Stock Movement Prediction with Deep Learning Models

Hans Easton
University of Washington
Seattle, WA 98103
hans00@cs.washington.edu

June 4, 2023

Abstract

In this project, I investigate and employ cutting-edge deep learning algorithms to predict stock market movements. The performance of these algorithms on established benchmarking datasets is evaluated. A stretch goal is evaluating the model's performance through their trading performance in the stock market.

1 Introduction

1.1 Background

With a total value of $80 trillion dollars, the stock market is one of the largest financial markets globally. Stock market analysis involves the examination and evaluation of the performance of a company's stock and the overall market as a whole, making it an interdisciplinary field that intersects finance and computer science. There are two primary types of stock analysis: fundamental analysis, which attempts to analyze and predict a company's stock value based on its intrinsic worth, and technical analysis, which primarily relies on analyzing charts and trends based on past market performance. Recent advancements in big data and the increasing parallel processing ability of graphics processing units (GPUs) have led to the extensive application of deep learning models in stock market analysis. Such models can incorporate both the fundamental and technical aspects of analysis by using a company's quarterly earning data and historical stock prices in the training process to predict future stock movements.

1.2 Motivation

For the stock movement prediction problem, the input is a series of historical stock data, and the goal is to predict whether the stock price will trend up or down in the following trading days. Asset management companies and investment banks are steadily increasing their research funding into artificial intelligence, particularly focusing on deep learning models, since higher model accuracies often mean increased profitability. Additionally, trained deep learning models can serve as the backend for automated trading bots, capable to make real-time trading decisions.

1.3 Goal

The goal of this research project is to benchmark the performance of recently developed deep learning models in stock movement prediction on established benchmark datasets. The stretch goal is to evaluate the profitability of trading based on the models' predictions on the market data for this year.

1.4 Related work

A recent paper by Jiang *et al.* [4] provides a comprehensive review of the advancements in deep learning techniques for stock market prediction over the last three to five years. Fei *et al.* [2] uses adversarial

training to enhance the generalization capabilities of neural network prediction models for stock market forecasting. Another paper by Xu *et al.* [6] proposes a deep learning model designed to address the inherent stochasticity in the stock market. Their model uses a generative deep learning framework known as StockNet, which incorporates both data from Twitter and historical stock prices as training inputs. Along similar lines, Chen *et al.* [5] proposes the use of sentiment transfer learning for stock market prediction. This method involves transferring sentiment information learned from news-rich stocks to news-poor ones, and the prediction performances of the latter are thus enhanced.

2 Data

2.1 Dataset

The acquisition of high-quality data is important for training a deep learning model. In the context of stock market prediction, the data primarily takes on two forms: market data and text data. Market data encompasses all trading activities in the stock market, including price fluctuations, trading volume, and the like. This information serves as input features for prediction models and can also be used to generate prediction targets, such as the closing price for the subsequent day. Meanwhile, text data is generated outside the stock market and includes social media posts, news articles, web searches, among others. Sentiment analysis techniques can be used to extract sentiment factors from this data, providing additional inputs for prediction. This research paper mainly focuses on deep learning models using intrinsic market data.

To evaluate the effectiveness of the algorithms, I first utilize two publicly available benchmark datasets specifically designed for stock movement prediction. The two datasets are ACL18 by Xu *et al.* [6] and KDD17 by Zhang *et al.* [7]. ACL18 encompasses historical data from January 1, 2014, to January 1, 2016, of 88 high-trade-volume stocks from the NASDAQ and NYSE markets. In contrast, KDD17 provides a more extensive history, ranging from January 1, 2000, to January 1, 2016, and covers 50 stocks in the U.S. markets.

2.2 Preprocessing

The two datasets include intrinsic stock features such as opening price, closing price, volume, and previous close - each providing key insights into stock market behavior. In data preprocessing, I generate 11 new features by normalizing the original features and capturing the ratio of change over various trading day windows. I also introduce a hyperparameter T, defined as the lag, indicating the number of consecutive trading days used for next day prediction. For instance, a lag of 3 uses data from three consecutive days to predict the fourth day's stock movement.

I filter out data with movement percentages between -0.5% and 0.5% as these do not indicate clear trends. I then label positive movement as +1 and negative movement as -1 for prediction output. In summary, the model uses a consecutive series of stock movement percentages as input, with lag T, for a binary classification task, producing a prediction of whether the stock price will rise (+1) or fall (-1) the following day.

3 Methods

3.1 Baselines

I use four methods as baselines for predicting stock price movements. The first method employs a technical indicator known as the Momentum Indicator, which measures the rate at which stock prices increase or decrease over a defined period. This method uses this rate to forecast the movement for the subsequent day, with a window set at 10 days. The second approach involves a technical indicator called Mean Reversion, grounded in the principle that stock prices tend to return to their average levels over time. This method is based on the assumption that extreme price fluctuations are challenging to maintain for extended periods.

The third and fourth methods are deep-learning methods. The third method employs a two-layer

neural network with ReLU activations and fully connected layers. The fourth method incorporates a Long Short-Term Memory (LSTM) network. This type of neural network is capable of learning long-term dependencies, which is particularly useful for time series data like stock prices.

3.2 Challenges

Predicting stock movements presents a significant challenge due to the stochastic nature of stock prices. The prices are affected by a myriad of dynamic factors, including economic conditions, company performance, investor sentiment, geopolitical events, and market news. These factors continuously evolve, causing fluctuating and unpredictable stock prices.

Although standard supervised learning methods can be employed to construct predictive models, these models often struggle with generalization capabilities, leading to overfitting. To address the stochastic nature of daily stock price inputs, I have implemented adversarial training as outlined in the aforementioned [2] paper.

Adversarial training aims to make the model more robust to input noise. For instance, if an input with a 3-day lag, represented as [+0.5%, +0.7%, +0.4%], correctly predicts a stock price rise the following day, the model should maintain the same prediction for a slightly modified input, such as [+0.52%, +0.68%, +0.41%]. This aligns with intuitive human reasoning where a trader would make consistent predictions given that the input percentages only vary by a small amount.

3.3 Model

The stock prediction model begins by encoding input sequence features via a fully-connected encoder, allowing a richer representation of the input. For a stock s at trading day t, I denote its input feature as x_t^s. The encoded feature is represented as $m_t^s = \tanh(W_m x_t^s + b_m)$, where W_m and bm are learnable parameters. This encoded feature is then passed through an LSTM layer, a variant of recurrent neural networks adept at handling sequences due to its capacity to capture and maintain long-term dependencies and temporal patterns in the data. The layer can be summarized with the equation $h_t^s = LSTM(x_t^s, h_{t-1}^s)$, where h_{t-1}^s is the learnable hidden representation of the previous timestamp. This feature proves particularly beneficial for stock data, which is inherently sequential.

The paper also introduces a temporal attention mechanism alongside the LSTM layer, which compresses the hidden representations at various time-steps into a singular representation through adaptive weights. The attention value for each timestamp t is denoted as $a_t^s = softmax(u_s^T tanh(W_a h_t^s + b_a))$, and the overall weighted attention is value is $a^s = \sum_{t=1}^{T} a_t^s h_t^s$, where variables u_s, W_a, and b_a are learnable parameters, and T is the length of the lag. The attention mechanism takes into account the fact that data at differing time-steps may contribute variably to the representation of the entire sequence. This mirrors real-world situations where specific days in the stock market hold greater significance than others. One such instance is the presence of support and resistance points, which are price points on a chart that are expected to draw the most buying or selling action.

The output of the temporal attention is concatenated with the final output of the LSTM layer and fed through a linear layer to produce a binary classification output. The concatenation further emphasizes the importance of the most recent timestamp, according to the work done by Fama *et al.*[1].

The model concludes with an adversarial training layer, which aims to increase model robustness. Adversarial training [3] trains the model using both training set examples and adversarial examples. However, this technique is not directly applicable to stock prediction due to its computational complexity and potential for uncontrollable gradient behavior in sequential data. Therefore, the model introduces perturbations only at the final layer, addressing these issues while maintaining computational feasibility.

To determine the magnitude of perturbations, the paper attempts to maximize the model's loss given the perturbation. The equation is given as $r_{adv}^s = argmax_{||r_{adv}^s \leq \epsilon||} l(y^s, y_{adv}^s)$, where r_{adv}^s is the perturbation under the constraint that its magnitude can-

not exceed a hyperparameter threshold ϵ. Because it's challenging to calculate the precise perturbation that maximizes the loss, a fast gradient approximation method is adopted. The method by Goodfellow *et al* [3] estimates the gradient of the loss function concerning the latent representation under an L2-norm constraint. Consequently, the loss function aims to maximize the regularized loss on both the original and adversarial inputs.

4 Evaluation

4.1 Metrics

I utilize two evaluation metrics to assess the performance of both the baseline models and the adversarial model. The first metric is model accuracy, simply defined as the percentage of instances where the model correctly predicts movement across all test samples. The second metric is the Matthews Correlation Coefficient (MCC), a measure particularly effective at avoiding bias due to data skew. The MCC is calculated using the formula:

$$MCC = \frac{tp \cdot tn - fp \cdot fn}{\sqrt{(tp + fp)(tp + fn)(tn + fp)(tn + fn)}}$$

where tp, fp, tn, fn is the number of true positive, false positive, true negative, and false negative samples respectively. Higher MCC value indicates bette performance, and vice versa.

4.2 Procedure

The KDD17 and ACL18 datasets are divided into training, validation, and testing subsets. The baseline and adversarial models are trained on the training set and tuned using the validation set via grid search. For the deep learning models, hinge loss function and Adam optimizer are employed. The testing set is reserved for unbiased final model evaluation using the two metrics proposed above. The code for this project can be accessed here: https://github.com/hanseaston/StockPrediction. I have put a significant amount of work into the code base. The original code, which the paper links to, is written in Tensorflow V1, and it is not particularly easy to understand. Therefore, I've rewritten the model from scratch using Tensorflow V2, and during the process, I've restructured the entire codebase as well.

	ACL18		KDD17	
Method	**Acc**	**MCC**	**Acc**	**MCC**
MOM	44.8	-0.04	40.2	-0.03
MR	46.1	-0.06	41.8	-0.04
NeuralNets	52	0.01	49	0.02
LSTM	55.8	0.09	55.2	0.11
Adv-Attn-LSTM	56.1	0.15	56.1	0.14

Table 1: Model Performances on ACL18 and KDD18 dataset

5 Results

Here's the preliminary findings of the results as shown in Table 1. Performance of technical indicators include MOM and MR, and the performance of deep learning models include NeuralNets, LSTM, and Adv-Attn-LSTM on the two benchmarking datasets.

5.1 Discussion

From Table 1, we can see that the adversarial model achieves the highest accuracy compared to other models. However, the degree of improvement is not substantial. During the evaluation process, I have noticed unstable performance. In some instances, the vanilla LSTM model almost reaches the same performance as the adversarial model. The results are also occasionally inconsistent with those presented in the paper. For instance, the paper reports that the LSTM model achieves an average accuracy of only 53.18% on the ACL18 dataset. Yet, in my own experiments, the LSTM model sometimes reachs as high as 56% accuracy, a clearly higher figure. Nevertheless, the Adv-Attn-LSTM model still remains the highest average performance compared to other models.

6 Deployability

6.1 Dataset

In order to determine whether the proposed model transfers well to more recent data, I have subscribed to and fetched API data from Polygon.io, a third-party financial market data provider. The API data I have obtained includes the 500 stocks in the S&P 500 categories and spans from January 2014 to May 2023. I follow similar data processing steps as the aforementioned procedures, but with a couple of additional features.

6.2 Preprocessing

In addition to the existing features, I have implemented code to parse the 10-day moving average, 20-day moving average, and 30-day moving average for both the stock prices and trading volumes. The motivation behind this addition is to provide the model with additional inputs for identifying the appropriate support and resistance levels in the stock market.

I also have a slight disagreement with the original paper's data filtering steps. To recap, the paper chooses to remove all training data with a price percentage change between -0.5% and 0.5%. While this decision may enhance the quality of the training data, it introduces a data shift between the training and test data. In real-world deployment scenarios, it is not uncommon for the model to encounter stock price changes within this range. If the model hasn't been exposed to similar inputs during training, it is likely to make incorrect predictions. Therefore, in my data preprocessing step, I opt not to include such filtering step.

6.3 Metrics

Finally, I propose directly using precision metrics to measure the performance of the model. If a trader is employing a simple trading strategy where they buy a stock if the model predicts a price increase and sell it the following day, the trader will place greater importance on the accuracy of positive predictions. In other words, the trader values the model's ability to make correct positive predictions more than it accidentally making incorrect negative predictions. Therefore, optimizing the model based on precision metrics is the most suitable approach for profitability, assuming the simplest trading strategy at present.

6.4 Initial results

Model	Precision(%)
MOM	47.1
MR	50.2
NeuralNets	50.8
LSTM	52.5
Adv-Attn-LSTM	53.8

Table 2: Model performances on Polygon dataset

As we can see, the findings are generally consistent with the results obtained from the two benchmarked datasets. The Adv-Attn-LSTM model demonstrates the highest precision accuracy compared to all the other models.

6.5 Threshold Bias

In this section, I propose a simple trick to increase overall precision for the model. This involves adjusting the decision threshold for binary classification in the model's last layer, which uses a sigmoid function. For instance, a 0.7 threshold means the model predicts a stock price rise only if the output is above 0.7. As the threshold increases, the model's accuracy generally improves, but the quantity of predictions also declines. This is because the model only makes predictions when it's highly confident. This higher threshold is useful for trading as the focus is more on prediction quality than quantity. The table indicates a consistent increase in accuracy as the threshold increases. The threshold greater than 0.7 is not shown in the table, as the model rarely is confident enough to produce very high output, thus there is a lack of predictions with threshold at 0.8.

Adv-Attn-LSTM @ 0.5	53.8%
Adv-Attn-LSTM @ 0.6	57.5%
Adv-Attn-LSTM @ 0.7	61.4%

Table 3: Model performances with 0.5, 0.6, 0.7 threshold.

6.6 Model Concensus

Finally, I propose a consensus model approach to enhance the robustness of predictions and increase the overall precision accuracy further. In this approach, I train three different models with different thresholds for model labels: the neutral threshold model, the positive threshold model, and the negative threshold model. These models differ in how they assign labels to the dataset.

The positive threshold model sets the label threshold at 1%, meaning that any stock price percentage greater than 1% is labeled as 1, and 0 otherwise. Similarly, the negative threshold model sets the label threshold at -1%, labeling any stock price percentage less than -1% as 1, and 0 otherwise. The neutral model uses the default threshold at 0%

Since the positive and negative models will have a disproportionate ratio of positive and negative labels, I employ the SMOTE technique to oversample the minority training data labels. After balancing the dataset, I use the same training procedure as in the previous section to train the three models.

As a consensus approach, the prediction that the stock price will go up is made only when both the positive and neutral models output a 1, and the negative model outputs a 0. This approach is motivated by the expectation that the three models will capture different upward and downward trend patterns in the input data. If all three models produce consistent predictions, a trading decision to buy the stock is made. It is expected that the number of positive predictions will decrease in quantity as occurrences become rarer, but the prediction precision will increase as a result.

As shown in the table, the model using the consensus approach outperforms each of the three models individually, with an average precision of 65.2%. It is worth noting that the models with the positive and negative thresholds, with precision accuracies of 52.9% and 54.5% respectively, perform worse than the neutral model at 61.4%. This is likely due to the imbalanced nature of the data distribution, even after employing the SMOTE technique for correction.

Model with positive threshold	52.9%
Model with neutral threshold	61.4%
Model with negative threshold	54.5%
Model with three consensus	65.2%

Table 4: Model performances with and without consensus approach.

7 Conclusion

The project has demonstrated preliminary promising results by utilizing deep learning models, adversarial training, attention mechanisms, and various training techniques for stock movement prediction. The best model achieves a precision accuracy of 65.2% on real market data. However, given the volatile nature of the stock market, further benchmarking and evaluation are necessary to measure the actual profitability of the model.

Future work includes model specialization for different market conditions, incorporation of diverse data sources like financial news for sentiment analysis, transfer learning for individual stock fine-tuning, and the development of trading bots to utilize model predictions. These efforts aim to further improve the model's performance, enhance profitability, and automate trading decisions. There's a lot of things to work on!

References

[1] Eugene Fama and French Kenneth. Size, value, and momentum in international stock returns. *Association for Computing Machinery, https://doi.org/10.1016/j.jfineco.2012.05.011*, 2012. 3

[2] Fuli Feng, Huimin Chen, Xiangnan He, Ji Ding, Maosong Sun, and Tat-Seng Chua. Enhancing stock

movement prediction with adversarial training. 2019. 1, 3

[3] J. Shlens I. Goodfellow and C. Szegedy. Explaining and harnessing adversarial examples. *ICLR*, 2015. 3, 4

[4] Weiwei Jiang. Applications of deep learning in stock market prediction: Recent progress. *Expert Systems With Applications*, 185:115537, 2021. 1

[5] Xiaodong Li, Haoran Xie, Raymond Lau, Tak-Lam Wong, and Fu Lee Wang. Stock movement prediction from tweets and historical prices. *EEE Access*, 2018. 2

[6] Yumo Xu and Shay B. Cohen. Stock movement prediction from tweets and historical prices. *arXiv preprint arXiv:1809.06582*, 2018. 2

[7] Liheng Zhang, Aggarwal Charu, and Qi Guo-Jun. Stock price prediction via discovering multi-frequency trading patterns. *Association for Computing Machinery, New York, NY, USA, 2141–2149. https://doi.org/10.1145/3097983.3098117*, 2017. 2