TEXT CLASSIFICATION

In [1]:
```
import nltk
from nltk.corpus import movie_reviews
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

Define a function for text preprocessing

In [2]:
```
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()
def preprocess(text):
    # Tokenize the text
    tokens = nltk.word_tokenize(text.lower())
    # Remove stop words and non-alphabetic characters
    tokens = [token for token in tokens if token.isalpha() and token not in stop_words]
    # Lemmatize the tokens
    tokens = [lemmatizer.lemmatize(token) for token in tokens]
    # Return the preprocessed text as a string
    return ' '.join(tokens)
```

Load the movie reviews dataset

In [3]:
```
documents = [(list(movie_reviews.words(fileid)), category)
             for category in movie_reviews.categories()
             for fileid in movie_reviews.fileids(category)]

preprocessed_documents = [(preprocess(' '.join(text)), label) for (text, label) in documents]
```

Feature Extraction (Use the CountVectorizer create bag-of-words features)

In [4]:
```
vectorizer = CountVectorizer()
X = vectorizer.fit_transform([text for text, _ in preprocessed_documents])
y = [label for _, label in preprocessed_documents]
```

NAIVE BAYES

In [5]:
```
# Step 3: Training
# Use Multinomial Naive Bayes to train the model
classifier = MultinomialNB()
classifier.fit(X, y)
```

Out[5]:
```
MultinomialNB()
```

In [6]:
```
# Step 4: Testing
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Fit the classifier on the training set
classifier.fit(X_train, y_train)
# Make predictions on the testing set
y_pred = classifier.predict(X_test)
# Calculate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)
```

```
Accuracy: 0.8075
```

In [7]:
```
# Step 5: Predictions
# Predict the sentiment of new reviews
new_reviews = [
    'This movie was amazing!',
    'I hated this movie.',
    'The movie was just okay.',
    'I absolutely love this movie!',
    'I have mixed feelings about this movie.',
    'This movie is terrible.',
    'I highly recommend this movie.',
    'I wouldn\'t watch this movie again.',
    'I don\'t know how to feel about this movie.',
    'This is the best movie ever!'
]
preprocessed_new_reviews = [preprocess(review) for review in new_reviews]
X_new = vectorizer.transform(preprocessed_new_reviews)
y_new_pred = classifier.predict(X_new)
for review, sentiment in zip(new_reviews, y_new_pred):
    print(review, sentiment)
```

```
This movie was amazing! pos
I hated this movie. neg
The movie was just okay. neg
I absolutely love this movie! neg
I have mixed feelings about this movie. pos
This movie is terrible. neg
I highly recommend this movie. pos
I wouldn't watch this movie again. neg
I don't know how to feel about this movie. neg
This is the best movie ever! pos
```

LOGISTIC REGGRESSION

In [8]:
```
# Step 3: Training
# Use Logistic Regression to train the model
classifier = LogisticRegression(solver='lbfgs', max_iter=1000)
classifier.fit(X, y)
```

Out[8]:
```
LogisticRegression(max_iter=1000)
```

In [9]:
```
# Step 3: Training
# Use Logistic Regression to train the model
classifier = LogisticRegression()
classifier.fit(X, y)

# Step 4: Testing
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Fit the classifier on the training set
classifier.fit(X_train, y_train)
# Make predictions on the testing set
y_pred = classifier.predict(X_test)
# Calculate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)

# Step 5: Predictions
# Predict the sentiment of new reviews
new_reviews = [
    'This movie was amazing!',
    'I hated this movie.',
    'The movie was just okay.',
    'I absolutely love this movie!',
    'I have mixed feelings about this movie.',
    'This movie is terrible.',
    'I highly recommend this movie.',
    'I wouldn\'t watch this movie again.',
    'I don\'t know how to feel about this movie.',
    'This is the best movie ever!'
]
preprocessed_new_reviews = [preprocess(review) for review in new_reviews]
X_new = vectorizer.transform(preprocessed_new_reviews)
y_new_pred = classifier.predict(X_new)
for review, sentiment in zip(new_reviews, y_new_pred):
    print(review, sentiment)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarn
ing: 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 (https://scikit-learn.org/stable/m
odules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit
-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(

Accuracy: 0.8175
This movie was amazing! neg
I hated this movie. neg
The movie was just okay. neg
I absolutely love this movie! neg
I have mixed feelings about this movie. neg
This movie is terrible. neg
I highly recommend this movie. neg
I wouldn't watch this movie again. neg
I don't know how to feel about this movie. neg
This is the best movie ever! neg

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarn
ing: 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 (https://scikit-learn.org/stable/m
odules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit
-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

In [10]:
```
new_reviews = [
    'This movie was amazing!',
    'I hated this movie.',
    'The movie was just okay.',
    'I absolutely love this movie!',
    'I have mixed feelings about this movie.',
    'This movie is terrible.',
    'I highly recommend this movie.',
    'I wouldn\'t watch this movie again.',
    'I don\'t know how to feel about this movie.',
    'This is the best movie ever!'
]
preprocessed_new_reviews = [preprocess(review) for review in new_reviews]
X_new = vectorizer.transform(preprocessed_new_reviews)
y_new_pred = classifier.predict(X_new)
for review, sentiment in zip(new_reviews, y_new_pred):
    print(review, sentiment)
```

```
This movie was amazing! neg
I hated this movie. neg
The movie was just okay. neg
I absolutely love this movie! neg
I have mixed feelings about this movie. neg
This movie is terrible. neg
I highly recommend this movie. neg
I wouldn't watch this movie again. neg
I don't know how to feel about this movie. neg
This is the best movie ever! neg
```