Intermediate level

Sentiment Analysis of X (Twitter) Data

Sentiment Analysis Approach

Two recommended approaches:

A. VADER (fast, no training) — good baseline for social media.
B. Transformer (fine-tuned BERT/RoBERTa) — higher accuracy, requires compute or a pre-trained model.

Statistical Analyses & Visualizations

- Sentiment distribution (bar chart / pie chart).
- Time-series of average sentiment score and counts by sentiment class.
- Top hashtags by sentiment (bar charts).
- Word clouds for Positive vs Negative tweets.
- Boxplots of likes/retweets grouped by sentiment to analyze engagement differences.
- Geographical heatmap of sentiment (if location available).

Appendix

Below is a complete, runnable Python script you can execute locally (or in Colab). It uses pandas, NLTK's VADER, matplotlib, seaborn, and wordcloud. Modify the dataset path if needed.

Required libraries

```
import pandas as pd
import numpy as np
import re
import matplotlib.pyplot as plt
import seaborn as sns
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import nltk
from wordcloud import WordCloud
nltk.download('vader_lexicon')
```

Load dataset

```
df = pd.read_csv('/path/to/your/tweets.csv', low_memory=False)
```

Basic inspection

```
print(df.columns)
print(df.shape)
print(df['text'].head())
```

Cleaning function

```
def clean_tweet(text):
    text = str(text)
    text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE)
    text = re.sub(r'@\w+', '', text)   # remove mentions
    text = re.sub(r'#', '', text)      # remove hashtag symbol (keep words)
    text = re.sub(r'[^A-Za-z0-9\s]', '', text)  # remove special chars (optional)
    text = text.lower().strip()
    return text

df['clean_text'] = df['text'].apply(clean_tweet)
```

VADER sentiment

```
sia = SentimentIntensityAnalyzer()
df['compound'] = df['clean_text'].apply(lambda x: sia.polarity_scores(x)['compound'])

def label_sentiment(comp):
    if comp >= 0.05:
        return 'Positive'
    elif comp <= -0.05:
        return 'Negative'
    else:
        return 'Neutral'

df['sentiment'] = df['compound'].apply(label_sentiment)
```

Sentiment distribution

```
sent_counts = df['sentiment'].value_counts(normalize=True) * 100
print(sent_counts)
```

Plot sentiment distribution

```
plt.figure(figsize=(6,4))
sns.countplot(x='sentiment', data=df, order=['Positive','Neutral','Negative'])
plt.title('Sentiment Distribution')
plt.savefig('sentiment_distribution.png', bbox_inches='tight')
```

Time series

```python
df['created_at'] = pd.to_datetime(df['created_at'], errors='coerce')
df_time = df.dropna(subset=['created_at']).copy()
df_time['date'] = df_time['created_at'].dt.date
time_trend = df_time.groupby(['date','sentiment']).size().unstack(fill_value=0)
time_trend_pct = time_trend.div(time_trend.sum(axis=1), axis=0)

time_trend_pct.plot(figsize=(12,5))
plt.ylabel('Proportion by day')
plt.title('Daily Sentiment Proportions')
plt.savefig('sentiment_trend.png', bbox_inches='tight')
```

Hashtag-level sentiment

```python
def extract_hashtags(text):
    return [tag.strip() for tag in re.findall(r'#(\w+)', str(text))]

df['hashtags'] = df['text'].apply(lambda x: extract_hashtags(x))
hashtag_exploded = df.explode('hashtags').dropna(subset=['hashtags'])
top_hashtags = hashtag_exploded['hashtags'].value_counts().nlargest(20).index.tolist()
hashtag_sent = hashtag_exploded[hashtag_exploded['hashtags'].isin(top_hashtags)].groupby(['hashtags','sentiment']).size().unstac
```

Engagement vs Sentiment

```python
if 'retweets' in df.columns or 'likes' in df.columns:
    agg_cols = []
    if 'retweets' in df.columns:
        agg_cols.append('retweets')
    if 'likes' in df.columns:
        agg_cols.append('likes')
    eng = df.groupby('sentiment')[agg_cols].median()
    print(eng)
```

Word clouds

```python
positive_text = ' '.join(df[df['sentiment']=='Positive']['clean_text'].astype(str).values)
negative_text = ' '.join(df[df['sentiment']=='Negative']['clean_text'].astype(str).values)

wc_pos = WordCloud(width=800, height=400, background_color='white').generate(positive_text)
wc_neg = WordCloud(width=800, height=400, background_color='white').generate(negative_text)

wc_pos.to_file('wordcloud_positive.png')
wc_neg.to_file('wordcloud_negative.png')
```