Stock Prices Forecasting Using Machine Learning

Anant Bhasin   
 *Department of Computer Science and Engineering*  
*Shiv Nadar University* Greater Noida, India  
ab342@snu.edu.in

Sisir Nalla   
 *Department of Computer Science and Engineering*  
*Shiv Nadar University* Greater Noida, India  
sn178@snu.edu.inRishika T D  
 *Department of Computer Science and Engineering*  
*Shiv Nadar University* Greater Noida, India  
rt347@snu.edu.in

Sparsh Ailawadi  
 *Department of Computer Science and Engineering*  
*Shiv Nadar University* Greater Noida, India  
sa342@snu.edu.in

***Abstract*— Stock market forecasting is a challenging research activity and in this paper, we perform technical analysis on historical stock market data to predict future prices using machine learning and deep learning techniques. We aim to perform and compare different machine learning algorithms and techniques and their accuracy in successfully predicting stock prices. We use the simple moving averages technique, kNN regression and finally LSTM RNNs to analyse the stock time series data by implementing them in Python code and discussing them theoretically.**

***Keywords—stock market prediction, LSTM, RNN, k-NN, moving averages***

# Introduction

Stock Market Prediction is the process of trying to predict the future value of a company stock or any other financial instruments traded on an exchange, based on their current information and trends. Such predictions could yield significant profits when successful and over time, many analysis techniques have been used to create stock market forecasts. This forecasting has posed as a challenging research activity that is now expanding with the availability of new data sources, markets, financial instruments, and algorithms. There is a lot of research being conducted and data mining done on stock market data in order to predict stock market prices with higher accuracies.

Stock Market analysis methodologies fall broadly into two categories — Fundamental Analysis and Technical Analysis.

* Fundamental Analysis consists of examining a stock’s economic and financial factors in order to measure a security’s intrinsic value. It analyses the company’s future profitability on the basis of macroeconomic factors such as its current business environment and financial performance, the state of the economy etc.
* Technical analysis consists of using charts and statistical figures to predict trends and future market behaviour.

Our focus will be on technical analysis. Technical analysis of stocks and trends have been performed for hundreds of years since the 17th century. In the modern day world, however, technical analysis owes heavily to Charles Dow, William P. Hamilton, Robert Rhea, Edson Gould and many others who proposed revolutionary theories to perform technical analysis on stock trends. They reflected a new point of view that the trends in a market can be measured in chart patterns of highs and lows and statistical indicators. The fundamental principle that technical analysis is based on is that all available information that may have an effect on the market is reflected in the market price. As a consequence, as they are already priced into a given security, there is no need to look at economic, fundamental or other developments.

In this paper, we attempt to perform technical analysis and predict stock market prices by employing machine learning and deep learning techniques. We will be using the moving averages method, k-nearest neighbours regression algorithm, and Long Short Term Memory RNNs for the predictions, analyze and compare the results in order to achieve higher accuracies in stock forecasting.

# Literature Review

As previously stated, stock market forecasting is a hot research area with high value. The stock market, with its large and dynamic information sources, also serves as a suitable environment for researchers in data mining and business. Stock forecasting becomes increasingly significant, particularly if certain rules could be established, helping investors make better decisions.

Moving average is a commonly used algorithm for technical analysis. In the study done by Lauren and Harlili [1], the simple moving averages technique was used to predict stock trends and compared with simple moving averages supported by news classification. Coming to kNNs, [2] the study done by the Alkhatib and his co-authors in stock price prediction using the k-NN algorithm on the stock prices of a sample of six major companies listed in the Jordanian stock exchange showed that the kNN algorithm was stable and robust with small error ratio, so the results were rational and reasonable. Lastly, when we look at the study done by Chen, Zhou and Dai for stock returns predictions in the Chinese stock market using the LSTM method, it proved to give significant accuracy compared to other machine learning techniques [3].

# Data Description

Time-series analysis is an extremely elementary concept that lets one find meaningful and relevant insights from data collected over time. Since stock prices vary with set intervals of time and the trend relies heavily on a specific feature of closing price, that varies with stamped time - time series analysis is a great tool that can be leveraged for stock market predictions.

The data being used for analysis has been taken from Quandl and represents the stock prices for Tata Global Beverages from 8th October 2013 to 8th October 2018. The data represents a series of data points indexed by time or a time series. The 5 years data is divided into train and validation sets for each of the implementations. The data is complete and as such no features are missing or inaccurate. The data represents typical stock market descriptors -

OPEN: Represents the price at which trading starts for a particular day

CLOSE: Represents the price at the end of a trading day

HIGH: The maximum price of a stock on a particular trading day

LOW: The minimum price of a stock on a trading day

LAST: The last traded price of a stock for a day

TOTAL TRADE QUANTITY: Represents the trade volume and is the total number of stocks of a company bought or sold on a day

TURNOVER: The turnover of the company on the particular day represented in lacs.

It should be kept in mind that stock trading floors are closed on weekends and national holidays and the same is reflected in the dataset as well since some dates are missing from the same.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Date | Open | High | Low | Last | Close | Total Trade Quantity | Turnover (Lacs) |
| 08/10/18 | 208 | 222.25 | 206.85 | 216 | 215.15 | 4642146 | 10062.83 |
| 05/10/18 | 217 | 218.6 | 205.9 | 210.25 | 209.2 | 3519515 | 7407.06 |
| 04/10/18 | 223.5 | 227.8 | 216.15 | 217.25 | 218.2 | 1728786 | 3815.79 |
| 03/10/18 | 230 | 237.5 | 225.75 | 226.45 | 227.6 | 1708590 | 3960.27 |

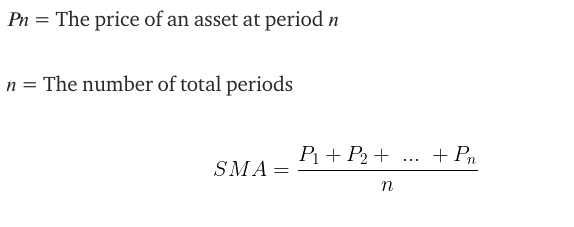
For a stock the profit/loss made is specified by the closing price of the same. Closing price hence is the factor that will be used for prediction purposes and will as such act as the target variable. We will be predicting the daily closing price for the stock of Tata Global Beverages. Total trade quantity is also a very important metric for a decrease in the same along with an increase in price shows a declining interest in a stock while a sizable price change on a large volume is also an indicator of a major change in the company’s outlook.

# Experimentation

* 1. *Moving Averages*

Moving average is a very common approach for time series analysis. Moving averages help eliminate a sense of randomness from older values by considering a specific range within the set timeframe. To make it clear, the moving average for a point in time k is obtained by averaging values of the time series within m periods of k. Therefore in order to calculate the moving average the data points further away in the past have lesser relevance as the moving average progresses through time.

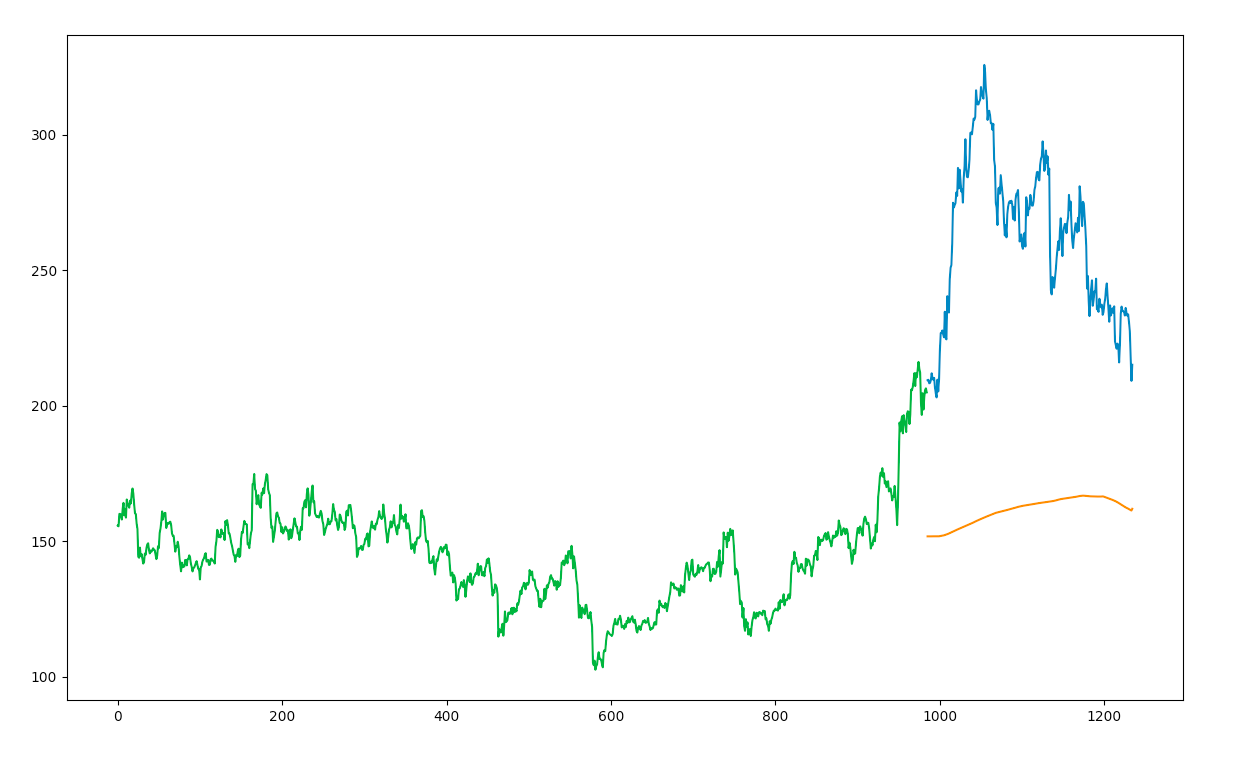
Mathematically, one can define the moving average as,



While moving average is easy to implement and extremely straightforward to understand, it comes with its fair share of issues that make it not the best jois for such a time series analysis, these drawbacks can be elaborated as,

1. The method operates under the assumption that trends are linear, which is not the case.
2. Trends values aren’t computed for all terms.
3. Simple moving averages can’t be used to forecast future trends, defeating the point of time series analysis.

Therefore, even though simple moving averages gives a very simple approach it is not the best choice for such an application. Much more sophisticated and accurate methods exist and the same will be explored further in the paper.



Prediction as done by moving averages method

Upon computation the approach gave a RMSE value of 104.5141. Clearly the inaccuracy is high and a better approach must be studied.

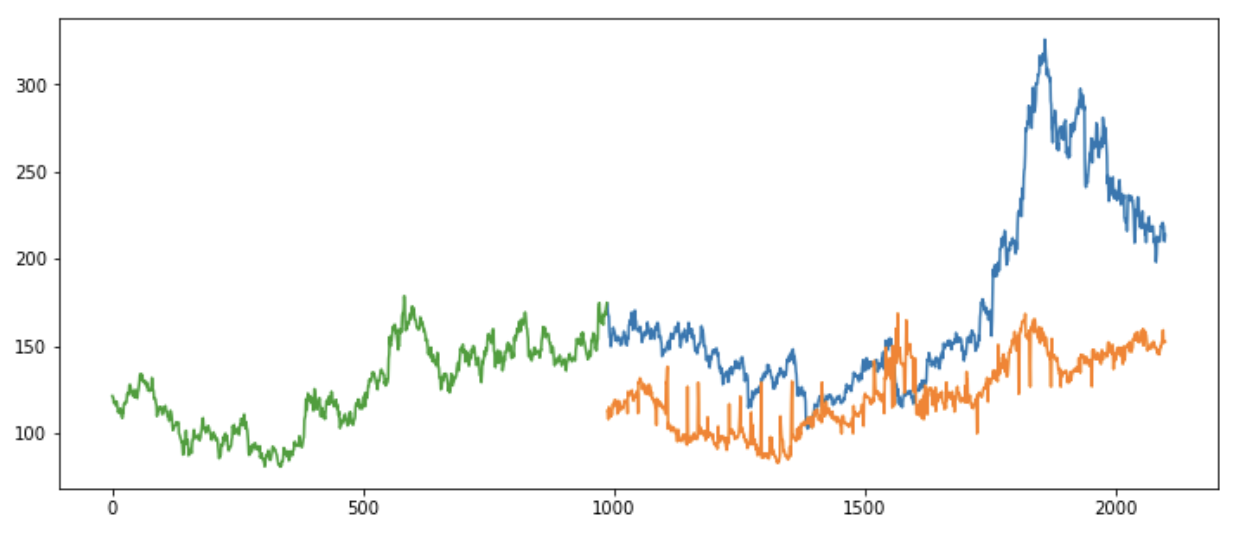
## k-Nearest Neighbours Algorithm

The k-nearest neighbour algorithm is a simple non-parametric method used for classification and regression. It belongs to the family of instance-based, competitive learning and lazy learning algorithms. When we need a prediction for a particular unseen test data value, the k-NN algorithm searches through the training dataset for the k most similar instances. A majority vote is then carried out among the prediction attributes of these k selected records and returned as the prediction for the unseen instance. Euclidean distance is one of the commonly used similarity metrics to take a decision for the data. In case of classification, the most prevalent class is returned, But in case of k-NN regression, the average of the prediction attributes of the k nearest neighbours are returned.

The kNN is powerful since it does not assume anything about the data except the fact that the distance measure can be consistently calculated between any 2 instances. Hence, we see that it does not assume a functional form and as a result, it is called non-linear.

It is possible to map a stock prediction problem into a similarity based classification. The test data and the previous years’ stock price data is mapped into a set of vectors where each vector represents N dimension for each stock feature. The closing price acts as the target variable.

In the Python code implementation for predicting stock prices using the k-NN algorithm, we make use of the sklearn.neighbors.KNeighborsRegressor() function that from the sklearn library that performs regression based on the k-nearest neighbours.The target is predicted by local interpolation of the targets associated of the nearest neighbors in the training set.



Prediction as done by kNN regression

The results of the predicted stock price for each individual company used in the sample with graphs for the actual and predicted prices are presented. The yellow lines in the graph (shown above) are the predicted values and the blue are the actual values of the stock price. From the graph itself we can tell that the k-NN does not predict the stock prices with the desired accuracy.

Moreover, we can see the predicted values have an RMSE value of 66.49. It has performed better than the simple moving averages technique but it is still not significant. We can also say from the negative R squared value that the k-NN model is not a good fit for predicting stock data.

# Our Approach

## Recurrent Neural Network

RNN are used in applications where the mean and variation keep changing as more data is collected. It sees applications in natural language processing, time sharing forecasting like stocks etc.

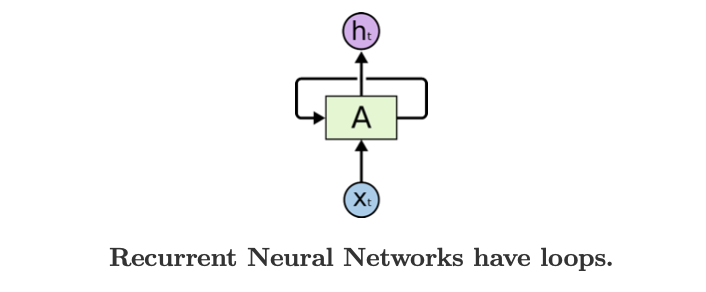
A simple example to understand the working of RNN would be a spam detector. When given a sentence the software has to detect whether it’s a spam subject or not. NLP is often used here along with algorithms like BOW (bag-of-words), TF-IDF or WORD2VEC, these algorithms only consider if the words in the sentence are +ve or -ve and based on that make their prediction.

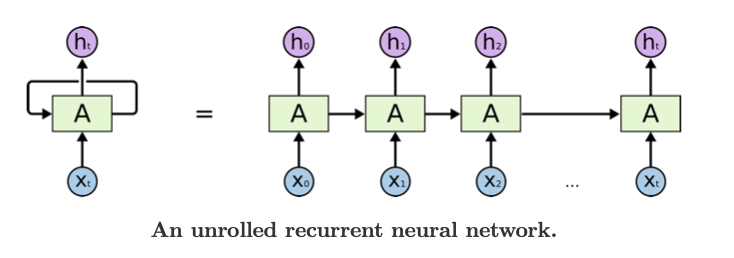
None of these algorithms consider the sequence of those words i..e the meaning of the sentence in making their predictions. This is very important in applications as well like speech detection and intent parsing in natural language processing applications like implementing voice assistants where not just the words but the sequence also carries a lot of importance.

This is what an RNN does, RNN passes the data from the previous layer to the next and the next layer may decide to retain/update/add new data to the incoming data from the previous layer.

With a system like this the flow of words in a sentence is taken into account while making a prediction.

Below is the structure of an RNN



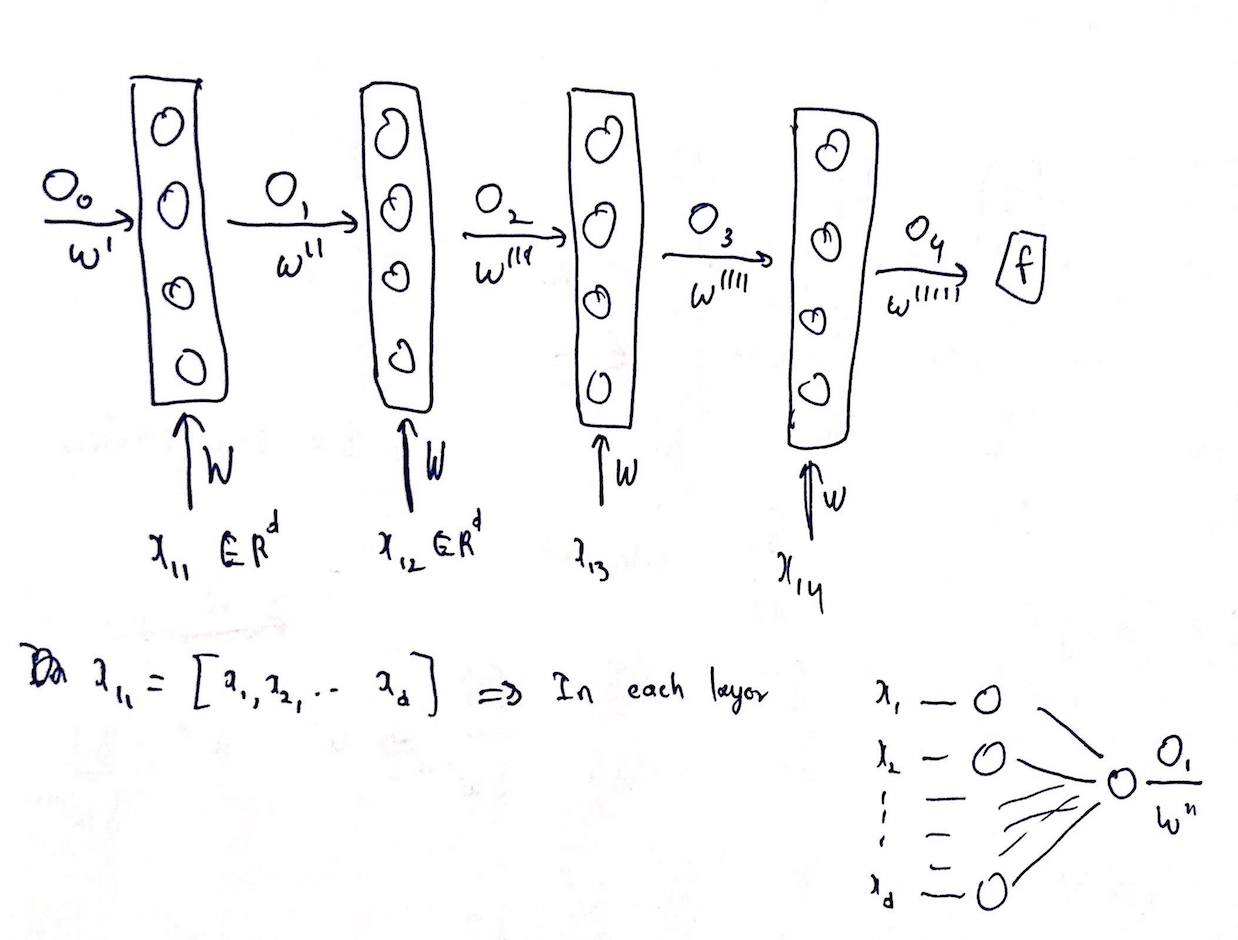
A is the hidden layer with a number of neurons inside. The input is **Xt** which is input at some time t and the output is **ht.** The output at time t is fed into the loop along with the new input at time **t+1** which gives an output **ht+1.** The outputs are continuously fed into the loop. The representative of the structure made in layers given below-

**Forward propagation**

The use case considered here to describe this technique is word processing.

Here the set of words are represented by **x**11,**x**12,**x**13,**x**14. The word is converted into a vector each of which is fed into the RNN in a timely manner. Let’s consider the word to be a d-dimensional vector. Each word is multiplied by the same weight vector and a bias is added to the result. An activation function is then applied to this result and we get out first output **O**1. **O**1 can be represented as -

The activation function - **sigmoid**

****

Here **O**o is the initial output that’s passed along with the weight vector **W’**. They are either zero padded values or randomly initialised.The output **O**1 is multiplied by the weight **W**’’ and is fed into the next layer (technically, the second loop). All this is done at time **t=1.**

At **t=2** the next word is processed. Vector **x12**  is multiplied by weights **W** and added to **O1 \* W’’ .** The activation function is applied to this result to produce output **O2.** The resulting equation is given by -

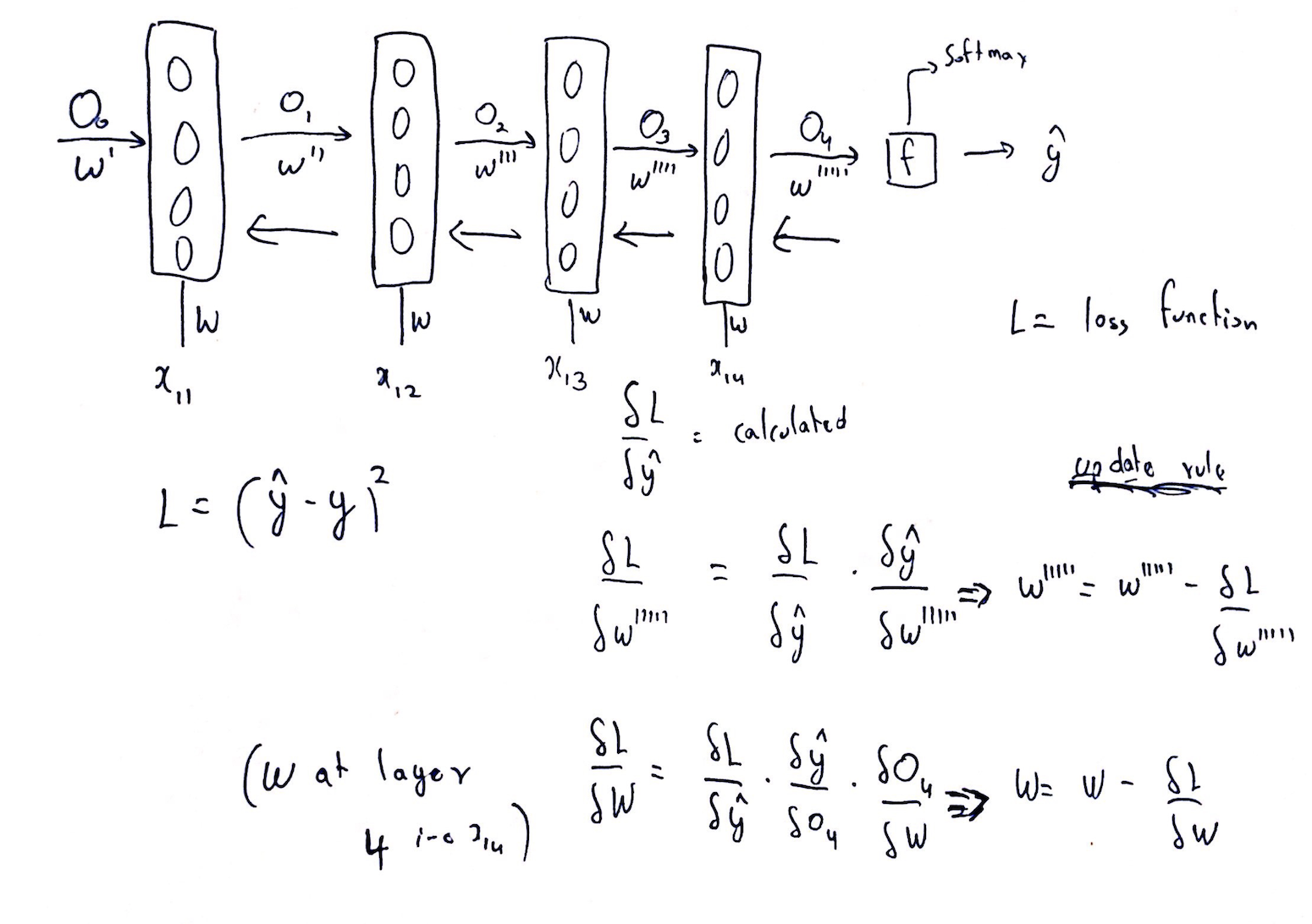
Similarly,

To the output **O4** a softmax function is applied to get a binary output between 0 and 1.

In this network the outputs of each layer are retained and sent to the next layer to get the next output. Hence, **O4** is dependent on **O3** & **x14**, **O3** is dependent on **O2** & **x13**, **O2** is dependent on **O1** & **x12** and finally **O1** is dependent on **Oo** & **x11**. Therefore, this dependency allows information to be passed on.

**Backward Propagation**

After getting the predicted value, a loss function is used ( in our case **mean squared error (MSE)** has been used) to calculate the loss and back propagation is started and keeps going till we have made the loss as minimum as possible.The weights are modified according the derivative of the loss function with respect to that weight. The derivatives are calculated using the chain rule.



All the weights are updated accordingly and once again forward propagation is started.

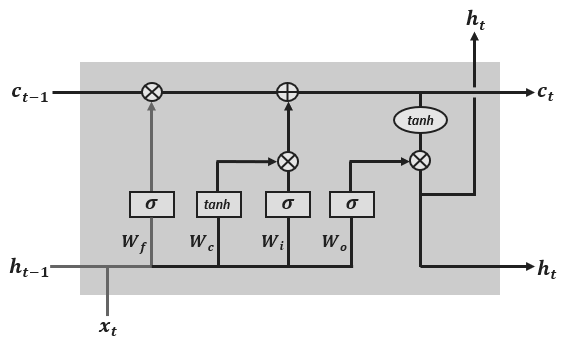
## Problem with RNN: Vanishing Gradient

This simplistic approach is useful in a lot of places but not for stock price forecasting, the main flaw being the RNN networks run into the problem of vanishing gradient. As we move from one loop to another the derivative of the loss function becomes very small and after a point the weight vectors become stagnant. The reason for this is the fact that we are using sigmod as our activation function and we use the same weight vector W for each input **x1i**. The derivative of sigmoid lies between 0 and 0.25 and this when multiplied by another number makes the result even smaller, hence the vanishing gradient problem.

These problems are solved by modifying the RNN to make what is called a Long Short Term Memory model. In this, not just the last but a number of elements before the element being processed contribute directly to the current output.

## Long Short Term Memory

An LSTM cell is shown below:



The LSTM has 4 components

* Memory cell
* Forget gate
* Input gate
* Output gate

**Memory cell**

The purpose of this cell is to remember/forget data from the previous cell. Remembering/forgetting is done based on the context of the input. Below is the memory cell



There are 2 operations that can be performed in this cell, pointwise multiplication and addition. For example, let’s assume that the previous cell’s output is a vector [1 2 3 4 5 ] and we get a vector [1 1 1 0 0 ] from the forget gate. Upon pointwise multiplication we get the vector [ 1 2 3 0 0 ] . Here the last 2 elements of the vector are 0 which implies that in this cell, the data corresponding to the 4th and 5th element of the array has been forgotten.

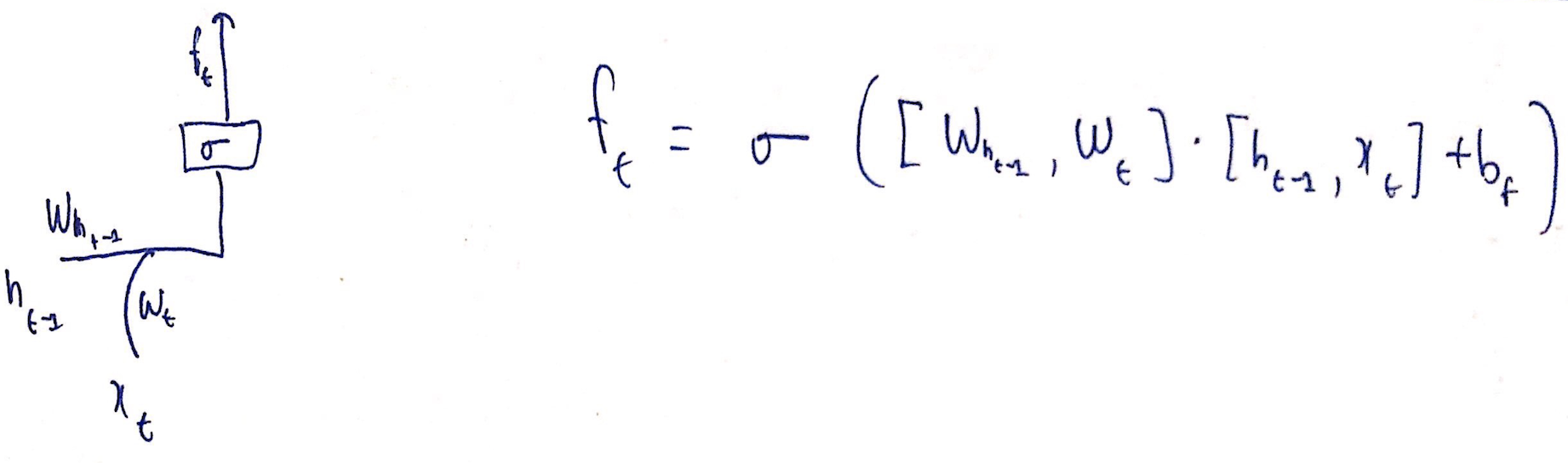
So if the context of the input (**xt**) is same as the incoming cell state (**Ct-1**), a vector of all 1’s would be passed from the forget gate and no data will be forgotten and if the context of the input (**xt**) is completely different from the previous cell, then a vector of all 0’s would be passed and the data would be forgotten.

At the +, new information is added or existing information is updated based on what we get from the input gate.

For example, in stocks, if the next sequence of stock prices are starkly different from the ones before it, then slowly in the consecutive layers the information regarding these prices will be forgotten and prediction will be made without a lot of influence by these previous values.

**Forget Gate**

The forget gate generates a number between 0 and 1 for every number in the cell state **Ct-1.** 0 implies, completely forget it and 1 implies keep this.

****

Weights are initialized on either side of the input of the cell and output of the previous cell. The number between 0 and 1’s are generated by the sigmoid function.

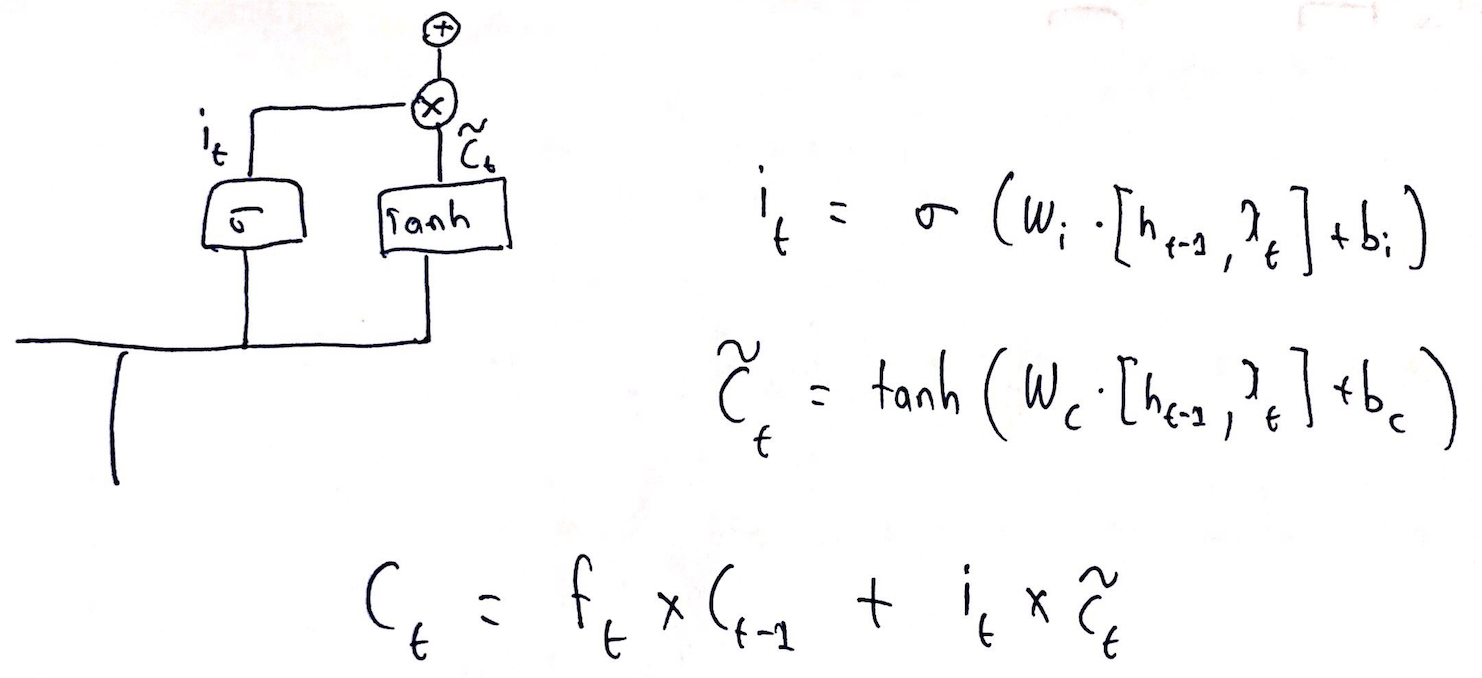
Example - Predicting next word based on the previous word. The previous cell state can include gender so that correct pronouns can be used. If the incoming input is a new subject, we’d want to forget the gender of the old subject and replace the old gender with the new gender.

**Input Layer**

This is where we update/add new data to the cell output **Ct**

The sigmoid decides which values could be updated. It produces an array of 0’s and 1’s.

The tanh function creates a new set of candidate values Ct (~). A pointwise addition operation is done and added to the cell state

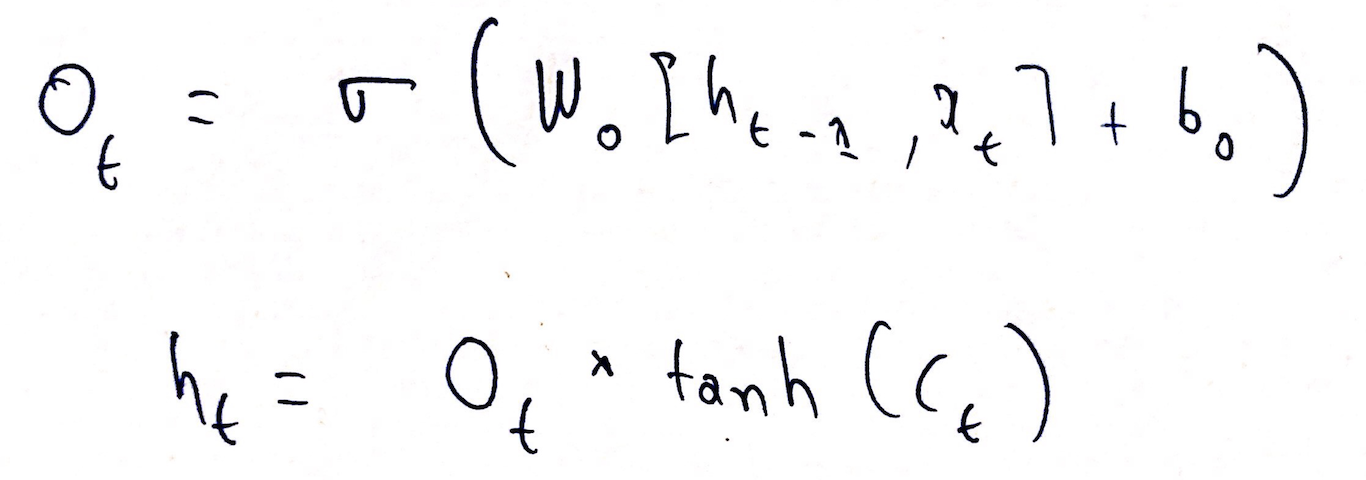
****

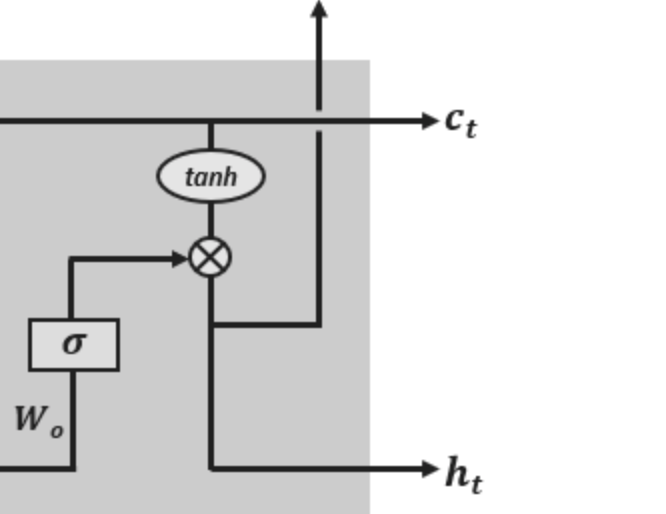
Example - this is where we omit the old gender and add the new gender to the data.

**Output layer**

Here we produce the output of the cell. The output is the cell state after being updated by the input layer and is filtered before being sent to the next cell.

We run a sigmoid to decide what part of the cell state we are going to output. The cell state is put through the tanh function to squash the value between -1 and +1 and is multiplied by the sigmoid to get what we want.

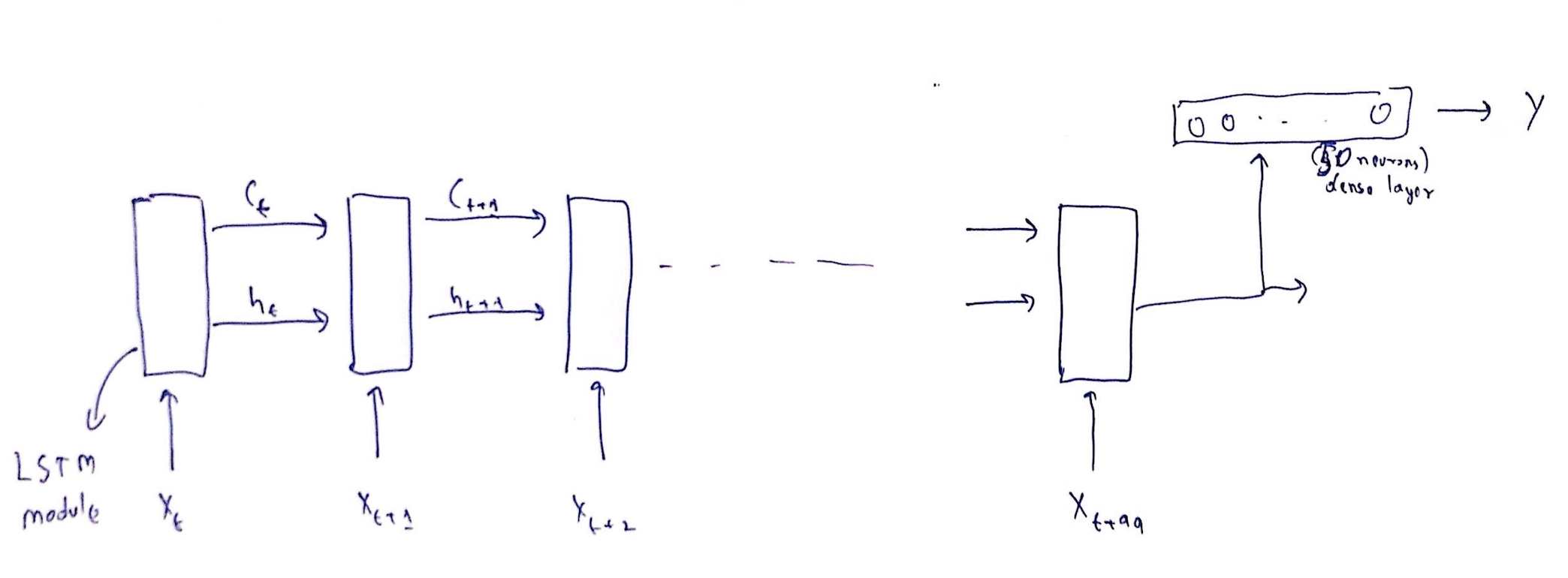




**Note -** If **xt** and **xt-1**are almost the same data points in terms of context then the new cell state will be the same as the previous cell state.

**Note -** As **Ct** is a function of **tanh**, the gradients are distributed between -1 and 1 and hence the problem of vanishing gradients is prevented.

## The working of LSTM cells

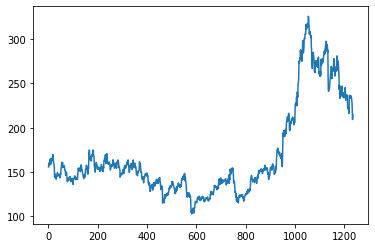
The Xt, Xt+1, .. Xt+99 is where the 100 slots where 100 data points go in and the predicted value for the data points is Y. The output of the last LSTM cell is squashed using a 50 neuron dense layer. The cell state (Ct +99) and ht+99 are passed into the 50 neuron dense network and then we get the predicted output. The loss function is then calculated and backprop is started . 

# Our Implementation And Results

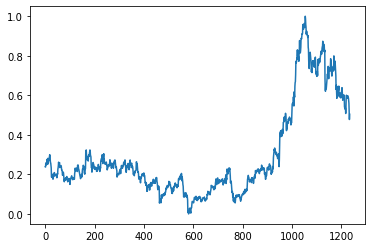
The implementation we are using for the stock price forecasting is a single LTSM layer neural network, implemented using the tensorflow/keras libraries in Python 3.

To begin, we are using the dataset containing the stock prices of the NSE listing TATAGLOBAL from 8th October 2013 to 8th October 2018 from a csv file attached to the submission.

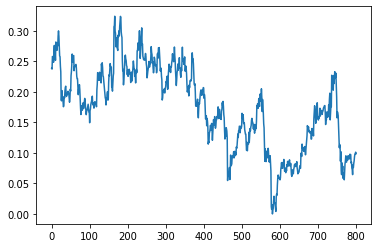
The dataset contains the opening and closing prices, the last trade price, the highest price and the lowest price for the day for TATAGLOBAL. We sequentially divided our dataset into a split of 65% and 35% for the training and the testing datasets, and trained our model on the closing price for everyday on the training dataset.



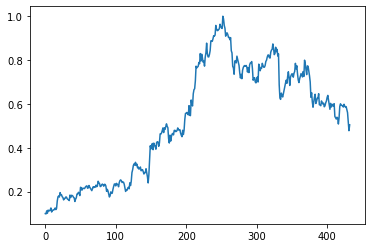
The initial data looked like the above. For the LSTM model to work efficiently with the data, we used a min-max scaler that linearly scaled the dataset to values between 0 and 1. Since all the values were linearly scaled with respect to the minimum and the maximum values of the dataset, the shape and hence proportional variations of the dataset were preserved and became optimised for the training of the LSTM model to begin. We had the below graph present after the scaling of the values:



The shapes of the training and the testing data parts are shown plotted below:



Training data; corresponds to the first 65% of the closing prices for the dataset

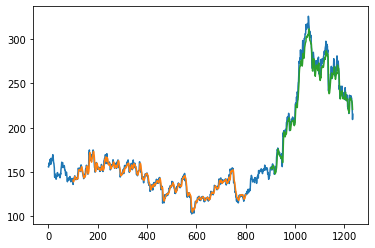


Testing data; corresponds to the last 35% of the closing prices for the dataset

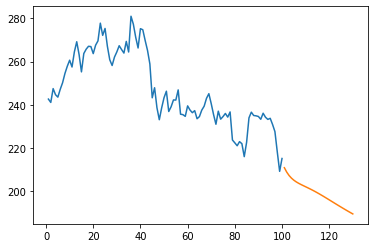
The single layer LSTM model we are using has 10,451 parameters, and then a dense layer for the outputting. The parameters passed were the number of neurons for the model layer and the input shape which is how tensorflow takes the input for the predefined LSTM model. The model was trained on the data for 100 epochs and the loss was reduced from 0.065 to 0.000198.

The root mean squared error calculated is 3.08 for the training data and 6.35 for the testing data. Hence, with the testing data, we’re actually getting an accuracy of 97.2% accuracy.

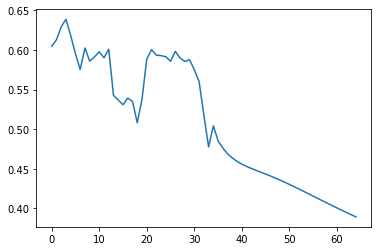
After training the model, we plotted the predictions for the testing data first and with the 97% accuracy model, we find it to be a plotted graph that almost coincides with the testing data itself. A plot showing the dataset, the training and testing predictions highlighted is shown below.



After this, we also predict stock prices for the next 10 days following this time period. This is done by predicting for every consecutive day, adding that prediction to the dataset itself and then reshaping the model, and then predicting the consecutive values. Those graphs are shown below too:



The orange line is the line with the predicted stock prices for the 10 days following our dataset. Blue line plot is the last stretch of the original dataset.



This is the whole dataset with the predictions appended to it.

# Conclusion

The conclusion we come to is that the KNN and the moving averages techniques are insufficient for this use case but using a modified RNN model such as the LSTM which is a time-dependent model and takes sequences into account is an apt solution to the problem statement.

##### Acknowledgment

We would like to extend our heartfelt gratitude to Dr. Snehasis Mukherjee for guiding us throughout the course of this project. His constant support, encouragement and insights have helped us approach this project from a holistic perspective. We would also like to thank the teaching assistant for the course Jaydeep Kishore for his insightful guidance.

##### References

1. K.Alkhatib, H.Najadat, I.Hmeidi and M.K.A.hatnawi, “Stock Price Prediction Using K-Nearest Neighbor (kNN) Algorithm” International Journal of Business, Humanities and Technology, vol. 3, pp. 32–44, March 2013.
2. S. Lauren and S. D. Harlili, "Stock trend prediction using simple moving average supported by news classification," *2014 International Conference of Advanced Informatics: Concept, Theory and Application (ICAICTA)*, Bandung, 2014, pp. 135-139, doi: 10.1109/ICAICTA.2014.7005929.
3. K. Chen, Y. Zhou and F. Dai, "A LSTM-based method for stock returns prediction: A case study of China stock market," 2015 IEEE International Conference on Big Data (Big Data), Santa Clara, CA, 2015, pp. 2823-2824, doi: 10.1109/BigData.2015.7364089.
4. Fundamental Analysis. (n.d.). Retrieved December 7, 2020, from https://www.investopedia.com/terms/f/fundamentalanalysis.asp
5. Wikipedia contributors. (n.d.). Stock market prediction. Retrieved December7,2020,https://en.wikipedia.org/wiki/Stock\_market\_prediction#References
6. Singh, Aishwarya. “Stock Price Prediction Using Machine Learning: Deep Learning.” Analytics Vidhya, 18 Oct. 2020, www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/.
7. https://colah.github.io/posts/2015-08-Understanding-LSTMs/
8. https://www.analyticsvidhya.com/blog/2018/10/introduction-neural-networks-deep-learning/