**“****PREDICTION OF STOCK PRICES USING NEURAL NETWORKS(LSTM)”**

**By**

**BADRI NARAAYANAN M**

Roll No: 2019201015

MBA (2019-2021)

**A PROJECT REPORT**

Submitted to the

**FACULTY OF MANAGEMENT STUDIES**

In partial fulfilment for the award of the degree

Of

**MASTER OF BUSINESS ADMINISTRATION**



**COLLEGE OF ENGINEERING, GUINDY**

**ANNA UNIVERSITY CHENNAI 600025**

**MAY 2021**

**BONAFIDE CERTIFICATE**

Certified that the Project report titled **“PREDICTION OF CLOSING PRICE OF STOCK BASED ON LSTM NEURAL NETWORKS”** is the bonafide work of **BADRI NARAAYANAN M (2019201015)** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**Signature of Guide**

**Dr. R. MAGESH Dr. S.N. GEETHA**

Professor and HOD Professor

Department of Management Studies Department of management studies

ANNA UNIVERSITY, CEG ANNA UNIVERSITY, CEG

GUINDY, Chennai. GUINDY, Chennai.

**Ⅰ**

**ABSTRACT**

Predicting stock prices is one of the most challenging tasks in the Stock Market with the real money at stake, and answerable to investors and other stakeholders. It is more complicated because of the volatile and non-linear nature of the stock markets. Stock markets are there in existence since 1602 AD, when the Dutch East India Company started Amsterdam Stock Exchange to deal with printed stocks and bonds. A typical stock analysis is done in two ways – fundamental and technical. Fundamental Analysis deals with analysing the qualitative aspects like management, industry, economy to forecast the stock performance, whereas technical analysis involves predicting the price movements based on past statistics.

In this paper, we are limiting our scope to predict the price movements for short term only using past data. We will do a linear regression to estimate the closing price of a stock and then predict the stock price using LSTM Neural Networks using past data. We collect ten years of data of five different stocks of five different industries from NSE and we compare the performance of linear regression and neural network.

**Acknowledgement**

I wish to express my heart-felt gratitude to our Head of the Department of Management Studies, **Dr.M.MAGESH** for his extensive support and encouragement to complete the project.

I take this opportunity to express my profound gratitude to my project guide

**Dr .S.N.Geetha, Professor**, Department of Management Studies, for her valuable guidance and useful suggestions, without which this internship project would not have been completed successfully.

I am very thankful to Ricabchand & Co. for giving me an opportunity to do the project in their organization.

Finally, I thank all my family members, friends, well-wishers for extending their support to me throughout the project.

I also sincerely express my gratitude to the almighty for giving me the strength and intelligence to complete my study.

* **Badri Naraayanan.M**

**Table of Contents**

[INTRODUCTION 8](#_Toc72011698)

[Industry Profile 8](#_Toc72011699)

[RESEARCH METHODOLOGY 9](#_Toc72011700)

[Statement of the Problem 9](#_Toc72011701)

[Objectives of the Problem 9](#_Toc72011702)

[Need for the study 9](#_Toc72011703)

[Scope of the study 9](#_Toc72011704)

[Research Gap 9](#_Toc72011705)

[Limitation 9](#_Toc72011706)

[Research Design 10](#_Toc72011707)

[REVIEW OF LITERATURE 11](#_Toc72011708)

[DATA ANALYSIS AND INTERPRETATION 15](#_Toc72011709)

[Data Preprocessing 15](#_Toc72011710)

[Data Model Training 18](#_Toc72011711)

[Predictions 19](#_Toc72011712)

[Multiple Linear Regression 19](#_Toc72011713)

[FINDINGS 35](#_Toc72011714)

[ANNEXURE 36](#_Toc72011715)

[Code 36](#_Toc72011716)

[REFERENCES 59](#_Toc72011717)

**List of Tables**

[Table 1: Correlation Co-efficient > 0.5 (Airtel) 20](#_Toc71995275)

[Table 2: Correlation Co-efficient > 0.5 (HDFC) 20](#_Toc71995276)

[Table 3: Correlation Co-efficient > 0.5 (India Cem) 20](#_Toc71995277)

[Table 4: Correlation Co-efficient > 0.5 (Sun Pharma) 21](#_Toc71995278)

[Table 5: Correlation Co-efficient > 0.5 (TCS) 21](#_Toc71995279)

**List of Figures**

[Figure 1: NIFTY Data 8](#_Toc72004848)

[Figure 2: Research Workflow 19](#_Toc72004849)

[Figure 3: Multiple Linear Regression 22](#_Toc72004850)

[Figure 4: LSTM 22](#_Toc72004851)

[Figure 5: Multiple Linear Regression - Actual Vs Prediction (Airtel) 23](#_Toc72004852)

[Figure 6: Multiple Linear Regression - Actual Vs Prediction (HDFC) 24](#_Toc72004853)

[Figure 7: Multiple Linear Regression - Actual Vs Prediction (India Cements) 25](#_Toc72004854)

[Figure 8: Multiple Linear Regression - Actual Vs Prediction (Sun Pharma) 26](#_Toc72004855)

[Figure 9: Multiple Linear Regression - Actual Vs Prediction (TCS) 27](#_Toc72004856)

[Figure 10: LSTM Neural Network - Actual Vs Prediction (Airtel) 28](#_Toc72004857)

[Figure 11: LSTM Neural Network - Actual Vs Prediction (HDFC) 29](#_Toc72004858)

[Figure 12: LSTM Neural Network - Actual Vs Prediction (India Cements) 30](#_Toc72004859)

[Figure 13: LSTM Neural Network - Actual Vs Prediction (Sun Pharma) 31](#_Toc72004860)

[Figure 14: LSTM Neural Network - Actual Vs Prediction (TCS) 32](#_Toc72004861)

[Figure 15: LSTM Neural Network (improved) - Actual Vs Prediction (Airtel) 33](#_Toc72004862)

[Figure 16: LSTM Neural Network (improved) - Actual Vs Prediction (HDFC) 34](#_Toc72004863)

[Figure 17: LSTM Neural Network (improved) - Actual Vs Prediction (India Cem) 35](#_Toc72004864)

[Figure 18: LSTM Neural Network (improved) - Actual Vs Prediction (Sun Pharma) 36](#_Toc72004865)

[Figure 19: LSTM Neural Network (improved) - Actual Vs Prediction (TCS) 36](#_Toc72004866)

# INTRODUCTION

## Industry Profile

Stock markets are there in existence since 1602 AD, when the Dutch East India Company started Amsterdam Stock Exchange to deal with printed stocks and bonds. Bombay Stock Exchange (BSE) and National Stock Exchange (NSE) are the most prominent stock exchanges in India. Sensex and NIFTY are the corresponding indices of these stock markets.

Generally, stock market indices are the forecast indicators of an economy.

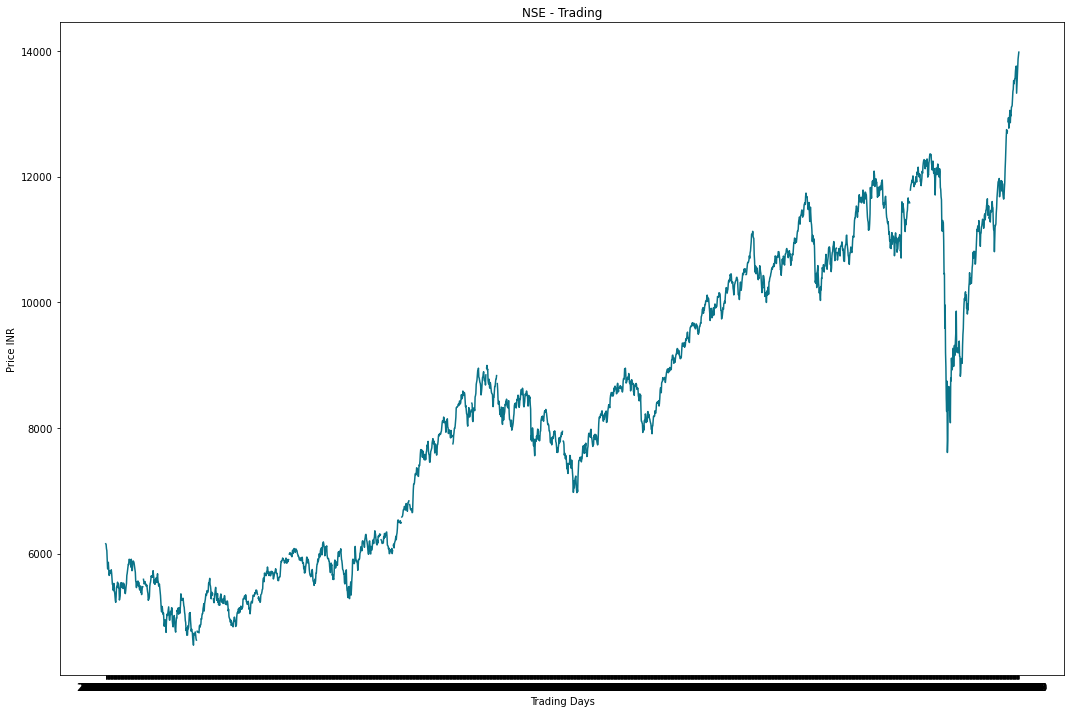


Figure 1: NIFTY Data

For training the model to predict the stock prices, we collect the data of stocks of 5 different industries.

1. IT - TCS
2. Cement – India Cements
3. Telecommunication - Airtel
4. Pharmaceutical – Sun Pharma
5. Banking – HDFC

# RESEARCH METHODOLOGY

## Statement of the Problem

Predicting stock market prices is one of the challenging tasks in the stock market. The high volatility and non-linear nature of the movement of the indices makes it more complicated.

## Objectives of the Problem

* To find the daily closing price of a stock in stock market by studying the existing patterns using Recurrent Neural networks and study the effectiveness of the measurement.

## Need for the study

* Prediction of stock prices more accurately is a more challenging task which requires extensive analysis, (fundamental and technical).
* Using machine learning and AI techniques, can open new doors and broaden the scope of predicting values as close as possible, as these techniques are already extensively used in identifying patterns in many fields such as image processing, voice recognition etc.,

## Scope of the study

In this study, we restrict ourselves to the technical analysis assuming the price itself comprises information of all the factors (both qualitative and quantitative).

## Research Gap

None.

## Limitation

We need to train the model for each stock separately as the price variation differs for each stock of each industry due to fluctuations caused due to systematic as well as unsystematic factors. Also, we are analysing on the data for only 10 years from Jan 2011 to Dec 2020.

## Research Design

The data required for this study is downloaded from NSE website.

Preliminary analysis is done on the data and additional features are manipulated like moving average and std deviations for the closing price.

Multiple Linear Regression of closing price is implemented on the features that best correlates with the closing price.

LSTM Neural network is implemented with the selected features of the training data.

Performance is measured using RMSE for each stock.

# REVIEW OF LITERATURE

* *Title*: Stock Closing Price Prediction using Machine Learning Techniques
* *Magazine*: Procedia Computer Science (Science Direct)
* *URL*: <https://www.sciencedirect.com/science/article/pii/S1877050920307924>
* *Description*: Accurate prediction of stock market returns is a very challenging task due to volatile and non-linear nature of the financial stock markets. With the introduction of artificial intelligence and increased computational capabilities, programmed methods of prediction have proved to be more efficient in predicting stock prices. The next day closing price is predicted for five companies belonging to different sectors of operation using Artificial Neural Network and Random Forest techniques. The financial data: Open, High, Low and Close prices of stock are used for creating new variables which are used as inputs to the model. The models are evaluated using standard strategic indicators: RMSE and MAPE.

----------------------------------------------------2--------------------------------------------------

* *Title*: Forecasting Market Price of Stock using Artificial Neural Network
* *Magazine*: International Journal of Computer Applications
* *URL*: <https://www.ijcaonline.org/research/volume124/number12/murkute-2015-ijca-905681.pdf>
* *Description*: Decision making in a stock market is not easy as it involves price trends, market nature, company's stability, different rumours, brand image, venture capitalist funds etc. methods like technical analysis, time series analysis and statistical analysis are an attempt to predict the price but unfortunately none of these methods are a consistently acceptable tool. Hence artificial neural network i.e. a field of Artificial Intelligence is a desired way to discover unknown and hidden patterns of the data. Back propagation algorithm is used to training session and Multilayer feed forward network is a network model for predicting price accordingly. This prediction would be done on various parameters that would be considered as input to the multilayer perceptron model.

-------------------------------------------------3-------------------------------------------------

* *Title*: Price Prediction of Share Market Using Artificial Neural Network 'ANN'
* *Magazine*: International Journal of Computer Applications
* *URL*: <https://www.researchgate.net/publication/226159428_Price_Prediction_of_Share_Market_Using_Artificial_Neural_Network_'ANN'/link/00b7d51643a469e06b000000/download>
* *Description*: Artificial Neural Network (ANN), a field of Artificial Intelligence (AI), is a popular way to identify unknown and hidden patterns in data which is suitable for share market prediction. For predicting of share price using ANN, there are two modules, one is training session and other is predicting price based on previously trained data. We used Backpropagation algorithm for training session and Multilayer Feedforward network as a network model for predicting price.

-----------------------------------------------4---------------------------------------------------

* *Title*: Parameters for Stock Market Prediction
* *Magazine*: International Journal of Computer Applications, Volume 4(2), 2013
* *URL*: https://www.ijsr.net/archive/v6i4/ART20172755.pdf
* *Description*:

Recurrent neural networks (RNN) have been proved one of the most powerful models for processing sequential data. Long Short-Term memory is one of the most successful RNNs architectures. The code is implemented in Keras.

--------------------------------------------------5----------------------------------------------------

* *Title*: Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling
* *Magazine*: Google, USA
* *URL*: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43905.pdf
* *Description*: Long Short-Term Memory (LSTM) is a specific recurrent neural network (RNN) architecture that was designed to model temporal sequences and their long-range dependencies more accurately than conventional RNNs. In this paper, we explore LSTM RNN architectures for large scale acoustic modeling in speech recognition

-----------------------------------------------6---------------------------------------------------

* *Title*: Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network
* *Magazine*: Elsevier journal “Physica D: Nonlinear Phenomena”, Volume 404, March 2020: Special Issue on Machine Learning and Dynamical Systems
* *URL*: https://arxiv.org/pdf/1808.03314.pdf
* *Description*: . The goal of this tutorial is to explain the essential RNN and LSTM fundamentals in a single document. Drawing from concepts in Signal Processing, we formally derive the canonical RNN formulation from differential equations. We then propose and prove a precise statement, which yields the RNN unrolling technique. We also review the difficulties with training the standard RNN and address them by transforming the RNN into the “Vanilla LSTM” network through a series of logical arguments

--------------------------------------------------7----------------------------------------------------

* *Title*: Multiple Linear Regression
* *Magazine*: Cathie Marsh Centre for Census and Survey Research
* *URL*: https://hummedia.manchester.ac.uk/institutes/cmist/archive-publications/working-papers/2008/2008-19-multiple-linear-regression.pdf
* *Description*: A multiple linear regression analysis is carried out to predict the values of a dependent variable, Y, given a set of p explanatory variables (x1,x2,….,xp). In these notes, the necessary theory for multiple linear regression is presented and examples of regression analysis with census data are given to illustrate this theory.

---------------------------------------------8-----------------------------------------------------

* *Title*: Stock Market Prediction with Multiple Regression,Fuzzy Type-2 Clustering and Neural Networks
* *Magazine*: Procedia Computer Science 6 (2011) 201–206
* *URL*: https://reader.elsevier.com/reader/sd/pii/S1877050911005035?token=68118A210577CC7C229A5960DB081E07456899305047AD59407277957D9FEE670AB4D064CA6E3F5874E9F24A66C90F8F&originRegion=eu-west-1&originCreation=20210414134135
* *Description*: Stock market forecasting research offers many challenges and opportunities, with the forecasting of individual stocks or indexes focusing on forecasting either the level (value) of future market prices, or the direction of market price movement. A three-stage stock market prediction system is introduced in this article. In the first phase, Multiple Regression Analysis is applied to define the economic and financial variables which have a strong relationship with the output.

# DATA ANALYSIS AND INTERPRETATION

Figure 2: Research Workflow

## Data Preprocessing

**Data Collection**: Downloaded in CSV format from official NSE website.

**Feature Addition**: Moving Average for 7 days, 14 days and 21 days, SD for 7 days, Open – Close, High – Low are added.

**Data Cleaning:** Remove the rows containing the empty column values.

**Data Split:** Data is split into 2 parts, training and testing in the ratio of 4:1

**Feature Extraction:** The correlations between the features and closing price for the training data is analysed and those with strong correlation are considered for further processing

Please find below the tables containing the correlation co-efficients of various features with closing price for each stock.

**Airtel**

Table 1: Correlation Co-efficient > 0.5 (Airtel)

|  |  |
| --- | --- |
| **Features** | **Correlation Coefficient** |
| ('Close Price', 'Turnover') | 0.522321139 |
| ('Close Price', 'MA(21)’) | 0.960588397 |
| ('Close Price', 'MA(14)’) | 0.974816874 |
| ('Close Price', 'MA(7)’) | 0.988527759 |
| ('Close Price', 'Prev Close') | 0.992418826 |
| ('Close Price', 'Open Price') | 0.993647361 |
| ('Close Price', 'Low Price') | 0.996716827 |
| ('Close Price', 'High Price') | 0.997466513 |
| ('Close Price', 'Average Price') | 0.998905942 |
| ('Close Price', 'Last Price') | 0.999805502 |

**HDFC**

Table 2: Correlation Co-efficient > 0.5 (HDFC)

|  |  |
| --- | --- |
| **Features** | **Correlation Coefficient** |
| ('Close Price', 'MA(21)’) | 0.5035171461472367 |
| ('Close Price', 'MA(14)’) | 0.6329072457192763 |
| ('Close Price', 'MA(7)’) | 0.8132102277631666 |
| ('Close Price', 'Prev Close') | 0.9057310179049453 |
| ('Close Price', 'Open Price') | 0.9117219694702485 |
| ('Close Price', 'Low Price') | 0.9713390511707097 |
| ('Close Price', 'High Price') | 0.9705169127349681 |
| ('Close Price', 'Average Price') | 0.9855342994035249 |
| ('Close Price', 'Last Price') | 0.9972407433736472 |

**India Cem**

Table 3: Correlation Co-efficient > 0.5 (India Cem)

|  |  |
| --- | --- |
| **Features** | **Correlation Coefficient** |
| ('Close Price', 'Turnover') | 0.536986882418957 |
| ('Close Price', 'MA(21)’) | 0.9756913647354248 |
| ('Close Price', 'MA(14)’) | 0.98201900375552 |
| ('Close Price', 'MA(7)’) | 0.9928825080847141 |
| ('Close Price', 'Prev Close') | 0.9961646852196477 |
| ('Close Price', 'Open Price') | 0.9963062802622816 |
| ('Close Price', 'Low Price') | 0.9982861068750881 |
| ('Close Price', 'High Price') | 0.9987026967975201 |
| ('Close Price', 'Average Price') | 0.9994122527462231 |
| ('Close Price', 'Last Price') | 0.999938882043492 |

**Sun Pharma**

Table 4: Correlation Co-efficient > 0.5 (Sun Pharma)

|  |  |
| --- | --- |
| **Features** | **Correlation Coefficient** |
| ('Close Price', 'MA(21)’) | 0.9684175710181946 |
| ('Close Price', 'MA(14)’) | 0.9792428606068188 |
| ('Close Price', 'MA(7)’) | 0.991009461814243 |
| ('Close Price', 'Prev Close') | 0.9897122723693583 |
| ('Close Price', 'Open Price') | 0.9977526495186073 |
| ('Close Price', 'Low Price') | 0.9989688180147768 |
| ('Close Price', 'High Price') | 0.9990599302669797 |
| ('Close Price', 'Average Price') | 0.9996625432809685 |
| ('Close Price', 'Last Price') | 0.9999456323830558 |

**TCS**

Table 5: Correlation Co-efficient > 0.5 (TCS)

|  |  |
| --- | --- |
| **Features** | **Correlation Coefficient** |
| ('Close Price', 'Turnover') | 0.536986882418957 |
| ('Close Price', 'MA(21)’) | 0.9765946880442554 |
| ('Close Price', 'MA(14)’) | 0.9843043832220104 |
| ('Close Price', 'MA(7)’) | 0.9928452075520686 |
| ('Close Price', 'Prev Close') | 0.9943310014927051 |
| ('Close Price', 'Open Price') | 0.9985007076389663 |
| ('Close Price', 'Low Price') | 0.9993255638939279 |
| ('Close Price', 'High Price') | 0.9994280309754727 |
| ('Close Price', 'Average Price') | 0.9997790499628239 |
| ('Close Price', 'Last Price') | 0.9999657306525533 |

**Data Normalization:** The data is minmax normalized and used for designing and training our model.

## Data Model Training

The training data is then used for training the model (Linear as well as LSTM).

#### Multiple linear Regression

#### 

Figure 3: Multiple Linear Regression

#### LSTM

#### Illustrated Guide to LSTM's and GRU's: A step by step explanation | by Michael Phi | Towards Data Science

Figure 4: LSTM

## Predictions

### Multiple Linear Regression

Once the model is trained using the training data, the testing data is fed to the model to test the accuracy of our model.

##### AIRTEL

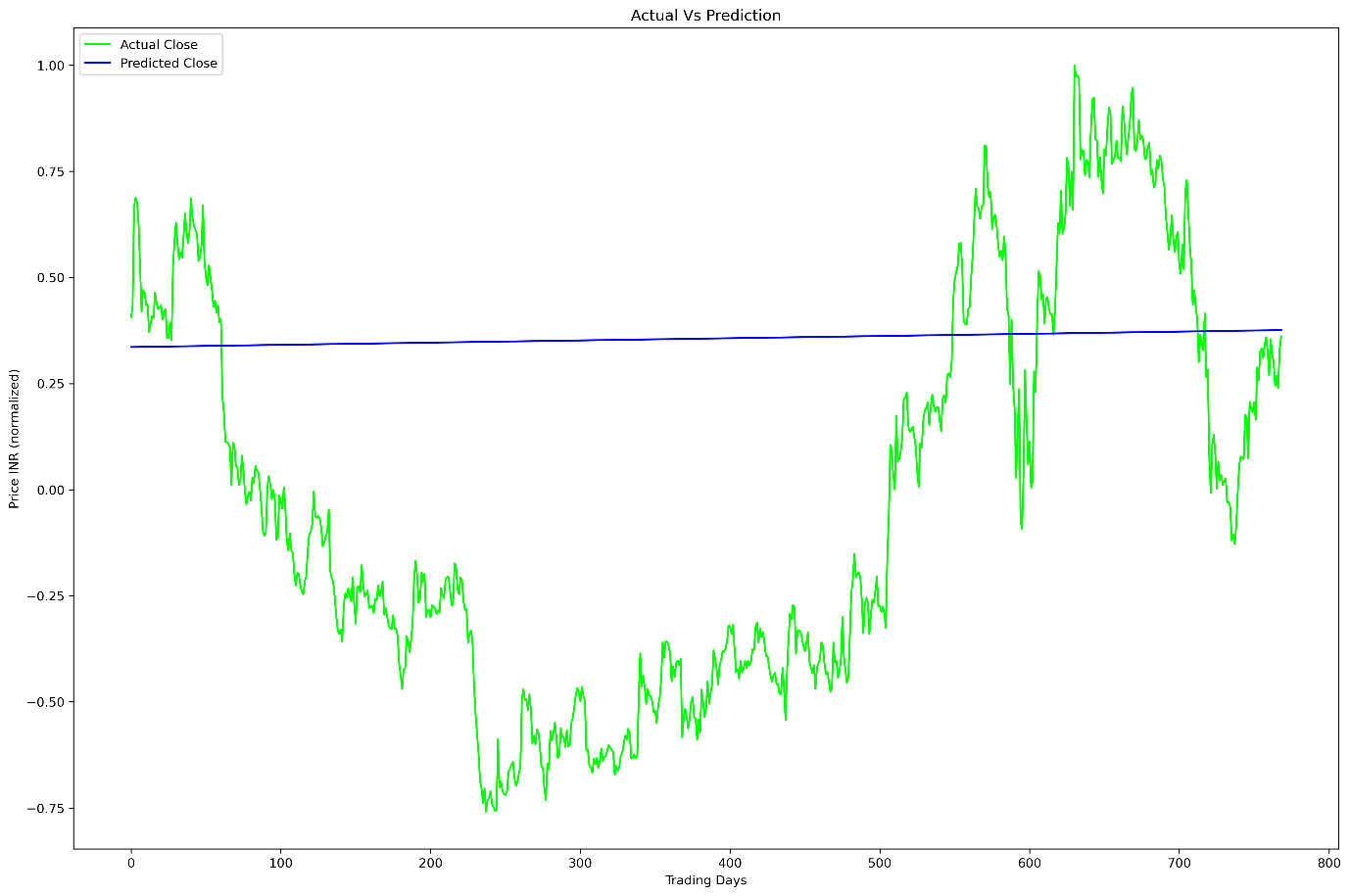


Figure 5: Multiple Linear Regression - Actual Vs Prediction (Airtel)

Train Score: 0.17863642 MSE (0.42265401 RMSE)

Test Score: 0.35805990 MSE (0.59838106 RMSE)

HDFC

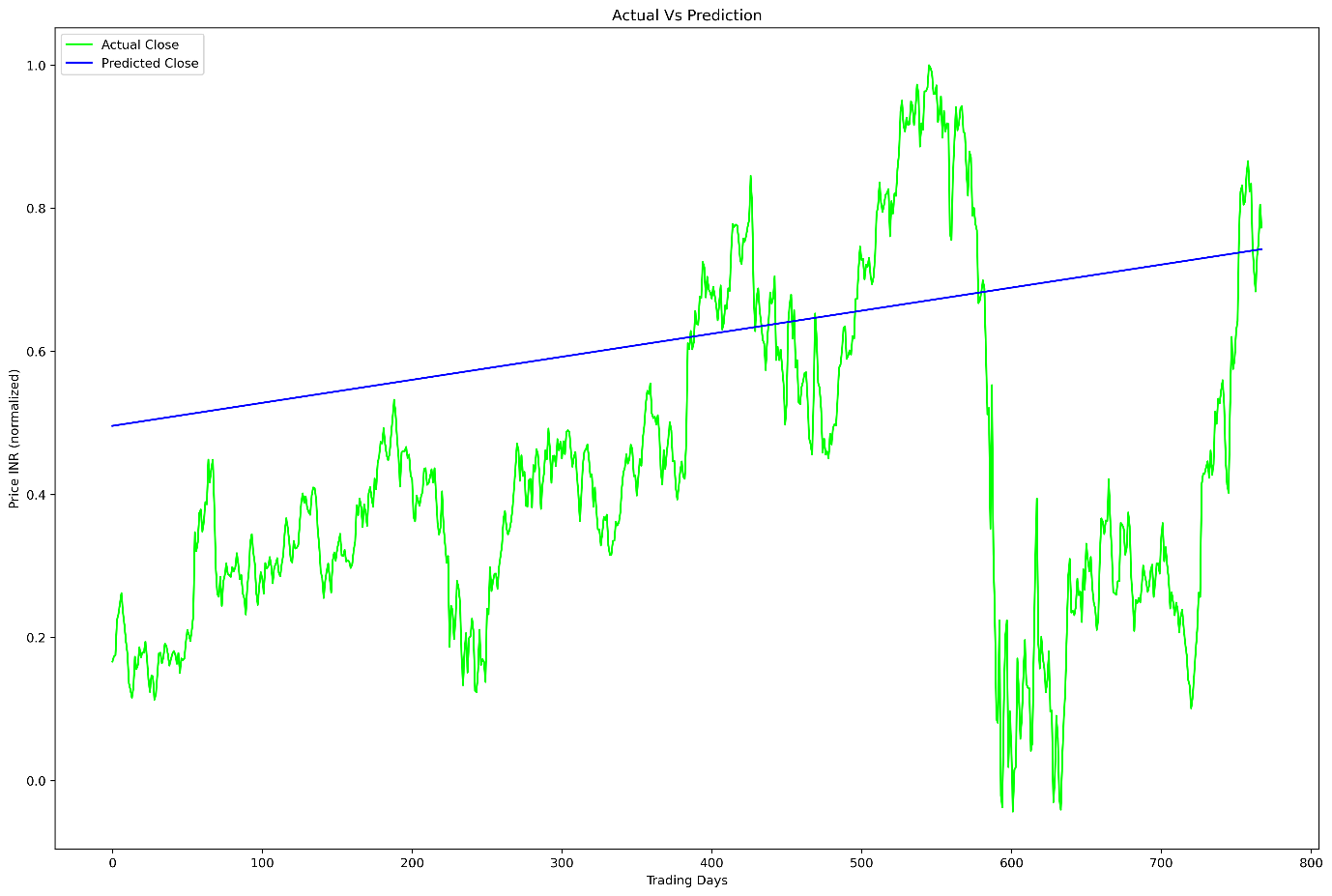


Figure 6: Multiple Linear Regression - Actual Vs Prediction (HDFC)

Train Score: 0.07378023 MSE (0.27162516 RMSE)

Test Score: 0.08099504 MSE (0.28459628 RMSE)

INDIA CEMENTS

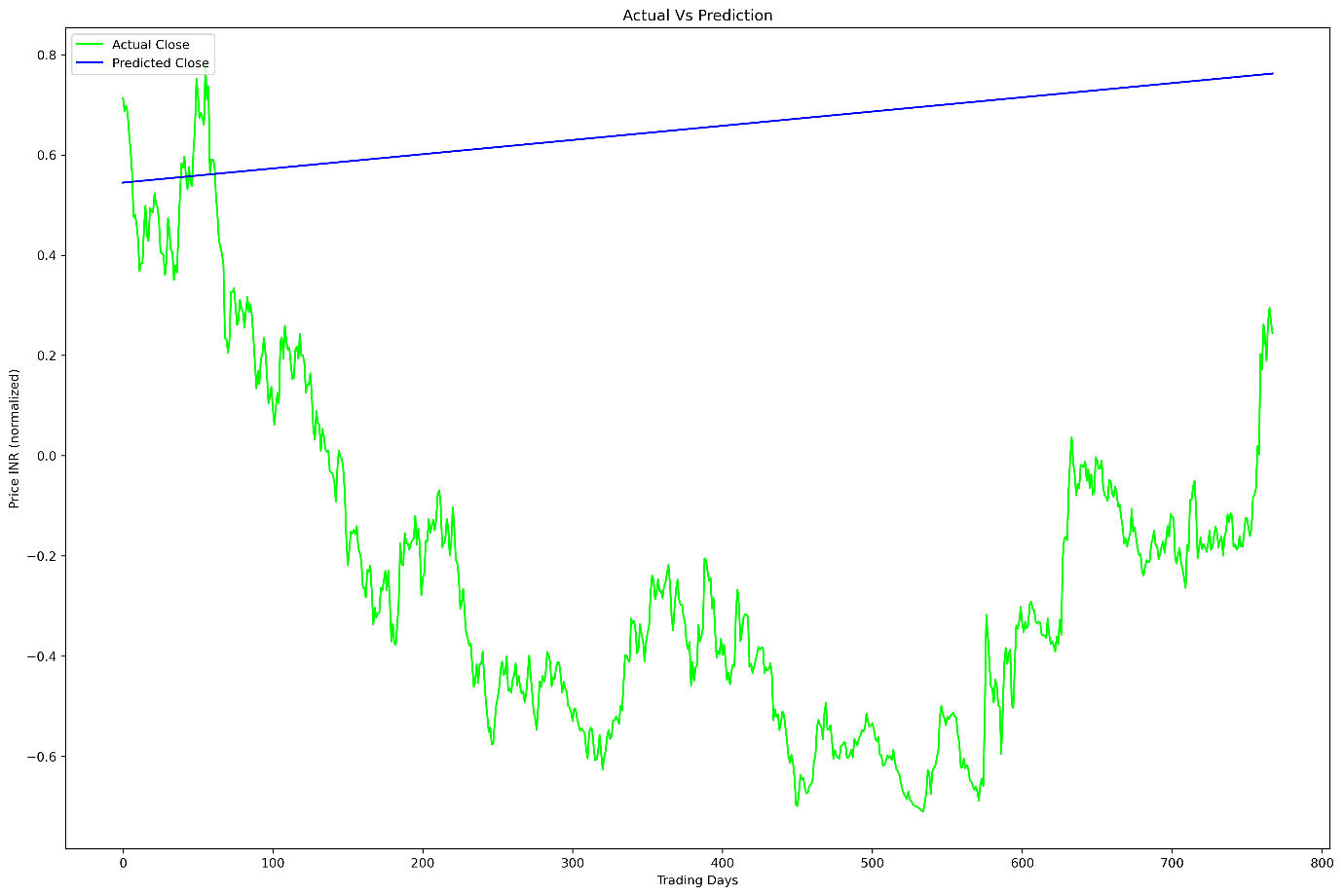


Figure 7: Multiple Linear Regression - Actual Vs Prediction (India Cements)

Train Score: 0.12913759 MSE (0.35935719 RMSE)

Test Score: 0.91428336 MSE (0.95618166 RMSE)

SUN PHARMA

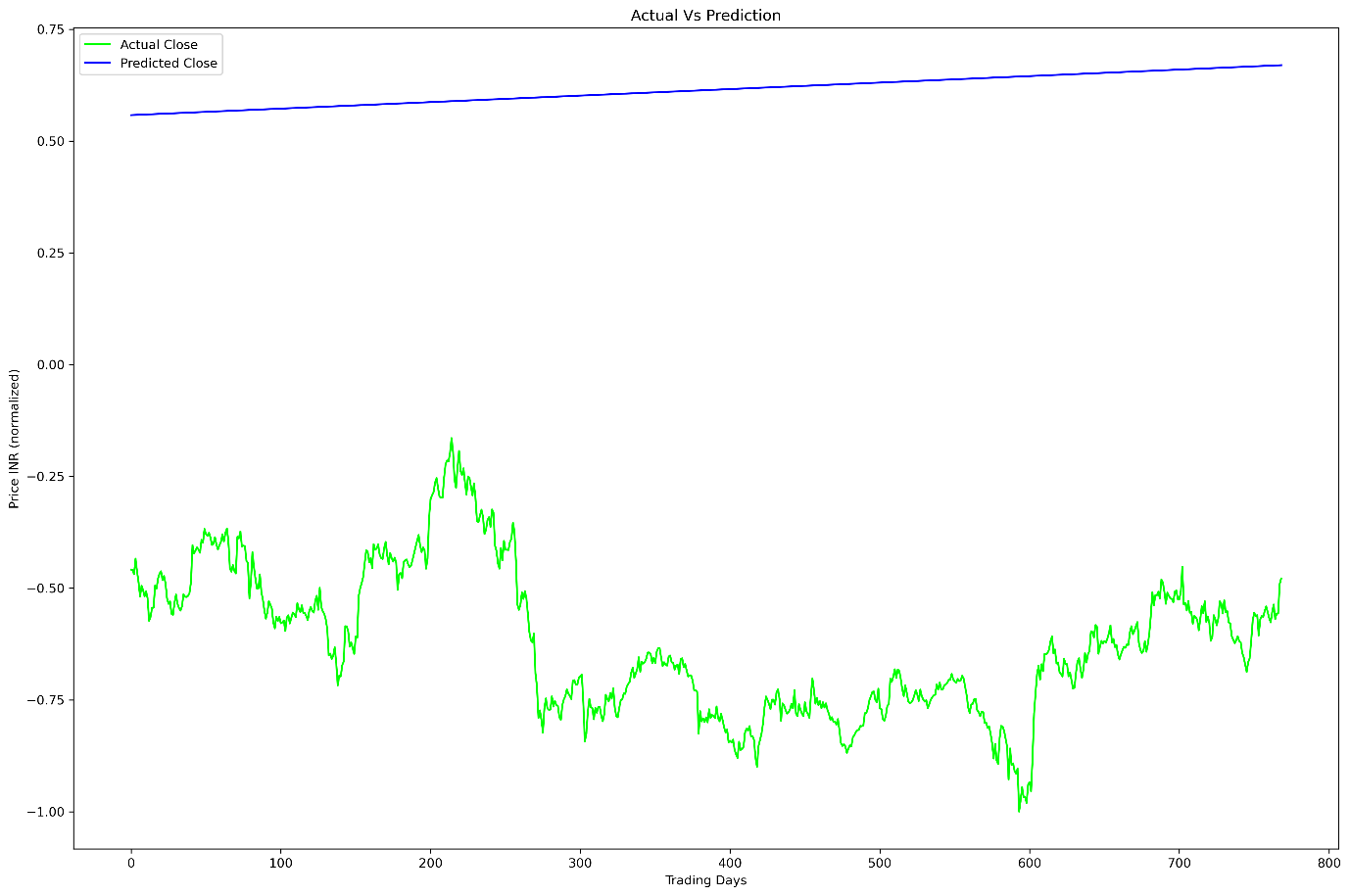


Figure 8: Multiple Linear Regression - Actual Vs Prediction (Sun Pharma)

Train Score: 0.22767407 MSE (0.47715204 RMSE)

Test Score: 1.56031364 MSE (1.24912515 RMSE)

TCS

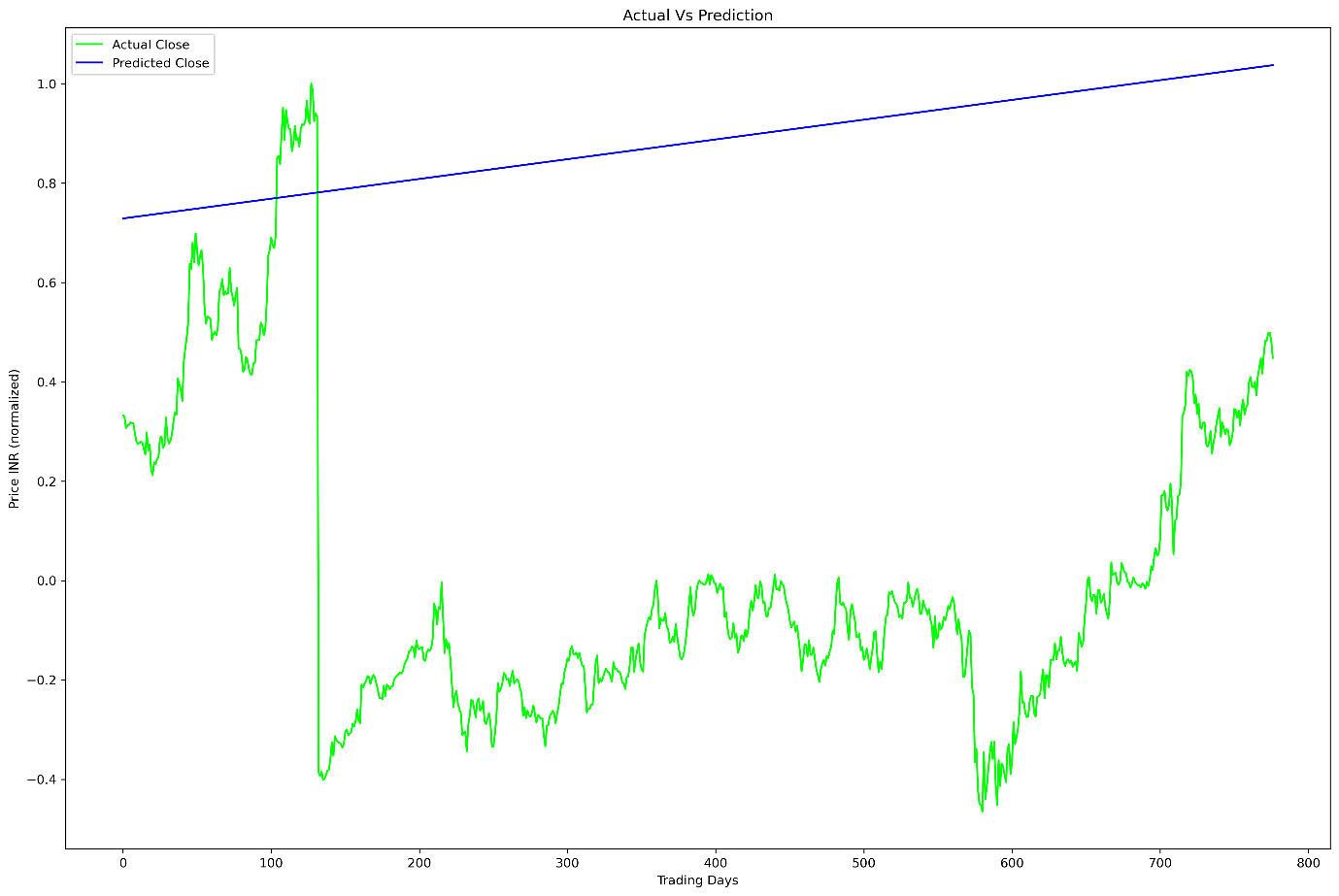


Figure 9: Multiple Linear Regression - Actual Vs Prediction (TCS)

Train Score: 0.05147397 MSE (0.22687875 RMSE)

Test Score: 0.87486308 MSE (0.93534116 RMSE)

### LSTM Recurrent Neural Networks

AIRTEL

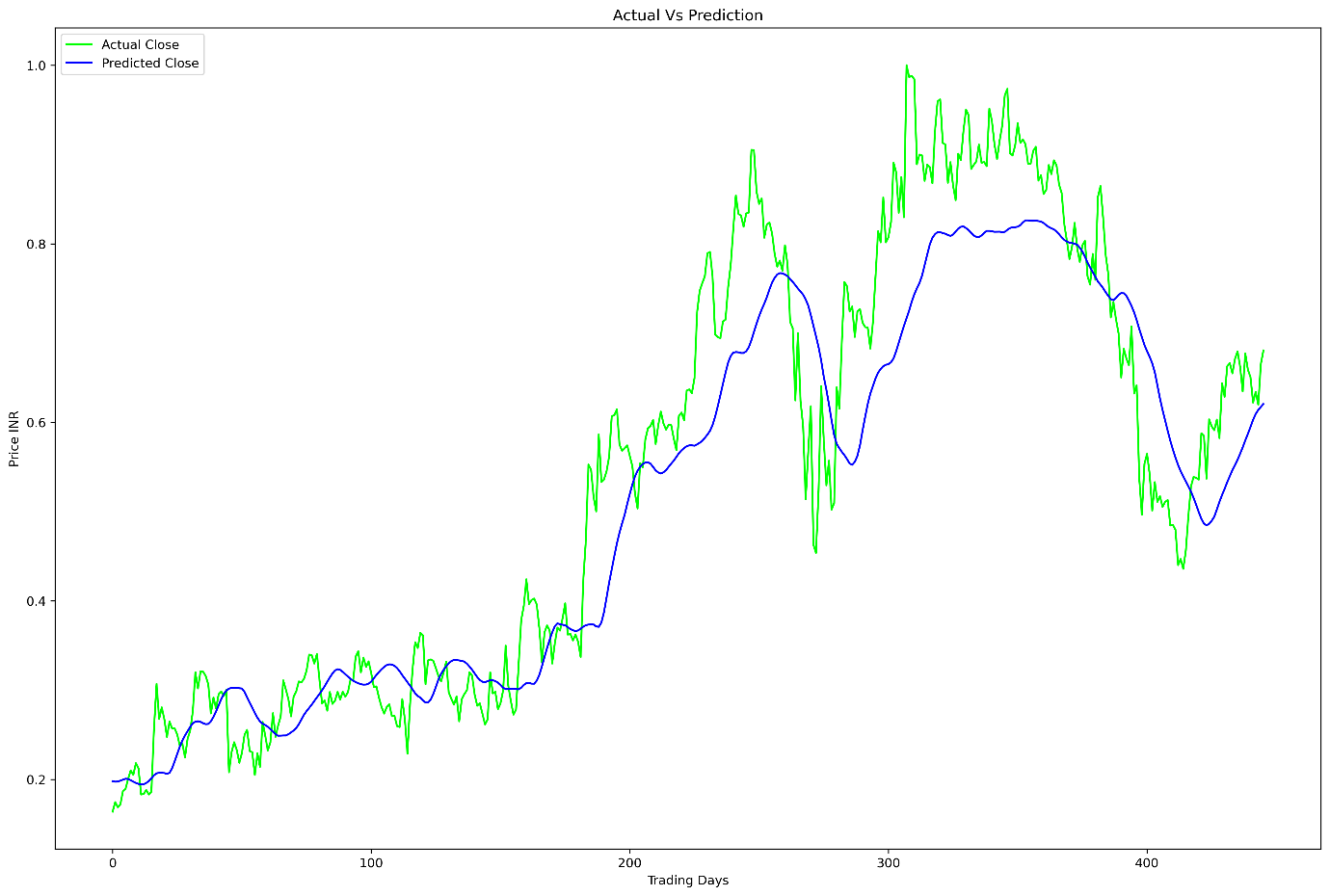


Figure 10: LSTM Neural Network - Actual Vs Prediction (Airtel)

Train Score: 0.00350620 MSE (0.05921316 RMSE)

Test Score: 0.00798955 MSE (0.08938428 RMSE)

HDFC

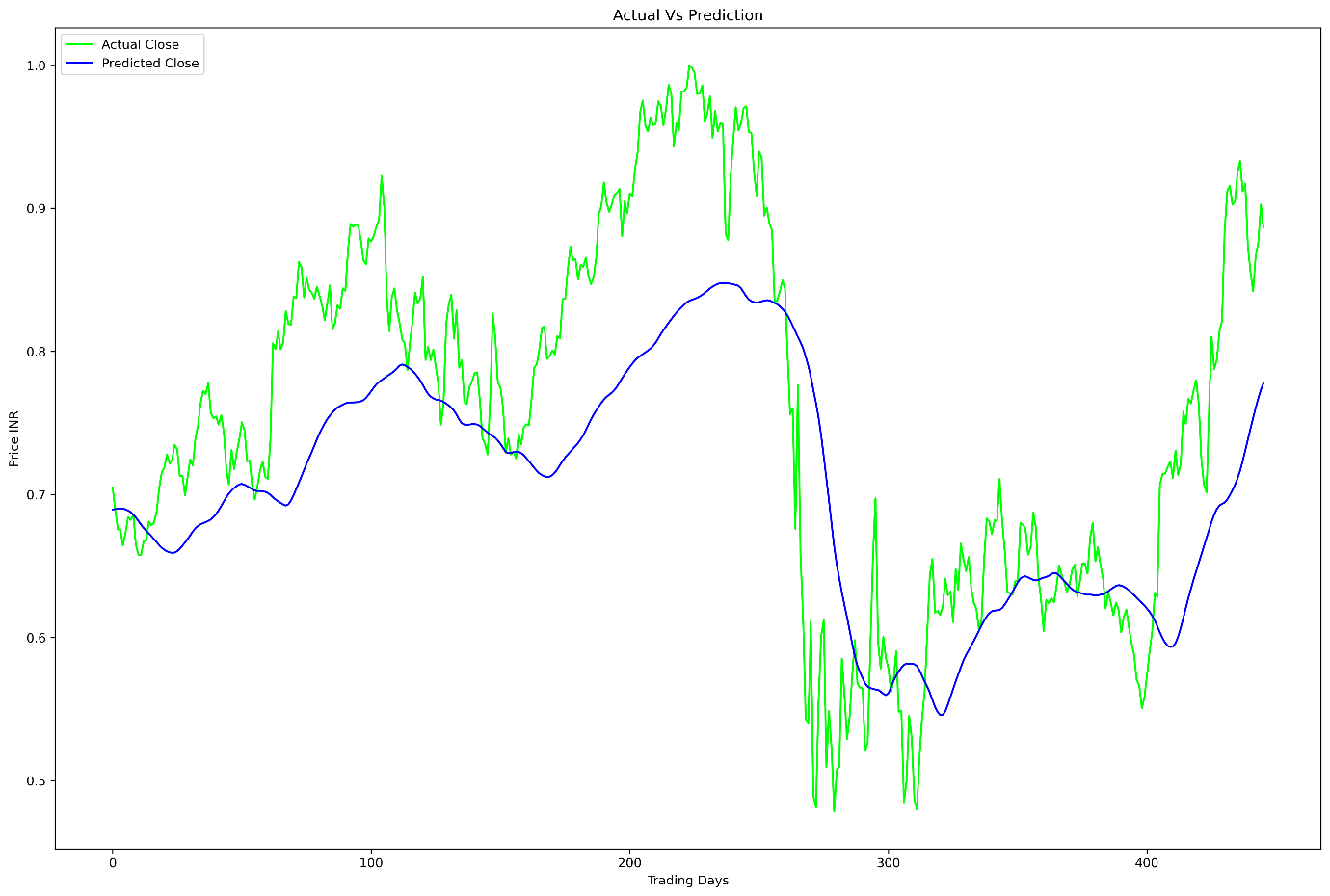


Figure 11: LSTM Neural Network - Actual Vs Prediction (HDFC)

Train Score: 0.00082917 MSE (0.02879529 RMSE)

Test Score: 0.00864996 MSE (0.09300517 RMSE)

INDIA CEM

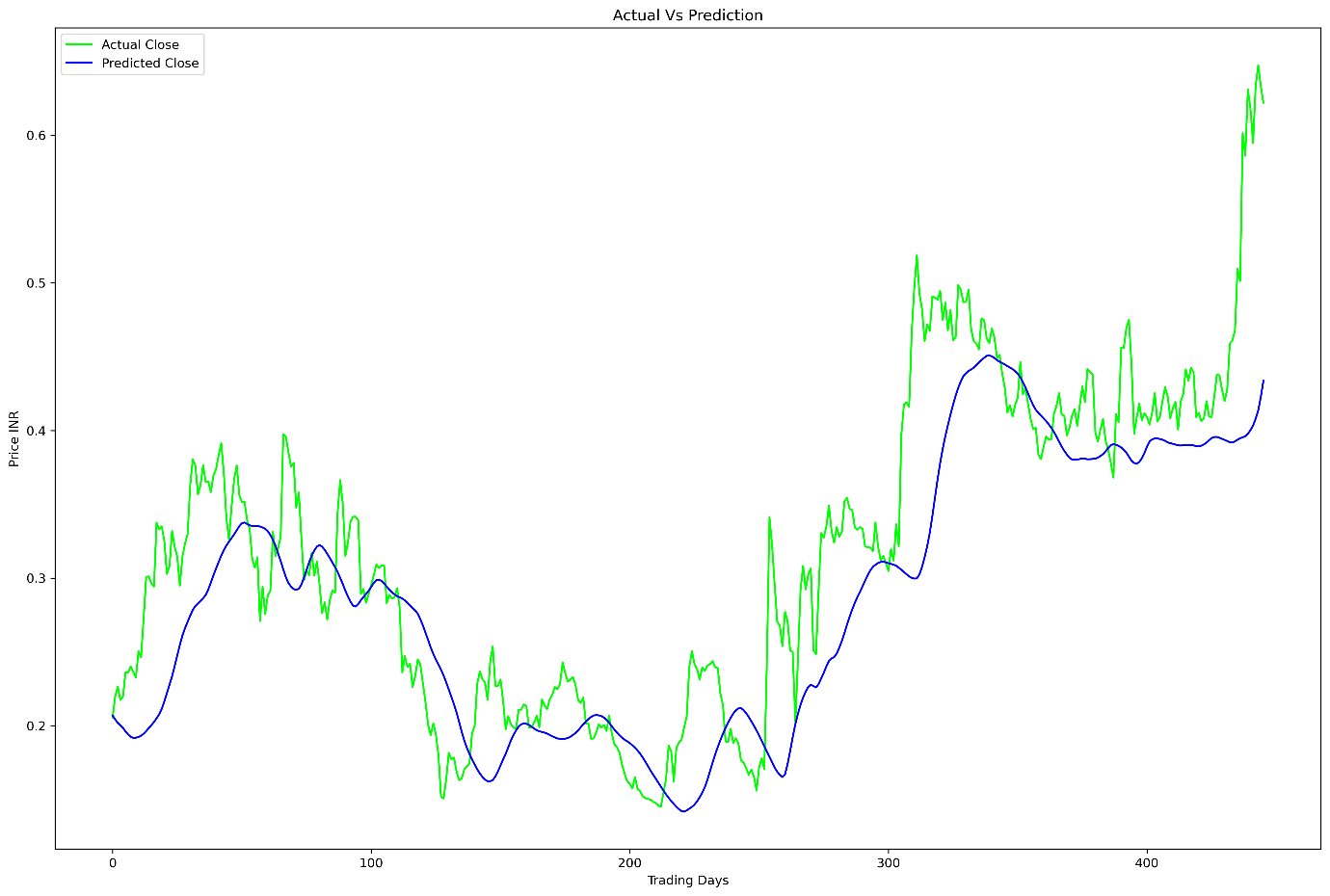


Figure 12: LSTM Neural Network - Actual Vs Prediction (India Cements)

Train Score: 0.00082917 MSE (0.02879529 RMSE)

Test Score: 0.00864996 MSE (0.09300517 RMSE)

SUNPHARMA

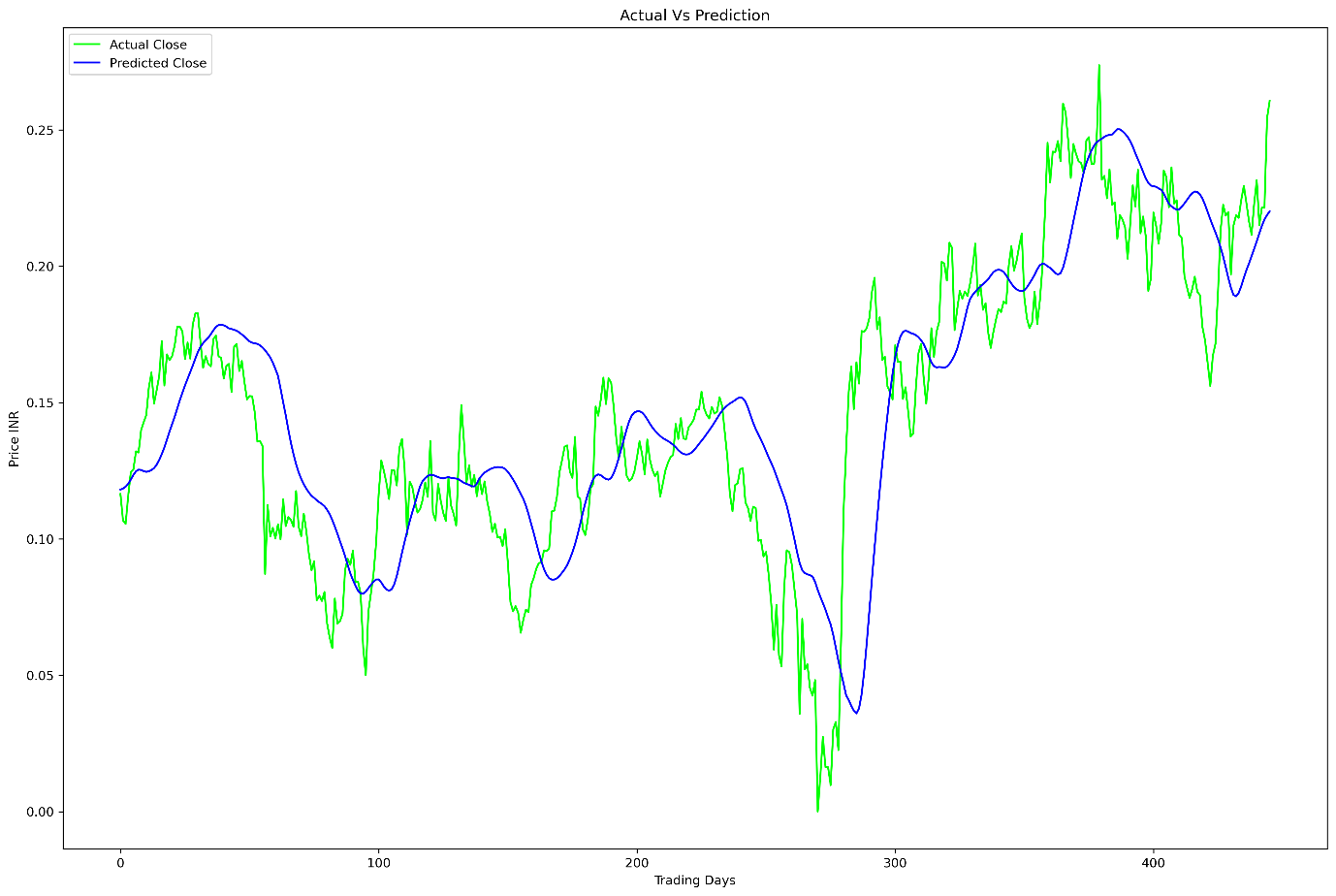


Figure 13: LSTM Neural Network - Actual Vs Prediction (Sun Pharma)

Train Score: 0.00482382 MSE (0.06945370 RMSE)

Test Score: 0.00109332 MSE (0.03306532 RMSE)

TCS

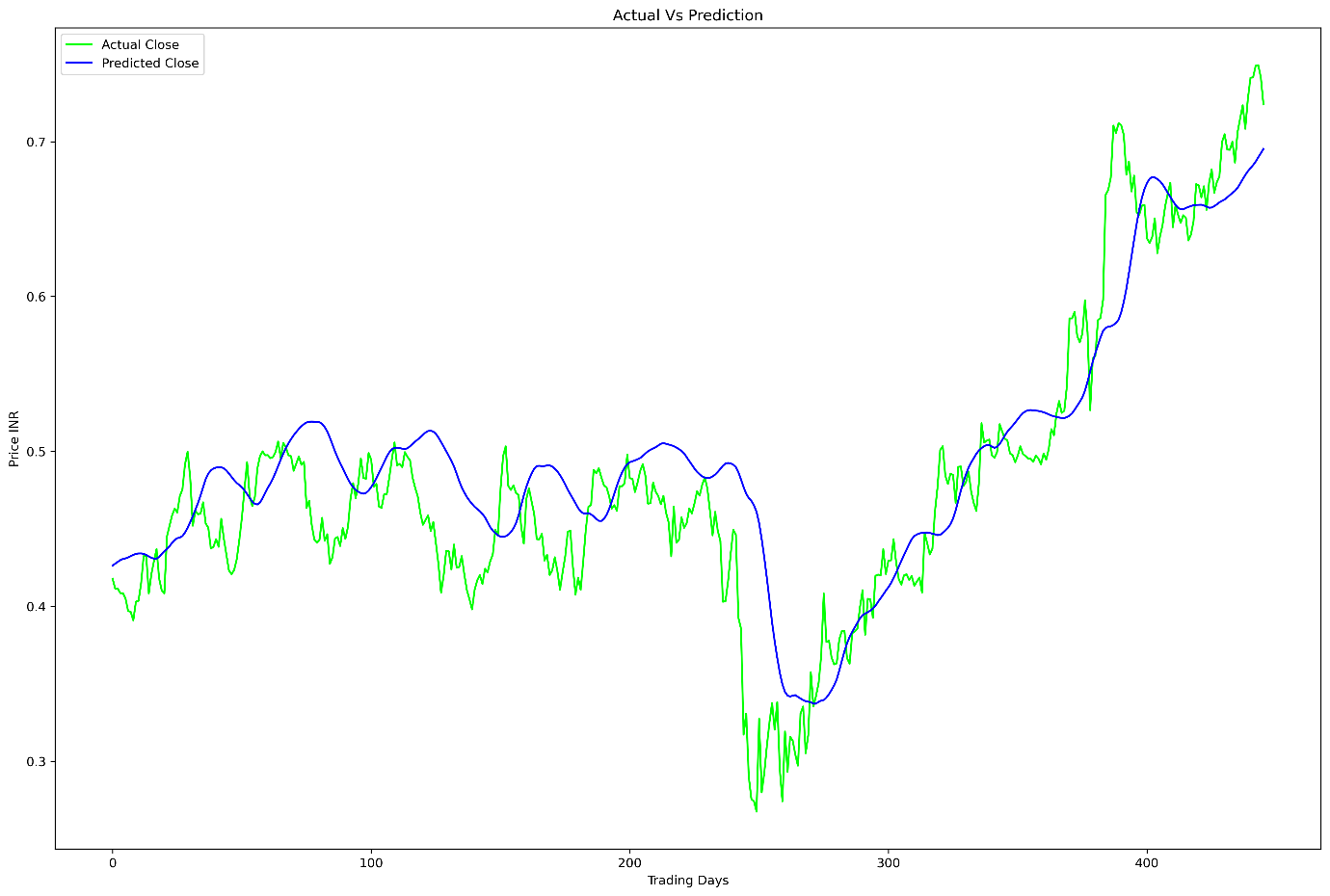


Figure 14: LSTM Neural Network - Actual Vs Prediction (TCS)

Train Score: 0.00329858 MSE (0.05743330 RMSE)

Test Score: 0.00205551 MSE (0.04533773 RMSE)

### LSTM Recurrent Neural Networks (Improved)

AIRTEL

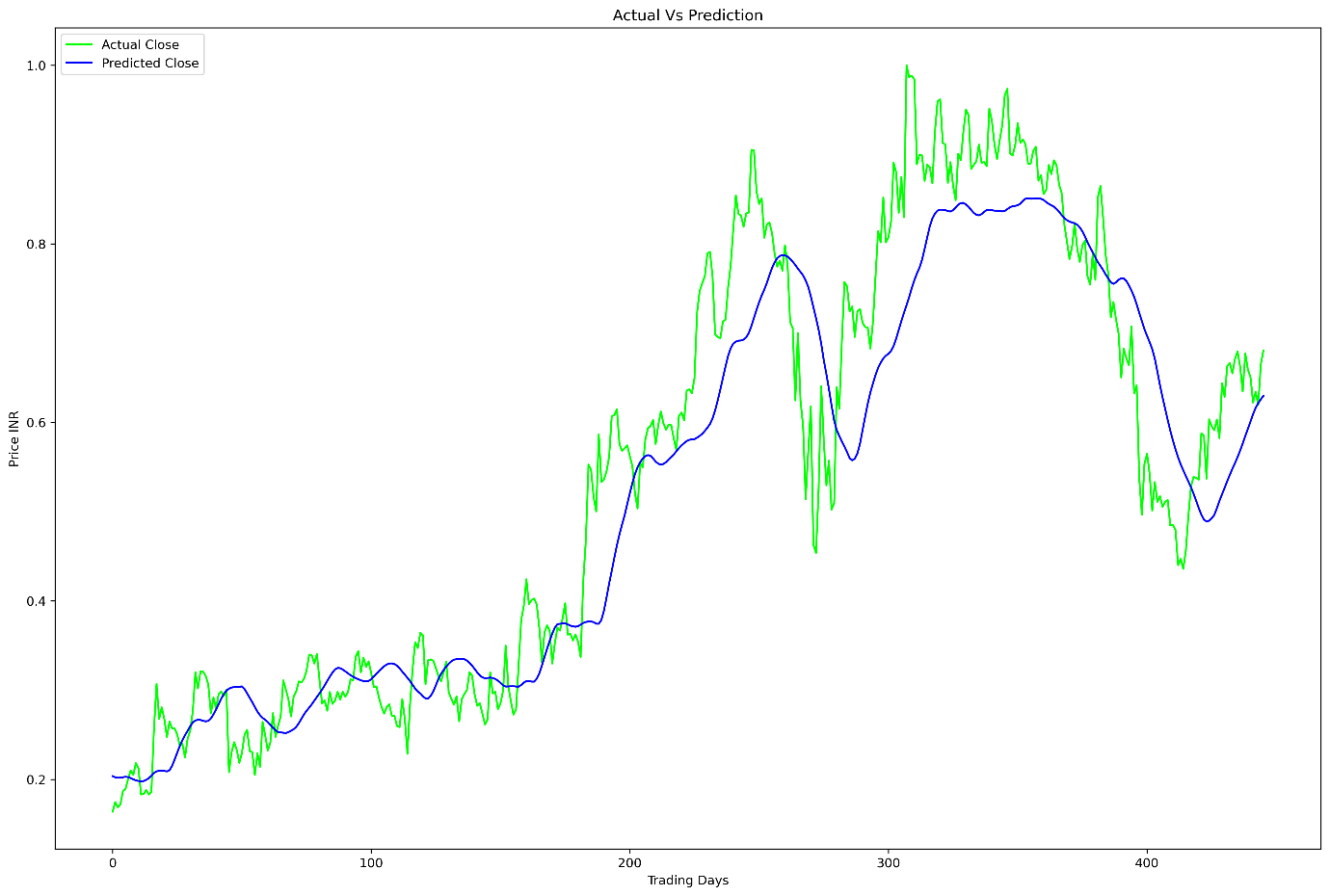


Figure 15: LSTM Neural Network (improved) - Actual Vs Prediction (Airtel)

Train Score: 0.00365975 MSE (0.06049587 RMSE)

Test Score: 0.00744119 MSE (0.08626235 RMSE)

HDFC

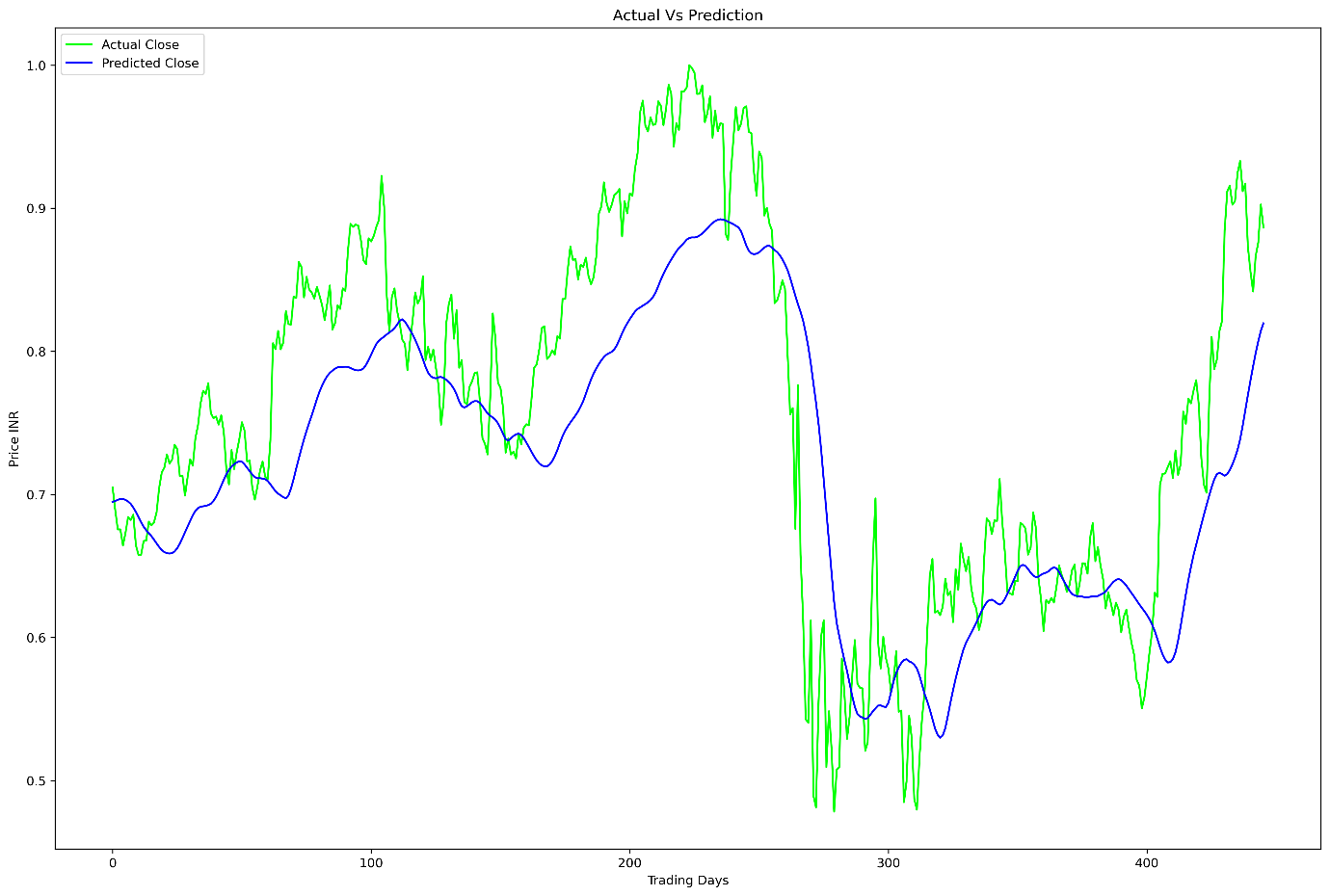


Figure 16: LSTM Neural Network (improved) - Actual Vs Prediction (HDFC)

Train Score: 0.00073833 MSE (0.02717214 RMSE)

Test Score: 0.00634284 MSE (0.07964192 RMSE)

INDIA CEM

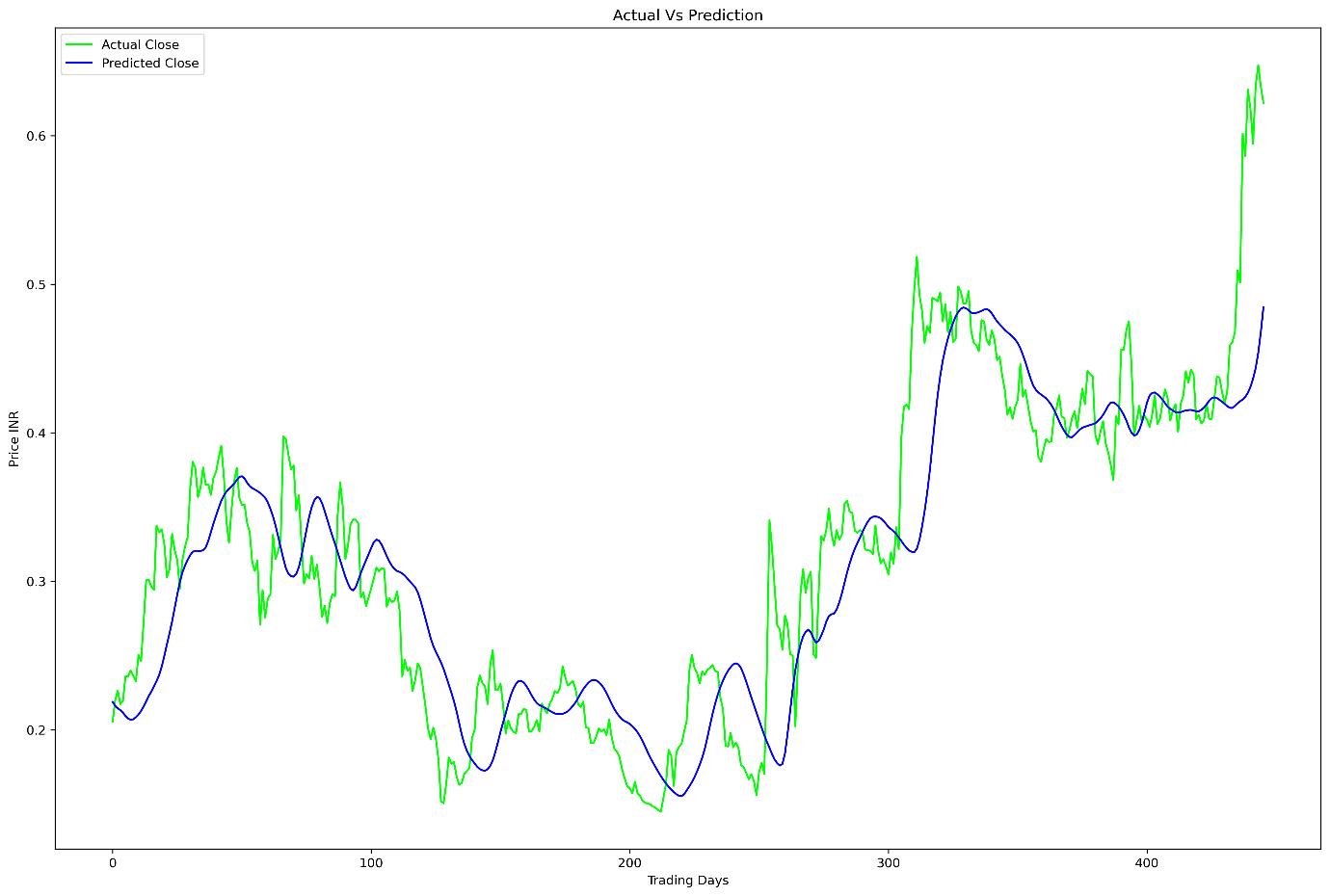


Figure 17: LSTM Neural Network (improved) - Actual Vs Prediction (India Cem)

Train Score: 0.00312831 MSE (0.05593127 RMSE)

Test Score: 0.00275089 MSE (0.05244896 RMSE)

SUN PHARMA

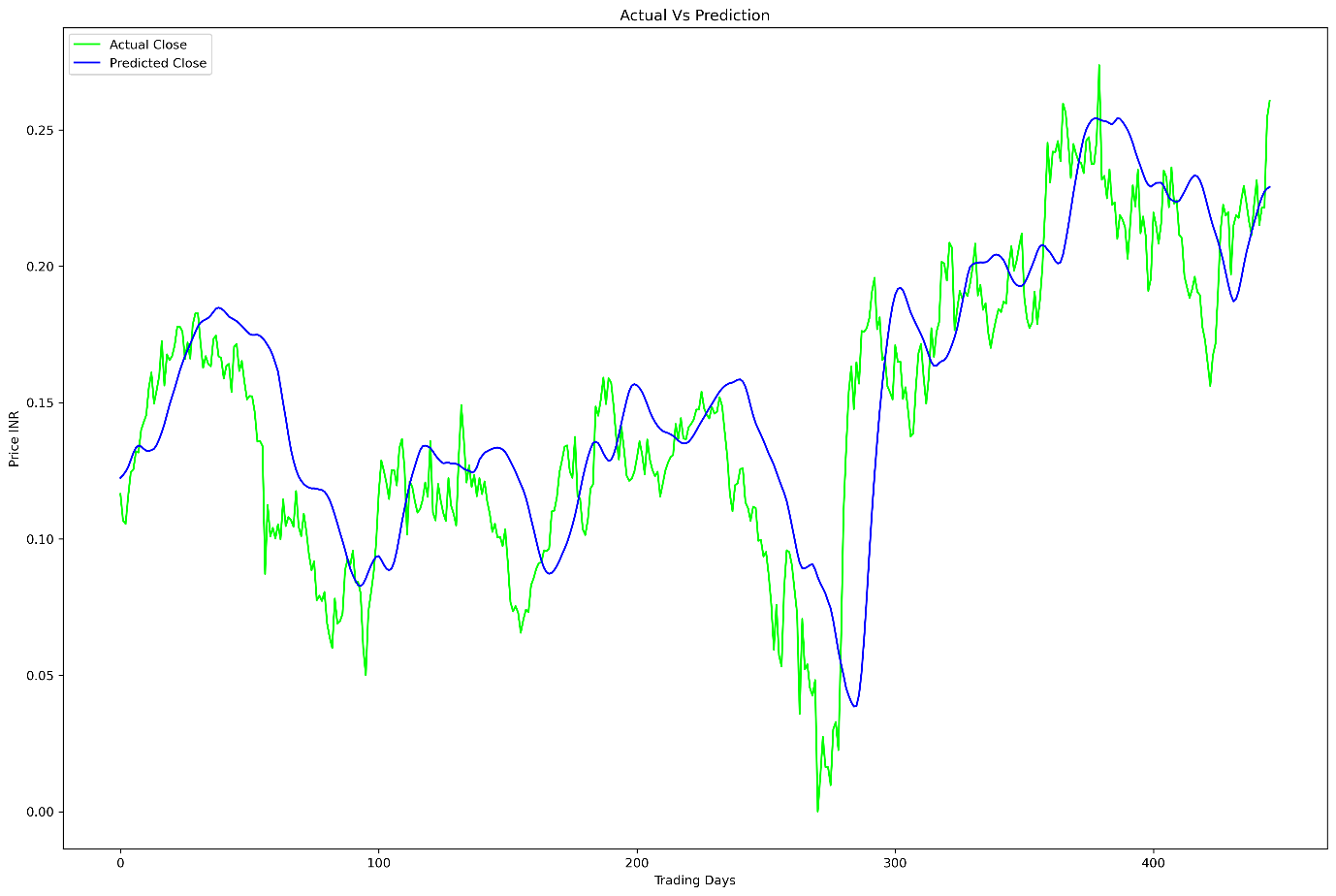


Figure 18: LSTM Neural Network (improved) - Actual Vs Prediction (Sun Pharma)

Train Score: 0.00406132 MSE (0.06372847 RMSE)

Test Score: 0.00106852 MSE (0.03268822 RMSE)

TCS

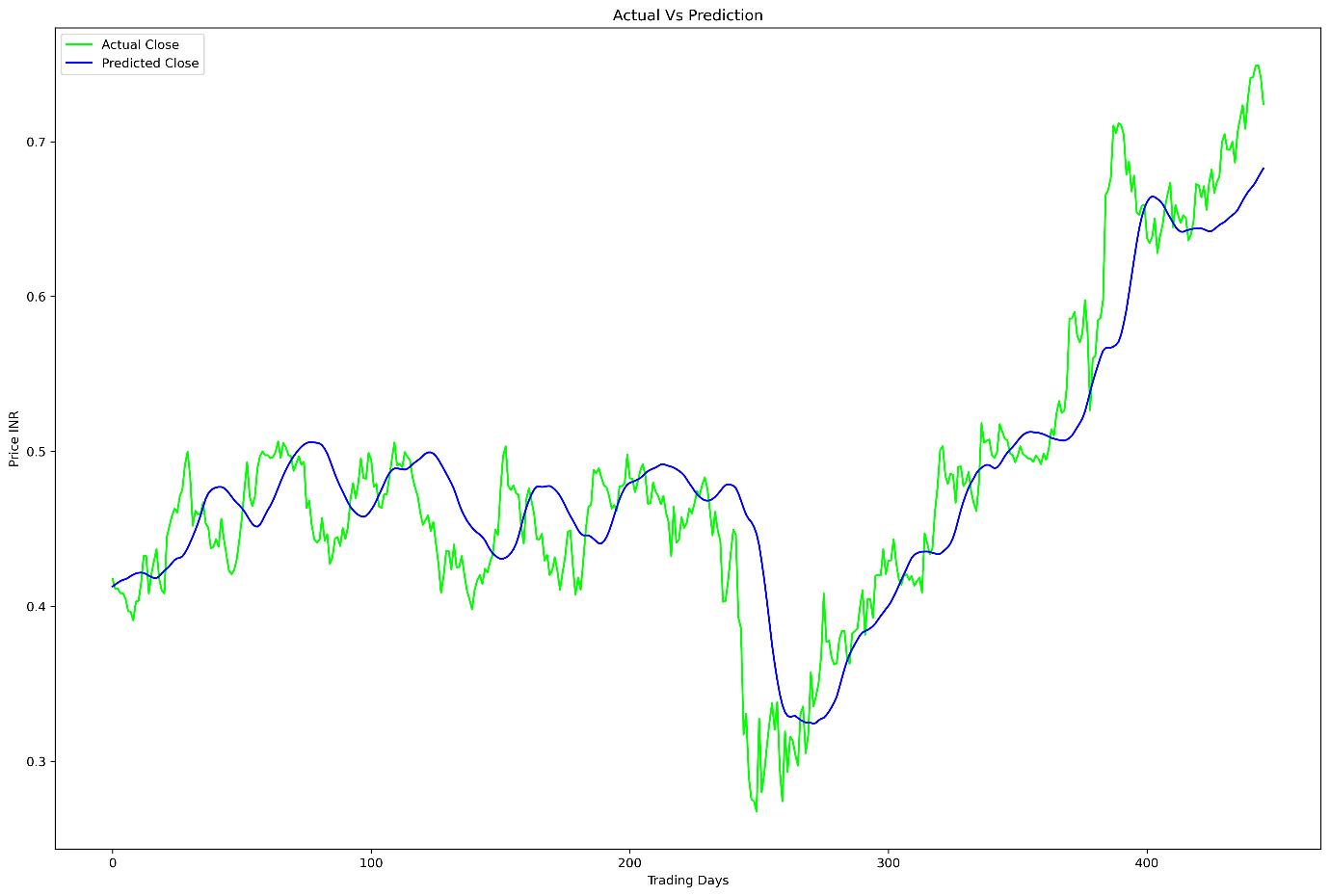
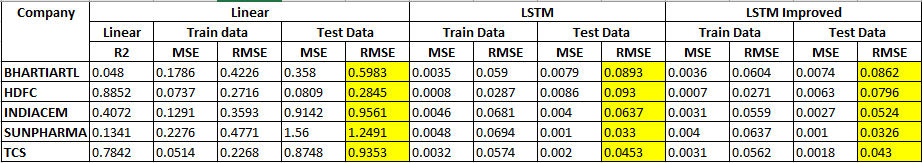


Figure 19: LSTM Neural Network (improved) - Actual Vs Prediction (TCS)

Train Score: 0.00316142 MSE (0.05622654 RMSE)

Test Score: 0.00185183 MSE (0.04303284 RMSE)

Performance Comparison of each model for different stocks



# FINDINGS

1. Maximum R2 value is 0.88 for HDFC for multivariate regression.
2. Apart from HDFC, R2 value is 0.78 for TCS.
3. Linear Regression works better only for HDFC and TCS.
4. We can see the RMSE value gets decreased irrespective of the stocks as we move towards LSTM.
5. LSTM is better technique for prediction than Linear Regression(Multi-variate) for all these stocks.

# ANNEXURE

**Tools Used**: Jupyter, Excel

**Implementation Language**: Python

**Modules used**: Keras, Pandas, Numpy, sklearn, matplotlib

Code:

import sys

import pandas as pd

import datetime

import pandas as pd

import numpy as np

In [ ]:

#Declaring All constants, labels etc.,

SYMBOL = 'Symbol'

SERIES = 'Series'

DATE = 'Date'

PREV\_CLOSE = 'Prev Close'

OPEN\_PRICE = 'Open Price'

HIGH\_PRICE = 'High Price'

LOW\_PRICE = 'Low Price'

LAST\_PRICE = 'Last Price'

CLOSE\_PRICE = 'Close Price'

AVERAGE\_PRICE = 'Average Price'

VOLUME = 'Total Traded Quantity'

TURNOVER = 'Turnover'

TRADES = 'No. of Trades'

DELIVERABLE\_QTY = 'Deliverable Qty'

DELQTY\_VOLUME\_PERCENT = '% Dly Qt to Traded Qty'

H\_MINUS\_L = 'H-L'

O\_MINUS\_C = 'O-C'

MA\_7 = 'MA(7)'

MA\_14 = 'MA(14)'

MA\_21 = 'MA(21)'

SD\_7 = 'SD(7)'

ITEM = 'Item'

#colors

LIGHT\_BLUE = '#0A7388'

BLUE = '#0000FF'

GREEN = '#00FF00'

#companies

TCS = 'tcs'

#plot

DEFAULT\_TITLE='NSE - Trading'

DEFAULT\_Y\_LABEL='Price INR'

DEFAULT\_X\_LABEL='Trading Days'

ACTUAL\_VS\_PREDICTION = 'Actual Vs Prediction'

ACTUAL\_CLOSE = 'Actual Close'

PREDICTED\_CLOSE = 'Predicted Close'

UPPER\_LEFT = 'upper left'

#DataFile Map

DATA\_TO\_READ = {

"1":"data\\BHARTIARTL\\Airtel.csv",

"2":"data\\HDFC\\Hdfc.csv",

"3":"data\\INDIACEM\\IndiaCem.csv",

"4":"data\\SUNPHARMA\\SunPharma.csv",

"5":"data\\TCS\\Tcs.csv",

}

company\_dict = {

"1" : "Airtel",

"2" : "Hdfc",

"3" : "IndiaCem",

"4" : "SunPharma",

"5" : "Tcs"

}

AVAILABLE\_SYMBOLS=["1. BHARTIARTL - airtel","2. HDFC - HDFC","3. INDIACEM - India Cements","4. SUNPHARMA - Sun Pharma","5. TCS - TCS"]

print(AVAILABLE\_SYMBOLS)

VALID\_SYMBOLS=["BHARTIARTL","HDFC","INDIACEM","SUNPHARMA","TCS"]

VALID\_NUMBERS=[1,2,3,4,5]

VALIDATION\_ERROR = "Please enter one of the supported options:"

VALIDATION\_OPTIONS = f"{VALIDATION\_ERROR} {VALID\_NUMBERS}\n"

function to get the stock from user[¶](file:///E:\GoogleDrive_sem1\GoogleDrive_sem1\sem%204\project\2019201015\Final%20Review\lstm_inputs_13042021.doc#function-to-get-the-stock-from-user)

In [ ]:

def get\_symbol\_to\_process():

print("1. BHARTIARTL - airtel \n 2. HDFC - HDFC \n 3. INDIACEM - India Cements \n 4. SUNPHARMA - Sun Pharma \n 5. TCS - TCS")

print("Choose from 1 to 5")

#print(f"Please give the symbol to run prediction for. {AVAILABLE\_SYMBOLS}")

symbol\_selected = input()

option=int(symbol\_selected)

print(option)

print(type(option))

global is\_valid\_symbol

global company

company = company\_dict[symbol\_selected]

if (not isinstance(option, int) or option<=0 or option>=6):

print(VALIDATION\_OPTIONS)

is\_valid\_symbol= False

#if(symbol\_selected<1 or symbol\_selected>5):

# print(VALIDATION\_OPTIONS)

# is\_valid\_symbol= False

else:

is\_valid\_symbol= True

print(f"Running prediction for {DATA\_TO\_READ[symbol\_selected]}")

return symbol\_selected

function to read csv into a dataframe[¶](file:///E:\\GoogleDrive_sem1\\GoogleDrive_sem1\\sem%204\\project\\2019201015\\Final%20Review\\lstm_inputs_13042021.doc" \l "function-to-read-csv-into-a-dataframe)

In [ ]:

def get\_historical\_data(filename):

col\_names = [SYMBOL ,SERIES ,DATE ,PREV\_CLOSE ,OPEN\_PRICE ,HIGH\_PRICE ,LOW\_PRICE ,LAST\_PRICE ,CLOSE\_PRICE ,AVERAGE\_PRICE ,VOLUME ,TURNOVER ,TRADES ,DELIVERABLE\_QTY ,DELQTY\_VOLUME\_PERCENT ,H\_MINUS\_L ,O\_MINUS\_C ,MA\_7 ,MA\_14 ,MA\_21 ,SD\_7]

stocks = pd.read\_csv(filename, header=0, names=col\_names)

df = pd.DataFrame(stocks)

return df

function to print the statistics of selected columns of data[¶](file:///E:\GoogleDrive_sem1\GoogleDrive_sem1\sem%204\project\2019201015\Final%20Review\lstm_inputs_13042021.doc#function-to-print-the-statistics-of-sel)

In [ ]:

def get\_data\_stats(data,columns\_to\_use):

print(f"\n {data[columns\_to\_use].describe()} \n")

function to get the features to use as input from the user[¶](file:///E:\GoogleDrive_sem1\GoogleDrive_sem1\sem%204\project\2019201015\Final%20Review\lstm_inputs_13042021.doc#function-to-get-the-features-to-use-as-)

In [ ]:

def get\_features\_to\_use(data):

avaliable\_features = list(set(data.columns) - set([CLOSE\_PRICE,DATE,SYMBOL,SERIES,TRADES]))

print(f"Select features from this \n {','.join(avaliable\_features)}")

#print('suggested features:')

columns\_as\_str = input()

features\_to\_use = columns\_as\_str.split(",")

features\_to\_use.append(CLOSE\_PRICE)

return features\_to\_use

Get the Symbol from User[¶](file:///E:\GoogleDrive_sem1\GoogleDrive_sem1\sem%204\project\2019201015\Final%20Review\lstm_inputs_13042021.doc#Get-the-Symbol-from-User)

In [ ]:

is\_valid\_symbol = False

symbol\_selected = None

while(not is\_valid\_symbol):

symbol\_entered = get\_symbol\_to\_process()

if(symbol\_entered):

filename = DATA\_TO\_READ[symbol\_entered]

Reading the data for the given input stock[¶](file:///E:\GoogleDrive_sem1\GoogleDrive_sem1\sem%204\project\2019201015\Final%20Review\lstm_inputs_13042021.doc#Reading-the-data-for-the-given-input-st)

In [ ]:

print(filename)

data = get\_historical\_data(filename)

print("Before",data)

data.dropna(inplace=True)

print("AFter",data)

In [ ]:

#import seaborn as sn

#import matplotlib.pyplot as plt

corrMatrix = data.corr()

corr\_pairs = corrMatrix.unstack()

sorted\_pairs = corr\_pairs.sort\_values(kind="quicksort")

strong\_pairs = sorted\_pairs[abs(sorted\_pairs) > 0.5]

mydict= strong\_pairs.to\_dict()

for k in mydict.keys():

if CLOSE\_PRICE in k[0]:

print(k,mydict[k])

#type(strong\_pairs)

#for var in strong\_pairs:

#print(var)

#print(strong\_pairs)

#sn.heatmap(corrMatrix, annot=True)

#plt.show()

In [ ]:

features\_to\_use = get\_features\_to\_use(data)

Normalizing the data[¶](file:///E:\GoogleDrive_sem1\GoogleDrive_sem1\sem%204\project\2019201015\Final%20Review\lstm_inputs_13042021.doc#Normalizing-the-data)

Now it is time to normalize the data. In the following cells we will normalise it for better prediction of data.

Step 1 : Get some stats.

In [ ]:

#data = data.dropna()

#features\_to\_use

get\_data\_stats(data,features\_to\_use)

Step 2 : Write function to normalize the data using min max scaling and Remove Unncessary data, i.e., Date and High value

In [ ]:

data.head

In [ ]:

import pandas as pd

# Import sklearn.preprocessing.StandardScaler

from sklearn.preprocessing import MinMaxScaler

def get\_normalised\_data(data):

"""

Normalises the data values using MinMaxScaler from sklearn

:param data: a DataFrame with columns as ['index','Open','Close','High','Low','Volume']

:return: a DataFrame with normalised value for all the columns except index

"""

# Initialize a scaler, then apply it to the features

scaler = MinMaxScaler()

numerical = features\_to\_use

data[numerical] = scaler.fit\_transform(data[numerical])

return data

def remove\_data(data):

stocks = pd.DataFrame()

stocks[ITEM] = range(0,data.shape[0])

print(data)

stocks[features\_to\_use] = data[features\_to\_use].apply(pd.to\_numeric,axis=1)

stocks.dropna(inplace= True)

#print(len(stocks))

return stocks

def remove\_data2(data):

"""

Remove columns from the data

:param data: a record of all the stock prices with columns as ['Date','Open Price','High Price','Low Price','Close Price','Volume']

:return: a DataFrame with columns as ['index','Open Price','Close Price','High Price','Low Price','Volume']

"""

# Define columns of data to keep from historical stock data

item = []

openf = []

close = []

high = []

low = []

volume = []

# Loop through the stock data objects backwards and store factors we want to keep

i\_counter = 0

for i in range(len(data) - 1, -1, -1):

' item.append(i\_counter)

' openf.append(data[OPEN\_PRICE][i])

close.append(data[CLOSE\_PRICE][i])

high.append(data[HIGH\_PRICE][i])

low.append(data[LOW\_PRICE][i])

volume.append(data[VOLUME][i])

i\_counter += 1

# Create a data frame for stock data

stocks = pd.DataFrame()

# Add factors to data frame

stocks[ITEM] = item

stocks[OPEN\_PRICE] = openf

stocks[CLOSE\_PRICE] = pd.to\_numeric(close)

stocks[HIGH\_PRICE] = pd.to\_numeric(high)

stocks[LOW\_PRICE] = pd.to\_numeric(low)

stocks[VOLUME] = pd.to\_numeric(volume)

# return new formatted data

return stocks

In [ ]:

print("before,",data)

stocks = remove\_data(data)

print(stocks.head())

print("--------------------------------------------------")

print(stocks.tail())

Step 2: Visualise raw data.

In [ ]:

#%matplotlib notebook

#%matplotlib inline

import matplotlib.pyplot as plt

plt.rcParams['figure.figsize'] = (18, 12)

def price(x):

"""

format the coords message box

:param x: data to be formatted

:return: formatted data

"""

return '$%1.2f' % x

def plot\_basic(stocks, title=DEFAULT\_TITLE, y\_label=DEFAULT\_Y\_LABEL, x\_label=DEFAULT\_X\_LABEL):

"""

Plots basic pyplot

:param stocks: DataFrame having all the necessary data

:param title: Title of the plot

:param y\_label: yLabel of the plot

:param x\_label: xLabel of the plot

:return: prints a Pyplot againts items and their closing value

"""

fig, ax = plt.subplots()

ax.plot(stocks[ITEM], stocks[CLOSE\_PRICE], LIGHT\_BLUE)

ax.format\_ydata = price

ax.set\_title(title)

# Add labels

plt.ylabel(y\_label)

plt.xlabel(x\_label)

plt.show()

def plot\_prediction(actual, prediction, title=ACTUAL\_VS\_PREDICTION, y\_label=DEFAULT\_Y\_LABEL, x\_label=DEFAULT\_X\_LABEL):

"""

Plots train, test and prediction

:param actual: DataFrame containing actual data

:param prediction: DataFrame containing predicted values

:param title: Title of the plot

:param y\_label: yLabel of the plot

:param x\_label: xLabel of the plot

:return: prints a Pyplot againts items and their closing value

"""

fig = plt.figure()

ax = fig.add\_subplot(111)

# Add labels

plt.ylabel(y\_label)

plt.xlabel(x\_label)

# Plot actual and predicted close values

plt.plot(actual, GREEN, label=ACTUAL\_CLOSE)

plt.plot(prediction, BLUE, label=PREDICTED\_CLOSE)

# Set title

ax.set\_title(title)

ax.legend(loc=UPPER\_LEFT)

plt.show()

def plot\_lstm\_prediction(actual, prediction, title=ACTUAL\_VS\_PREDICTION, y\_label=DEFAULT\_Y\_LABEL, x\_label=DEFAULT\_X\_LABEL):

"""

Plots train, test and prediction

:param actual: DataFrame containing actual data

:param prediction: DataFrame containing predicted values

:param title: Title of the plot

:param y\_label: yLabel of the plot

:param x\_label: xLabel of the plot

:return: prints a Pyplot againts items and their closing value

"""

fig = plt.figure()

ax = fig.add\_subplot(111)

# Add labels

plt.ylabel(y\_label)

plt.xlabel(x\_label)

#list(range(1,prediction.shape))

# Plot actual and predicted close values

actual\_x = list(range(0,len(actual)))

prediction\_x = list(range(0,len(prediction)))

plt.plot(actual\_x,actual, GREEN, label=ACTUAL\_CLOSE)

plt.plot(prediction\_x,prediction, BLUE, label=PREDICTED\_CLOSE)

#plt.plot(actual,list(range(1,len(actual))), '#00FF00', label='Adjusted Close')

#plt.plot(prediction,list(range(1,len(prediction))), '#0000FF', label='Predicted Close')

# Set title

ax.set\_title(title)

ax.legend(loc=UPPER\_LEFT)

plt.show()

In [ ]:

plot\_basic(stocks,'Past Values - '+ symbol\_entered)

Step 3 : Normalise the data using minmaxscaler function

In [ ]:

stocks\_normalized = get\_normalised\_data(stocks)

get\_data\_stats(stocks\_normalized,features\_to\_use)

Step 4 : Visualize the data again

In [ ]:

plot\_basic(stocks\_normalized, title = 'Past Values (Normalized) -' + symbol\_entered, y\_label = 'Price - INR (normalized)')

Step 5: Log the normalised data for future resuablilty

In [ ]:

normalized\_name = symbol\_entered + '\_normalized.csv'

stocks\_normalized.to\_csv(normalized\_name,index= False)

Bench Mark Model[¶](file:///E:\GoogleDrive_sem1\GoogleDrive_sem1\sem%204\project\2019201015\Final%20Review\lstm_inputs_13042021.doc#Bench-Mark-Model)

In this section, let us try to fit linear regression model for the given data and consider it as benchmark model

Step 1: Load the preprocessed data

In [ ]:

import math

import pandas as pd

import numpy as np

from IPython.display import display

from sklearn import linear\_model

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error

from sklearn.model\_selection import TimeSeriesSplit

import visualize as vs

import stock\_data as sd

import LinearRegressionModel

stocks\_normalized = pd.read\_csv(normalized\_name)

display(stocks\_normalized.head())

Step 2: Split data into train and test pair

In [ ]:

import numpy as np

import math

def scale\_range(x, input\_range, target\_range):

"""

Rescale a numpy array from input to target range

:param x: data to scale

:param input\_range: optional input range for data: default 0.0:1.0

:param target\_range: optional target range for data: default 0.0:1.0

:return: rescaled array, incoming range [min,max]

"""

range = [np.amin(x), np.amax(x)]

x\_std = (x - input\_range[0]) / (1.0\*(input\_range[1] - input\_range[0]))

x\_scaled = x\_std \* (1.0\*(target\_range[1] - target\_range[0])) + target\_range[0]

return x\_scaled, range

def train\_test\_split\_linear\_regression(stocks):

"""

Split the data set into training and testing feature for Linear Regression Model

:param stocks: whole data set containing ['Open','Close','Volume'] features

:return: X\_train : training sets of feature

:return: X\_test : test sets of feature

:return: y\_train: training sets of label

:return: y\_test: test sets of label

:return: label\_range: scaled range of label used in predicting price,

"""

# Create numpy arrays for features and targets

feature = []

label = []

# Convert dataframe columns to numpy arrays for scikit learn

for index, row in stocks.iterrows():

# print([np.array(row['Item'])])

feature.append([(row[ITEM])])

label.append([(row[CLOSE\_PRICE])])

# Regularize the feature and target arrays and store min/max of input data for rescaling later

feature\_bounds = [min(feature), max(feature)]

feature\_bounds = [feature\_bounds[0][0], feature\_bounds[1][0]]

label\_bounds = [min(label), max(label)]

label\_bounds = [label\_bounds[0][0], label\_bounds[1][0]]

feature\_scaled, feature\_range = scale\_range(np.array(feature), input\_range=feature\_bounds, target\_range=[-1.0, 1.0])

label\_scaled, label\_range = scale\_range(np.array(label), input\_range=label\_bounds, target\_range=[-1.0, 1.0])

# Define Test/Train Split 80/20

split = .315

split = int(math.floor(len(stocks[ITEM]) \* split))

# Set up training and test sets

X\_train = feature\_scaled[:-split]

X\_test = feature\_scaled[-split:]

y\_train = label\_scaled[:-split]

y\_test = label\_scaled[-split:]

return X\_train, X\_test, y\_train, y\_test, label\_range

def train\_test\_split\_lstm(stocks, prediction\_time=1, test\_data\_size=450, unroll\_length=50):

"""

Split the data set into training and testing feature for Long Short Term Memory Model

:param stocks: whole data set containing ['Open','Close','Volume'] features

:param prediction\_time: no of days

:param test\_data\_size: size of test data to be used

:param unroll\_length: how long a window should be used for train test split

:return: X\_train : training sets of feature

:return: X\_test : test sets of feature

:return: y\_train: training sets of label

:return: y\_test: test sets of label

"""

# training data

test\_data\_cut = test\_data\_size + unroll\_length + 1

#print(stocks,stocks[0:-prediction\_time - test\_data\_cut])

x\_train = stocks[0:-prediction\_time - test\_data\_cut]

y\_train = stocks[prediction\_time:-test\_data\_cut][CLOSE\_PRICE]

# test data

x\_test = stocks[0 - test\_data\_cut:-prediction\_time]

y\_test = stocks[prediction\_time - test\_data\_cut:][CLOSE\_PRICE]

return x\_train, x\_test, y\_train, y\_test

def unroll(data, sequence\_length=24):

"""

use different windows for testing and training to stop from leak of information in the data

:param data: data set to be used for unrolling

:param sequence\_length: window length

:return: data sets with different window.

"""

result = []

for index in range(len(data) - sequence\_length):

result.append(data[index: index + sequence\_length])

return np.asarray(result)

In [ ]:

X\_train, X\_test, y\_train, y\_test, label\_range= train\_test\_split\_linear\_regression(stocks\_normalized)

print("x\_train", X\_train.shape)

print("y\_train", y\_train.shape)

print("x\_test", X\_test.shape)

print("y\_test", y\_test.shape)

Step 3: Train a Linear regressor model on training set and get prediction

In [ ]:

from sklearn import linear\_model

model = LinearRegressionModel.build\_model(X\_train,y\_train)

print('r2')

print(model.score(X\_train,y\_train))

Step 4: Get prediction on test set

In [ ]:

predictions = LinearRegressionModel.predict\_prices(model,X\_test, label\_range)

Step 5: Plot the predicted values against actual

In [ ]:

plot\_prediction(y\_test,predictions,y\_label = 'Price INR (normalized)')

Step 6: measure accuracy of the prediction

In [ ]:

trainScore = mean\_squared\_error(X\_train, y\_train)

print('Train Score: %.4f MSE (%.4f RMSE)' % (trainScore, math.sqrt(trainScore)))

testScore = mean\_squared\_error(predictions, y\_test)

print('Test Score: %.8f MSE (%.8f RMSE)' % (testScore, math.sqrt(testScore)))

Long-Short Term Memory Model[¶](file:///E:\GoogleDrive_sem1\GoogleDrive_sem1\sem%204\project\2019201015\Final%20Review\lstm_inputs_13042021.doc#Long-Short-Term-Memory-Model)

In this section we will use LSTM to train and test on our data set.

Basic LSTM Model[¶](file:///E:\GoogleDrive_sem1\GoogleDrive_sem1\sem%204\project\2019201015\Final%20Review\lstm_inputs_13042021.doc#Basic-LSTM-Model)

First lets make a basic LSTM model.

Step 1 : import keras libraries for smooth implementaion of lstm

In [ ]:

import math

import pandas as pd

import numpy as np

from IPython.display import display

from keras.layers.core import Dense, Activation, Dropout

from keras.layers.recurrent import LSTM

from keras.models import Sequential

from keras.metrics import mean\_squared\_error

from sklearn.model\_selection import StratifiedKFold

import lstm, time #helper libraries

#import visualize as vs

# import stock\_data as sd

In [ ]:

stocks\_normalized = pd.read\_csv(normalized\_name)

stocks\_data = stocks\_normalized.drop([ITEM], axis =1)

display(stocks\_data.head())

Step 2 : Split train and test data sets and Unroll train and test data for lstm model

In [ ]:

X\_train, X\_test,y\_train, y\_test = train\_test\_split\_lstm(stocks\_data, 5)

unroll\_length = 50

X\_train = unroll(X\_train, unroll\_length)

#print('after unroll: type(X\_train)')

#print(type(X\_train))

X\_test = unroll(X\_test, unroll\_length)

y\_train = y\_train[-X\_train.shape[0]:]

y\_test = y\_test[-X\_test.shape[0]:]

print("x\_train", X\_train.shape)

print("y\_train", y\_train.shape)

print("x\_test", X\_test.shape)

print("y\_test", y\_test.shape)

Step 3 : Build a basic Long-Short Term Memory model

In [ ]:

print(company)

In [ ]:

from datetime import datetime

#ct = datetime.datetime.now()

date = datetime. now(). strftime("%Y\_%m\_%d-%I:%M:%S\_%p")

print(f"filename\_{date}")

#print(str(ct))

In [ ]:

from datetime import datetime

# ct stores current time

#ct = datetime.datetime.now()

# build basic lstm model

lstm\_model = lstm.build\_basic\_model(input\_dim = X\_train.shape[-1],output\_dim = unroll\_length, return\_sequences=True)

# Compile the model

start = time.time()

lstm\_model.compile(loss='mean\_squared\_error', optimizer='adam')

print(filename)

#ct = datetime.datetime.now()

date = datetime. now(). strftime("%Y\_%m\_%d\_%I\_%M\_%S")

print(date)

lstm\_model.save(company+'\_model\_'+ str(date) +'.h5')

#pickle.dump(lstm\_model, open(filename+"\_model.pickle", 'wb'))

#filehandler = open(filename+"\_model.pickle",'w')

#print(pickle.dumps(lstm\_model).hex())#,filehandler)

print('compilation time : ', time.time() - start)

In [ ]:

from keras.models import load\_model

lstm\_model = load\_model('Tcs\_model.h5')

Step 4: Train the model

In [ ]:

lstm\_model.fit(

X\_train,

y\_train,

epochs=1,

validation\_split=0.05)

Step 5: make prediction using test data

In [ ]:

predictions = lstm\_model.predict(X\_test)

Step 6: Plot the results

In [ ]:

plot\_lstm\_prediction(y\_test,predictions)

Step 7: Get the test score.

In [ ]:

trainScore = lstm\_model.evaluate(X\_train, y\_train, verbose=0)

print('Train Score: %.8f MSE (%.8f RMSE)' % (trainScore, math.sqrt(trainScore)))

testScore = lstm\_model.evaluate(X\_test, y\_test, verbose=0)

print('Test Score: %.8f MSE (%.8f RMSE)' % (testScore, math.sqrt(testScore)))

Improved LSTM Model[¶](file:///E:\GoogleDrive_sem1\GoogleDrive_sem1\sem%204\project\2019201015\Final%20Review\lstm_inputs_13042021.doc#Improved-LSTM-Model)

Step 1: Build an improved LSTM model

In [ ]:

# Set up hyperparameters

batch\_size = 100

epochs = 5

# build improved lstm model

lstm\_model\_improved = lstm.build\_improved\_model( X\_train.shape[-1],output\_dim = unroll\_length, return\_sequences=True)

start = time.time()

#final\_model.compile(loss='mean\_squared\_error', optimizer='adam')

lstm\_model\_improved.compile(loss='mean\_squared\_error', optimizer='adam')

print('compilation time : ', time.time() - start)

Step 2: Train improved LSTM model

In [ ]:

lstm\_model\_improved.fit(X\_train,

y\_train,

batch\_size=batch\_size,

epochs=epochs,

verbose=2,

validation\_split=0.05

)

Step 3: Make prediction on improved LSTM model

In [ ]:

# Generate predictions

predictions = lstm\_model\_improved.predict(X\_test, batch\_size=batch\_size)

Step 4: plot the results

In [ ]:

plot\_lstm\_prediction(y\_test,predictions)

Step 5: Get the test score

In [ ]:

trainScore = lstm\_model\_improved.evaluate(X\_train, y\_train, verbose=0)

print('Train Score: %.8f MSE (%.8f RMSE)' % (trainScore, math.sqrt(trainScore)))

testScore = lstm\_model\_improved.evaluate(X\_test, y\_test, verbose=0)

print('Test Score: %.8f MSE (%.8f RMSE)' % (testScore, math.sqrt(testScore)))

In [ ]:

rangef = [np.amin(stocks\_data[CLOSE\_PRICE]), np.amax(stocks\_data[CLOSE\_PRICE])]

#Calculate the stock price delta in $

true\_delta = testScore\*(rangef[1]-rangef[0])

print('Delta Price: %.6f - RMSE \* Adjusted Close Range' % true\_delta)

# REFERENCES

https://en.wikipedia.org/wiki/Euronext\_Amsterdam