SALARY PREDICTION

A project Report Submitted for

3rd Year

CSE-DS

by

Name of Students	Roll No.
Anish Kumar	2301331549001
Aryan Yadav	2301331549002
Ravi Kumar	2301331549004
Rudra Srivastava	2301331549006
Sunitendra Srivastava	2301331549007
Tejas Kumar	2301331549009
Vishal Kumar	2301331549010

Submitted to

Mr. Krishna Arjun

(Trainer)

NOIDA INSTITUTE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institute of AKTU, Lucknow)

Affiliated to

D.R. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW

CONTENTS

Chapter 1- Introduction

1.1 Objective

The primary objective of the Email/SMS Spam Classifier project is to develop a robust and efficient machine learning model capable of accurately identifying and filtering out spam messages from legitimate ones in email and SMS communications. The project aims to achieve the following specific objectives:

1. **Spam Detection Accuracy**: Implement algorithms and techniques to train the classifier to accurately distinguish between spam and non-spam messages with a high level of precision and recall.

2. **Adaptability and Scalability**: Design the classifier to be adaptable to evolving spamming techniques and scalable to handle large volumes of incoming messages across diverse communication platforms.

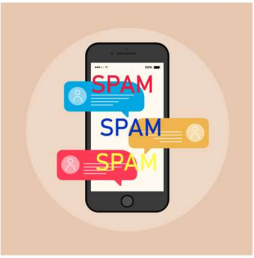
3. **User-Friendly Interface**: Develop an intuitive and user-friendly interface that allows users to interact with the classifier effectively, customize spam filtering preferences, and manage spam detection settings with ease.

4. **Real-Time Detection**: Enable real-time detection and classification of incoming messages to minimize the risk of users being exposed to spam and other malicious content.

5. **Continuous Improvement**: Implement mechanisms for continuous monitoring, feedback collection, and model refinement to enhance the classifier's performance over time and adapt to changing spamming patterns and trends.

6. **Integration and Compatibility**: Ensure seamless integration of the classifier into existing email and SMS platforms, frameworks, and workflows, making it compatible with a wide range of communication tools and services.

7. **Security Enhancement**: Enhance communication security by providing users with a reliable defines against phishing attacks, malware distribution, and other forms of cyber threats propagated through spam messages.



1.2Problem definition

The problem at hand revolves around the pervasive threat of spam messages infiltrating email and SMS communications, causing inconvenience, security risks, and potential harm to users. With the exponential growth of digital communication, distinguishing between genuine messages and spam has become increasingly challenging. Spam messages often include fraudulent schemes, phishing attempts, and malware distribution, posing significant risks to individuals, businesses, and organizations. These unwanted messages not only inundate inboxes but also undermine productivity and trust in digital communication channels. The problem lies in the need for a reliable and efficient system capable of automatically identifying and filtering out spam messages, thereby mitigating the impact of spam and enhancing communication security. The Email/SMS Spam Classifier project aims to address this problem by developing a machine learning-based solution that can accurately classify incoming messages as spam or legitimate, thereby providing users with a proactive defense mechanism against spam and fostering a safer digital communication environment.

1.3 Scope

The scope of the Email/SMS Spam Classifier project encompasses the development of a machine learning-based system designed to identify and classify spam messages in both email and SMS communications. The project includes the following components and considerations:

1. Data Collection and Preparation: The project involves gathering a diverse dataset comprising both spam and legitimate messages for training and validation purposes. This dataset may include textual content, sender information, metadata, and other relevant features extracted from email and SMS messages.
2. Feature Extraction and Engineering: The project entails extracting pertinent features from the collected data, such as text content, linguistic patterns, sender reputation, and message metadata. Feature engineering techniques may be employed to enhance the discriminative power of the classifier.
3. Machine Learning Model Development: The core of the project involves developing and training machine learning models, such as supervised learning algorithms (e.g., Support Vector Machines, Random Forests, or Neural Networks), to classify incoming messages as either spam or non-spam based on the extracted features.
4. Model Evaluation and Validation: The performance of the developed classifier is evaluated using metrics such as accuracy, precision, recall, and F1-score. Cross-validation techniques may be employed to assess the robustness and generalization ability of the model.

5. Integration with Communication Platforms: The classifier is designed to be integrated seamlessly with existing email and SMS communication platforms, enabling real-time spam detection and filtering. Compatibility with popular email clients, messaging apps, and communication APIs is considered within the project scope.
6. User Interface Development: A user-friendly interface is developed to facilitate user interaction with the classifier. This interface allows users to configure spam filtering preferences, manage whitelists and blacklists, and view classified messages conveniently.
7. Adaptability and Continuous Improvement: The classifier is designed to be adaptable to evolving spamming techniques and trends. Mechanisms for continuous monitoring, feedback collection, and model retraining are implemented to ensure the classifier's effectiveness and relevance over time.
8. Security and Privacy Considerations: The project incorporates security measures to safeguard user privacy and prevent sensitive information leakage. Data encryption, access control, and compliance with privacy regulations (e.g., GDPR) are integral aspects of the project's scope.

1.4 Definitions, Acronyms and Abbreviations

Definitions, Acronyms, and Abbreviations:

- NLP: Natural Language Processing
- SMS: Short Message Service
- API: Application Programming Interface
- GDPR: General Data Protection Regulation
- F1-score: Harmonic mean of precision and recall
- GUI: Graphical User Interface
- ML: Machine Learning
- SVM: Support Vector Machine
- Top of Form

1.5 Technologies to be used.

Technologies to be Used:

1. Python: Python will serve as the primary programming language for developing the Email/SMS Spam Classifier due to its versatility, rich ecosystem of libraries for

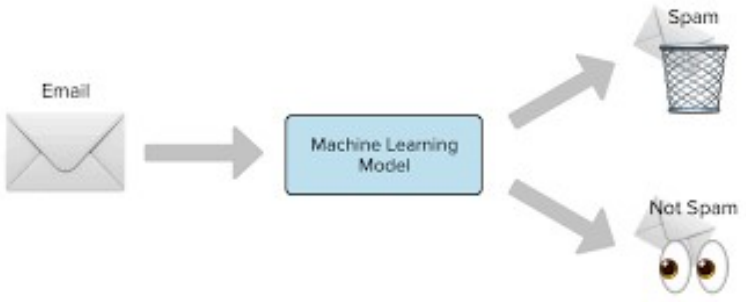
machine learning and natural language processing, and ease of integration with various platforms.

2. Machine Learning Libraries: Popular machine learning libraries such as Scikit-learn, TensorFlow, or PyTorch will be utilized for building and training the classifier model. These libraries offer a wide range of algorithms and tools for developing robust machine learning models.
3. Natural Language Processing (NLP) Libraries: NLP libraries like NLTK (Natural Language Toolkit) or spaCy will be employed for text processing tasks such as tokenization, stemming, and feature extraction from textual content in emails and SMS messages.
4. Data Visualization Tools: Tools like Matplotlib or Seaborn will be used for visualizing data distributions, model performance metrics, and other relevant insights during the exploratory data analysis and model evaluation stages.
5. Version Control: Version control systems such as Git will be utilized for collaborative development, code management, and tracking changes throughout the project's lifecycle.
6. Development Environment: Integrated Development Environments (IDEs) like PyCharm, Jupyter Notebooks, or Visual Studio Code will be used for coding, debugging, and testing the project components

Chapter 2- Software Requirement Specifications

2.1 Introduction

In today's interconnected world, where digital communication has become ubiquitous, the proliferation of spam emails and text messages has emerged as a pervasive nuisance and a significant security concern. The Email/SMS Spam Classifier project endeavours to tackle this pressing issue by leveraging the power of machine learning to develop an effective solution for filtering out unwanted and potentially harmful messages. The exponential growth of internet usage and the widespread adoption of smartphones have facilitated an unprecedented surge in the volume of electronic messages exchanged daily. However, amidst this deluge of communication, malicious actors exploit these channels to disseminate spam—unsolicited and often deceptive messages intended to defraud recipients, distribute malware, or engage in phishing attacks. Such activities not only compromise individual privacy and security but also pose substantial risks to businesses and organizations, leading to financial losses, reputational damage, and operational disruptions. Recognizing the critical need to address this menace, the Email/SMS Spam Classifier project seeks to develop an intelligent system capable of automatically identifying and segregating spam messages from legitimate ones. At its core, the project harnesses the principles of natural language processing (NLP) and machine learning, drawing upon vast datasets comprising both spam and non-spam messages to train and refine its classification model. The classification process entails extracting relevant features from incoming messages, such as textual content, sender information, metadata, and linguistic patterns. One of the distinguishing features of the Email/SMS Spam Classifier project is its adaptability and scalability. As spamming techniques evolve and adapt to circumvent conventional filtering methods, the classifier employs sophisticated algorithms capable of detecting emerging patterns and variations in spam messages. This adaptability ensures that the classifier remains effective in combating new and evolving forms of spam, safeguarding users against ever-changing threats in the digital landscape. Moreover, the project emphasizes user-centric design principles, prioritizing ease of integration and usability to facilitate seamless integration into existing communication platforms and workflows. By providing users with a transparent and intuitive interface for managing spam filters and preferences, the classifier enhances user control and empowers individuals and organizations to customize their spam detection mechanisms according to their specific needs and preferences. In conclusion, the Email/SMS Spam Classifier project represents a proactive and innovative approach to addressing the pervasive threat of spam in digital communication channels. By harnessing the capabilities of machine learning and natural language processing, the project aims to fortify communication security, mitigate the risks associated with spam, and foster a safer and more productive online environment for users worldwide.



2.1.1 Purpose

The purpose of the Email/SMS Spam Classifier project is to develop a robust and efficient system for automatically identifying and filtering out spam messages from legitimate ones in email and SMS communications. By leveraging machine learning algorithms and natural language processing techniques, the project aims to enhance communication security, minimize the risk of users being exposed to spam, and foster a safer digital communication environment.

2.1.2 Project Scope

The scope of the Email/SMS Spam Classifier project encompasses the following key aspects:

Data Collection and Preparation: Gathering a diverse dataset comprising both spam and legitimate messages for training and validation purposes.

- Feature Extraction and Engineering: Extracting pertinent features from the collected data, such as text content, linguistic patterns, sender reputation, and message metadata.
- Machine Learning Model Development: Developing and training machine learning models, such as supervised learning algorithms, to classify incoming messages as either spam or non-spam based on the extracted features.
- Model Evaluation and Validation: Evaluating the performance of the developed classifier using metrics such as accuracy, precision, recall, and F1-score.
- Integration with Communication Platforms: Integrating the classifier seamlessly with existing email and SMS communication platforms to enable real-time spam detection and filtering.
- User Interface Development: Developing a user-friendly interface that allows users to configure spam filtering preferences, manage whitelists and blacklists, and view classified messages conveniently.
- Adaptability and Continuous Improvement: Designing the classifier to be adaptable to evolving spamming techniques and trends, with mechanisms for continuous monitoring, feedback collection, and model retraining.

By addressing these aspects within the project scope, the Email/SMS Spam Classifier project aims to provide users with a reliable and effective solution for mitigating the impact of spam and enhancing communication security in digital environments.

2.2 Overall Description

The Email/SMS Spam Classifier project aims to develop a robust and efficient system for automatically detecting and filtering out spam messages from legitimate ones in email and SMS communications. Leveraging machine learning algorithms and natural language processing techniques, the project endeavours to enhance communication security and minimize the risk of users being exposed to spam.

2.2.1 Product Perspective

The Email/SMS Spam Classifier will function as an independent system that integrates seamlessly with existing email clients and SMS messaging applications. It will operate in real-time, analysing incoming messages and categorizing them as either spam or legitimate based on learned patterns and features.

2.2.2 Product Features

Key features of the Email/SMS Spam Classifier include:

- Automatic Spam Detection: The classifier automatically analyses incoming messages and identifies spam based on predefined criteria and learned patterns.
- User Interface: A user-friendly interface allows users to configure spam filtering preferences, view classified messages, and manage whitelists and blacklists.
- Adaptability: The classifier is designed to adapt to evolving spamming techniques and trends through continuous monitoring, feedback collection, and model retraining.
- Integration: Seamless integration with email clients and SMS messaging applications enables real-time spam detection and filtering without disrupting user workflows.

2.2.3 User Classes and Characteristics

Users of the Email/SMS Spam Classifier include individuals, businesses, and organizations who rely on email and SMS communications for personal and professional purposes. Users are concerned about the security and privacy risks associated with spam messages and seek an effective solution for mitigating these risks.

2.2.4 Operating Environment

The Email/SMS Spam Classifier will operate in various environments, including desktop computers, mobile devices, and server systems. It will support multiple operating systems, such as Windows, macOS, Linux, iOS, and Android.

2.2.5 Design and Implementation Constraints

Design and implementation constraints of the Email/SMS Spam Classifier include:

- Scalability: The system must be able to handle large volumes of incoming messages efficiently, without sacrificing performance.
- Resource Consumption: The classifier should be optimized to minimize resource consumption, such as memory and processing power, particularly in resource-constrained environments.

2.2.6 Assumptions and Dependencies

The Email/SMS Spam Classifier operates under the following assumptions:

- Availability of Training Data: Sufficient training data comprising both spam and legitimate messages is available for model development and evaluation.
- User Cooperation: Users will provide feedback on misclassified messages to facilitate model refinement and improvement over time.
- Compatibility: The classifier integrates seamlessly with popular email clients and SMS messaging applications, ensuring compatibility with diverse user environments.

2.3 External Interface Requirements

External Interface Requirements:

1. **Input Interface**: The system should accept input in the form of email messages or SMS texts for classification.
2. **Output Interface**: The system should provide classification results indicating whether the input message is spam or not.
3. **Integration Interface**: If integrated into an existing platform, the classifier should seamlessly interact with the user interface, APIs, or databases.
4. **User Interface**: A user-friendly interface may be required for users to interact with the classifier, such as a web-based form or mobile app.
5. **Compatibility:** The classifier should be compatible with common email clients or SMS applications to facilitate easy deployment and usage.

2.3.1 User Interfaces

Description: The user interface (UI) serves as the primary means for users to interact with the Email/SMS Spam Classifier system. It provides functionalities for configuring spam filtering preferences, viewing classified messages, managing whitelists/blacklists, and providing feedback.

Functional Requirements:

- Display options for configuring spam filtering preferences, including sensitivity levels, whitelist/blacklist rules, and notification settings.
- Present a dashboard or message inbox for users to view classified messages, with options to mark messages as spam or legitimate.
- Include mechanisms for reporting misclassified messages and providing feedback to improve the classifier's performance.
- Ensure responsiveness and intuitive navigation to enhance user experience across different devices and screen sizes.

Platform: The user interface should be accessible through web browsers, desktop applications, and mobile applications, ensuring compatibility with various devices and operating systems.

2.3.2 Hardware Interfaces

Description: Hardware interfaces specify the physical connections and requirements necessary for the Email/SMS Spam Classifier system to operate effectively.

Functional Requirements:

1. The system should be compatible with standard computing devices such as desktop computers, laptops, tablets, and smartphones.
2. Ensure compatibility with hardware components such as processors, memory, and storage to support efficient execution of the classifier algorithms.
3. No specific hardware dependencies beyond those required for running the chosen programming languages and libraries (e.g., Python runtime environment).

Platform: The Email/SMS Spam Classifier system should be deployable on any hardware platform that meets the specified computing requirements, without requiring specialized hardware configurations.

2.3.3 Software Interfaces

Description: Software interfaces define the interactions between the Email/SMS Spam Classifier system and other software components, including email clients, SMS messaging applications, databases, and external APIs.

Functional Requirements:

1. Support for integration with standard email protocols such as IMAP, SMTP, and POP3 for communication with email servers and clients.
2. Compatibility with APIs or SDKs provided by SMS messaging platforms for sending and receiving SMS messages and accessing message metadata.
3. Interface with relational or NoSQL databases for storing training data, model parameters, user preferences, and feedback.
4. Ensure compatibility with operating systems (e.g., Windows, macOS, Linux) and runtime environments required for deploying and running the classifier system.

Platform: The software interfaces should be compatible with a variety of email clients, SMS messaging platforms, database management systems, and operating systems commonly used in the target environment.

2.3.4 Communications Interfaces

Description: Communications interfaces enable real-time communication between the Email/SMS Spam Classifier system and external email servers, SMS gateways, and user devices.

Functional Requirements:

1. Implement protocols for secure and efficient communication with email servers (e.g., IMAP, SMTP) to fetch incoming messages for classification.
2. Support for sending and receiving SMS messages via SMS gateways or APIs provided by mobile network operators or third-party service providers.
3. Ensure reliable and low-latency communication channels to minimize message processing delays and ensure timely spam detection.
4. Implement encryption and authentication mechanisms to secure communication channels and protect sensitive user data during transmission.

Platform: The communications interfaces should be compatible with standard email protocols, SMS messaging APIs, and network protocols used for internet communication, ensuring interoperability across different platforms and network environments.

2.4 Other Non-functional Requirements:

2.4.1 Performance Requirements:

- **Speed:** The classifier should provide near real-time classification of incoming emails or SMS messages.
- **Scalability**: The system should be able to handle a large volume of incoming messages efficiently, especially during peak usage times.

- **Accuracy**: The classifier should achieve high accuracy in distinguishing between spam and legitimate messages.

2.4.2 Safety Requirements:

- **Data Integrity**: Ensure that the classifier does not alter the content of messages during the classification process.
- **User Privacy**: Protect user data and ensure that sensitive information within messages is not exposed or misused.

2.4.3 Security Requirements:

- **Data Security:** Implement measures to protect classified data from unauthorized access or tampering.
- **Authentication:** Authenticate users who access the classifier to prevent unauthorized usage.
- **Secure Communication:** Ensure that communication between the classifier and external systems is encrypted to prevent eavesdropping or data interception.

2.4.4 Software Quality Attributes:

- **Reliability:** The classifier should consistently provide accurate results without errors or failures.
- **Maintainability:** The system should be designed with clear code structure and documentation to facilitate easy maintenance and updates.
- **Usability:** Provide a user-friendly interface that is intuitive and easy to navigate for users interacting with the classifier.
- **Portability:** Ensure that the classifier can be deployed across different platforms and environments with minimal effort.

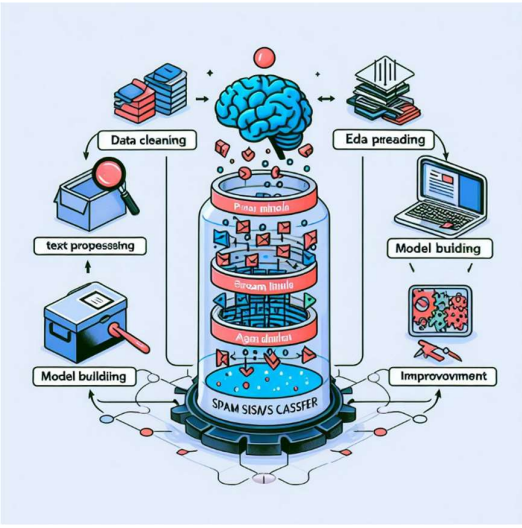
Chapter 3- System Design

3.1 Design Methodology:

- Data Cleaning: Utilize techniques such as removing duplicates, handling missing values, and standardizing text formats to ensure data quality and consistency.
- EDA (Exploratory Data Analysis): Employ statistical methods and data visualization techniques to gain insights into the characteristics of the dataset, such as the distribution of spam and non-spam messages, and identify patterns or trends.
- Text Preprocessing: Implement text preprocessing steps including tokenization, stop-word removal, and stemming or lemmatization to prepare the text data for model building.
- Model Building: Select appropriate machine learning or natural language processing models such as Naive Bayes, Support Vector Machines, or neural networks, and train them using the preprocessed data.
- Improving model performance: Fine-tune model hyperparameters, experiment with different feature engineering techniques, and consider ensemble methods or advanced algorithms to enhance the classification performance.

3.2 Software Development Model:

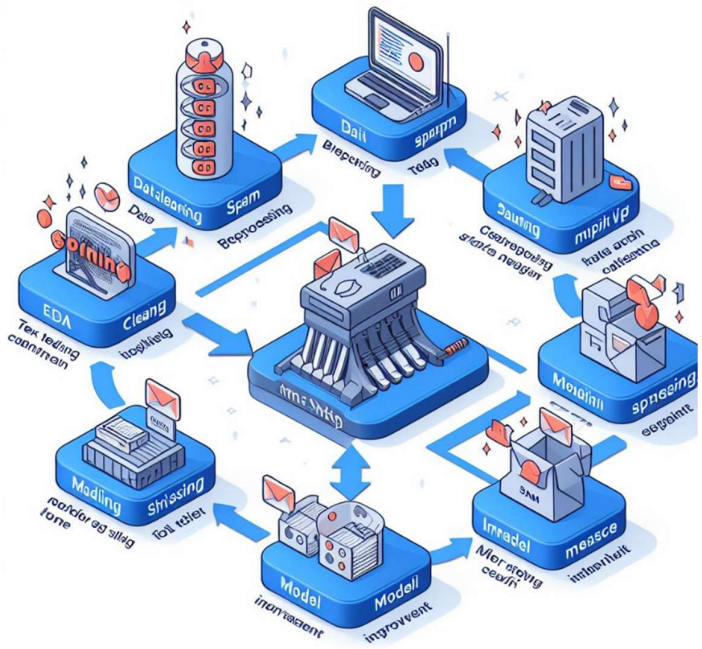
- Iterative Development: Adopt an iterative approach to software development, where each stage of the project (data cleaning, EDA, model building, etc.) is continuously refined based on feedback and insights gained from the previous stages.
- Agile Methodology: Embrace the Agile methodology to prioritize collaboration, adaptability, and customer feedback throughout the development process, ensuring rapid and incremental delivery of valuable software.



3.3 DFD's (Data Flow Diagrams):

- **Data Cleaning DFD**: Illustrate the flow of data through various cleaning processes such as removing duplicates, handling missing values, and standardizing formats.
- **EDA DFD**: Represent the flow of data during exploratory data analysis, showcasing the statistical analyses and visualizations performed to gain insights into the dataset.
- **Text Preprocessing DFD**: Depict the flow of text data through preprocessing steps like tokenization, stop-word removal, and stemming/lemmatization.
- **Model Building DFD**: Demonstrate the flow of data through the model building process, including feature extraction, model training, and evaluation.
- **Improving Model Performance DFD**: Outline the iterative process of optimizing model performance through techniques such as hyperparameter tuning, feature engineering, and model selection.
- **Creating the Project in PyCharm DFD**: Show the flow of development tasks and resources within the PyCharm IDE, including coding, debugging, version control, and project management activities.

These design elements provide a structured overview of the system development process, from data pre-processing to model building and software implementation, ensuring clarity and coherence in the project design and execution.



Chapter 4- System Implementation

4.1 Coding:

1. **Coding Environment**: Utilize Jupyter notebook IDE for coding the Email/SMS Spam Classifier project.

```
[4]: import numpy as np
     import pandas as pd

[5]: df = pd.read_csv('spam.csv', encoding='latin1')

[6]: df.sample(5)
```

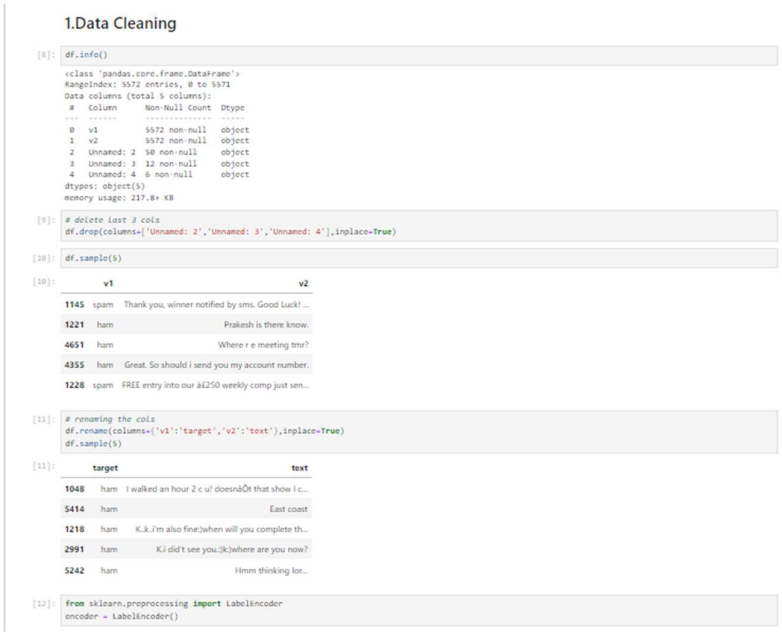
[6]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
5037	ham	Thanks for being there for me just to talk to ...	NaN	NaN	NaN
4334	ham	Now u sound like manky scouse boy steve,like! ...	NaN	NaN	NaN
2639	ham	Why she wants to talk to me	NaN	NaN	NaN
5077	spam	Do you want a New Nokia 3510i colour phone Del...	NaN	NaN	NaN
821	ham	On the road so cant txt	NaN	NaN	NaN

```
[7]: df.shape
```

```
[7]: (5572, 5)
```

2. **Implementation of Data Cleaning:** Write Python code to perform data cleaning tasks such as removing duplicates, handling missing values, and standardizing text formats.



1.Data Cleaning

```
[8]: df.info()
```

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

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

[10]: df.sample(5)
```

[10]:

	v1	v2
1145	spam	Thank you, winner notified by sms. Good Luck! ...
1221	ham	Prakesh is there know.
4651	ham	Where r e meeting tmr?
4355	ham	Great. So should i send you my account number.
1228	spam	FREE entry into our â£250 weekly comp just sen...

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

[11]:

	target	text
1048	ham	I walked an hour 2 c u! doesnâÒt that show I c...
5414	ham	East coast
1218	ham	K..k..i'm also fine:)when will you complete th...
2991	ham	K.i did't see you.:)k:)where are you now?
5242	ham	Hmm thinking lor...

```
[12]: from sklearn.preprocessing import LabelEncoder
      encoder = LabelEncoder()
```

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

[13]:

	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...

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

```
[14]: target    0
      text      0
      dtype: int64
```

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

```
[15]: 403
```

```
[16]: df = df.drop_duplicates(keep='first')
```

```
[17]: df.duplicated().sum()
```

```
[17]: 0
```

```
[18]: df.shape
```

```
[18]: (5169, 2)
```

3. **Implementation of Exploratory Data Analysis (EDA):** Develop Python scripts to conduct EDA, including statistical analysis and data visualization using libraries such as Pandas, NumPy, and Matplotlib/Seaborn.

2.EDA

```
[19]: df.head()
```

[19]:

	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...

```
[20]: df.value_counts()
```

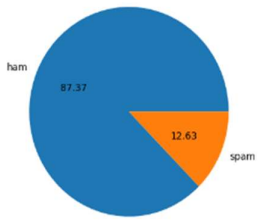
```
[20]: target  text
      0       &lt;#&gt;  in mca. But not conform.                                                                    1
              Thats cool. i liked your photos. You are very sexy!                                                  1
              That's good, because I need drugs                                                                    1
              That's fine, have him give me a call if he knows what he wants or has any questions                  1
              That's fine, I'll bitch at you about it later then                                                   1
                                                                                                                   ..
              I want to send something that can sell fast.  &lt;#&gt; k is not easy money.                          1
              I want to see your pretty pussy...                                                                   1
              I want to lick your pussy now...                                                                     1
              I want to go to perumbavoor                                                                          1
      1       we tried to contact you re your response to our offer of a new nokia fone and camcorder hit reply or call 08000930705 for delivery   1
      Name: count, Length: 5169, dtype: int64
```

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

```
[21]: target
      0    4516
      1     653
      Name: count, dtype: int64
```

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

```
[22]: ([<matplotlib.patches.Wedge at 0x28fd66c5100>,
        <matplotlib.patches.Wedge at 0x28fd66c4fb0>],
       [Text(-1.0144997251399075, 0.4251944351600247, 'ham'),
        Text(1.014499764949479, -0.4251943401757036, 'spam')],
       [Text(-0.5533634864399495, 0.2319242373601344, '87.37'),
        Text(0.5533635081542612, -0.2319241855038377, '12.63')])
```



[28]:
```
df['num_sentences'] = df['text'].apply(lambda x:len(nltk.sent_tokenize(x)))
```

[29]:
```
df.head()
```

[29]:

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

[30]:
```
df[['num_characters','num_words','num_sentences']].describe()
```

[30]:

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

[31]:
```
#ham
df[df['target'] == 0][['num_characters','num_words','num_sentences']].describe()
```

[31]:

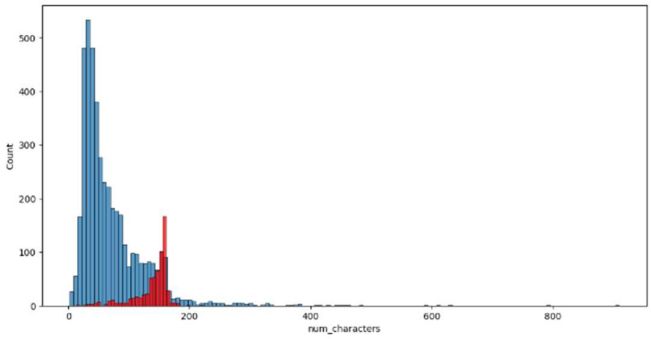
	num_characters	num_words	num_sentences
count	4516.000000	4516.000000	4516.000000
mean	70.459256	17.123782	1.820195
std	56.358207	13.493970	1.383657
min	2.000000	1.000000	1.000000
25%	34.000000	8.000000	1.000000
50%	52.000000	13.000000	1.000000
75%	90.000000	22.000000	2.000000
max	910.000000	220.000000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

[32]:
```
#spam
df[df['target'] == 1][['num_characters','num_words','num_sentences']].describe()
```

[33]:
```
import seaborn as sns
```

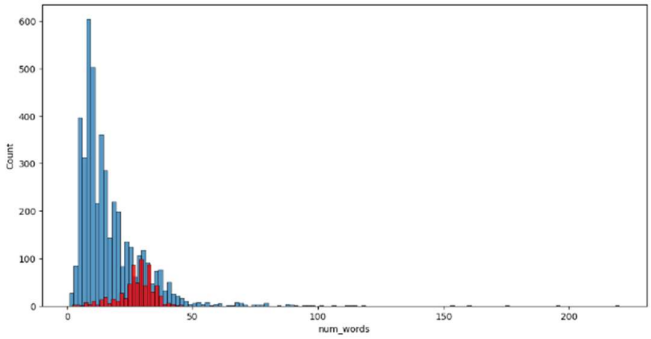
[34]:
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_characters'])
sns.histplot(df[df['target'] == 1]['num_characters'],color='red')
```

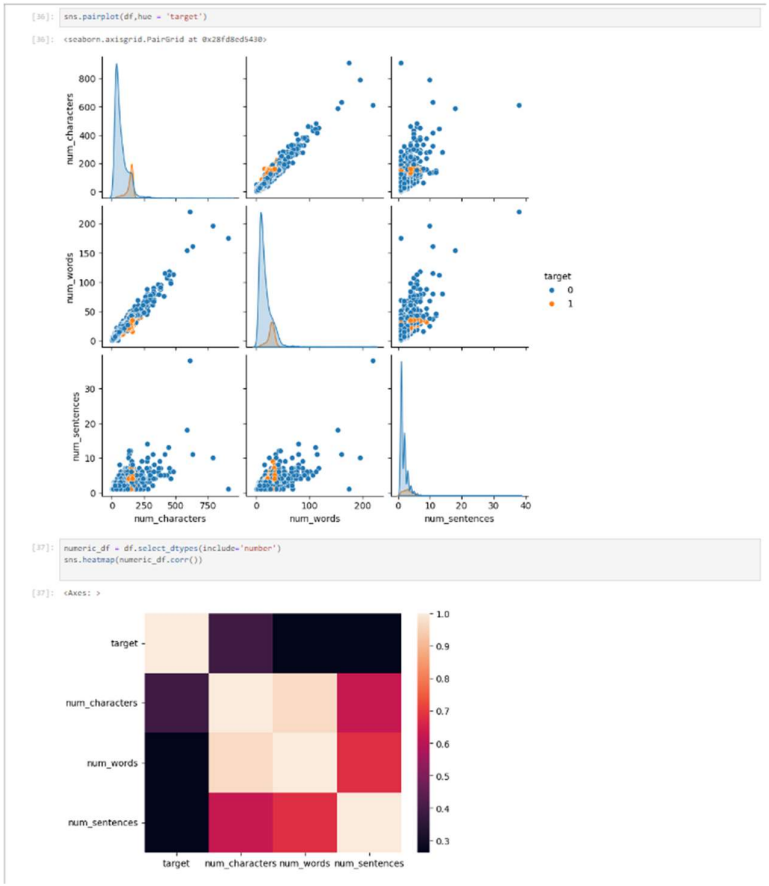
[34]:
```
<Axes: xlabel='num_characters', ylabel='Count'>
```



[35]:
```
plt.figure(figsize=(12,6))
sns.histplot(df[df['target'] == 0]['num_words'])
sns.histplot(df[df['target'] == 1]['num_words'],color='red')
```

[35]:
```
<Axes: xlabel='num_words', ylabel='Count'>
```





4. **Text Preprocessing Implementation:** Implement text preprocessing techniques like tokenization, stop-word removal, and stemming/lemmatization using appropriate libraries such as NLTK or spaCy.

[52]:
```
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

def transform_text(text):
    text = text.lower()

    text = nltk.word_tokenize(text)

    ps = PorterStemmer()

    y = []

    for i in text:
        if i.isalnum():
            y.append(ps.stem(i))

    y = [i for i in y if i not in stopwords.words('english') and i not in string.punctuation]

    return " ".join(y)

transformed_text = transform_text('I loved the YT lecture on machine learing who arout you ?')
print(transformed_text)
```

```
love yt lectur machin lear arout
```

[39]:
```
import string
string.punctuation
```

[39]:
```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

[53]:
```
transform_text('I loved the YT lecture on machine learing who arout you ?')
```

[53]:
```
'love yt lectur machin lear arout'
```

[49]:
```
df['text'][0]
```

[49]:
```
'Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...'
```

[50]:
```
from nltk.stem.porter import PorterStemmer
ps= PorterStemmer()
ps.stem('dancing')
```

[50]:
```
'danc'
```

[ ]:

[55]:
```
df['transformed_text'] = df['text'].apply(transform_text)
```

[56]:
```
df.head()
```

[56]:

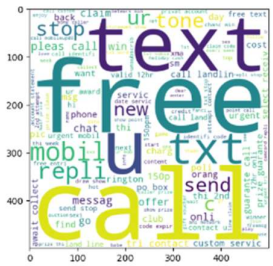
	target	text	num_characters	num_words	num_sentences	transformed_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail onli bugi n great ...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155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

[62]:
```
from wordcloud import WordCloud
wc = WordCloud(width=500,height=500,min_font_size=10,background_color='white')
```

[63]:
```
spam_wc = wc.generate(df[df['target'] == 1]['transformed_text'].str.cat(sep=" "))
```

[64]:
```
plt.imshow(spam_wc)
```

[64]:
```
<matplotlib.image.AxesImage at 0x28fdd2b54c0>
```



[65]:
```
ham_wc = wc.generate(df[df['target'] == 0]['transformed_text'].str.cat(sep=" "))
```

[66]:
```
plt.imshow(ham_wc)
```

[66]:
```
<matplotlib.image.AxesImage at 0x28fda4f8440>
```

[67]:
```
df.head()
```

[67]:

	target	text	num_characters	num_words	num_sentences	transformed_text
0	0	Go until jurong point, crazy.. Available only ...	111	24	2	go jurong point crazi avail onli bugi n great ...
1	0	Ok lar... Joking wif u oni...	29	8	2	ok lar joke wif u oni
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2	free entri 2 wkli comp win fa cup final tkt 21...

5. **Model Building Code:** Write Python code to select, train, and evaluate machine learning or natural language processing models for spam classification, incorporating libraries like scikit-learn or TensorFlow/Keras.

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

[152]: x = tfidf.fit_transform(df['transformed_text']).toarray()

[153]: x.shape

[153]: (5169, 3000)

[154]: y= df['target'].values

[155]: y

[155]: array([0, 0, 1, ..., 0, 0, 0])

[156]: from sklearn.model_selection import train_test_split

[157]: X_train,X_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=2)

[158]: from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
       from sklearn.metrics import accuracy_score,confusion_matrix,precision_score

[159]: gnb = GaussianNB()
       mnb = MultinomialNB()
       bnb = BernoulliNB()

[160]: 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))

       0.8713733075435203
       [[790 106]
        [ 27 111]]
       0.511520737327189
```

6. **Optimization and Refinement:** Refine the codebase iteratively based on performance evaluation and feedback, optimizing model hyperparameters and feature engineering strategies.

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

[160]: 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))

       0.8713733075435203
       [[790 106]
        [ 27 111]]
       0.511520737327189

[161]: 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))

       0.9729206963249516
       [[896   0]
        [ 28 110]]
       1.0

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

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

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

7. **Creating a Website Using Streamlit:** Utilize PyCharm IDE for coding the Email/SMS Spam Classifier project and create a web page.

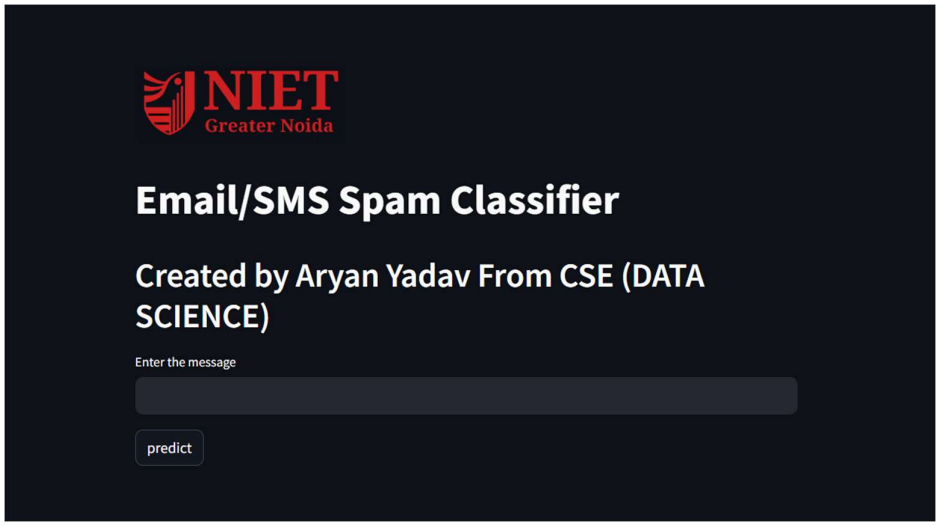
```
import streamlit as st
import pickle
import nltk
import string
import sklearn

from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

ps = PorterStemmer()


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

    text = nltk.word_tokenize(text)

    ps = PorterStemmer()

    y = []

    for i in text:
        if i.isalnum():
            y.append(ps.stem(i))

    y = [i for i in y if i not in stopwords.words('english') and i not in string.punctuation]

    return " ".join(y)

```

```
tfidf = pickle.load(open('vectorizer.pkl', 'rb'))
model = pickle.load(open('model.pkl', 'rb'))
st.markdown(
    "\n"
    "    <style>\n"
    "    .reportview-container {\n"
    "        background: #;\n"
    "    }\n"
    "    </style>\n"
    "    ",
    unsafe_allow_html=True
)
from PIL import Image
image = Image.open('C:\\Users\\Aryan\\Downloads\\niet 3.png')
col1, col2, col3 = st.columns(3)
with col1:
    st.image(image)
st.title("Email/SMS Spam Classifier")
st.header("Created by Aryan Yadav From CSE (DATA SCIENCE)")

input_sms = st.text_input("Enter the message")
if st.button('predict'):
```


```
tfidf = pickle.load(open('vectorizer.pkl', 'rb'))
model = pickle.load(open('model.pkl', 'rb'))
st.markdown(
    "\n"
    "    <style>\n"
    "    .reportview-container {\n"
    "        background: #;\n"
    "    }\n"
    "    </style>\n"
    "    ",
    unsafe_allow_html=True
)
from PIL import Image
image = Image.open('C:\\Users\\Aryan\\Downloads\\niet 3.png')
col1, col2, col3 = st.columns(3)
with col1:
    st.image(image)
st.title("Email/SMS Spam Classifier")
st.header("Created by Aryan Yadav From CSE (DATA SCIENCE)")

input_sms = st.text_input("Enter the message")
if st.button('predict'):
```


4.2 Testing:

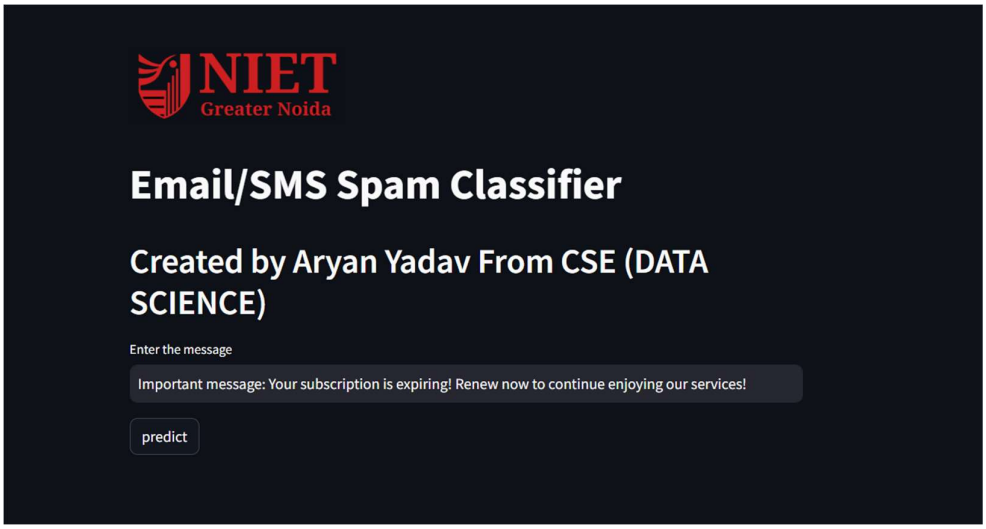
The working of our model involves passing a message or email through a machine learning classifier, which has been trained on a dataset of labelled spam and non-spam messages. Upon clicking the predict button, the model analyses the input message using natural language processing techniques and makes a prediction based on learned patterns and features. If the model identifies the message as spam, it outputs "spam"; otherwise, it outputs "not spam." This prediction is made in real-time, providing users with immediate feedback on the classification of the input message.



To test your Email/SMS Spam Classifier project, you can follow these steps:

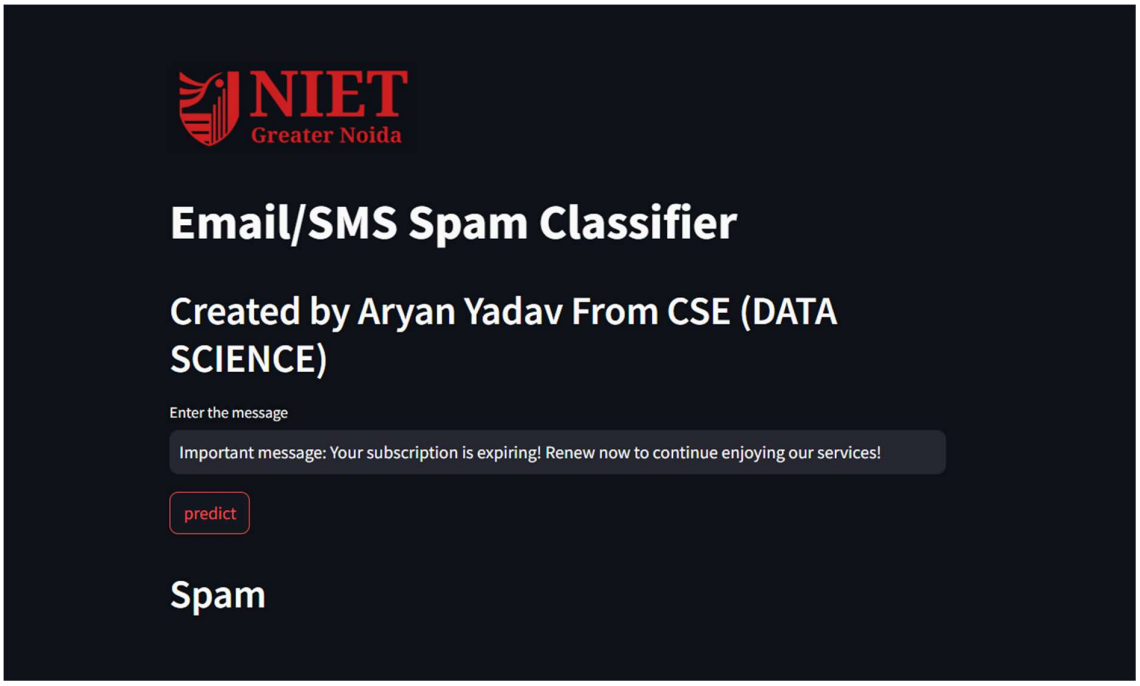
1. **Input Test Data**: Enter various sample email or SMS messages into the "Enter the message" field to evaluate how well the classifier identifies spam.

   When I pass this SMS that is spam. (Important message: Your subscription is expiring! Renew now to continue enjoying our services!)



2. **Analyze Output**: Observe the prediction results to determine if they are accurate.

- When I analyze the prediction results it was correct


NIET
Greater Noida
Email/SMS Spam Classifier
Created by Aryan Yadav From CSE (DATA SCIENCE)
Enter the message
Important message: Your subscription is expiring! Renew now to continue enjoying our services!
predict
Spam

Chapter 5 - Conclusions and Future Scope:

5.1 Conclusion:

- Summary of Achievements: Summarize the key achievements and outcomes of the Email/SMS Spam Classifier project.
- Effectiveness of the Classifier: Discuss the effectiveness of the implemented classifier in accurately distinguishing between spam and non-spam messages.
- Challenges and Limitations: Address any challenges or limitations encountered during the project implementation and how they were mitigated.
- Impact and Significance: Highlight the significance of the classifier in addressing the issue of email and SMS spam, and its potential benefits for users and organizations.

5.2 Future Scope:

- **Enhancements to the Classifier:** Propose potential enhancements or refinements to further improve the performance and functionality of the classifier, such as: Experimenting with advanced machine learning algorithms or deep learning architectures. Incorporating additional features or data sources for better classification accuracy. Implementing real-time monitoring and updating mechanisms to adapt to evolving spam patterns.
- **Integration with Email/SMS Platforms**: Explore opportunities to integrate the classifier into email clients, messaging apps, or spam filtering services to provide users with seamless spam detection capabilities.
- **User Feedback and Iterative Development:** Emphasize the importance of gathering user feedback to iteratively enhance the classifier based on user needs and preferences.
- **Deployment and Scalability:** Discuss plans for deploying the classifier in production environments and scaling it to handle large volumes of messages effectively.
- **Research Directions:** Identify potential research directions related to email and SMS spam detection, such as analysing emerging spam tactics, exploring novel feature engineering techniques, or investigating the impact of contextual information on classification accuracy.
- **Collaborations and Partnerships**: Explore opportunities for collaborations with industry partners or academic institutions to further develop and validate the classifier's performance in real-world scenarios.

These future scope suggestions outline the potential avenues for extending the Email/SMS Spam Classifier project and its continued evolution to meet evolving user needs and technological advancements.

Chapter 6 - References:

6.1 Books:

- Mitchell, T. (1997). Machine Learning. McGraw-Hill Education.
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.

6.2 URLs:

- Brownlee, J. (2019). How to Develop a Naive Bayes Classifier from Scratch in Python. https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/
- NLTK Documentation. (n.d.). Natural Language Toolkit. Available: https://www.nltk.org/
- Scikit-learn Documentation. (n.d.). Machine Learning in Python. Available: https://scikit-learn.org/stable/
- Machine learning IBM : https://www.ibm.com/topics/machine-learning
- Project Idea: https://medium.com/codex/creating-a-spam-email-classifier-using-python-245754443d71