```
# Import all the required packages to train the model
import numpy as np
import pandas as pd
import sklearn
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Load the spam mail dataset from the dataset
df = pd.read_csv('C:\Users\hp\OneDrive\Desktop\Ironhack\datasets\spam\spam.csv', encoding='latin1')
df.head()
```

```
	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
```

```
df.shape
```

```
(5572, 5)
```

Input data Cleaning or Data Processing
Check the structure of the dataset

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column  Unnamed: 2  Unnamed: 3  Unnamed: 4
---  ------  --------  --------  --------  --------
 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
```

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

```
	v1	v2
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...
```

```
# rename the column of the dataset
df.rename(columns='v1' 'Spam or Ham' 'v2' 'Message',inplace = True)
df.head()
```

```
	Spam or Ham	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...
```

Cleaned Dataset
```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
df['Spam or Ham'] = df['Spam or Ham'].astype('Spam or Ham')
```

```
	Spam or Ham	Message
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...
```

```
# Find if there existing nulls or null values
df.isnull().sum()
print("
print("Number of duplicates:",df.duplicated().sum())
```

```
Spam or Ham    0
Message        0
dtype: int64

Number of duplicates:  403
```

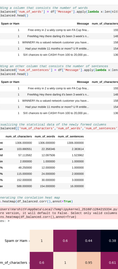
```
# Find the duplicates
df[df.duplicated()].shape
print("Number of duplicates:",df.duplicated().sum())
df.drop_duplicates(inplace=True)
df.shape
```

```
Number of duplicates:  403
(5169, 2)
```

```
# Group frequencies data analysis
target_counts = df['Spam or Ham'].value_counts()
print(target_counts)
```

```
0    4516
1     653
Name: Spam or Ham, dtype: int64
```

```
plt.bar(target_counts.index, target_counts.values, tick_label=['ham', 'spam'])
plt.xlabel('Category')
plt.ylabel('Count')
plt.title('Distribution of Ham and Spam Messages')
plt.show()
```




```
# Find to balance the data set using under sampling technique
spam_df df[df['Spam or Ham'] == 1]
ham_df df[df['Spam or Ham'] == 0]
# Randomly sample ham to match the size of spam
ham_df_balanced = ham_df.sample(spam_df.shape[0], random_state=42)
df_balanced = pd.concat([spam, ham_df_balanced])
# Visualize the balanced class distribution
plt.bar(df_balanced['Spam or Ham'].value_counts().plot(kind='bar')
plt.xlabel('Spam or Ham')
plt.ylabel('Count')
plt.title('Balanced Distribution of Spam and Ham Messages (Undersampling)')
plt.show()
```

```
# Find the statistical data of the newly formed columns
df_balanced['num_of_characters'] df_balanced['Message'].apply(lambda x len(x))
df_balanced.head()
```

```
	Spam or Ham	Message	num_of_characters
0	1	Free entry in 2 a wkly comp to win FA Cup fina...	155
5	1	Freephone Mobile Update this Weekend between ...	149
9	1	WINNER!!As a valued network customer you have ...	158
10	1	Had your mobile 11 months or more? U R entitle...	160
11	1	SIX chances to win CASH From 100 to 20,000 po...	136
```

```
# Adding an other column that contains the number of sentences
df_balanced['num_of_sentences'] df_balanced['Message'].apply(lambda x len(nltk.sent_tokenize(x)))
df_balanced.head()
```

```
	Spam or Ham	Message	num_of_characters	num_of_words
0	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	27
5	1	Freephone Mobile Update this Weekend between ...	149	29
9	1	WINNER!!As a valued network customer you have ...	158	29
10	1	Had your mobile 11 months or more? U R entitle...	160	35
11	1	SIX chances to win CASH From 100 to 20,000 po...	136	31
```

```
# Introducing the statistical data of the newly formed columns
df_balanced[['num_of_characters', 'num_of_words', 'num_of_sentences']].describe()
```

```
	num_of_characters	num_of_words	num_of_sentences
count	1306.000000	1306.000000	1306.000000
mean	153.098591	27.352604	3.455828
std	57.11562	12.997609	1.557892
min	13.000000	2.000000	1.000000
25%	95.750000	17.000000	2.000000
50%	148.000000	27.000000	3.000000
75%	196.000000	36.000000	4.000000
max	395.000000	104.000000	38.000000
```

Generating the correlation heat map
```
# Generating the correlation heat map
df_balanced.corr(numeric_only=True)
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_25180\1436423204.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
	Spam or Ham	num_of_characters	num_of_words	num_of_sentences
Spam or Ham	1.000000	0.384456	0.384456	0.338126
num_of_characters	0.384456	1.000000	0.965753	0.684141
num_of_words	0.384456	0.965753	1.000000	0.679971
num_of_sentences	0.338126	0.684141	0.679971	1.000000
```

Input data Processing
Convert all the words into lower case and removing the stopwords
```
import nltk
from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()

def transform_message(Message):
    Message = Message.lower()
    Message = nltk.word_tokenize(Message)
    y = []
    for i in Message:
        if i.isalnum():
            y.append(i)
    Message = y[:]
    y.clear()
    for i in Message:
        if i not in stopwords.words('english') and i not in string.punctuation:
            y.append(i)
    Message = y[:]
    y.clear()
    for i in Message:
        y.append(ps.stem(i))
    return " " .join(y)
```

Adding a column that is an improved version of the original message
```
df_balanced['transformed_message'] df_balanced['Message'].apply(transform_Message)
df_balanced.head()
```

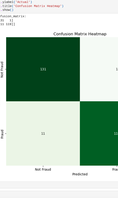
```
	Spam or Ham	Message	num_of_characters	num_of_words	num_of_sentences	transformed_message
0	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	27	2	free entri 2 wkli comp win fa cup final tkts ...
5	1	Freephone Mobile Update this Weekend between ...	149	29	3	freephon mobil updat weekend free text or ...
9	1	WINNER!!As a valued network customer you have ...	158	29	3	winner!!as valu network customer have select ...
10	1	Had your mobile 11 months or more? U R entitle...	160	35	3	mobil 11 month u r entitl update latest colour ...
11	1	SIX chances to win CASH From 100 to 20,000 po...	136	31	1	six chance win cash 100 pound txt ch send co...
```

Adding a workcloud for the spam outcomes
```
from wordcloud import WordCloud
wc = WordCloud(width=500,height=500,min_font_size=10,background_color='white')
spam_wc = wc.generate(df_balanced[df_balanced['Spam or Ham'] == 1]['transformed_message'].str.cat(sep=" "))
plt.figure(figsize=(15,6))
plt.imshow(spam_wc)
```

```
<matplotlib.image.AxesImage at 0x1d8b6d37f8>
```

Adding a workcloud for the ham outcomes
```
ham_wc = wc.generate(df_balanced[df_balanced['Spam or Ham'] == 0]['transformed_message'].str.cat(sep=" "))
plt.figure(figsize=(15,6))
plt.imshow(ham_wc)
```

```
<matplotlib.image.AxesImage at 0x1d8b6d37f8>
```



Removing the length of spam corpus
```
spam_corpus = []
for msg in df_balanced[df_balanced['Spam or Ham'] == 1]['transformed_message'].tolist():
    for word in msg.split():
        spam_corpus.append(word)
len(spam_corpus)
```

```
9939
```

```
# from collections import Counter
spam_word_counts = Counter(spam_corpus).most_common(30)
```

```
# Convert to dataframe
spam_word_df = pd.DataFrame(spam_word_counts, columns=['word', 'count'])
```

```
# Plot the barplot
plt.figure(figsize=(10, 6))
sns.barplot(x='word', y='count', data=spam_word_df)
plt.xticks(rotation='vertical')
plt.show()
```


Removing the length of ham corpus
```
ham_corpus = []
for msg in df_balanced[df_balanced['Spam or Ham'] == 0]['transformed_message'].tolist():
    for word in msg.split():
        ham_corpus.append(word)
len(ham_corpus)
```

```
9224
```

```
# from collections import Counter
ham_word_counts = Counter(ham_corpus).most_common(30)
```

```
# Convert to dataframe
ham_word_df = pd.DataFrame(ham_word_counts, columns=['word', 'count'])
```

```
# Plot the barplot
plt.figure(figsize=(10, 6))
sns.barplot(x='word', y='count', data=ham_word_df)
plt.xticks(rotation='vertical')
plt.show()
```


Input Model Building
```
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)
X = tfidf.fit_transform(df_balanced['transformed_message'].lower())
y = df_balanced['Spam or Ham'].values
```

```
# Train test split
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
print("X_train shape",X_train.shape)
print("X_test shape",X_test.shape)
print("y_train shape",y_train.shape)
print("y_test shape",y_test.shape)
```

```
X_train shape (1044, 3000)
X_test shape (262, 3000)
y_train shape (1044,)
y_test shape (262,)
```

Start the model
```
# Import the naive bayes classifier
from sklearn.naive_bayes import BernoulliNaiveBayes
bnb = BernoulliNB()
df_fitted_bnb = bnb.fit(X_train,y_train)
y_pred = df_fitted_bnb.predict(X_test)
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       
```