Real-Time Traffic Prediction with Kafka

Ayush Tripathi

94879-A1

Programming Assignment

Introduction

In this project, we explore the use of Apache Kafka for real-time data streaming and traffic flow prediction. As urbanization increases, traffic congestion becomes a critical issue affecting transportation efficiency and quality of life. This project aims to address this by building a Kafka-based pipeline to stream, process, and predict traffic flow data in real time.

The dataset used for this project is the Traffic Flow Forecasting dataset, which contains hourly traffic data, including measurements of flow, speed, and occupancy across various highways. This data is ideal for time series analysis and prediction, which we perform after streaming it through Kafka.

The project is divided into three main phases:

1. Kafka Setup and Data Streaming
2. Exploratory Data Analysis on Time Series Data
3. Traffic Flow Prediction using a Basic Predictive Model

Methods

Kafka Setup and Data Streaming

The first phase involved setting up Kafka and streaming traffic data in real time. Apache Kafka was configured with a producer and a consumer to simulate real-time traffic data flow. The producer streamed traffic data into a Kafka topic, and the consumer processed the data in real time. Python was used to implement both the producer and consumer scripts, introducing 1-second intervals between data streams to simulate real-world traffic conditions.

- Producer Script: Reads traffic data from a CSV file and streams it into Kafka.
- Consumer Script: Consumes traffic data from the Kafka topic, storing it locally for analysis.

Exploratory Data Analysis

In this exploratory data analysis (EDA), several visualizations and statistical methods were applied to understand the temporal patterns in traffic flow data.

The training data was loaded from two pickle files, tra_X_tr.pkl and tra_Y_tr.pkl, using the pickle module. The data in tra_X_tr.pkl was converted from a sparse matrix to a dense matrix and stored in a pandas DataFrame, df_tra_X_tr. The target variable, df_tra_Y_tr, was similarly loaded into a DataFrame from tra_Y_tr.pkl.

To visualize traffic flow over time for a specific location, the traffic data from df_tra_Y_tr was extracted for Location 1. A time series plot was generated, with time intervals (15-minute intervals) as the x-axis and traffic flow as the y-axis. This plot provides insight into how traffic flow fluctuates over time for a single location.

Autocorrelation and partial autocorrelation were used to detect temporal dependencies and patterns. Using the plot_acf and plot_pacf functions from the statsmodels library, autocorrelation (ACF) and partial autocorrelation (PACF) plots were created for Location 1. The number of lags was set to the minimum of 50 or half the length of the series.

To understand the distribution of traffic flow at Location 1, a histogram and kernel density estimate (KDE) plot were created. The histogram visualizes the distribution of traffic values, while the KDE provides a smoothed estimate of the density.

A heatmap was created using Seaborn to visualize traffic flow across all locations over time. Each cell in the heatmap represents the traffic flow for a specific location at a specific time, allowing for the identification of spatial and temporal patterns.

To explore daily traffic trends, traffic data was truncated to ensure a complete set of 15-minute intervals per day (96 intervals per day). The average daily traffic flow was calculated by averaging traffic flow at each 15-minute interval across all complete days. A line plot was generated to visualize this average daily traffic pattern.

These visualizations were used to identify trends, seasonality, and anomalies in traffic flow data, which inform further model development and feature engineering.

Traffic Flow Prediction

Traffic data from two primary sources were loaded: tra_X_tr.pkl for features and tra_Y_tr.pkl for target values, both serialized in pickle format. The feature data, initially in sparse format, was converted to a dense pandas DataFrame, df_tra_X_tr, to facilitate manipulation. The target variable df_tra_Y_tr underwent a similar transformation.

To enhance the predictive power of the models, several features were engineered:

- Time-based Features: The hour_of_day and day_of_week were computed from the index, assuming a continuous capture of data at regular intervals.
- Rolling Statistics: Rolling mean and standard deviation over a one-hour period (4 intervals of 15 minutes each) were computed to capture short-term trends and fluctuations in traffic flow. Missing values resulting from the rolling computations were imputed using backward filling.

A simple Linear Regression model was utilized to predict traffic flow based solely on the rolling_mean feature. This model served as a baseline to assess the linear relationship between traffic flow and temporal trends. The model's performance was evaluated using Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), providing initial insights into its predictive accuracy.

For a more nuanced analysis considering autocorrelation in the time series data, an AutoRegressive Integrated Moving Average (ARIMA) model was employed. Configured with parameters (5,1,0), this model aimed to forecast traffic flow at a single location based on its past values, taking into account the non-stationarity and autoregressive nature of the traffic data. The training process involved fitting the ARIMA model to the data of one specific location, and its efficacy was similarly evaluated using MAE and RMSE.

Additionally, a Kafka Producer and Consumer script were integrated to simulate a real-time traffic data processing environment. The Kafka Producer is initialized with JSON serialization capabilities and it streams feature data to a Kafka topic traffic_features. Each DataFrame row was converted into a dictionary and sent sequentially, facilitating real-time data handling. The Kafka Consumer is configured to deserialize JSON data from the traffic_features topic, it was set to continuously listen for new messages. Upon receiving data, the consumer extracted necessary features to input into the ARIMA model for real-time prediction.

The consumer applied the ARIMA model to newly ingested data to predict future traffic flow, leveraging the temporal dynamics captured by the model. This approach highlights the feasibility of integrating predictive modeling with real-time data streams to facilitate dynamic traffic management systems.

These methods underscore the integration of statistical modeling with real-time data processing, showcasing a practical approach to traffic flow prediction that can be scaled and adapted for live traffic management systems. The methods employed not only provided a foundation for advanced predictive analytics but also demonstrated the effective use of Kafka for real-time data streaming and processing in a traffic context.

Results

Exploratory Data Analysis

This section aims to discuss the results of the EDA. Below are several descriptions and analyses of the data visualizations. Any key patterns, trends, and anomalies in the data will also be identified. Finally, the results from the visualizations will enforce how these findings will influence the model selection and feature engineering in the next section.

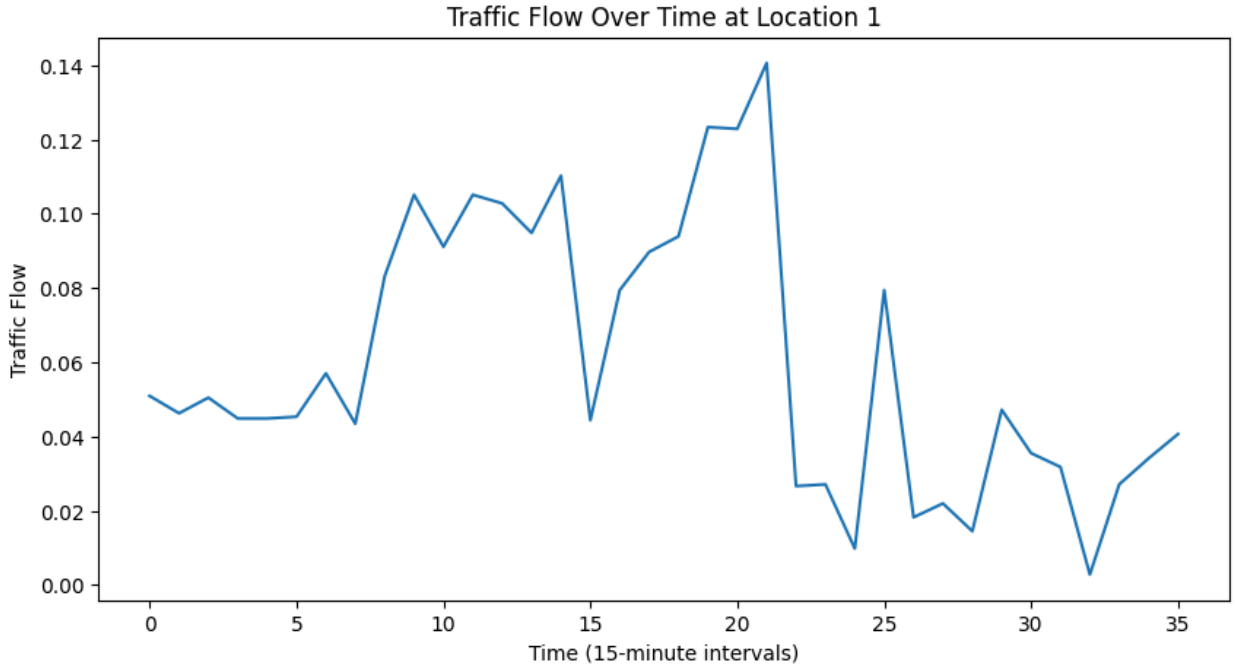

Figure 1: Traffic flow at Location 1

The traffic flow at Location 1 demonstrates significant variability within a relatively short time frame (35 intervals of 15 minutes each). Initially, traffic flow starts low, increases steeply to a peak around the 15th interval, declines slightly, and peaks again around the 23rd interval before a sharp drop and a final rise towards the end of the observed period. This pattern suggests multiple high-activity periods, possibly correlating with typical rush hours.

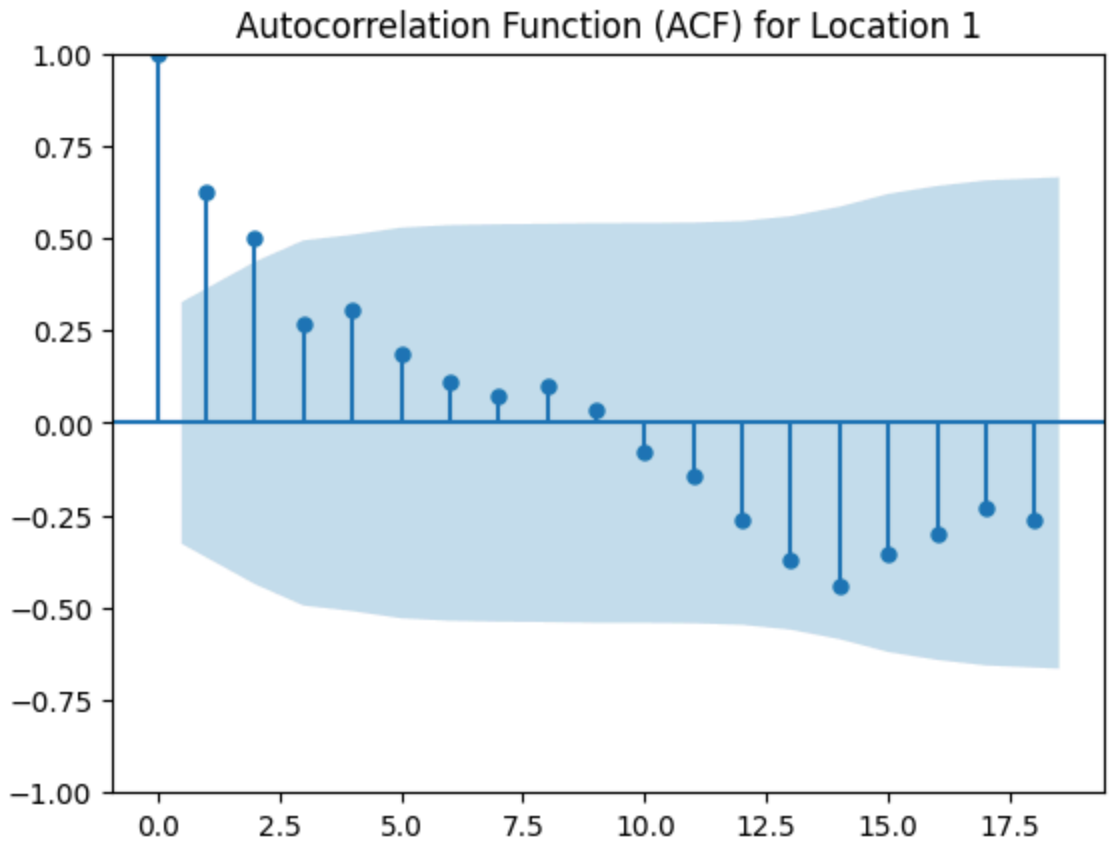

Figure 2: Autocorrelation Function for Location 1

The Autocorrelation Function (ACF) for Location 1 reveals that the traffic flow data exhibit strong positive autocorrelation at initial lags, which gradually decreases but stays significant up to about 18 lags. This suggests that the traffic flow at any given time is significantly influenced by the traffic flow of the previous 4.5 hours, with diminishing influence as the time lag increases.

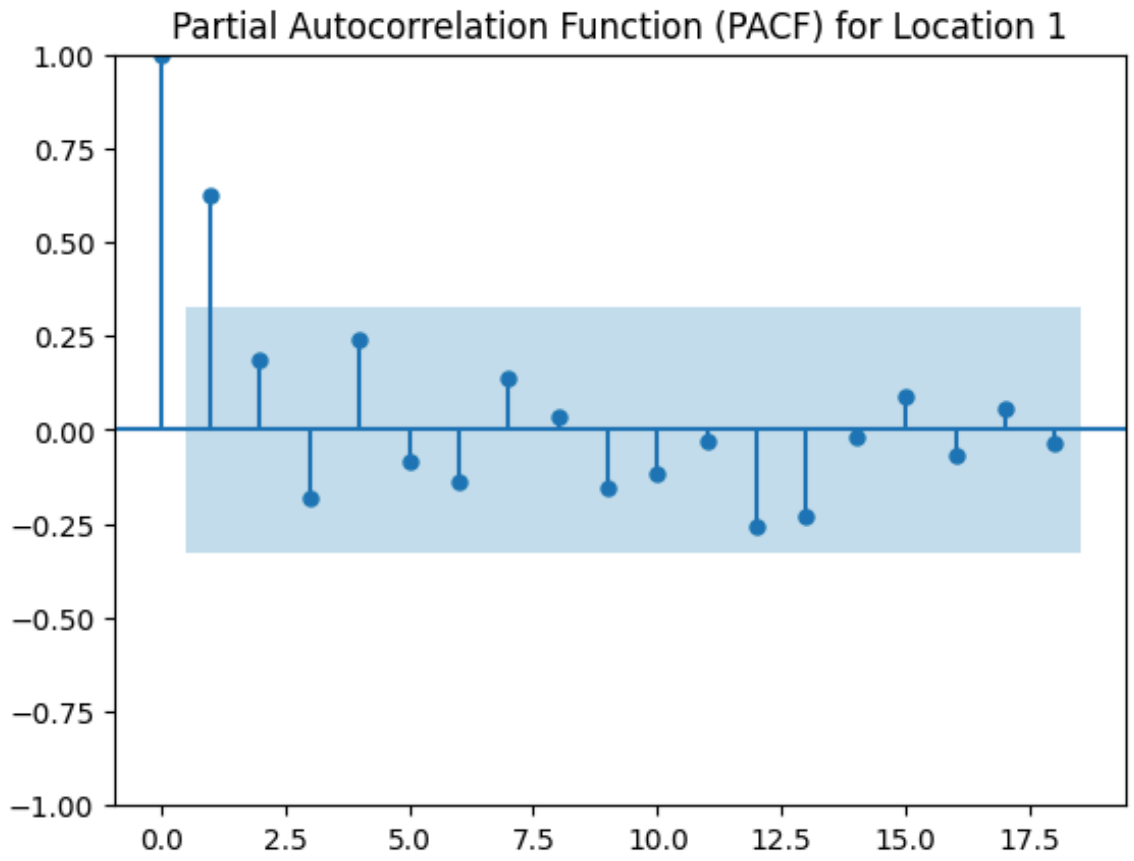

Figure 3: Partial Autocorrelation Function for Location 1
The Partial Autocorrelation Function (PACF) shows a sharp drop after the first lag, which remains relatively stable and around zero thereafter. This indicates that any autocorrelation in traffic flow after the first lag can be explained by its direct correlation with prior values without needing further lags to explain additional variance.

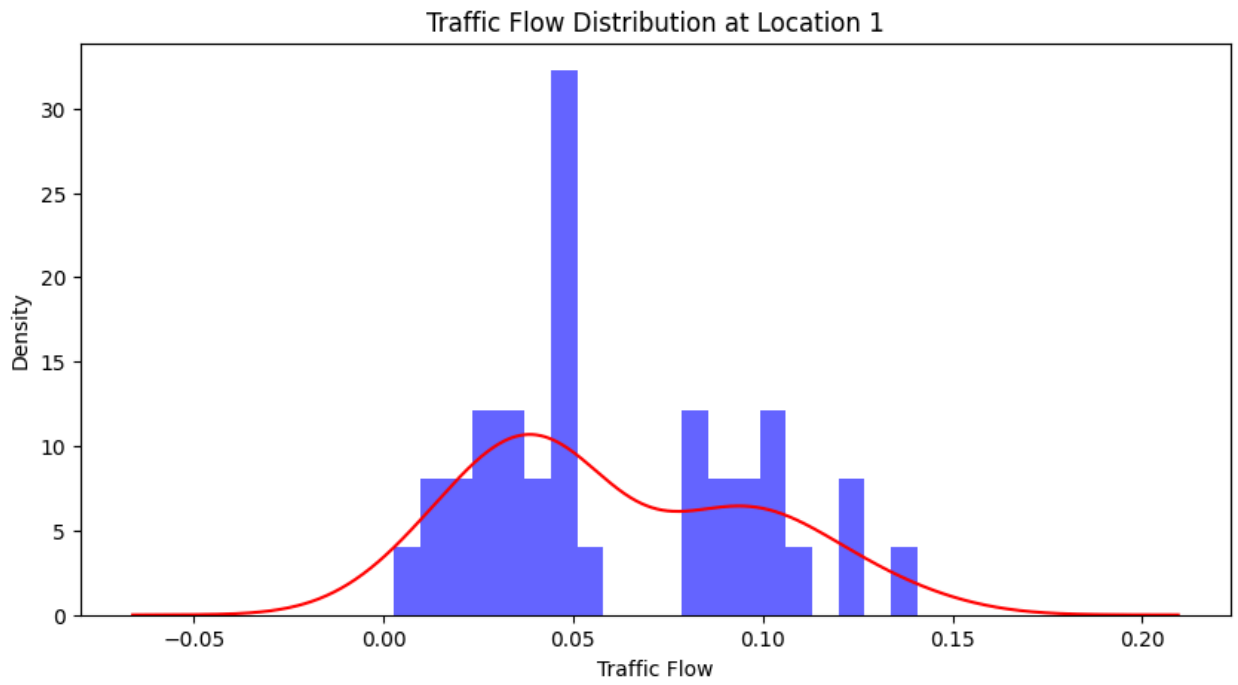

Figure 4: Traffic Flow Distribution at Location 1

The distribution of traffic flow at Location 1 shows a multi-modal pattern, with one predominant peak around 0.05 and smaller peaks around 0.10 and 0.15. This bimodal distribution suggests different typical traffic states, possibly distinguishing between peak and off-peak traffic conditions.

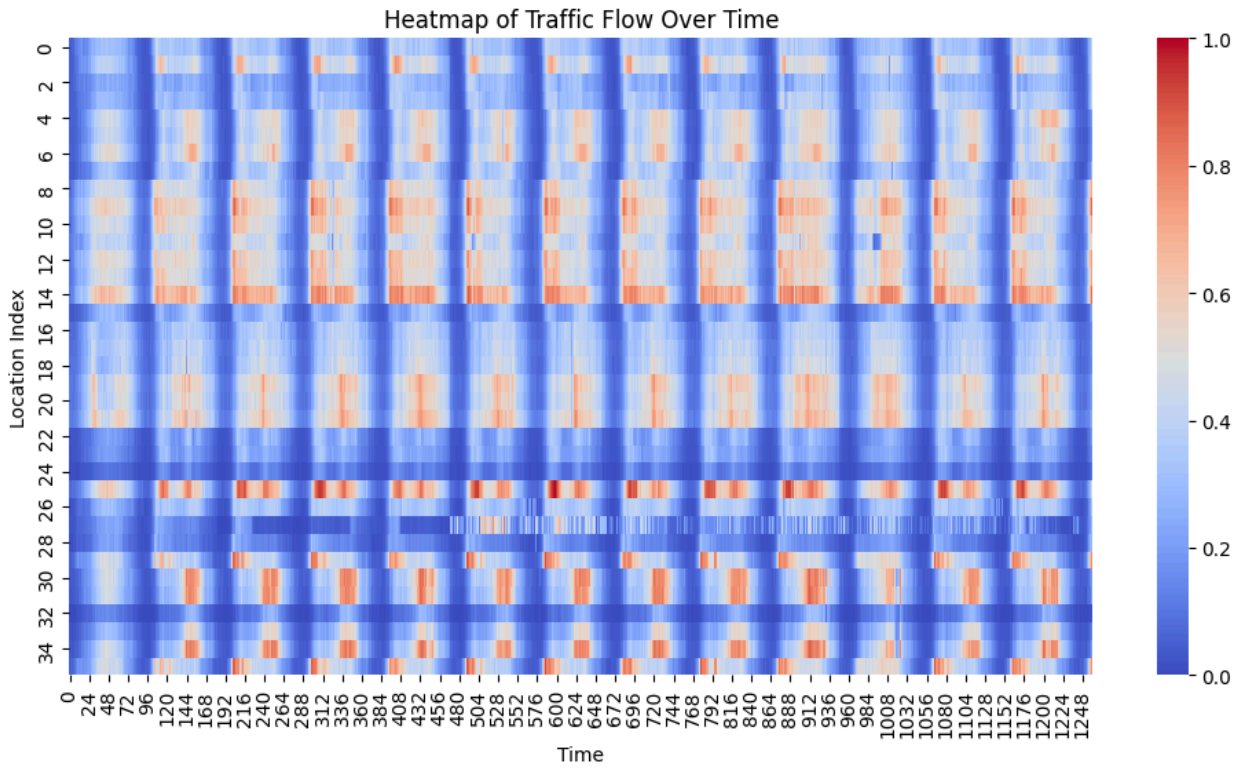

Figure 5: Heatmap of Traffic Flow Over Time

The heatmap across all locations shows clear patterns of variation, with periodic intervals of higher and lower traffic flows. Notably, regular intervals of increased traffic can be observed, suggesting daily cyclic activity likely aligned with typical morning and evening rush hours.

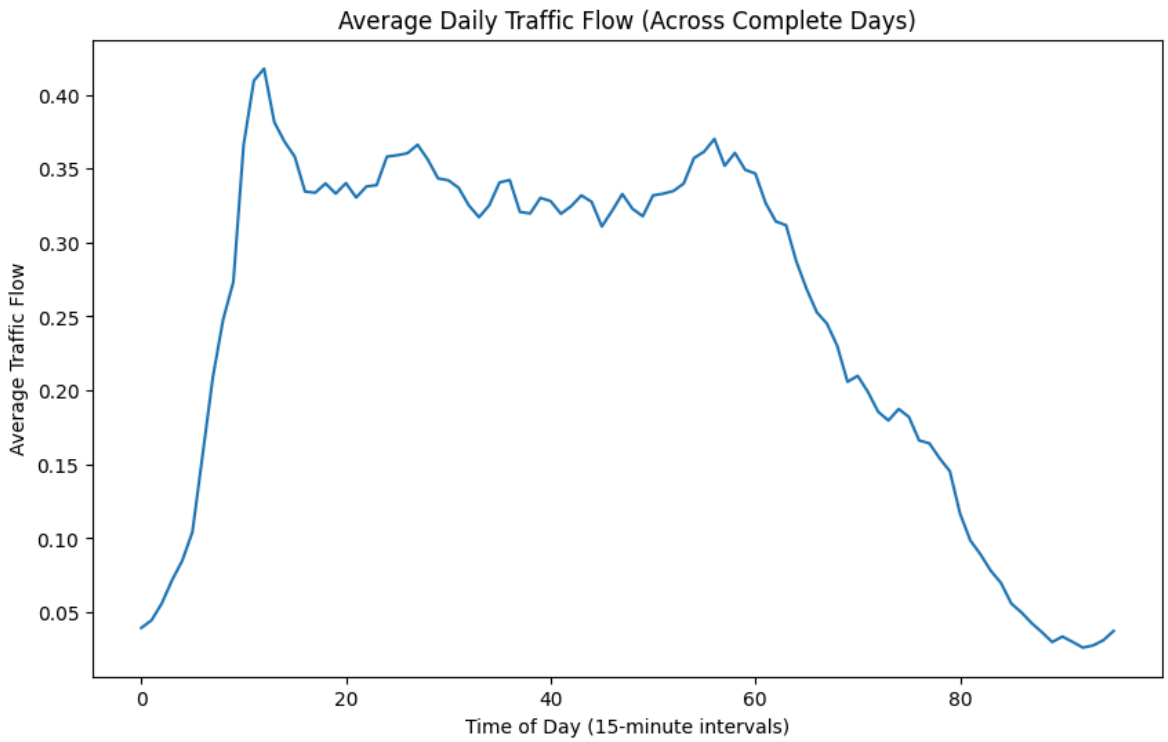

Figure 6: Average Daily Traffic Flow
The average daily traffic flow across complete days displays a pronounced peak during the early intervals, which gradually decreases as the day progresses. This could indicate a substantial volume of traffic during morning hours, which tapers off towards the evening.

The cyclic patterns observed in the time-series and heatmap data suggest that time-based features such as hour of the day, day of the week, and possibly week of the year could be crucial predictors for traffic flow models. Furthermore, the strong autocorrelation at initial lags supports the use of ARIMA models for forecasting, as these models are well-suited to leveraging this autocorrelation. The multi-modal distribution of traffic flow suggests that non-linear models like decision trees or neural networks might be effective in capturing the complex patterns in traffic flow beyond what linear models might predict. Given the significant daily patterns observed, rolling averages and standard deviations could serve as effective features, capturing short-term trends and variability in traffic flow, enhancing model performance in both regression and classification tasks.

These findings will guide the development of predictive models and the selection of features that effectively capture the underlying dynamics of traffic flow, improving the accuracy and reliability of traffic predictions.

Traffic Flow Prediction

This section aims to discuss the selected and implemented basic predictive model. Two predictive models, Linear Regression and ARIMA, were implemented to forecast traffic flow

based on time-series data. The models leveraged features such as rolling statistics (e.g., rolling mean and standard deviation), and temporal indicators like hour of the day and day of the week.

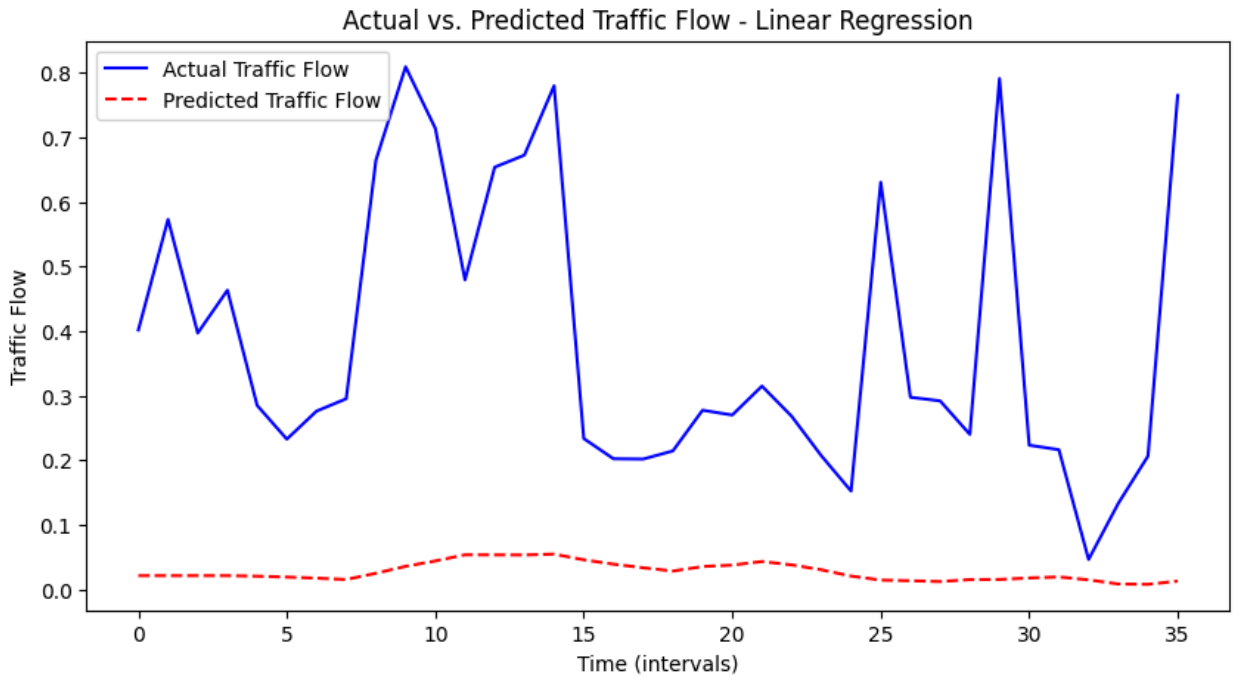

Figure 1: Actual vs Predicted Traffic Flow from Linear Regression

The Linear Regression model was trained on the engineered features and evaluated on a separate test set. The results indicated a mean absolute error (MAE) of 0.3585 and a root mean squared error (RMSE) of 0.4171 on the testing data. The plot of actual vs. predicted traffic flow (Figure 1) shows that the Linear Regression model could not capture the variability and peaks in the traffic flow data, resulting in a significant underestimation of traffic flow values.

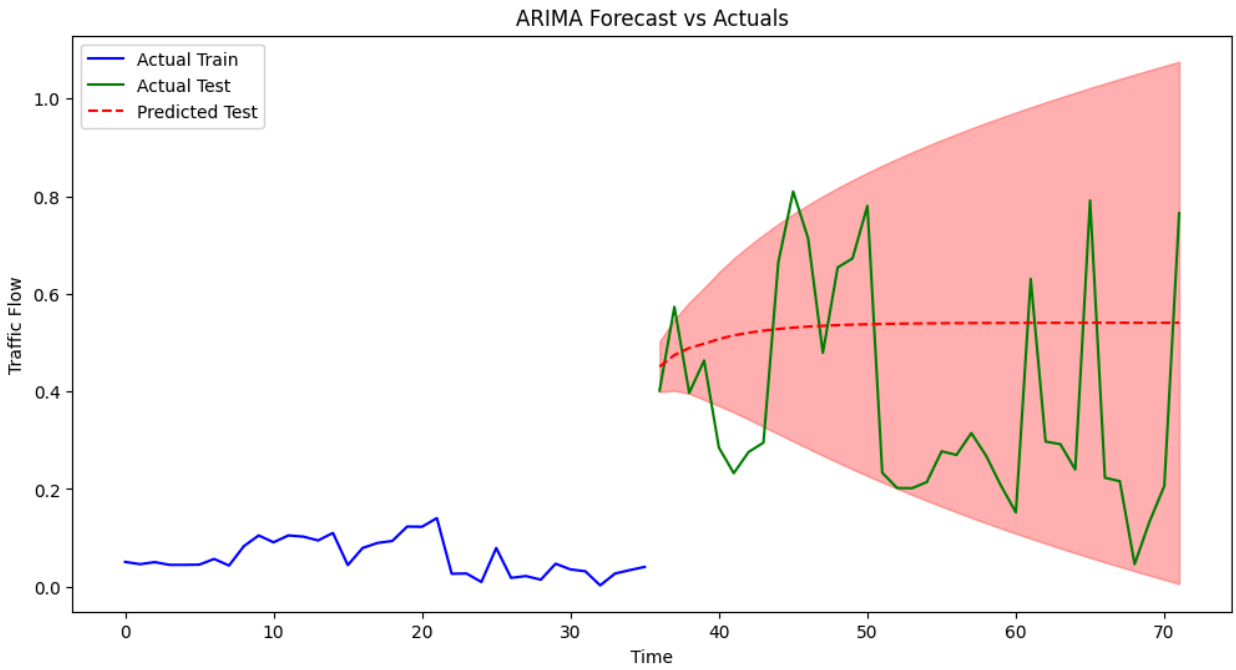

Figure 2: ARIMA Forecast vs Actuals

The ARIMA model was specified with parameters (5,1,0) and trained on univariate time-series data. The testing performance of the ARIMA model showed an MAE of 0.2414 and an RMSE of 0.2635. In Figure 2, the ARIMA model predictions are overlaid with the actual data, showing better responsiveness to changes in traffic flow compared to Linear Regression, though it still had issues with peak predictions. These metrics indicate the average magnitude of the errors in a set of predictions, without considering their direction. The lower values suggest a good fit between the predicted and actual values, with the MAE indicating that, on average, the model's predictions deviate from the actual values by about 0.2414. The RMSE value offers a slightly more sensitive measure since it squares the errors before averaging, highlighting larger errors. The RMSE of 0.2635 suggests a relatively low spread of errors around the mean error, which is favorable for predictive accuracy.

The Kafka streaming platform was utilized to simulate real-time data handling and prediction. The Kafka Producer sent feature data to a Kafka topic, and the Kafka Consumer used the ARIMA model to make real-time predictions. The predicted traffic flow from the Kafka real-time streaming was observed to be 0.4508, demonstrating the model's practical application in a streaming context.

The Linear Regression model, while straightforward and easy to implement, struggled with the nonlinear nature of the time-series traffic flow data. In contrast, the ARIMA model, more suited for time-series forecasting, showed improved performance but still lacked precision in peak predictions. The disparity in model performance underscores the need for more complex models or additional feature engineering to better capture the dynamics of traffic flow. Future iterations might explore more sophisticated models like LSTM (Long Short-Term Memory networks) or hybrid models that combine machine learning with traditional time-series forecasting techniques.

These approaches could potentially enhance accuracy and adapt better to variability in traffic data. These findings can guide further iterations of the project, particularly in refining models and optimizing feature selection to improve prediction accuracy further.

Conclusion

This project successfully demonstrates the application of Apache Kafka for real-time data streaming and predictive analysis in traffic management. By harnessing real-time traffic data, conducting time-series analysis, and deploying predictive models, we were able to forecast traffic flow patterns with an MAE of 0.358 and RMSE of 0.417 for linear regression, and an MAE of 0.241 and RMSE of 0.264 for the ARIMA model. These results signify a reasonable degree of accuracy, indicating the models' potential utility in practical traffic management scenarios.

The introduction of more complex machine learning techniques, such as neural networks or ensemble models, could further refine accuracy in future work. These models are particularly adept at understanding complex and nonlinear time-series data, which is characteristic of traffic flows. To accommodate larger datasets and enhance real-time processing capabilities, fine-tuning the Kafka setup could provide more robust scalability and efficiency. Furthermore, Initial model validations were limited to a single location due to computational constraints. Future developments should expand testing to multiple locations to ensure the models' adaptability and scalability across varied traffic ecosystems. Finally, Integrating additional data such as weather conditions, ongoing events, and roadworks could significantly improve the predictive models by providing more context to the traffic flow data.

By advancing these aspects, the project can evolve into a more versatile tool for traffic management, offering precise and actionable insights for urban planning and congestion mitigation. The real-time predictive capabilities demonstrated by the Kafka-integrated system highlight its potential to transform traffic management strategies in urban environments.