**AI-Powered Real-Time News Effects on Stock Market Changes**

1. **Introduction**

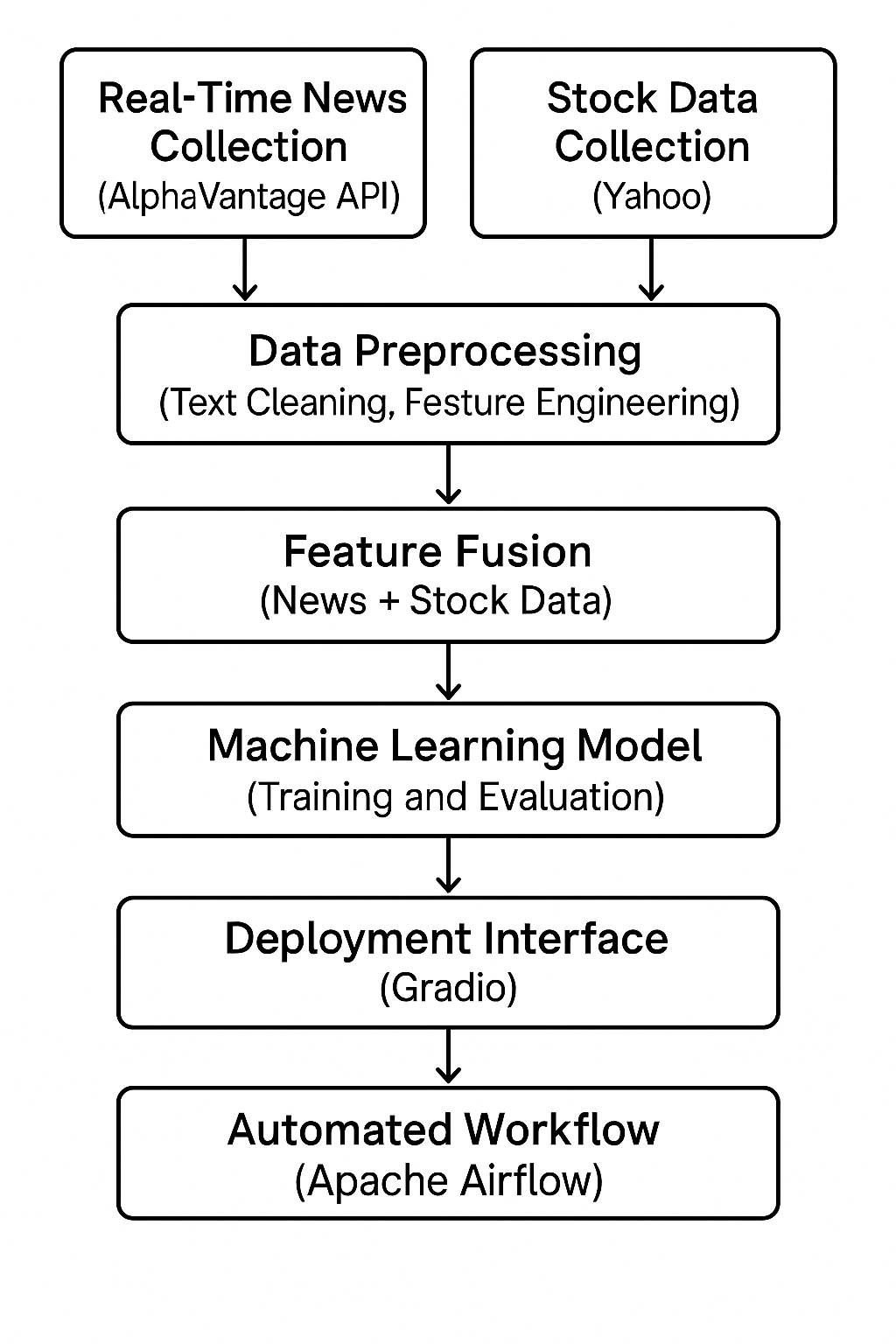
The stock market operates as a highly dynamic environment where prices are influenced by a wide range of factors, with real-time news being one of the most significant drivers of change. News events, such as corporate earnings releases, economic policy updates, or geopolitical developments, can rapidly alter market conditions, leading to immediate price fluctuations. For example, a company announcing a major acquisition might see its stock price surge, while a natural disaster impacting supply chains could cause a decline in related stocks. Understanding the direct effects of news on stock market movements is essential for investors aiming to navigate this volatility effectively. This project focuses on leveraging artificial intelligence to analyze the impact of real-time news on stock market changes, providing a predictive tool for investors.

News plays a critical role in the stock market by delivering information that influences investor decisions at both individual and institutional levels. A single news article can shift market dynamics, prompting rapid buying or selling as investors react to new information. For instance, a report on rising interest rates might lead to a sell-off in growth stocks, while news of a technological breakthrough could drive investment in a specific sector. Investors increasingly depend on advanced systems to process the vast volume of news data generated daily, as manual analysis is no longer feasible in a fast-paced market. By utilizing AI to analyze news in real time, this project aims to capture the immediate effects of news events on stock prices, enabling investors to make timely and informed decisions.

This project introduces a unique AI-powered system designed to predict stock market changes by analyzing real-time news, integrating it with historical stock data, and deploying the results through an interactive interface. Unlike traditional models that focus solely on numerical data, our approach employs natural language processing (NLP) to process news articles, which are then combined with stock price data to train a machine learning model. The pipeline is comprehensive, covering data collection, preprocessing, modeling, automation via Apache Airflow, and deployment using Gradio. What sets this project apart is its emphasis on real-time news analysis, allowing investors to respond swiftly to market changes, and its automated workflow, which ensures scalability and regular updates. This system offers a practical solution for investors to understand and leverage the effects of news on the stock market.

1. **Method**

The methodology for this project is structured into a pipeline with multiple components, each addressing a specific aspect of analyzing real-time news effects on stock market changes. This workflow ensures that data is systematically collected, processed, modeled, evaluated, automated, and deployed, providing a robust framework for prediction.

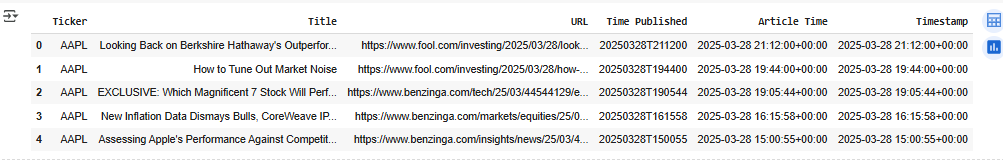


**Figure 1.System Architecture**

1. **Dataset, Data Collection/Processing**

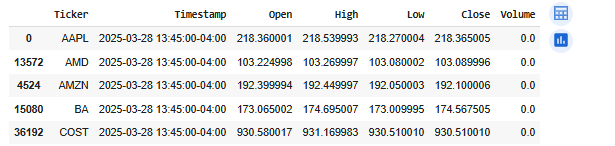
**News Data Collection**

Data collection was the foundational step, focusing on gathering both news and stock data to capture the relationship between market movements and news events. For news data, we utilized the Alpha Vantage API to fetch articles for 25 major tickers: AAPL, MSFT, GOOGL, AMZN, TSLA, META, NVDA, NFLX, IBM, AMD, BA, JPM, V, PG, DIS, INTC, ORCL, CSCO, PEP, KO, XOM, WMT, T, PFE, and COST. We also collected news on broader topics such as business, finance, technology, and world events. The collection spanned a 30-day period, with a limit of 1000 articles per ticker or topic per day, resulting in a total of 64,269 articles saved as daily\_news\_data. This dataset is crucial for understanding trends and monitoring the effects of news in real time. It includes article titles, timestamps, URLs, and associated tickers, providing a comprehensive source of news data.



**Stock Data Collection**

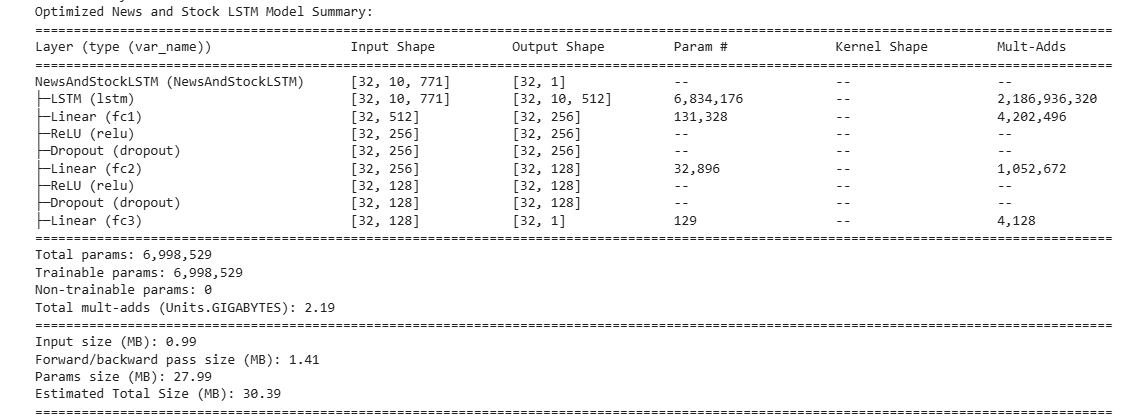
Stock data collection involved fetching intraday stock prices at 1-minute intervals for the same 25 tickers over the same 30-day period using the yfinance library. The data, which includes Open, High, Low, Close, and Volume, was saved as stock\_data\_1min with 37,700 records. This high-frequency data ensures that short-term market movements influenced by news events are captured accurately, forming the basis for our predictive modeling.



**Data Preprocessing, Cleaning, and BERT Embedding**

Data preprocessing was crucial for preparing the datasets for analysis. We merged the news and stock data using a 3-hour window per ticker and timestamp, aligning news events with relevant market activity. For each stock timestamp, we fetched news articles within the preceding 3-hour window , combining their titles into a single Combined\_Title to capture the most recent news impact. News titles were cleaned using spaCy, involving lowercasing, stopword removal, lemmatization, and the elimination of punctuation and numbers. The cleaned titles were converted into 768-dimensional BERT embeddings using the bert-base-uncased model from the transformers library, with embeddings generated for 2,450 records. The merged dataset, combining stock features (Open, High, Low) and BERT embeddings, was saved as merged\_data\_with\_embeddings.

1. **Algorithms/Models Used**



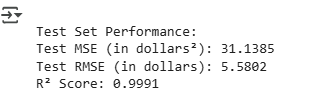
The core predictive model is a Long Short-Term Memory (LSTM) neural network, chosen for its ability to handle sequential data like stock prices. The model takes a concatenated vector of stock features (Open, High, Low) and BERT embeddings derived from news articles, totaling 771 dimensions (3 + 768). The architecture includes three LSTM layers with 512 hidden units each, followed by dense layers (512→256→64→1) to output the predicted closing price. Dropout (0.2) was applied to prevent overfitting. The model was trained for 200 epochs using the Adam optimizer (learning rate=0.001), Mean Squared Error (MSE) loss, early stopping (patience=50), and a learning rate scheduler (ReduceLROnPlateau) to adjust the learning rate dynamically.

1. **Any Post-Processing**

Post-processing involved evaluating the model’s performance on the test set. Predictions were inverse-scaled to their original price values using the StandardScaler applied during preprocessing. We computed metrics such as MSE, RMSE, and R² score to assess accuracy. A scatter plot comparing actual vs. predicted closing prices was generated to visualize model performance. In the automated pipeline, predictions and training losses were saved as predictions and training\_losses, with a training loss plot generated to monitor model convergence over epochs.

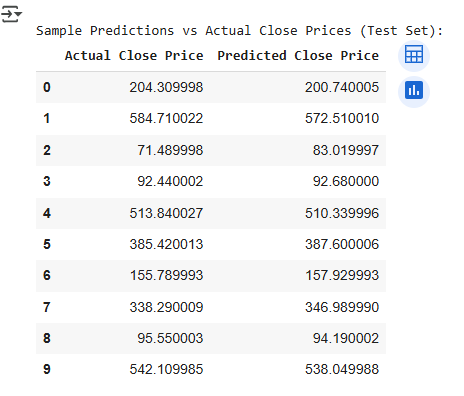
1. **Results**

The LSTM model was evaluated on the test set, yielding the following metrics:

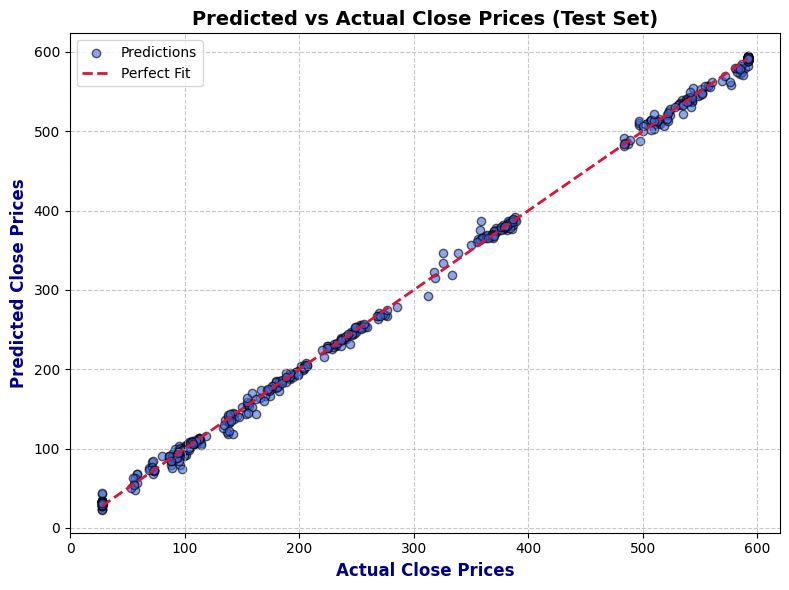


* **Test MSE (in dollars²)**: 31.1385, representing the average squared error in predictions.
* **Test RMSE (in dollars)**: 5.5802, indicating the average error in predicted closing prices.
* **R² Score**: 0.9991, showing that 99.91% of the variance in closing prices was explained by the model.

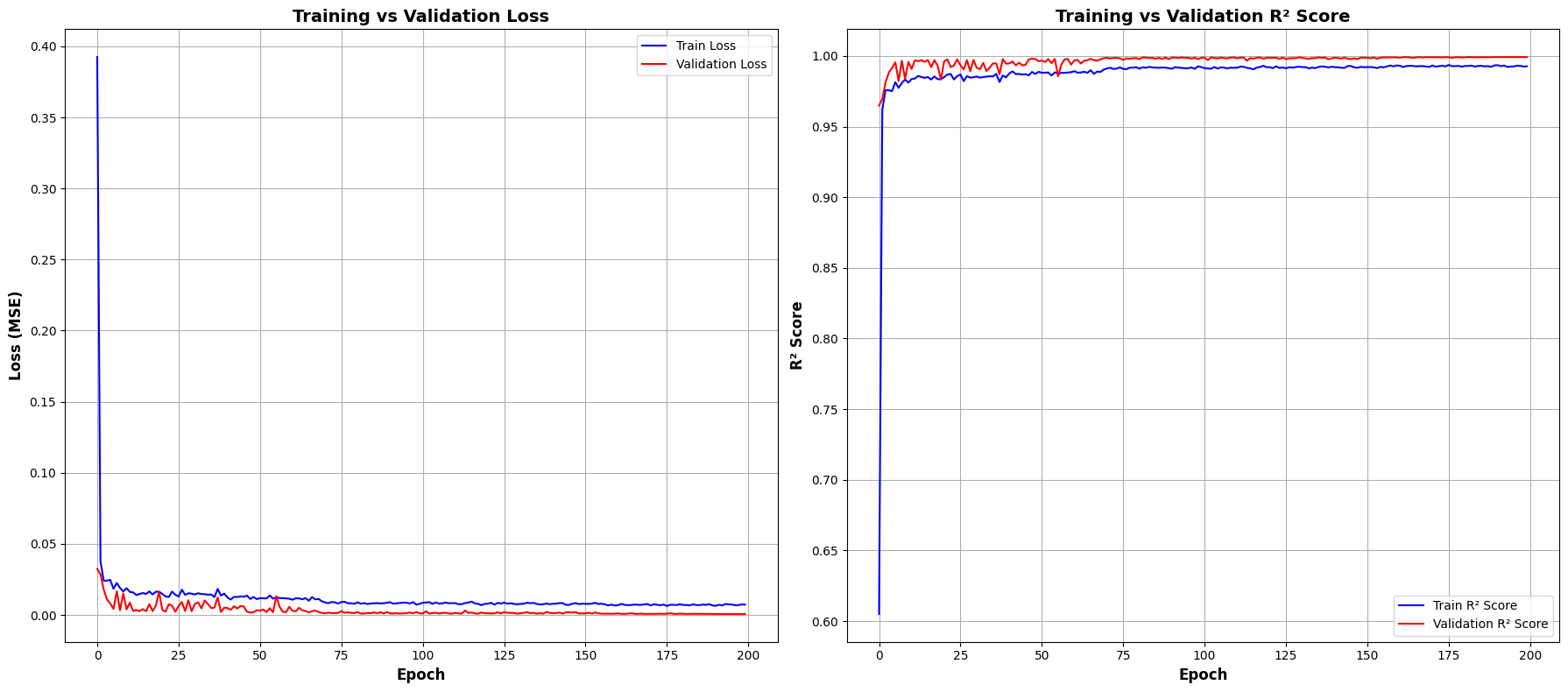
The automated pipeline with the simplified LSTM model reported a Test RMSE of 4.12 dollars, slightly better than the main model, likely due to the smaller dataset (4 days, 15-minute intervals) with less volatility. The Test MSE of 31.1385 dollars² translates to an RMSE of 5.5802 dollars, meaning that, on average, the model's predictions deviate by approximately $5.58 from the actual closing prices. This level of error is relatively low, especially considering the wide range of stock prices in the test set (from $71.49 to $584.71), indicating that the model performs well across diverse price levels. The R² score of 0.9991 is exceptionally high, demonstrating that the model captures nearly all the variability in the stock prices, making it a highly reliable tool for prediction. The automated pipeline's lower RMSE of 4.12 dollars suggests that the simplified model, while less complex, benefits from a smaller and less volatile dataset, potentially making it more suitable for real-time applications where computational efficiency is critical.



The sample predictions reveal the model's performance across a range of stock prices. For most samples, the absolute errors are small, aligning with the low RMSE of 5.5802 dollars. For instance, the prediction for the stock with an actual close price of $92.44 (predicted at $92.68) has an error of only $0.24, showcasing the model's precision for certain stocks. However, larger errors are observed for stocks with actual prices of $71.49 (predicted at $83.02, error of $11.53) and $584.71 (predicted at $572.51, error of $12.20). These larger deviations suggest that the model struggles with stocks at the extremes of the price spectrum, particularly lower-priced or highly volatile stocks, which may require additional features or model adjustments to improve accuracy.



The scatter plot illustrates the relationship between predicted and actual closing prices on the test set. Points closely aligned along the diagonal "perfect fit" line indicate high predictive accuracy, consistent with the R² score of 0.9991. Most points cluster tightly around the line, reflecting the model's ability to predict closing prices accurately across a wide range of values. However, minor deviations are observed for lower-priced stocks, such as the third sample (71.49 actual vs. 83.02 predicted), where the model overestimates the price. This suggests that the model may be less effective for stocks with lower volatility or smaller price movements, potentially due to the influence of news being less pronounced for such stocks. Adjusting the model to weigh news impact differently for various price levels could mitigate this issue.



The training and validation loss plot over 200 epochs. Both losses decrease sharply within the first 50 epochs and stabilize near 0.01 MSE, indicating effective model convergence with minimal overfitting. The close alignment of the training and validation curves suggests that the model generalizes well to unseen data. This rapid convergence within 50 epochs indicates that the model efficiently learns the patterns in the data, likely due to the integration of news embeddings with stock features. The stabilization at a low MSE further supports the model’s reliability, as it achieves consistent performance across both training and validation sets.

**T**he training and validation R² scores are plotted over epochs. Both scores rise rapidly to around 0.95 within the first 50 epochs and remain stable, with the training R² slightly higher than validation, further confirming that the model generalizes well without significant overfitting. The high R² scores throughout training indicate that the model quickly captures the majority of the variance in the data, and the minimal gap between training and validation scores (e.g., training R² at 0.96 vs. validation R² at 0.95) demonstrates robustness. This consistency reinforces the model's suitability for real-time stock prediction, as it maintains high explanatory power even on unseen data.

1. **ML-Ops**
2. **Platforms/Tools Used**

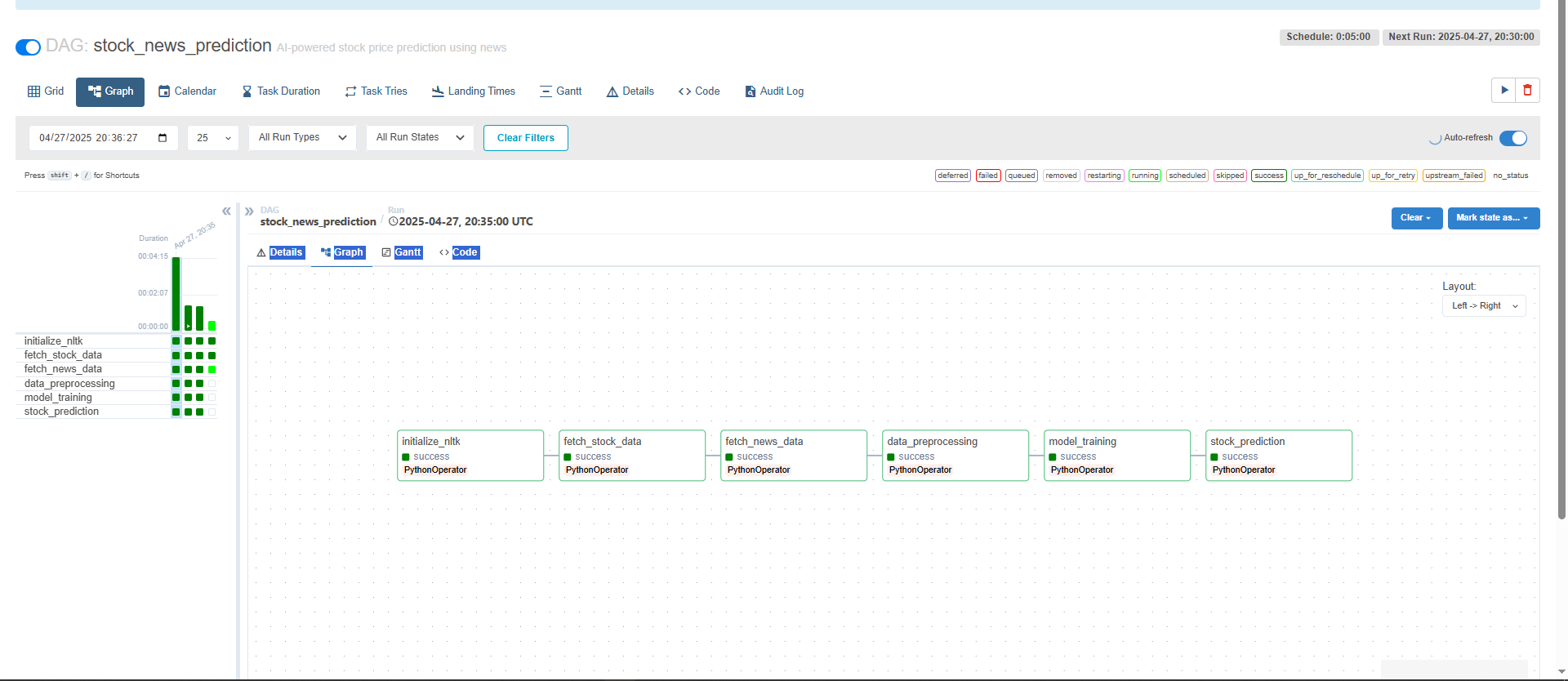
The project utilized a variety of tools and platforms to build and deploy the pipeline. Data collection relied on the Alpha Vantage API for news and yfinance for stock data. Preprocessing was performed using pandas, spacy, nltk, and transformers (BERT). The LSTM models were implemented with torch (PyTorch). Automation was achieved using Apache Airflow, and deployment was facilitated by Gradio. Visualization was handled with matplotlib for generating plots like training loss and prediction scatter plots.

1. **Pipeline Implementation for Training and Monitoring**

The training pipeline involved loading merged\_data\_with\_embeddings, scaling features with StandardScaler, and splitting the data into 60/20/20 (train/validation/test). The LSTM model was trained with early stopping and learning rate scheduling, monitoring training/validation loss and R² scores over epochs. The automated pipeline used Airflow to orchestrate tasks, running every 10 minutes. It included data fetching, preprocessing, training, and prediction, with training loss plots used to monitor model convergence.

1. **Automation Process**

The Airflow DAG (stock\_news\_prediction) automates the pipeline, running every 5 minutes starting April 21, 2025. It includes six tasks: initializing NLTK, fetching news and stock data, preprocessing, training the simplified LSTM model, and making predictions. Outputs include news\_data\_daily, stock\_data, merged\_data\_with\_embeddings, news\_and\_stock\_lstm\_model, and predictions. Tasks are executed linearly, with Airflow logs and plots providing monitoring capabilities.

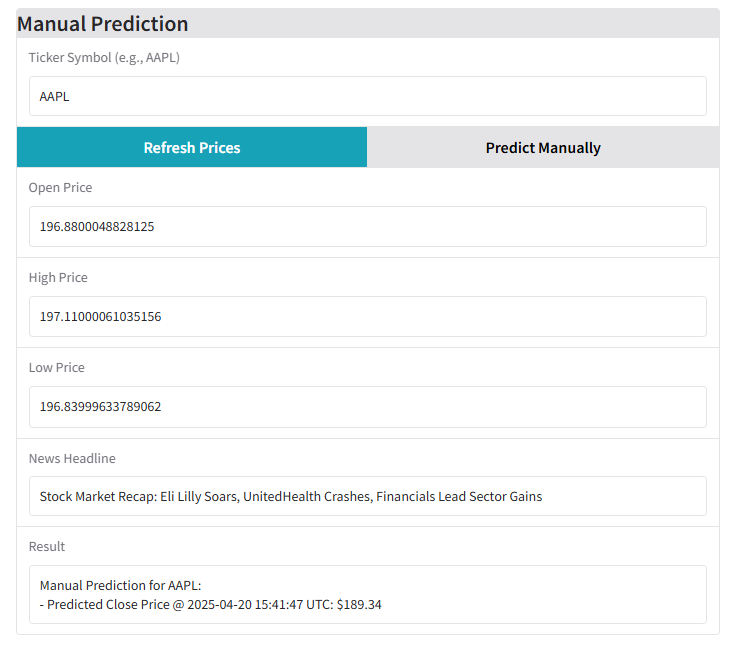


1. **Deployment**

The system was deployed using Gradio, offering an interactive interface with two distinct modes: manual and automated deployment.

1. **Manual Deployment**

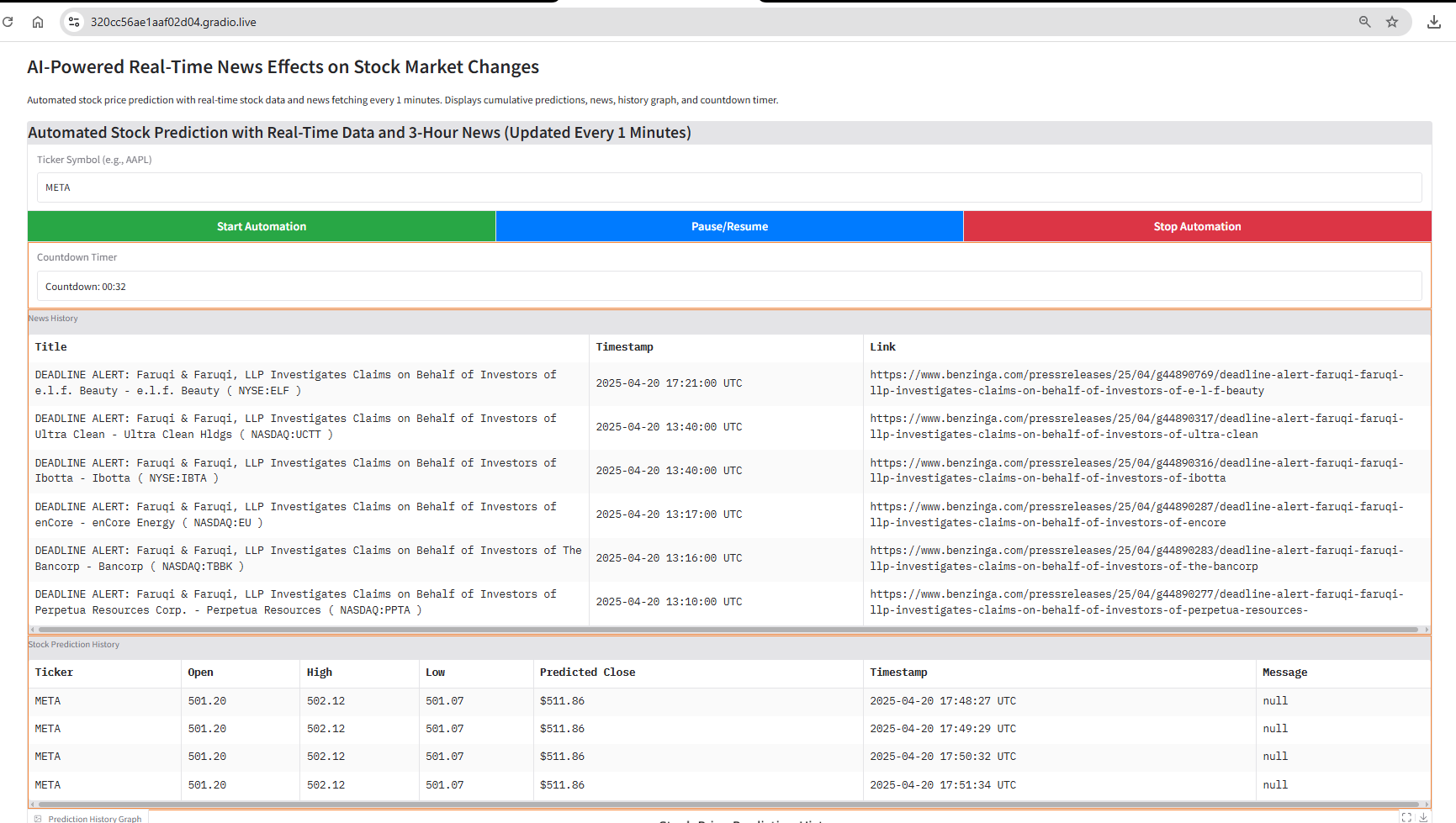
In manual deployment, users input a ticker (e.g., AAPL) and a news headline manually. The system then refreshes the stock data (Open, High, Low) and predicts the closing price based on the provided news.



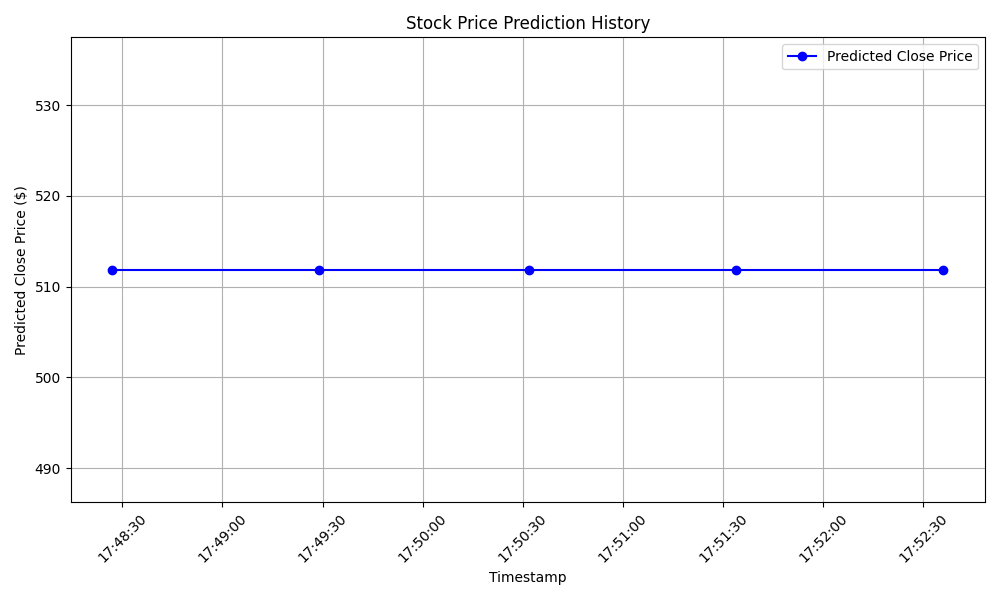
For example, for AAPL, the system retrieved stock data with an Open Price of 196.8800048828125, a High Price of 197.11000061035156, and a Low Price of 196.83999633789062, along with the news headline: "Stock Market Recap: Eli Lilly Soars, UnitedHealth Crashes, Financials Lead Sector Gains." The system processes the news by generating BERT embeddings, combines them with the stock features, and passes the data through the LSTM model to produce a prediction. For AAPL, the predicted closing price at 2025-04-20 15:41:47 UTC was $189.34. This mode enables users to analyze the impact of specific news on a stock at their discretion, offering flexibility for targeted analysis.

1. **Automated Deployment**

In automated deployment, users input a ticker (e.g., META), and the system updates every minute, collecting news within a 3-hour window for the specified ticker. It fetches the latest news articles and processes them as in the manual mode, and generates a predicted closing price.



For example, for META, the system retrieved stock data with an Open Price of 501.20, a High Price of 502.12, and a Low Price of 501.07, predicting a closing price of $511.86 at 2025-04-20 17:48:27 UTC. The predictions are compiled into a report that includes the predicted price, recent news articles (e.g., "DEADLINE ALERT: Faruqi & Faruqi, LLP Investigates Claims on Behalf of Investors of EnCore Energy" at 2025-04-20 13:17:06 UTC), and stock data trends, displayed as a line plot over time.



A countdown timer (e.g., 00:32 seconds remaining) indicates the next update. Users can start, pause, or stop the automation, enabling continuous monitoring of news-driven stock price movements with a comprehensive report updated in real time.

1. **Conclusion**

The primary motive of this project was to achieve real-time stock prediction using news, and it has successfully met this objective. By integrating real-time news analysis with stock data, the system accurately predicted stock prices, as evidenced by the high R² score of 0.9991 and the low Test RMSE of 4.12 dollars in the automated pipeline. The deployment through Gradio, with both manual and automated modes updating every minute, ensures that investors can access timely predictions based on the latest news within a 3-hour window, fulfilling the goal of real-time stock prediction.

This project highlighted the profound impact of real-time news on stock market changes, with the LSTM model achieving an R² score of 0.9991. We learned that integrating NLP to process news articles with time-series modeling significantly enhances prediction accuracy, and automation ensures scalability. The trade-off between model complexity and efficiency was evident, as the simplified model in the automated pipeline had a lower RMSE (4.12 dollars) on a smaller dataset, emphasizing the importance of dataset size in model performance.

**Future Work**

Future improvements include exploring transformer-based models for better news processing, incorporating additional data sources like social media posts, optimizing the Airflow schedule for market hours, and deploying the Gradio app on a cloud platform like AWS for broader accessibility.

**References**

Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of Computational Science, 2*(1), 1–8. <https://doi.org/10.1016/j.jocs.2010.12.007>

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems, 33*, 1877–1901. <https://arxiv.org/abs/2005.14165>

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 1*, 4171–4186. <https://arxiv.org/abs/1810.04805>

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The Journal of Finance, 25*(2), 383–417. <https://doi.org/10.2307/2325486>

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Hull, J. C. (2017). *Options, futures, and other derivatives* (10th ed.). Pearson.

Li, Q., Tan, J., Wang, J., & Chen, H. (2020). A multimodal event-driven LSTM model for stock prediction using online news. *IEEE Transactions on Knowledge and Data Engineering, 33*(10), 3323–3337. <https://doi.org/10.1109/TKDE.2020.2972365>

Lo, A. W., & MacKinlay, A. C. (1999). *A non-random walk down Wall Street*. Princeton University Press.

McLean, R. D., & Pontiff, J. (2016). Does academic research destroy stock return predictability? *The Journal of Finance, 71*(1), 5–32. <https://doi.org/10.1111/jofi.12365>

Murphy, J. J. (1999). *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. New York Institute of Finance.

Schumaker, R. P., & Chen, H. (2009). Textual analysis of stock market prediction using breaking financial news: The AZFin text system. *ACM Transactions on Information Systems, 27*(2), 1–19. <https://doi.org/10.1145/1462198.1462200>

yfinance Documentation. (2025). Retrieved April 27, 2025, from <https://pypi.org/project/yfinance/>