Stock Market Analysis Dashboard

Project Overview
This is a comprehensive stock analysis dashboard built using Streamlit that provides various analytical tools and predictive capabilities for Indian stock market analysis. The project combines real-time data fetching, technical analysis, fundamental analysis, sentiment analysis, and machine learning predictions.

Technical Architecture

Core Technologies

1. Frontend: Streamlit
2. Data Sources: yfinance (Yahoo Finance API), GNews
3. Analysis Libraries**:
4. NLTK (Natural Language Processing)
5. scikit-learn (Machine Learning)
6. TA (Technical Analysis)
7. Plotly (Interactive Visualization)
8. Pandas & NumPy (Data Processing)

Main Components

1. Data Fetching Layer:

- `fetch_stock_data()`: Retrieves historical stock data using yfinance
- `fetch_market_data()`: Gets current market indices data

1. -Caching implemented using `@st.cache_data` for performance optimization

2. Analysis Modules:

- Market Status Analysis
- Current Price Tracking
- Historical Price Analysis
- Stock Comparison
- Time Series Analysis
- Fundamental Analysis
- Technical Analysis
- Predictive Analysis ("Gyaani Baba")

3. Sentiment Analysis System:

- Uses GNews for real-time news fetching
- NLTK's VADER sentiment analyzer for sentiment scoring
- Provides both individual article sentiment and aggregate sentiment metrics

Feature Details

1. Overall Market Status:

- - Displays real-time data for major indices (NIFTY, SENSEX, etc.)
- - Shows current prices, changes, and percentage changes
- - Includes an intraday NIFTY chart

2. Stock Analysis Features:

- Current Price: Real-time price tracking with news sentiment
- Price Between Dates: Historical price analysis with interactive charts
- Stock Comparison: Multi-stock comparative analysis
- Time Series Analysis: Trend analysis with visualizations
- Fundamental Analysis: Key metrics like P/E ratio, market cap, etc.

3. Technical Analysis:

- SMA (Simple Moving Average) - 50 and 200 days
- RSI (Relative Strength Index)
- MACD (Moving Average Convergence Divergence)
- Interactive charts for each indicator

4. News Sentiment Analysis System:

Input → News Fetching → Sentiment Analysis → Aggregation → Output

- News Fetching: Uses GNews API to get recent news articles
- Sentiment Analysis Process:
  1. Article title extraction
  2. VADER sentiment scoring (-1 to +1)
  3. Sentiment classification (Positive/Negative/Neutral)
  4. Aggregate sentiment calculation

5. Predictive Analysis System
The "Gyaani Baba" prediction system uses Random Forest Regression with the following features:
- Price data (Open, High, Low, Close, Volume)

- Technical indicators (SMA, RSI, MACD)
- Performance metrics (R-squared, RMSE)
- Future price predictions with confidence metrics

News Sentiment Analysis Deep Dive

General Concept

News sentiment analysis is a natural language processing (NLP) technique that determines the emotional tone and opinion expressed in news articles. It's particularly valuable in financial markets where news can significantly impact stock prices.

Process Flow:

1. Text Collection: Gathering relevant news articles
2. Preprocessing: Cleaning and normalizing text
3. Sentiment Scoring: Applying sentiment analysis algorithms
4. Aggregation: Combining individual scores into meaningful metrics

Implementation in This Project

The project uses VADER (Valence Aware Dictionary and sEntiment Reasoner), which is specifically attuned to social media and news text. The implementation:

1. Fetches recent news using GNews API
2. Analyzes article titles using VADER
3. Provides individual article sentiment scores
4. Calculates aggregate sentiment metrics
5. Uses sentiment data as a market sentiment indicator

LSTM (Long Short-Term Memory) Overview

General Concept

LSTM is a type of recurrent neural network (RNN) architecture designed to handle the vanishing gradient problem and effectively learn long-term dependencies in sequential data.

Key Components:

1. **Input Gate**: Controls what new information to store
2. **Forget Gate**: Controls what information to discard
3. **Output Gate**: Controls what information to output
4. **Cell State**: Long-term memory component
5. **Hidden State**: Short-term memory component

Potential Implementation for Stock Prediction

While the current project uses Random Forest, LSTM could be implemented for potentially better sequence prediction:

```python
def lstm_prediction(symbol, days=120):
    # Data preparation
    data = fetch_stock_data(symbol, ...)
    scaler = MinMaxScaler()
    scaled_data = scaler.fit_transform(data)

    # Create sequences
    X, y = create_sequences(scaled_data, sequence_length=60)

    # Build LSTM model
    model = Sequential([
        LSTM(50, return_sequences=True),
        LSTM(50),
        Dense(1)
    ])

    # Train and predict
    model.fit(X, y, epochs=100)
    predictions = model.predict(...)

    return scaler.inverse_transform(predictions)
```