In [21]:
```
import nltk
import os
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix

# Setup NLTK data path
nltk_data_dir = '/Users/shivrajchaudar/nltk_data'
nltk.data.path.append(nltk_data_dir)

# Downloads
nltk.download('punkt', download_dir=nltk_data_dir)
nltk.download('stopwords', download_dir=nltk_data_dir)
nltk.download('wordnet', download_dir=nltk_data_dir)
nltk.download('averaged_perceptron_tagger', download_dir=nltk_data_dir)

print("NLTK Data Files:", os.listdir(nltk_data_dir))
```

```
NLTK Data Files: ['tokenizers', 'taggers', 'corpora']
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/shivrajchaudar/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/shivrajchaudar/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/shivrajchaudar/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /Users/shivrajchaudar/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

In [22]:
```
# Sample text for tokenization and POS tagging
text = """Albert Einstein was a theoretical physicist who developed the
He was born in Germany in 1879 and is considered one of the most influen
```

In [23]:
```
# Sentence Tokenization
sentences = sent_tokenize(text)
print("Sentence Tokenization:", sentences)

# Word Tokenization
words = word_tokenize(text)
print("Word Tokenization:", words)
```

```
Sentence Tokenization: ['Albert Einstein was a theoretical physicist who d
eveloped the theory of relativity.', 'He was born in Germany in 1879 and i
s considered one of the most influential physicists.']
Word Tokenization: ['Albert', 'Einstein', 'was', 'a', 'theoretical', 'phys
icist', 'who', 'developed', 'the', 'theory', 'of', 'relativity', '.', 'H
e', 'was', 'born', 'in', 'Germany', 'in', '1879', 'and', 'is', 'considere
d', 'one', 'of', 'the', 'most', 'influential', 'physicists', '.']
```

In [24]:

```
# Stopwords Removal
stop_words = set(stopwords.words('english'))
filtered_words = [w for w in words if w.lower() not in stop_words]
print("Filtered Words:", filtered_words)
```

```
Filtered Words: ['Albert', 'Einstein', 'theoretical', 'physicist', 'develo
ped', 'theory', 'relativity', '.', 'born', 'Germany', '1879', 'considere
d', 'one', 'influential', 'physicists', '.']
```

In [25]:

```
# Stemming
stemmer = PorterStemmer()
stemmed = [stemmer.stem(w) for w in filtered_words]
print("Stemmed Words:", stemmed)

# Lemmatization
lemmatizer = WordNetLemmatizer()
lemmatized = [lemmatizer.lemmatize(w) for w in filtered_words]
print("Lemmatized Words:", lemmatized)
```

```
Stemmed Words: ['albert', 'einstein', 'theoret', 'physicist', 'develop', '
theori', 'rel', '.', 'born', 'germani', '1879', 'consid', 'one', 'influent
i', 'physicist', '.']
Lemmatized Words: ['Albert', 'Einstein', 'theoretical', 'physicist', 'deve
loped', 'theory', 'relativity', '.', 'born', 'Germany', '1879', 'considere
d', 'one', 'influential', 'physicist', '.']
```

In [26]:

```
# POS Tagging
pos_tags = nltk.pos_tag(filtered_words)
print("POS Tagging:", pos_tags)
```

```
POS Tagging: [('Albert', 'NNP'), ('Einstein', 'NNP'), ('theoretical', 'J
J'), ('physicist', 'NN'), ('developed', 'VBD'), ('theory', 'JJ'), ('relati
vity', 'NN'), ('.', '.'), ('born', 'VBN'), ('Germany', 'NNP'), ('1879', 'C
D'), ('considered', 'VBD'), ('one', 'CD'), ('influential', 'JJ'), ('physic
ists', 'NNS'), ('.', '.')]
```

In [27]:

```
# Sample dataset
docs = [
    # Class 0 – Einstein / related
    "Einstein developed the theory of relativity.",
    "He was a physicist from Germany.",
    "Einstein received the Nobel Prize in Physics.",
    "Einstein's work led to the development of quantum theory.",
    "The photoelectric effect was explained by Einstein.",
    "Einstein's ideas influenced modern physics.",
    "He is one of the most famous scientists in history.",
```

```
    "Einstein challenged Newtonian physics.",
    "He was known for his wild hair and deep thoughts.",
    "Einstein published many scientific papers.",

    # Class 1 — General Physics
    "Physics is the study of matter and energy.",
    "Relativity changed how we understand space and time.",
    "Quantum mechanics is another area of physics.",
    "The laws of motion were developed by Newton.",
    "Thermodynamics deals with heat and temperature.",
    "Electromagnetism describes electric and magnetic fields.",
    "Particle physics explores subatomic particles.",
    "Classical physics describes everyday physical phenomena.",
    "Gravity is a fundamental force in the universe.",
    "Light can behave as both a wave and a particle."
]

labels = [0]*10 + [1]*10

from sklearn.linear_model import LogisticRegression
model = LogisticRegression()

# Preprocess function for classification
def preprocess(text):
    words = word_tokenize(text)
    words = [w for w in words if w.lower() not in stop_words]
    return ' '.join([lemmatizer.lemmatize(w.lower()) for w in words])

# Apply preprocessing
docs_cleaned = [preprocess(doc) for doc in docs]

# TF-IDF Vectorization
vectorizer = TfidfVectorizer(ngram_range=(1, 2), stop_words='english')
X = vectorizer.fit_transform(docs_cleaned)
print("TF-IDF Vocabulary:", vectorizer.get_feature_names_out())
```

```
TF-IDF Vocabulary: ['area' 'area physic' 'behave' 'behave wave' 'challenge
d'
 'challenged newtonian' 'changed' 'changed understand' 'classical'
 'classical physic' 'deal' 'deal heat' 'deep' 'deep thought' 'describes'
 'describes electric' 'describes everyday' 'developed' 'developed newton'
 'developed theory' 'development' 'development quantum' 'effect'
 'effect explained' 'einstein' 'einstein challenged' 'einstein developed'
 'einstein idea' 'einstein published' 'einstein received' 'einstein work'
 'electric' 'electric magnetic' 'electromagnetism'
 'electromagnetism describes' 'energy' 'everyday' 'everyday physical'
 'explained' 'explained einstein' 'explores' 'explores subatomic' 'famous'
 'famous scientist' 'field' 'force' 'force universe' 'fundamental'
 'fundamental force' 'germany' 'gravity' 'gravity fundamental' 'hair'
 'hair deep' 'heat' 'heat temperature' 'history' 'idea' 'idea influenced'
 'influenced' 'influenced modern' 'known' 'known wild' 'law' 'law motion'
 'led' 'led development' 'light' 'light behave' 'magnetic'
 'magnetic field' 'matter' 'matter energy' 'mechanic' 'mechanic area'
 'modern' 'modern physic' 'motion' 'motion developed' 'newton' 'newtonian'
 'newtonian physic' 'nobel' 'nobel prize' 'paper' 'particle'
 'particle physic' 'phenomenon' 'photoelectric' 'photoelectric effect'
 'physic' 'physic describes' 'physic explores' 'physic study' 'physical'
 'physical phenomenon' 'physicist' 'physicist germany' 'prize'
 'prize physic' 'published' 'published scientific' 'quantum'
 'quantum mechanic' 'quantum theory' 'received' 'received nobel'
 'relativity' 'relativity changed' 'scientific' 'scientific paper'
 'scientist' 'scientist history' 'space' 'space time' 'study'
 'study matter' 'subatomic' 'subatomic particle' 'temperature' 'theory'
 'theory relativity' 'thermodynamics' 'thermodynamics deal' 'thought'
 'time' 'understand' 'understand space' 'universe' 'wave' 'wave particle'
 'wild' 'wild hair' 'work' 'work led']
```

In [28]:

```python
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, labels, test_size=0.33, random_state=42, stratify=labels)

# Model Training
model = MultinomialNB()
model.fit(X_train, y_train)

# Prediction
y_pred = model.predict(X_test)

# Evaluation
print("\n--- Classification Results ---")
print("Predicted Labels:", y_pred)
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
--- Classification Results ---
Predicted Labels: [1 1 0 0 1 0 0]
Accuracy: 0.7142857142857143
Confusion Matrix:
 [[3 1]
 [1 2]]
Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.75      0.75         4
           1       0.67      0.67      0.67         3

    accuracy                           0.71         7
   macro avg       0.71      0.71      0.71         7
weighted avg       0.71      0.71      0.71         7
```