Install & Import Required Libraries

```
import pandas as pd
import numpy as np
import nltk
import re
import matplotlib.pyplot as plt

from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import stopwords
```

Load the Dataset

```
df = pd.read_csv("/content/IMDB Dataset.csv", on_bad_lines='skip', doublequote=False)
df.head()
```

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	Basically there's a family where a little boy ...	negative
2	I sure would like to see a resurrection of a u...	positive
3	This show was an amazing, fresh & innovative i...	negative
4	Encouraged by the positive comments about this...	negative

Next steps: Generate code with df New interactive sheet

Separate Positive and Negative Reviews

```
positive_reviews = df[df['sentiment'] == 'positive']['review']
negative_reviews = df[df['sentiment'] == 'negative']['review']

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

```
Positive reviews: 3846
Negative reviews: 3481
```

Download NLTK Stopwords

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

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

Text Preprocessing Function

```
def preprocess_text(text):
    text = text.lower()
    text = re.sub(r'<.*?>', '', text)
    text = re.sub(r'[^a-z\s]', '', text)
    tokens = text.split()
    tokens = [word for word in tokens if word not in stop_words]
    return " ".join(tokens)
```

Apply Preprocessing

```
positive_clean = positive_reviews.apply(preprocess_text)
negative_clean = negative_reviews.apply(preprocess_text)
```

Create Two TF-IDF Models

```
tfidf_pos = TfidfVectorizer(max_features=1000)
tfidf_neg = TfidfVectorizer(max_features=1000)

pos_tfidf_matrix = tfidf_pos.fit_transform(positive_clean)
neg_tfidf_matrix = tfidf_neg.fit_transform(negative_clean)
```

Extract Top 15 TF-IDF Terms

```
def top_tfidf_terms(tfidf_matrix, feature_names, n=15):
    scores = np.mean(tfidf_matrix.toarray(), axis=0)
    tfidf_dict = dict(zip(feature_names, scores))
    return sorted(tfidf_dict.items(), key=lambda x: x[1], reverse=True)[:n]

top_pos_terms = top_tfidf_terms(pos_tfidf_matrix, tfidf_pos.get_feature_names_out())
top_neg_terms = top_tfidf_terms(neg_tfidf_matrix, tfidf_neg.get_feature_names_out())
```

Display Top Terms

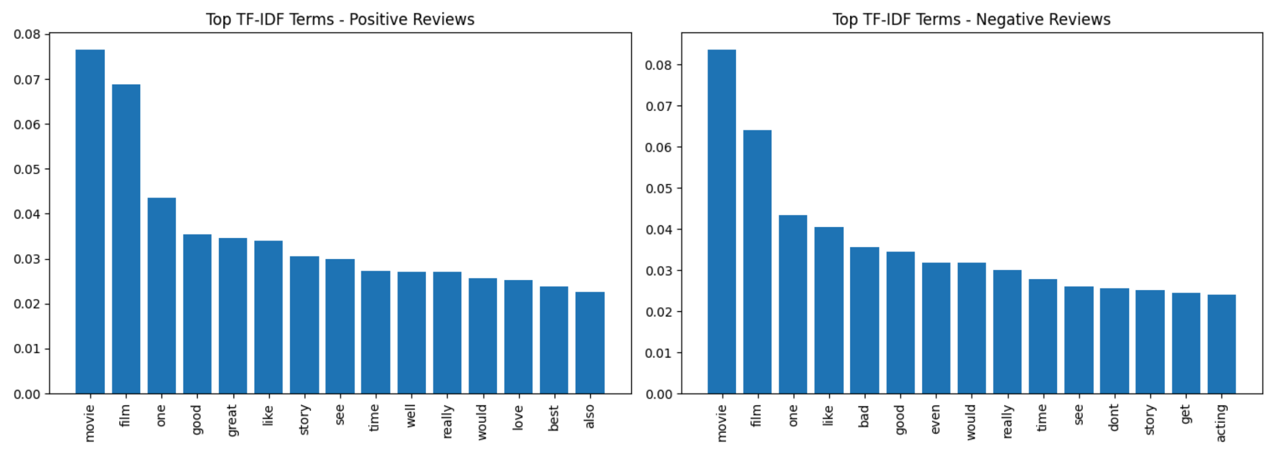
```
print("Top 15 TF-IDF Terms in Positive Reviews")
for term, score in top_pos_terms:
    print(term, ":", round(score, 4))

print("\nTop 15 TF-IDF Terms in Negative Reviews")
for term, score in top_neg_terms:
    print(term, ":", round(score, 4))
```

```
Top 15 TF-IDF Terms in Positive Reviews
movie : 0.0765
film : 0.0688
one : 0.0436
good : 0.0353
great : 0.0345
like : 0.0341
story : 0.0305
see : 0.0299
time : 0.0273
well : 0.0271
really : 0.0271
would : 0.0256
love : 0.0253
best : 0.0239
also : 0.0226

Top 15 TF-IDF Terms in Negative Reviews
movie : 0.0836
film : 0.064
one : 0.0434
like : 0.0405
bad : 0.0357
good : 0.0345
even : 0.0319
would : 0.0318
really : 0.0301
time : 0.0277
see : 0.0259
dont : 0.0255
story : 0.0251
get : 0.0244
acting : 0.0241
```

Visualization – Side-by-Side Bar Charts

```
pos_terms, pos_scores = zip(*top_pos_terms)
neg_terms, neg_scores = zip(*top_neg_terms)

plt.figure(figsize=(14,5))

plt.subplot(1,2,1)
plt.bar(pos_terms, pos_scores)
plt.xticks(rotation=90)
plt.title("Top TF-IDF Terms - Positive Reviews")

plt.subplot(1,2,2)
plt.bar(neg_terms, neg_scores)
plt.xticks(rotation=90)
plt.title("Top TF-IDF Terms - Negative Reviews")

plt.tight_layout()
```

```
    plt.show()
```



Top TF-IDF Terms - Positive Reviews
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
movie
film
one
good
great
like
story
see
time
well
really
would
love
best
also
Top TF-IDF Terms - Negative Reviews
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
movie
film
one
like
bad
good
even
would
really
time
see
dont
story
get
acting