Stock Price Prediction with Sequential Models

Yuanhang Luo
Department of Computer Science
Stanford University
`royluo@stanford.edu`

Abstract

This is a deep learning project on predicting stock market prices using neural sequential models, including RNN and Transformers. The dataset we use are stock price data including open price, high price, low price, close price, and volume traded on each day. We aim to predict the stock price of the next day using data from previous days, with nested sliding window to prepare the training and testing data. Experiements show that LSTM performs well on the task and outperforms Transformer.

1 Introduction

Stock prices are always changing, and it has been a very interesting problem to predict the stock prices based on available information. If we were able to predict changes of stock prices successfully for more than half of the time, we will be making well-informed decisions on investment and trading.

There are many factors that are influencing stock prices in the market, and many of them cannot be quantified or easily obtainable. However, historical prices and trading volumes do provide some information in deciding the future prices. Therefore, in this project, we aim to use neural network models to predict stock prices using history stock prices and trading volumes.

2 Related work

There are several areas of related works: traditional statistical methods and machine learning (especially neural network) methods.

Ariyo et al. [1] uses autoregressive integrated moving average (ARIMA) models to predict stock prices. ARIMA models are efficient and robust when it comes to short-term time series prediction especially in the finance field. Lee et al.[2] and Merh et al.[3] compared the performance of neural network models and ARIMA models.

As for the neural network side of researches, Lu et al. [4] proposes a CNN-BiLSTM-AM model , combining Convolutional Neural Networks, Bidirectional Long Short-Term Memory, and Attention Mechanism to predict stock prices. It shows the power of how modern and complex neural networks can be applied on stock price prediction tasks. Selvin et al.[5] proposes a model independent approach to find the latent dynamics underneath the stock prices instead of simply predicting the future stock prices. Sonkiya et al. [6] proposes a model using BERT and GAN. It utilizes features including news, technical indicators, and historical prices. It is using ensemble methods with high performance.

3 Dataset and Features

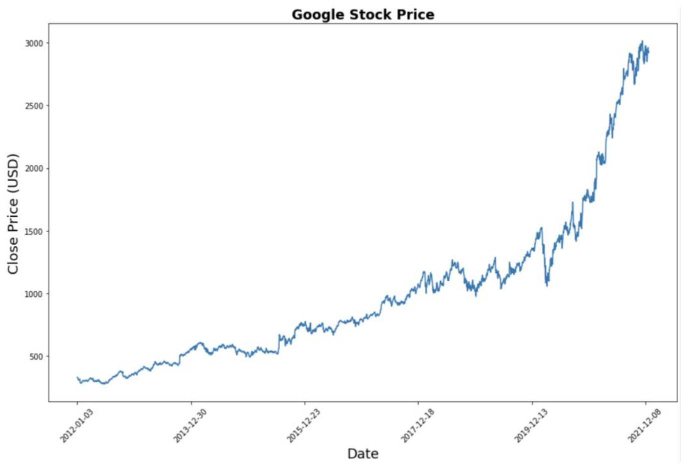
3.1 Dataset

The dataset we are using is Google stock prices from 2012 Jan 3 to 2021 Dec 30. The dataset is downloaded from Yahoo Finance[7].

Each data entry has information of a day when stocks are actively being traded. Since there are dates that stock is not being traded, there are 2,516 rows in the dataset, instead of 3,650 rows.

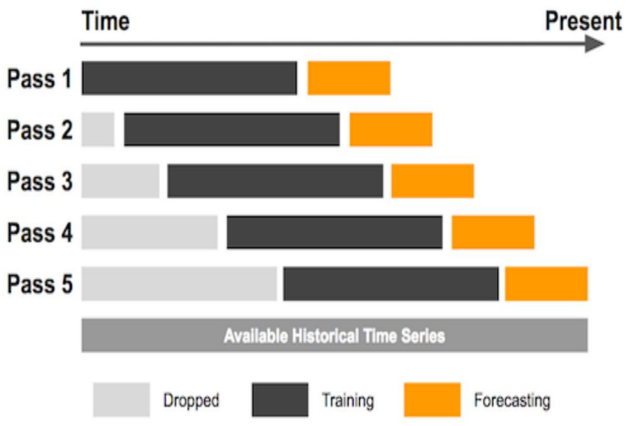
For each row (each trading day), there are five features: open price, high price, low price, close price, and volume. The feature names are self-explanatory. We will be using these features to predict the close price of the next day.

Below is a figure of the close price in the dataset.



3.2 Preprocessing

Nested Sliding Window We use nested sliding window method to process dataset into training and testing dataset. For the entire dataset, we first use a big sliding window to select 5 passes of windowed datasets from it. Then for each pass of the big window, we use sliding window method to convert data into our desired format, and set the 80%:20% training and testing split. Below is a illustration of the nested sliding window.



For each pass of dataset, first we define the length of the history data in the past to predict the next day price. This length called seq_length is used as the size of the window to slide over the historical data.

Then we select the previous seq_length days of the historical price and volume data, and the corresponding label of this window of data will be the close price of the next day. For example, seq_length is 10. Then day 0 - day 9's prices and volume will be the features, and the close price on day 10 will be the label.

To avoid data leakage, sliding window will be applied to the training and testing set respectively after the dataset split.

Scaling The magnitudes of the prices and volume are very different. Prices range from several hundred to a few thousand. While the trading volume range from many thousand to over 20 million. Considering the large difference in these values, we use two methods for scaling the data. The scaling is separate for prices and volumes.

Min Max Scaling We choose min and max value for the prices data, and scale the prices based on this pair of min and max value. Then we choose a different pair of min and max value for the volume data, and scale the volume based on this pair.

The formula we use for min-max scale is

$$X_{scaled} = \frac{2(X - X_{min})}{X_{max} - X_{min}} - 1$$

Standardization We standardize the data into 0 mean and 1 standard deviation. The formula is:

$$X_{scaled} = \frac{X - X_{mean}}{X_{std}}$$

To avoid data leakage, we only use the training set data to determine the min and max values. Then we scale the test data based on the training set min and max values.

4 Methods

4.1 Models

4.1.1 Long Short-Term Memory

Model Architecture One model we use is a Long short-term memory (LSTM) network [8]. LSTM network is a Recurrent Neural Network (RNN) capable of learning long term dependencies through forget gate, input gate, output gate, and cell state.

To increase the complexity of the LSTM network, we might stack L layers of LSTM together, so the hidden states of the cells in the previous layer $h_n^{(t-1)}$ serve as the input to the corresponding cells in the next layer $h_n^{(t)}$.

The input of the model are the sliding windows of historical prices and volume, with the feature dimension of 5. The output of the model is the predicted close price of the next day. We consider the output state of the final LSTM cell $h^L_{seq\_length}$ as the prediction result, so its dimension is 1. The length of the LSTM sequence is the seq_length we defined for the sliding window.

4.1.2 Transformer

Model Architecture Another model we use is a transformer network[9]. It is made up of positional encoding, embedding layers, and self-attention.

Transformer is very different from traditional Recurrent Neural Network models, because it is disregarding the time / order information in a sequence. Instead, "attention is all you need". The sequence is processed as a whole rather than sequentially as in regular RNN models.

Considering the fact that stock prices data relies very heavily on time, it will be impossible to ignore the valuable information. Therefore, we need positional encoding to encode the time information. It helps assign a value $P_{k,j}$ to a particular data v_k given its position k in the sequence. More specifically, it is defined as

$$P(k, 2i) = sin(\frac{k}{n^{2id}})$$

Table 1: RMSE

Pass Number	Training LSTM	Training LSTM	Training Transformer	Testing Transformer
1st	7.28	9.57	88.89	37.22
2nd	9.67	8.24	46.10	72.07
3rd	10.22	20.24	53.64	177.25
4th	15.75	42.07	46.64	123.78
5th	26.19	82.11	120.04	423.26

$$P(k, 2i+1) = cos(\frac{k}{n^{2id}})$$

where d is the embedding dimension, and n is a scalar.

The inputs of the model are sliding windows of historical prices and volume, and the outputs are the corresponding close prices.

4.2 Loss Function

The nature of the task is a regression task, so the loss function is Mean Squared Error (MSE) between the predicted and true value.

$$MSE = mean(l_1, ..., l_N), l_n = (y_{predicted} - y_{true})^2$$

5 Experiments/Results/Discussion

5.1 Evaluation Metrics

Given that the task is a regression task, we use Root Mean Squared Error (RMSE) to evaluate our results on training and testing sets.

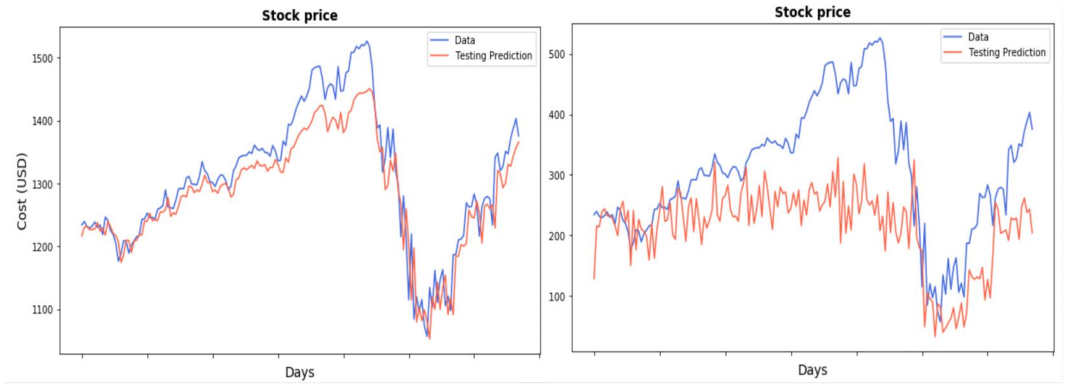
5.2 Experiments and Results

We have run several experiments on testing performances between LSTM and Transformer models. Here are some results.

Generally, from table1, we can see a trend of increasing training and testing RMSE in both models. This might be due to the high variance of the dataset. As we see, the stock prices become more and more volatile and have higher and higher variance as time goes on from 2012. Therefore, it is more and more difficult for the model to learn enough information to make a powerful prediction of the future.

To mitigate this, we can make the model to predict less far in the future due to the possible different distribution of past data and future data. We might also need more data, and increase the complexity of the model.

Another difference lies between the performance of LSTM and Transformer model. LSTM is performing better than Transformer in many experiments, as seen from the following graph. This might be due to the lack of long term dependencies between the data. It could also be due to the fact that the information is not fitting in well into this Transformer model and we need to investigate more about how we can best fit the Transformer model with time-series data.



LSTM vs Transformer prediction

5.3 Discussion

There are several things insightful about this project. One of them is the high variance of data and its impacts. Stock prices have been very volatile, and in our dataset, it can be seen especially after 2018. The trend and patterns become very different from the data before. Therefore, it is inevitable that the prediction of stock prices is a very difficult task - it is very difficult for neural networks to learn from the data with very different patterns in the past and make predictions about the future.

Another thing is the many factors that are influencing stock prices. Past stock prices and trading volumes do carry much information, but there are also lots of other factors that are influencing the stock market. Those factors are not included in the dataset, and they are making really huge impacts on the stock prices.

Additionally, it is interesting to see that LSTM is outperforming Transformer in the experiment settings of this project. Even though we did some results analysis on this, it will still be interesting to see how stock prices can be better predicted using Transformer models to outperform LSTM models.

6 Conclusion/Future Work

In this project, we have explored various sequential models for the task of stock price prediction. So far as we explored, Transformers are not outperforming LSTM for this task. However, there are many aspects that might be limiting the performance of Transformers in this project, including the lack of long term dependencies between data, lack of dataset, and lack of model complexity and training power.

In future, it is interesting to explore more into how time series stock price data can be applied on Transformer models. One direction will be the positional encoding. There could be better ways to represent time information in time series data, different from the current prevalent positional encoding method used in natural language processing. Another direction could be investigating how the data and the model can be fit together better. For example, feature dimensions, sequence length, input and output formats, etc.

Moreover, it will be interesting to see how we could incorporate other information into the stock price prediction model. One interesting direction will be including news into the prediction model, because it is influencing human belief and decisions.

7 Contributions

Yuanhang Luo contributed entirely to the whole project.

References

[1] Adebiyi A. Ariyo, Adewumi O. Adewumi, and Charles K. Ayo. Stock price prediction using the arima model. In *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, pages 106–112, 2014.

[2] Kyung Joo Lee, Sehwan Yoo, and John Jongdae Jin. Neural network model vs. sarima model in forecasting korean stock price index (kospi). 2007.

[3] Nitin Merh, Vinod Prakash Saxena, and Kamal Raj Pardasani. A comparison between hybrid approaches of ann and arima for indian stock trend forecasting. 2010.

[4] Wenjie Lu, Jiazheng Li, Jingyang Wang, and Lele Qin. A cnn-bilstm-am method for stock price prediction. *Neural Computing and Applications*, 33:1–13, 05 2021.

[5] Sreelekshmy Selvin, R Vinayakumar, E. A Gopalakrishnan, Vijay Krishna Menon, and K. P. Soman. Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1643–1647, 2017.

[6] Priyank Sonkiya, Vikas Bajpai, and Anukriti Bansal. Stock price prediction using bert and gan, 2021.

[7] Yahoo Finance. Goog stock prices. `https://finance.yahoo.com/quote/GOOG/history`. Accessed: 2022-05-01.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.