Stock Price Predection

Avi Sharma
2022119

Baljyot Singh Modi
2022133

I. Problem Statement

Predicting Stock Prices with Machine Learning

The stock market, a dynamic ecosystem where billions of dollars change hands in the blink of an eye, remains an enigma to many. Investors, analysts, and traders continuously seek to unravel its mysteries, striving to predict the unpredictable: stock prices. In this quest for foresight, machine learning emerges as a beacon of hope, offering sophisticated tools and techniques to decode the market's complexities.

At the heart of this endeavor lies a fundamental problem: the inherent unpredictability of stock prices. Traditional financial models grapple with this challenge, often faltering in the face of sudden market shifts, economic trends, and unforeseen events. However, armed with vast troves of historical data and cutting-edge algorithms, machine learning endeavors to navigate this turbulent landscape with finesse.

The problem our project aims to solve is multifaceted. Firstly, we confront the unpredictability of stock prices head-on, seeking to harness the power of machine learning to discern patterns, trends, and anomalies hidden within vast datasets. Secondly, we endeavor to provide investors and stakeholders with actionable insights, empowering them to make informed decisions in an ever-changing market environment. Finally, we strive to push the boundaries of predictive analytics, leveraging a diverse range of machine learning techniques to refine and enhance our forecasting capabilities.

Central to our approach is the utilization of various machine learning models, each offering unique insights and advantages. Linear regression, a stalwart of statistical modeling, provides a foundational framework for understanding the relationship between independent variables and stock prices. Meanwhile, sophisticated algorithms such as XGBoost and LSTM delve deeper, capturing nonlinear dependencies and temporal dynamics within the data.

Moreover, our project incorporates cutting-edge methodologies such as FBProphet, a powerful tool for time series forecasting developed by Facebook. By combining these diverse techniques, we aim to construct a comprehensive predictive framework capable of adapting to the nuances of different stocks, sectors, and market conditions.

In conclusion, our journey to predict stock prices with machine learning is a testament to human ingenuity and technological innovation. As we navigate the complexities of the market, armed with algorithms and data, we embark on a quest for insight, understanding, and foresight. While the future remains uncertain, our pursuit of knowledge remains unwavering, driven by the belief that with the right tools and techniques, we can illuminate even the darkest corners of the financial world.

Identify applicable funding agency here. If none, delete this.

II. DataSets

In our project we have made use of the YFinance / Yahoo Finance API . The Yahoo Finance API provides access to a vast number of datasets of historical stock prices and financial information for publicly traded companies. This api contains information on a wide range of stocks, including stocks from major indices such as the SP 500 and NASDAQ, as well as smaller, less well-known companies. The data includes a variety of financial metrics, including opening and closing prices, daily high and low prices, trading volume, and more, making it a valuable resource for investors and analysts for analyzing and forecasting stock prices. While training for all our models we have these four datasets -

1) ***AAPL****:* Apple Inc. Common Stock (AAPL)

2) ***NIFTY50****:* The NIFTY 50 is a benchmark Indian stock market index that represents the weighted average of 50 of the largest Indian companies listed on the National Stock Exchange.

3) ***SPY****:* The SPDR SP 500 ETF trust is an exchange-traded fund which trades on the NYSE Arca under the symbol SPY.

4) ***INFY****:* Infosys Limited Stock

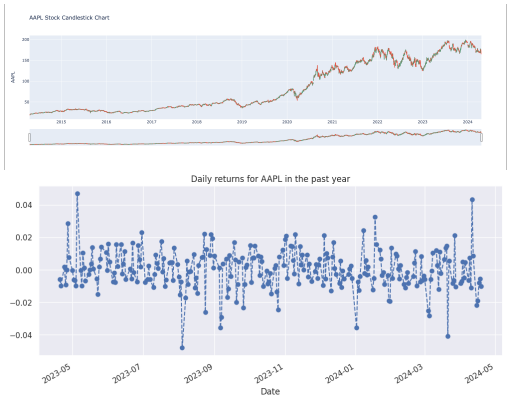
All these 4 stocks have very different trends in themselves. This allows us to test our models against them to make sure our models are not overfitting or memorizing trends and are robust to any kind of data.

III. Experimental Analysis

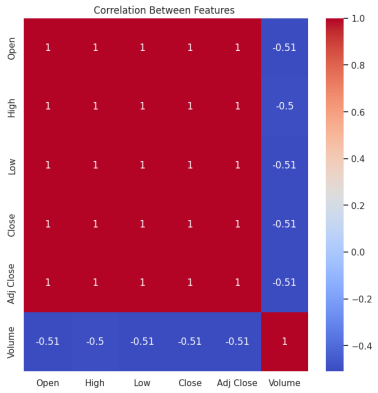
We followed a step wise approach towards applying any of the techniques described above to our data to ensure the robustness of the results.We did sufficient EDA and Feature Engineering to ensure our models gave the best possible predictions.

A. Exploratory Data Analysis

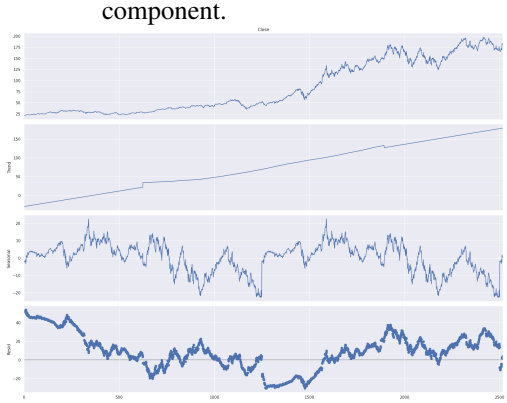
Before starting with building a model that solves the objective , we started of with exploring the dataset to find out which features can be of use to us. We start of by analyzing the trend of the stock over the whole period as well as its daily returns in the past year.



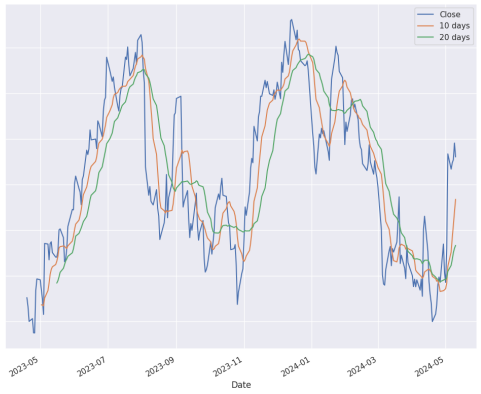
Another key component to be analyzed in our data is the correlation between features. We have plotted the correlation between the principal features in our dataset which were the opening price , closing price , adj close , volume , high and low. Upon plotting a heatmap indicative of the degree of correlation between the features , we observed that most of the were highly correlated.



For applying models like FBProphet , it was also vital for us to know the underlying structure of our time series , hence we decomposed it into its individual trend and seasonality component.



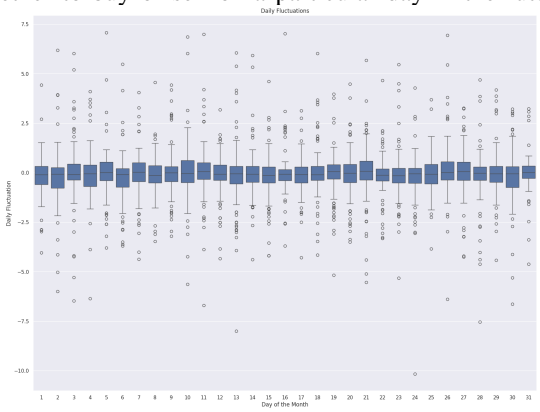
Rolling and Moving averages through different time intervals also provide a good base line idea about the movement of the stock prices and can help boost the model predictions, hence we visualized the moving averages for the past 10 and 20 days respectively.



B. Feature Engineering

Feature Engineering is the process of creating new features or transforming existing features to improve the performance of a machine-learning model. It involves selecting relevant information from raw data and transforming it into a format that can be easily understood by a model. The goal is to improve model accuracy by providing more meaningful and relevant information.A few relevant features we created for our dataset were -

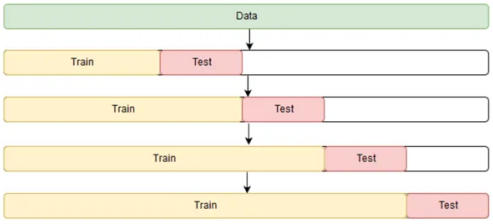
1) ***Daily Max Fluctuation****:* This gave the max fluctuation in the stock price within any particular trading day. In esssence gave the difference between the daily High and Low. Its other counterpart Daily Fluctutation too gives the fluctuation between the price from when the day opens to when it closes. These two features we particularly helpful to use when we were creating a SVC classifier strategy to classify whether to buy or sell on a particular day in the future.



2) ***Simple Moving Averages****:* A simple moving average (SMA) is an arithmetic moving average calculated by adding recent prices and then dividing that figure by the number of time periods in the calculation average. For example, one could add the closing price of a security for a number of time periods and then divide this total by that same number of periods. We have used this parameter by computing the SMA for the past 5 days and used this as feature in the XGBoost Regressor to enhance the predictions. We have attached a photo too in the relevant section showcasing that this feature also carried significant importance in the training stage.

C. Time Series Cross Validation

Since we are dealing with time series data we have used the method of Time Series Cross Validation on a rolling basis. The algorithm proceeds just like normal cross validation except for the change that the data chunk sliced for testing always comes from a later time from which the data for training the model comes. The forecasted labels are then included as part of the next training dataset and subsequent data points are forecasted. The below image is indicative of how the algorithm proceeds. Before implementing all our proposed models we have splitted the dataset into train , test and validation set and evaluated its performances at different percentages of test-train data splits.



D. Sequence Generation for LSTM Model

- **Data Scaling**:
  - We scale the data using MinMaxScaler from scikit-learn's preprocessing module.
  - This scaling operation ensures that all data values fall within the range of 0 to 1.
  - Scaling helps stabilize and accelerate the training process of the LSTM model.
- **Sliding Window Approach**:
  - For **Univariate LSTM**:
    - We utilize a sliding window approach with a parameter called "step" to create sequences of input features (x_{train}) and corresponding target values (y_{train}).
    - Each window represents a sequence of past data points (input features), and the next data point after the window serves as the target value (output).
  - For **Multivariate LSTM**:
    - Similar to univariate LSTM, we use a sliding window approach with a parameter called "step".
    - However, in multivariate LSTM, we consider the previous step values of all the features as input features (x_{train}).
    - The next data point after the window still serves as the target value (y_{train}).

E. Stationarity, ADF Test, and ARIMA Modeling

- **Stationarity**: Time series stationarity implies that its statistical properties remain constant over time. A stationary time series exhibits constant mean, variance, and autocorrelation. Stationarity is essential for many time series analysis techniques, such as ARIMA modeling. Differencing is often used to make non-stationary series stationary by removing trends or seasonality.
- **Augmented Dickey-Fuller (ADF) Test**: A statistical test used to determine the stationarity of a time series. The null hypothesis of the ADF test is that the time series has a unit root, indicating non-stationarity. A low p-value (< 0.05) from the ADF test suggests rejecting the null hypothesis, indicating stationarity. Conversely, a high p-value (> 0.05) indicates insufficient evidence to reject the null hypothesis, implying non-stationarity.

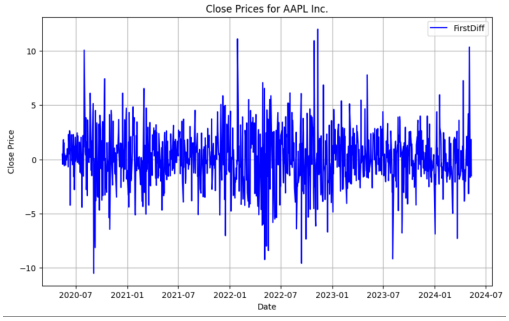


Fig. 1. After First Difference

Autocorrelation Function (ACF): The ACF measures the correlation between a time series and its lagged values. It plots the correlation coefficients for each lagged observation against the number of lags. A strong positive correlation at a specific lag indicates that values at that lag are highly correlated with the current values. A strong negative correlation suggests an inverse relationship between the current values and values at the lag.

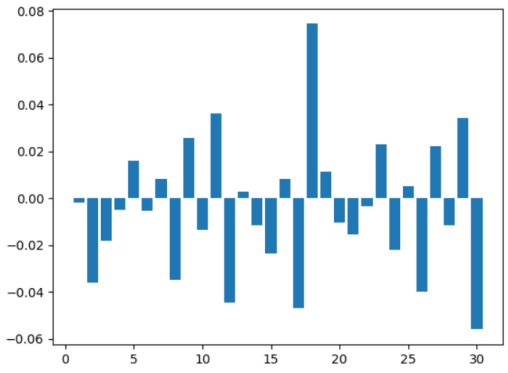


Fig. 2. ACF Plot of First Difference of Apple Plot

Partial Autocorrelation Function (PACF): The PACF measures the correlation between a time series and its lagged values while controlling for the intervening lags. It represents the correlation between the current observation and its lag, accounting for the influence of all shorter lags. A strong partial autocorrelation at a specific lag indicates a direct relationship between the current observation and that lag. It helps identify the order of an autoregressive

(AR) process by showing significant partial autocorrelations beyond a certain lag.

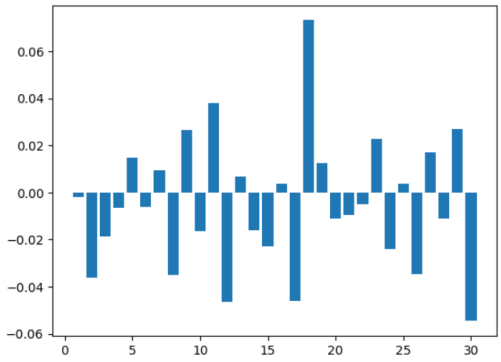


Fig. 3. PACF Plot of First Difference of Apple Plot

IV. Methodology

A. Support Vector Classifiers(SVC)

Support Vector Classifier (SVC) when used for classification, is a powerful supervised learning algorithm used for classification tasks.The basic idea behind SVC is to find the hyperplane that best separates the different classes in the feature space. This hyperplane is chosen in such a way that it maximizes the margin, which is the distance between the hyperplane and the nearest data points from each class, called support vectors. We have used SVC as a part of a strategy which forecasts whether one should buy or sell in the future. **Strategy**: We first preprocess the data and construct two new features that represent the Daily fluctuation in the price of the stock as well as Daily Fluctuation in the Max-Min difference in the stock price. These two features indirectly help us in analyzing whether the stock price went up or down just like a candle stick. We then one hot encode these features to assign categories 1/0 indicating buy and sell respectively. Hence is the Daily flutuation comes out to be positive then we one hot encode it as class 1 indicating that we should buy. Subsequently we train our SVC classifier on the same and then test our

B. Linear Regression

Linear regression is a fundamental statistical technique used to model the relationship between a dependent variable (often denoted as y) and one or more independent variables (often denoted as $x_1, x_2, ..., x_n$). The relationship is modeled as a linear equation:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n + \varepsilon$$

where: - β_0 is the intercept (the value of y when all independent variables are zero), - $\beta_1, \beta_2, ..., \beta_n$ are the coefficients of the independent variables (representing the change in y for a one-unit change in each respective independent variable), - $x_1, x_2, ..., x_n$ are the independent variables,

The goal of linear regression is to estimate the coefficients $(\beta_0, \beta_1, ..., \beta_n)$ that minimize the sum of squared differences between the observed and predicted values of y. This is typically done using the method of least squares. In our model we have selected the features Open , Close , Low and High in our dataset for the linear regression. This formulates this linear equation -

$$y = \beta_0 + \beta_1(Open) + \beta_2(Close) + \beta_3(High) + \beta_4(Low)$$

Hence using these features we have trained our model to predict the Closing Prices of the stock . Here also we have used time series cross validation to evaluate results across various split percentages of the test and train data to get the best result.

C. XGBoost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples. It is a combination of weak

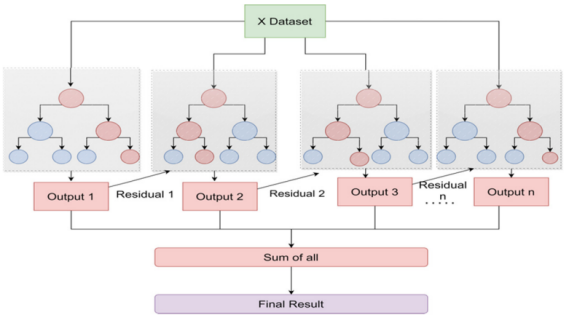


Figure 5. XG-Boost algorithm.

learners such as decision trees. It is a good prediction model for stock forecasting as it works on a sequential model that considers the gradient for each iteration so that the weights are updated for each iteration of the decision tree (Zhu and He 2022).

D. FBProphet

Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well.

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

We use a decomposable time series model with three main model components: trend, seasonality, and holidays. They are combined in the following equation:
g(t): piecewise linear or logistic growth curve for modelling non-periodic changes in time series s(t): periodic changes (e.g. weekly/yearly seasonality) h(t): effects of holidays (user provided) with irregular schedules t: error term accounts for any unusual changes not accommodated by the model

E. Univariate LSTM

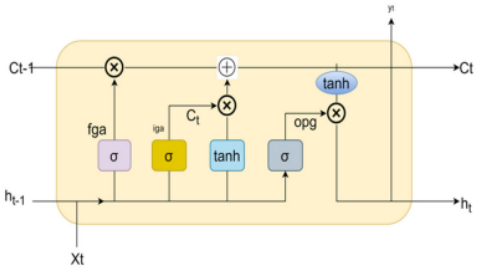


Fig. 4. LSTM Structure

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture designed to overcome the limitations of traditional RNNs in capturing long-range dependencies in sequential data. LSTMs utilize a memory cell with self-connected recurrent units and specialized gating mechanisms, including input, output, and forget gates, to selectively retain and update information over multiple time steps. This enables LSTMs to effectively learn and remember patterns in sequential data, making them particularly well-suited for tasks such as time-series forecasting and natural language processing
1) Model Architecture: The proposed LSTM model architecture consists of multiple Long Short-Term Memory (LSTM) layers stacked on top of each other. Each LSTM layer contains a specified number of memory units or neurons. In this architecture, we employ a stacked LSTM model with three LSTM layers.

- **Input Layer:** The input layer accepts sequences of input features with a predefined window size (*step*). Each input sequence represents past data points.
- **LSTM Layers:** The LSTM layers are responsible for learning and capturing temporal dependencies in the input sequences. Each LSTM layer consists of a certain number of memory units or neurons. The first two LSTM layers have 64 neurons each, and the last LSTM layer has 50 neurons.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 100, 64)           16896

 lstm_1 (LSTM)               (None, 100, 64)           33024

 lstm_2 (LSTM)               (None, 50)                23000

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 72971 (285.04 KB)
Trainable params: 72971 (285.04 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

- **Dense Output Layer:** The output layer is a dense layer with a single neuron, responsible for predicting the next value in the sequence.

F. Multivariate LSTM

: In Multivariate LSTM, multiple time series variables are used as input features to predict future values. Each input sequence consists of observations from multiple variables at different time steps. Multivariate LSTM models capture complex relationships between variables, allowing for more accurate predictions. They are useful when variables exhibit interdependencies and can provide better insights into the underlying dynamics of the system.

```
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 100, 128)          69120

 lstm_1 (LSTM)               (None, 100, 128)          131584

 lstm_2 (LSTM)               (None, 50)                35800

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 236555 (924.04 KB)
Trainable params: 236555 (924.04 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Fig. 5. Multivariate LSTM Structure

G. ARIMA (AutoRegressive Integrated Moving Average)

AutoRegressive (AR) Model:
The autoregressive (AR) component of ARIMA represents the dependency between an observation and lagged observations.
The AR(p) model is defined as:

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + \varepsilon_t$$

Where:

- Y_t is the value at time t.
- c is a constant.
- $\phi_1, \phi_2, \ldots, \phi_p$ are model parameters.
- $Y_{t-1}, Y_{t-2}, \ldots, Y_{t-p}$ are lagged values.
- ε_t is the error term.

Moving Average (MA) Model:

The moving average (MA) component of ARIMA represents the dependency between an observation and a residual error from a moving average model applied to lagged observations.
The MA(q) model is defined as:

$$Y_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q}$$

Where:

- Y_t is the value of the time series at time t.
- μ is the mean of the series.
- ε_t is the residual error at time t.
- $\theta_1, \theta_2, \ldots, \theta_q$ are the parameters of the model.
- $\varepsilon_{t-1}, \varepsilon_{t-2}, \ldots, \varepsilon_{t-q}$ are the residual errors from previous time steps.

Differencing (I): The differencing component is used to make the time series stationary by removing trends or seasonality. It involves taking differences between consecutive observations.
The differenced series Y_t is defined as:

$$Y_t = X_t - X_{t-1}$$

If the series is still non-stationary after differencing, further differencing may be applied until stationarity is achieved.
ARIMA Model:
The ARIMA model is represented as ARIMA(p, d, q), where:
p is the order of the autoregressive component. d is the degree of differencing. q is the order of the moving average component.
The ARIMA(p, d, q) model equation is:

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p}$$

$$+ \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q} (1)$$

ARIMA models are widely used for forecasting time series data in various fields, including economics, finance, and climate science. They provide a flexible framework for capturing different time series patterns and trends.

V. Results

A. Support Vector Classifiers(SVC)

Model Score: The model score, commonly known as R^2 (R-squared), reflects the proportion of variance in the dependent variable explained by the regression model, with higher values indicating better fit. It is calculated as the ratio of the regression sum of squares (SSR) to the total sum of squares (SST).

Train/Test Split : 80 - 20
model's score for the Test Set : 0.488
model's score for the Validation Set : 0.519

B. XGBoost

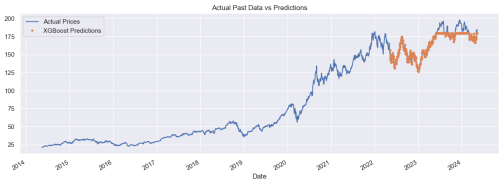


Fig. 6. XGBoost on AAPL

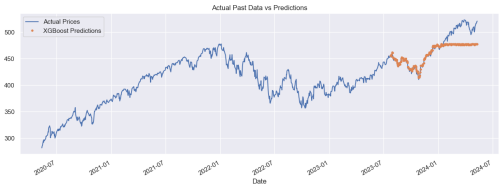


Fig. 7. XGBoost on SPY

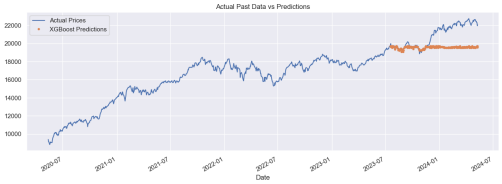


Fig. 8. XGBoost on Nifty50

TABLE I
XGBoost on Different Stocks

Stock	RMSE	MAE
Apple	5.23	2.98
SPY	19.86	12.86
Nifty50	1734.91	1302.35
Infosys	0.10	0.08

A. *Prophet*

TABLE I
PROPHET ON DIFFERENT STOCKS

Stock	RMSE	MAE
Apple	63.26	61.01
SPY	65.83	52.43
Nifty50	2257.25	1994.14
Infosys	2.06	1.65

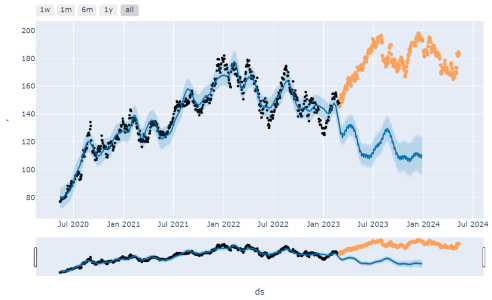


Fig. 1. AAPL

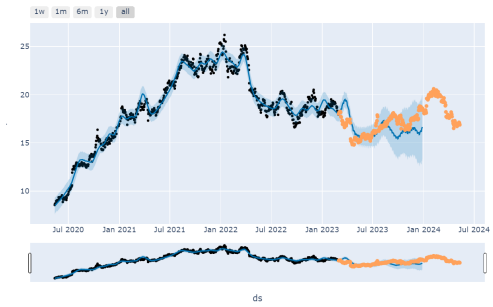


Fig. 2. INFY

Fig. 3. NIFTY50

B. *UniVariateLSTM*

TABLE II
UNIVARIATELSTM ON DIFFERENT STOCKS

Stock	RMSE	MAE
Apple	3.85	3.15
SPY	6.79	6.79
Nifty50	579.90	521.29
Infosys	0.47	0.39

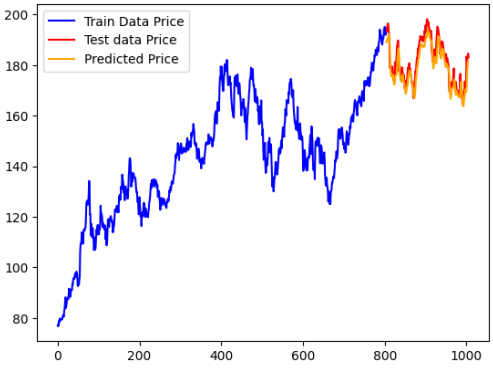


Fig. 4. aapl

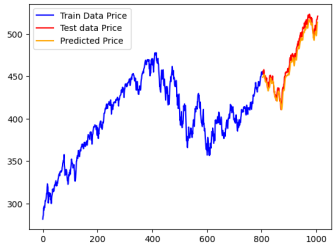


Fig. 5. spy

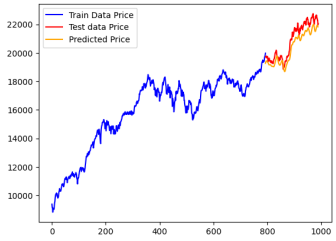


Fig. 6. nifty

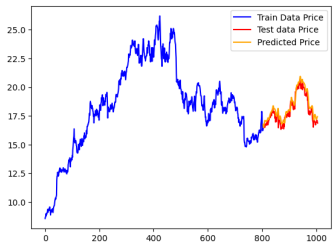


Fig. 7. infy

A. MultiVariateLSTM

TABLE I
MULTIVARIATELSTM ON DIFFERENT STOCKS

Stock	RMSE	MAE
Apple	4.11	3.32
SPY	6.65	5.29
Nifty50	631.12	538.22
Infosys	0.48	0.36

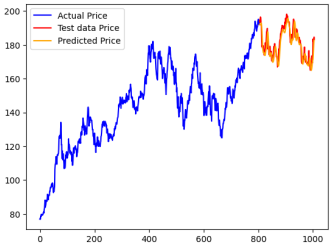


Fig. 1. aapl

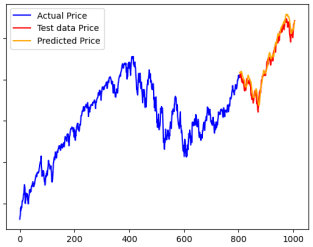


Fig. 2. spy

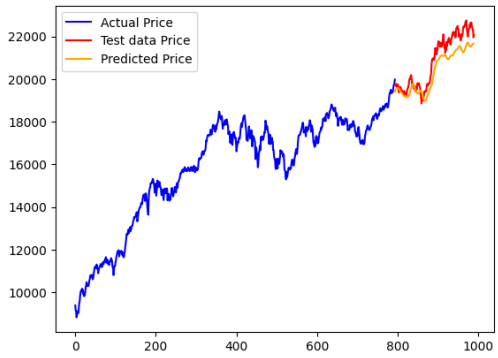


Fig. 3. nifty

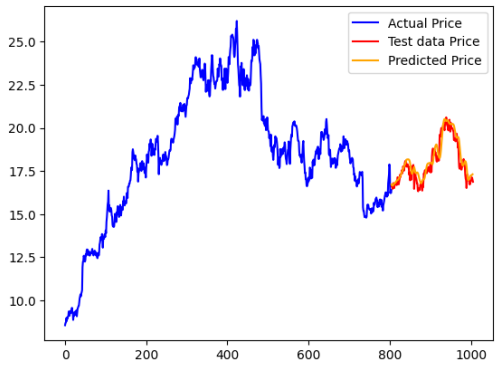
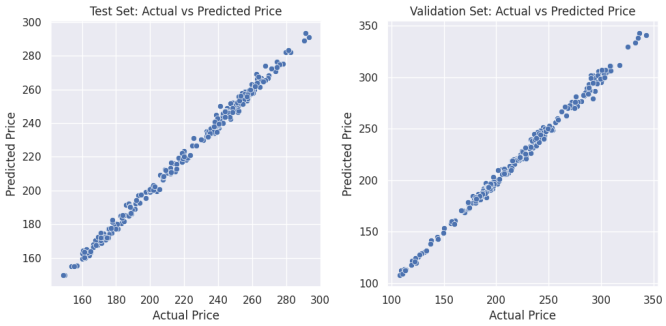


Fig. 4. infy

B. Linear Regression

We attached below our observation / results given by our model for the best split -



Train - Test Split : 80.0 - 20
Coefficients - [-6.7e-01 9.1e-01 7.6e-01 2.5e-10]
MSE for the Test Set: 5.40
MSE for the Val Set: 11.16

C. ARIMA

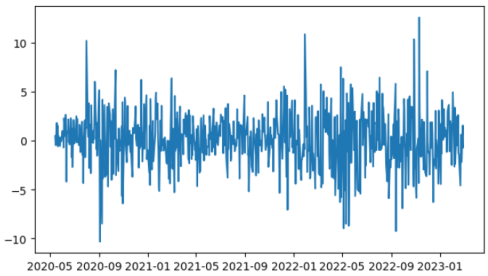


Fig. 5. Residual Plot of Apple Stock

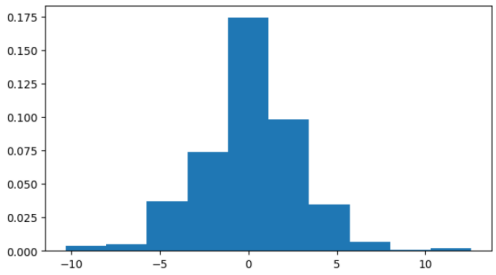


Fig. 6. Histogram of Residuals

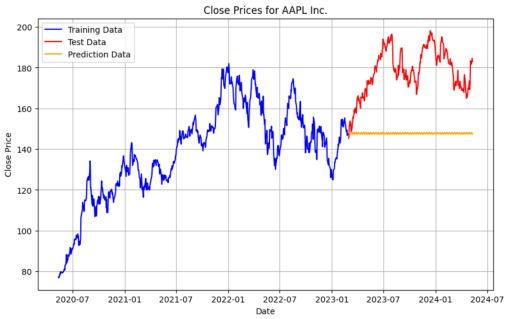


Fig. 7. ARIMA on Apple Stock

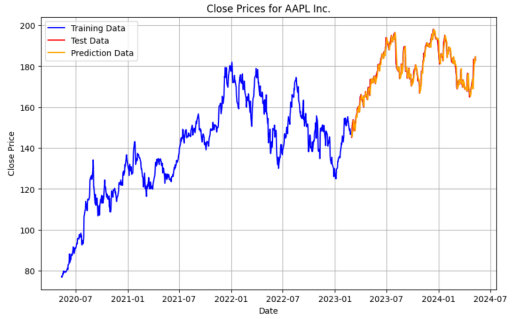


Fig. 8. Arima with Rolling Forecast Origin

TABLE II
ARIMA ON DIFFERENT STOCKS

Stock	RMSE	MAE
Apple	2.30	1.74
SPY	3.94	2.84
Nifty50	134.00	104.27
Infosys	0.27	0.18

I. CONCLUSION AND LIMITATIONS

A. Conclusion

In this study, we explored various machine learning models for stock price prediction, including Support Vector Classifiers (SVC), Linear Regression, and Long Short-Term Memory (LSTM) networks , Arima , Prophet , XGBoost etc. We leveraged the YFinance dataset, which provides historical stock prices and financial metrics for publicly traded companies, focusing primarily on Apple ,SPY, Nifty50,Infosys stocks. Our findings indicated how the ARIMA model outperformed almost all the other models on all the different evaluation metrics that we could use.

A notable finding was that both Multivariate and Univariate LSTMs outperformed all ensemble methods like XGBoost. However, the difference between them was marginal, suggesting that Univariate LSTM can achieve comparable performance while requiring less computational resources. This also implies that we could have employed XGBoost to identify the best features and then applied Multivariate LSTM solely on these features.

The Ensemble learning methods showed that they could perform well at the initial stages of testing only , after which the predictions in most cases cannot accurately predict the trend of the time series. A possible reason for this could be due to the non-stationarity of the series meaning the both the mean and the upper limit of the price point increase with time and go much beyond what our model has seen while training , this could be a potential bottleneck , preventing it from predicting accurately. Additionally, we observed an unusual scenario with Infosys stock, where the highest value was reached only in the training set (a situation uncommon in stock markets where stocks are expected to grow higher). In this case, we found that XGBoost outperformed ARIMA, which is otherwise the best-performing model.

TABLE III
COMPARISON OF METRICS FOR INFOSYS

Model	MAPE	MAE	MSE	RMSE	R2	SMAPE
XGBoost	0.44	0.08	0.01	0.10	0.99	0.44
Univariate LSTM	2.17	0.39	0.22	0.47	0.84	2.14
Prophet	9.24	1.65	4.23	2.06	-1.17	9.56
Multivariate LSTM	2.03	0.36	0.23	0.48	0.83	2.01
ARIMA	1.07	0.18	0.07	0.27	0.96	1.06

TABLE IV
COMPARISON OF EVALUATION METRICS ON APPLE

Model	MAPE	MAE	MSE	RMSE	SMAPE
XGB	1.63	2.98	27.35	5.23	1.67
Prophet	33.82	61.01	4002.21	63.26	41.22
ARIMA	0.99	1.74	5.31	2.30	0.99
Univ LSTM	1.73	3.15	14.81	3.85	1.74
Multiv LSTM	1.83	3.32	16.91	4.11	1.84

TABLE V
COMPARISON OF METRICS ON SPY

Model	MAPE	MAE	MSE	RMSE	SMAPE
XGBoost	2.56	12.86	394.35	19.86	2.63
Univariate LSTM	1.21	5.74	46.13	6.79	1.22
Prophet	10.94	52.43	4333.10	65.83	11.93
Multivariate LSTM	1.14	5.29	44.22	6.65	1.13
ARIMA	0.64	2.84	15.54	3.94	0.64

B. Limitation

In both the Univariate and Multivariate LSTM model,after applying MinMax scaling, the maximum value in the dataset is transformed to 1, setting the upper limit for the scaled values. While this normalization facilitates model training and convergence, it poses a limitation in scenarios like stock prediction. In stock markets, prices can continually rise, leading to new maximum values over time. Consequently, if the model's training range is limited by the maximum value observed during preprocessing, it may be unable to accurately predict values beyond that range, potentially underestimating future price movements as it never encounters values above 1 during training.

REFERENCES

[1] Krish Naik, "Stock Market Forecasting with LSTM", GitHub, 2019. [Online]. Available: https://github.com/krishnaik06/Stock-MArket-Forecasting

[2] GeeksforGeeks, "Predicting Stock Price Direction using Support Vector Machines", GeeksforGeeks, 2022. [Online]. Available: https://www.geeksforgeeks.org/predicting-stock-price-direction-using-support-vector-machines/

[3] Soumya Bhattacharjee, "Cross-Validation in Time Series", Medium, 2018. [Online]. Available: https://medium.com/@soumyachess1496/cross-validation-in-time-series-566ae4981ce4

[4] Saahil Madge, "Predicting Stock Prices using Machine Learning", Princeton University, 2020. [Online]. Available: https://www.cs.princeton.edu/sites/default/files/uploads/saahil_madge.pdf

[5] Gaurang Sonkavde, Deepak Sudhakar Dharrao, Anupkumar M. Bongale, Sarika T. Deokate, Deepak Doreswamy, and Subraya Krishna Bhat, "Forecasting Stock Market Prices Using Machine Learning and Deep Learning Models: A Systematic Review, Performance Analysis and Discussion of Implications", MDPI, 2021. [Online]. Available: https://www.mdpi.com/2227-7072/11/3/94

[6] Rohan Paul, ”Machine Learning and Deep Learning Code for Time Series Analysis”, GitHub, 2021. [Online]. Available: https://github.com/rohan-paul/MachineLearning-DeepLearning-Code-for-my-YouTube-Channel/tree/master/Time-Series
[7] Facebook, ”Prophet: Forecasting at Scale”, Facebook, 2022. [Online]. Available: https://facebook.github.io/prophet/
[8] Krish Naik, ”Stock Market Forecasting with LSTM”, GitHub, 2019. [Online]. Available: https://github.com/krishnaik06/Stock-MArket-Forecasting
[9] Gaurang Sonkavde, Deepak Sudhakar Dharrao, Anupkumar M. Bongale, Sarika T. Deokate, Deepak Doreswamy, and Subraya Krishna Bhat, ”Forecasting Stock Market Prices Using Machine Learning and Deep Learning Models: A Systematic Review, Performance Analysis and Discussion of Implications”, MDPI, 2021. [Online]. Available: https://www.mdpi.com/2227-7072/11/3/94
[10] ”statsmodels.tsa.arima.model.ARIMA”, Statsmodels, 2022. [Online]. Available: