```
data = '''Holi is one of the most colorful and joyful festivals of India.
It is celebrated in the month of March and marks the arrival of spring.
People throw colors on each other and enjoy the day with music, dance, and sweets like gujiya.
The festival starts with Holika Dahan, which shows the victory of good over evil.
Holi teaches us love, friendship, and forgiveness.
On this day, people forget old fights and celebrate together.
Holi spreads happiness and brings people closer to each other.'''
```

```
Text data loaded into 'data' variable.
```

```
import nltk
import string
import re
from collections import Counter
from nltk.stem import WordNetLemmatizer
try:
    nltk.data.find('corpora/stopwords')
except LookupError:
    nltk.download('stopwords')
try:
    nltk.data.find('corpora/wordnet')
except LookupError:
    nltk.download('wordnet')
lower_cased_text = data.lower()
print("--- Lowercased Text ---")
print(lower_cased_text)
```

```
--- Lowercased Text ---
holi is one of the most colorful and joyful festivals of india.
it is celebrated in the month of march and marks the arrival of spring.
people throw colors on each other and enjoy the day with music, dance, and sweets like gujiya.
the festival starts with holika dahan, which shows the victory of good over evil.
holi teaches us love, friendship, and forgiveness.
on this day, people forget old fights and celebrate together.
holi spreads happiness and brings people closer to each other.
[nltk_data] Downloading package wordnet to /root/nltk_data...
```

```
import string
no_punctuation_text = lower_cased_text.translate(str.maketrans('', '', string.punctuation))
print("\n--- Text after Punctuation Removal ---")
print(no_punctuation_text)
```

```

--- Text after Punctuation Removal ---
holi is one of the most colorful and joyful festivals of india
```

```
it is celebrated in the month of march and marks the arrival of spring
people throw colors on each other and enjoy the day with music dance and sweets like gujiya
the festival starts with holika dahan which shows the victory of good over evil
holi teaches us love friendship and forgiveness
on this day people forget old fights and celebrate together
holi spreads happiness and brings people closer to each other
```

```
from nltk.tokenize import word_tokenize
tokens = word_tokenize(no_punctuation_text)
print("\n--- Tokenized Text ---")
print(tokens)
```

```
--- Tokenized Text ---
['holi', 'is', 'one', 'of', 'the', 'most', 'colorful', 'and', 'joyful', 'festivals', 'of', 'india', 'it', 'is', 'celebrated', 'in', 'the', 'month'
```

```
import re
no_numbers_tokens = [token for token in tokens if not token.isdigit()]
print("\n--- Tokens after Number Removal ---")
print(no_numbers_tokens)
```

```
--- Tokens after Number Removal ---
['holi', 'is', 'one', 'of', 'the', 'most', 'colorful', 'and', 'joyful', 'festivals', 'of', 'india', 'it', 'is', 'celebrated', 'in', 'the', 'month'
```

```
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
no_stopwords_tokens = [token for token in no_numbers_tokens if token not in stop_words]
print("\n--- Tokens after Stopword Removal ---")
print(no_stopwords_tokens)
```

```
--- Tokens after Stopword Removal ---
['holi', 'one', 'colorful', 'joyful', 'festivals', 'india', 'celebrated', 'month', 'march', 'marks', 'arrival', 'spring', 'people', 'throw', 'colo
```

```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
lemmatized_tokens = [lemmatizer.lemmatize(token) for token in no_stopwords_tokens]
print("\n--- Lemmatized Tokens ---")
print(lemmatized_tokens)
```

```
--- Lemmatized Tokens ---
['holi', 'one', 'colorful', 'joyful', 'festival', 'india', 'celebrated', 'month', 'march', 'mark', 'arrival', 'spring', 'people', 'throw', 'color'
```

```
Start coding or generate with AI.
```

```
import string
import nltk
from nltk.corpus import stopwords
from collections import Counter
try:
    nltk.data.find('corpora/stopwords')
except LookupError:
    nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
processed_tokens = []
for token in tokens:
    token_lower = token.lower()
    token_no_punct = token_lower.translate(str.maketrans('', '', string.punctuation))
    if token_no_punct and token_no_punct not in stop_words:
        processed_tokens.append(token_no_punct)
word_frequencies = Counter(processed_tokens)
print("Most frequent words (top 10):")
for word, count in word_frequencies.most_common(10):
    print(f"{word}: {count}")
```

```
Text processing complete. Word frequencies calculated.
lower case : .
No punctuation :
Total number of tokens: 97
Most frequent words (top 10):
holi: 3
people: 3
day: 2
one: 1
colorful: 1
joyful: 1
festivals: 1
india: 1
celebrated: 1
month: 1
```

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
processed_text_string = ' '.join(lemmatized_tokens)
print("\n--- Bag-of-Words Vectorization ---")
count_vectorizer = CountVectorizer()
bow_matrix = count_vectorizer.fit_transform([processed_text_string])
vocabulary_bow = count_vectorizer.get_feature_names_out()
print("Vocabulary (Bag-of-Words):", vocabulary_bow)
print("Bag-of-Words Matrix:\n", bow_matrix.toarray())
print("\n--- TF-IDF Vectorization ---")
tfidf_vectorizer = TfidfVectorizer()
```

```
tfidf_matrix = tfidf_vectorizer.fit_transform([processed_text_string])
vocabulary_tfidf = tfidf_vectorizer.get_feature_names_out()
print("Vocabulary (TF-IDF):", vocabulary_tfidf)
print("TF-IDF Matrix:\n", tfidf_matrix.toarray())
```

```
--- Bag-of-Words Vectorization ---
Vocabulary (Bag-of-Words): ['arrival' 'brings' 'celebrate' 'celebrated' 'closer' 'color' 'colorful'
 'dahan' 'dance' 'day' 'enjoy' 'evil' 'festival' 'fight' 'forget'
 'forgiveness' 'friendship' 'good' 'gujiya' 'happiness' 'holi' 'holika'
 'india' 'joyful' 'like' 'love' 'march' 'mark' 'month' 'music' 'old' 'one'
 'people' 'show' 'spread' 'spring' 'start' 'sweet' 'teach' 'throw'
 'together' 'victory']
Bag-of-Words Matrix:
 [[1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 3 1 1 1
  1 1 1 1 1 1]]

--- TF-IDF Vectorization ---
Vocabulary (TF-IDF): ['arrival' 'brings' 'celebrate' 'celebrated' 'closer' 'color' 'colorful'
 'dahan' 'dance' 'day' 'enjoy' 'evil' 'festival' 'fight' 'forget'
 'forgiveness' 'friendship' 'good' 'gujiya' 'happiness' 'holi' 'holika'
 'india' 'joyful' 'like' 'love' 'march' 'mark' 'month' 'music' 'old' 'one'
 'people' 'show' 'spread' 'spring' 'start' 'sweet' 'teach' 'throw'
 'together' 'victory']
TF-IDF Matrix:
 [[0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.25  0.125 0.125
  0.25  0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.375 0.125 0.125 0.125
  0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.375 0.125 0.125 0.125
  0.125 0.125 0.125 0.125 0.125 0.125]]
```

```
print("\n--- Feature Matrix Shapes ---")
print("Bag-of-Words Matrix Shape:", bow_matrix.shape)
print("TF-IDF Matrix Shape:", tfidf_matrix.shape)
print("\n--- Sample Feature Names (Bag-of-Words) ---")
print(vocabulary_bow[:5])
print("\n--- Sample Feature Names (TF-IDF) ---")
print(vocabulary_tfidf[:5])
```

```
--- Feature Matrix Shapes ---
Bag-of-Words Matrix Shape: (1, 42)
TF-IDF Matrix Shape: (1, 42)

--- Sample Feature Names (Bag-of-Words) ---
['arrival' 'brings' 'celebrate' 'celebrated' 'closer']

--- Sample Feature Names (TF-IDF) ---
['arrival' 'brings' 'celebrate' 'celebrated' 'closer']
```

```
from sklearn.model_selection import train_test_split
import numpy as np
```

```
num_samples = tfidf_matrix.shape[0]
y = np.zeros(num_samples, dtype=int)
if num_samples > 1:
    X_train, X_test, y_train, y_test = train_test_split(
        tfidf_matrix, y, test_size=0.2, random_state=42, stratify=y
    )
else:
    print("Warning: Only one sample found. Train-test split will not be meaningful for evaluation. Assigning all data to training set.")
    X_train, y_train = tfidf_matrix, y
    X_test = np.array([]).reshape(0, tfidf_matrix.shape[1] if tfidf_matrix.shape[0] > 0 else 0)
    y_test = np.array([])

print("--- Train-Test Split Details ---")
print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"y_test shape: {y_test.shape}")
```

```
Warning: Only one sample found. Train-test split will not be meaningful for evaluation. Assigning all data to training set.
--- Train-Test Split Details ---
X_train shape: (1, 42)
X_test shape: (0, 42)
y_train shape: (1,)
y_test shape: (0,)
```

```
from sklearn.naive_bayes import MultinomialNB
naive_bayes_model = MultinomialNB()
naive_bayes_model.fit(X_train, y_train)
print("--- Naive Bayes Model Trained ---")
print("Model parameters:")
print(f"  Class log probabilities (prior): {naive_bayes_model.class_log_prior_}")
print(f"  Feature log probabilities given class (conditional probability):\n{naive_bayes_model.feature_log_prob_}")
print(f"  Number of training samples encountered for each class: {naive_bayes_model.class_count_}")
print(f"  Number of samples encountered for each (feature, class) combination: {naive_bayes_model.feature_count_}")
```

```
--- Naive Bayes Model Trained ---
Model parameters:
  Class log probabilities (prior): [0.]
  Feature log probabilities given class (conditional probability):
[[-3.75341798 -3.75341798 -3.75341798 -3.75341798 -3.75341798 -3.75341798
  -3.75341798 -3.75341798 -3.75341798 -3.64805746 -3.75341798 -3.75341798
  -3.64805746 -3.75341798 -3.75341798 -3.75341798 -3.75341798 -3.75341798
  -3.75341798 -3.75341798 -3.55274728 -3.75341798 -3.75341798 -3.75341798
  -3.75341798 -3.75341798 -3.75341798 -3.75341798 -3.75341798 -3.75341798
  -3.75341798 -3.75341798 -3.55274728 -3.75341798 -3.75341798 -3.75341798
  -3.75341798 -3.75341798 -3.75341798 -3.75341798 -3.75341798 -3.75341798]]
  Number of training samples encountered for each class: [1.]
  Number of samples encountered for each (feature, class) combination: [[0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.25  0.125 0.125
  0.25  0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.375 0.125 0.125 0.125
  0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.375 0.125 0.125 0.125
  0.125 0.125 0.125 0.125 0.125 0.125]]
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
print("--- Model Evaluation ---")
if X_test.shape[0] == 0:
    print("No test data available for evaluation as only one sample was provided.")
    print("To perform a meaningful evaluation, please use a dataset with multiple samples.")
else:
    y_pred = naive_bayes_model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average='weighted', zero_division=0)
    recall = recall_score(y_test, y_pred, average='weighted', zero_division=0)
    f1 = f1_score(y_test, y_pred, average='weighted', zero_division=0)
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1-Score: {f1:.4f}")
    print("\n--- Classification Report ---")
    target_names = [f'Class {i}' for i in np.unique(y_test)]
    print(classification_report(y_test, y_pred, target_names=target_names, zero_division=0))
    print("\n--- Confusion Matrix ---")
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(6, 5))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=target_names, yticklabels=target_names)
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title('Confusion Matrix')
    plt.show()
```

```
--- Model Evaluation ---
No test data available for evaluation as only one sample was provided.
To perform a meaningful evaluation, please use a dataset with multiple samples.
```