Comprehensive Documentation

Setup Requirements

- **Python Version:** Python 3.8 or above due to dependencies on recent library features.
- **Dependencies Installation:** Required Python libraries can be installed via pip:

```
pip install pandas numpy matplotlib seaborn plotly scikit-learn requests python-dotenv
```

Environment Setup

- Create a `.env` file in the root directory of the project to store API keys securely:

```
ALPHAVANTAGE_API_KEY='your_alphavantage_api_key_here'
NEWS_API_KEY='your_news_api_key_here'
```

Usage Instructions

- Run the script from the command line:

```
python main.py
```

2. Code Explanations

Data Fetching and Caching

- **`fetch_stock_data`** : Fetches stock data using the Alpha Vantage API. It first attempts to load cached data; if unavailable or stale, it fetches new data, respects API rate limits, and caches the new data.
- **`cache_data` and `load_cached_data`** : Manage the caching mechanism to reduce API calls, using pickle to serialize data with a freshness timestamp.

Predictive Modeling

- **`preprocess_data`** : Prepares data by scaling feature columns necessary for modeling.
- **`train_model`** : Trains a gradient boosting regressor model using the prepared data.
- **`predict_future_prices_hermite`** : Uses Hermite spline interpolation for future stock price predictions based on the trained model.

Visualization

- `plot_combined` : Generates both static and interactive plots for stock prices and volumes.
- `plot_sentiment_trends` : Plots sentiment trends derived from news headline analysis.

Sentiment Analysis

- `fetch_news_headlines` , `analyze_sentiment` , `integrate_sentiment_data` : These functions collectively fetch news headlines, analyze their sentiment using a pre-trained BERT model, and integrate sentiment scores into the stock data.

3. Concise Report on Methodologies

Predictive Modeling

- Utilizes machine learning models to forecast future stock prices based on historical data. This project implements a Gradient Boosting Regressor for robust predictions against non-linear data trends and employs Cubic Hermite Spline for interpolation to predict future stock price movements based on the last observed trends.
- The data is preprocessed using a standard scaler to normalize feature scales, ensuring better performance and stability of the machine learning model.

API Management

- **Rate Limiting**: Ensures that the script adheres to API rate limits using a custom-built rate limiter that delays requests based on the last call time.
- **Data Caching**: Implements data caching to optimize API usage. Data is cached with a freshness timestamp, allowing the script to decide whether to use cached data or fetch new if it's stale.
- **Error Handling**: Robust error handling is built into the data fetching functions to gracefully manage and log issues like network errors or data access problems.

Deliverables

1. **Integrated Python Script**: All functionalities (data fetching, processing, predictive modeling, visualization, and sentiment analysis) are encapsulated in a single Python script (`main.py`).
2. **Documentation**: Detailed documentation is provided, outlining setup, installation, and usage.
3. **Methodological Report**: A concise report explaining the methodologies employed, focusing on predictive modeling and API management.