Financial Forecasting Analysis

Integrating Sentiment Analysis and Global Signals for Enhanced Stock Market Prediction: Indian Stock Market

Team 21 - Trading titans: Amit Kumar Jha, Shamil Murzin and Yang Yan
(SIADS 699 Capstone project - August 2023)

1. Introduction

1.1 Traditional Approaches to Stock Price Forecasting

The journey of stock price prediction dates back to the early 20th century, with pioneers like Yule and Walker introducing the autoregressive model (AR). The framework evolved through the contributions of Whittle and Box-Jenkins, encompassing moving averages and seasonality. These statistical methods laid the foundation but faced challenges in accommodating the nonlinear nature of stock prices.

1.2 Evolution Toward Machine Learning

Learning Modern stock price prediction techniques embrace data-driven learning. The advent of Machine Learning (ML) and Artificial Intelligence (AI) marked a shift from assumptions to adaptability. Ensemble methods like Random Forest and Gradient Boosting, alongside neural network models, emerged to capture the dynamics of nonstationary stock prices.

1.3 Project statement

In a landscape dominated by a market capitalization of 3.2 trillion USD, India's stock market stands as the fifth-largest in the world. However, despite its immense significance, the exploration of stock price analysis has predominantly centered around the USA, leaving a void in our understanding of the Indian stock market. Our project aims to bridge this gap by delving deep into the intricacies of the Indian stock market, employing a comprehensive array of statistical techniques for the purpose of predicting stock prices. By unraveling the complexities unique to the Indian context, we strive to contribute valuable insights and a robust analysis that will shape the future of stock price prediction in this dynamic market.

2. Data Sources

2.1 Yahoo Finance

We collected historical stock data from Yahoo Finance[1], including price, volume, and other relevant information for various Indian stocks. We also used this to get foreign market index data like, Shanghai, Singapore and Nikkei index and crude oil prices and currency rates.

2.2 Economic Times

We scraped approx 1.5 million news articles from one of the most popular financial news daily Economic Times[2] for the period of Jan-2008 to May-2023 to extract sentiment and topic information to capture market sentiments.

2.3 BSE India

Financial results of a company gives us insight into the real value of a company.
We scraped quarterly/yearly financial results of companies from BSE India[3] from Apr-2008 till May-2023, enriching our dataset with valuable fundamental information.

3. Methodology

3.1 Unsupervised Learning for Stock Picks

In our effort to find a robust forecasting technique, we did not just choose the so-called sexy stocks. Instead we decided to choose the stocks, which are most dissimilar based on their time series characteristics. This approach ensures that our modeling techniques can be utilized for all kinds of stocks with completely different characteristics. To do this we followed below steps:

1. Select the latest five months of data from Jan-23 to May-23
2. Used different scaling/signal processing steps to normalize the data:
   a. MinMax scaling, b. Standard Scaling
   c. Log Returns Scaling, d. Implemented signal processing by CEEMDAN,
3. Further we fit the data through yellowbrick's KElbowVisualizer using tslearn’s TimeSeriesKMeans method for data scaled with different methods to find the optimal clusters for each stock data processed through all kinds of scaling
4. We found thirteen clusters for data scaled with Standard Scaling to be optimal. [Appendix A]
5. Now we created a dissimilarity matrix for all the tickers using Dynamic Time Warping as metric and extracted the pair of stocks with the most dissimilarity distance across clusters for each cluster.
6. We chose eight stocks from 13 clusters, for the feasibility of analysis in limited time.
7. We further downsized it to five tickers by analyzing the data availability for these stocks, which led us to choose these stocks:
   ['EIHOTEL.BO', 'ELGIEQUIP.BO', 'IPCALAB.BO', 'PGHL.BO', 'TV18BRDCST.BO'].

3.2 Data Cleansing and Splitting

For financial data, we deleted columns that contain only NaN values and performed forward filling, followed by backward filling to fill in the remaining missing values. For training/evaluation/test, we use the strategy of 80/10/10 ratio. That is 80% of the data was used for training while the next 10% of data was used for evaluation and model finetuning. Last 10% of the data was used for testing and final reporting purposes.

We used 10% of the news articles for topic modeling. These 10% of the news articles were taken in a way so as to be 10% from all the months. This was done to make sure that our topic model has all kinds of news to understand the underlying topics of news.

3.3 Feature Engineering

Creating Technical Features

Feature engineering is an integral part of any machine learning problem and we spent a lot of time creating lots of features. Apart from using daily prices, we created 86 technical features using python library ta(technical analysis)[4]. Further, we created rolling averages for all these technical features, to create additional features, for the window of 5, 10, 20, 50, 100, 200 days, which gave us 84 * 6 = 504 additional features.

Creating Fundamental Features

Warren Buffett's enduring wisdom—*'It's far better to buy a wonderful company at a fair price than a fair company at a wonderful price'*—holds a mirror to our approach as we delve into the quarterly/yearly results of companies to create fundamental features. [5] This quote resonates as a reminder that the intrinsic strength of a company, reflected in its fundamentals and financial health, is of paramount importance.

Creating Sentiment Features

Our intuition behind using sentiment features was that performance of a company is shaped mainly by three factors - general economy/political environment of the country, changing market conditions of a particular industry and finally actions/performance of the company itself. We tried to capture these signals through daily news, focused industry news and particular company news by extracting sentiments for each category using Vader/TextBlob.

To compute industry news sentiment, first we identified the industry/topic of a news article by topic modeling through BERTopic and then computed the aggregated industry wise news sentiments. For our chosen five stocks- EIHOTEL, ELGIEQUIP, IPCALAB, PGHL, TV18BRDCST, which belong to Hotel, heavy machines, pharma and news broadcasting industry, we identified the topic ids to be [33, 921, 495, 495, 385], based on manual topic exploration (Figure 1).

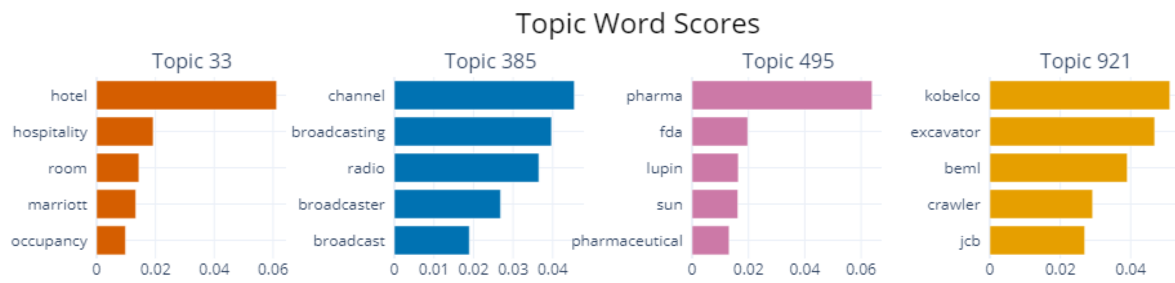

Figure 1 Topic Word scores

Which were then merged with corresponding manually identified industry/topic of the ticker to extract the industry news sentiment for the particular ticker. To compute ticker news sentiment, we filtered the news containing the company name and computed their sentiment.

Further, to avoid the data leak, as we are using daily stock data, we used yesterday's news sentiments to predict today's stock price. We understand that it may not be the best approach as if a news came after opening the market, stock prices would already accommodate the impact of the news and would not wait for the next day's market opening.

However, we had to make a trade off between the frequency of ticker data we use and the extraction of sentiments at the same frequency level.

Creating Global Impact Features

In the current globalized world, whatever small/big a company is, it can not insulate itself from ever changing global economic conditions. Like steep depreciation of Rupee means that industries heavily dependent on import would have to wear the wrath of steep higher import costs.[7]

To capture these signals, we used foreign indexes like Shanghai, Nikkei and Singapore market etc. Additionally we used oil price, dollar index, Rupee rate to capture global economic conditions. Here instead of using these daily raw data, we computed daily percentage change to use as features.

In total using these methods we crafted a whopping 633 features.

Transforming TimeSeries to Supervised Data

While the deep learning model LSTM (Long Short-Term Memory) networks possess inherent temporal sense due to their architecture designed to handle sequential and time-dependent data. None of the supervised learning methods have any temporal sense and if used directly, they would have only one day of data in the training and would not be able to learn from past days of the data.

Hence, we needed to convert our time series to data in a way that it would have a temporal sense. For this, we used a ten day window, and in a rolling fashion flattened 10 days data to a row and predicted an 11th day high price using our custom target explained later in the report. This also meant that our training would start with prediction for the 11th day and would have 10 days less data in training/evaluation.

Target feature
From a traders perspective, the next High price has more significance as if a trader knows, tomorrow's high price he/she may decide to enter when it is lower than predicted higher price with a margin and sell at predicted high price.
However, predicting the next day high turned out to be a lot noisier. Stock prices can experience significant intraday volatility, with price fluctuations influenced by a wide range of factors, including market sentiment, news, economic data, and trading activities.

Log transformation is suggested to stabilize variance of a time series with non-constant variance. Hence we decided to experiment with a custom target of ln(high/yesterday_close), while taking the log can help to make the series stationary dividing the series by yesterday_close can normalize the data. As stock prices may change significantly upwards over a period of 10-15 years, by dividing the high by yesterday's close price, our target becomes range bound and is easier to scale.

We confirmed the same by performing the Adfuller test for high price and our custom target and as can be seen, p-value with our custom target is much lower than 0.05 that is time series with custom is stationary (Figure 2).

```
# ADF test
result = adfuller(combined_df['ln_target'], autolag='AIC')

print("ADF statistic:", result[0])
print("ADF p-value:", result[1])
```

```
ADF statistic: -12.90548692306805
ADF p-value: 4.161531746555872e-24
```

Figure 2. Adfuller Test Result - Custom Target(ln(high/yesterday_close))

3.4 Feature Importance and Dimensionality Reduction

Now the challenge was to select a feasible number of features which strike a balance between model complexities, model accuracy, explainability of the model and finally the cost of computing these features.

Base models for all tickers were used to compute feature importance with TreeSHAP, an algorithm to compute SHAP values for Decision Trees based models.[8] SHAP (SHapley Additive exPlanation) is a game theoretic approach to explain the output of any machine learning model. The goal of SHAP is to explain the prediction for any instance x as a sum of contributions from its individual feature values.

As different algorithms have distinct ways of learning and making predictions, we decided to find the top features specific to the models we are using instead of one size fits all approach.

Post which we computed the average SHAP values for all the tickers for a model and chose top fifty features for a specific model to reduce the feature space from 633 to 50 (Figure 3). For the LSTM models we used the Captum library, as SHAP doesn’t work well with PyTorch models.

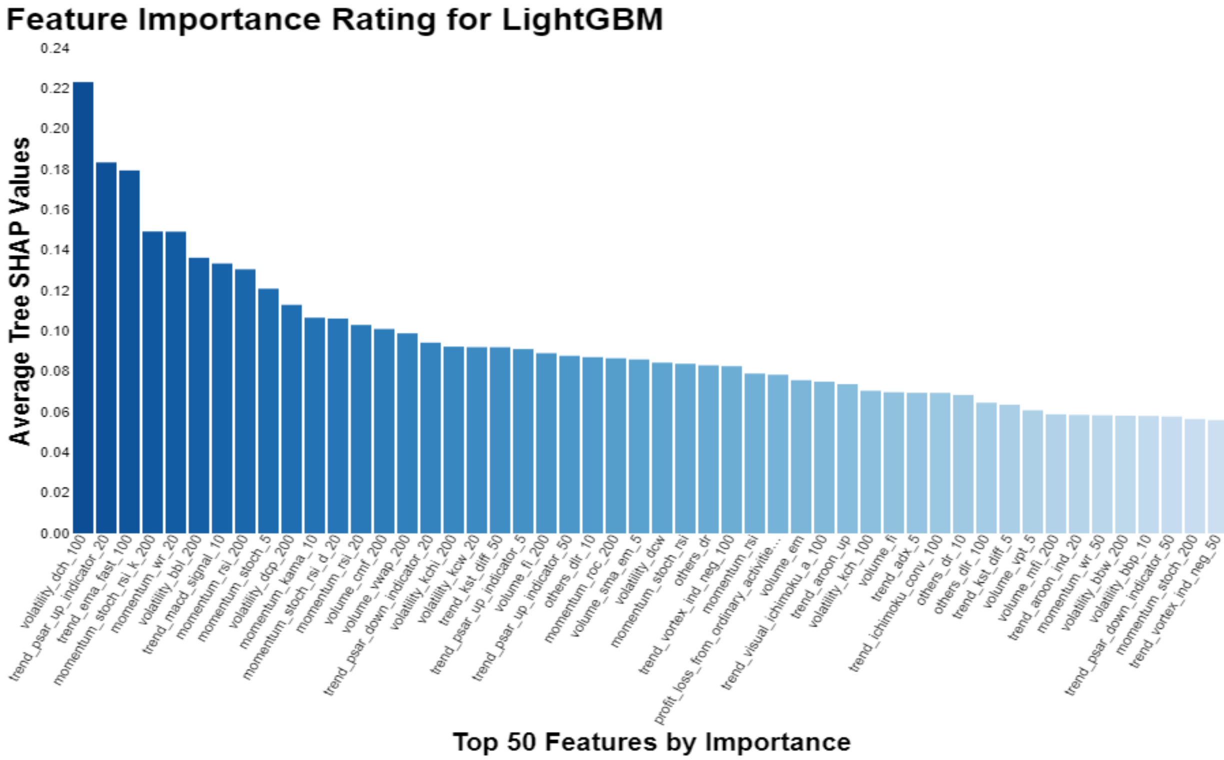

Figure 3. SHAP features importance for LGBM model.

As can be seen there are many features in the top 50 which have a moving window of 200 days. It may be largely due to the fact that large changes in the longer window features indicate a significant change in the company, momentum, volatility and volume.

For LSTM models we used the Captum library to compute feature importance.

3.5 Market Sentiment Analysis

We created six sentiment features corresponding to daily news, industry news and stock news each using Vader/TextBlob. However, when we ran the TreeSHAP to find the top 50 features, none of these features came in the top 50, instead they were all ranked lower than 50+ and often were ranked 300+. Hence to quantify the news sentiment's impact we performed two experiments:

3.5.1 Sentiment Correlation Hypothesis Testing

In the first experiment, we did hypothesis testing to find if there is a correlation between stock price and different kinds of news sentiment(daily/topci/ticker news sentiment) (Figure 4). We found that while four of the stocks have moderate positive correlation one had weak negative correlation(TV18BRDCST). Weak negative correlation may be attributed to the fact that if there is any positive news for a stock, stock rallies that day and it often goes down the next day owing to profit booking by traders.

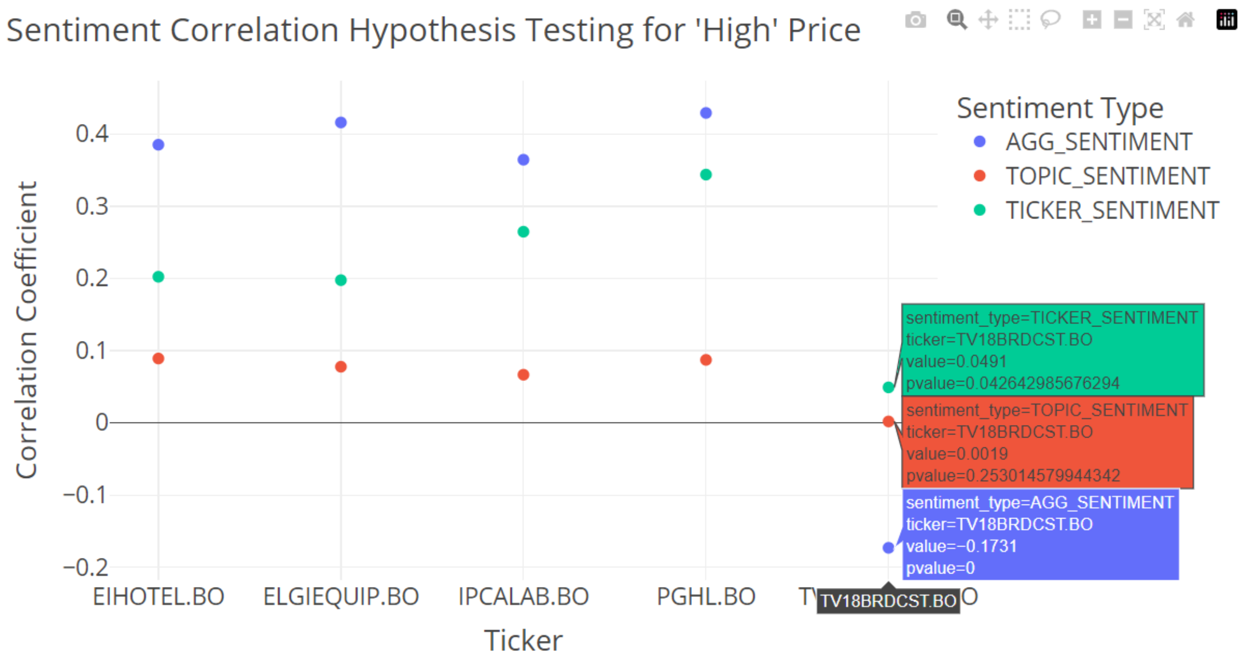

Figure 4 Sentiment Correlation Hypothesis

3.5.2 Stock Price Prediction(With/Wo Sentiment scores)

In our second experiment, we ran our baseline models for five tickers with top 50 features(without sentiment)/top 44 features + sentiment features. Here our intuition was to find if our six sentiment features are better/worse than the worse six features from the top 50 features.

Through experiments we found that for the RandomForest model MAPE score improves/worsens for two/two tickers while almost no change is found for one ticker(PGHL). Similar results were observed for other models (Table 1).

Ticker/Sentiment_Impact	LinearRegression	RandomForest	LightGBM	Prophet	LSTM
EIHOTEL	0.010	-0.130	-0.100	-0.030	-0.670
ELGIEQUIP	-0.020	0.110	0.000	-0.030	-0.570
IPCALAB	0.040	0.160	-1.270	-0.011	0.930
PGHL	-0.030	0.020	0.040	-0.010	0.050
TV18BRDCST	-0.070	-0.120	-0.120	-0.002	0.760

Table 1. Sentiment MAPE scores impact for different methods.

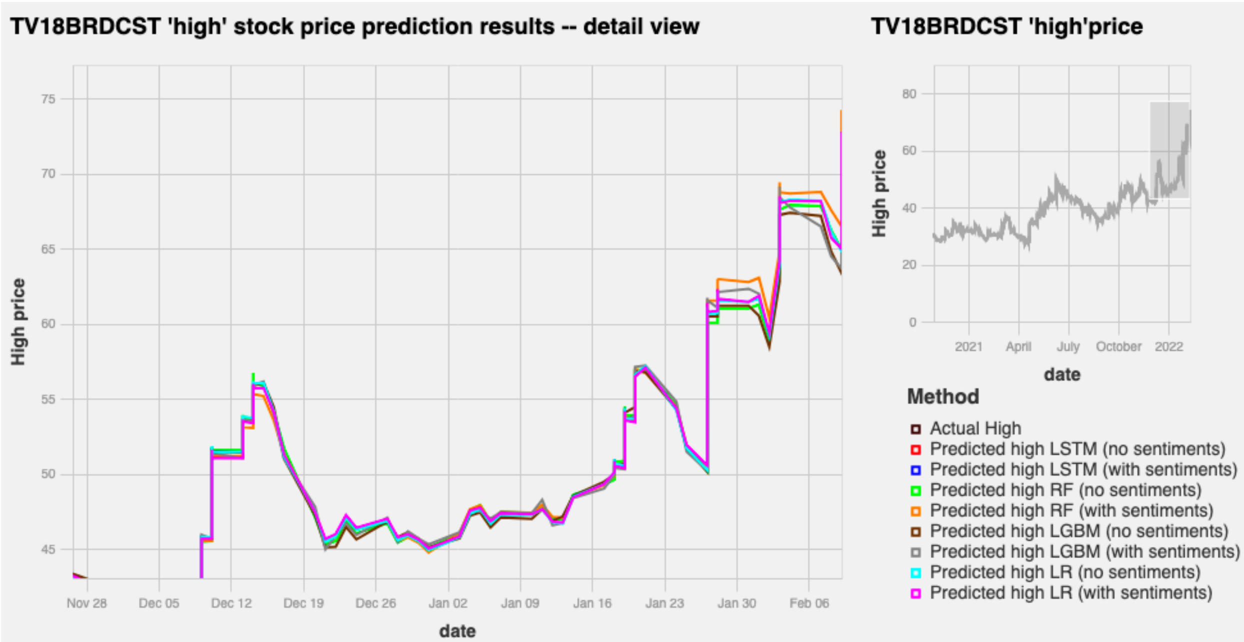

Figure 5 1-day stock prediction example.

Detailed scores for all the models with/without sentiment scores are provided in Appendix C.

4. Result and Discussion

4.1 Numeric Accuracy Results:

MAPE	Linear Regression	RandomForest	LightGBM	Prophet	LSTM
EIHOTEL	1.48	2.02	1.52	**0.88**	2.13
ELGIEQUIP	1.83	2.88	1.83	**1.33**	13.99
IPCALAB	1.72	3.31	1.21	**1.09**	4.54
PGHL	1.14	1.7	1.08	**0.66**	7.66
TV18BRDCST	1.77	1.85	1.68	**1.16**	10.32

Table 2. Numeric accuracy results.

In our experiments, we evaluated five models to forecast stock prices (“High” price). These models are: Linear Regression, Random Forest, LightGBM, LSTM, and Prophet (Figure 6). While there are multiple metrics in use for regression problems - mean absolute error(mae), mean squared error(mse), root mean squared error(rmse) and mean absolute percentage error(mape), mape is often preferred. We chose mape as it is easy to understand for both developers and users and it can be compared for stocks even if they have varied price ranges. It calculates the averaged percentage of absolute distance between predicted and actual output (Table 2).

Figure 6. Different methods predictions for IPCALAB ticker.

Linear Regression

Linear Regression, here proves that simplicity is the king. With no parameters to tune for, yet it performs as one of the best models. If a simple model such as Linear Regression can give such a good performance, then we must have done really good feature engineering. Which helps the model derive the relationship between independent and dependent variables easier.

This also gives an insight on how to keep things simple and do extensive feature engineering in solving a business problem through data science.

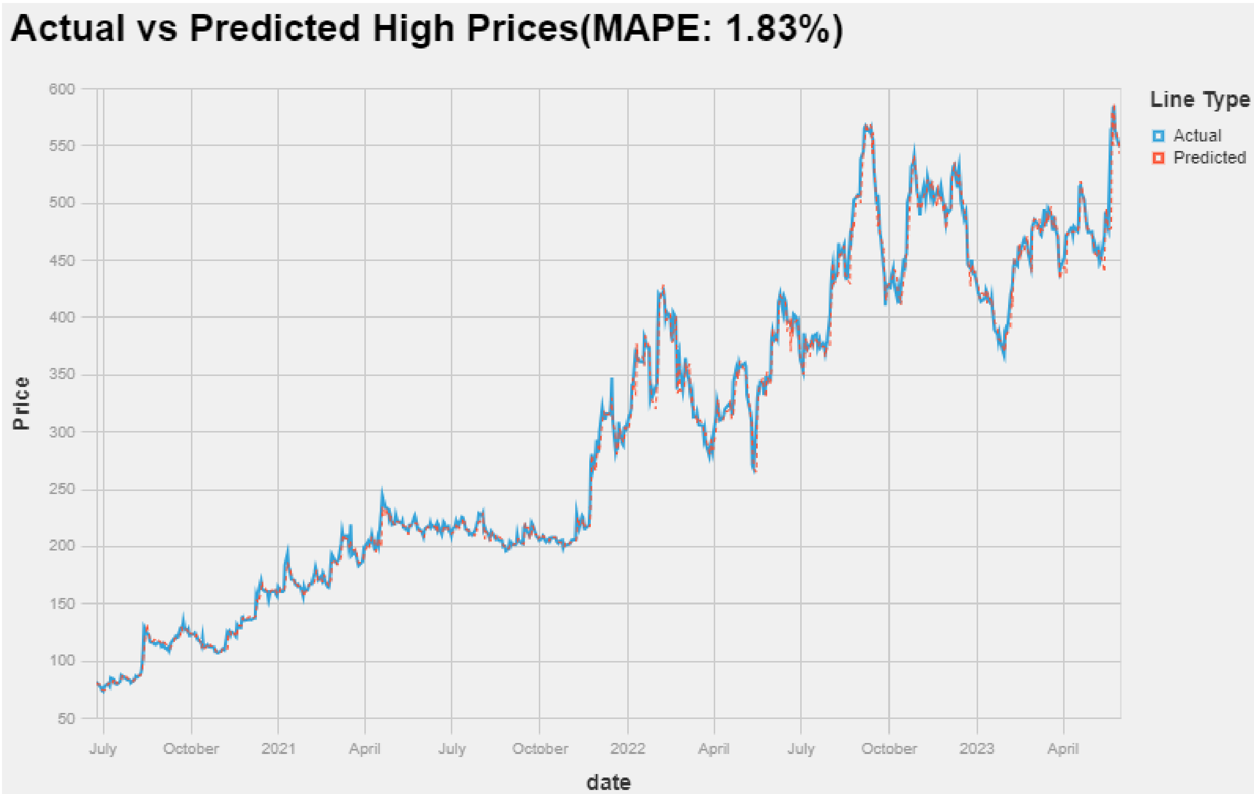

Figure 7. High price prediction with LR method for ELGIEQUIP.

Random Forest

We got mape score in the range of 1.7% to 3.3%, which is not bad. However it is worse than LinearRegression. RandomForest is known to perform well however its performance here may suggest model overfitting or need for more fine tuning.

LightGBM

LightGBM is easily the second best model after Prophet for our stock price predictions. It gives us the mape score in the range of 1.1 to 1.8% and is always either second best or is tied with the second best model.

Prophet

Despite having very few hyperparameters to fine-tune, this simple model from META delivered the best performance. This model provided the MAPE score in the range of 0.66-1.33% for all the tickers. (Figure 8).

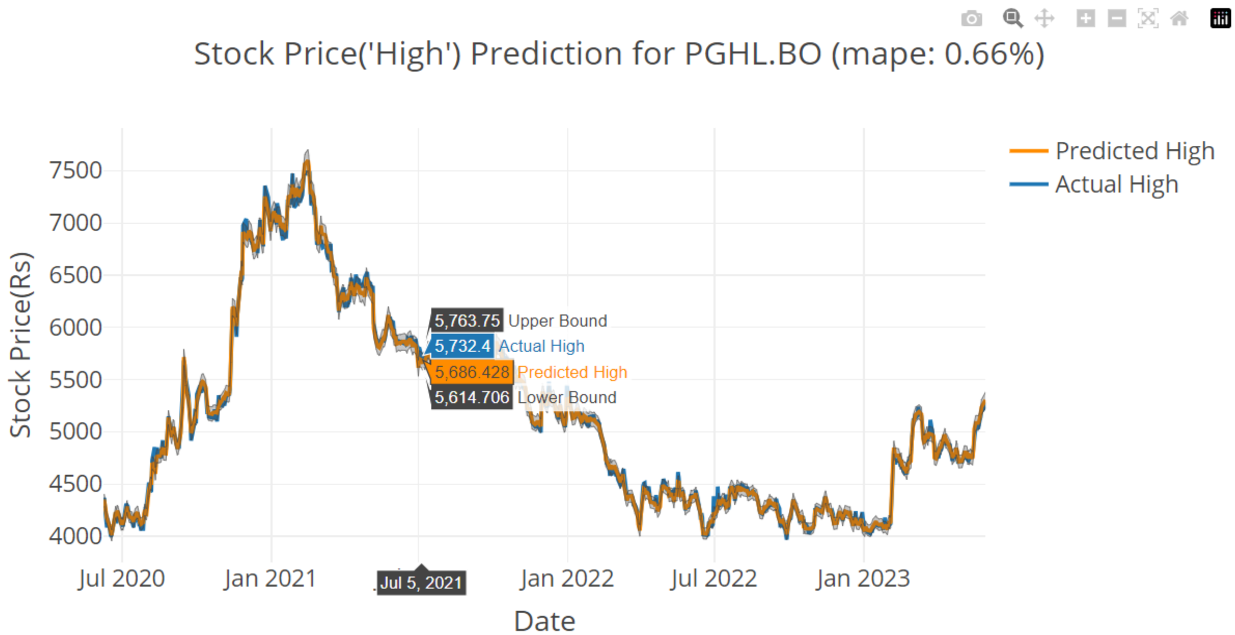

Figure 8. High price prediction with Prophet method.

LSTM

While we expected LSTM to perform better, we didn't get the scores we were expecting, even after much fine tuning. Its MAPE score of 10/14% is easily off the charts and we wanted to investigate this as to why this is so. However, for lack of time, we could not do this.

In general, the LSTM showed less accuracy, which can be explained by the need for better tuning of the models.