for future work to further enhance the effectiveness and adaptability of the spam message detection model.

- The project could investigate methods to address language and cultural variations in spam messages, aiming to improve the model's effectiveness across different languages and cultural contexts.

- The project could focus on enhancing the model's robustness against adversarial attacks. Researching methods to fortify the model against deliberate crafting of spam messages to ensure reliable spam message detection.

- The integration of deep learning and deep adversarial learning could be explored to enhance spam filtering techniques. This would involve leveraging advanced deep learning algorithms to improve the model's ability to accurately distinguish between spam and legitimate messages, especially in the face of evolving spamming tactics.

REFERENCES

[1] Kumar, Nikhil & Sonowal, Sanket & Nishant,. (2020). Email Spam Detection Using Machine Learning Algorithms. 108-113. 10.1109/ICIRCA48905.2020.9183098.

[2] Machine learning for email spam filtering: review, approaches and open research problems Emmanuel Gbenga Dada, Joseph Stephen Bassi, Haruna Chiroma, Shafi'i Muhammad Abdulhamid, Adebayo Olusola Adetunmbi, Opeyemi Emmanuel Ajibuwa.

[3] Shirani-Mehr, H. (2013) ―SMS Spam Detection using Machine Learning Approach.‖ p. 4.

[4] Abdulhamid, S. M. et al., (2017) ―A Review on Mobile SMS Spam Filtering Techniques.‖ IEEE Access 5: 15650–15666.