STOCK PRICE PREDICTION USING GRAPH NEURAL NETWORKS

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

Amay Gada
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
ahgada@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.

Data choices

A task like stock price prediction requires temporal aspects which is readily available. However, to explore relationships between companies, there is a need for graph like data. Such data is not readily available. As we explored possible datasets, we were left with 2 choices. The following is a report of the same.

1. **Option 1:**

   We found a dataset off a Kaggle competition. The dataset listed around 300000 pairs of companies along with the partnerships that they had. The partnership included distinct categories like "competitor", "partners", "investor", etc.

   Our initial approach was to build a potential recommendation system. The approach was very simple. We chose a company as root (potentially the top company an investor chooses). Using the root, we did a biased walk on the graph to build a subset of the graph that we would train on.

   While we were very close to using this dataset, the dataset had certain aspects that were not convincing. The issues were as follows:

   - No ticker name - If we were to go with this dataset, we would have had to extract 100s of ticker names manually.
   - Links not completely accurate - There were companies that were non-intuitively connected. That made us come to a conclusion that parts of a large company can invest in other parts of another large company. An entire link between the 2 would be somewhat accurate.
   - Dataset was static - The dataset did not allow us to verify 2 very important things. How important the relationship was, and if the relationship was still valid today.
   - Currency conversion - Lastly, the stocks very multinational and currency conversion was an overhead as well.

   Hence this dataset was not the best choice to go ahead with.

2. **Option2:**

   The second option was creating our own graph using mathematical arsenal like correlation.

Making our own graph

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:

- 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.

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

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:

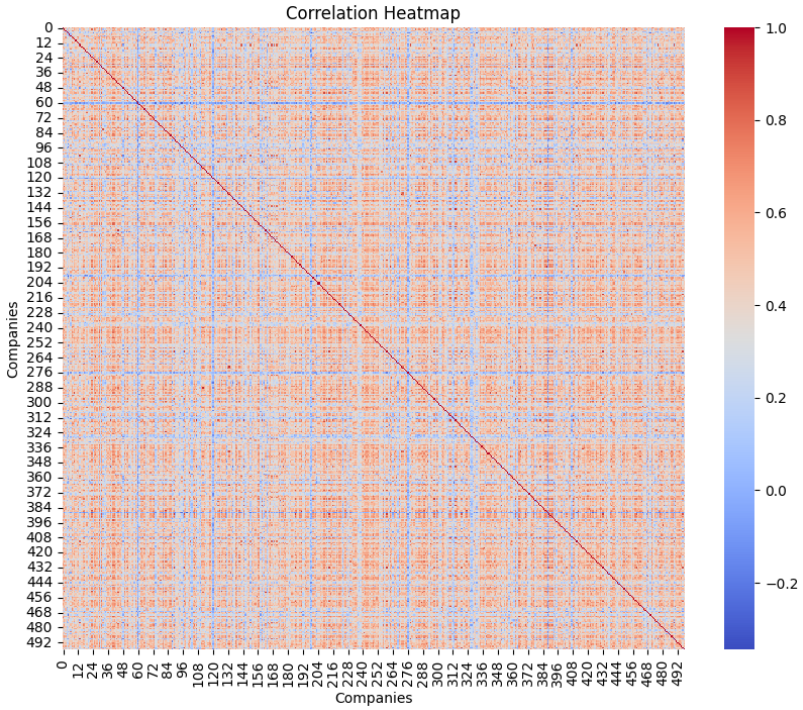

Figure 1: Correlation matrix heatmap

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.

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

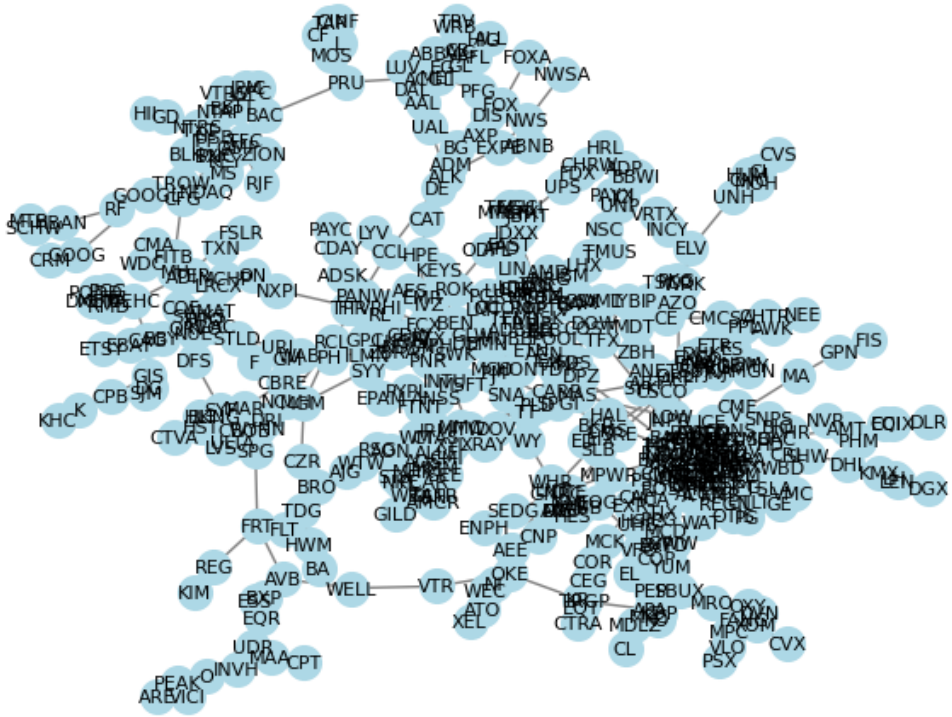

Figure 2: MST from the adjacency matrix

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. Modelling the problem

We explore a single layered spatial temporal GNN architecture (Figure 3) 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 graph convolution layers stacked over an LSTM layer that leverage the historical closing prices on top of the network structure to predicts closing prices in the future for each stock.

Modelling this task in this way was a challenging task, and we had several roadblocks in the way. This section describes the training challenges as a journey. As we move along, we will highlight the roadblock at each step and the solution we 'tried to' employ to overcome it. We will also include the results (good and bad) that led us to the next consecutive roadblock.

3.1 Experimental Setup

- 32 GB RAM
- 2 GB Quadro P1000 GPU
- Python3

- PyG, PyG Temporal, Pytorch

3.2 Data split

We ensured that all the three splits (train, test and split) were independent of each other and randomized.

- Train - 65%
- Val - 15%
- Train - 20%

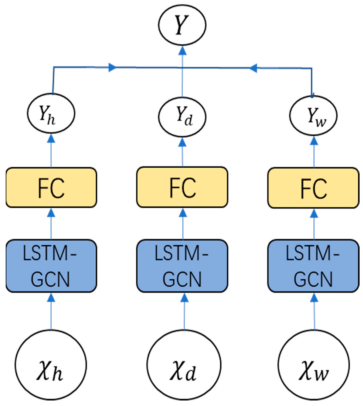

Figure 3: GCN + LSTM model

3.3 The training journey

The initial design of our model was very simple. We used a single layer LSTM with a single layer of Graph convolution. The forward pass of the model looked something like this:

Listing 1: Forward pass code for the initial 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
```

The first roadblock that we faced was that the GPU memory we had was very low for training a graph of 500 companies together. Moreover, cloud platforms weren't desirable since we had to build pyG temporal from scratch, everytime the session crashed. That took atleast 40 minutes each time.

As a result, we decided to not use more than 100 companies on our local machine.

Next we began training on an MST created by a 100 companies. We used a learning rate of 0.01 with the Adam optimizer, and scheduled a learning rate decay with a patience of 6 and factor of 0.05. By the end of the training, and severe tuning, we saw noticeable convergence. Figure 4 shows the training graph over 20 epochs.

To test this further, we compared the predicted stock price on the test set. The results we saw led to roadblock 2. Figure 5 shows the the predicted price (green) vs the actual price (blue) over the test set (50 days).

Referring figure 5, it is obvious that the model, even though converged, could not learn as much as we wanted it to. This indeed was our roadblock 2. But luckily we found out that we made a fundamental error.

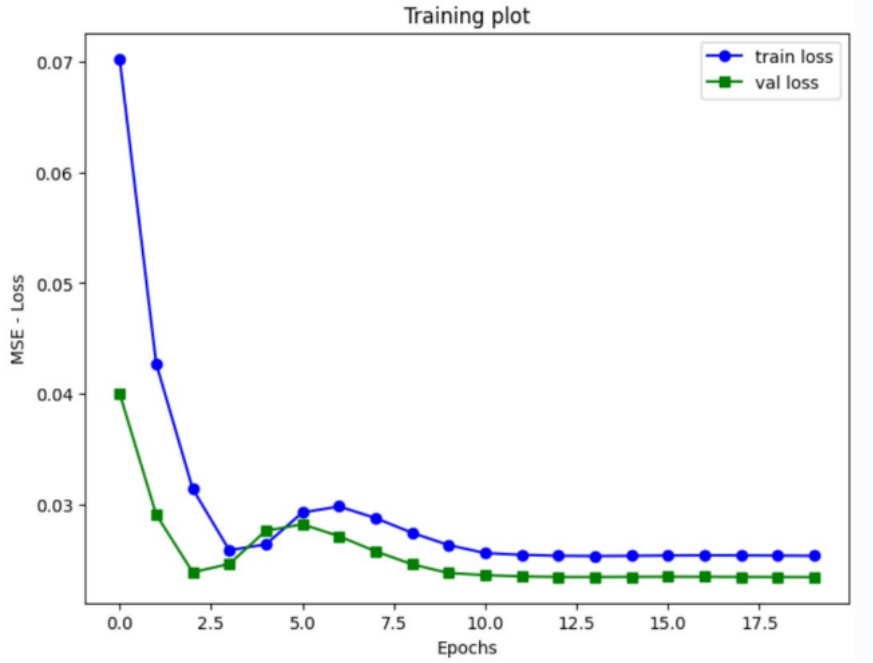

Figure 4: Initial training plot

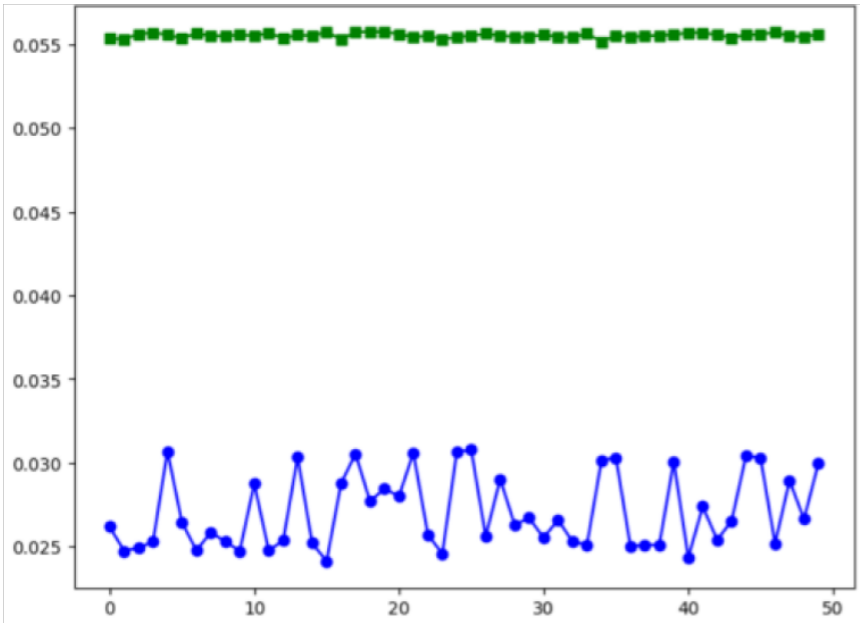

Figure 5: Flawed prediction plot

Loss functions allow optimization. However, if the loss gets too small, it can hinder convergence. What we did earlier was normalize the stock price globally. We performed min max scaling across all companies. Hence, a company with an extremely large price point, squished the prices of other companies as well. While this was not an issue logically, it led to the loss function outputting very small values.

We fixed this by normalizing locally (per company), and we saw some improvement. Figure 6, Figure 7 and Figure 8 show the prediction plot in comparison to the actual stock prices at epoch 5, epoch 10, and epoch 15 of training.

The training plot for the new approach is shown in figure 9.

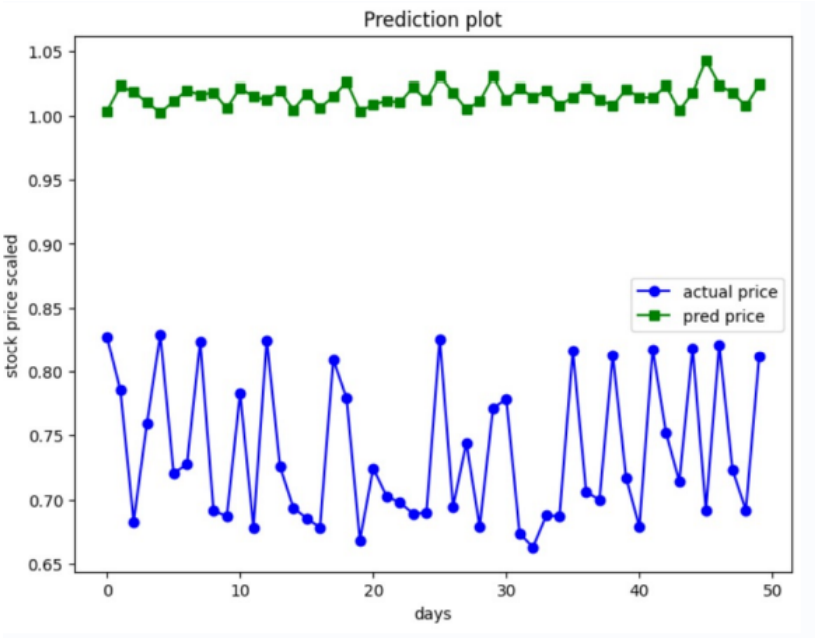

Figure 6: Prediction plot at epoch 5 - local normalization

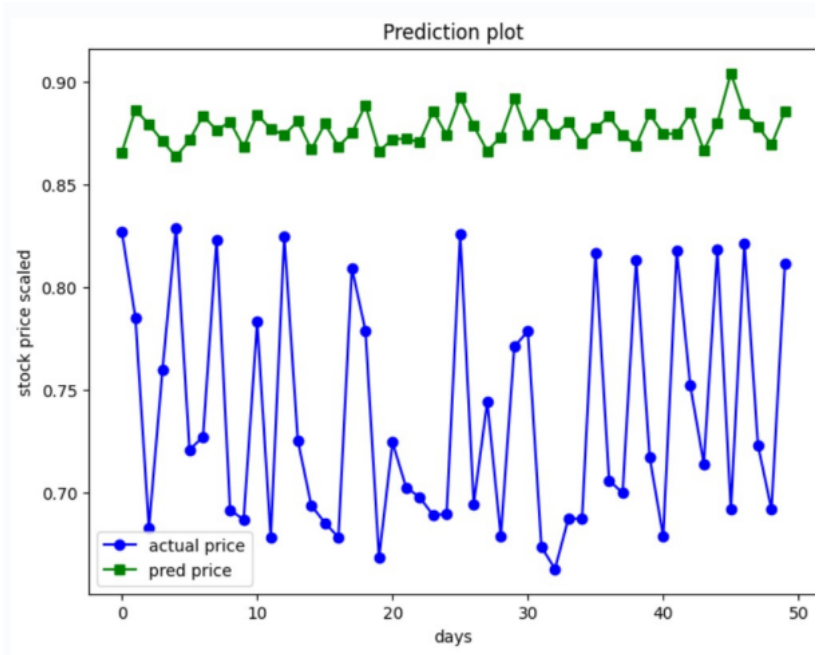

Figure 7: Prediction plot at epoch 10 - local normalization

Referring figure 8 and 9, we can come to two conclusions. Firstly, the model is able to learn patterns correctly however, the price it learns is for some reason squished. Secondly, the model cannot really be converged further. We reached the minimum loss that the model could potentially achieve.

These conclusions, lead us to roadblock 3. How to "unsquish" the predicted results. A larger model perhaps?

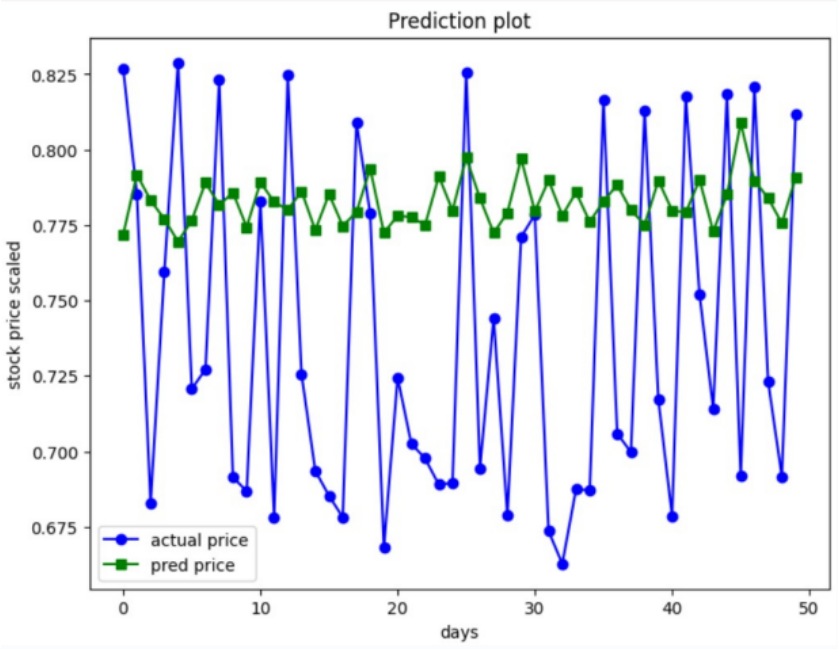

Figure 8: Prediction plot at epoch 15 - local normalization

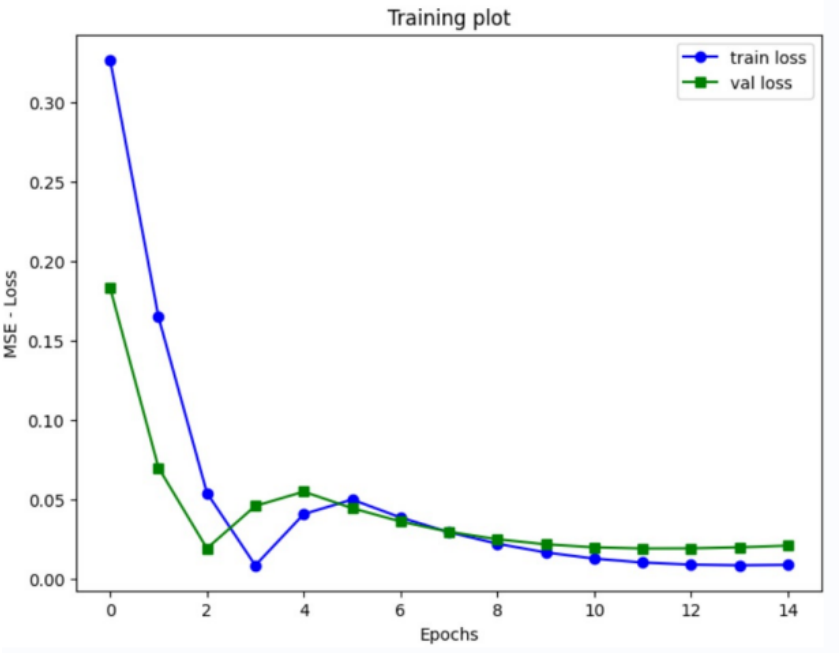

Figure 9: Training plot - Local Normalization

We created a new model on top of the current model to improve the predictions. The new model increased the complexity by introducing an extra layer of Graph convolution, and an extra layer of the LSTM model as well. The hypothesis was that the extra parameters would help the model perform better.

The forward pass in the new model looked something like this:

Listing 2: Forward pass code for the new model

```
def forward(self, x, edge_index, edge_weight):
    h = None
    c = None
    h1 = None
    c1 = None
    for i in range(self.window_size):
        input_x = x[:,i].view(num_nodes,5)
        input_x = self.linear_s(input_x)
        h1 ,c1 = self.recurrent(input_x, edge_index, edge_weight, h1,
            c1)
        h1 = self.a(h1)
        h, c = self.recurrent2(h1, edge_index, edge_weight, h, c)
        h = F.relu(h)
        h = self.linear(h)
        h = self.linear_e(h)
    return h
```

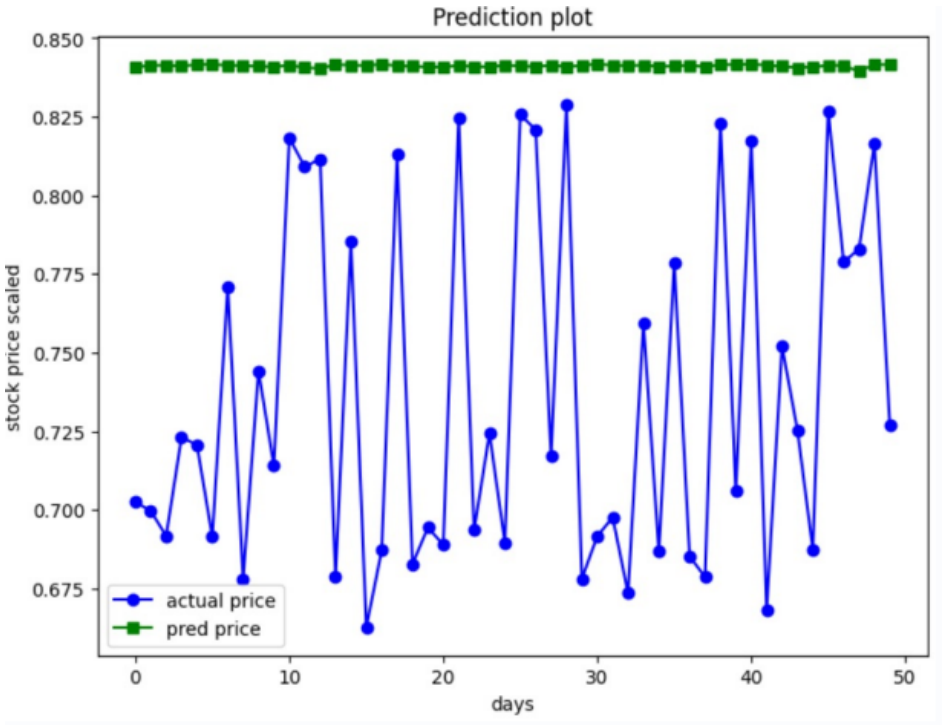

Figure 10: Prediction plot - New Model

Figure 10 shows the prediction plot for the new model. Evidently, it performs worse.

In a final attempt to solve roadblock 3, we chose to dig deeper. It was clear that the loss function was maybe an issue. This is because, we already had scaled down values for the stock prices. In a highly scaled graph environment, the average loss is bound to get small. Squaring on top of that did not help. Hence, we decided to use Huber loss with delta=1.

The training plot for the same is given in figure 11, and the prediction plot for the same is given in figure 12.

Referring both the figures, we see that huber loss did not help that much either.

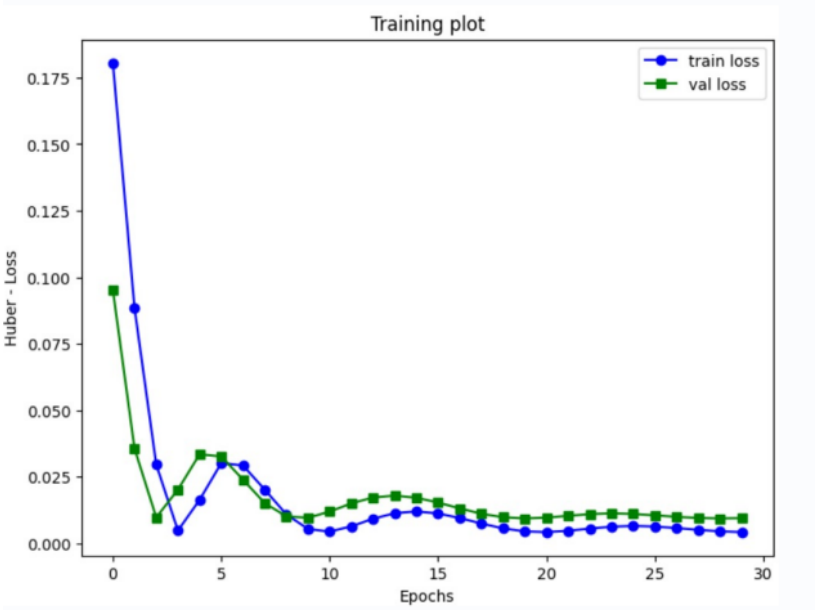

Figure 11: Training plot - Huber loss

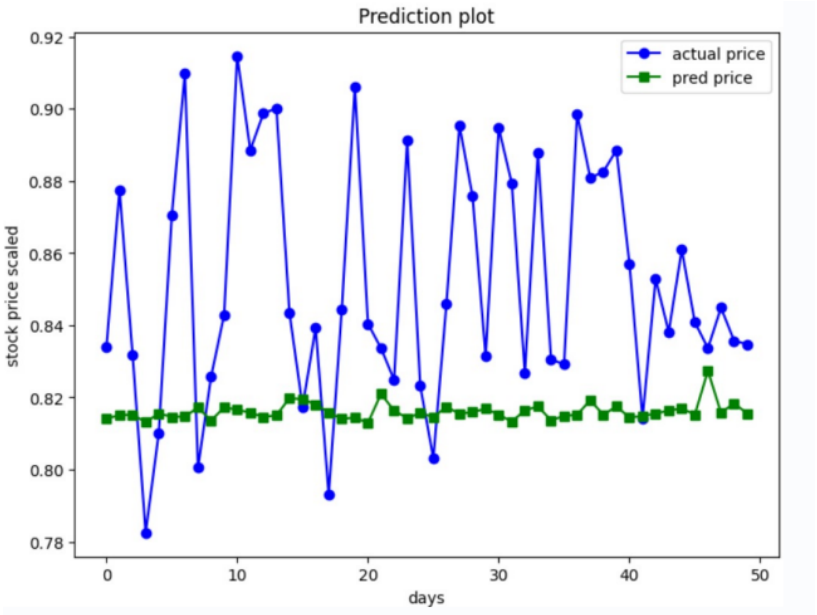

Figure 12: Prediction plot - Huber Loss

Conclusion

Maybe graph neural networks are not the best for day-to-day predictions. Maybe long-term predictions may cause better graph signals per node. Right now, the signals maybe do not change as much. There is not a lot of price variance in a 31-day period.

Over and above that, stock price prediction using the given OCHLV input is not enough for any machine learning model. There are a lot of factors affecting the price of any given stock.

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. Hence, we would like to do the following to improve the predictions and to also strengthen our conclusions.

Firstly, we would like to employ long term predictions to allow for a decent variance in the signals per node. Secondly, we would like to explore models like ASTGCN, that learn edge weights for us. This can be beneficial as we can rely on a the model to learn weights instead of writing fixed correlation matrices, Lastly, we can also create dynamic correlation matrices and use architectures like Dynamic TGCNs to allow for dynamic edges during prediction.

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.com/amaygada/Stock-Price-Prediction-using-GCNs