Deep analysis of stock market behaviour using Long Short Term Memory, ARIMA and Prophet Models

Shivam Basia
7838-4123
shivam.basia@ufl.edu

Shashank Kumar
7144-3673
sh.kumar@ufl.edu

Richa Gupta
3634-2982
richagupta@ufl.edu

Abstract

In the current economic climate, modeling and profiting from equity-related risks profoundly impacts an investor's gains and is based on the intricate stock index movement. An increase or decrease in the price of these stocks of publicly held companies can determine the future of an entire business and Investor wealth. By deploying machine learning models, it is possible to predict the future value of stock prices and generate significant profits. Most of the existing methods rely only on linear and non-linear algorithms that consider daily closing prices and correlation between stocks. Researchers have discovered dynamic architectures that factor in multi-modal signals in the data by using deep learning architectures. Here, we are implementing various algorithms namely Long Short Term Memory, ARIMA as well as Prophet and comparing each model to see which model performs better for the out-of-sample financial movements and can be applied as a trading strategy. Our empirical results show that the Long Short Term Memory has the highest accuracy in terms of long period stock price prediction for a multi-feature setting as well as for a single-feature setting consisting of daily closing prices.

1 Introduction

1.1 Background

Stock market prices are unpredictable and non-stationary. As a result, we cannot model stock prices precisely over time since we use many consistent patterns in data. There is a notable amount of risk involved with investing in stock markets. An uninformed trader may be exposed to financial risk and incur monetary losses, whereas a careful investment will maximize profits. Prices are driven by a broad range of factors, including but not limited to political events, economic conditions and economic stability.

The state-of-the-art models to predict the stock prices are centered around a fundamental concept called time series forecasting. Time series forecasting refers to statistical predictions made using scientific models dependent on historic time-stamped data. These models only provide a probabilistic measure of the possible outcomes at a specific time in the future. Time series analysis is particularly complex due to a time component that adds an explicit order dependence to the observations. This additional dimension to the features serves both as a constraint and a source of information. This concept is highly used in modeling seasonal patterns and trends for an entity.

Analyzing any index's periodic variance and flow can help investors trade efficiently and incur profits. There will always be turbulence in the financial data, making it difficult to find reliable patterns. Machine learning algorithms are needed to find hidden patterns in data and to know how they will affect turbulent structures in the future. A trend chart can be plotted by developing models that take discrete time series as input, like stock prices. The potential for machine learning to simplify the whole process lies in its ability to analyze vast amounts of data, spot significant patterns, and determine which decision to make based on predicted asset prices.

1.2 Motivation

Presently, the stock prices trend forecasts are done using various models ranging from linear to non-linear. Models referred to as linear are those whose features combine linearly. The learning process calculates one weight per feature based on the training data to develop a predictive or estimation model. One of the most popular linear models used to predict stock prices is ARIMA, in which we apply some predefined equations to a univariate time series. The model introduced for one series does not apply to another, making it impossible to identify generalized patterns in the combined data.

In contrast, non-linear models directly employ the sigmoid, ReLU, or exponential functions. Convex optimization techniques are used when non-Linear models have more than one minima where tailor-series are applied to approximate the differential equation of nth order. Deep neural networks employ architectures like Recursive Neural Networks (RNN), Long Short Term Memory (LSTM), and CNN (Convolutional Neural Network) to approximate non-linear functions. Their applications range from natural language processing to time series analysis.Various studies have examined specific types of models, which evaluate their performance according to predetermined criteria based on various data types. Each of these studies discusses the pros and cons of the model in question.

Our goal with stock price prediction is to determine the future value of several companies' stocks. We plan to predict the stock prices using linear models like ARIMA, Exponential Moving Average, and some non-linear models like LSTM(Long Short Term Memory)with Recurrent Neural Network, and Prophet Forecasting model. All the results will be analyzed to determine which algorithm has the most significant potential for producing better accuracy calculated as MAPE (Mean Absolute Percentage Error).

1.3 Problem Statement and Optimization Problem

Stock Market forecasting is an Optimization problem which is also inspired by an optimization algorithm called stock exchange trading optimization (SETO) for solving numerical and engineering problems.The inspiration source of this optimization problem is the behavior of traders and stock price changes in the stock market. Traders use various fundamental and technical analysis methods to gain maximum profit.So, our problem statement is to optimize the profits of the Stocks by predicting the future term prices of each stocks.

In the previous works, researchers have been using a variety of models for predicting stock price trends. In this work, we tried using three of the most widely used models for time-series forecasting namely Long Short Term Memory(LSTM), ARIMA and Prophet for long term Stock Market Prediction. We are trying to predict the Stock Prices for long term as long term investing comes with its own benefits like no or less taxations on Long Term Capital Gains which in turn is more motivating for investors.

2 Literature Survey

Different models have been developed and analyzed to predict the stock market value.

- In a study titled "Stock Market Prediction Using Linear Regression and SVM," Linear Regression was shown to work superior to SVM for stock market analysis[PDJ$^+$21].
- F. B. Oriani and G. P. Coelho studied the impact of technical indicators on stock forecasting by using Exponential Moving Average and Weighted Moving Average[OC16].
- C. K. Ayo, A. O. Adewumi, and A. A. Ariyo proposed a linear model to make predictions based on the autoregressive integrated moving average (ARIMA) model[AAA14].
- Using Facebook Prophet and Arima models, another study forecasts future retail valuations of stocks, allowing us to make significant comparisons[GKG$^+$21].
- R. Achkar, F. Elias-Sleiman, H. Ezzidine, and N. Haidar presented an approach to predicting stock market ratios using artificial neural networks considering two different techniques-BPA-MLP and LSTM-RNN[AESEN18].

- Sreelekshmy Selvin, Vinayakumar R, Gopalakrishnan E.A, Vijay Krishna Menon, and Soman K.P used three different deep learning architectures: LSTM RNN, and CNN, for the price prediction of NSE listed companies[SVG+17].

Percentage errors were used to measure the models' performance. A new method of predicting stock markets has also been proposed by researchers, which uses a Bayesian network, and a dynamical Bayesian factor graph that incorporates investor sentiment and market factors to improve forecasting performance using a Long Short-Term Memory (LSTM) neural network.

3 Methodology and Algorithms

3.1 Methodology

3.1.1 Environment

Models were developed in python3 using libraries such as pandas, numpy and sklearn. Google Colab was the Integrated Development Environment that was used to develop and run the model. Google Colab uses 16 GB GPU. Additional libraries for Prophet were also installed in Google Colab.The graphs was prepared with the Plotly library.

3.1.2 Dataset and pre-processing

The dataset was compiled from stock market datasets from multiple companies with daily closing prices of the stock. Data was collected from the NASDAQ website for the last 10 years for the FAANG (Facebook, Apple, Amazon, Netflix, Google) stocks. We used the FAANG stocks as they are the most trusted stocks as well as this stocks have presence for more than a decade.

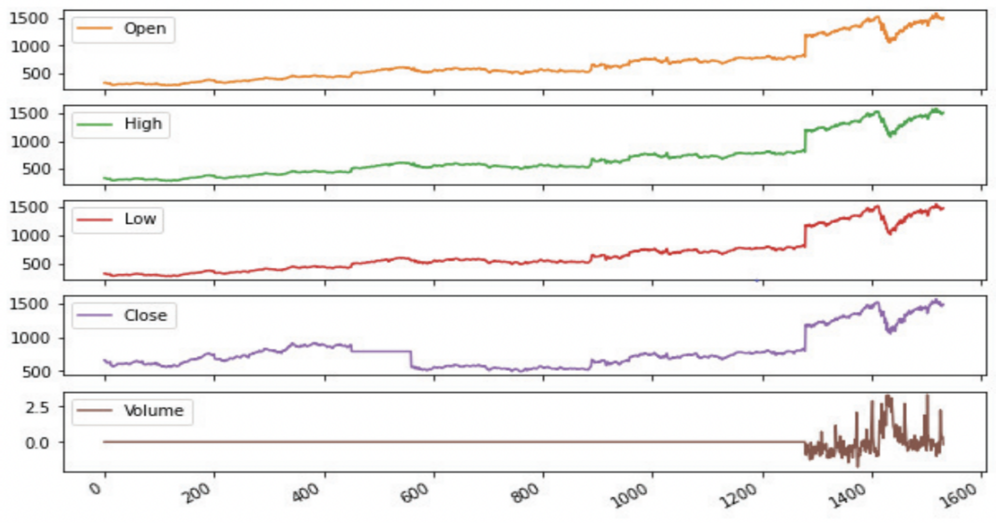


Figure 1: Data Visualization plot for Amazon

The data files came with the following features:

- Open: Opening stock price of the day.
- Close: Closing stock price of the day.
- High: Highest stock price of the data.
- Low: Lowest stock price of the day.
- Volume: Number of stocks traded.
- Date: Date on which data is recorded.

All the five features are plotted against the sixth feature, i.e., the date of the record. The date is plotted on the x-axis and other features on the y-axis. In our initial plots, we observed some missing or NaN values for the features closing price and date. These columns were mean imputed in the data and then plotted again as shown in Figure 1

Similar imputations have been done for all other companies, where missing values are either replaced by mean or the modal values of the feature columns. The samples have been sorted on the date to make them a time series.

3.2 Algorithms and Model Development

Following algorithms have been studied and implemented for our datasets:

3.2.1 Recurrent Neural Network and Long-Short-Term-Memory

RNNs are ANNs that employ previous outputs as inputs, thus remembering the current information cycle through a loop. The system considers both the current input and what it has learnt from the past inputs when making a decision.

An RNN can be described as a sequence of neural connections that we train consecutively with back-propagation. Figure 2 shows a typical unrolled RNN where neural network A has input x_t and output as h_t. Information is transferred between steps using a loop-like structure.

RNN is unrolled after the equal sign. As the different time steps are visualized and information is passed from one-time step to the next, there's no cycle after the equal sign.

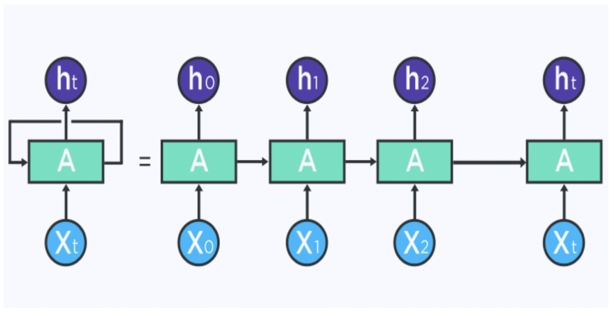


Figure 2: Unrolled version of RNN

Modules (or cells) of an LSTM feature five essential components that let them model both long-term and short-term data.

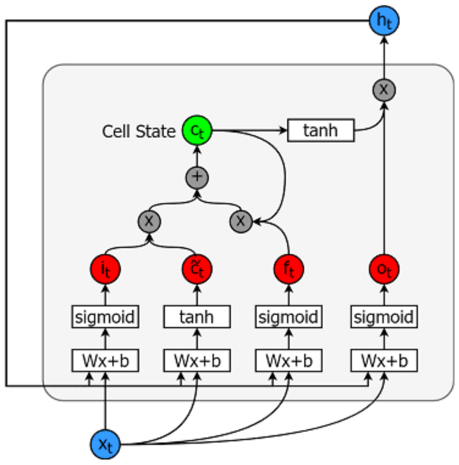


Figure 3: A cell unit of an LSTM

- Cell state (c_t) - Contains both short-term and long-term memories within the cell.
- Hidden state (h_t) - Calculated as an output state information based on current inputs and previously hidden states. Automatically chooses whether to retrieve long-term memory, short-term memory, or both types of memory to make the next prediction.
- Input gate (i_t) - Determines the amount of information flowing to the cell from the current input.
- Forget gate (f_t) - Determines the amount of information to be transferred from the current input and previous cell state to the current cell state.
- Output gate (o_t) - Controls the amount of information from the current cell state flowing into the hidden state.

Each of these entities can be calculated using the following equations.

$$i_t = \sigma(W_{ix}X_t + W_{ih}h_{t-1} + b_i) \tag{1}$$

$$\tilde{c_t} = \sigma(W_{cx}X_t + W_{ch}h_{t-1} + b_c) \tag{2}$$

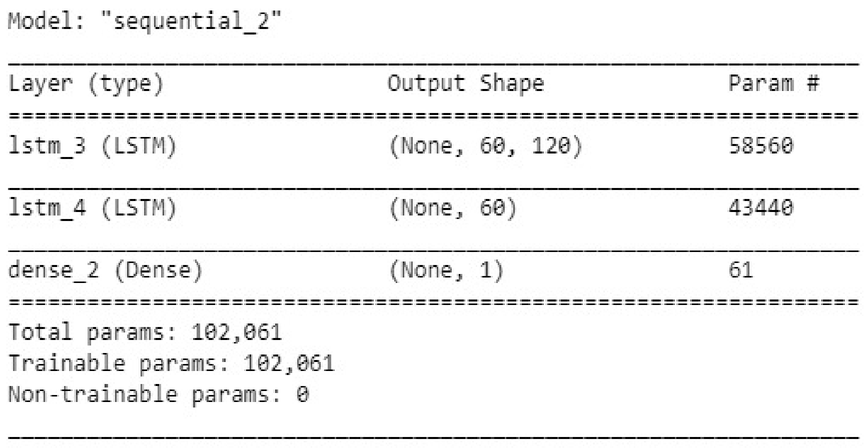
$$f_t = \sigma(W_{fx}X_t + W_{fh}h_{t-1} + b_f) \tag{3}$$

$$o_t = \sigma(W_{ox}X_t + W_{oh}h_{t-1} + b_o) \tag{4}$$

$$c_t = f_t c_{t-1} + i_t \tilde{c_t} \tag{5}$$

$$h_t = o_t \tanh(c_t) \tag{6}$$

LSTM Model Development We developed the stacked LSTM model by using multiple hidden layers of LSTM with a dropout between the layers for regularization, similar to Deep Recurrent Neural Network model. The stacked LSTM model used here has 120 neurons in the first layer with a dropout between layers for regularization and 60 neurons in the second layer, and 50 epochs. The iterative update of weights using the training data is essential for training the model. An overview of model summary is presented in Figure 4:



```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 60, 120)           58560
_________________________________________________________________
lstm_4 (LSTM)                (None, 60)                43440
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 61
=================================================================
Total params: 102,061
Trainable params: 102,061
Non-trainable params: 0
_________________________________________________________________
```

Figure 4: LSTM Model Summary

3.2.2 ARIMA Model

ARIMA (Autoregressive Integrated Moving Average) is a statistical analysis linear model that accepts time-series data as input to analyze or predict future trends based on past values.

Prior to implementing an Arima Model on a Dataset, it is important to make sure that Stationary Test has has been conducted.

Stationary test To apply an ARIMA model, we take **non-stationary data** and then stationarize it. In this project, we have used Augmented Dickey-Fuller Test to check the data.

The purpose of this test is to determine whether a series is stationary by examining how the essence of a unit root manifests itself. P-value must be less than 0.05 for data to be stationary.
Here, P-value obtained for data is P=0.99.

- **Null Hypothesis:** Series possesses a unit root.
- **Alternate Hypothesis:** Series is not stationary.

Analysis of the p-value above through stationary test indicates that the data is non-stationary. Hence, ARIMA model can be used.

Work-flow of an ARIMA model is described in Figure 5.

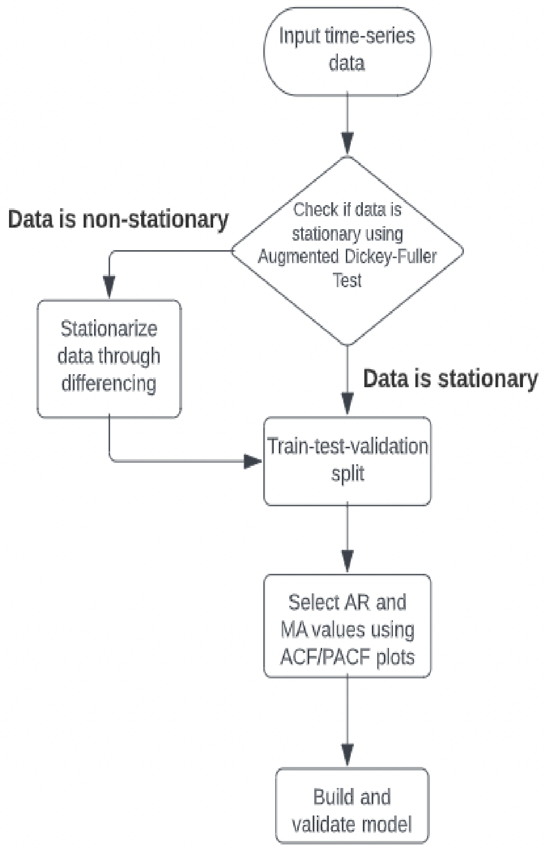

Input time-series data
Check if data is stationary using Augmented Dickey-Fuller Test
Data is non-stationary
Stationarize data through differencing
Data is stationary
Train-test-validation split
Select AR and MA values using ACF/PACF plots
Build and validate model

Figure 5: ARIMA Model Workflow

The components of ARIMA are all parameterised with a standard notation. To indicate the model type, an ARIMA model with integer parameters (p, d, and q) would be a standard notation. Those parameters can be expressed as:

- p (lag order): The number of lag observations in the model. A common representation of an autoregressive model of order p can be written as:

$$y(t) = c + \phi_1 y(t) - 1 + \phi_2 y(y) - 2 + ... + \phi_p y(t) - p + \varepsilon(t) \tag{7}$$

  where $\varepsilon(t)$ represents white noise.
- q (order of the moving average): The size of the moving average window. A common representation of a moving average model of order q can be written follows:

$$y(t) = c + \varepsilon(t) + \theta_1 \varepsilon(t) - 1 + \theta_2 \varepsilon(t) - 2 + ... + \theta_q \varepsilon(t) - q \tag{8}$$

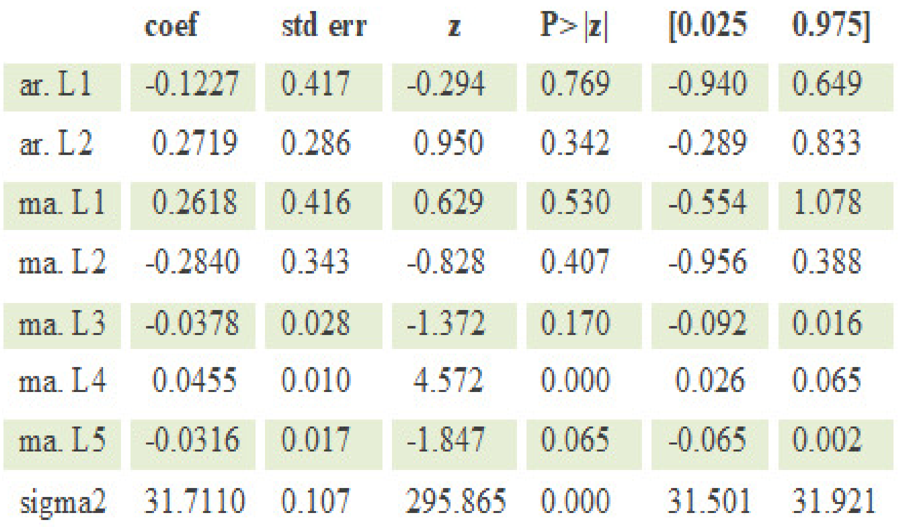
  where $\varepsilon(t)$ represents white noise.
- d (degree of difference): The number of times the raw observations are differenced. The formula for ARIMA after 1st order differencing could be represented as:

$$y'(t) = c + \phi_1 y'(t) - 1 + ... + \phi_p y'(t) - p + \theta_1 \varepsilon(t) - 1 + ... + \theta_q \varepsilon(t) - q + \varepsilon(t) \tag{9}$$

  where y'(t) represents the series of difference.

ARIMA Model Development We developed the ARIMA Model on training dataset and selected the highest quality of ARIMA Model by utilizing the AIC, AME, ASE bases. An overview of model summary is presented in Table 1.

Table 1: ARIMA Model Summary



	coef	std err	z	P>|z|	[0.025	0.975]
ar. L1	-0.1227	0.417	-0.294	0.769	-0.940	0.649
ar. L2	0.2719	0.286	0.950	0.342	-0.289	0.833
ma. L1	0.2618	0.416	0.629	0.530	-0.554	1.078
ma. L2	-0.2840	0.343	-0.828	0.407	-0.956	0.388
ma. L3	-0.0378	0.028	-1.372	0.170	-0.092	0.016
ma. L4	0.0455	0.010	4.572	0.000	0.026	0.065
ma. L5	-0.0316	0.017	-1.847	0.065	-0.065	0.002
sigma2	31.7110	0.107	295.865	0.000	31.501	31.921

The Model summary in Table 1 depicts the best ARIMA model is ARIMA(2,1,5).

Various forecasting techniques may be possible for the same dataset. Monitoring the sources is crucial to determine whether a prediction technique utilizes comprehensively available data. The diagnostic plot of our Arima model is presented below in Figure 6:

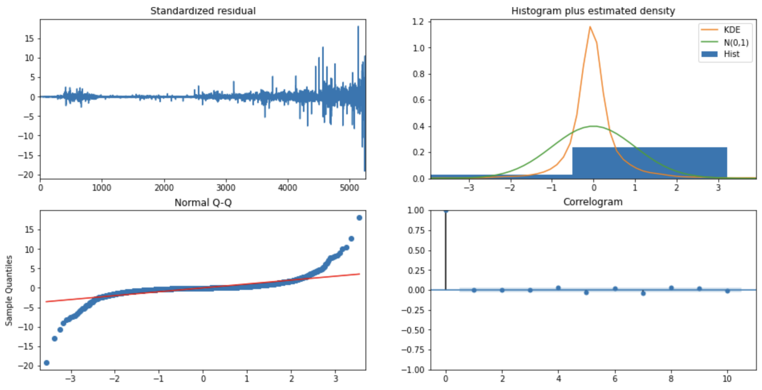


Figure 6: ARIMA Diagnostic Model

Correlograms for residuals are meant to be flat which means that they should be flat and lie between the shaded regions. In our Arima Model Diagnosis, all the points lie within the shaded region.

3.2.3 Prophet Model

To forecast univariate time-series datasets, Facebook's Core Data Science team designed the Prophet library as an open-source project. The program uses a unique hyperparameter setting to automatically find a good set of parameters for the model to provide the best possible forecasts. The analysis looks for changes in trends due to various products and for outlier seasonal effects like weekly, monthly, and yearly cycles. Seasonality can be modelled as an additive element in the same way as exponential smoothing.

Prophet forecasting model equation is given as:

$$y(t) : g(t) + s(t) + h(t) + \epsilon(t) \tag{10}$$

where
y(t) – additive regression model
g(t) – trend factor
s(t) - seasonality component
h(t) - holiday component
ϵ(t) – error

The trend and seasonality parameters used by prophet models are calculated as below:

- Change points in trends that are piece-wise linear or logistic in nature are automatically detected by Prophet by analyzing the data. A parameter called $changepoint\_prior\_scale$ is used to adjust the trend flexibility and tackle over-fitting or under-fitting.
- Fourier series are used to model a yearly seasonal component.

$$s(t) = \sum_{n=1}^{N}(a_n cos(\frac{2\pi nt}{P}) + b_n sin(\frac{2\pi nt}{P})) \tag{11}$$

  where P = period (365.25 for yearly data and 7 for weekly data),
  Parameters $a_1, b_1, ..., a_N, b_N$ are estimated for a given N.
- Dummy variables are used to create a weekly seasonal component.
- Users provide a list of important holidays.

Prophet Model development As described in component modesl, out data contained all 3 components - daily, weekly and yearly. The trend and seasonality parameters can also be overridden by using add_seasonality method. Prophet time series variable should be y-Target and ds-Datetime. We converted the data frame according to these specifications and fit the model to predict the future value of stocks.

4 Data Analysis

In order to predict the stock price of Amazon, Apple, Meta, Google and Netflix, we analyzed weekly and yearly trend of these stocks of past 10 years. From the analysis we observed important features like whether the stock performed well while going from Tuesday to Wednesday or in a particular month. The stock prices were predicted using the 2 trends - Day of the Year (yearly trend) on left and Day of the week (weekly trend) on right as shown in Figure 7, 8, 9, 10 and 11.

From the graphs, we inferred that stocks of the major companies perform well during July-September quarter and fall during January-March quarter due to taxation cycle.

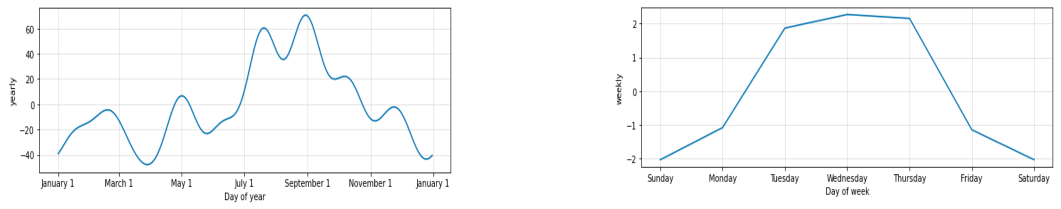


Figure 7: Amazon Stock Day of Year and Day of Week Trend

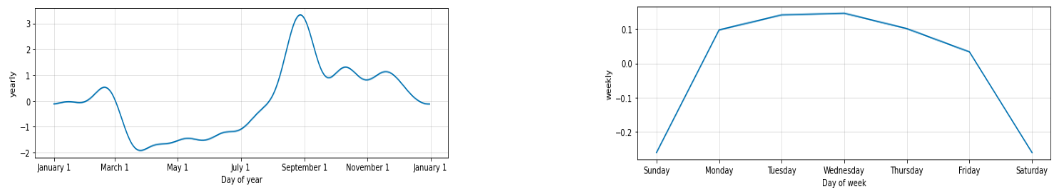


Figure 8: Apple Stock Day of Year and Day of Week Trend

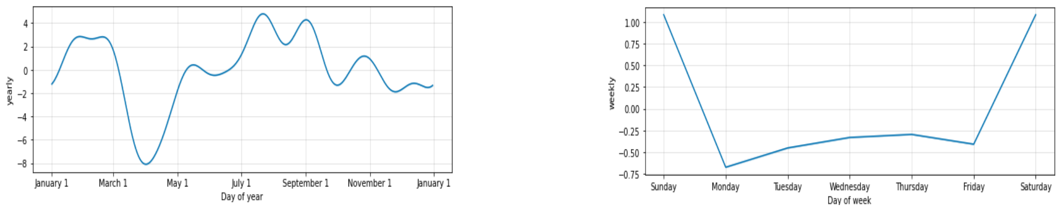


Figure 9: Meta Stock Day of Year and Day of Week Trend

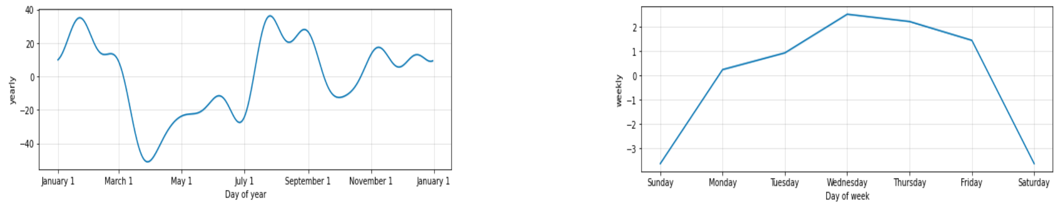


Figure 10: Google Stock Day of Year and Day of Week Trend

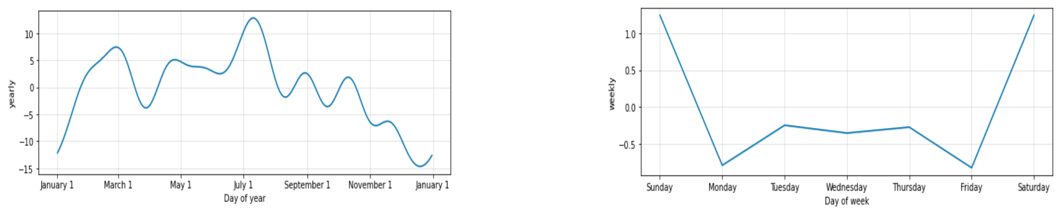


Figure 11: Netflix Stock Day of Year and Day of Week Trend

5 Experiment and Result Analysis

For accuracy and sustainability analysis of the three models, following three metrics were used:

1. **RMSE:** Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are. In other words, it tells how concentrated the data is around the line of best fit.

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2} \tag{12}$$

2. **MAPE:** The mean absolute percentage error (MAPE) is defined as the average of the absolute percentage errors of forecasts. Error is the difference between the observed value and the forecasted value.

$$MAPE = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{e_t}{y_t}\right| \tag{13}$$

3. **MAE:** Mean Absolute Error (MAE) is the mean of difference between the observed and forecasted values. It is the average of the residuals in the dataset.

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|e_t| \tag{14}$$

The train-test split ratio used for the model was 9:1 (9 years data used for training and 1 year data used for testing). The three metrics calculated for the three models for the above data split is presented in Table 2:

Table 2: **Experimental Observations on Stock Price Prediction.**

Models	ARIMA			PROPHET			LSTM		
Companies	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE
Amazon	269.29	186.80	5.63	839.59	714.11	21.53	92.76	73.32	2.21
Apple	22.57	18.04	11.71	23.05	20.41	14.68	4.31	3.44	2.22
Meta	53.18	37.92	10.83	83.22	54.26	15.51	10.61	7.41	2.12
Google	671.72	587.93	22.05	729.45	661.91	24.83	201.96	183.06	6.86
Netflix	100.85	67.42	12.26	147.19	101.25	18.41	20.34	13.31	2.42

The Stock Prices of Amazon, Apple, Meta, Google and Netflix were predicted using all three models (LSTM on the left, ARIMA in the centre and Prophet on the right) as shown in Figure 12, 13, 14, 15 and 16.

It is evident from the plots and accuracy metrics that LSTM outperformed both Prophet and ARIMA models in terms of long term stock market prediction.

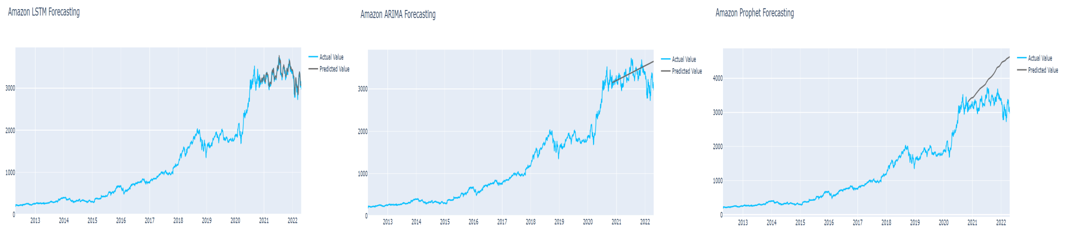


Figure 12: Amazon Stock Price Prediction

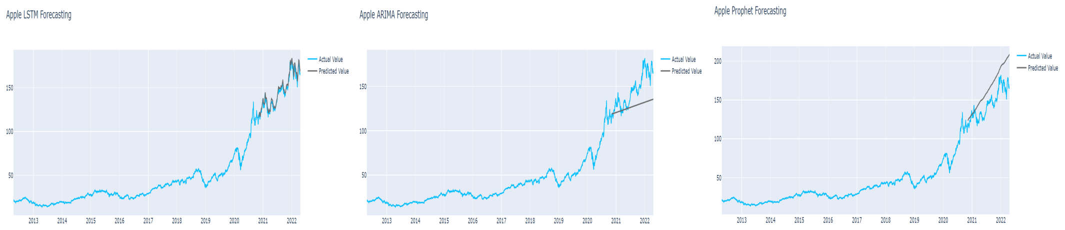


Figure 13: Apple Stock Price Prediction

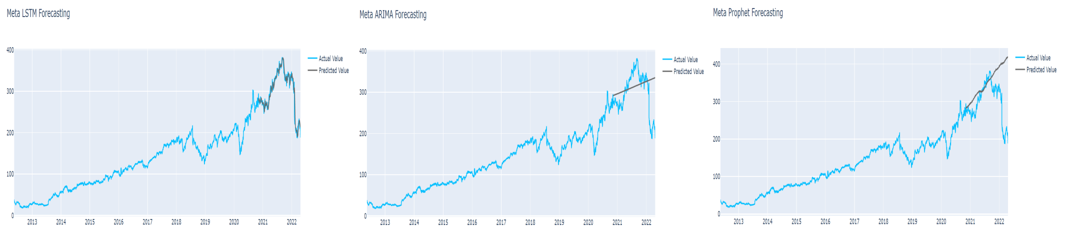


Figure 14: Meta Stock Price Prediction

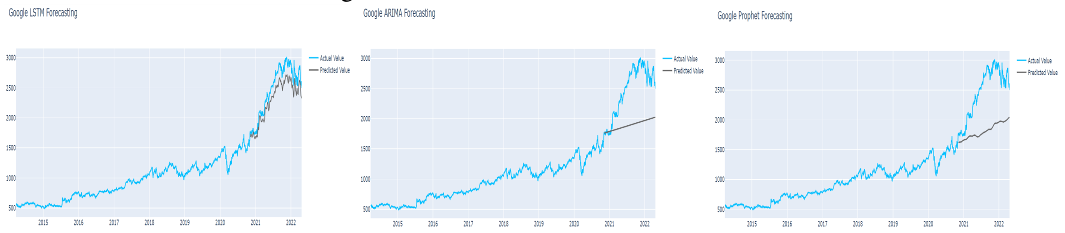


Figure 15: Google Stock Price Prediction

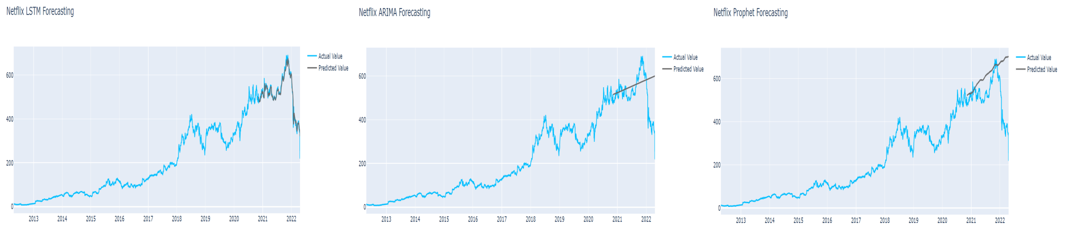


Figure 16: Netflix Stock Price Prediction

6 Conclusion

This paper compared how well ARIMA, LSTM, and Prophet Model can handle time-series data with no seasonality based on stock price data. The prediction was made on ten-year Stock price data from Nasdaq. We discussed the theory behind the back propagation algorithm and recurrent neural networks to construct a stable program that could learn from the past data and predict the future of given stocks. Both the techniques used were relatively accurate. As is evident from the Table 2, the percentage error in LSTM is less compared to ARIMA or Prophet. Given a choice between

ARIMA and Prophet, ARIMA performed slightly better than Prophet. Therefore, an investor may prefer LSTM which has the minimum percentage error amongst all the models studied. Hence, we conclude that LSTM is the most effective and promising tool for stock market prediction.

7 Future Scope

The unavailability or data sparsity may affect the performance of prediction techniques. The demand for predictive tools is increasing drastically, specially in the field of stock market. Hence, new models and technologies are needed with improved accuracy.

Our current model analyzes the seasonal trend of the stock prices from Nasdaq dataset of the past years. However, the stock market prices are deeply impacted by a variety of different external factors like political and social media influence. The future development in the prediction techniques might include a sentimental analysis of social media trends and news feeds, incorporate them as features into our machine learning models and improve the model accuracy for more realistic stock market price predictions.

8 Acknowledgement

We would like to thank Professor Kejun Huang for the Machine Learning course and giving us the opportunity to apply our knowledge by building a project for the course. Each member contributed equally to the implementation, analysis and documentation of this study.

References

[AAA14] A. A. Ariyo, A. O. Adewumi, and C. K. Ayo. Stock price prediction using the arima model. *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, pages 106–112, 2014. doi: `10.1109/UKSim.2014.67`.

[AESEN18] R. Achkar, F. Elias-Sleiman, H. Ezzidine, and N.Haidar. Comparison of bpa-mlp and lstm-rnn for stocks prediction,. *2018 6th International Symposium on Computational and Business Intelligence (ISCBI)*, pages 48–51, 2018. doi: `10.1109/ISCBI.2018.00019`.

[GKG+21] 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)*, pages 1–7, 2021. doi: `10.1109/I2CT51068.2021.9418057`.

[OC16] F. B. Oriani and G. P. Coelho. Evaluating the impact of technical indicators on stock forecasting. *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, 2016. doi: `10.1109/SSCI.2016.7850017`.

[PDJ+21] B. Panwar, G. Dhuriya, P. Johri, S. Singh Yadav, and N. Gaur. Stock market prediction using linear regression and svm. *2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, pages 629–631, 2021. doi: `10.1109/ICACITE51222.2021.9404733`.

[SVG+17] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman. Stock price prediction using lstm, rnn and cnn-sliding window model. *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1643–1647, 2017. doi: `10.1109/ICACCI.2017.8126078`.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Refer section 5 Experimental and Results Analysis
   (b) Did you describe the limitations of your work? [Yes] Refer section 7 Future Scope
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Refer section 3 Methodology
   (b) Did you include complete proofs of all theoretical results? [Yes] Refer section 5 Experimental and Results Analysis
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The code for this project is attached with this report on Canvas.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Refer section 3 Methodology and Algorithms
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Refer section 3 Methodology and Algorithms
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Refer supplemental material and assets attached with the report in Canvas.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] The Data/Environment used for this research does not contain any personally identifiable information or offensive content
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]