Transformer-Based News-Sensitive Stock Price Prediction

Huijie Pan* Yifan Wang* Haichao Ji* Leheng Lu* Baihan Liu*
University of Michigan, Ann Arbor
{phjus, evanwyf, haichao, leheng, baihanl}@umich.edu

1. Introduction

The stock market is known for its unpredictability and complexity. In order to make better trading decisions, from the starting day of the stock market, people are making continuous efforts to analyze patterns of the market and make price predictions. However, this ambitious task requires a tremendous amount of both financial and technical knowledge to understand the hidden trends that the market follows. It's an extremely dynamic research field that has huge potential for people to explore.

On the other hand, precise stock price prediction is much more than purely empirical data analysis. Professional news reporters and journalists are publishing insights into the market. Some journals effectively reflect the orientation of the market. Some suggest the absolute opposite. After hundreds and thousands of viewers read their articles, it will definitely make a difference in their trading strategies. The sentiments of these news reports will definitely affect buyers' behavior and more or less impacts the market.

To accommodate both technical stock market history purely based on data and relevant sentiment factors from news reports. We are proposing a deep-learning approach that fuses both numerical and textual information using Transformer to make accurate stock price predictions.

2. Related Work

There has been extensive research on stock market prediction over the past, including the use of machine learning and deep learning methods to try to explore an accurate way to predict the stock market. Accurate stock market predictions can help investors make informed decisions about buying and selling stocks, thereby maximizing their profits and minimizing their risks.

Stock market prediction is based on a prediction of time series data. Researchers have been exploring different traditional machine learning approaches such as support vector machine [8] [1] and K-Nearest Neighbor [1]. These methods utilize feature engineering and traditional machine learning models to predict the stock market.

2.1. Support Vector Machine(SVM)

Support Vector Machines are a type of supervised machine learning algorithm used for both classification and regression analysis. They usually work by finding the optimal hyperplane in a high-dimensional space that separates the different classes of data. SVMs are effective in predicting stock prices for the following reasons. Stock price prediction involves analyzing a large number of features, such as past stock prices, trading volumes, and technical indicators. SVMs are able to handle this high-dimensional data and identify patterns. In addition, stock prices can be influenced by a wide range of factors, including unexpected news events and changes in market sentiment. Furthermore, SVMs are effective in dealing with this kind of noisy or overlapping data. Complex and nonlinear factors can influence stock prices. Introducing kernel functions in SVMs can make a nonlinear classification problem linearly separable by transforming the original space into a high-dimensional space. Researchers used correlation-based SVM filters and quasi-linear SVMs on Taiwan stock market datasets, and the experimental results demonstrated a good feature selection and prediction power[8]. However, SVMs can be computationally expensive, sensitive to outliers, and require careful tuning of parameters to achieve optimal performance[10].

2.2. Long Short-Term Memory(LSTM)

Long Short-Term Memory (LSTM) is a recurrent neural network variant designed to retain long-term dependencies. It has an additive connection to previous memories passed through time. LSTM is capable of capturing long-term dependencies, handling sequential data, and learning from past mistakes, which is particularly useful in stock price prediction, where long-term trends and patterns are important. Besides, LSTM can be trained using different input features and configurations and provide insight into which features are important in predicting stock prices. It

*Equal contribution.

is useful for understanding the underlying factors driving stock prices. Researchers proposed a combined effort of using efficient machine learning techniques coupled with LSTM to use them to predict the stock prices with a high level of accuracy[5]. However, LSTM lacks a mechanism to index the memory while writing and reading the data. The number of memory cells is linked to the size of the recurrent weight matrices[10].

2.3. Hybrid model with ARIMA and Neural Network

Autoregressive Integrated Moving Average (ARIMA) is a statistical method designed for time series data, which combines three parameters p (the number of autoregressive terms), d (the number of differences), and q (the number of moving average terms). It's trained on historical data to estimate parameters and then can be used to predict future data based on them. Although ARIMA is a powerful model for stationary time series data, the stock price data are usually non-stationary. Also, there are some other limitations of predicting stock prices by ARIMA[9]. ARIMA is a linear model that it assumes a linear relationship on the past and future data, while the stock prices data usually have a non-linear pattern due to some external factors like event news and the performance of companies. It only takes historical values into consideration to predict future values. Besides, ARIMA models are most powerful for short-term prediction, while they are limited for longer-term forecasting tasks. Accordingly, some hybrid models like ARIMA-LSTM [3] are suggested to be more effective in predicting stock prices by capturing non-linear patterns using neural networks. The neural network can address images (CNN) and texts (RNN) containing useful information for prediction.

2.4. Transformer

In recent years, researchers have focused on studying deep neural networks to predict the stock market. Especially after the transformer architecture is introduced [11], it has become the most popular deep learning architecture to solve the problem. Transformer is known for handling sequential data. Its encoder-decoder and positional encoding architecture can be able to handle long sequences much better than recurrent neural networks. The self-attention mechanism will also help detect long-range dependencies, which is particularly important for NLP tasks that involve understanding the context of words and sentences. It is promising that the same feature will be beneficial for time-series data which has similar behavior and characteristics as NLP tasks.

2.5. Linear Model

More recently, in 2022-2023, ideas suggesting that the Transformer model is not suitable for long-term time series forecasting tasks are starting to emerge. These opinions state that we have to give credit to Transformer as one of the most successful models in semantic association tasks. But the attention mechanism naturally "forgets" some time-dependent information and thus may struggle to analyze trends of time-series data. To validate their claims, several simple linear models are proposed. The outcomes have shown that some of these simple architectures can actually outperform some well-designed time-series transformers. LTSF-Linear [12] uses a single-layer fully connected neural network, and TiDE [2] proposed a residual encoder-decoder structure. They both yield impressive results compared to transformer-based models. At the same time, another advantage of these linear models is they significantly reduce the computation needed to run and train the model. This is great news for users who don't have access to substantial computational powers.

3. Proposed Method/Approach

3.1. Data Collection

To begin with, we collect stock data from 2014-01-01 to 2015-12-31 from Yahoo Finance by utilizing the yfinance library to download the data. These data comprise the essential features that aid us in building our baseline model. To enhance our data with more numerical features, we use the Stockstats library to compute various stock indicators, such as the Moving Average Convergence/Divergence (MACD) and Relative Strength Index (RSI). More than 80 indicators are generated from the original dataset, which is later utilized in feature engineering. Apart from the numerical features, we also acquire text data in the form of tweet data from various open-source datasets.

3.2. Data Preprocessing

3.2.1 Data Features and Features Creation

The followings table are descriptions of stock price data features. Table 1 shows the features of the original data set. Table 2 shows the supported Statistics/Indicators.

For each point of price in the data set, its n pre-days price value were used to fit an ARIMA model. For each ARIMA model, several combination of parameters (p,d,q) were attempted and the best one with the lowest AIC value was selected to predict the price on the following day. Unlike the hybrid model with ARIMA used to predict the price for the residuals which are fitted by LSTM methods later in those related work, predicted values by their corresponding ARIMA models compose a feature called "arima_res" in our model. Also, the residuals of ARIMA models were obtained as another feature to possibly test which were better to improve the model.

For the tweet textual data, it is crawled from Twitter API. Every company has several news each day. All Twit-

Feature	Description
Date	The date of the trading day
High	The highest stock price of the trading day.
Low	The lowest stock price of the trading day.
Open	The open price of the trading day.
Close	The close price of the trading day.
Adj Close	The adjusted close price of the trading day.
Volume	The volume of stock traded during the trading day.

Table 1. Variables of the Original Data Set

Feature	Description
macd	Moving Average Convergence Divergence
boll_ub	Upper Bollinger Bands
boll_lb	Lower Bollinger Bands
rsi_30	30 Periods Relative Strength Index
cci_30	30 Periods Commodity Channel Index
dx_30	30 Periods Directional Index
close_30_sma	30 periods Exponential Moving Average of the Close
close_60_sma	60 periods Exponential Moving Average of the Close
arima_res	predicted results by ARIMA model based on n pre-days
residuals	the difference between the ground truth and arima_res

Table 2. Supported Statistics/Indicators

ter APIs that return Tweets provide that data encoded using JavaScript using JavaScript Object Notation(JSON).

3.2.2 Exploratory Data Analysis

We first perform some exploratory data analysis for stock price. For a single stock analysis, the stock Apple Inc.(AAPL) is used for illustrations.

- Daily, weekly, yearly periodic trend of the stock price are plotted using additive model Prophet, as shown in Figure 1.
- Anomaly detection shows that there are no missing values and outliers in this dataset.
- Distribution of daily return of stock price are plotted, as shown in Figure 2, and it approximately follows a normal distribution and most values are between -0.03 and 0.03.
- Heatmap are plotted, as shown in Figure 4, to find the correlations between different stock features. From the heatmap, we find that the variables Open, High, Low, boll_ub, boll_lb, close_30_sma, close_60_sma and arima_res have relatively large positive correlations with the predicted variable Adj Close. The variable Volume has relatively large negative correlation with Adj Close.
- Principal component analysis are used to lower the dimensions of the data and get the amount of variance explained by each of the selected components.We choose the variable that are highly pairwise correlated and have large correlations with the predicted variable Adj Close to perform principal component analysis. These variables are Open, High, Low, Volume, boll_ub, boll_lb, close_30_sma and close_60_sma. There is an elbow in the scree plot at the 3th principal component, which suggests that there maybe little benefit to examining more than four or more principal components. From the three dimensions visualization plot, as shown in Figure 5, we can see that the adjusted close price is highly related to the first principal components.

Next, we analyze the relationship between different stocks.

- The adjusted close price of multiple stocks over time are plotted, as shown in Figure 3 and their stock prices are in different scale.
- Hierarchical clustering method, as shown in Figure 6, is used to cluster the stocks based on their dissimilar matrix. We can divide the stocks into multiple clusters by drawing a cutting read line. In this dendrogram, the red line divide the stocks into five clusters. Stocks in one cluster are relatively similar to each other.

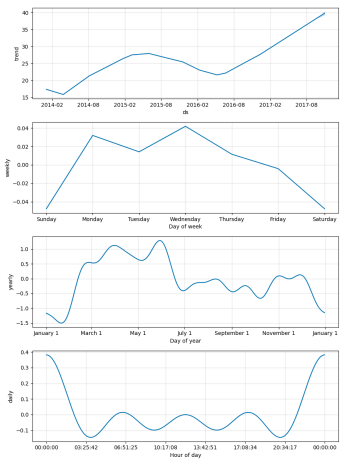

Figure 1. Periodic Trend of the Stock Price.

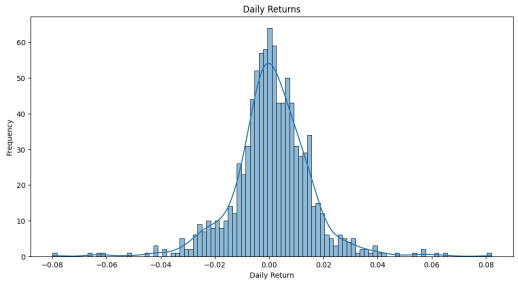

Figure 2. Distribution of Daily Return of the Stock Price.

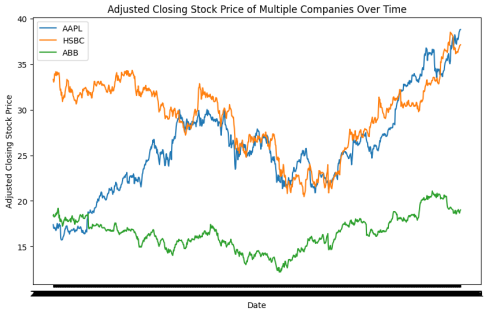

Figure 3. Adjusted Closing Stock Price for Multiple Stocks.

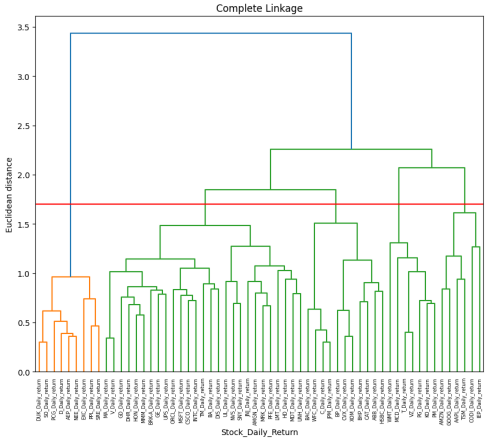

Figure 6. Stock Similarity Dendrogram.

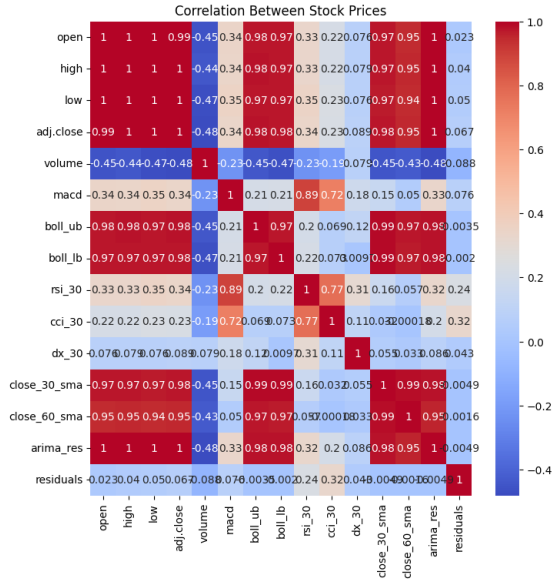

Figure 4. Heatmap of features.

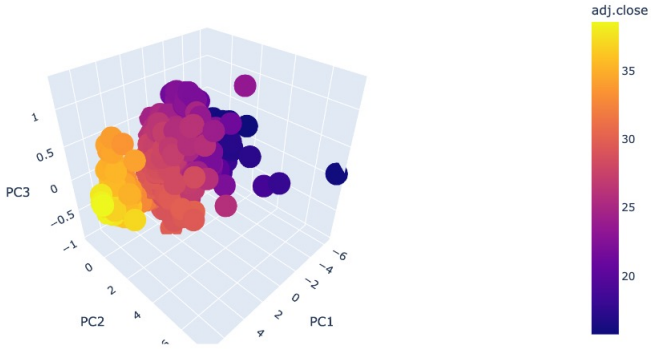

Figure 5. 3D Visualiation of Principal Component Analysis.

3.2.3 Preprocessing Steps

Based on the findings in exploratory data analysis, we perform the following data preprocessing tasks for stock price data.

- Delete the dates in price data set on which the stock price change rate of these dates are less than 0.5% from the previous dates to reduce the noise in stock price.
- Balance the up and down of the stock price so that accuracy would be a good evaluation metric for our classification tasks.
- Use Min-Max Scaler to normalize our data so that the prices of each company are on the same scale.
- Process data using 5, 15, and 30 lookback sliding windows which is a technique often used in time series

analysis to process data by breaking it into smaller, overlapping sections or windows of fixed length.

For the tweet data, news corresponding to the dates and stocks are selected in stock price data set. We use the pre-trained Bert [4] model to pre-process the text data and include the embedding as features with the base features.

3.3. Models

Predicting stock market prices is a highly complex task. To construct a thorough model, we endeavor to consider as many factors that can influence stock market prices as possible. Along with numerical data from the stock market, we also aim to incorporate news associated with each stock. Social text data, such as news articles and comments, play a crucial role in the stock market. Recently, there has been extensive research on how to combine numerical and text data in stock market prediction.

3.3.1 Transformer

The Transformer architecture has proven to be a potent tool in various deep-learning tasks. Nonetheless, it is not adept at handling time series data and modeling the relationship between each timestep. Consequently, we employ Time2Vec [7] to encode the stock time series data. Similar to LSTM, we train Transformer models using both the original dataset and the augmented dataset. Additionally, we aim to predict the stock price for the next P days rather than solely the next day. The Transformer architecture is illustrated in Figure 7.

The process of historical stock embedding preprocesses the stock data and generates a sequence of length L as the lookback window with D dimensional features. The self-attention encoder network receives the window data from the $[1, L-1]$ day, and subsequently, the data from the Lth day, in addition to the encoder output, are used as input to the decoder. The decoder outputs the prediction of the stock price for the next P days.

3.3.2 LTSF-Linear

The structure 8 of LTSF-Linear[12] is fairly straightforward. Same pre-processed historical stock price embedding technique as the transformer model, generating a sequence of L length look-back window with the feature dimension D as input. Build a fully connected neural network that has a total of 3 layers: input layer, hidden layer, and output layer. The input layer has the dimension of $L-1$ size with D channels. Then, fully connect the input layer to the hidden layer that has dimension 1 with D channels. Finally, connect all these D channels into a single scalar which will be our final prediction. There's also a normalized version called NLinear. The only change to the original LTSF-Linear is subtracting the input by the last value of the sequence. Then, the input goes through a linear layer, and the subtracted part is added back before making the final prediction. The paper states that this will improve the performance when there is a distribution shift in the dataset. We implemented both approaches and hope to see how do they behave differently on our dataset.

3.3.3 Pretrained Bert+ LSTM

To extract valuable information from Twitter text, we utilize Google's pre-trained Bert to obtain text features and apply attention mechanism to process the extracted feature information and capture important details. Additionally, we use normalization strategy to deal with stock price and ensure experimental results' stability. We then integrate the text features and stock price data using the concatenation method and feed them into an LSTM with temporal attention mechanism to achieve accurate predictions.

The feature extraction part of Twitter involves a transformer encoder with two submodels: a multihead attention mechanism and a feedforward neural network, followed by addition and normalization. We use the pre-trained Bert in hugging face and obtain the 768-dimensional vector of the last layer's [CLS] token hidden state, denote as c.

To integrate different types of data, adjusted close price p_c and c, we used the concatenation method and input the fused data into the LSTM model considering the nature of financial time series data. we denote the output of hidden state of LSTM as h_t.

The proposed framework allows for predicting both the trend of the target trading day and the fluctuations of other trading days during the lag period, which provides useful auxiliary information for predicting the target trading day. To achieve accurate predictions for both target and other trading days, we utilized a temporal attention mechanism to obtain important information.

Temporal attention has been extensively used in financial market analysis tasks, and some researchers have enhanced this mechanism to adapt it to the overall framework.it used information score and dependency score to calculate the contribution weight of the data and form the temporal attention model. The temporal attention model was able to generate predictions for target and other trading days. The over all structure is show in figure 9

4. Experimental results

4.1. Dataset

Data on daily stock prices from January 1st, 2014, to December 31st, 2015, have been obtained from the Yahoo finance website. This dataset comprises the open, high, low, close, and adjusted close prices of 88 different stocks,

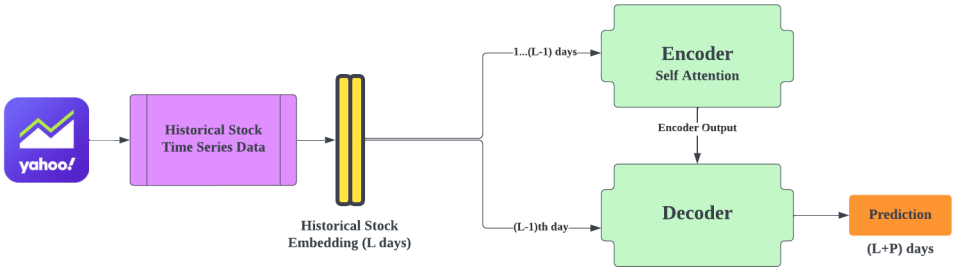

Figure 7. Transformer Architecture.

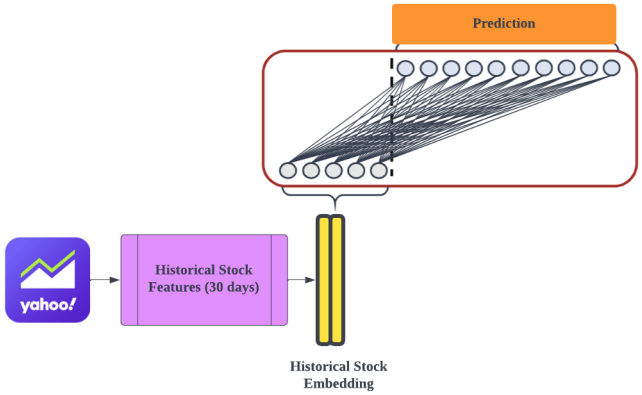

Figure 8. Linear Model Architecture.

as well as the trading volume on each day for a period of two years. The tweet text is adapted from the StockNet [6] dataset. These tweet texts correspond to the specific stock ticker and are assigned to specific dates. We pre-processed the data using techniques described in section 3.3 to ensure data balance and eliminate excessively small changes in stock price (less than 0.5% change). The dataset is split into two parts: a training set (80%) and a validation set (20%).

4.2. LSTM

We use different settings to train the LSTM baseline model. We first use the base features only, and then we add some economic indices which are augmented from the original dataset. The results are not far away from 0.5, so it suggests that the model prefers to make prediction randomly.

	30 days
5 features	0.475
15 features	0.510

Table 3. LSTM rise/fall Acc

4.3. Transformer

Our result shows that the transformer performs slightly better than LSTM and can have more potential. The setting for training transformer model is the same as training LSTM model with additional setting. The result is shown below:

	30 days
5 features	0.508
15 features	0.513

Table 4. Transformer rise/fall Acc

The transformer model performs slightly better than LSTM. However, no matter what settings we choose, the performances of the models are similar. In addition, both LSTM and Transformer models have the same problem by looking at their plots (The prediction plot for transformer 10). They all tend to prefer using the previous day's stock price as the prediction for the next day. That's why the plot suggests that they look similar, but the prediction is shifted right. Therefore, we didn't include more results from our experiments even though we have tried a lot of different settings in these models.

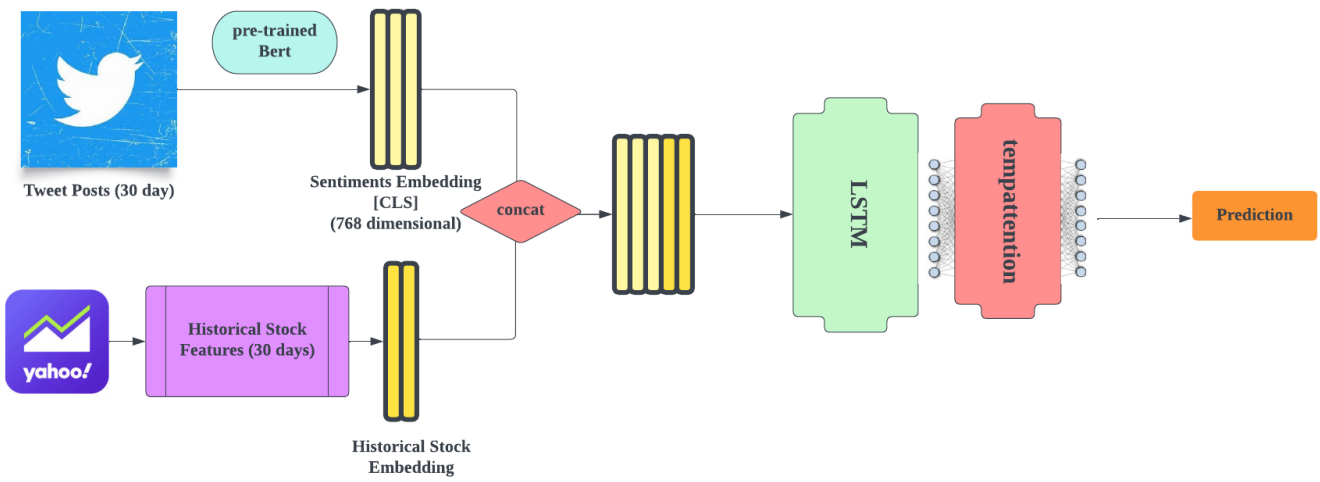

Figure 9. Pretrained Bert+ LSTM

4.4. LTSF-Linear & Nlinear

We did ablation studies for both LTSF-Linear and NLinear. We set 2 variables for the study. They are amounts of extracted features and lookback windows size. For the feature extraction, we want to observe how the change in the number of features will have effects on the linear model's performance. So we split it into 3 different settings:

- 2 features: adjusted close and volume
- 5 features (original stock features): open, high, low, volume, and adjusted close prices
- 15 features: 5 original stock features + 10 calculated economic indexes

At the same time, we also want to investigate if our linear model is more suitable for tracing long-term dependencies or short-term dependencies. So, for the lookback window sizes, we also have 3 different settings: 5 days, 15 days, and 30 days. The results of both Linear and Nlinear are shown in the following tables 5 6.

In regards to LTSF-Linear, with only adjusted close and volume as features (2-feature set), the model achieved the highest accuracy with a lookback window of 5 days (0.549). In contrast, using the 5 original stock features, the model's performance improved as the lookback window size increased, reaching the highest accuracy at 15 days (0.536) and marginally decreasing at 30 days (0.526). The performance with 15 features was relatively stable across all lookback window sizes.

For Nlinear model, it exhibited a similar pattern to the LTSF-Linear model, although the accuracies were generally lower, and the highest accuracy was achieved for 15 features and 15 days lookback window.

Our results indicate that, for both LTSF-Linear and LTSF-NLinear models, using a larger number of features generally improved the performance. The addition of economic indices provided some benefit, especially for the LTSF-NLinear model, suggesting that these extra features contain valuable information for predicting stock price movements. The models' performances were generally better with a 15-day lookback window. This suggests that both models are more suitable for capturing mid-term dependencies. The decrease in performance at longer lookback windows might be attributed to the increased difficulty of predicting stock price movements over a longer period or the dilution of relevant information in the larger window.

	5 days	**15 days**	**30 days**
2 features	**0.549**	0.516	0.497
5 features	0.508	0.536	0.526
15 features	0.531	0.518	0.517

Table 5. LTSF-Linear rise/fall Acc

	5 days	**15 days**	**30 days**
2 features	0.530	0.507	0.502
5 features	0.510	0.526	0.498
15 features	0.512	**0.543**	0.513

Table 6. LTSF-NLinear rise/fall Acc

4.5. Pretrained Bert+ LSTM

The table displays the performance of various model variants, revealing that the Pre-trained + LSTM (W/O Tweet) model has the poorest results, suggesting that social media text data contains valuable information. Additionally, even when compared to the baseline model, Pre-trained + LSTM (W/O Tweet) is still unsatisfactory, demonstrating

	3 days	5 days	15 days
Pretrain+LSTM	0.530	**0.552**	0.537
Pretrain+LSTM w/o tempAttention	0.510	0.527	0.516
Pretrain+LSTM w/o price	0.512	0.532	0.533
Pretrain+LSTM w/o tweet	0.487	0.491	0.482

Table 7. Pretrained Bert+ LSTM rise/fall Acc

the importance of extracting text features as input. Conversely, Pre-trained + LSTM (W/O price) produces highly competitive results, similar to those of Pre-trained + LSTM, indicating that historical prices have a significant impact on stock movement prediction. These results confirm the positive effects of tweets and historical prices, respectively, for predicting stock movement.

In Pre-trained + LSTM, the basic forecast unit is the trading day, making the choice of time window crucial for achieving accurate experimental results. However, when forecasting over a period of three calendar days, the presence of multiple trading days in the lag period, such as Monday's sports forecast, imposes limitations. As shown in the table, experiments with a time window of 10 calendar days did not produce better results than those with a lag of 5. This can be attributed to the fact that many random factors influence the stock market and changes in the target trading day are more likely to be affected by the price of the adjacent trading day. Furthermore, investor comments often reflect current market conditions and are unsuitable for predicting long-term trading behavior.

5. Conclusion & Future Works

Stock market prediction is a challenging problem. Investment in the stock market always comes with risks. However, if we can create a model that can predict most of the trends in the stock market, then it will be good news for all investors. So far, we have tried 3 different approaches.

The LTSF-Linear model has been proven to perform surprisingly well though it is very simple. It can capture the relationship better when dealing with short to mid-term time series dependencies. This might be the result of the model architecture being too simple and thus cannot keep track of more complex relationships. It is also shown that it generally performs better as the number of features increases. Future works can be to increase the complexity of the model so it can learn more sophisticated patterns from the time series data. Also, future research should focus on optimizing performance at different lookback window sizes or exploring alternative feature extraction techniques.

Since the transformer models and the LSTM models both have the preference to use the previous day's stock price as the prediction for the next day, we can try to not include the target price in training and let the model predict the target price by the given features. We can also try to make predictions for more days and use a different loss function to measure the area between the ground truth and the prediction instead of using MSE error. Since the performance of the transformer-based model increases as the dataset size increases, maybe we can try using a huge dataset that might capture more trends so that the model can perform better.

Features that are highly correlated with the adjusted stock price can be chosen to run the models. In addition, we can perform stock price predictions within one cluster generated by hierarchical clustering since the daily return of the stocks in a single cluster have a relatively high correlation with each other. Furthermore, we can use the principal components created in the dimensional reduction technique as our new input data set.

6. Author Contributions

Yifan and Leheng reviewed existing stock price prediction literature and proposed our customized model structure. Yifan also implemented the LTSF-Linear and Nlinear models, ran the Linear model experiments, and plotted all the model structure diagrams. Leheng created the LSTM and Transformer models and did many different experiments in different settings. Baihan Liu do some exploratory data an and did the exploratory data analysis, plot some visualizations, and wrote the data preprocessing steps. Haichao Ji did literature reviews on forecasting stock prices by hybrid model with ARIMA and Neural Network. Huijie obtained the experiment results from the baseline model. Huijie also desigend and trained the Pre-trained LSTM. All co-authors were involved in writing this report.

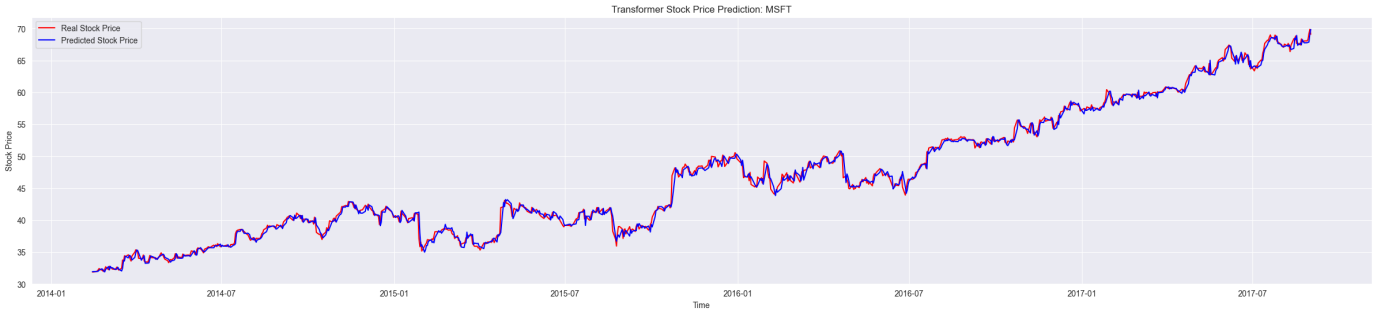

Figure 10. Transformer Stock Price Prediction.

References

[1] Yingjun Chen and Yongtao Hao. A feature weighted support vector machine and k-nearest neighbor algorithm for stock market indices prediction. *Expert Systems with Applications*, 80:340–355, 2017. 1

[2] Abhimanyu Das, Weihao Kong, Andrew Leach, Rajat Sen, and Rose Yu. Long-term forecasting with tide: Time-series dense encoder. *arXiv preprint arXiv:2304.08424*, 2023. 2

[3] Emmanuel Dave, Albert Leonardo, Marethia Jeanice, and Novita Hanafiah. Forecasting indonesia exports using a hybrid model arima-lstm. *Procedia Computer Science*, 179:480–487, 2021. 2

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. 5

[5] Shilpa Gite, Hrituja Khatavkar, Ketan Kotecha, Shilpi Srivastava, Priyam Maheshwari, and Neerav Pandey. Explainable stock prices prediction from financial news articles using sentiment analysis. *PeerJ Computer Science*, 7:e340, 2021. 2

[6] Umang Gupta, Vandana Bhattacharjee, and Partha Sarathi Bishnu. Stocknet—gru based stock index prediction. *Expert Systems with Applications*, 207:117986, 2022. 6

[7] Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. Time2vec: Learning a vector representation of time, 2019. 5

[8] Yuling Lin, Haixiang Guo, and Jinglu Hu. An svm-based approach for stock market trend prediction. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2013. 1

[9] Andreea-Cristina Petrică, Stelian Stancu, and Alexandru Tindeche. Limitation of arima models in financial and monetary economics. *Theoretical & Applied Economics*, 23(4), 2016. 2

[10] Seyed Hossein Razavi Hajiagha Shahvaroughi Farahani, Milad. Forecasting stock price using integrated artificial neural network and metaheuristic algorithms compared to time series models. *Soft computing*, 25(13), 2021. 1, 2

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. 2

[12] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? 2023. 2, 5