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

# Read the CSV file into a DataFrame
df = pd.read_csv('your_file.csv')
```

In [2]:
```
data2 = df.copy()
```

In [3]:
```
from sklearn.preprocessing import LabelEncoder

# Initialize the LabelEncoder
label_encoder = LabelEncoder()

# Fit and transform the 'sentiment' column
data2['sentiment_encoded'] = label_encoder.fit_transform(data2['Sentiment'])

# Display the DataFrame with the encoded sentiment column
data2.head()
```

Out[3]:

	Product Name	Brand Name	Price	Rating	Reviews	Review Votes	Sentiment	Tokenized	Without_Stopwords
0	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	5	i feel so lucky to have found this used (phone...	1.0	positive	['i', 'feel', 'so', 'lucky', 'to', 'have', 'fo...	['feel', 'lucky', 'found', 'used', '(', 'phone...
1	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	4	nice phone, nice up grade from my pantach revu...	0.0	positive	['nice', 'phone', ',', 'nice', 'up', 'grade', ...	['nice', 'phone', ',', 'nice', 'grade', 'panta...
2	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	5	very pleased	0.0	positive	['very', 'pleased']	['pleased']
3	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	4	it works good but it goes slow sometimes but i...	0.0	positive	['it', 'works', 'good', 'but', 'it', 'goes', '...	['works', 'good', 'goes', 'slow', 'sometimes',...
4	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	4	great phone to replace my lost phone. the only...	0.0	positive	['great', 'phone', 'to', 'replace', 'my', 'los...	['great', 'phone', 'replace', 'lost', 'phone',...

In [4]:

```
# Check the distribution of the neutral , positive and negative reviews
data2.Sentiment.value_counts()
```

Out[4]:

```
Sentiment
positive    236886
negative     84902
neutral      27682
Name: count, dtype: int64
```

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 349470 entries, 0 to 349469
Data columns (total 11 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   Product Name         349470 non-null  object 
 1   Brand Name           294786 non-null  object 
 2   Price                349470 non-null  float64
 3   Rating               349470 non-null  int64  
 4   Reviews              349470 non-null  object 
 5   Review Votes         349470 non-null  float64
 6   Sentiment            349470 non-null  object 
 7   Tokenized            349470 non-null  object 
 8   Without_Stopwords    349470 non-null  object 
 9   Without_Punctuation  349470 non-null  object 
 10  Lemmatized           349470 non-null  object 
dtypes: float64(2), int64(1), object(8)
memory usage: 29.3+ MB
```

In [6]:

```
# Assuming data2 is your DataFrame
neg = data2.loc[data2.Sentiment == 'negative']
pos = data2.loc[data2.Sentiment == 'positive'].sample(n=len(neg), random_state=42)
# Count the number of negative reviews
nue = data2.Sentiment.value_counts()['negative']

# Select all available 'neutral' reviews if there are fewer of them than negative revi
if nue >= data2.Sentiment.value_counts().get('neutral', 0):
    neutral = data2.loc[data2.Sentiment == 'neutral']
else:
    neutral = data2.loc[data2.Sentiment == 'neutral'].sample(n=nue, random_state=42)
```

In [7]:

```
import nltk
nltk.download('wordnet')
```

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

Out[7]:

```
True
```

In [8]:

```
import nltk
from nltk.corpus import stopwords as sw
import string
import matplotlib.pyplot as plt
%matplotlib inline

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

lemmatizer = nltk.WordNetLemmatizer()
stopwords = sw.words('english')
stopwords = stopwords + ['not_' + w for w in stopwords]

# transform punctuation to blanks
trans_punct = str.maketrans(string.punctuation,' '*len(string.punctuation))

# pad punctuation with blanks
pad_punct = str.maketrans({key: " {0} ".format(key) for key in string.punctuation})
# remove "_" from string.punctuation
invalidChars = str(string.punctuation.replace("_", ""))
```

In [12]:

```
data2.head()
```

Out[12]:

	Product Name	Brand Name	Price	Rating	Reviews	Review Votes	Sentiment	Tokenized	Without_Stopwords
0	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	5	i feel so lucky to have found this used (phone...	1.0	positive	['i', 'feel', 'so', 'lucky', 'to', 'have', 'fo...	['feel', 'lucky', 'found', 'used', '(', 'phone...
1	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	4	nice phone, nice up grade from my pantach revu...	0.0	positive	['nice', 'phone', ',', 'nice', 'up', 'grade', ...	['nice', 'phone', ',', 'nice', 'grade', 'panta...
2	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	5	very pleased	0.0	positive	['very', 'pleased']	['pleased']
3	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	4	it works good but it goes slow sometimes but i...	0.0	positive	['it', 'works', 'good', 'but', 'it', 'goes', '...	['works', 'good', 'goes', 'slow', 'sometimes',...
4	"clear clean esn" sprint epic 4g galaxy sph-d7...	samsung	199.99	4	great phone to replace my lost phone. the only...	0.0	positive	['great', 'phone', 'to', 'replace', 'my', 'los...	['great', 'phone', 'replace', 'lost', 'phone',...

In [13]:

```
from sklearn.model_selection import train_test_split

# Split the data into training (60%) and temporary data (40%)
X_train_temp, X_temp, Y_train_temp, Y_temp = train_test_split(data2['Lemmatized'], dat

# Split the temporary data into testing (50%) and validation (50%)
X_test, X_validation, Y_test, Y_validation = train_test_split(X_temp, Y_temp, test_si

print("Train:", X_train_temp.shape, Y_train_temp.shape)
print("Test:", X_test.shape, Y_test.shape)
print("Validation:", X_validation.shape, Y_validation.shape)
```

```
Train: (209682,) (209682,)
Test: (69894,) (69894,)
Validation: (69894,) (69894,)
```

In [14]:

```
#Using TF*IDF Vectorizer
```

In [19]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()
tf_x_train = vectorizer.fit_transform(X_train_temp)
tf_x_test = vectorizer.transform(X_test)
tf_x_validation = vectorizer.transform(X_validation)

print("TF-IDF transformation completed for training, testing, and validation data.")
```

```
TF-IDF transformation completed for training, testing, and validation data.
```

In [20]:

```
##model
```

In [23]:

```
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Assuming you have already defined and processed your tf_x_train and Y_train datasets
# Make sure your data is properly preprocessed and split into training and testing set

# Create and train the LinearSVC classifier
clf = LinearSVC(random_state=0)
clf.fit(tf_x_train, Y_train_temp)

# Predict on the test dataset
y_test_pred = clf.predict(tf_x_test)

# Calculate accuracy
accuracy = accuracy_score(Y_test, y_test_pred)

# Calculate precision, recall, and F1 score
precision = precision_score(Y_test, y_test_pred, average='weighted')
recall = recall_score(Y_test, y_test_pred, average='weighted')
f1 = f1_score(Y_test, y_test_pred, average='weighted')

print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
```

```
C:\Users\SACHIN\anaconda3\Lib\site-packages\sklearn\svm\_classes.py:32: FutureWarnin
g: The default value of `dual` will change from `True` to `'auto'` in 1.5. Set the va
lue of `dual` explicitly to suppress the warning.
  warnings.warn(
Accuracy: 0.8822216499270323
Precision: 0.8750275158356623
Recall: 0.8822216499270323
F1 Score: 0.8670074765251397
```

In [26]:

```
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

```python
# Create and train the LinearSVC classifier
clf = LinearSVC(random_state=0)
clf.fit(tf_x_train, Y_train_temp)

# Predict on the validation set
y_val_pred = clf.predict(tf_x_validation)

# Calculate accuracy on the validation set
accuracy_val = accuracy_score(Y_validation, y_val_pred)

# Calculate precision, recall, and F1 score on the validation set
precision_val = precision_score(Y_validation, y_val_pred, average='weighted')
recall_val = recall_score(Y_validation, y_val_pred, average='weighted')
f1_val = f1_score(Y_validation, y_val_pred, average='weighted')

print(f"Validation Accuracy: {accuracy_val}")
print(f"Validation Precision: {precision_val}")
print(f"Validation Recall: {recall_val}")
print(f"Validation F1 Score: {f1_val}")
```

```
C:\Users\SACHIN\anaconda3\Lib\site-packages\sklearn\svm\_classes.py:32: FutureWarnin
g: The default value of `dual` will change from `True` to `'auto'` in 1.5. Set the va
lue of `dual` explicitly to suppress the warning.
  warnings.warn(
Validation Accuracy: 0.8797178584713995
Validation Precision: 0.8730284814930125
Validation Recall: 0.8797178584713995
Validation F1 Score: 0.8644891018932364
```

In [27]:

```python
from sklearn.linear_model import LogisticRegression

# Create and train the Logistic Regression classifier
lr = LogisticRegression(random_state=0)
lr.fit(tf_x_train, Y_train_temp)

# Predict on the test dataset
y_test_pred = lr.predict(tf_x_test)

# Calculate accuracy
accuracy = accuracy_score(Y_test, y_test_pred)

# Calculate precision, recall, and F1 score
precision = precision_score(Y_test, y_test_pred, average='weighted')
recall = recall_score(Y_test, y_test_pred, average='weighted')
f1 = f1_score(Y_test, y_test_pred, average='weighted')

print(f"Logistic Regression Test Accuracy: {accuracy}")
print(f"Logistic Regression Test Precision: {precision}")
print(f"Logistic Regression Test Recall: {recall}")
print(f"Logistic Regression Test F1 Score: {f1}")
```

```
Logistic Regression Test Accuracy: 0.8703036025982201
Logistic Regression Test Precision: 0.8539144128722995
Logistic Regression Test Recall: 0.8703036025982201
Logistic Regression Test F1 Score: 0.852573765650536
```

```
C:\Users\SACHIN\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:460: Co
nvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

In [28]:

```python
from sklearn.linear_model import LogisticRegression

# Create and train the Logistic Regression classifier
lr = LogisticRegression(random_state=0)
lr.fit(tf_x_train, Y_train_temp)

# Predict on the validation set
y_val_pred = lr.predict(tf_x_validation)

# Calculate accuracy on the validation set
accuracy_val = accuracy_score(Y_validation, y_val_pred)

# Calculate precision, recall, and F1 score on the validation set
precision_val = precision_score(Y_validation, y_val_pred, average='weighted')
recall_val = recall_score(Y_validation, y_val_pred, average='weighted')
f1_val = f1_score(Y_validation, y_val_pred, average='weighted')

print(f"Logistic Regression Validation Accuracy: {accuracy_val}")
print(f"Logistic Regression Validation Precision: {precision_val}")
print(f"Logistic Regression Validation Recall: {recall_val}")
print(f"Logistic Regression Validation F1 Score: {f1_val}")
```

```
Logistic Regression Validation Accuracy: 0.8681718030159956
Logistic Regression Validation Precision: 0.8511710541616365
Logistic Regression Validation Recall: 0.8681718030159956
Logistic Regression Validation F1 Score: 0.8501002450298084
```

```
C:\Users\SACHIN\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:460: Co
nvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

In [29]:

```python
from sklearn.ensemble import RandomForestClassifier

# Create and train the Random Forest classifier
rf = RandomForestClassifier(random_state=0)
rf.fit(tf_x_train, Y_train_temp)

# Predict on the test dataset
y_test_pred = rf.predict(tf_x_test)

# Calculate accuracy
accuracy = accuracy_score(Y_test, y_test_pred)

# Calculate precision, recall, and F1 score
precision = precision_score(Y_test, y_test_pred, average='weighted')
```

```
recall = recall_score(Y_test, y_test_pred, average='weighted')
f1 = f1_score(Y_test, y_test_pred, average='weighted')

print(f"Random Forest Test Accuracy: {accuracy}")
print(f"Random Forest Test Precision: {precision}")
print(f"Random Forest Test Recall: {recall}")
print(f"Random Forest Test F1 Score: {f1}")

# For the validation set, you can use the same code structure as the LinearSVC example
```

```
Random Forest Test Accuracy: 0.9231264486221993
Random Forest Test Precision: 0.9245049678548426
Random Forest Test Recall: 0.9231264486221993
Random Forest Test F1 Score: 0.9175294919843003
```

In [30]:

```
from sklearn.ensemble import RandomForestClassifier

# Create and train the Random Forest classifier
rf = RandomForestClassifier(random_state=0)
rf.fit(tf_x_train, Y_train_temp)

# Predict on the validation set
y_val_pred = rf.predict(tf_x_validation)

# Calculate accuracy on the validation set
accuracy_val = accuracy_score(Y_validation, y_val_pred)

# Calculate precision, recall, and F1 score on the validation set
precision_val = precision_score(Y_validation, y_val_pred, average='weighted')
recall_val = recall_score(Y_validation, y_val_pred, average='weighted')
f1_val = f1_score(Y_validation, y_val_pred, average='weighted')

print(f"Random Forest Validation Accuracy: {accuracy_val}")
print(f"Random Forest Validation Precision: {precision_val}")
print(f"Random Forest Validation Recall: {recall_val}")
print(f"Random Forest Validation F1 Score: {f1_val}")
```

```
Random Forest Validation Accuracy: 0.9223109279766504
Random Forest Validation Precision: 0.9235644512699824
Random Forest Validation Recall: 0.9223109279766504
Random Forest Validation F1 Score: 0.9170647620247578
```

In [31]:

```
from sklearn.naive_bayes import MultinomialNB

# Create and train the Naive Bayes (Multinomial) classifier
nb = MultinomialNB()
nb.fit(tf_x_train, Y_train_temp)

# Predict on the test dataset
y_test_pred = nb.predict(tf_x_test)

# Calculate accuracy
accuracy = accuracy_score(Y_test, y_test_pred)

# Calculate precision, recall, and F1 score
precision = precision_score(Y_test, y_test_pred, average='weighted')
recall = recall_score(Y_test, y_test_pred, average='weighted')
f1 = f1_score(Y_test, y_test_pred, average='weighted')

print(f"Naive Bayes Test Accuracy: {accuracy}")
print(f"Naive Bayes Test Precision: {precision}")
print(f"Naive Bayes Test Recall: {recall}")
print(f"Naive Bayes Test F1 Score: {f1}")

# For the validation set, you can use the same code structure as the LinearSVC example
```

```
Naive Bayes Test Accuracy: 0.8162360145362978
Naive Bayes Test Precision: 0.8162440552773624
Naive Bayes Test Recall: 0.8162360145362978
Naive Bayes Test F1 Score: 0.7754745045113425
```

In [32]:

```
from sklearn.naive_bayes import MultinomialNB

# Create and train the Naive Bayes (Multinomial) classifier
nb = MultinomialNB()
nb.fit(tf_x_train, Y_train_temp)

# Predict on the validation set
y_val_pred = nb.predict(tf_x_validation)

# Calculate accuracy on the validation set
accuracy_val = accuracy_score(Y_validation, y_val_pred)

# Calculate precision, recall, and F1 score on the validation set
precision_val = precision_score(Y_validation, y_val_pred, average='weighted')
recall_val = recall_score(Y_validation, y_val_pred, average='weighted')
f1_val = f1_score(Y_validation, y_val_pred, average='weighted')

print(f"Naive Bayes Validation Accuracy: {accuracy_val}")
print(f"Naive Bayes Validation Precision: {precision_val}")
print(f"Naive Bayes Validation Recall: {recall_val}")
print(f"Naive Bayes Validation F1 Score: {f1_val}")
```

```
Naive Bayes Validation Accuracy: 0.8164792399919879
Naive Bayes Validation Precision: 0.8240364742365126
Naive Bayes Validation Recall: 0.8164792399919879
Naive Bayes Validation F1 Score: 0.7753845266198462
```

In [ ]: