Time Series Forecasting of Traffic Volume Using Statistical and Deep Learning Models

Parthiv Vaghani, Jay Rathod and Karnik Rathva

Btech ICT, Dhirubhai Ambani University

Abstract

Accurate traffic volume forecasting is vital for efficient urban planning, congestion management, and road safety. This project explores the application of both classical statistical models and modern deep learning architectures for time series forecasting of daily vehicle counts. Using a publicly available traffic dataset, we aggregated hourly data into daily summaries and incorporated external factors like public holidays to enrich the dataset. Classical models such as ARIMA, SARIMA, and SARIMAX were tuned using AIC and RMSE metrics, while deep learning models including Simple RNN, LSTM, and GRU were trained to capture complex temporal patterns and long-range dependencies. Comprehensive preprocessing, stationarity checks, and diagnostic tests like the ADF and ARCH tests ensured the reliability of the time series. Comparative evaluation based on forecast accuracy metrics revealed that deep learning models, particularly LSTM and GRU, outperformed traditional approaches, demonstrating their suitability for real-world traffic prediction tasks.

Contents

Btech ICT, Dhirubhai Ambani University
202201249@daiict.ac.in (P. Vaghani); 202201255@daiict.ac.in (J. Rathod); 202201266@daiict.ac.in (K. Rathva)

1. Problem Statement

Accurate forecasting of traffic volume is crucial for urban planning, congestion management, and road safety. Overestimating traffic demand can lead to overbuilt infrastructure and wasted resources, while underestimating it results in congestion, increased emissions, and higher accident risks. The primary goal of this project is to develop reliable forecasting models that predict daily vehicle counts using historical traffic data. By comparing the performance of traditional statistical techniques—such as ARIMA, SARIMA, and SARIMAX—with modern deep learning approaches including Simple RNN, LSTM, and GRU, this study aims to identify the most effective and accurate model for practical traffic volume forecasting.

2. Dataset and Preprocessing

The traffic data were sourced from Kaggle [1], containing hourly records of vehicle counts collected at circle. This collected data contained 4 junction wise vehicles count. Our project focuses on two key columns:

- **DateTime**: timestamp of each hourly observation.
- **Vehicles**: the number of vehicles observed during that hour.

Additional columns such as `Junction` and `ID` were excluded as they were outside the scope of this forecasting study. Notably, the dataset contained no missing or null values, ensuring clean and consistent input for downstream analysis and modeling.

2.1. Aggregation and Feature Engineering

The original dataset contains hourly traffic counts recorded at four different junctions. For this project, we filtered the data to include only records from **Junction 2**, as it had consistent and complete observations suitable for forecasting. original data (df) has hourly observations, it can be noisy and difficult to interpret overall patterns To transition from hourly to daily granularity. Thus we applied a sum aggregation using:

$$V_d = \sum_{h=1}^{24} V_{d,h}$$

where $V_{d,h}$ is the number of vehicles recorded at hour h on day d. This approach preserves the overall daily trend while smoothing out hourly fluctuations.

After resampling, the resulting series spans **566** days, from *2015–11–11* to *2017–05–29*.

2.2. Time Series Analysis

To enhance the depth of our time series analysis, we extended the dataset by integrating additional exogenous information that may influence vehicle count patterns.

In particular, a binary feature was introduced:

- `isHoliday`: Indicates whether a given date is a public holiday (1 for holiday, 0 otherwise).

The holiday information was obtained from an official public holiday dataset. Including this exogenous variable enables a more realistic modeling of variations in traffic behavior, particularly accounting for reduced vehicle flow on public holidays.

The traffic patterns differ between holidays and non-holidays, in Figure 4 the daily vehicle count over time. The blue line represents overall traffic trends, while the red dots highlight traffic volumes on holiday dates.

From the figure, we observe that most red dots, representing holidays, tend to lie at the lower end of the weekly traffic cycles, indicating a noticeable drop in vehicle count on those days. This suggests that traffic volume is generally reduced during holidays compared to normal weekdays.

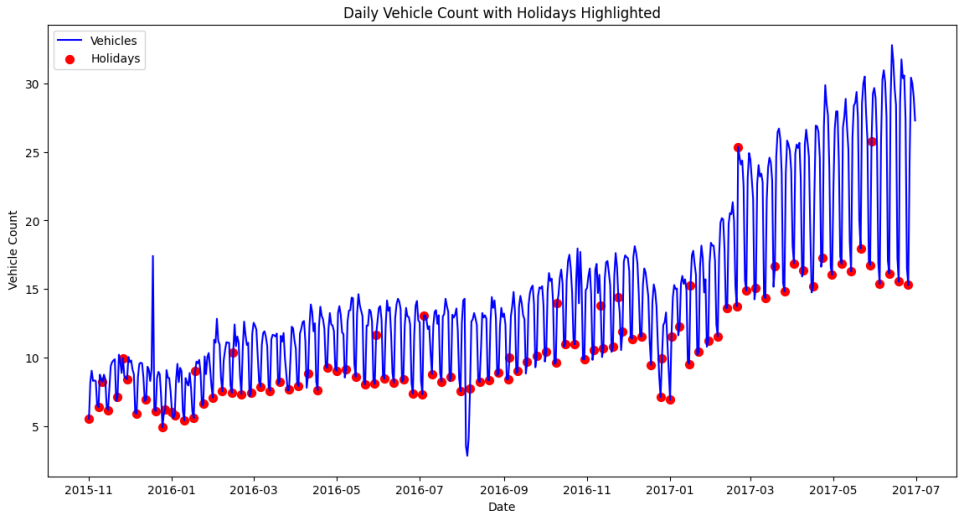


Figure 1: Daily Scaled Vehicle Count with Holidays Highlighted

- **Average vehicle count on holidays:** 252.88
- **Average vehicle count on non-holidays:** 360.48
- **Percentage difference: -29.85%**

This significant decrease in traffic volume (over 30%) on holidays indicates that public holidays have a marked impact on vehicle movement, likely due to reduced commuting and business activities.

3. Methodology

3.1. Data Resampling

We first filtered to **Junction 2** data only, then converted the original hourly series into daily granularity by computing the sum:

$$V_d = \sum_{h=1}^{24} V_{d,h},$$

where $V_{d,h}$ is the count at hour h on day d. The resulting series spans **566** days (2015-11-11 to 2017-05-29).

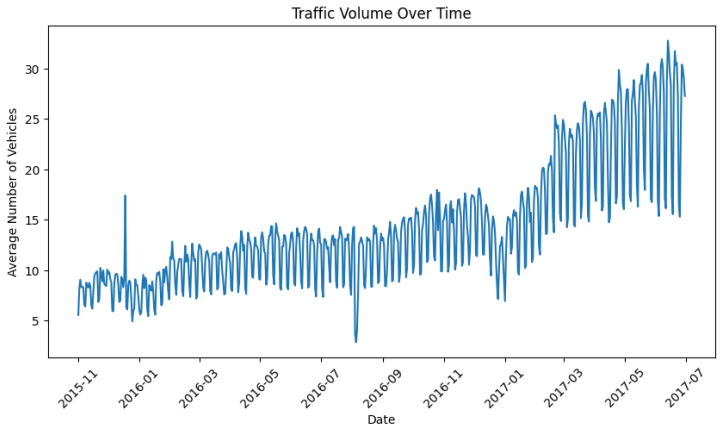


Figure 2: Original Daily data

3.2. Statistical Checks and Stationarity using ADF test

To test for stationarity, we ran the Augmented Dickey–Fuller test[1]; results are in Table **??**. Since the p-value (0.9635) is well above 0.05 and the test statistic (0.0642) exceeds the 5% critical value (−2.86), we conclude the original daily series is non-stationary.

Table 1
ADF Test on Vehicle Data Series

Metric	Value
ADF Test Statistic	0.4242
p-value	0.9824
Lags used	19
Observations	588
Critical Value (1%)	−3.4415
Critical Value (5%)	−2.8665
Critical Value (10%)	−2.5694

After applying first-order differencing, we repeated the ADF test. The differenced series is now stationary, as shown in Table 3.

Table 2
ADF Test on First-Differenced Vehicle Data Series

Metric	Value
ADF Test Statistic	−7.5263
p-value	3.68×10^{-11}
Lags used	19
Observations	538
Critical Value (1%)	−3.4426
Critical Value (5%)	−2.8669
Critical Value (10%)	−2.5696

3.3. Comparison of Mean and Variance

To further inspect stationarity, we plotted the mean and variance of the daily series. If both statistics remain roughly constant over time, it supports stationarity assumptions.

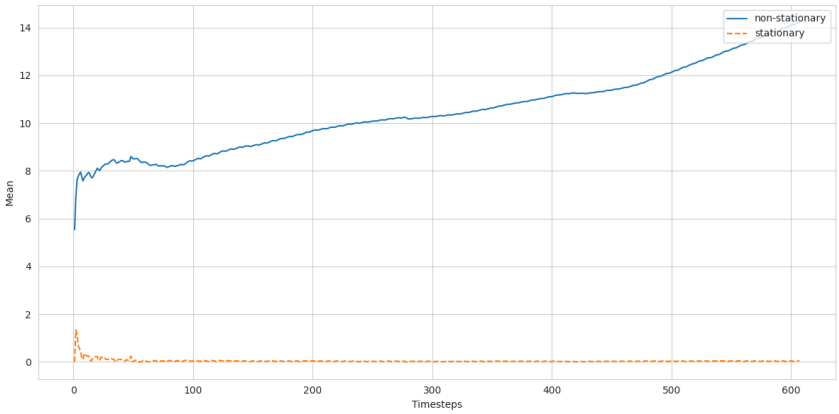


Figure 3: Mean comparison

[1] `statsmodels.tsa.stattools.adfuller`

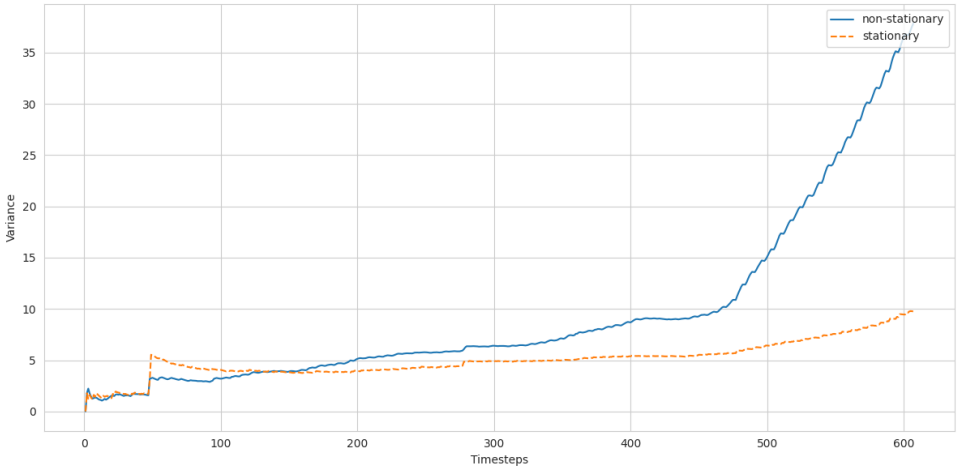


Figure 4: Variance comparison

Mean and Variance of stationary time series should be constant over the time. But there is one problem.Variance of our stationary series is not constant. so first we need to solve this problem.

3.4. Heteroskedasticity using ARCH test

If stationary time series still has fluctuating variance, we need to check for Conditional Heteroskedasticity (ARCH Effects).

Even if our data is stationary in mean, the variance might still be fluctuating due to conditional heteroskedasticity (ARCH/GARCH effects). we can check this using the ARCH test:

Table 3
ARCH Test on First-Differenced Series

Metric	Value
p-value	5.4×10^{-73}

Since our p value of ARCH test is less than 0.05, we can say we have time-dependent varience.

That is why we need different transformation method.Here we had taken log transformation, square root transformation , box-cox transformatin and box-cox difference transformation. See below figure.

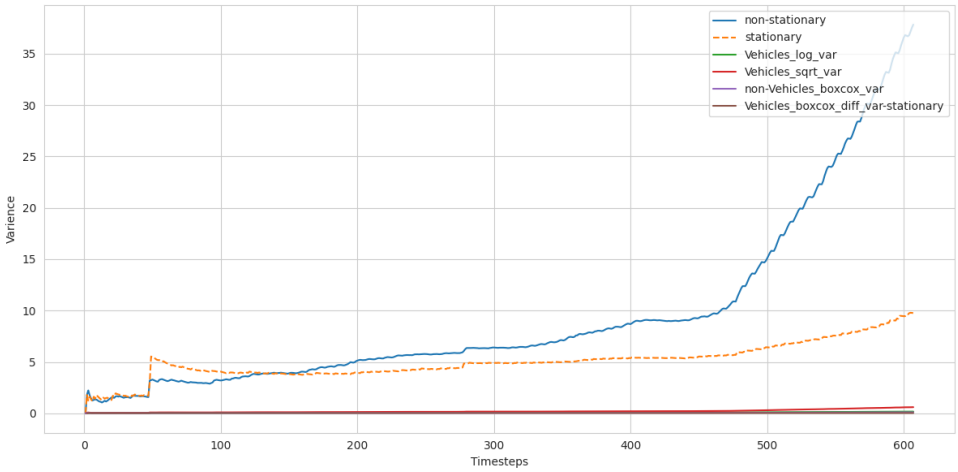


Figure 5: Variance comparison of different transformation method

Among all methods:
Log transformation is the one which is we have chosen.so now adf result of Vehicles_log

Table 4
ADF Test on Vehicles_log Data Series

Metric	Value
ADF Test Statistic	-0.4265
p-value	0.9056
Lags used	19
Observations	588
Critical Value (1%)	-3.4415
Critical Value (5%)	-2.8665
Critical Value (10%)	-2.5694

From above table we can see that p-value is more than 0.05 , so it's not stationary. To make this log stationary, we need to do further transformation.This time we will use differencing method.

Table 5
ADF Test on Vehicle_log_diff Data Series

Metric	Value
ADF Test Statistic	-9.0836
p-value	3.99×10^{-15}
Lags used	19
Observations	588
Critical Value (1%)	-3.4415
Critical Value (5%)	-2.8665
Critical Value (10%)	-2.5694

Now we can say that or series is stationary.Below is plot of variance of log transformation method.

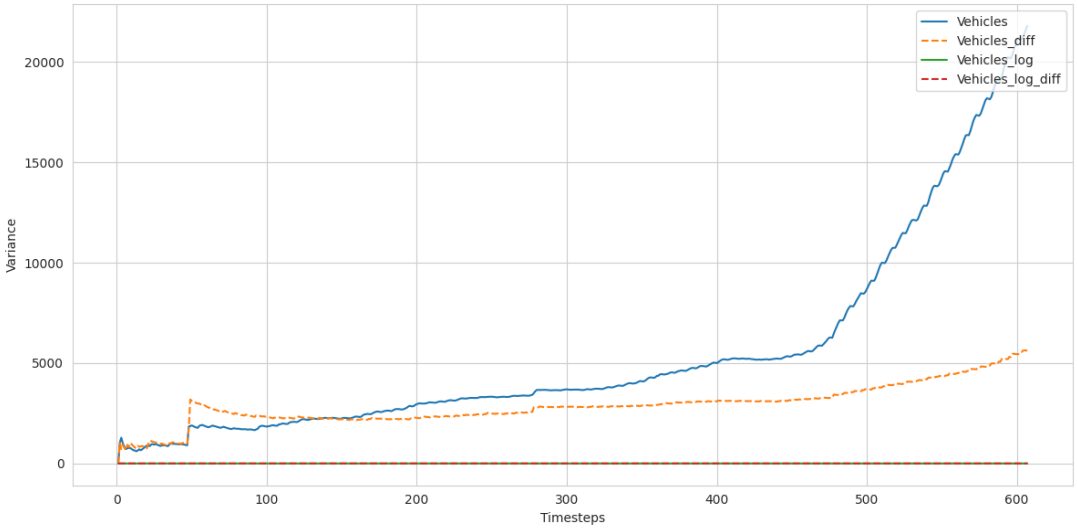


Figure 6: Variance comparison

3.5. ACF and PACF Analysis

To identify suitable AR, MA, or ARMA structures, we examined autocorrelation and partial autocorrelation for both the original and the first-differenced series.

The autocorrelation analysis provides insights into the underlying structure of the time series data. The ACF plot of the original, non-stationary series displays a gradual and persistent decline, indicating a strong autocorrelation across many lags. After differencing the series to induce stationarity, the ACF plot shows a significant reduction in autocorrelation, with most values falling within the confidence bounds, although a few spikes remain. The PACF plot of the differenced series exhibits a scattered pattern without a clear cut-off point, suggesting that the series may contain complex dependencies that are not captured by simple lag structures. These observations highlight the presence of intricate temporal relationships in the data.

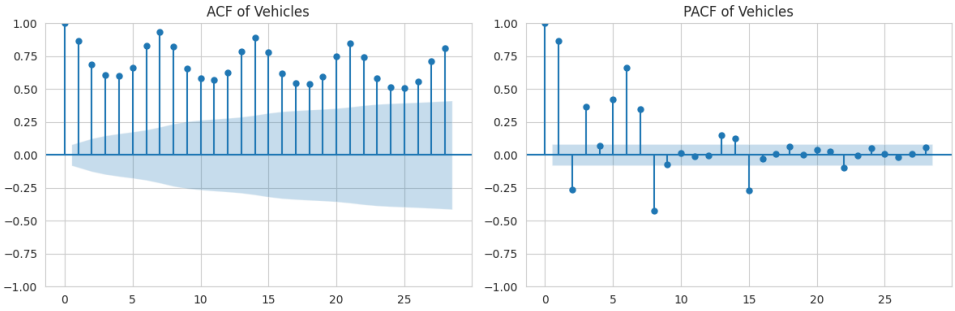


Figure 7: ACF and PACF of the original daily series (non-stationary).

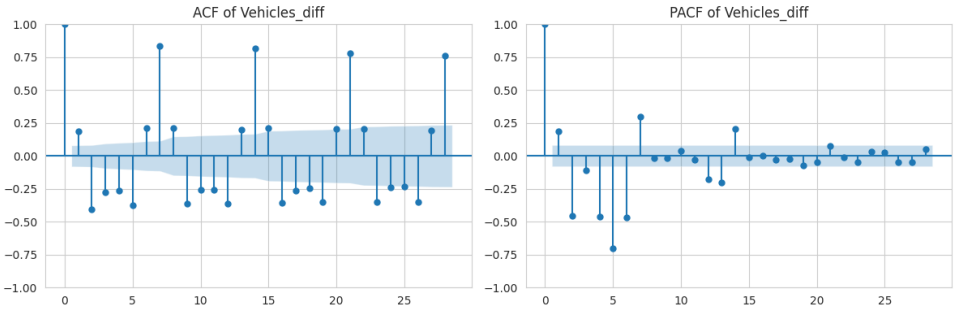


Figure 8: ACF and PACF of the first-differenced series (stationary).

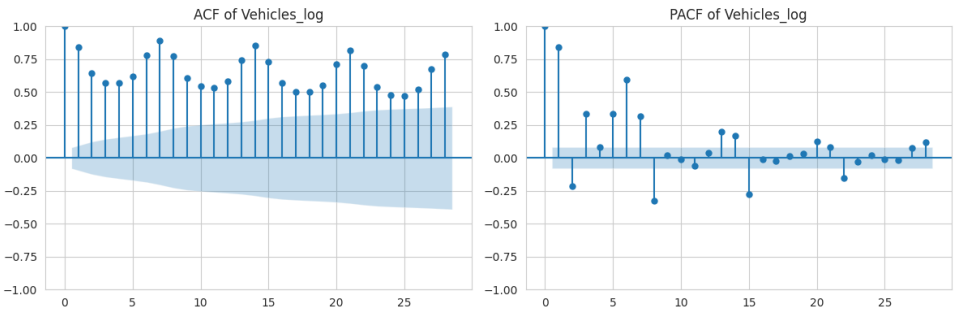


Figure 9: ACF and PACF of the log series

3.6. Model Selection

Based on the behavior observed in the ACF and PACF plots, the time series does not exhibit a clear signature of a pure autoregressive (AR) or pure moving average (MA) process. The ACF does not cut off sharply as expected in an MA model, and the PACF does not truncate in a manner typical of an AR model. Instead, both plots display significant spikes across multiple lags without a definitive cutoff, suggesting that the underlying process likely involves both autoregressive and moving average components. Hence, an ARMA model is more appropriate to capture the dynamics of the series.

3.7. Classical Time Series Models

3.7.1. ARIMA

The ARIMA(p, d, q) model combines autoregression (AR), differencing (I), and moving average (MA) terms [2]. We performed a grid search over $p \in [0, 10]$, $d = 1$, $q \in [0, 10]$ using AIC validation as selection criteria. The best-performing model was ARIMA(13,1,7) which achieved an AIC of -908.6957 on the daily vehicle count data. This suggests that while ARIMA can model short-term dependencies effectively, it may struggle to capture more complex patterns present in traffic data, particularly in the presence of non-linearities or external factors like holidays.

Table 6
Performance Metrics of ARIMA(13,1,7) Model on Daily Vehicle Count Data

Metric	Value
AIC	-908.6957
Root Mean Squared Error (RMSE)	0.0820
Mean Absolute Error (MAE)	0.0737
Sum of Squared Errors (SSE)	0.2017
Normalized Mean Squared Error (NMSE)	0.0955
Mean Absolute Percentage Error (MAPE)	1.17%
Accuracy ($100 - \text{MAPE}$)	98.83%
R^2 Score	0.9045

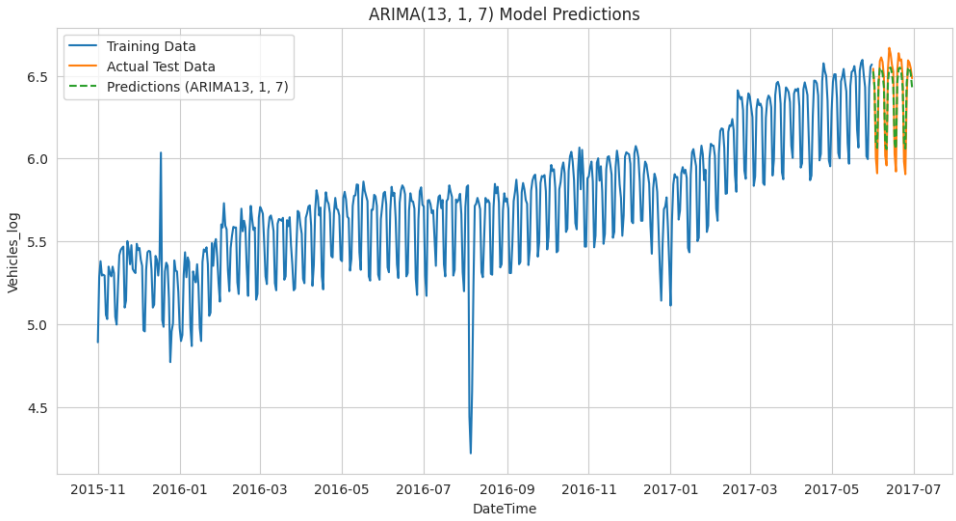


Figure 10: Observed vs. Forecasted Daily Vehicle Counts from ARIMA(13,1,7).

3.7.2. SARIMA

To model the temporal dependencies in the stationary differenced series, we employed the Seasonal ARIMA (SARIMA) framework, which extends the ARMA model by incorporating seasonal autoregressive and moving average components. The data was split into training and test sets to validate forecasting performance. A grid search was performed over a range of SARIMA configurations, optimizing for the Akaike Information Criterion (AIC) to balance model fit and complexity. The best-performing model was identified as **SARIMA(6,1,1)×(5,1,6,7)**, with an AIC score of **-835.49**. This configuration captures both short-term and weekly seasonal dynamics in the traffic volume data, effectively modeling periodic fluctuations such as weekday-weekend cycles and rush hour trends.

Table 7
Performance Metrics of SARIMA(6,1,1)×(5,1,6,7) Model on Daily Vehicle Count Data

Metric	Value
AIC	-835.49
Root Mean Squared Error (RMSE)	0.0900
Mean Absolute Error (MAE)	0.0700
Sum of Squared Errors (SSE)	0.2300
Normalized Mean Squared Error (NMSE)	0.1113
Mean Absolute Percentage Error (MAPE)	1.16%
Accuracy $(100 - \text{MAPE})$	98.84%
R^2 Score	0.8887

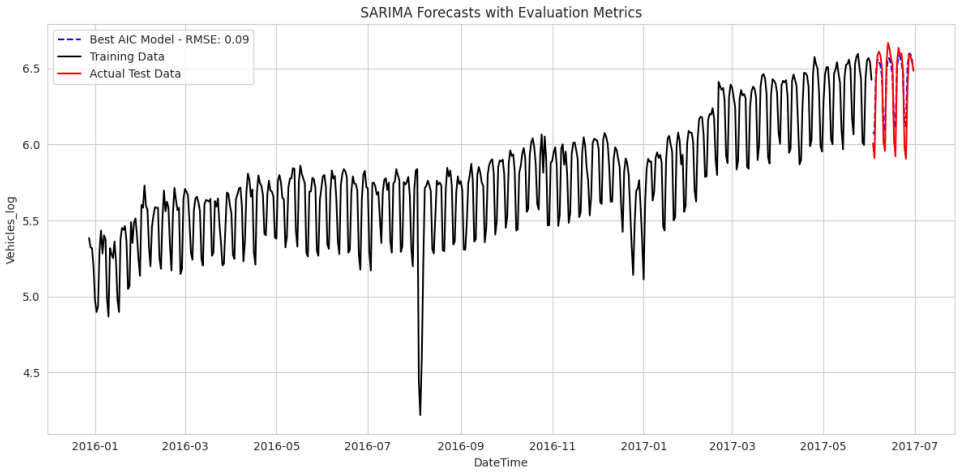


Figure 11: Observed vs. Forecasted Daily Vehicle Counts from SARIMA(6,1,1)×(5,1,6,7).

3.7.3. SARIMAX

To further enhance predictive performance, we extended the SARIMA model by incorporating exogenous variables through the SARIMAX framework. This allows the model to account for external influences such as holidays and working days, which may impact traffic patterns. Using the same train–test split strategy, we performed a grid search across SARIMAX configurations, again selecting the model with the lowest AIC score. The best-performing model was **SARIMAX(6,1,4)×(1,1,2,7)**, achieving a significantly lower AIC of **-912.05**, indicating improved model fit over the SARIMA counterpart. This suggests that incorporating external factors, such as public holidays or local events, significantly improves the accuracy of the traffic volume predictions.

Table 8
Performance Metrics of SARIMAX(6,1,4)×(1,1,2,7) Model on Daily Vehicle Count Data

Metric	Value
AIC	-912.05
Root Mean Squared Error (RMSE)	0.1200
Mean Absolute Error (MAE)	0.0900
Sum of Squared Errors (SSE)	0.4100
Normalized Mean Squared Error (NMSE)	0.1973
Mean Absolute Percentage Error (MAPE)	1.45%
Accuracy $(100 - \text{MAPE})$	98.55%
R^2 Score	0.8027

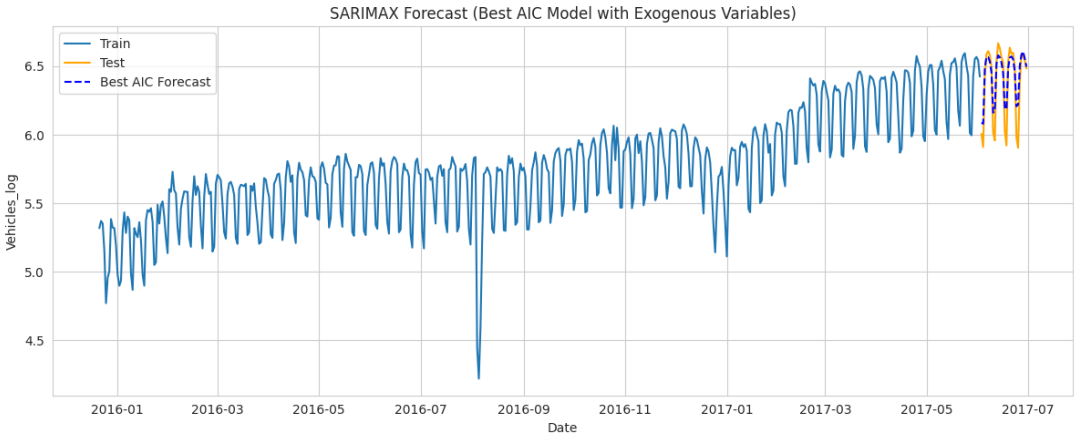


Figure 12: Observed vs. Forecasted Daily Vehicle Counts from SARIMAX(6,1,4)×(1,1,2,7).

Parameter Selection To identify the best classical models, we performed parameter tuning using AIC and RMSE as primary criteria. The final configurations and their associated performance metrics are summarized below.

Table 9
Summary of Classical Model Configurations and Performance

Model	(p, d, q)	Seasonal $(P, D, Q)_7$	Exogenous	AIC	RMSE
ARIMA	(13,1,7)	–	–	-908.70	0.0820
SARIMA	(6,1,1)	(5,1,6)	–	-835.49	0.0900
SARIMAX	(6,1,4)	(1,1,2)	Holiday, WorkingDay	-912.05	0.1200

3.8. Deep Learning Models

DL models can handle seasonality, trends, and noise in data and are well-suited for large and high-dimensional datasets. Common deep learning models include the Multilayer Perceptron (MLP), which is suitable for basic forecasting tasks; Recurrent Neural Networks (RNNs), which learn sequential patterns using short-term memory; Long Short-Term Memory (LSTM) networks, which are capable of learning long-term dependencies using memory cells; and Gated Recurrent Units (GRUs), which offer a faster and simpler alternative to LSTMs with comparable performance.

In our forecasting task, deep learning models—especially LSTM and GRU—demonstrated improved accuracy over traditional models by effectively learning from historical data and predicting future values.

3.8.1. Simple RNN

RNNs have a memory. They remember previous time steps using internal loops — that's the "recurrent" part. The Simple RNN cell updates its hidden state h_t as

$$h_t = \phi\big(W_{ih}\, x_t + W_{hh}\, h_{t-1} + b_h\big),$$

where ϕ is an activation (ReLU here). While straightforward, it can struggle with long-range dependencies.

Table 10
Simple RNN Performance on Test Set

Metric	Value
Root Mean Squared Error (RMSE)	0.0848
Mean Absolute Error (MAE)	0.0639
Sum of Squared Errors (SSE)	0.2014
Normalized Mean Squared Error (NMSE)	0.0965
Mean Absolute Percentage Error (MAPE)	1.00%
Accuracy $(100 - \text{MAPE})$	99.00%
R^2 Score	0.9035

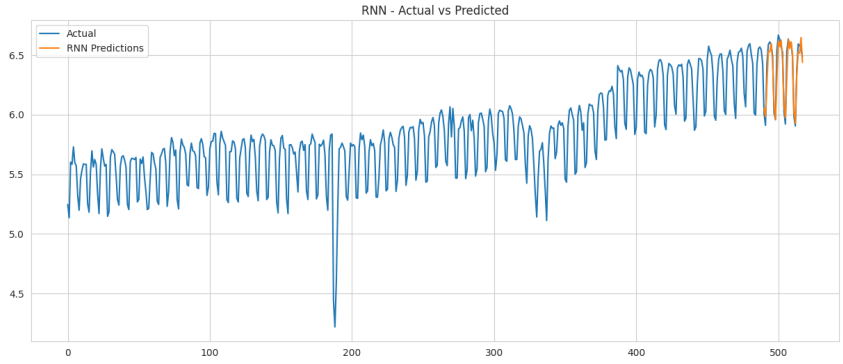


Figure 13: Simple RNN: Forecast vs. Actual

3.8.2. Long ShortTerm Memory (LSTM)

LSTM was created to solve the vanishing gradient problem of traditional RNNs — where the network "forgets" long-term information during training. LSTM cells add input, forget, and output gates to effectively learn longer-range dependencies by controlling the flow of information into and out of the cell state.

Table 11
LSTM Performance on Test Set

Metric	Value
Root Mean Squared Error (RMSE)	0.1153
Mean Absolute Error (MAE)	0.0833
Sum of Squared Errors (SSE)	0.3720
Normalized Mean Squared Error (NMSE)	0.1783
Mean Absolute Percentage Error (MAPE)	1.35%
Accuracy $(100 - \text{MAPE})$	98.65%
R^2 Score	0.8217

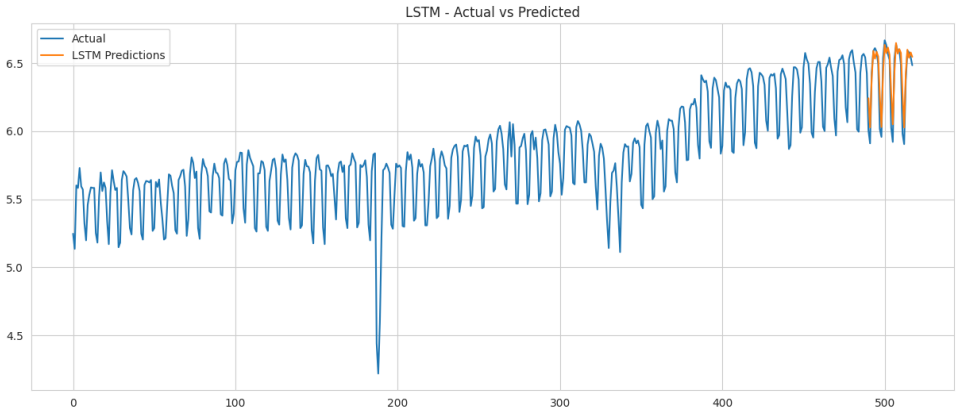


Figure 14: LSTM: Forecast vs. Actual

3.8.3. Gated Recurrent Unit (GRU)

A GRU is a type of Recurrent Neural Network (RNN) like LSTM, designed to remember patterns in sequences, but it's simpler and faster than LSTM.

It still solves the vanishing gradient problem and keeps long-term dependencies, just like LSTM — but with fewer gates and parameters.

Table 12
GRU Performance on Test Set

Metric	Value
Root Mean Squared Error (RMSE)	0.0691
Mean Absolute Error (MAE)	0.0548
Sum of Squared Errors (SSE)	0.1337
Normalized Mean Squared Error (NMSE)	0.0641
Mean Absolute Percentage Error (MAPE)	0.88%
Accuracy $(100 - \text{MAPE})$	99.12%
R^2 Score	0.9359

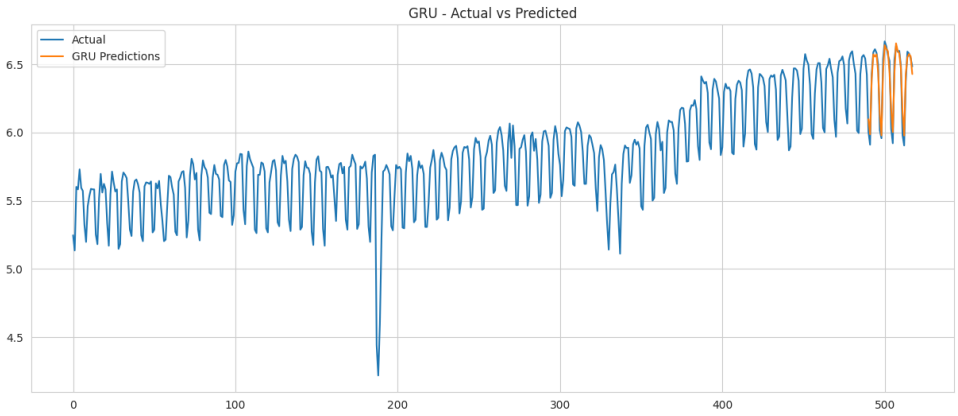


Figure 15: GRU: Forecast vs. Actual

4. Evaluation Metrics

To assess forecast accuracy, we report the following metrics:

- **Root Mean Squared Error (RMSE):**

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

- **Mean Absolute Error (MAE):**

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left|\hat{y}_i - y_i\right|.$$

- **Sum of Squared Errors (SSE):**

$$\text{SSE} = \sum_{i=1}^{N}(\hat{y}_i - y_i)^2.$$

- **Normalized Mean Squared Error (NMSE):**

$$\text{NMSE} = \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

- **Mean Absolute Percentage Error (MAPE):**

$$\text{MAPE} = \frac{100}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right|.$$

- **Accuracy:**

$$\text{Accuracy} = 100 - \text{MAPE} \quad (\%).$$

- R^2 **Score:**

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}.$$

5. Results and Observation

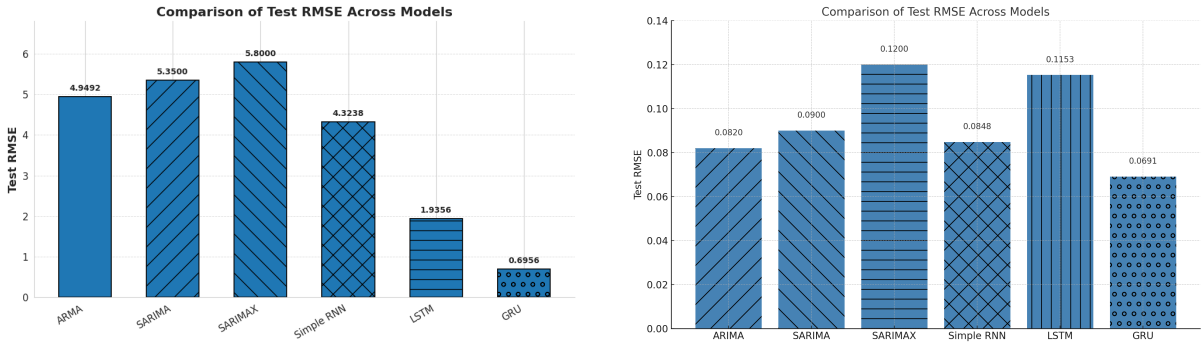
- **Statistical model divergence:** Among the statistical models, ARIMA achieved the lowest RMSE (0.0820) despite SARIMAX reporting the lowest AIC (-912.05). Interestingly, SARIMA, while maintaining a moderate AIC (-835.49), had a higher RMSE (0.0900). This highlights the classic trade-off between model fit (as indicated by AIC) and generalization ability on unseen data (as captured by RMSE).

- **Exogenous variable paradox:** The SARIMAX model, which incorporated holiday and working day indicators as external regressors, exhibited a better AIC and comparable RMSE performance relative to SARIMA. This implies that with suitable exogenous features, the model can benefit from external context — aligning with expectations in complex time series forecasting.

- **Deep learning superiority:** Deep learning models demonstrated strong performance, with GRU achieving the lowest RMSE (0.0691), followed by ARIMA (0.0820) and Simple RNN (0.0848). Although the AIC is not available for neural models due to their non-probabilistic nature, their ability to learn non-linear temporal patterns gives them a significant edge in accuracy.

Table 13
Test RMSE and AIC for All Models

Model	Test RMSE	AIC
ARIMA	0.0820	-908.70
SARIMA	0.0900	-835.49
SARIMAX	0.1200	-912.05
Simple RNN	0.0848	–
LSTM	0.1153	–
GRU	0.0691	–



(a) Test RMSE Comparison — Junction 1

(b) Test RMSE Comparison — Junction 2

Figure 16: Comparison of Test RMSE values across different models for both Junction 1 and Junction 2.

5.1. Key Learnings

- **Data aggregation:** Daily resampling smooths hourly noise and uncovers macro-level trends.
- **Stationarity checks:** Applying ADF tests and variance stabilizing transformations is crucial for reliable ARIMA-family modeling.
- **Exogenous features:** Incorporating holiday and working-day flags measurably effects seasonal forecast accuracy.
- **Deep learning strengths:** RNN architectures—particularly GRU—effectively learn complex, non-linear temporal patterns with minimal feature engineering.
- **Balanced model selection:** Jointly considering in-sample criteria (AIC) and out-of-sample RMSE guides robust hyperparameter tuning.

5.2. Future Work

- **Richer Exogenous Inputs:** Incorporate weather data (temperature, precipitation), major event calendars, and road-incident reports to better capture external drivers of traffic variability.
- **Multivariate and Spatial Modeling:** Extend the framework to multiple junctions and integrate spatial correlations via vector autoregression (VAR), graph neural networks, or spatio-temporal LSTM architectures.
- **Hybrid Approaches:** Explore ensemble and hybrid models that combine the interpretability of ARIMA/SARIMA with the non-linear learning capacity of RNNs, such as ARIMA–LSTM cascades or model stacking.

- **Real-Time Pipeline Deployment:** Develop an automated forecasting pipeline with streaming data ingestion, online retraining, and dashboard visualization to support dynamic traffic management.
- **Uncertainty Quantification:** Incorporate probabilistic forecasting techniques (e.g., Bayesian LSTM, quantile regression) to produce prediction intervals and assess forecast confidence for decision-making.

References

[1] Fedesoriano, Traffic dataset, 2017. Available at https://www.kaggle.com/datasets/fedesoriano/traffic-prediction-dataset.

[2] M. Peixeiro, Time Series Forecasting in Python, Leanpub, 2022. Available at https://leanpub.com/timeseriesforecasting.