**Time Series Forecasting for Hotel Bookings using Streamlit**

Akash Anand T, Arjun Sreekumar, Ayantika Chatterjee, Sonia Thomas, Taranjeet Kamboj

**Abstract - This paper presents an end-to-end forecasting solution using time series analysis on hotel booking data. The solution includes decomposition of temporal components, model-based forecasting, and deployment of an interactive web application using Streamlit. The performance of ARIMA, ETS, Prophet, and LSTM models is compared using standard metrics (RMSE, MAE, MAPE, MSE). The app enables user uploads, interactive model selection, and visualization.**

**Keywords— time series, forecasting, hotel bookings, Streamlit, ARIMA, ETS, Prophet, LSTM.**

# I. INTRODUCTION

Time series forecasting is a vital technique for predicting future values based on historical data and is widely applied in domains such as finance, retail, healthcare, and hospitality. In the hotel industry, accurate forecasting of bookings can enhance inventory management, staffing, and revenue optimization. This project focuses on forecasting monthly hotel bookings using classical statistical models and modern machine learning approaches.

We apply and compare four forecasting models: ARIMA, ETS, Prophet, and LSTM. Each model is evaluated based on its accuracy and ability to capture trend and seasonality in the data. To enhance interpretability, we use time series decomposition to separate components such as trend, seasonality, and residuals.

To make the solution interactive and user-friendly, a web application was developed using Streamlit. The app allows users to upload their own time series data, apply different forecasting models, visualize results, and interpret model performance through metrics and plots.

# II. DATASET AND EDA INSIGHTS

The dataset used for this project consists of historical records of hotel bookings, with each entry indexed by arrival\_date. The original data was aggregated to a monthly frequency, resulting in a time series of total\_bookingsper month. This aggregation allowed for effective trend and seasonal pattern analysis while reducing noise from daily fluctuations. The resulting series spans multiple years, making it highly suitable for time series forecasting due to its consistent granularity and length.

An initial review of the dataset revealed a well-structured temporal index, although certain preprocessing steps were necessary to ensure data quality and continuity. Specifically, there were instances of missing values and duplicate date entries. These issues were addressed through a combination of forward-fill imputation and group-by aggregation to ensure that each month was represented exactly once. These transformations produced a complete and evenly spaced time series, which is essential for decomposition and model stability.

**Exploratory Data Analysis (EDA)** was conducted to uncover insights about the data’s temporal behavior. A key observation was the presence of strong seasonal fluctuations, with consistent peaks occurring in the summer months, particularly in June and July. This seasonal spike is expected in the hospitality sector due to increased vacation-related travel during warmer months. Conversely, the data displayed recurring dips in January and February, aligning with post-holiday slowdowns in demand.

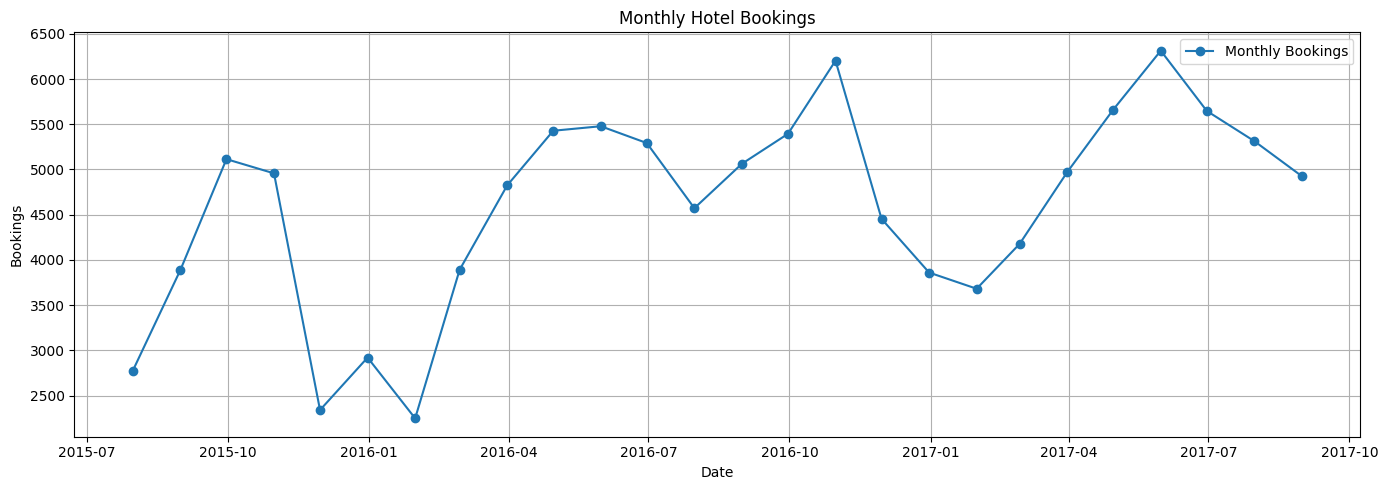


Fig. 1. Monthly Hotel Booking Trend (2015-17)

Overall, the dataset proved to be rich in patterns, minimally noisy, and well-structured after preprocessing. It was suitable for both **classical forecasting models** like ARIMA and ETS, and **advanced neural approaches** such as LSTM, enabling a comprehensive comparison across methodologies.

# III. TIME SERIES DECOMPOSITION

To better understand the structure of the hotel bookings dataset, both additive and multiplicative decompositions were performed using the seasonal\_decompose() function from Statsmodels, with a fixed periodicity of 12 to reflect monthly seasonality. These decompositions allowed us to isolate and observe the temporal behavior of trend, seasonal effects, and irregular variations over time.

**In the additive decomposition**, the trend component showed a smooth and steady rise in hotel bookings across the time frame, indicating a gradual growth in customer demand. The seasonal component revealed consistent booking spikes in the middle of each year, particularly in June and July, suggesting strong summer travel patterns. Lower values were observed around January and February, aligning with typical off-peak periods in the hospitality sector. The residual component exhibited mild volatility, especially in the early part of the dataset where some irregularities and missing values were originally present. Overall, the additive model captured the dataset’s general structure well, providing a clear breakdown of time-based influences.

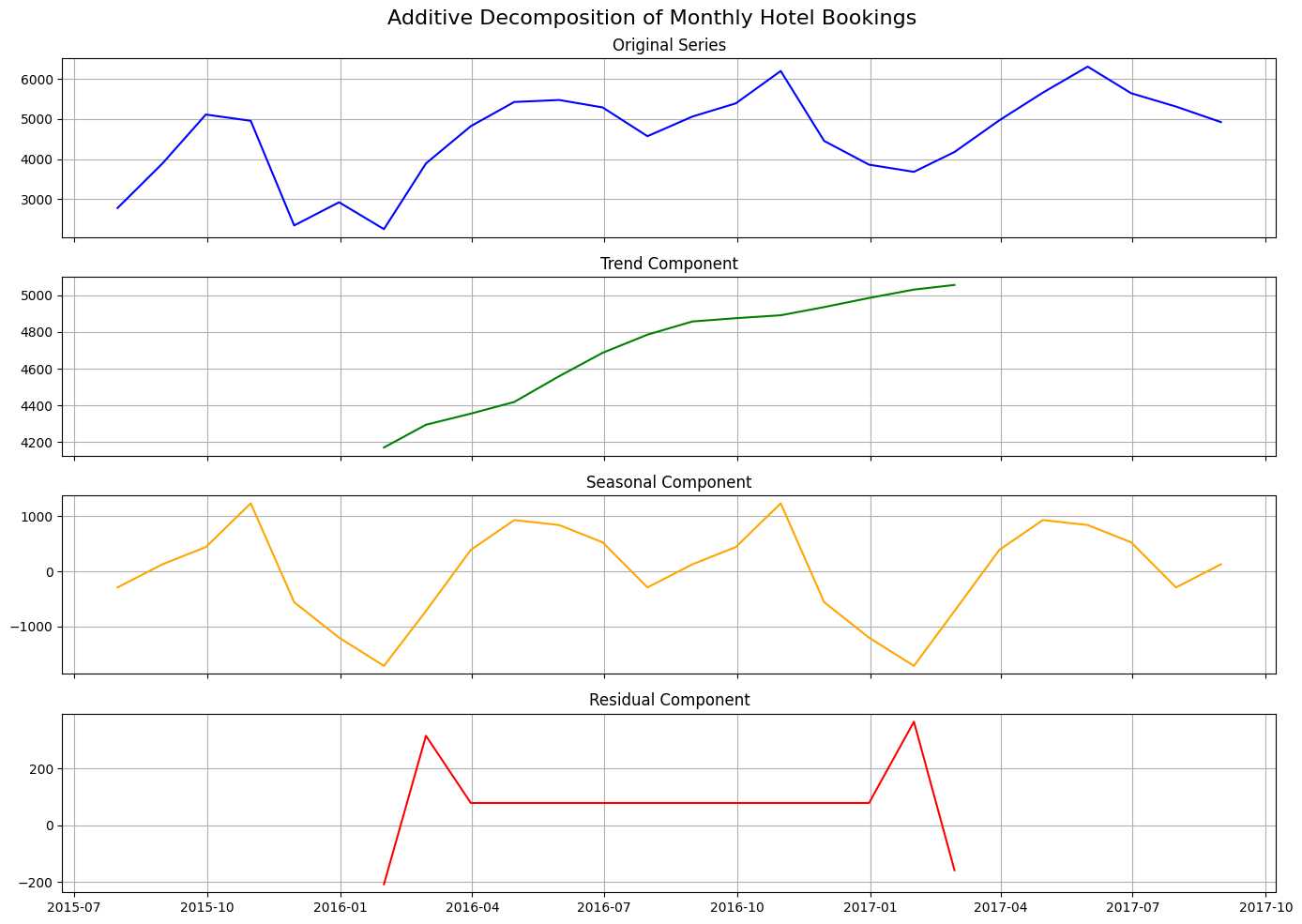


Fig. 2. Additive Decomposition of Monthly Hotel Bookings

**In the multiplicative decomposition**, similar patterns were observed but with amplified seasonal fluctuations. The peaks in summer months appeared more prominent due to the interaction between the increasing trend and the proportional nature of multiplicative seasonality. This decomposition emphasized how seasonal effects intensified as total bookings increased over time. The residuals in this model were more normalized and consistent, especially after the series was shifted. The multiplicative decomposition provides stronger contrast between high- and low-demand periods and is particularly useful when seasonality scales with the overall volume, which is characteristic of this dataset.

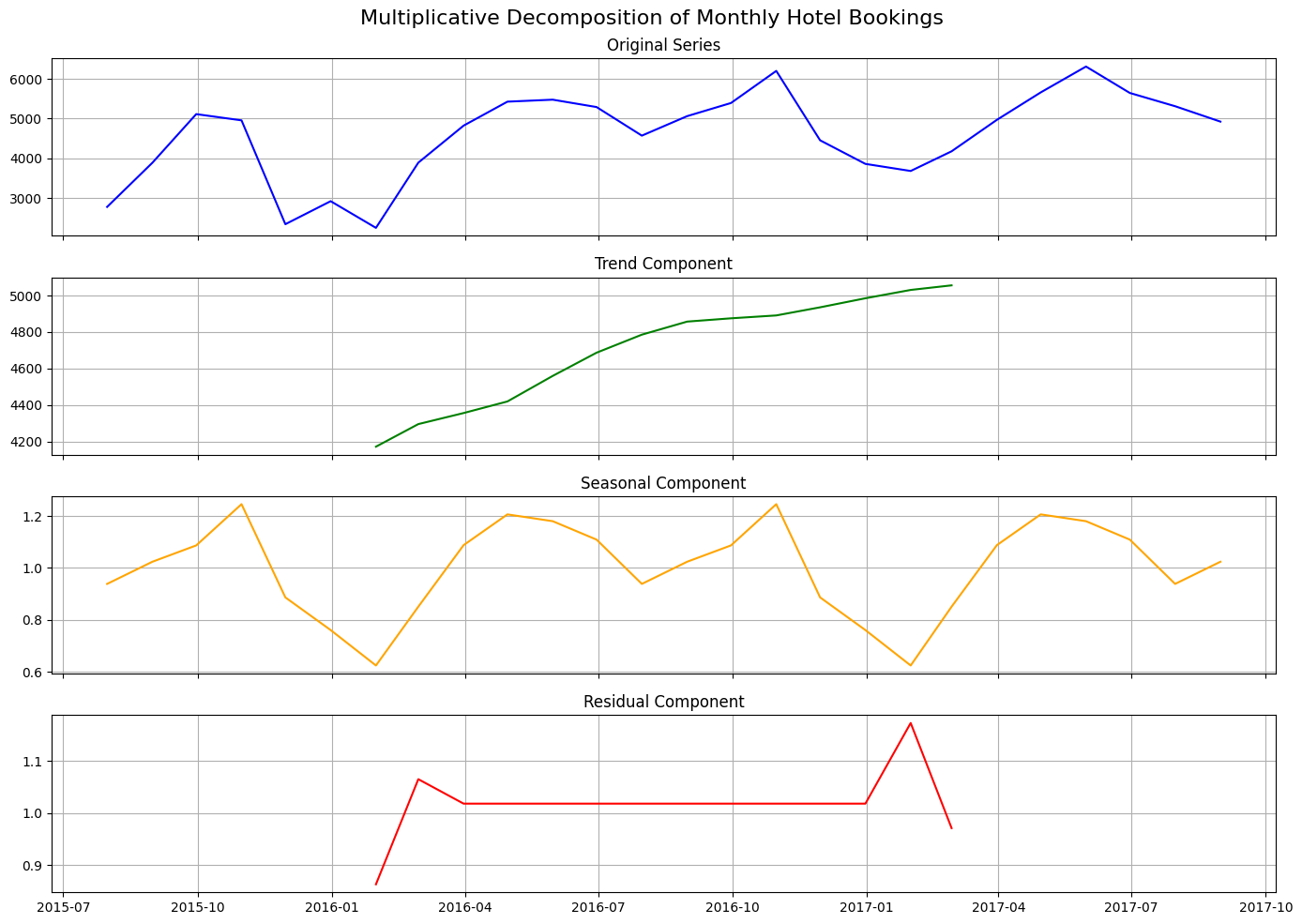


Fig. 3. Multiplicative Decomposition of Monthly Hotel Bookings

Both decomposition types offered valuable insights. While additive decomposition gave a clearer baseline view of seasonal stability, multiplicative decomposition highlighted the growing amplitude of seasonality across the timeline. The combination of both views guided the design and selection of forecasting models, ensuring that trend and seasonality were accurately captured and represented in the final model predictions.

# IV. FORECASTING MODEL COMPARISON

To assess the predictive performance of various time series forecasting techniques on the hotel bookings dataset, four models were implemented and evaluated: ARIMA, ETS, Prophet, and LSTM. Each model represents a distinct approach to modeling temporal patterns, ranging from classical statistical methods to advanced machine learning.

The dataset was split using an 80/20 ratio, with 80% of the monthly data used for training and the remaining 20% reserved for testing. Forecast accuracy was assessed using four commonly used performance metrics: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Mean Squared Error (MSE).

Table I. Model Performance Summary

| **Model** | **RMSE** | **MAE** | **MAPE (%)** | **MSE** |
| --- | --- | --- | --- | --- |
| ARIMA | 1475.78 | 1390.05 | 24.85% | 2.18M |
| ETS | 1436.55 | 1123.49 | 19.73% | 2.06M |
| Prophet | 584.72 | 445.61 | 7.73% | 2.06M |
| LSTM | 937.97 | 774.65 | 13.49% | 0.88M |

## A. Model Descriptions and Evaluation

**ARIMA (AutoRegressive Integrated Moving Average)** is a classical statistical model that combines autoregressive terms, differencing, and moving averages to capture linear relationships in stationary time series. While it handled the underlying trend reasonably well, ARIMA underperformed compared to newer models due to its limited ability to capture non-linear or seasonal variation without manual tuning.

**ETS (Error-Trend-Seasonality)** modeling incorporates smoothing techniques for level, trend, and seasonal components. In this case, we implemented an additive version with a seasonal period of six months. ETS slightly outperformed ARIMA in all metrics, particularly in terms of MAE and MAPE. However, it still struggled to capture some of the larger seasonal peaks and troughs seen in the dataset.

**Prophet**, developed by Facebook, provided the most accurate results across all metrics. It is designed to handle missing data, irregular time intervals, and strong seasonal effects. Prophet’s built-in ability to model changepoints and weekly/monthly seasonality made it particularly effective for our dataset, which exhibits clear mid-year demand surges. With the lowest RMSE (584.72), MAPE (7.73%), and MAE, Prophet is the most suitable model for **short-term, high-precision forecasting** in this case.

**LSTM (Long Short-Term Memory)** is a type of recurrent neural network capable of learning long-term dependencies and non-linear temporal patterns. While LSTM did not outperform Prophet in short-term forecasting, it significantly outperformed both ARIMA and ETS. LSTM was particularly effective at modeling more complex and dynamic fluctuations, making it a strong candidate for **long-range forecasting** scenarios or datasets with more noise or irregular seasonality. Its ability to learn from lagged sequences without requiring extensive feature engineering makes it highly flexible for time series modeling.

From the analysis, **Prophet** demonstrated the best performance on this dataset, especially in terms of precision and robustness to seasonal variation. **LSTM** followed as the second-best model, with good adaptability to nonlinear patterns. While **ARIMA** and **ETS** remain valid options, they were less accurate overall due to the dataset's seasonality and trend dynamics.

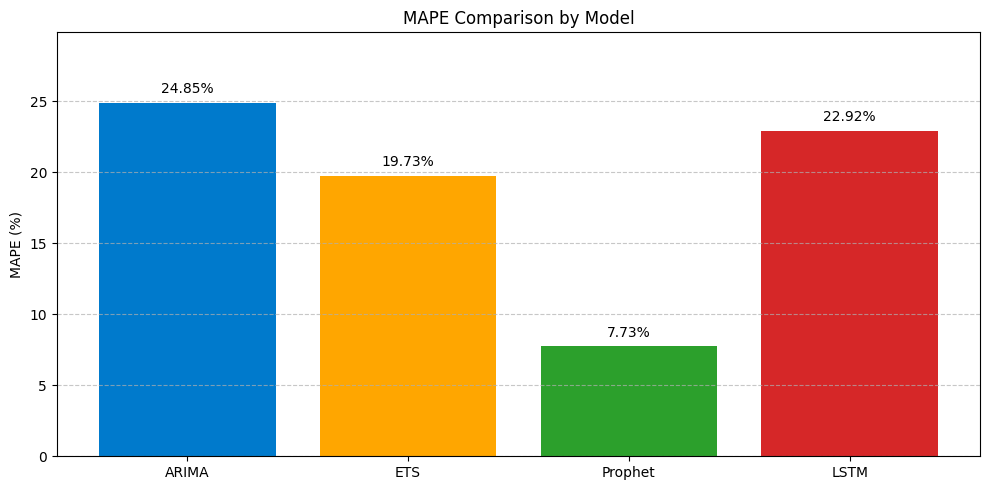


Fig. 4. MAPE (%) Comparison by Model

## V. STREAMLIT APP AND DEPLOYMENT

To facilitate user interaction, reproducibility, and ease of experimentation, a fully functional web application was developed using the Streamlit framework. The app serves as an end-to-end solution for time series forecasting, allowing users to upload their own dataset, explore seasonal patterns through decomposition, select forecasting models, and visualize performance metrics—all within an intuitive browser-based interface.

## A. Application Features

The web app was designed with user flexibility in mind. It accepts time series datasets in .csv format and guides the user through a streamlined workflow. Upon file upload, the application displays a preview of the dataset and prompts the user to select two key columns: the date/time column and the value column to be forecasted.

Once columns are selected, the app automatically sets the time index and handles missing timestamps by applying forward-fill imputation. Duplicate date entries are resolved using aggregation, ensuring the time series is formatted with unique and regular intervals—an essential requirement for decomposition and modeling.

A core feature of the application is the ability to perform time series decomposition. Users can choose between Additive or Multiplicative decomposition based on the nature of their data. The decomposition is visually presented in a four-panel plot showing the original series, trend, seasonal component, and residuals. In the case of multiplicative decomposition, the application automatically detects zero or negative values and applies a corrective shift to the series to ensure compatibility with the model. This built-in error-handling mechanism prevents application crashes and maintains forecast reliability.

Following decomposition, users can proceed to forecasting model selection. The app supports four forecasting models:

* ARIMA, a traditional statistical model suited for linear and stationary time series.
* ETS, which captures level, trend, and seasonality using exponential smoothing.
* Prophet, a decomposable model capable of handling complex seasonality, changepoints, and missing data.
* LSTM, a deep learning model well-suited for nonlinear and long-range temporal patterns.

Once a model is selected, users define the forecast horizon by adjusting a slider for the number of future periods (in months). The app then automatically splits the dataset into training and testing portions using an 80/20 rule, trains the selected model, generates forecasts, and overlays them on the actual data for comparison. Forecasting accuracy is assessed using four metrics: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Mean Squared Error (MSE). These metrics are displayed with proper formatting for ease of interpretation.

A key strength of the app is its modular handling of model-specific requirements. For example, in the LSTMimplementation, the app automatically scales the input data using MinMax normalization, reshapes sequences into lag-based training windows, and reconstructs the original scale of predictions for evaluation. This allows neural forecasting without requiring the user to manage the underlying architecture or transformations.

## B. Additional Capabilities

In addition to model forecasting and metric evaluation, the app allows users to download a cleaned version of their dataset after preprocessing. This is useful for exporting time-aligned data for use in other analytical tools or reports. Future versions may include options to download the forecasted values and decomposition components as well.

The application also incorporates minor enhancements to improve usability, including:

* Automatic detection and pre-selection of common column names like arrival\_date and total\_bookings.
* Conditional formatting and fallback messages when certain components are missing.
* In-app warnings and instructions for multiplicative model constraints.

## C. Deployment via Streamlit Cloud

The app was deployed publicly using Streamlit Cloud, which offers seamless integration with GitHub and rapid cloud deployment. After pushing the project code and dependencies to a public GitHub repository, the app was launched using Streamlit’s deployment interface. This eliminated the need for complex infrastructure such as virtual machines or web servers.

To ensure full reproducibility, a requirements.txt file was included in the repository. This file listed all the necessary libraries including streamlit, statsmodels, prophet, tensorflow, matplotlib, scikit-learn, pandas, and numpy. Streamlit Cloud uses this file to build the application environment automatically.

Upon deployment, the app was assigned a custom public URL, which can be shared with peers, instructors, or clients. The deployed version remains synchronized with the GitHub repository, so any code updates automatically reflect in the live app.

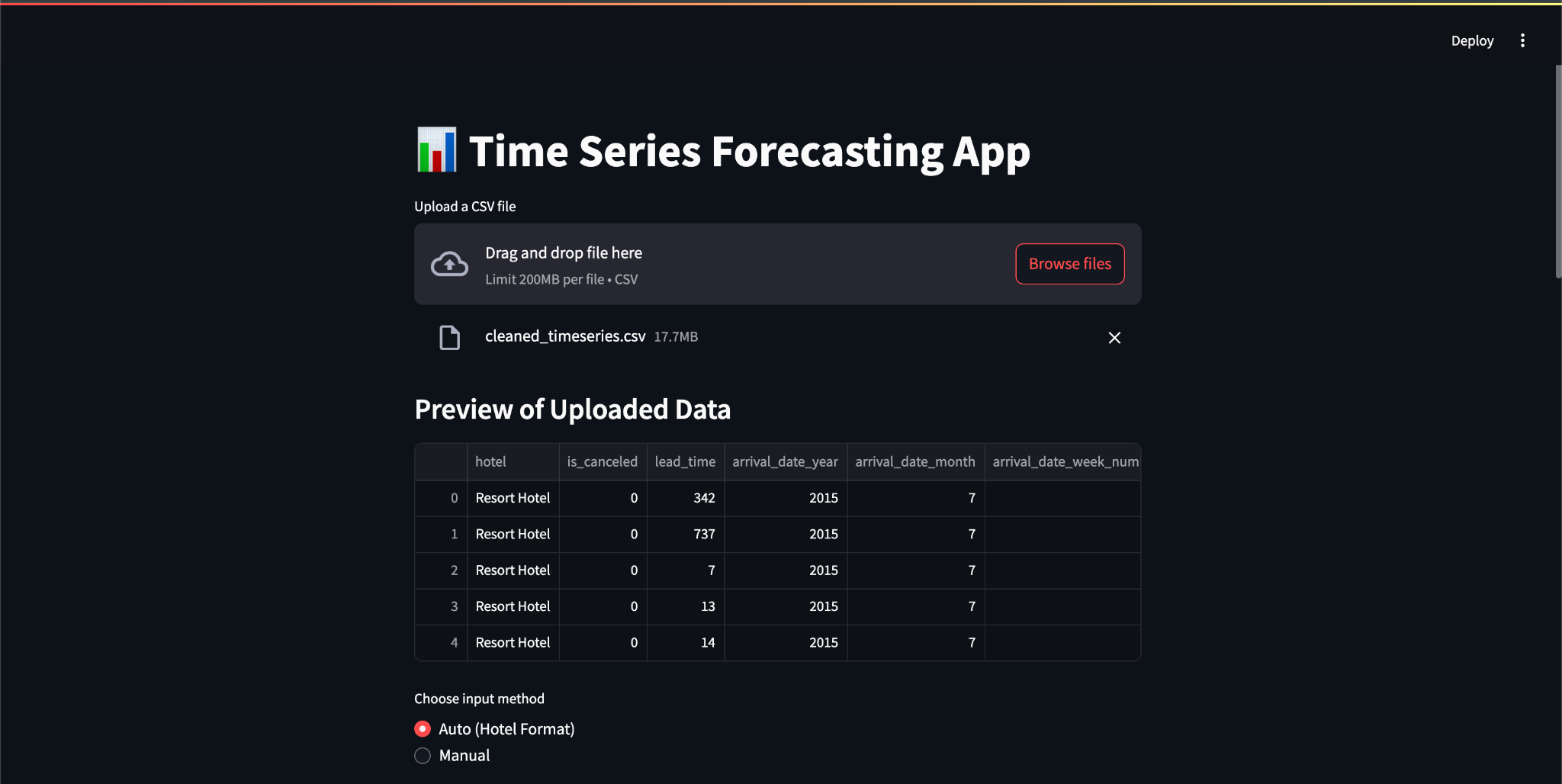


Fig. 5. CSV Upload and Data Preview Interface

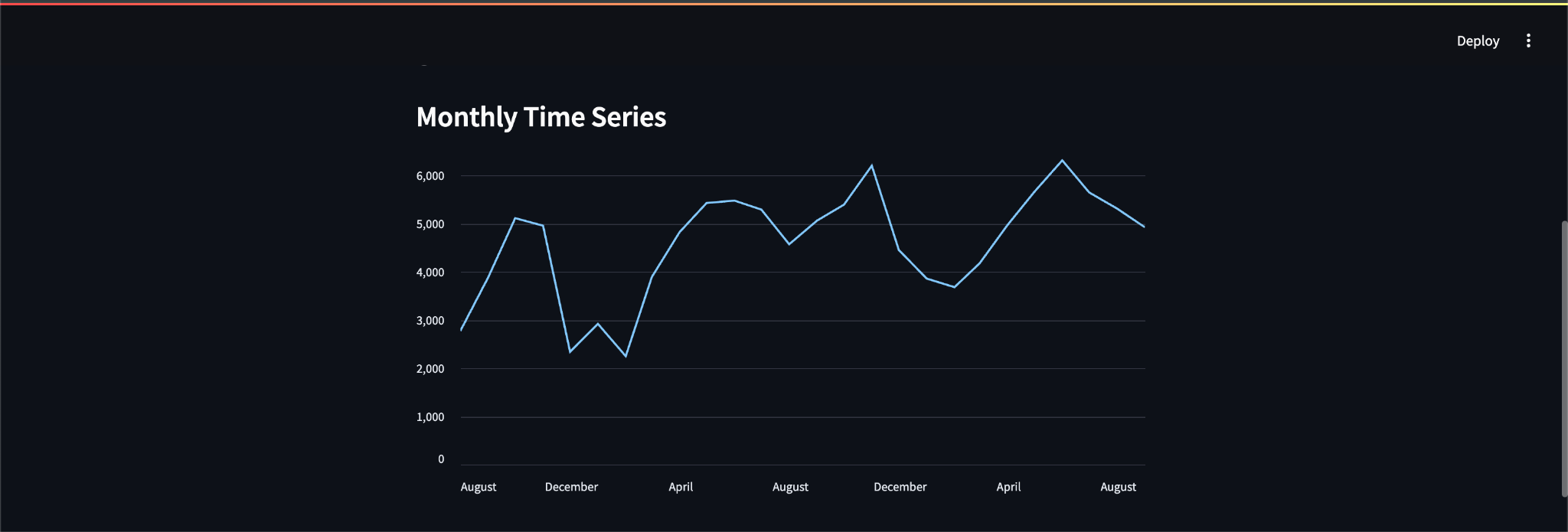


Fig. 6. Time Series Visualization

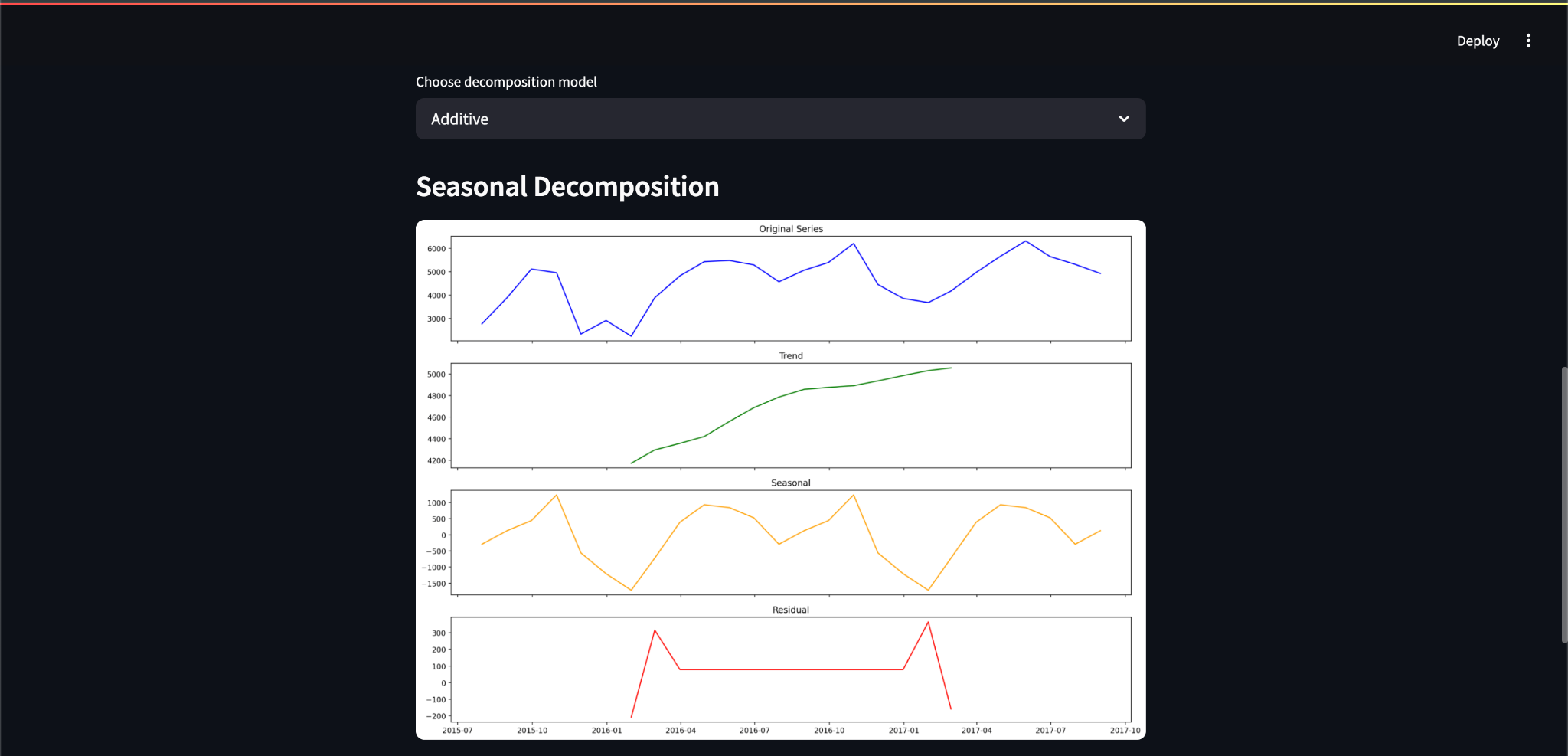


Fig. 7. Time Series Decomposition Plot (Additive and Multiplicative)



Fig. 8. Model Forecast and Evaluation Metrics

# VI. CONCLUSION

This project successfully implemented a comprehensive time series forecasting pipeline for hotel bookings using both classical statistical models and modern deep learning techniques. Through decomposition, we identified clear seasonal patterns and an upward trend in booking volumes, guiding the selection of appropriate forecasting models. Among the models evaluated, Prophet delivered the highest accuracy for short-term forecasts, effectively capturing trend shifts and seasonal fluctuations. LSTM, while slightly less accurate in the short term, showed strong potential for modeling complex, long-range temporal dependencies. ARIMA and ETS provided baseline comparisons and performed reasonably on stable data. The integration of all models into a Streamlit-based web application enabled real-time interactivity, ease of use, and deployment accessibility. Users can upload datasets, apply decomposition, visualize forecasts, and compare models within a single interface. Overall, the solution is robust, modular, and generalizable for use in industries such as hospitality, finance, and healthcare where time-based decision-making is critical.

## REFERENCES

[1] J. Mostipak, “Hotel booking demand,” Kaggle, 2020. [Online]. Available: <https://www.kaggle.com/datasets/jessemostipak/hotel-booking-demand>

[2] Streamlit, “Streamlit: The fastest way to build and share data apps,” [Online]. Available: [https://streamlit.io](https://streamlit.io/)

[3] Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: principles and practice.

[4] Facebook Prophet. [Online]. Available: <https://facebook.github.io/prophet/>

[5] Brownlee, J. (2017). Deep Learning for Time Series Forecasting. Machine Learning Mastery.

*[6] IEEE, “Conference templates,” [Online]. Available:* [*https://www.ieee.org/conferences/publishing/templates.html*](https://www.ieee.org/conferences/publishing/templates.html)

[7] TensorFlow, <https://www.tensorflow.org/>