Nifty techniques for predicting NIFTY:
A report on efficient stock prediction approaches.

A Project Report

Submitted by

Arhya Singh

Darshil Shah

Rushank Shah

Yash Oza

Under the Guidance of
Prof. Abhay Kolhe and Prof Archana Nanade

in partial fulfillment for the award of the degree of

BTech Integrated
IN THE BRANCH OF STUDY
Computer engineering

At

MPSTME, NMIMS.
March 2024

Table of contents

1. Introduction

Stock market or stock exchange is essentially the buying and selling of shares belonging to publicly held companies. In India trading is enabled by the National Stock Exchange (NSE), controlled and operated under stringent regulations by the Securities and Exchange Board of India (SEBI).

NIFTY, often known as the National Stock Exchange and Fifty, is a market index. The top 50 equity stocks traded in the market are listed in the benchmark-based NIFTY 50 index. The largest and most liquid Indian equities, or blue-chip companies — profitable, steady enterprises that are regarded as safe investments in their respective industries — are tracked by the NIFTY 50 index. The twelve Indian economic sectors that these stocks cover are information technology, consumer goods, financial services, media and entertainment, metals, pharmaceuticals, telecommunications, cement and related products, cars, energy, fertilizers and pesticides, and other services.

Every six months, the NIFTY Index is reconstructed using stock performance as a basis. Fulfilling liquidity requirements and financial norms are among the eligibility conditions. Index funds and ETFs are subject to quarterly screenings to verify compliance with SEBI laws on portfolio concentration.

To manage risk and make well-informed decisions, traders and investors need to be able to predict the NIFTY 50 index. It influences trading in derivatives and investment decisions by reflecting the mood of the market and the state of the economy. Precise forecasts support corporate decision-making and economic analysis, enabling a range of financial operations in the Indian stock market.

Univariate analysis is a statistical method for analyzing a singular variable. In context of NIFTY 50, we use the closing price as our single variable, predicting its trend without considering its relationship with other variables like Opening, High, Low, Volume etc. Through univariate analysis, we can understand NIFTY trends over a certain period of time.

In this report, we propose and discuss various methods for predicting trends in the NIFTY index by means of univariate operations.

2. Literature Survey

[1] used a variety of neural network models like LSTM, RNN, GRU and Transformers for multivariate time series data prediction and used evaluation metrics like RMSE and MAE.

[2] ARIMA model is widely used in time series analysis, it has variety of hyperparameters that let the programmer cover all the possibilities like seasonality. This paper explained why it is important to choose correct parameters for ARIMA model and how significantly the performance can be improved if we increase the training set over time and refit the model. This enables the model to predict the values with more accuracy as it is has more up to date data.

[3] proposed the Facebook Prophet model which allows user the to perform prediction on time series data and to customize the model according to data keeping in mind the various frequent and non-frequent changes. [4] uses Facebook prophet and ARIMA models for stock price prediction.

[5] Transformers came to popularity for Language translation and became a breakthrough in every field. Their ability to use the multi-head attention to incorporate large sequences make them the perfect candidate for time series forecasting. [6] uses Transformer to predict the stock prices of China's energy vehicle industry

[7] The authors argue that while some believe this prediction is impossible, their work demonstrates achieving some accuracy using historical data . They collected data on NIFTY 50 stock prices and used it to train various models including multivariate linear regression and LSTM models. Their findings suggest that a univariate LSTM model, considering one week of prior data, is the most accurate for predicting the daily opening value of the NIFTY 50 index.

[8] This paper focused on prediction of the stock prices of a company using normal RNNs, LSTM, GRU models. The findings from this paper indicated that both LSTM and GRU models achieved good results, GRU even emerged as a better and more favorable model since it uses less parameters and had a faster processing speed.

3. Proposed Work

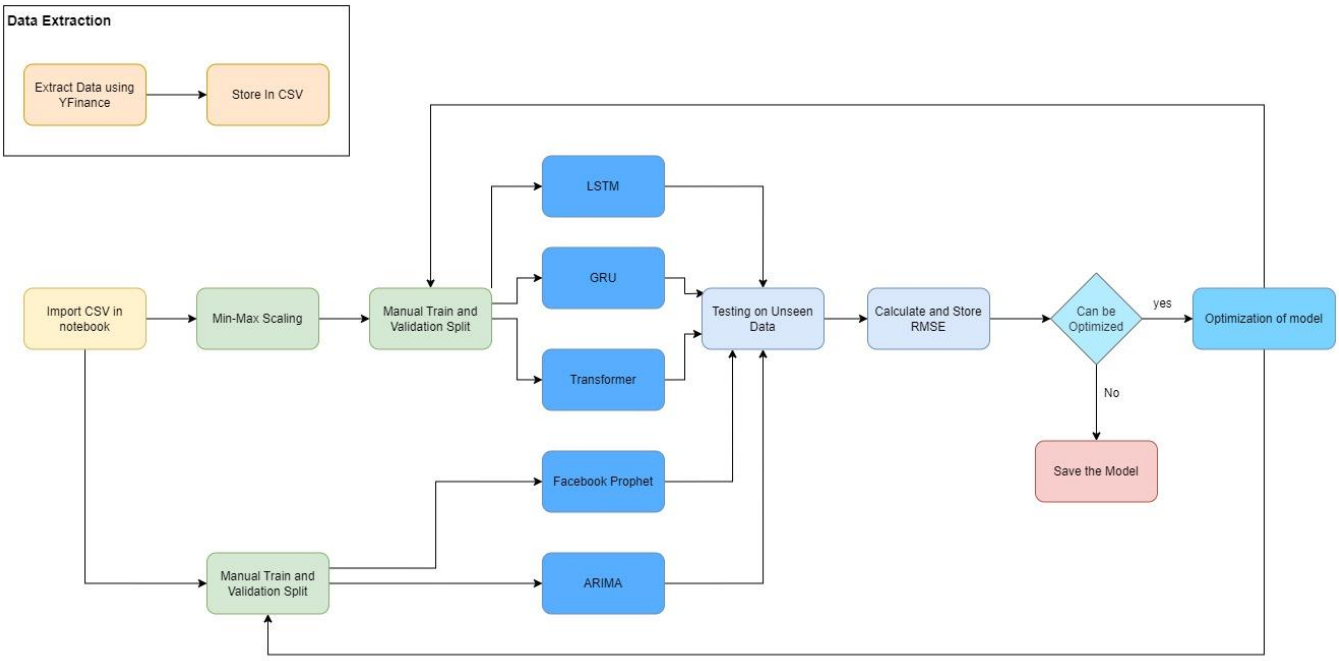


Figure 1: Project Pipeline

The flowchart depicts our approach towards data extraction and training different models. We are performing manual train and validation split because we are training on time series data, where the sequence is vital and to avoid false trend analysis and prediction due to random shuffling. The reason we are not scaling the data for ML models is that both ARIMA and Facebook Prophet assume data is stationary. Scaling will not affect the stationarity of data but can introduce unnecessary transformations that could mask the underlying pattern the model is trying to capture.

3.1 About Dataset

We are extracting Nifty 50 Price data using Yahoo Finance (yfinance) library in Python. Yfinance gives us the flexibility to extract data for a specific range and also lets us define the interval. Currently, in this project, we are training the models on daily data.

Dataset contains the following columns

- **Date**: date for which the prices are saved
- **Open**: opening price/starting price of the day
- **High**: Highest price nifty touched on that day
- **Low**: lowest price nifty touched on that day
- **Close**: closing price of the day
- **Volume**: total number of shares bought or sold on that day

We are training the model on data from 1st January 2020 till 31st December 2023 and testing it on data from 1st January 2024 till 15th March 2024. We are using lookback value as 7 during training

the models i.e. the model will predict the current value based on the past 7 values. Since we are planning to use it for short term predictions, we decided to keep the lookback value small.

3.2 Arima Model

ARIMA stands for Autoregressive Integrated Moving Average. ARIMA models assume that the future will be similar to the past. It is autoregressive as future values of a changing variable are based on its own past values, Integrated is nothing but differencing, sometimes, data has trends and jumps which makes it non-stationary. This makes it difficult to analyze, therefore, we basically take the difference between each data point and the one before it. This can help flatten out trends and make the data more stable for analysis. A moving average is like smoothing out that noise by taking an average of past errors (the difference between the expected value and the actual value). This can help us see the underlying pattern of the data more clearly.

3.2.1 Arima Model Parameters:

p: autoregressive term, it considers the impact of the past p values on the current value.

d: degree of differencing, this parameter addresses non-stationarity in the data, meaning the trend or seasonality changes over time. Differencing involves subtracting the previous value from the current value. The value of d indicates how many times this differencing is performed to achieve stationarity. To fit arima, we need time series to be stationary, d is responsible for making it stationary.

q: moving average term, size of the moving average window. It considers the average of past residuals (errors) to account for randomness in the data.

3.2.2 Best performing model

After performing optimization, the SARIMAX model with parameters, p=0, d=1 and q=0 had the best performance. SARIMAX is a Seasonal Arima model with exogenous variables. Since we are keeping seasonality as false, it is the same as the normal Arima Model. Seasonal is kept false as we want to focus on short term predictions.

Stock data, being very volatile, does not necessarily have any seasonal patterns, it is very random.

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=11766.140, Time=0.05 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=11768.136, Time=0.09 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=11768.139, Time=0.11 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=11766.233, Time=0.03 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=11770.139, Time=0.15 sec

Best model:  ARIMA(0,1,0)(0,0,0)[0] intercept
Total fit time: 0.476 seconds
                               SARIMAX Results
================================================================================
Dep. Variable:                      y   No. Observations:                  900
Model:               SARIMAX(0, 1, 0)   Log Likelihood               -5881.070
Date:                Mon, 18 Mar 2024   AIC                          11766.140
Time:                        14:27:41   BIC                          11775.743
Sample:                             0   HQIC                         11769.808
                                  - 900
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      8.1007      5.937      1.364      0.172      -3.536      19.737
sigma2      2.816e+04    762.513     36.932      0.000    2.67e+04    2.97e+04
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):               987.30
Prob(Q):                              0.92   Prob(JB):                         0.00
Heteroskedasticity (H):               0.47   Skew:                            -0.87
Prob(H) (two-sided):                  0.00   Kurtosis:                         7.83
===================================================================================
```

Figure 2: ARIMA Best Performing Model

3.2.3 How we got to best model

On trying to calculate the values of p, d and q using the graphs, the model performance was not better than previous model. Performing GridSearchCV gave the parameters which performed better on validation set had poor performance on test set (unseen data). Auto Arima model tries to consider all the possibilities and gives the best performing model parameters based on AIC and BIC.

3.2.4 Diagnostic Graphs of Arima Model

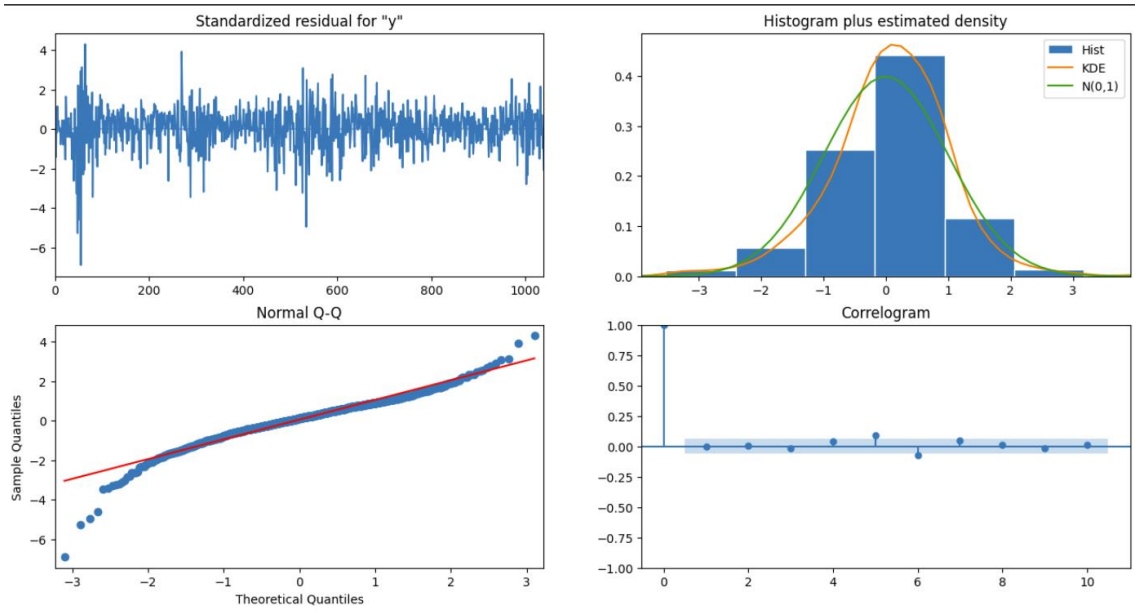


Figure 3: ARIMA Diagnostic Graphs

Standardized Residuals Plot: This plot shows the standardized residuals over time. Ideally, the residuals should fluctuate randomly around a zero mean with no discernible patterns. This indicates that the model's errors are random and not systematically biased.

Correlogram: This plot depicts the correlation of the residuals with their lagged versions (shifted versions of themselves). Ideally, this plot should show no significant spikes at lags beyond the first few, indicating that there are no remaining autocorrelations in the residuals.

QQ Plot (Quantile-Quantile Plot): This plot compares the distribution of the standardized residuals with a normal distribution. The points should roughly follow a straight diagonal line, indicating that the residuals are normally distributed. This is a desired characteristic because many statistical tests rely on the normality of residuals.

3.3 LSTM

LSTMs are Long Short Term Memory Neural Network. By having a longer short term memory it aims to eliminate the vanishing gradients problem faced by recurrent neural networks, The short term memory in question is the network's ability to remember past knowledge and discarding irrelevant data. LSTMs can capture long term dependencies. Having the ability to deal with larger 'gaps' between relevant information is what makes LSTMs have an edge over RNNs. We have leveraged this ability of LSTMs in our prediction models, as we are dealing with time series data.

3.3.1 LSTM Parameters

Input Size (input_size): This is the dimensionality of the input features.

Hidden Size (hidden_size): This parameter determines the number of units in each LSTM hidden state and cell state.

Number of Stacked Layers (num_stacked_layers): This parameter defines how many LSTM layers are stacked on top of each other.

Dropout (dropout): Dropout is a regularization technique used to prevent overfitting by randomly setting a fraction of input units to zero during training.

3.3.2 LSTM Architecture

Best model

4 layers, 2 Stacks, no dropout,

0.01 LR, Optimizer Adam,

25 epochs, with Bayesian Optimization

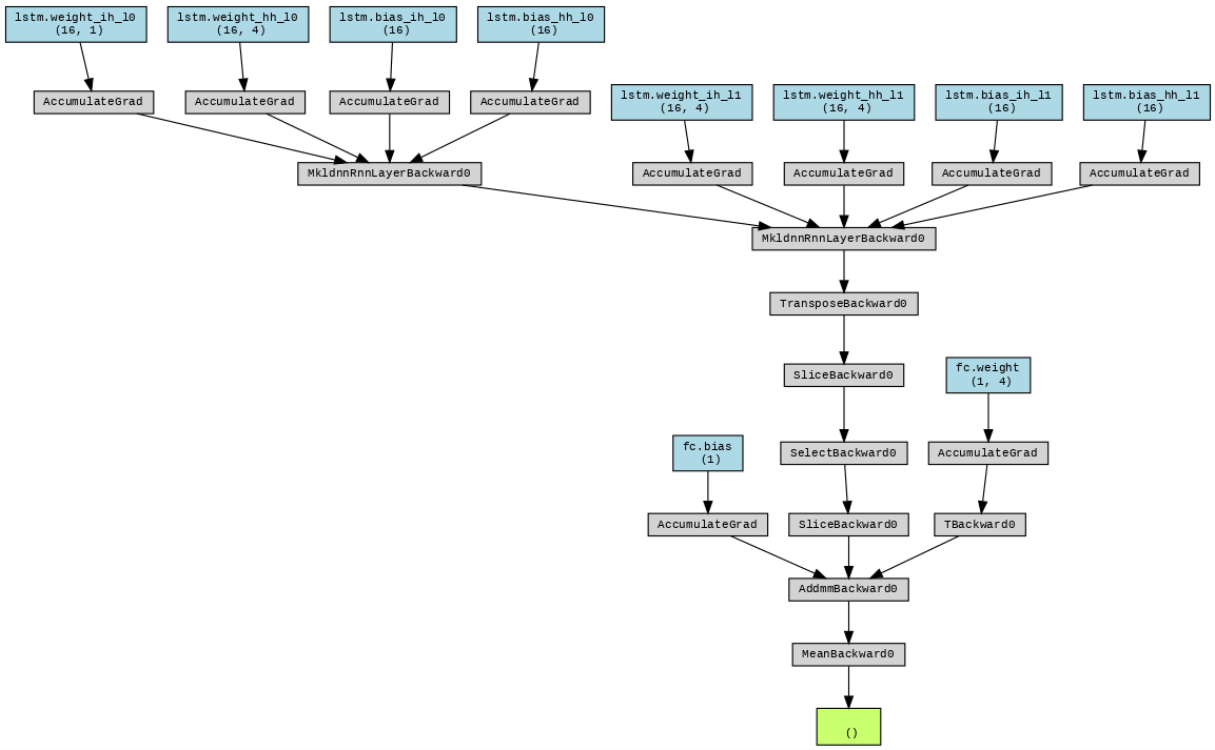


Figure 4: Stacked LSTM

3.3.3 How we got the Best Model

Vanilla LSTMs have one hidden layer and are singular models, unlike stacked LSTMs where two or more LSTMs are sequentially linked.

The LSTM was fed a matrix containing 6 days' worth of historical data and is predicting the closing price for the 7th day. To get the right set of hyperparameters, we employed Bayesian Optimization, a dependent informed search technique. Bayesian Optimization is similar to manual optimization, as it relies on past results to speed up the search. Despite employing Bayesian Optimization our model needed manual hyperparameter tuning.

Other model, 2 layers, 1 Stacks, 0.197 dropout, 0.016 LR, Optimizer Adam, 25 epochs. With Bayesian Optimization

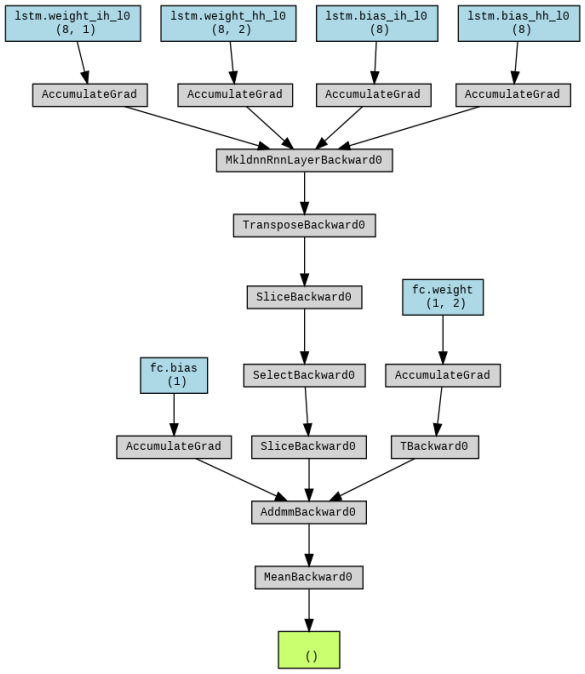
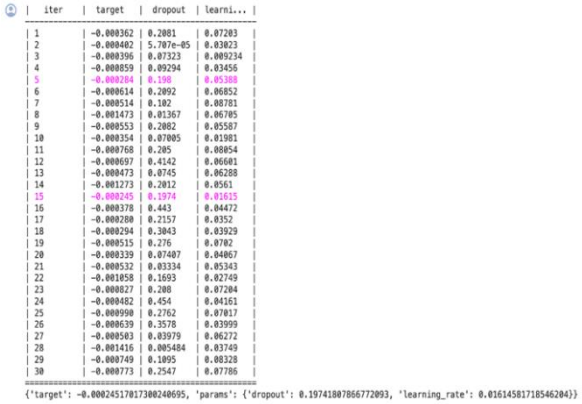


Figure 5: Basic LSTM

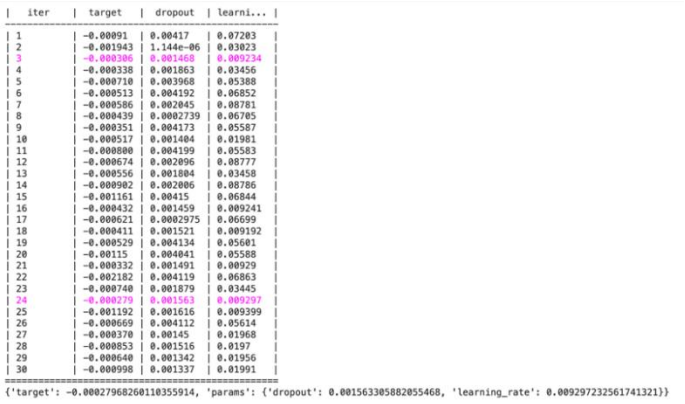
Bayesian Search



iter	target	dropout	learni...
1	-0.000362	0.2081	0.07203
2	-0.000402	5.707e-05	0.03023
3	-0.000396	0.07323	0.009234
4	-0.000859	0.09294	0.03456
5	-0.000284	0.198	0.05388
6	-0.000614	0.2092	0.06852
7	-0.000514	0.102	0.08781
8	-0.001473	0.01367	0.06705
9	-0.000553	0.2082	0.05587
10	-0.000354	0.07005	0.01981
11	-0.000768	0.205	0.08054
12	-0.000697	0.4142	0.06601
13	-0.000473	0.0745	0.06288
14	-0.001273	0.2012	0.0561
15	-0.000245	0.1974	0.01615
16	-0.000378	0.443	0.04472
17	-0.000280	0.2157	0.0352
18	-0.000294	0.3043	0.03929
19	-0.000515	0.276	0.0702
20	-0.000339	0.07407	0.04067
21	-0.000532	0.03334	0.05343
22	-0.001058	0.1693	0.02749
23	-0.000827	0.208	0.07204
24	-0.000482	0.454	0.04161
25	-0.000990	0.2762	0.07017
26	-0.000639	0.3578	0.03999
27	-0.000503	0.03979	0.06272
28	-0.001416	0.005484	0.03749
29	-0.000749	0.1095	0.08328
30	-0.000773	0.2547	0.07786

{'target': -0.0002451701730024069 5, 'params': {'dropout': 0.19741807866772093, 'learning_rate': 0.016145817185462 04}}

Figure 7: Bayesian Search 1



iter	target	dropout	learni...
1	-0.00091	0.00417	0.07203
2	-0.001943	1.144e-06	0.03023
3	-0.000306	0.001468	0.009234
4	-0.000338	0.001863	0.03456
5	-0.000710	0.003968	0.05388
6	-0.000513	0.004192	0.06852
7	-0.000586	0.002045	0.08781
8	-0.000439	0.0002739	0.06705
9	-0.000351	0.004173	0.05587
10	-0.000517	0.001404	0.01981
11	-0.000880	0.004199	0.05583
12	-0.000674	0.002096	0.08777
13	-0.000556	0.001804	0.03458
14	-0.000902	0.002006	0.08786
15	-0.001161	0.00415	0.06844
16	-0.000432	0.001459	0.009241
17	-0.000621	0.0002975	0.06699
18	-0.000411	0.001521	0.009192
19	-0.000529	0.004134	0.05601
20	-0.00115	0.004041	0.05588
21	-0.000332	0.001491	0.00929
22	-0.002182	0.004119	0.06863
23	-0.000740	0.001879	0.03445
24	-0.000279	0.001563	0.009297
25	-0.001192	0.001616	0.009399
26	-0.000669	0.004112	0.05614
27	-0.000370	0.00145	0.01968
28	-0.000853	0.001516	0.0197
29	-0.000640	0.001342	0.01956
30	-0.000998	0.001337	0.01991

{'target': -0.0002796826011035591 4, 'params': {'dropout': 0.0015633058B2055468, 'learning_rate': 0.009297232561741321}}

Figure 6: Bayesian Search 2

3.4 Facebook Prophet

Facebook Prophet is a forecasting tool provided by Meta. Prophet is a decomposable time series model meaning that it decomposes the time series into several parts. Prophet is a model that is easy to use for various forecasting applications at scale. It is reliable as it is used by Meta itself for forecasting. It Helps us to create a reasonable and accurate forecast that can be used by people who are not experts in this field. Prophet also helps to deal with outliers and missing data. It incorporates several Parameters like changepoints, Growth, Seasonality, and holidays.

3.4.1 Prophet Components

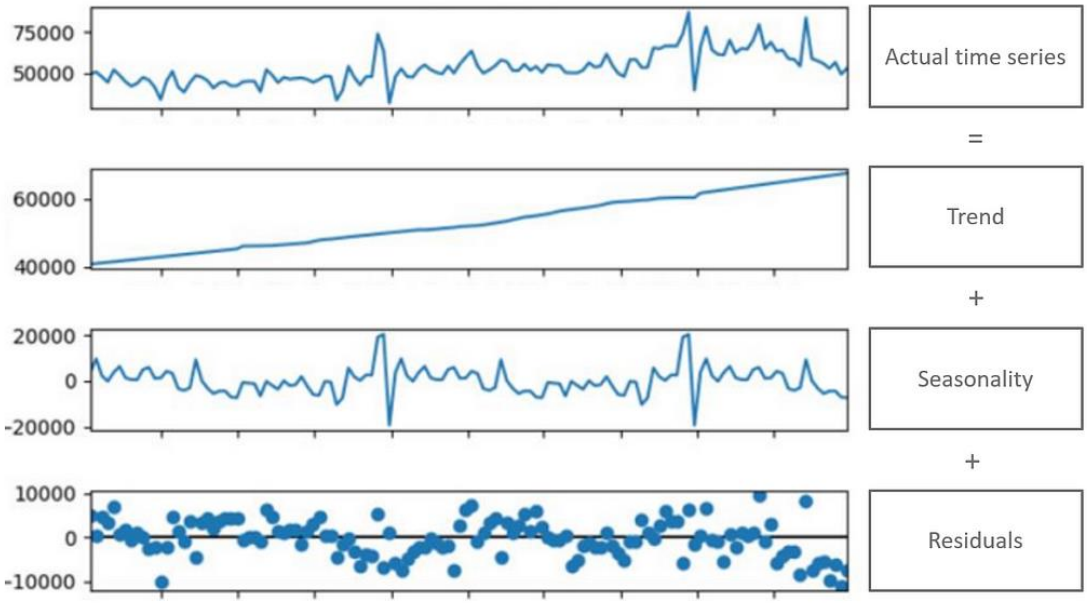


Figure 8: Prophet Components

1. Trend- It tells us about the direction of the time series data.
2. Seasonality – represents the periodic changes that might occur yearly/weekly/monthly.
3. Holidays – tells us about the effects of holidays which occur at uneven intervals
4. Error/Residuals – random fluctuations or irregularities in the data.

3.4.2 How we got to best Model

Changepoints are the points in your data where there are sudden and abrupt changes in the trend. An example of this would be if you had a campaign and suddenly you got 50 000 more constant visitors to your website. The changepoint will be the timeslot where this big change occurred.

Seasonality in Facebook Prophet refers to recurring patterns or fluctuations in a time series data that happen with a fixed frequency over time. These patterns can repeat daily, weekly, monthly, yearly, or at some other regular interval. Seasonality captures the influence of such periodic fluctuations on the time series data.

Following are the hyperparameters which were tuned

changepoint_range: This is the proportion of the history in which the trend is allowed to change.

changepoint_prior_scale: The other parameter, changepoint_prior_scale, is there to indicate how flexible the changepoints are allowed to be. In other words, how much can the changepoints fit to the data.

seasonaliy_prior_scale: This parameter will again allow your seasonalities to be more flexible.

```
    changepoint_prior_scale  seasonality_prior_scale  changepoint_range  \
0                       0.1                     0.01               0.80
1                       0.1                     0.01               0.90
2                       0.1                     0.01               0.95
3                       0.1                     0.01               1.00
4                       0.1                     0.10               0.80
..                      ...                      ...                ...
75                      0.5                     1.00               1.00
76                      0.5                    10.00               0.80
77                      0.5                    10.00               0.90
78                      0.5                    10.00               0.95
79                      0.5                    10.00               1.00

           rmse      MAPE
0    273.961502  0.997679
1    452.806400  1.564928
2    330.778651  1.162224
3    812.338020  3.317047
4    270.000233  1.016501
..          ...       ...
75   658.155235  2.712695
76  1803.470388  7.639365
77  1199.581653  5.124703
78   917.151546  2.968846
79   710.861861  2.950308

[80 rows x 5 columns]
```

Figure 9: Prophet Hyperparameter Tuning

3.4.3 Analysis

Below is the Graph of the Prophet components consisting of the trend, weekly and yearly seasonality. Early seasonality tells us that the highest closing price are observed onwards 2023 and the graph also shows a steadily increasing trend since 2020. Weekly graph tells us that the highest closing price are observed on Wednesday. Monthly forecast graph tells that the highest closing price were observed in March and August

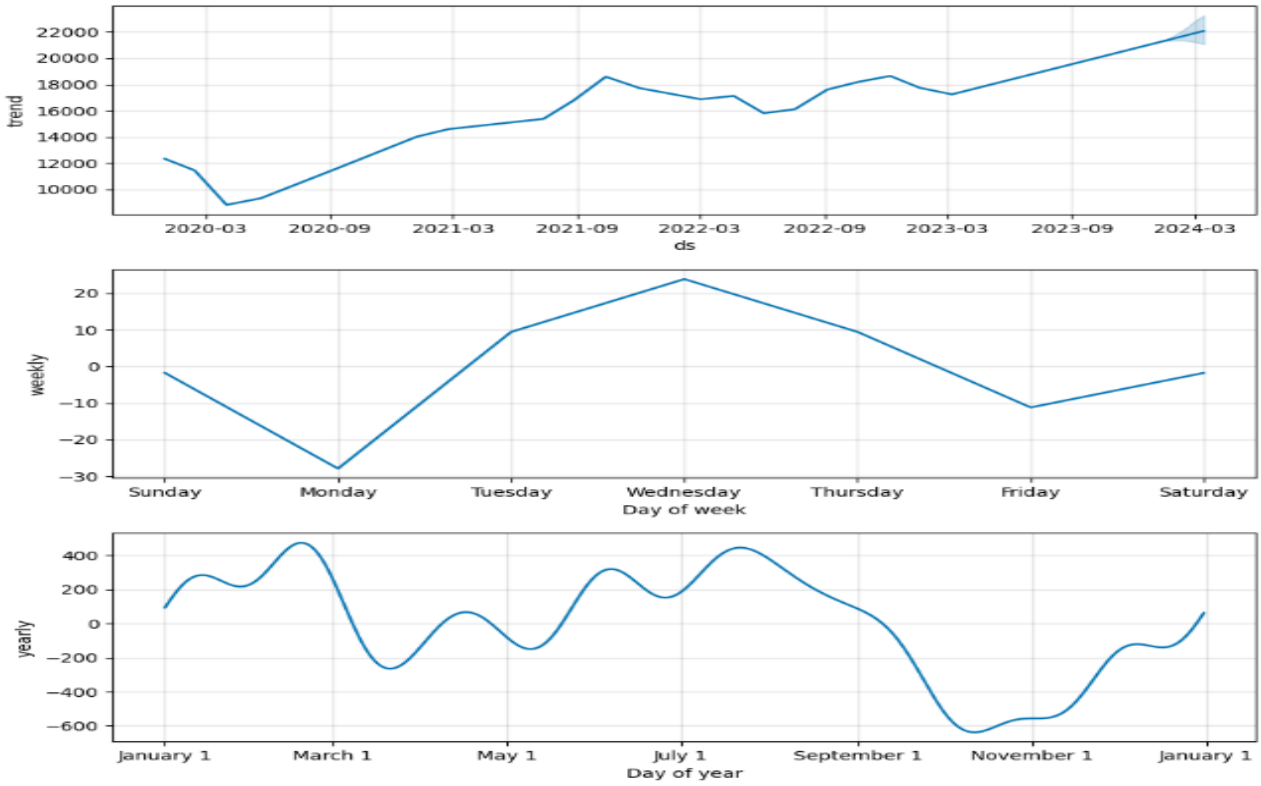


Figure 10: Seasonality Analysis

3.5 Transformers

Transformer is an encoder-decoder model that uses self-attention for the prediction of sequences. The Transformer used for predicting stock prices is a decoder-only Transformer which has three decoders stacked sequentially which means the output of one decoder is the input of another.

3.5.1 Parameters

1. Input dim: Set to 1 as we are performing univariate analysis
2. Feature size: Number of features set to 8.
3. Num_layers: No. of layers of decoder which we have set to 3.
4. Dropout: Dropout probability set to 0.

3.5.2 Model Architecture

The input to the transformer is univariate as it consists of closing prices. The lookback of 7 is kept for our dataset which means our transformer after learning the sequence of the closing price of 6 days predicts the closing price of the 7th day. The first decoder block is consisting of masked self-attention which helps to look at the closing price of all the 6 days and encode each of the prices and is followed by a Feed-Forward Network.

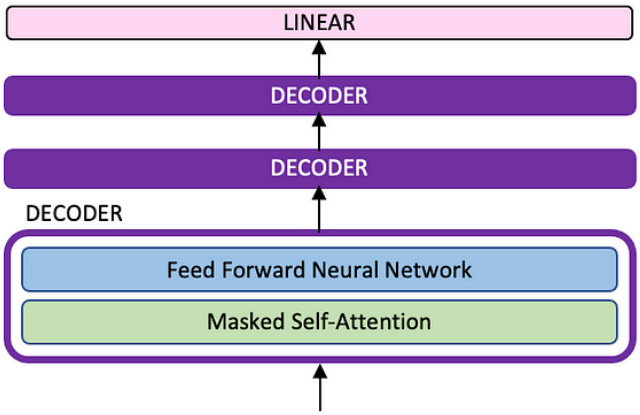


Figure 11: Transformer Architecture

3.5.3 How we Got to Best Model

The transformer model's predictions closely align with those of the test data, Transformer did not require much hyperparameter-tuning/optimization to obtain the best RMSE. Reducing the learning rate and simplifying the model by reducing the features and layers helped to gain the RMSE of 173.32. Early stopping was implemented with the help of learning rate scheduler and helped to save time.

3.6 Gated Recurrent Unit (GRU)

GRU or Gated Recurrent Unit is a type of recurrent neural network which was developed as an alternative to LSTM (Long Short Term Memory) networks. There are 3 types of GRU Models, which are Basic GRU Model, Stacked GRU Model, and Bidirectional GRU Model. In a GRU model, much like the LTSM model, we use gates to control the information flow. These gates are also responsible for updating of the hidden layer after each iteration. It consists of 2 gates-

• **Reset Gate:** It is responsible for determining how much of the information from the previous hidden state should be forgotten.

• **Update Gate:** It is responsible for determining how much of the information from the previous hidden state should be carried forward and used to update the hidden state.

3.6.1 GRU Parameters

GRU Models are mainly used for sequential data (speech, text or time series). The GRU model has many hyper-parameters which can be updated to provide us the most optimal solution. These mainly are

number of GRU units- These specify the number of hidden units in each GRU layer. It helps us in handling the model complexity and its ability to analyze and understand patterns

number of hidden layers- This specifies the number of layers to be stacked. Stacking multiple layers is usual when we need to build a highly complex model.

3.6.2 GRU Architecture

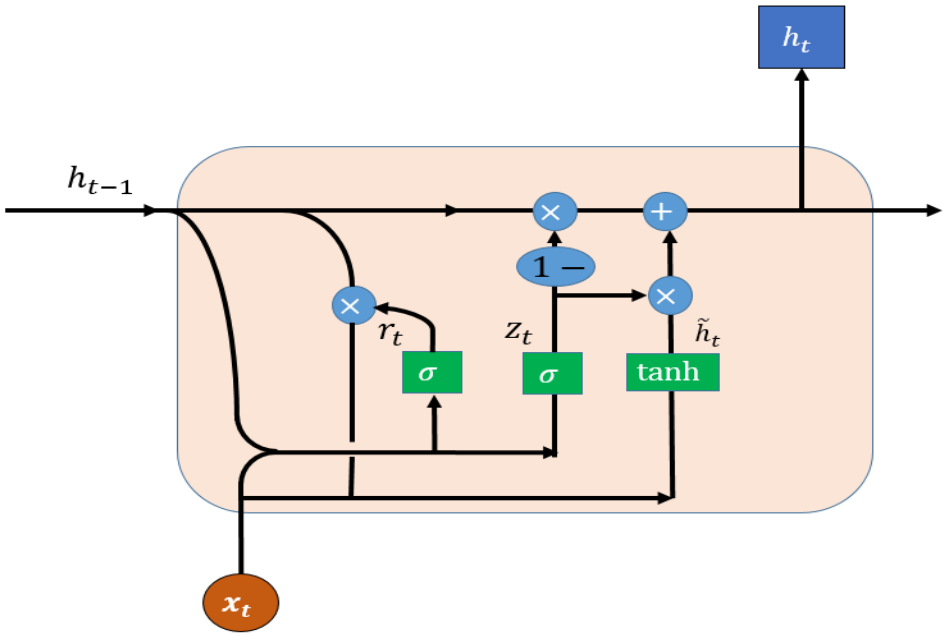


Figure 12: GRU Architecture

Model Architecture

```
1 model
```

```
GRUModel(
  (gru): GRU(1, 256, batch_first=True, dropout=0.2)
  (fc): Linear(in_features=256, out_features=1, bias=True)
  (relu): ReLU()
)
```

Figure 13: Best Performing Model Architecture

3.6.3 How we got the Best Model

Since we are making a univariate prediction, we used the basic GRU model architecture for our model as it is sufficient enough for our prediction task and is computationally less expensive than a stacked gru model or a bidirectional gru model.

We used a learning rate of 0.001 on our model to ensure that our model doesn't overfit or underfit.

We added a dropout of 0.2 to avoid overfitting occurring.

By experimenting with different number of layers and GRU Units, we were able to determine that we got the most optimal solution by keeping the number of hidden layers as 1, while keeping the number of GRU units as 256. This helped us in building an efficient model while not making it very complex as well. Making the model complex led to it overfitting.

We ran the model with 200 epochs using the Adam Optimizer with the Loss Criterion as Mean Squared Error as by experimenting with different optimizers like Adam, SGD, RMS Prop, we saw that Adam provided us with the least loss.

4. Result and Discussion

4.1 Prediction Graph of ARIMA Model

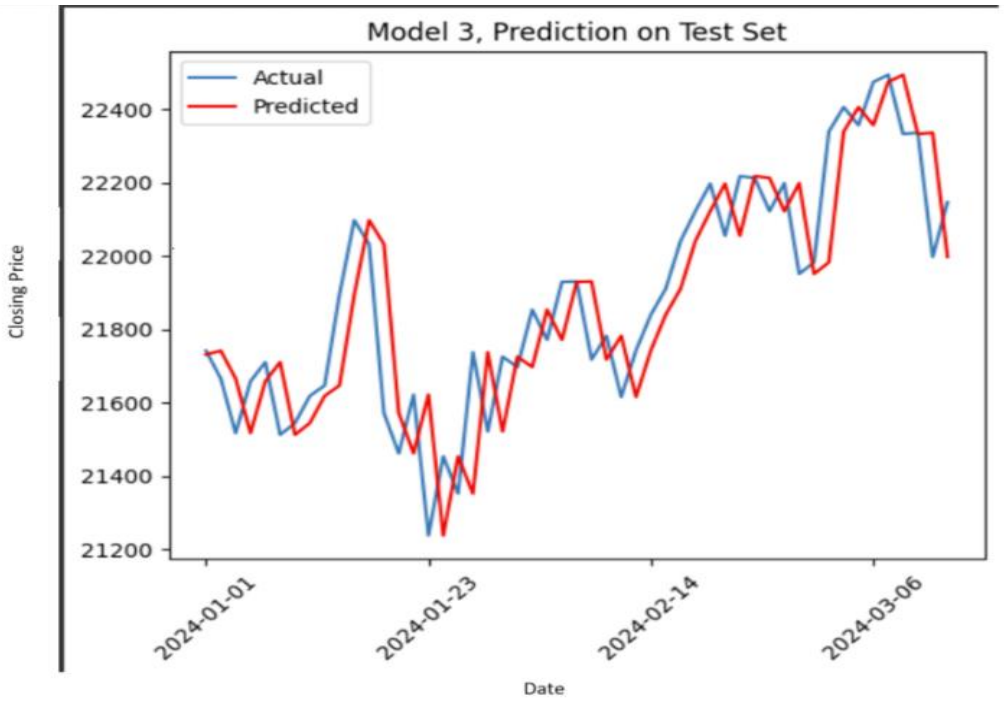


Figure 14: Prediction Graph of ARIMA

4.2 Prediction Graph of LSTM

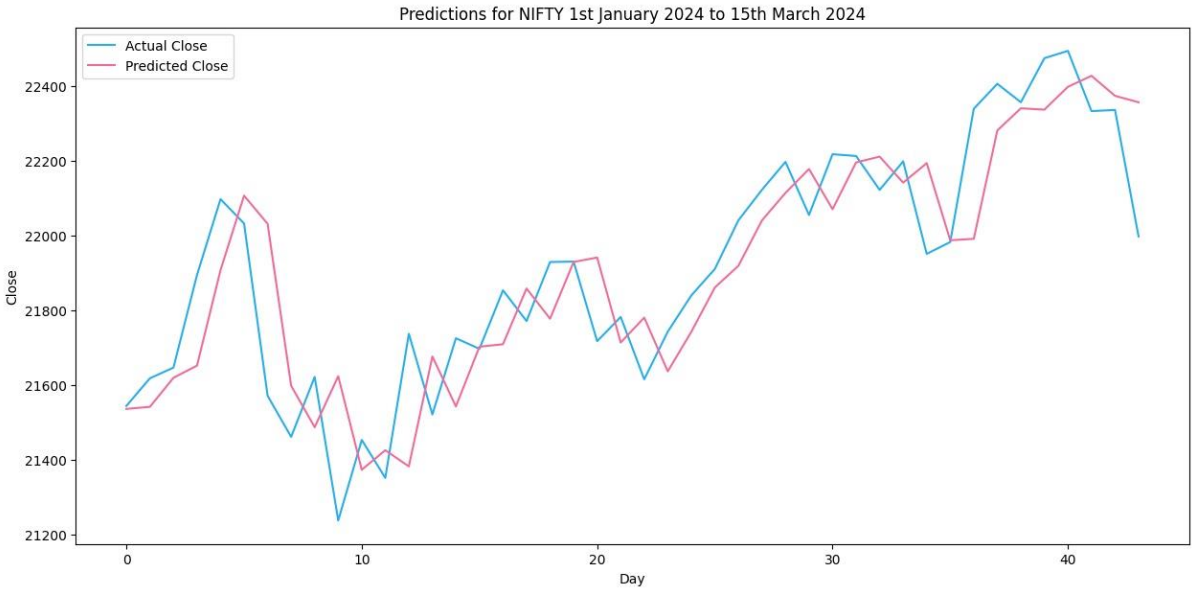


Figure 15: Prediction Graph of LSTM

4.3 Prediction Graph of Facebook Prophet

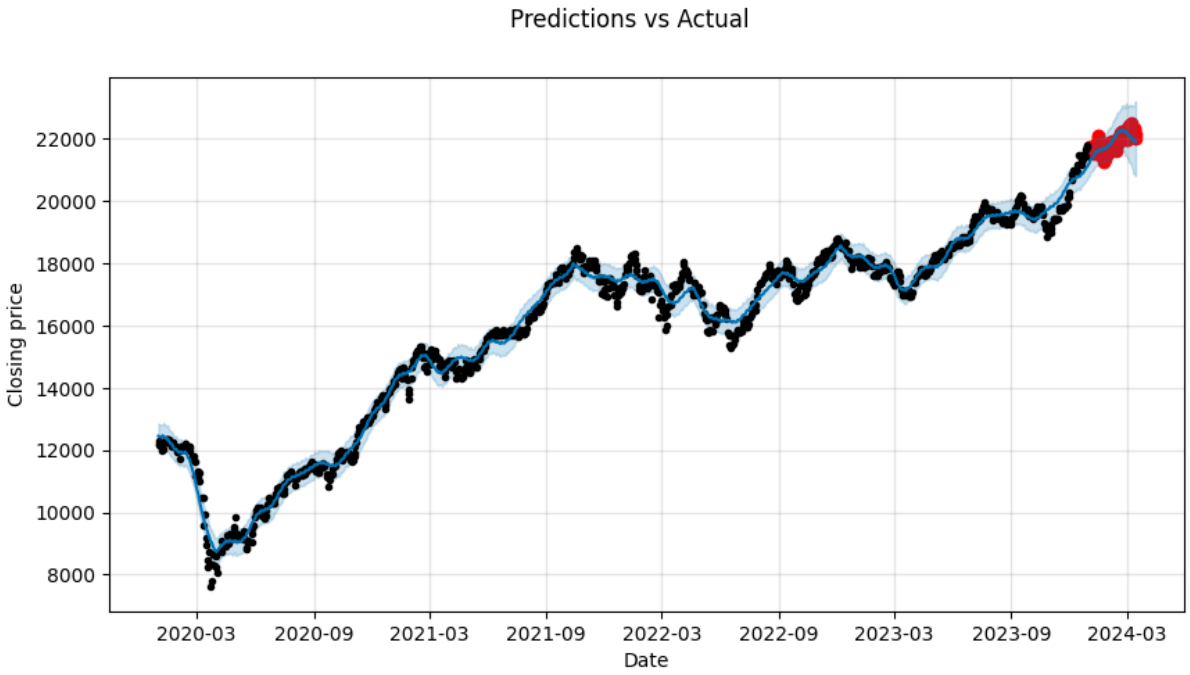


Figure 16: Prediction Graph of Facebook Prophet

The Blue line here shows the Prophet's prediction on trained and test data. Black dots are the actual values of the closing price of train data and red dots are the actual values of the closing prices of the test data. The focus of this graph is to look at the prediction of the test data and see that the model's prediction fits the closing prices accurately.

4.4 Prediction Graph of Transformers

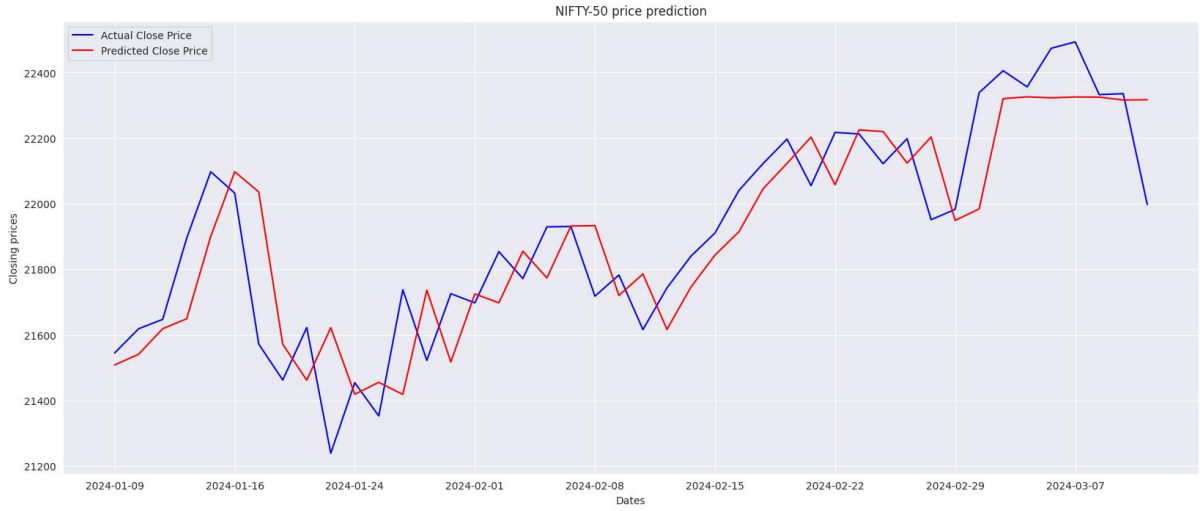


Figure 17: Prediction Graph of Transformers

4.5 Prediction Graph of GRU

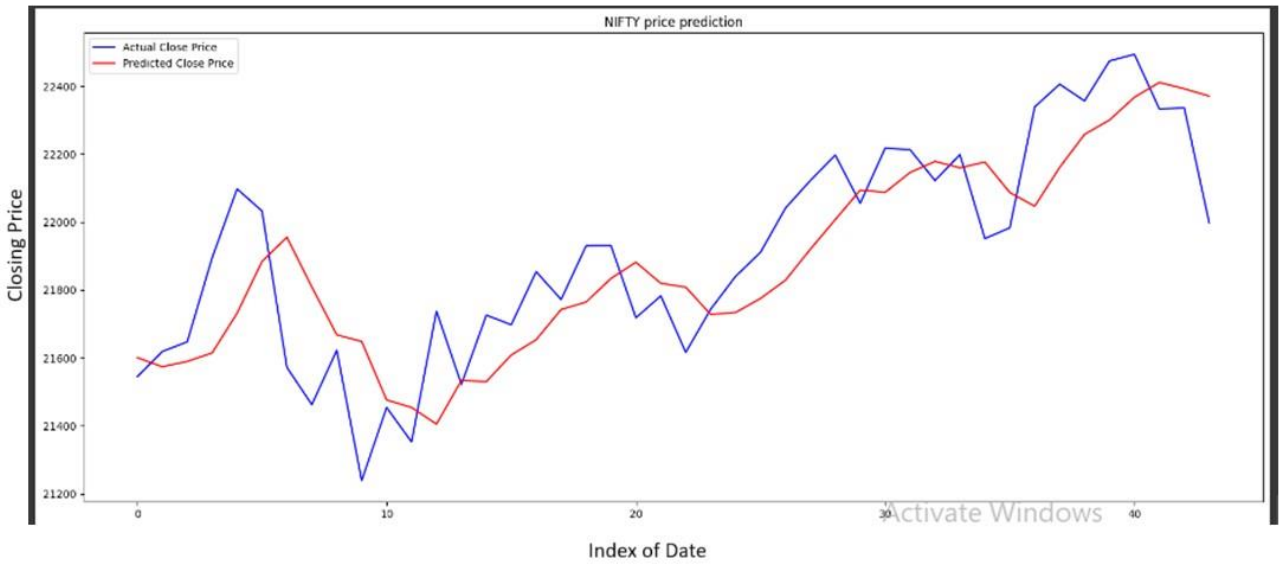


Figure 18: Prediction Graph of GRU

4.6 Interpretation of Results

From the observed graphs, we can say that ARIMA Model and LSTM generalized well and could predict the Close Price on unseen data between January 2024 and March 2024 with the least error, followed by Transformers closely in terms of performance. However, when we observe GRU and Prophet models, we observe that they grossly underfit the unseen data suggesting that the models are too complex and have overfit the training data, producing subpar results.

Model	**Test RMSE**
LSTM	171.4
Arima	172.68
Transformers	173.324
GRU	189.53
FB Prophet	270

5. Conclusion

The Root Mean Squared Error (RMSE) stands as one of the primary metrics for evaluating regression models, quantifying the average disparity between predicted values and actual values. It offers insight into the model's predictive accuracy by estimating its capability to forecast the target value effectively.

After comparatively observing and analyzing the models, Long Short Term Memory was found to be the best model with an RMSE of 171.4, followed by Arima models with an RMSE of 172.68

The results obtained via LSTMs are subject to the random nature of the neural network weight initialization process. Even when all other factors are controlled, training LSTMs involve various underlying processes that may have inherent randomness, such as CPU or GPU operations. Hence reproducibility of results under the same conditions is arduous.

ARIMA models do not depend on such random seed, therefore, same results can be reproduced. Based on the parameters tuned, the model can perform long term predictions as well as short term predictions. Moreover, forecasting in the case of univariate predictions is difficult for neural networks like LSTM, Transformers and GRU. ARIMA model being light weight in processing, and have been around longer than DL, are widely used in the industry for time series forecasting.

6. Future Scope

For now, we have performed univariate analysis and are trying to predict the closing value referring to past closing values only. We can perform multivariate analysis and try to predict the closing value referring to other factors like open, volume, high and low. We can also try to calculate some **technical indicators** during the pre-processing and use it as a feature.

Currently, we are predicting the closing value at day level. We can train the model on 5-minute data and try to predict the closing value at intervals of 5 minutes. Using 5-minute data and the predictions on it, we can try to create an automatic trading algorithm.

We can also try to improve the performance of the current models by using evolutionary optimizers for our DL networks. In the context of optimizing neural networks for time series prediction, evolutionary algorithms like **Particle Swarm Optimization** may offer advantages in exploring complex and multimodal hyperparameter spaces efficiently. PSO excels in the global exploration of complex, multimodal landscapes by maintaining a swarm of candidate solutions. PSO requires less hyperparameter tuning compared to traditional gradient-based optimization techniques. Stock prediction is formulated as a non-convex optimization problem and PSO can be more robust in handling non-smooth, non-convex, or noisy objective functions.

In our forthcoming research, we aim to devise a graph structure that captures the correlations among stocks and utilizes price performance as the training feature.

In the future, we also plan to utilize state-of-the-art models such as **Temporal Fusion Transformers** for predicting stock prices.

7. References

[1] Shi, J., Jain, M. and Narasimhan, G., 2022. Time series forecasting (tsf) using various deep learning models. arXiv preprint arXiv:2204.11115.

[2] C. Deemee, K. Ngampis, T. Noraset, T. Thaipisutikul, M. -T. Sun and K. Kitchat, "Statistical Comparison ARIMA Order Performance In Stock Market," 2023 7th International Conference on Information Technology (InCIT), Chiang Rai, Thailand, 2023, pp. 81-85, doi: 10.1109/InCIT60207.2023.10413152. keywords: {Training;Time series analysis;Predictive models;Data models;Stock markets;Forecasting;Task analysis;ARIMA model;Stock price forecast;Stock Market;Stock price prediction;Historical Stock data}

[3] Taylor, S.J. and Letham, B., 2018. Forecasting at scale. The American Statistician, 72(1), pp.37-45.

[4] A. Garlapati, D. R. Krishna, K. Garlapati, N. m. Srikara Yaswanth, U. Rahul and G. Narayanan, "Stock Price Prediction Using Facebook Prophet and Arima Models," 2021 6th International Conference for Convergence in Technology (I2CT), Maharashtra, India, 2021, pp. 1-7, doi: 10.1109/I2CT51068.2021.9418057.

[5] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. Advances in neural information processing systems. 2017;30.

[6] Wang, Qianzhun & Yuan, Yingqing. (2023). Stock Price Forecast: Comparison of LSTM, HMM, and Transformer. 10.2991/978-94-6463-198-2_15.

[7] Mehtab, S., Sen, J., Dutta, A. (2021). Stock Price Prediction Using Machine Learning and LSTM-Based Deep Learning Models. In: Thampi, S.M., Piramuthu, S., Li, KC., Berretti, S., Wozniak, M., Singh, D. (eds) Machine Learning and Metaheuristics Algorithms, and Applications. SoMMA 2020. Communications in Computer and Information Science, vol 1366. Springer, Singapore.

[8] Y. Liu, "Stock Prediction Using LSTM and GRU," 2022 6th Annual International Conference on Data Science and Business Analytics (ICDSBA), Changsha, China, 2022, pp. 206-211, doi: 10.1109/ICDSBA57203.2022.00054. keywords: {Machine learning algorithms;Fitting;Psychology;Companies;Data science;Complexity theory;Stock markets;RNN;LSTM;GRU;stock price;big data}