An Application Predicting Stock Price Trends using Neural Network

Yi Man
yi.man@colorado.edu

Han Ngo
han.ngo@colorado.edu

1. Abstract

Stock price trend prediction is an active research area for data scientists. Therefore, in recent years, significant efforts have been put into developing models that can predict for future trend of a specific stock or overall market. Also, computer scientists and researchers showed that deep learning technologies e.g. artificial neural network algorithms come handy and useful and handling historical stock data and so help predict stock price fluctuations. We create Stockalyzer to help us project the tendency in stock price changes with regards to past datasets. Understanding trading strategies, we use deep learning model and develop the financial comparisons to forecast the interactivity between change in price in overall market which will help us foresee stock trend movement and better decide which may have more or less risk when it comes to buying/selling stock.

The application of knowledge is all about buying and selling stocks. With stockalyzer, hedge funds, investment firms, and daily traders who seek financial security when it comes to making a decision can now know how to predict stock price in a specific time frame, can project financial analysis and comparison between different stocks, and better decide which stock they should invest in a overall target market.

2. Introduction

The overall purpose of our work will help clarify such following questions:

1. Given a set of symbols (company indexes), how would the model predict stock price change in a specified time frame?
2. How efficient does the neural network work towards different sets of data? Does it cause too much over-fitting or under-fitting?
3. What would be a good decision in different situations, e.g. if we decided to previously buy the stock and there is an increase in the stock price?
4. Once decision is made, based on historical data source, how does the program generate analysis for us to see if the action yields any profits?

3. Literature Survey

Financial stock analysis is a well-studied area that has seen much prior work. There are approaches that have been utilized to build efficient predictive models from Deep Learning technologies such as Neural Network [1], Support Vector Machines [2], Q-Learning, etc. to identify the correlation between past datasets and current public opinions' impact on stock movements. A much more detailed and well-developed version has been developed by Facebook team using additive models to predict certain sets of stocks in current years.

Moreover, collaborative open-source work has also been performed by Kaggle community members and competition candidates interested in exploring this domain. While many of these projects provide bases analysis of stock trends as a function of time, Facebook's Prophet model seems to be the most popular library to be used currently for its animating visualization and high validation accuracy.

4. Data Source

Index file:
https://raw.githubusercontent.com/datasets/s-and-p-500-companies/master/data/constituents-financials.csv/

Attributes: There are many features that we do not actually need in the context of just specifying the symbol (e.g. GOOG, MMM, ABT, etc.) and using the integrated list of those symbols to query data from Yahoo! Finance

Data file for 500 companies:
https://finance.yahoo.com/quote/%5EGSPC/history?p=%5E GSPC/
Attributes: Aspects relating to daily numerical value of a stock, this dataset contains the four columns with the opening and closing price per day and the extreme high and low price movements for individual stock for each day. Additionally noted and used in our context for comparison: Volume (the number of shares that got traded during a single day) and Adj Close (the closing price of the day that has been slightly adjusted to include any actions that occurred at any time before the next day's open. This column is to examine historical returns or when we are performing a detailed analysis on historical returns.)

5. Procedure

Data Collection and Integration
We have collected constituent financials and download the full S&P 500 data from Yahoo! Finance-GSPC to our dataset directory. The Python script load.py will fetch the prices of individual stocks in S&P 500 for 4000 days maximum, each saved to our dataset directory.

Data Cleaning
We will clean up the data set by removing non-applicable columns including Ex-dividend, Split ratio, Adj. Open, Adj. High, Adj. Low.

Data preprocessing
We will use content in one sliding windows to make prediction for the next and ensuring there is no overlap between two

consecutive windows. Layer inputs and outputs will form an unrolled version of recurrent neural network for training on Tensorflow. After that, for model validation and accuracy evaluation, we will use the latest 25% of data as the testing data.

We do some sort of cleaning and transformation for the comparison part. As we have different tables of stocks based on each every single day when there exists a unique price of this stock but not the other. And for monthly/quarterly overview, we calculated the cumulative daily return values for each month/quarter of a stock (for all attributes that we need) and concatenate data from different symbols of stocks in a specific time frame for comparison.

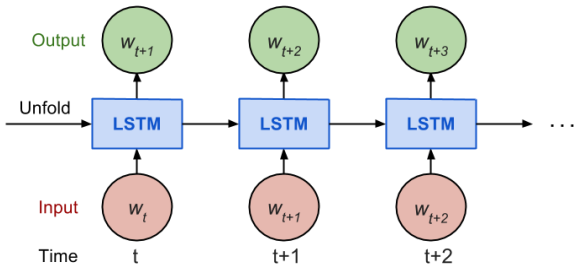
Data normalization

We will normalize the close prices in each sliding window. The task becomes predicting the relative change rates instead of the absolute values. In a normalized sliding window at time t, all the values are divided by the previous price to help prevent out-of-scale issues.

Model construction

The model is expected to learn the price sequences of different stocks in time. Due to the different underlying patterns, we will use embedding. We will create a Tensorflow graph using input variables from datasets that we have and the available optimizing algorithm built in TF/Keras library for gradient descent optimization.

For our focus in data with time frames, we found that recurrent neural network come accordingly useful. We construct the model first without no hyper-parameters being initiated and run some trials and errors with different set of variables and depend on Keras library for creating LSTM (Long Short Term memory) layers combined with a number of fully connected ones.



The RNN model we are about to build has LSTM cells as basic hidden units. We use values from the very beginning in the first sliding window W_0 to the window W_t at time t:

$W0=(p_0,p_1,\ldots,p_{(w-1)})$

$W1=(p_w,p_{w+1},\ldots,p_{2w-1})$

$Wt=(p_{tw},p_{tw+1},\ldots,p_{(t+1)w-1})$

to predict the prices in the following window W_{t+1}:

$$W_{t+1}=(p_{(t+1)w},p_{(t+1)w+1},\ldots,p_{(t+2)w-1})$$

Training/Testing

We will initiate different hyperparameters, start a running tf session, and saved our train model at the end which will be later used to evaluate on the last 25% of our datasets. There are a lot of approaches that machine learning enthusiasts pursue to forecast stock trends as the aforementioned

in literature survey section. For the sake of learning the whole process of data mining, we decided to try Neural Network approach. Moreover, reading researches about the relevance between public opinions and stock movement motivates us to discover add on a new feature to the training data sets.

5. Evaluation Methods

We will use mean square error as the loss metric and the RMSPropOptimizer algorithm which is built in Tensorflow for gradient descent optimization

6. Tools

For communication and sharing purposes, our group will work on the project through Github. The code will be mostly based on Python for submodules and Jupyter Notebook for most of the interactions.

We plan to utilize Pandas and Numpy libraries to aid data cleaning process. Tableau and Matplotlib for basic graphing visualizations.

For everything related to recurrent network, we utilize Tensorflow and its well-known library Keras to help with the whole process of building model, training and evaluating.

Most interactions will be based on Ubuntu CLI and data training process will also make use of our GPU instance on AWS.

7. Key Results

A - Neural Network Prediction Model

We wrote a script to download the needed data up-to-date dated back to 4000 days ago for each of the 500 S&P companies. They are all stored in folder ‘data’. We also completed the code to load the training data from csv files into the appropriately shaped numpy array and split them in batches of training/testing data with proportion of 9/1.

After we have the data ready, we noticed that all the data scale in each column differ a lot. So before throwing batches of data into the Recurrent Neural Network, we used Standard Scaler in sklearn to help us complete the normalization preprocessing step.

Now that we had the normalized dataset, we wanted the LSTM to learn the sin wave from a set window size of data that we would feed it and then hopefully we could ask the LSTM to predict the next N-steps in the series and it will keep spitting out the sin wave. After some researches, we noted that one of the important things was also to normalize the sliding window. Because running the adjusted returns of a stock index through a network would make the optimization process not converge to any sort of optimums for such large numbers. So to combat this, we have researched ways and decided to take each n-sized window of training/testing data and normalize each one to reflect percentage changes from the start of that window (so the data at point i=0 will always be 0). We used the following equations to normalise and subsequently de-normalise at the end

of the prediction process to get a real world number out of the prediction.

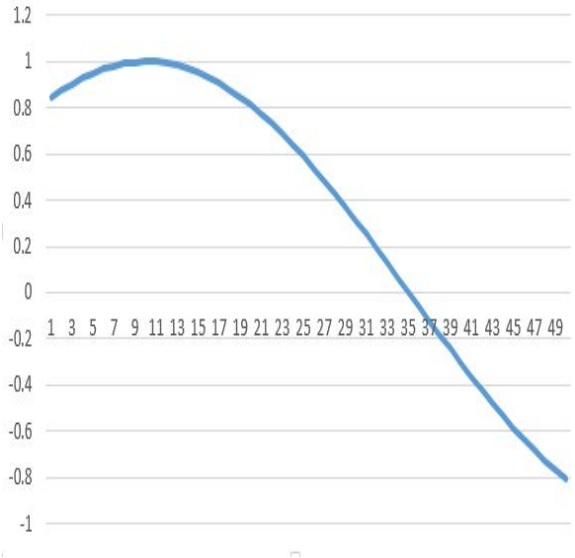
Normalization formula:

$$n_i = \left(\frac{p_i}{p_0}\right) - 1$$

Denormalization formula:

$$p_i = p_0(n_i + 1)$$

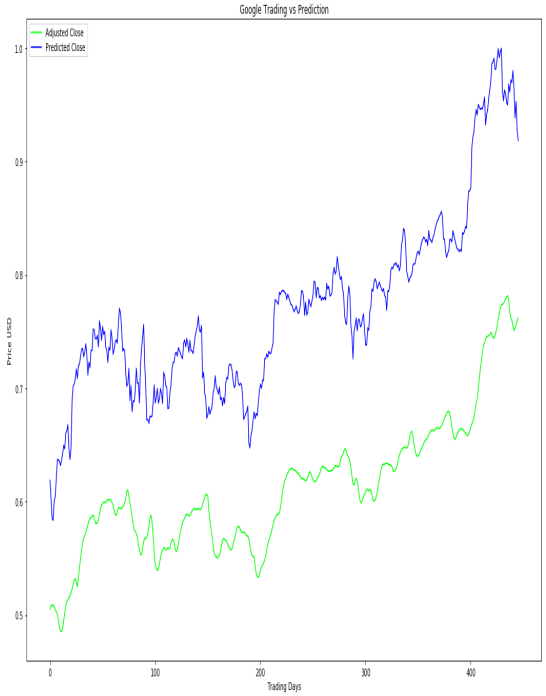
About the model, the way Keras LSTM layers work is by taking in a numpy array of 3 dimensions (N, W, F) where N is the number of training sequences, W is the sequence length and F is the number of features of each sequence. We chose to go with a sequence length (read window size) of 50 which allows for the network so get glimpses of the shape of the sin wave at each sequence and hence hopefully teach itself to build up a pattern of the sequences based on the prior window received. The sequences themselves are sliding windows and hence shift by 1 each time, causing a constant overlap with the prior windows.

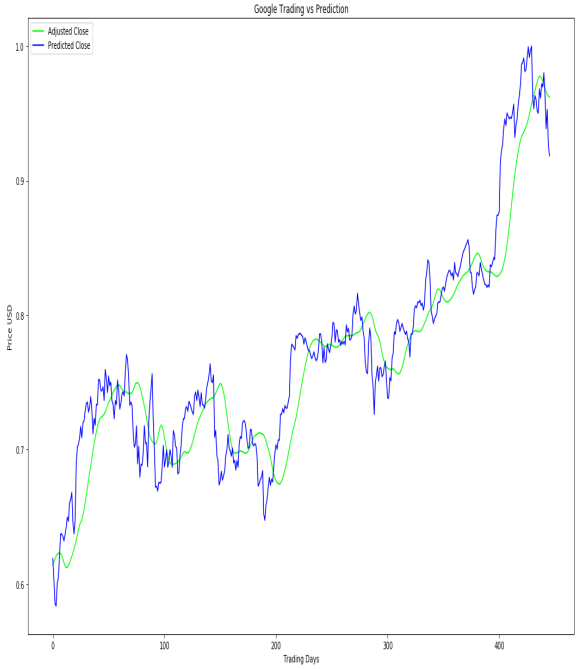


Example of ‘High’ data sequence from A.csv file

For the model construction, we have had some time to learn Tensorflow and Keras. So we started with using a simple network structure where we initially have one input layer (consisting a sequence of size 50 data values) which feeds into an LSTM layer with 50 neurons, that in turn feeds into another LSTM layer with 100 neurons which then feed into a fully connected layer with a linear activation function which will be used to give the prediction of the next time step.

This is the result from a simple network which comprises of three layers (2 LSTMs and an activation layer). With a number of 10 epochs, the model achieved 0.2607 RMSE which is pretty high.





Simple model result summary

Compilation time	~300s
Training RMSE score	~0.2607
Testing RMSE score	~0.347

We have built another network which adds some more dropout layers in between LSTM layers to avoid the issue of overfitting in training stage. With the same number of epochs being 10, we have a set of predicted values better aligned with the ground truth data.

Improved model result summary

Compilation time	~350s
Training RMSE score	~0.0318
Testing RMSE score	~0.0397

After the model is initiated with a set of weights, we use the model to predict stock value of the next day and calculate the percentage difference between the last day and the next day. Positive means there shows an increase in closing value within that two days and the negatives means the opposite.

B - Volatility Calculations and Comparisons

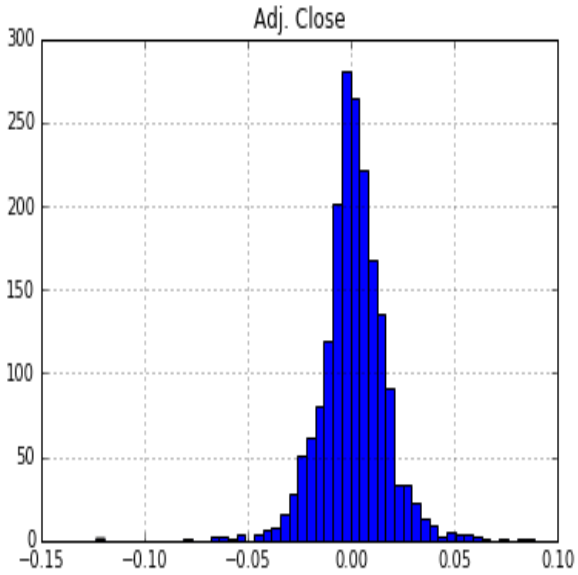
As we have an idea of our data, what time series data is about and how we can use pandas to quickly explore your data, we want to dive deeper into some of the common financial analyses relating to stock returns and volatility.

Returns

The simple **daily percentage change** doesn't take into account dividends and other factors and represents the amount of percentage change in the value of a stock over a single day of trading. We then summarize those values and plot the distribution of daily percentage change with regards to an individual stock.

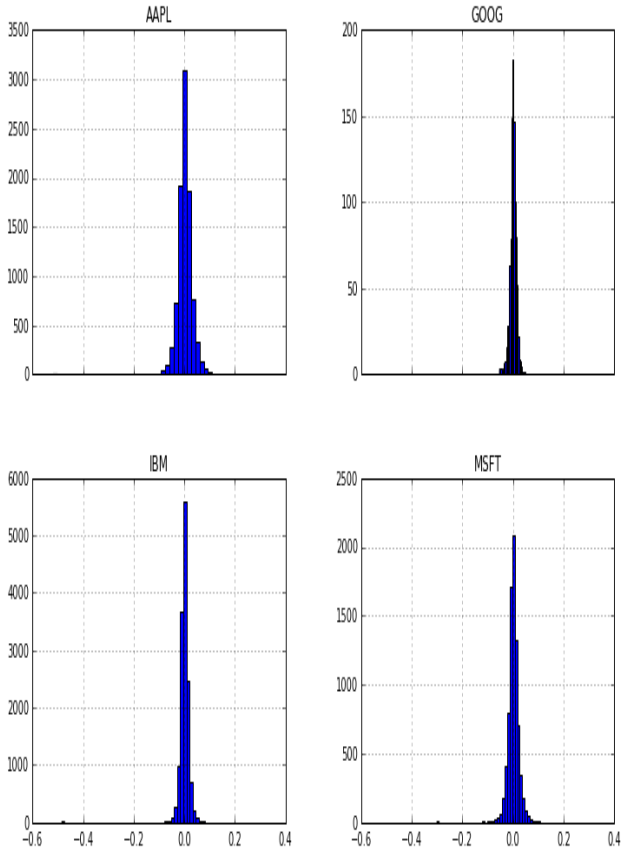
Summary

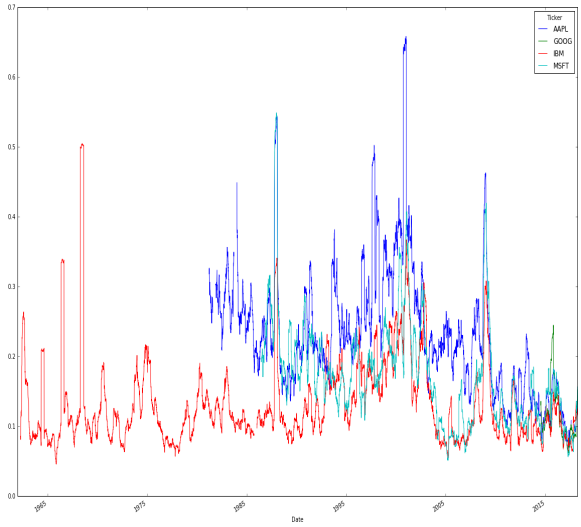
count	4000.000000
mean	0.000940
std	0.015737
min	-0.123549
25%	-0.006818
50%	0.000547
75%	0.009411
max	0.088741



It is beneficial to be able to interpret the result of the histogram. As the mean is very close to the 0.00 bin also and that the standard deviation is 0.02, the daily percentage change is shown to be pretty stable and normal.

We often see that these numbers don't really say much when we don't compare them to other stock. That's why we often see examples where two or more stocks are compared. So we decided to get more data from Yahoo! Finance so that you can calculate the daily percentage change and compare the results. As we get the stock data from Apple, Microsoft, IBM, and Google, here is the four sub-plots that we have to better visualize how the overall market performs in terms of daily percentage change.





Generally the main takeaway from the graph is the more zigzagging the line gets, the higher the volatility, the riskier the investment in that stock, which results in investing in one over another.

Volatility Calculation

The volatility of a stock is a measurement of the change in variance in the returns of a stock over a specific period of time. It is common to compare the volatility of a stock with another stock to get a feel for which may have less risk or to a market index to examine the stock's volatility in the overall market.

10 - Applications of Knowledge

Buying and selling or trading is essential when we are talking about stocks, but certainly not limited to it: trading is the act of buying or selling an asset, which could be financial security, like stock, a bond or a tangible product, such as gold or oil.

To achieve a profitable return, investors who seek financial security would like to base on a decent algorithmic model to decide whether to go long or short in stock markets: Either thinking that the stock price will go up to sell at a higher price in the future, or you sell your stock, expecting you can buy it back at a lower price and realize a profit.

Moreover, developing such a trading strategy is something that needs going through a couple of phases. Right now with Stockalyzer, users can have a better view at the targeted stock market where they want to compare and visualize the stock volatilities in a specific time frame. This way, they can get a starting point of how to analyze, optimize and improve their approach before applying it to real markets.