```
#Step 1: Import Required Libraries
import nltk
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from nltk.corpus import movie_reviews, stopwords
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
#Step 2: Download Required NLTK Resources
nltk.download('movie_reviews')
nltk.download('punkt')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package movie_reviews to /root/nltk_data...
[nltk_data]   Unzipping corpora/movie_reviews.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
#Step 3: Load Positive and Negative Reviews
positive_reviews = []
negative_reviews = []

for fileid in movie_reviews.fileids('pos'):
    positive_reviews.append(movie_reviews.raw(fileid))

for fileid in movie_reviews.fileids('neg'):
    negative_reviews.append(movie_reviews.raw(fileid))

print("Positive reviews:", len(positive_reviews))
print("Negative reviews:", len(negative_reviews))
```

```
Positive reviews: 1000
Negative reviews: 1000
```

```
# Step 4: Text Preprocessing Function
# Includes:
# - Lowercasing
# - Tokenization
# - Stopword removal
stop_words = set(stopwords.words('english'))

def preprocess(text):
    tokens = word_tokenize(text.lower())
```

```
    tokens = [word for word in tokens if word.isalpha() and word not in stop_wo
    return " ".join(tokens)
```

```
# Step 5: Apply Preprocessing
positive_cleaned = [preprocess(review) for review in positive_reviews]
negative_cleaned = [preprocess(review) for review in negative_reviews]
```

```
#Step 6: Create Separate TF-IDF Models
tfidf_pos = TfidfVectorizer(max_features=2000)
tfidf_neg = TfidfVectorizer(max_features=2000)

pos_tfidf_matrix = tfidf_pos.fit_transform(positive_cleaned)
neg_tfidf_matrix = tfidf_neg.fit_transform(negative_cleaned)
```

```
#Step 7: Extract Top 15 TF-IDF Terms
pos_scores = np.mean(pos_tfidf_matrix.toarray(), axis=0)
pos_terms = tfidf_pos.get_feature_names_out()

pos_df = pd.DataFrame({'Term': pos_terms, 'Score': pos_scores})
top_pos = pos_df.sort_values(by='Score', ascending=False).head(15)
top_pos
```

	Term	Score
657	film	0.079054
1167	movie	0.050451
1246	one	0.046418
1022	like	0.032546
1688	story	0.027380
767	good	0.026163
1019	life	0.025340
51	also	0.025163
1801	time	0.024804
572	even	0.024066
276	character	0.023952
278	characters	0.023878
1983	would	0.023693
1937	well	0.022962
1860	two	0.022796

Next steps: Generate code with top_pos New interactive sheet

```
# Negative Reviews
neg_scores = np.mean(neg_tfidf_matrix.toarray(), axis=0)
neg_terms = tfidf_neg.get_feature_names_out()

neg_df = pd.DataFrame({'Term': neg_terms, 'Score': neg_scores})
top_neg = neg_df.sort_values(by='Score', ascending=False).head(15)
top_neg
```

	Term	Score
651	film	0.075606
1165	movie	0.061308
1240	one	0.047131
1032	like	0.036248
566	even	0.029654
1982	would	0.027339
762	good	0.027325
134	bad	0.026679
1796	time	0.025971
1678	story	0.025584
741	get	0.025283
1171	much	0.024681
290	character	0.023802
292	characters	0.023514
1318	plot	0.023323

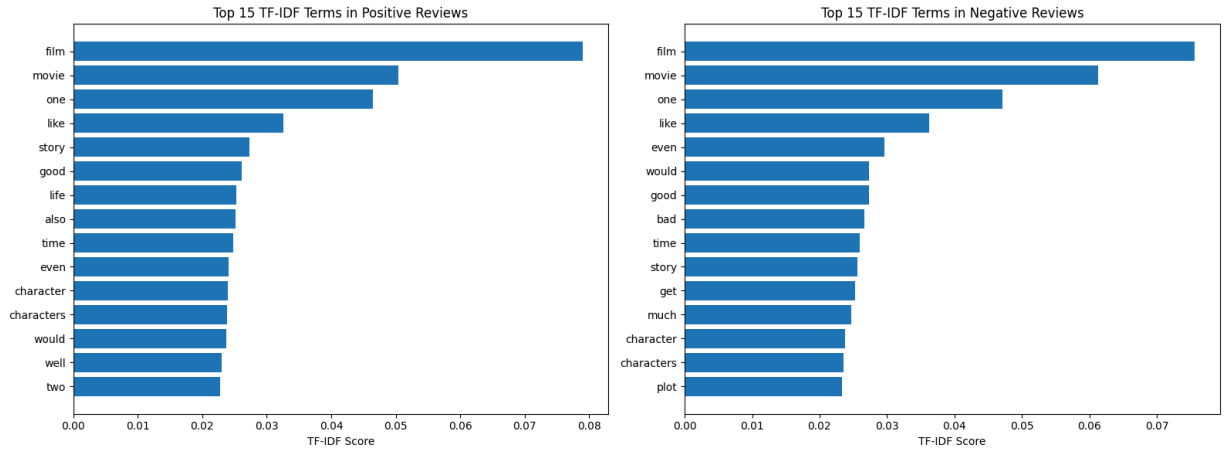
Next steps: Generate code with top_neg New interactive sheet

```
#Step 8: Visualization – Side-by-Side Bar Charts
plt.figure(figsize=(16,6))

# Positive Reviews
plt.subplot(1,2,1)
plt.barh(top_pos['Term'], top_pos['Score'])
plt.title("Top 15 TF-IDF Terms in Positive Reviews")
plt.xlabel("TF-IDF Score")
plt.gca().invert_yaxis()
```

```
# Negative Reviews
plt.subplot(1,2,2)
plt.barh(top_neg['Term'], top_neg['Score'])
plt.title("Top 15 TF-IDF Terms in Negative Reviews")
plt.xlabel("TF-IDF Score")
plt.gca().invert_yaxis()

plt.tight_layout()
plt.show()
```



ıces", "beautiful"

positivity

ɔpointment

.c vocabulary, helping distinguish between positive and negative opinions.