Traffic Flow Prediction Using PEMS-08 Dataset:

A Comparative Analysis of Time Series Models

Aung Chan Nyein1,2\* and Sandhya Avasthi2,3*†*

1\*College of Informatics and Computing Sciences – Graduate School, Batangas State University, Golden Country Homes, Alangilan, Batangas City, 4200, Batangas, Philippines.

2Department of CSE, ABES Engineering College, Crossings Republik, Ghaziabad, 201009, Uttar Pradesh, India.

\*Corresponding author(s). E-mail(s): 24-05039@g.batstate-u.edu.ph; Contributing authors: sandhya avasthi@yahoo.com;

*†*These authors contributed equally to this work.

Abstract

Traffic flow prediction is crucial for urban planning and management. This study explores predictive models using the PEMS-08 dataset from San Bernardino, California, recorded in July–August 2016. By leveraging occupancy as a proxy, the study ranks locations with the highest traffic levels and evaluates various time series forecasting techniques. The findings highlight the effectiveness of these models in accurately predicting traffic flow patterns, ultimately contributing to more efficient urban mobility solutions. The use of Long Short-Term Memory (LSTM) networks in traffic forecasting not only improves prediction accuracy but also creates new possibilities for smart city projects, where real-time data can greatly influence infrastructure planning and resource allocation. LSTM neural network significantly outperforms traditional models, including Kalman Filter, ARIMA, SARIMA, Exponential Smoothing, Moving Average, and Facebook Prophet. The LSTM achieves the lowest errors (MAE: 0.030, RMSE: 0.031, MAPE: 46 %), demonstrating its superior ability to capture complex temporal dependencies and non-linear patterns. This approach also shows potential for optimizing police patrols by identifying high-traffic locations at specific times, underscoring the value of deep learning methods in traffic management.

Keywords: LSTM, Time Series Forecasting, Traffic Flow Prediction, PEMS-08 Dataset

1 Introduction

Traffic flow prediction is crucial for strengthening the urbanization procedures of modern society, especially for urban and sub-urban regions. [1] It assists in traffic management, infrastructure planning, and optimization of traffic signals, which leads traffic authorities to make data-driven decisions with valuable insights from the forecasting model using time series data analysis. To state the complex spatial-temporal dependencies inherent in traffic data, several models have been developed to acquire valuable insights for effective management. [2] These models strengthen advanced machine learning techniques to improve the accuracy and adaptability of prediction to traffic scenarios. This study aims to enhance predictive accuracy by leveraging time series forecasting models, ranging from traditional statistical methods (e.g., ARIMA, SARIMA) to advanced machine learning approaches (e.g., LSTM networks). Challenges such as seasonal variability and abrupt changes in traffic dynamics underscore the importance of selecting appropriate models for specific scenarios. [3] The paper is organized as follows: Section 2 reviews related work, Section 3 details the methodology, Section 4 presents the results, and Section 5 discusses the implications and conclusions.

1.1 Time Series Forecasting Techniques

Time series data is a collection of random variables indexed in the order they are obtained in time. Analyzing time series data involves understanding its components, such as Trend: A long-term pattern or direction in the data that can be increasing or decreasing. Seasonality: Recurring patterns that occur at fixed intervals that can be daily, weekly, or yearly. Cycle: Fluctuations around the trend that are not fixed in duration, unlike seasonality. Irregularity: Random fluctuations that do not follow any detectable pattern. Time series decomposition aims to separate these components to understand and model the underlying patterns in the data. [4] [5] [6]

1.2 Stationary Time Series

A time series is considered strictly stationary when the properties remains unchanged over the period. This is to say that the joint probability distribution of any set of observations is the same as that of the observations altered by a constant time lag. A weak stationary time series has a constant mean and an autocovariance function that only depends on the time lag between observations, not their specific time points and it is usually referred to as stationary. Having a stationary series is important for time series analysis because it involves a level of regularity and predictability in the data. [6]

1.3 Non-stationary Time Series

The statistical properties, such as mean, variance, or covariance, that change over the time period are called non-stationary time series. A simple example is a random walk, where the value at each time point depends on the previous value plus a random shock. [6]

1.4 Time Series Analysis and Forecasting

Time series analysis includes examining time series data to extract meaningful insights in order to understand underlying patterns. Forecasting is a major part of time series analysis that aims to predict future values of the series based on historical data. Good forecasting depends on identifying and modeling the components of a time series, such as trend, seasonality, cycle, and irregularity. [5] [6]

1.5 Converting Non-stationary Time Series to Stationary

Most of the time series analysis techniques need stationary data. Many methods can be used to make non-stationary into stationary series. Differencing: Calculates the difference between consecutive observations or observations at seasonal lags. For example, differencing a random walk removes the dependence on the previous value and yields a stationary white noise process. Seasonal differencing: Calculates the difference between an observation and the matching observation from the previous season. Log transformation: Logarithmic transformation makes the data stabilize the variance of a series that exhibits non-constant variance. [3] [6]

2 Related Works

Several studies have applied statistical and machine learning models to traffic forecasting. [7]Time Series Forecasting Models such as ARIMA, SARIMA, Kalman Filters, Exponential Smoothing, GARCH, and Facebook Prophet have been used widely to predict traffic flow. However, recent advancements in deep learning, particularly LSTM networks, offer significant improvements in capturing temporal dependencies. [1] This study builds upon these advancements by focusing on traffic data from the PEMS08 dataset.

3 Literature Review

According to the paper, “Experimental Analysis of Time Series Models for Traffic Flow Prediction” by Nivedita Tiwari and Lalji Prasad, SARIMA had the best prediction accuracy and the lowest root mean squared error (RMSE) compared to LSTM, and ARIMA models performed to the traffic data. The authors examined related studies and pointed out the various uses of machine learning and deep learning in predicting traffic flow. They talked about earlier research that applied LSTM to predict traffic flow. The authors also noted studies on how rainfall affects traffic speed predictions using SVM. [8]

Regarding the paper by Guangyu Wang, Yujie Chen, Ming Gao, Z. Wu, Jiafu Tang, and Junbo Zhao, TITAN, a new mixed model of experts for predicting traffic flow, combines sequence-focused, variable-focused, and knowledge-based modeling techniques. The model uses cosine normalizing to add prior knowledge during the training process. Tests on the METR-LA and PEMS-BAY datasets demonstrate that TITAN surpasses other models with an average enhancement of 9% in all forecast periods. TITAN’s strength lies in its capability to effectively manage sudden changes in traffic flow, which is vital for making precise predictions. [9]

This paper presents a new model for predicting traffic flow using Fourier graph convolution. The authors add a feature embedding module to capture various time-related aspects. They merge Fourier graph convolution with gated residual learning to pull out spatial-temporal features. Tests on the PeMSD4 and PeMSD8 datasets show that the model effectively captures the spatial and temporal dependencies of traffic data, outperforming several benchmarks, including HA, ARIMA, GRU, STGCN, ASTGCN, STSGCN, AGCRN, and F-GCN. [10]

The study, “Temporal Graph Learning Recurrent Neural Network for Traffic Forecasting: by Sanghyun Lee and Chayoung Park, offers a quick summary of different deep learning models used for time series data, including Recurrent Neural Networks (RNNs). It highlights key RNN designs like Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTMs). This material also presents graph-based neural networks to add structural details to time series analysis.[11]

The authors of the research “Forecasting Day-Ahead Traffic Flow Using Functional Time Series Approach”, introduce a practical time series method for predicting traffic flow for the next day. The researchers suggest a modeling structure that combines the functional autoregressive (FAR) model with the seasonal autoregressive integrated moving average (SARIMA) model. They analyzed traffic flow data from a busy road in Dublin, Ireland, to test the effectiveness of their proposed method. The researchers discovered that the combined FAR(1)-SARIMA model provided the highest forecasting accuracy, surpassing the individual FAR(1) and SARIMA models.[12]

This paper introduces CALTM, a model that uses context-aware long-term memory to predict holiday traffic patterns. CALTM integrates scene and sequential context to overcome the challenges recurrent neural networks face in long-term predictions. The model incorporates a new plug-in module, S2M2, to find historical time series that are similar based on contextual features, such as holiday and weekend trends. Tests on a real-world ETC dataset demonstrate that CALTM significantly exceeds the performance of baseline methods like N-BEATS, LSTM, and SeasESOpt.[13]

“A New Perspective on Traffic Flow Prediction: A Graph Spatial-Temporal Network with Complex Network Information”, a study by Zhiqiu Hu, Fengjing Shao, and Rencheng Sun, presents a new graph spatial-temporal network (GSTNCNI) for predicting traffic flow that includes complex network information. The authors apply complex network theory to study the traffic road network structure and identify features like degree centrality, clustering coefficient, betweenness centrality, closeness centrality, point strength, and average shortest path length. These features are combined into a graph neural network to enhance the precision of traffic flow prediction. Tests on traffic flow data from California freeways show that GSTNCNI surpasses current graph convolutional neural network benchmarks, demonstrating strong accuracy and reliability.[14]

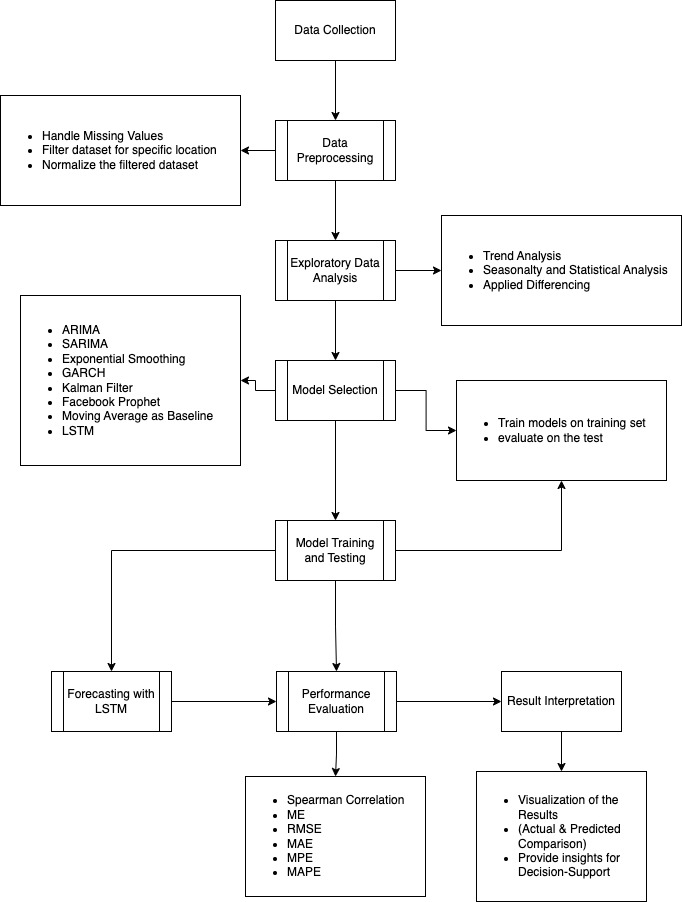
This study looks into using multi-head attention transformers to predict traffic flow. The researchers suggest a transformer model that uses the multi-head attention technique to understand long-term patterns in traffic data. The model is tested against LSTM and GRU models with traffic flow data from the California Performance Measurement System (PeMS). The findings indicate that the transformer model is more effective than both LSTM and GRU, showing notable gains in accuracy.[15]

This chapter of the book gives a detailed look at methods for predicting traffic flow, emphasizing large-scale spatiotemporal data. The writers examine a range of deep learning models, such as RNNs, CNNs, and graph convolutional networks (GCNs). They explore how these models can be applied in various situations, such as assessing safety for truck-related road hazards and analyzing the capacity of toll booths. The writers stress the significance of traffic flow prediction in improving safety and efficiency in transportation.[16]

This Master’s thesis investigates how to predict traffic speed, flow, and congestion with deep learning techniques. The author tested and analyzed long short-term memory (LSTM), gated recurrent unit (GRU), and hybrid CNN-LSTM models, comparing them to linear regression (LR). The research revealed that deep learning models were more effective than traditional linear regression, especially in understanding spatial and temporal patterns in traffic data. GRU, being simpler in design, outperformed LSTM and CNN-LSTM during longer training periods (six months), implying that extensive road network and time data might not be crucial for precise long-term forecasts. However, CNN-LSTM excelled with smaller datasets (three months), showing its capability to grasp complex spatial and temporal trends when data is scarce. The author utilized six and three months of traffic data from the California Department of Transportation (Caltrans) to train the models and assessed their performance using mean squared error (MSE) and mean absolute error (MAE).[17]

4 Methodology

This section will cover the steps and processes applied to gathering, aggregating, cleaning, analyzing the data, and managing the forecasting models and time series analysis methodologies used to perform predictions and evaluation. The following figure represents the process flow of this paper.



**Fig.1** Process Flow Diagram

4.1 Data Collection

The dataset, PEMS-08 (Performance Measurement System), was sourced from Kaggle and contains traffic data from San Bernardino recorded between July and August 2016. It includes flow, occupancy, and speed metrics captured every 5 minutes across 170 locations. It includes three key features: flow, occupancy, and speed, described as follows: Flow: The number of vehicles passing through a detector within a 5-minute interval, measured as vehicles per 5 minutes. Occupancy: The percentage of time within a 5-minute interval that a detector was occupied by a vehicle. Speed: The average speed of vehicles passing through the detector during a 5-minute interval, measured in miles per hour (mph). [17] [18]

4.2 Data Pre-processing

This paper explores an interesting question: What if a model could predict and rank locations with the highest traffic? Such a predictive model could help optimizing traffic control by identifying the best locations to monitor at specific times. To simplify the problem, the variable occupy is used as a proxy for estimating traffic levels, and itserves as the target variable for prediction. The problem is defined as follows: Given predictions for N time steps of the occupy variable across K locations, the goal is to develop a predictive model that minimizes the total error, expressed as:

where *“pt”* represents the “Spearman correlation coefficient” between the predicted and actual values at time step *“t”*.

The Spearman correlation coefficient measures the strength and direction of the relationship between two variables based on their rank order, rather than their abso lute values. This approach focuses on optimizing patrol locations by ranking traffic levels. The current strategy is to first minimize the Root Mean Square Error (RMSE) when predicting the actual values of the target variable, occupy. The ranks of these predictions will then be used for further optimization. To achieve this, an “LSTM (Long Short-Term Memory) neural network” will be trained, and its performance will be compared against a baseline created using the moving average of historical occupy values. The following steps have been applied to the dataset: [2][3][19][20][21]

• Handle missing values using mean imputation.

• Scale features using Min-Max normalization.

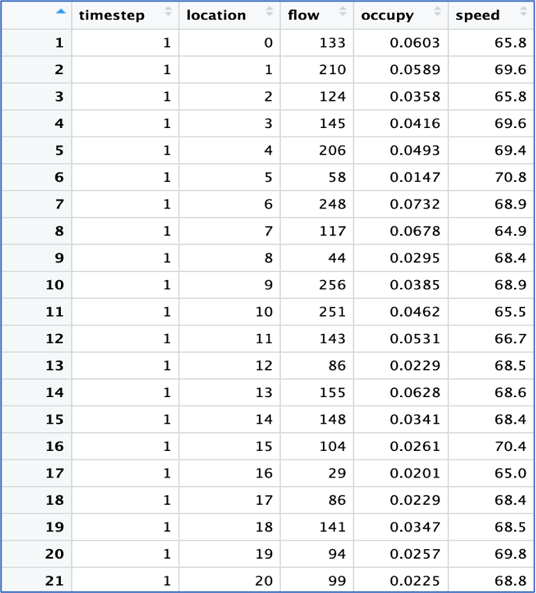
• Aggregate data into hourly intervals for better manageability.

4.3 Exploratory Data Analysis and Data Visualization

The traffic dataset was analyzed to gain insights into its structure and properties, particularly focusing on a specific location (location 50). The following steps and methods were applied:

4.4 Data Loading and Examination

The dataset was read into the R-Programming framework and inspected using functions like head(), summary(), and str() to understand its structure, types of variables, and basic statistics.

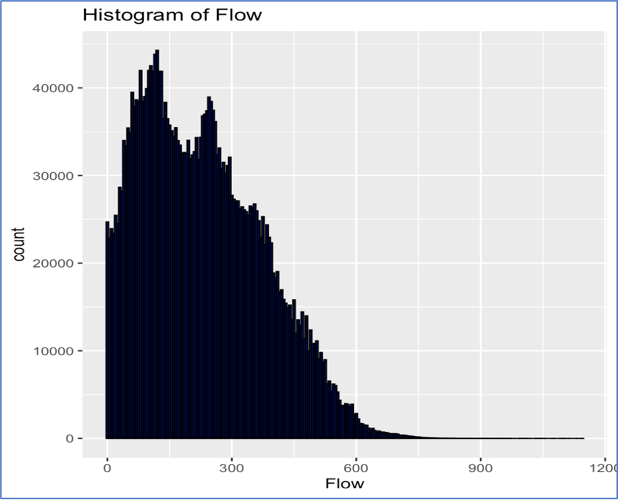


**Fig.2** Summary of Traffic Flow Dataset

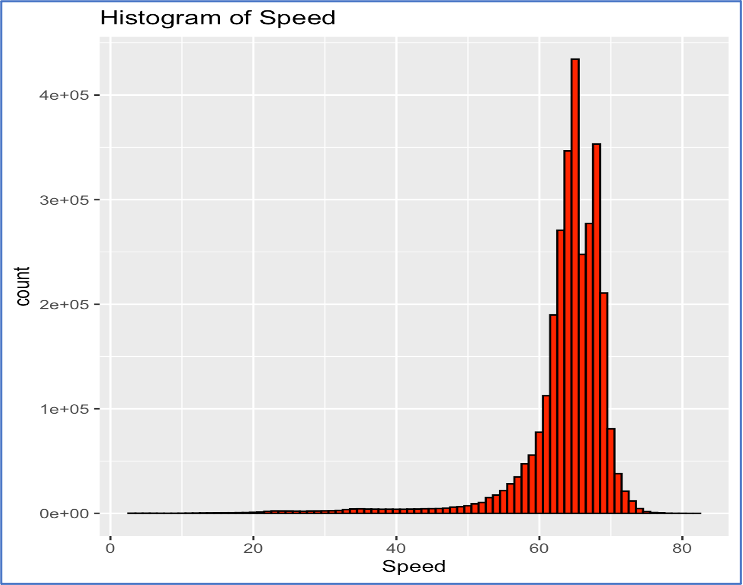
4.5 Exploratory Data Analysis - EDA

Histograms were created for three key variables: flow, speed, and occupy. These visu alizations provided an overview of the distribution of each variable. A subset of data for location 50 was extracted, and the first 1,000 rows of this filtered data were used for detailed analysis. Line graphs were plotted to observe trends over time for occupy, speed, and flow.

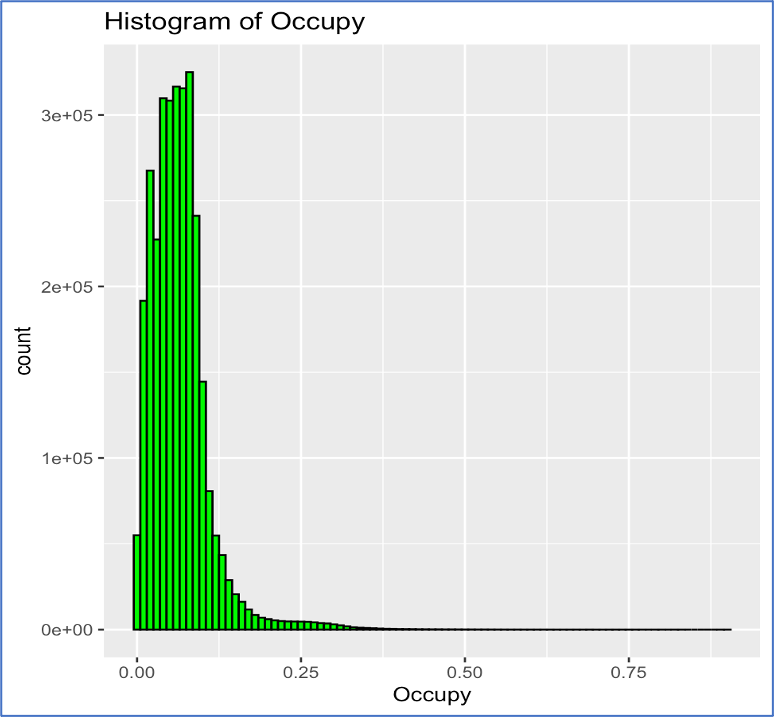
Regarding the data visualizations, the distributions of flow, occupy, and speed show that the values are spread out well and roughly follow a normal distribution, though they are slightly skewed. However, the values for these variables differ signifi cantly in scale and magnitude. This indicates the need for normalization and scaling before proceeding with further analysis to ensure consistency and comparability across the variables, which may involve techniques such as min-max scaling or z-score normalization to bring all variables onto a similar scale and improve the performance of subsequent analytical methods. [21][22]



**Fig.3** Histogram of Flow

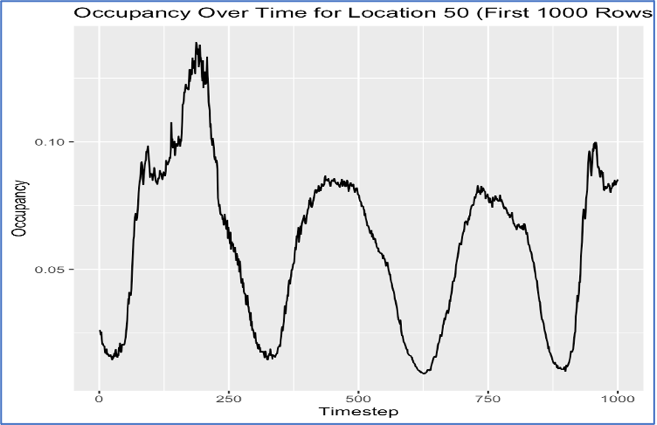


**Fig.4** Histogram of Speed

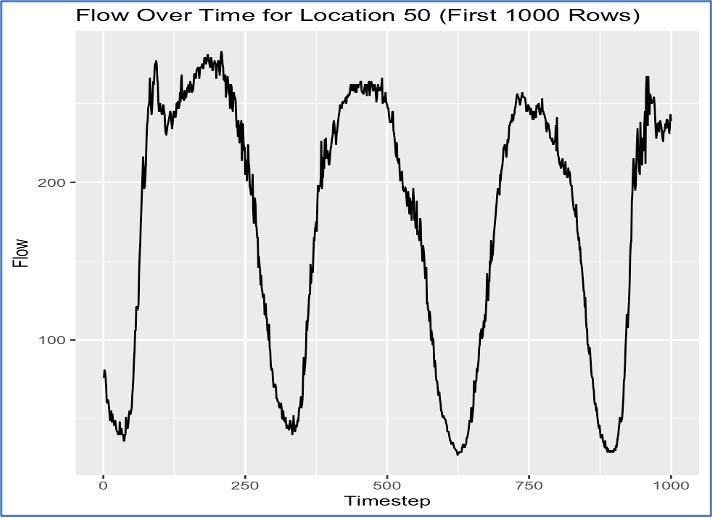


**Fig.5** Histogram of Occupy

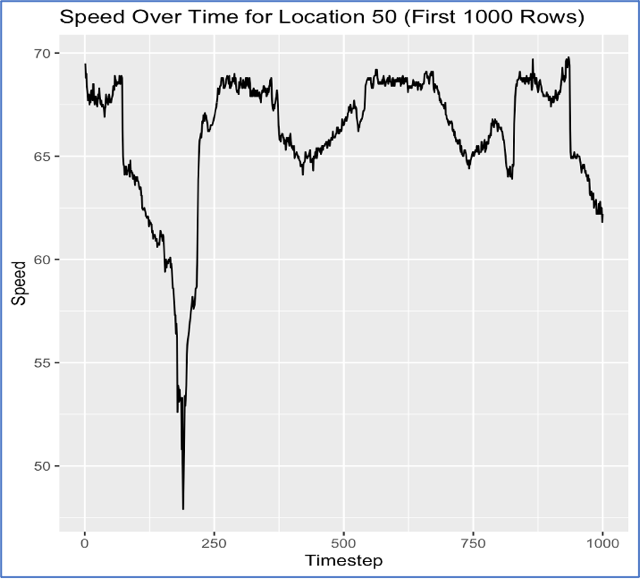
Time Series Visualization: Line plots were constructed to examine trends over temporal dimensions for the three principal variables, speed, flow, and occupy. These visual representations proved instrumental in elucidating seasonal patterns and various trends within the dataset, yielding significant insights for subsequent modeling and analysis.



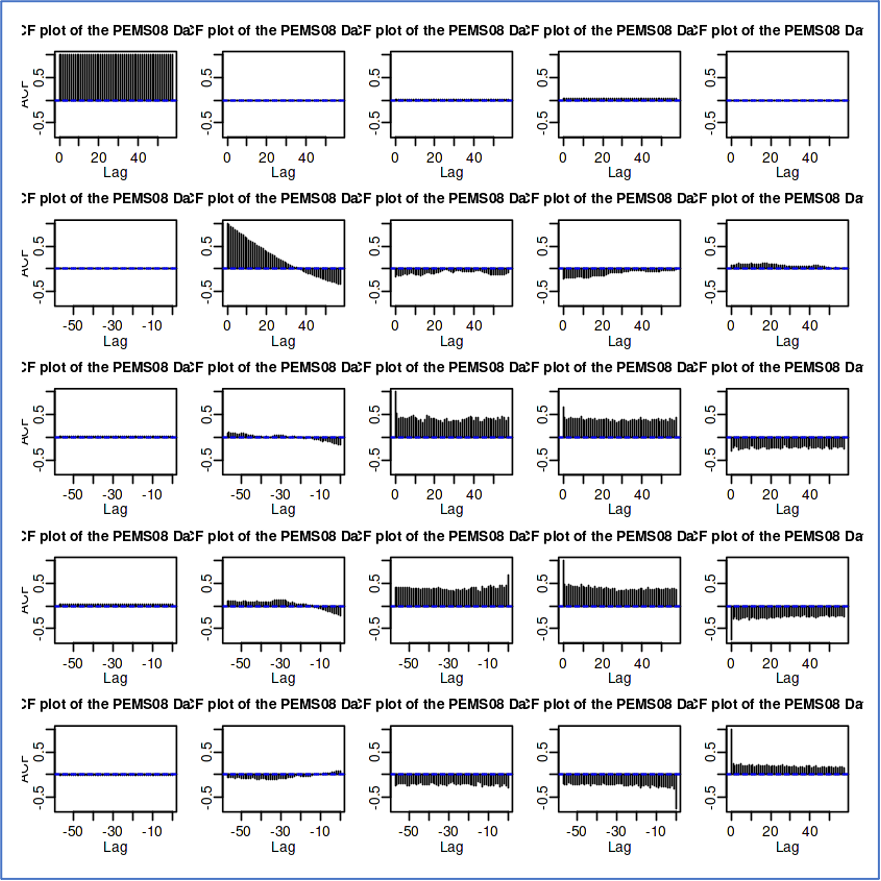
**Fig. 6** Occupancy Over Time for Location 50



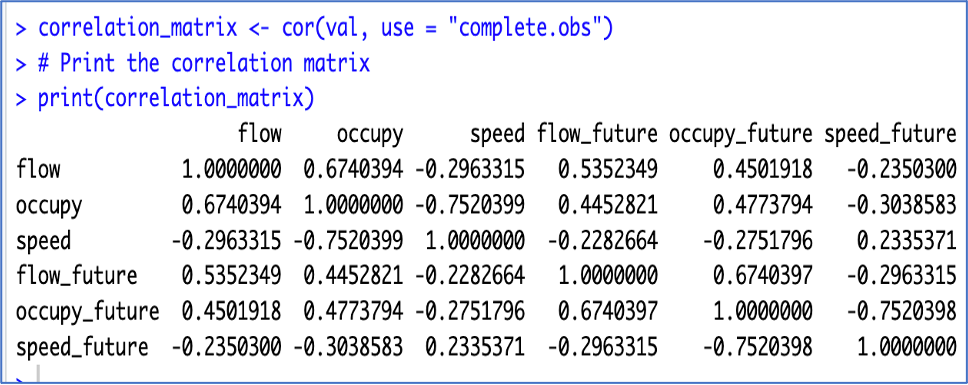
**Fig. 7** Flow Over Time for Location 50



**Fig. 8** Speed Over Time for Location 50



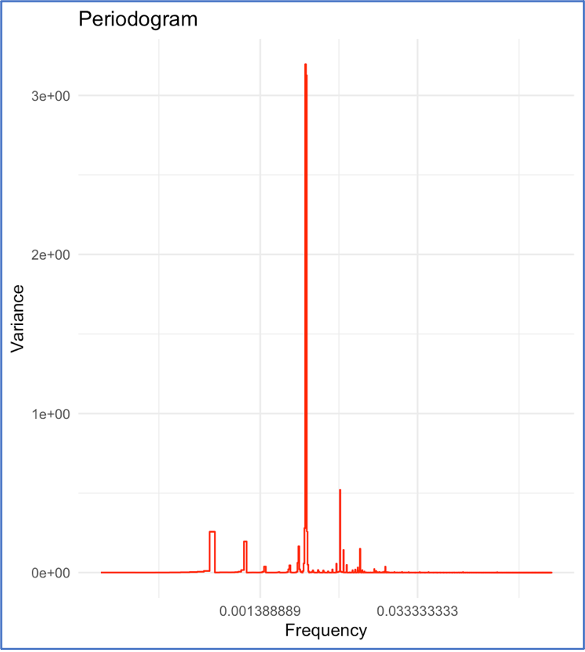
**Fig. 9** ACF Plot



**Fig. 10** Correlation Matrix

Positive Correlations: flow and occupy (0.674): A moderately robust positive correlation is evident, indicating that as traffic flow escalates, occupancy is likely to increase concurrently. flow and flow future (0.535): This finding denotes a substantial temporal dependency; the current flow is a significant determinant of future flow. occupy and occupy future (0.477): A comparable temporal dependency is observed with respect to occupancy. Negative Correlations: speed and occupancy (-0.752): A robust negative correlation suggests that increased occupancy is generally associated with diminished speeds, which is a logical conclusion within the context of traffic dynamics. Speed and flow future (-0.296): This denotes that present speeds may exert a slight inverse effect on future flow rates. Occupy future and speed future (-0.752): This further corroborates the association between elevated occupancy levels and a decline in speeds anticipated in the future as well. Weak or Near-Zero Correlations: flow and speed (- 0.296): A weak negative correlation suggests that flow does not significantly influence velocity, although there exists a slight negative association. Velocity and flow future (-0.228): This indicates that the present velocity has a negligible direct effect on subsequent flow. [1][2][23]

A periodogram was also constructed to evaluate the seasonality of the occupancy variable. The periodogram illustrates the intensity of the frequencies inherent in the time series and serves as a tool for discerning the predominant frequencies or periodicities within the dataset. The graph presented above indicates a discernible pattern in the occupancy variable on a daily basis, which is logically consistent, given that numerous individuals operate and commute according to a predetermined schedule. (For instance, peak hours post-work are likely to exhibit increased traffic.) Consequently, it is imperative to incorporate a temporal feature into our predictive model. [5][24][25]



**Fig. 11** Periodogram of Flow for Location 50

4.6 Stationary Testing

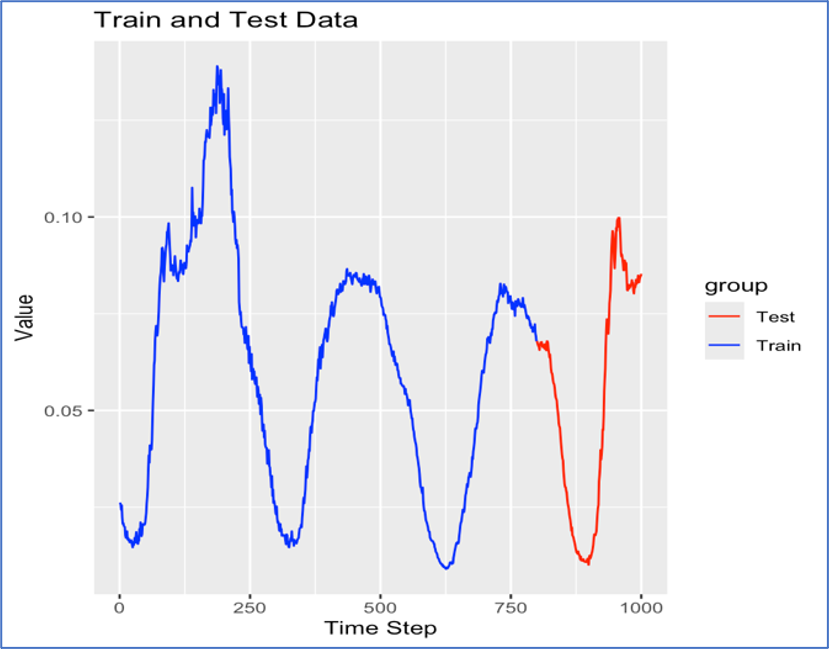
The Augmented Dickey-Fuller (ADF) test was applied to the time series data of flow, speed, and occupy to check the stationarity. For transformation purposes, the firstorder differencing was used as a transformation method to stabilize the mean and remove trends from the non-stationary time series data. The results of the transfor mations indicated that the differenced series achieved stationarity, allowing for more accurate modeling and forecasting of the underlying patterns in the data. [2][3]

* Augmented Dickey-Fuller Test for Flow
* Dickey-Fuller = -2.641, Lag order = 9, p-value = 0.307
* Alternative hypothesis: stationary
* Augmented Dickey-Fuller Test for Speed
* Dickey-Fuller = -2.2388, Lag order = 9, p-value = 0.4772
* Alternative hypothesis: stationary
* Augmented Dickey-Fuller Test for Occupy
* Dickey-Fuller = -2.3697, Lag order = 9, p-value = 0.4218
* Alternative hypothesis: stationary

4.7 Splitting Train and Test Set

The target variable, “occupy” was prepared for the purpose of model training and evaluation through the following methodologies:

* Addressing Missing Values: The missing entries within the ”occupy” variable were substituted with the arithmetic mean of the existing observations to ensure uniformity and to mitigate potential complications during the analytical process.
* Data Partitioning: The initial 799 observations of the ”occupy” variable were desig nated for incorporation into the training dataset. The subsequent 200 observations (ranging from index 800 to 1,000) were allocated for the purpose of model testing. [19]



**Fig. 12** Train and Test Split

4.8 Model Development and Implementation

All the modeling methodologies and forecasting analyses presented in this section were executed within the R programming framework. The study deployed the following time series forecasting model:

ARIMA – Autoregressive Integrated Moving Average, which is widely utilized for its effectiveness in capturing various patterns within time series data. This model allows for the incorporation of both autoregressive and moving average components, making it suitable for handling non-stationary datasets by differencing them to achieve stationarity. [6][7]

Mathematic Formula of ARIMA:

(2)

Where,

yt = The actual value of the time series at time (3)

c = A constant term (if included). (4)

ϕi = Coefficients of the autoregressive (AR) terms. (5)

yt−i = Lagged values of the time series (AR terms). (6)

ϵt = White noise (error term) a time t (7)

θj = Coefficients of the moving average (MA) terms. (8)

ϵt−j = Lagged error terms (MA terms). (9)

p = Order of the AR part. (10)

d = Degree of differencing (11)

q = Order of the MA part. (12)

(13)

SARIMA – ARIMA with Seasonal Component, which is one of the ARIMA models with seasonal terms, enabling it to capture seasonal patterns and trends in data more effectively. [7][19] Mathematic Formula of SARIMA:

(14)

Where,

yt = The actual value of the time series at time t, (15)

c = A constant term (if included). (16)

ϕi = Coefficients of the non-seasonal AR terms. (17)

Φi = Coefficients of the seasonal AR terms. (18)

θi = Coefficients of the non-seasonal MA terms. (19)

Θi = Coefficients of the seasonal MA terms. (20)

Yt−s = Lagged values of the seasonal component. (21)

s = Seasonal period. (22)

P, Q = Orders of the seasonal AR and MA terms. (23)

p, q = Orders of the non-seasonal AR and MA terms. (24)

d, D = Degrees of non-seasonal and seasonal differencing. (25)

(26)

GARCH – Generalized AutoRegressive Conditional Heteroskedasticity, which is particularly useful for modeling time series data with volatility clustering, allowing researchers to better understand and predict changes in variance over time. [4]

Mathematic Formula of GARCH:

(27)

Where,

σ2t = Conditional variance at time t, (28)

ϵt = Residuals (errors) at time t, (29)

α0 = Constant term. (30)

αi = Coefficients of the past squared errors (ARCH terms). (31)

βj = Coefficients of the past variances (GARCH terms). (32)

p, q = Orders of the ARCH and GARCH components. (33)

(34)

Exponential Smoothing, a forecasting method that applies decreasing weights to past observations, giving more importance to recent data points while smoothing out fluctuations, making it ideal for short-term forecasts. [1][6]

Mathematic Formula of Exponential Smoothing:

(35)

Where,

= Forecasted value at time t, (36)

yt = Actual value at time t, (37)

α = Smoothing parameter (0 ¡ α ≤ 1). (38)

(39)

Kalman Filter, a mathematical algorithm that uses a series of measurements observed over time to estimate the unknown state of a dynamic system, providing optimal estimates even in the presence of noise and uncertainty. [7]

Mathematic Formula of Kalman Filter:

(40)

Where,

= State estimate at time t, (41)

A = State transition matrix. (42)

B = Control input matrix. (43)

ut = Control vector at time t. (44)

Kt = Kalman gain at time t. (45)

yt = Observed value at time t. (46)

H = Observation model. (47)

(48)

Facebook Prophet - an additive method that facilitates forecasting time series data by incorporating seasonal effects and holiday effects, enabling users to generate reliable predictions even with missing values or outliers in the dataset. [26]

Mathematic Formula of Facebook Prophet:

(49)

Where,

yt = Observed value at time t, (50)

g(t) = Trend function (linear or logistic). (51)

s(t) = Seasonal component. (52)

h(t) = Holiday effects. (53)

ϵt = Error term. (54)

(55)

LSTM with Moving Average Baseline - In terms of model implementation and development results, the findings show that each model has its own advantages and disadvantages, with performance differing based on the specific features of the dataset being studied. The Moving Average - MA as a baseline for LSTM modeling acts as an important step in assessing the effectiveness of more advanced models by offering a straightforward method to smooth out short-term variations and emphasize long-term trends in the traffic dataset. Consequently, the following steps were taken to obtain the results. [3][7]

Mathematic Formula of MA:

(56)

Where,

= Forecasted value at time t, (57)

yt−i = Actual value of the time series at lag i, (58)

n = Number of periods for averaging. (59)

(60)

Mathematic Formula of LSTM:

(61)

(62)

(63)

(64)

(65)

(66)

where,

ft = Forget gate activation at time t, (67)

it = Input gate activation at time t, (68)

ot = Output gate activation at time t, (69)

= Cell candidate (new candidate values for the cell state). (70)

ct = Cell state at time t, (71)

ht = Hidden state at time t, (72)

xt = Input vector at time t, (73)

σ = Sigmoid activation function,

σ(x) = 1 + e−x, (74)

tanh = Hyperbolic tangent function, tanh(x) = ex − e−x ex + e−x, (75)

Wf , Wi, Wc, Wo = Weight matrices for forget, input, cell, and output gates. (76)

bf , bi, bc, bo = Bias vectors for forget, input, cell, and output gates. (77)

⊙ = Element-wise (Hadamard) product. (78)

4.9 Understanding the Metrics

The following performance evaluation metrics have been used to measure how well the models are fit for predicting future traffic flow and to identify any discrepancies between the predicted and actual values. Each metric has a specific role in evaluating various elements of model performance, enabling a detailed analysis that guides future enhancements and adjustments in predictive modeling methods.

Spearman Correlation: The Spearman correlation is a statistical technique used to measure the strength and direction of the association between two ranked variables. Unlike the Pearson correlation, which assesses linear relationships, the Spearman correlation focuses on monotonic relationships, where the variables tend to move in the same or opposite directions, regardless of the specific form of the relationship. In the context of traffic flow prediction, the Spearman correlation can be valuable for evaluating how well the ranked predictions from a model align with the ranked actual traffic data. [1][25]

ME (Mean Error): The mean error (ME) provides a simple measure of the average bias in your traffic flow forecasts. It is calculated by averaging the differences between the predicted traffic values and the actual observed traffic values. A negative ME value suggests that the model tends to under-predict traffic flow, while a positive ME value indicates a tendency to over-predict. [1][20][24]

RMSE (Root Mean Squared Error): The root mean squared error (RMSE) is widely used to assess the accuracy of predictive models. It quantifies the average magnitude of the errors between the predicted values and the true values. A lower RMSE value generally indicates a better fit and higher predictive accuracy. [1][3][25]

MAE (Mean Absolute Error): The mean absolute error (MAE) is another common metric for evaluating model performance. It calculates the average absolute difference between the predicted and actual values. The MAE is less sensitive to outliers compared to RMSE, as it does not square the errors. [3][25][26]

MPE (Mean Percentage Error): The mean percentage error (MPE) expresses the average error as a percentage of the actual values. It helps to understand the relative magnitude of the errors. [3]

MAPE (Mean Absolute Percentage Error): Similar to MPE, the mean absolute percentage error (MAPE) provides a measure of prediction accuracy in percentage terms. However, it uses the absolute values of the errors, making it less sensitive to the direction of the errors (overprediction vs. underprediction). [2][3]

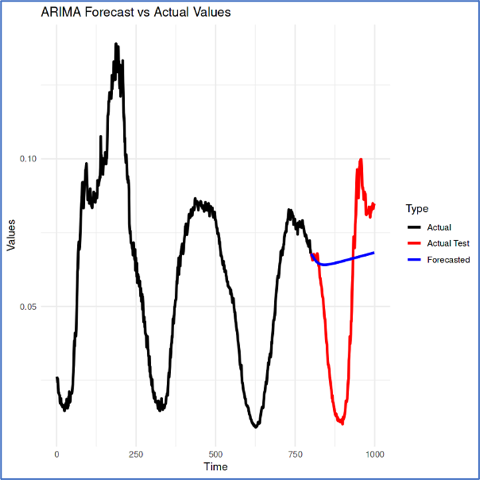
4.10 Prediction with ARIMA

The ARIMA (2,1,2) model was configured with 2 autoregressive (AR) and 2 moving average (MA) components, demonstrating a strong positive correlation with the immediate prior value and a weaker negative correlation with the value from two steps back. The MA terms reveal a strong negative influence from the prior error, coupled with a positive influence from the error two steps prior. While the model effectively captures general cyclic patterns in the training data and performs well in identifying linear trends and stationary components, it exhibits limitations when faced with abrupt changes in the test data. Forecasted values initially align closely with the actual data, but as time progresses, they increasingly deviate, highlighting the model’s struggles with high variability and significant deviations during testing. These weaknesses suggest challenges in modeling non-linear or seasonal dependencies, particularly in the context of traffic flow data.

4.10.1 Forecast vs Actual Graph (ARIMA)

Observed Trend: The ARIMA model captures the general cyclic patterns of traffic flow in the training segment (black line). However, it struggles to align with the actual test data (red line) during periods of abrupt changes, showing limitations in handling high variability in traffic dynamics.

Forecasting Gap: The forecasted values (blue line) show reasonable alignment at the start of the test period but increasingly deviate as the test period progresses. This highlights the ARIMA model’s inability to fully capture the complexities of traffic flow during unpredictable or high-variance periods.



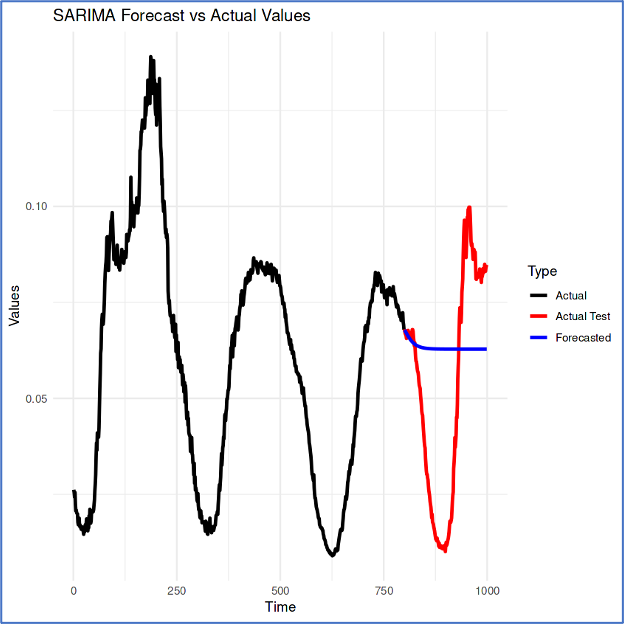
**Fig. 13** ARIMA: Actual Vs Forecast

4.11 Prediction with SARIMA

The ARIMA(2,1,2) model with seasonality demonstrated a good fit and effectively captured trends and seasonality in the training data. However, it faced challenges in the test set, particularly during high-variance periods, leading to noticeable deviations from actual values. While the model successfully aligns with the cyclic nature of traffic flow, its limitations in handling abrupt changes suggest that enhancements or integration with other models, such as machine learning, could improve its predictive performance.

4.11.1 Forecast vs Actual Graph (SARIMA)

Observed Trend: The model captures the cyclic nature of traffic flow, as evident from the alignment with actual data in training and testing segments. Forecasting Gap: While the forecasted values (blue line) closely align with the test data in the initial phase, there is a noticeable deviation as the test period progresses. This suggests the SARIMA model may not fully capture all complexities of traffic flow dynamics during high-variance periods.



**Fig. 14** SARIMA: Actual Vs Forecast

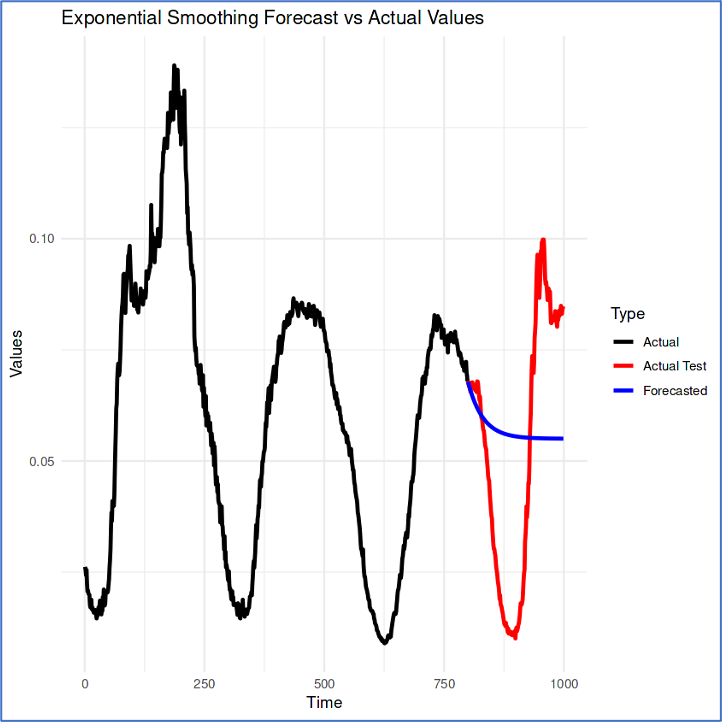
4.12 Prediction with Exponential Smoothing

The Exponential Smoothing model demonstrates strong performance in forecasting traffic flow during the training phase, effectively capturing trends and seasonality with minimal bias and errors. However, it faces challenges when applied to the test set, particularly in handling sudden changes in traffic patterns, resulting in underestimation and larger prediction errors. While it generally follows overall trends, the model struggles during sharp fluctuations, indicating a need for additional models or enhancements to improve accuracy in dynamic traffic conditions.

4.12.1 Forecast vs Actual Graph (Exponential Smoothing)

Observed Trend: The Exponential Smoothing model effectively reflects the overall trend and seasonal changes in traffic flow during the training phase (black line). However, in the testing period (red line), the model struggles to adjust to sudden shifts, especially in regions with steep peaks and valleys.

Forecasting Gap: The predicted values (blue line) offer a smoother forecast of traffic flow but show significant differences from the actual test data, particularly during times of quick changes. This indicates that the Exponential Smoothing model is not very effective at managing high variability and sudden shifts in traffic trends.



**Fig. 15** Exponential Smoothing: Actual Vs Forecast

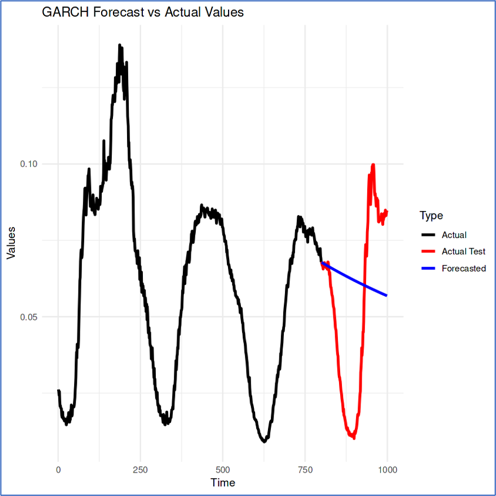
4.13 Prediction with GARCH

The GARCH model was employed to forecast occupancy levels in a traffic flow dataset, showing a small bias in predictions (ME: 9.20e-03) and moderate prediction errors (RMSE: 3.23e-02, MAE: 2.77e-02). However, it exhibited significant deviations between predicted and actual values, particularly during large traffic flow variations, as indicated by high MPE (101.09%) and MAPE (122.36%). While the model effectively captures general trends and short-term volatility in stationary series with heteroscedasticity, it struggles with sharp fluctuations and complex dependencies. To improve performance, integrating the GARCH model with other approaches that address trends, seasonality, and abrupt changes in traffic flow is recommended.

4.13.1 Forecast vs Actual Graph (GARCH)

Observed Trend: The GARCH model captures the general trends in the training segment (black line) but struggles to align with the actual test data (red line), particularly during sharp changes in traffic flow.

Forecasting Gap: The forecasted values (blue line) deviate significantly from the actual test data as the test period progresses, showing limitations in adapting to abrupt traffic fluctuations.



**Fig. 16** GARCH: Actual Vs Forecast

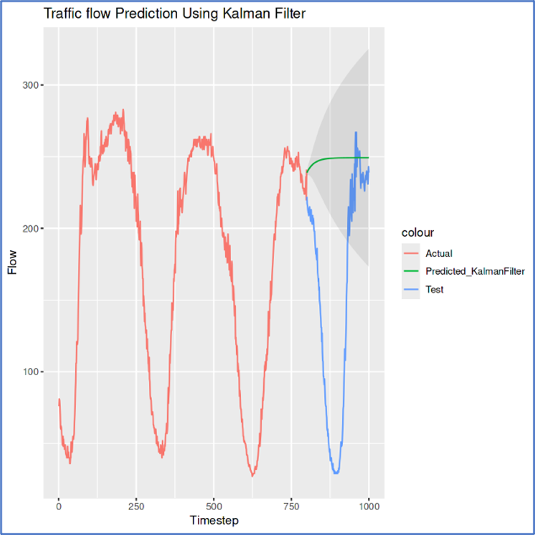
4.14 Prediction with Kalman Filter

The Kalman Filter model, combined with ARIMA, was utilized to forecast traffic flow occupancy. It demonstrated strong performance during the training phase, effectively capturing regular traffic patterns and providing insights into prediction confidence through uncertainty bounds. However, in the test phase, the model faced challenges, exhibiting significant performance degradation due to its inability to adapt to abrupt shifts and high-variance conditions, leading to noticeable discrepancies between predicted and actual values. Overall, while the model excels in stable periods, it struggles during dynamic changes in traffic flow.

4.14.1 Forecast Vs Actual Graph (Kalman Filter)

Observed Trend: The model captures cyclic traffic flow patterns during training (red line). The forecast (green line) initially aligns well with actual data but shows discrepancies during high-variance periods. Uncertainty (shaded area) increases in the test period, indicating reduced confidence in longer predictions.

Forecasting Gap: A noticeable mismatch exists between the forecast (green line) and actual test values (blue line) during high variability or abrupt changes, highlighting the Kalman Filter’s limitations in adapting to rapid shifts in traffic dynamics.



**Fig. 17** Kalman Filter: Actual Vs Forecast

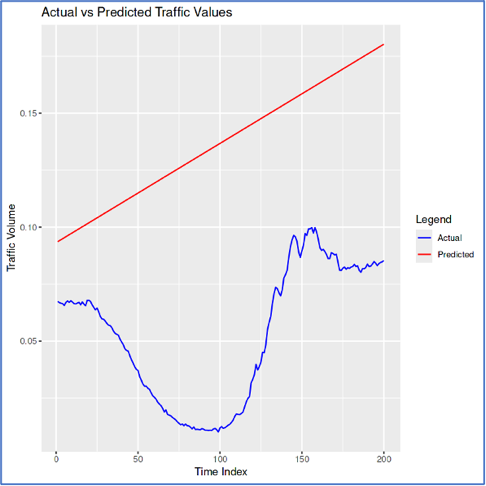
4.15 Prediction with Facebook Prophet

The Prophet model was applied to traffic data, demonstrating strong performance during the training phase with low error metrics and effective long-term trend identification. However, it struggled in the test phase, showing moderate bias and significant underestimation, which indicated its inability to capture seasonality and high-frequency fluctuations in traffic patterns. The model’s linear predictions resulted in poor generalization for testing, highlighting its unsuitability for short-term traffic flow forecasting characterized by high variability. While Prophet can identify general trends, it lacks the precision needed for dynamic datasets with abrupt shifts or irregular fluctuations, making it more suitable for datasets with distinct seasonality and trend patterns.

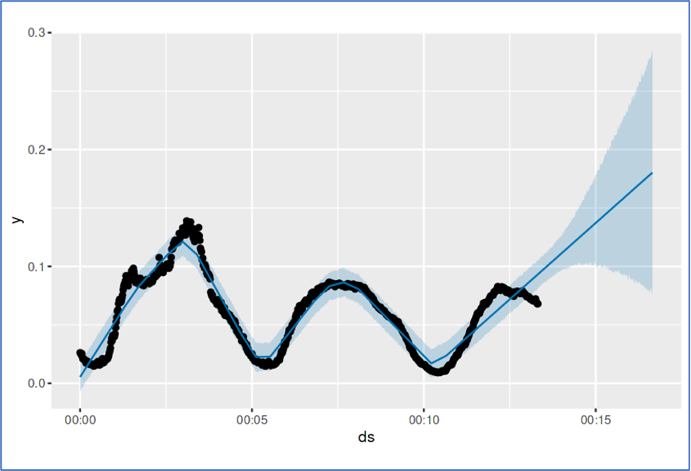
4.15.1 Forecast Vs Actual Graph (Facebook Prophet)

Observed Trend: The Facebook Prophet model shows a steady upward trend in its predictions (red line), which does not match the actual traffic data (blue line) closely. The model doesn’t catch the fluctuations of real data.

Forecasting Gap: The predicted traffic values consistently differ from the actual data throughout the period, indicating the model struggles to adjust to the changing and non-linear characteristics of traffic flow.



**Fig. 18** Prophet: Actual Vs Forecast

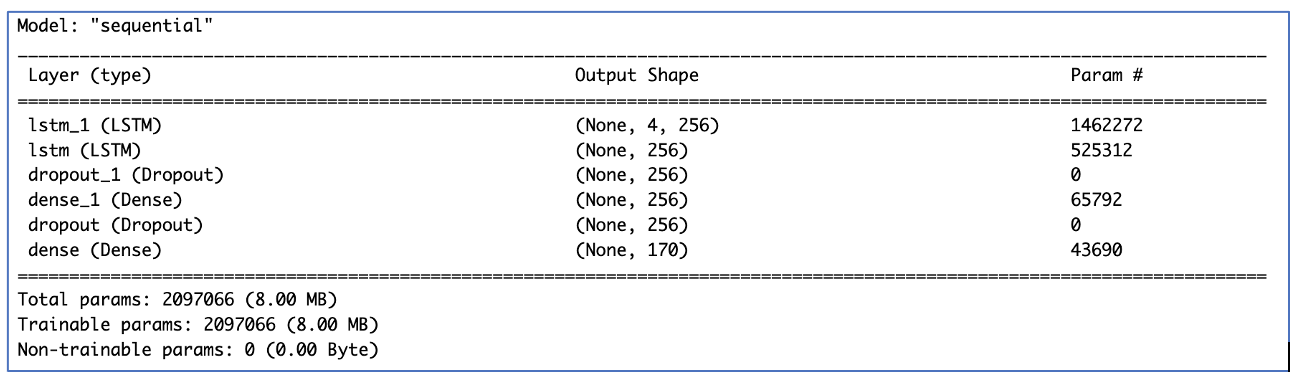


**Fig. 19** Forecasting with Prophet

4.16 Prediction with LSTM Model & Moving Average Baseline

The process of preparing traffic data for training an LSTM model with moving average as a baseline involves the following key steps: Data Selection and Pre-processing: The data is organized into hourly blocks from 5-minute segments for each site. Feature Engineering: Additional features like time of day and day of the week are included and transformed into one-hot vectors. Creating Lags: Lag features are generated using a sliding window method, with time windows forming inputs (X) and the occupancy for the next time step as the target variable (Y). Data Splitting: The dataset is divided into training (80%) and testing (20%) sets Scaling: Min-Max scaling is applied to normalize both input features and target variables. Reshaping: The data is transformed into 2D arrays for scaling and returned to its original size afterward. Final Data Preparation: The train and test datasets are verified for correct dimensions, ensuring they are ready for model training and assessment. Overall, these steps ensure that the traffic data is cleaned, important features are included, lag variables are created, and the data is standardized for effective model training.

Train the Dataset: To effectively conduct a comprehensive analysis and make accurate predictions, it is essential to meticulously train the traffic dataset so that it aligns perfectly with the parameters and requirements of the moving average model, which is designed to provide insights into trends over time.



**Fig. 20** Summary of Dataset After Training

Build Long Short-Term Model (LSTM) Model: Build the LSTM model to process data in order and maintain a hidden state. Long Short-Term Memory Networks, or LSTM, are neural networks that help information stay over time in deep learning. This type of Recurrent Neural Network effectively addresses the vanishing gradient issue that RNNs encounter. Hochreiter and Schmidhuber created LSTM to fix the issues present in traditional RNNs and machine learning methods. The outcome is shown in the following figure:

Model Architecture: A Sequential model is employed, meaning the layers are arranged in a linear fashion. The initial layer is an LSTM (Long Short-Term Memory) layer containing 256 units. This layer handles the input data and provides sequences for the next LSTM layer. The second LSTM layer also consists of 256 units, but it does not output sequences since it is followed by a dense layer.

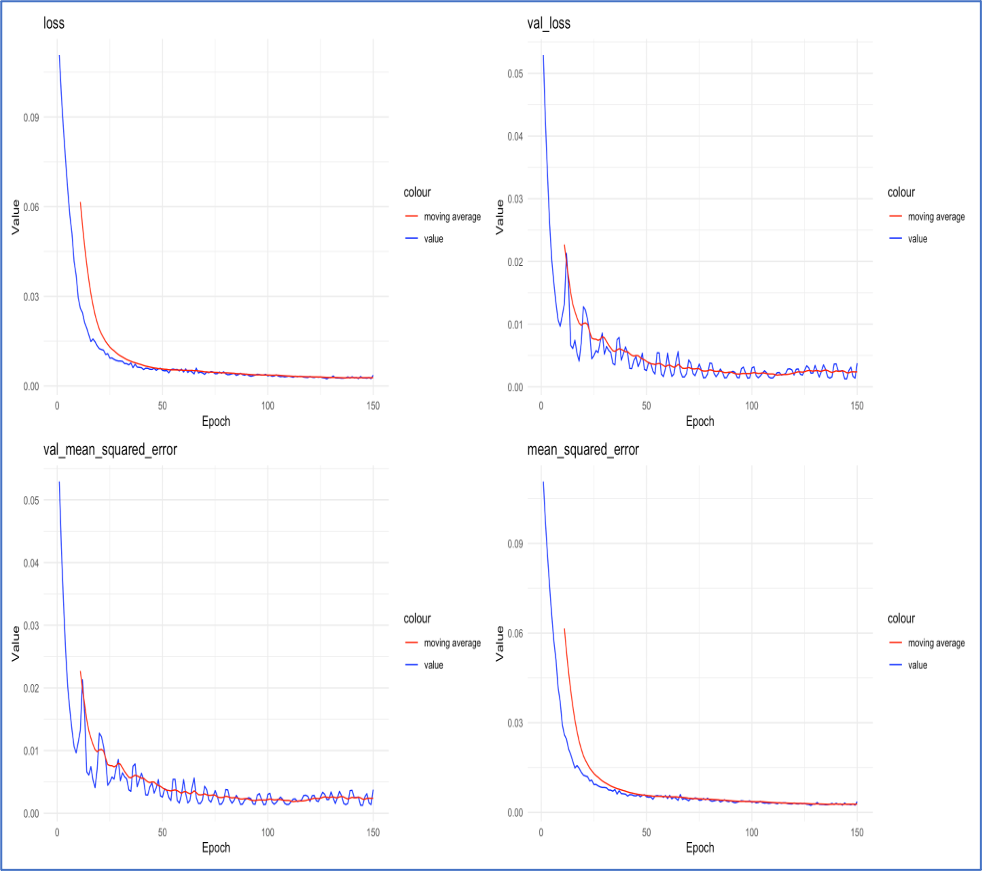
Dropout layers, with a rate of 0.2, are included to avoid overfitting by randomly zeroing some input values during training. A Dense layer with 256 units and ReLU activation processes the features further. Another Dropout layer is incorporated at the same rate of 0.2 for continued regularization. The last Dense layer has 170 units (matching the number of locations in the dataset) and uses a linear activation function. This layer generates the final predicted values for each location.

Model Compilation: The model is set up with the Adam optimizer, a smart algorithm that adjusts the learning rate while training.

The Mean Squared Error (MSE) loss function is chosen since this is a regression task aimed at reducing the gap between predicted and actual values. The model also utilizes Mean Squared Error as a metric to monitor its performance during training.

Model Training: The model is trained using the training data for 150 epochs, which means it will process the entire training data 150 times. A batch size of 32 is applied, indicating the model will adjust its weights after processing 32 samples at once. A validation split of 0.1 means that 10% of the training data will serve for validation during training to track performance on new data. The training progress is shown using the verbose = 2 option, which gives a detailed log of the training process.

Plotting the Training History: After completing the training, the plot(history) command helps to display the training progress, showing the loss and performance metrics throughout the epochs. The dataset was prepared to create a deep LSTM model featuring two LSTM layers, dropout for regularization, and dense layers for making predictions. The model is set up with an Adam optimizer and MSE loss function, trained for 150 epochs, and assessed using validation data. The training history is subsequently displayed to track the model’s performance throughout training.



**Fig. 21** LSTM Model Training History

Epoch: This indicates the number of times the model reviews all the training data. More epochs help the model learn better, but too many can lead to overfitting. Here, the model will train for 150 epochs.

Batch Size: This refers to how many data samples the model handles before it updates its weights. A smaller batch size allows the model to adapt more quickly to changes, while a larger batch size results in steadier training. In this case, a batch size of 32 is utilized. Furthermore, a validation split of 0.1 is applied, indicating that 10% of the training data is reserved for validation. This portion is excluded from training, but it aids in assessing the model’s performance on unfamiliar data during training. This ensures that the model does not overfit and can perform well on new data. By default, the final 10% of the dataset is designated for validation. The validation loss starts to level off after 120 training epochs, while the training loss continues to decrease. If training continues beyond this point, the model may become too tailored to the training data, resulting in lower training loss but increased validation loss. To prevent overfitting, it’s advisable to limit the training epochs to roughly this stage, which in this case is 150 epochs.

Predict & Score the Model: This process includes transforming data, creating and training an LSTM model, normalizing values, assessing predictions using RMSE, and displaying the outcomes to contrast predictions with real and baseline figures.

Prediction and Scoring: The predict and score function is utilized to predict the target values (Y) with the trained model. It also checks how the model’s predictions match the real values, calculating the RMSE for both the model and a basic moving average. The moving average is calculated using a 12-step window to smooth the real data.

Plotting: The plot prediction function shows the real values, the moving average, and the model’s forecasts over time. This allows us to evaluate the model’s performance compared to the baseline.

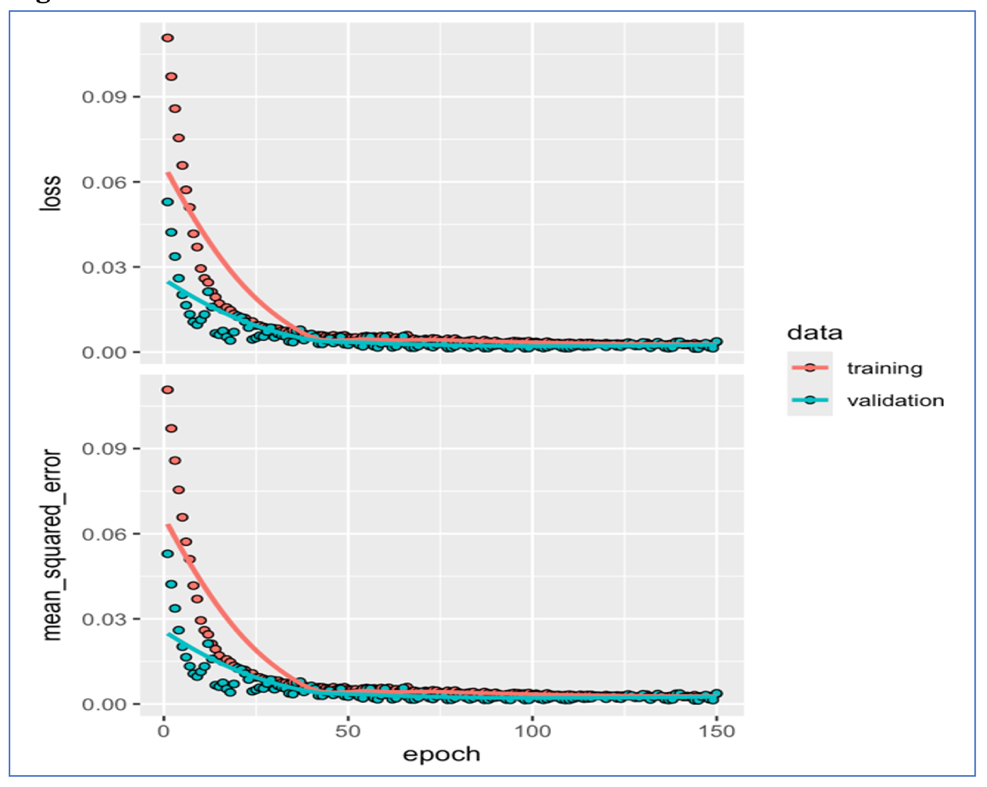
5 Results

To thoroughly evaluate the effectiveness and performance of the models and framework, it was necessary to utilize the most common performance metrics, which include the Spearman Correlation, Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE), all of which are essential for providing a comprehensive assessment of the framework’s accuracy and reliability. Train Dataset Results: The results from training the Root Mean Square Error (RMSE) and the Spearman correlation coefficient show very strong performance, indicating that the model being tested is better than the traditional moving average method and is not facing any underfitting issues. However, it is important to remember that the best measure of the model’s overall success is its performance metrics on the test dataset since neural network models often face overfitting challenges, which can distort the results and not accurately represent true predictive abilities.

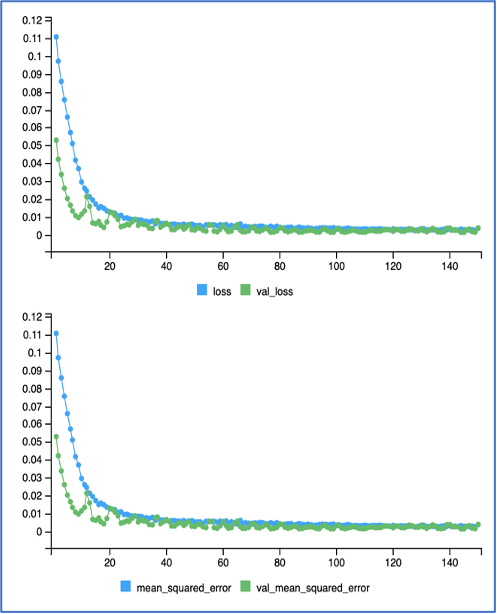
Test Dataset Results: The evaluation metrics, particularly the test Root Mean Square Error (RMSE) and the Spearman Correlation, show outstanding performance that greatly surpasses the baseline set by the moving average calculation, suggesting strong analytical ability. This finding strongly indicates that the model does not fall into the usual traps of overfitting, thus proving its ability to generalize well when used with new, unseen data sets.

5.1 LSTM: Validation of Lost & MSE

The validation loss starts to level off after 120 training epochs, while the training loss continues to decrease. If training continues beyond this point, the model may become too tailored to the training data, resulting in lower training loss but increased validation loss. To prevent overfitting, it’s advisable to limit the training epochs to roughly this stage, which in this case is 150 epochs.



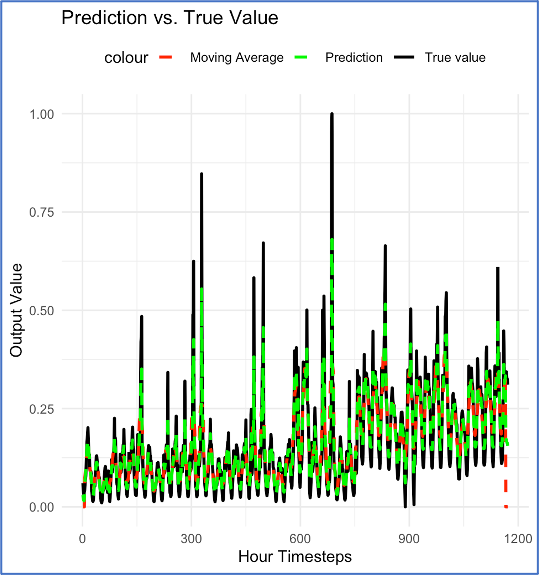
**Fig. 22** Training Loss & Validation Loss



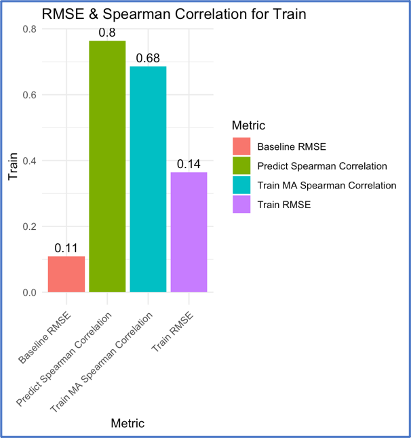
**Fig. 23** Validation of Loss & MSE

5.2 LSTM: Train Dataset Line Plot

The model tested shows strong performance based on RMSE and Spearman correlation, outperforming traditional moving averages without underfitting issues. However, the true measure of success lies in its performance on the test dataset, as neural networks can often face overfitting, which may skew results.



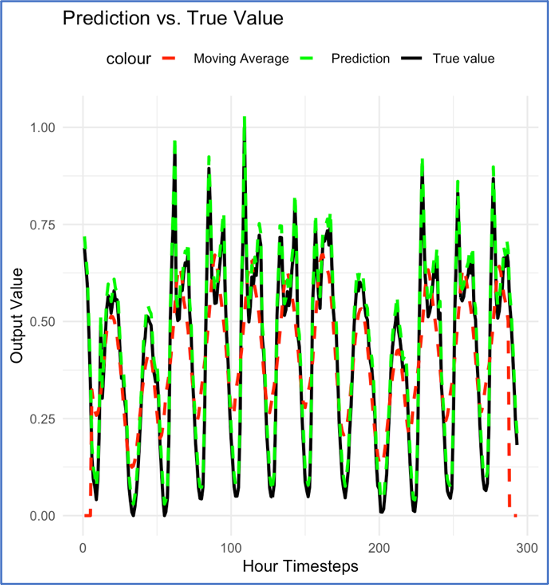
**Fig. 24** LSTM: Actual Vs Forecast (Train Set)



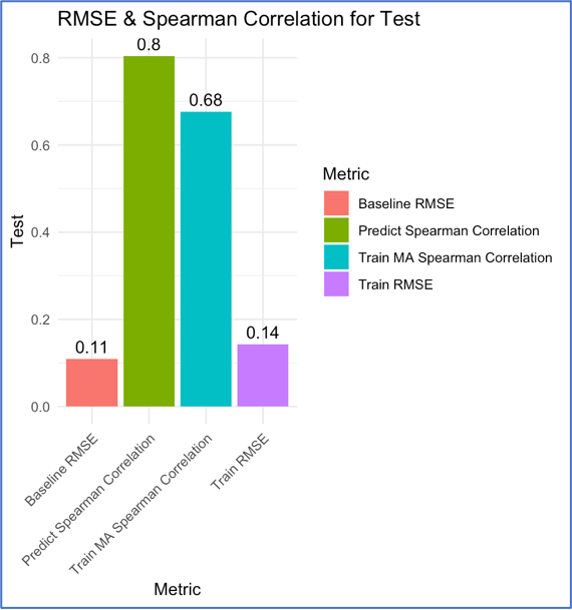
**Fig. 25** RMSE & Spearman Correlation (Train)

5.3 LSTM: Test Dataset Line Plot

The evaluation metrics, including test RMSE and Spearman correlation, demonstrate exceptional performance that exceeds the baseline established by the moving average. This suggests a strong analytical capability and indicates that the model effectively avoids overfitting, allowing it to generalize well with new, unseen data sets.



**Fig. 26** LSTM: Actual Vs Forecast (Test Set)



**Fig. 27** RMSE & Spearman Correlation (Test)

6 Equations

Spearman Correlation

(79)

Where,

(80)

(81)

(82)

Mean Absolute Error – MAE

) = (83)

Root Mean Square Error – RMSE

(84)

Mean Absolute Percentage Error – MAPE

(85)

Where,

(86)

(87)

(88)

7 Tables

7.1 Performance Evaluation Metrics

The comparative analysis reveals that SARIMA performs well for seasonal patterns, but its error increases during high variability. GARCH captures volatility effectively but fails to address abrupt traffic shifts. LSTM consistently outperforms other models by adapting to non-linear trends and minimizing prediction errors across all metrics. The table below summarizes the performance metrics of different model:

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **MAE** | **RMSE** | **MAPE** |
| **LSTM**  ARIMA  SARIMA  Exponential Smoothing  GARCH  Kalman Filter  Facebook Prophet  Moving Average | **0.030**  0.025  0.026  0.025  2.77  0.97  0.084  0.026 | **0.031**  0.031  0.031  0.030  3.23  0.091  0.089  0.03 | **46%**  124%  121%  106%  122%  3193%  327%  115% |

**Table 1** Model Performance Metrics

8 Conclusion

This study evaluated time series models for traffic flow prediction, demonstrating that while traditional models like SARIMA and GARCH address specific aspects (seasonality and volatility, respectively), LSTM networks excel in capturing complex temporal dependencies. The findings underscore the potential of integrating advanced machine-learning techniques with statistical models to improve predictive accuracy and scalability. Future research should focus on hybrid modeling approaches and real-time data integration for smarter urban mobility solutions.

The results highlight the need to choose the right models based on the type of traffic data and the prediction goals. For example, SARIMA is suggested for situations where accuracy is crucial, while GARCH is suitable for exploring traffic unpredictability. Future research might consider combining statistical methods with machine learning approaches like LSTM or attention mechanisms to improve prediction accuracy and scalability. These improvements will be key to achieving smarter and more sustainable transportation systems.

References

[1] Lu, W., Rui, Y., Yi, Z., Ran, B., Gu, Y.: A hybrid model for lane-level traffic flow forecasting based on complete ensemble empirical mode decomposition and extreme gradient boosting. IEEE Access 8, 42042–42054 (2020) https://doi.org/ 10.1109/ACCESS.2020.2977219

[2] Zheng, J., Huang, M.: Traffic flow forecast through time series analysis based on deep learning. IEEE Access 8, 82562–82570 (2020) https://doi.org/10.1109/ ACCESS.2020.2990738

[3] Katambire, V.N., Musabe, R., Uwitonze, A., Mukanyiligira, D.: Forecasting the traffic flow by using arima and lstm models: Case of muhima junction. Forecasting 5(4), 616–628 (2023) https://doi.org/10.3390/forecast5040034

[4] Tunnicliffe Wilson, G.: Time series analysis: Forecasting and control,5th edition, by george e. p. box, gwilym m. jenkins, gregory c. reinsel and greta m. ljung, 2015. published by john wiley and sons inc., hoboken, new jersey, pp. 712. isbn: 978-1-118-67502-1. Journal of Time Series Analysis 37, (2016) https://doi.org/ 10.1111/jtsa.12194

[5] Hyndman, R.: Forecasting: Principles and Practice. OTexts, ??? (2018)

[6] Shumway, R.H., Stoffer, D.S.: Time Series Analysis and Its Applications. Springer, ??? (2017)

[7] V., R., S., G.V.: Hybrid time-series forecasting models for traffic flow prediction. Promet 34(4) (2022) https://doi.org/10.7307/ptt.v34i4.3998

31

[8] Tiwari, N., Prasad, L.: Experimental analysis of time series models for traf fic flow prediction. SAMRIDDHI Journal of Physical Sciences, Engineering and Technology 15(3), 312–320 (2023) https://doi.org/10.18090/samriddhi.v15i03.05

[9] Wang, G., Chen, Y., Gao, M., Wu, Z., Tang, J., Zhao, J.: A time series is worth five experts: Heterogeneous mixture of experts for traffic flow prediction. arXiv (2024) https://doi.org/10.48550/arXiv.2409.17440 . arXiv:2409.17440

[10] Liao, L., Hu, Z., Hsu, C.-Y., Su, J.: Fourier graph convolution network for time series prediction. Mathematics 11(7), 1649 (2023) https://doi.org/10.3390/ math11071649

[11] Lee, S., Park, C.: Temporal graph learning recurrent neural network for traffic forecasting. arXiv (2024) https://doi.org/10.48550/arXiv.2406.02726 . arXiv:2406.02726

[12] Shah, I., Muhammad, I., Ali, S., Ahmed, S., Almazah, M.M.A., Al-Rezami, A.Y.: Forecasting day-ahead traffic flow using functional time series approach. Mathematics 10(22), 4279 (2022) https://doi.org/10.3390/math10224279

[13] Jin, C., Chen, J., Wu, S., Wu, H., Wang, S., Ying, J.: Caltm: A context-aware long-term time-series forecasting model. Computational Modeling in Engineering Sciences 139(1), 873–891 (2024) https://doi.org/10.32604/cmes.2023.043230

[14] Hu, Z., Shao, F., Sun, R.: A new perspective on traffic flow prediction: A graph spatial-temporal network with complex network information. Electronics 11(15), 2432 (2022) https://doi.org/10.3390/electronics11152432

[15] Reza, S., Ferreira, M.C., Machado, J.J.M., Tavares, J.M.R.S.: A multi-head attention-based transformer model for traffic flow forecasting with a comparative analysis to recurrent neural networks. Expert Systems with Applications 202, 117275 (2022) https://doi.org/10.1016/j.eswa.2022.117275

[16] Shah, S.A.A.: Deep Learning based Prediction of Vehicular Speed, Flow and Traffic Congestion using Real World Data (2023)

[17] Wang, P., Shi, T., He, R., Yuan, W.: Prediction of large scale spatio-temporal traffic flow data with new graph convolution model. In: Niu, M. (ed.) Intelligent Electronics and Circuits - Terahertz, ITS, and Beyond. IntechOpen, ??? (2022). https://doi.org/10.5772/intechopen.101756

[18] Choe, T., Skabardonis, A., Varaiya, P.: Freeway performance measurement sys tem: Operational analysis tool. Transportation Research Record 1811 (2002) https://doi.org/10.3141/1811-08

[19] Tiwari, N., Prasad, L.: Experimental analysis of time series models for traf fic flow prediction. SAMRIDDHI Journal of Physical Sciences, Engineering and Technology 15(3), 312–320 (2023) https://doi.org/10.18090/samriddhi.v15i03.05

[20] Jiang, W., Han, J., Liu, H., Tao, T., Tan, N., Xiong, H.: Interpretable Cascading Mixture-of-Experts for Urban Traffic Congestion Prediction. arXiv:2406.12923 (2024). https://doi.org/10.48550/arXiv.2406.12923

[21] Zhao, Z., Pan, X.: A new traffic flow prediction based on multi-scale convolutional neural network combined with LSTM neural network. In Review (2023). https: //doi.org/10.21203/rs.3.rs-2954224/v1

[22] Liu, H., Zhu, C., Zhang, D., Li, Q.: Multi-Scale Spatial-Temporal Recurrent Net works for Traffic Flow Prediction. arXiv:2310.08138 (2023). https://doi.org/10. 48550/arXiv.2310.08138

[23] Tm, V., Cn, C.: Traffic prediction using machine learning. SAMRIDDHI Journal of Physical Sciences, Engineering and Technology 15(3), 341–345 (2023) https: //doi.org/10.18090/samriddhi.v15i03.10

[24] Wang, L., Pan, K.: Quantitative analysis and model prediction of urban traffic flow optimization by geographic information technology. Applied Math ematics and Nonlinear Sciences 9(1), 20243265 (2024) https://doi.org/10.2478/ amns-2024-3265

[25] Sun, S., Yan, H., Lang, Z.: A study on traffic congestion prediction based on random forest model. Highlights in Science, Engineering and Technology 101, 738–749 (2024) https://doi.org/10.54097/fmmtqw30

[26] Feng, T., et al.: The comparative analysis of sarima, facebook prophet, and lstm for road traffic injury prediction in northeast china. Frontiers in Public Health 10, 946563 (2022) https://doi.org/10.3389/fpubh.2022.946563