STOCK PRICE PREDICTION USING GRAPH NEURAL NETWORKS

Amey Meher
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
avmeher@ncsu.edu

Deep Mehta
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
dmmehta2@ncsu.edu

Background

In today's dynamic financial markets, making informed and strategic decisions regarding long-term stock investments is a formidable challenge. The traditional methods for predicting stock prices over extended time horizons often fall short due to the myriad of factors influencing market dynamics. Investors face difficulties in understanding complex relationships between stocks, sectors, and macroeconomic indicators, making it challenging to identify attractive investment opportunities. Moreover, the inherent volatility of the stock market further complicates the task of devising effective long-term investment strategies. This project addresses this multifaceted problem by leveraging the power of graph machine learning to model and predict stock price movements while utilizing the intricate web of relationships between different stocks and financial entities.

Introduction

In recent years, graph machine learning techniques have gained prominence for their ability to capture complex relationships in financial data. This project aims to apply graph machine learning methods to predict stock prices and explore relationships between stocks, indices, and other relevant factors. More specifically, our problem statement is on the basis of previous 30 days of daily stock OHLC values, predict the closing price of the stock on the 31st day.

The primary objectives of this project are:

- Develop a graph-based dataset: Create a structured graph dataset that represents relationships between stocks, financial indices, economic indicators, and other relevant factors. This dataset will be used for long-term stock price predictions and relationship exploration.
- Apply graph embedding techniques: Utilize graph embedding methods to capture the underlying patterns and dependencies in the dataset. Techniques such as Graph Convolutional Networks (GCNs) will be explored.
- Predict stock prices: Build predictive models using embedded graph data to forecast stock closing prices for daily data.
- Explore stock relationships: Analyze the graph data to identify and quantify relationships between different stocks and sectors. Discover patterns and dependencies that can inform investment decisions.
- Evaluate model performance: Assess the accuracy, precision, recall, and F1-score of the prediction models. Evaluate their effectiveness in identifying attractive long-term investments.

Literature survey

The foundational understanding of graph neural networks for stock market predictions was derived from a seminal paper [1]. This source provided insights into the utilization of graph structures based on correlation values of the logarithmic return series of stocks. Techniques presented in this paper served as a basis for creating a graph in our work, emphasizing the significance of the graph's structure in capturing stock relations. Additionally, the adoption of the Minimum Spanning Tree (MST) for graph generation was inspired by the methodologies discussed in this reference.

The integration of temporal aspects into graph convolutional networks was informed by the paper titled "T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction" [2]. This source provided a foundational understanding of how temporal information could be incorporated into graph-based models. Leveraging the concepts from this paper,

our work devised an approach that captures the temporal dynamics of stock data through the integration of temporal graph convolutional layers.

The integration of deep learning techniques with knowledge graphs for stock price trend prediction was explored through the paper titled "An integrated framework of deep learning and knowledge graph for prediction of stock price trend: An application in Chinese stock exchange market" [3]. Drawing inspiration from this work, our research integrated graph convolutional layers with Long Short-Term Memory (LSTM) networks, creating a comprehensive model that incorporates historical closing prices and network structure to predict future closing prices for individual stocks.

Our approach involves stacking multiple graph convolution layers over LSTM sequence-to-sequence models to capture both spatial dependencies of stock relations and temporal dynamics. This novel methodology draws on the insights gathered from the surveyed literature, creating a robust framework for stock market analysis and prediction.

Methodology

1. **Data collection**

   Utilizing the Yahoo Finance API, we meticulously gathered stock prices for the S&P 500 stocks, considering two distinct timeframes: Daily and Monthly. The purpose of this dual-data approach was to facilitate both the construction of a comprehensive graph representation and the training of a predictive model.

   1.1 **Data for graph generation:**

   In this phase, we gathered monthly closing prices for all stocks, spanning a 5-year historical period. This choice of this extended timeframe was motivated by the belief that trends manifest more significantly when observed over a longer duration. We aimed to capture the essence of long-term market behavior, allowing for a more comprehensive analysis of evolving patterns, cyclical trends, and overarching market sentiments. The belief that trends manifest more significantly over a longer duration is rooted in the acknowledgment that short-term fluctuations may be influenced by transient factors.

   Furthermore, opting for a broader timeframe not only facilitated the identification of enduring trends but also had a practical implication on the representation of the graph. Given the less frequent occurrence of significant changes over this extended 5-year period, the graph could be effectively represented in a static manner. This choice simplifies the visualization and analysis of the graph, as it becomes a snapshot of the more stable and enduring relationships between stocks over time.

   1.2 **Data for training the model:**

   We collected Daily OHLCV values of all the stocks for the previous 1 year data. We wanted to train our model on the daily data so as to predict the daily closing prices of the 31st day when the history data of 30 days was available to us.

   Features used:

   - Ticker Symbol - A unique identifier for each traded security, the ticker symbol streamlines the tracking and analysis of specific stocks on the stock exchange.
   - Open price - Representing the initial traded value of a stock on a given day, the open price offers insights into early market sentiment and trading activities.
   - Close price - The final traded value of a stock at the end of a trading day, the close price is a pivotal indicator of overall market sentiment, extensively used in technical analysis.
   - High price - Denoting the peak traded value during a specific trading session, the high price highlights intraday price fluctuations and identifies peak price levels.
   - Low price - Signifying the lowest traded value within a specific trading period, the low price is crucial for assessing the minimum point reached during the trading day.
   - Volume traded on the day - Reflecting the total number of shares traded for a stock during a specified timeframe, volume is a key metric indicating market interest and often accompanying significant price movements.

2. **Data pre-processing**

   2.1 **Monthly data pre-processing for graph generation:**

2.1.1 Calculating the logarithmic return series

We used logarithmic returns instead of normal returns for calculating correlations between stocks due to their more desirable statistical properties. Logarithmic returns are additive, maintaining consistency in time aggregation and avoiding biases in averaging. They possess a mathematical convenience, simplifying calculations and ensuring compatibility with continuous compounding assumptions. Additionally, logarithmic returns stabilize volatility and offer an interpretation close to percentage changes when returns are small, making them more intuitive for investors and analysts. Overall, the use of logarithmic returns enhances the robustness and reliability of correlation analysis in financial contexts.

$$r_{i,\Delta} = log(C_i(t)) - log(C_i(t - \Delta t)$$

In our case, we have taken the value of Δt to be 1, where we are considering the return between T and T-1 days.

2.1.2 Calculating the Pearson's correlation coefficient

Pearson's correlation coefficient is often preferred for calculating the correlation between stocks in financial analysis. It measures the linear relationship between two variables, providing insights into the strength and direction of their association. It is also standardized, producing values between -1 and 1, where -1 indicates a perfect negative linear relationship, 1 indicates a perfect positive linear relationship, and 0 indicates no linear relationship. This standardized scale facilitates easy interpretation and comparison of correlation values.

$$\rho_{ij}(t, \Delta t, \tau, T) = \frac{\sum_{s=t}^{t-(T-1)} [r_{i,\Delta t}(s) - \bar{r}_{i,\Delta t}(t)][r_{j,\Delta t}(s-\tau) - \bar{r}_{j,\Delta t}(t-\tau)]}{\sqrt{\text{Var}[\boldsymbol{r}_{i,\Delta t}(t)]\text{Var}[\boldsymbol{r}_{j,\Delta t}(t-\tau)]} \times T}.$$

The graphs based on the correlation values will be an undirected graph. Below is a visualization of the heatmap of the correlation matrix that was calculated:

2.1.3 Calculating the distance between the stocks

These correlation values cannot be used as a metric as they do not obey the axioms of triangle inequality. Therefore, to use these values as a metric, we use the below formula to convert the correlation values to a distance metric that could be used to represent the dissimilarity between the stocks. Here, the minimum value is 0 and the maximum value is 2.

$$d_{i,j} = \sqrt{2(1 - \rho_{i,j})}$$

We then use this distance matrix as the adjacency matrix for the graph that we are generating.

2.1.4 Extracting meaningful edges

For extracting meaningful edges for the graph, we get the MST for this distance matrix. In paper [1], we get to know that the MST would represent the stocks in a hierarchical way which would give us strong interdependencies among the stocks. The stocks are arranged in sectors after getting the MST from this adjacency matrix. We will then use this graph with the edge weights from the distance matrix that we obtained earlier.

2.2 Daily data pre-preocessing for model training:

2.2.1 Data filtering:

To facilitate training, we collected daily data spanning the past year for all stocks in our analysis. However, some recently launched companies lacked sufficient historical data. Consequently, we opted to exclude these companies from our training dataset by removing the corresponding rows, ensuring that our model is trained on a robust dataset with a consistent timeframe for all included stocks.

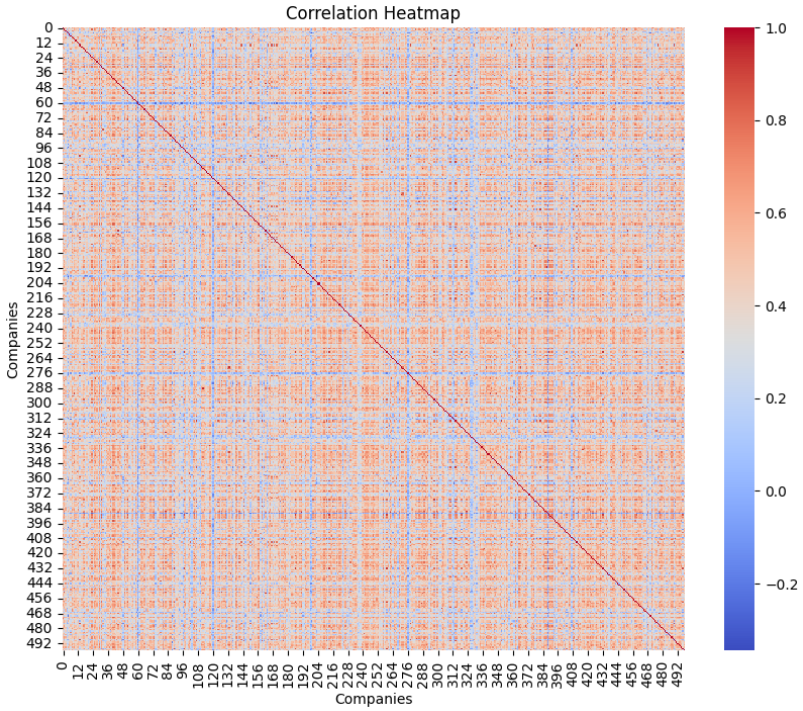

Figure 1: Correlation matrix heatmap

2.2.2 Data Normalization:

To enhance the robustness and effectiveness of our model, we normalized all Open, High, Low, Close, and Volume (OHLCV) values for each company, scaling them to a range between 0 and 1. This normalization process serves several crucial purposes. Firstly, it mitigates the impact of varying scales among different features, preventing certain parameters, such as higher stock prices, from disproportionately influencing the model's predictions. Additionally, normalization facilitates convergence during the training process, as it helps prevent numerical instabilities and ensures that each feature contributes proportionally to the overall learning. Moreover, normalizing the data can improve the model's generalization performance, making it more adaptable to unseen data by avoiding biases introduced by differing scales.

3. Model architecture

We explore a neural network architecture that learns from both the spatial stock relational data and time-series of historical stock price changes to forecast closing price of stocks at a future time.

The spatial dependency of the stock relations is learnt through multiple graph convolution layers stacked over multiple LSTM, sequence to sequence model, layers that leverage the historical closing prices on top of the network structure to predicts closing prices in the future for each stock.

Listing 1: Forward pass code for the model

```
def forward(self, x, edge_index, edge_weight):
    h = None
    c = None
    for i in range(self.window_size):
        input_x = x[:,i].view(num_nodes,5)
        h ,c = self.recurrent(input_x, edge_index, edge_weight, h, c)
    h = F.relu(h)
    h = self.linear(h)
    h = self.linear_e(h)
    return h
```

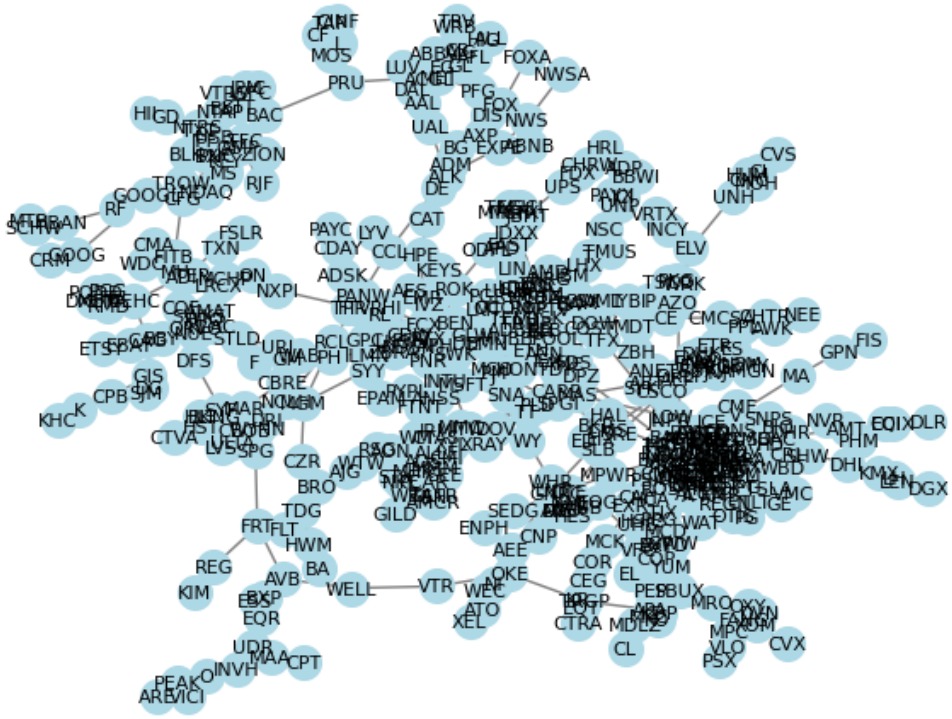

Figure 2: MST from the adjacency matrix

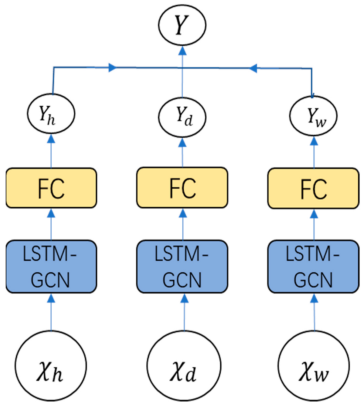

Figure 3: GCN + LSTM model

Experiments

Creating the Train/Test Dataset: We started by collecting daily data spanning the previous year. This data was then subjected to a preprocessing phase to ensure it was in a suitable format for our model. Once preprocessed, we divided the data into two distinct sets: a training set and a testing set. The training set comprised 80% of the data, while the remaining 20% was allocated to the testing set.

Training the Graph Neural Network (GNN) Model: The training process involved creating batches of data, each containing information from 30 consecutive timesteps. These batches were then fed into the GNN model, which was

trained to predict the outcome at the 40th timestep based on the input data. After each prediction, the sliding window of data was shifted forward by one timestep, and the process was repeated with the new batch of data.

Evaluating the Performance of the Model: To gauge the effectiveness of our GNN model, we compared the stock prices predicted by the model with the actual stock prices. This comparison was quantified using specific metrics, such as the Mean Square Error (MSE) and the Mean Absolute Error (MAE). These metrics provided a numerical measure of the model's accuracy, allowing us to assess its performance objectively.

Results

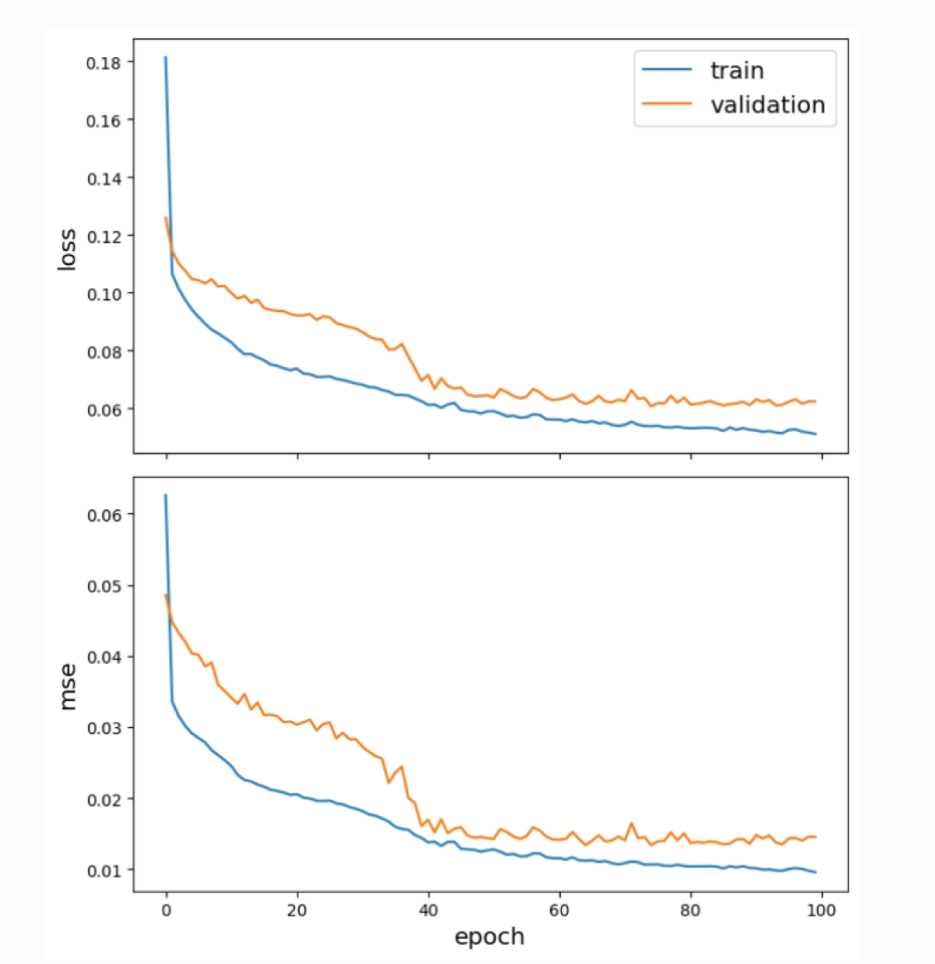

Figure 4: Train and Validation Error

The graph 4 presents the Mean Absolute Error (MAE) and Mean Square Error (MSE) for both training and validation data over a span of 100 epochs.

From the graph, we can observe the following:

- Training Data: The MAE and MSE for the training data show a decreasing trend as the number of epochs increases. This indicates that the model is learning and improving its predictions on the training data with each epoch.
- Validation Data: The MAE and MSE for the validation data, however, remain relatively constant throughout the epochs. This suggests that the model's performance on unseen data (validation data) is not improving significantly with more training.
- Overfitting: The decreasing error on the training data and the constant error on the validation data suggest that the model might be overfitting. Overfitting occurs when a model learns the training data too well, capturing

noise along with the underlying pattern. As a result, it performs well on the training data but fails to generalize to unseen data.

In conclusion, while the model seems to perform well on the training data, its performance on the validation data suggests there might be overfitting. To address this, we could consider strategies such as early stopping, regularization, or increasing the amount of training data.

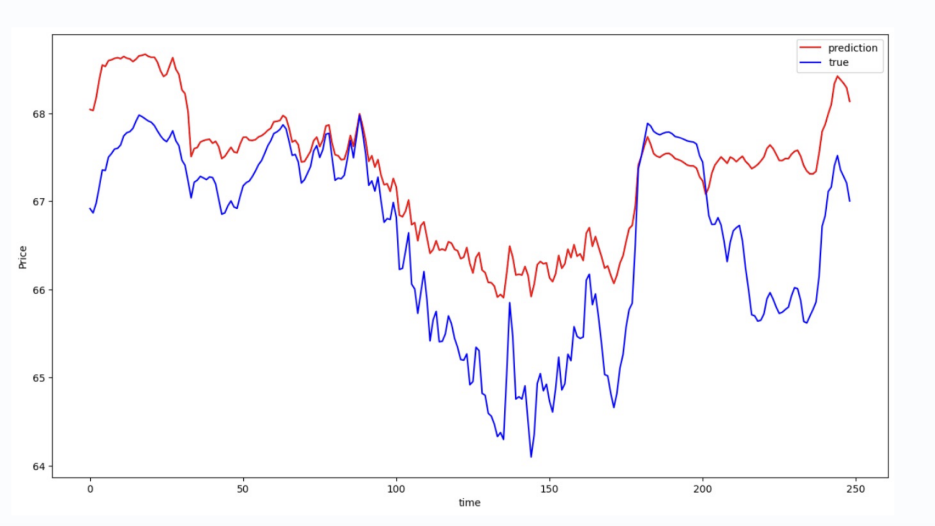

Figure 5: Prediction Results

Here are some observations from the graph:

- General Trend: The predicted and true stock prices seem to follow a similar trend. This indicates that the model is able to capture the overall movement of the stock prices.
- Prediction Accuracy: There are instances where the predicted prices are higher than the true prices, and vice versa. This suggests that while the model is able to capture the general trend, it may not be as accurate in predicting the exact stock prices.
- Overestimation and Underestimation: The model appears to overestimate the stock prices when the true prices are low, and underestimate when the true prices are high. This could be a sign of the model's bias or its inability to capture extreme values.

In conclusion, while the model seems to capture the overall trend of the stock prices, there is room for improvement in its prediction accuracy, particularly in capturing extreme values. Further tuning of the model or incorporation of additional relevant features could potentially enhance its performance.

Conclusion

Based on the analysis of the Graph Neural Network (GNN) model's performance and the stock price prediction graph, we can draw the following conclusions:

1. **Model Performance**: The GNN model shows a decreasing trend in error rates (both Mean Absolute Error and Mean Square Error) for the training data as the number of epochs increases, indicating that the model is learning and improving its predictions on the training data. However, the error rates for the validation data remain relatively constant, suggesting that the model may be overfitting the training data and not generalizing well to unseen data.
2. **Stock Price Predictions**: The predicted stock prices generally follow the same trend as the true stock prices, indicating that the model is able to capture the overall movement of the stock prices. However, there are instances where the model overestimates or underestimates the true prices, suggesting that it may not be as accurate in predicting the exact stock prices.

In conclusion, while the GNN model shows promise in capturing the overall trend of the stock prices, there is room for improvement in its prediction accuracy and its ability to generalize to unseen data. Further tuning of the

model, incorporation of additional relevant features, or strategies to prevent overfitting could potentially enhance its performance.

Future Work

Based on our experimental results, we encountered challenges with convergence in our predictions, deviating from the anticipated behavior. We therefore aim to explore alternative models, specifically ASTGCN (Attention-based Spatial-Temporal Graph Convolutional Network) and Dynamic TGCNs (Temporal Graph Convolutional Networks), for our prediction tasks.

References

[1] A survey of the application of graph-based approaches in stock market analysis and prediction - Suman Saha, Junbin Gao, Richard Gerlach

[2] T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction - Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, Haifeng Li

[3] An integrated framework of deep learning and knowledge graph for prediction of stock price trend - Jiawei Long, Zhaopeng Chen, Weibing He, Taiyu Wu, Jiangtao Ren

[4] HATS: A Hierarchical Graph Attention Network for Stock Movement Prediction - Raehyun Kim, Chan Ho So, Minbyul Jeong, Sanghoon Lee, Jinkyu Kim, Jaewoo Kang

[5] Exploring Graph Neural Networks for Stock Market Predictions with Rolling Window Analysis - Daiki Matsunaga, Toyotaro Suzumura, Toshihiro Takahashi

[6] Incorporating Corporation Relationship via Graph Convolutional Neural Networks for Stock Price Prediction - Yingmei Chen, Zhongyu Wei

[7] Knowledge-Driven Event Embedding for Stock Prediction - Xiao Ding, Yue Zhang, Ting Liu, Junwen Duan

[8] metapath2vec: Scalable Representation Learning for Heterogeneous Networks - Yuxiao Dong, Nitesh V. Chawla, Ananthram Swami

Code

The code for data collection and model training can be found here: https://github.ncsu.edu/avmeher/Stock-Market-Analysis-Using-GNN.git