Objectives of the project

Data Collection:
The project uses a CSV file (`financial_news_data.csv`) of real-time scraping from The Guardian. (Total data scraping of 60k)

Sample:

```
webTitle,sectionName,publishedDate,id,webUrl,sectionId,tags,companyName,sourceType,topic,keywords
```

```
UK mortgage rates rising; Meta fined for breaching EU antitrust rules – as it happened,Business,2025-11-18 17:15:02+00:00,business/live/2024/nov/14/uk-pension-megafund-renters-rising-demand-falling-supply-stock-markets-ftse-pound-us-dollar-business-live-news,https://www.theguardian.com/business/live/2024/nov/14/uk-pension-megafund-renters-rising-demand-falling-supply-stock-markets-ftse-pound-us-dollar-business-live-news,business,N/A,UK,News,Finance,"eu, mortgage, rates"
```

Status: Modified from original objective, but implemented as per your requirements.

Data Cleaning:
Implemented in the `preprocess_data` function in `src/data_preprocessing/preprocess.py`.

Status: Completed.

Sentiment Analysis Model (Hybrid approach):

a. Rule-based model with custom lexicon:
Implemented in `src/models/lexicon_model.py` and `src/models/custom_lexicon.py`.

Status: Completed.

b. Machine learning model:

Implemented in `src/models/ml_model.py`.

Uses the Financial PhraseBank dataset as requested.

Status: Completed.

Comparison with VADER:

- VADER model implemented in `src/models/vader_model.py`.
- Comparison included in the dashboard.

Status: Completed.

Real-time Processing:

- The current implementation doesn't use Flask for real-time processing.
- Instead, it uses Dash, which is built on Flask, for the dashboard and allows for real-time updates.

Status: Modified from original objective, but real-time capability is present through Dash.

UI Development:

- Dashboard implemented using Dash (Python) instead of React.js.
- Displays company sentiment trends and provides insights.

Status: Completed

Main Objectives:

1. Creating and comparing the hybrid model with the pre-trained VADER model:
Status: Completed. The dashboard shows comparisons between different models, including VADER.

2. Developing the dashboard to present the sentiment analysis outputs in a user-friendly way:
Status: Completed. The dashboard provides various visualizations and interactive elements.

I'll provide a more detailed explanation of the dashboard technology we're using.

Dashboard Technology:

We are using Dash, a Python framework for building analytical web applications. Dash is built on top of Flask, Plotly.js, and React.js, which allows us to create interactive web-based dashboards using pure Python.

Comprehensive List of Components, Algorithms, Techniques, Libraries, and Modules:

1. Data Collection and Preprocessing:

1. Guardian API (for initial data collection)
2. Pandas (for data manipulation and cleaning)
3. NLTK (for text preprocessing)

2. Sentiment Analysis Models:

a. Rule-based Lexicon Model:

1. Custom financial lexicon
2. NLTK for tokenization

b. Machine Learning Model:

1. Scikit-learn (for implementing Multinomial Naive Bayes)
2. TfidfVectorizer (for feature extraction)
3. Financial PhraseBank dataset (for training)

c. VADER Sentiment Analysis:

1. vaderSentiment library

d. Hybrid Sentiment Analyzer:

1. Combination of rule-based and machine learning approaches

3. Unsupervised Learning (implemented but not fully utilized):

1. Scikit-learn (KMeans clustering)
2. Gensim (Word2Vec for word embeddings)

4. Dashboard Development:

1. Dash (main framework for the dashboard)
2. Plotly (for interactive visualizations)
3. Dash Bootstrap Components (for responsive layout and UI components)

5. Data Visualization Techniques:

1. Line charts (for sentiment trends)
2. Pie charts (for sentiment distribution)
3. Bar charts (for sentiment comparison across models)

6. Additional Libraries and Modules:

1. NumPy (for numerical computations)
2. Joblib (for model persistence)
3. DateTime (for handling date and time operations)

7. Algorithms and Techniques:

1. Text cleaning and normalization
2. Tokenization
3. Stop word removal
4. TF-IDF (Term Frequency-Inverse Document Frequency)
5. Naive Bayes classification
6. K-means clustering
7. Word embeddings

8. Project Structure and Organization:

1. Modular design with separate files for different components (models, preprocessing, dashboard)
2. Use of Python classes for encapsulating model functionality