Explainable AI for high-risk applications: comparative analysis of traffic forecasting models in smart cities[⋆]

Anastasiya Doroshenko [*†] and Dmytro Savchuk[†]

Lviv Polytechnic National University, Stepana Bandery Street, 12, Lviv, Lviv region, 79000

Abstract

Accurate short-term forecasting of traffic parameters is crucial for improving traffic management and reducing the risk of accidents. This study conducted a comprehensive comparison of six approaches to time series forecasting. These include moving average, ARIMA (2,0,3), seasonal SARIMA (2,0,3) × (2,1,3,24), exponential smoothing (Holt–Winters), Kalman filter, and long short-term memory (LSTM) neural network. The study uses the PEMS08 dataset, which contains 5-minute time intervals with measured traffic flow, occupancy, and speed. Each model is trained on historical data and evaluated on a delayed test set using root mean square error (RMSE), mean absolute error (MAE), Spearman's rank correlation coefficient (ρ), and coefficient of determination (R^2).

The LSTM model used in the study has two LSTM layers, dropout regularization, and dense output layers, outperforming all baseline models, achieving a test RMSE of 0.0195, MAE of 0.00767, Spearman's ρ of 0.81364, and R^2 of 0.80810. These results demonstrate the advantage of LSTM in capturing complex nonlinear and long-term dependencies inherent in traffic dynamics. Our findings highlight the potential of deep learning for real-time traffic forecasting and accident prevention and show the way for future improvements using hybrid and ensemble methods.

Keywords

Road traffic, data, feature, dataset, prediction, Moving Average, ARIMA, SARIMA, Exponential Smoothing, Kalman Filter, neural network, RNN, LSTM, RMSE, MAE, Spearman correlation, R^2. [1]

1. Introduction

Modern smart cities face constant growth in traffic flow, which leads to an increase in traffic accidents and congestion. Accurate forecasting of traffic parameters, namely flow, occupancy, and average speed, can not only prevent accidents but also optimize infrastructure management in real time [1], [2].

One of the key challenges in applying artificial intelligence systems for traffic management in smart cities [3] is that even minor prediction errors can have serious consequences for the safety of road users. Inaccurate forecasting of traffic intensity or lane occupancy may lead to inefficient traffic light control or incorrect information messages, which in turn can cause congestion and hazardous situations. In this context, the accuracy of models remains critically important, but today this is not sufficient: traffic management systems must be not only accurate but also explainable. This necessity is driven by both technical and socio-legal factors [4], [5].

The use of explainable artificial intelligence (XAI) methods ensures transparency of the modeling process and provides the ability to analyze the causes of erroneous forecasts in case of incidents [6]. This contributes to building trust among users, traffic system operators, and regulatory bodies [7]. Moreover, XAI enables the implementation of the human-in-the-loop principle, which ensures human oversight over critical safety-related decisions [8].

This issue becomes particularly significant in the context of regulatory requirements. The EU Artificial Intelligence Act (AI Act, 2024/1689) classifies traffic management systems as high-risk, for which explainability, transparency, and auditability of algorithms are mandatory [9]. Therefore, the development of models that can balance high predictive accuracy with interpretability is of

[⋆] *CIAW-2025: Computational Intelligence Application Workshop, September 26-27, 2025, Lviv, Ukraine*
[1*] Corresponding author.
[†] These authors contributed equally.

anaastasiia.v.doroshenko@lpnu.ua (A. Doroshenko); dmytro.y.savchuk@lpnu.ua (D. Savchuk)
0000-0002-7214-5108 (A. Doroshenko); 0009-0006-0937-6161 (D. Savchuk)

fundamental importance for the practical deployment of intelligent traffic management systems in smart cities.

The goal of this work is to create an effective system for short-term forecasting of key traffic parameters, namely: traffic intensity, lane congestion, and average speed. This is done based on real five-minute measurements from the PEMS08 detector network. Such forecasts make it possible to predict peak loads, identify critical situations, and adjust transport infrastructure management in a timely manner, which directly contributes to reducing the number and severity of road accidents.

In modern "smart" cities, the growth of traffic flow leads to congestion and accidents. The main task is to use historical 5-minute sensor data (PEMS08) [10] to predict future traffic flow, occupancy, and speed. This will enable preventive measures to be implemented, optimize infrastructure use in real time, and reduce the number and severity of road accidents.

The main problems encountered in solving this issue include: the limitations of linear models in capturing deep nonlinearities and long-term correlations, instability due to gaps or anomalies in the data (sensor errors, sharp jumps), seasonal cyclicality with a 24-hour period, which requires special models or additional differentiations, the complexity of balancing model adaptability and avoiding overfitting on peak or rare events.

The main approaches to solving this problem can be divided into four groups:

- statistical models (ARIMA, SARIMA, Theta, STL, ETS),
- classical machine learning-based models (Random Forest, XGBoost, GBM),
- specialized deep models (CNN, GNN, STGCN, DCRNN, Graph WaveNet),
- hybrid-ensemble approaches (ARIMA-NN, Kalman Filter + RNN).

Six approaches to time series analysis and forecasting were selected for the study.

1. The first is a simple moving average, which serves as a baseline, providing an initial idea of average trends.
2. Next, classic linear models are applied: ARIMA (p, d, q), which combines autoregression, differentiation, and moving average.
3. Seasonal extension SARIMA (p, d, q) × (P, D, Q, m) with a daily lag m = 24 (hours), which allows for regular traffic cyclicality.
4. Exponential smoothing according to Holt-Winters provides an adaptive mechanism for determining the level, trend, and multiplicative seasonality without explicit use of differentiation.
5. An alternative to statistical methods is the one-dimensional Kalman filter, which combines a priori assumptions about dynamics with constant updating of the system state in the presence of measurement noise.
6. Finally, in comparison with traditional models, a powerful deep RNN architecture based on LSTM is investigated, capable of capturing complex nonlinear and long-term dependencies.

In this study, we consciously selected a broad spectrum of models for experiments: from classical statistical approaches (Moving Average, ARIMA (2, 0, 3), SARIMA (2, 0, 3) × (2, 1, 3, 24), Holt–Winters exponential smoothing) to methods capable of handling noisy and incomplete data (Kalman filter), as well as modern deep learning architectures (LSTM). Special attention was also paid to hybrid and ensemble methods (e.g., ARIMA and NN, Kalman Filter and RNN), which combine the interpretability of statistical approaches with the predictive power of neural networks.

This strategy makes it possible not only to perform a quantitative comparison of forecasting accuracy but also to assess the potential for explainability of each method, which represents an important step toward the development of safe and regulation-compliant traffic management systems.

2. Data preparation

2.1. Initial Dataset

First, it is necessary to conduct a review of the dataset that will be used in the work. So, the **PEMS** [10] dataset was used. All data sets are collected by the California Department of Transportation Performance Measurement System in real time every 30 seconds. The collected data is then aggregated to 5 minutes, which in turn means that there are 12 points in the stream data for each hour.

This dataset includes the following [11]:

1. **PEMS03.** Contains statistical data on traffic flows for the three months from September 1, 2018, to November 30, 2018, including 358 sensors. The total number of observed traffic data points is 26208.
2. **PEMS04.** Contains statistical data on traffic flow, speed, and traffic occupancy for two months from January 1, 2018, to February 28, 2018, including 307 sensors. The total number of observed traffic data points is 16992.
3. **PEMS07.** Contains three months of traffic flow statistics for the period from May 1, 2017, to August 31, 2017, including 883 sensors. The total number of observation traffic flow data points is 28224.
4. **PEMS08.** Contains three months of traffic flow, speed, and road congestion statistics for the period from July 1, 2016, to August 31, 2016, including 170 sensors. The total number of observed traffic flow data points is 17856.
5. **PEMS(BAY).** Contains six months of traffic speed statistics from January 1, 2017, to May 31, 2017, including 325 sensors in the Bay Area. The total number of observed traffic data points is 16937179.

Thus, in our case, the PEMS-08 dataset was used. This dataset includes three characteristics: traffic flow, traffic occupancy, and traffic speed. Detailed information about the characteristics is provided below:

1. The **"flow"** feature in the PEMS08 dataset represents the number of vehicles passing through a loop detector over a given period (in our case, 5 minutes). It is measured in vehicles per 5-minute interval.
2. The **"occupancy"** feature represents the proportion of time during a time interval (5 minutes) during which the detector was occupied by a vehicle. It is measured in percentage.
3. The **"speed"** feature represents the average speed of vehicles passing through the loop detector during a time interval (5 minutes). It is measured in miles per hour (mph).

So, let's look at the features of our dataset in more detail Table 1:

Table 1
Source Dataset

Name	Type	Description
Lane N Flow	Veh/5-min	Number of vehicles passing through a loop detector over a 5 min period.
Lane N Occupancy	%	Proportion of time during which the detector was occupied by a vehicle.
Lane N Speed	mph	Average speed of vehicles passing through the loop detector during a 5 min interval.

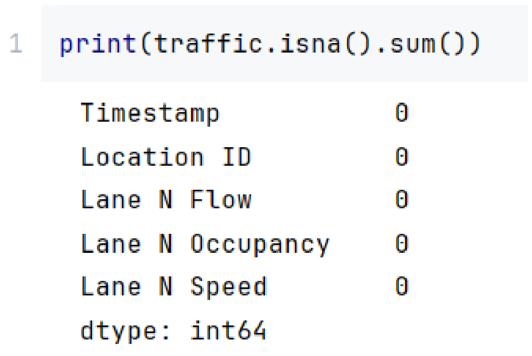
After reviewing the data set, it is necessary to process it. Specifically, checking for missing values, identifying outliers using interquartile range and Z-scores, and analyzing the distribution and temporal patterns for each of the three key indicators.

The histograms of flow, occupancy, and speed showed significant asymmetry and the presence of isolated extreme jumps, which can distort the results of models that are sensitive to scale and assumptions about normal distribution. Time series confirmed the presence of a daily cycle with an interval of approximately 24 hours and random jumps, indicating both regular "morning" and "evening" peaks and sudden events (accidents, repairs, sensor errors).

2.2. Check for Null values

The first step in examining a dataset is to check for null values. Missing values can distort statistical estimates and invalidate further modeling. The following section presents a concise procedure for identifying null records in our traffic dataset.

To do this, we use the pandas library to calculate the total number of null values (NaN) in each column (Figure 1).



```
print(traffic.isna().sum())
```

```
Timestamp           0
Location ID         0
Lane N Flow         0
Lane N Occupancy    0
Lane N Speed        0
dtype: int64
```

Figure 1: Number of missing values for the dataset.

Thus, all columns show zero values for missing data, indicating that there are no blank spaces for our features.

2.3. Researching the features of a dataset

When studying features, it is important to visualize their distribution to identify patterns, outliers, and potential obstacles to modeling. Below are histograms of the values of the features **"Lane N Flow"** (Figure 2), **"Lane N Occupancy"** (Figure 3), and **"Lane N Speed"** (Figure 4) which show the distribution of traffic depending on our feature.

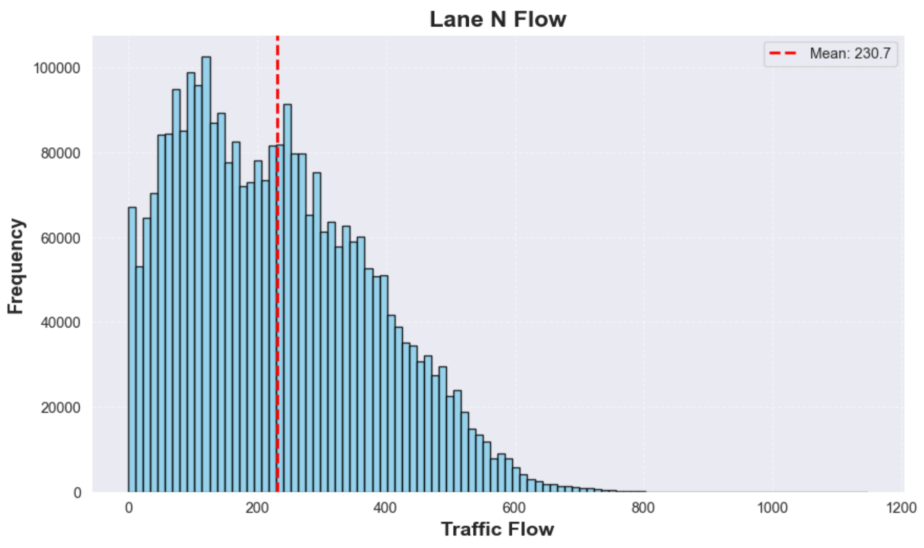


Figure 2: Frequency of traffic flow.

In the figure above, flow values ranging from 0 to 1200 are represented by binary columns of varying heights. The average value, marked by a red dotted line at approximately 230.7, indicates the central tendency, while the elongated right tail indicates rare but significant peak flow loads.

Figure 3 below illustrates the distribution of lane occupancy percentages in the range from 0.0 to 0.8. Here, the light blue bars are concentrated in the lower range, and the average value of about 0.1 confirms the predominantly low occupancy. The right tail of the distribution indicates isolated episodes of high traffic density, which are most likely to occur during peak hours or extraordinary events.

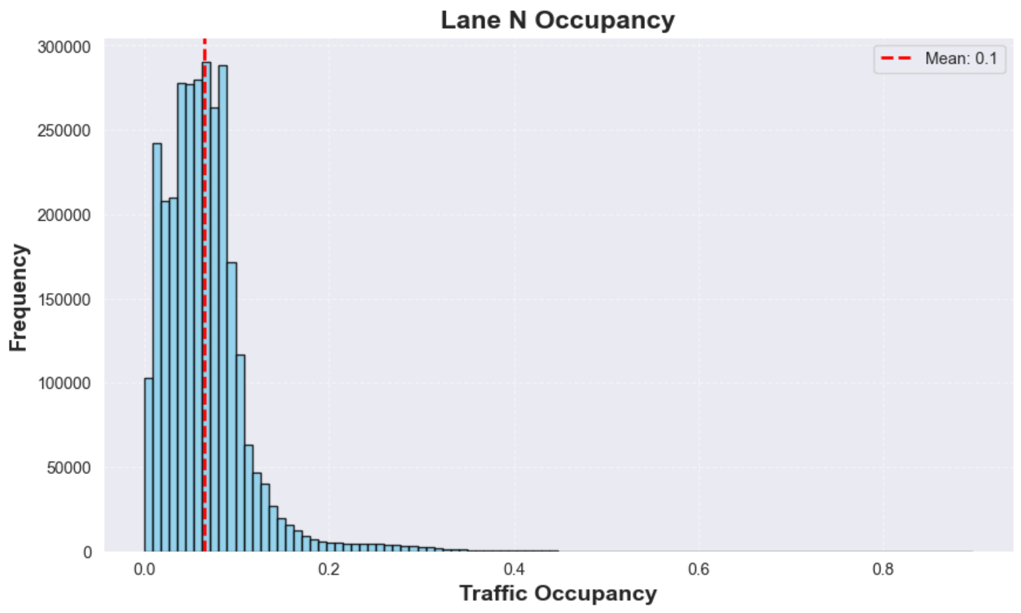


Figure 3: Frequency of traffic occupancy.

The histogram below (Figure 4) illustrates the distribution of vehicle speeds. The X-axis shows speeds ranging from 0 to 80 mph, while the Y-axis shows the frequency of observations up to 350,000. The highest concentration of observations is at speeds between 50 and 70 mph, and the average speed, marked by the red dotted line at 63.8 mph, indicates the central tendency of the data.

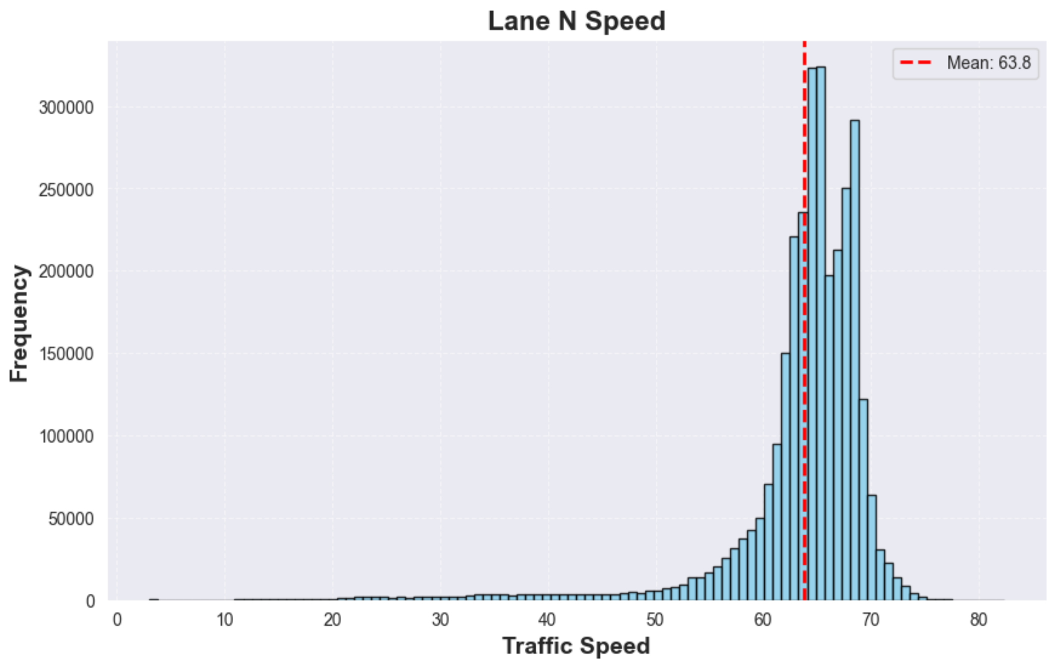


Figure 4: Frequency of traffic speed.

If we look at all the histograms together, we can see a noticeable asymmetry in the distributions and a few outliers that could mess up the results of models that are sensitive to feature scale. For proper further analysis, it's worth thinking about normalizing the features or removing the outliers. This approach will make sure the machine learning algorithms work more consistently and accurately with our dataset.

Next, let's perform the same analysis for our features, but in this case, depending on time (Figures 5, 6, 7).

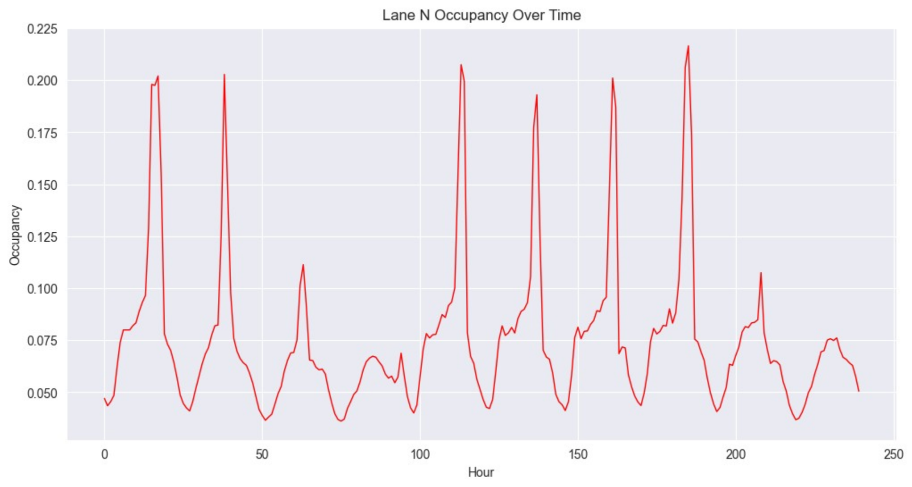


Figure 5: Traffic occupancy overtime.

Figure 5 illustrates the change in lane occupancy over 250 hours. The X-axis shows the time in hours, and the Y-axis shows the time during which the sensor detects the presence of a vehicle in the lane (from 0.05 to 0.225).

From the time series we can clearly see a daily cycle: occupancy peaks reach almost 0.20 approximately every 24 hours, and troughs drop to 0.08 during the night or morning off-peak hours. This periodicity corresponds to typical hourly traffic peaks morning and evening. Isolated jumps above 0.20 may indicate traffic incidents or atypically intense periods of traffic.

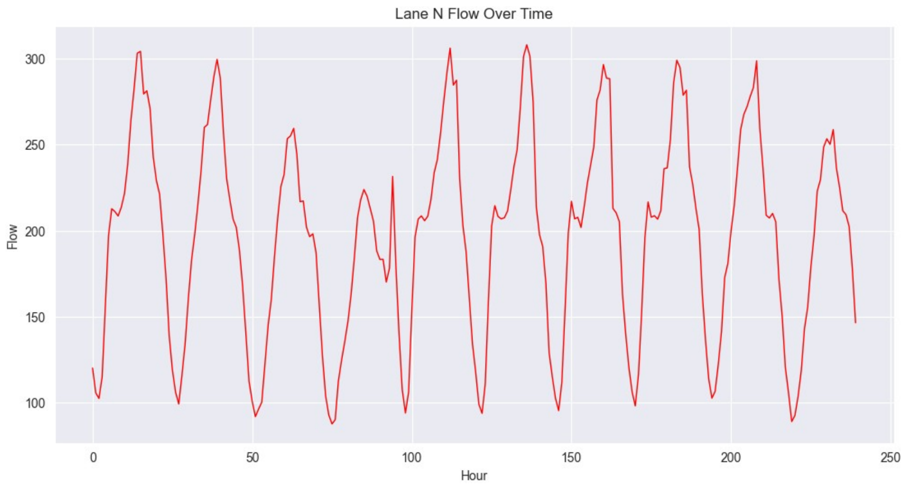


Figure 6: Traffic flow overtime.

The graph above (Figure 6) shows the dynamics of traffic flow intensity over a certain period. The X-axis represents time in hours, and the Y-axis represents traffic flow ranging from 100 to 300. The wave-like nature of the line indicates periodic fluctuations in traffic volume, alternating between relatively high peaks of ~300 and low troughs less than 100.

Analysis of the curve shows stable cycles with an interval of about 24 hours, which corresponds to the daily traffic pattern: morning and evening rush hours create recurring peaks, while nighttime periods are local minimum. The average traffic flow is around 200 units, but sometimes there are abnormal spikes above 300 or sudden drops to 100, which may be caused by extraordinary events, accidents, or meter errors.

Figure 7 shows changes in average traffic flow speed in the range from 0 to 250 hours. The X-axis shows time in hours, and the Y-axis shows speed in the range from 35 to 65 mph. The line is wave like in nature, with periodic fluctuations and noticeable sharp drops and rises. This shape indicates daily repetition of traffic patterns and, at the same time, random disruptions in the road network.

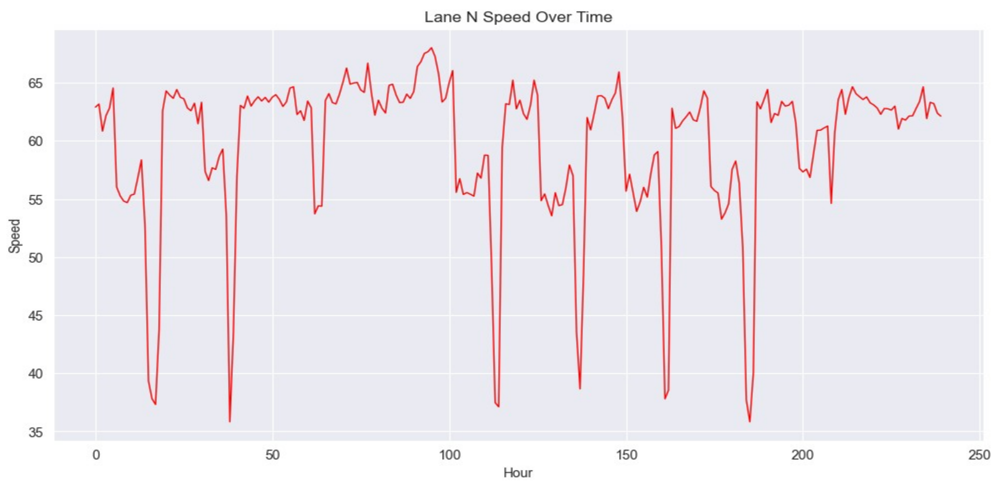


Figure 7: Traffic speed overtime.

The graph clearly shows cycles with an interval of approximately 24 hours, corresponding to morning and evening rush hours with traffic slowdowns. The amplitude of fluctuations reaches over 15 mph when comparing local minimums (around 35 mph) with peak values (60 mph). In addition to regular seasonal changes, there are abnormal jumps above 60 mph and drops to around 35 mph, which may indicate accidents, changing weather conditions, or traffic control system intervention.

2.4. Outlier detection

Traffic data often contains sudden spikes or drops in values caused by sensor errors, equipment malfunctions, or extreme road conditions. If these anomalous observations are not separated from the main dataset, subsequent risk prediction models may distort their estimates and lose stability.

In this section, we will consider an approach based on interquartile range (IQR) analysis [12] (Figure 8), which clearly defines the lower and upper limits of normal values and the distribution of Z-scores (Figure 10) of the flow, where we will mark critical values that should be considered potential outliers.

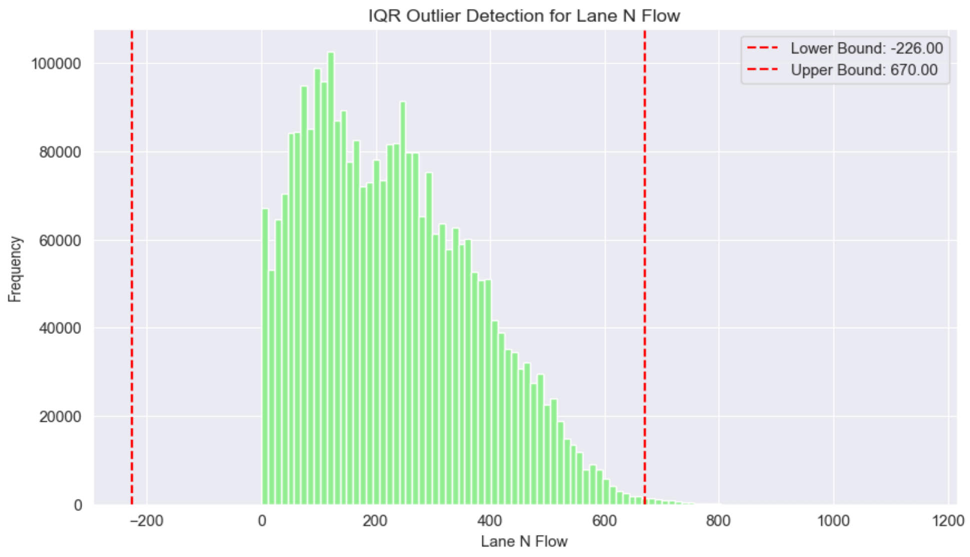


Figure 8: IQR outlier detection.

The distribution histogram (Figure 8) shows the values of Lane N flow in the range from 200 to 1200 units. The horizontal axis shows the measured flow values, while the vertical axis shows their frequency of occurrence in the data, which reaches up to 100000 observations. The green color of the bars emphasizes the accumulation of most values in the range from 0 to 400, indicating the bulk of the data.

On the graph, two vertically dotted red lines outline the rejection limits using the interquartile range method: the lower limit is 226 and the upper limit is 670. Values outside this range will be considered potential outliers. The lower limit takes on negative values due to mathematical calculation, although the actual flow cannot be negative, and the upper limit separates excessively large values.

```python
Q1 = traffic['Lane N Flow'].quantile(0.25)
Q3 = traffic['Lane N Flow'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers_iqr = traffic[(traffic['Lane N Flow'] < lower_bound) | (traffic['Lane N Flow'] > upper_bound)]
print("IQR outliers count for Lane N Flow:", len(outliers_iqr))
```

```
IQR outliers count for Lane N Flow: 7154
```

Figure 9: IQR outlier count.

This approach helps to automatically spot observations that are way different from the main dataset. Any points with a flow rate below 226 or above 670 should be considered outliers. They may indicate measurement errors, equipment malfunctions, or rare extreme situations on the road.

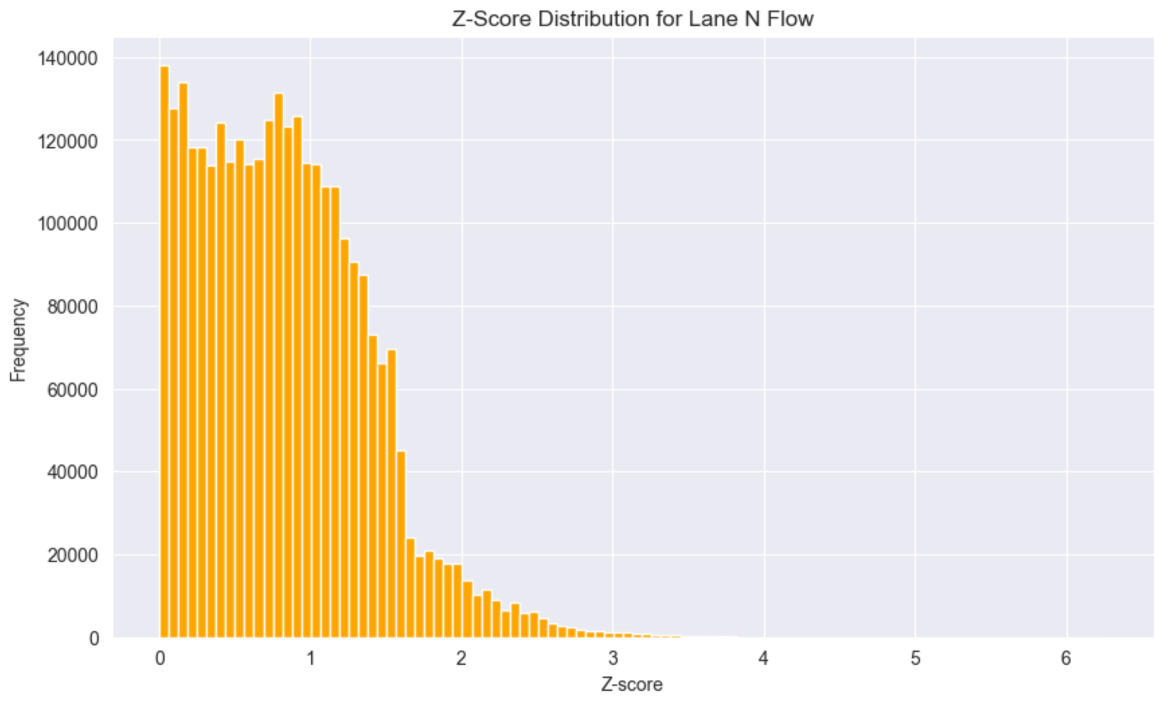
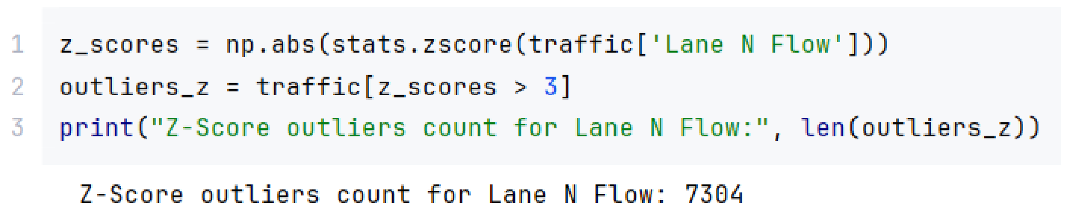


Figure 10: Z − score distribution.

The histogram above shows the distribution of absolute Z-scores in the range from 0 to 6. The horizontal axis shows the Z-scores, which are measures of the deviation of each measurement from the mean in units of standard deviation [13]. The vertical axis shows the frequency, with a maximum of approximately 140000 observations.

The golden color of the bars emphasizes the concentric cluster of the most frequent values near $Z \approx [0 - 1]$, which corresponds to typical deviations within one sigma. Further on, the frequency gradually decreases, forming a long right “tail” to $Z \approx 6$. This distribution pattern indicates a classic normal distribution of residuals with a small number of extreme observations.

The critical threshold for labeling outliers is usually considered to be $Z \geq 3$. In this graph, the bars above this threshold are almost invisible compared to the whole dataset, which emphasizes the rarity of extreme deviations. These isolated observations may be the result of serious road events, measurement inaccuracies, or equipment malfunctions.



```python
z_scores = np.abs(stats.zscore(traffic['Lane N Flow']))
outliers_z = traffic[z_scores > 3]
print("Z-Score outliers count for Lane N Flow:", len(outliers_z))
```

```
Z-Score outliers count for Lane N Flow: 7304
```

Figure 11: Z – score outliers' count.

2.5. Correlation Matrix

In this section, we will explore the interrelationships between traffic features using a correlation matrix. This approach allows us to visualize the strength and direction of linear dependencies between key features of the dataset: time, location, traffic flow, occupancy, and speed. Correlation analysis helps to identify which variables move in unison or in opposite directions, which is critical for optimizing forecasting models and data clustering [14].

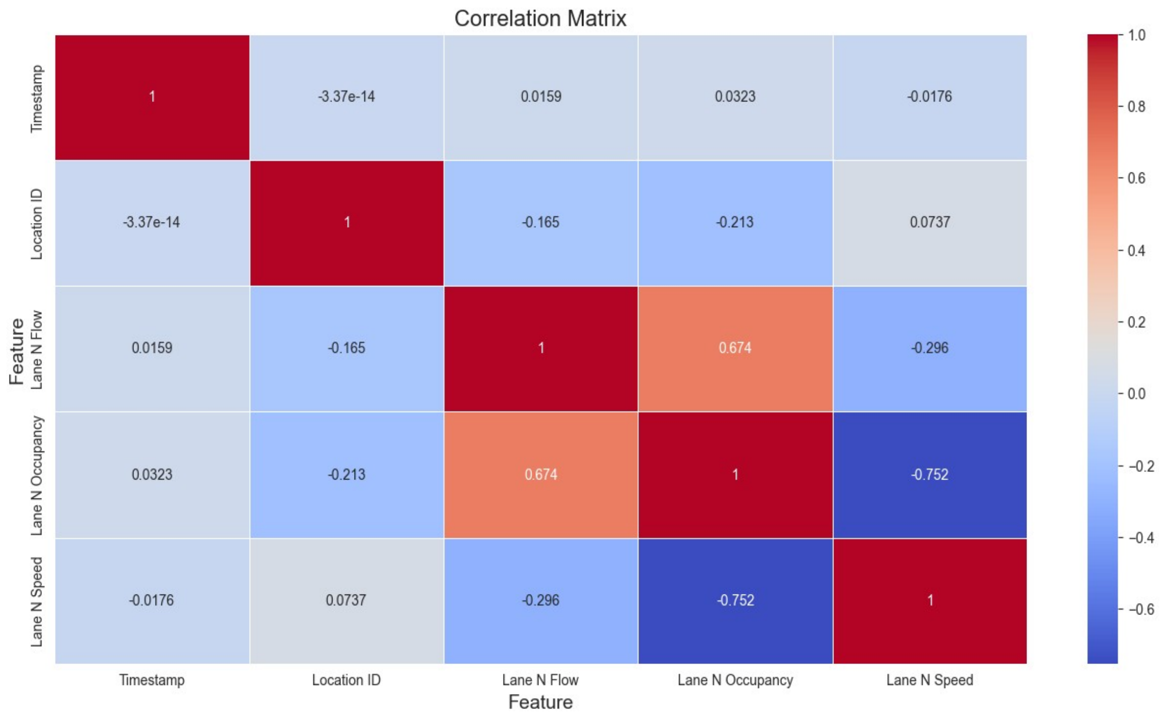


Figure 12: Correlation matrix for traffic features.

The correlation matrix above reflects the strength and direction of the relationships between the key features of the dataset: timestamp, location ID, lane flow, lane occupancy, and average lane speed.

The color scheme ranges from light blue to brown shades, illustrating correlation coefficients from -1 to +1, where warmer colors indicate strong positive relationships and cooler colors indicate negative relationships.

- A zero-correlation value between Timestamp and Location ID indicates that timestamps are independent of specific points in the sensor network.
- In contrast, traffic flow and lane occupancy show a high positive correlation (≈ 0.67), reflecting the pattern that as traffic flow increases, lane occupancy also increases.
- Conversely, lane occupancy and traffic speed are strongly negatively correlated (≈ -0.75), i.e., an increase in lane occupancy leads to a significant decrease in average speed.
- A less pronounced but still negative correlation between flow and speed (≈ -0.30) also confirms that more intense periods of traffic are accompanied by some slowing down.
- The correlations between traffic parameters and Location ID remain weak (maximum 0.21), indicating relatively uniform behavior of indicators in different road sectors.

This distribution of relationships allows us to conclude that clustering or simultaneous modeling of Flow, Occupancy, and Speed in a multidimensional space should be prioritized, leaving time and location characteristics primarily as constant contexts for data aggregation.

3. Machine learning methods

3.1. Model Types

In the current chapter, we will look at five key approaches to time series forecasting, which are shown in Table 2. The first two methods: ARIMA and its seasonal extension SARIMA — are based on linear modeling of autocorrelations and integrated trends and are classics of one-dimensional time series analysis.

The exponential smoothing method offers flexible trend and seasonality extraction through adaptive smoothing coefficients that automatically adjust to changes in the data.

The Kalman filter transfers the task of estimating dynamics to the state space, allowing observations and priori assumptions about wave processes to be combined into a single probabilistic model.

Finally, recurrent neural networks (RNNs), specifically LSTM architectures, demonstrate the ability to capture complex nonlinear dependencies and long-term patterns in data, which is particularly useful when forecasting traffic with numerous influences and variable time intervals.

In the following sections, we will describe in detail the principles of each of these models, their strengths and weaknesses, and the criteria for selecting the most appropriate approach for traffic parameter forecasting tasks.

Table 2
List of used Models

№	Model	Reference
1	ARIMA	[15]
2	SARIMA	[16]
3	Exponential Smoothing	[17]
4	Kalman Filter	[18]
5	RNN (LSTM)	[19]

3.2. ARIMA

In this section, we will take a detailed look at the **ARIMA (Autoregressive Integrated Moving Average)** model, which combines autoregression, integration (differentiation), and moving average for time series modeling [20].

ARIMA (p, d, q) consists of three components that consider autocorrelation, trend, and noise:

1. The autoregressive component AR(p) describes the dependence of the current value on the previous p lags.

$$Y_t = \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + \ldots + \alpha_p Y_{t-p} + e_t, \tag{1}$$

Y_t - *current value*, $\alpha_1, \alpha_2, \ldots, \alpha_p$ - *coefficients*, e_t - *error*

2. The integrated component I(d) brings the series to stationarity by means of d-fold differentiation.

$$\Delta Y_t = Y_t - Y_{t-1}, \tag{2}$$

Δ - *difference operator*

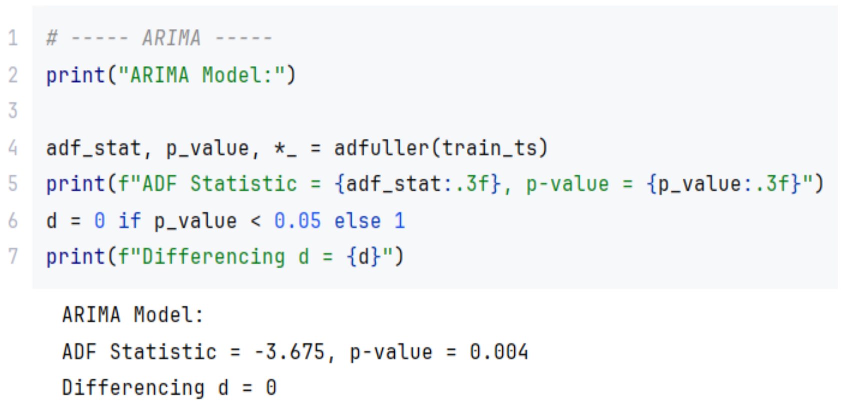
3. The moving average MA(q) models the influence of q previous random errors on the current value.

$$Y_t = \mu + e_t + \Omega_1 e_{t-1} + \Omega_2 e_{t-2} + \ldots + \Omega_p e_{t-q}, \tag{3}$$

μ - *constant*,
e_t - *error at time* t,
$\Omega_1, \Omega_2, \ldots, \Omega_p$ - *moving average coefficients*

The key idea is to apply a linear combination of previous observations and errors after bringing the series to a stationary state.

To determine the degree of differentiation d, a stationarity test is performed, most often the Augmented Dickey-Fuller (ADF) test (Figure 13).



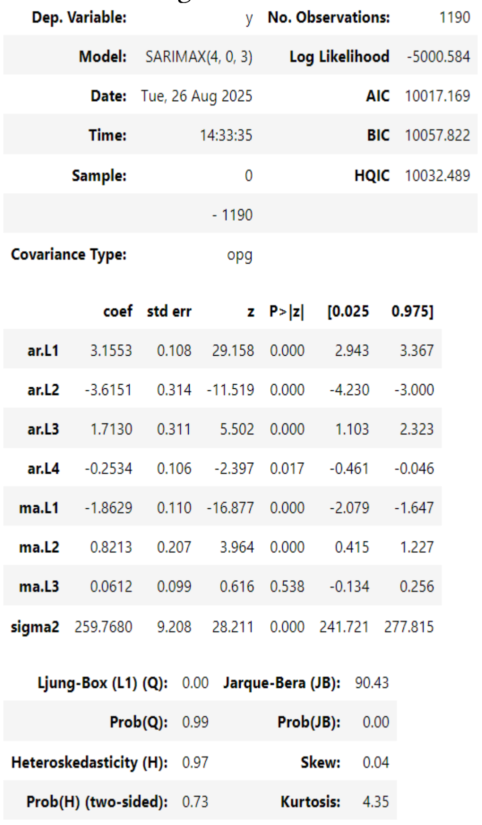
```
# ----- ARIMA -----
print("ARIMA Model:")

adf_stat, p_value, *_ = adfuller(train_ts)
print(f"ADF Statistic = {adf_stat:.3f}, p-value = {p_value:.3f}")
d = 0 if p_value < 0.05 else 1
print(f"Differencing d = {d}")
```

```
ARIMA Model:
ADF Statistic = -3.675, p-value = 0.004
Differencing d = 0
```

Figure 13: ARIMA model.

From the figure above we can see that p-value = 0.004. If p-value < 0.05, the series is considered stationary (d = 0). Otherwise, one differentiation is applied (d = 1).

Figure 14 below shows the results of fitting the ARIMA model for the specified parameters.



Dep. Variable:	y	**No. Observations:**	1190
Model:	SARIMAX(4, 0, 3)	**Log Likelihood**	-5000.584
Date:	Tue, 26 Aug 2025	**AIC**	10017.169
Time:	14:33:35	**BIC**	10057.822
Sample:	0	**HQIC**	10032.489
	- 1190		
Covariance Type:	opg		

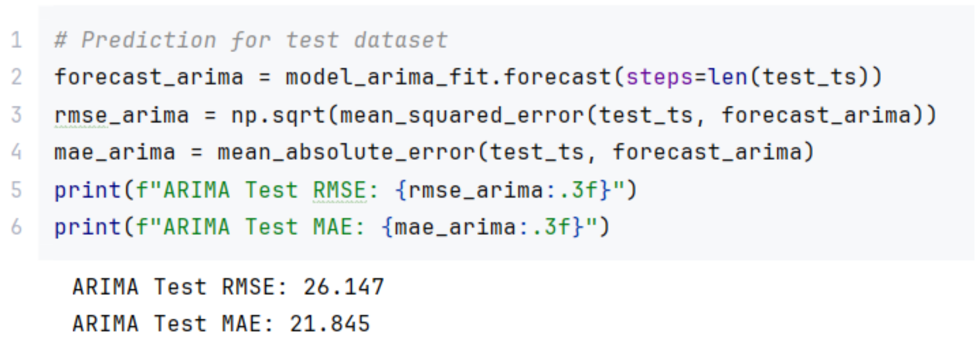
	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
ar.L1	3.1553	0.108	29.158	0.000	2.943	3.367
ar.L2	-3.6151	0.314	-11.519	0.000	-4.230	-3.000
ar.L3	1.7130	0.311	5.502	0.000	1.103	2.323
ar.L4	-0.2534	0.106	-2.397	0.017	-0.461	-0.046
ma.L1	-1.8629	0.110	-16.877	0.000	-2.079	-1.647
ma.L2	0.8213	0.207	3.964	0.000	0.415	1.227
ma.L3	0.0612	0.099	0.616	0.538	-0.134	0.256
sigma2	259.7680	9.208	28.211	0.000	241.721	277.815

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	90.43
Prob(Q):	0.99	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.97	**Skew:**	0.04
Prob(H) (two-sided):	0.73	**Kurtosis:**	4.35

Figure 14: ARIMA settings.

After fitting the ARIMA model to the training sample, a forecast was made for the test interval using the *model_arima_fit.forecast()* method, which allowed us to obtain predicted values for each step of the test series. To quantitatively assess the accuracy of the forecast, two metrics were calculated (Figure 15): **root mean square error (RMSE)** and **mean absolute error (MAE)**.

The RMSE value of 26.147 indicates the root mean square distance between the actual and predicted flow intensity indicators. This metric is sensitive to large deviations and demonstrates how significantly the peak errors of the model can differ. The MAE, which was 21.845, reflects the average absolute deviation without considering the direction of the error.



```python
# Prediction for test dataset
forecast_arima = model_arima_fit.forecast(steps=len(test_ts))
rmse_arima = np.sqrt(mean_squared_error(test_ts, forecast_arima))
mae_arima = mean_absolute_error(test_ts, forecast_arima)
print(f"ARIMA Test RMSE: {rmse_arima:.3f}")
print(f"ARIMA Test MAE: {mae_arima:.3f}")
```

```
ARIMA Test RMSE: 26.147
ARIMA Test MAE: 21.845
```

Figure 15: ARIMA results.

Figure 16 below compares of the actual values of the time series and the forecast obtained by the ARIMA model for the interval from 1200 to 1500 observations. The blue solid line corresponds to the actual data, while the orange dashed line corresponds to the ARIMA forecast.

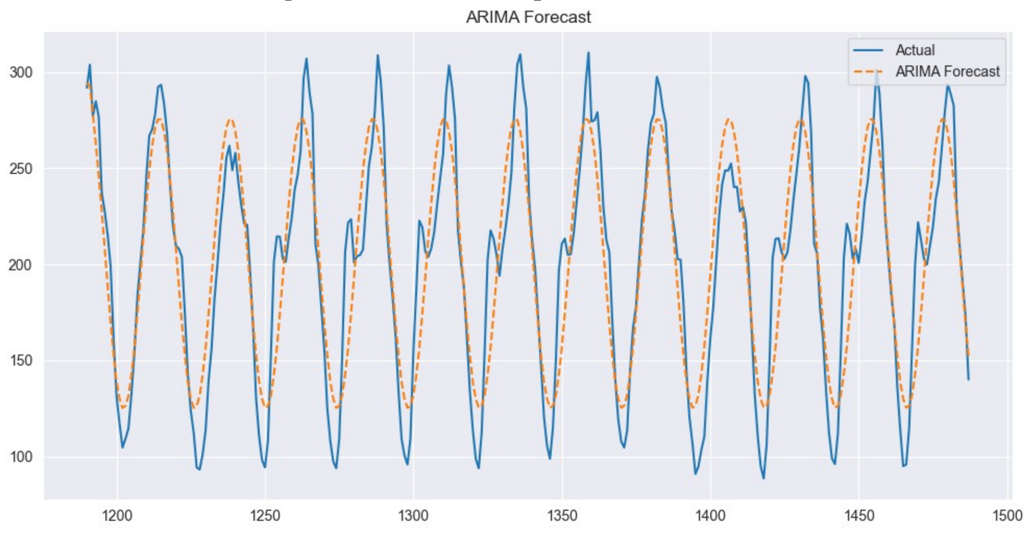


Figure 16: ARIMA forecast.

3.3. SARIMA

The simplest way to extend ARIMA for seasonal data is to apply SARIMA, which considers both irregular and seasonal components [21].

SARIMA is an improved version of the ARIMA model that takes into account seasonal fluctuations in time series. When data has a clear seasonal pattern (for example, with fluctuations on an annual or monthly basis), the standard ARIMA model is unable to work effectively with such patterns. SARIMA adds seasonal components to the ARIMA model, allowing for both nonlinear changes over time and regular seasonal variations to be considered.

$$\Phi_p\left(B^S\right)*\left(1 - B^S\right)^D Y_t = \Omega_q\left(B^S\right)*e_t, \tag{4}$$

$B -$ *lag operator*,

$s -$ *seasonal period length*,

$\Phi_p\left(B^S\right) -$ *seasonal component AR*,

$\Omega_q\left(B^S\right)$ - *seasonal component MA*,

$\left(1 - B^S\right)^D$ - *seasonal difference operation for stationarity*

First, we need to determine the seasonal interval m, which corresponds to the length of our observation cycle. For monthly data - 12, for weekly data - 52, and for hourly data - 24. After that, both regular and seasonal differentiation are performed with orders d and D, respectively, to make the series stationary on both short and long seasonal intervals.

Diagnostics help to select the optimal values of p, q for the irregular part and P, Q for the seasonal part. First, the autocorrelation (ACF) and partial autocorrelation (PACF) (Figure 17) of the original series are estimated to be approximately determined by the determine d and the usual parameters AR(p) and MA(q). Then, after seasonal differentiation, ACF/PACF analysis allows us to understand whether seasonal lags are needed in the autoregression (P) or moving average (Q).

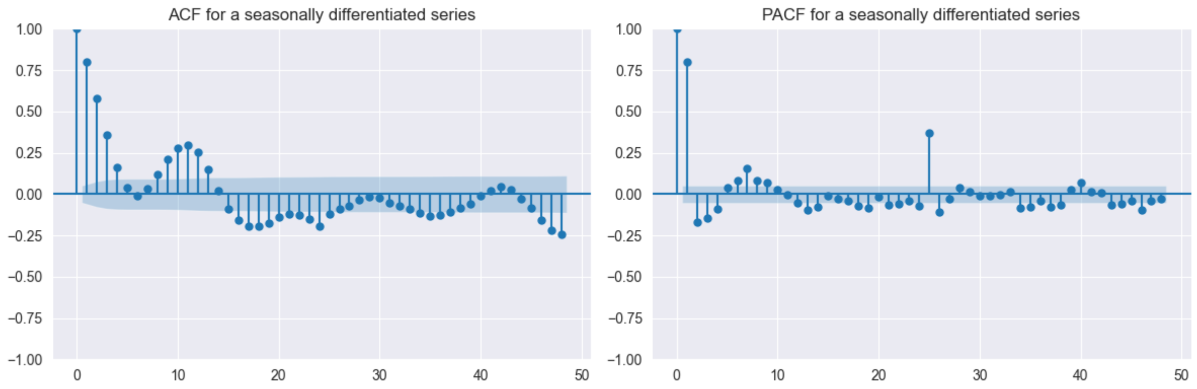
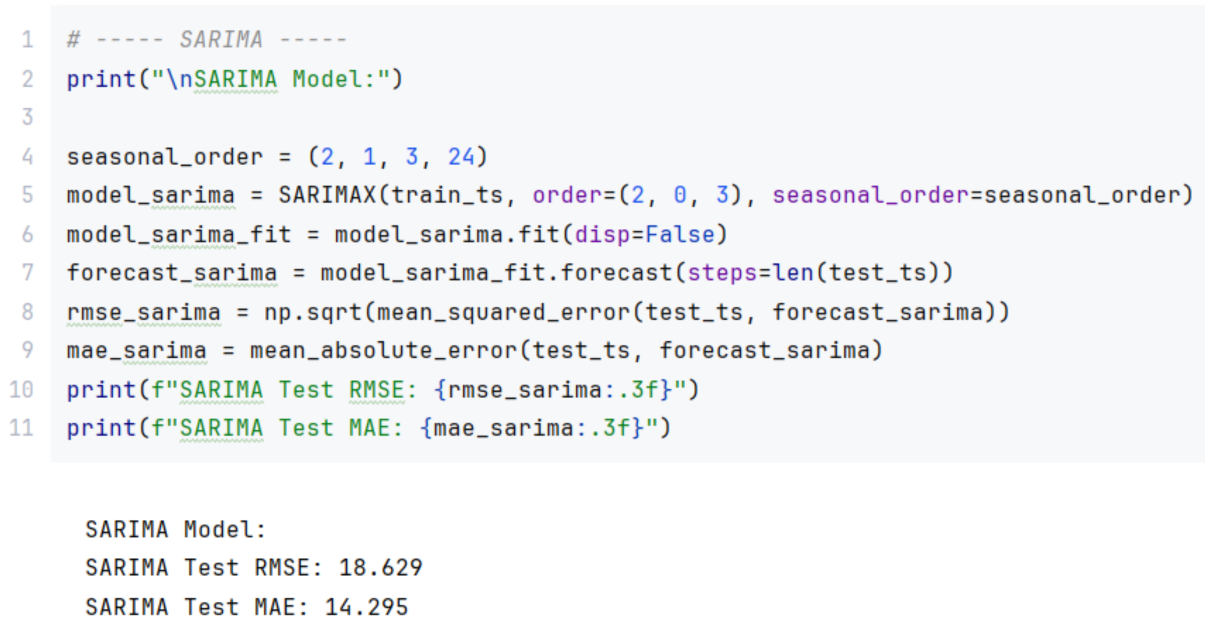


Figure 17: SARIMA - ACF and PACF correlation.

To quantitatively assess the accuracy of the forecast, we calculated the RMSE and MAE. From Figure 18 we can see that RMSE = 18.629 and MAE = 14.295. This demonstrates a noticeable reduction in errors compared to the linear ARIMA model. This improved accuracy confirms the importance of considering seasonal fluctuations with a 24 - hour lag in road traffic modeling.



```
# ----- SARIMA -----
print("\nSARIMA Model:")

seasonal_order = (2, 1, 3, 24)
model_sarima = SARIMAX(train_ts, order=(2, 0, 3), seasonal_order=seasonal_order)
model_sarima_fit = model_sarima.fit(disp=False)
forecast_sarima = model_sarima_fit.forecast(steps=len(test_ts))
rmse_sarima = np.sqrt(mean_squared_error(test_ts, forecast_sarima))
mae_sarima = mean_absolute_error(test_ts, forecast_sarima)
print(f"SARIMA Test RMSE: {rmse_sarima:.3f}")
print(f"SARIMA Test MAE: {mae_sarima:.3f}")
```

```
SARIMA Model:
SARIMA Test RMSE: 18.629
SARIMA Test MAE: 14.295
```

Figure 18: SARIMA results.

The constructed SARIMA (2,0,3) × (2,1,3,24) model demonstrates high accuracy: RMSE = 18.63 and MAE = 14.30. On the graph (Figure 18), the actual values and the forecast overlap almost perfectly, confirming the model's ability to capture daily seasonality and short autocorrelation effects. Figure 19 illustrates the results of the SARIMA model prediction.

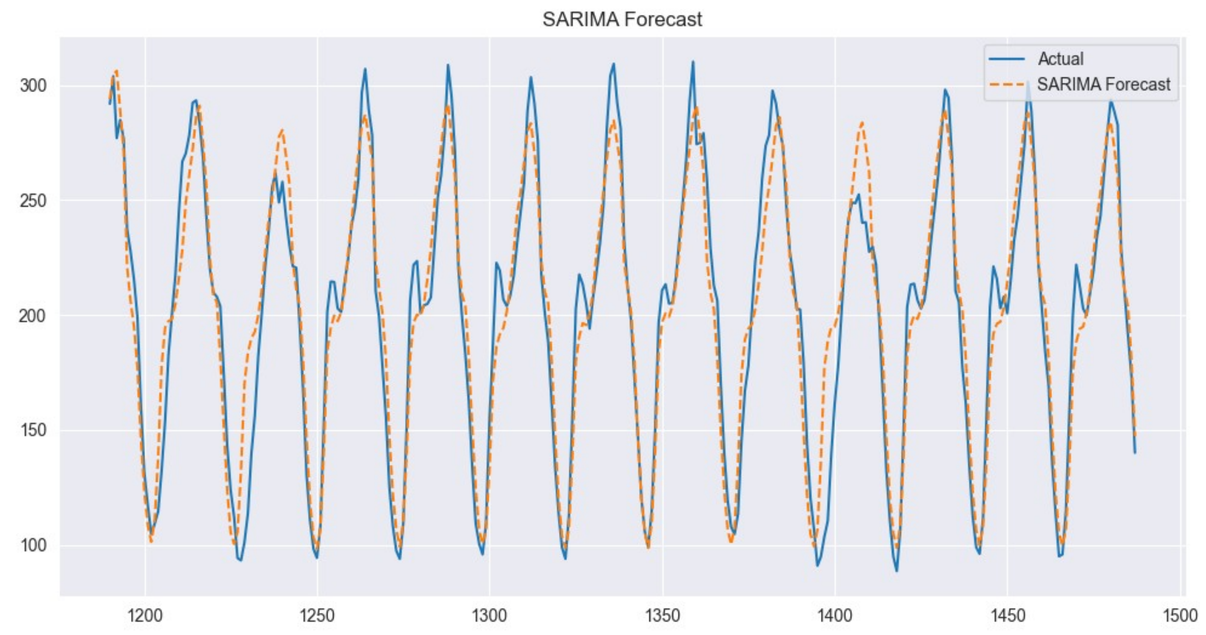


Figure 19: SARIMA forecast.

3.4. Exponential Smoothing

Exponential smoothing (ES) is a method that gives greater weight to recent observations in a time series and uses an exponential function to "smooth" the data. A distinctive feature of this method is that it does not treat all previous data equally but applies a decreasing weighting value that decreases for older observations [22].

The basic idea of the method is to predict future values based on historical data, reducing the influence of older observations.

$$¥_{t+1} = \alpha Y_t + (1-\alpha) ¥_t , \tag{5}$$

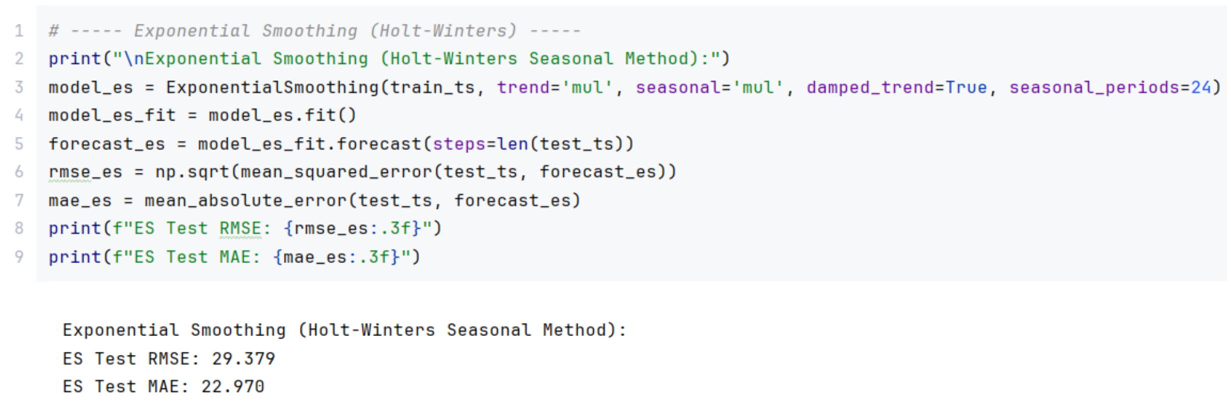
Y_t - *actual value at time* t,

$¥_t$ - *predicted value at time* t,

α - *smoothing parameter (0 < α < 1), which determines the weight given to the latest observations*

The exponential smoothing approach (Holt-Winters) allows simultaneous modeling of the level, trend with attenuation, and multiplicative seasonality with a period (m = 24). It is based on recursive updating of components according to optimized smoothing coefficients.

From Figure 20 we can see that Exponential smoothing showed higher errors compared to SARIMA, indicating its limited ability to capture complex autocorrelation relationships in road traffic data.



```
# ----- Exponential Smoothing (Holt-Winters) -----
print("\nExponential Smoothing (Holt-Winters Seasonal Method):")
model_es = ExponentialSmoothing(train_ts, trend='mul', seasonal='mul', damped_trend=True, seasonal_periods=24)
model_es_fit = model_es.fit()
forecast_es = model_es_fit.forecast(steps=len(test_ts))
rmse_es = np.sqrt(mean_squared_error(test_ts, forecast_es))
mae_es = mean_absolute_error(test_ts, forecast_es)
print(f"ES Test RMSE: {rmse_es:.3f}")
print(f"ES Test MAE: {mae_es:.3f}")
```

```
Exponential Smoothing (Holt-Winters Seasonal Method):
ES Test RMSE: 29.379
ES Test MAE: 22.970
```

Figure 20: Exponential Smoothing results.

Figure 21 below shows actual data and ES forecast (orange dotted line). It displays how accurately the model predicted the results.

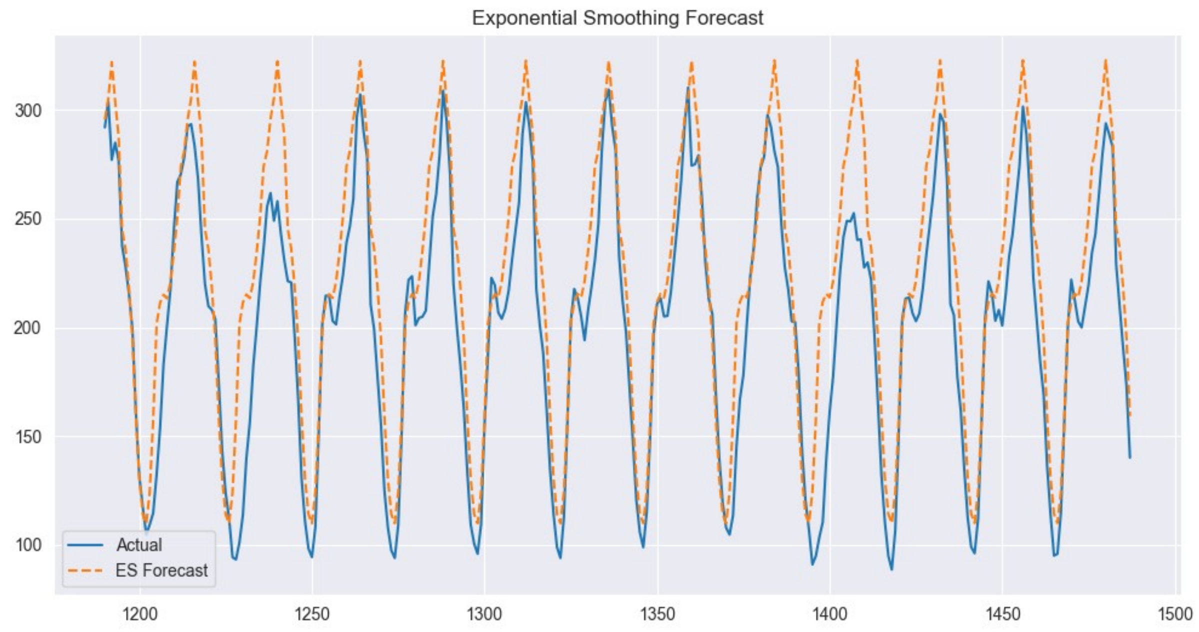


Figure 21: Exponential Smoothing forecast.

3.5. Kalman Filter

As an alternative to traditional models, we used a one-dimensional Kalman filter for traffic forecasting. This method combines a simple dynamic state model with consideration of process noise and measurements, allowing the forecast to be flexibly adapted to data uncertainty [23].

$$X_k = F_k X_{k-1} + B_k u_k + w_k, \tag{6}$$

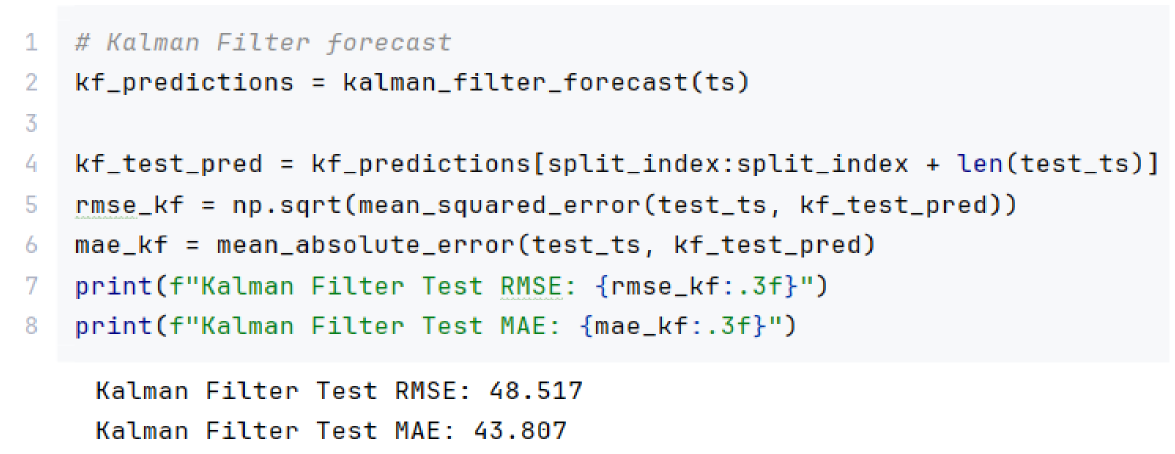
x_k - current state value,

F_k - is a state transition model that applies to the previous state X_{k-1}

B_k - is a model of the input control signal applied to the control vector u_k,

w_k - is a noise

As a result, the Kalman filter (Figure 22) showed the highest errors among the models considered, indicating the limited ability of a simple one-dimensional filter to account for the complex autocorrelation structure in traffic data.



```
# Kalman Filter forecast
kf_predictions = kalman_filter_forecast(ts)

kf_test_pred = kf_predictions[split_index:split_index + len(test_ts)]
rmse_kf = np.sqrt(mean_squared_error(test_ts, kf_test_pred))
mae_kf = mean_absolute_error(test_ts, kf_test_pred)
print(f"Kalman Filter Test RMSE: {rmse_kf:.3f}")
print(f"Kalman Filter Test MAE: {mae_kf:.3f}")
```

```
Kalman Filter Test RMSE: 48.517
Kalman Filter Test MAE: 43.807
```

Figure 22: Kalman Filter forecast.

The graph (Figure 23) below compares actual values (solid blue line) and Kalman Filter predictions (dotted orange line).

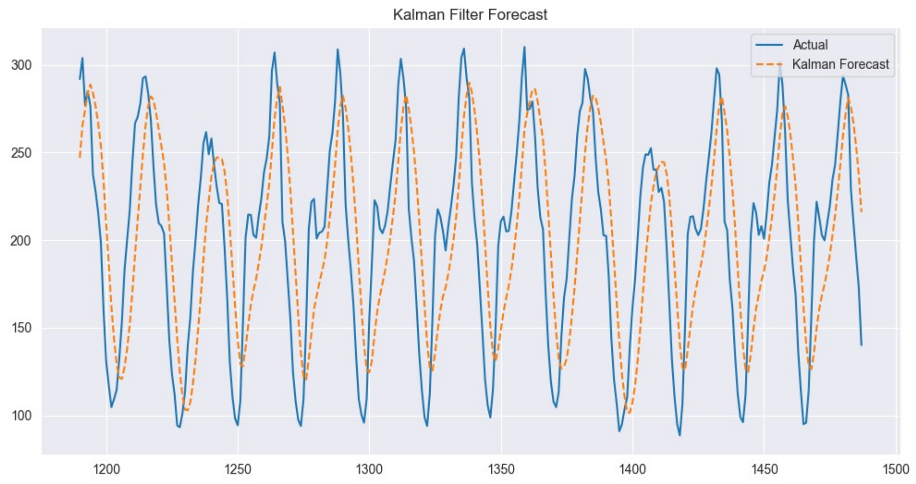


Figure 23: Kalman Filter forecast.

3.6. RNN (LSTM)

This section discusses the application of recurrent neural networks (RNN), in particular LSTM (Long Short-Term Memory) architecture, for transport traffic forecasting. Unlike classical statistical models, which are based on well-defined autocorrelation and seasonality structures, LSTM is capable of automatically detecting complex temporal dependencies, including long-term patterns, nonlinear relationships, and contextual changes [24].

Unlike conventional RNNs, LSTM models have internal memory that allows them to store relevant information over time steps. This is particularly important for traffic, where current traffic intensity depends not only on the previous hour, but also on conditions throughout the day or even week. Thanks to input, output, and forget gates, LSTM controls which information to store, which to update, and which to ignore, which in turn enables flexible learning without gradient loss.

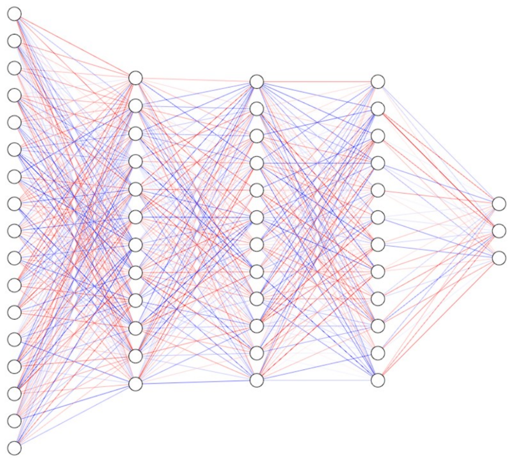


Figure 24: LSTM model (symbolic designation).

Figure 24 is a brief description of the sequential stack of layers implemented in our model:

1. **First LSTM layer:** reads 24-time intervals with 4590 features (sensors/road markers) and generates a 256-dimensional vector at each step, storing information about short-term and medium-term dependencies.
2. **Second LSTM layer:** analyzes the sequence of hidden states of the first layer and returns only the final 256-dimensional vector. This narrows the time dimension, leaving a concise description of the background.
3. **Dropout layers:** applied twice to randomly disable ~20% of neurons during training. This reduces the risk of overfitting, especially when combined with deep layers.
4. **First Dense Layer:** enables the model to build more complex nonlinear combinations of hidden features before the final prediction.
5. **Second Dense Layer:** converts the 256-dimensional vector into 170 prediction values (for each of the 170 sensors of the network) using linear activation.

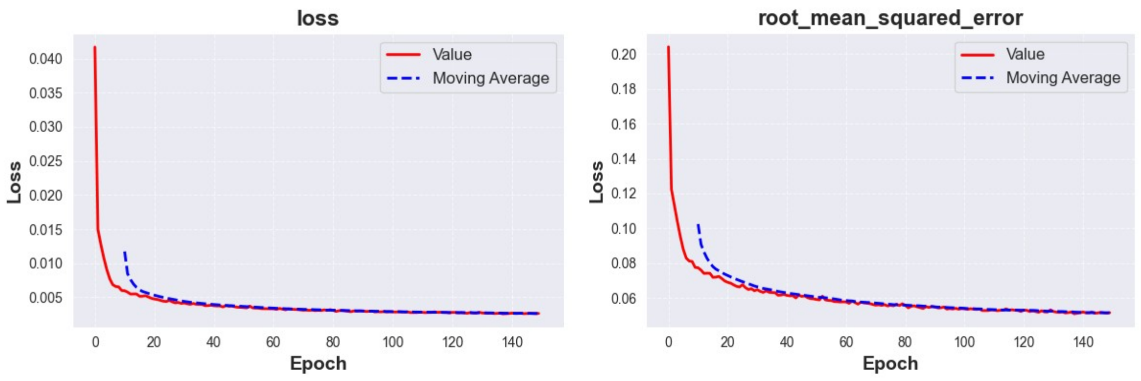


Figure 25: LSTM model training (loss, RMSE).

Figure 25 above shows a graph of loss convergence and RMSE. Figure 26 shows the graphs of validation losses and validation RMSE.

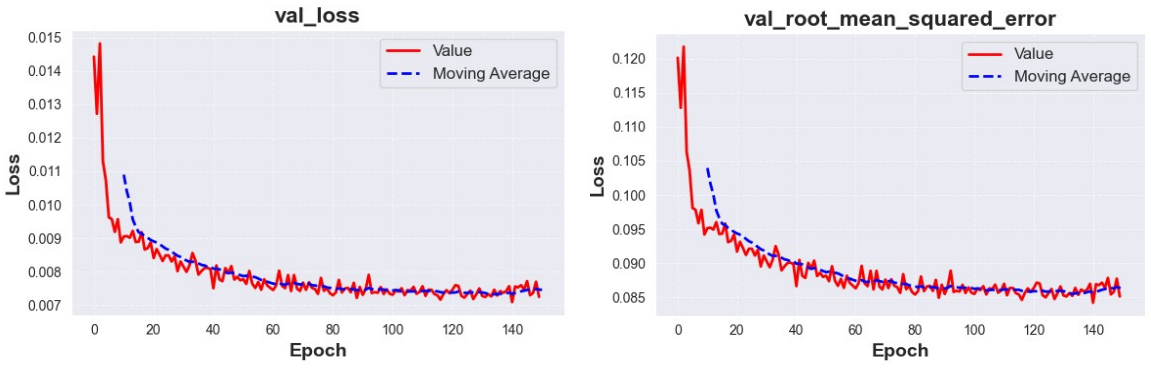


Figure 26: LSTM model training (val-loss, val-RMSE).

Table 3 below shows the key quality metrics of the LSTM model on the training and test data samples.

Table 3
LSTM model training results

№	Metric	Train data	Test data
1	Moving Average RMSE	0.0239	0.0265
2	Prediction RMSE	0.0144	0.0195
3	Moving Average Spearman Correlation	0.71511	0.72089
4	Prediction Spearman Correlation	0.80191	0.81364
5	Mean Absolute Error (MAE)	0.00590	0.00767
6	Determination Coefficient (R^2)	0.85994	0.80810

In this table, “Moving Average” [25] refers to the simple moving average as the baseline, and “Prediction” refers to the results of the LSTM model itself. It shows the high accuracy of LSTM predictions compared to simple methods, good consistency between the results on the training and test sets (no significant overfitting), significant progress in ranking (Spearman) and in reproducing the variable flow (R^2).

Figures 27 and 28 demonstrate how the LSTM model worked and predicted data for the test and training datasets.

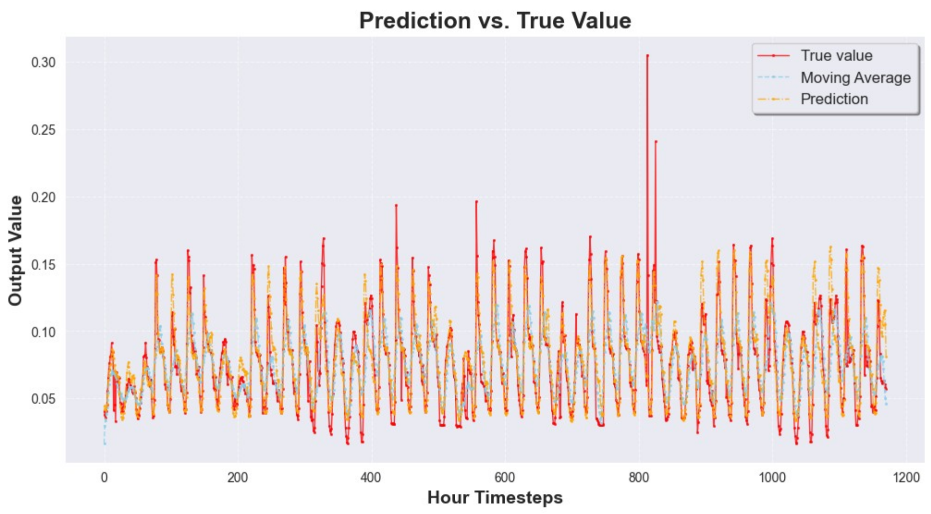


Figure 27: LSTM model prediction on the train data.

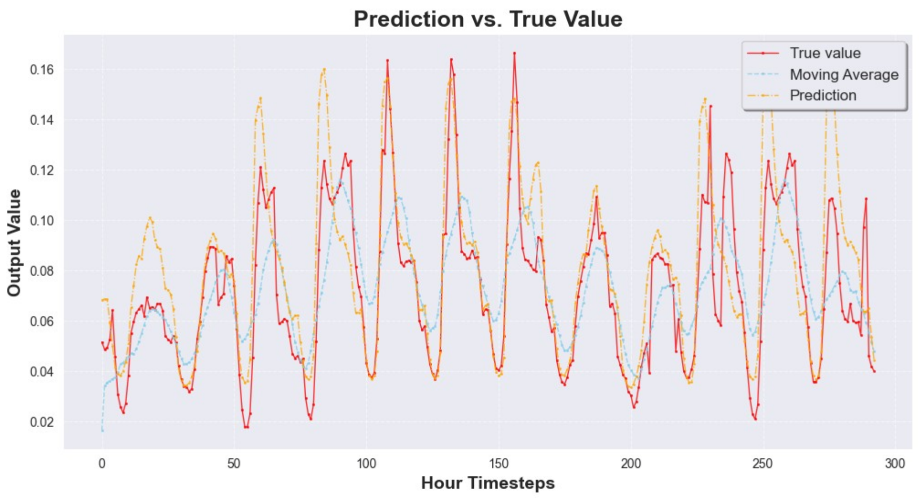


Figure 28: LSTM model prediction on the test data.

4. Models Comparison

Table 4 presents a comparative analysis of six time series forecasting models based on four criteria: RMSE, MAE [26], Spearman ρ [27], and R^2 [28]. This allows us to evaluate not only the absolute accuracy of predictions, but also the quality of ranking and the proportion of explained variance.

Table 4
Model comparison

№	Model	RMSE	MAE	Spearman ρ	R^2
1	Moving Average	0.0265	-	0.72089	-
2	ARIMA(2,0,3)	24.57	19.34	-	-
3	SARIMA (2,0,3) × (2,1,3,24)	18.63	14.30	-	-
4	Exponential smoothing (Holt-Winters)	29.38	22.97	-	-
5	Kalman Filter	48.52	43.81	-	-
6	LSTM	0.0195	0.00767	0.81364	0.80810

Therefore, in terms of accuracy and consistency of results, LSTM is the best solution among the models that were considered. Seasonal ARIMA methods are a viable alternative to traditional ARIMA, but they are inferior to the deep learning approach in terms of error rate and rating.

Conclusions

The results of our study demonstrate that while deep learning models such as LSTM achieve the highest predictive accuracy in short-term traffic forecasting, classical statistical methods (ARIMA, SARIMA, Holt–Winters) and state-space models (Kalman filter) retain significant value due to their transparency and interpretability. These models enable clearer analysis of model behavior, making it possible to identify the factors driving predictions and to diagnose errors when anomalies occur.

During the research, a systematic comparison of six approaches to short-term forecasting of traffic parameters based on PEMS08 data was conducted. This, in turn, made it possible to identify the strengths and weaknesses of each method.

After data preparation, each of the six approaches was trained on the historical set and tested on the delayed test sample using four metrics: RMSE, MAE, Spearman's correlation, and the coefficient of determination R^2. The simplest moving average method showed an RMSE of 0.0265, ARIMA showed an RMSE of 24.57 and MAE of 19.34, while seasonal SARIMA managed to significantly reduce errors to an RMSE of 18.63 and MAE of 14.30 by taking daily cycles into account. Exponential smoothing and the Kalman filter proved to be less accurate, with RMSE of 29.38 and 48.52, respectively. This highlights their limitations in modeling complex traffic autocorrelation.

The culmination of the research is the implementation of a two-layer LSTM network solution with two memory layers of 256 units each, two dropout layers (20%), and two dense layers for generating predictions for 170 detectors simultaneously. This architecture allowed us to take full advantage of the flexibility of LSTM in storing additional information about short-term and long-term dependencies, controlling the flow of gradients, and avoiding overfitting. The model demonstrated the best results: RMSE = 0.0195, MAE = 0.00767, Spearman = 0.81364, and R^2 = 0.80810.

Our results demonstrate the advantages of deep learning-based approaches for real-time monitoring and forecasting of road conditions. The next stage of development involves integrating hybrid schemes that combine statistical and neural models, as well as using ensembles to further improve the stability and accuracy of forecasts. This approach opens the way to create highly reliable accident warning systems and adaptive traffic management in the smart cities of the future.

Among the models considered, the LSTM network demonstrated the highest accuracy across all key metrics (RMSE, MAE, Spearman's ρ, and R^2), confirming its ability to capture complex nonlinear dependencies and long - term temporal patterns. The SARIMA seasonal model proved to be significantly more accurate than the traditional ARIMA due to its consideration of daily seasonality, while exponential smoothing and the one-dimensional Kalman filter demonstrated limited effectiveness in modeling the complex autocorrelation structure of traffic flows.

The use of outlier detection methods based on interquartile range and Z-score ensured the removal of anomalous values from the data and increased the stability of all models. Correlation analysis confirmed a strong positive relationship between flow and congestion and a pronounced negative correlation between congestion and speed, justifying the joint modeling of these three characteristics.

From the perspective of trustworthy and responsible AI, the trade-off between accuracy and explainability is particularly relevant for intelligent traffic management systems in smart cities. Since such systems are classified as high-risk under the EU AI Act, achieving regulatory compliance requires not only high accuracy but also interpretability, transparency, and the possibility of human oversight. Explainable AI techniques therefore represent an essential complement to advanced predictive models, allowing stakeholders to understand the reasoning behind forecasts and to build confidence in their safe deployment.

The inclusion of hybrid and ensemble approaches, such as ARIMA, NN, Kalman Filter and RNN, highlights a promising direction for future research. These methods combine the interpretability of statistical models with the expressive power of neural networks, offering a pathway to systems that can simultaneously meet accuracy requirements and regulatory demands for explainability.

So, another important outcome of this research is the recognition that accuracy alone is not sufficient for intelligent traffic forecasting systems deployed in smart cities. Although deep learning models such as LSTM demonstrated superior predictive performance across all evaluation metrics, their “black box” nature creates challenges in understanding the logic of the decision-making process. For high-risk domains such as traffic management, which directly affect public safety, explainability and transparency are equally important requirements.

Statistical models, including ARIMA, SARIMA, Holt–Winters, and the Kalman filter, offer a natural advantage in this regard. Their parameters can be directly interpreted in terms of lag effects, trends, and seasonal cycles, allowing operators to trace the origins of predictions and identify the causes of anomalies. For example, SARIMA explicitly incorporates daily periodicity, while Holt–Winters decomposes forecasts into level, trend, and seasonal components. Such transparency facilitates trust and provides a means of diagnosing model errors.

In contrast, LSTM networks capture complex nonlinear and long-term dependencies but operate as opaque systems. To address this limitation, the application of Explainable AI (XAI) methods such as SHAP, LIME, or attention-based mechanisms can provide insight into the most influential time steps and input variables that shaped a given prediction. This allows system operators and regulators to analyze why an incorrect forecast occurred and to establish safeguards against potential adverse consequences.

Hybrid and ensemble approaches represent a particularly promising solution, as they combine the interpretability of statistical models with the flexibility of neural networks. In models such as ARIMA+NN or Kalman Filter and RNN, the statistical component remains transparent and explainable, while the neural component refines the prediction by learning nonlinear dependencies. This provides a balanced compromise between interpretability and predictive accuracy, which is essential for the trustworthy deployment of AI in critical infrastructures.

Finally, the explainability of AI systems in traffic management is not merely a technical advantage but a regulatory obligation. According to the EU Artificial Intelligence Act (2024/1689), traffic management systems are classified as high-risk, requiring transparency, accountability, and the ability to audit algorithmic decisions. Therefore, integrating explainable models and XAI techniques into forecasting architectures is a necessary step toward compliance with legal frameworks and the development of safe, reliable, and socially responsible traffic management systems in smart cities.

In conclusion, our findings emphasize that the future of real-time traffic forecasting in smart cities should not be determined solely by predictive accuracy. Instead, it must also incorporate explainability, robustness, and compliance with emerging regulatory frameworks. This integrated perspective will enable the development of traffic management systems that are not only technically effective but also safe, trustworthy, and socially responsible.

Acknowledgements

This work was realized within the framework of the Erasmus+ Jean Monnet Module 2022 «101085626 – TrustAI – ERASMUS-JMO-2022-HEI-TCH-RSCH».

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly in order to: Grammar and spelling check. After using this service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] A. M. Nagy and V. Simon, 'Survey on traffic prediction in smart cities', *Pervasive and Mobile Computing*, vol. 50, pp. 148–163, Oct. 2018, doi: 10.1016/j.pmcj.2018.07.004.

[2] A. Doroshenko and D. Savchuk, 'Predicting road accidents in smart cities: machine learning approach for enhanced safety', in: V. Teslyuk, N. Kryvinska, A. Poniszewska-Maranda, I. Miladinovic, V. Lytvyn, and V. Vysotska (Eds.), Proceedings of the Computational Intelligence Application Workshop (CIAW 2024), Lviv, Ukraine, October 10-12, 2024, pp. 122–136. [Online]. Available: https://ceur-ws.org/Vol-3861/paper8.pdf.

[3] V. Teslyuk, P. Denysyuk, N. Kryvinska, K. Beregovska, T. Teslyuk. Neural controller for smart house security subsystem. Procedia Computer Science. Proceedings of the10th Intern. conf. on emerging ubiquitous systems and pervasive networks EUSPN-2019, Proceedings of the 9th Intern. conf. on current and future trends of information and communication technologies in

healthcare ICTH-2019), Affiliated workshops, Vol. 160, 2019, P. 394–401. doi: 10.1016/j.procs.2019.11.075

[4] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, ‘A Survey of Methods for Explaining Black Box Models’, *ACM Comput. Surv.*, vol. 51, no. 5, pp. 1–42, Sept. 2019, doi: 10.1145/3236009.

[5] A. Barredo Arrieta et al., ‘Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI’, Information Fusion, vol. 58, pp. 82–115, June 2020, doi: 10.1016/j.inffus.2019.12.012.

[6] D. Savchuk and A. Doroshenko, ‘Explainable AI methods to increase trustworthiness in healthcare’, in Responsible and Explainable Artificial Intelligence in Healthcare, Elsevier, 2025, pp. 55–89. doi: 10.1016/B978-0-443-24788-0.00003-0.

[7] G. Li, Y. Yang, S. Li, X. Qu, N. Lyu, and S. E. Li, ‘Decision making of autonomous vehicles in lane change scenarios: Deep reinforcement learning approaches with risk awareness’, *Transportation Research Part C: Emerging Technologies*, vol. 134, p. 103452, Jan. 2022, doi: 10.1016/j.trc.2021.103452.

[8] J. Wu, Z. Huang, Z. Hu, and C. Lv, ‘Toward Human-in-the-Loop AI: Enhancing Deep Reinforcement Learning via Real-Time Human Guidance for Autonomous Driving’, Engineering, vol. 21, pp. 75–91, Feb. 2023, doi: 10.1016/j.eng.2022.05.017

[9] European Union, Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (AI Act), OJ L 168, 12.6.2024, pp. 1–169.

[10] California Department of Transportation, Caltrans PeMS. URL: https://pems.dot.ca.gov/.

[11] GitHub - juyongjiang/TimeSeriesDatasets: This repository collects public benchmark datasets for Time Series Forecasting (TSF). GitHub. URL: https://github.com/juyongjiang/TimeSeriesDatasets.

[12] The Interquartile Range: Theory and Estimation. Digital Commons @ East Tennessee State University. URL: https://dc.etsu.edu/etd/1030.

[13] Z-Score: Definition, Formula, Calculation & Interpretation. Simply Psychology. URL: https://www.simplypsychology.org/z-score.html.

[14] What is a correlation matrix?. Educative: Interactive Courses for Software Developers. URL: https://www.educative.io/answers/what-is-a-correlation-matrix.

[15] V. N. Katambire, R. Musabe, A. Uwitonze, and D. Mukanyiligira, ‘Forecasting the Traffic Flow by Using ARIMA and LSTM Models: Case of Muhima Junction’, Forecasting, vol. 5, no. 4, pp. 616–628, Nov. 2023, doi: 10.3390/forecast5040034.

[16] Y. Wang, R. Jia, F. Dai, and Y. Ye, ‘Traffic Flow Prediction Method Based on Seasonal Characteristics and SARIMA-NAR Model’, Applied Sciences, vol. 12, no. 4, p. 2190, Feb. 2022, doi: 10.3390/app12042190.

[17] K. Y. Chan, T. S. Dillon, J. Singh, and E. Chang, ‘Traffic flow forecasting neural networks based on exponential smoothing method’, in 2011 6th IEEE Conference on Industrial Electronics and Applications, Beijing, China: IEEE, June 2011, pp. 376–381. doi: 10.1109/ICIEA.2011.5975612.

[18] S. V. Kumar, ‘Traffic Flow Prediction using Kalman Filtering Technique’, Procedia Engineering, vol. 187, pp. 582–587, 2017, doi: 10.1016/j.proeng.2017.04.417.

[19] A. Belhadi, Y. Djenouri, D. Djenouri, and J. C.-W. Lin, ‘A recurrent neural network for urban long-term traffic flow forecasting’, Appl Intell, vol. 50, no. 10, pp. 3252–3265, Oct. 2020, doi: 10.1007/s10489-020-01716-1.

[20] S. V. Kumar and L. Vanajakshi, ‘Short-term traffic flow prediction using seasonal ARIMA model with limited input data’, Eur. Transp. Res. Rev., vol. 7, no. 3, p. 21, Sept. 2015, doi: 10.1007/s12544-015-0170-8.

[21] N. Deretić, D. Stanimirović, M. A. Awadh, N. Vujanović, and A. Djukić, ‘SARIMA Modelling Approach for Forecasting of Traffic Accidents’, Sustainability, vol. 14, no. 8, p. 4403, Apr. 2022, doi: 10.3390/su14084403.

[22] E. Ostertagová and O. Ostertag, ‘Forecasting using simple exponential smoothing method’, Acta Electrotechnica et Informatica, vol. 12, no. 3, Jan. 2012, doi: 10.2478/v10198-012-0034-2.

[23] K. A. Momin, S. Barua, Md. S. Jamil, and O. F. Hamim, ‘Short duration traffic flow prediction using kalman filtering’, presented at the 6TH INTERNATIONAL CONFERENCE ON CIVIL ENGINEERING FOR SUSTAINABLE DEVELOPMENT (ICCESD 2022), Khulna, Bangladesh, 2023, p. 040011. doi: 10.1063/5.0129721.

[24] R. Fu, Z. Zhang, and L. Li, ‘Using LSTM and GRU neural network methods for traffic flow prediction’, in 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), Wuhan, Hubei Province, China: IEEE, Nov. 2016, pp. 324–328. doi: 10.1109/YAC.2016.7804912.

[25] Fernando J. Moving Average (MA): Purpose, Uses, Formula, and Examples. Investopedia. URL: https://www.investopedia.com/terms/m/movingaverage.asp.

[26] T. Chai and R. R. Draxler, ‘Root mean square error (RMSE) or mean absolute error (MAE)?’, Feb. 28, 2014, Numerical methods. doi: 10.5194/gmdd-7-1525-2014.

[27] J. W. Gooch, ‘Spearman Rank Correlation Coefficient’, in Encyclopedic Dictionary of Polymers, J. W. Gooch, Ed., New York, NY: Springer New York, 2011, pp. 996–997. doi: 10.1007/978-1-4419-6247-8_15382.

[28] F. Pyrczak and D. M. Oh, Making Sense of Statistics: A Conceptual Overview, 8th edn. New York: Routledge, 2023. doi: 10.4324/9781003299356.