NOTEBOOK

Zerodha Stock Price Predictor

Predict closing price of a stock using neural networks

Learn

#1

Introduction

ZERODHA

"They pioneered the discount broking model in India. Now, they are breaking ground with their technology."

Zerodha was founded on 15 August 2010 by brothers Nithin Kamath and Nikhil Kamath. The name Zerodha is a combination of the words Zero and Rodha, a Sanskrit term for barriers. The aim was to convey to people that there would be Zero Barriers when it comes to investing with Zerodha.

What makes them different from other retail brokers is that they charge no brokerage on equity delivery trades and for other categories, the commission charges are as low as Rs.20 or 0.03% of the transaction. They took this step to attract young investors who generally don't invest due to high commission.

Technology-wise also Zerodha is quite different from other retail brokers in the market. They offer various services like Kite, coin, and many more.

#1

Introduction

ZERODHA

A stock (also known as equity) is a security that represents the ownership of a fraction of a corporation. This entitles the owner of the stock to a proportion of the corporation's assets and profits equal to how much stock they own. Units of stock are called "shares."

Tata Consultancy Services (TCS) is an Indian multinational information technology services and consulting company. TCS is a global leader in IT services, consulting & business solutions with a large network of innovation & delivery centres. The below graph shows the stock price of TCS from 2004. You can observe that TCS stock price rose from 120 INR to 3500+ INR. Isn't it would be great if you knew that its price will rose that much?

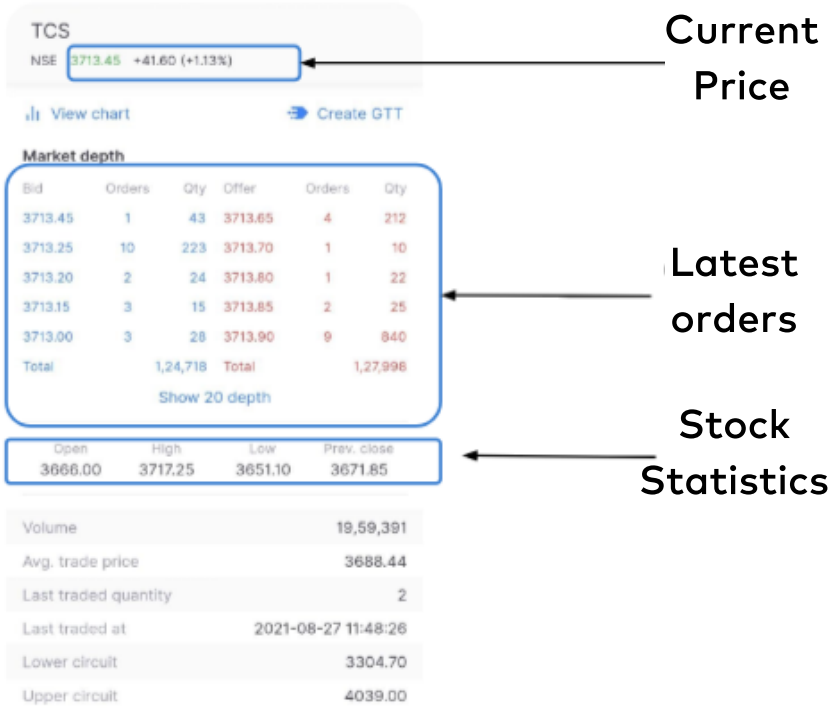
Zerodha Kite

The picture below is of Kite's app which is online trading software. We can get any information about a particular stock. The most important for traders is the current price of the stock, the latest orders of the stock, and other stock statistics like previous closing price, the opening price of the stock.

#1

Introduction

ZERODHA



Zerodha is planning to offer a new feature to their PRO users. They want to show predicted today's closing price of stocks. This will help all the traders in choosing when to sell or buy a stock. After implementing this they can also show next week trend of stocks to their PRO users.

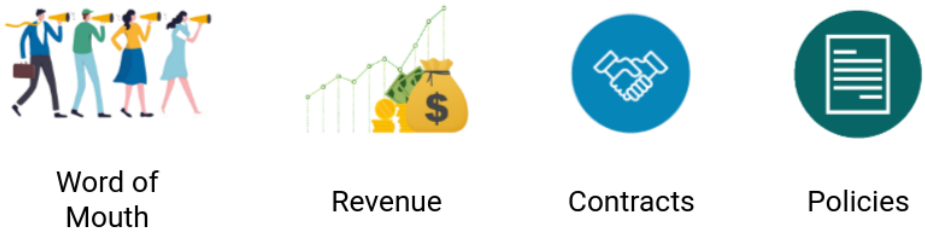
To implement this their engineers have to use time series forecasting. We will understand each term, and how it works in upcoming chapters.

#2

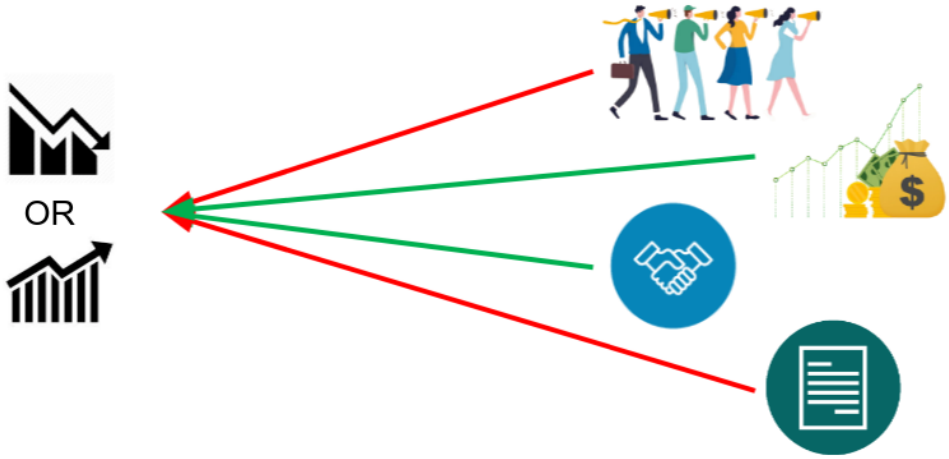
Neural Network

We need a system or algorithm that will take factors on stock price depends on input, and predict the closing price of the stock. We will be using a neural network.

Let us say the closing price of a stock depends on word of mouth of the company, the company's revenue, contracts of the company and various company and government policies.



We will take these factors after analysing each of them, and fed them to our neural network and predict whether our price will increase or not.



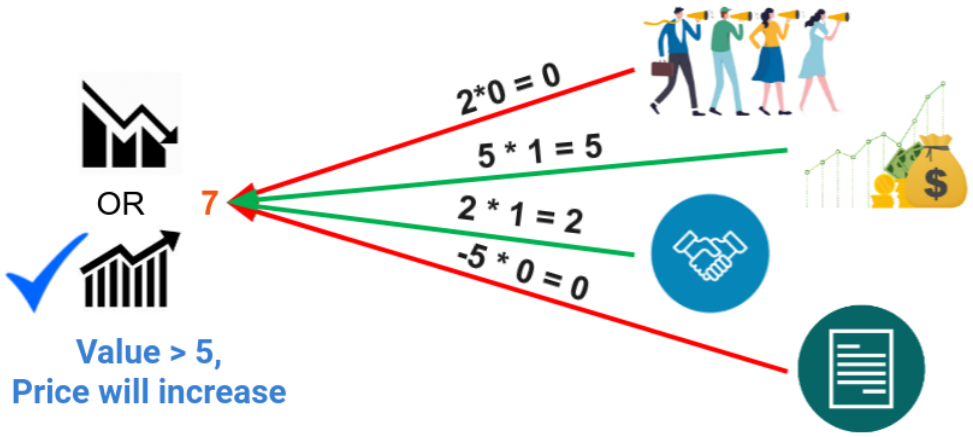
Each lines (red and green) are weights which will scale all the respective input. We need them because some factor will increase the price, and some factor will decrease the price but at different scale. So weight can be positive or negative.

#2

Neural Network

Let us say that there's no word of mouth of the company (input will be 0), the company's revenue is increasing daily (input will be 1), the company have good contracts (input will be 1), and government policies are not favourable to the company (input will be 0)

Now we will scale these inputs by multiplying them by their respective weights as shown in the figure below. Later we will add all the scaled inputs and we will get a number. Now we can decide if this number is greater than 5 then price of the stock will increase and if it's less, the price of the stock will decrease.



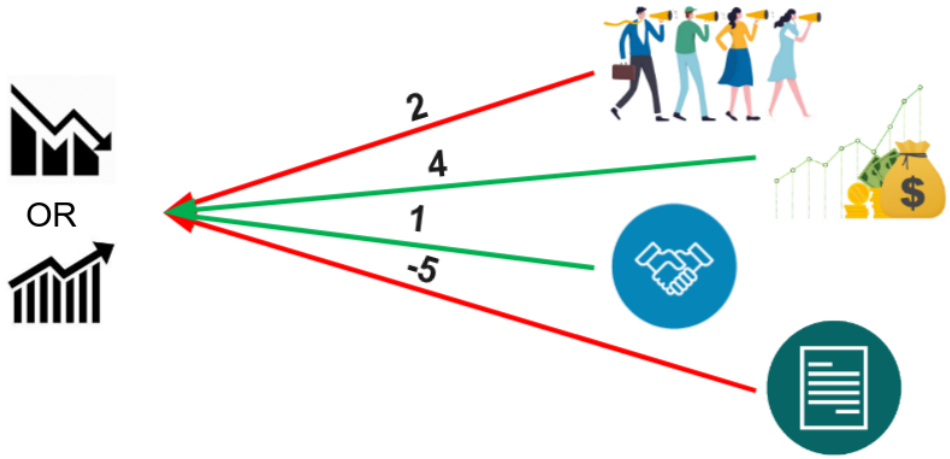
After all the calculations we receive 7 as output, which is greater than 5. So we can say that our price will increase. After this, we can wait for one day and check out our prediction if its incorrect then we need to make a few modifications

As our output depends on weights which we can change to change the output. After creating a neural network, we need to train them which is a process in which we update the weights so that we will get correct output.

#2

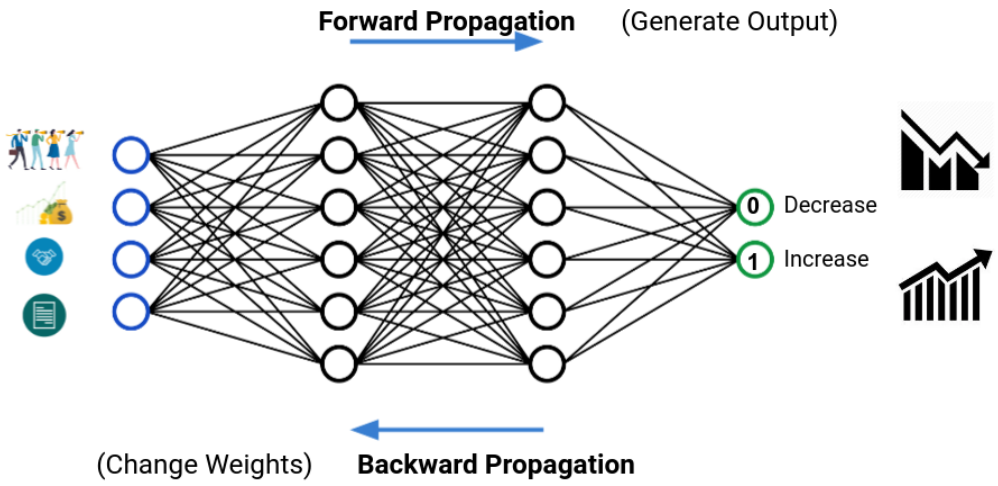
Neural Network

Let us change the current weight of our neural network. As only revenue and contracts were activated (input were 1), so our neural network needs to change their weights.



So we changed their weights from 5 to 4, and 2 to 1, respectively. Now for next day prediction we will use these weights instead of old ones.

The training of neural networks involves two steps, forward propagation and backward propagation. We can also add more neurons (building blocks of neural network) to our neural network. The figure below shows a neural network and the training process.



#2

Neural Network

In the above figure, we can observe that there are multiple layers of neurons. The first layer is called the input layer because through this layer our neural network receives input. The next two layers are called hidden layers, these layers receive data from the previous layer and perform some calculations on it, and send the output to the next layer. The last layer of the neural network is called the output layer because the final output of the neural network is given through this layer.

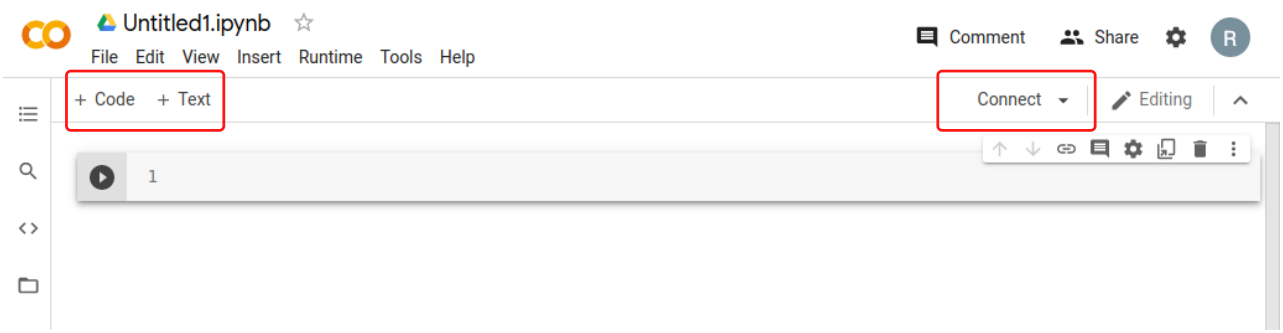
Our first step is to generate the output, i.e. predict whether the stock price will increase or not. This step is called forward propagation. The next step is backward propagation, in which we update or change weights so that we will get the correct output.

#Extra

Introduction to Colab

Nearly all machine learning and deep learning algorithms require good hardware. What if ... you don't have good hardware? Should you drop your dream to be a data scientist? No, there's an alternative Let us introduce you to Colab.

Colab is a service provided by Google which lets you access a virtual machine hosted on google servers. These virtual machines have dual-core Xeon processors, with 12GB of RAM. You can even use GPU for your neural networks. Colab is an interactive python notebook (ipynb), which means that with writing python code, you can also write normal text, include images.
To create a new colab notebook, just go to "https://colab.research.google.com", and create a new notebook. You will get something like below



You can connect to runtime by clicking the "Connect" button in the right corner. You can add a new Code cell, or text cell using respective buttons in the toolbar.

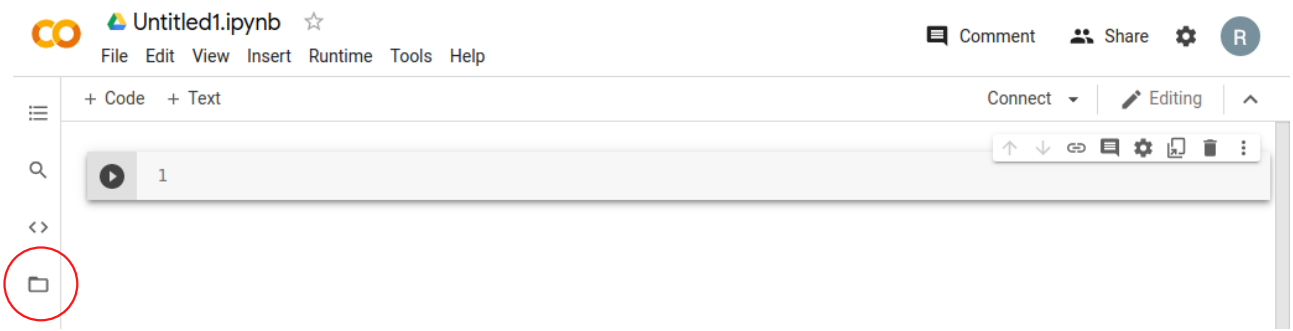
#Extra

Introduction to Colab

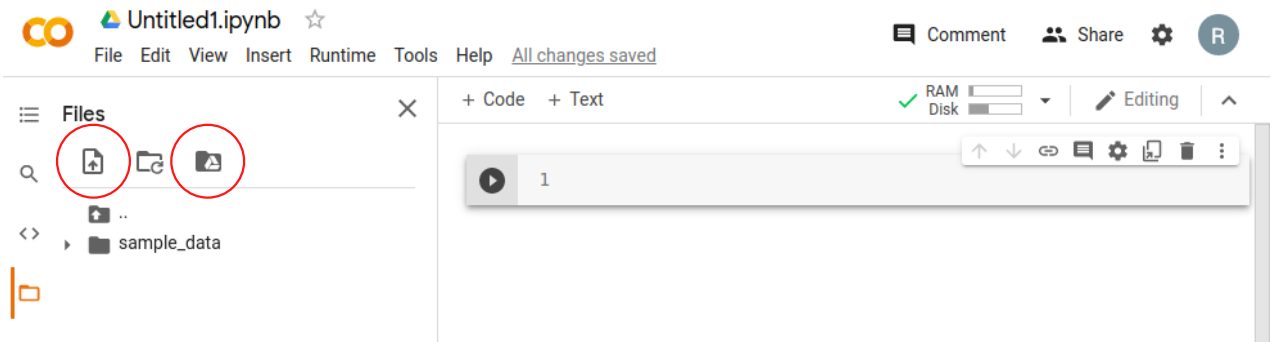
Uploading files

Sometimes you need to use a file from your PC. For that, you can upload the required files to google colab. Colab provides 100 GB in a colab session. If you want to access some files from your drive, you can even do so by connecting the drive to colab.

To upload something, open file pane from left toolbar.



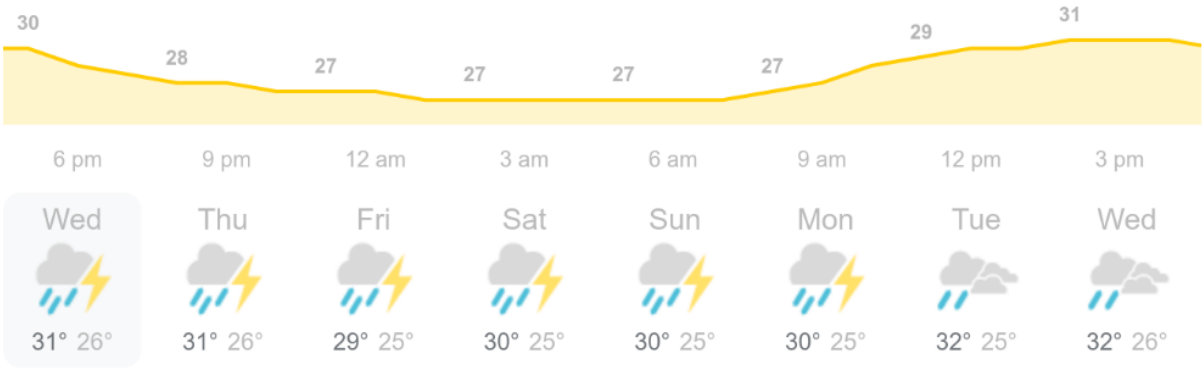
Now, just upload the file using first button, and you can also mount google drive using last button



#3

Time Series

In mathematics, a time series is a series of data points indexed (or listed or graphed) in time order. In time series, data at time T depends on previous time data. (timestamp). The perfect example of a time series is of weather.



In the above figure, we can see how temperature is indexed by the time of the day. Between 6 AM to 12 AM, the temperature is dropping and it's constant till 6 AM. After that, the temperature keeps on increasing.

Our stock's data is also a time series, as we have different prices for each day.

	Opening price	Highest price for the day	Lowest price for the day	Closing price	Number of shares traded
	Open	**High**	**Low**	**Close**	**Volume**
2002-08-12	38.724998	40	38.724998	39.700001	212976
2002-08-13	39.75	40.387501	38.875	39.162498	153576
2002-08-14	39.25	39.25	35.724998	36.462502	822776

#3

Time Series

Downloading data

```
!wget https://techlearn-
cdn.s3.amazonaws.com/bs_zerodha_stock_price/tcs_share.csv
```

Importing library

Pandas is a library that is used for data manipulation and analysis. We will use if to read stocks data.

```
import pandas as pd
```

Loading data

```
data = pd.read_csv('tcs_share.csv')
```

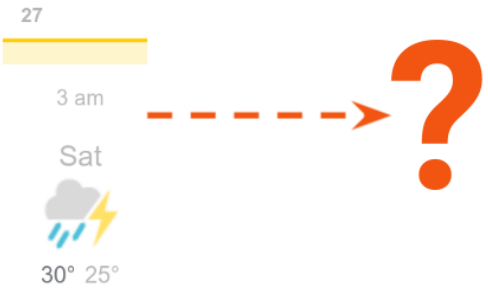
Right now, the Date column is just strings, we need to convert them to datetime objects and drop null columns (columns with no data)

```
data['Date'] = pd.to_datetime(data['Date'])
data = data.dropna()
```

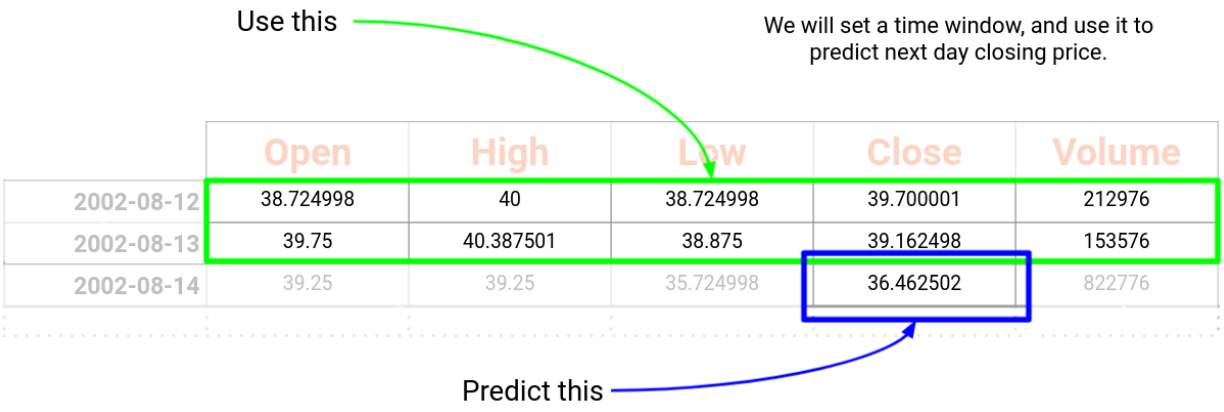
#3

Time Series

Imagine you were asked to predict temperature after one hour, and you were only provided with the current temperature. Can you predict it?
Nobody cannot predict next hour temperature without providing few previous temperature readings.



Similarly, if we need to predict today's closing price of a stock we need at least two days of data on the stock. We say this as a window. We can set the window of 2 days, 7 days, or even 60 days, we just need to create a batch of these windowed data points, like in the figure below.



Splitting data into training and testing set

```
train_size = 0.8        # 80%
split_index = int(train_size * data.shape[0])
factors_column = ['Open', 'High', 'Low', 'Close', 'Volume']
y_col_index = 3 # Close
train_set = data[factors_column].values[:split_index]
test_set = data[factors_column].values[split_index:]
```

#3

Time Series

Scaling data

The price of stocks varies from 0.4 INR to 7000 INR. This is a large range of price which affects the performance of a neural network. So we will scale these prices between a range of 0-1.

```
from sklearn.preprocessing import MinMaxScaler

sc = MinMaxScaler(feature_range = (0, 1))
train_set_scaled = sc.fit_transform(train_set)
test_set_scaled = sc.fit_transform(test_set)
```

Generate windowed timestamp data

We will create a function that will take whole data as input, and create batches of data according to window size.

```
import numpy as np

def generate_data(series, y_col_index, time_window=60):
    X = []
    y = []
    for i in range(60, len(series)):
        X.append(series[i-time_window: i])
        y.append(series[i, y_col_index])
    return (np.array(X), np.array(y))
```

Now we will use above-defined question to create windowed data.

```
X_train, y_train = generate_data(train_set_scaled,
y_col_index=y_col_index)
X_test, y_test = generate_data(test_set_scaled,
y_col_index=y_col_index)
```

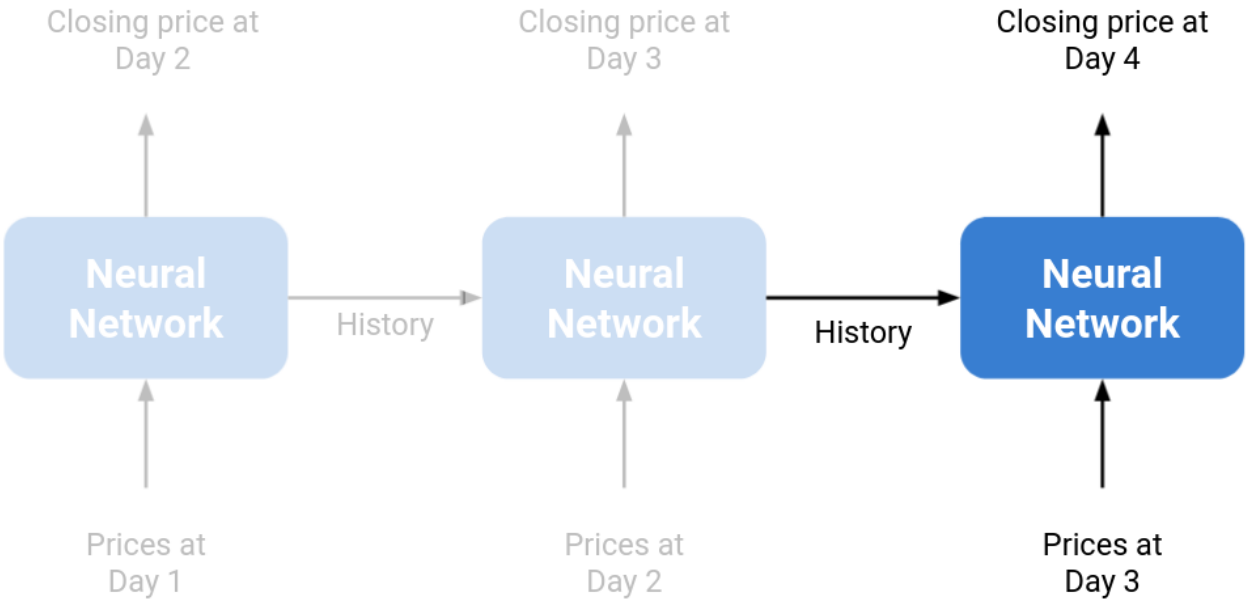
#4

Intuition

If we try to read chapter 4 of a novel without reading 1st, 2nd, and 3rd chapters, then we won't be understanding anything because we would be lacking the context of the novel and the characters that were introduced in the initial chapters. So, previous knowledge of chapters is important in understanding the next chapters.

Like we saw that previous knowledge is required to understand new chapters of a novel easily, similarly in time series (stock price data) at a particular time depends on previous data, so we need to modify our neural network because they can't see back in time.

We need to create a neural network that can see back in time, or should have a memory of previous data.



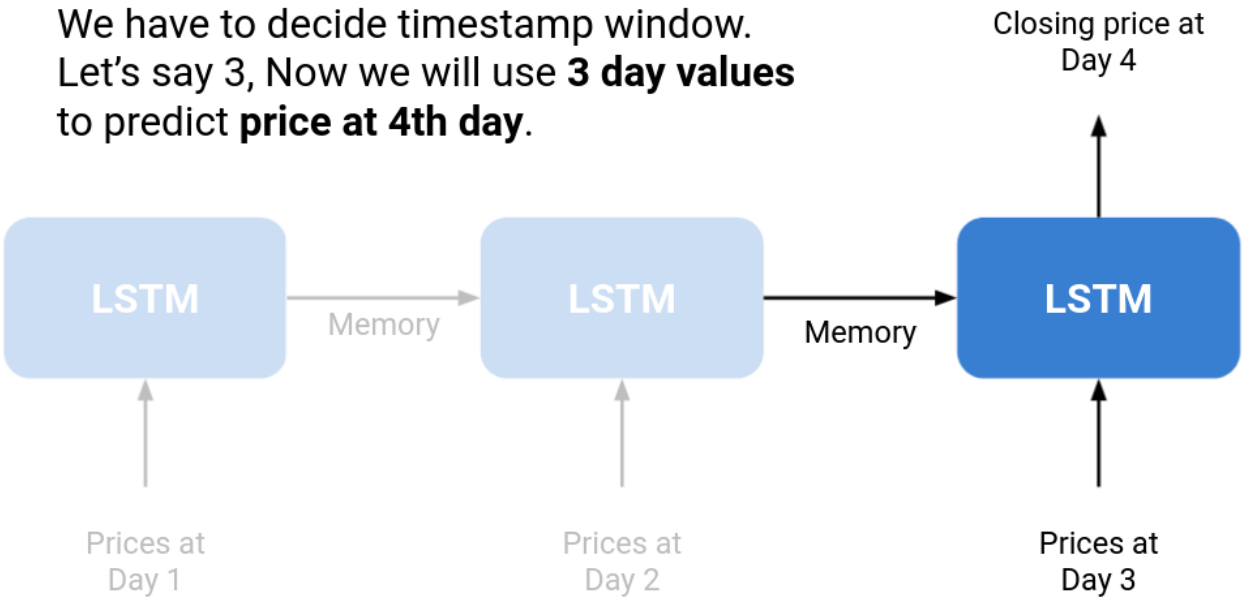
So, we will use something similar to the above neural network. This neural network will take the price of the stock for a day, and predict the next price. It will also pass its memory to itself which will be used to predict next day price.

#4

LSTM

Long Short-Term Memory (LSTM) networks are a type of neural network capable of learning time series and predict the next value. It works in a similar way like we developed our intuition. It maintains a memory state of old timestamp data which is used in the next prediction.

LSTM will take whole window data (60 days data of stock price), and will predict the price for the 61st day. After taking the whole window data, it will pass data for each data through itself, and maintain a memory for itself. This memory will be passed to itself when it will receive the next day's price.

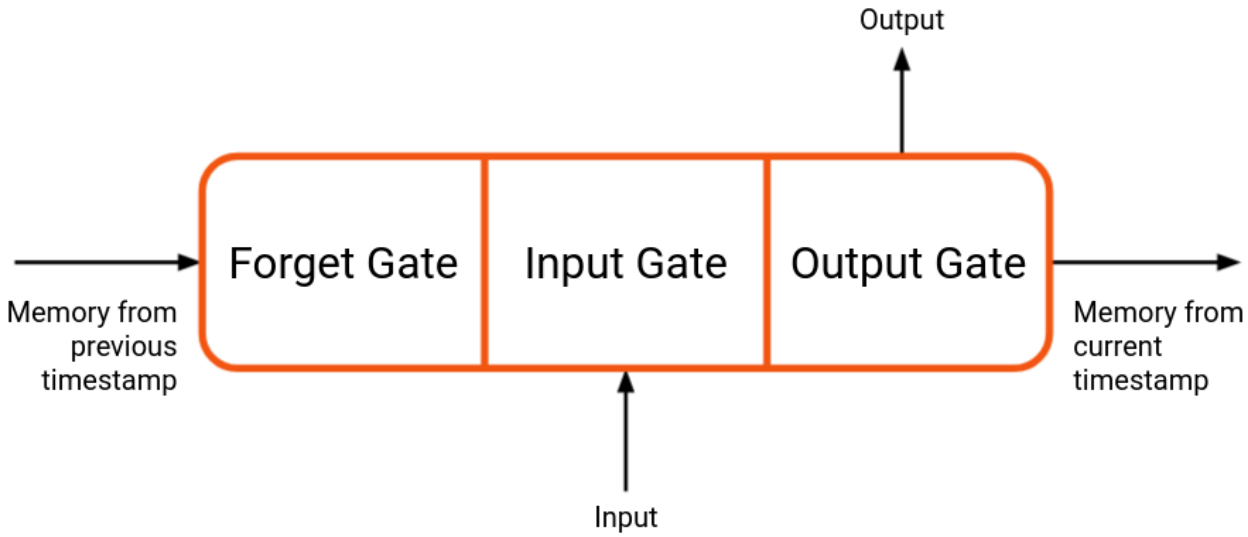


In the above picture, one thing we need to note is that all the lstm in the picture are the same but from a different time. The horizontal lines which get passed to lstm of different times are memory states which hold the information of previous data which was processed by lstm.

#4

LSTM

Internal Structure of LSTM



The above figure shows the internal structure of the LSTM cell. LSTM cells receive two data, the first is the memory from the previous timestamp, and the second is the data that needs to be analyzed or pricessed. There are three gates in a LSTM cell:

1. Forget Gate
2. Input Gate
3. Output Gate

Forget Gate

This gate receives memory from the previous timestamp and it chooses what should be remembered and forgotten from the previous timestamp memory. Only relevant information is remembered, the rest of the information is deleted from the memory.

#4

LSTM

Input Gate

This gate receives memory from forget gate, and data that needs to be analyzed. The received data contains some relevant information and some irrelevant. Its the job of input gate to find relevant information and add it to the memory.

Output Gate

This gate receives memory from the input gate. This gate job is to generate the output and prepare memory for the next timestamp lstm. It does this by predicting price for next day using the memory state of lstm.

Creating LSTM network

Importing required library

We will be using Keras to build our lstm network

```
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
```

#4

LSTM

Now that we have imported library, we will create a multi layered lstm network

```
model = Sequential()

model.add(LSTM(units = 50, return_sequences = True, input_shape =
X_train.shape[1:]))
model.add(Dropout(0.2))

model.add(LSTM(units = 20, return_sequences = True))
model.add(Dropout(0.2))

# model.add(LSTM(units = 10, return_sequences = True))
model.add(LSTM(units = 10))
model.add(Dropout(0.2))

# model.add(LSTM(units = 10))
# model.add(Dropout(0.2))

model.add(Dense(units = 1))
```

Compiling and training lstm network

```
model.compile(optimizer = 'adam', loss = 'mean_squared_error')
model.fit(X_train, y_train, epochs = 20, batch_size = 64)
```

#4

LSTM

Testing trained LSTM network

Now, we will test our trained lstm network. As we have scaled our price between 0 and 1, we need to inverse the scaling process.

```
test_prediction = model.predict(X_test)
test_prediction = (test_prediction * sc.data_range_[y_col_index]) +
sc.data_min_[y_col_index]
```

We can also plot this result using matplotlib

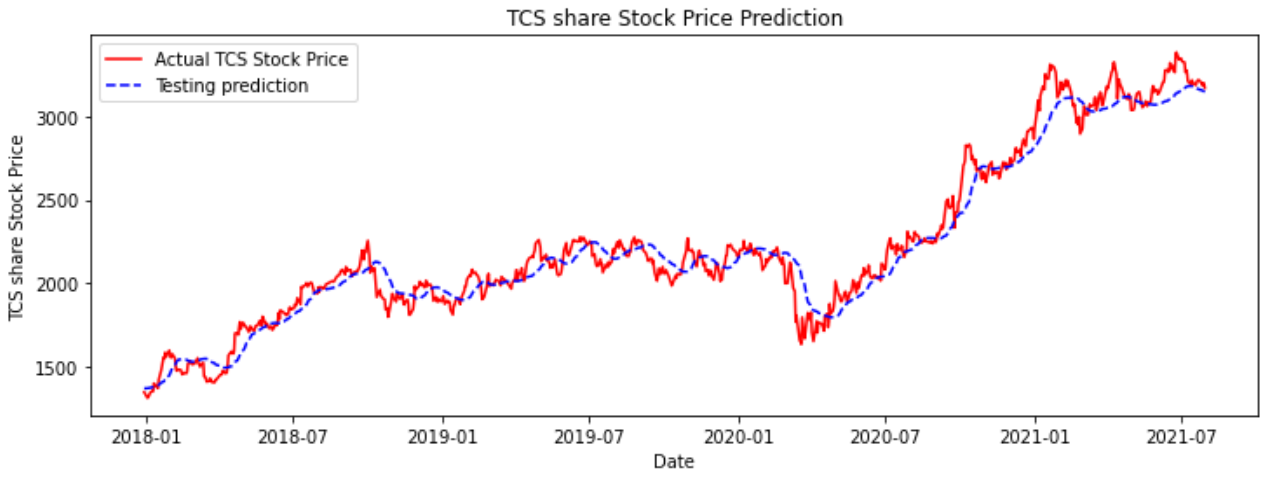
```
fig = plt.figure(figsize = (15, 5))
plt.plot(data.Date.values[ split_index+60 : ], data.Close.values[
split_index+60: ], 'r-', label = 'Actual TCS Stock Price')
plt.plot(data['Date'].values[split_index+60 : ], test_prediction, 'b-
-', label = 'Testing prediction')

plt.title('TCS share Stock Price Prediction')
plt.xlabel('Date')
plt.ylabel('TCS share Stock Price')
plt.legend()
plt.show()
```

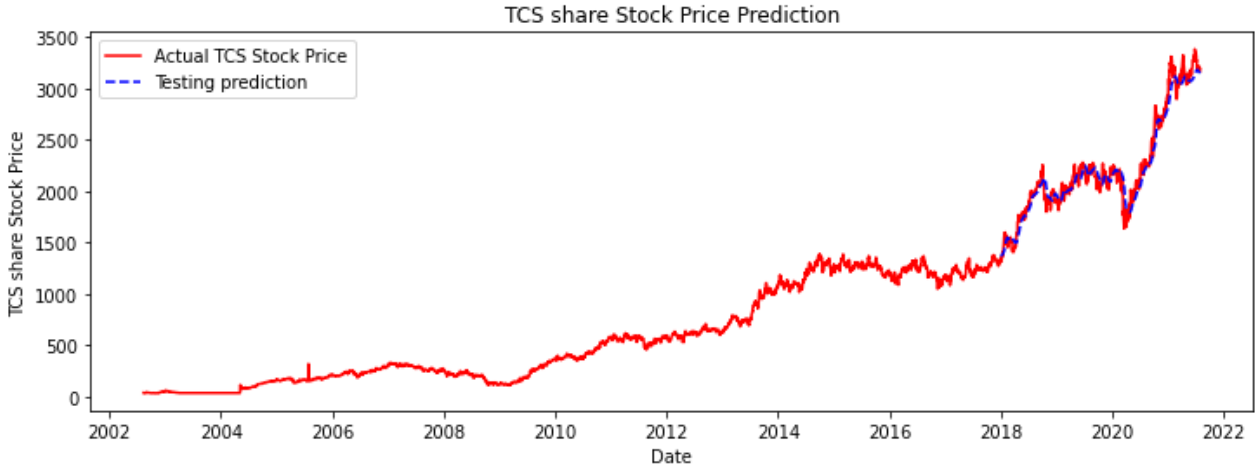
#4

LSTM

Result



The blue dashed line represents the predicted price of TCS stock, and the red line represents our actual price. You can observe that our network was accurate enough to predict when our price will increase and when the price will decrease.



The above graph shows training data used and testing data. Initial 80% of data is used for training, and the rest 20% of data is used for testing. You can observe that our network is able to predict price how efficiently.

Learn

Knowledge Discovery through Brand Stories

Visit Us at :
techlearn.live

Email : support@techlearn.live
Phone : +91-9154796743