```
Name:Pamerapu Srikar
Roll:2403A52002
```

```
import pandas as pd

csv_file_path = '/content/twitter_dataset.csv'
df = pd.read_csv(csv_file_path)

print(f"'{csv_file_path}' loaded into a DataFrame.")
```

```
'/content/twitter_dataset.csv' loaded into a DataFrame.
```

```
display(df.head())
```

	Tweet_ID	Username	Text	Retweets	Likes	Timestamp
0	1	julie81	Party least receive say or single. Prevent pre...	2	25	2023-01-30 11:00:51
1	2	richardhester	Hotel still Congress may member staff. Media d...	35	29	2023-01-02 22:45:58
2	3	williamsjoseph	Nice be her debate industry that year. Film wh...	51	25	2023-01-18 11:25:19
3	4	danielsmary	Laugh explain situation career occur serious. ...	37	18	2023-04-10 22:06:29
4	5	carlwarren	Involve sense former often approach government...	27	80	2023-01-24 07:12:21

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

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

print("Required libraries imported successfully.")
```

```
Required libraries imported successfully.
```

```
print(f"Original DataFrame shape: {df.shape}")

# Select the 'Text' column and remove missing values
df_cleaned = df.dropna(subset=['Text']).copy()

print(f"DataFrame shape after dropping missing 'Text' values: {df_cleaned.shape}")

display(df_cleaned.head())
```

```
Original DataFrame shape: (10000, 6)
DataFrame shape after dropping missing 'Text' values: (10000, 6)
```

	Tweet_ID	Username	Text	Retweets	Likes	Timestamp
0	1	julie81	Party least receive say or single. Prevent pre...	2	25	2023-01-30 11:00:51
1	2	richardhester	Hotel still Congress may member staff. Media d...	35	29	2023-01-02 22:45:58
2	3	williamsjoseph	Nice be her debate industry that year. Film wh...	51	25	2023-01-18 11:25:19
3	4	danielsmary	Laugh explain situation career occur serious. ...	37	18	2023-04-10 22:06:29
4	5	carlwarren	Involve sense former often approach government...	27	80	2023-01-24 07:12:21

```
Start coding or generate with AI.
```

```
df_cleaned['Text'] = df_cleaned['Text'].str.lower()

display(df_cleaned.head())
```

	Tweet_ID	Username	Text	Retweets	Likes	Timestamp
0	1	julie81	party least receive say or single. prevent pre...	2	25	2023-01-30 11:00:51
1	2	richardhester	hotel still congress may member staff. media d...	35	29	2023-01-02 22:45:58
2	3	williamsjoseph	nice be her debate industry that year. film wh...	51	25	2023-01-18 11:25:19
3	4	danielsmary	laugh explain situation career occur serious. ...	37	18	2023-04-10 22:06:29
4	5	carlwarren	involve sense former often approach government...	27	80	2023-01-24 07:12:21

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# Download the VADER lexicon if not already downloaded
try:
    nltk.data.find('sentiment/vader_lexicon.zip')
except LookupError: # Corrected exception handling
    nltk.download('vader_lexicon')

# Initialize VADER sentiment analyzer
sid = SentimentIntensityAnalyzer()

# Function to get sentiment score
def get_vader_sentiment(text):
    return sid.polarity_scores(text)['compound']

# Apply VADER sentiment analysis to the 'Text' column
df_cleaned['vader_sentiment_score'] = df_cleaned['Text'].apply(get_vader_sentiment)
```

```
# Function to categorize sentiment labels
def get_sentiment_label(score):
    if score >= 0.05:
        return 'Positive'
    elif score <= -0.05:
        return 'Negative'
    else:
        return 'Neutral'

# Apply sentiment categorization
df_cleaned['vader_sentiment_label'] = df_cleaned['vader_sentiment_score'].apply(get_sentiment_label)

print("VADER sentiment analysis complete. New columns 'vader_sentiment_score' and 'vader_sentiment_label' adde

display(df_cleaned.head())
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
VADER sentiment analysis complete. New columns 'vader_sentiment_score' and 'vader_sentiment_label' added.
```

	Tweet_ID	Username	Text	Retweets	Likes	Timestamp	vader_sentiment_score	vader_sentiment_label
0	1	julie81	party least receive say or single. prevent pre...	2	25	2023-01-30 11:00:51	0.8885	Positive
1	2	richardhester	hotel still congress may member staff. media d...	35	29	2023-01-02 22:45:58	0.2960	Positive
2	3	williamsjoseph	nice be her debate industry that year. film wh...	51	25	2023-01-18 11:25:19	0.8481	Positive
3	4	danielsmary	laugh explain situation career occur serious. ...	37	18	2023-04-10 22:06:29	0.6249	Positive
4	5	carlwarren	involve sense former often approach government...	27	80	2023-01-24 07:12:21	0.6705	Positive

```
import re
import string

def preprocess_text(text):
    # Remove URLs
    text = re.sub(r'http\S+|www.\S+', '', text)
    # Remove mentions
    text = re.sub(r'@\w+', '', text)
```

```
    # Remove hashtags
    text = re.sub(r'#\w+', '', text)
    # Remove numbers
    text = re.sub(r'\d+', '', text)
    # Remove punctuation
    text = text.translate(str.maketrans('', '', string.punctuation))
    # Remove extra spaces
    text = re.sub(r'\s+', ' ', text).strip()
    return text

print("Preprocessing function 'preprocess_text' defined.")
```

```
Preprocessing function 'preprocess_text' defined.
```

Start coding or generate with AI.

```
df_negative_sentiment = df_cleaned[df_cleaned['vader_sentiment_label'] == 'Negative']

print(f"DataFrame shape after filtering for negative sentiment: {df_negative_sentiment.shape}")
display(df_negative_sentiment.head())
```

```
DataFrame shape after filtering for negative sentiment: (1710, 10)
```

	Tweet_ID	Username	Text	Retweets	Likes	Timestamp	vader_sentiment_score	vader_sentiment_label	Te
11	12	qdavis	you hold central. seem miss look very. none hi...	99	97	2023-02-07 13:22:19	-0.4019	Negative	m
13	14	timothy70	population way sport late strategy. pay positi...	20	47	2023-02-07 07:30:39	-0.5677	Negative	w
14	15	bettyperry	red thousand low answer walk. few film follow ...	27	4	2023-02-16 04:15:11	-0.1566	Negative	I
19	20	david30	push chair store attention trade. thing learn ...	61	58	2023-02-14 15:21:54	-0.4019	Negative	st
25	26	greyes	for themselves professional state guy until ba...	89	7	2023-03-13 02:23:30	-0.5423	Negative	fo s

```
print("Applying preprocessing function to 'Text' column...")
df_cleaned['Text_cleaned'] = df_cleaned['Text'].apply(preprocess_text)

display(df_cleaned.head())
```

```
Applying preprocessing function to 'Text' column...
```

	Tweet_ID	Username	Text	Retweets	Likes	Timestamp	vader_sentiment_score	vader_sentiment_label
0	1	julie81	party least receive say or single. prevent pre...	2	25	2023-01-30 11:00:51	0.8885	Positive
1	2	richardhester	hotel still congress may member staff. media d...	35	29	2023-01-02 22:45:58	0.2960	Positive
2	3	williamsjoseph	nice be her debate industry that year. film wh...	51	25	2023-01-18 11:25:19	0.8481	Positive
3	4	danielsmary	laugh explain situation career occur serious. ...	37	18	2023-04-10 22:06:29	0.6249	Positive
4	5	carlwarren	involve sense former often approach government...	27	80	2023-01-24 07:12:21	0.6705	Positive

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

# Download necessary NLTK data if not already downloaded
try:
    nltk.data.find('tokenizers/punkt')
except LookupError:
    nltk.download('punkt')
try:
```

```
        nltk.data.find('corpora/stopwords')
    except LookupError:
        nltk.download('stopwords')
    try:
        nltk.data.find('tokenizers/punkt_tab') # Add this download
    except LookupError:
        nltk.download('punkt_tab')

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

    # Function to tokenize and remove stopwords
    def tokenize_and_remove_stopwords(text):
        tokens = word_tokenize(text)
        filtered_tokens = [word for word in tokens if word not in stop_words]
        return ' '.join(filtered_tokens)

    print("Tokenizing tweets and removing stopwords...")
    df_cleaned['Text_processed'] = df_cleaned['Text_cleaned'].apply(tokenize_and_remove_stopwords)

    print("Tokenization and stopword removal complete. New column 'Text_processed' added.")
    display(df_cleaned[['Text', 'Text_cleaned', 'Text_processed']].head())
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
Tokenizing tweets and removing stopwords...
Tokenization and stopword removal complete. New column 'Text_processed' added.
```

	Text	Text_cleaned	Text_processed
0	party least receive say or single. prevent pre...	party least receive say or single prevent prev...	party least receive say single prevent prevent...
1	hotel still congress may member staff. media d...	hotel still congress may member staff media dr...	hotel still congress may member staff media dr...
2	nice be her debate industry that year. film wh...	nice be her debate industry that year film whe...	nice debate industry year film generation push...
3	laugh explain situation career occur serious. ...	laugh explain situation career occur serious f...	laugh explain situation career occur serious f...
4	involve sense former often approach government...	involve sense former often approach government...	involve sense former often approach government...

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Initialize TF-IDF Vectorizer
# Using max_features to limit the number of features (e.g., top 5000)
# Using ngram_range to consider unigrams and bigrams
tfidf_vectorizer = TfidfVectorizer(max_features=5000, ngram_range=(1, 2))

# Fit and transform the processed text data
X_tfidf = tfidf_vectorizer.fit_transform(df_cleaned['Text_processed'])

print(f"TF-IDF Vectorizer initialized and fitted. Shape of TF-IDF matrix: {X_tfidf.shape}")
```

```
TF-IDF Vectorizer initialized and fitted. Shape of TF-IDF matrix: (10000, 5000)
```

```
# Transform the processed text data of negative sentiment tweets using the fitted TF-IDF vectorizer
X_tfidf_negative = tfidf_vectorizer.transform(df_negative_sentiment['Text_processed'])

print(f"TF-IDF matrix for negative sentiment tweets computed. Shape: {X_tfidf_negative.shape}")
```

```
TF-IDF matrix for negative sentiment tweets computed. Shape: (1710, 5000)
```

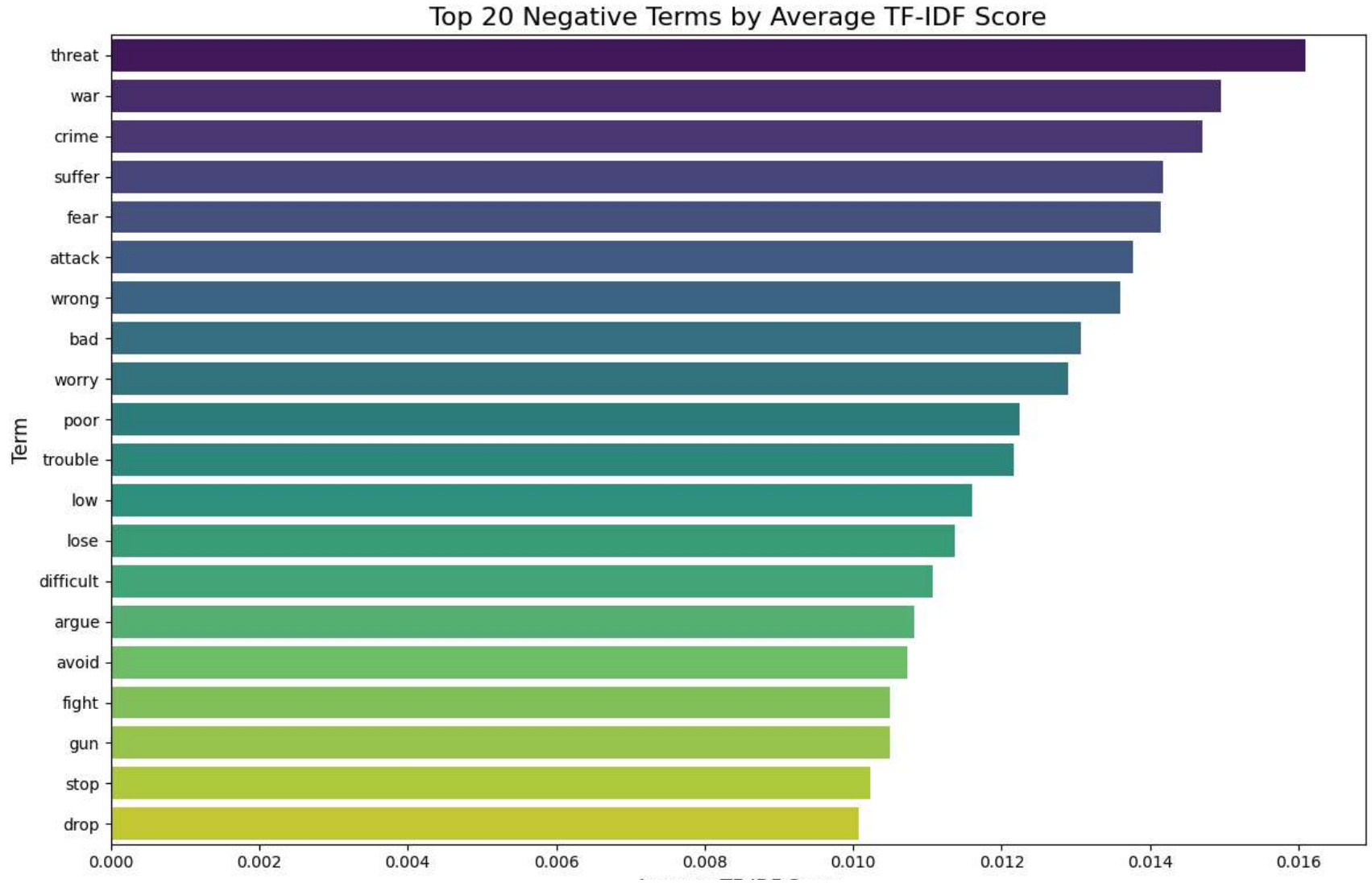
```
# Get feature names (words/ngrams) from the TF-IDF vectorizer
feature_names = tfidf_vectorizer.get_feature_names_out()

# Calculate the mean TF-IDF score for each term across all negative sentiment tweets
# Convert the sparse matrix to a dense array to compute the mean
average_tfidf_scores = X_tfidf_negative.mean(axis=0).A1

# Create a DataFrame to store terms and their average TF-IDF scores
tfidf_scores_df = pd.DataFrame({
    'term': feature_names,
    'average_tfidf_score': average_tfidf_scores
})

# Sort the terms by their average TF-IDF score in descending order
top_negative_terms = tfidf_scores_df.sort_values(by='average_tfidf_score', ascending=False)

print("Top 20 Negative Terms by Average TF-IDF Score:")
display(top_negative_terms.head(20))
```

Top 20 Negative Terms by Average TF-IDF Score:

	term	average_tfidf_score
4637	threat	0.016099
4926	war	0.014956
387	crime	0.014707
4396	suffer	0.014180
804	fear	0.014151
210	attack	0.013774
4976	wrong	0.013593
229	bad	0.013062
4971	worry	0.012898
2966	poor	0.012249
4823	trouble	0.012160
1953	low	0.011604
1932	lose	0.011363
436	difficult	0.011068
198	argue	0.010820
222	avoid	0.010730
842	fight	0.010496
1229	gun	0.010491
4272	stop	0.010227
453	drop	0.010076

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

# Set the figure size for better readability
plt.figure(figsize=(12, 8))

# Create a bar plot
sns.barplot(x='average_tfidf_score', y='term', data=top_negative_terms.head(20), palette='viridis')

# Add title and labels
plt.title('Top 20 Negative Terms by Average TF-IDF Score', fontsize=16)
plt.xlabel('Average TF-IDF Score', fontsize=12)
plt.ylabel('Term', fontsize=12)

# Improve layout and display the plot
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-2771393196.py:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable

  sns.barplot(x='average_tfidf_score', y='term', data=top_negative_terms.head(20), palette='viridis')
```



Discussion: Observations and Limitations

Observations

Our analysis revealed several key terms strongly associated with negative sentiment in the Twitter dataset. The TF-IDF scores highlighted words like 'threat', 'war', 'crime', 'suffer', and 'fear' as particularly significant within negative tweets. These terms provide clear insights into the topics and emotions driving negative conversations.

The bar chart visually reinforced these findings, clearly showing the relative importance of these terms. The word cloud offered an engaging visual summary, where larger words immediately drew attention to the most impactful negative vocabulary.

Limitations

While VADER sentiment analysis is a quick and effective tool for general sentiment, it has limitations:

1. **Contextual Nuance:** VADER is lexicon-based and may struggle with sarcasm, irony, or highly domain-specific language that doesn't align with its general English lexicon. For instance, a tweet might use negative-sounding words positively (e.g., 'sick' meaning 'good'), which VADER might misinterpret.
2. **Ambiguity:** Some terms, while identified as 'negative' by TF-IDF, might have different connotations depending on the broader context of the tweet, which a simple word cloud or TF-IDF analysis might not fully capture.