Sentiment Analysis Report

Objective

The goal of this task was to perform sentiment analysis on a set of tweets using the VADER (Valence Aware Dictionary and Sentiment Reasoner) model. The focus was on extracting sentiment polarity (positive, neutral, or negative) using lexicon-based scoring.

Dataset

We worked with the dataset tweets-data.csv, which includes over 3000 tweets. For performance and reproducibility, a random sample of 500 rows was selected using random_state=42.

```
[2] import pandas as pd
    import re
    import nltk
    from nltk.sentiment import SentimentIntensityAnalyzer
```

Load the Dataset

```
[4] df = pd.read_csv('tweets-data.csv')
    sample_df = df.sample(n=500, random_state=42).copy()
    df
```

	Unnamed: 0	Date Created	Number of Likes	Source of Tweet	Tweets	hashtag
0	0	2023-06-25 19:16:20+00:00	0	NaN	@jacksonhinklle #wagner with 6.2 billion dolla...	wagner
1	1	2023-06-25 19:16:18+00:00	0	NaN	Pobrecito es discapacitado\n#Reddetuiterosdemo...	wagner
2	2	2023-06-25 19:16:07+00:00	0	NaN	News from the EIR Daily Alert\n\n"#Putin Addre...	wagner
3	3	2023-06-25 19:15:56+00:00	0	NaN	It's Messi day #Messi🐐 #Messi36 #Russia #bigst...	wagner
4	4	2023-06-25 19:15:54+00:00	0	NaN	Il passaggio chiave di Machiavelli era questo ...	wagner

Step 1: Data Cleaning

```
def clean_tweet(text):
    text = re.sub(r"@\w+", "", text)
    text = re.sub(r"http\S+|www\S+", "", text)
    text = re.sub(r"[^a-zA-Z\s]", "", text)
    text = text.lower().strip()
    return text

sample_df['cleaned_tweet'] = sample_df['Tweets'].apply(clean_tweet)
```

Tweets were first preprocessed using a custom function:

- Removed usernames, URLs, punctuation, and symbols using re.sub.
- Converted all characters to lowercase.
- Trimmed extra whitespace.

A new column, cleaned_tweet, was added to hold these cleaned texts. This ensured better compatibility with lexicon matching in VADER.

Step 2: Sentiment Detection with VADER

```
[6] from nltk.sentiment import SentimentIntensityAnalyzer
    nltk.download('vader_lexicon')

    sia = SentimentIntensityAnalyzer()

    def get_sentiment(text):
        score = sia.polarity_scores(text)
        compound = score['compound']
        if compound >= 0.05:
            sentiment = 'Positive'
        elif compound <= -0.05:
            sentiment = 'Negative'
        else:
            sentiment = 'Neutral'
        return pd.Series([sentiment, compound])

    [nltk_data] Downloading package vader_lexicon to /root/nltk_data...
```

Using nltk.sentiment.vader.SentimentIntensityAnalyzer, we defined a function that:

- Computes sentiment scores for each tweet (negative, neutral, positive, compound).
- Labels the sentiment based on the compound score thresholds:
  - >= 0.05: Positive
  - <= -0.05: Negative
  - Otherwise: Neutral

Each tweet was assigned two new outputs:

- sentiment_label (string)

- sentiment_score (float)

Step 3: Results

```
[8] sample_df[['sentiment_label', 'sentiment_score']] = sample_df['cleaned_tweet'].apply(get_sentiment)
```

Results

```
[9] sample_df.to_csv("sentiment_output.csv", index=False)
```

We applied the function using .apply() to every row of the cleaned_tweet column. Two new columns were added:

- sentiment_label: one of Positive, Negative, or Neutral
- sentiment_score: the actual compound score from VADER

The final results were exported as a new CSV file:
sentiment_output.csv

Observations

- The pipeline is fast and efficient, well-suited for short texts like tweets.
- Lexicon-based models are interpretable but may lack contextual understanding.
- VADER handled informal language (emojis, slang) reasonably well due to its tuning for social media.

Final Note

This task successfully demonstrates the value of rule-based sentiment analysis using VADER. While it's a lightweight and quick method, for complex language or nuanced tone, hybrid or deep learning models may perform better.