


Name: V.Vamshi  
HTNO: 2403A52016

```
import pandas as pd
df = pd.read_csv('/Tweets.csv')
display(df.head())
```

	tweet_id	airline_sentiment	airline_sentiment_confidence	negative
0	570306133677760513	neutral	1.0000	
1	570301130888122368	positive	0.3486	
2	570301083672813571	neutral	0.6837	
3	570301031407624196	negative	1.0000	B:
4	570300817074462722	negative	1.0000	C

```
display(df[['text', 'airline_sentiment']].head())
```

	text	airline_sentiment	
0	@VirginAmerica What @dhepburn said.	neutral	
1	@VirginAmerica plus you've added commercials t...	positive	
2	@VirginAmerica I didn't today... Must mean I n...	neutral	
3	@VirginAmerica it's really aggressive to blast...	negative	
4	@VirginAmerica and it's a really big bad thing...	negative	

```
import nltk

# Download necessary NLTK data directly
```

```

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('punkt_tab') # Added to resolve the new LookupError

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

# Get English stopwords
stop_words = set(stopwords.words('english'))

# Function to tokenize and remove stopwords
def preprocess_text(text):
    # Convert to lowercase
    text = text.lower()
    # Tokenize the text
    tokens = word_tokenize(text)
    # Remove stopwords and non-alphabetic tokens
    filtered_tokens = [word for word in tokens if word.isalpha() and word not in stop_words]
    return " ".join(filtered_tokens)

# Apply the preprocessing to the 'text' column
df['processed_text'] = df['text'].apply(preprocess_text)

# Display the original and processed text for comparison
display(df[['text', 'processed_text']].head())

```

```

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data] Unzipping tokenizers/punkt_tab.zip.

```

	text	processed_text
0	@VirginAmerica What @dhepburn said.	virginamerica dhepburn said
1	@VirginAmerica plus you've added commercials t...	virginamerica plus added commercials experienc...
2	@VirginAmerica I didn't today... Must mean I n...	virginamerica today must mean need take anothe...
3	@VirginAmerica it's really aggressive to blast...	virginamerica really aggressive blast obnoxio...
4	@VirginAmerica and it's a really big bad thing...	virginamerica really big bad thing

```

from sklearn.feature_extraction.text import TfidfVectorizer

# Initialize TF-IDF Vectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=5000) # Limiting to top 5000 fe

```

```
# Fit and transform the processed text
tfidf_matrix = tfidf_vectorizer.fit_transform(df['processed_text'])

# Display the shape of the TF-IDF matrix and some feature names
print(f"Shape of TF-IDF matrix: {tfidf_matrix.shape}")
print(f"Top 10 feature names: {tfidf_vectorizer.get_feature_names_out()[:10]}")
```

```
Shape of TF-IDF matrix: (14640, 5000)
Top 10 feature names: ['aa' 'aadvantage' 'aafail' 'aarp' 'abc' 'abcnetwork' 'abi'
'ability' 'able' 'aboard']
```

```
# Filter for negative sentiment tweets
negative_tweets_df = df[df['airline_sentiment'] == 'negative']

# Transform the processed text of negative tweets using the *already fitted* TF
tfidf_matrix_negative = tfidf_vectorizer.transform(negative_tweets_df['processe

# Get feature names (terms)
feature_names = tfidf_vectorizer.get_feature_names_out()

# Sum TF-IDF scores for each term across all negative tweets
# The .A converts the sparse matrix to a dense NumPy array for summation
sum_tfidf_scores_negative = tfidf_matrix_negative.sum(axis=0).A1

# Create a Series of terms and their summed TF-IDF scores
tfidf_scores_series_negative = pd.Series(sum_tfidf_scores_negative, index=featu

# Sort the terms by their TF-IDF scores in descending order and get the top N
top_n_negative_terms = tfidf_scores_series_negative.nlargest(20) # Display top

print("Top 20 TF-IDF terms for Negative Sentiment:")
display(top_n_negative_terms)
```

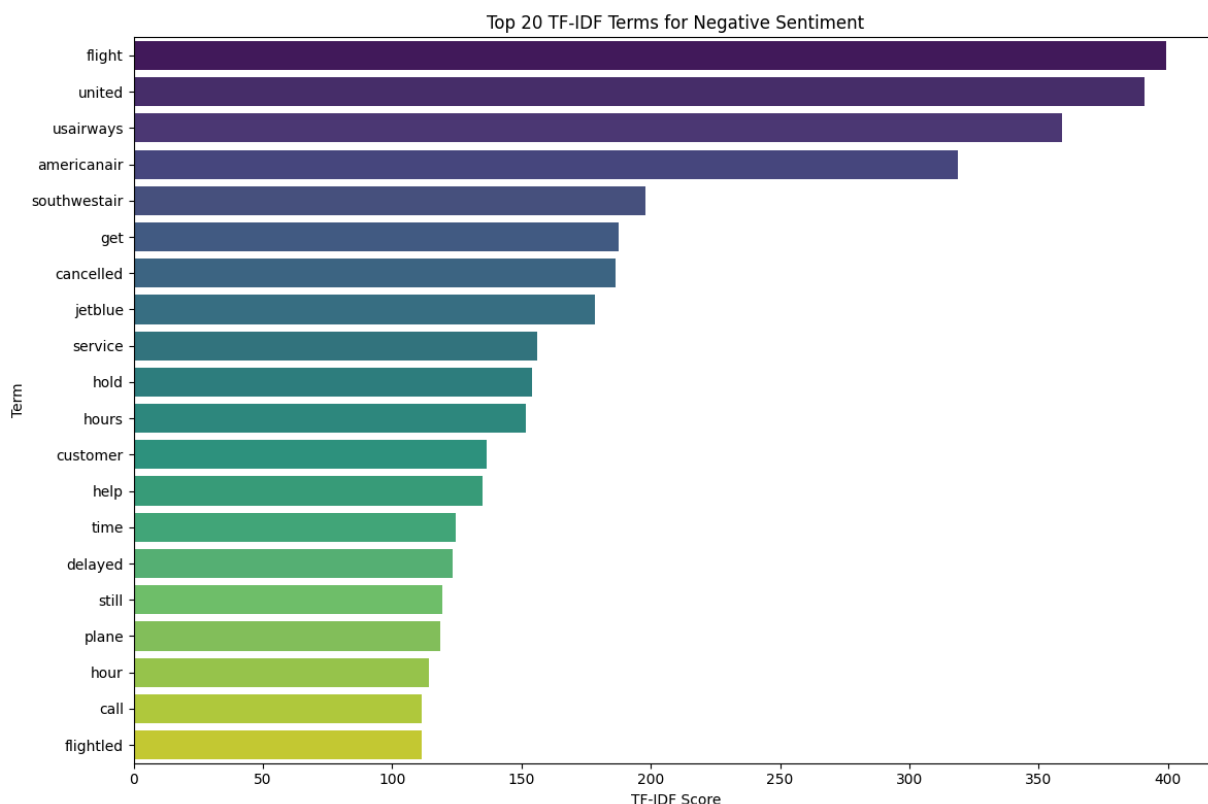
Top 20 TF-IDF terms for Negative Sentiment:

	0
<b>flight</b>	399.280532
<b>united</b>	390.873367
<b>usairways</b>	358.955895
<b>americanair</b>	318.814738
<b>southwestair</b>	197.952409
<b>get</b>	187.455508
<b>cancelled</b>	186.415954
<b>jetblue</b>	178.373214
<b>service</b>	156.102443
<b>hold</b>	154.205142
<b>hours</b>	151.847064
<b>customer</b>	136.386221
<b>help</b>	134.771582
<b>time</b>	124.744890
<b>delayed</b>	123.243031
<b>still</b>	119.499013
<b>plane</b>	118.396010
<b>hour</b>	114.349198
<b>call</b>	111.339799
<b>flightled</b>	111.233616

**dtype:** float64

```
import matplotlib.pyplot as plt
import seaborn as sns

# Create a bar chart for the top 20 negative TF-IDF terms
plt.figure(figsize=(12, 8))
sns.barplot(x=top_n_negative_terms.values, y=top_n_negative_terms.index, palette='magma')
plt.title('Top 20 TF-IDF Terms for Negative Sentiment')
plt.xlabel('TF-IDF Score')
plt.ylabel('Term')
plt.tight_layout()
plt.show()
```



```
# Install the wordcloud library if not already installed
!pip install wordcloud
```

```
Requirement already satisfied: wordcloud in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: numpy>=1.19 in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: pillow in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dis
Requirement already satisfied: cyclor>=0.10 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/di
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/di
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dis
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packag
```

```
from wordcloud import WordCloud

# Generate a word cloud for negative sentiment terms
# We'll use the tfidf_scores_series_negative for weights
wordcloud = WordCloud(width=800, height=400, background_color='white').gen

plt.figure(figsize=(15, 7))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud for Negative Sentiment Terms')
plt.show()
```

