Final Report: Real-Time Traffic Prediction with Kafka and Machine Learning

-Rujuta Parulekar

Introduction

Urbanization has significantly increased the demand for better traffic management systems. Efficiently predicting traffic flow patterns in real-time can provide immense value in reducing congestion and improving mobility within cities. In this project, I aim to develop a real-time traffic prediction system using Apache Kafka for data streaming and Machine Learning models for predictive analysis. The project is divided into three phases: Kafka setup for real-time data streaming, Exploratory Data Analysis (EDA) on time-series data, and developing a basic predictive model for traffic flow forecasting.

The following report elaborates on each phase, highlighting the methodology, results, and analysis, along with suggested improvements. The deliverables include data streaming scripts, time-series analysis, and model predictions, which are evaluated using relevant metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). Additionally, feature engineering and hyperparameter tuning are performed to improve the prediction accuracy of the models.

Algorithm Overview

The following algorithm outlines the complete approach to real-time traffic flow prediction:

1. Kafka Setup:

- Initialize Kafka producer and consumer.
- Stream traffic data in real-time with appropriate intervals.
- Consume and process data continuously.

2. Exploratory Data Analysis (EDA):

- Load traffic data from the consumer.
- Plot traffic flow over time and generate ACF/PACF plots to identify patterns.
- Identify key insights such as temporal dependencies and seasonality.

3. Feature Engineering:

- Create rolling averages to smooth traffic data.
- Add time-based features like `hour_of_day` and `day_of_week`.
- Introduce lag features to incorporate previous traffic flow values.

4. Model Training:

- Train a Linear Regression model using the engineered features.
- Train a Random Forest model and perform hyperparameter tuning using Grid Search.

5. Model Evaluation:

- Evaluate both models using MAE and RMSE.
- Cross-validate the Random Forest model to ensure robustness.

6. Visualization:

- Plot the actual vs predicted traffic flow for each model.
- Compare models and provide insights into their performance.

Phase 1: Kafka Setup and Data Streaming

In Phase 1, the objective was to set up Apache Kafka for real-time data streaming. The traffic flow dataset was loaded into a Kafka producer, which streamed the data to a Kafka topic in real-time, simulating real-world scenarios where traffic data is continuously updated. The Kafka consumer was designed to subscribe to this topic and process the streamed data as it arrived.

1. **Producer:** The Kafka producer script (`kafka_producer.py`) reads the traffic flow dataset in small intervals and sends it as messages to the Kafka topic. A one-second delay was introduced between messages to simulate real-time streaming.

2. **Consumer:** The Kafka consumer script (`kafka_consumer.py`) subscribes to the `traffic_topic`, consuming the data in real-time. The data was processed and stored locally for further analysis in Phase 2.

The real-time data streaming ensures that the traffic data is continuously processed, mimicking real-world scenarios such as traffic monitoring and live predictions for congestion management.

Key Achievements:

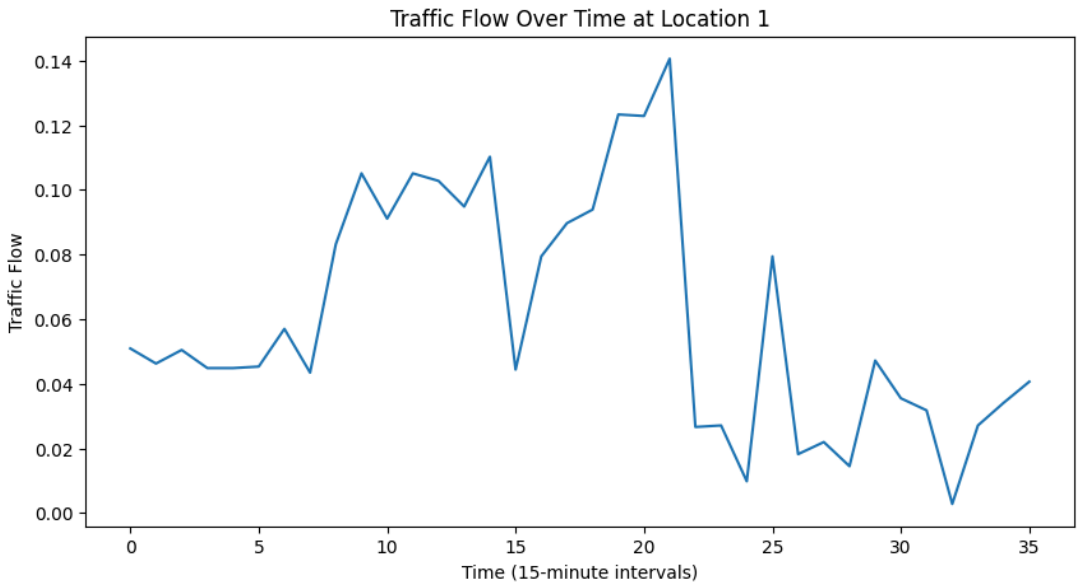
- Real-time simulation of traffic flow using Kafka.
- Proper handling of data integrity between the producer and consumer.

Phase 2: Exploratory Data Analysis (EDA)

In Phase 2, the goal was to explore and analyze the time-series nature of the traffic flow data. By performing EDA, I aimed to understand temporal patterns, such as daily trends and potential seasonality in traffic flow.

Data Visualization:

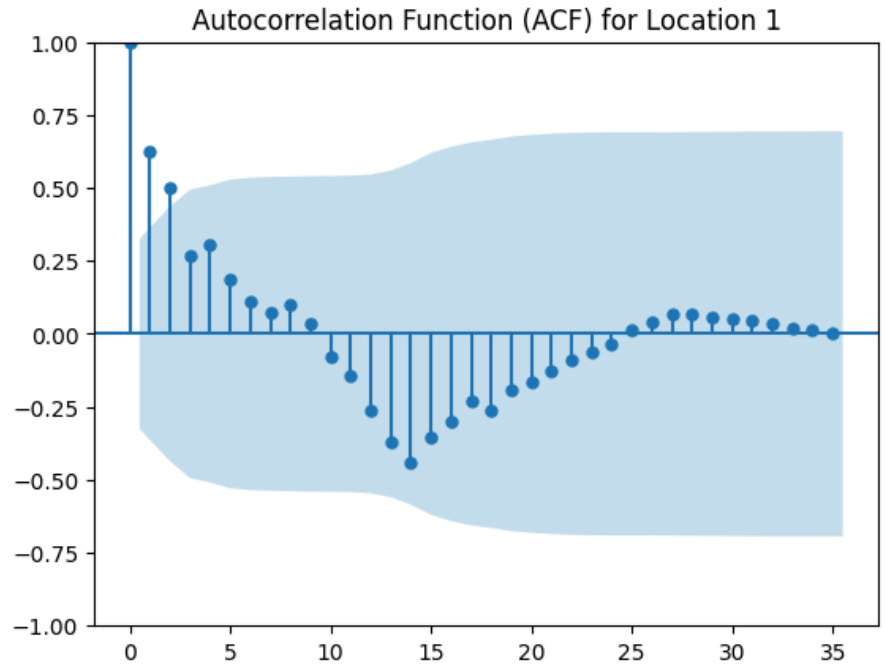
1. **Time-Series Plot:** The time-series plot shows the traffic flow over time at Location 1. The traffic flow fluctuates with several peaks and dips over the course of 35 time intervals (each representing 15 minutes). Significant traffic spikes are observed around interval 20, followed by sharp declines.

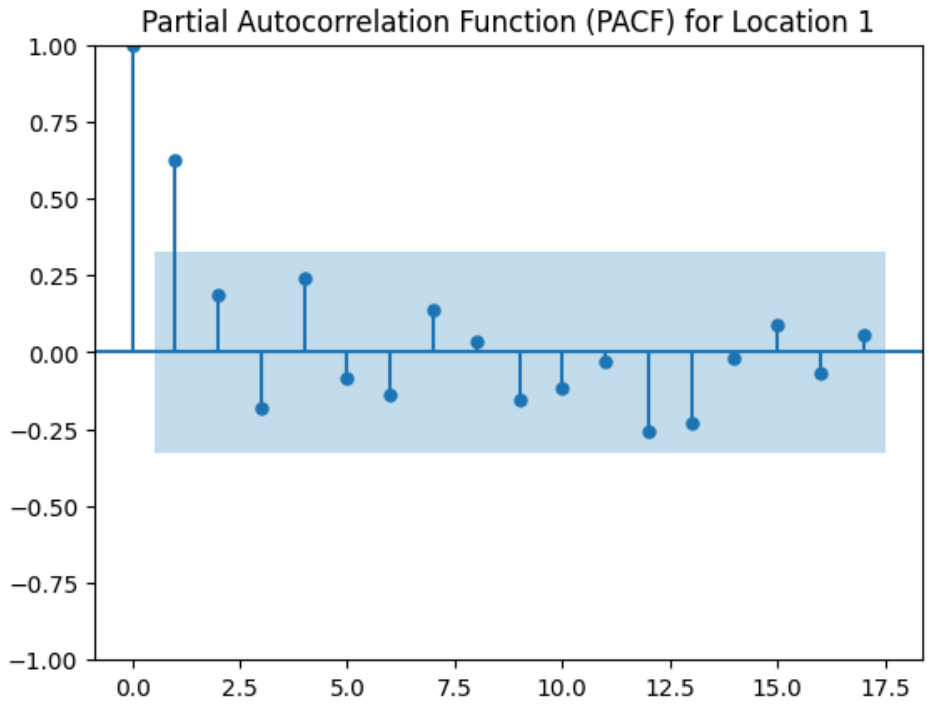


2. **Autocorrelation and Partial Autocorrelation (ACF/PACF):**

ACF Plot: The autocorrelation function (ACF) shows that traffic flow is positively correlated over the first 10-15 lags, meaning that traffic flow is dependent on its recent past.

PACF Plot: The partial autocorrelation function (PACF) indicates that only the first lag has a strong influence on future traffic flow, suggesting that recent intervals are more informative than distant ones.





Key Insights:

- Traffic flow exhibits short-term dependencies, as shown in the ACF and PACF plots.
- These insights led to the development of lag-based features and rolling averages for the predictive model in Phase 3.

Phase 3: Basic Traffic Flow Prediction

The objective of Phase 3 was to develop and evaluate predictive models using the insights gained from the EDA. Two models were implemented: Linear Regression and Random Forest. The models were evaluated using performance metrics such as MAE and RMSE, and their predictions were visualized.

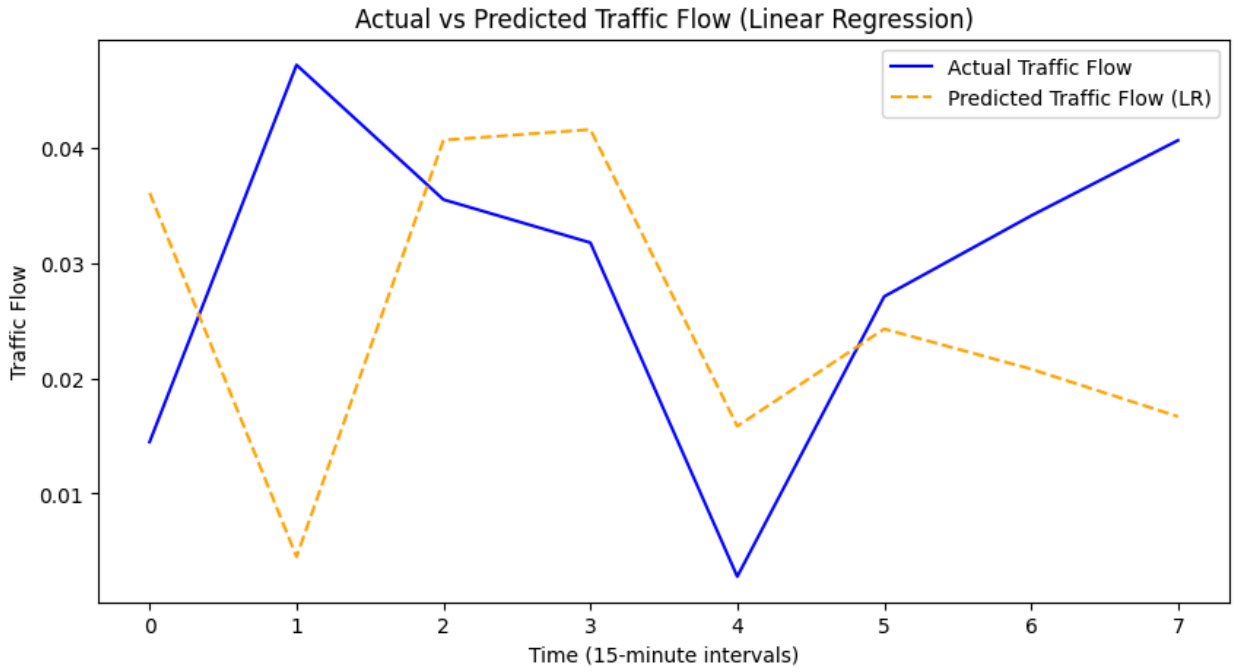
Feature Engineering:

Based on the EDA, the following features were created:
Rolling Mean: A 1-hour rolling average (4 intervals) was used to smooth the traffic data.
Time-based Features: `hour_of_day` and `day_of_week` features were added to capture periodic patterns in traffic.
Lag Features: `lag_1` and `lag_2` were introduced to incorporate previous traffic values as predictors.

Model 1: Linear Regression:

The linear regression model was trained using the time-based, lag, and rolling average features. While the model captured the general trend in traffic flow, it struggled to predict sudden spikes or drops.

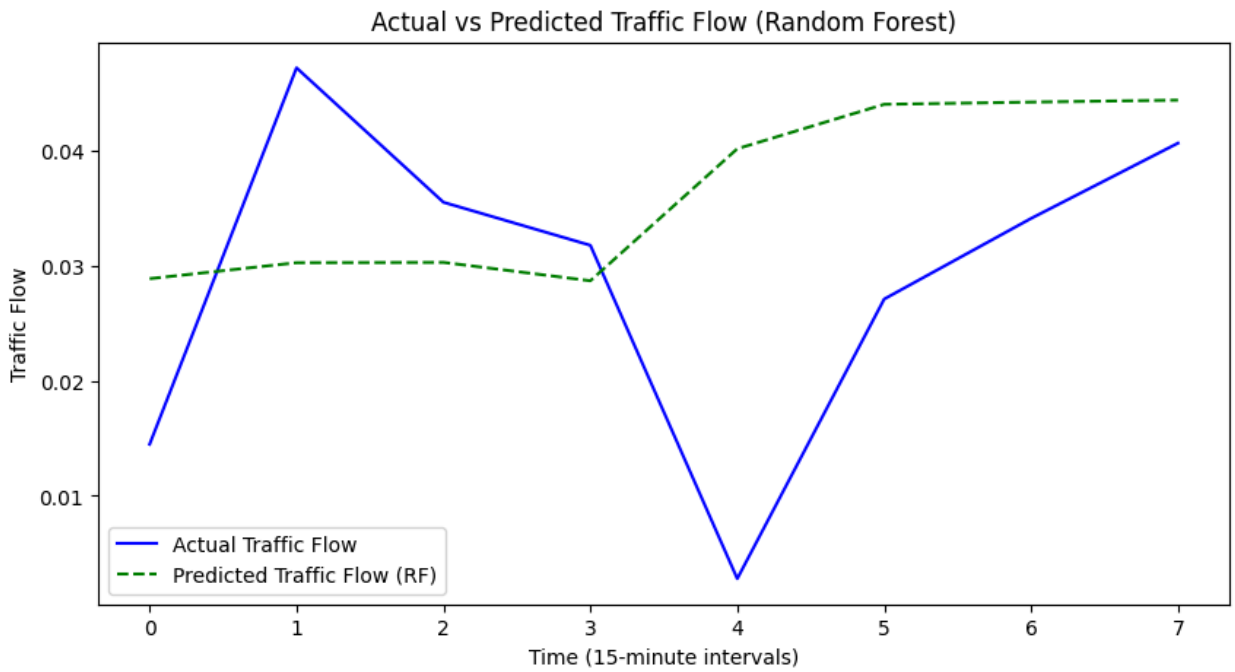
MAE: 0.0165
RMSE: 0.0204



Model 2: Random Forest:

The Random Forest model provided better performance by capturing non-linear patterns in the data. However, it still faced challenges with over-smoothing predictions and missed sudden spikes.

MAE: 0.0135
RMSE: 0.0171
Cross-Validation MAE: 0.0275



Comparison of Models:

Random Forest outperformed Linear Regression in both MAE and RMSE, indicating better predictive accuracy. However, both models struggled with sharp traffic changes, suggesting the need for more advanced models like Gradient Boosting or time-series-specific models like ARIMA.

Conclusion

In this project, I successfully implemented a real-time traffic flow prediction system using Apache Kafka for data streaming and Machine Learning models for forecasting. Both the Linear Regression and Random Forest models showed promising results, with Random Forest outperforming Linear Regression. However, both models struggled with predicting sudden spikes in traffic flow, highlighting the need for more sophisticated models in future iterations.

By following the algorithm outlined above, further improvements can be made through advanced models, hyperparameter tuning, and the incorporation of additional features such as weather conditions or event-based traffic anomalies.

Next Steps:

Model Refinement: Experiment with models like Gradient Boosting and ARIMA to improve accuracy during sudden traffic changes.

Hyperparameter Tuning: Further tune the Random Forest and other models for optimal performance.

Additional Features: Incorporate weather or event-based data to improve the model's understanding of external factors influencing traffic flow.