Building a Smarter AI-Powered Spam classifier

Phase - III

Team Name : Proj_208227_Team_1

Development Part - 1

INTRODUCTION:

○ Short Message Service (SMS) has become a primary mode ofcommunication, connecting people, businesses, and services in real-time. However, this convenience also brings with it the menace of SMS spam, unwanted and often malicious messages that disrupt our daily lives and pose security risks. To combat this growing issue, the development of an SMS spam classifier powered by advanced AI and machine learning techniques is of paramount importance.

○ An SMS spam classifier is a sophisticated tool that employscutting-edge technology to sift through the ever-increasing volume of text messages, distinguishing between legitimate communication and unwanted spam. This classifier plays a crucial role in enhancing the mobile experience byensuring that our SMS inboxes are free from clutter, annoyance, and potential security threats.

○ To train a smart SMS spam classifier, it's essential to accumulate a diverse dataset containing both spam and legitimate messages. This dataset serves as the foundation for the AI model's training.

○ Relevant features from SMS messages, such as text content, sender information, timestamps, and even message length, are extracted to beused in training the classifier.

○ To adapt to evolving SMS spam tactics, the classifier must bedesigned for continuous learning. Regular updates with new data ensure its relevance and effectiveness over time.

○ Implementing feedback loops where users can report SMS spamhelps the system improve its accuracy and adapt to new forms of spam.The classifier should be designed to handle a high volume of SMS messages in real- time, ensuring that legitimate messages are not delayed.

Given data set:

	type	text
0	ham	Hope you are having a good week. Just checking in
1	ham	K..give back my thanks.
2	ham	Am also doing in cbe only. But have to pay.
3	spam	complimentary 4 STAR Ibiza Holiday or £10,000 ...
4	spam	okmail: Dear Dave this is your final notice to...
...	...	...
5554	ham	You are a great role model. You are giving so ...
5555	ham	Awesome, I remember the last time we got someb...
5556	spam	If you don't, your prize will go to another cu...
5557	spam	SMS. ac JSco: Energy is high, but u may not kn...
5558	ham	Shall call now dear having food

5559 rows × 2 columns

Necessary Steps to Follow:

1.IMPORT LIBRARIES:

Start by importing the necessary libraries

Program:

In[1]:

```
import numpy import pandas as pd
 import pandas as pd
import matplotlib.pyplot as plt
 import seaborn as sns
 import re
```

2. Load Your Dataset:

Load your dataset into a Pandas DataFrame. You can typically find spam or non spam message datasets in CSV format, but you can adapt this code to other formats as needed.

Program:

In[2]:

```
df=pd.read_csv("C:/Users/ELCOT/Downloads/sms_spam.csv")
```

3. Explarotary Data Analysis(EDA)

Perform EDA to understand your data better. This includes get the length of each message, exploring the data's statistics, and visualizing imbalanced data

In[3]:

```
# let's get length of each message

import pandas as pd

df=pd.read_csv("C:/Users/ELCOT/Downloads/sms_spam.csv")

df['text_length'] = df['text'].apply(lambda x: len(x.split(" ")))

df.head()
```

out[3]:

	type	text	text_length
0	ham	Hope you are having a good week. Just checking in	10
1	ham	K..give back my thanks.	4
2	ham	Am also doing in cbe only. But have to pay.	10
3	spam	complimentary 4 STAR Ibiza Holiday or £10,000 ...	21
4	spam	okmail: Dear Dave this is your final notice to...	27

In[4]:

df.describe()

out[4]:

	type	text
count	5559	5559
unique	2	5156
top	ham	Sorry, I'll call later
freq	4812	30

Visualizing Imbalanced Data:

In[5]:

df['type'].value_counts()

out[5]:

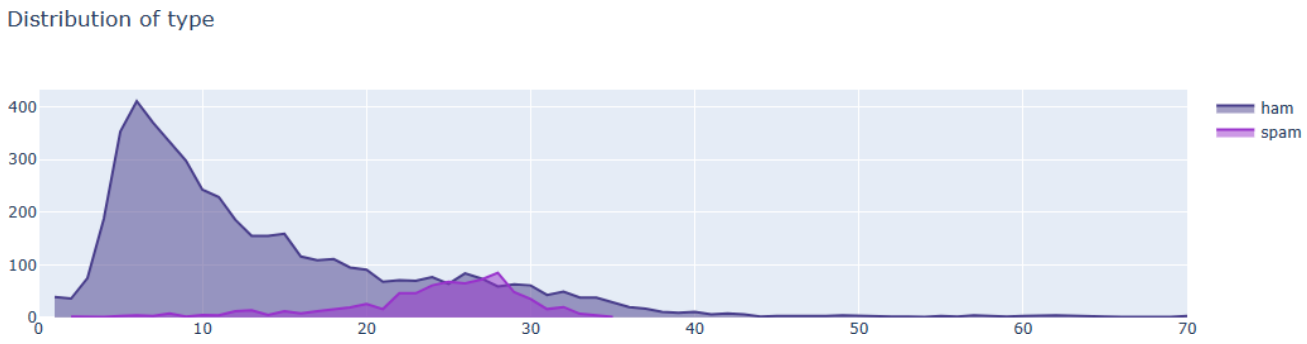
```
type
ham     4812
spam     747
Name: count, dtype: int64
```

In[6]:

```
from plotly import graph_objs as go
Ham_len = df[df['type']=='ham']['text_length'].value_counts().sort_index()
Spam_len= df[df['type']=='spam']['text_length'].value_counts().sort_index()
fig = go.Figure()
fig.add_trace(go.Scatter(
   x = Ham_len.index ,
```

```
    y = Ham_len.values ,
    name= 'ham' ,
    fill= 'tozeroy',
    marker_color = 'darkslateblue',))
fig.add_trace(go.Scatter(
x = Spam_len.index ,
    y = Spam_len.values ,
    name = 'spam' ,
    fill = 'tozeroy',
    marker_color = 'darkorchid' ,))
fig.update_layout( title = 'Distribution of type')
fig.update_xaxes(range =[0,70])
Fig.show()
```

Out[6]:



Importance of Loading and Preprocessing dataset:

Loading and preprocessing the dataset is crucial in building an accurate SMS spam classifier. It helps in preparing the data for analysis by cleaning, transforming, and organizing it. This ensures that the classifier receives high-quality data, leading to better predictions.

Challenges involved in loading and preprocessing a sms spam classifier dataset:

There are a number of challenges involved in loading and preprocessing a sms spam classifier dataset, including:

❖ Text Cleaning:

Text data often contains special characters, punctuation, and non-standard spellings. Preprocessing, such as tokenization, stopword removal, and stemming, can be challenging to perform correctly.

❖ Stopword Lists and Stemming Rules:

Choosing the appropriate list of stopwords and stemming rules for your specific dataset and language can be challenging. Different languages and contexts may require different lists and rules.

❖ Feature Extraction:

Selecting the right technique for converting text data into numerical features, such as TF-IDF or word embeddings, can be challenging. The choice of feature extraction method can impact the model's performance significantly.

❖ Split the Data:

Split the dataset into training and testing the sets.

1. Loading the dataset:

- ✓ Loading the dataset using machine learning is the process of bringing the data into the machine learning environment so that it can be used to train and evaluate a model.
- ✓ Thespecific steps involved in loading the dataset will vary depending on the machine learning library or framework that is being used. However, there are some general steps that are common to most machine learning frameworks:

a.Identify the dataset:

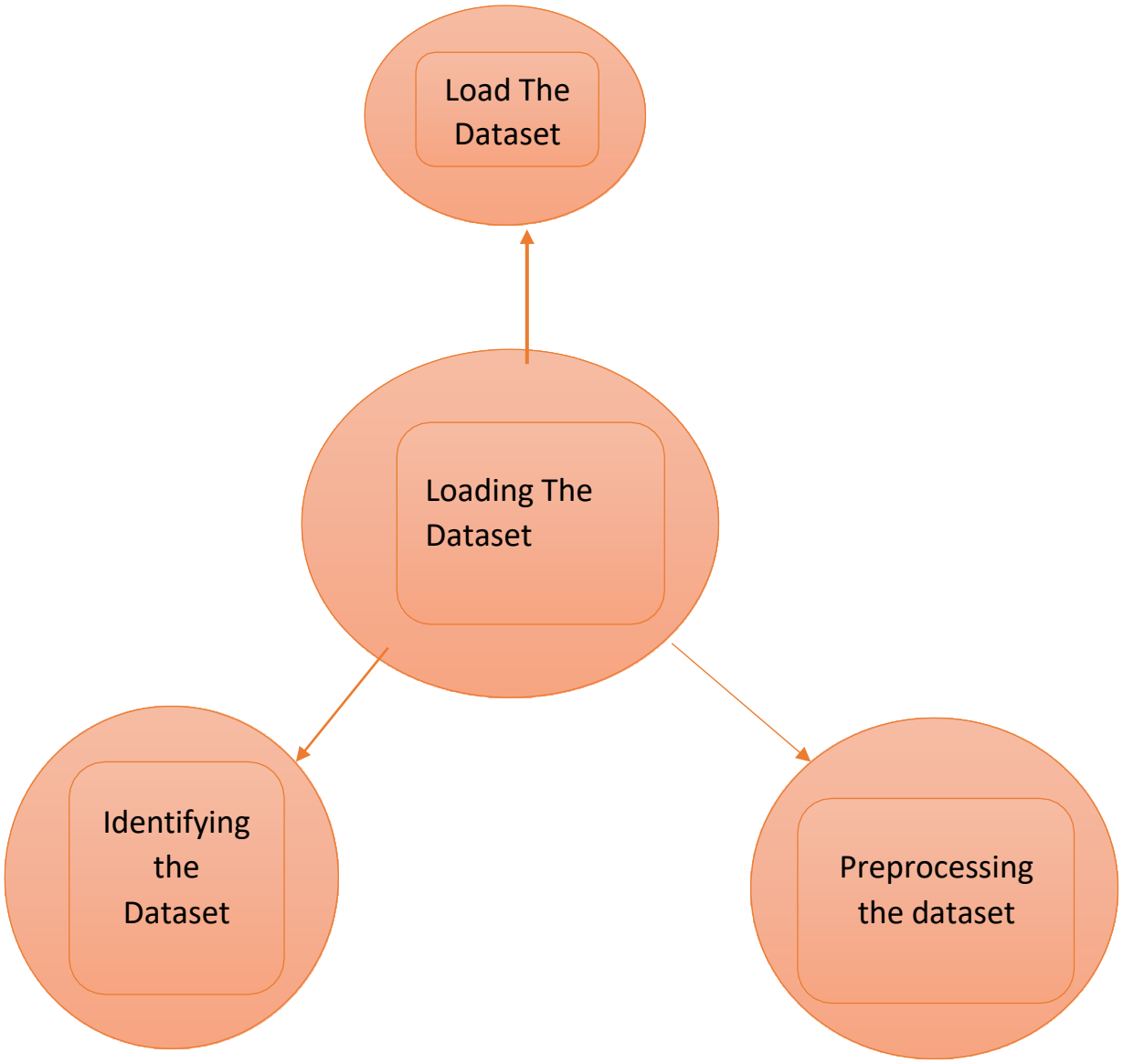
The first step is to identify the dataset that you want to load. This dataset may be stored in a local file, in a database, or in a cloud storage service.

b.Load the dataset:

Once you have identified the dataset, you need to load it into the machine learning environment. This may involve using a built-in function in the machine learning library, or it may involve writing your own code.

c.Preprocess the dataset:

Once the dataset is loaded into the machine learning environment, you may need to preprocess it before you can start training and evaluating your model. This may involve cleaning the data, transforming the data into a suitable format, and splitting the data into training and test sets.



Program:

```
import numpy import pandas as pd
 import pandas as pd
import matplotlib.pyplot as plt
 import seaborn as sns
 import re
from nltk.corpus import stopwords
from plotly import graph_objs as go
 from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer,TfidfTransformer
from sklearn.model_selection import train_test_split
 from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score,classification_report, confusion_matrix
```

Loading the Dataset:

```
df=pd.read_csv("C:/Users/ELCOT/Downloads/sms_spam.csv")
```

Data Exploration: Dataset:

	type	text
0	ham	Hope you are having a good week. Just checking in
1	ham	K..give back my thanks.
2	ham	Am also doing in cbe only. But have to pay.
3	spam	complimentary 4 STAR Ibiza Holiday or £10,000 ...
4	spam	okmail: Dear Dave this is your final notice to...
...	...	...
5554	ham	You are a great role model. You are giving so ...
5555	ham	Awesome, I remember the last time we got someb...
5556	spam	If you don't, your prize will go to another cu...
5557	spam	SMS. ac JSco: Energy is high, but u may not kn...
5558	ham	Shall call now dear having food

5559 rows × 2 columns

2. Preprocessing the Dataset:

Preprocessing the data is a crucial step when building a SMS spam classifier. It involves cleaning and transforming the text data to make it suitable for machine learning algorithms. Here's a step-by-step guide on how to preprocess the data for a SMS spam classifier:

1. Text Cleaning:

Clean the text data by performing the following tasks:

- Remove any special characters, punctuation, and numbers.
- Convert text to lowercase for uniformity.
- Tokenize the text (split it into words).

In[8]:

```
def clean_text(text):
    '''Do lowercase, remove text in square brackets,links, punctuation
    and words containing numbers.'''
    text = str(text).lower()
    text = re.sub(r'\[.*?\]', '', text)
    text = re.sub(r'https?://\S+|www\.\S+', '', text)
    text = re.sub(r'<.*?>+', '', text)
    text = re.sub(r'[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub(r'\n', '', text)
    text = re.sub(r'\w*\d\w*', '', text)
    return text
```

In[9]:

```
def preprocessing(text):
cleaned_text = clean_text(text)
# remove stopwords
cleaned_text = ' '.join(word for word in cleaned_text.split(' ') if word not in stop_words)
 # do stem method
 cleaned_text = ' '.join(stemmer.stem(word) for word in cleaned_text.split(' '))
 return cleaned_text
```

In[10]:

```
import re

import string

df['Cleaned_Message'] = df['text'].apply(preprocessing)

# let's show new length after process

df['New_length'] = df['text'].apply(lambda x: len(x.split(' ')))

df.head()
```

out[10]:

	type	text	Cleaned_Message	New_length
0	ham	Hope you are having a good week. Just checking in	hope good week check	10
1	ham	K..give back my thanks.	kgive back thank	4
2	ham	Am also doing in cbe only. But have to pay.	also cbe pay	10
3	spam	complimentary 4 STAR Ibiza Holiday or £10,000 ...	complimentari star ibiza holiday £ cash need ...	21
4	spam	okmail: Dear Dave this is your final notice to...	okmail dear dave final notic collect tenerif ...	27

2.Stopword Removal:

Remove common stopwords (e.g., 'the', 'and', 'in') because they don 't carry much information for classification.

In[11]:

```
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
df['text'] = df['text'].apply(lambda x: [word for word in x if word not in stop_words]
)
df.head()
```

out[11]:

	type	text
0	ham	[h, p, e, , u, , r, e, , h, v, n, g, , , ...
1	ham	[k, ., ., g, v, e, , b, c, k, , , h, n, k, .]
2	ham	[, l, , n, g, , n, , c, b, e, , n, l, ., ...
3	spam	[c, p, l, e, n, r, , 4, , r, , b, z, , h, ...
4	spam	[k, l, :, , e, r, , v, e, , h, , , u, r, ...

3.Feature Extraction: Convert the text data into numerical features using techniques like TF-IDF (Term Frequency-Inverse Document Frequency).

Do TF-IDF Method

In[11]:

```
from sklearn.preprocessing import LabelEncoder

from sklearn.model_selection import train_test_split

Encoder = LabelEncoder()

y_label  = Encoder.fit_transform(df['text'])

train_data= df['Cleaned_Message']

X_train ,X_test , y_train , y_test = train_test_split(train_data , y_label , test_size=0.2 , random_state=42)
```

In[12]:

```
from sklearn.feature_extraction.text import TfidfTransformer , CountVectorizer

TF_model = TfidfTransformer()

Vec_model= CountVectorizer()

Vec_model.fit(X_train)

X_train_vec = Vec_model.transform(X_train)

X_test_vec = Vec_model.transform(X_test)

TF_model.fit(X_train_vec)

X_train_tfidf = TF_model.transform(X_train_vec)

X_test_tfidf = TF_model.transform(X_test_vec)

X_train_tfidf
```

Out[12]:

```
<4447x5994 sparse matrix of type '<class 'numpy.float64'>'
        with 34854 stored elements in Compressed Sparse Row format>
```

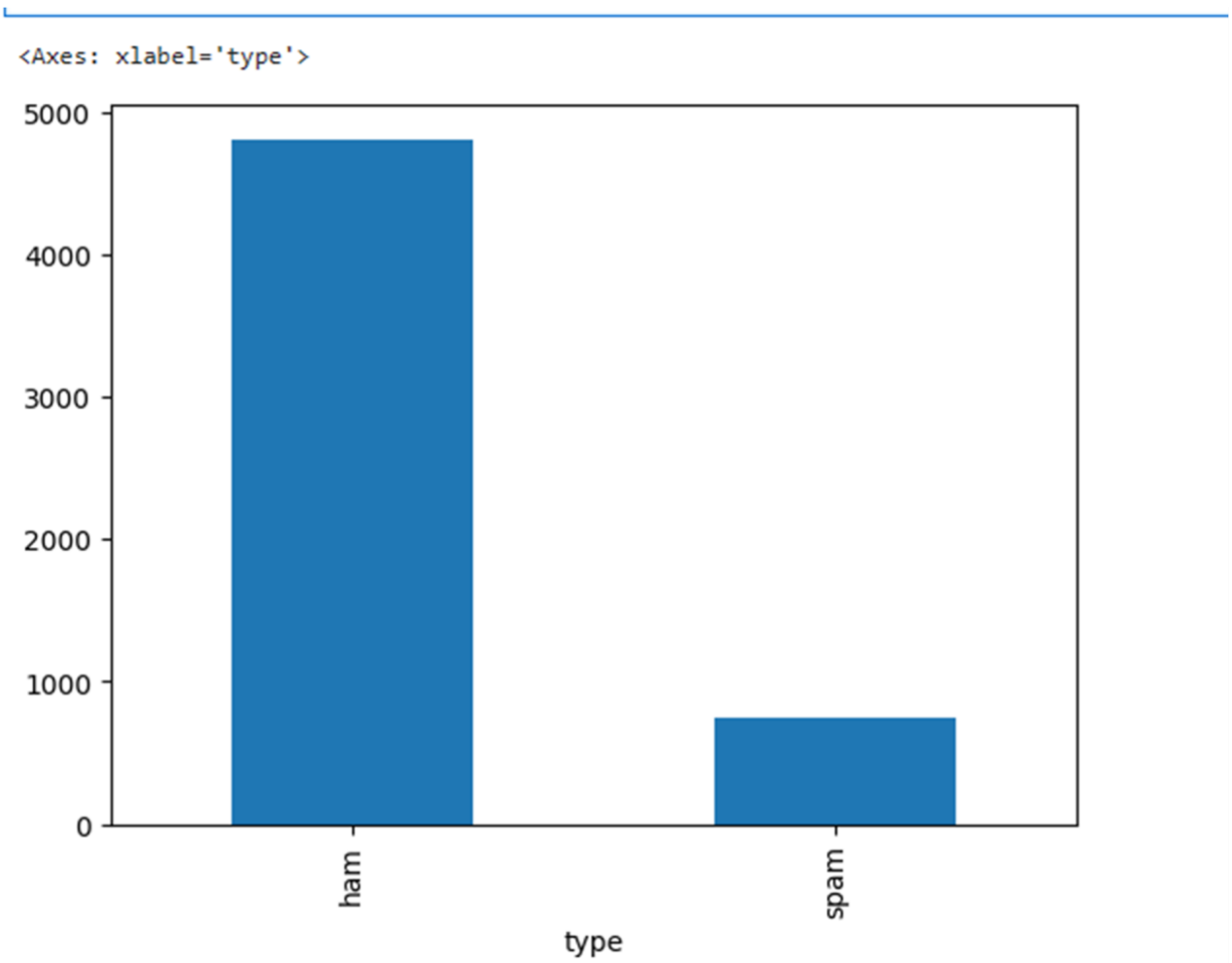
Creating Histogram

A histogram is a graphical representation of the distribution of a dataset. It can be used to visualize the distribution of ham and spam words in a dataset.

In[13]:

df.type.value_counts().plot.bar()

out[13]:



Create a column named length:

In[10]:

df['spam'] = df['type'].map({'spam': 1, 'ham': 0}).astype(int)

In[11]:

df['length'] = df['message'].apply(len)

In[12]:

df.head(10)

Out[12]:

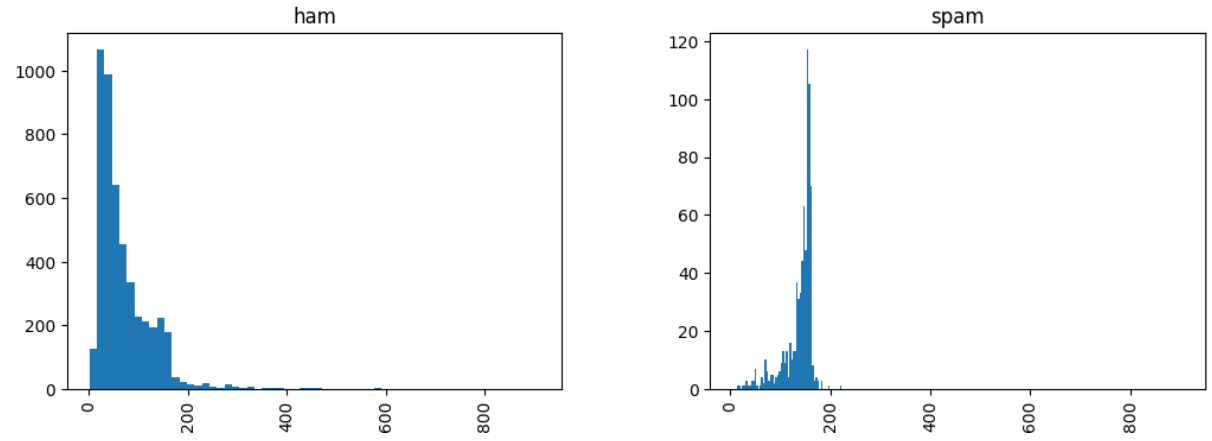
	type	text	spam	length
0	ham	Hope you are having a good week. Just checking in	0	49
1	ham	K..give back my thanks.	0	23
2	ham	Am also doing in cbe only. But have to pay.	0	43
3	spam	complimentary 4 STAR Ibiza Holiday or £10,000 ...	1	149
4	spam	okmail: Dear Dave this is your final notice to...	1	161
5	ham	Aiya we discuss later lar... Pick u up at 4 is...	0	50
6	ham	Are you this much buzy	0	22
7	ham	Please ask mummy to call father	0	31
8	spam	Marvel Mobile Play the official Ultimate Spide...	1	160
9	ham	fyi I'm at usf now, swing by the room whenever	0	46

In[14]:

```
df.hist(column='length',by='type',bins=60,figsize=(12,4));

plt.xlim(-40,950);
```

out[14]:



Word cloud:

A word cloud is a visual representation of text data that displays the most frequently occurring words in a dataset. It can be used to identify the most common words in spam messages and help improve the accuracy of a spam classifier. You can use Python's word cloud library to create a word cloud .

In[15]:

```
data_ham = df[df['spam'] == 0].copy()

data_spam = df[df['spam'] == 1].copy()
```

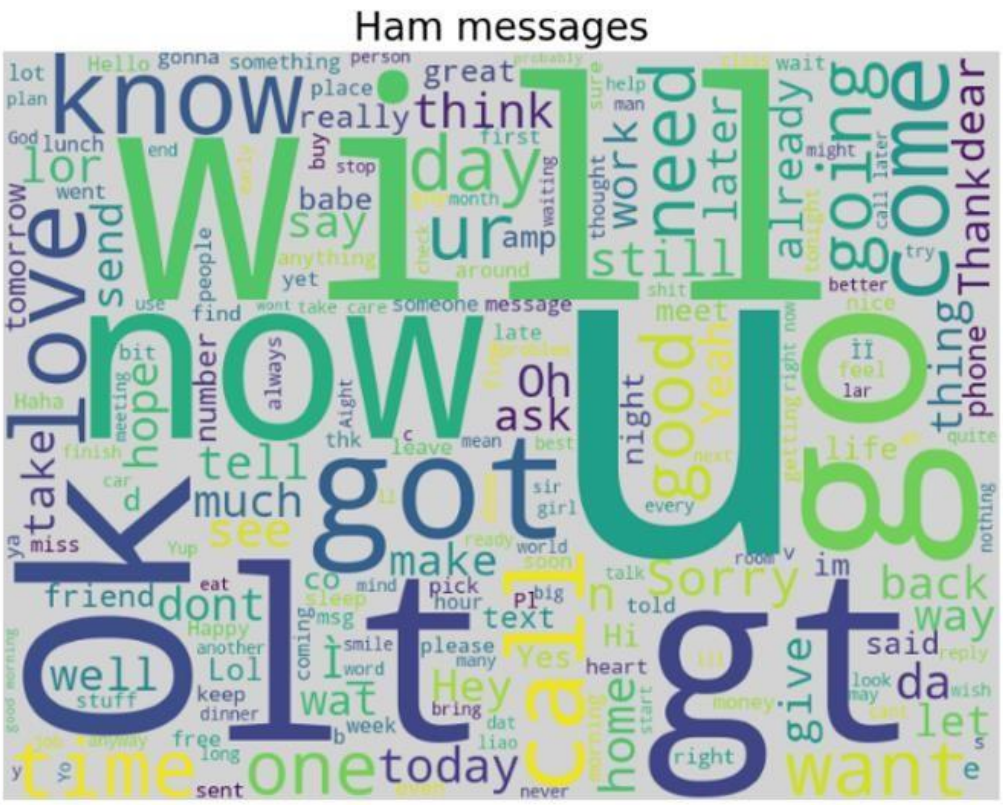
In[16]:

```
def show_wordcloud(data_spam_or_ham, title):
 text = '
'.join(data_spam_or_ham['message'].astype(str).tolist())
   stopwords = set(wordcloud.STOPWORDS)
  fig_wordcloud =
wordcloud.WordCloud(stopwords=stopwords,background_color='lightgrey',
                colormap='viridis', width=800, height=600).generate(text)
   plt.figure(figsize=(10,7),  frameon=True)
   plt.imshow(fig_wordcloud)
   plt.axis('off')
 plt.title(title, fontsize=20 )
 plt.show()
```

In[17]:

```
show_wordcloud(data_ham, "Ham messages")
```

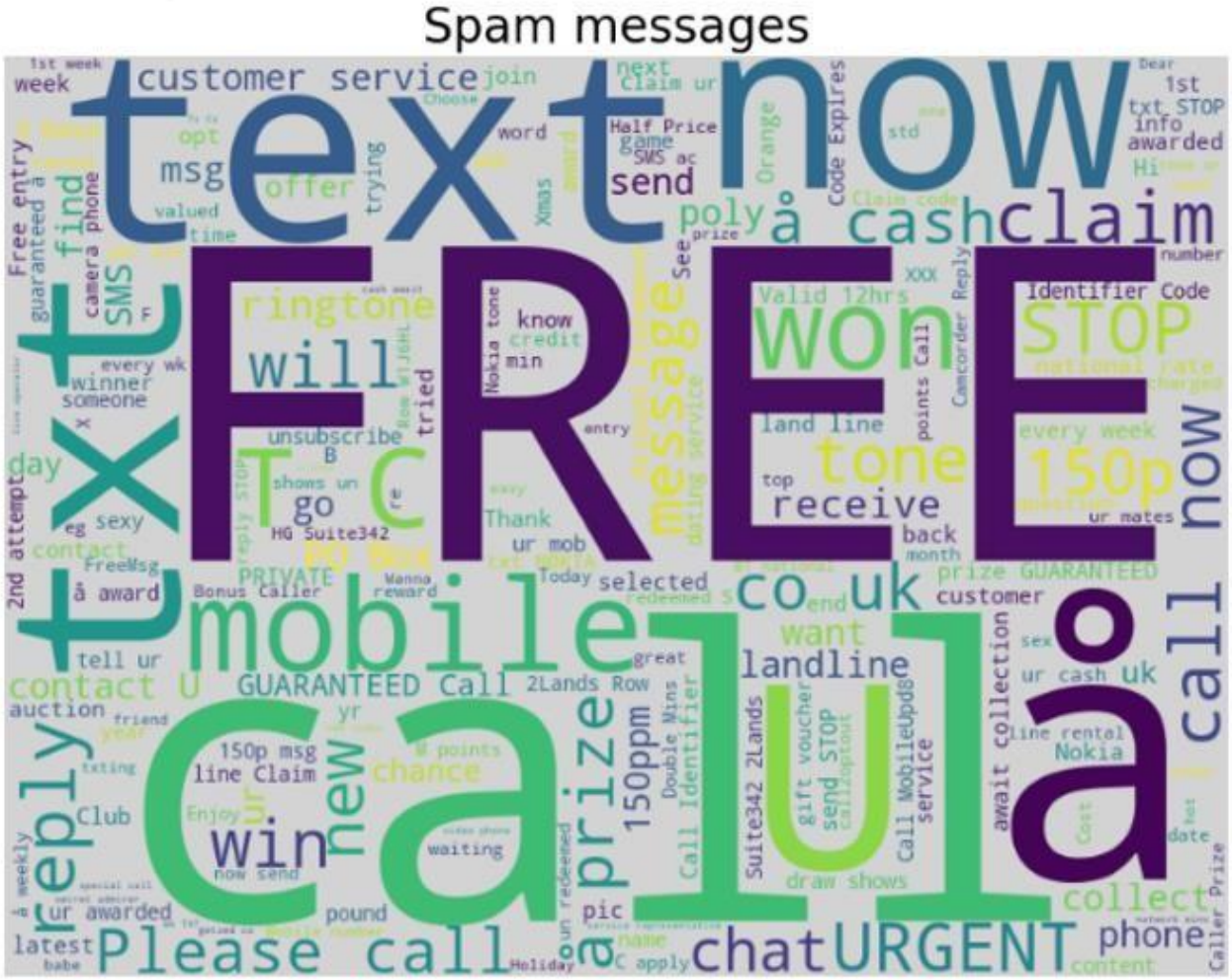
Out[17]:



In[18]:

show_wordcloud(data_spam, "Spam messages")

Out[18]:



In[19]:

plt.figure(figsize=(12,6))

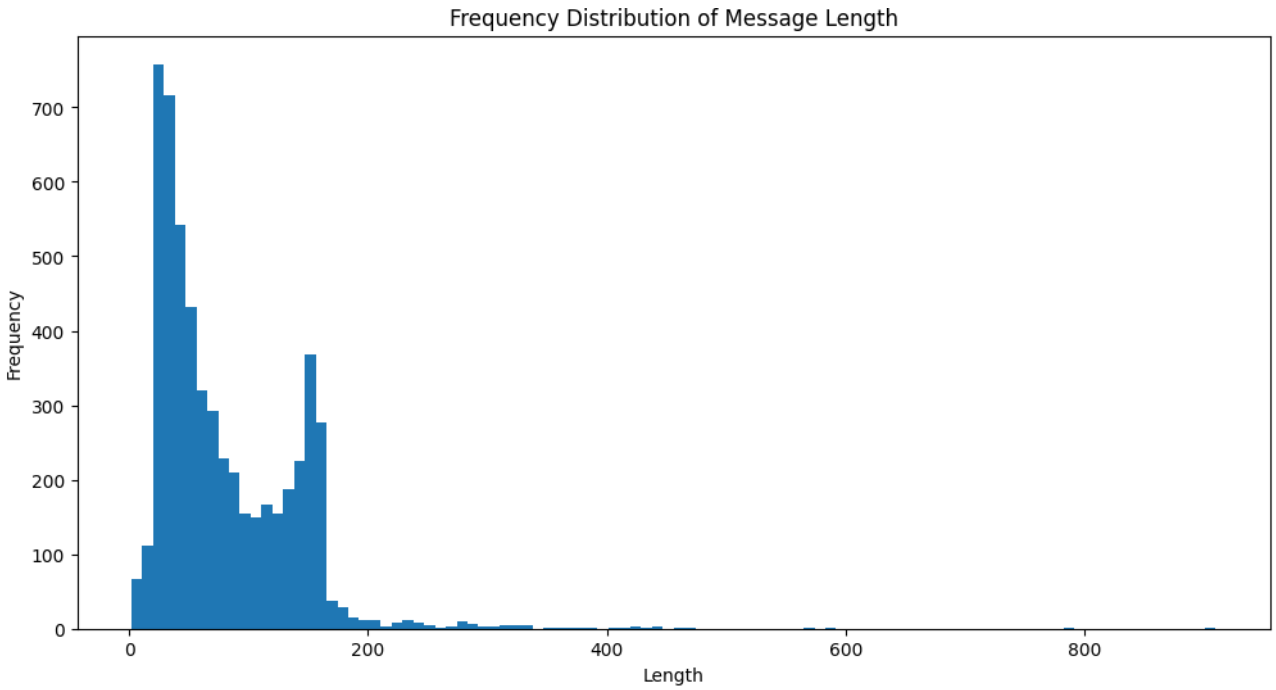
df['length'].plot(bins=100, kind='hist') # with 100 length bins (100 length intervals)

plt.title("Frequency Distribution of Message Length")

plt.xlabel("Length")

plt.ylabel("Frequency")

out[19]:



Tokenization:

In[20]:

```
nltk.download('punkt')

df["No_of_Characters"] = df["text"].apply(len)

df["No_of_Words"]=df.apply(lambda row:
nltk.word_tokenize(row["text"]), axis=1).apply(len)

df["No_of_sentence"]=df.apply(lambda row:
nltk.sent_tokenize(row["text"]), axis=1).apply(len)

df.describe().T
```

out[20]:

	count	mean	std	min	25%	50%	75%	max
spam	5559.0	0.134377	0.341087	0.0	0.0	0.0	0.0	1.0
length	5559.0	79.781436	59.105497	2.0	35.0	61.0	121.0	910.0
No_of_Characters	5559.0	79.781436	59.105497	2.0	35.0	61.0	121.0	910.0
No_of_Words	5559.0	18.400432	13.182246	1.0	9.0	15.0	27.0	196.0
No_of_sentence	5559.0	2.006296	1.540083	1.0	1.0	2.0	3.0	38.0

Pair plot:

In[21]:

Import seaborn as sns

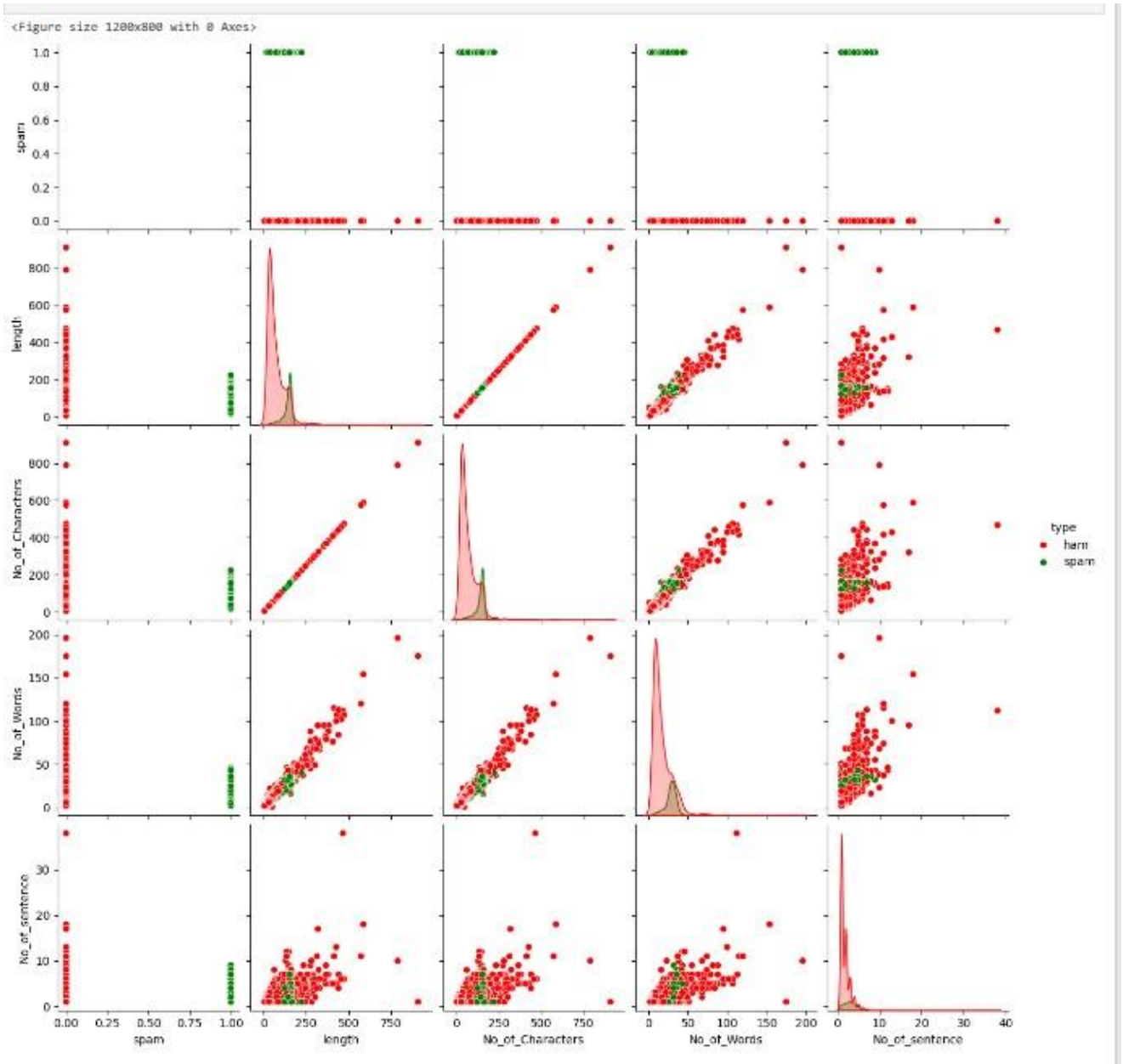
plt.figure(figsize=(12,8))

cols=["red","green"]

fg = sns.pairplot(data=df, hue="type",palette=cols)

plt.show(fg)

out[21]:



Now, you have preprocessed the SMS data, cleaned the text, removed stopwords, and converted it into numerical features using TF-IDF. You can use this preprocessed data to train and evaluate a SMS spam classifier

Conclusion:

In conclusion, loading and preprocessing a dataset for an SMS spam classifier is a crucial and sometimes challenging process that lays the foundation for building an effective and accurate spam detection model.

- SMS spam datasets are often imbalanced, which require careful handling to prevent model bias.
- Cleaning and preprocessing text data, including tokenization, stopword removal, and stemming, are essential for effective feature extraction.

Choosing the right feature extraction method, such as TF-IDF or word embeddings, can significantly impact the model's performance.

- Successfully addressing these challenges through a systematic and well-thought-out approach is critical for building a robust SMS spam classifier.