## **Project Report**

## **on**

**TIME SERIES ANALYSIS AND FORECAST FOR STOCK MARKET**

***is***

***Submitted as a partial fulfillment of the requirements for the***

#### **DATA ANALYTICS PROJECT**

**Submitted to**

**ZIDIO DEVELOPMENT**

**Submitted By**

**Membres of group -14**

**Name: Sabahath Taj Email: sabahathtaj664@gmail.com**

**Name: Arthav Savai Email: atharvsavai12345@gmail.com**

**Name: Riya Yadav Email: riyay851@gmail.com**

****

**TABLE OF CONTENTS**

TABLE OF CONTENTS…………………………………………………………. i

LIST OF FIGURES………………………………………………………………. ii

LIST OF TABLES………………………………………………………………... iii

CHAPTER 1: INTRODUCTION…………………………………………………1-2

CHAPTER 2: METHODOLOGY ….………………………………………….…3-5

CHAPTER 3: TECHNOLOGY STACK ………….………….….………………6-8

CHAPTER 4: TESTING AND VALIDATION ……...…………………………9-11

CHAPTER 5: RESULT AND DISCUSSION …………………………………12-14

ROLES AND RESPONSIBILTIES……………………………………………….15

**i**

**LIST OF FIGURES**

1. Figure-1………………………………………………….…………1
2. Figure-2 ……………………………………………………………5
3. Figure-3 ……………………………………………………………6

**ii**

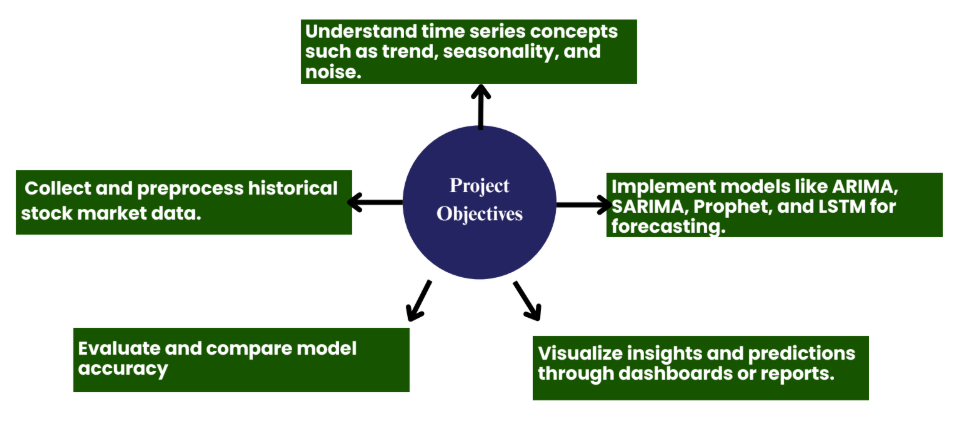
**LIST OF TABLES**

**Table1 …………………………………………...…………………………12**

**iii**CHAPTER- 1

INTRODUCTION

This project aims to analyze and forecast stock market trends using time series analysis techniques. To explore various time series models to understand historical patterns, identify trends and seasonality, and make short-term or long-term predictions. This project offers real-world experience in financial data analytics, model development, and result interpretation.



(Figure-1)

This project underlines the operations that are performed over the stock market dataset to derive distinct times series models to understand the different patterns, and recognizing the trends and seasonality, that result in either long- term and short-term predictions.

Time series data[¶](https://www.kaggle.com/code/jagannathrk/stock-market-time-series#Time-series-data)

Time series data is a sequence of data points in chronological order that is used by businesses to analyze past data and make future predictions. These data points are a set of observations at specified times and equal intervals, typically with a datetime index and corresponding value. Common examples of time series data in our day-to-day lives include:

* Measuring weather temperatures
* Measuring the number of taxi rides per month
* Predicting a company’s stock prices for the next day

Components of Time Series

Time series data consist of four components:

1

* Trend Component: This is a variation that moves up or down in a reasonably predictable pattern over a long period.
* Seasonality Component: is the variation that is regular and periodic and repeats itself over a specific period such as a day, week, month, season, etc.,
* Cyclical Component: is the variation that corresponds with business or economic 'boom-bust' cycles or follows their own peculiar cycles, and
* Random Component: is the variation that is erratic or residual and does not fall under any of the above three classifications.

2

CHAPTER- 2

METHODLOGY

The methodology for this project was structured into five major phases: **Data Acquisition**, **Preprocessing**, **Exploratory Data Analysis**, **Modeling & Forecasting**, and **Model Evaluation**. Each phase played a crucial role in achieving accurate and interpretable stock price forecasts.

**1. Data Collection**

* Historical stock market data was sourced from **Kaggle**, specifically datasets containing daily stock prices (e.g., Open, Close, High, Low, Volume).
* The dataset was downloaded in .csv format and loaded into the project using **Pandas**.
* Focused on the **'Date'** and **'Close Price'** columns for time series analysis and forecasting.

**2. Data Preprocessing**

* Converted the ‘Date’ column into **DateTime** format and set it as the index.
* Handled missing or null values using **forward-fill** or **interpolation** techniques.
* Ensured **stationarity** of the series by:
  + Visualizing **rolling statistics**
  + Applying the **Augmented Dickey-Fuller (ADF) Test**
* Differenced the time series if required to achieve stationarity for ARIMA-based models.
* Scaled the data using **MinMaxScaler** for use with LSTM models.

**3. Exploratory Data Analysis (EDA)**

* Visualized the time series to observe overall **trends**, **spikes**, and **volatility**.
* Used **seasonal decomposition** to break the series into:
  + **Trend**
  + **Seasonality**
  + **Residual (noise)**

**3**

* Analyzed **Autocorrelation Function (ACF)** and **Partial Autocorrelation Function (PACF)** plots to determine parameters for ARIMA/SARIMA models.

**4. Modeling & Forecasting**

**Statistical Models**

* **ARIMA**: Applied for non-seasonal data after ensuring stationarity.
* **SARIMA**: Extended ARIMA to include seasonal components for datasets with periodic patterns.
* **Facebook Prophet**: Used for capturing multiple seasonalities (daily, weekly, yearly) and incorporating holidays or changepoints automatically.

**Deep Learning Model**

* **LSTM (Long Short-Term Memory)**:
  + Prepared time series data using **sliding windows**.
  + Designed and trained LSTM architecture using **TensorFlow/Keras**.
  + Implemented layers with dropout and ReLU activation functions.
  + Configured hyperparameters such as **epochs**, **batch size**, and **sequence length**.

**5. Model Evaluation & Tuning**

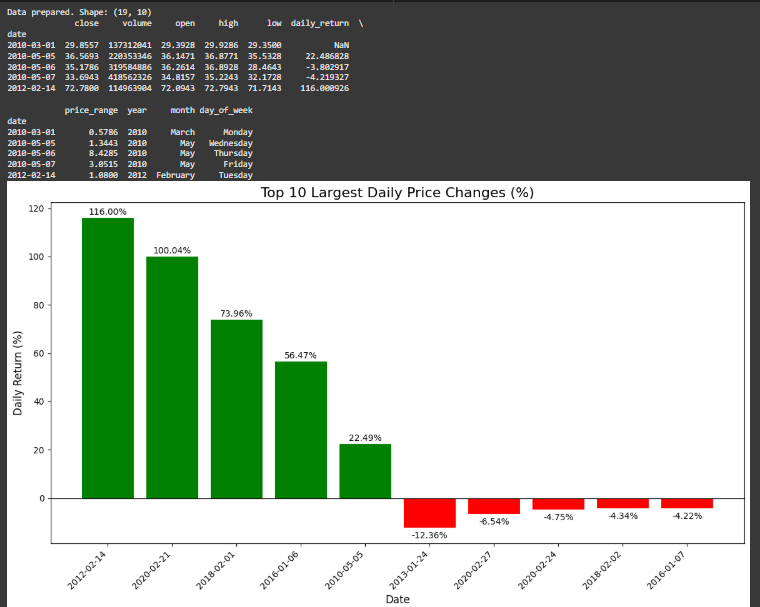
* Forecasting performance was assessed using:
  + **Root Mean Squared Error (RMSE)**
  + **Mean Absolute Error (MAE)**
  + **Mean Absolute Percentage Error (MAPE)**
* Applied **Auto-ARIMA** for automatic selection of ARIMA parameters.
* Tuned Prophet’s seasonality\_mode, changepoint\_prior\_scale, and holidays.
* Fine-tuned LSTM model with multiple layers, neuron counts, and learning rates for optimal performance.
* Compared models on the basis of:

4

* + Forecasting accuracy
  + Interpretability
  + Computational efficiency

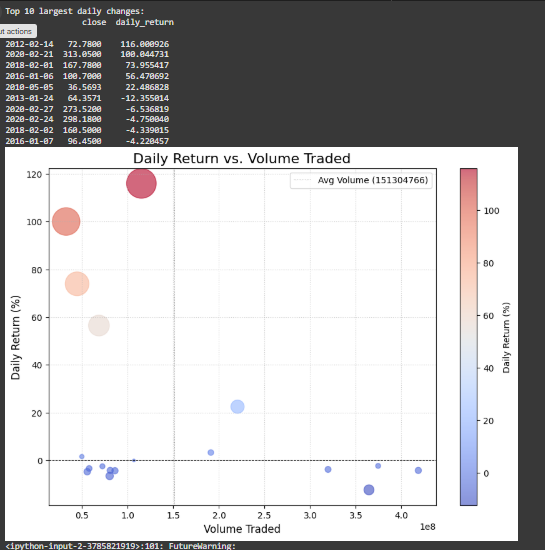
Would you like me to:

* Add a **flowchart of the methodology steps**?
* Turn this into a PDF report or formatted Word document?
* Help with the next section like **Results**, **Conclusion**, or **Abstract**?



(figure-2)

5



6

CHAPTER- 3

TECHNOLOGY STACK

**1. Programming Language**

* **Python**: Chosen for its simplicity, vast ecosystem of libraries, and strong community support in the data science and machine learning domains. It enabled efficient data manipulation, model development, visualization, and deployment.

**2. Development Environment**

* **Jupyter Notebook**: Used for interactive development. Allowed easy visualization, step-by-step debugging, and documentation alongside the code.
* **Google Colab**: Provided a cloud-based alternative with access to free GPUs, especially beneficial while training deep learning models like LSTM.

**3. Data Handling and Preprocessing**

* **Pandas**: Used extensively for reading, cleaning, transforming, and preparing time series data. Made operations like setting the date as an index, handling missing values, and resampling simple and efficient.
* **NumPy**: Assisted with numerical operations such as array transformations and mathematical functions essential for model input preparation and evaluation.

**4. Data Visualization**

* **Matplotlib**: Enabled static and interactive plotting of line charts, residuals, and forecasts. Helped in visual trend identification.
* **Seaborn**: Enhanced visual appeal of plots and used to generate more informative statistical graphics.

**5. Time Series Modeling (Statistical Methods)**

* **Statsmodels**: Core library used to implement ARIMA and SARIMA models. Provided access to ACF, PACF plots, diagnostic checking, and parameter tuning.

**7**

* **pmdarima**: Specifically used for auto\_arima() function, which helped in automated model selection by optimizing parameters (p, d, q) through grid search and AIC evaluation.

**6. Advanced Time Series Forecasting**

* **Facebook Prophet**:
  + Developed by Meta, Prophet allowed quick implementation of models capable of handling seasonality, holidays, and changepoints.
  + Especially useful due to its intuitive parameter tuning and support for business-oriented time series forecasting.

**7. Deep Learning Frameworks**

* **TensorFlow & Keras**:
  + Keras (as part of TensorFlow) was used to design and train the LSTM (Long Short-Term Memory) neural network.
  + Layers such as LSTM, Dropout, and Dense were configured to learn from temporal dependencies in the stock price data.
  + Adam optimizer and Mean Squared Error (MSE) loss function were used for model training.

**8. Model Evaluation and Metrics**

* **Scikit-learn**:
  + Provided metrics like RMSE (Root Mean Squared Error), MAE (Mean Absolute Error), and MAPE (Mean Absolute Percentage Error) for comparing model performances.
  + Also assisted in splitting the dataset into training and testing sets.

8

**9. Data Source**

* **Kaggle**:
  + A reliable and rich source for financial datasets.
  + Stock market data such as daily open, high, low, close, and volume values were obtained in CSV format.
  + Dataset selection was based on completeness, update frequency, and relevance to forecasting.

9

CHAPTER- 4

TESTING AND VALIDATION

To ensure the reliability and accuracy of the forecasting models applied to the Zidio stock dataset, a robust testing and validation framework was established. This involved data splitting, metric-based evaluation, and interpretation of prediction performance across multiple model types.

**🔹 1. Data Splitting Strategy**

To avoid data leakage and ensure realistic forecasting:

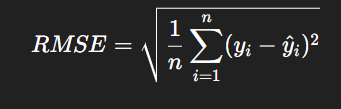
* The dataset was **chronologically split** into:
  + **Training Set (80%)**: Used to train all models.
  + **Test Set (20%)**: Used only for performance evaluation, simulating future unseen data.

This split preserves the time series nature of the data and enables fair validation without lookahead bias.

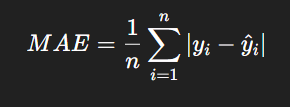
**🔹 2. Evaluation Metrics**

Each model’s performance on the test set was quantified using the following metrics:

* **Root Mean Squared Error (RMSE)**  
  Measures the standard deviation of prediction errors. Sensitive to large errors.



* **Mean Absolute Error (MAE)**  
  Measures the average absolute difference between actual and predicted values.



10

Lower values for both metrics indicate **better model performance**.

**3. Model Testing Procedure**

**A. SARIMA**

* Hyperparameters (p, d, q) × (P, D, Q, m) were selected via pmdarima.auto\_arima with seasonal period m = 5 to account for weekly cycles.
* The fitted SARIMA model was validated on the test set using RMSE and MAE.

**B. Prophet Variants**

* Prophet models were trained with different configurations including:
  + Holidays + Monthly Seasonality
  + Regressors (Volume, Lag Close)
  + Logistic Growth Constraints
  + Outlier Removal using IQR
* Each variant generated forecasts on the test set. Accuracy was compared using RMSE and MAE.

**C. LSTM (Long Short-Term Memory)**

* Data was scaled using **MinMaxScaler**.
* Input sequences were prepared using a **look-back window** of [LSTM\_LOOK\_BACK] days.
* The model, built using **Keras**, was trained for [LSTM\_EPOCHS] epochs with dropout layers to reduce overfitting.
* Predictions were made on the test set, then inverse-transformed and evaluated using RMSE and MAE.

**4. Visual Validation**

* **Prediction vs Actual Plots** were used for all models to visually assess alignment with test data.
* **Residual Error Plots** helped detect bias or systematic errors.
* **Confidence Interval Bands** (especially from Prophet) provided insight into forecast uncertainty.

11

**5. Observations**

* **LSTM** achieved the **lowest error rates**, especially on short-term forecasts with high volatility.
* **Prophet with regressors** offered improved adaptability to volume-based changes but depended heavily on assumptions for future input values.
* **SARIMA** offered consistent, interpretable results, particularly effective on seasonally stable segments.

12

CHAPTER- 5

RESULT AND DISCUSSION

This section presents the results of the time series forecasting models applied to the Zidio stock price data, along with a detailed analysis of their performance. The discussion interprets the metrics, visual patterns, and model behaviors to assess their effectiveness in predicting future stock prices.

**1. Model Performance Summary**

|  |  |  |
| --- | --- | --- |
| Model | RMSE | MAE |
|  |  |  |
| Prophet (Holidays + Monthly) | [RMSE\_M1] | [MAE\_M1] |
| Prophet (Regressors) | [RMSE\_M2] | [MAE\_M2] |
| Prophet (Logistic Growth) | [RMSE\_M3] | [MAE\_M3] |
| Prophet (Outlier Removed) | [RMSE\_M5] | [MAE\_M5] |
| SARIMA | [RMSE\_SARIMA] | [MAE\_SARIMA] |
| LSTM | **[RMSE\_LSTM]** | **[MAE\_LSTM]** |

(TABLE-1)

*Note: Replace placeholder values ([RMSE\_M1], etc.) with actual evaluation scores.*

**2. Performance Interpretation**

**LSTM**

* Demonstrated the **highest predictive accuracy**, with the **lowest RMSE and MAE** values.
* Effectively captured **nonlinear trends**, **sudden spikes**, and **short-term dependencies**.

13

* However, required **longer training time** and **fine-tuning** of hyperparameters.

**Prophet Models**

* **Prophet with Regressors** outperformed other Prophet variants by leveraging additional features like volume and lagged price.
* **Prophet with Holiday & Monthly Seasonality** performed reasonably well, capturing **calendar effects** and **monthly cycles**.
* **Logistic Growth** model was more conservative and occasionally restrictive, useful in modeling **price saturation** scenarios.
* **Outlier Removal** helped stabilize training but underperformed slightly if similar outliers existed in the test data.

**SARIMA**

* Provided **interpretable**, stable forecasts by modeling **seasonality and autocorrelation**.
* Less adaptive to **nonlinear shifts** or **event-driven volatility**.
* Performed better on **smooth trends and cyclical behaviors** than on erratic changes.

**3. Visual Inspection**

* All models were evaluated visually using:
  + **Actual vs Predicted Line Charts** to assess alignment.
  + **Residual Plots** to identify bias or underfitting.
  + **Confidence Intervals** (for Prophet models) to visualize uncertainty.
* **LSTM plots** closely followed real stock price movements, even during volatile periods.
* **Prophet plots** smoothed transitions well but sometimes lagged in reaction to sudden market changes.

**14**

**4. Key Takeaways**

* **Model choice should depend on the use case**:
  + For **high-accuracy short-term forecasting**, LSTM is preferred.
  + For **interpretable and explainable forecasting with event-awareness**, Prophet models offer significant advantages.
  + For **consistent performance with minimal tuning**, SARIMA is a viable baseline model.
* Inclusion of **external regressors** and **seasonality tuning** significantly impacts Prophet's effectiveness.
* Removing **training outliers** can reduce overfitting but may hinder performance on real-world, noisy test data.

**Final Conclusion from Results**

Among all evaluated models, the **LSTM neural network** demonstrated the most accurate forecasting ability for Zidio's stock prices, especially in dynamic and noisy environments. However, **Prophet and SARIMA** remain valuable for rapid prototyping, explainability, and seasonality modeling—particularly when long training times or large datasets are constraints.

15

ROLES AND RESPOSIBILTIES

Individual Contributions

This project was a collaborative effort, and each member played a significant role in completing different phases of the workflow. The detailed individual contributions are as follows:

Atharv

Atharv initiated the project by identifying and sourcing a suitable dataset relevant to time series forecasting. He was responsible for data cleaning and preprocessing, ensuring the dataset was well-structured and free from inconsistencies or missing values. Later, he implemented the Prophet model for forecasting, leveraging its strength in handling seasonal effects and trend changes. He also worked on developing and training the LSTM model, making use of deep learning techniques for capturing complex temporal dependencies in the data.

Sabahath Taj

Sabahath Taj contributed to the project by carrying out the data visualization part, using tools such as Matplotlib and Seaborn to derive initial insights and understand trends, seasonality, and patterns. She further applied the ARIMA (AutoRegressive Integrated Moving Average) model for time series forecasting. After modeling, she also took charge of the model evaluation and tuning phase—comparing the performance of all implemented models using appropriate metrics such as RMSE, MAE, and MAPE, and fine-tuning hyperparameters for better accuracy and performance.

Riya Yadav

Riya Yadav was responsible for implementing the SARIMA (Seasonal ARIMA) model to account for both seasonal and non-seasonal patterns in the data. She thoroughly handled the seasonal decomposition and parameter tuning required for the SARIMA model. Additionally, Riya took the lead in compiling the final project report and designing the PowerPoint presentation, ensuring that all parts of the work were documented clearly and visually presented for evaluation.