Lab 3 : Sentiment/Classification analysis

Q 1] You are given a dataset containing a collection of emails labeled "Spam" or "Not Spam." Your task is to build a Naive Bayes classifier that can predict whether a new email is spam or not based on its content.

In [1]:
```
import pandas as pd

spam_dataset = pd.read_csv("spam_ham_dataset.csv", header=0,names=['Text Length', 'Label', 'Email', 'spam'])
```

In [2]:
```
spam_dataset.head()
```

Out[2]:

	Text Length	Label	Email	spam
0	605	ham	Subject: enron methanol ; meter # : 988291\r\n...	0
1	2349	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	3624	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	4685	spam	Subject: photoshop , windows , office . cheap ...	1
4	2030	ham	Subject: re : indian springs\r\nthis deal is t...	0

In [3]:
```
ham=spam_dataset[spam_dataset['spam']==0]
spam = spam_dataset[spam_dataset['spam']==1]

print('Spam Percentage =',(len(spam)/len(spam_dataset))*100,'%')
print('Ham Percentage =',(len(ham)/len(spam_dataset))*100,'%')
```

```
Spam Percentage = 28.98859021465867 %
Ham Percentage = 71.01140978534133 %
```

In [4]:
```
spam_dataset.drop(columns=['Label'], inplace=True)
spam_dataset.head(5)
```

Out[4]:

	Text Length	Email	spam
0	605	Subject: enron methanol ; meter # : 988291\r\n...	0
1	2349	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	3624	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	4685	Subject: photoshop , windows , office . cheap ...	1
4	2030	Subject: re : indian springs\r\nthis deal is t...	0

Data Cleaning and Preprocessing

- LowerCase
- Tokenisation
- Removing special characters and "Subject:"
- Removing stop words

In [5]:
```
# Remove leading spaces first, then remove "Subject:"
spam_dataset["Email"] = spam_dataset["Email"].str.strip().str.replace(r"^Subject:\s*", "", case=False, regex=True).st
spam_dataset.head(5)
```

Out[5]:

	Text Length	Email	spam
0	605	enron methanol ; meter # : 988291\r\nthis is a...	0
1	2349	hpl nom for january 9 , 2001\r\n(see attached...	0
2	3624	neon retreat\r\nho ho ho , we ' re around to t...	0
3	4685	photoshop , windows , office . cheap . main tr...	1
4	2030	re : indian springs\r\nthis deal is to book th...	0

In [6]:
```
# Lowercase and removing special characters.
import re
def clean_data(text):
    text = text.lower()
```

```
    text = re.sub(r'[^a-z0-9]', ' ', text)
    return text

spam_dataset["Email"] = spam_dataset["Email"].apply(clean_data)
spam_dataset.head(10)
```

Out[6]:

	Text Length	Email	spam
0	605	enron methanol meter 988291 this is a f...	0
1	2349	hpl nom for january 9 2001 see attached f...	0
2	3624	neon retreat ho ho ho we re around to tha...	0
3	4685	photoshop windows office cheap main tr...	1
4	2030	re indian springs this deal is to book the ...	0
5	2949	ehronline web address change this message is ...	0
6	2793	spring savings certificate take 30 off sa...	0
7	4185	looking for medication we re the best sour...	1
8	2641	noms actual flow for 2 26 we agree ...	0
9	1870	nominations for oct 21 23 2000 see at...	0

In [7]:

```
# Remove stopwords
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')

stop_words = set(stopwords.words('english'))

def remove_stopwords(text):
    filtered_words = [word for word in text.split() if word not in stop_words]
    return ' '.join(filtered_words)

spam_dataset["Email"] = spam_dataset["Email"].apply(remove_stopwords)
spam_dataset.head(10)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\alens\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[7]:

	Text Length	Email	spam
0	605	enron methanol meter 988291 follow note gave m...	0
1	2349	hpl nom january 9 2001 see attached file hplno...	0
2	3624	neon retreat ho ho ho around wonderful time ye...	0
3	4685	photoshop windows office cheap main trending a...	1
4	2030	indian springs deal book teco pvr revenue unde...	0
5	2949	ehronline web address change message intended ...	0
6	2793	spring savings certificate take 30 save 30 use...	0
7	4185	looking medication best source difficult make ...	1
8	2641	noms actual flow 2 26 agree forwarded melissa ...	0
9	1870	nominations oct 21 23 2000 see attached file h...	0

Applying TF-IDF

In [8]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()

spamham_vectorizer = vectorizer.fit(spam_dataset['Email'])
# spamham_vectorizer.vocabulary_
```

In [9]:
```
vector = vectorizer.transform(spam_dataset['Email'])
df = pd.DataFrame(vector.todense(), columns=vectorizer.get_feature_names_out())
df
```

Out[9]:

	00	000	0000	000000	000000000002858	000000000049773	000080	000099	0001	00018	...	zynve	zyqt
0	0.104068	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
1	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
2	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
3	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
4	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
5166	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
5167	0.000000	0.123354	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
5168	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
5169	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	
5170	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	

5171 rows × 50303 columns

In [10]:
```
vector.shape
```

Out[10]:
```
(5171, 50303)
```

Train and Test Split

In [11]:
```
label = spam_dataset['spam']
X = vector
y = label
print(f'X shape: {X.shape}, Y shape: {y.shape}')
```

```
X shape: (5171, 50303), Y shape: (5171,)
```

In [12]:
```
# Train Split : 80% and Test Split of 20%
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)
```

In [13]:
```
# Training Model
from sklearn.naive_bayes import MultinomialNB

NB_classifier=MultinomialNB()
NB_classifier.fit(X_train,y_train)
```

Out[13]:
```
MultinomialNB
MultinomialNB()
```

Performance Evaluation

In [14]:
```
# Calculating the accuracy
from sklearn.metrics import accuracy_score

y_predict_train=NB_classifier.predict(X_train)
y_predict_test=NB_classifier.predict(X_test)

train_accuracy = accuracy_score(y_predict_train, y_train)
test_accuracy = accuracy_score(y_predict_test, y_test)

print("Training Accuracy:", train_accuracy)
print("Testing Accuracy:", test_accuracy)
```

```
Training Accuracy: 0.9613152804642167
Testing Accuracy: 0.9120772946859903
```

In [15]:
```
from sklearn.metrics import precision_score, recall_score, f1_score

# Precision, Recall, and F1-score
train_precision = precision_score(y_train, y_predict_train, average='binary')
train_recall = recall_score(y_train, y_predict_train, average='binary')

test_precision = precision_score(y_test, y_predict_test, average='binary')
test_recall = recall_score(y_test, y_predict_test, average='binary')
f1 = f1_score(y_test, y_predict_test, average='binary')

print("Training Precision:", train_precision)
```

```
print("Testing Precision:", test_precision)
print("\nTraining Recall:", train_recall)
print("Testing Recall:", test_recall)
print("\nF1-score:", f1)
```

```
Training Precision: 1.0
Testing Precision: 0.9906542056074766

Training Recall: 0.8664440734557596
Testing Recall: 0.7043189368770764

F1-score: 0.8233009708737864
```

In [16]:
```
from sklearn.metrics import classification_report

print("Classification Report For Test Data:\n", classification_report(y_test, y_predict_test))
```

```
Classification Report For Test Data:
              precision    recall  f1-score   support

           0       0.89      1.00      0.94       734
           1       0.99      0.70      0.82       301

    accuracy                           0.91      1035
   macro avg       0.94      0.85      0.88      1035
weighted avg       0.92      0.91      0.91      1035
```

Q2] You have a dataset containing articles from different domains, such as Sports, Politics, and Technology. Your task is to train a Naive Bayes classifier that can automatically categorize a new document into one of these domains based on its content.

In [17]:
```
# importing dataset
article_dataset = pd.read_csv("BBC_News_dataset.csv")
article_dataset.head()
```

Out[17]:

	ArticleId	Text	Category
0	1833	worldcom ex-boss launches defence lawyers defe...	business
1	154	german business confidence slides german busin...	business
2	1101	bbc poll indicates economic gloom citizens in ...	business
3	1976	lifestyle governs mobile choice faster bett...	tech
4	917	enron bosses in $168m payout eighteen former e...	business

In [18]:
```
article_dataset.drop(columns=['ArticleId'], inplace=True)
print("Unique categories are: \n")
target_categories = article_dataset['Category'].unique()
print(target_categories)
article_dataset['Category'].value_counts()
```

```
Unique categories are:

['business' 'tech' 'politics' 'sport' 'entertainment']
```

Out[18]:
```
Category
sport            346
business         336
politics         274
entertainment    273
tech             261
Name: count, dtype: int64
```

In [19]:
```
# Convert Categorical value into numerical value
article_dataset['CategoryId'] = article_dataset['Category'].factorize()[0]
article_dataset.head()
```

Out[19]:

	Text	Category	CategoryId
0	worldcom ex-boss launches defence lawyers defe...	business	0
1	german business confidence slides german busin...	business	0
2	bbc poll indicates economic gloom citizens in ...	business	0
3	lifestyle governs mobile choice faster bett...	tech	1
4	enron bosses in $168m payout eighteen former e...	business	0

In [20]:
```
category = article_dataset[['Category', 'CategoryId']].drop_duplicates().sort_values('CategoryId')
category
```

Out[20]:

	Category	CategoryId
0	business	0
3	tech	1
5	politics	2
6	sport	3
7	entertainment	4

Data Cleaning and Preprocessing

- Remove punctuations
- Lowercase all words
- Remove Stop words
- Lemmatize

In [21]:
```
# Remove punctuations and Lowercase
article_dataset["Text"] = article_dataset["Text"].apply(clean_data)
article_dataset.head(10)
```

Out[21]:

	Text	Category	CategoryId
0	worldcom ex boss launches defence lawyers defe...	business	0
1	german business confidence slides german busin...	business	0
2	bbc poll indicates economic gloom citizens in ...	business	0
3	lifestyle governs mobile choice faster bett...	tech	1
4	enron bosses in 168m payout eighteen former e...	business	0
5	howard truanted to play snooker conservative...	politics	2
6	wales silent on grand slam talk rhys williams ...	sport	3
7	french honour for director parker british film...	entertainment	4
8	car giant hit by mercedes slump a slump in pro...	business	0
9	fockers fuel festive film chart comedy meet th...	entertainment	4

In [22]:
```
# Remove stopwords
article_dataset["Text"] = article_dataset["Text"].apply(remove_stopwords)
article_dataset.head(10)
```

Out[22]:

	Text	Category	CategoryId
0	worldcom ex boss launches defence lawyers defe...	business	0
1	german business confidence slides german busin...	business	0
2	bbc poll indicates economic gloom citizens maj...	business	0
3	lifestyle governs mobile choice faster better ...	tech	1
4	enron bosses 168m payout eighteen former enron...	business	0
5	howard truanted play snooker conservative lead...	politics	2
6	wales silent grand slam talk rhys williams say...	sport	3
7	french honour director parker british film dir...	entertainment	4
8	car giant hit mercedes slump slump profitabili...	business	0
9	fockers fuel festive film chart comedy meet fo...	entertainment	4

In [23]:

```
# Lemmatizing the Words
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk.download('punkt_tab')

lemmatizer = WordNetLemmatizer()

def lemmatize_text(text):
    words = word_tokenize(text)
    lemmatized_words = [lemmatizer.lemmatize(word) for word in words]
    return ' '.join(lemmatized_words)

article_dataset["Text"] = article_dataset["Text"].apply(lemmatize_text)

article_dataset.head(5)
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\alens\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\alens\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package punkt_tab to
[nltk_data]     C:\Users\alens\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
```

Out[23]:

	Text	Category	CategoryId
0	worldcom ex bos launch defence lawyer defendin...	business	0
1	german business confidence slide german busine...	business	0
2	bbc poll indicates economic gloom citizen majo...	business	0
3	lifestyle governs mobile choice faster better ...	tech	1
4	enron boss 168m payout eighteen former enron d...	business	0

Applying TF-IDF

In [24]:
```
vectorizer = TfidfVectorizer()

article_vectorizer = vectorizer.fit_transform(article_dataset['Text'])

df = pd.DataFrame(article_vectorizer.todense(), columns=vectorizer.get_feature_names_out())
df
# spamham_vectorizer.vocabulary_
```

Out[24]:

	00	000	0001	000bn	000m	000th	001	001and	001st	0051	...	zombie	zone	zonealarm	zoom	zooropa	zo
0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.023132	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.018154	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1485	0.0	0.032602	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
1486	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
1487	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
1488	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	
1489	0.0	0.025612	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	

1490 rows × 22045 columns

Train & Test Split

In [25]:
```
label = article_dataset['CategoryId']
X = article_vectorizer
y = label
print(f'X shape: {X.shape}, Y shape: {y.shape}')
```

```
X shape: (1490, 22045), Y shape: (1490,)
```

In [26]:
```
# Train Split : 80% and Test Split of 20%
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
```

In [27]:
```
# Training Model
NB_classifier.fit(X_train,y_train)
```

Out[27]:

MultinomialNB

MultinomialNB()

Performance Evaluation

In [28]:

```
y_pred_train = NB_classifier.predict(X_train)
y_pred_test = NB_classifier.predict(X_test)

train_accuracy = accuracy_score(y_train, y_pred_train)
train_precision = precision_score(y_train, y_pred_train, average='micro')
train_recall = recall_score(y_train, y_pred_train, average='micro')

test_accuracy = accuracy_score(y_test, y_pred_test)
test_precision = precision_score(y_test, y_pred_test, average='micro')
test_recall = recall_score(y_test, y_pred_test, average='micro')
test_f1 = f1_score(y_test, y_pred_test, average='micro')

print(f"Training Accuracy: {train_accuracy:.4f}")
print(f"Testing Accuracy: {test_accuracy:.4f}")
print("\n")
print(f"Training Precision: {train_precision:.4f}")
print(f"Testing Precision: {test_precision:.4f}")
print("\n")
print(f"Training Recall: {train_recall:.4f}")
print(f"Testing Recall: {test_recall:.4f}")
print(f"\nF1-score : {test_f1:.4f}")

print("\nClassification Report:\n")
print(classification_report(y_test, y_pred_test))
```

```
Training Accuracy: 0.9933
Testing Accuracy: 0.9564


Training Precision: 0.9933
Testing Precision: 0.9564


Training Recall: 0.9933
Testing Recall: 0.9564

F1-score : 0.9564

Classification Report:
```

	precision	recall	f1-score	support
0	0.90	0.98	0.94	63
1	0.98	0.91	0.94	55
2	0.98	0.94	0.96	64
3	0.94	1.00	0.97	59
4	1.00	0.95	0.97	57
accuracy			0.96	298
macro avg	0.96	0.96	0.96	298
weighted avg	0.96	0.96	0.96	298

Q3] You are given a dataset containing movie reviews labeled as Positive or Negative. Your task is to build a Naive Bayes classifier to predict the sentiment of a new review.

In [50]:
```
movie_review_dataset = pd.read_csv("movie_review_dataset.csv", nrows=5000)
movie_review_dataset.head(5)
```

Out[50]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

In [51]:
```
print("Unique Sentiments are: \n")
target_sentiment = movie_review_dataset['sentiment'].unique()
print(target_sentiment)
```

```
Unique Sentiments are:

['positive' 'negative']
```

In [52]:
```
# Convert Categorical value into numerical value
movie_review_dataset['sentimentId'] = movie_review_dataset['sentiment'].map({'positive': 1, 'negative': 0})
movie_review_dataset.head()
```

Out[52]:

	review	sentiment	sentimentId
0	One of the other reviewers has mentioned that ...	positive	1
1	A wonderful little production. The...	positive	1
2	I thought this was a wonderful way to spend ti...	positive	1
3	Basically there's a family where a little boy ...	negative	0
4	Petter Mattei's "Love in the Time of Money" is...	positive	1

Data Cleaning and Preprocessing

- Remove punctuations and special characters
- Lowercase all words

- Remove Stop words
- Remove most frequent words
- Lemmatize

In [53]:
```
# Remove punctuations and Lowercase
movie_review_dataset["review"] = movie_review_dataset["review"].apply(clean_data)
movie_review_dataset.head(10)
```

Out[53]:

	review	sentiment	sentimentId
0	one of the other reviewers has mentioned that ...	positive	1
1	a wonderful little production br br the...	positive	1
2	i thought this was a wonderful way to spend ti...	positive	1
3	basically there s a family where a little boy ...	negative	0
4	petter mattei s love in the time of money is...	positive	1
5	probably my all time favorite movie a story o...	positive	1
6	i sure would like to see a resurrection of a u...	positive	1
7	this show was an amazing fresh innovative i...	negative	0
8	encouraged by the positive comments about this...	negative	0
9	if you like original gut wrenching laughter yo...	positive	1

In [54]:
```
# Remove stopwords
movie_review_dataset["review"] = movie_review_dataset["review"].apply(remove_stopwords)
movie_review_dataset.head(10)
```

Out[54]:

	review	sentiment	sentimentId
0	one reviewers mentioned watching 1 oz episode ...	positive	1
1	wonderful little production br br filming tech...	positive	1
2	thought wonderful way spend time hot summer we...	positive	1
3	basically family little boy jake thinks zombie...	negative	0
4	petter mattei love time money visually stunnin...	positive	1
5	probably time favorite movie story selflessnes...	positive	1
6	sure would like see resurrection dated seahunt...	positive	1
7	show amazing fresh innovative idea 70 first ai...	negative	0
8	encouraged positive comments film looking forw...	negative	0
9	like original gut wrenching laughter like movi...	positive	1

In [55]:

```
# Removing most frequent words
from collections import Counter

cnt = Counter()
for text in movie_review_dataset["review"].values:
    for word in text.split():
        cnt[word] += 1

FREQWORDS = set([w for (w, wc) in cnt.most_common(10)])

def remove_most_freqwords(text):
    return " ".join([word for word in text.split() if word not in FREQWORDS])

movie_review_dataset["review"] = movie_review_dataset["review"].apply(remove_most_freqwords)


movie_review_dataset.head()
```

Out[55]:

	review	sentiment	sentimentId
0	reviewers mentioned watching 1 oz episode hook...	positive	1
1	wonderful little production filming technique ...	positive	1
2	thought wonderful way spend hot summer weekend...	positive	1
3	basically family little boy jake thinks zombie...	negative	0
4	petter mattei love money visually stunning wat...	positive	1

In [56]:
```
# Lemmatizing the Words
movie_review_dataset["review"] = movie_review_dataset["review"].apply(lemmatize_text)

movie_review_dataset.head(5)
```

Out[56]:

	review	sentiment	sentimentId
0	reviewer mentioned watching 1 oz episode hooke...	positive	1
1	wonderful little production filming technique ...	positive	1
2	thought wonderful way spend hot summer weekend...	positive	1
3	basically family little boy jake think zombie ...	negative	0
4	petter mattei love money visually stunning wat...	positive	1

Applying TF-IDF

In [57]:
```
vectorizer = TfidfVectorizer()

movie_review_vectorizer = vectorizer.fit_transform(movie_review_dataset['review'])
```

Train & Test Split

In [58]:
```
label = movie_review_dataset['sentimentId']
X = movie_review_vectorizer
# X = X.astype("float32")
```

```
y = label
print(f'X shape: {X.shape}, Y shape: {y.shape}')
```

```
X shape: (5000, 34697), Y shape: (5000,)
```

In [59]:
```
# Train Split : 80% and Test Split of 20%
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
```

In [60]:
```
# Training Model
NB_classifier.fit(X_train,y_train)
```

Out[60]:
```
MultinomialNB
MultinomialNB()
```

Performance Evaluation

In [62]:
```
y_pred_train = NB_classifier.predict(X_train)
y_pred_test = NB_classifier.predict(X_test)

train_accuracy = accuracy_score(y_train, y_pred_train)
train_precision = precision_score(y_train, y_pred_train)
train_recall = recall_score(y_train, y_pred_train)

test_accuracy = accuracy_score(y_test, y_pred_test)
test_precision = precision_score(y_test, y_pred_test)
test_recall = recall_score(y_test, y_pred_test)
test_f1 = f1_score(y_test, y_pred_test)

print(f"Training Accuracy: {train_accuracy:.4f}")
print(f"Testing Accuracy: {test_accuracy:.4f}")
print("\n")
print(f"Training Precision: {train_precision:.4f}")
print(f"Testing Precision: {test_precision:.4f}")
print("\n")
print(f"Training Recall: {train_recall:.4f}")
print(f"Testing Recall: {test_recall:.4f}")
print(f"\nF1-score : {test_f1:.4f}")
```

```
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred_test))
```

```
Training Accuracy: 0.9490
Testing Accuracy: 0.8320


Training Precision: 0.9894
Testing Precision: 0.9175


Training Recall: 0.9054
Testing Recall: 0.7383

F1-score : 0.8182

Classification Report:

              precision    recall  f1-score   support

           0       0.77      0.93      0.84       488
           1       0.92      0.74      0.82       512

    accuracy                           0.83      1000
   macro avg       0.84      0.83      0.83      1000
weighted avg       0.85      0.83      0.83      1000
```

Q4] You have a dataset of customer product reviews labeled as Positive, Neutral, or Negative. Your goal is to develop a Naive Bayes classifier that can predict the sentiment of a given product review.

In [64]:
```
# importing dataset
product_review_dataset = pd.read_csv("product_review_dataset.csv")
product_review_dataset.head()
```

Out[64]:

	Review	Sentiment
0	This product exceeded my expectations! It's hi...	Positive
1	The product was decent. It worked fine, but it...	Neutral
2	I had a terrible experience with this company....	Negative
3	It's an okay product. Nothing to write home ab...	Neutral
4	Disappointed with the product. It didn't meet ...	Negative

In [67]:
```
print("Unique sentiment are: \n")
target_sentiment = product_review_dataset['Sentiment'].unique()
print(target_sentiment)
product_review_dataset['Sentiment'].value_counts()
```

```
Unique sentiment are:

['Positive' 'Neutral' 'Negative']
```

Out[67]:
```
Sentiment
Positive    129
Negative    129
Neutral     128
Name: count, dtype: int64
```

In [69]:
```
# Convert Categorical value into numerical value
product_review_dataset['SentimentId'] = product_review_dataset['Sentiment'].map({'Positive': 1, 'Negative': -1, 'Neut
product_review_dataset.head()
```

Out[69]:

	Review	Sentiment	SentimentId
0	This product exceeded my expectations! It's hi...	Positive	1
1	The product was decent. It worked fine, but it...	Neutral	0
2	I had a terrible experience with this company....	Negative	-1
3	It's an okay product. Nothing to write home ab...	Neutral	0
4	Disappointed with the product. It didn't meet ...	Negative	-1

Data Cleaning and Preprocessing

- Remove punctuations and special characters
- Lowercase all words
- Remove Stop words
- Lemmatize

In [72]:
```
# Remove punctuations and Lowercase
product_review_dataset["Review"] = product_review_dataset["Review"].apply(clean_data)
product_review_dataset.head(10)
```

Out[72]:

	Review	Sentiment	SentimentId
0	this product exceeded my expectations it s hi...	Positive	1
1	the product was decent it worked fine but it...	Neutral	0
2	i had a terrible experience with this company ...	Negative	-1
3	it s an okay product nothing to write home ab...	Neutral	0
4	disappointed with the product it didn t meet ...	Negative	-1
5	avoid this company at all costs the service i...	Negative	-1
6	i had a terrible experience with this company ...	Negative	-1
7	avoid this company at all costs the service i...	Negative	-1
8	this product exceeded my expectations it s hi...	Positive	1
9	this product is outstanding it s exactly what...	Positive	1

In [73]:
```
# Remove stopwords
product_review_dataset["Review"] = product_review_dataset["Review"].apply(remove_stopwords)
product_review_dataset.head(10)
```

Out[73]:

	Review	Sentiment	SentimentId
0	product exceeded expectations high quality per...	Positive	1
1	product decent worked fine anything special	Neutral	0
2	terrible experience company customer service r...	Negative	-1
3	okay product nothing write home	Neutral	0
4	disappointed product meet expectations	Negative	-1
5	avoid company costs service terrible products ...	Negative	-1
6	terrible experience company customer service r...	Negative	-1
7	avoid company costs service terrible products ...	Negative	-1
8	product exceeded expectations high quality per...	Positive	1
9	product outstanding exactly looking works perf...	Positive	1

In [74]:
```
# Lemmatizing the Words
product_review_dataset["Review"] = product_review_dataset["Review"].apply(lemmatize_text)

product_review_dataset.head(5)
```

Out[74]:

	Review	Sentiment	SentimentId
0	product exceeded expectation high quality perf...	Positive	1
1	product decent worked fine anything special	Neutral	0
2	terrible experience company customer service r...	Negative	-1
3	okay product nothing write home	Neutral	0
4	disappointed product meet expectation	Negative	-1

Applying TF-IDF

In [75]:
```
vectorizer = TfidfVectorizer()

product_review_vectorizer = vectorizer.fit_transform(product_review_dataset['Review'])
```

Train and Test Split

In [76]:
```
label = product_review_dataset['SentimentId']
X = product_review_vectorizer
# X = X.astype("float32")
y = label
print(f'X shape: {X.shape}, Y shape: {y.shape}')
```

```
X shape: (386, 154), Y shape: (386,)
```

In [80]:
```
# Train Split : 80% and Test Split of 20%
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3)
```

In [81]:
```
# Training Model
NB_classifier.fit(X_train,y_train)
```

Out[81]:

MultinomialNB

```
MultinomialNB()
```

In [82]:
```
y_pred_train = NB_classifier.predict(X_train)
y_pred_test = NB_classifier.predict(X_test)

train_accuracy = accuracy_score(y_train, y_pred_train)
train_precision = precision_score(y_train, y_pred_train, average='micro')
train_recall = recall_score(y_train, y_pred_train, average='micro')

test_accuracy = accuracy_score(y_test, y_pred_test)
test_precision = precision_score(y_test, y_pred_test, average='micro')
test_recall = recall_score(y_test, y_pred_test, average='micro')
test_f1 = f1_score(y_test, y_pred_test, average='micro')

print(f"Training Accuracy: {train_accuracy:.4f}")
print(f"Testing Accuracy: {test_accuracy:.4f}")
print("\n")
```

```
print(f"Training Precision: {train_precision:.4f}")
print(f"Testing Precision: {test_precision:.4f}")
print("\n")
print(f"Training Recall: {train_recall:.4f}")
print(f"Testing Recall: {test_recall:.4f}")
print(f"\nF1-score : {test_f1:.4f}")

print("\nClassification Report:\n")
print(classification_report(y_test, y_pred_test))
```

```
Training Accuracy: 0.9963
Testing Accuracy: 1.0000


Training Precision: 0.9963
Testing Precision: 1.0000


Training Recall: 0.9963
Testing Recall: 1.0000

F1-score : 1.0000

Classification Report:

              precision    recall  f1-score   support

          -1       1.00      1.00      1.00        35
           0       1.00      1.00      1.00        46
           1       1.00      1.00      1.00        35

    accuracy                           1.00       116
   macro avg       1.00      1.00      1.00       116
weighted avg       1.00      1.00      1.00       116
```