Capstone Project

Data Scientist Nanodegree Program

Avinash Kumar

16th May,2020

Project Definition

Project Overview

Stock market prediction is the act of trying to determine the future value of a company stock or other financial instrument traded on an exchange. The successful prediction of a stock's future price could yield significant profit. Stock market allows us to buy and sell units of stocks (ownership) of a company. If the company's profits go up, then we own some of the profits and if they go down, then we lose profits with them. Prediction provides knowledgeable information regarding the current status of the stock price movement.

This project will help the Intra-day Traders to get insights of the next day's stock market and will help them to prepare their buy and sell strategies. In this project, Stock market forecasting has gained more attention recently, maybe because of the fact that if the direction of the market is successfully predicted the investors may be better guided. The profitability of investing and trading in the stock market to a large extent depends on the predictability. If anysystem be developed which can consistently predict the trends of the dynamic stock market, would make the owner of the system wealthy.

I have used Tata Consultancy Services Pvt. Ltd. historical data which is retrieved from Yaho o Finance [1]. Tata Consultancy Services Limited provides information technology (IT) and IT enabled services worldwide. It operates through Banking, Financial Services and Insurance, Manufacturing, Retail and Consumer Business, Communication, Media and Technology and Others segments.

Problem Statement

The aim of this project is to predict the next day's closing prices of a Tata Consultancy Services stock using the deep learning model Long Short Term Memory [2] and also to optimize the hyperparameters of the network and do feature selection on the dataset. The final model is useful to get the next day's closing price.

[1] https://in.finance.yahoo.com/quote/TCS.NS/profile?p=TCS.NS

[2] C. Olah, "Understanding lstm networks https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Metrics

R2 Score : In statistics, the coefficient of determination, denoted R2 or r2 and pronounced "Rsquared", is the proportion of the variance in the dependent variable that is predictable from the independent variable(s). The R2 coefficient of determination is a statistical measure of how well the regression predictions approximate the real data points.

$$R^2 = 1 - \frac{Sum\_of\_squared\_residuals/n}{variance_{y\_actual}}$$

Mean Squared Error (MSE) : It is the mean squared error regression loss.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (predicted\_value_i - actual\_value_i)^2}{n}}$$

Analysis

Data Exploration

The data used in this project is of the Tata Consultancy Services Pvt. Ltd. taken from June 22, 2009 to June 19,2019 i.e a period of 10 years. This is a series of data points indexed in time order.

Some terminologies related to stock price are given below:

Date : the date of the day on which the trading occurs

Open : price of the first trade for any listed stock is its daily opening price

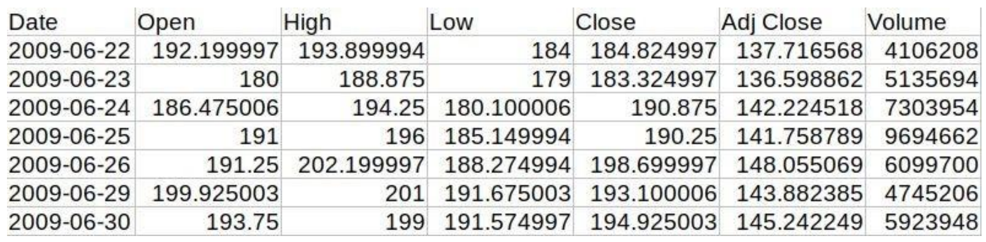
Close : the last price at which a stock trades during a regular trading session

High : the highest price at which a stock is traded during the course of the day

Low : the lowest price at which a stock is traded during the course of the day

Volume : the number of shares or contracts traded in an entire market during a given period of time The dataset looks

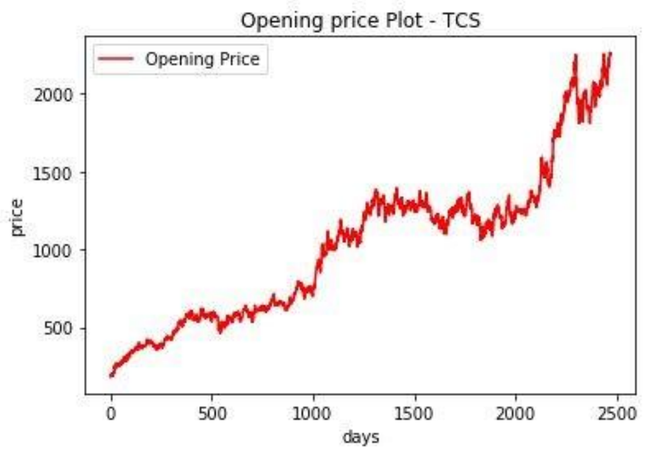
like the following :

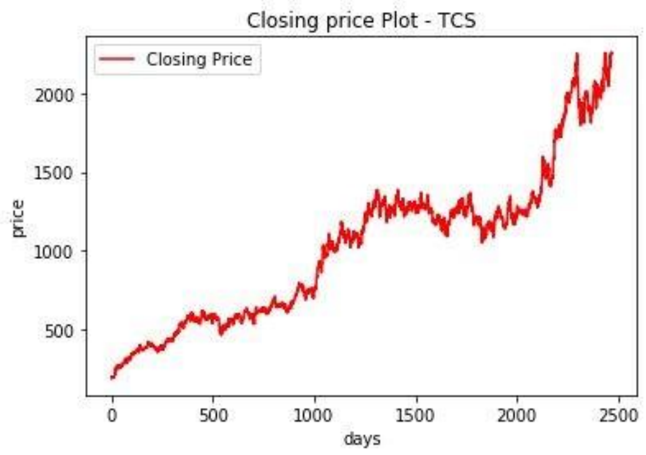


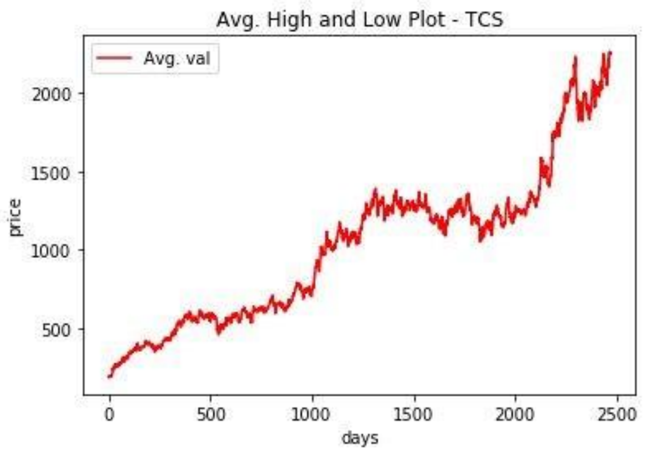
Date	Open	High	Low	Close	Adj Close	Volume
2009-06-22	192.199997	193.899994	184	184.824997	137.716568	4106208
2009-06-23	180	188.875	179	183.324997	136.598862	5135694
2009-06-24	186.475006	194.25	180.100006	190.875	142.224518	7303954
2009-06-25	191	196	185.149994	190.25	141.758789	9694662
2009-06-26	191.25	202.199997	188.274994	198.699997	148.055069	6099700
2009-06-29	199.925003	201	191.675003	193.100006	143.882385	4745206
2009-06-30	193.75	199	191.574997	194.925003	145.242249	5923948

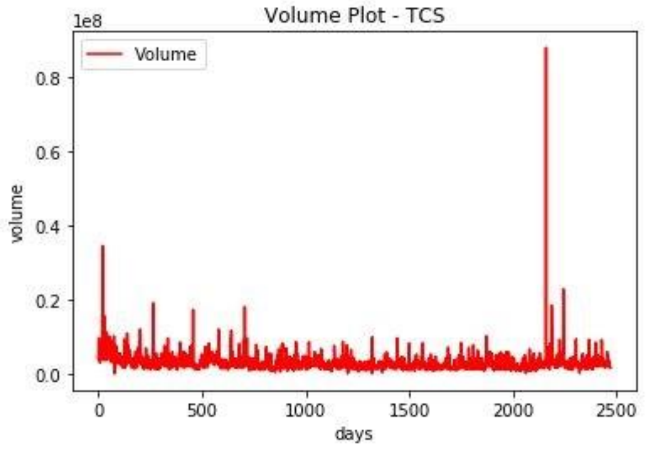
Data Visualization

Below are the plot of all the features which are taken into account to train the model. The X axis represents the number of days from June 22, 2009 onwards and Y axis represents price or volume. From the plot, you can see the continuous growth in the stock of the company but stock was flat from 1300-2100 which will be around 2015-2017.









Methodology

Data Preprocessing

The preprocessing is done in the following step:

- First, the null values are dropped from the dataset
- Second, the normalization of the column values is performed using MinMaxScaler ● Third, creating 3D input tensor

We performed normalization of the data with the help of the following:

$$X_{changed} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

After Normalizing, I split the data into 4:1 ratio. So, after splitting the shape of the set are as follows:

- Training set : (1971, 4)

- Testing set : (553, 4)

After Splitting, I reshaped the data to form the 3D input tensor where the shape of the set are as follows:

- Training set : (1911, 60, 4)
- Testing set : (493, 60, 4)

Implementation

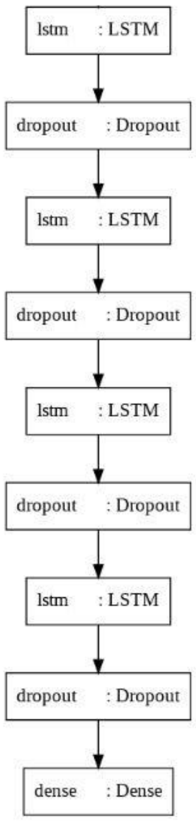
The implementation process are as follows:

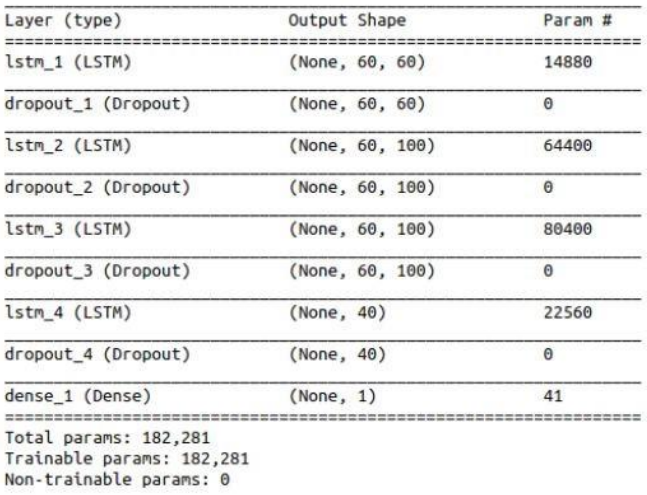
- Loading the dataset and splitting into training and testing set by 80:20 ratio
- Creating the 3D input tensor
- Training the model
- Validating the model ● Plotting the results

In this project, I have incorporated a stacked LSTM network to work on the problem of stock price prediction which is a type of sequence-to-sequence time series problem. A stacked LSTM architecture can be defined as an LSTM model comprising of multiple LSTM layers. Stacking LSTM layers makes the model deeper which helps the model to store more complex information and trends. The additional hidden layers are understood to recombine the learned representation from prior layers and create new representations at high levels of abstraction for next layer.

Here I have introduced 4 LSTM layers, 4 dropout layers and 1 dense layer at the end. In my case, the input LSTM network changes according to the number of features taken into account. In order to feed the input features to the neural network, I first create a 3-dimensional numpy tensor of input features.

There are 60 input nodes in the first LSTM layer which should be equal to the number of timesteps. Then, there are 3 LSTM hidden layers where first two hidden layers have 100 hidden nodes and last LSTM hidden layer has 40 hidden nodes. At the end, there is a Dense layer having only 1 node which outputs the result. After every LSTM layer, I have introduced a Dropout layer to avoid overfitting of the model with a dropout rate of 20% at each layer. In every LSTM layer, I have used a Linear activation function and at the Dense layer, used tanh activation function.





```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 60, 60)            14880
_________________________________________________________________
dropout_1 (Dropout)          (None, 60, 60)            0
_________________________________________________________________
lstm_2 (LSTM)                (None, 60, 100)           64400
_________________________________________________________________
dropout_2 (Dropout)          (None, 60, 100)           0
_________________________________________________________________
lstm_3 (LSTM)                (None, 60, 100)           80400
_________________________________________________________________
dropout_3 (Dropout)          (None, 60, 100)           0
_________________________________________________________________
lstm_4 (LSTM)                (None, 40)                22560
_________________________________________________________________
dropout_4 (Dropout)          (None, 40)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 41
=================================================================
Total params: 182,281
Trainable params: 182,281
Non-trainable params: 0
```

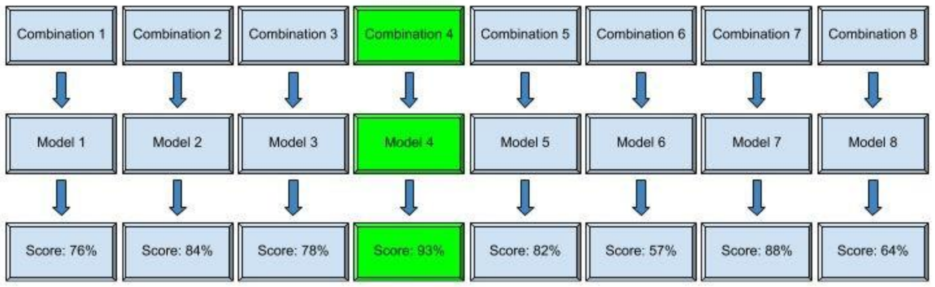
Sometimes after completing the training, I may not get the perfect model because of local minima of cost function. So, I try to incorporate some optimizer in our training topology like Adam or Rmsprop to avoid this type of problems. Here, I have used Adam optimizer which will help to attain the global minima of cost function. To train the model, I have sent the input samples with a batch size of 32. In very first run, the first input LSTM layer will fetch very first timestep from the input tensor containing stock trend of starting 60 days of the dataset. In case of single attribute model, the timestep will be a (60, 1) matrix whereas in case of multiple attribute model eg. 4 feature attributes, thetimestep will be a (60, 4) matrix. After this, a feed forward pass is done and I get an output which I use to calculate the cost function. Here, I have used Mean Squared Error as the cost function to calculate the error.

Refinement

Hyperparameter Optimization

To perform the refinement process, I have incorporated Hyperparameter Optimization and Feature selection using trial and error method.

To get the best possible hyperparameters, I have used trial and error method. In this method, I have created a list of all possible combinations of the hyperparameters and trained our model on each of the combinations and validated the model using the test set. The parameters with the highest validation score will be chosen. Below is an example image.



Feature Selection

Feature Selection is the process where we automatically or manually select those features which contribute most to our prediction variable or output in which we are interested in. Here, we have again used trial and error method to select the best features. We have created a list of all possible combinations of feature attributes and trained the model on each combination and validated the model with the testing set. The combinations with the highest validation score will be selected.

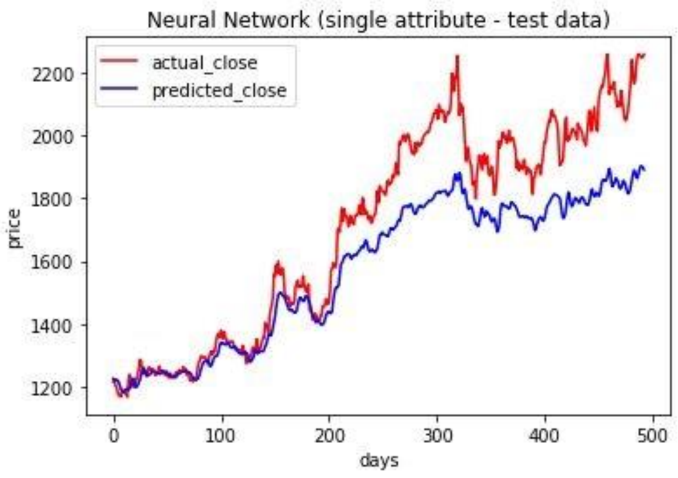
- Reduces Overfitting : Less redundant data means less opportunity to make decisions based on noise.
- Improves Accuracy : Less misleading data means modeling accuracy improves.
- Reduces Training Time : Fewer data points reduce algorithm complexity and algorithms train faster.

Results

Model Evaluation and Validation

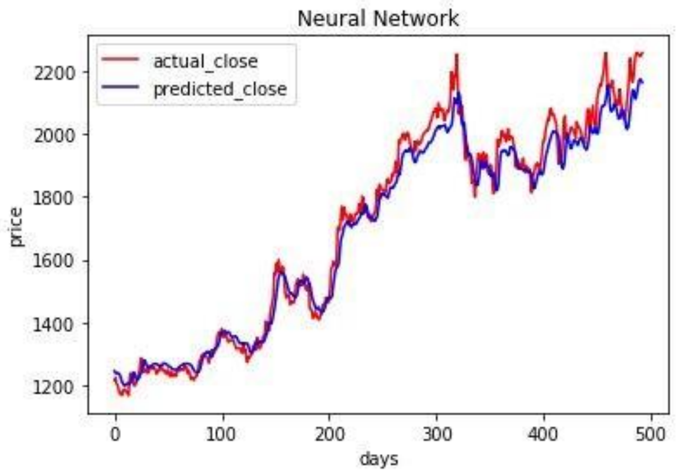
Single Feature Model

In the model, the model achieved an r2 score of 0.74709 and mse value of 568.16 on the test set.



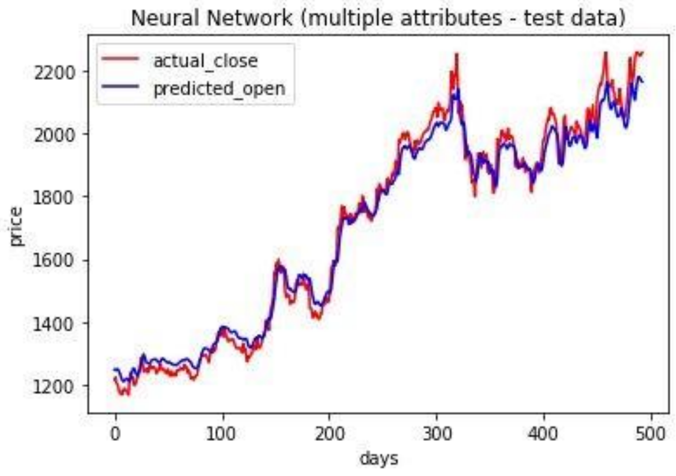
Hyperparameter Optimized Model

In the process of hyperparameter optimization, the best model achieved an r2 score of 0.97720 and mse value of 2575.15 on the test set.



Final Model

After feature selection process, the best model achieved an r2 score of 0.98379 and mse value of 1830.83 on the test set.



As the r2 score value is nearly close to 1, so we can say that the model is pretty good in predicting the next day's closing price.

Justification

We trained the model using the data of Tata Consultancy Services stock and first predicted the closing prices using single attribute only yielding an r2 score of 0.74709 and mse value of 568.16 on the test set. Then after doing hyperparameter optimization and feature selection, we have seen the results yielding an r2 score of 0.98379 and mse value of 1830.83 on the test set in predicting the closing price. And came to the conclusion that our LSTM model is good at predicting time dependent data analysis problem when proper optimization techniques are used.

Conclusion

Reflection

Using neural networks to forecast stock market prices will be a continuing area of research as researchers and investors strive to outperform the market, with the ultimate goal of bettering their returns. Since there are many indeterminate parameters that directly or indirectly affect stock market, each and every one of them cannot be taken into account. So our model only depends on the relationship of our selected parameters of the stock market. As I have only limited computational power, so I only optimized selected number of hyperparameters only.

Improvement

It is unlikely that new theoretical ideas will come out of this applied work. However, interesting results and validation of theories will occur as neural networks are applied to more complicated problems. To improve it further, we can also apply sentiment analysis using news headline and can club together the results of the two models to get the more concrete predictions. We can also take more number of hyperparameters into account to optimize the model further, but make sure you have powerful GPU to train the model as training these models take a lot of time.