2. SYSTEM REQUIREMENTS

2.1 HARDWARE REQUIREMENTS

- **Processor -**Intel Core i5
- **System with CPU & GPU**
- **RAM -** 8GB or above

2.2 SOFTWARE REQUIREMENTS

- **Operating System -** Windows 10
- **Coding Language -** Python
- **Software -** Python IDEs, Google Colab, MS Excel.

3. LIBRARIES USED

3.1 NumPy - NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

3.2 Pandas - Pandas is an open-source library that is made mainly for working with relational or labelled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.

3.3 Matplotlib - Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

3.4 Sklearn - Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python.

It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn.

3.5 Keras – Keras is an Open Source Neural Network library written in Python that runs on top of Theano or Tensorflow. It is designed to be modular, fast and easy to use. It was developed by François Chollet, a Google engineer. Keras doesn't handle low-level computation. Instead, it uses another library to do it, called the "Backend.

Keras is high-level API wrapper for the low-level API, capable of running on top of TensorFlow, CNTK, or Theano. Keras High-Level API handles the way we make models, defining layers, or set up multiple input-output models. In this level, Keras also compiles our model with loss and optimizer functions, training process with fit function. Keras in Python doesn't handle Low-Level API such as making the computational graph, making tensors or other variables because it has been handled by the "backend" engine.

3.6 Tensorflow - TensorFlow is an open source framework developed by Google researchers to run machine learning, deep learning and other statistical and predictive analytics workloads. Like similar platforms, it's designed to streamline the process of developing and executing advanced analytics applications for users such as data scientists, statisticians and predictive modelers.

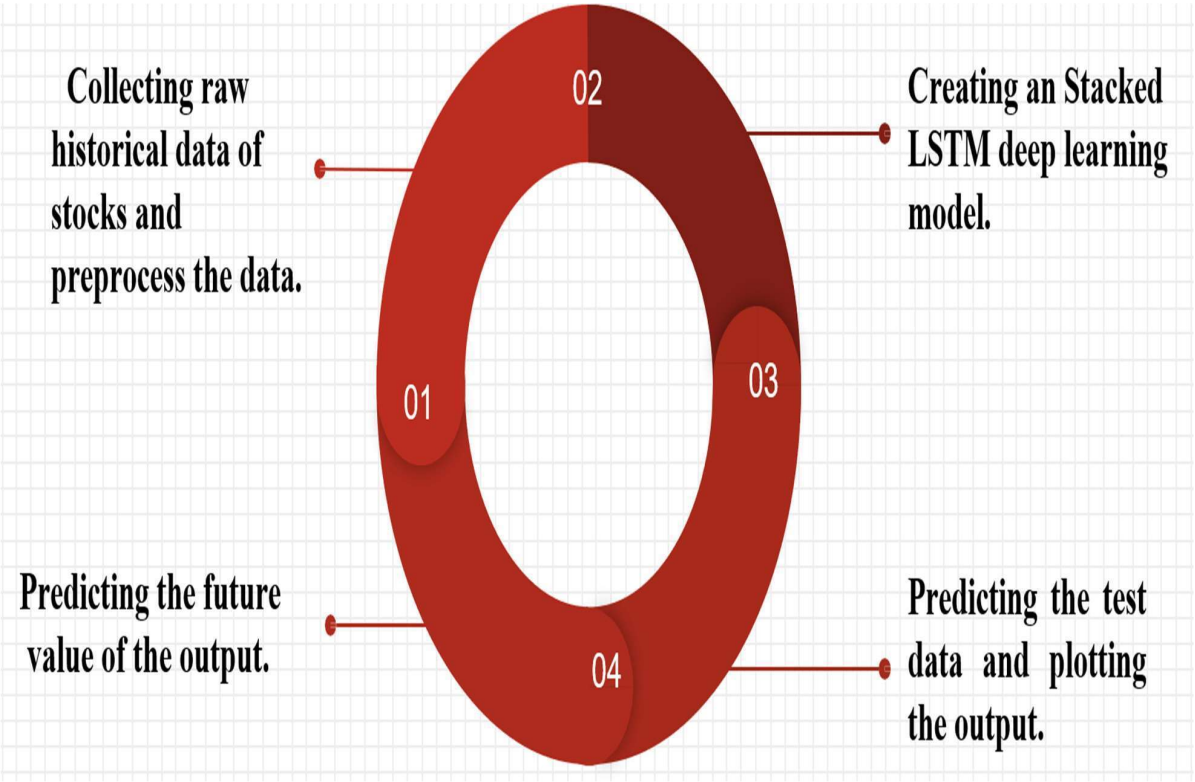
4. PROPOSED METHOD

Step-1: Collecting raw historical data of stocks and preprocess the data.

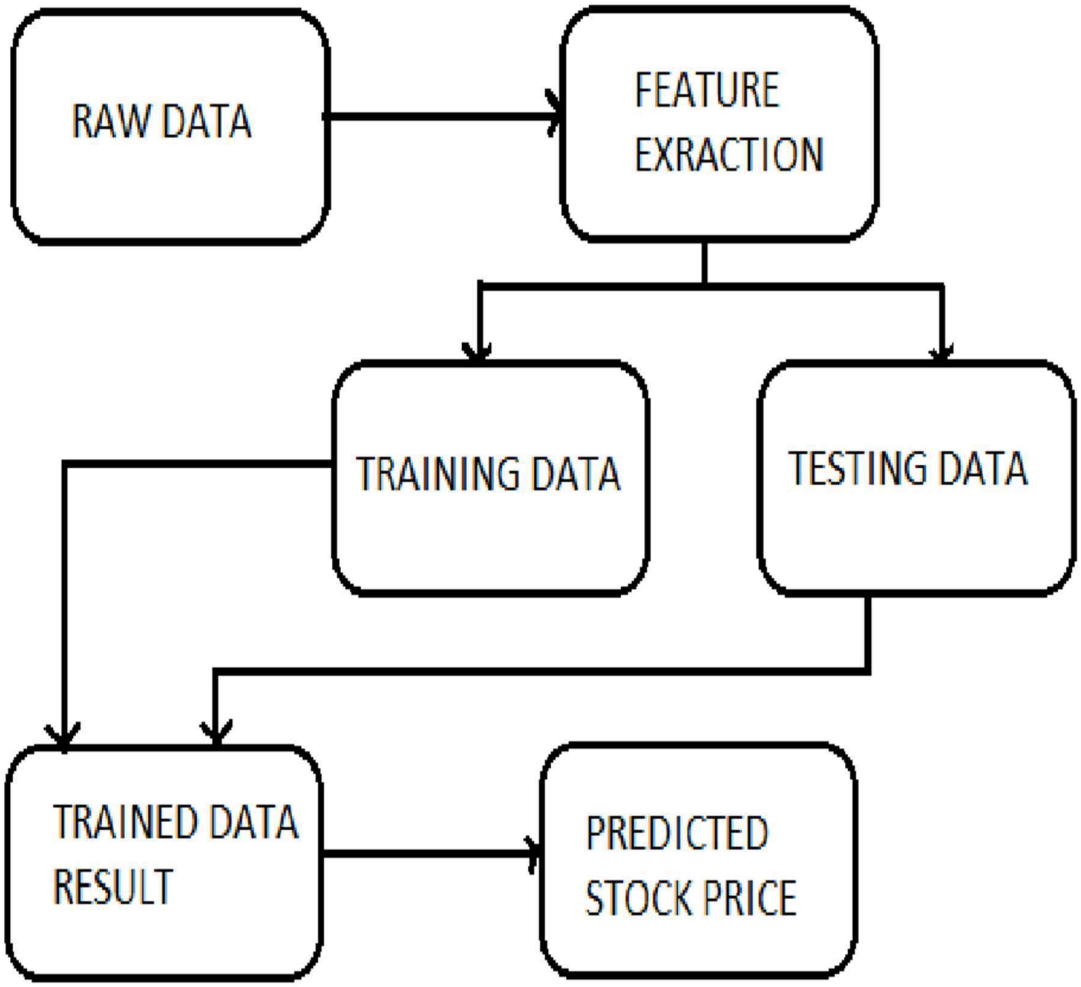
Step-2: Creating an Stacked LSTM deep learning model.

Step-3: Predicting the test data and plotting the output.

Step-4: Predicting the future value of the output.



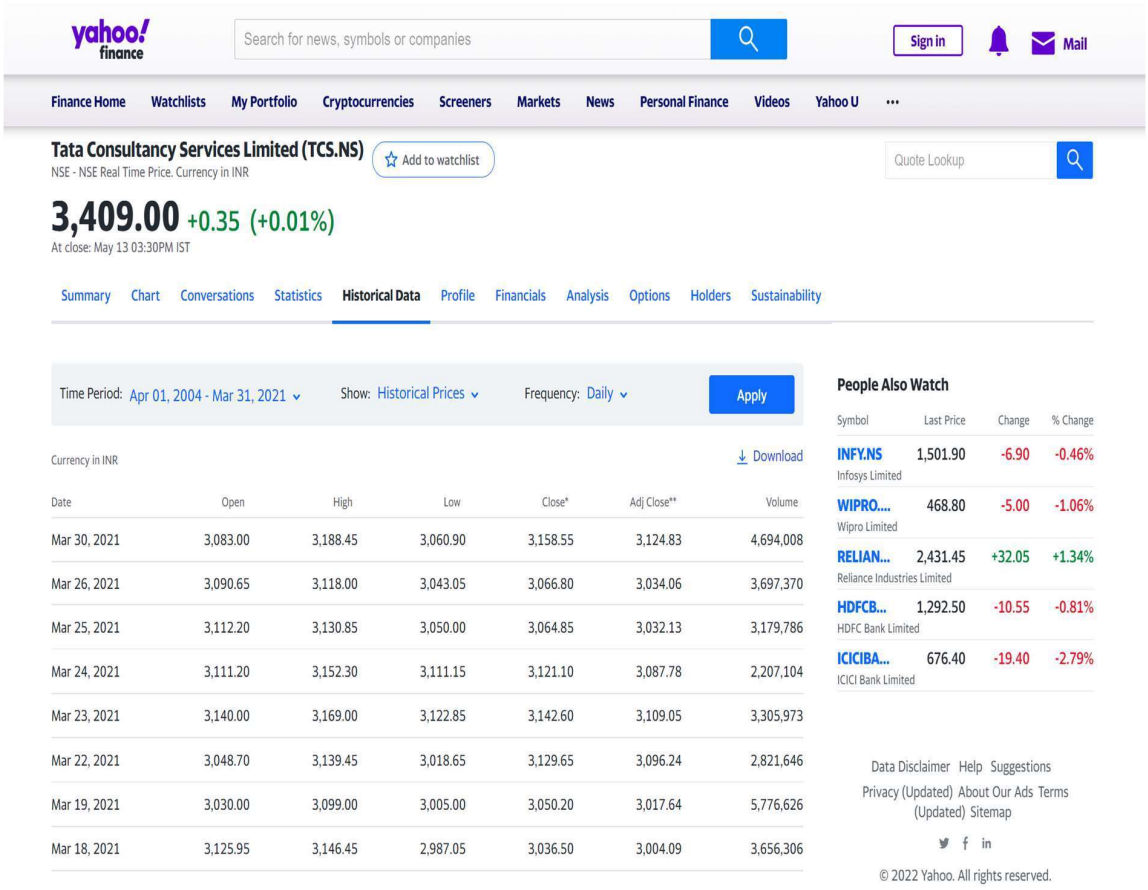
5. WORKFLOW PROCESS



6. IMPLEMENTATION

6.1 STEPS:

Step-1: Collecting historical data of various stocks using Yahoo finance. Each and every stock data was taken from 01-04-2004 to 31-03-2021.



yahoo! finance | Search for news, symbols or companies | Sign in | Mail

Finance Home | Watchlists | My Portfolio | Cryptocurrencies | Screeners | Markets | News | Personal Finance | Videos | Yahoo U | ...

Tata Consultancy Services Limited (TCS.NS)
NSE - NSE Real Time Price. Currency in INR
Add to watchlist | Quote Lookup

3,409.00 +0.35 (+0.01%)
At close: May 13 03:30PM IST

Summary | Chart | Conversations | Statistics | **Historical Data** | Profile | Financials | Analysis | Options | Holders | Sustainability

Time Period: Apr 01, 2004 - Mar 31, 2021 | Show: Historical Prices | Frequency: Daily | Apply

Currency in INR | Download

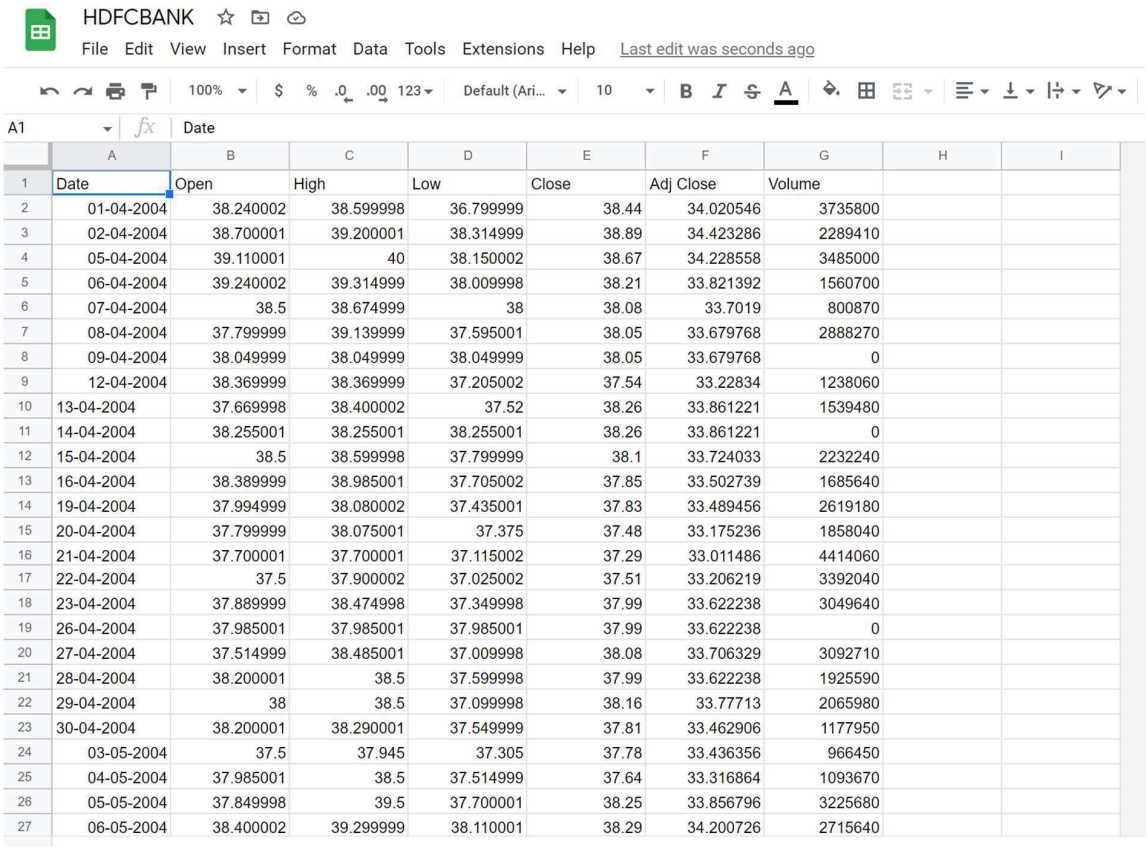
Date	Open	High	Low	Close*	Adj Close**	Volume
Mar 30, 2021	3,083.00	3,188.45	3,060.90	3,158.55	3,124.83	4,694,008
Mar 26, 2021	3,090.65	3,118.00	3,043.05	3,066.80	3,034.06	3,697,370
Mar 25, 2021	3,112.20	3,130.85	3,050.00	3,064.85	3,032.13	3,179,786
Mar 24, 2021	3,111.20	3,152.30	3,111.15	3,121.10	3,087.78	2,207,104
Mar 23, 2021	3,140.00	3,169.00	3,122.85	3,142.60	3,109.05	3,305,973
Mar 22, 2021	3,048.70	3,139.45	3,018.65	3,129.65	3,096.24	2,821,646
Mar 19, 2021	3,030.00	3,099.00	3,005.00	3,050.20	3,017.64	5,776,626
Mar 18, 2021	3,125.95	3,146.45	2,987.05	3,036.50	3,004.09	3,656,306

People Also Watch

Symbol	Last Price	Change	% Change
INFY.NS Infosys Limited	1,501.90	-6.90	-0.46%
WIPRO.... Wipro Limited	468.80	-5.00	-1.06%
RELIAN... Reliance Industries Limited	2,431.45	+32.05	+1.34%
HDFCB... HDFC Bank Limited	1,292.50	-10.55	-0.81%
ICICIBA... ICICI Bank Limited	676.40	-19.40	-2.79%

Data Disclaimer | Help | Suggestions | Privacy (Updated) | About Our Ads | Terms (Updated) | Sitemap

Step-2: Preprocessing the dataset and we then remove null values in the csv file.



HDFCBANK

File Edit View Insert Format Data Tools Extensions Help Last edit was seconds ago

A1 | Date

	A	B	C	D	E	F	G	H	I
1	Date	Open	High	Low	Close	Adj Close	Volume		
2	01-04-2004	38.240002	38.599998	36.799999	38.44	34.020546	3735800		
3	02-04-2004	38.700001	39.200001	38.314999	38.89	34.423286	2289410		
4	05-04-2004	39.110001	40	38.150002	38.67	34.228558	3485000		
5	06-04-2004	39.240002	39.314999	38.009998	38.21	33.821392	1560700		
6	07-04-2004	38.5	38.674999	38	38.08	33.7019	800870		
7	08-04-2004	37.799999	39.139999	37.595001	38.05	33.679768	2888270		
8	09-04-2004	38.049999	38.049999	38.049999	38.05	33.679768	0		
9	12-04-2004	38.369999	38.369999	37.205002	37.54	33.22834	1238060		
10	13-04-2004	37.669998	38.400002	37.52	38.26	33.861221	1539480		
11	14-04-2004	38.255001	38.255001	38.255001	38.26	33.861221	0		
12	15-04-2004	38.5	38.599998	37.799999	38.1	33.724033	2232240		
13	16-04-2004	38.389999	38.985001	37.705002	37.85	33.502739	1685640		
14	19-04-2004	37.994999	38.080002	37.435001	37.83	33.489456	2619180		
15	20-04-2004	37.799999	38.075001	37.375	37.48	33.175236	1858040		
16	21-04-2004	37.700001	37.700001	37.115002	37.29	33.011486	4414060		
17	22-04-2004	37.5	37.900002	37.025002	37.51	33.206219	3392040		
18	23-04-2004	37.889999	38.474998	37.349998	37.99	33.622238	3049640		
19	26-04-2004	37.985001	37.985001	37.985001	37.99	33.622238	0		
20	27-04-2004	37.514999	38.485001	37.009998	38.08	33.706329	3092710		
21	28-04-2004	38.200001	38.5	37.599998	37.99	33.622238	1925590		
22	29-04-2004	38	38.5	37.099998	38.16	33.77713	2065980		
23	30-04-2004	38.200001	38.290001	37.549999	37.81	33.462906	1177950		
24	03-05-2004	37.5	37.945	37.305	37.78	33.436356	966450		
25	04-05-2004	37.985001	38.5	37.514999	37.64	33.316864	1093670		
26	05-05-2004	37.849998	39.5	37.700001	38.25	33.856796	3225680		
27	06-05-2004	38.400002	39.299999	38.110001	38.29	34.200726	2715640		

Step-3: Loading the dataset in Google colab.

```
[1] import pandas as pd
```

```
[2] df=pd.read_csv('/content/HDFCBANK.csv')
```

```
[3] df.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	01-04-2004	38.240002	38.599998	36.799999	38.44	34.020546	3735800
1	02-04-2004	38.700001	39.200001	38.314999	38.89	34.423286	2289410
2	05-04-2004	39.110001	40.000000	38.150002	38.67	34.228558	3485000
3	06-04-2004	39.240002	39.314999	38.009998	38.21	33.821392	1560700
4	07-04-2004	38.500000	38.674999	38.000000	38.08	33.701900	800870

```
[4] df1=df.reset_index()['Close']
```

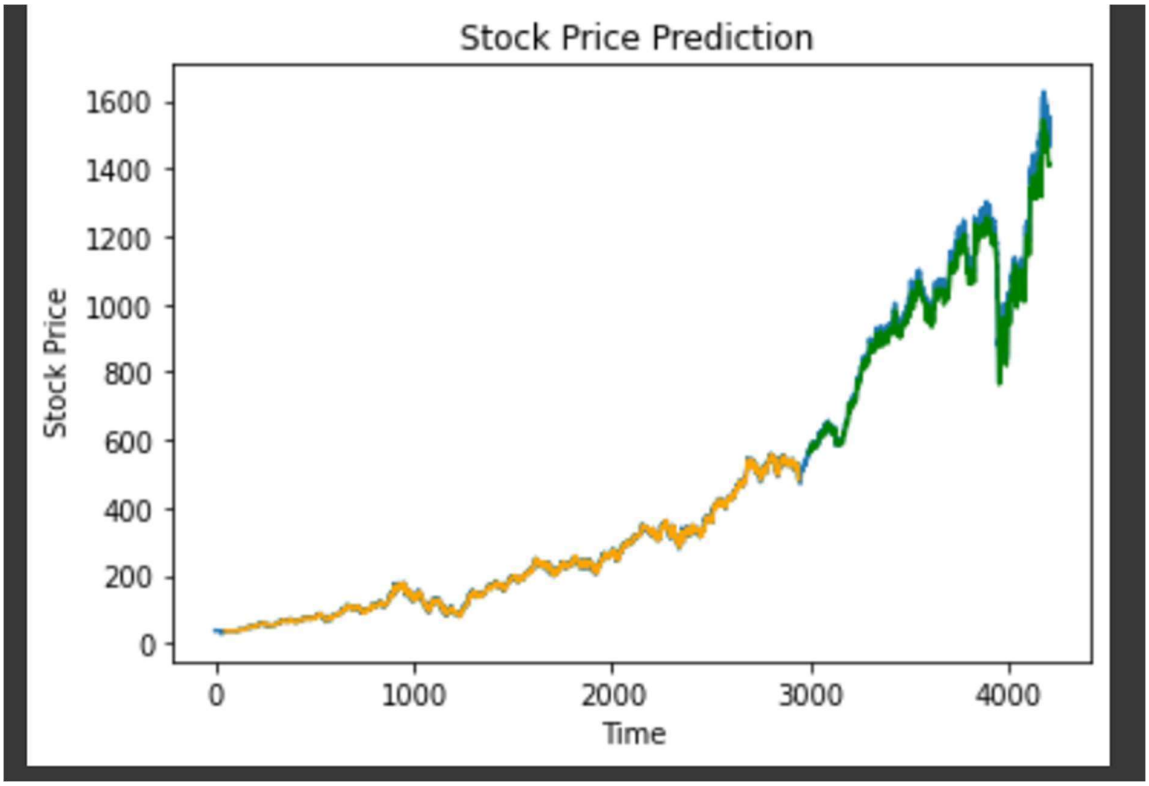
Step-4: Now we split the dataset into training and testing data. 70% is for training and 30% is for testing.

```
[10] training_size=int(len(df1)*0.70)
     test_size=len(df1)-training_size
     train_data,test_data=df1[0:training_size,:],df1[training_size:len(df1),:1]

[11] training_size,test_size

     (2943, 1262)
```

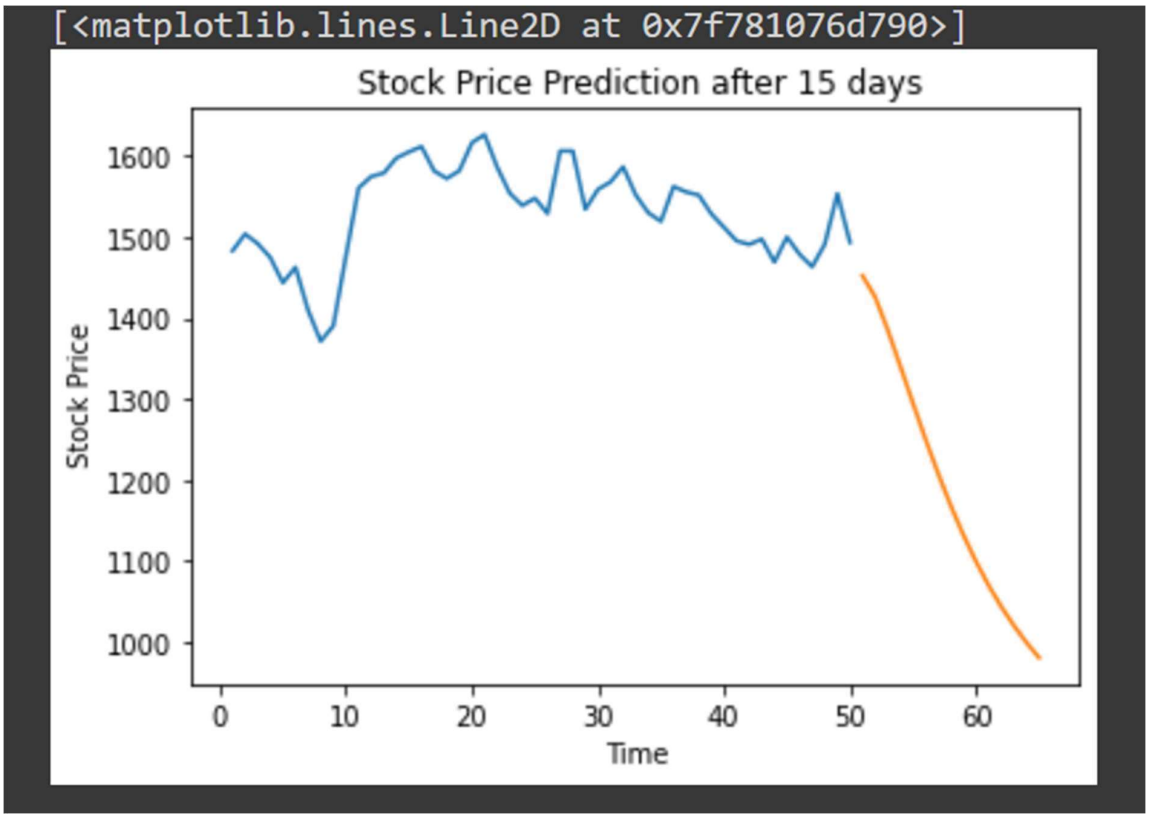
Step-5: We train the model using tensorflow.keras library.

```
[17] X_train =X_train.reshape(X_train.shape[0],X_train.shape[1] , 1)
     X_test = X_test.reshape(X_test.shape[0],X_test.shape[1] , 1)

     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Dense
     from tensorflow.keras.layers import LSTM

[19] model=Sequential()
     model.add(LSTM(50,return_sequences=True,input_shape=(k,1)))
     model.add(LSTM(50,return_sequences=True))
     model.add(LSTM(50))
     model.add(Dense(1))
     model.compile(loss='mean_squared_error',optimizer='adam')

     model.summary()

     model.fit(X_train,y_train,validation_data=(X_test,ytest),epochs=100,batch_size=64,verbose=1)

     Epoch 1/100
     46/46 [==============================] - 13s 73ms/step - loss: 0.0019 - val_loss: 0.0034
     Epoch 2/100
     46/46 [==============================] - 2s 44ms/step - loss: 5.0798e-05 - val_loss: 0.0046
     Epoch 3/100
     46/46 [==============================] - 2s 44ms/step - loss: 4.6509e-05 - val_loss: 0.0046
     Epoch 4/100
     46/46 [==============================] - 2s 44ms/step - loss: 4.7446e-05 - val_loss: 0.0043
     Epoch 5/100
     46/46 [==============================] - 2s 44ms/step - loss: 4.4929e-05 - val_loss: 0.0050
     Epoch 6/100
     46/46 [==============================] - 2s 44ms/step - loss: 4.5328e-05 - val_loss: 0.0049
```

Step-6: Now test model and plotted output is obtained.



Step-7: Now we predict the future values.



Step-8: We then continue the same process for all the remaining stocks.

6.2 DATASET

In this project we used data of four stocks i.e; AXISBANK, ONGC, TCS, MARUTI and NIFTY index. Stocks data range from FY2005 to FY2021 and index data from FY2008 to FY2021. Due to covid we also consider the data of above stocks and index from YR2020 to YR2021. The data is collected in .csv format.

6.3 PARAMETERS USED

Parameter Used	Meaning
Date	Date of stock price
Open	Open price of a share
Close	Closing price of a share
Volume/ trade quantity	Number of shares traded
High	Highest share value for the day
Low	Lowest share value for the day
Turnover	Total Turnover of the share