Stock Price Forecasting Using Deep Learning and Sentiment Analysis

Group 38
CS 182 – Introduction to Machine Learning
Yang Jiao, Yuechen Wu, Quan Li

***Abstract*—This paper explores the impact of integrating news sentiment into machine learning models for daily stock price prediction. We evaluate the performance of XGBoost, LSTM, and CNN-LSTM architectures, each with and without FinBERT-extracted sentiment features. All models are trained and tested in an online learning setup using one year of financial data for AAPL, MSFT, and GOOGL. The goal is to assess whether sentiment enhances predictive accuracy and which model configuration yields the best results.**

I. Introduction

Stock prices are notoriously volatile and influenced by myriad factors, making short-term prediction extremely challenging. Nevertheless, accurate stock price forecasting is of great interest due to the potential for significant financial gains. In recent years, researchers and practitioners have increasingly looked beyond purely historical price patterns by integrating textual sentiment data (e.g., news or social media sentiment) into predictive models [1].

Seminal work by Tetlock [2] provided early evidence that the tone of news media can predict market movements – in particular, high levels of media pessimism were found to robustly predict downward pressure on stock prices. Subsequent studies have corroborated that investor sentiment extracted from daily news narratives can significantly influence stock price swings [3]. Quantitative hedge funds have also recognized the value of such alternative data signals; for example, Two Sigma, a prominent quant fund, sponsored a public data science competition to leverage news analytics for stock return prediction [4].

Parallel to these developments, advances in machine learning (ML) and deep learning have provided powerful tools for modeling complex, non-linear patterns in financial time series. On one hand, gradient-boosted decision tree algorithms like XGBoost [5] have become popular for regression and forecasting tasks due to their efficiency and accuracy on structured data. On the other hand, deep neural networks have shown exceptional promise in sequence modeling: Long Short-Term Memory (LSTM) networks, a type of recurrent neural network, are well-suited for capturing long-term temporal dependencies in stock data and often outperform traditional time-series models [6]. Researchers have progressively moved from single-model approaches to hybrid architectures that combine the strengths of multiple methods. For instance, a CNN-LSTM architecture can fuse a convolutional neural network's ability to extract local features from price sequences with an LSTM's ability to model temporal dynamics, thereby achieving improved trend prediction performance [7].

Another key development has been in natural language processing for sentiment analysis. Transformer-based language models like BERT have revolutionized text analysis, and domain-specific variants such as FinBERT have achieved state-of-the-art results in financial sentiment classification [8]. FinBERT is a BERT model pre-trained on large financial corpora, which allows it to recognize nuanced sentiment in finance-related text and outperform generic sentiment models on financial data. It can categorize news headlines as positive, negative, or neutral in sentiment, effectively quantifying the market sentiment reflected in news. The availability of such advanced tools enables researchers to incorporate more reliable sentiment signals into forecasting models.

Indeed, recent studies have directly examined the impact of adding news sentiment features in stock prediction models by comparing performance with and without sentiment inputs [9]. These comparisons generally find that sentiment-enriched models outperform their price-only counterparts, confirming that textual sentiment contributes useful predictive information. For example, Gu *et al.* [9] introduced a hybrid FinBERT-LSTM model that integrates news-derived sentiment with historical pricing data, and reported an enhancement in prediction precision by leveraging the sentiment features.

Given this background, the aim of our project is to rigorously evaluate whether incorporating news sentiment can improve the accuracy of stock price forecasts, and to compare the effectiveness of different modeling approaches in doing so. In particular, we focus on three widely-used predictive models – a boosted tree model and two deep learning models (including a hybrid) – and test each model in two settings: with vs. without sentiment input features. By examining these 2 $times3$ combinations (sentiment/no-sentiment × XGBoost, LSTM, CNN-LSTM), we can quantify the value-added by news sentiment and identify which type of model benefits most from it.

The study is conducted using one year of daily data for three major technology stocks (Apple, Microsoft, and Google), with sentiment features extracted from daily news headlines via FinBERT. Ultimately, this work seeks to determine whether news sentiment analytics (powered by FinBERT) can boost short-term stock price prediction performance, and how the performance of a classic ML model compares to modern

neural network models in this context.

II. Methodology

Our methodology involves an empirical comparison of six forecasting model variants, arising from the combination of three modeling techniques with the inclusion or exclusion of sentiment features. The three base models chosen represent a mix of state-of-the-art approaches in stock prediction: a tree-based ensemble model, a recurrent neural network model, and a hybrid CNN-RNN model. We pair each of these with either purely price-based features or augmented features including news sentiment. This 2
$times$3 experimental design allows us to isolate the impact of the sentiment feature for each model type.

All models are deployed in an online learning framework, meaning they are updated incrementally over time rather than trained just once offline—an approach that mimics how a model in production could be retrained as new daily data arrives.

The models and tools used can be summarized as follows:

A. XGBoost

This is a gradient-boosted decision tree algorithm known for its high efficiency and predictive performance in regression tasks
citechen2016xgboost. XGBoost builds an ensemble of shallow trees in sequence, each correcting errors of the previous, to capture non-linear relationships in the data. It has proven successful in many time-series forecasting competitions and serves as a strong ML baseline for our study. In our implementation, XGBoost is used to predict the next day's price based on numerical features, and in the sentiment-augmented case it can naturally incorporate the sentiment score as an additional input feature.

B. LSTM

Long Short-Term Memory networks are a type of recurrent neural network designed to handle sequential data and long-range dependencies. An LSTM maintains an internal memory state that enables it to learn temporal patterns over extended sequences without vanishing gradient issues
citefischer2018deep. LSTMs have been widely applied in stock market prediction, where they can learn complex patterns such as trends and seasonality from historical price sequences. In our setup, the LSTM model ingests a window of past days' prices (and sentiment values, if applicable) and outputs a forecast for the next day's price. The network's architecture (number of LSTM layers, units, etc.) and training epochs were tuned based on validation performance.

C. LSTM-CNN

This is a hybrid model combining a one-dimensional Convolutional Neural Network with an LSTM, aiming to exploit both local feature extraction and long-term sequence modeling citelu2020cnn. The CNN component acts on the time-series data (e.g., a sliding window of past prices) to detect short-term patterns or motifs—such as sudden jumps or local trends—by applying convolutional filters. The LSTM component then takes the CNN-extracted features (sequentially over time) and learns the temporal dependencies among those higher-level features. By uniting the spatial pattern recognition of CNNs with the temporal memory of LSTMs, CNN-LSTM architectures have achieved superior performance in capturing stock price dynamics. Our CNN-LSTM model uses a small convolutional network (with a few filters and kernel size on the order of days) whose outputs feed into an LSTM layer, which then predicts the next day's price. As with the plain LSTM, we test this model with and without the sentiment feature included in its input sequence.

D. FinBERT (Financial BERT)

FinBERT is a specialized language model for financial text, obtained by fine-tuning the general BERT model on large-scale financial corpora
citearaci2019finbert. It is used in our pipeline to extract daily sentiment scores from news headlines. FinBERT can classify a given piece of news text as having positive, negative, or neutral sentiment relevant to financial markets. We leverage a pre-trained FinBERT model to process all news headlines for each stock on each day. Each headline's sentiment is inferred (e.g., categorized as Positive, Negative, or Neutral) and then mapped to a numeric sentiment score (for instance, +1, –1, or 0 respectively) that reflects its polarity. We aggregate these scores across all headlines of a given stock for a given day (e.g., by averaging) to derive a single daily sentiment feature. This approach yields a time series of sentiment values aligned with the stock price data. By using FinBERT's domain-specific capabilities, we aim to capture more accurate and context-aware sentiment signals from the news, as generic sentiment analyzers often misclassify financial jargon.

The daily FinBERT sentiment score is then included as an additional input feature for the "with sentiment" model variants. (For the "no sentiment" models, this feature is omitted.)

In all cases, the predictive target is the stock's next-day closing price, and the models are trained to minimize prediction error on this continuous target. We employ a rolling online learning procedure during the test phase: the models are first trained on the training dataset (historical data up to a cutoff), and then they make day-by-day predictions on the test set in chronological order. After each day's prediction is made and the actual price for that day is revealed, we update the model by incorporating that new data point (this could mean retraining or incrementally updating the model parameters) before predicting the following day.

This simulates a real-world scenario where a model is regularly updated with the latest information, and it helps prevent performance degradation that could occur if a model trained only on older data is applied far into the future. The tree-based model (XGBoost) is updated by training on new data points (taking advantage of its ability to continue training with additional boosting rounds), while the neural networks

are fine-tuned on the new day or retrained on an expanding window including the new observation. All model retraining uses only information that would be available up to that point in time to avoid look-ahead bias.

We carefully tuned each model's hyperparameters using the validation set. For XGBoost, this involved parameters like the number of trees, max tree depth, and learning rate. For LSTM and CNN-LSTM, we experimented with the length of the input window (e.g., using the past 5 days vs. past 10 days of prices), the number of LSTM layers and hidden units, dropout rates to mitigate overfitting, and the training epochs. We selected configurations that gave the lowest error on the validation data for each model. These chosen hyperparameters were then fixed when training on the combined training+validation data and evaluating on the test set (to ensure the test results reflect a model that had no direct knowledge of test outcomes).

III. Data

We evaluate our models on daily stock data combined with daily news headline sentiment for three well-known companies: Apple (AAPL), Microsoft (MSFT), and Google/Alphabet (GOOGL). The time span of the dataset covers approximately one year of trading days for each stock.

Using the `yfinance` API (which provides programmatic access to Yahoo Finance), we retrieved daily price data for each stock, including features such as the opening price, closing price, highest and lowest prices, and trading volume. In our study, we focus primarily on the *closing price* as the variable to predict and use recent closing prices—along with potentially derived technical indicators such as returns or moving averages—as input features. Each stock's price series is aligned with its corresponding news data by date.

For the same date range, we collected news headlines related to each of the three companies. These headlines were obtained from public news feeds, such as Yahoo Finance's news section for each ticker, which aggregates relevant financial articles. We filtered headlines by stock ticker or company name to ensure relevance. Since the goal is daily forecasting, we grouped news by day: all news headlines for stock X on date D are treated as the news corpus for X on that day. While not all headlines carry equal informational weight, for simplicity, we treated each as contributing equally to the sentiment score of that day.

Using FinBERT, we computed a *daily sentiment score* for each stock. Every headline was fed into the FinBERT model to infer its sentiment classification (positive, negative, or neutral). These classes were mapped to numeric values: $+1$ for positive, 0 for neutral, and -1 for negative. The sentiment scores of all headlines pertaining to a given stock on a given day were then averaged, yielding a single scalar sentiment feature for that day. This daily score serves as a proxy for overall market sentiment toward the company on that day, with values above zero indicating bullish sentiment, values below zero indicating bearish sentiment, and near-zero values reflecting mixed or neutral views.

This daily sentiment feature is merged with the stock's price data. Specifically, the input feature set on day D may include a sequence of closing prices from the past n days and, optionally, the sentiment score on day D. In our experimental setup, we align sentiment with the day *prior* to prediction: to forecast the closing price on day $D+1$, we provide the model with day D's sentiment score and recent price history up to day D. This alignment ensures that the model only has access to information available before the prediction date, preventing any look-ahead bias.

After constructing the dataset of combined price and sentiment features, we split the data chronologically into training, validation, and test sets in a $70\%/20\%/10\%$ ratio. For example, in a 252-trading-day year, approximately 176 days would be used for training, 50 for validation, and 26 for testing. This temporal split mimics realistic forecasting workflows, where future data remains unseen during model development. The training set is used to fit model parameters, the validation set is used for hyperparameter tuning and model selection, and the test set is used only for final evaluation.

We ensured the chronological integrity of all splits to prevent leakage of future information. For instance, no data from 2024 would be used in any form during model training or validation if predictions are made on 2024 data.

Prior to model training, we applied basic preprocessing. We handled missing values and removed non-trading days to maintain consistency across time series. Input features were scaled appropriately: raw price values were normalized or transformed (e.g., log-returns or percent changes) and then scaled to zero mean and unit variance using statistics computed from the training set. This scaling prevents model training from being affected by absolute price magnitudes. The sentiment feature, which is naturally bounded between -1 and $+1$, did not require additional scaling. All transformations were fitted on the training set and applied to validation and test data using the same parameters, thereby preserving proper training-test isolation.

IV. Empirical Study

In our experimental setup, we train and evaluate all six model configurations—XGBoost, LSTM, and CNN-LSTM, each with and without sentiment features—on each of the three target stocks: AAPL, MSFT, and GOOGL. The task is to perform one-day-ahead prediction of the closing price, using information up to the current day.

During the training phase, each model is fitted on the training data using its respective configuration. For neural networks (LSTM and CNN-LSTM), early stopping based on validation loss is employed to prevent overfitting. For the XGBoost model, the number of boosting rounds is determined similarly using the validation set.

Each model is then evaluated in a rolling online prediction fashion on the test set. On each day D of the test period:

1) The model receives the input feature vector constructed from day D (including price history and sentiment, if applicable).

2) It outputs a forecast for the closing price on day $D+1$.
3) After observing the true closing price for day $D+1$, the model is updated to incorporate this new observation (either through partial retraining or incremental fitting).

This procedure continues sequentially until the end of the test set, emulating a live deployment scenario in which the model makes predictions on future data and continually learns from new market conditions.

We evaluate prediction performance using two common regression metrics:

- **Mean Absolute Error (MAE)**:

$$\text{MAE} = \frac{1}{T}\sum_{t=1}^{T}|\hat{y}_t - y_t|$$

- **Root Mean Squared Error (RMSE)**:

$$\text{RMSE} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\hat{y}_t - y_t)^2}$$

where $\hat{y}_t$ is the predicted price and y_t is the true price on day t.

These metrics are computed across the entire test set for each stock and model variant. MAE measures the average absolute deviation between predicted and actual values, while RMSE penalizes larger errors more strongly. Both offer insights into model reliability.

Our comparative analysis focuses on two core questions:

- Does incorporating FinBERT-based sentiment improve forecast accuracy?
- Which model family—XGBoost, LSTM, or CNN-LSTM—delivers superior performance under each setting?

By comparing paired versions of each model (with and without sentiment), we assess the marginal value of sentiment features. Additionally, cross-model comparisons allow us to identify which architectures are better suited for this sentiment-aware forecasting task.

All models are tuned using the same protocol, trained on identical data segments, and evaluated over the same test dates to ensure fairness in comparison.

V. Results

(The results is in Table1.)

TABLE I
MAE and RMSE of Different Methods

Method	Sentiment	DataSet	MAE	RMSE
LSTM	No	APPL	0.0067	0.0078
LSTM	No	GOOGL	0.0112	0.0113
LSTM	No	MSFT	0.0122	0.0123
LSTM	Yes	APPL	0.0055	0.0055
LSTM	Yes	GOOGL	0.0112	0.0113
LSTM	Yes	MSFT	0.0122	0.0123
CNN+LSTM	No	APPL	0.0087	0.0099
CNN+LSTM	No	GOOGL	0.0073	0.0081
CNN+LSTM	No	MSFT	0.0132	0.0141
CNN+LSTM	Yes	APPL	0.0066	0.0066
CNN+LSTM	Yes	GOOGL	0.0102	0.0103
CNN+LSTM	Yes	MSFT	0.0137	0.0138
XGBoost	No	APPL	0.0013	0.0017
XGBoost	No	GOOGL	0.0053	0.0072
XGBoost	No	MSFT	0.0025	0.0032
XGBoost	Yes	APPL	0.0066	0.0066
XGBoost	Yes	GOOGL	0.0102	0.0103
XGBoost	Yes	MSFT	0.0137	0.0138

VI. Discussion

1. Without Sentiment Factors

- **XGBoost** achieved the best performance across all datasets, with the lowest MAE and RMSE. This indicates its high efficiency and stability in handling raw time-series data.
- **LSTM** and **CNN+LSTM** showed relatively higher errors but still demonstrated good predictive performance. This suggests that, in the absence of sentiment factors, these deep learning models are competitive for time-series prediction tasks, albeit slightly less accurate than XGBoost.

2. With Sentiment Factors

- **APPL Dataset**: **LSTM** performed the best, with significant reductions in MAE and RMSE. This highlights the positive impact of sentiment factors on predictive performance for this dataset.
- **GOOGL and MSFT Datasets**: **LSTM** still outperformed other models, but the changes in MAE and RMSE were minimal. This suggests that sentiment factors have limited impact on predictive performance for these datasets.
- **XGBoost** experienced a significant increase in MAE and RMSE after incorporating sentiment factors, indicating a negative impact on model performance. This may be due to incompatibility between sentiment factors and the model's structure or feature handling.

3. Impact of Sentiment Factors

- **APPL Dataset**: The addition of sentiment factors significantly enhanced model performance, highlighting their importance for predictive accuracy.
- **GOOGL and MSFT Datasets**: The impact of sentiment factors was limited or even negative, suggesting weak correlations between sentiment factors and stock price fluctuations or the need for optimized sentiment factor handling.

4. CNN+LSTM and LSTM Models

- The **CNN+LSTM** model first uses **CNN** to extract local spatial features and then employs **LSTM** to model temporal relationships, capturing complex patterns more effectively. In contrast, **LSTM** processes univariate time-series data (e.g., daily stock prices) and captures long-term dependencies through memory units, making it suitable for learning patterns across time steps.
- In this experiment, the data consisted of a single time-series (only closing prices) with smooth trends and low

noise. As a result, **LSTM** outperformed **CNN+LSTM** in some cases. However, **CNN+LSTM** generally outperforms **LSTM** in capturing short-term technical patterns in stock prices in practical applications.

- After incorporating sentiment factors, the performance improvement of **CNN+LSTM** was greater than that of **LSTM**. This is attributed to the convolutional layers in **CNN** that integrate sentiment factors with other technical features more effectively.

5. XGBoost

- **XGBoost** is a gradient boosting ensemble learning algorithm based on decision trees, whereas **LSTM** and **CNN+LSTM** are recurrent neural networks in deep learning. These models fundamentally differ: one is a tree-based model, and the other is a neural network-based model.
- **XGBoost** excels at handling structured feature data and can easily incorporate technical indicators and date features. In contrast, **LSTM** is more suited for pure time-series data. In this experiment, **XGBoost** performed best without sentiment factors but saw a significant decline in performance after their inclusion, suggesting the need for optimized sentiment factor handling.

CONCLUSIONS

The impact of sentiment factors on different models and datasets varies significantly. For the **APPL** dataset, sentiment factors notably enhance the predictive performance of **LSTM** and **CNN+LSTM**. However, for the **GOOGL** and **MSFT** datasets, the impact is minimal or negative. Moreover, **XGBoost** struggles with sentiment factors, indicating potential incompatibility. Future research should explore the compatibility of sentiment factors with different models and optimize their handling to improve overall model performance.

REFERENCES

[1] K. R. Dahal *et al.*, "A comparative study on effect of news sentiment on stock price prediction with deep learning architecture," *PLOS ONE*, vol. 18, no. 4, p. e0284502, 2023.

[2] P. C. Tetlock, "Giving content to investor sentiment: The role of media in the stock market," *The Journal of Finance*, vol. 62, no. 3, pp. 1139–1168, 2007.

[3] S. S. Ragunathan *et al.*, "Effects of daily news sentiment on stock price forecasting," *arXiv preprint arXiv:2308.08549*, 2023.

[4] Two Sigma Investments, "Using news to predict stock movements," https://www.kaggle.com/competitions/two-sigma-financial-news, 2019, accessed: 2025-05-01.

[5] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD)*, 2016, pp. 785–794.

[6] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018.

[7] W. Lu *et al.*, "A cnn-lstm-based model to forecast stock prices," *Complexity*, vol. 2020, p. Article ID 6622927, 2020.

[8] D. T. Araci, "Finbert: Financial sentiment analysis with pre-trained language models," *arXiv preprint arXiv:1908.10063*, 2019.

[9] W. Gu, Y. Zhong, S. Li *et al.*, "Predicting stock prices with finbert-lstm: Integrating news sentiment analysis," *arXiv preprint arXiv:2407.16150*, 2024.