Stock Movement Prediction using Social Media Sentiment Analysis

1. Introduction

The goal of this project is to predict the stock movement of Nvidia (NVDA) by analyzing public sentiment from social media posts and combining this with financial data. This report outlines the steps taken to collect, process, analyze, and model the data. The project integrates sentiment analysis from Reddit posts with historical financial data and predicts stock price movement using machine learning.

2. Data Collection

2.1 Web Scraping Process

The data collection process involves scraping Reddit posts for Nvidia-related content. The *nvidia_stock_scraper.py* script utilizes the Reddit API (PRAW library) to collect posts from subreddits such as 'stocks', 'investing', 'wallstreetbets', and others. The primary focus was to gather posts mentioning "NVIDIA", "NVDA", or related keywords.

Challenges Encountered:

- **API Limits:** The Reddit API has rate limits, which can restrict the number of posts that can be scraped in a given period. This was mitigated by limiting the number of posts retrieved per subreddit and managing the rate of requests.
- **Data Quality:** Not all posts were relevant, and some contained noisy or irrelevant text. This was addressed during the preprocessing phase by cleaning the text data and focusing on posts that provided sentiment-rich information.

Solution:

- The script filters out posts that have already been seen, ensuring that no duplicate data is collected. The final data is stored in a JSON file for further analysis.

2.2 Features Extracted

The following features were extracted from the Reddit posts:

- **Text Data:** The title and body of each post were combined into a single text field.
- **Engagement Metrics:** Upvotes and the number of comments were included to capture the level of engagement.
- **Post Timestamp:** The date of each post was collected to align with stock price data.

The key feature for analysis was the **sentiment of the posts**, which was determined using the Vader Sentiment Analyzer.

3. Sentiment Analysis

The *nvidia_stock_sentiment_analysis.py* script performs sentiment analysis on the text data. It uses the Vader Sentiment Analyzer to calculate sentiment scores, including positive, negative, neutral, and compound scores. The script also preprocesses the text by:

- Removing URLs and non-alphanumeric characters.
- Converting text to lowercase.
- Removing stop words and applying lemmatization.

Challenges Encountered:

- **Sentiment Noise:** Reddit posts can contain mixed sentiments, making it challenging to extract accurate sentiment. This was mitigated by aggregating sentiment scores at the daily level.
- **Missing Data:** Some posts lacked engagement metrics or sentiment scores. Missing values were imputed using median values for engagement and mean values for sentiment scores.
- Merged sentiment data with stock prices on a common *date* column.

Solution:

- The aggregated sentiment data was merged with Nvidia stock price data from Yahoo Finance (*yfinance*).

4. Stock Data Analysis

The stock data for Nvidia was fetched using the *yfinance* library, covering the period from 2010 to 2024. Several technical indicators were calculated:

- **Simple Moving Average (SMA):** A 5-day moving average.
- **Exponential Moving Average (EMA):** A 50-day EMA.
- **Relative Strength Index (RSI):** A 14-day RSI to measure momentum.
- **Bollinger Bands:** Calculated for price volatility.

The financial data was merged with the aggregated sentiment data on the date field, creating a comprehensive dataset for model training.

5. Model Training and Evaluation

The goal was to predict whether Nvidia's stock price would go up or down based on the features extracted from sentiment and financial data. A Random Forest Classifier model was used for this classification task.

5.1 Feature Engineering

Features used for prediction include sentiment scores (positive, neutral, negative, compound), engagement metrics (upvotes, comments), and technical indicators (SMA, EMA, RSI, Bollinger Bands, and

LogMomentum). The target variable was created by shifting the stock's closing price to predict the next day's movement (up or down).

5.2 Model Performance

The model was trained on the historical data, and performance metrics were evaluated using:

- **Accuracy:** The proportion of correct predictions.
- **Classification Report:** Precision, recall, and F1-score for the momentum prediction.

Challenges Encountered:

- **Overfitting:** The Random Forest model with a large number of trees and deep trees risked overfitting to training data. This was mitigated by tuning hyperparameters using RandomizedSearchCV.

Solution:

- The model achieved an accuracy of **56%**, and the classification report showed a balanced performance across both classes (up and down).

6. Code Breakdown and Methodologies

A. Data Collection: ***nvidia_stock_scraper.py***

Objective:

Scrape Reddit posts discussing NVIDIA using specific keywords from relevant subreddits.

Process:

1. **Reddit API Integration:**
   - Used the *praw* library to interact with Reddit's API. Authentication credentials were configured for secure access.
   - Scraped subreddits like *stocks*, *investing*, and *techstock* for NVIDIA-related discussions.
2. **Keyword Matching:**
   - Used a predefined list of keywords like *NVIDIA, NVDA,* and *NVIDIA stock* to filter posts.
   - Ensured relevant data by iterating over multiple subreddits and keywords.
3. **Data Extraction:**
   - Extracted metadata such as *title, text, upvotes, num_comments,* and *created_utc*.
   - Filtered duplicate posts using a *seen_post_ids* set to avoid redundancy.

Output:

- Saved data as *nvidia_stock_posts.json* in JSON format for subsequent processing.

B. Data Preprocessing and Sentiment Analysis: ***nvidia_stock_sentiment_analysis.py***

Objective:

Prepare data for analysis and derive sentiment scores from Reddit discussions.

Process:

1. **Data Loading and Cleaning:**
   - Loaded the JSON file into a Pandas DataFrame.
   - Combined *title* and *text* into a single column for comprehensive sentiment analysis.
2. **Text Preprocessing:**
   - Removed URLs, non-alphanumeric characters, and numeric values.
   - Applied stopword removal and lemmatization using NLTK.
3. **Sentiment Analysis:**
   - Used the ***VADER*** sentiment analyzer to compute *compound, positive, neutral,* and *negative* scores for each post.
   - Aggregated scores by date to correlate sentiment trends with stock performance.

Output:

- Generated a ***stock_analysis.csv*** file containing both sentiment metrics and stock indicators.

C. Feature Engineering: ***nvidia_stock_sentiment_analysis.py***

Objective:

Enhance data with additional features for better predictive modeling.

Features Added:

1. **Technical Indicators:**
   - **Simple Moving Average (SMA):** Captures short-term price trends.
   - **Exponential Moving Average (EMA):** Emphasizes recent price data.
   - **Relative Strength Index (RSI):** Measures price momentum.
   - **Bollinger Bands:** Represents price volatility.
   - **Log Momentum:** Quantifies price movement over a 5-day window.
2. **Handling Missing Data:**
   - Imputed missing sentiment scores with mean values.
   - Filled gaps in upvotes and comments using the median.

D. Model Training: ***model_training.py***

Objective:

Build a machine learning model to predict stock movement (up or down) based on engineered features.

Process:

1. **Feature Selection:**
   - Selected features like sentiment scores, upvotes, comments, and technical indicators for modeling.
   - Defined *momentum* as the target variable, indicating whether the next day's closing price increased.
2. **Data Splitting and Scaling:**
   - Split the data into training (80%) and testing (20%) sets.
   - Used *MinMaxScaler* to normalize features for better model performance.
3. **Model Selection and Hyperparameter Tuning:**
   - Choose *RandomForestClassifier* for its robustness in handling structured data.
   - Performed hyperparameter tuning using *RandomizedSearchCV* to optimize parameters like:
     - Number of estimators (*n_estimators*).
     - Maximum tree depth (*max_depth*).
4. **Evaluation Metrics:**
   - Computed *accuracy_score* and *classification_report* to evaluate model performance.
   - Focused on precision and recall to assess the balance of predictions.

Output:

- Trained model saved as a *.joblib* file for deployment or further use

E. User Input Prediction: ***user_predict_input.py***

- **Purpose:** Provides a user-friendly interface for stock prediction.
- **Process:**
  - Accepts real-time sentiment scores and technical indicator values.
  - Normalizes the input using the same *MinMaxScaler* as during training.
  - Loads the pre-trained Random Forest model using *Joblib*.

Output:

- A binary prediction: ***1*** (price increase) or ***0*** (price decrease).

7. Results and Insights

1. **Model Performance:**
   - Achieved an accuracy of approximately 56% on the test set.
   - The model effectively captured sentiment-driven market movements but faced challenges with noisy data.
2. **Key Findings:**
   - Positive sentiment correlated with upward stock movement.

    - Technical indicators like RSI and Bollinger Bands significantly contributed to prediction accuracy.

6. Suggestions for Future Work

6.1 Data Expansion

- **Additional Social Media Platforms:** To improve the model, future work can incorporate data from Twitter, StockTwits, or other platforms where investors discuss Nvidia stock.
- **More Financial Indicators:** Including other technical indicators such as MACD, moving averages for longer periods, and volume-based indicators could provide more insight into price movement.
- **Fundamental Data:** Including Nvidia's earnings reports, P/E ratio, and other financial metrics could improve the model's ability to predict stock price movements.

6.2 Model Improvements

- **Advanced Models:** Instead of Random Forest, models like XGBoost, LSTM (for time series analysis), or reinforcement learning could be explored for better accuracy and performance.
- **Sentiment Analysis Enhancements:** Refining the sentiment analysis to include fine-grained categories (e.g., extreme positive/negative sentiments) may provide a better indication of market movements.

6.3 Performance Monitoring

- **Live Prediction Integration:** A live prediction system that updates daily or weekly based on new Reddit posts and stock data could be implemented to keep the model updated and relevant.

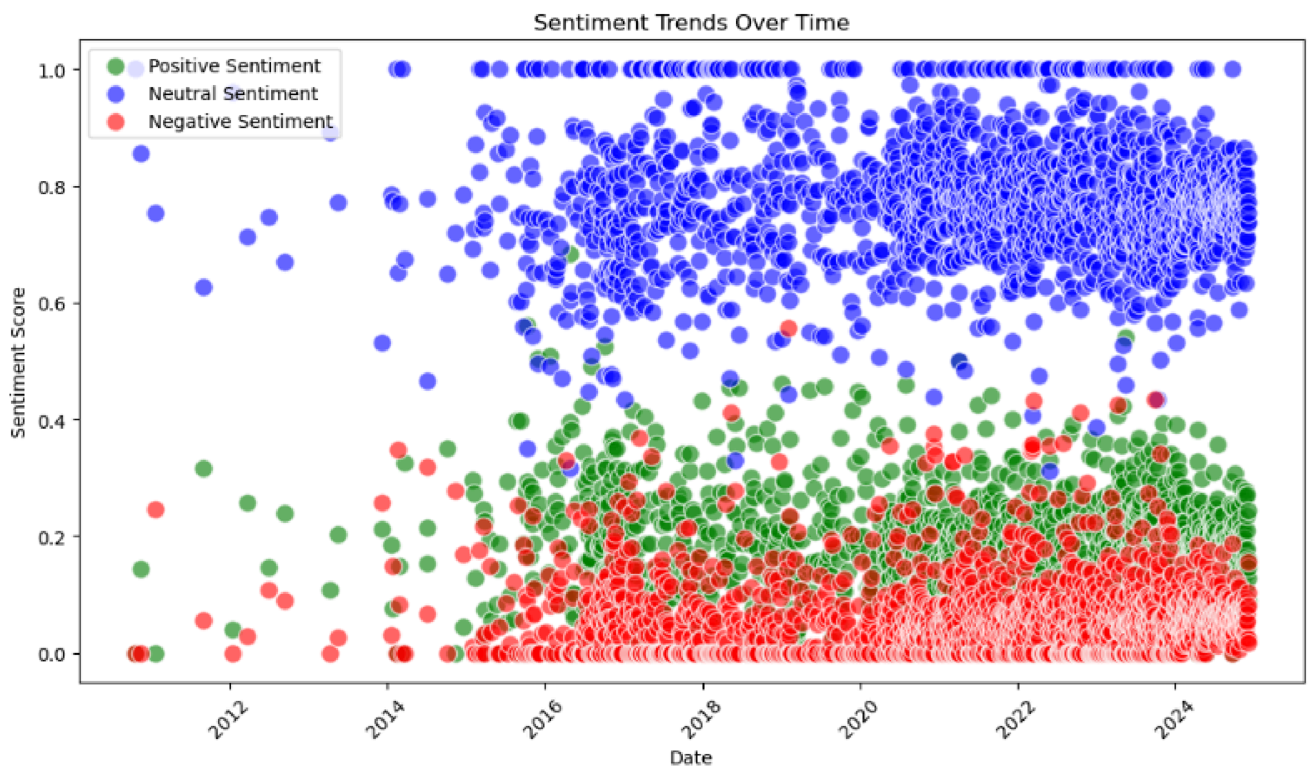
7. Conclusion

This project demonstrated the potential of combining sentiment analysis from social media with traditional financial data to predict stock movements. The model successfully predicted the direction of Nvidia's stock price with decent accuracy, though improvements in model sophistication and data quality could enhance its predictive power.

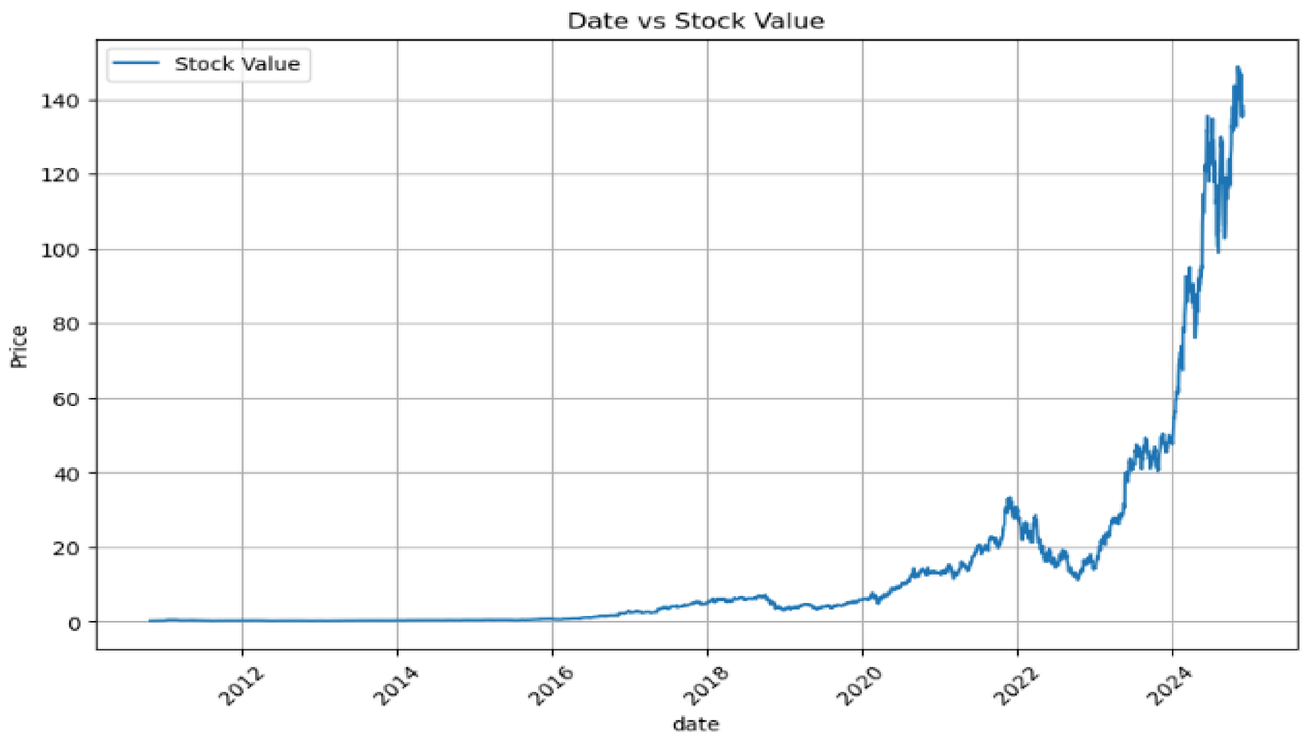
8. Appendices

8.1 Diagrams and Charts

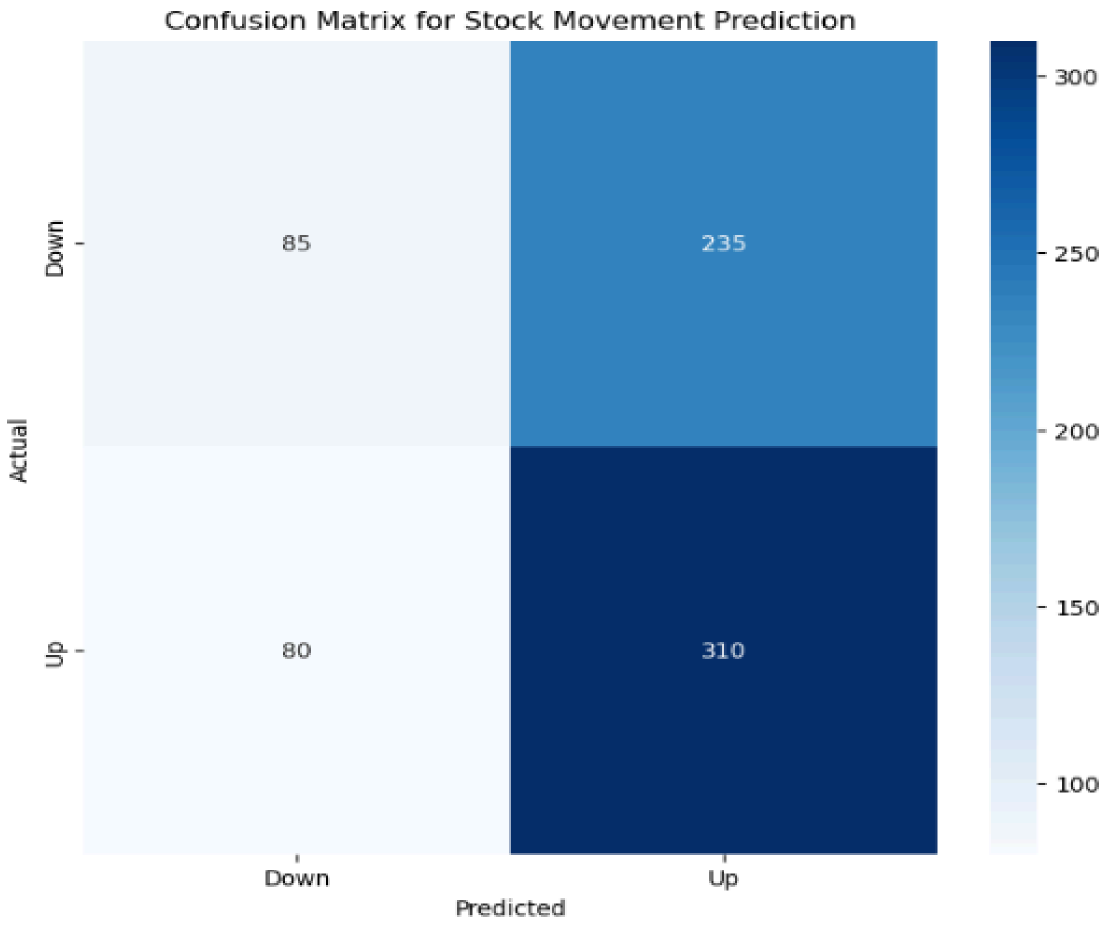
- **Sentiment Distribution by Date:** A chart showing the sentiment trends over time.



- **Stock Price vs. Time:** A graph illustrating the relationship between time and stock price movements.



- **Confusion Matrix for Stock Movements Prediction:**



8.2 Code Files

- *Jupyter_Notebooks_Scripts.ipyn*b: Complete code for data processing, analysis, and visualization.
- *nvidia_stock_scraper.py*: Scraping script to gather Reddit posts.
- *nvidia_stock_sentiment_analysis.py*: Script for sentiment analysis.
- *model_training.py*: Code for training and evaluating the machine learning model.
- *User_input_predict.py*: Predict next day trend