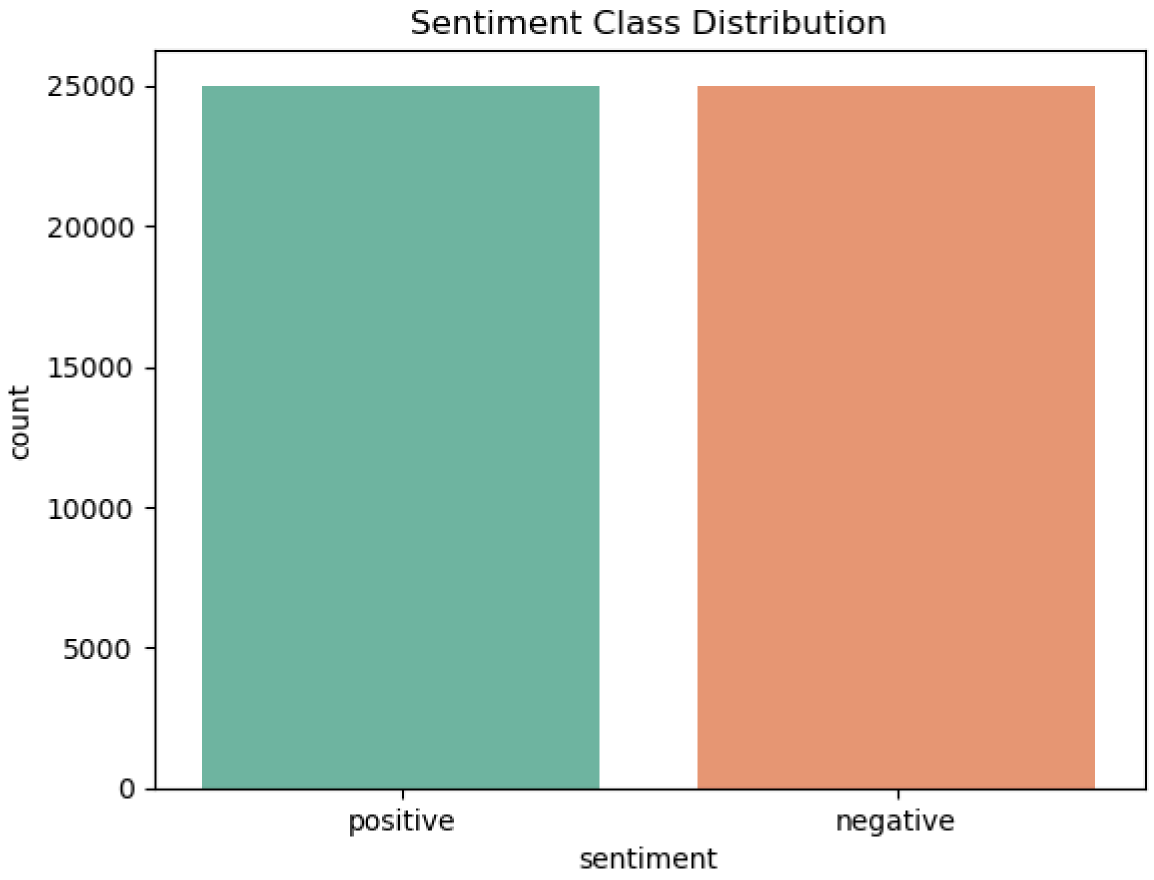
In [1]:

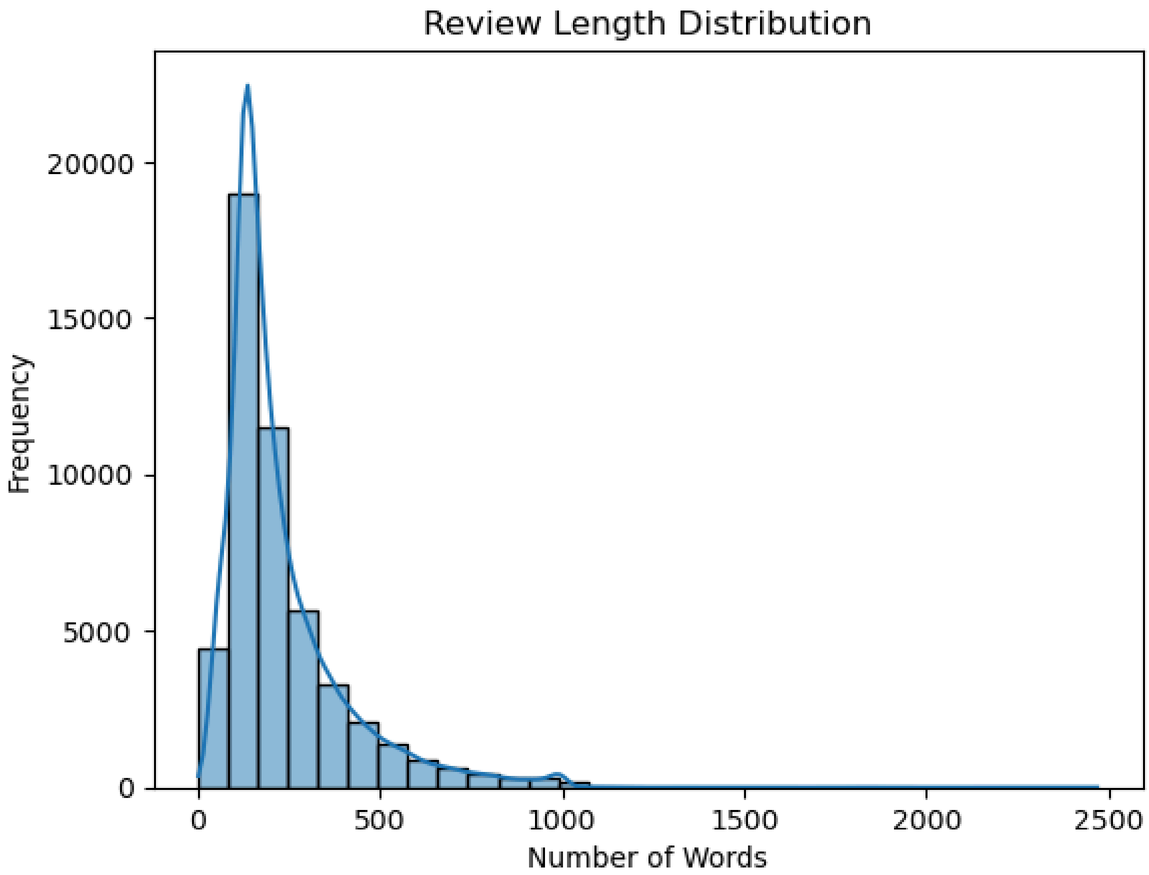
```
# Import required libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
import re
import string
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

# NLTK downloads (only needed once)
nltk.download('stopwords')
nltk.download('wordnet')

# Load the dataset
file_path = r"C:\Users\AkshS\OneDrive\Desktop\Imdb.xlsx"
df = pd.read_excel(file_path)

# Display the first few rows
print("Sample Data:")
print(df.head())

# Basic info
print("\nDataset Info:")
print(df.info())

# Check for missing values
print("\nMissing Values:")
print(df.isnull().sum())

# Class distribution
print("\nSentiment Distribution:")
print(df['sentiment'].value_counts())

# Review length analysis
df['review_length'] = df['review'].apply(lambda x: len(str(x).split()))
print("\nReview Length Description:")
print(df['review_length'].describe())

# Visualization: Sentiment count
sns.countplot(x='sentiment', data=df, palette='Set2')
plt.title('Sentiment Class Distribution')
plt.show()

# Visualization: Review lengths
sns.histplot(df['review_length'], bins=30, kde=True)
plt.title('Review Length Distribution')
plt.xlabel('Number of Words')
plt.ylabel('Frequency')
plt.show()

# Clean the reviews
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def clean_text(text):
    text = text.lower()
    text = re.sub(r'<.*?>', '', text)  # remove HTML
    text = text.translate(str.maketrans('', '', string.punctuation))  # remove p
    text = re.sub(r'\d+', '', text)  # remove numbers
    words = text.split()
    words = [lemmatizer.lemmatize(word) for word in words if word not in stop_wo
    return ' '.join(words)

df['clean_review'] = df['review'].astype(str).apply(clean_text)

print("\nOriginal vs Cleaned:")
print(df[['review', 'clean_review']].head())
```

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

```
Sample Data:
                                              review sentiment
0  One of the other reviewers has mentioned that ...  positive
1  A wonderful little production. <br /><br />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

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   review     50000 non-null  object
 1   sentiment  50000 non-null  object
dtypes: object(2)
memory usage: 781.4+ KB
None

Missing Values:
review       0
sentiment    0
dtype: int64

Sentiment Distribution:
sentiment
positive    25000
negative    25000
Name: count, dtype: int64

Review Length Description:
count    50000.000000
mean       231.137900
std        171.339334
min          1.000000
25%        126.000000
50%        173.000000
75%        280.000000
max       2470.000000
Name: review_length, dtype: float64
```

```
C:\Users\AkshS\AppData\Local\Temp\ipykernel_9176\516829515.py:41: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.countplot(x='sentiment', data=df, palette='Set2')
```





```
Original vs Cleaned:
                                              review  \
0  One of the other reviewers has mentioned that ...
1  A wonderful little production. <br /><br />The...
2  I thought this was a wonderful way to spend ti...
3  Basically there's a family where a little boy ...
4  Petter Mattei's "Love in the Time of Money" is...

                                        clean_review
0  one reviewer mentioned watching oz episode you...
1  wonderful little production filming technique ...
2  thought wonderful way spend time hot summer we...
3  basically there family little boy jake think t...
4  petter matteis love time money visually stunni...
```

In [2]:

```python
from sklearn.feature_extraction.text import TfidfVectorizer

# TF-IDF vectorization
vectorizer = TfidfVectorizer(max_features=5000)
X = vectorizer.fit_transform(df['clean_review'])

print("TF-IDF Matrix Shape:", X.shape)

# Add simple textual features
df['char_count'] = df['review'].apply(lambda x: len(str(x)))
df['avg_word_length'] = df['char_count'] / df['review_length']

print("\nTextual Feature Samples:")
print(df[['review_length', 'char_count', 'avg_word_length']].head())
```

```
TF-IDF Matrix Shape: (50000, 5000)

Textual Feature Samples:
   review_length  char_count  avg_word_length
0            307        1761         5.736156
1            162         998         6.160494
2            166         926         5.578313
3            138         748         5.420290
4            230        1317         5.726087
```

In [5]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC

# Convert target labels
y = df['sentiment']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

# Logistic Regression
lr = LogisticRegression()
lr.fit(X_train, y_train)

# Naive Bayes
nb = MultinomialNB()
nb.fit(X_train, y_train)

# Support Vector Machine
svm = LinearSVC()
svm.fit(X_train, y_train)
```

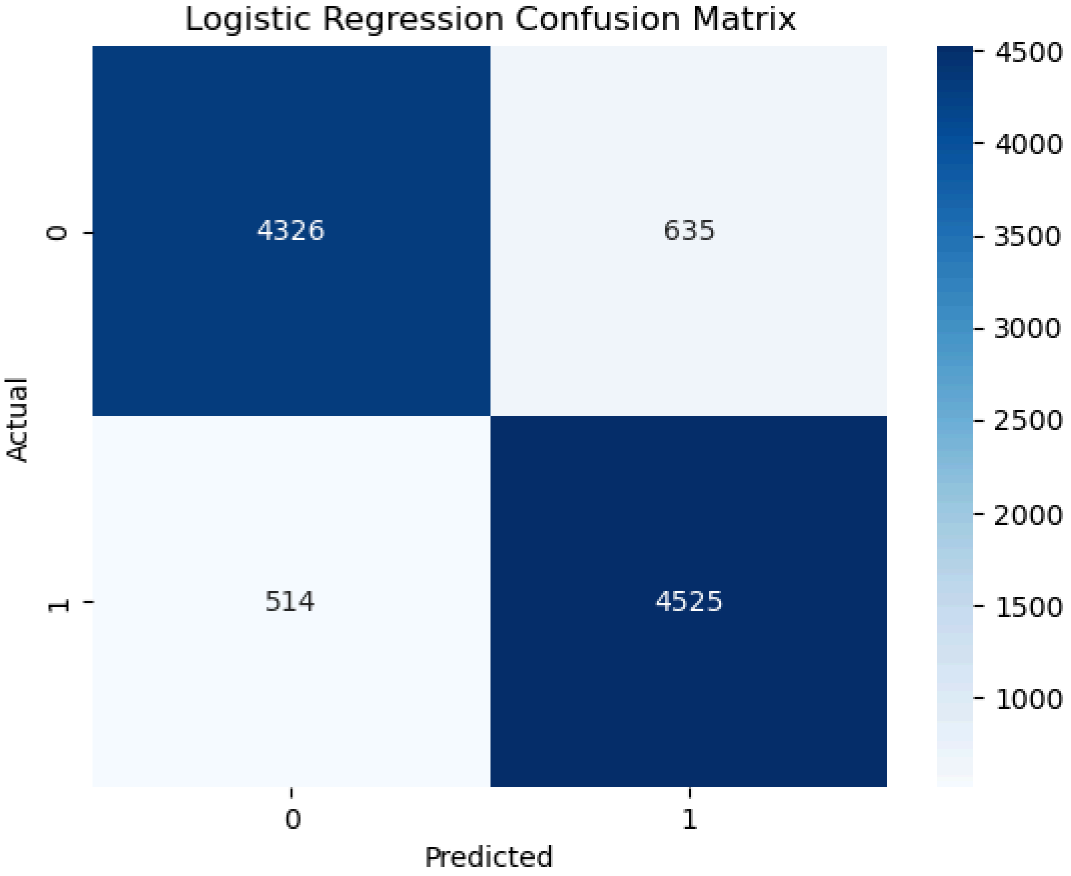
Out[5]:

```
▾ LinearSVC ⓘ ?

LinearSVC()
```

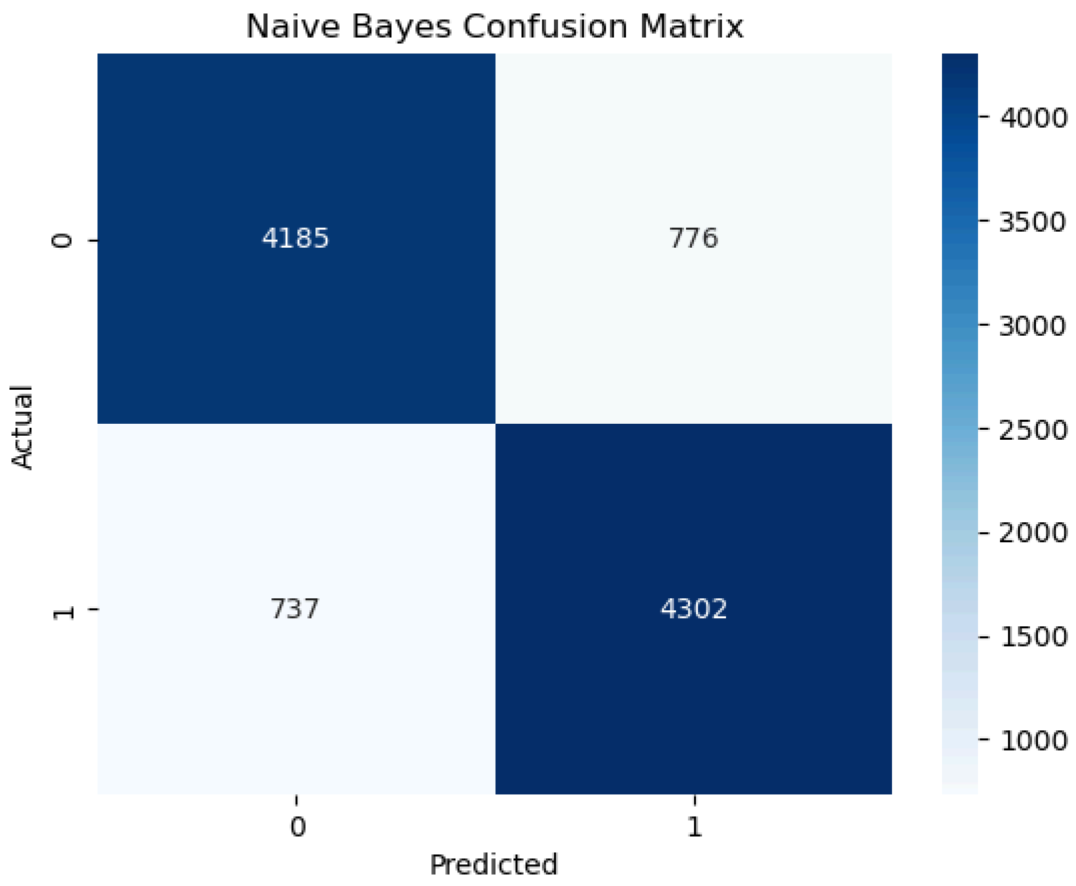
In [7]:

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_sc
import seaborn as sns

# Function to evaluate models
def evaluate_model(model, name):
    print(f"\n{name} Model Evaluation:")
    y_pred = model.predict(X_test)
    print(classification_report(y_test, y_pred))
    print("Accuracy:", accuracy_score(y_test, y_pred))
    cm = confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(f'{name} Confusion Matrix')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()

# Evaluate all models
evaluate_model(lr, "Logistic Regression")
evaluate_model(nb, "Naive Bayes")
evaluate_model(svm, "SVM")
```

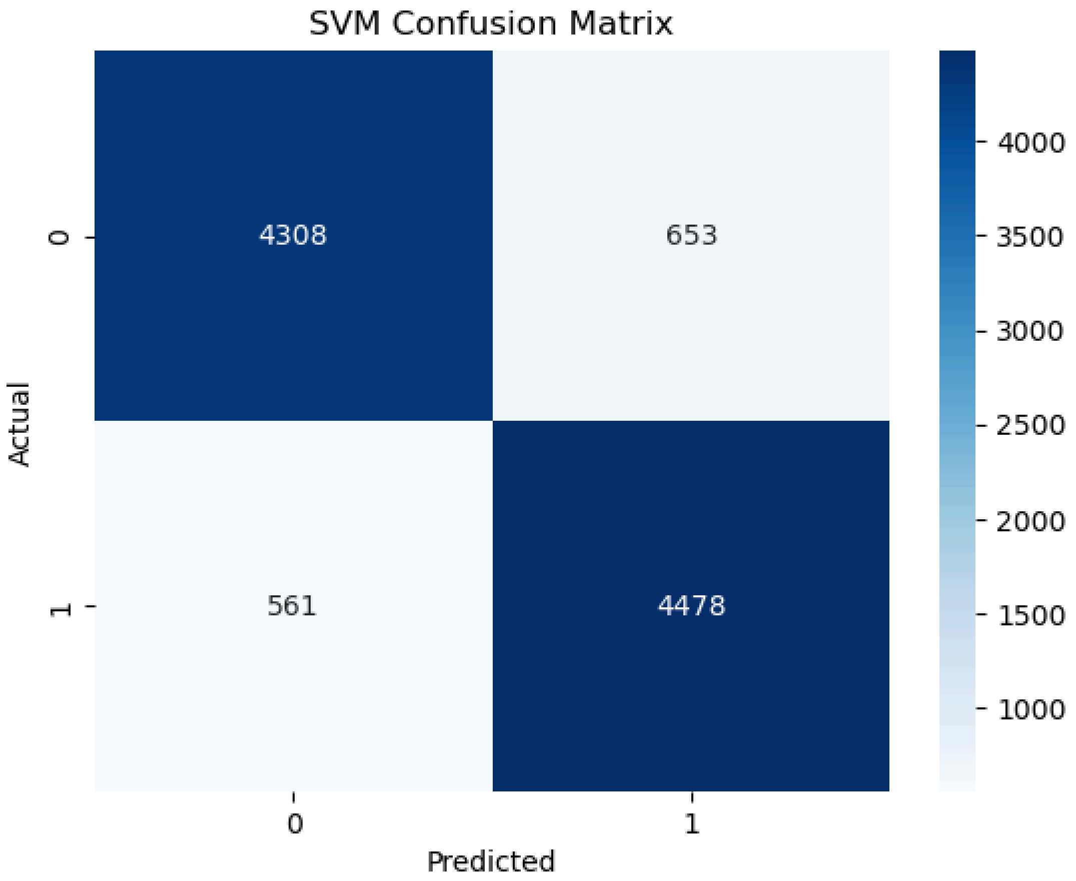
```
Logistic Regression Model Evaluation:
              precision    recall  f1-score   support

    negative       0.89      0.87      0.88      4961
    positive       0.88      0.90      0.89      5039

    accuracy                           0.89     10000
   macro avg       0.89      0.88      0.89     10000
weighted avg       0.89      0.89      0.89     10000

Accuracy: 0.8851
```



```
Naive Bayes Model Evaluation:
              precision    recall  f1-score   support

    negative       0.85      0.84      0.85      4961
    positive       0.85      0.85      0.85      5039

    accuracy                           0.85     10000
   macro avg       0.85      0.85      0.85     10000
weighted avg       0.85      0.85      0.85     10000

Accuracy: 0.8487
```



```
SVM Model Evaluation:
              precision    recall  f1-score   support

    negative       0.88      0.87      0.88      4961
    positive       0.87      0.89      0.88      5039

    accuracy                           0.88     10000
   macro avg       0.88      0.88      0.88     10000
weighted avg       0.88      0.88      0.88     10000

Accuracy: 0.8786
```


SVM Confusion Matrix
Actual
Predicted
0
1
0
1
4308
653
561
4478
4000
3500
3000
2500
2000
1500
1000

```
In [ ]:
```

```
In [ ]:
```