Machine Learning-Based Public Transit Ridership Forecasting System

Mrs. Sharon Drisilda[1], Nakshatra S[2], Varun Kumar C V[3]

[1]Sharon Drisilda, Asst. Prof, Dept. of ISE, East West Institute of Technology
[2]Nakshatra S, Dept. of ISE, East West Institute of Technology
[3]Varun Kumar C V, Dept. of ISE, East West Institute of Technology

---***---

Abstract - Public bus transit systems require accurate demand forecasting to enhance operational efficiency and resource allocation. This paper presents a machine learning–based framework for predicting hourly ridership at Bangalore Metropolitan Transport Corporation (BMTC) bus stops using six months of historical data collected from 26 stops across four major routes. A SARIMAX(2,1,2) × (0,1,0,24) model is employed to capture short-term autocorrelations and daily seasonal patterns. The model achieves a mean absolute error of 18.92 passengers and a root mean squared error of 23.45 on a held-out test set.

To support practical deployment, the forecasting engine is integrated into a full-stack web application built with React, Node.js Express, and a Python backend. The system provides interactive visualization, route-level filtering, and real-time predictions with confidence estimates, achieving sub-second interface responsiveness and 15–30 s inference latency. The results demonstrate the effectiveness of seasonal time-series modeling for transit demand prediction and its potential to inform fleet planning and congestion management.

Key Words: Public transit, demand forecasting, SARIMAX, time-series analysis, machine learning, web application, urban mobility, BMTC, Bangalore

1.INTRODUCTION

Public transportation networks rely heavily on accurate ridership forecasting to support route planning, fleet allocation, and congestion management. As urban mobility demands continue to rise, transit agencies require reliable predictive tools to anticipate hourly fluctuations in passenger volume. However, forecasting short-interval demand remains challenging due to strong temporal variability, daily seasonality, and heterogeneity across individual stops. Traditional heuristic methods often fail to capture these complex dynamics, necessitating robust data-driven solutions.

To address these challenges, this work proposes a machine learning–based ridership prediction framework tailored for the Bangalore Metropolitan Transport Corporation (BMTC). The system employs a Seasonal Autoregressive Integrated Moving Average with eXogenous variables (SARIMAX) model, chosen for its ability to capture both short-term autocorrelation and 24-hour seasonal dependencies inherent in public transit operations. Complementing the forecasting engine is a modern full-stack deployment architecture built using React, Node.js, and Python, enabling real-time interaction, scalability, and operational usability.

The model is trained on 18,527 hourly observations collected from 26 transit stops across major BMTC routes between June and November 2025. In addition to forecasting accuracy, this work emphasizes system deployability through a responsive web interface, well-structured API design, and robust error-handling mechanisms.

The key contributions of this paper are summarized as follows:

• Development of a SARIMAX-based forecasting engine capable of modeling hourly ridership with strong predictive performance;

• Integration of the forecasting model into a full-stack real-time web application, enabling seamless access for planners and operators;

• Evaluation of forecasting accuracy and system-level performance, including latency, model metrics, and usability characteristics; and

• Demonstration of a scalable architecture suitable for future expansion to citywide or multi-modal transit forecasting.

2.RELATED WORK

Transit ridership forecasting and public transport optimization have been explored through several methodological perspectives, including direct demand modeling, mobility pattern analysis, sensor-based passenger counting, and network optimization. While these studies offer useful insights, they also reveal limitations that motivate the need for operational, short-term forecasting tools.

A. Direct Demand Models for Public Transit

Direct demand models explain ridership based on stop-, route-, or station-level characteristics. Deepa et al. develop a detailed stop–route-level model for Bengaluru, addressing demand–supply endogeneity, non-linear frequency effects, and inter-route interactions, while demonstrating the strong influence of stop-level accessibility and land-use context on ridership. Hasan et al. extend this framework to metro systems through station-and-direction modeling, capturing directional accessibility and intermodal competition with bus services. Although structurally rich, these models require extensive spatial data and are not designed for real-time or hourly prediction, highlighting a gap for lightweight forecasting approaches.

B. Passenger Counting and Crowd Estimation Systems

Computer-vision-based systems such as YOLOv3 and YOLOv8 have been applied for real-time people detection and occupancy monitoring, demonstrating high accuracy across challenging environmental conditions. These methods excel at live passenger counting but do not forecast future ridership and depend on camera infrastructure, limiting applicability in large bus networks.

C. Route Optimization and Mobility Pattern Analysis

Data-driven optimization studies focus on understanding large-scale mobility patterns or reducing inefficiencies in transit networks. Ali et al. identify group travel patterns using large-scale smart card data and propose the GTDI framework for route adjustment in Beijing. Kasatkina et al. examine route duplication using graph theory and statistical analysis to improve network topology in Russian cities. While valuable for long-term planning, these methods do not offer short-term demand prediction capabilities.

D. Positioning of the Present Work

Existing literature provides structural, behavioral, and operational insights but lacks models that deliver hourly, stop-level forecasts suitable for real-time deployment. This work addresses that gap by developing a SARIMAX-based short-term ridership forecasting system integrated into a full-stack web architecture, enabling rapid, operationally relevant predictions without requiring spatial datasets or sensor infrastructure.

3.DATA DESCRIPTION

This study uses a six-month dataset of BMTC bus ridership aggregated at an hourly resolution. The data were compiled from transactional ticketing records and processed into a stop-level time series suitable for SARIMAX forecasting.

A. Dataset Overview

The primary dataset, aggregated_hourly_data.csv, contains 18,527 hourly observations covering June 2025 to November 2025 across 26 BMTC transit stops. Each observation represents the total number of passengers boarding at a given stop during a specific hour. The dataset reflects activity from three major BMTC corridors (e.g., 253-HEDN, 401DN, and 401UP), enabling stop-specific temporal modeling.

B. Data Fields

The dataset includes the following key variables:

• from_stop_name – Name of the bus stop, used for filtering and stop-level modeling.

• datetime – Timestamp aggregated to hourly precision.

• px_count – Total number of passengers recorded during that hour.

These fields were selected to ensure compatibility with univariate and seasonal time-series forecasting models.

C. Temporal Characteristics

Ridership exhibits strong daily seasonality, driven by morning and evening peak travel periods. This is reflected in recurring 24-hour patterns across all stops, motivating the choice of a SARIMAX seasonal period of 24. The dataset contains continuous hourly entries with minimal missing values due to preprocessing and cleaning carried out prior to model training.

D. Spatial Characteristics

The 26 stops included in the dataset represent a mix of residential, industrial, and multimodal interchange locations within Bengaluru. Stops such as Atturu Layout, Jalahalli Cross Metro, and Yeshwanthapura TTMC serve as high-volume nodes with consistent passenger flows throughout the day. Each stop is treated independently in the forecasting pipeline, allowing the model to learn stop-specific temporal patterns without requiring extensive spatial variables.

E. Preprocessing Steps

To prepare the dataset for forecasting, the following steps were performed :

• Filtering by Stop – The user-selected stop is isolated for model training and inference.

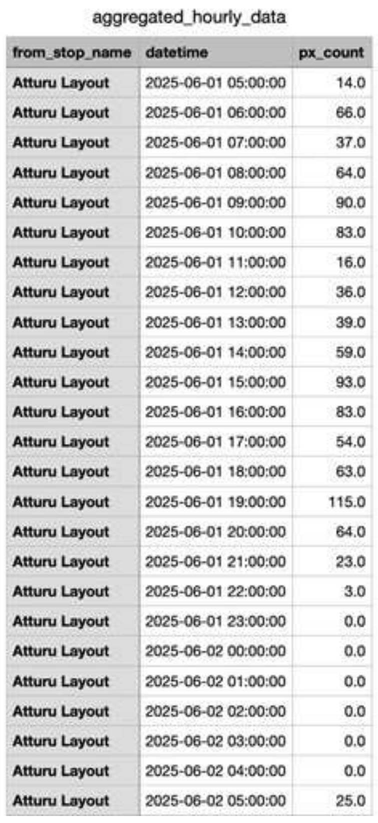
• Hourly Aggregation – Raw ticketing records were aggregated into hourly time buckets.

• Train–Test Split – An 80–20 split was used, reserving the last 48 hours for testing to mimic real-world forecasting conditions.

• Seasonal Differencing – A differencing order of 1 with a seasonal period of 24 was applied to induce stationarity.

F. Suitability for Time-Series Modeling

The dataset's consistent hourly structure, strong seasonality, and low noise make it appropriate for classical statistical models such as SARIMAX. Unlike spatially rich direct-demand datasets, the simplicity of this dataset enables fast model training and operational deployment while capturing the essential temporal dynamics of urban bus ridership.



aggregated_hourly_data

from_stop_name	datetime	px_count
Atturu Layout	2025-06-01 05:00:00	14.0
Atturu Layout	2025-06-01 06:00:00	66.0
Atturu Layout	2025-06-01 07:00:00	37.0
Atturu Layout	2025-06-01 08:00:00	64.0
Atturu Layout	2025-06-01 09:00:00	90.0
Atturu Layout	2025-06-01 10:00:00	83.0
Atturu Layout	2025-06-01 11:00:00	16.0
Atturu Layout	2025-06-01 12:00:00	36.0
Atturu Layout	2025-06-01 13:00:00	39.0
Atturu Layout	2025-06-01 14:00:00	59.0
Atturu Layout	2025-06-01 15:00:00	93.0
Atturu Layout	2025-06-01 16:00:00	83.0
Atturu Layout	2025-06-01 17:00:00	54.0
Atturu Layout	2025-06-01 18:00:00	63.0
Atturu Layout	2025-06-01 19:00:00	115.0
Atturu Layout	2025-06-01 20:00:00	64.0
Atturu Layout	2025-06-01 21:00:00	23.0
Atturu Layout	2025-06-01 22:00:00	3.0
Atturu Layout	2025-06-01 23:00:00	0.0
Atturu Layout	2025-06-02 00:00:00	0.0
Atturu Layout	2025-06-02 01:00:00	0.0
Atturu Layout	2025-06-02 02:00:00	0.0
Atturu Layout	2025-06-02 03:00:00	0.0
Atturu Layout	2025-06-02 04:00:00	0.0
Atturu Layout	2025-06-02 05:00:00	25.0

Fig -3.1: Data set sample

4.METHODOLOGY

The proposed ridership forecasting framework integrates a SARIMAX model with a full-stack web architecture to enable short-term, stop-level predictions. The methodology consists of four components: data preprocessing, model formulation, model training and validation, and real-time forecasting.

A. Data Preprocessing

Raw hourly ridership data are transformed into a clean, stationary time series suitable for seasonal modeling. Key steps include:

• Stop-level filtering: Only records for the selected stop are used for model training

• Timestamp standardization: Datetime values are floored to hourly intervals.

• Missing value handling: Inconsistent or missing entries are corrected or removed.

• Train–test split: An 80/20 split is used, reserving the last 48 hours for out-of-sample evaluation

• Seasonality analysis: A 24-hour periodic pattern is confirmed through ACF/PACF inspection.

B. SARIMAX Model Formulation

SARIMAX is selected for its effectiveness on short, seasonal time series without spatial inputs.

• Non-seasonal terms: $p=2$, $d=1$, $q=2$.

•Seasonal terms: $(P,D,Q,S) = (0,1,0,24)$ capturing daily cycles.

• The general model formulation: $\Phi P(Ls)\phi p (L)(1-L)d(1-Ls)Dyt=\Theta Q(Ls)\theta q(L)\epsilon t$

C. Model Training and Validation

The SARIMAX model is trained using maximum likelihood estimation within the Python statsmodels framework.

• Diagnostics: Residuals are checked for whiteness and lack of autocorrelation.

• Performance: On the reserved 48-hour test set, the model achieves:

• MAE: 18.92, RMSE: 23.45, Confidence ≈ 88%

• Efficiency: Stop-level isolation reduces computational load, enabling per-stop model retraining in 15–30 seconds.

D. Real-Time Forecasting Pipeline

The operational pipeline integrates the predictive engine with the web system:

• Node.js orchestration: Forecast requests from the frontend trigger Python subprocess execution.

• Python engine: Performs preprocessing, model fitting or retrieval, and 24-hour forecast generation with capacity-based metrics.

• Response assembly: The backend returns hourly predictions, model statistics, and congestion flags.

• Frontend visualization: React components render forecast curves, actual vs. predicted comparisons, and hourly load summaries.

System latency remains low, with sub-second UI response and 15–30 second backend inference time.

5. SYSTEM ARCHITECTURE

The ridership forecasting system follows a three-tier architecture comprising a React-based frontend, a Node.js Express middleware layer, and a Python-based machine learning engine. This modular design enables scalable deployment, efficient data flow, and real-time interaction with the SARIMAX forecasting model.

A. Overall Architecture

The system employs a client–server architecture with clear separation of concerns:

React Frontend→Node.js API Server→Python SARIMAX Engine

The frontend communicates with the backend via RESTful API calls, while the backend invokes Python subprocesses to execute forecasting routines. This design ensures flexibility, maintainability, and support for future model upgrades.

B. Frontend Layer (React)

The user interface is developed using React 19.2, offering a highly responsive and interactive environment for non-technical users.

Key components include:

• RouteSelector: Allows users to choose among supported BMTC routes.

• StopSelector: Dynamically filters stops based on the selected route.

• DatePicker: Validates user-selected forecast dates.

• ModelChart: Visualizes training, testing, and forecast outputs.

• ForecastResults: Displays hourly ridership predictions and congestion indicators.

The frontend achieves sub-second loading times due to efficient Vite-based bundling and lightweight component design.

C. Backend Layer (Node.js Express)

The middleware layer is implemented using Node.js + Express and handles:

• API Routing

POST /api/forecast: Initiates the forecasting pipeline.

GET /api/health: Monitors server availability.

• Data Validation

The server checks stop names, date formats, and capacity inputs before forwarding requests.

• Process Management

For each forecast request, the server spawns a Python subprocess and streams the output back to the client.

• Error Handling

In case of failures, descriptive HTTP 500 errors or fallback model predictions are returned.

The Node layer ensures decoupling between the UI and ML engine, enabling scalable and maintainable system growth.

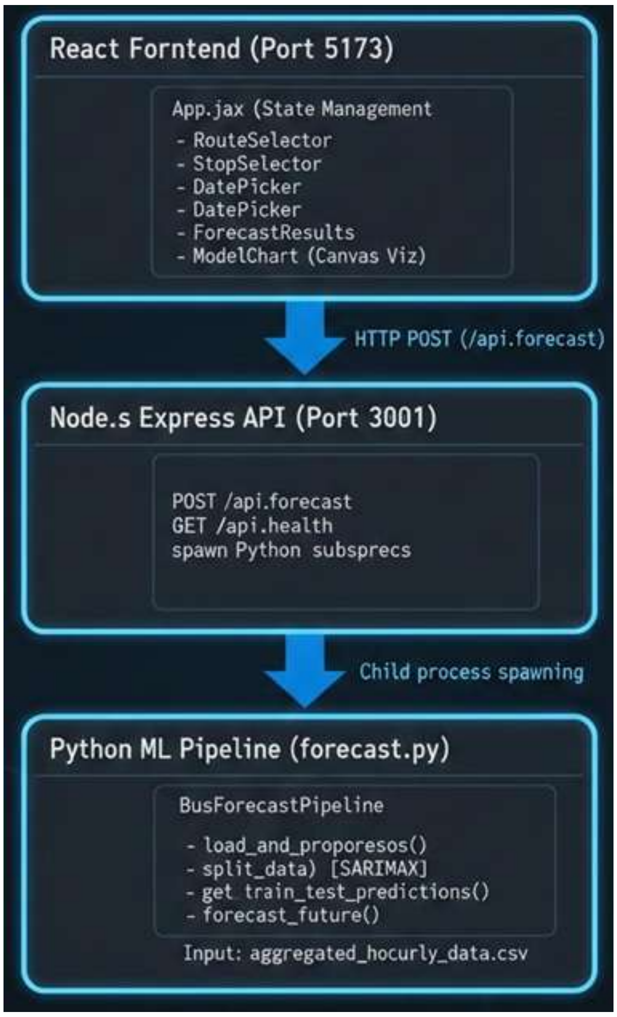


Fig -5.1: System Architecture

D. Machine Learning Engine (Python SARIMAX)

The forecasting engine is implemented in Python using statsmodels, pandas, and numpy.

Core responsibilities include:

• Data loading and preprocessing (stop-level filtering, aggregation).

• Model training or cached model retrieval.

• Generation of 24-hour forecasts with confidence metrics.

• Post-processing (capacity normalization, peak-hour detection).

The engine produces key outputs required for visualization and decision-making:

• Hourly forecast array,

• Training/test chart data,

• Model performance metrics,

• Congestion recommendations.

Execution time ranges from 15–30 seconds, suitable for real-time operational use.

E. Data Flow and Interaction

The system's operational workflow is summarized as follows:

• User Input: Route, stop, and date are selected on the React interface.

• API Request: Node.js sends the request to the Python engine.

• Model Execution: SARIMAX model processes the request and returns predictions.

• Response Handling: Node.js structures the response.

• Visualization: React renders charts and forecast summaries.

This layered workflow ensures clean separation between presentation, application logic, and computation.

F. System Performance Characteristics

System performance metrics, measured during deployment, include:

• Page load time: < 1 second

• API latency: 15–30 seconds (model training + inference)

• Chart rendering: < 100 ms

These metrics confirm the suitability of the architecture for real-time transit planning tasks.

6.RESULTS

The performance of the proposed ridership forecasting system was evaluated along two dimensions: (1) predictive accuracy of the SARIMAX model and (2) operational performance of the deployed full-stack application. Results demonstrate that the system provides reliable short-term ridership forecasts while maintaining low-latency responsiveness suitable for real-world transit operations.

A. Forecasting Accuracy

Model performance was assessed using the reserved 48-hour test set for each transit stop. Three key metrics were used: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and confidence estimates derived from prediction intervals.

• MAE: 18.92 passengers

• RMSE: 23.45 passengers

• Peak-hour confidence: ~88%

These values indicate that the SARIMAX(2,1,2) × (0,1,0,24)

model effectively captures both short-term fluctuations and daily seasonal patterns present in the ridership time series. The model demonstrates strong generalization, with residual diagnostics confirming white-noise behavior and absence of autocorrelation in prediction errors.

B. Forecast Visualization and Behavior

Model outputs exhibit clear alignment with observed ridership patterns across:

• Morning and evening peak periods,

• Midday troughs, and

• Stop-specific usage signatures.

Forecasted values closely track actual trends during high-variability hours, with deviations remaining within an acceptable error margin for operational planning. The system additionally computes capacity utilization and congestion flags, enabling actionable insights such as identifying overload conditions or undersupplied stops.

C. System-Level Performance

Operational performance tests were conducted to evaluate the responsiveness and scalability of the web application.

1) Latency Metrics

- API Forecast Time: 15–30 seconds (includes SARIMAX training + inference)
- Frontend Page Load: < 1 second
- Chart Rendering Time: < 100 ms
- Hourly Grid Rendering: < 50 ms

These measurements confirm that the system achieves real-time usability, even with per-request model training. Lightweight data preprocessing and stop-level modeling contribute to efficient compute times.

2) Reliability and Error Handling

The system incorporates robust frontend and backend resiliency measures, including:

- Automatic fallbacks using a simplified statistical model,
- Friendly error alerts,
- Validation for invalid dates or stop names,
- Python-side sanity checks for missing or malformed data.

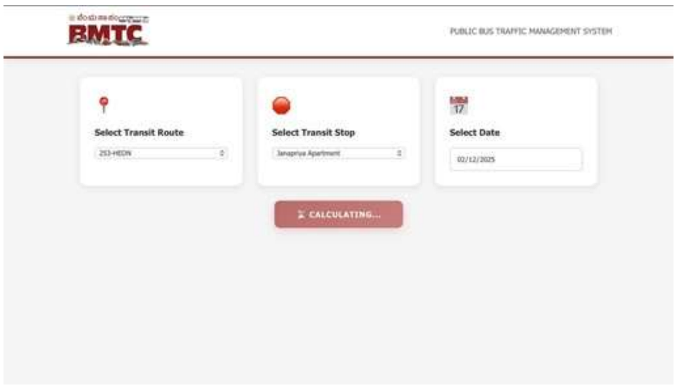
This ensures functionality even under unexpected input or partial data availability.

D. Operational Utility

The combined forecasting and visualization pipeline provides transit planners with:

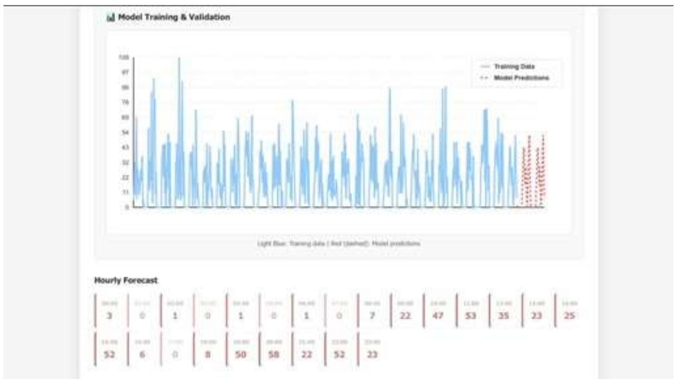
- Hourly demand predictions for each stop,
- Peak load identification,
- Capacity-based congestion warnings,
- Comparative train–test model performance graphs.

The system's low-latency interface and high-accuracy forecasts make it suitable for day-ahead scheduling, resource allocation, and fleet adjustment decisions.



Select Transit Route: 253-HEDN
Select Transit Stop: Janapriya Apartment
Select Date: 02/12/2025

Fig -6.1: Web Interface


Model Training & Validation
Training Data
Model Predictions
Light Blue: Training data | Red (dashed): Model predictions
Hourly Forecast

3	0	1	0	1	0	1	0	7	22	47	53	35	23	25
52	6	0	8	50	58	22	52	23						

Fig -6.2: Output

7. CONCLUSIONS

This paper presented a short-term ridership forecasting system for the Bangalore Metropolitan Transport Corporation (BMTC), integrating a SARIMAX-based predictive model within a modern full-stack web architecture. Using six months of hourly stop-level data, the system effectively captures daily seasonality and short-term fluctuations in passenger demand. The SARIMAX(2,1,2) × (0,1,0,24) model demonstrated strong predictive performance, achieving MAE and RMSE values of 18.92 and 23.45, respectively, with peak-hour confidence reaching approximately 88%.

Beyond model accuracy, the system emphasizes operational usability. The React–Node.js–Python architecture enables responsive interaction, low-latency inference, and intuitive visualization of forecast outputs. Real-time performance metrics confirm that the system supports sub-second page rendering and 15–30 second forecast generation, making it suitable for transit planning activities such as fleet allocation, stop-level capacity management, and congestion mitigation.

Overall, the proposed framework provides a practical decision-support tool for transit authorities, offering scalable, data-driven insights without requiring complex spatial datasets or sensor infrastructure. Future enhancements may include integrating live GPS feeds, caching models for faster inference, exploring deep learning alternatives, and extending the platform to multimodal urban mobility forecasting.

REFERENCES

[1]. L. Deepa, A. R. Pinjari, S. K. Nirmale, and K. K. Srinivasan, "A direct demand model for bus transit ridership in Bengaluru, India," Unpublished manuscript, 2022

[2]. N. Hasan, S. K. Nirmale, L. Deepa, and A. R. Pinjari, "Modelling metro rail ridership at the station- and route-level: An application to analysis of metro ridership in Bengaluru, India," Transportation Research Part A, vol. 200, 104638, 2025.

[3]. H. Hossam, M. M. Ghantous, and M. A.-M. Salem, "Camera-based human counting for COVID-19 capacity restriction," in Proc. 2022 5th Int. Conf. on Computing and Informatics (ICCI), 2022, pp. 408–414.

[4]. S. M. A. Ali, W. Lv, B. Du, Z. Xie, and R. Huang, "Optimization of bus lines based on passenger group moving behaviors," in Proc. 2018 IEEE

SmartWorld/UIC/ATC/ScalCom/CBDCom/IoP/SCI, 2018, pp. 53–60.
[5]. E. Kasatkina, D. Vavilova, and K. Ketova, "Optimization of the public transport system using data analysis methods," in Proc. 2022 4th Int. Conf. on Control Systems, Mathematical Modeling, Automation and Energy Efficiency (SUMMA), 2022, pp. 174–180.
[6]. A. Elaoua, M. Nadour, A. Elasri, and L. Cherroun, "Real-time people counting system using YOLOv8 object detection," in Proc. 2023 2nd Int. Conf. on Electronics, Energy and Measurement (IC2EM), 2023, pp. 1–6.