Step 1: Import Required Libraries

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

from sklearn.feature_extraction.text import TfidfVectorizer
from wordcloud import WordCloud
nltk.download('stopwords')
from nltk.corpus import stopwords
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Step 2: Load the Dataset

```
df = pd.read_csv('arxiv_data.csv.zip')
```

```
# Display first 5 rows
df.head()
```

	titles	summaries	terms
0	Survey on Semantic Stereo Matching / Semantic ...	Stereo matching is one of the widely used tech...	['cs.CV', 'cs.LG']
1	FUTURE-AI: Guiding Principles and Consensus Re...	The recent advancements in artificial intellig...	['cs.CV', 'cs.AI', 'cs.LG']
2	Enforcing Mutual Consistency of Hard Regions f...	In this paper, we proposed a novel mutual cons...	['cs.CV', 'cs.AI']
3	Parameter Decoupling Strategy for Semi-supervi...	Consistency training has proven to be an advan...	['cs.CV']
4	Background-Foreground Segmentation for Interio...	To ensure safety in automated driving, the cor...	['cs.CV', 'cs.LG']

Next steps: Generate code with df | New interactive sheet

Step 3: Select One Research Category

```
cs_df = df[df['terms'].str.contains('cs', na=False)]
cs_df = cs_df.head(5)
abstracts = cs_df['summaries'].values
```

Step 4: Text Cleaning Function

```
def clean_text(text):
    text = text.lower()
    text = re.sub(r'\d+', '', text)
    text = re.sub(r'[^a-z\s]', '', text)
    return text
cleaned_abstracts = [clean_text(text) for text in abstracts]
```

Step 5: TF-IDF Vectorization

```
tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_features=1000)
tfidf_matrix = tfidf_vectorizer.fit_transform(cleaned_abstracts)
feature_names = tfidf_vectorizer.get_feature_names_out()
```

Step 6: Top 20 TF-IDF Terms Across Corpus

```
scores = np.sum(tfidf_matrix.toarray(), axis=0)
term_scores = pd.DataFrame({'Term': feature_names, 'Score': scores})
top_20_terms = term_scores.sort_values(by='Score', ascending=False).head(20)
top_20_terms
```

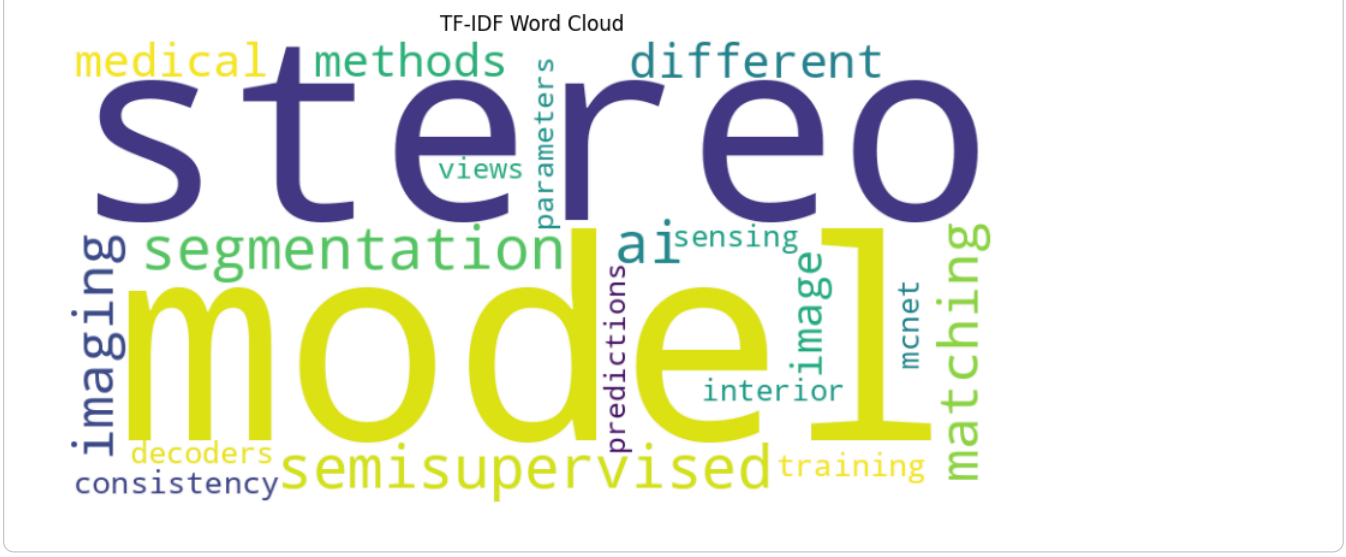
	Term	Score
232	model	0.526339
348	stereo	0.492781
324	semisupervised	0.471568
321	segmentation	0.467531
11	ai	0.456115
224	matching	0.394225
184	imaging	0.390956
110	different	0.380359
228	methods	0.370196
226	medical	0.365653
181	image	0.315975
72	consistency	0.307256
364	training	0.300054
225	mcnet	0.271548
91	decoders	0.271548
197	interior	0.268796
325	sensing	0.268796
279	predictions	0.252485
261	parameters	0.245062
388	views	0.245062

Next steps: Generate code with top_20_terms New interactive sheet

Step 7: Word Cloud Visualization (TF-IDF Weighted)

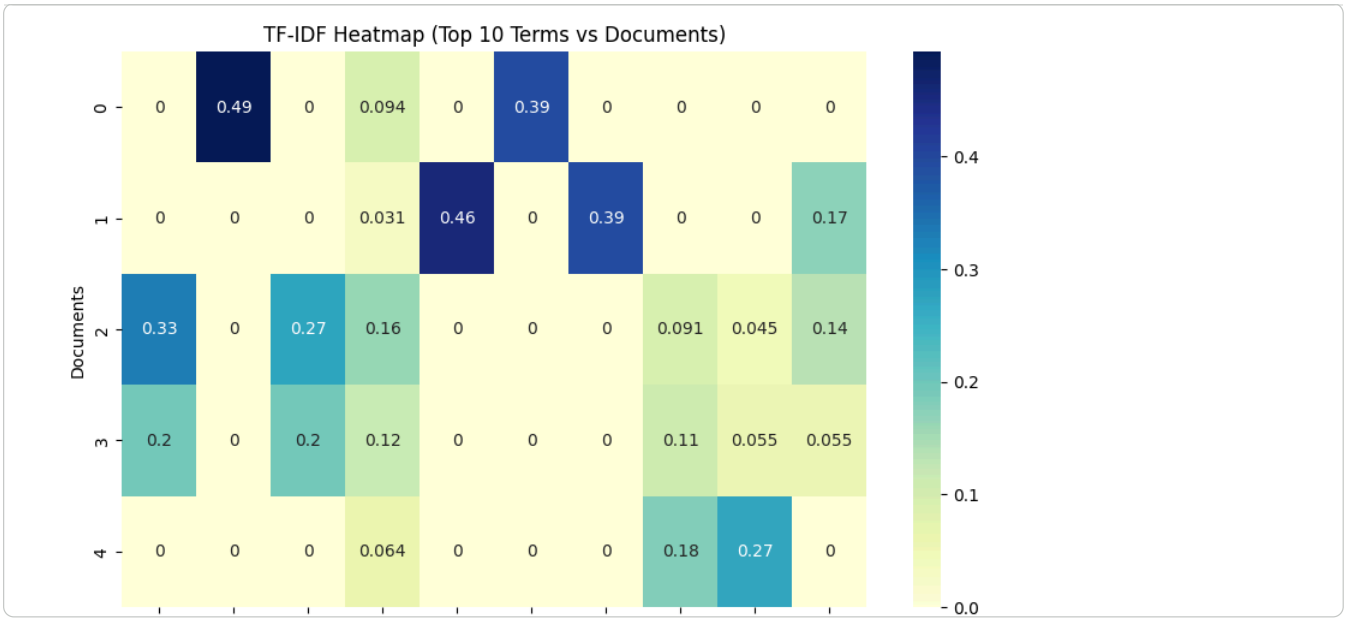
```
oud(width=800, height=400, background_color='white')
_from_frequencies(dict(zip(top_20_terms['Term'], top_20_terms['Score'])))
=(10,5))
ud)

Word Cloud')
```



Step 8: Heatmap of TF-IDF Scores

```
top_10_terms = top_20_terms.head(10)['Term'].values
heatmap_df = pd.DataFrame(tfidf_matrix.toarray(), columns=feature_names)
heatmap_df = heatmap_df[top_10_terms]
plt.figure(figsize=(10,6))
sns.heatmap(heatmap_df, annot=True, cmap='YlGnBu')
plt.title('TF-IDF Heatmap (Top 10 Terms vs Documents)')
plt.xlabel('Terms')
plt.ylabel('Documents')
plt.show()
```


TF-IDF Heatmap (Top 10 Terms vs Documents)
Documents
0
1
2
3
4
0 0.49 0 0.094 0 0.39 0 0 0 0
0 0 0 0.031 0.46 0 0.39 0 0 0.17
0.33 0 0.27 0.16 0 0 0 0.091 0.045 0.14
0.2 0 0.2 0.12 0 0 0 0.11 0.055 0.055
0 0 0 0.064 0 0 0 0.18 0.27 0
0.4
0.3
0.2
0.1
0.0