Exploring Various Machine Learning Methods in Stock Price Prediction

Tomasz Jezak, Matthew Napoli, David Cruz-Sanchez, Ibrahim Bakhiet

Abstract:

Our economy is based on a number of sectors, with the stock market being one of the most significant. In order to try and anticipate the price of a later stock price given a company's dataset, this project involved building three algorithms. This project's primary objective, in our opinion, is to use academic research and contemporary technology to forecast a company's trading stock price based on its stock history. The group used three different programs: Transformers, Reinforcement Learning (RL), and Long-Short Term Memory (LSTM). Our goal for this project is to determine which program will produce the most accurate stock prediction, minimal loss, mean absolute error (MAE), and mean absolute percent error (MAPE) by analyzing the Starbucks historical dataset that is accessible on the Yahoo finance website and applying it to our programs. Because the LSTM algorithm could calculate the minimum error loss in a basic model using a single LSTM cell, we were able to determine that it made effective use of the dataset.

Introduction:

The project's primary goal was to forecast a company's future stock price by using its historical stock price data. We are aware that a company's trading stock price is influenced by a number of different elements. In order to take it into account, we look at the company's historical stock data, which shows us how its price has fluctuated over time. This will also address the outside variables that impact the value of the stock price. We decided to concentrate on this subject because precise stock price forecasting will enable investors and financial advisors to make well-informed decisions that may have a positive effect on the economy. Because of the project's importance, personnel will be able to maximize results by maximizing benefits and reducing risks.

Background:

For stock price prediction, researchers used conventional statistical models before implementing machine learning techniques. When observations were collected over a period of time, time-series analysis (such ARIMA) was used; however, it should be noted that these analyses are obviously reliant on time, which adds another factor to take into account (Talaviya et al). But in order to maximize trading tactics, machine learning techniques—like reinforcement learning, which we used—have been increasingly popular in stock forecasts in recent years. Within the category of reinforcement learning, there are also other sub-techniques that are used, such as Temporal-Difference (TD), where the algorithm learns from raw experience without the need for a model given the dynamics of its environment, such as rewards and the likelihood of the next state (Lee et al). With the use of recurrent neural networks (RNNs) and LSTM, usage has increased. Transformer-based models have been revived to show temporal patterns in stocks, despite their original purpose being the natural language process (Muhammad et al).

Methods:

The team applied a comparative analysis strategy to the several methods (LSTM, RL, and Transformers). Long-range dependencies in the sequential data were captured by the LSTM model by handling sequential memory. When it comes to capturing temporal dependencies in stock price movements and changes, LSTMs shine. In essence, they retain or identify trends and patterns seen in the dataset. This method updates the weights based on past stock prices using backpropagation, which aids in forecasting future stock prices. Trading methods were optimized by the RL model to gain insights from the historical information. The Advantage Actor Critic Tutorial (A2C) algorithm, which uses both actor and critic networks, was the one used for reinforcement learning. At each stage, the actor makes a decision. The critic then assesses the actor's performance and offers suggestions. The actor considers the criticism and applies it to the next stage, selecting the better course of action (Wang et al). To optimize financial returns, the agent also used a Deep Q-Networks (DQN) technique. The transformer model processed

historical data and highlighted pertinent elements using an encoder-decoder architecture, while the decoder produced predictions for future stock prices (Muhammad). We calculated the loss/error for each model using MAE and MAPE, and then we used this metric to compare the models' respective levels of effectiveness and efficiency.

Theory:
The capacity of the LSTM model to identify long-term dependencies is centralized. Because LSTM cells are able to sustain themselves, pertinent data and information can be remembered and stored for extended periods of time. Since Q-learning is an evolved version of it, Q-learning provides the theoretical basis for Deep Q-Network (DQN). We now have a solid grasp of the stock market thanks to the developed ideas of Q-learning, neural network function, TD, and other related topics. This understanding may then be applied to a variety of other industries. In the domain of stock market prediction, the theoretical underpinnings of transformers enable us to find significant patterns that can facilitate the capture and identification of stock data via feedforward networks, normalization, pre-training, etc.

Data Description:
To obtain the complete dataset of Starbucks' historical stock value from the earliest date to the most recent, we employed the “yfinance” package. This provided us with sufficient information to generate fair and accurate predictions. Apart from the extensive duration of data, it offers a varied range of values that aid in instructing our models about external events that need to be taken into account. In order to access and read the data appropriately, we divided it into five categories when we called it: "open, high, low, close, and volume." To make it easier to facilitate, we also preprocessed the data using normalization.

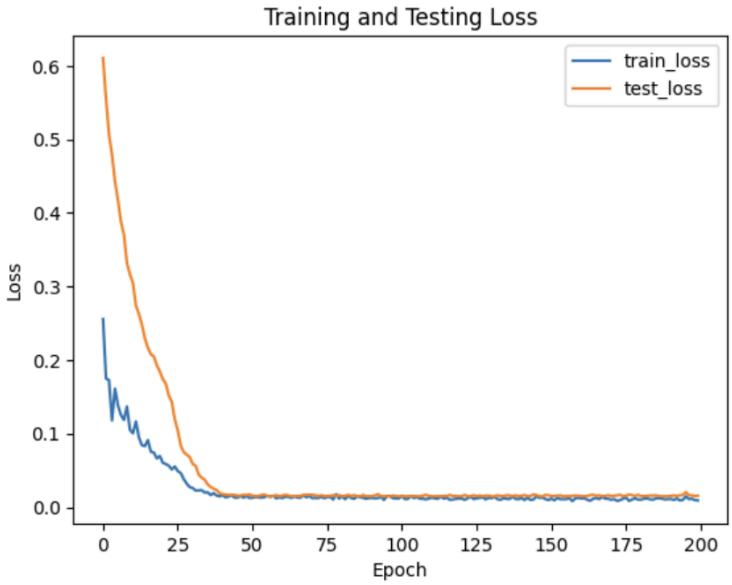
Results:
After comparing the models, we concluded that the LSTM model was the most effective since it could deliver the lowest error bound with just one LSTM cell. Because we thought the exploration rate was insufficient to fully examine the reward, the outcome RL had a somewhat bigger loss error calculated, which varied in various situations. The RL approach was developed from a fundamental standpoint. The transformer model produced impressive results, but occasionally it was unreliable because the encoders would become extremely unstable. We are continuously learning about this technology because encoders are a relatively new development in this industry.

GRU Model:
An extensive synopsis of the obstacles encountered and the performance results obtained throughout the development of our stock price prediction GRU model is given in this part. Since we just used OHLCV data, which is basic pricing data, we decided to employ a single GRU cell as the primary component of our data. We considered that the 9,000 parameters being tuned between our GRU cell and the Dense layer were sufficiently sophisticated to fit the data. GRU produced a somewhat better training/testing accuracy and was noticeably faster across 200 epochs, which is why we selected it over the LSTM cell.

Model Development and Challenges:
We imported and utilized the data using Python's {yfinance} model. This is a publicly available Python module that lets users input price history and get trading statistics on a certain Ticker. Typically, you may find this information online via Yahoo Finance. We encountered numerous difficulties when initially developing the GRU model, as it would frequently break. We spend a great deal of time attempting to determine why our label forecasts turned out to be wholly incorrect. We utilized MinMaxScaling to normalize the data after realizing that this was a critical error. After doing this, we discovered that even if our mistakes were less, the model should be optimized for a certain, valuable statistic. We selected closing prices because they provide crucial details about the price at which a stock has most recently traded, as opposed to High/Low, which provides information about the maximum and lowest prices that a stock has

traded at during the previous timestep but not the most recent price at which a stock is trading. Since Yahoo Finance is the primary source of free financial data, the data has a resolution of daily timesteps. We decided to fix our model to trade Starbucks stock {SBUX} because of its large amount of data and because it is not as liquid as an index like $SPY. Our hypothesis was that the model could learn clearer deterministic factors instead of an index that is noisy and liquid like $SPY.

Training Process and Hyperparameter Tuning:



The last 20 years' worth of daily price data served as our training set, and this produced the best results for testing and training errors, both of which were less than 0.01. Since SGD was one of the class's suggested optimizers, we chose it; there were no appreciable differences between ADAM and SGD. Additionally, we liked SGD's more adjustable parameters. Our LSTM model was trained for 200 epochs in the context of stock prediction using a learning rate of 0.001, a momentum of 0.1, and these parameters. Stable convergence throughout training is sometimes facilitated by starting with a suitable learning rate of 0.001. Momentum at 0.1, which incorporates a portion of the prior update, can aid in accelerating convergence. We chose 200 epochs since we saw that testing and training errors were still slightly declining. This suggests that there is a balance between training the model and preventing overfitting, however it is important to keep an eye out for overfitting as training goes on. Since our loss function evaluated at the validation data converged to lower losses much more quickly than with Mean Squared Error, or MSE, we ultimately used a custom function to calculate the Mean Absolute Error to optimize only for the closing price, or MAE, over MSE. We preferred MAE because it provides error in absolute terms and was easier to understand. The size of the sliding glass window, which we named {data_samples} = 30, determined the length of this 3d array. We experimented with this hyperparameter to split the data and found that going over 30 yielded a higher testing error and lower training error, while going below 30 yielded both a higher training and testing error. These steps were taken in order to create the LSTM model.

Performance and Results:

We discovered the strength of our model and were ultimately quite pleased with the outcomes of our model. Even though our goal was to reduce testing errors, we were unable to achieve an MAE of less than 0.01. Furthermore, we observed that a large portion of the testing forecasts appeared to be underestimating the price. Based on our theory, this was probably because the bulk of the price history was on the lower end. When we extended our model to other tickers, we found that it likewise functioned effectively, with train/test errors of about ~0.02.

Challenges:
Our largest obstacle was attempting to assess model performance using test data. As previously indicated, we experimented with a wide range of loss functions and measures before settling on one that performed as planned. Although one might assume that a basic neural network should be sufficient, it is surprising to learn that many small details have a significant impact on the performance of the model. especially for tasks like data splicing and preprocessing that aren't directly connected to machine learning.

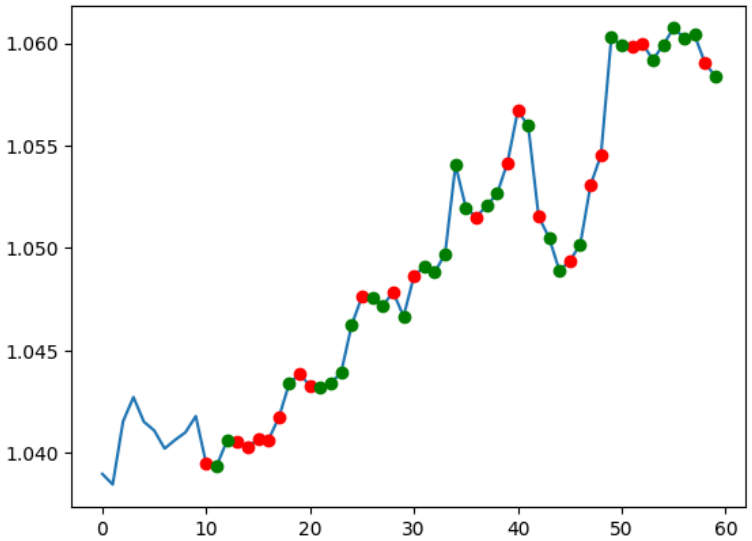
Conclusion:
This was an excellent illustration of the power of a single LSTM cell and highly educational. It would be interesting to investigate if this approach could be applied to minute-by-minute data.

Reinforcement Learning Model (RL):
The project made use of a customized Gym environment and the features of Stable Baselines 3 to help the agent learn in a stock market simulation.

Model Development:
With the use of the Gym environment and the StockPredictionEnv, the RL model sought to create a realistic stock trading simulation. In order to optimize financial returns through strategic trading decisions, especially between long and short stock positions, the agent opted for a DQN with an MLP policy.

Training Process and Hyperparameter Tuning:
The agent completed a rigorous 210,000 timestep training program. The agent's policy and value estimations were improved by this extended training period, as evidenced by the policy and value losses that decreased over time, pointing to a steady and efficient learning trajectory. In order to allow the agent to learn and adjust to different trading strategies, hyperparameters were meticulously adjusted to create a delicate balance between exploration and exploitation.

Challenges:
There was a steep learning curve involved in navigating the intricacies of reinforcement learning algorithms because these approaches differed greatly from the more conventional machine learning techniques that the group was more used to. The intricacies of environment setup, understanding the dynamics of agent-environment interactions, and mastering the nuances of algorithms like Deep Q-Networks demanded both time and dedicated effort. A significant observation was the agent's initial predisposition towards short positions, as evidenced by consistently red plots in its trading strategy visualization. This behavior pointed towards possible issues in the exploration strategy or the reward structure of the model. Another challenge was ensuring that the observation shape remained consistent at (5, 2) across the model's training steps, a critical requirement for the agent's decision-making process.



Note this plot of stock price (y-axis) over time (x-axis), with green points indicating buy decisions and red points signifying sell decisions made by the agent.

Advantages of the RL Model:
The model exhibited a critical ability in the erratic realm of financial trading: the ability to adjust its strategies in reaction to shifting conditions in the stock market. This adaptability is critical in an environment where market dynamics are prone to abrupt, dramatic shifts. The model's achievement in financial performance was another noteworthy plus. The model showed a 4% rise in capital with a total profit of 1.04, which is in line with the main goal of capital growth in stock trading. This finding is particularly noteworthy since it suggests that the model could be a helpful resource for financial decision-making. Additionally, a high explained variance of 0.697 was attained by the model. In comparison to the rewards the model was given, this statistic showed that the model's predictions were quite accurate. This degree of precision is necessary for a model to operate in the complex and data-driven stock trading business since it shows a firm understanding of the underlying patterns and trends in the market data.

Disadvantages of the RL Model:
Notwithstanding these advantages, there were certain difficulties with the project. The average episode reward of 0 indicated that the model tended to take a neutral position, which was one of the key causes for concern. This trading strategy's neutrality could point to possible constraints in the environment's reward structure or a very cautious approach on the part of the agent. In any case, it suggests a course of action that needs to be followed in order to encourage more decisive and potentially successful trading actions. Even though the low entropy loss was an indication of policy convergence, it also raised questions about the model's exploration method. A very low entropy loss could mean that the model is not looking at the action space in enough detail, which could lead to a lack of diversity in the trading strategies the model identifies. This part of the learning process is critical, particularly in a complex and varied environment like the stock market where exposure to a variety of scenarios is essential for creating reliable and successful trading methods.

Conclusion:
In stock trading simulations, the created reinforcement learning model showed encouraging results and produced moderate financial benefits. The project's results demonstrate the promise of reinforcement learning for financial applications; nonetheless, it should be noted that much more work needs to be done. More hyperparameter tinkering and making sure the model generalizes to stock data other than Starbucks are two areas of future improvement. We would also think about enhancing feature engineering, exploring new avenues, and fine-tuning the reward structure. Additionally, investigating different algorithms or ensemble techniques may offer insightful information about how to improve the trading strategies of the model.

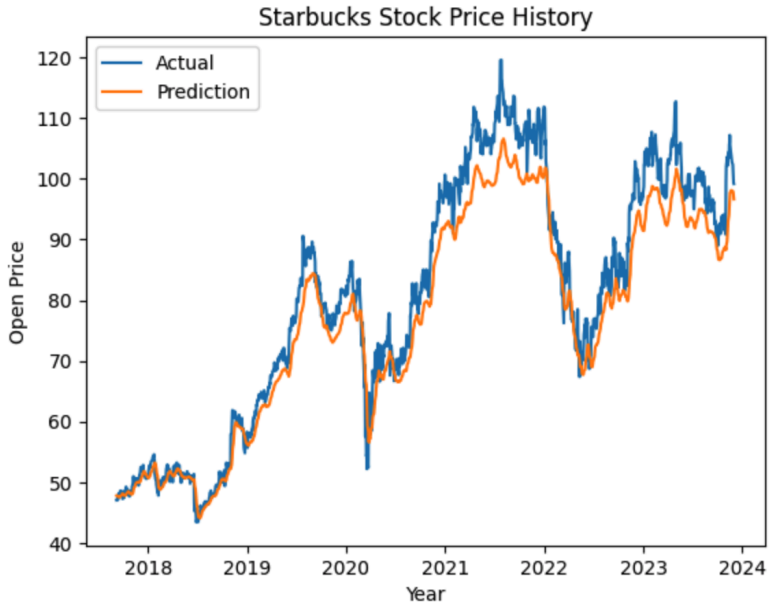
Transformer Model:
The Encoder Transformer model forecasted the future closing stock values by applying self-attention processes to time-series data.

Model Development:
Our aim is to estimate the closing price of the stock on the following day by taking data from the yfinance library and splitting it into sets of five days that contain the high, low, and closing price of the stock. Our Transformer architecture consists of a completely connected network after a stack of encoders that make use of the self-attention mechanism. One layer in each encoder is made up of the multi-head attention layer, which operates as a parallelized collection of attention mechanisms and stabilizes training by coming after layer normalization. Overfitting is avoided with the help of dropout. The input enters a deep,

fully linked layer after passing through the encoder stack. The temporal dimension is compressed by global average pooling, and the network's resilience is then increased using dropout.

Challenges:
Transformers' strange and intricate architecture made it a difficult method to approach. Given what we have learned about feed-forward networks, recurrence, and the multi-head attention mechanism—which, like RNNs, learns information over time and has some notion of attention—the typical encoder-decoder setup requires a conceptual overhaul. However, this module contains multiple attention mechanisms that simultaneously focus attention on different parts of a sequence. Although we initially intended to employ both the encoder and decoder design, in actuality the encoder-only architecture made sense, therefore we had some difficulty implementing the model. Since the decoder does not need to understand the input, we kept using the stacked encoder model, which produced largely satisfactory results.

Training Process and Hyperparameter Tuning:
Some of the models from which we gained knowledge influenced our selection of hyperparameters. The options included 38 batch sizes, 1e-6 training rates, 0.10 dropout rates, and 100 epochs of training. The model's capacity for prediction did not increase after 100 epochs. We tried changing the batch size, but this made the system perform worse, so we decided to stick with it. Despite training for 100 epochs, our model immediately plateaued; within a few steps, it was able to achieve a loss below 0.001, and it remained there for the remainder of the training. Over time, the mean absolute error steadily dropped to a level of 0.006, and the mean absolute percentage error peaked at around 2000.



Advantages of the Transformer Model:
Transformers have the advantage of having fewer parameters than LSTMs and other RNNs, and because of their parallelized attention mechanism structure, they also have substantially faster training periods. Additionally, the multi-head attention mechanism is more adapted to establishing long-term dependencies, which improves the predictive capacity of the model. Our model used 194 KB, far less than predicted, and employed approximately 50,000 trainable parameters. Even though the model didn't perform as well as

the GRU model, it still produced passable results. Given how straightforward the encoder construction was, there is still potential for development in this model. Malibari et al., a model that impacted our work, employed a Transformer model akin to the Vision Transformed and predicted future closing prices of two Saudi indices with over 94% accuracy, demonstrating the Transformer's potency in price movement prediction.

Disadvantages of the Transformer Model:
Transformers are far more unstable than RNNs like LSTM, despite having a respectable overall performance. Unlike our GRU training results, different training iterations produced significantly varied error rates. Frequently, using the same hyperparameters produced entirely different outcomes.

Conclusion

We used a multimodal approach to stock market prediction in our project, combining Transformers, Reinforcement Learning (RL), and Gated Recurrent Units (GRU) to demonstrate our inventiveness and ingenuity. We were able to leverage the distinct advantages of multiple AI models—each selected for its applicability and promise in the intricate and dynamic field of financial data analysis—by combining a variety of modeling approaches. We made a strategic choice in using GRUs and Transformers since we knew that stock market data is sequential and time-sensitive. GRUs work particularly well for time series forecasting because they are good at capturing time dependencies. Transformers provide a more sophisticated method of representing financial time series and are excellent at handling long-range dependencies. When combined, these models demonstrate our dedication to utilizing cutting-edge AI methods to improve stock price forecast accuracy. By including Reinforcement Learning in our toolkit, we were able to move past simple prediction and into the domain of strategic trading choices. The dynamic and frequently unpredictable nature of stock trading is a good fit for RL's ability to learn from and adapt to interactions with the market environment. In addition, our goal was to evaluate the challenges, results, and effectiveness of GRU, Transformers, and RL to identify the advantages and disadvantages of each AI technique. Our project's investigation of these diverse approaches showed a high degree of technical proficiency as well as a strong dedication to comprehending and utilizing sophisticated AI techniques in useful, real-world situations.

Works Cited

Lee, Jae W. “Stock price prediction using reinforcement learning.” https://ieeexplore.ieee.org/abstract/document/931880/authors.

Muhammad, Tashreef. “Transformer-Based Deep Learning Model for Stock Price Prediction: A Case Study on Bangladesh Stock Market.” https://arxiv.org/abs/2208.08300.

Talaviya, Avikumat. “Time-series analysis and stock price forecast modeling: All you need to know.” https://medium.com/@avikumart_/time-series-analysis-and-stock-price-forecast-modeling-all-you-need-to-know-fe66c06e50ae#:~:text=Time%2Dseries%20analysis%20is%20the,how%20different%20strategies%20affect%20performance.

Wang, Mike. “Advantage Actor-Critic Tutorial: minA2C.” advantage-actor-critic-tutorial-mina2c-7a3249962fc8.