A

Learning Project-II Report

On

"SPAM MAIL PREDICTION USING MACHINE LEARNING WITH PYTHON"

Submitted in partial fulfillment of

The requirements for the 4thSemester Sessional Examination of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

By

SUMAN KUMAR RANA (22UG010854)

SURYA PRAKASH (22UG010972)

MOHILESH MADAN PATRA (22UG010985)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GIET UNIVERSITY, GUNUPUR

2023 - 24

GIET UNIVERSITY

Gunupur-765022, At – Gobriguda, Po- Kharling, Dist. - Rayagada, Odisha

Contact:- +91 7735745535, 06857-250170,172,Visit us:- **www.giet.edu**

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the project work entitled "SPAM MAIL PREDICTION USING MACHINE LEARNING WITH PYTHON "is done by Name- SUMAN KUMAR RANA, SURYA PRAKASH, MOHILESH MADAN PATRA

Regd. No. – 22UG010854, 22UG010972, 22UG010985 in partial fulfillment of the requirements for the 4th Semester Sessional Examination of Bachelor of Technology in Computer Science and Engineering during the academic year 2023-24. This work is submitted to the department as a part of evaluation of 4th Semester Learning Project-II.

Proctors/Class Teacher *Project Coordinator, 2nd Year*

HoD, CSE, 2nd Year

ACKNOWLEDGEMENT

We express our sincere gratitude to Itishree Panda of Computer science and engineering for giving us an opportunity to accomplish the project. Without his/her active support and guidance, this project report has not been successfully completed.

We also thanks Mr. Bhavani Sankar Panda, Project Coordinator, Dr. Premanshu Sekhar Rath, Head of the Department of Computer Science and Engineering, Prof. (Dr.) Kakita Murali Gopal, Dy. Dean, Computational Science, SOET for their consistent support, guidance and help.

We also thanks to our friends and family members for their constant support during the execution of the project.

SUMAN KUMAR RANA (22UG010854)

SURYA PRAKASH (22UG010972)

MOHILESH MADAN PATRA (22UG010985)

TABLE OF CONTENTS

CHAPTER – 1

1.1 ABSTRACT

In this project, we developed a robust spam email classifier using Python libraries and Google Colab's collaborative environment. Leveraging NLTK, Scikit-learn. we implemented advanced natural language processing techniques for text classification and feature extraction.

Initially, the dataset was cleaned by removing unnecessary columns and duplicate entries. Exploratory Data Analysis (EDA) revealed insights into the imbalance of spam and ham emails. Visualizations and statistical analyses provided valuable information about the dataset's characteristics.

For data preprocessing, we employed techniques such as tokenization, removal of stop words and punctuation, and stemming. The transformed text data was visualized using word clouds and analyzed to identify frequent words in spam and ham emails.

Model building involved text vectorization using TF-IDF, followed by training and evaluation of several classifiers, including Gaussian Naive Bayes, Multinomial Naive Bayes, and others. Due to the dataset's imbalance, Multinomial Naive Bayes exhibited the best performance.

To further enhance model performance, various techniques were explored, including changing TF-IDF parameters, scaling numerical features, and ensemble methods such as Voting Classifier and Stacking Classifier. These approaches improved both accuracy and precision in classifying spam emails.

Finally, the trained model was saved using pickle for future use. The project demonstrates an effective solution for classifying spam emails, utilizing machine learning techniques and collaborative tools to streamline the development process.

1.2 INTRODUCTION

In today's digital era, email communication stands as an indispensable tool for personal and professional interactions. However, amidst the convenience and efficiency of email, there exists a persistent and disruptive menace: spam. Spam emails, characterized by their unsolicited and often malicious content, pose a substantial threat to individuals, businesses, and organizations worldwide. From phishing attempts seeking sensitive information to malware-laden messages aiming to compromise systems, the repercussions of falling prey to spam can be severe, leading to financial loss, data breaches, and compromised security.

To combat this pervasive issue, we introduce a comprehensive machine learning project focused on the classification of spam emails. The primary objective of this endeavor is to develop an accurate and reliable system capable of discerning between legitimate (ham) and spam emails with precision and efficiency. By harnessing the power of advanced algorithms, natural language processing (NLP) techniques, and collaborative development environments, our aim is to provide users with a robust tool to bolster email security and streamline productivity.

This project unfolds through several key stages, beginning with the acquisition and preprocessing of a diverse dataset comprising email samples from various sources and categories. Through meticulous data cleaning and preparation, we ensure the dataset's integrity and suitability for analysis. Exploratory data analysis (EDA) techniques are then employed to glean insights into the dataset's distribution, patterns, and characteristics, guiding our approach to feature extraction.

Utilizing advanced NLP techniques such as tokenization, stop word removal, and stemming, we extract meaningful features from the email text, transforming raw data into a format conducive to machine learning model training. Subsequently, we embark on the process of model selection, training, and evaluation, employing a range of classifiers and ensemble methods to identify the most effective approach for spam classification.

Finally, the culmination of this project is the deployment of the trained model as an interactive web application, accessible to users for real-time email classification. Through meticulous documentation and ongoing maintenance, we endeavor to ensure the project's accessibility, reliability, and efficacy in combating the pervasive threat of email spam.

1.2.1 PURPOSE

The purpose of the project is to develop a highly accurate and efficient solution for classifying spam emails, thereby enhancing email security and productivity for users. With the exponential growth of digital communication, the incessant influx of spam emails poses a significant threat, ranging from phishing scams to malware distribution. Addressing this challenge requires a sophisticated approach that can reliably differentiate between legitimate and malicious emails.

Through the utilization of machine learning techniques and collaborative tools, the project aims to achieve the following objectives:

1.Effective Spam Detection: By leveraging advanced natural language processing (NLP) algorithms and feature extraction methods, the project aims to accurately classify incoming emails as either spam or ham (non-spam). This ensures that users can swiftly identify and filter out unwanted emails, thereby minimizing the risk of falling victim to various online threats.

2.Streamlined Workflow: The integration of Google Colab's collaborative environment facilitates seamless collaboration among project contributors, enabling efficient code development, data preprocessing, model training, and evaluation. This streamlined workflow accelerates the project's progress and allows for rapid iteration and experimentation, leading to quicker deployment of effective solutions.

3.User-Friendly Interface: The transformation of the trained machine learning model into an interactive web application using Streamlit enhances accessibility for users. With a simple text input, individuals can easily determine the spam or ham status of their emails, empowering them to make informed decisions about their inbox management.

4.Continuous Improvement: Through rigorous evaluation and experimentation, the project aims to continually refine and improve the spam email classifier's performance. By exploring various techniques such as data preprocessing, feature engineering, and ensemble methods, the project seeks to enhance the model's accuracy and robustness in handling diverse email datasets.

5.Knowledge Sharing: The project also serves as a platform for knowledge sharing and collaboration within the machine learning community. By documenting the development process, sharing code snippets, and discussing insights and challenges, the project contributes to the collective understanding and advancement of spam detection techniques.

Overall, the project's purpose is to provide users with a reliable and user-friendly tool for mitigating the risks associated with spam emails, ultimately promoting a safer and more efficient email experience in today's digital landscape.

1.2.2 SCOPE

The scope of the project encompasses various aspects related to spam email classification, including data preprocessing, feature extraction, model building, evaluation, and deployment. Here's a detailed breakdown of the scope:

1.Data Preprocessing: The project involves cleaning and preprocessing the email dataset to remove irrelevant columns, handle missing values, and eliminate duplicate entries. Text data undergoes normalization techniques such as lowercasing, tokenization, removal of stop words and punctuation, and stemming. This ensures that the data is in a suitable format for further analysis and model training.

2.Feature Extraction: Feature engineering plays a crucial role in identifying informative patterns in the email data. The project explores techniques such as TF-IDF (Term Frequency-Inverse Document Frequency) for text vectorization, which captures the importance of words in distinguishing between spam and ham emails. Additionally, numerical features such as the number of characters, words, and sentences in each email are extracted to provide additional context for classification.

3.Model Building: Various machine learning algorithms are evaluated for their effectiveness in classifying spam emails. This includes traditional algorithms such as Naive Bayes, Decision Trees, and ensemble methods like Random Forest and Gradient Boosting. The project also considers deep learning approaches using neural networks for more complex pattern recognition.

4.Evaluation Metrics: The performance of each model is assessed using appropriate evaluation metrics such as accuracy, precision, recall, and F1-score. Cross-validation techniques are employed to ensure robustness and generalization of the models. Special consideration is given to handling the imbalance in the dataset to prevent biased performance metrics.

5.Deployment: Upon selecting the best-performing model, it is deployed as an interactive web application using Streamlit. This allows users to input text and receive real-time predictions on whether the email is spam or ham. Additionally, the trained model can be saved and reused for future classification tasks.

6.Scalability and Generalization: The project aims to develop a scalable solution that can handle large volumes of emails while maintaining high classification accuracy. Generalization of the model is also crucial, ensuring that it can adapt to new and unseen email patterns and remain effective over time.

7.Documentation and Maintenance: Comprehensive documentation is provided for the project, including explanations of algorithms, code snippets, and usage instructions for the web application. Regular maintenance and updates are planned to incorporate new techniques and improve the model's performance based on user feedback and emerging email spam trends.

Overall, the scope of the project encompasses the entire lifecycle of spam email classification, from data preprocessing to model deployment, with a focus on delivering a reliable and user-friendly solution for email security.

1.2.3 FEATURES

The project encompasses a variety of features aimed at delivering a comprehensive and effective solution for spam email classification. Here's a detailed overview of the project's features:

1.Data Cleaning and Preprocessing: The project begins with thorough data cleaning and preprocessing to ensure that the email dataset is free from inconsistencies, missing values, and duplicate entries. Techniques such as removing unnecessary columns, handling missing data, and eliminating duplicates are employed to prepare the dataset for further analysis.

2.Exploratory Data Analysis (EDA): EDA techniques are applied to gain insights into the characteristics of the email dataset. Visualizations and statistical analyses are used to understand the distribution of spam and ham emails, identify patterns, and detect any potential biases or anomalies in the data.

3.Text Feature Extraction: Advanced natural language processing (NLP) techniques are utilized to extract meaningful features from the text data. This includes tokenization, removal of stop words and punctuation, stemming, and vectorization using techniques such as TF-IDF (Term Frequency-Inverse Document Frequency).

4.Model Building and Evaluation: The project explores a variety of machine learning algorithms, including traditional classifiers like Naive Bayes and Decision Trees, as well as ensemble methods such as Random Forest and Gradient Boosting. Models are evaluated using appropriate performance metrics such as accuracy, precision, recall, and F1-score, with a focus on handling the imbalance in the dataset to prevent biased evaluations.

5.Interactive Web Application: The trained machine learning model is deployed as an interactive web application using Streamlit. This allows users to input text and receive real-time predictions on whether the email is spam or ham. The web application provides a user-friendly interface for easy access and utilization of the spam email classification system.

6.Scalability and Performance: The project is designed to be scalable, capable of handling large volumes of emails while maintaining high classification accuracy. Efforts are made to optimize the performance of the model, ensuring efficient processing and classification of emails in real-time.

7.Documentation and Maintenance: Comprehensive documentation is provided for the project, including explanations of algorithms, code snippets, and usage instructions for the web application. Regular maintenance and updates are planned to incorporate new techniques, address user feedback, and adapt to emerging email spam trends.

The project's features combine to deliver a robust, scalable, and user-friendly solution for spam email classification, empowering users to effectively manage their email security and productivity.

CHAPTER – 2

SYSTEM ANALYSIS

2.1 HARDWARE REQUIREMENT

LAPTOP/PC:

Operating System Version –

Microsoft Windows 7/8/10 (32- or 64-bit)

Linux

MacOS

RAM –

4 GB RAM minimum; 8 GB RAM recommended.

Free digital storage –

2 GB of available digital storage minimum, 4 GB Recommended

2.2 SOFTWARE REQUIREMENT

1. VS CODE:

Visual Studio Code is an editor that favours simplicity over having an endless assortment of bells and whistles. Most of the functionality is exposed by typing in the thing you need into the top search bar. Everything you would expect to be there is there: debugging, breakpoints, etc.

2. GOOLE CLOUD:

Google Drive integration with Google Colab, often referred to as "Google Drive collab," is a powerful feature that allows users to seamlessly access and manipulate files stored in their Google Drive directly within the Google Colab environment. Google Colab (short for Colaboratory) is a cloud-based Jupyter notebook environment that enables users to write and execute Python code collaboratively.

3. PYTHON 3.11 :

Python is a high-level programming language renowned for its simplicity, readability, and versatility. With an intuitive syntax resembling plain English, Python is accessible to beginners and conducive to rapid development. Its interpreted nature allows for interactive coding experiences, fostering experimentation and exploration. Supported by a vibrant community and a vast ecosystem of libraries and frameworks, Python excels in diverse domains including web development, data analysis, machine learning, scientific computing, automation, and more. Cross-platform compatibility enables seamless deployment across different operating systems, while high-level language features such as dynamic typing, automatic memory management, and extensive standard libraries empower developers to express complex ideas concisely and efficiently. Python's dominance in data science and machine learning is underscored by its rich toolkit of libraries and frameworks, making it the language of choice for practitioners and researchers alike. Overall, Python's simplicity, versatility, and robust ecosystem make it a go-to language for programmers across industries and skill levels.

CHAPTER – 3

3.1 LANGUAGE USED

PYTHON:

Python is extensively used in machine learning (ML) due to its simplicity, versatility, and rich ecosystem of libraries and frameworks tailored for ML tasks. Here are some key ways Python is utilized in ML:

1. Libraries and Frameworks: Python offers a wide range of libraries and frameworks specifically designed for ML tasks. Some of the most popular ones include:

- scikit-learn: A comprehensive library for classical ML algorithms such as linear regression, decision trees, random forests, support vector machines, and more.

- TensorFlow and Keras: TensorFlow is a powerful open-source framework for building and training deep learning models, while Keras provides a high-level API for building neural networks.

- PyTorch: Another popular deep learning framework known for its flexibility and ease of use, particularly favored by researchers and practitioners in academia.

- pandas: A data manipulation library that provides data structures and functions for working with structured data, essential for data preprocessing and analysis in ML workflows.

2. Data Manipulation and Preprocessing: Python's libraries such as pandas and NumPy provide powerful tools for data manipulation and preprocessing. These libraries enable tasks such as loading data from various sources, cleaning and transforming data, handling missing values, encoding categorical variables, and scaling numerical features.

3. Model Building and Training: Python's ML libraries facilitate the building and training of ML models across various domains. Developers can leverage algorithms and techniques provided by these libraries to train models on labeled datasets, optimize model parameters, and evaluate model performance using metrics such as accuracy, precision, recall, and F1-score.

4. Model Evaluation and Validation: Python provides tools for evaluating and validating ML models to ensure their effectiveness and generalizability. Techniques such as cross-validation, hyperparameter tuning, and model selection help optimize models for performance and robustness.

5. Deployment and Integration: Python enables the deployment and integration of ML models into production environments. Frameworks like Flask and Django facilitate the development of web services and

APIs for serving ML models, while tools like TensorFlow Serving and TensorFlow Lite enable deployment on various platforms and devices.

6. Visualization and Interpretation: Python's libraries for data visualization, such as Matplotlib, Seaborn, and Plotly, help visualize data distributions, model predictions, and evaluation metrics. Visualization aids in understanding data patterns, model behavior, and communicating insights to stakeholders.

7. Research and Experimentation: Python's ease of use and flexibility make it ideal for conducting research and experimentation in ML. Researchers and practitioners can quickly prototype ideas, test algorithms, and analyze results using Python's interactive environments like Jupyter notebooks.

Python's extensive ecosystem of libraries, ease of use, and flexibility make it the language of choice for machine learning practitioners across various domains, from data preprocessing and model building to deployment and interpretation of results.

PYTHON STREAMLIT:

Streamlit is an open-source Python library that allows developers to create web applications for machine learning and data science projects with minimal effort. It simplifies the process of building interactive web-based user interfaces (UIs) for data-driven applications, enabling developers to focus on writing Python code rather than dealing with the complexities of web development.

Here are some key features and aspects of Streamlit:

1. Simple and Intuitive: Streamlit provides a simple and intuitive API that allows developers to create interactive web apps using familiar Python syntax. Developers can quickly prototype and iterate on their ideas without needing expertise in web development.

2. Reactive Programming: Streamlit uses reactive programming concepts, where changes to UI elements automatically trigger updates to the underlying data and vice versa. This enables the creation of dynamic and responsive applications that react to user inputs in real-time.

3. Rapid Development: With Streamlit, developers can build fully functional web applications in just a few lines of code. The library handles many common web development tasks, such as layout management, widget creation, and data visualization, automatically, saving developers time and effort.

4. Wide Range of Widgets: Streamlit provides a wide range of built-in widgets for creating interactive elements such as sliders, buttons, dropdowns, checkboxes, and text inputs. Developers can easily incorporate these widgets into their applications to enable user interaction and control.

5. Integration with Python Ecosystem: Streamlit seamlessly integrates with popular Python libraries for data manipulation, analysis, and visualization, such as pandas, NumPy, Matplotlib, Plotly, and scikit-learn. Developers can leverage the capabilities of these libraries within their Streamlit applications to create rich, data-driven experiences.

6. Customization and Styling: While Streamlit simplifies many aspects of web development, it also offers customization options for developers who want to tailor the appearance and behavior of their applications. Developers can customize the layout, styling, and interaction behavior of their Streamlit apps using CSS and custom JavaScript.

7. Deployment Options: Streamlit applications can be deployed easily to various platforms, including local servers, cloud services such as Heroku, and containerized environments like Docker. Streamlit provides built-in commands for deploying applications, making the deployment process straightforward and accessible to developers.

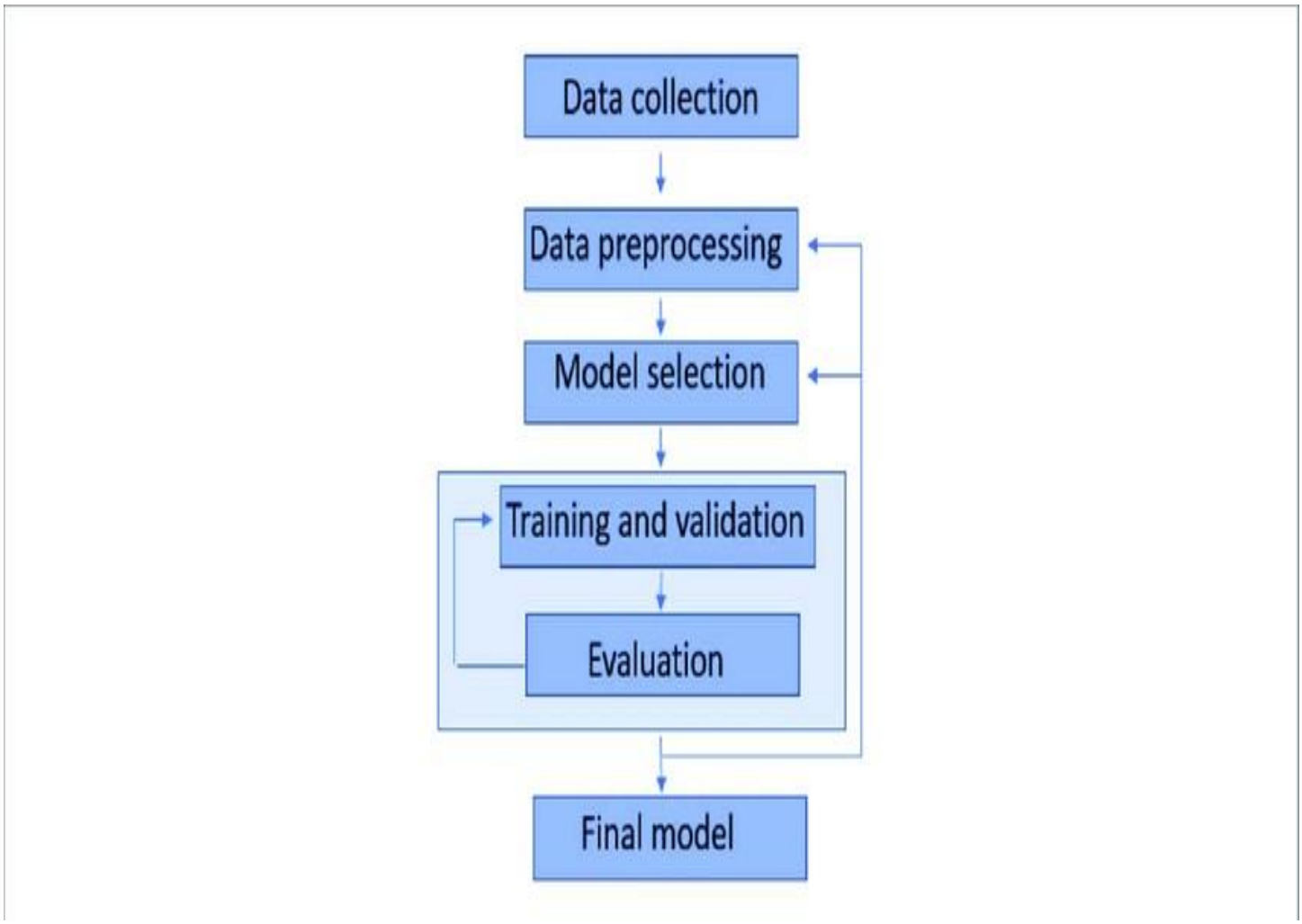
8. Active Community and Ecosystem: Streamlit has a vibrant and growing community of developers who contribute to the library's development, share tips and best practices, and create extensions and plugins to enhance its functionality. The active community provides support, resources, and inspiration for developers building Streamlit applications.

Streamlit empowers Python developers to create interactive web applications for data science and machine learning projects quickly and efficiently, enabling them to share their work, insights, and findings with a wider audience.

CHAPTER – 4

SYSTEM DESIGN & SPECIFICATION

4.1 FLOW CHART

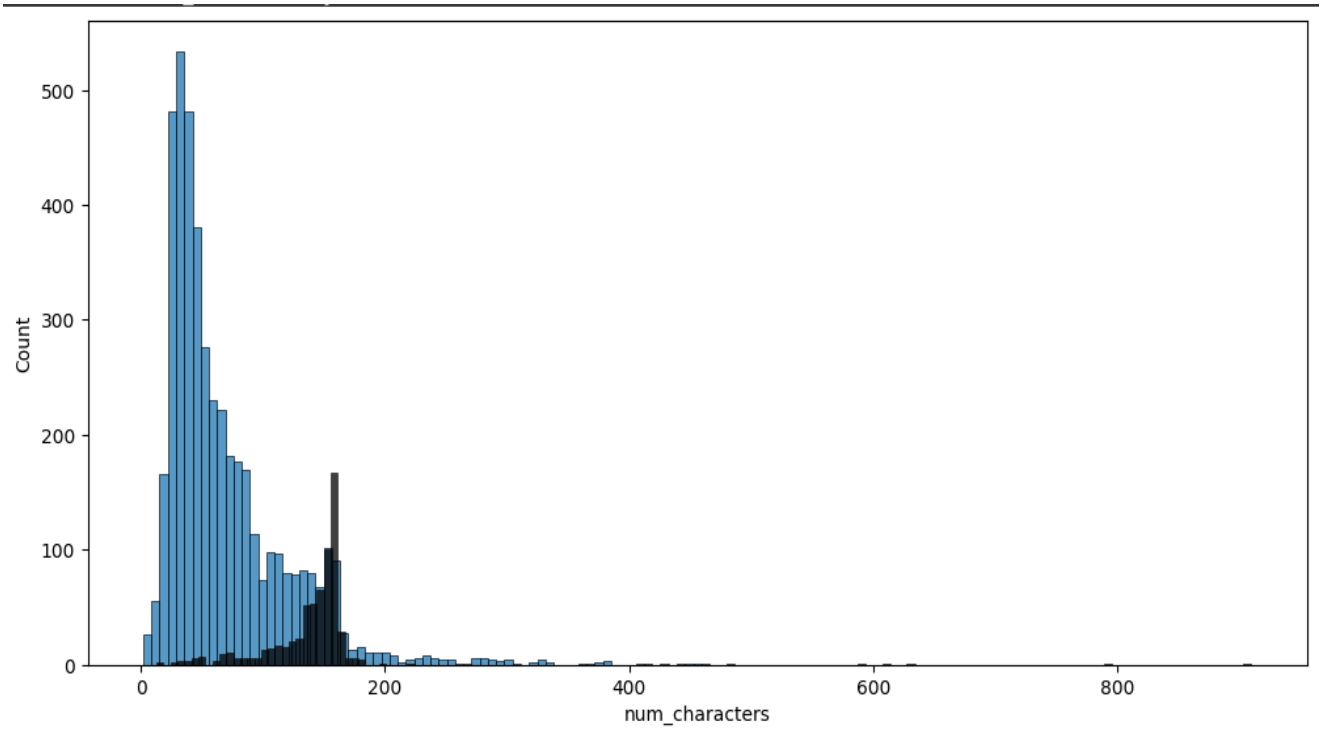


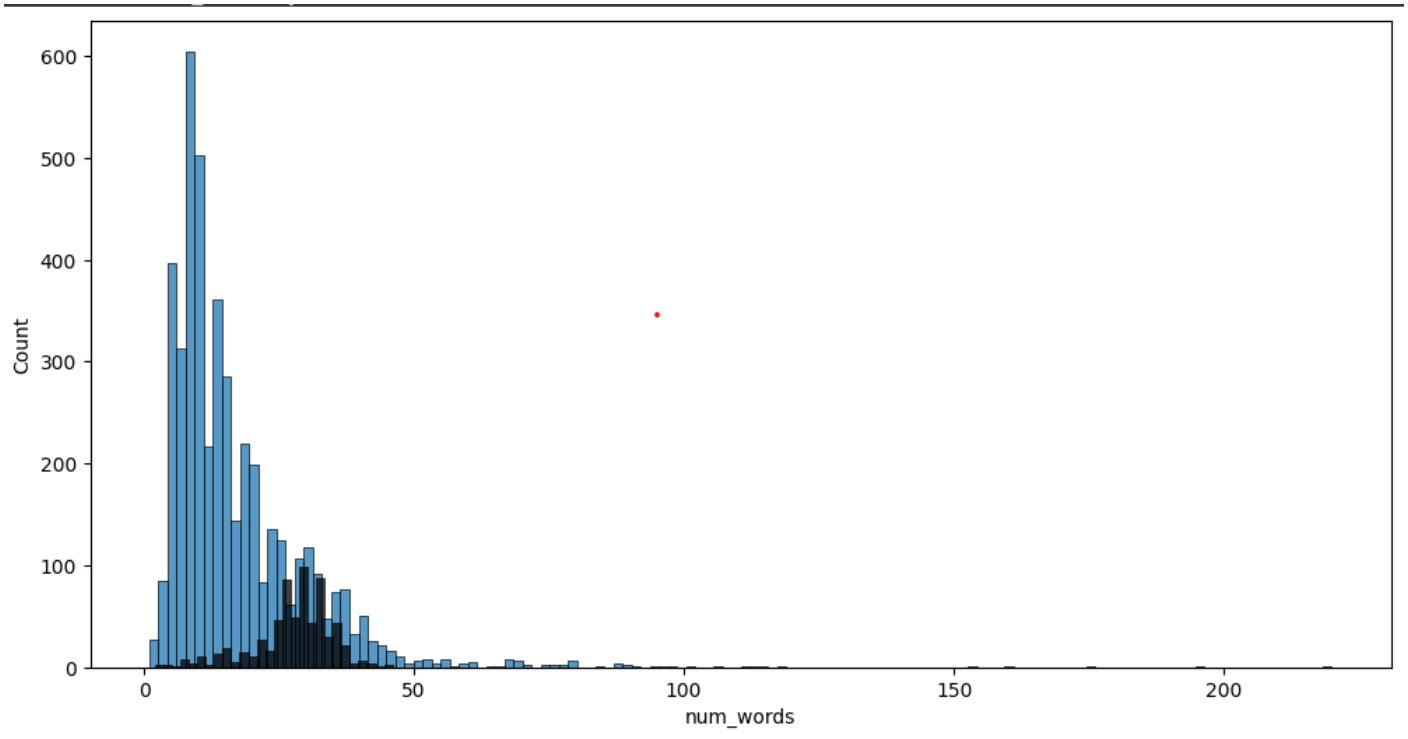
4.2 SCREEN SHOTS

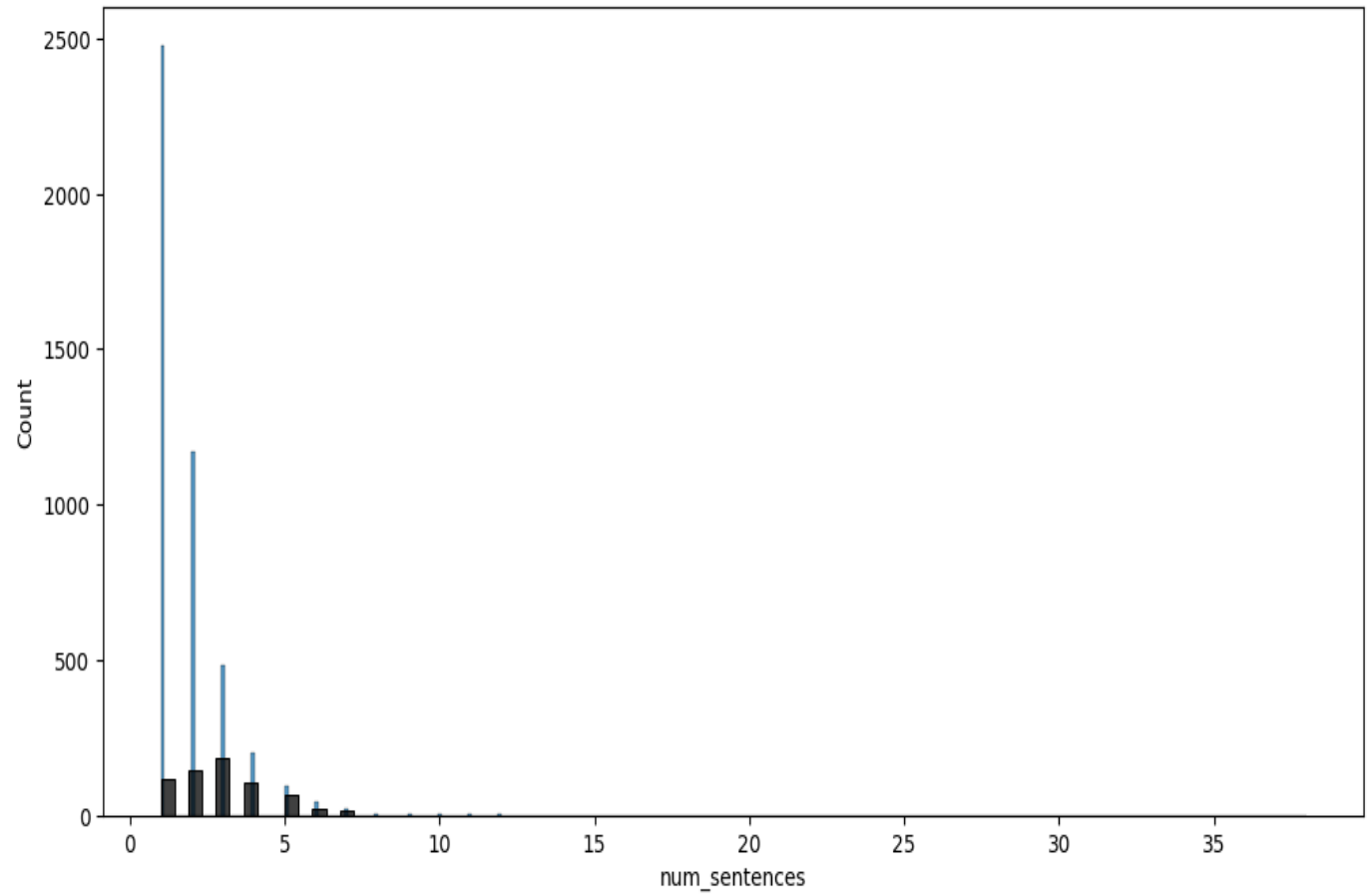
DATASET

	v1	v2
2	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
3	ham	Ok lar... Joking wif u oni...
4	spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
5	ham	U dun say so early hor... U c already then say...
6	ham	Nah I don't think he goes to usf, he lives around here though
7	spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs to send, å£1.50 to rcv
8	ham	Even my brother is not like to speak with me. They treat me like aids patent.
9	ham	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to copy your friends Callertune
10	spam	WINNER!! As a valued network customer you have been selected to receivea å£900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.
11	spam	Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free! Call The Mobile Update Co FREE on 08002986030
12	ham	I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.
13	spam	SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 info
14	spam	URGENT! You have won a 1 week FREE membership in our å£100,000 Prize Jackpot! Txt the word: CLAIM to No: 81010 T&C www.dbuk.net LCCLTD POBOX 4403LDNW1A7RW18
15	ham	I've been searching for the right words to thank you for this breather. I promise i wont take your help for granted and will fulfil my promise. You have been wonderful and a blessing at all time
16	ham	I HAVE A DATE ON SUNDAY WITH WILL!!
17	spam	XXXMobileMovieClub: To use your credit, click the WAP link in the next txt message or click here>> http://wap. xxxmobilemovieclub.com?n=QJKGIGHJJGCBL
18	ham	Oh k...i'm watching here:)
19	ham	Eh u remember how 2 spell his name... Yes i did. He v naughty make until i v wet.
20	ham	Fine if thatåÕs the way u feel. thatåÕs the way its gota b
21	spam	England v Macedonia - dont miss the goals/team news. Txt ur national team to 87077 eg ENGLAND to 87077 Try:WALES, SCOTLAND 4txt/ì¼1.20 POBOXox36504W45WQ 16+
22	ham	Is that seriously how you spell his name?
23	ham	I‰Û÷m going to try for 2 months ha ha only joking
24	ham	So ì_ pay first lar... Then when is da stock comin...
25	ham	Aft i finish my lunch then i go str down lor. Ard 3 smth lor. U finish ur lunch already?
26	ham	Ffffffffff. Alright no way I can meet up with you sooner?
27	ham	Just forced myself to eat a slice. I'm really not hungry tho. This sucks. Mark is getting worried. He knows I'm sick when I turn down pizza. Lol
28	ham	Lol your always so convincing.
29	ham	Did you catch the bus ? Are you frying an egg ? Did you make a tea? Are you eating your mom's left over dinner ? Do you feel my Love ?
30	ham	I'm back & we're packing the car now, I'll let you know if there's room
31	ham	Ahhh. Work. I vaguely remember that! What does it feel like? Lol
32	ham	Wait that's still not all that clear, were you not sure about me being sarcastic or that that's why x doesn't want to live with us
33	ham	Yeah he got in at 2 and was v apologetic. n had fallen out and she was actin like spoilt child and he got caught up in that. Till 2! But we won't go there! Not doing too badly cheers. You?
34	ham	K tell me anything about you.
35	ham	For fear of fainting with the of all that housework you just did? Quick have a cuppa
36	spam	Thanks for your subscription to Ringtone UK your mobile will be charged å£5/month Please confirm by replying YES or NO. If you reply NO you will not be charged
37	ham	Yup... Ok i go home look at the timings then i msg ì_ again... Xuhui going to learn on 2nd may too but her lesson is at 8am
38	ham	Oops, I'll let you know when my roommate's done

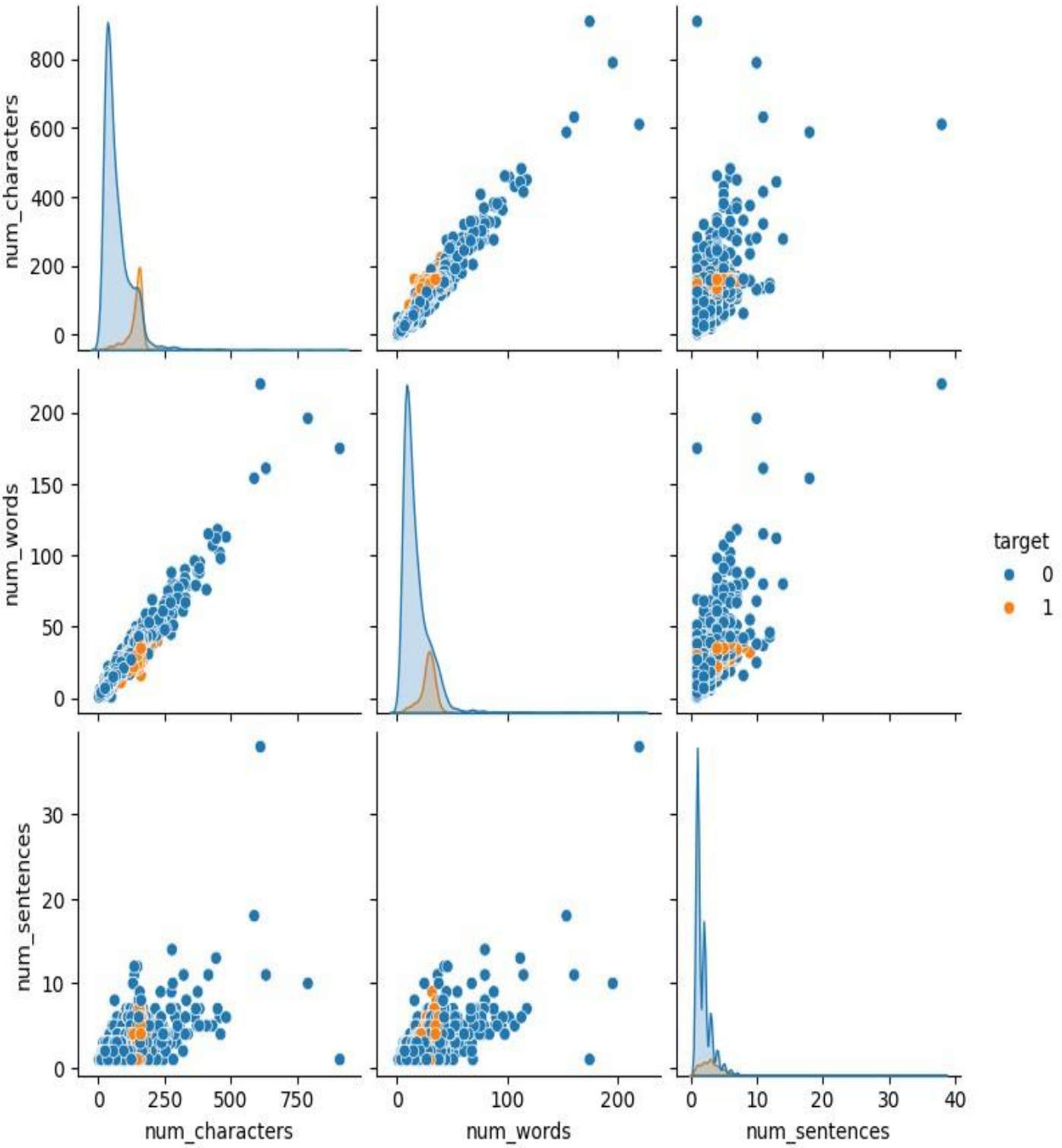
Exploratory Data Analysis (EDA)



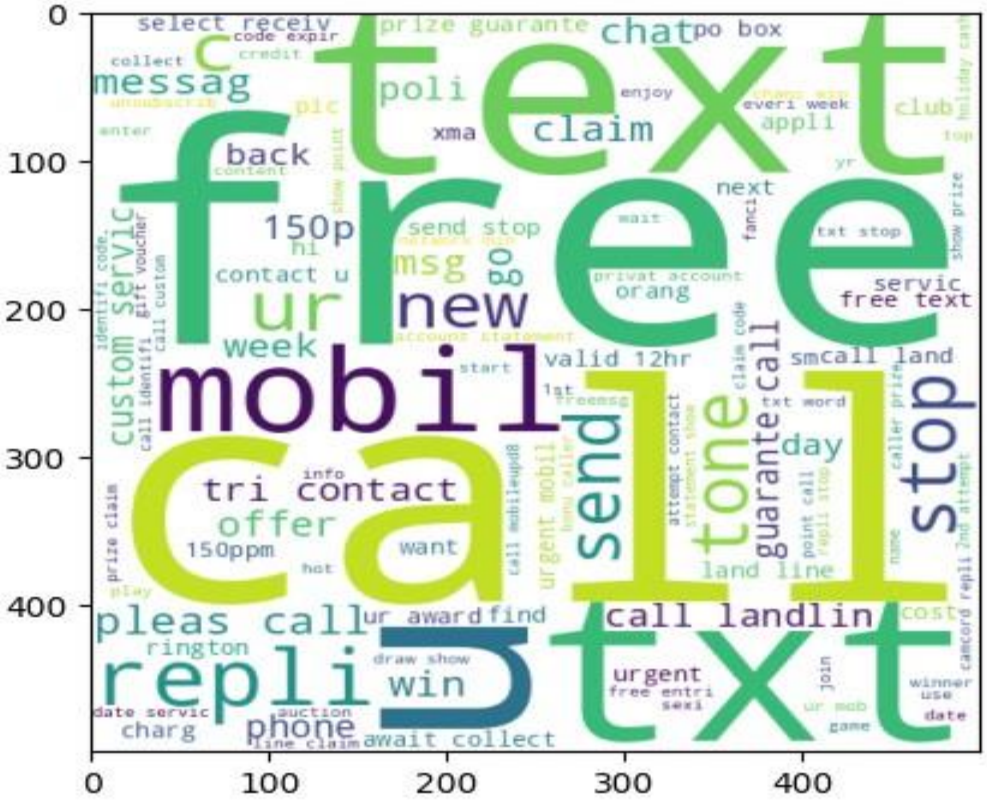


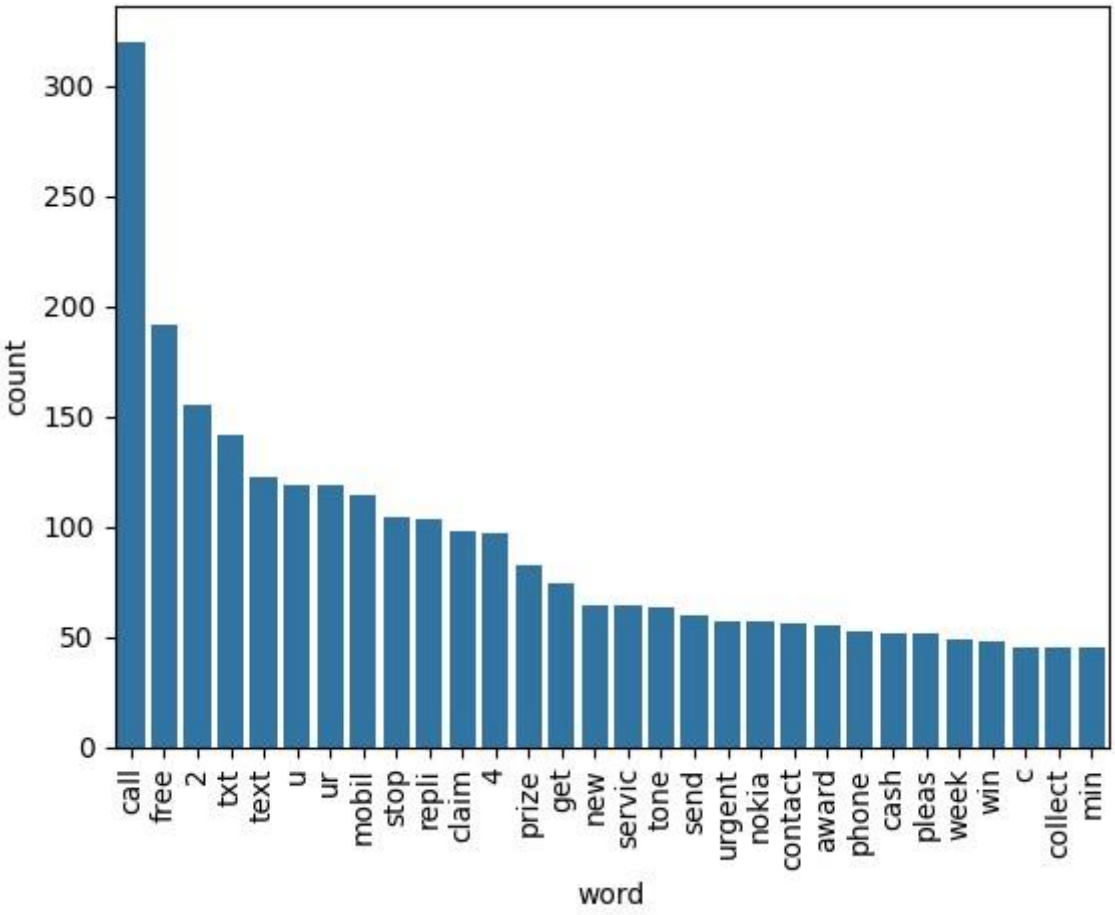

2500
2000
1500
1000
500
0
Count
0
5
10
15
20
25
30
35
num_sentences

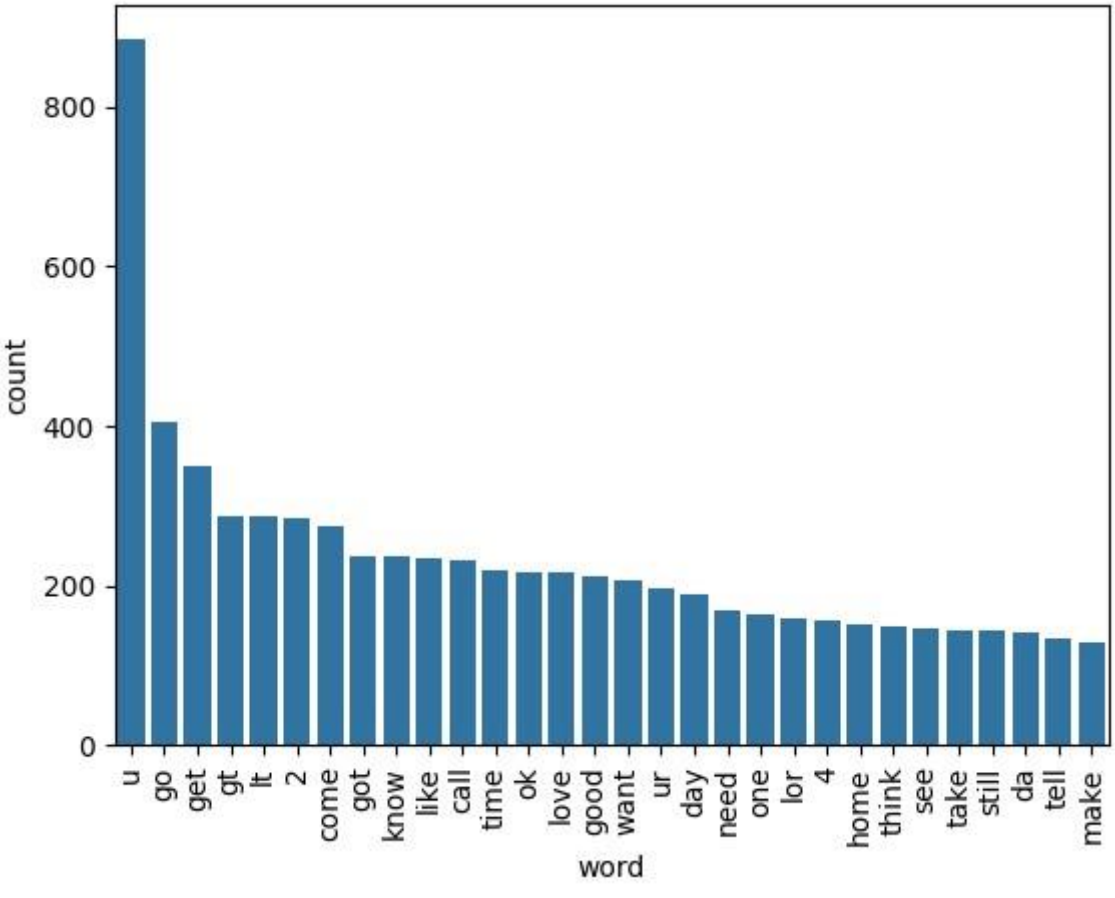

target
num_characters
num_words
num_sentences
1
0.38
0.26
0.26
0.38
1
0.97
0.62
0.26
0.97
1
0.68
0.26
0.62
0.68
1
target
num_characters
num_words
num_sentences
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3


num_characters
num_words
num_sentences
target
0
1
0
250
500
750
0
100
200
0
10
20
30
40

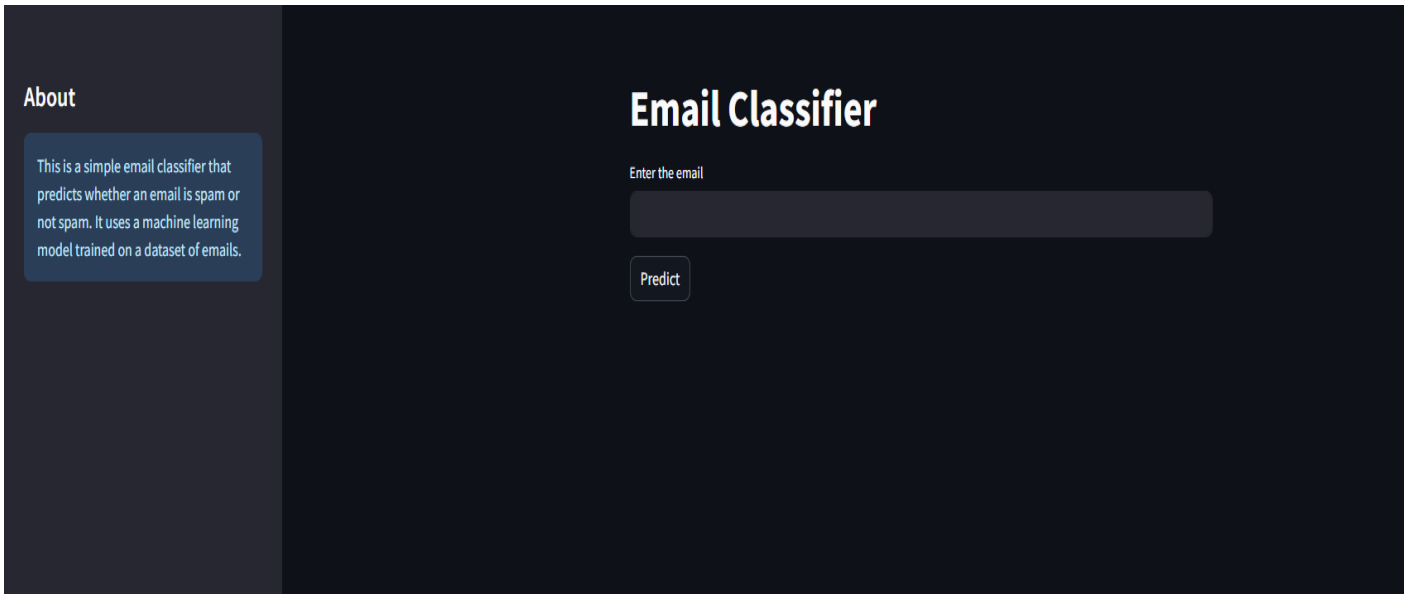
DATA PREPROCESSING:

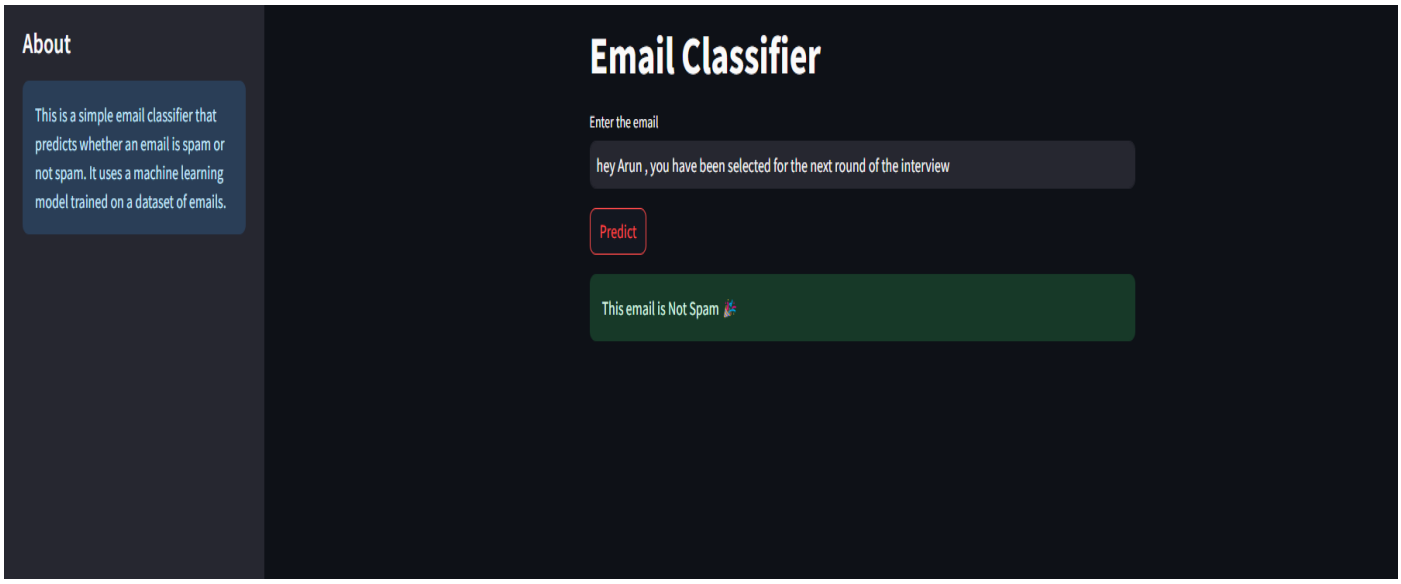


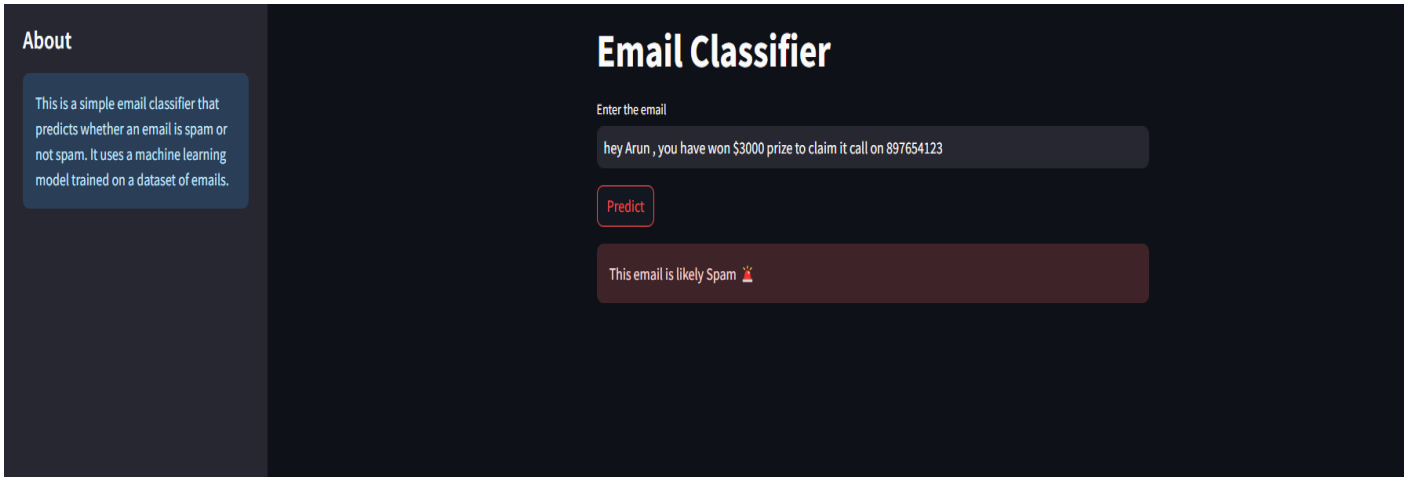

count
0
50
100
150
200
250
300
call
free
2
txt
text
u
ur
mobil
stop
repli
claim
4
prize
get
new
servic
tone
send
urgent
nokia
contact
award
phone
cash
pleas
week
win
c
collect
min
word


count
0
200
400
600
800
u
go
get
gt
lt
2
come
got
know
like
call
time
ok
love
good
want
ur
day
need
one
lor
4
home
think
see
take
still
da
tell
make
word

USER INPUT SECTION:







CHAPTER - 5

PROJECT CODING

5.1 Importing Libraries and Dataset

```
import numpy as np
import pandas as pd

df = pd.read_csv('spam.csv')

df.sample(5)

df.shape
```

5.2 Data Cleaning

```
df.info()

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

df.sample(5)

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

from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()

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

df.head()

# missing values
df.isnull().sum()
```

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

```
# check for duplicate values
df.duplicated().sum()

# remove duplicates
df = df.drop_duplicates(keep='first')

df.duplicated().sum()

df.shape
```

5.3 EDA

```
df.head()

df['target'].value_counts()
```

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

# Data is imbalanced

import nltk

!pip install nltk

nltk.download('punkt')

df['num_characters'] = df['text'].apply(len)

df.head()

# num of words
df['num_words'] = df['text'].apply(lambda x:len(nltk.word_tokenize(x)))

df.head()

df['num_sentences'] = df['text'].apply(lambda x:len(nltk.sent_tokenize(x)))

df.head()

df[['num_characters','num_words','num_sentences']].describe()

# ham
df[df['target'] == 0][['num_characters','num_words','num_sentences']].describe()

#spam
df[df['target'] == 1][['num_characters','num_words','num_sentences']].describe()

import seaborn as sns

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

plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_words'])
sns.histplot(df[df['target'] == 1]['num_words'],color='red')

sns.pairplot(df,hue='target')

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

5.4 Data Preprocessing

- Lower case
- Tokenization
- Removing special characters
- Removing stop words and punctuation
- Stemming

```
nltk.download('stopwords')

from nltk.corpus import stopwords

stopwords.words('english')

import string
```

```
string.punctuation
```

```
from nltk.stem.porter import PorterStemmer
```

```
ps=PorterStemmer()
```

```
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)
```

```
df['transformed_text'] = df['text'].apply(transform_text)
```

```
df.head()
```

```
from wordcloud import WordCloud
wc =WordCloud(width=500,height=500,min_font_size=10,background_color='white')
```

```
spam_wc =wc.generate(df[df['target']==1]['transformed_text'].str.cat(sep=""))
plt.imshow(spam_wc)
```

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

```
plt.figure(figsize=(15,6))
plt.imshow(ham_wc)
```

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

```
len(spam_corpus)
```

```
from collections import Counter
sns.barplot(pd.DataFrame(Counter(spam_corpus).most_common(30))[0],pd.DataFrame(Counter
(spam_corpus).most_common(30))[1])
plt.xticks(rotation='vertical')
plt.show()
```

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

len(ham_corpus)

from collections import Counter
sns.barplot(pd.DataFrame(Counter(ham_corpus).most_common(30))[0],pd.DataFrame(Counter(
ham_corpus).most_common(30))[1])
plt.xticks(rotation='vertical')
plt.show()

# Text Vectorization
# using Bag of Words
df.head()
```

5.5 Model Building

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

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

# appending the num_character col to X
X = np.hstack((X,df['num_characters'].values.reshape(-1,1)))

X.shape

y = df['target'].values

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)

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))

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))

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))

# tfidf --> MNB
```

```
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

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)

clfs = {
    'SVC' : svc,
    'KN' : knc,
    'NB': mnb,
    'DT': dtc,
    'LR': lrc,
    'RF': rfc,
    'AdaBoost': abc,
    'BgC': bc,
    'ETC': etc,
    'GBDT':gbdt,
    'xgb':xgb
}

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

train_classifier(svc,X_train,y_train,X_test,y_test)

accuracy_scores = []
precision_scores = []

for name,clf in clfs.items():

    current_accuracy,current_precision = train_classifier(clf, X_train,y_train,X_test,
y_test)

    print("For ",name)
    print("Accuracy - ",current_accuracy)
```

```
    print("Precision - ",current_precision)

    accuracy_scores.append(current_accuracy)
    precision_scores.append(current_precision)

performance_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy':accuracy_scores,'Pre
cision':precision_scores}).sort_values('Precision',ascending=False)

performance_df

performance_df1 = pd.melt(performance_df, id_vars = "Algorithm")

performance_df1

sns.catplot(x = 'Algorithm', y='value',
               hue = 'variable',data=performance_df1, kind='bar',height=5)
plt.ylim(0.5,1.0)
plt.xticks(rotation='vertical')
plt.show()

# model improve
# 1. Change the max_features parameter of TfIdf

temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_max_ft_3000':accuracy_scores
,'Precision_max_ft_3000':precision_scores}).sort_values('Precision_max_ft_3000',ascend
ing=False)

temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_scaling':accuracy_scores,'Pr
ecision_scaling':precision_scores}).sort_values('Precision_scaling',ascending=False)

new_df = performance_df.merge(temp_df,on='Algorithm')

new_df_scaled = new_df.merge(temp_df,on='Algorithm')

temp_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy_num_chars':accuracy_scores,'
Precision_num_chars':precision_scores}).sort_values('Precision_num_chars',ascending=Fa
lse)

new_df_scaled.merge(temp_df,on='Algorithm')

# Voting Classifier
svc = SVC(kernel='sigmoid', gamma=1.0,probability=True)
mnb = MultinomialNB()
etc = ExtraTreesClassifier(n_estimators=50, random_state=2)

from sklearn.ensemble import VotingClassifier

voting = VotingClassifier(estimators=[('svm', svc), ('nb', mnb), ('et', etc)],voting='
soft')

voting.fit(X_train,y_train)

VotingClassifier(estimators=[('svm', SVC(gamma=1.0, kernel='sigmoid', probability=True
)), ('nb', MultinomialNB()),'et', ExtraTreesClassifier(n_estimators=50, random_state=2
))], voting='soft')

y_pred = voting.predict(X_test)
print("Accuracy",accuracy_score(y_test,y_pred))
print("Precision",precision_score(y_test,y_pred))
```

```
# Applying stacking
estimators=[('svm', svc), ('nb', mnb), ('et', etc)]
final_estimator=RandomForestClassifier()

from sklearn.ensemble import StackingClassifier

clf = StackingClassifier(estimators=estimators, final_estimator=final_estimator)

clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print("Accuracy",accuracy_score(y_test,y_pred))
print("Precision",precision_score(y_test,y_pred))

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

5.6 Python Streamlit Code

```
import streamlit as st

import pickle

import string

import nltk

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

from sklearn.feature_extraction.text import TfidfVectorizer

# Download NLTK resources

nltk.download('punkt')

nltk.download('stopwords')

# Load the pre-trained model and TF-IDF vectorizer

with open('model1.pkl', 'rb') as model_file:

    model = pickle.load(model_file)

with open('vectorizer1.pkl', 'rb') as vectorizer_file:

    tfidf = pickle.load(vectorizer_file)

ps = PorterStemmer()

# Function for text transformation

def transform_text(text):

    text = text.lower()

    text = nltk.word_tokenize(text)

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

    text = [i for i in y if i not in stopwords.words('english') and i not in string.pu
nctuation]
```

```
    y = [ps.stem(i) for i in text]
    return " ".join(y)
# Streamlit UI
st.set_page_config(page_title="Email Classifier", page_icon="📧")
st.title("Email Classifier")
input_mail = st.text_input("Enter the email")
if st.button('Predict'):
    # 1. Preprocessing
    transformed_mail = transform_text(input_mail)
    # 2. Vectorize
    vector_input = tfidf.transform([transformed_mail])
    # 3. Predict
    result = model.predict(vector_input)[0]
    # 4. Display
    if result == 1:
        st.error("This email is likely Spam 🚨")
    else:
        st.success("This email is Not Spam 🎉")
# Sidebar with information
st.sidebar.title("About")
st.sidebar.info(
    "This is a simple email classifier that predicts whether an email is spam or not s
pam. "
    "It uses a machine learning model trained on a dataset of emails."
)
```

CONCLUSION

In conclusion, our project presents a robust solution for spam email classification, leveraging advanced machine learning techniques and collaborative development environments. Through meticulous data preprocessing, feature extraction, and model building, we have successfully developed a system capable of accurately distinguishing between spam and legitimate emails with high precision.

The deployment of the trained model as an interactive web application further enhances its accessibility and usability, allowing users to quickly and easily classify emails in real-time. By streamlining email management and bolstering security, our project contributes to a safer and more efficient digital communication experience.

Moving forward, continuous maintenance and updates will be crucial to ensure the project remains effective in the face of evolving spam tactics and user needs. Additionally, further exploration of advanced techniques such as deep learning and ensemble methods could enhance the model's performance and scalability.

Overall, our project underscores the importance of leveraging machine learning and collaborative tools to address contemporary challenges in email security, empowering users to mitigate the risks associated with spam and safeguard their digital communication channels effectively.

LIMITATIONS

One limitation of the project lies in the reliance on supervised learning techniques, which necessitate labeled data for model training. While the availability of labeled datasets facilitates the development of accurate classifiers, it also imposes constraints on scalability and adaptability. As spam tactics evolve over time, the model's effectiveness may diminish without continuous updates and retraining on new data. Additionally, the project's performance may be influenced by the quality and representativeness of the training data, leading to biases or inaccuracies in classification.

Furthermore, the project's focus on textual features may overlook other potentially informative attributes present in email metadata or attachments. Incorporating additional features such as sender information, email headers, and attachment analysis could enhance the model's discriminatory power and robustness.

Moreover, the imbalance in the dataset, where spam emails may outnumber legitimate ones, poses a challenge for model evaluation and may lead to biased performance metrics. Addressing this imbalance through techniques like oversampling, under sampling, or cost-sensitive learning is essential to ensure fair evaluation and effective classification.

Lastly, the project's reliance on open-source libraries and tools may limit customization and flexibility, constraining the implementation of specialized techniques or optimizations tailored to specific use cases or domains.

REFERENCES

Books:

"Python Data Science Handbook" by Jake VanderPlas - This book offers an in-depth exploration of data science tools and techniques using Python. It covers data manipulation, visualization, and machine learning with examples and case studies that can be applied to spam email classifier projects.

Websites and Online Resources:

1. Kaggle (https://www.kaggle.com/) - Kaggle is a platform for data science competitions and datasets. It offers a wide range of datasets related to spam email and sms, along with notebooks and kernels shared by the community, which can serve as valuable resources for learning and building predictive models.

2. GitHub (https://github.com/) - GitHub hosts numerous open-source projects and repositories related to machine learning and healthcare. You can find Python code, libraries, and resources for building spam email classifier by searching GitHub repositories and exploring projects shared by the community.

4. Official Documentation and Tutorials:
   - scikit-learn Documentation (https://scikit-learn.org/stable/documentation.html)
   - NLTK Documentation: (https://www.nltk.org/)
   - Streamlit Documentation (https://docs.streamlit.io/)