Stock Price Prediction with Long Short Term Memory Neural Networks

By: Sachin Khoja (18NA30015)

Abstract

Stock price prediction is a really interesting avenue in the current world of financial markets. It has now been attracting many data science and machine learning graduates. On one hand ,we have proponents of efficient market hypothesis who claim that stock prices can not be predicted as stock prices are quite volatile and unpredictable which means that there are no consistent patterns in data that allows us to model stock prices over time near-perfectly. However, researchers have worked on bringing a technique or hybrid modelling so as to identify patterns in stock price trends. Successful prediction model for stock market price prediction can be really profitable for traders. But this model to be designed is affected by various other factors apart from the feature vectors which are taken sometimes in an unexpected manner. The complexity is inflated by the economic progress and the government policies as well. The dynamics of stock volatility adds to the complexity of the problem. But now with the advancing technologies, professionals are turning to data scientists and machine learning analysts for guidance towards market dynamics.

In this study, we use a type of Recurrent Neural Network (RNN) that is Long Short Term Memory (LSTM) neural networks to predict the stock price with the help of previous data which is made available by kaggle. The Data set is then analysed and normalised and then the model is created and fine-tuned through the hyper parameters and other parameters.
Then the model is trained and tested to give desired results to predict the VWAP(Volume Weighted Average Price) of the stock.

Introduction

Here we target to build a model which predicts stock market prices such that a buyer can correctly Know when to buy or when to sell the stocks. so we build a time series model which predicts this. Stock

Market dataset file includes various terms. We are giving a brief introduction to the various terms related to that. PREV CLOSE means the closing price of a stock the previous day which means it is the price at which the stock of that particular type was exchanged just before the closure of the stock exchange market. OPEN means the price at which the stock was traded for the first time after opening of a stock exchange market on the day. HIGH means the highest price at which the stock was traded on that particular day whereas LOW means lowest price at which the stock was traded on that particular day. VOLUME means the total amount of shares which were traded on that day by adding them for every transaction which occurred that day. TURNOVER Is the total money transaction which is reflected at the end of day through the stock transactions i.e. VWAP multiplied by volume. VWAP is a very interesting feature since it is the average price at which stocks were traded during the whole day. This average is taken with weight as volume i.e total number of shares traded in that day. In this project, we have trained the model to predict the VWAP.

Long short term memory (LSTM) is a type of recurrent neural network used for time series prediction which was first introduced in 1997. It Can predict an arbitrary number of steps in the future. LSTMs are often used for Problems in language processing. As it utilizes the feedback connections so it can auto-correlate the words in the form of time series the words are occurring. So , it has been used traditionally for tasks such as handwriting recognition ,speech recognition etc.

The Idea behind applying LSTMs in the prediction of the stock market is that LSTMs are uniquely suited to handle time series with varying time periods between events of significance as they are architectured to extract previous information in memory through the feedback networks which is a characteristic feature of LSTM.

LSTM Module has five essential components which as its name suggests extracts features from both long-term as well as short term data. It consists of cell state, hidden state, input
Gate, forget gate and output gate. Cell State represents the internal memory of a cell which stores both short term and long term memories. Hidden state contains output state information with respect to current input, previous Hidden State and Current cell input which is used for predicting future stock market prices. However , the best part is according to requirements, hidden states can decide to only employ short or long or both types of memory stored in the cell state to make predictions. Input gate decides how much information from current Input and previous cell state flows into the current sale state. Output gate describes the information which flows from current cell state into hidden state so

that if needed our LSTM algorithm can only extract the long-term or short-term memories or both. LSTMs proficiency to handle the time series with inconsistent timings between desired events make it a nice algorithm model for predicting financial assets.

Contribution

My contribution in this project is regarding designing the hyperparameters for the networks and selecting the feature vector and doing the further analysis to create the model. The testing , training and error part has been done by my colleague.

The steps which I have taken to design the model are mentioned here after. we start with reading and analysing the data .Then we normalise the data for our further analysis. We then do the data augmentation and convert it to time series and supervised learning problems. Thereafter we create the model with appropriate layers and finetune it. We import the following libraries such as Pandas, keras, SR learn metric And preprocessing and Math.

Our primary data source is from kaggle and it provides daily data for a span of 18 years.
We have analysed the stock market data of Maruti corporation and used it for our analysis. Stock prices come in different flavours and there are various entries such as open,closed , high , low , volume and vwap. However we have taken a special feature VWAP as our target vector as it signifies all other variables related to the stock market as it is the average price at which stocks are traded for a day. Here we Sort the data by date so that it is in increasing time series. Then we visualise the target variable VWAP which gives us a fair idea of the change in stock price during the entire period. Thereafter splitting of data is done into the training set and testing sets . We have followed nearly a 80:20 percent ratio of training and testing data sets. We have taken the data till 2017 to be part of the training set and thereafter we have taken it to be a testing set. The .csv file containing these data sets was taken from https://www.kaggle.com/rohanrao/nifty50-stock-market-data.

VWAP Training & Testing Data

The data now has to be normalised as without normalisation it can be used for further analysis as it varies quite in the range especially the volume criteria . We have used the Min-Max scaler from sci-kit learn which subtracts a minimum of data set from all values there by marking a scale from min to Max which is then divided by Max- Min of the data set to scale the value between 0 to 1. Also , we convert the data frame to ndarray in the process.

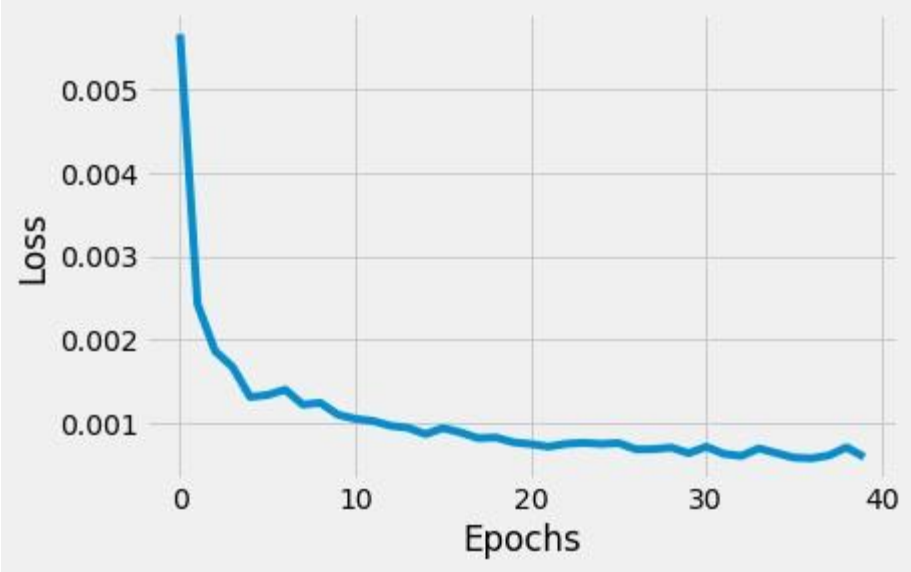
For converting the data to time-series and supervised learning problems , we have chosen the following hyperparameters to be decided. Since LSTMs have a long term memory state , we create a data structure with 80 timesteps and 1 output. So for each element of the training set we have 80 previous Training sets of elements. Now While testing these sets we have used the testing inputs and the previous 80 data sets as well as to predict for the first training data set . So, we have chosen our time step to be 80 units. So, the network trains according to the stock prices of the previous 80 Days. We define the batch size as 30 and we have used this batch size to train our model. So, we are updating weights after every batch size. As batch size is an important hyperparameter as it decides the speed of training and on the other hand model's ability to generate different data. As using small batch size reduces the speed of training and on the other hand using too big batch size reduces the model's ability to generate different data. So, we have decided the batch size by taking consideration of both the factors by finding the sweet spot which itself is a topic of research.

Next We decide about the model which we are going to use. We have implemented the sequential model of LSTM which consists of three different hidden layers, the input layer and output dense layer network. Sequential model is appropriate for a plain stack of layers where each layer has exactly one Input tensor and one output tensor. Since we don't have multiple input/outputs and we don't need to layer sharing nor we want to have non-linear topology , we can use the sequential model. The number of neurons in each layer from input to output side is decreasing as we don't want the nature of our LSTM network to be very computationally complex. We have chosen our output dense network of one unit which is then processed by the activation function. We have chosen our first input layer to be of 100 units ,second layer with 80 units , third layer with 60 units and fourth layer with 50 units and finally the output layer having one unit. We have chosen linear activation functions since the model which we have utilised after scaling has quite even properties and we have to show dependencies of previous days stock price efficiently. Another factor which is important is the dropout factor which we have chosen accordingly. Dropout can be interpreted as a regularization model where Input and recurrent connections to LSTM units are probabilistically excluded from weight and activation updates while training a network. The Dropout factor helps us to reduce the overfitting of data and thus improves our model performance. As We want to retain various parts of the data set while predicting and dropout Some part probabilistically. As preferred in various research papers we have taken our dropout rate to be 20% i.e. it sets 20% of inputs to zero and rest are carried forward.

Another aspect which is important is the Optimizer we are using to build upon gradient Descent to combat the problem of pathological curvature and speed up search at the same learning step is adapted accordingly. Out of popular optimizers, we may find Momentum to be most prevalent and Adam looking most promising on paper. However 5GD with Momentum seems to find more flatter minima than Adam while adaptive methods tend to converge quickly to sharper minima. Flatter Minima generalizes better than sharper ones.
We tried using RMS prop Optimizer Here In this project as we are focusing Finally on making the root mean square error minimum as possible and train the data accordingly. The default value of learning rates for RMS prop is 0.001 according to official documentation. The root mean square came out to be 299.358751619 .

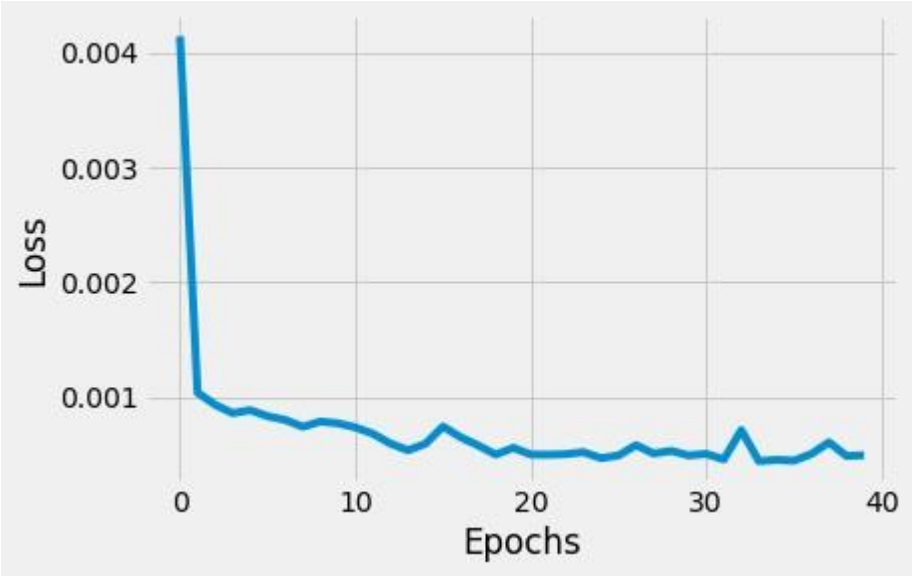
Next we tried using Adam’s optimizer. Adam’s Optimisation is a stochastic gradient

Descent method based on adaptive estimation of first order and second order moments. The default value of learning rate is 0.001 as in documentation. The root mean square error while using Adam's Optimizer came to be 262.75854.



Loss vs Epoch (RMSProp Optimiser)

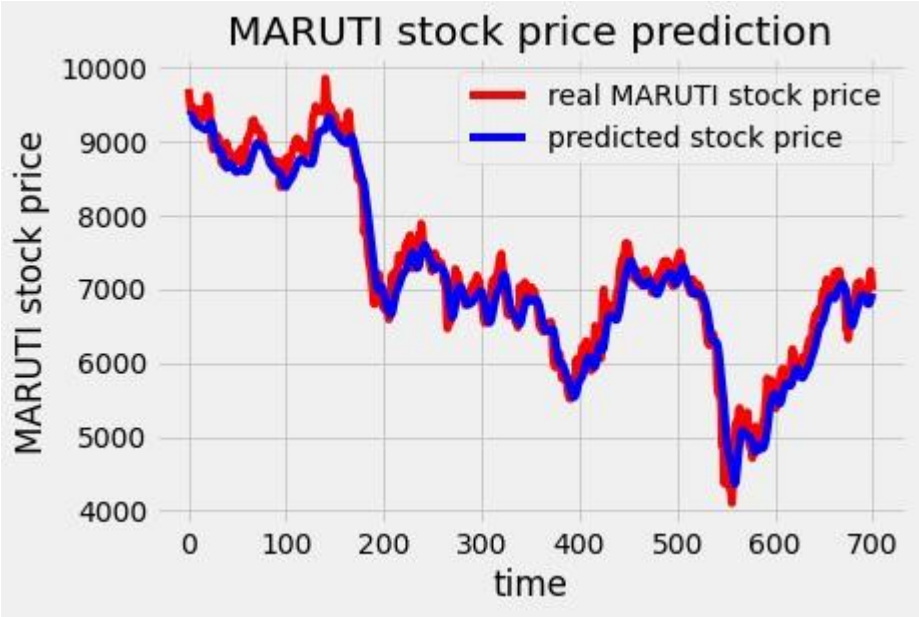
We decided to use Adam's Optimizer here as we concluded from the fact that root mean square Error for Adam's optimizer is Significantly less than RMSProp optimizer. This is so because while the RMS prop algorithm chose a different learning rate for each parameter whereas in Adam's Optimizer algorithm the exponential average of gradient as well as the squares of gradient for each parameter is computed and learning rate is calculated by using these two features for each parameter. So, Adam's optimizer comprises the heuristic of both Momentum and RMSProp optimizers. So, it's better to use Adam's Optimizer in our model.



Loss vs Epoch (RMSProp Optimiser)

Result

By doing this analysis using the LSTM network , we can observe through the graph that it is approximately correctly predicting the attribute “VWAT” and the root mean square is 262.75854387 Which is quite good as for the testing part the stock of VWAP attribute is much greater than 5000. So, we have an error of the order of 1.5-2 percent. So, we can say that the model is predicting in some sense the right value of the attribute VWAP.The figure in next page describes the predicted stock price VWAP against the real stock price for the testing data set.The time axis here represents the number of days.



Drawbacks

As much as our model predicts the price of the stock we cannot fully trust the model to accurately predict in short time frame and invest as there are many other factors which are taken into account while determining the stock prices on any day like some news article or volatility may depend on the volume of the trades done and also maybe because of some lucrative announcements which can be made by some companies or due to some policy of ease/ difficulty of business being announced by the government. This comes under fundamental analysis of stocks which is different from technical analysis.

Technical analysis includes orders flows numbers. For example in 2016 , after the announcement of demonetisation , the stock prices of digital payment companies like Paytm gradually increased. In 2019 as well we observed that sales of automobile companies went down and so their performance in stocks degraded. But in 2020 after lockdown and covid-19 the stocks of Maruti surprisingly went up. So , we can't give the input of a feel-good parameter which can't be evaluated in the model. Another factor which may affect is the sudden market crash due to some unexpected news which hasn't happened earlier.

Conclusion and further advancement

Our model has fared in an efficient way and this model can be used for other stocks as well.

The first attribute to consider is time series frequently. Our model uses daily stock price data as input. However different frequencies of data can be used. Another area to consider for further research is that we can include market sentiment as a feature vector to further correctly build the model . Another area which can be considered is the use of Bayesian Optimisation or some other highly Optimisation process instead of Adam Optimisation to reduce the loss. It Would be worth considering some other architecture or may be considering changes in the hyperparameters of the model we used i.e. learning rate of optimiser / number of layers and number of hidden units in each layer. We can also Tinker with the tuning of drop out layers and experimenting with different activation functions.

References:

https://keras.io/guides/sequential_model/

http://cs230.stanford.edu/projects_fall_2019/reports/26254244.pdf

Xiong Ruoxuan, Eric P. Nichols, Yuan Shen, Deep Learning Stock Volatility with Google Domestic Trends, 2015.