# “SUMMER INTERNSHIP PROJECT REPORT”

## Infosys Stock Prediction

## Submitted By

ABHINOV PHUKON

ROLL NO. 172010007003

B.E COMPUTER SCIENCE

Academy Session- 2017-2021


**Barak Valley Engineering College**

**Nirala, Karimganj - 788701, Assam, INDIA**

**September 2020**

# Preface:

The prediction of the future has always fascinated mankind due to the possible benefits of this knowledge (Thierry, 1996). This is especially true in the financial world. New tools and techniques for prediction are growing away from their original environment like the mathematical and computing world and find their way into all kinds of professional applications such as finance or engineering (B.K. Wong, 1995). Here we are trying to predict Infosys (INFY) stock price Prediction.

# Content:

# Abstract

Predicting stock market prices is a complex task that traditionally involves extensive human-computer interaction. Due to the correlated nature of stock prices, conventional batch processing methods cannot be utilized efficiently for stock market analysis. We propose an online learning algorithm that utilizes a kind of recurrent neural network (RNN) called Long Short Term Memory (LSTM), where the weights are adjusted for individual data points using stochastic gradient descent. This will provide more accurate results when compared to existing stock price prediction algorithms. The network is trained and evaluated for accuracy with various sizes of data, and the results are tabulated. A comparison with respect to accuracy is then performed against an Artificial Neural Network.

# Introduction

There are a lot of complicated financial indicators and also the fluctuation of the stock market is highly violent. However, as the technology is getting advanced, the opportunity to gain a steady fortune from the stock market is increased and it also helps experts to find out the most informative indicators to make a better prediction. The prediction of the market value is of great importance to help in maximizing the profit of stock option purchase while keeping the risk low. Recurrent neural networks (RNN) have proved one of the most powerful models for processing sequential data. Long Short-Term memory is one of the most successful RNNs architectures. LSTM introduces the memory cell, a unit of computation that replaces traditional artificial neurons in the hidden layer of the network. With these memory cells, networks are able to effectively associate memories and input remote in time, hence suit to grasp the structure of data dynamically over time with high prediction capacity.

# Stock Price Prediction Using LSTM Recurrent Neural Network

Long-Short-Term Memory Recurrent Neural Network belongs to the family of deep learning algorithms. It is a recurrent network because of the feedback connections in its architecture. It has an advantage over traditional neural networks due to its capability to process the entire sequence of data. Its architecture comprises the *cell*, *input gate*, *output gate* and *forget gate*.

The cell remembers values over arbitrary time intervals, and the three gates regulate the flow of information into and out of the cell. The cell of the model is responsible for keeping track of the dependencies between the elements in the input sequence. The input gate controls the extent to which a new value flows into the cell, the forget gate controls the extent to which a value remains in the cell, and the output gate controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit.

However, there are some variants of the LSTM model such as Gated Recurrent Units (GRUs) that do not have the output gate. LSTM Networks are popularly used on time-series data for classification, processing, and making predictions. The reason for its popularity in time-series application is that there can be several lags of unknown duration between important events in a time series.

# Stock Prediction

In this task, the stock price of Infosys (INFY) is predicted using the LSTM Recurrent Neural Network. Our task is to predict stock prices for a few days, which is a time series problem. The LSTM model is very popular in time-series forecasting, and this is the reason why this model is chosen in this task.

This data set contains 5141 observations. After preprocessing, only dates and Open column are taken, as these columns have main significance in the dataset. The LSTM model is trained on this entire dataset, and for the testing purpose, a new dataset is fetched. The stock prices for this new duration will be predicted by the already trained LSTM model, and the predicted prices will be plotted against the original prices to visualize the model's accuracy.

# Implementation

## *First we need to import necessary libraries*

**Importing Required Libraries**

In [27]:
```
import pandas as pd
import numpy as np
import datetime as dt
from datetime import datetime

import matplotlib.pyplot as plt

import numpy as np

from sklearn.preprocessing import MinMaxScaler

### Create the Stacked LSTM model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
from keras.layers import Dropout
```

## *Then we load the data set*

In [28]:
```
#read the file
df = pd.read_csv("/content/datasets_423609_1456925_INFY.csv")

#print the head
df.head()
```

Out[28]:

| | Date | Symbol | Series | Prev Close | Open | High | Low | Last | Close | VWAP | Volume | Turnover | Trades | Deliverable Volume | %Deliverble |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2000-01-03 | INFOSYSTCH | EQ | 14467.75 | 15625.00 | 15625.20 | 15625.00 | 15625.20 | 15625.20 | 15625.18 | 5137 | 8.026657e+12 | NaN | NaN | NaN |
| 1 | 2000-01-04 | INFOSYSTCH | EQ | 15625.20 | 16800.00 | 16875.25 | 16253.00 | 16875.25 | 16855.90 | 16646.38 | 56186 | 9.352937e+13 | NaN | NaN | NaN |
| 2 | 2000-01-05 | INFOSYSTCH | EQ | 16855.90 | 15701.00 | 16250.00 | 15507.45 | 15507.45 | 15507.45 | 15786.38 | 164605 | 2.598516e+14 | NaN | NaN | NaN |
| 3 | 2000-01-06 | INFOSYSTCH | EQ | 15507.45 | 15256.65 | 15300.00 | 14266.85 | 14266.85 | 14266.85 | 14462.82 | 81997 | 1.185908e+14 | NaN | NaN | NaN |
| 4 | 2000-01-07 | INFOSYSTCH | EQ | 14266.85 | 13125.50 | 13125.50 | 13125.50 | 13125.50 | 13125.50 | 13125.50 | 7589 | 9.960942e+12 | NaN | NaN | NaN |

## *Preprocessing and feature extraction*

```
df = df['Open'].values
df = df.reshape(-1, 1)
print(df.shape)
df[:5]
```

```
(5141, 1)
```

Out[4]:
```
array([[15625.  ],
       [16800.  ],
       [15701.  ],
       [15256.65],
       [13125.5 ]])
```

*Now we spilt them into training and testing set 80% train and 20 %test and just subtracting 50 so that it overlap and we can visualize easier. We will use the min/max scalar to scale our data between zero and Run first of all we will fit and transfer our training data and we'll print the first five rows.*

*Then we'll transform our data with the prefix scaler. Next off we will create a function which we can use to create datasets as an argument df*

*We will create our training data so we call creates data set with our data sets trained and then we will save it in x_train and y_train and variable and we are going to put the first row of the x .Reshape features for LSTM Layer*

## *Building Model*

In [38]:
```
# create and fit the LSTM network
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(x_train.shape[1],1)))
model.add(Dropout(rate = 0.2))

model.add(LSTM(units=50, return_sequences = True))
model.add(Dropout(rate = 0.2))

model.add(LSTM(units=50, return_sequences = True))
model.add(Dropout(rate = 0.2))

model.add(LSTM(units=50, return_sequences = False))
model.add(Dropout(rate = 0.2))

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

## *Compile the model*

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

Compiling the model with **Stochastic Gradient Descent** algorithm

Using Mean Squared Error as Loss Function

In [39]:
```
model.compile(loss='mean_squared_error', optimizer='adam')
```

*Check whether we save the model or not*

```
In [14]: #check whetert we save the model or not

if(not os.path.exists('stock_prediction.h5')):
    model.fit(x_train, y_train, epochs=50, batch_size=32)
    model.save('stock_prediction.h5')
```

*Load the model*

```
In [15]: #load the model

model = load_model('stock_prediction.h5')
```

*Then we Plot the actual and predicted price and at last we print the predictions.*

```
In [46]: X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
         preds = model.predict(X_test)
         preds = scaler.inverse_transform(preds)
```

```
In [47]: preds
```

```
Out[47]: array([[1289.8188 ],
                [1303.6515 ],
                [1308.7808 ],
                [1311.2748 ],
                [1307.949  ],
                [1296.3668 ],
                [1284.7076 ],
                [1277.8086 ],
                [1273.4635 ],
                [1259.9264 ],
                [1248.3718 ],
                [1251.974  ],
                [1250.0455 ],
                [1242.9579 ],
                [1235.1604 ],
                [1236.6605 ],
                [1246.135  ],
                [1255.4777 ],
                [1262.39   ],
```

# conclusion

Of course the model we have proposed here using Stock Prediction using Recurrent Neural Network is not very advance but we can improve the model using deep learning techniques and some technical indicators (like RSI, MACD, Moving Averages, Bollinger's Band etc. ) for more accurate results.

We can use self recurrent neural network architecture to comprise globally recurrent neurons and both adaptive and sliding mode approach can be used for forecasting and classification of time series databases. An evolving recurrent fuzzy neural network combined with FLANN can also be considered as an alternative model for stock market trend and volatility prediction. This can be further expanded to a type-2 recurrent fuzzy neural model for making trading decision like buying, selling or holding stocks. Although there has been some study regarding trend analysis and trading of stock market indices; a detailed study is, however, required for developing an intelligence system for making decisions regarding buying and selling of stocks and portfolios. In this regard it is suggested to develop both Type-1 and Type-2 fuzzy logic systems and effective learning procedures for taking accurate decisions in stock market trading. Some forecasting procedures have been suggested for day ahead electricity price in deregulated electrical markets. However, it is suggested to explore the possibility of using both support vector and extreme learning machine regressions for more effective prediction of spikes in the electricity market price. This will be required by the electricity producers and brokers of electricity markets for accruing profits.