PHASE – 2

(Import the dataset and perform data cleaning & data analysis)

NAME	**HEMANATH M D**
REGISTER NUMBER	**61772221L01**
NM_ID	**aut2221l301**
PROJECT TITLE	**BUILDING A SMARTER AI-POWERED SPAM CLASSIFIER**

IMPORT DATASET :

Importing a dataset means bringing external data into our programming environment, typically in a format that allows us to work with and analyse that data using our programming language.

- **Choose a Programming Language:** we typically use a programming language to work with datasets. Python is a popular choice for data analysis.
- **Select a Data Format:** Datasets can come in various formats, such as CSV, Excel, JSON, SQL databases, or more specialized formats like HDF5. we need to know the format of your dataset to choose the right method for importing it. Here we using CSV dataset format.
- **Install Necessary Libraries:** Depending on the format of our dataset, we might need to install specific libraries to handle it.
- **Import Required Libraries:** Import the necessary libraries into our code using import statements.

```
In [5]: # Importing necessary libraries
        import pandas as pd
        import seaborn as sns

In [6]: # Read the file
        df = pd.read_csv("spam.csv")

In [7]: df
Out[7]:
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN
...	...	...	...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...	NaN	NaN	NaN
5568	ham	Will Ì_ b going to esplanade fr home?	NaN	NaN	NaN
5569	ham	Pity, * was in mood for that. So...any other s...	NaN	NaN	NaN
5570	ham	The guy did some bitching but I acted like i'd...	NaN	NaN	NaN
5571	ham	Rofl. Its true to its name	NaN	NaN	NaN

5572 rows × 5 columns

- “import pandas as pd” and “import seaborn as sns” are lines that bring in two helpful tools for working with data.
- “df = pd.read_csv("spam.csv")” reads a file named "spam.csv" (dataset) and stores it in a table-like structure called a DataFrame.
- “print(df)” shows us what's inside the table. It's like looking at the first few rows of your data so we can understand what's in there.
- So, this code is all about getting data from a file, organizing it in a table, and taking a peek at the data to see what's in it.

DATA CLEANING :

The purpose of data cleaning is to make our data more reliable and usable:

- **Remove Errors:** Fix wrong or unrealistic values in the dataset.
- **Fill the Missing Information:** If some data is missing, we can estimate or find the missing values so our data is complete.
- **Make it Consistent:** If we have the same information in different formats, like "New York" and "NY," we can make it all look the same.
- **Remove Duplicates:** Sometimes, the same information appears more than once. Cleaning helps us to remove duplicates so we don't count things twice.

By doing this, data cleaning ensures that the data we work with is accurate, trustworthy, and ready for analysis.

The dataset shall be cleaned through the following processes:

- Checking the number of columns.
- Changing misspelt column names to the correct names.
- Checking for missing values.
- Checking for Duplicate values.

```
In [8]: # Checking for number of columns
        df.columns

Out[8]: Index(['v1', 'v2', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], dtype='object')

In [9]: # Rename the column "v1" to "Mail_Type"
        df.rename(columns={"v1":"Mail_Type"},inplace=True)

In [10]: # Rename the column "v2" to "Mail_Message"
         df.rename(columns={"v2":"Mail_Message"},inplace=True)

In [11]: df.columns

Out[11]: Index(['Mail_Type', 'Mail_Message', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], dtype='object')

In [12]: # To check for the missing values
         df.isnull().sum()

Out[12]: Mail_Type          0
         Mail_Message       0
         Unnamed: 2      5522
         Unnamed: 3      5560
         Unnamed: 4      5566
         dtype: int64
```

- 'df.columns' is like looking at the labels on a set of drawers. It shows us the names of the different parts (columns) in our dataset. In this case, there are columns named "Mail_Type," "Mail_Message," and some others that don't have clear names.
- df.rename(columns={"v1":"Mail_Type"}, inplace=True) and df.rename(columns={"v2":"Mail_Message"}, inplace=True) are like relabeling our drawers so that they have more meaningful names. Instead of "v1" and "v2," now we have "Mail_Type" and "Mail_Message."
- After renaming, df.columns shows us the updated labels of our drawers.
- df.isnull().sum() is like checking if there's anything missing in our drawers. It tells us how many missing (empty) values there are in each drawer. In this case, "Mail_Type" and "Mail_Message" have no missing values, but the other drawers have lots of missing stuff.

```
In [18]: # Drop the columns 'unnamed: 2' 'unnamed: 3 'unnamed: 4'
         df.drop(columns=['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],inplace=True)
```

```
In [15]: df.columns

Out[15]: Index(['Mail_Type', 'Mail_Message'], dtype='object')

In [16]: df.isnull().sum()

Out[16]: Mail_Type       0
         Mail_Message    0
         dtype: int64

In [19]: # checking for information concerning the dataset
         df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Mail_Type     5572 non-null   object
 1   Mail_Message  5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

- **df.drop(columns=['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], inplace=True):** Think of our dataset as a collection of drawers. This line of code removes (or drops) some drawers labeled 'Unnamed: 2,' 'Unnamed: 3,' and 'Unnamed: 4.' It's like getting rid of drawers we don't need.

- **df.columns:** This line shows us the labels on our remaining drawers. After the drop, we have only two drawers left, 'Mail_Type' and 'Mail_Message.'

- **df.isnull().sum():** It's like counting how many empty spaces are inside your remaining drawers. In this case, both 'Mail_Type' and 'Mail_Message' have no missing (empty) values.

- **df.info():** This provides detailed information about our dataset, like its size and what's inside. It tells us that we have a DataFrame with two columns, 'Mail_Type' and 'Mail_Message,' and no missing values. It also tells us the memory usage (how much space it takes in your computer's memory).

```
In [28]: # Check for the unique values in the column "Mail_Type"
         df1["Mail_Type"].unique()

Out[28]: array(['ham', 'spam'], dtype=object)
```

```
In [29]: # checking for duplicate values
         df1.duplicated()

Out[29]: 0       False
         1       False
         2       False
         3       False
         4       False
                 ...
         5567    False
         5568    False
         5569    False
         5570    False
         5571    False
         Length: 5572, dtype: bool
```

```
In [17]: # Drop the duplicate values
         df2 = df1.drop_duplicates()
```

```
In [18]: # to confirm the change
         df2.duplicated().sum()

Out[18]: 0

In [19]: # Saving the new dataset into a csv file
         df2.to_csv("cleaned_spam.csv",index=False)
```

- **df1["Mail_Type"].unique():** This line is like looking at a specific drawer labeled "Mail_Type" in our data. It shows us all the different things (unique values) we find in that drawer. In this case, there are two unique values: 'ham' and 'spam.'
- **df1.duplicated():** It's like checking if any information in our data is repeated. It tells us for each row in our data if it's a duplicate (a repeat) or not. If it's not a duplicate, it says 'False,' and if it is a duplicate, it says 'True.'
- **df2 = df1.drop_duplicates():** we're making a new collection of drawers (a new dataset) by removing the duplicates. It's like taking out repeated pieces of information from our data.
- **df2.duplicated().sum():** This checks if there are any duplicates left in our new dataset (the new collection of drawers). If the sum is zero, it means there are no duplicates left.
- **df2.to_csv("cleaned_spam.csv", index=False):** It's like taking our cleaned data and putting it in a new file named "cleaned_spam.csv." The "index=False" part just means we don't want to include the row numbers when saving the data.

DATA PREPROCESSING:

In [8]:
```python
import re

def clean_text(text):
    text = text.lower()
    text = re.sub(r"[^a-zA-Z0-9]", " ", text)
    return text

df['Mail_Message'] = df['Mail_Message'].apply(clean_text)
```

In [9]:
```python
df
```

Out[9]:

	Mail_Type	Mail_Message
0	ham	go until jurong point crazy available only ...
1	ham	ok lar joking wif u oni
2	spam	free entry in 2 a wkly comp to win fa cup fina...
3	ham	u dun say so early hor u c already then say
4	ham	nah i don t think he goes to usf he lives aro...
...	...	...
5164	spam	this is the 2nd time we have tried 2 contact u...
5165	ham	will b going to esplanade fr home
5166	ham	pity was in mood for that so any other s...
5167	ham	the guy did some bitching but i acted like i d...
5168	ham	rofl its true to its name

In [10]:
```
!pip install nltk
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: nltk in c:\programdata\anaconda3\lib\site-packages (3.8.1)
Requirement already satisfied: click in c:\programdata\anaconda3\lib\site-packages (from nltk) (8.0.4)
Requirement already satisfied: joblib in c:\programdata\anaconda3\lib\site-packages (from nltk) (1.2.0)
Requirement already satisfied: regex>=2021.8.3 in c:\programdata\anaconda3\lib\site-packages (from nltk) (2022.7.9)
Requirement already satisfied: tqdm in c:\programdata\anaconda3\lib\site-packages (from nltk) (4.65.0)
Requirement already satisfied: colorama in c:\programdata\anaconda3\lib\site-packages (from click->nltk) (0.4.6)
```

In [11]:
```python
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Raman\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
```

Out[11]: True

```python
import nltk
from nltk.corpus import stopwords
import string
from nltk.stem.porter import PorterStemmer
nltk.download('stopwords')
ps = PorterStemmer()

def transfrom_text(text):
    # Tokenization
    text = nltk.word_tokenize(text)
    y=[]

    # Removing Special character
    for i in text:
        if i.isalnum():
            y.append(i)

    text = y[:]
    y.clear()

    # Removing stopwords and punctuations
    for i in text:
        if i not in stopwords.words('english') and i not in string.punctuation:
            y.append(i)
    text = y[:]
    y.clear()

    # Stemming
    for i in text:
        y.append(ps.stem(i))
    text = y[:]
    y.clear()
    return " ".join(text)


df['transformed_text'] = df['Mail_Message'].apply(transfrom_text)
```

In [25]:
```python
df
```

Out[25]:

	Mail_Type	Mail_Message	transformed_text
0	ham	go until jurong point crazy available only ...	go jurong point crazi avail bugi n great world...
1	ham	ok lar joking wif u oni	ok lar joke wif u oni
2	spam	free entry in 2 a wkly comp to win fa cup fina...	free entri 2 wkli comp win fa cup final tkt 21...
3	ham	u dun say so early hor u c already then say	u dun say earli hor u c alreadi say
4	ham	nah i don t think he goes to usf he lives aro...	nah think goe usf live around though
...	...	...	...
5164	spam	this is the 2nd time we have tried 2 contact u...	2nd time tri 2 contact u u 750 pound prize 2 c...
5165	ham	will b going to esplanade fr home	b go esplanad fr home
5166	ham	pity was in mood for that so any other s...	piti mood suggest
5167	ham	the guy did some bitching but i acted like i d...	guy bitch act like interest buy someth els nex...
5168	ham	rofl its true to its name	rofl true name

5169 rows × 3 columns

In [26]:
```python
df.to_csv('preprocessed_dataset.csv')
```

1. Cleaning the Text:

- import re brings in a library for text manipulation.
- clean_text(text) is a function that does several things to make text consistent:

  text = text.lower() makes all text lowercase.
- re.sub(r"[^a-zA-Z0-9]", " ", text) removes anything that isn't a letter or number and replaces it with a space.
- df['Mail_Message'].apply(clean_text) applies this cleaning to each message in the 'Mail_Message' column in our dataset, making the text consistent and easier to work with.
- print(df) displays the cleaned dataset.

2. Natural Language Processing (NLP) Preprocessing:

- !pip install nltk installs the Natural Language Toolkit (NLTK) library.
- import nltk and nltk.download('punkt') load and download necessary data for NLTK.
- from nltk.corpus import stopwords and nltk.download('stopwords') import a list of common words (stopwords) in English and download related data.
- ps = PorterStemmer() creates a stemmer, which reduces words to their base form.

3. Text Transformation:

- transfrom_text(text) is a function that processes text:
- text = nltk.word_tokenize(text) splits the text into words.
- A loop filters out special characters and non-alphanumeric characters.
- Another loop removes stopwords (common words like "the" and "and") and punctuation.
- Yet another loop performs stemming to reduce words to their base form (e.g., "running" becomes "run").
- The final result is a cleaned and transformed text.
- df['Mail_Message'].apply(transfrom_text) applies this text transformation to each message in the 'Mail_Message' column.
- print(df) shows the updated dataset with the transformed text.

4. Saving the Preprocessed Data:

- df.to_csv('preprocessed_dataset.csv') saves the preprocessed dataset to a new CSV file called 'preprocessed_dataset.csv.' This file contains the cleaned and transformed text for further analysis.

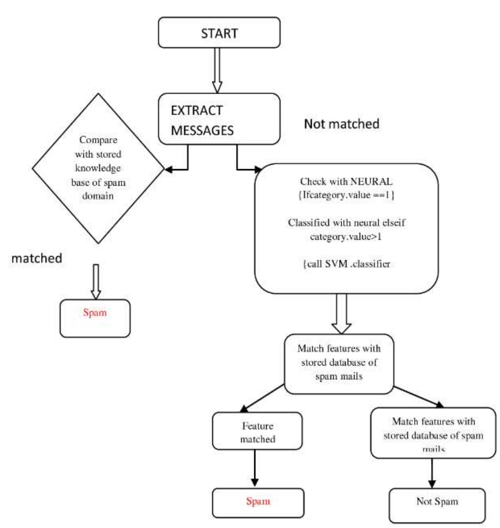
PDATA ANALYSIS:

Model : Neural Networks

Data analysis using neural networks is like teaching a computer to recognize and understand patterns in data, just like how we humans recognize things.

- **Data Input:** Imagine we have a lot of emails - some are spam (unwanted junk emails) and some are not. Each email is like a piece of data.
- **Learning:** we use a neural network to teach our computer to recognize patterns in these emails. For instance, it learns that spam emails often contain words like "buy now," "free," or "win."
- **Recognition:** After learning from the data, the computer can look at a new email and decide whether it's spam or not. If it sees lots of words like "buy now" and "free," it might say, "This is likely spam."
- **Improvement:** The more emails we give it, the better it gets at recognizing spam. It learns to spot more patterns and becomes better at avoiding false alarms (marking non-spam as spam) or missing real spam.
- **Applications:** we can use this spam classifier to automatically filter out spam emails from our inbox, keeping our email safe and organized.

The purpose of this data analysis with a neural network is to save our time by automatically identifying and filtering out annoying and potentially harmful spam emails. It's like having a smart assistant who's really good at spotting and getting rid of junk mail for us.



The dataset shall be analysed through the following processes:

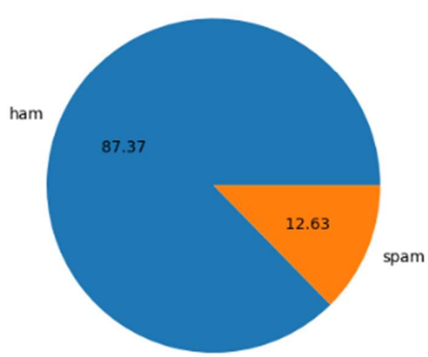
- Exploratory Data Analysis (EDA).
- Length of the Mail_Message
  - Num_characters
  - Num_words
  - Num_sentence
- Displaying Histogram.
- Word Frequency Analysis.
- Message length Analysis.

Exploratory Data Analysis (EDA):

Checking ham and spam count:

```
In [8]: df['Mail_Type'].value_counts()

Out[8]: Mail_Type
        ham     4516
        spam     653
        Name: count, dtype: int64
```

```
In [17]: # EDA
         !pip install matplotlib
         plt.pie(df['Mail_Type'].value_counts(), labels = ['ham','spam'], autopct = "% 0.2f")

         Defaulting to user installation because normal site-packages is not writeable
         Requirement already satisfied: matplotlib in c:\programdata\anaconda3\lib\site-packages (3.7.2)
         Requirement already satisfied: contourpy>=1.0.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib) (1.0.5)
         Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\site-packages (from matplotlib) (0.11.0)
         Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\anaconda3\lib\site-packages (from matplotlib) (4.25.0)
         Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib) (1.4.4)
         Requirement already satisfied: numpy>=1.20 in c:\programdata\anaconda3\lib\site-packages (from matplotlib) (1.24.3)
         Requirement already satisfied: packaging>=20.0 in c:\programdata\anaconda3\lib\site-packages (from matplotlib) (23.1)
         Requirement already satisfied: pillow>=6.2.0 in c:\programdata\anaconda3\lib\site-packages (from matplotlib) (9.4.0)
         Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib) (3.0.9)
         Requirement already satisfied: python-dateutil>=2.7 in c:\programdata\anaconda3\lib\site-packages (from matplotlib) (2.8.2)
         Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib)
         (1.16.0)

Out[17]: ([<matplotlib.patches.Wedge at 0x1e7ff4c69d0>,
           <matplotlib.patches.Wedge at 0x1e7ff508d90>],
          [Text(-1.0144997251399075, 0.4251944351600247, 'ham'),
           Text(1.014499764949479, -0.4251943401757036, 'spam')],
          [Text(-0.5533634864399495, 0.2319242373600l344, ' 87.37'),
           Text(0.5533635081542612, -0.23192418555038377, ' 12.63')])
```



Code Explanation:

- df['Mail_Type'].value_counts(): This line is like counting how many times we find different things in a drawer labeled "Mail_Type." In this case, there are 4,516 emails labeled "ham" (non-spam) and 653 emails labeled "spam."
- # EDA: This comment suggests that we're moving into Exploratory Data Analysis (EDA). EDA is like taking a closer look at your data to understand it better.
- !pip install matplotlib: This line installs a library called Matplotlib, which is used for creating visualizations like charts and graphs.
- plt.pie(df['Mail_Type'].value_counts(), labels=['ham', 'spam'], autopct="%0.2f"): This code creates a pie chart. It takes the counts we found in step 1 and represents them in a pie chart. The "ham" slice is larger because there are more "ham" emails. The "autopct" part adds percentage labels to the chart.

Finding Length of the Mail_Message:

1. **Checking number of characters:**

```
In [18]: # Length of Mail_Message
         # Checking number of characters
         df['num_characters'] = df['Mail_Message'].apply(len)
```

2. **Checking number of Words:**

```
In [22]: # Number of words in Mail_Message
         df['num_words'] = df['Mail_Message'].apply(lambda x : len(nltk.word_tokenize(x)))
```

3. **Checking number of Sentence:**

```
In [26]: #number of sentence in Mail_Message
         df['num_sentence'] = df['Mail_Message'].apply(lambda x : len(nltk.sent_tokenize(x)))
```

Output:

```
In [27]: df
```

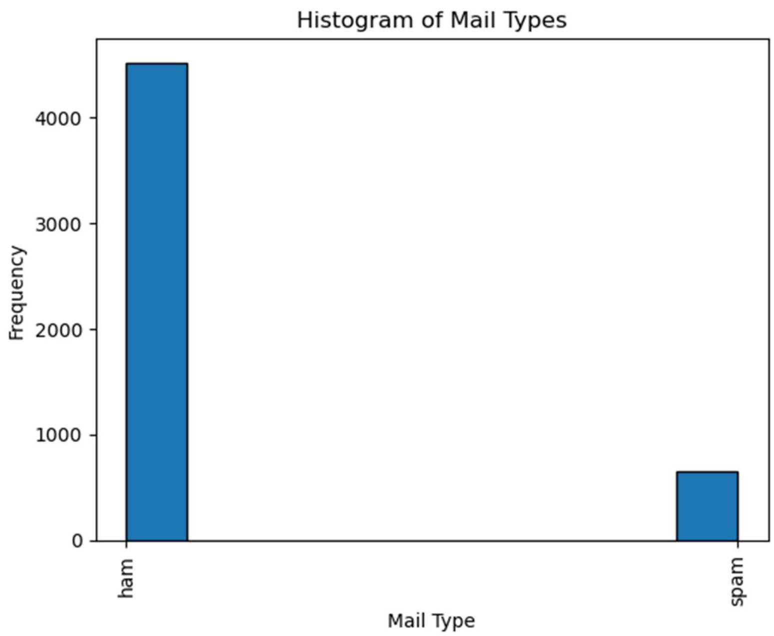
Out[27]:

	Mail_Type	Mail_Message	num_characters	num_words	num_sentence
0	ham	Go until jurong point, crazy.. Available only ...	111	24	2
1	ham	Ok lar... Joking wif u oni...	29	8	2
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	155	37	2
3	ham	U dun say so early hor... U c already then say...	49	13	1
4	ham	Nah I don't think he goes to usf, he lives aro...	61	15	1
...	...	...	...	...	...
5164	spam	This is the 2nd time we have tried 2 contact u...	161	35	4
5165	ham	Will Ì_ b going to esplanade fr home?	37	9	1
5166	ham	Pity, * was in mood for that. So...any other s...	57	15	2
5167	ham	The guy did some bitching but I acted like i'd...	125	27	1
5168	ham	Rofl. Its true to its name	26	7	2

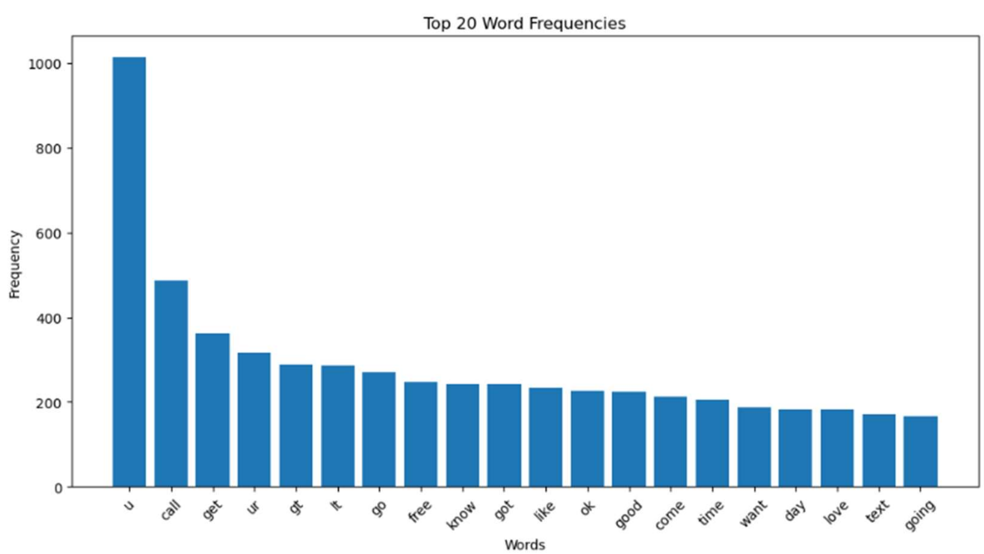
5169 rows × 5 columns

Displaying Histogram:

In [3]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load your dataset into a DataFrame, replace 'your_data.csv' with your actual data source
df = pd.read_csv('cleaned_spam.csv')

# Create a histogram for the 'Mail_Type' column
plt.hist(df['Mail_Type'], bins=10, edgecolor='black')
plt.xlabel('Mail Type')
plt.ylabel('Frequency')
plt.title('Histogram of Mail Types')
plt.xticks(rotation=90)  # Rotate x-axis labels for better visibility

# Display the histogram
plt.show()
```



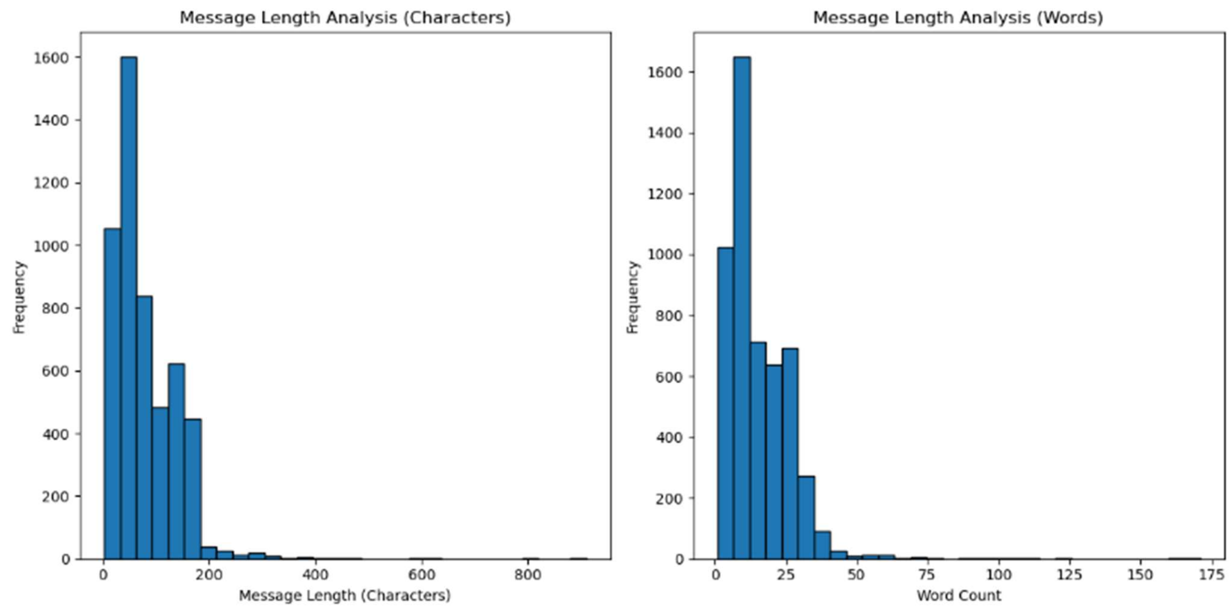
Word Frequency Analysis:



In [11]:
```python
# Word Frequency Analysis

import pandas as pd
import nltk
from nltk.corpus import stopwords
from collections import Counter
import matplotlib.pyplot as plt

# Download NLTK stopwords dataset
nltk.download('stopwords')

# Assuming you have a DataFrame 'df' with a 'Mail_Message' column containing text
# Replace 'your_data.csv' with your actual data source
df = pd.read_csv('cleaned_spam.csv')

# Combine all text into a single string
text = ' '.join(df['Mail_Message'])

# Tokenize the text
words = nltk.word_tokenize(text)

# Convert to lowercase and remove stopwords
stop_words = set(stopwords.words('english'))
filtered_words = [word.lower() for word in words if word.isalpha() and word.lower() not in stop_words]

# Count word frequencies
word_freq = Counter(filtered_words)

# Get the most common words and their frequencies
most_common_words = word_freq.most_common(20)  # Change the number to get the top N words

# Plot the word frequencies
plt.figure(figsize=(12, 6))
words, frequencies = zip(*most_common_words)
plt.bar(words, frequencies)
plt.xlabel('Words')
plt.ylabel('Frequency')
plt.title('Top 20 Word Frequencies')
plt.xticks(rotation=45)
plt.show()
```

Message length Analysis:

In [17]:
```python
# message length Analysis

import pandas as pd
import matplotlib.pyplot as plt

# Assuming you have a DataFrame 'df' with a 'Mail_Message' column containing text messages
# Replace 'your_data.csv' with your actual data source
df = pd.read_csv('cleaned_spam.csv')

# Calculate message length in characters
df['Message_Length'] = df['Mail_Message'].str.len()

# Calculate message length in words
df['Word_Count'] = df['Mail_Message'].str.split().apply(len)

# Plot the distribution of message lengths
plt.figure(figsize=(12, 6))

# Histogram of message length in characters
plt.subplot(1, 2, 1)
plt.hist(df['Message_Length'], bins=30, edgecolor='black')
plt.xlabel('Message Length (Characters)')
plt.ylabel('Frequency')
plt.title('Message Length Analysis (Characters)')

# Histogram of message length in words
plt.subplot(1, 2, 2)
plt.hist(df['Word_Count'], bins=30, edgecolor='black')
plt.xlabel('Word Count')
plt.ylabel('Frequency')
plt.title('Message Length Analysis (Words)')

plt.tight_layout()  # Ensures proper spacing of subplots
plt.show()
```



Code Explanation:

Import Libraries: The code begins by importing two important libraries: pandas for data handling and matplotlib.pyplot for creating visual plots and charts.

Load Data: It assumes we have a dataset in a CSV file (replace 'your_data.csv' with our actual file name) and uses pd.read_csv to load this data into a DataFrame called 'df.'

Calculate Message Length:

1. It creates two new columns in the DataFrame: 'Message_Length' and 'Word_Count.'
2. 'Message_Length' stores the number of characters in each text message.
3. 'Word_Count' stores the number of words in each text message.

Plot Distribution:

- The code sets up a plot area using plt.figure(figsize=(12, 6)), which is like preparing a canvas for drawing.
- It creates two histograms (bar charts) side by side to analyze the distribution of message lengths.
- The first histogram shows the distribution of message lengths in characters.
- The second histogram shows the distribution of message lengths in words.

Customizing Plots:

- For each histogram, it specifies the number of bins (divisions) and the color of the edges of the bars to make the plot visually appealing.
- It labels the X and Y axes and gives each histogram a title.
- Show the Plots: plt.tight_layout() ensures that the two plots have proper spacing between them, and plt.show() displays the plots on the screen.