```
import pandas as pd        # data handling
import numpy as np         # numerical operations
import re                  # regex for text cleaning
import nltk                # tokenization, stopwords
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.decomposition import LatentDirichletAllocation, NMF
import matplotlib.pyplot as plt

# Why each library:
# pandas → load and manage dataset
# numpy → numerical operations
# re → regex-based cleaning
# nltk → stopwords, tokenization
# sklearn.feature_extraction → Bag-of-Words & TF-IDF matrices
# sklearn.decomposition → LDA & NMF models
# matplotlib → optional visualization of topics
```

Gemini

```
 # Example: load research abstracts (
 data = pd.read_csv("research_abstrac
 # Create a dummy CSV file for demons
 import os
 if not os.path.exists('research_abst
     dummy_data = {'abstract': ['This
     pd.DataFrame(dummy_data).to_csv(

 data = pd.read_csv('research_abstrac
 documents = data['abstract'].dropna(

 print("Number of documents:", len(dc
 print("Sample document:\n", document
 print('Number of documents:', len(dc
 print('Sample document:\n', document
```

```
Number of documents: 4
Sample document:
 This is an abstract about topic modeling with LDA.
```

```
nltk.download('stopwords')
stop_words = set(nltk.corpus.stopwords.words('english'))

def preprocess(text):
    text = text.lower()                          # lowercase
    text = re.sub(r'[^a-z\s]', '', text)         # remove punctuation/numbers
    tokens = [w for w in text.split() if w not in stop_words]  # remove stopwords
    return " ".join(tokens)

cleaned_docs = [preprocess(doc) for doc in documents]
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
# Bag-of-Words for LDA
count_vectorizer = CountVectorizer(max_df=0.95, min_df=2)
count_data = count_vectorizer.fit_transform(cleaned_docs)

# TF-IDF for NMF
```

```
tfidf_vectorizer = TfidfVectorizer(max_df=0.95, min_df=2)
```

```
lda = LatentDirichletAllocation(n_components=5, random_state=42)
lda.fit(count_data)

# Extract topics
def display_topics(model, feature_names, no_top_words):
    for idx, topic in enumerate(model.components_):
        print("Topic %d:" % (idx))
        print([feature_names[i] for i in topic.argsort()[:-no_top_words - 1:-1]])

display_topics(lda, count_vectorizer.get_feature_names_out(), 10)
```

```
Topic 0:
['abstract', 'document']
Topic 1:
['abstract', 'document']
Topic 2:
['abstract', 'document']
Topic 3:
['document', 'abstract']
Topic 4:
['abstract', 'document']
```

```
nmf = NMF(n_components=5, random_state=42)
nmf.fit(tfidf_data)

display_topics(nmf, tfidf_vectorizer.get_feature_names_out(), 10)
```

```
Topic 0:
['abstract', 'document']
Topic 1:
['document', 'abstract']
Topic 2:
['document', 'abstract']
Topic 3:
['abstract', 'document']
Topic 4:
['document', 'abstract']
```