Technical Report: Final Project DS 5220: Supervised Machine Learning and Learning Theory

Team Members:
Dilep Shetty Ittanguru Venkatesh
ittanguruvenkatesh.d@northeastern.edu
Sai Nikhil Kunapareddy
kunapareddy.s@northeastern.edu

Khoury College of Computer Sciences
Data Science Program

July 30, 2024

Contents

1 Introduction

In this project, we aim to utilize time series forecasting to predict the future closing stock value of the S&P 500 index. We will use four different models: ARIMA (Autoregressive Integrated Moving Average), SARIMAX (Seasonal Autoregressive Integrated Moving Average with Exogenous Variables), LSTM (Long Short-Term Memory) Network, and Meta's Prophet. These models will be employed to make predictions and a comparative analysis will be performed at the end. Timeline considered for analysis will be from 2010 to 2019 (inclusive). This decision is taken to exclude the influence of unpredictable market behaviors caused by financial crises and pandemics.

The S&P 500 index is a highly recognized and widely followed equity index in the United States. The S&P 500, short for Standard & Poor's 500, comprises the top 500 publicly traded companies in the country. It serves as a crucial indicator of the overall performance of the U.S. economy and provides insights into the performance of various industries.

Time series forecasting involves predicting future values by analyzing a series of past data points. It is widely used in fields such as finance, economics, and weather forecasting. This technique relies on identifying patterns, trends, and cycles in historical data and using them to make predictions about future values.

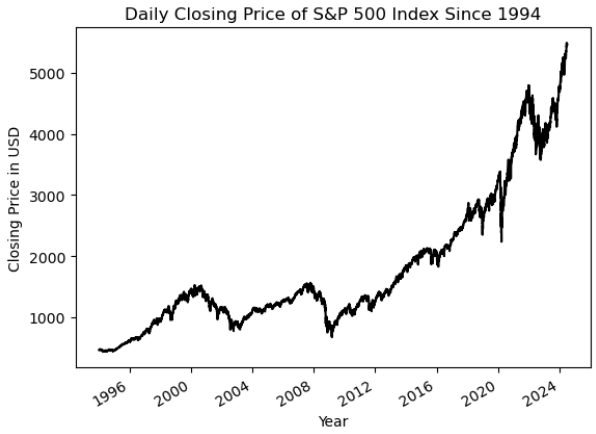

Figure 1: Closing stock price value over years

2 Literature Review

Existing research on stock price analysis employs a range of statistical, machine learning and hybrid methodologies. Time series models like ARIMA and its variants are widely used for their ability to capture linear trends and seasonal patterns. Machine learning techniques such as Support Vector Machines (SVMs) and Artificial Neural Networks (ANNs), particularly Long Short-Term Memory (LSTM) networks, have shown promise in modeling complex, nonlinear relationships in stock price data. Inaddition to these two categories hybrid machine learning approaches like SVM-KNN and ARMA-LSTM are used for stock price forecasting to find a good balance between model complexity and computability.

Despite the advancements, significant gaps remain in the literature. The integration of diverse data sources, such as social media sentiment and macroeconomic indicators, is still underexplored, potentially limiting the accuracy of predictive models. Real-time prediction capabilities and the explainability of complex models like ANN's and LSTM's are critical areas needing further development. Moreover, many models struggle with robustness against market anomalies and lack comprehensive comparative studies to evaluate their performance across different market conditions. Addressing these gaps could enhance the reliability and utility of stock price prediction models, providing more robust tools for investors and policymakers.

3 Methodology

The project employs a comprehensive methodology for analyzing and predicting S&P 500 closing stock price. Yahoo finance API is utilized to collect historical data for the stock index. Preprocessing steps like handling missing values, feature engineering and dataset schema manipulation were done for the data to adapt based on the model requirements. Exploratory data analysis (EDA) is performed to understand the distribution and change of the closing price over years. The index performance is compared with the top and bottom five percent of the organizations included in the index to understand their influence. Time series analysis involved differencing and decomposing data and fitting ARIMA and SARIMA models. Machine learning models, including LSTM network and Meta's PROPHET are considered to take care of non-linearity and local fluctuations. The performance evaluation metrics used to compared the model accuracy are MAE(Mean Absolute Error) and RMSE(Root Mean Squared Error) and results are visualized using tools Matplotlib.

Mean Absolute Error is a measure of prediction accuracy in a regression model, calculated as the average of the absolute differences between the predicted and actual values. Root Mean Square Error is calculated as the square root of the average of the squared differences between predictions and actual values. These specific metrics are the most popular evaluation criteria for regression models and forecasting

The formula for Mean Squared Error (MSE) is:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

The formula for Root Mean Squared Error (RMSE) is:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where y_i represents the actual observed value at the i-th instance. $\hat{y}_i$ represents the predicted value at the i-th instance.

3.1 Data Collection

The data source for this project has been taken from Yahoo finance API. Which provided us with daily stock information (daily low price, high price, closing price) of the requested stock index. The collection method involved calling the API by providing required stock index and time period of interest and storing the information in a pandas dataframe.

3.2 Data Preprocessing

Data Cleaning
Duplicate values and 'NaN' values does not exist in the dataset as it is extracted from

a reputed API. All the necessary corrections and precautions were taken care by the developers. Few preliminary checks like computing the count of 'NaN' values, requesting data type of each feature, were done as a part of cross validation.

Feature Engineering
While implementing statistical methods like ARIMA and SARIMA, differencing of multiple orders for daily and seasonal windows were done on the closing price feature column as a part of feature engineering. Linear and exponential moving averages were computed to understand the trend of the stock under various time windows.

3.3 Analysis Techniques

ARIMA stands for AutoRegressive Integrated Moving Average. It is widely used for time series forecasting, capturing three aspects of the data: autoregression(AR), differencing(I) and moving averages(MA). Autoregression is the ability of the model to predict future values with the help of historical data. Differencing is a technique used to achieve stationarity (constant mean and standard deviation over time), that is removing any effect of upward or downward trend. Moving average window includes previous prediction errors in the model. The value of this window will define the number of lagged residual values to be included in the model.

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q} + \epsilon_t \tag{1}$$

where:

- Y_t is the value of the time series at time t.
- $\phi_1, \phi_2, \ldots, \phi_p$ are the coefficients of the autoregressive (AR) part.
- $\theta_1, \theta_2, \ldots, \theta_q$ are the coefficients of the moving average (MA) part.
- ϵ_t is the white noise error term at time t.
- p is the order of the autoregressive part.
- d is the order of differencing needed to make the series stationary.
- q is the order of the moving average part.

SARIMA stands for Seasonal AutoRegressive Integrated Moving Average. This model is an extension of previously discussed ARIMA model with an additional focuses on seasonality. This model assumes that the data considered has periodic fluctuations.

$$\Phi(B^s)\phi(B)Y_t = \Theta(B^s)\theta(B)\epsilon_t \tag{2}$$

where:

- Y_t is the value of the time series at time t.
- $\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p$ is the autoregressive (AR) polynomial of order p.

- $\theta(B) = 1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q$ is the moving average (MA) polynomial of order q.
- $\Phi(B^s) = 1 - \Phi_1 B^s - \Phi_2 B^{2s} - \cdots - \Phi_P B^{Ps}$ is the seasonal autoregressive polynomial of order P and seasonality s.
- $\Theta(B^s) = 1 + \Theta_1 B^s + \Theta_2 B^{2s} + \cdots + \Theta_Q B^{Qs}$ is the seasonal moving average polynomial of order Q and seasonality s.
- ϵ_t is the white noise error term at time t.
- p is the order of the non-seasonal autoregressive part.
- d is the order of differencing needed to make the series stationary.
- q is the order of the non-seasonal moving average part.
- P is the order of the seasonal autoregressive part.
- D is the order of seasonal differencing.
- Q is the order of the seasonal moving average part.
- s is the length of the seasonal cycle.

LSTM (Long Short-Term Memory) networks are a type of recurrent neural network (RNN) designed to learn and remember dependencies in sequence data over long periods. Unlike traditional RNNs, LSTMs are equipped with a special architecture that includes gates—namely, the input gate, forget gate, and output gate—which regulate the flow of information. This gating mechanism allows LSTMs to effectively retain relevant information and forget irrelevant details, making them particularly well-suited for time series forecasting.

Meta's Prophet is an open-source forecasting tool designed to handle time series data with strong seasonal effects and holidays. It is based on an additive model where non-linear trends can be fit using yearly, weekly, and daily seasonality, along with holiday effects. One of the key features of Prophet is its ability to automatically detect changes in trends and manage missing data, outliers, and shifts in seasonality. The model also includes components like trend changepoints, seasonal components, and holiday effects, which allow it to adapt to various temporal patterns in the data.

4 Results

Three different ARIMA models were computed with different orders of differencing. The first model comprises daily differencing, while the second model is computed with seasonality of one year. Finally, the last model contains both daily and seasonal differencing effects.

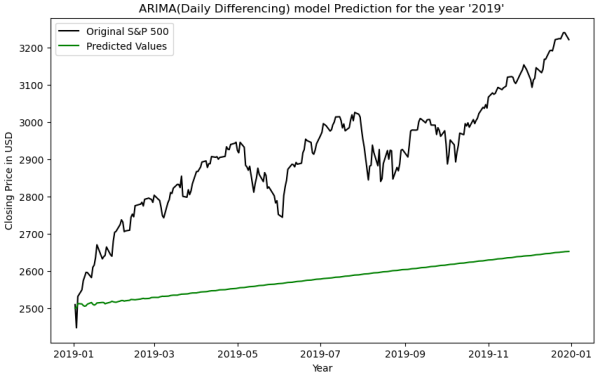

Figure 2: ARIMA (Daily Differencing) Predictions

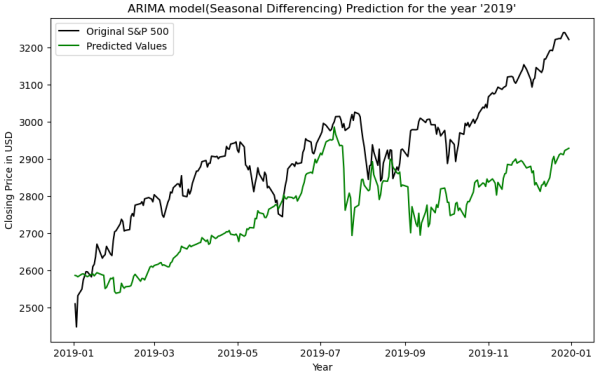

Figure 3: ARIMA (Seasonal Differencing) Predictions

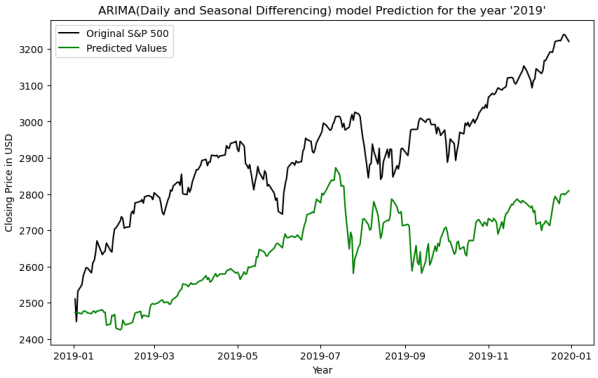

Figure 4: ARIMA (Daily and Seasonal Differencing) Predictions

The ARIMA model with seasonal differencing exhibited the best performance in comparison to its counterparts. Including only daily differencing gave poor results without

capturing the local fluctuations by just considering the broader upward trend in the dataset. Adding seasonal differencing to the data helped in capturing the local variations in the data but their combination created an offset in prediction away from actual values.

SARIMA model is an extension to the ARIMA model, which includes seasonal components in various aspects. This model was able to capture every fluctuation and performed way better than any ARIMA model.

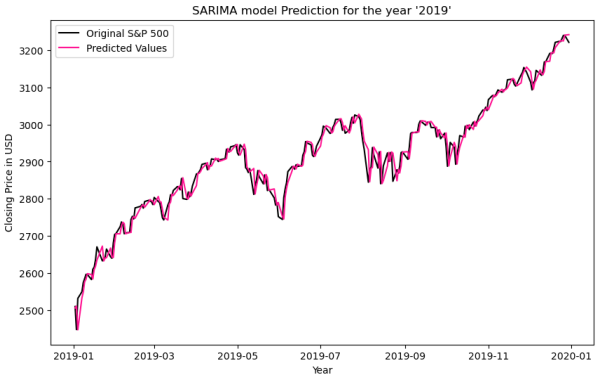

Figure 5: SARIMA Model Predictions

The best model performance in this project is exhibited by the LSTM network. As expected, the model is able to predict every local fluctuation with the given time windows.

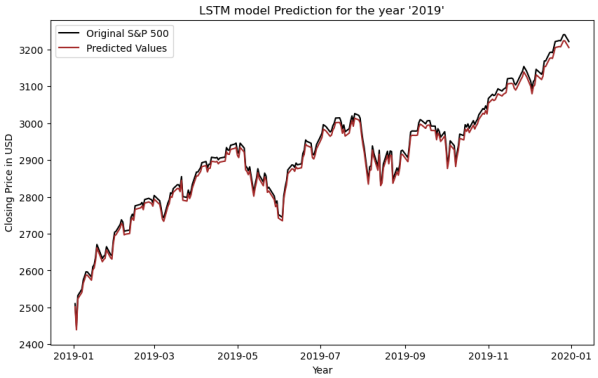

Figure 6: LSTM Network Predictions

Finally Meta's Prophet model is used with the help of API calls. Interestingly, this model couldn't perform better than LSTM network. We believe this sophisticated model can be trained to predict better with hyperparameter tuning but that front is not explored in this project.

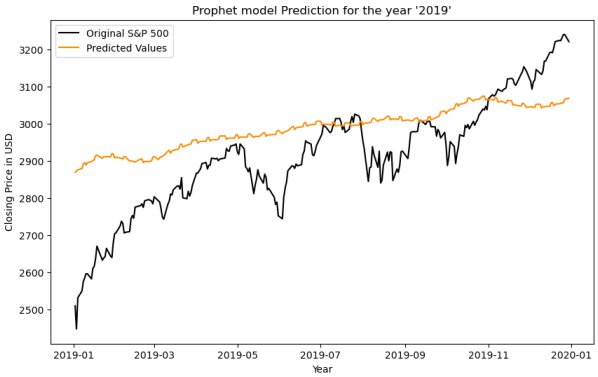

Figure 7: Meta's Prophet model prediction

5 Discussion

Given below is a table comparing the performance of all the six models with model evaluation metrics like mean absolute error and root mean square error.

Table 1: Performance comparison of different statistical and machine learning models

	Model Evaluation Metrics	
	Mean Absolute Error	Root Mean Square Error
ARIMA (Daily Differencing)	123350	351
ARIMA (Seasonal Differencing)	33409	182
ARIMA (Daily and Seasonal Differencing)	82643	287
SARIMA	505	22
LSTM	122	11
Prophet	16749	129

As we are considering error metrics like mean absolute error and root mean square error, it is important to understand that these values by themselves add less value but they can be a good comparative metric. LSTM network performed the best with respect to both MAE and RMSE followed by SARIMA. Prophet and ARIMA models are far behind(different order) especially when we compare their MAE values.

While the success of the LSTM model is expected but the failure of Meta's prophet isn't. Upon introspection, its poor performance might have something to do with modelling limitations without any hyperparameter tuning. Moreover, this model in general expects seasonality in the input data which is not really the case in stock market datasets.

6 Conclusion

As expected LSTM Network exhibited the best performance being a neural network it levaraged its ability to capture non-linear information in the index data. Interestingly Meta's prophet couldn't perform as expected but this is believed due to modelling limitations rather than it's ability. SARIMA performed the second best with the help of considering seasonality of the data, emphasizing the importance of relatively simpler statistical methods. Finally, ARIMA models performed the worst as expected because of their inherent limitations that only rely on regression.

7 References

[1] Pham Hoang Vuong, Lam Hung Phu, T. Hong, Le Nhat Duy, Pham The Bao, and Tan Dat Trinh, "A bibliometric literature review of stock price forecasting: From statistical model to deep learning approach," Science progress, vol. 107, no. 1, Jan. 2024, doi: https://doi.org/10.1177/00368504241236557.
[2] Modelling logic was adapted from https://www.kaggle.com/kushal1506.

A Appendix A: Code

Differencing function used in ARIMA and SARIMA models:

```
def difference(dataset, interval=1): #inputs required: time-series data and time period (default = 1)
    diff = list() #initializing a list
    for i in range(interval, len(dataset)):
        value = dataset[i] - dataset[i-interval] #differencing based on requested interval value
        diff.append(value)
    return np.array(diff)
```

Listing 1: Differencing function

LSTM Network architecture defintion

```
#creating model architecture
model = Sequential()
model.add(LSTM(128,return_sequences=True,input_shape=(None,1)))
model.add(LSTM(64,return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
```

Listing 2: Network architecture

B Appendix B: Additional Figures

Design Methodology Diagram

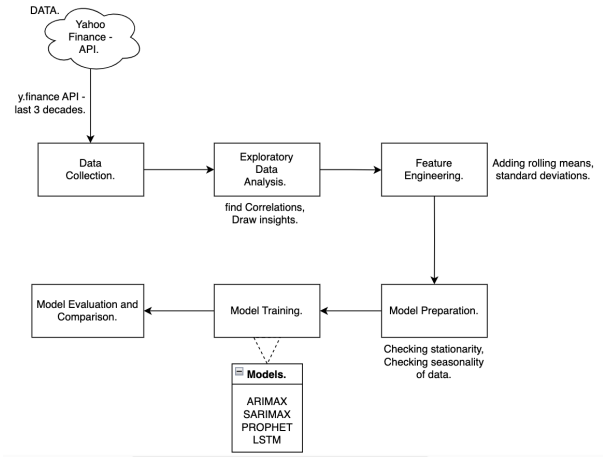

Figure 8: Design Methodology Workflow.

Exploratory Data Analysis (EDA)

Exploratory analysis was performed to understand the frequency of occurrence of daily index closing values.

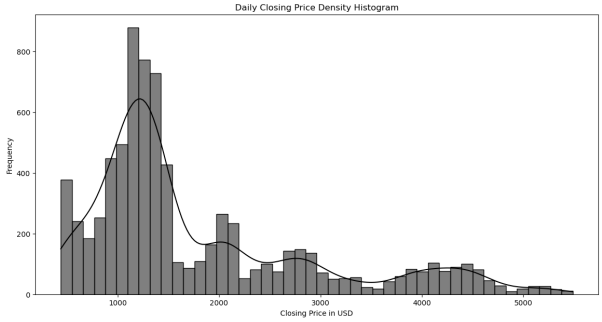

Figure 9: Density histogram of daily closing price

A comparison of the index trends with the top and bottom five percent of the participating organizations is performed.

Figure 10: S&P Index comparison with top five percent participating organizations

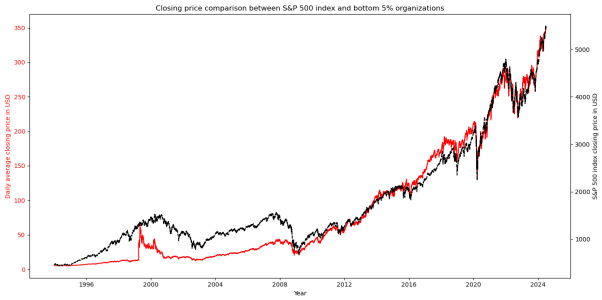

Figure 11: S&P Index comparison with bottom five percent participating organizations