In [9]:
```
import nltk
import string
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import warnings
warnings.simplefilter('ignore')
```

In [10]:
```
# Download necessary NLTK resources
nltk.download('punkt')
```

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

Out[10]:
```
True
```

In [11]:
```
# Download necessary NLTK resources
nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package punkt_tab to
[nltk_data]     C:\Users\abishek\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt_tab.zip.
```

Out[11]:
```
True
```

In [12]:
```
# Download necessary NLTK resources
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\abishek\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

Out[12]:
```
True
```

In [13]:
```
# Download necessary NLTK resources
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\abishek\AppData\Roaming\nltk_data...
```

Out[13]:
```
True
```

In [14]:
```
# Initialize lemmatizer and stopwords
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))
```

In [15]:
```
# Function to preprocess text
def preprocess_text(text):
    # Convert to lowercase
    text = text.lower()

    # Remove punctuation
    text = text.translate(str.maketrans('', '', string.punctuation))
    # Tokenize text
    tokens = word_tokenize(text)

    # Remove stopwords and lemmatize tokens
    processed_tokens = [
        lemmatizer.lemmatize(word) for word in tokens if word not in stop_words and
    ]

    # Join tokens back into a single string
    return ' '.join(processed_tokens)
```

In [16]:
```
# Sample text data
documents = [
    "The quick brown fox jumped over the lazy dog.",
    "I love programming in Python, especially for data analysis.",
    "Natural language processing is fascinating!"
]
```

In [17]:
```
# Preprocess each document
preprocessed_docs = [preprocess_text(doc) for doc in documents]
```

In [18]:
```
# Initialize TF-IDF Vectorizer
vectorizer = TfidfVectorizer()
```

In [19]:
```
# Fit and transform the preprocessed text data
tfidf_matrix = vectorizer.fit_transform(preprocessed_docs)
```

In [20]:
```
# Convert the TF-IDF matrix to an array for better readability
tfidf_array = tfidf_matrix.toarray()

# Get the feature names (vocabulary)
feature_names = vectorizer.get_feature_names_out()
```

In [21]:
```
# Print the TF-IDF matrix
print("TF-IDF Matrix:")
for i, doc in enumerate(tfidf_array):
    print(f"\nDocument {i + 1}:")
    for word, score in zip(feature_names, doc):
        if score > 0:  # Only print words with non-zero TF-IDF scores
            print(f"{word}: {score}")
```

```
TF-IDF Matrix:

Document 1:
brown: 0.4082482904638631
dog: 0.4082482904638631
fox: 0.4082482904638631
jumped: 0.4082482904638631
lazy: 0.4082482904638631
quick: 0.4082482904638631

Document 2:
analysis: 0.4082482904638631
data: 0.4082482904638631
especially: 0.4082482904638631
love: 0.4082482904638631
programming: 0.4082482904638631
python: 0.4082482904638631

Document 3:
fascinating: 0.5
language: 0.5
natural: 0.5
processing: 0.5
```

In [ ]: