Analysis of Stock Market Forecasting Techniques

Aniket Gupta *Kunal Khetan*

2017EE10438 *2017MT60778*

1. INTRODUCTION

Over 70% of trades in US right now are being handled by bots. Algorithmic trading has revolutionized the stock market.

In this project, we worked with historical data of stocks of publicly listed companies. We implemented a variety of machine learning techniques to predict future stock prices using the prices of previous days.

First, we have given an overview of all the machine learning techniques that we have used. Then, we have shown our results on day-wise data[i] and 1-minute data[ii] of different companies.

2. SETUP

Source of day-wise data –

- NSE website - https://www1.nseindia.com/index_nse.htm
- Yahoo Finance - https://finance.yahoo.com/

Source of 1-minute tick data – Kaggle Dataset – www.kaggle.com/hk7797/stock-market-india/data

3. OVERVIEW OF ML TECHNIQUES USED

We worked upon mainly 3 different machine learning models to predict stock prices –

A. Auto-Regressive Integrated Moving Average (ARIMA) model
B. Long Short-Term Memory (LSTM) model
C. Evolution Strategy Agent (Reinforcement Learning)

A. Auto-Regressive Integrated Moving Average (ARIMA) model

ARIMA is a simple time series model that can be used to train and then forecast future time points. It can capture complex relationships as it takes error terms and observations of lagged terms. This model relies on regressing a variable on past values.

An ARIMA model is characterized by 3 terms: p, d, q where –

(a) p is the order of the Auto Regressive term. It refers to the number of lags of Y to be used as predictors.

(b) q is the order of the Moving Average term. It refers to the number of lagged forecast errors that should go into the ARIMA Model.
(c) d is the minimum number of differencing required to make the time series stationary. And if the time series is already stationary, then d = 0.

$$\Delta Y_t = \beta_1 \Delta Y_{t-1} + \beta_2 \Delta Y_{t-2} + \ldots + \beta_p \Delta Y_{t-p} + \varepsilon_t - \emptyset_1 \varepsilon_{t-1} - \emptyset_2 \varepsilon_{t-2} - \ldots - \emptyset_q \varepsilon_{t-q}$$

which can be written in words as

Predicted Y_t = Constant + Linear combination Lags of Y (up to p lags) + Linear Combination of Lagged forecast errors (up to q lags)

B. Long Short-Term Memory (LSTM) model

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. Unlike standard feedforward neural networks, LSTM has feedback connections. These networks are highly suitable for classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series.

LSTM cell (the memory part of the LSTM unit) components –

(a) Forget Gate "f_t" (a neural network with sigmoid) - Controls the extent to which a value remains in the cell
(b) Input Gate "I_t" (a NN with sigmoid) - Controls the extent to which a new value flows into the cell
(c) Output Gate "O_t" (a NN with sigmoid) - Controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit.

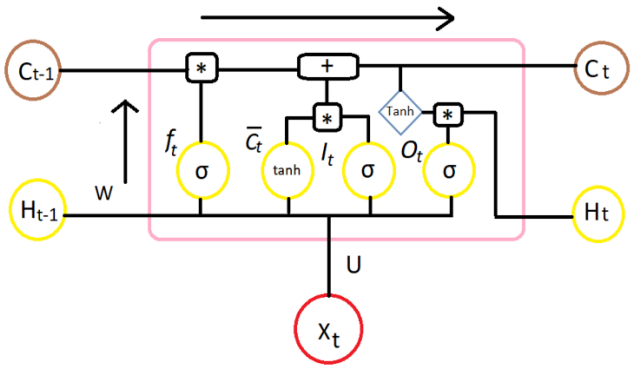


Figure 1: Source - https://medium.com/deep-math-machine-learning-ai/chapter-10-1-deepnlp-lstm-long-short-term-memory-networks-with-math-21477f8e4235

X_t = Input Vector | H_{t-1} = Previous cell output | C_{t-1} = Previous cell memory
H_t = Current cell output | C_t = Current cell memory

Here is a mathematical representation of the gates –

$$f_t = \sigma(X_t * U_f + H_{t-1} * W_f)$$

$$\bar{C}_t = \tanh(X_t * U_c + H_{t-1} * W_c)$$

$$I_t = \sigma(X_t * U_i + H_{t-1} * W_i)$$
$$O_t = \sigma(X_t * U_o + H_{t-1} * W_o)$$
$$C_t = f_t * C_{t-1} + I_t * \bar{C}_t$$
$$H_t = O_t * \tanh(C_t)$$

Usually, the activation function of the forget, input and output gates is the sigmoid function. A Recurrent Neural Network using LSTM units can be trained on a collection of training data, by applying an optimization algorithm, like gradient descent, combined with backpropagation in order to change each weight of the LSTM network in balance to the derivative of the error with respect to corresponding weight.

C. Evolution Strategy Agent (Reinforcement Learning)

We can define[iii] an evolution strategy as an algorithm that provides the user a set of candidate solutions to evaluate a problem. The evaluation is based on an *objective function* that takes a given solution and returns a single *fitness* value. Based on the fitness results of the current solutions, the algorithm will then produce the next generation of candidate solutions that is more likely to produce even better results than the current generation. The iterative process will stop once the best-known solution is satisfactory for the user.

Here is a summary of how the agent works –

(a) In this strategy, we have a model initialized with random weights. The input to the model is the close prices of the past k many days, where k is the window size to be set, and the output is the decision i.e. Buy, Sell or Hold.
(b) We create a population of weights of size n by adding random noise to the weights. Then we do stock prediction using all the weights in the population and get rewards for each member of the population. The *objective function* is the reward function which is simply the profit. The reward is the *fitness value*. We update the weights according to this formula –

$$\theta_{t+1} \leftarrow \theta_t + \frac{\alpha}{n\sigma} \sum_{j=1}^{n} F_j \epsilon_j$$

Here θ is the weight matrix. α is the learning rate. F_j is the reward for member j of the population. ε_j is the noise added to member j of the population. n is the size of the population and σ is a factor by which we had multiplied all the noises.
(c) We call each iteration a generation. In every generation, the weights get better and better.

4. EXPERIMENTS AND RESULTS

A. Auto-Regressive Integrated Moving Average (ARIMA) model

a) We used 90% of the data to train our ARIMA model and tested our predictions on the remaining 10% of the data.

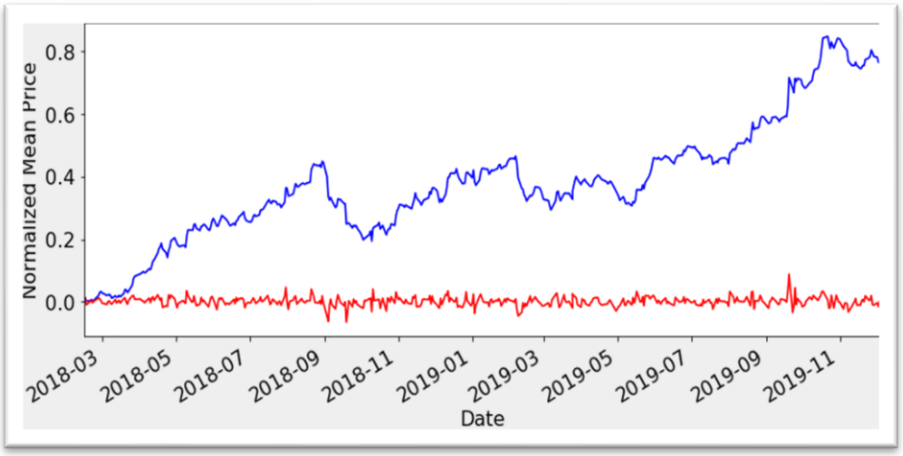
b) We scaled all the data so that it lies in the range 0 to 1

$$X_{std} = (X - X_{min})/(X_{max} - X_{min})$$

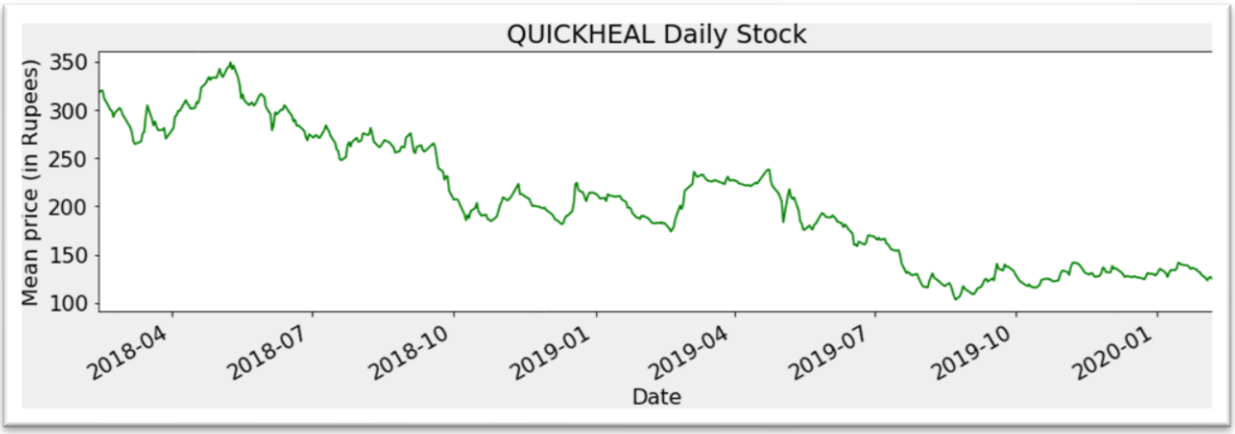
c) To check if the data is stationary, we used Augmented Dickey-Fuller test.
Suppose, the series is of the form –

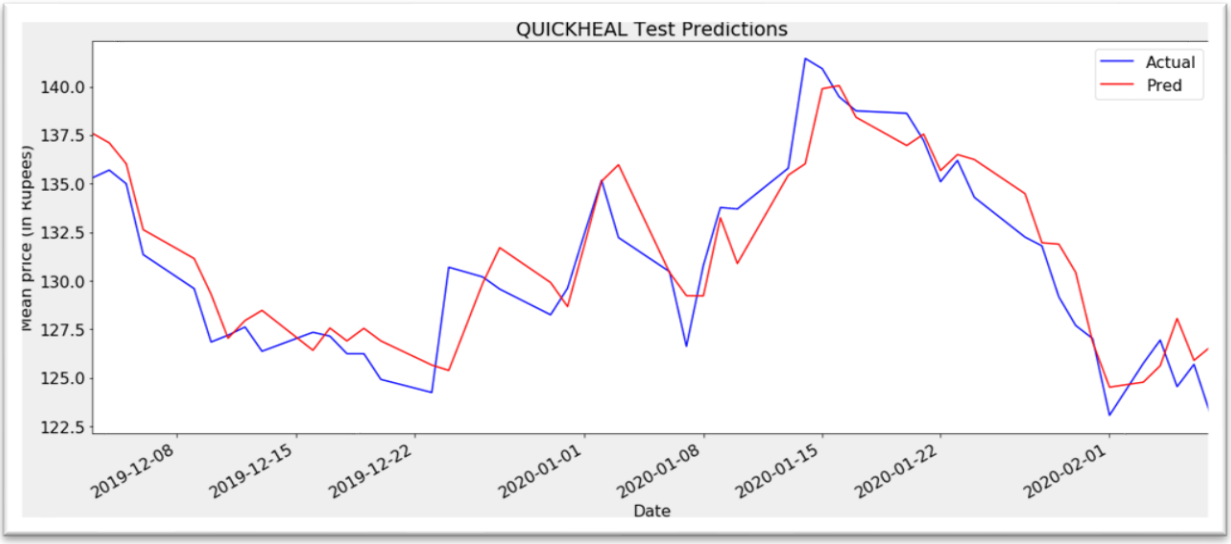
$$Y_t = \alpha + \emptyset Y_{t-1} + e_t$$
$$\Delta Y_t = Y_t - Y_{t-1} = \alpha + \gamma Y_{t-1} + e_t$$

For the series to be stationary, γ should be zero, ΔY_t will be a random walk.
We use this test to find the parameter d in ARIMA. The following is the result of this test on *Nestle* stocks. The blue series is the original time series. The red series represents the series after one differencing. We can see that after one level of differencing, the series is stationary.



Following are the results obtained on training the agent on different Indian companies' stocks –

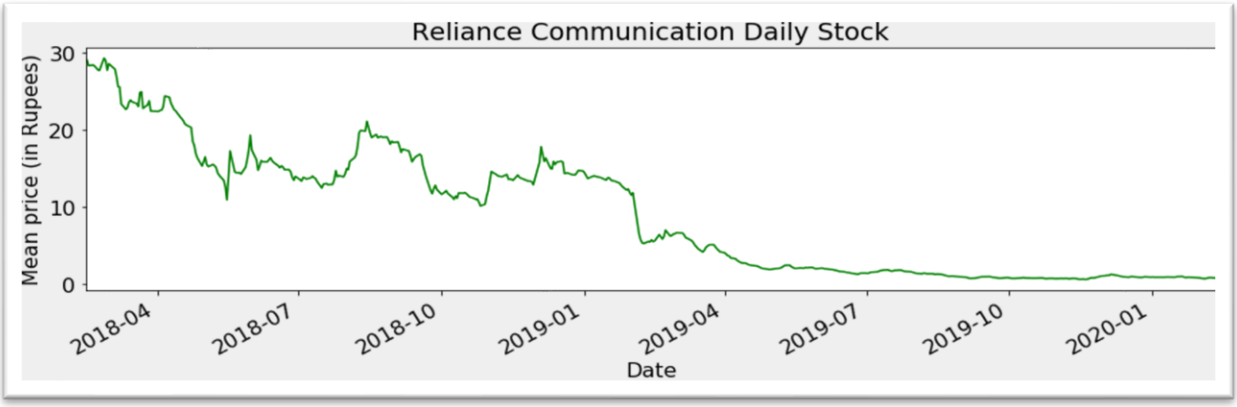


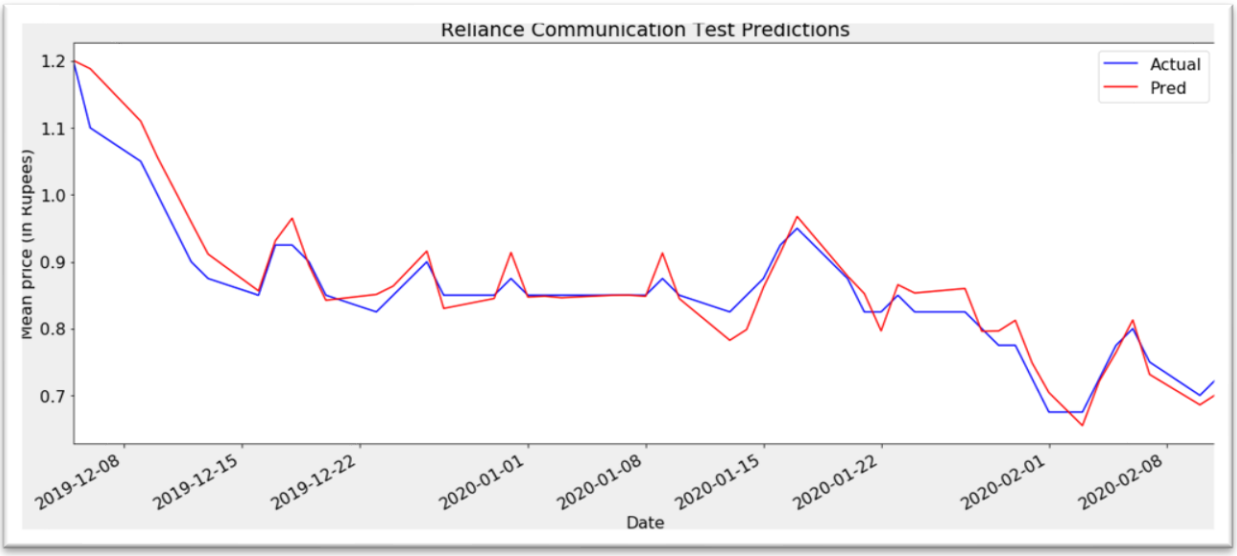


Mean Absolute Percentage Error – 1.14%

Normalized Root Mean Squared Error – 0.011

Correlation Coefficient - 0.92

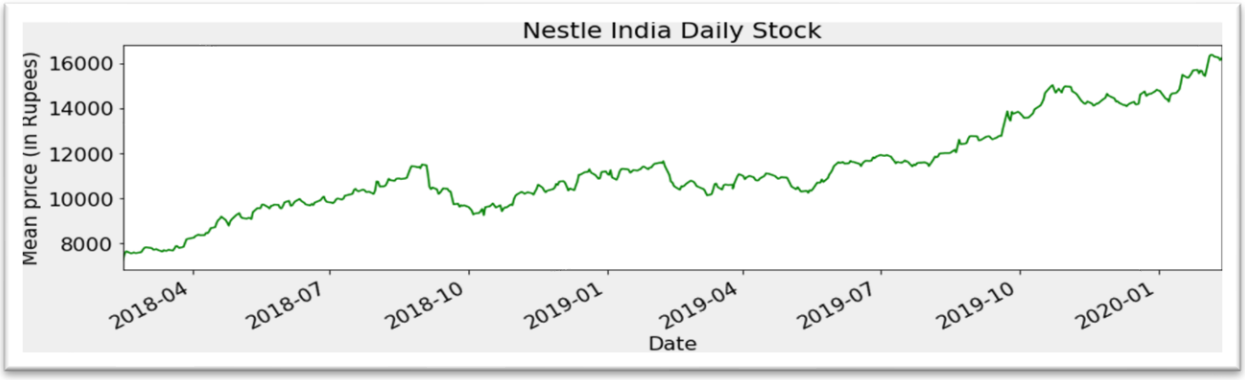


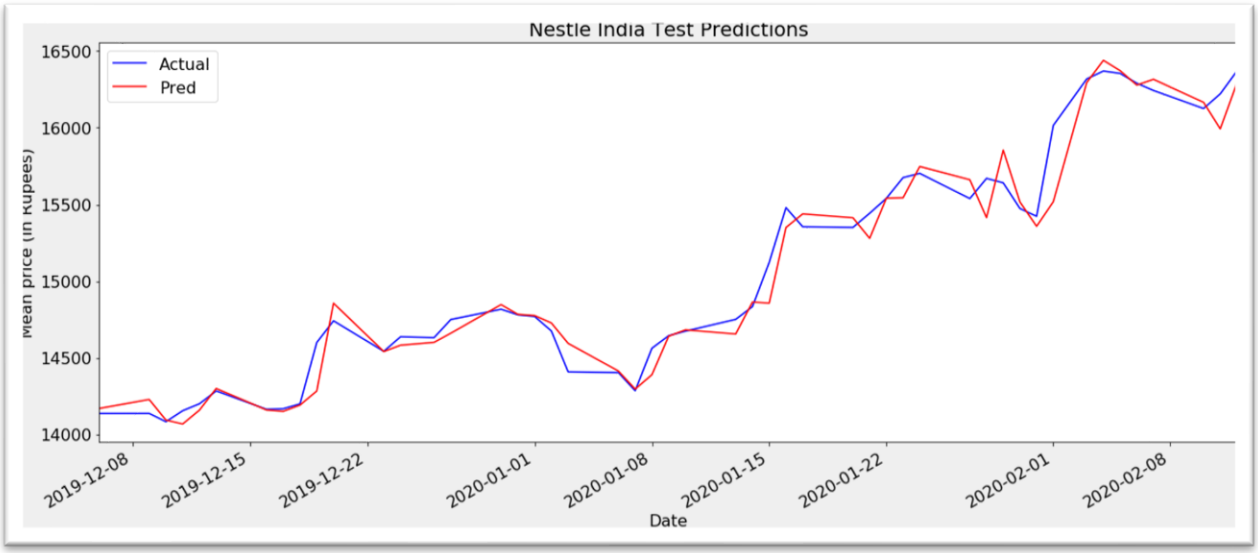


Mean Absolute Percentage Error – 2.56%

Normalized Root Mean Squared Error – 0.034

Correlation Coefficient - 0.97

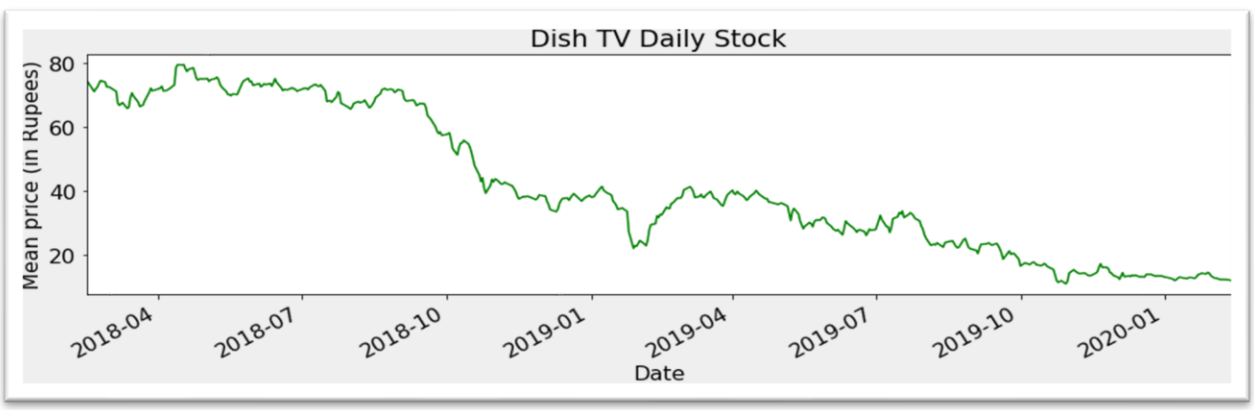


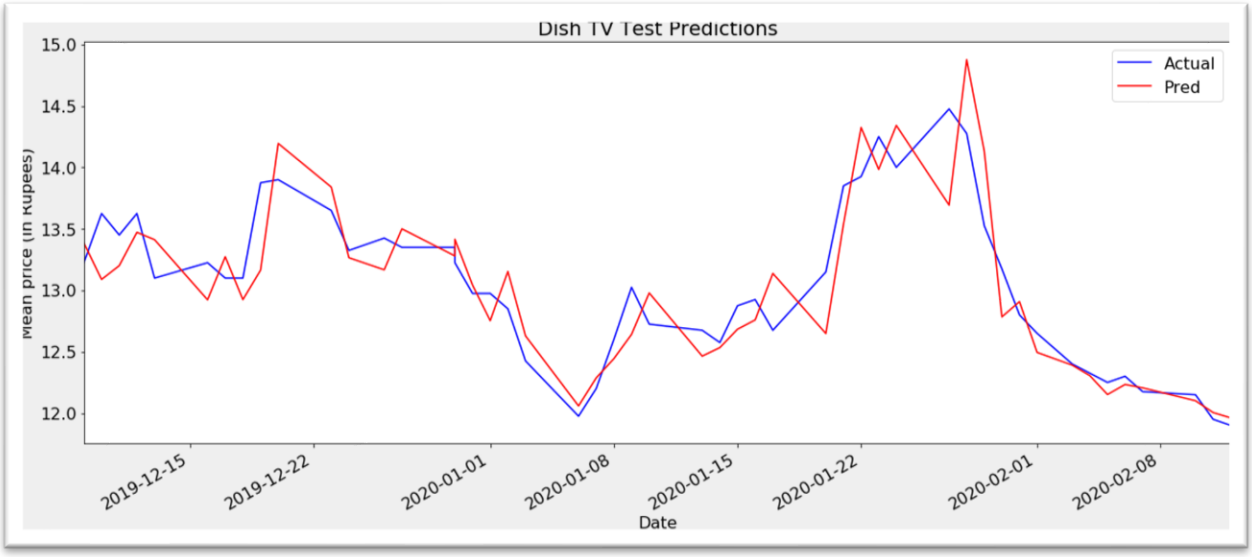


Mean Absolute Percentage Error – 0.55%

Normalized Root Mean Squared Error – 0.008

Correlation Coefficient - 0.98





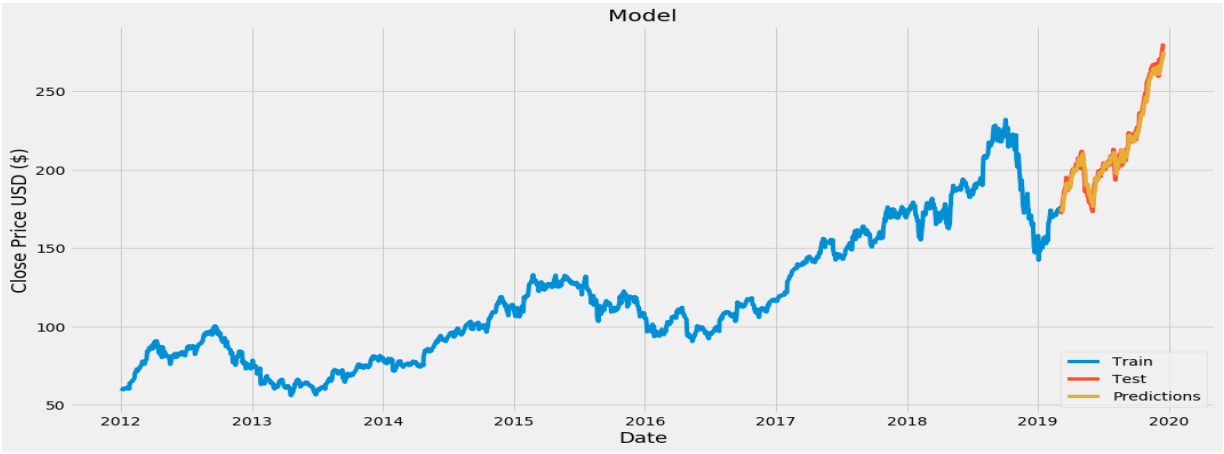
Mean Absolute Percentage Error – 1.78%

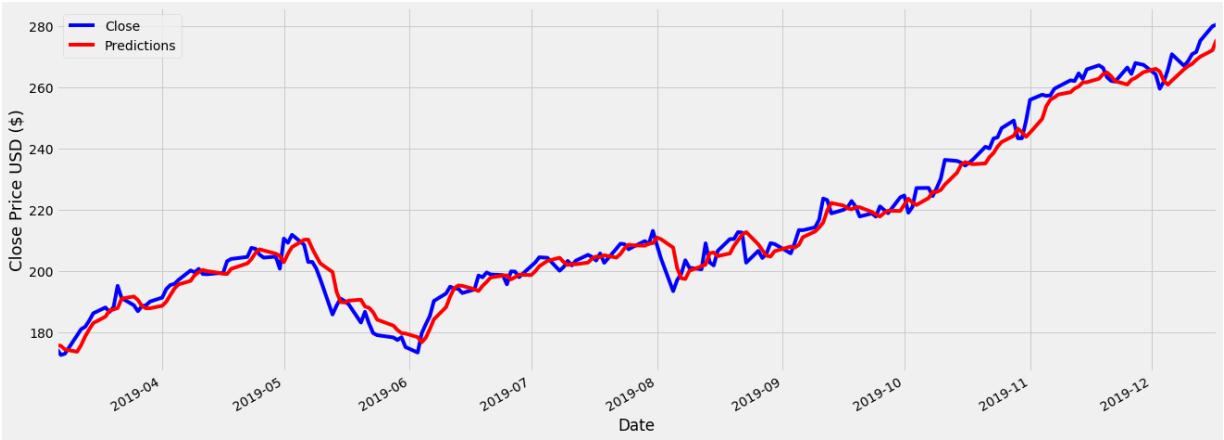
Normalized Root Mean Squared Error – 0.023

Correlation Coefficient - 0.89

B. <u>Long Short-Term Memory (LSTM) model</u>

Apple | Simple LSTM model | Day-wise Data | (90-10 Train-Test ratio)



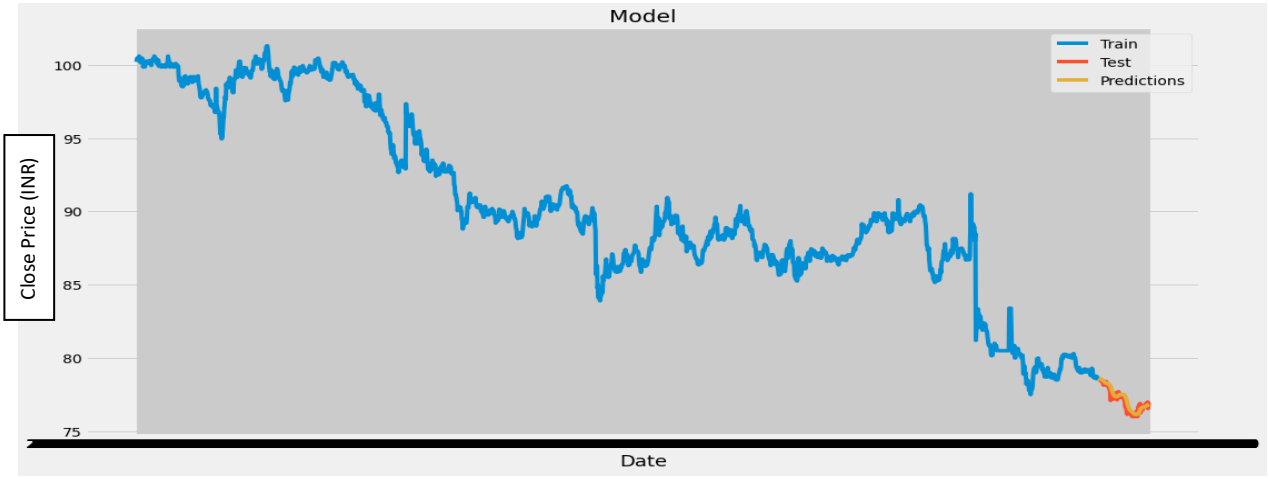


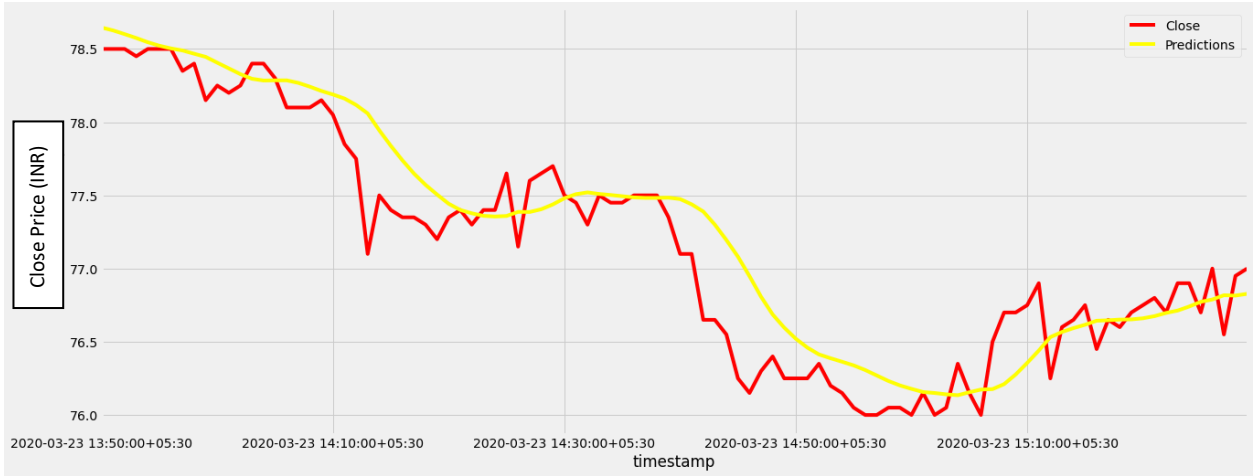
Correlation Coefficient = 0.991

Normalized Root Mean Squared Error = 0.208

Root Mean Squared Error = 3.837

Apollo Tyre | Simple LSTM model | 1-minute data | (90-10 Train-Test ratio)





Correlation Coefficient = 0.943

Normalized Root Mean Squared Error = 0.0581

Root Mean Squared Error = 0.284

Observation:

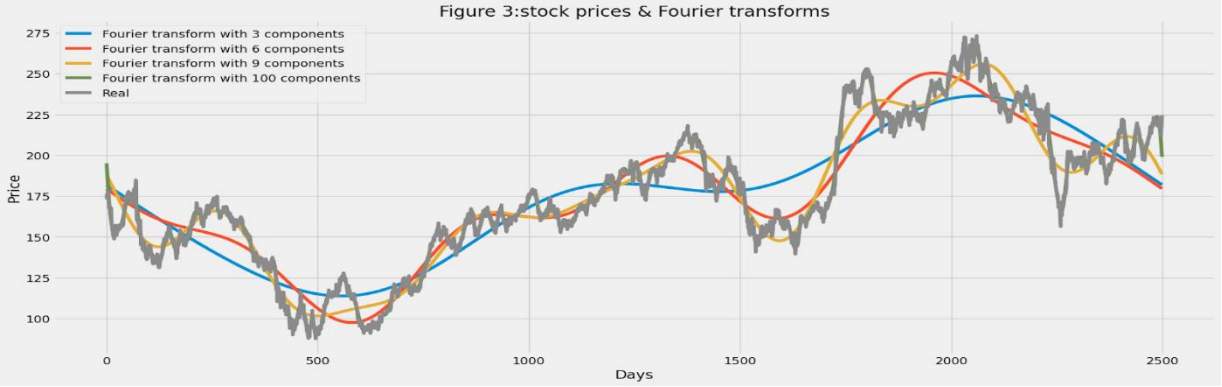
Training an LSTM model is much more difficult on 1-minute data as there is too much fluctuation in the stock price.

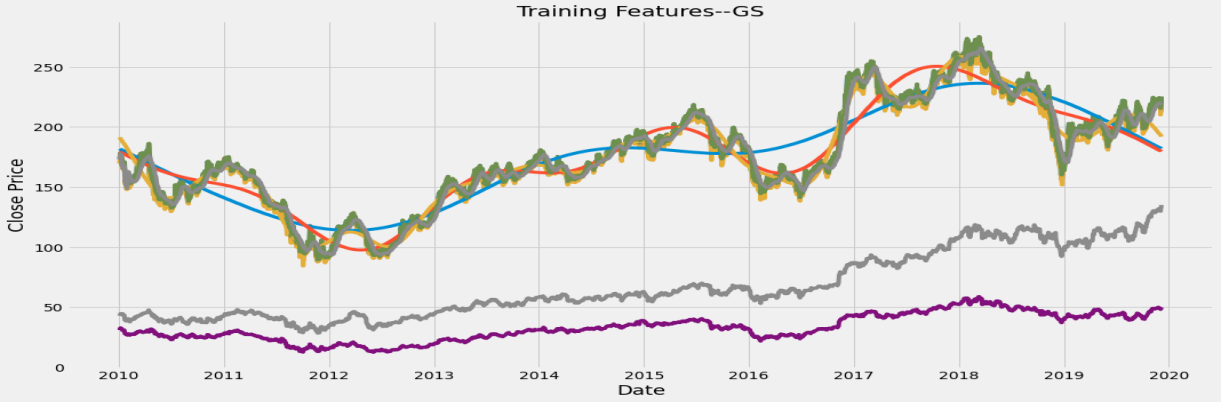
Now, we tried to make a few changes to improve our LSTM model. We worked on day-wise data with much more fluctuations. In data with less fluctuations, a simple LSTM model with closing price as input performs well as we see in case of Apple.

We made a new LSTM model for Goldman Sachs stocks -

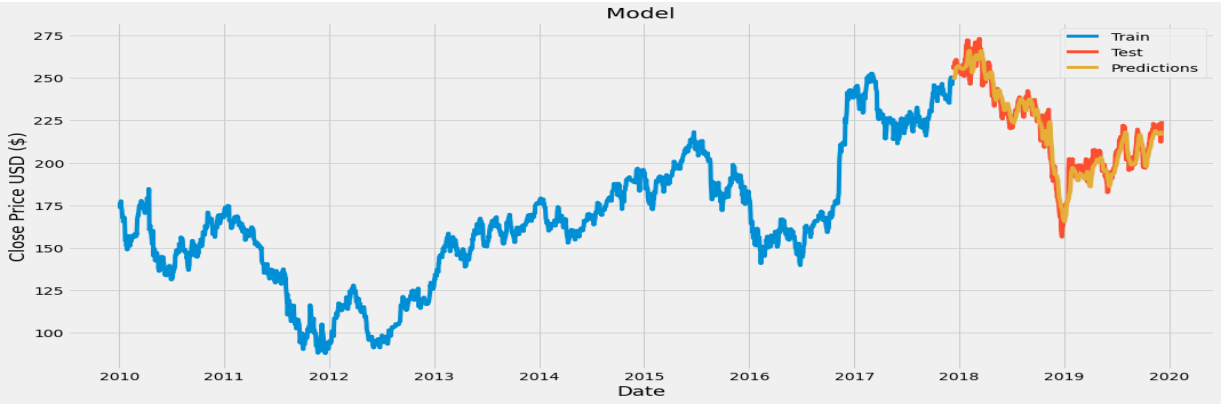
a) Since companies' stocks depend on other companies too, we considered the stock of other companies while training. We added JP Morgan and Morgan Stanley stocks' closing price as a feature to the training dataset.
b) We included popular indicators such as moving average, exponential moving average etc. as independent features.
c) We used Fourier Transform to eliminate a lot of noise and create approximations of the real stock movement. Having trend approximations can help the LSTM network pick its prediction trends more accurately. We used Fourier transform with 3,6,9,100 components as 4 different features to train our model.

Our experiments & results on Goldman Sachs day-wise data (80-20 Train-Test ratio) –





(Purple – Morgan Stanley) (Grey – JP Morgan)



Correlation Coefficient = 0.966

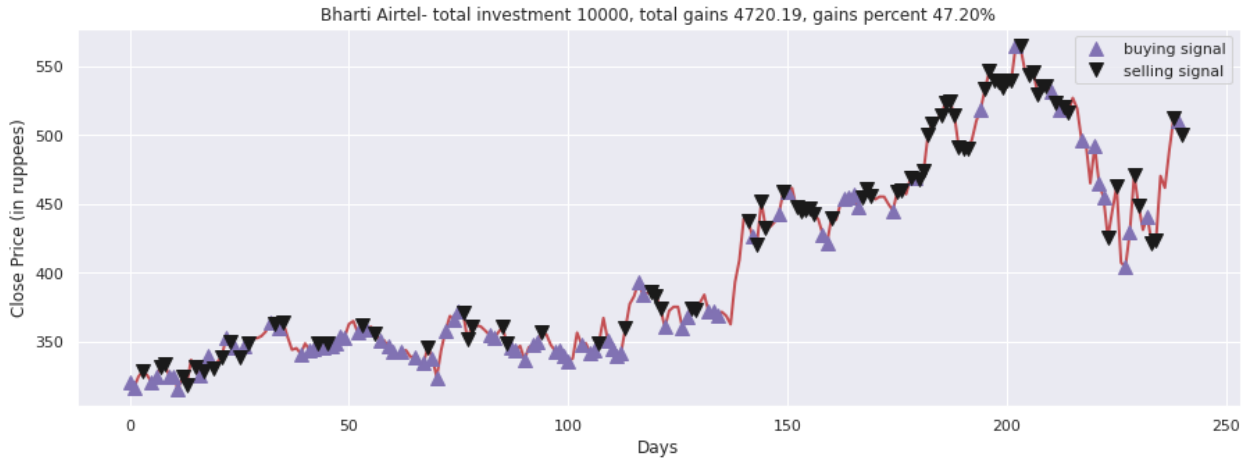
Normalized Root Mean Squared Error = 35.38

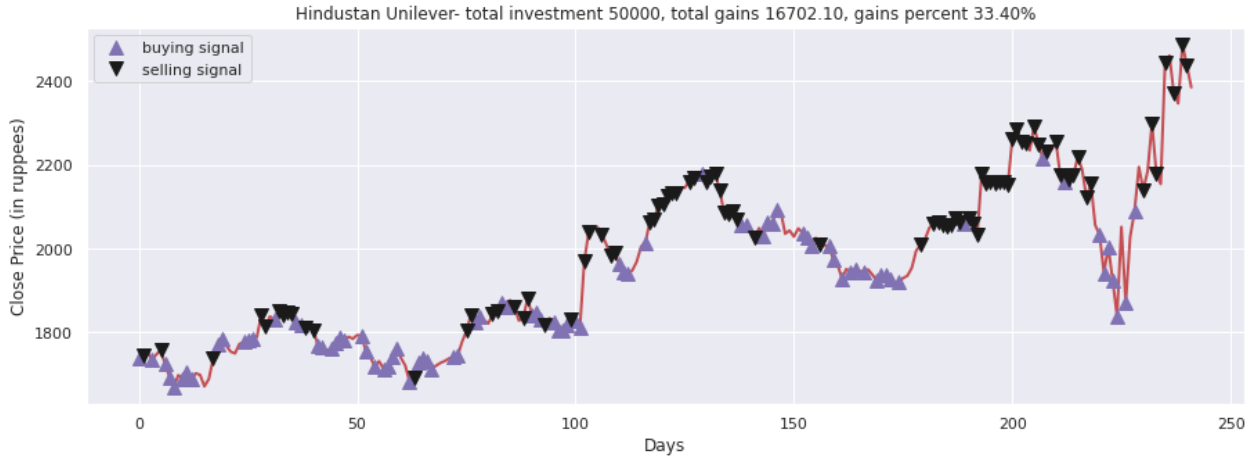
Root Mean Squared Error = 0.190

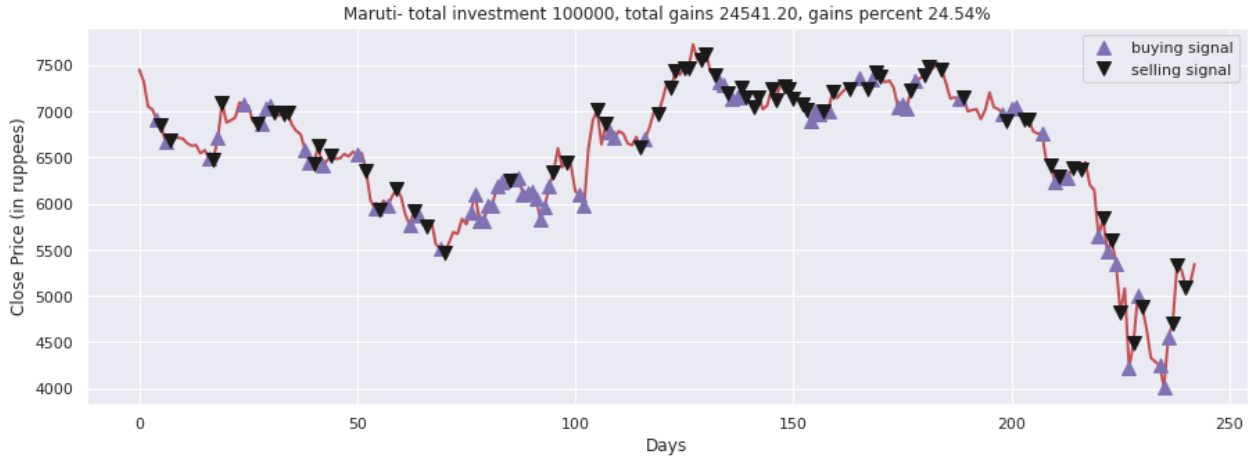
C. <u>Evolution Strategy Agent (Reinforcement Learning)</u>

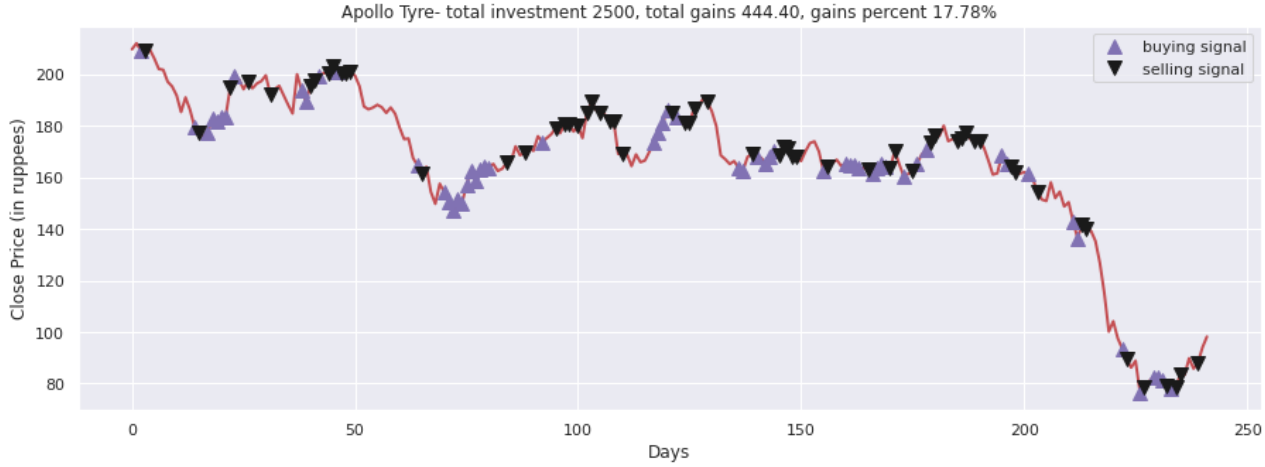
The agent was given some initial balance which it could use to buy or sell stocks. We find the total gains that the agent makes after trading after a few days -

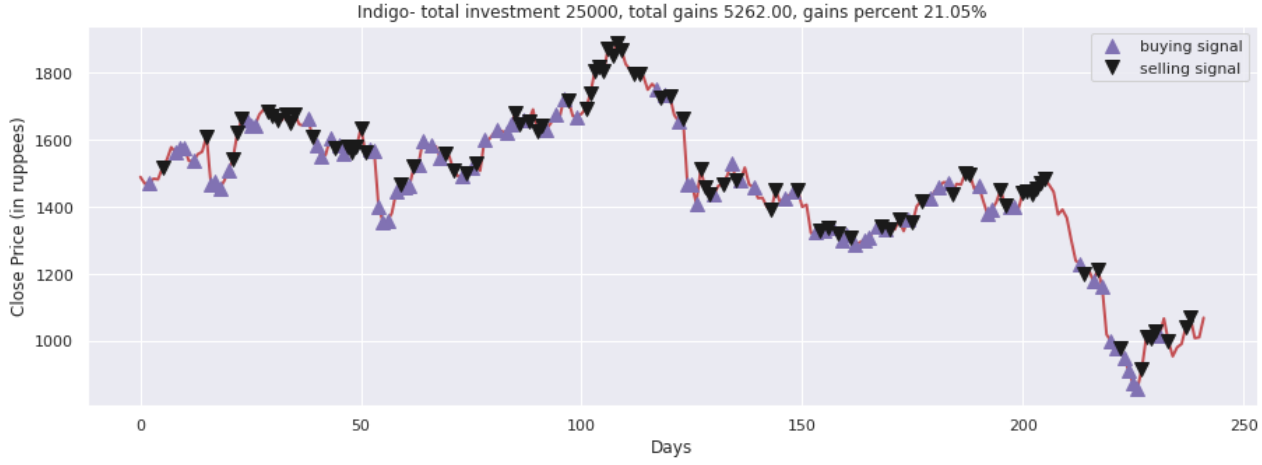
Following are the results obtained on training the agent on different Indian companies' stocks –






Maruti- total investment 100000, total gains 24541.20, gains percent 24.54%
buying signal
selling signal
Close Price (in ruppees)
Days
7500
7000
6500
6000
5500
5000
4500
4000
0
50
100
150
200
250


Apollo Tyre- total investment 2500, total gains 444.40, gains percent 17.78%
buying signal
selling signal
Close Price (in ruppees)
Days
200
180
160
140
120
100
80
0
50
100
150
200
250


Indigo- total investment 25000, total gains 5262.00, gains percent 21.05%
buying signal
selling signal
Close Price (in ruppees)
Days
1800
1600
1400
1200
1000
0
50
100
150
200
250

5. DISCUSSION

Previously we were not able to predict the very high fluctuations but by using correlated asset, technical indicators, denoising (Fourier Transform), we were able to predict them.

Observations related to ARIMA and LSTM:

a) ARIMA yields better results in forecasting short term, whereas LSTM yields better results for long term modelling.
b) LSTM is undoubtedly more complicated and difficult to train and, in most cases, does not exceed the performance of a simple ARIMA model.
c) As LSTMs are equipped to learn long term correlations in a sequence, they can model complex multivariate sequences without the need to specify any time window.

Now, to use both ARIMA and LSTM in a single model as an ensemble, we propose the following:

As we know that a time series equation can be decomposed into three components-Trend, Cyclic/Seasonal, Random Noise. First describes the overall movement while the second points to periodic fluctuations. However, we have also seen that ARIMA is effective in estimating the seasonal component. Cyclic, this sequence keeps the same values for a given period. On the other hand, it is this seasonal component (in addition to noise to a lesser extent) that makes the trajectory of the time series erratic and volatile. Therefore, the idea is to transform the time series into smooth function(trend) by isolating it from seasonal component using ARIMA model

$Y_t = m_t + s_t + \varepsilon_t$

Three functions are obtained: the trend m_t, the seasonal component s_t and the noise ε_t. s_t and ε_t are retained to reproduce seasonality and put the predicted values into confidence intervals using ε_t.

Now, m_t is a smooth curve. Therefore, we can use neural network to predict it. In addition, we take the best performance of both ARIMA and LSTM models: the first is optimal for the seasonal component and the second is effective for the trend. Once this is done, we can cross-check the predicted trend m_t and the reproduced cyclical component st to construct a forecast of the original series Y_t.

Observations related to Evolution Strategy:

The evolution training agent performs really well and has earned approximately 20% profit or more across 250 days in most datasets.

It is not possible to compare the results of ARIMA and LSTM with the Evolution Strategy Agent. This is because the evolution strategy agent is trained to make a decision, i.e. whether to Buy, Sell or Hold. But, in ARIMA and LSTM, we predict the values of the time series using the past data points. If we use these predicted values to make decisions, we would have to apply a new algorithm altogether which would have to make decisions on how long to hold before selling etc.

[i] finance.yahoo.com
[ii] kaggle.com/hk7797/stock-market-india/data
[iii] https://blog.otoro.net/2017/10/29/visual-evolution-strategies/