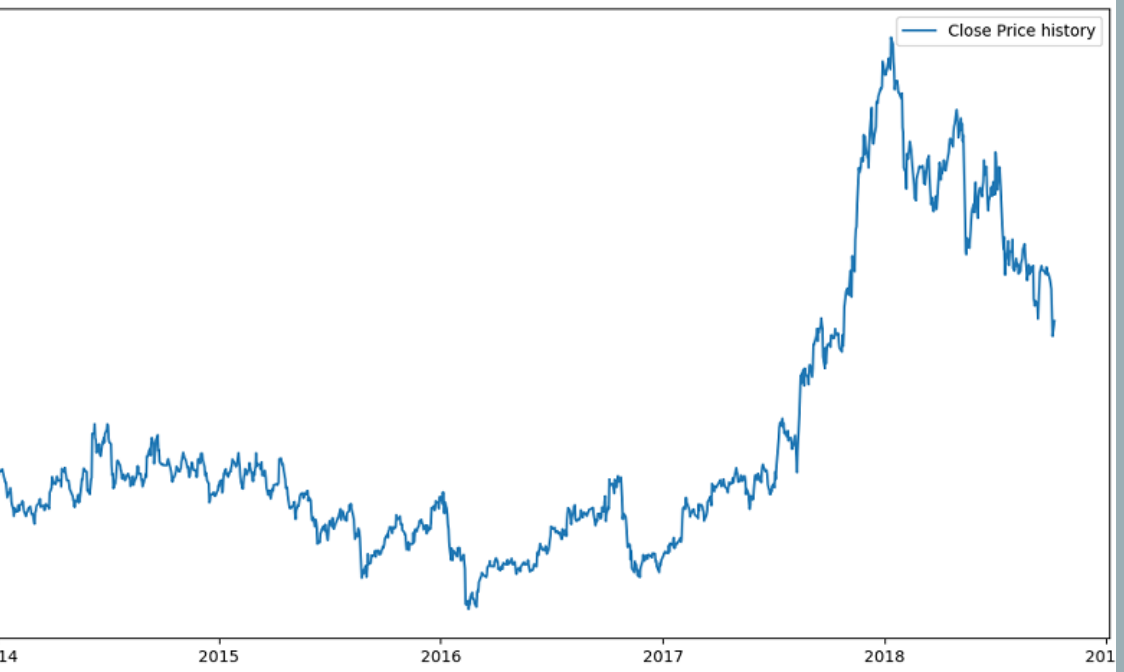
STOCK PRICE PREDICTON

NAME – ARYAN SAINI



INTRODUCTION

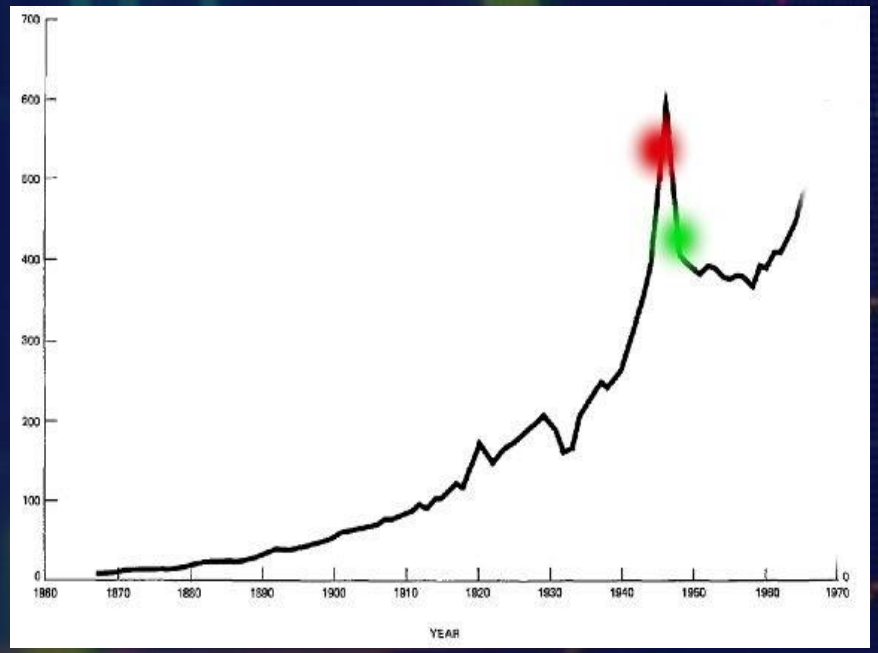
A stock market is a public market for the trading of company stock.

Stock market allows us to buy and sell units of stocks (ownership) of a company.

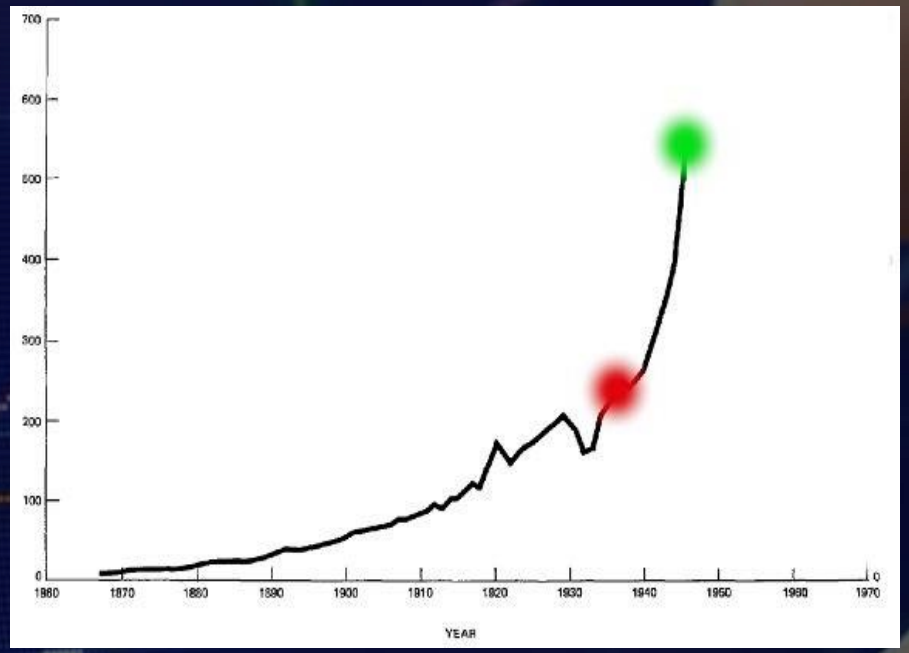
If the company's profits go up, then we own some of the profits and if they go down, then we lose profits with them.

If more seller than buyers, stock price tend to fall. Conversely, when more buyers than sellers, stock prices tend to rise.

INTRODUCTION



Loss



Profit

Red is purchased stock.
Green is sold stock.

PROBLEM STATEMENT

- To accurately predict the future closing value of a given stock across a given period of time in the future.
- Use different machine learning and deep learning models available and compare them in terms of graphical analysis.

HOW TO READ A STOCK TABLE?

Date- day on which the stock is traded

High- **high** is the **highest** price at which a **stock** traded during the course of the day

Close- refers to the last **price** at which a **stock** trades during a regular trading session

	Date	Open	High	Low	Close	Volume
2	20-Jul-17	997.00	998.68	984.62	992.19	1418385
3	19-Jul-17	990.01	995.60	987.01	992.77	1412148

Open- price of the first trade for any listed **stock** is its daily **opening** price.

Low- lowest price at which a **stock** trades over the course of a **trading** day.

Volume- the number of **shares** or contracts traded in a security or an entire **market** during a given period of time

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df=pd.read_csv("NSE-TATA.csv")

print(df.head())
```

DATASET

To build the stock price prediction model, we will use the NSE TATA GLOBAL dataset. This is a dataset of Tata Beverages from Tata Global Beverages Limited, National Stock Exchange of India.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\AL-code\Stock_Prediction> python demo.py
         Date    Open    High     Low    Last   Close  Total Trade Quantity  Turnover (Lacs)
0  2018-10-08  208.00  222.25  206.85  216.00  215.15             4642146.0          10062.83
1  2018-10-05  217.00  218.60  205.90  210.25  209.20             3519515.0           7407.06
2  2018-10-04  223.50  227.80  216.15  217.25  218.20             1728786.0           3815.79
3  2018-10-03  230.00  237.50  225.75  226.45  227.60             1708590.0           3960.27
4  2018-10-01  234.55  234.60  221.05  230.30  230.90             1534749.0           3486.05
PS E:\AL-code\Stock_Prediction>
```


PREPARING THE DATA

DEALING WITH MISSING DATA

As there we are having no missing value in your data .We can move forward.

test2.py | stock_pred.py

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df=pd.read_csv("NSE-TATA.csv")

df["Date"]=pd.to_datetime(df.Date,format="%Y-%m-%d")
df.index=df['Date']
print(df.isnull().sum())
'''
```

```
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\AL-code\Stock_Prediction> python demo.py
Date                    0
Open                    0
High                    0
Low                     0
Last                    0
Close                   0
Total Trade Quantity    0
Turnover (Lacs)         0
dtype: int64
PS E:\AL-code\Stock_Prediction>
```

VISUALIZE THE CLOSING PRICE HISTORY

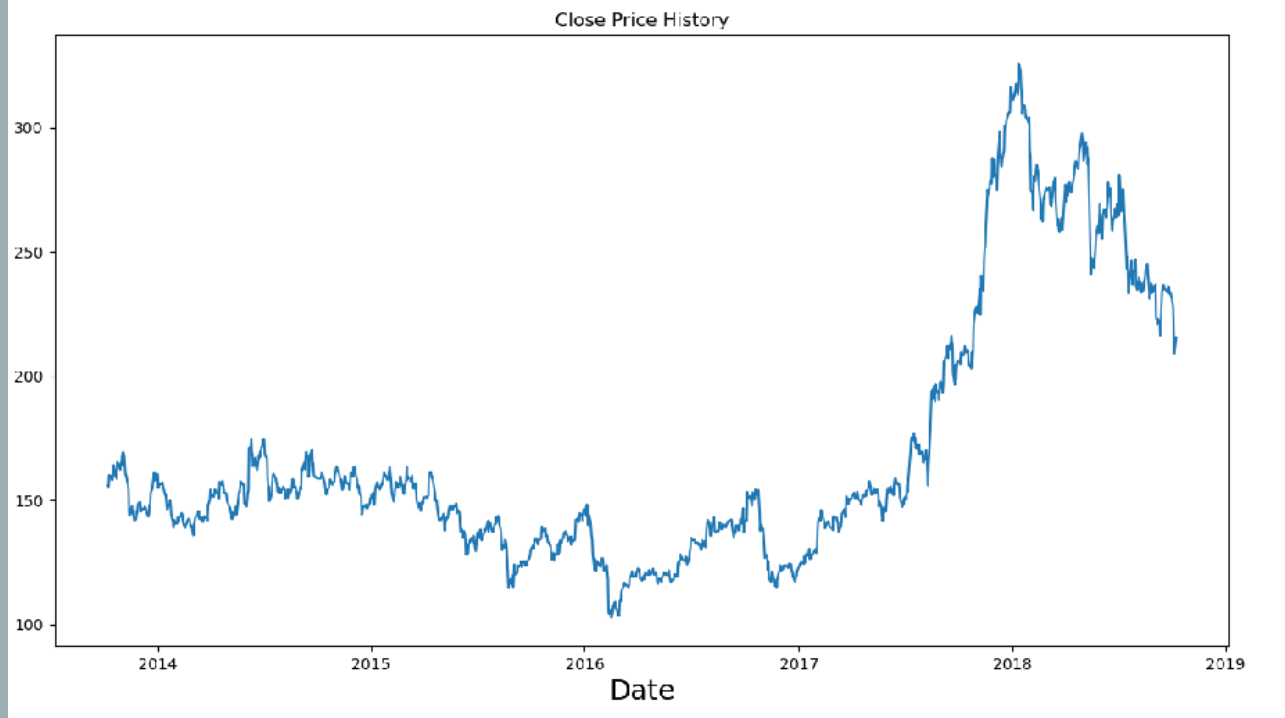
```
File  Edit  View  Selection  Find  Packages  Help

test2.py   stock_pred.py   demo.py

import matplotlib.pyplot as plt
df=pd.read_csv("NSE-TATA.csv")
df["Date"]=pd.to_datetime(df.Date,format="%Y-%m-%d")
df.index=df['Date']
plt.figure(figsize=(16,8))
plt.title('Close Price History')
plt.plot(df['Close'])
plt.xlabel('Date',fontsize=18)
plt.ylabel('Close Price USD ($)',fontsize=18)
plt.show()
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6
```



SORT THE DATASET ON DATE TIME AND FILTER "DATE" AND "CLOSE" COLUMNS

We have sort your data into ascending order and make a new dataframe for your target variable.

demo.py — E:\AL-code\Stock_Prediction — Atom

File Edit View Selection Find Packages Help

test2.py stock_pred.py

```
1  import pandas as pd
2  import numpy as np
3  import matplotlib.pyplot as plt
4
5  df=pd.read_csv("NSE-TATA.csv")
6  df["Date"]=pd.to_datetime(df.Date,format="%Y-%m-%d")
7  df.index=df['Date']
8
9  data=df.sort_index(ascending=True,axis=0)
10 new_dataset=pd.DataFrame(index=range(0,len(df)),columns=['Date
11
12 for i in range(0,len(data)):
13     new_dataset["Date"][i]=data['Date'][i]
14     new_dataset["Close"][i]=data["Close"][i]
15
16 new_dataset.index=new_dataset.Date
17 new_dataset.drop("Date",axis=1,inplace=True)
18 final_dataset=new_dataset.values
19
```

NORMALIZE THE NEW FILTERED DATASET:

Now scale the data set to be values between 0 and 1 inclusive, I do this because it is generally good practice to scale your data before giving it to the neural network.

Denormalized Dataset

Open	Close	Volume
997.00	992.19	1418385
990.01	992.77	1412148
973.36	986.95	1413335
976.32	975.96	1660464

Normalized Dataset

Open	Close	Volume
0.012051212203759196	0.015141267590269444	0.37724770218929105
0.01419776627621902	0.010657794262026843	0.3256440180584465
0.009893761917733584	0.010112359550561806	0.1898197785299343
0.010874421138654333	0.007407003381695226	0.24270137132323968
0.008368292018523571	0.010297807352459915	0.22490508770060744

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


DATA SPLITTING

X_TRAIN, Y_TRAIN AND RESHAPE

Reshape the data to be 3-dimensional in the form [number of **samples**, number of **time steps**, and number of **features**]. This needs to be done, because the LSTM model is expecting a 3-dimensional data set.

```
56
57  new_dataset.index=new_dataset.Date
58  new_dataset.drop("Date",axis=1,inplace=True)
59
60  final_dataset=new_dataset.values
61
62  train_data=final_dataset[0:987,:]
63  valid_data=final_dataset[987:,:]
64
65  scaler=MinMaxScaler(feature_range=(0,1))
66  scaled_data=scaler.fit_transform(final_dataset)
67
68  x_train_data,y_train_data=[],[]
69
70  for i in range(60,len(train_data)):
71      x_train_data.append(scaled_data[i-60:i,0])
72      y_train_data.append(scaled_data[i,0])
73
74  x_train_data,y_train_data=np.array(x_train_data),np.array(y_train
75
76  x_train_data=np.reshape(x_train_data,(x_train_data.shape[0],x_tra
77
78
```

BUILD THE LSTM MODEL

Build the LSTM model to have two LSTM layers with 50 neurons and two Dense layers, one with 25 neurons and the other with 1 neuron.

```
77
78
79
80  lstm_model=Sequential()
81  lstm_model.add(LSTM(units=50,return_sequences=True,input_shape=(x_train_data.shape[1],1)))
82  lstm_model.add(LSTM(units=50))
83  lstm_model.add(Dense(1))
84
85  lstm_model.compile(loss='mean_squared_error',optimizer='adam')
86  lstm_model.fit(x_train_data,y_train_data,epochs=1,batch_size=1,verbose=2)
87
88  inputs_data=new_dataset[len(new_dataset)-len(valid_data)-60:].values
89  inputs_data=inputs_data.reshape(-1,1)
90  inputs_data=scaler.transform(inputs_data)
91
92
```

TAKE A SAMPLE OF A DATASET TO MAKE STOCK PRICE PREDICTIONS USING THE LSTM MODEL:

So the first column in the '**x_train**' data set will contain values from the data set from index 0 to index 59 (60 values total) and the second column will contain values from the data set from index 1 to index 60 (60 values) and so on and so forth.

The '**y_train**' data set will contain the 61st value located at index 60 for it's first column and the 62nd value located at index 61 of the data set for it's second value and so on and so forth.

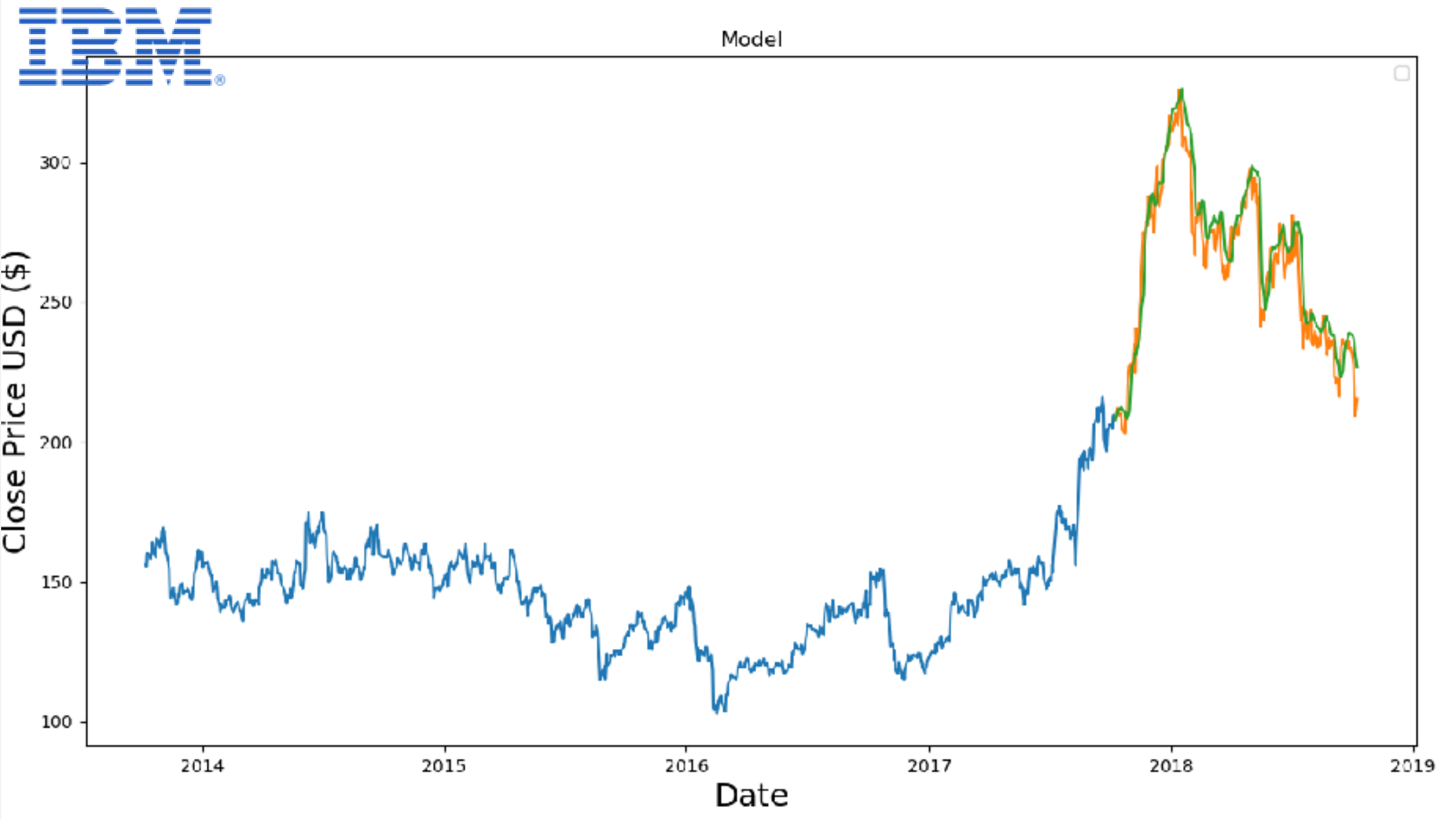
```
X_test=[]
for i in range(60,inputs_data.shape[0]):
    X_test.append(inputs_data[i-60:i,0])
X_test=np.array(X_test)
X_test=np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))
closing_price=lstm_model.predict(X_test)
closing_price=scaler.inverse_transform(closing_price)
```

VISUALIZE THE PREDICTED STOCK COSTS WITH ACTUAL STOCK COSTS:

```
train_data=new_dataset[:987]
valid_data=new_dataset[987:]
valid_data['Predictions']=closing_price

model_F_score(valid_data['Predictions'],valid_data['Close'])

plt.plot(train_data["Close"])
plt.plot(valid_data[['Close',"Predictions"]])
plt.legend()
plt.show()
```

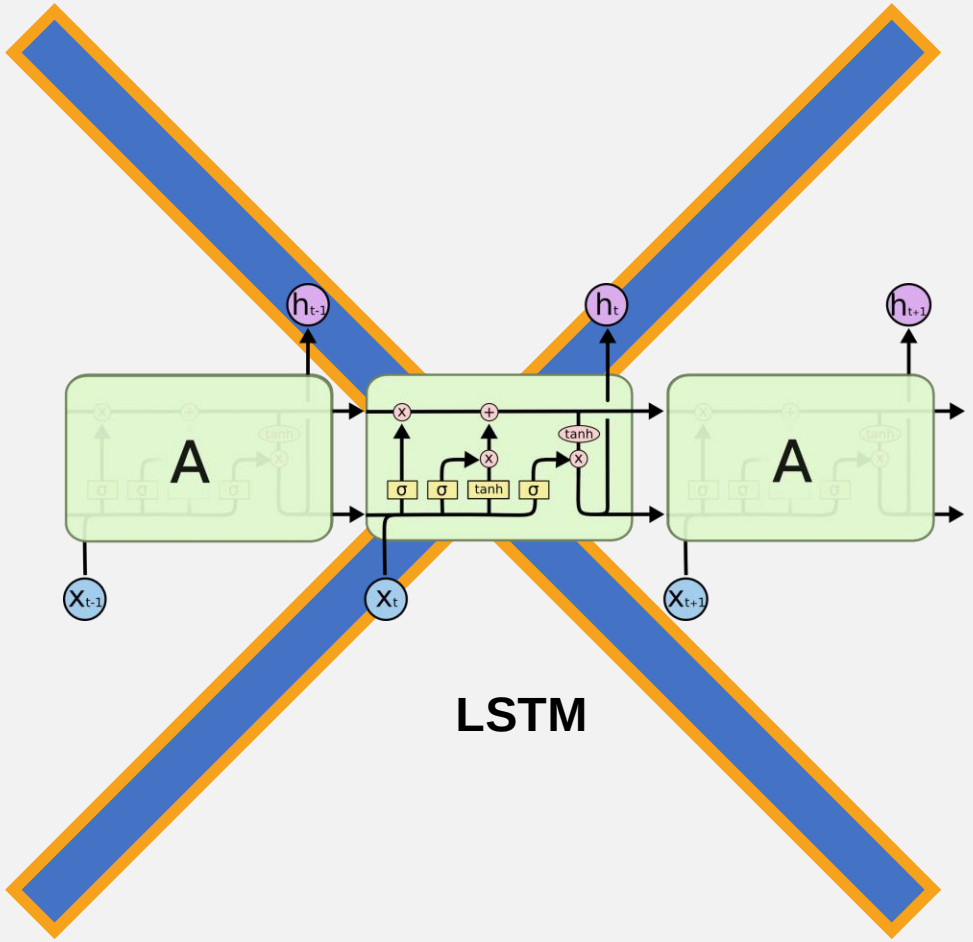

Model
Close Price USD ($)
300
250
200
150
100
2014
2015
2016
2017
2018
2019
Date

HOW LSTM WORKS ?

LSTM

LSTM (Long Short Term Memory)

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for **long periods of time** is practically their default behavior, not something they

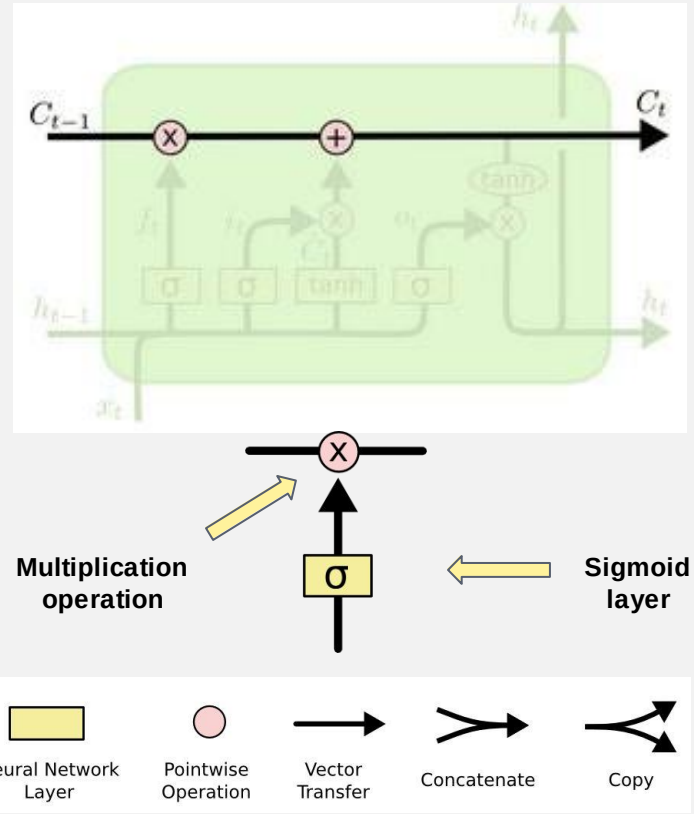


LSTM

HOW IT WORKS?

LSTM (How it works?)

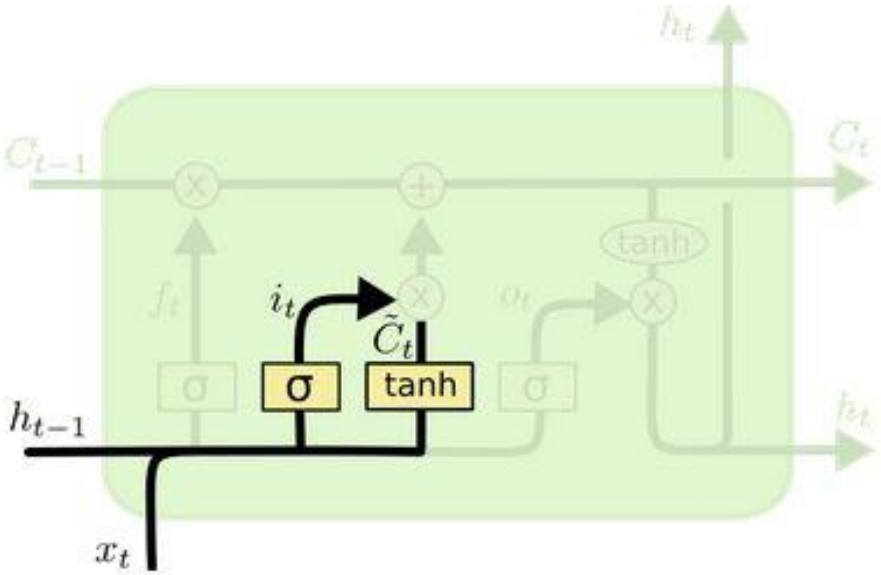
- The key to LSTM is the **Memory cell state** which stores the information. It runs straight down the **entire chain**.
- LSTM has the ability to **remove or add information** to these cell state, regulated by structures called **gates**.



HOW IT WORKS?

LSTM (How it works?)

- **First**, forget gate looks at h_{t-1} and x_t and outputs a number between 0 and 1.
- 1 represents "keep the information" and 0 represents "remove the information".
- **Second**, input gate decides which values will be updated, in order to do that a tanh layer creates a vector of C_t **(bar)**.
- Combining these two, create an update to the state.



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

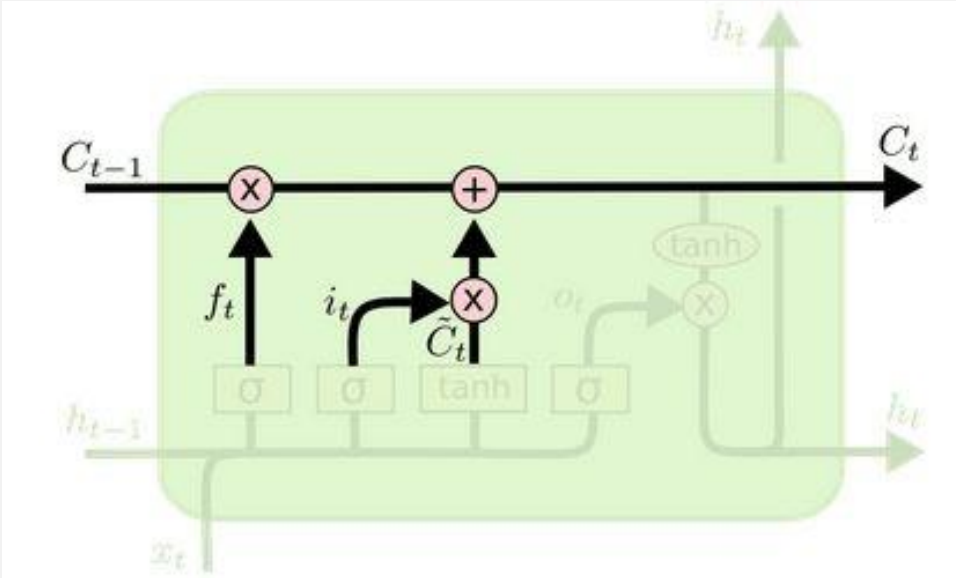
$$h_t = o_t * \tanh\left(C_t\right)$$

required for updating the weight

HOW IT WORKS?

LSTM (How it works?)

- **Third**, It's time to update the old cell C_{t-1} to C_t.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

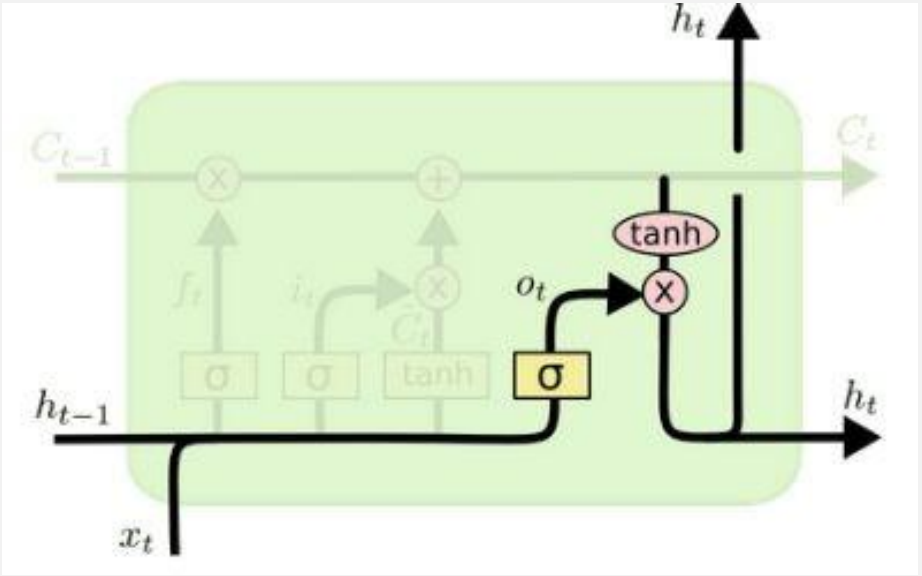
New Cell state

HOW IT WORKS?

LSTM (How it works?)

Fourth output will based on our cell.

A sigmoid layer will decides what parts of the cell state we're going to output.



Output Layer

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Current hidden layer information

THANK YOU