Streamlined Traffic Prognosis using Flexible Reservoir Sampling and Regression Methods

V. R. Srividhya
Department of CSE, B.M.S. College of Engineering, Affiliated to Visvesvaraya Technological University, Belagavi-590018, India
vrs.phd@gmail.com (corresponding author)

Kayarvizhy
Department of CSE, B.M.S. College of Engineering, Affiliated to Visvesvaraya Technological University, Belagavi-590018, India
kayarvizhyn.cse@bmsce.ac.in

Received: 9 December 2024 | Revised: 12 January 2025 | Accepted: 15 January 2025

 DOI: https://doi.org/10.48084/etasr.9889

ABSTRACT

The increased use of Internet of Things (IoT) technologies has resulted in an exponential increase in real-time data streams, particularly in smart city applications, especially for traffic management. Accurate prediction of traffic parameters in such environments is critical for optimizing traffic flow, reducing congestion, and enabling efficient resource management. This study presents an approach to the prediction of traffic intensity and occupancy using IoT streaming data, time series analysis, and machine learning algorithms. The proposed method includes preprocessing steps such as data interpolation to handle missing values and temporal alignment, followed by feature extraction and model training using a combination of regression and sampling techniques. Experiments were carried out on a real-world IoT traffic dataset, and the results show significant improvements in the prediction accuracy in terms of MAPE values. It also predicts the complex event of congestion, using a rule-based algorithm. The proposed method can pave the way for smarter and more efficient urban infrastructure.

Keywords-IoT streaming; traffic data; adaptive; reservoir sampling; regression; prediction

I. INTRODUCTION

The Internet of Things (IoT) has gained widespread prominence around the world. It is a well-connected network of elements, including a variety of sensors. These devices continuously generate data by monitoring various parameters, which may be analog, digital, or combined. The purpose of networking these devices extends beyond passive observation. The establishment of this interconnected infrastructure enables the development and implementation of a wide range of groundbreaking applications. These data can offer meaningful information to improve the quality of applications. However, there is a growing need for real-time or near-real-time analysis. Supply chain systems [1], intelligent traffic monitoring [2], network monitoring [3], and fraud detection [4] are a few use cases that require the deployment of such analytical frameworks. Under these circumstances, historical data serve as a foundational resource, enabling the transformation of streaming data, analyzed in real-time or near-real-time, into practical intelligence.

Sensor data is continuously generated in real time as event streams, exhibiting intricate patterns, each corresponding to a distinct occurrence. Accurate interpretation of these unique events requires consideration of the prevailing context while minimizing errors, thus facilitating decision-making. This necessity underscores the relevance of the domain known as Complex Event Processing (CEP). CEP encompasses a collection of techniques for capturing and analyzing data streams as they arrive to identify threats and opportunities in real time. CEP enables systems and applications to respond to trends, events, and patterns in the data [5]. It is used predominantly in scenarios characterized by high-speed event generation, where strict requirements for minimal latency must be upheld without compromise. CEP is used in a wide variety of applications such as data center security [6] and energy management systems [7].

However, in-depth domain knowledge is required to set up CEP. It is primarily declarative but also reactive by nature. This is achieved by correlating data streams as they arrive. Historical data can be used by combining CEP with Machine Learning (ML). In the specific case of disaster management systems, such as tsunamis [8], the idea is to predict the disaster, rather than just detecting it. ML algorithms, combined with statistical data, have found great use in predicting overall production costs in supply chain management [9] and energy consumption [10]. Although ML algorithms can leverage historical data, their primary drawback is that they lack scalability and the capacity to process multiple data streams

simultaneously. Hence, there is a need to combine the strengths of the ML and CEP techniques. Correct conclusions can be drawn by CEP using input from accurate predictions of ML techniques.

Event processing encompasses analyzing events and identifying patterns. Complex events are a combination of events that, when analyzed, can alert to several useful information, ranging from opportunities to threats. There are many applications where CEP is used with IoT, acting on data streams to infer details of complex events. This is based on a number of techniques, including event filtering, aggregation, pattern detection, and abstraction [11]. In [12], an Event Driven Architecture (EDA) was proposed based on CEP, where large-scale traffic data was analyzed. This approach facilitates automated solutions for decision support systems, enhancing traffic management. Event Processing Agents (EPAs) were utilized to connect streams of progressively abstract types of events. ESPER CEP was proposed for the implementation, overcoming its inherent ability to realize distributed architecture using message-oriented middleware. In [13], a distance-based event detection system was proposed to detect complex low-level events such as "ball possession" and "kicking the ball" in football. In [14], CEP technology was used to analyze data streams and uncover attacks on IoT applications that used the MQTT and CoAP protocols.

Progress has been made in using predictive analytics in conjunction with statistics and CEP to provide predictive solutions. In [15], a probabilistic event processing network was utilized to detect complex events, employing a multi-layered Bayesian model to predict future events. This approach utilized the Expectation Maximization (EM) algorithm, which incurs a high computational cost. As the size of the training dataset increases exponentially, the complexity of the model also increases, rendering it inept for massive IoT applications. In [16], an algorithm based on the sliding window model on data streams was proposed to extract Weighted Maximal Frequent Patterns (WMFPs). Deep learning approaches have also been used for time series prediction. In [17], the LSTM and GRU methods were used to predict confirmed and death cases of COVID-19 in three countries.

Data streams refer to data that increase over time. The salient characteristics of the data streams [18] can be classified as follows:

- The upcoming data in the data stream remain unseen until it arrives, in contrast to historical data that provide visibility into how the data flows over time.
- The data elements follow a time series pattern, as they arise one by one.
- Data streams are quite large, hence, there is a difficulty in storing such huge amounts of data.

Due to these characteristics, deterministic algorithms cannot be applied to data streams. However, there is an increasing need to analyze data streams in many applications, such as anomaly detection [19], monitoring of IP packets [20], health analytics [21], and intelligent transportation systems [15].

Data streams are formulated as $D_n = \{x_1, x_2, .., x_n\}$, where x_i denotes the elements entering over time and observed at time t. Access is currently limited to this data, with no visibility into the upcoming data stream. Any data generation that follows this fashion is categorized as streaming data. The data sources can be telephone records, web documents, clickstreams, financial market data, video and audio streams, etc. An extensive amount of work has been conducted to understand the behavior of these streams [22].

Time series analysis [23], is naturally time-dependent and typically has consistent time stamps. Therefore, at any point in time t, $D_t = \{x_1, x_2, .., x_t\}$ is only observed. Each of these x_i values can be a group of values. Also, since all the data cannot be stored in memory, there is a need to find a method to compute $F(D_t)$ at any time t. Several methods have been described in various works [24]. One way to achieve this is through sampling. Sampling techniques have been extensively examined for a variety of domains, and optimal sampling methods have been presented [25-26]. Several sampling techniques can be used for IoT streaming data, such as time-based, event-based, and adaptive sampling techniques. These techniques improve real-time performance and reduce storage requirements. This can also lead to an increased error rate unless the appropriate technique is chosen.

Reservoir sampling [27] is a significant adaptive statistical technique that was developed to allow analysis of large-scale datasets while optimizing memory utilization, which is a crucial consideration given the limitations of the period it was formulated. This work explores an adaptation of reservoir sampling to enable data prediction by integrating historical and near real-time streaming data, implemented within the framework of Adaptive Reservoir Sampling Regression (AdReSaR). In previous studies on reservoir sampling, reservoir methods were generalized to select balanced samples [28], but the decision on a unit at the beginning of the stream can be made quite late. In [29], a data analytics system was explored in the realm of edge computing for approximate computing.

This work utilizes the combined strengths of CEP and ML. The AdReSaR algorithm is used for prediction, utilizing reservoir sampling for model training alongside a regression technique. The trained model can be updated with the arrival of new data. The sampling size is determined using reservoir sampling to optimize performance. This is later fed into a regression model to predict the values. The contributions of this study are as follows:

- Presents a system based on ML and CEP for forecasting IoT applications.
- Proposes an adaptive prediction algorithm for IoT data streams using reservoir sampling and regression.
- Performs comparative analysis with other experimental techniques.
- Evaluates the AdReSaR algorithm on near-real-time streaming data for congestion prediction.

II. PROPOSED ARCHITECTURE

Figure 1 illustrates the proposed system. Streaming data from a real-time data source is fed to the AdReSaR algorithm along with historical data. The predicted data are fed into the CEP system. The output of the CEP system is the predicted complex event.

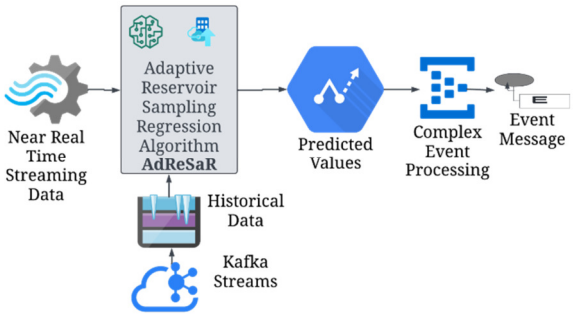


Fig. 1. Proposed system using AdReSaR.

A. Preliminaries

1) Dataset Description

The busiest traffic areas in Madrid include several key zones, particularly in the city center and around the major intersections and transportation hubs. Traffic congestion is prevalent in some important places such as Atocha, Puerta del Sol, M-30, and M-40. The Madrid city council offers a wealth of information related to traffic and public services within the city. Several traffic observation sensors at fixed high-traffic spots in the city measure various traffic characteristics. Table I presents details on the specific parameters that were used in this study.

TABLE I. PARAMETERS CONSIDERED

Parameter	Explanation
idelem	Unique identifier used to locate traffic measurement points on a map
intensidad	Vehicle count passing through a measurement point per hour.
ocupacion	Occupancy (%), how much of time vehicles were occupying the lanes

These data are published in XML format [30]. Data was collected for a month (March 2024) and stored as historical data. The AdReSaR algorithm was applied to the historical data to predict the results for *intensidad* and *ocupacion* for two locations.

2) Time Series Data

Data are recorded for each observation at regular intervals. The order of the observations matters much and the data are used to analyze changes or patterns. Let y_t represent the value of a time series at time t. The time series equation is shown in (1), where $f(t)$ is the function representing the underlying pattern of data and ε_t the error at time t.

$$y_t = f(t) + \varepsilon_t \quad (1)$$

3) Regression

Regression is a statistical method to model the relationship between one or more independent variables (often referred to as predictors or features) and a dependent variable (also known as the outcome or response). Regression analysis aims to understand and quantify the relationship between variables and make predictions. Its mathematical representation is shown in (2), where β_0 and β_1 are the y-intercepts and slope coefficients, ε is the error term, y is the dependent variable, and x is the independent variable.

$$y = \beta_0 + \beta_1 x + \varepsilon \quad (2)$$

Other statistical methods, such as ARIMA and enhanced ARIMA models [31], have been applied for regression tasks. There is a current trend toward utilizing ML models, such as Support Vector Regression (SVR), K-Nearest Neighbors (KNN), and Artificial Neural Networks (ANN), due to their predictive power and flexibility.

4) Support Vector Regression (SVR)

SVR is an ML algorithm for regression analysis [32], adapted from support vector machines which were originally used for classification. Unlike the other algorithms, it focuses on fitting the best hyperplane within a certain margin (ε tube) around the actual values. Another advantage of SVR is the availability of kernel functions, which are useful for transforming the data input into a higher dimension, making it possible to find a hyperplane that separates or fits the data more effectively. SVR is versatile, handling both linear and nonlinear regression tasks, through the use of various kernel functions. The Radial Basis Function (RBF) can tune the gamma parameter. This study experimented with other kernels before choosing RBF due to its performance.

5) Reservoir Sampling

Sampling is the process of selecting a subset from a larger population and using it for analysis, based on the properties of the smaller subset chosen. There are several sampling methods whose efficiency depends on the dataset considered. A simple random sampling can be represented as in (3), where N is the population and n is the sample size. Here, each individual in sampling has an equal opportunity to be selected.

$$P\,(selecting\ specific\ sample) = n!/(N.(N-1).(N-2).(N-n+1)) \quad (3)$$

Reservoir sampling employs a method that manages an infinite data stream of size R and gathers a randomly distributed sample of at least R elements from the unbounded data stream. This is significantly important because this sampling method ensures that the earliest R components primarily remain in the reservoir. This is fulfilled when the i^{th} item is identified (where $i > R$). The objective is to sample k elements from a stream. Upon reaching the n^{th} element, the following conditions must be met:

- Every element has exactly a k/n probability of being in the sample.
- Exactly k elements have been sampled.

It takes as input a stream of items and an integer k (the desired size of the reservoir), and it outputs a reservoir containing a random sample of size k.

B. Adaptive Reservoir Sampling Regression Algorithm (AdReSaR)

AdReSaR represents the initial step in addressing dynamic IoT data. Traditionally, ML models are trained on historical data to facilitate prediction. The quantitative characteristics (statistical characteristics) of the data may change. This can make it difficult to update the model to suit this change, and not updating it will result in a deterioration of the prediction model's effectiveness. In such a typical scenario, it is best to retrain the model by applying sampling and regression techniques on historical data and near-real-time data streams. This is materialized by assessing the prediction error and fine-tuning the model accordingly. Figure 2 shows the main steps of this algorithm. The objective is to predict parameters of time series data using various models, interpolation techniques, and sampling strategies to minimize prediction errors.

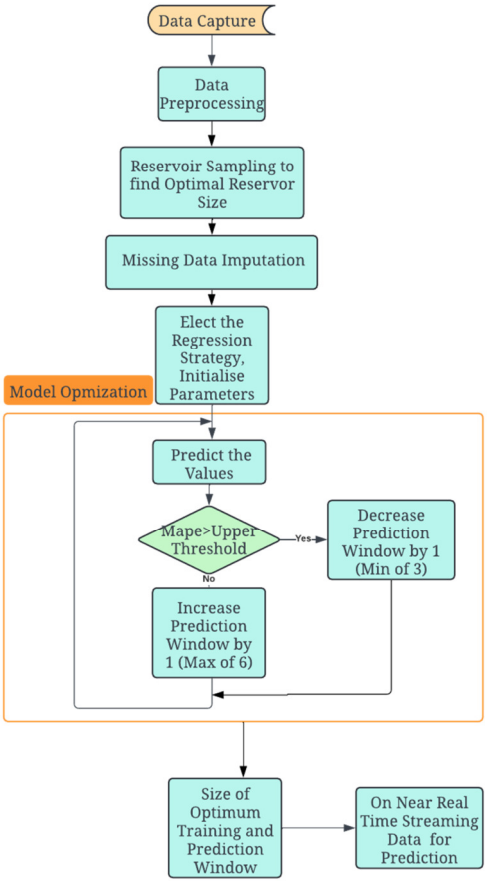


Fig. 2. The AdReSaR algorithm.

1) Data Capture, Storage, Preprocessing and Reservoir Sampling

Data were collected using Kafka, a distributed streaming platform that is scalable and fault-tolerant, allowing the storage and processing of data streams [33]. Historical data was acquired from streaming sources [30] at 15-minute intervals and subsequently archived in a file for subsequent retrieval and analysis. To augment the data's temporal resolution, interpolation techniques were employed for the parameters of *intensidad* and *ocupacion* to resample the data at 5-minute and 1-minute intervals. Reservoir sampling was implemented on the above data to find the reservoir size. In this reservoir of sample points, missing values were imputed using cubic and linear interpolation.

2) Model Optimization

The parameters for the training and prediction windows were initialized. The regression models considered include the Gaussian Process Regressor (GPR) with a Matérn kernel, the Stochastic Gradient Descent (SGD) regressor with an RBF kernel, and the Support Vector Regression (SVR) with an RBF kernel, both with and without sampling. The analysis involved predicting the values using the chosen regression technique. The values were predicted and the prediction error was determined in terms of Mean Absolute Percentage Error (MAPE). The MAPE value was compared to predefined upper and lower threshold values, and the prediction window size was adjusted accordingly, either increasing or decreasing by 1. The upper and lower threshold values for MAPE were established at 10 and 5 respectively. The final prediction window was chosen as the one with the lowest MAPE, as it is desirable to minimize MAPE [34]. This prediction window size can be applied to near-real-time streaming data to find the predicted values of the parameter of interest.

3) Complex Event Processing (CEP) System

The above-predicted parameter values (historical and near-real-time predicted) can be sent to a CEP system to gather useful insights such as congestion prediction, environmental impact monitoring [35], integrated smart city solutions [36], etc. Figure 3 depicts a typical CEP system.

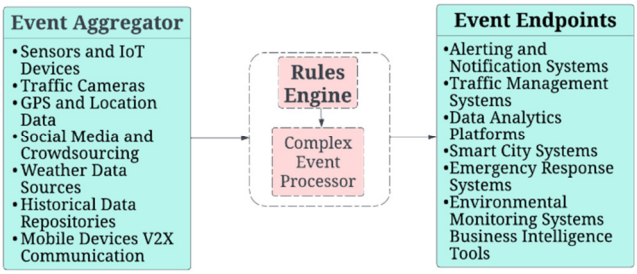


Fig. 3. A typical CEP system.

III. EXPERIMENTS AND TRIALS

A. Rule Base for Congestion Prediction

The data collected in the previous steps were processed to detect a complex congestion event based on threshold values ($Intensidad_{thresh}$, $Ocupacion_{thresh}$) incorporated into a simple rule-based engine. This can be verified graphically by plotting a graph of *intensidad* vs *ocupacion*. Algorithm 1 shows a sample rule-based algorithm for predicting congestion.

```
Algorithm 1: Rule-based Congestion
  Prediction
Input: (Intensidad, Ocupacion)
Output: Congestion Condition
If (Intensidad<Intensidad_thresh) ∧
  (Ocupacion< Ocupacion_thresh) then
  Congestion_Condition = 'Free-flowing'
elif(Intensidad>=Intensidad_thresh)∧
  (Ocupacion<Ocupacion_thresh) then
  Congestion_Condition = 'Efficient Movement'
elif(Intensidad>=Intensidad_thresh)∧
  (Ocupacion>=Ocupacion_thres) then
  Congestion_Condition = 'Congestion'
else:
  Congestion_Condition = 'Slow Movement'
```

B. Comparison Experiments

Experiments were conducted using combinations of regression and sampling approaches to predict the values of *intensidad* and *ocupacion* and the complex event of congestion. The list of techniques used and their prediction errors are illustrated in terms of MAPE:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n} |(X_i - Y_i)/X_i| \quad (4)$$

where X_i denotes the actual value, Y_i denotes the predicted value, and n is the number of fitted points (total predicted values). The prediction results for traffic intensity (*intensidad*) and occupancy (*ocupacion*) for two different locations were plotted, indicating close alignment with the original data. This level of accuracy was achieved because the model incorporates any prediction errors and updates itself accordingly, due to the efficient algorithm design, effectively preventing errors from propagating. Since the error of predicted values is minimized, subsequent inputs to the CEP module can also produce results with smaller errors. The same method for near-real-time streaming data was adopted, by streaming live data from the website for 400 minutes.

IV. COMPARATIVE ANALYSIS OF PERFORMANCE

Experiments were carried out employing various approaches, including multiple regression methods, sampling techniques, and a combination of both regression and sampling, to predict the values of *intensidad* and *ocupación*.

A. Methods for Performance Comparison

The following methods were employed to compare performance under different conditions.

- GPR with Matérn kernel,
- SGD regressor with an RBF kernel,
- SVR with an RBF kernel with and without sampling.

Additionally, wherever appropriate, linear and cubic interpolation at 1-minute and 5-minute intervals was explored to address data continuity requirements.

1) Gaussian Process Regressor (GPR) with Matérn Kernel

GPR with a Matérn kernel [37] predicts values by considering the covariance between data points, providing probabilistic predictions, and capturing nonlinear relationships. The Matérn kernel allows GPR to handle smoothness in the data, where the degree of smoothness is controlled by its parameters. Although this approach can yield good results, it takes more time to run.

2) Stochastic Gradient Descent (SGD) Regressor

The SGD regressor [38] performs linear regression using SGD, in which the model parameters are updated iteratively for each training example, making it efficient for large datasets. The model minimizes the loss function by adjusting the weights based on gradient updates. It stops either after reaching the maximum iterations or when the error improvement becomes smaller than the threshold.

3) Support Vector Regression (SVR)

SVR was used with an RBF kernel with and without sampling. SVR [39] aims to identify a function that approximates the target values within a specified margin of error (ε) while maintaining the smoothest possible model. This approach minimizes complexity and deviation from the true values. SVR is versatile, handling both linear and nonlinear regression tasks using various kernel functions. Common kernel functions are linear (for data that are linearly separable), polynomial (for polynomial relationships between input features and output), sigmoid (for some types of nonlinear relationships), and RBF (for nonlinear relationships). The RBF kernel is a common choice in SVR because it can capture non-linear relationships by mapping the input space into a higher-dimensional space. The key parameters of SVR with RBF are: The C regularization parameter (higher values of C aim to minimize error more strictly but may lead to overfitting) and gamma (the kernel coefficient of RBF) [40]. The values of the parameters used for these experiments were $C = 1e3$ and $gamma = 0.1$. These values were chosen because larger or smaller values can lead to overfitting or underfitting. The use of SVR also requires preprocessing. For SVR, it is common to scale both the features and the target variable, because it is sensitive to the magnitude of the data.

The imputation of missing data was also examined. Several techniques such as mean/median/mode imputation, hot-deck imputation, KNN imputation, and simple interpolation techniques can be used. Given the nature of time series data and the need to estimate missing values based on neighboring time points to maintain continuity, interpolation was selected over other methods. Linear and cubic interpolations were examined. This method was also combined with reservoir sampling before applying regression. This allowed analyzing model behavior with different sample sizes while preserving the dataset's statistical properties.

B. Results and Analysis

Figures 4 and 5 and Table III show the results. Table II delineates the legend for the abbreviations employed in the graphs, whereas Table III shows the MAPE accuracy metrics

achieved by the algorithms across the intensity and occupancy parameters for two different locations.

The results show that sampling offered significant improvement in MAPE values. Comparing 1-minute to 5-minute interpolation, the former had lower MAPE values compared to the latter. The comparison of linear with cubic interpolation shows that for 1-minute interpolation, linear yielded better results, while for 5-minute interpolation, cubic yielded lower MAPE.

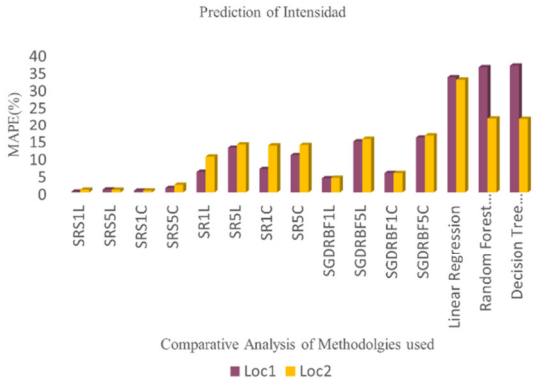


Fig. 4. Comparative prediction accuracy for *intensidad* at locations 1 and 2.

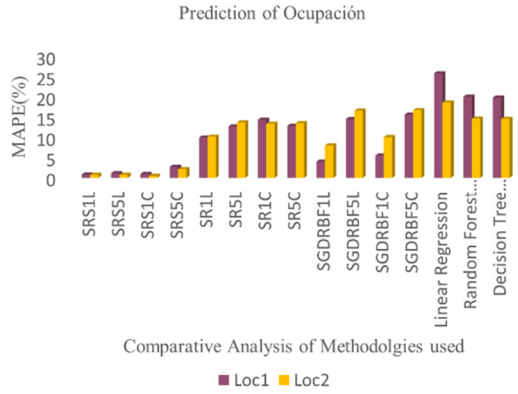


Fig. 5. Comparative prediction accuracy for *ocupación* at locations 1 and 2.

TABLE II. ABBREVIATIONS USED IN COMPARISONS

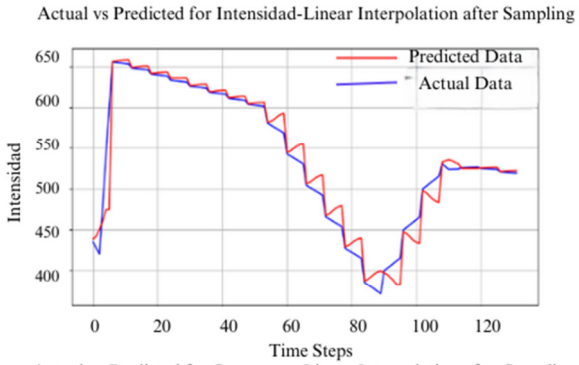
Algorithm	Details
SRS1L	SVR with sampling (1-minute linear interpolation)
SRS5L	SVR with sampling (5-minute linear interpolation)
SRS1C	SVR with sampling (1-minute cubic interpolation)
SRS5C	SVR with sampling (5-minute cubic interpolation)
SR1L	SVR without sampling (1-minute linear interpolation)
SR5L	SVR without sampling (5-minute linear interpolation)
SR1C	SVR without sampling (1-minute cubic interpolation)
SR5C	SVR without sampling (5-minute cubic interpolation)
SGDRBF1L	SGD without sampling (1-minute linear interpolation)
SGDRBF5L	SGD without sampling (5-minute linear interpolation)
SGDRBF1C	SGD without sampling (1-minute cubic interpolation)
SGDRBF5C	SGD without sampling (5-minute cubic interpolation)
Lin Reg	Linear regression
Ran For Reg	Random forest regression
Dec Tree Reg	Decision Tree regression

TABLE III. MAPE VALUES FOR ALGORITHMS

Method	Intensity		Occupancy	
	Loc1	Loc2	Loc1	Loc2
SRS1L	0.2	0.76	0.82	0.76
SRS5L	0.8	0.75	1.14	0.75
SRS1C	0.48	0.52	0.96	0.52
SRS5C	1.24	2.16	2.74	2.16
SR1L	5.84	10.17	9.95	10.17
SR5L	12.73	13.69	12.7	13.69
SR1C	6.66	13.36	14.35	13.36
SR5C	10.61	13.5	12.84	13.5
SGDRBF1L	4.03	4.13	4.03	8.01
SGDRBF5L	14.51	15.23	14.51	16.57
SGDRBF1C	5.53	5.53	5.53	10.04
SGDRBF5C	15.63	16.21	15.63	16.71
Lin Reg	33.16	32.50	25.80	18.6
Ran For Reg	36.06	21.35	20.04	14.63
Dec Tree Reg	36.53	21.25	19.82	14.57

Figures 6 and 7 show the predictive analysis on near-real-time streaming data, demonstrating a strong alignment between predicted and actual values, akin to the results obtained from historical data. This approach exhibits potential for future extension to real-time applications, enabling on-the-fly analysis of streaming data.

(a)



(b)

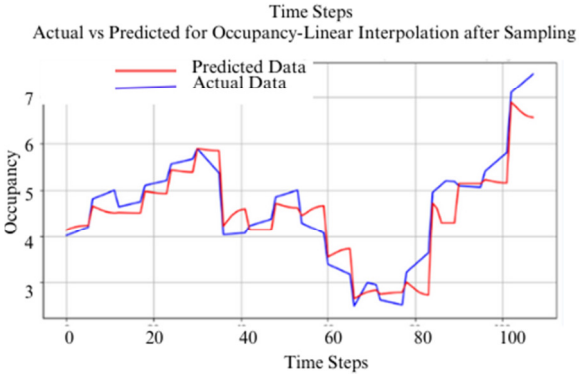


Fig. 6. Prediction results on near-real-time streaming data for (a) *intensidad* and (b) *ocupación* for Location 1.

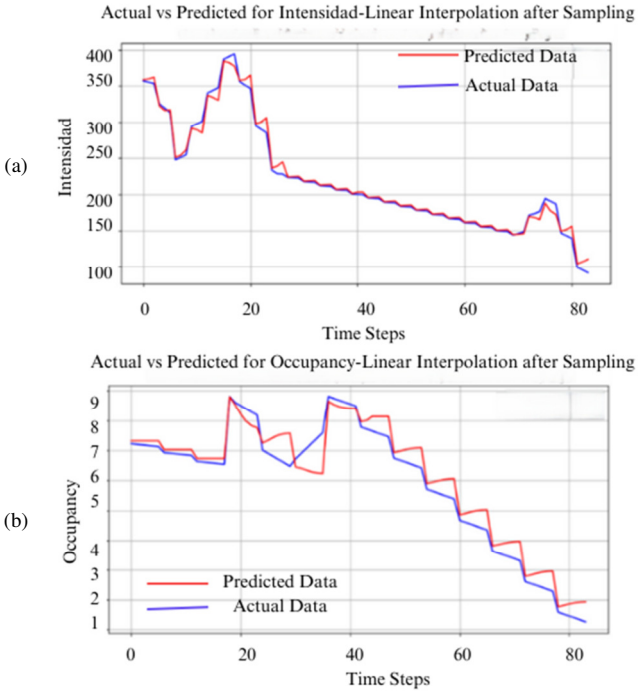


Fig. 7. Prediction results on near-real-time streaming data for *intensidad* and *ocupación* for Location 2.

C. Congestion Prediction Using CEP Rules

Figure 8 shows an example use case of a CEP system with threshold values.

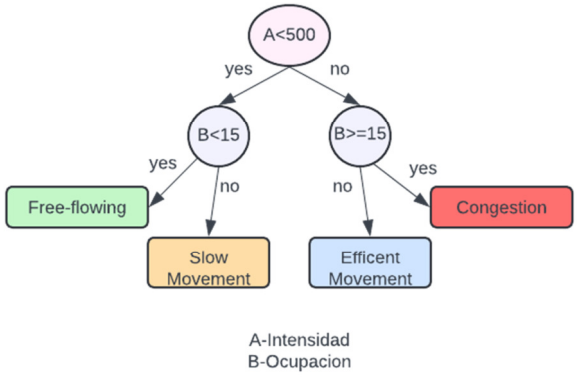


Fig. 8. An example use case for threshold values for congestion prediction.

By generating a graphical representation and implementing the prescribed rule-based framework, potential congestion points can be effectively identified. From the rule set in Figure 8, the graph in Figure 9 highlights these probable congestion instances. This can be extended to be implemented on a continuous stream of incoming data using CEP systems for large-scale and real-time complex event prediction.

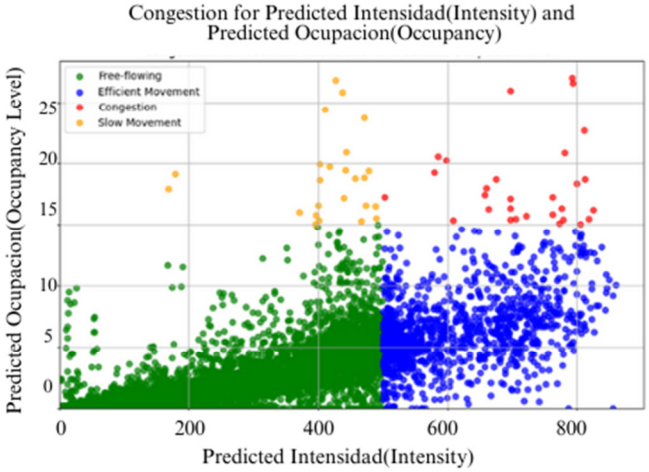


Fig. 9. Congestion prediction for Location 1 using the rules in Figure 8.

V. CONCLUSION AND FUTURE WORK

This study showed that AdReSaR, the proposed algorithm, achieves satisfactory accuracy in predicting traffic intensity and occupancy levels. Its adaptability to non-stationary data leads to significant improvements in prediction performance over traditional regression methods. Beyond forecasting traffic parameters, such as intensity and occupancy, the proposed framework integrates a rule-based algorithm to predict complex events, such as congestion, providing actionable insights for traffic management systems. The findings suggest that adopting the proposed method could significantly enhance real-time decision-making in traffic management systems. Ultimately, this research contributes to ongoing efforts to improve IoT applications in urban environments, paving the way for smarter cities and such real-time applications. Additionally, a simple rule-based approach was used to experiment with CEP, which can be scaled up.

Future research can further explore the integration of real-time data streams to improve the algorithm's responsiveness to dynamic traffic conditions. The proposed method can be adapted for use in other IoT applications, such as environmental monitoring and resource management. Collaborating with urban planners and data scientists could provide valuable insights into practical implementations. The rise of edge computing presents an opportunity to implement the proposed algorithm in decentralized environments, improving scalability and real-time processing.

REFERENCES

[1] B. Yan and G. Huang, "Supply chain information transmission based on RFID and internet of things," in *2009 ISECS International Colloquium on Computing, Communication, Control, and Management*, Sanya, China, Aug. 2009, pp. 166–169, https://doi.org/10.1109/CCCM.2009.5267755.

[2] L. Xiao and Z. Wang, "Internet of things: A new application for intelligent traffic monitoring system," *Journal of networks*, vol. 6, no. 6, pp. 887–894, 2011.

[3] A. Gupta, R. Birkner, M. Canini, N. Feamster, C. Mac-Stoker, and W. Willinger, "Network Monitoring as a Streaming Analytics Problem," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*,

Atlanta, GA USA, Nov. 2016, pp. 106–112, https://doi.org/10.1145/3005745.3005748.

[4] Y. Vivek, V. Ravi, A. A. Mane, and L. R. Naidu, "ATM Fraud Detection using Streaming Data Analytics." arXiv, Mar. 08, 2023, https://doi.org/10.48550/arXiv.2303.04946.

[5] A. Moraru and D. Mladenić, "Complex event processing and data mining for smart cities," in *15th International Multiconference on Information Society*, Ljubljana, Slovenia, 2012.

[6] K. A. Alaghbari, M. H. M. Saad, A. Hussain, and M. R. Alam, "Complex event processing for physical and cyber security in datacentres - recent progress, challenges and recommendations," *Journal of Cloud Computing*, vol. 11, no. 1, Oct. 2022, Art. no. 65, https://doi.org/10.1186/s13677-022-00338-x.

[7] J. Y. C. Wen *et al.*, "A complex event processing architecture for energy and operation management: industrial experience report," in *Proceedings of the 5th ACM international conference on Distributed event-based system*, New York, NY, USA, Jul. 2011, pp. 313–316, https://doi.org/10.1145/2002259.2002300.

[8] S. Shetty, "Disaster Management (Early Information about the Occurrence of Tsunami)," *International Journal for Research in Applied Science and Engineering Technology*, vol. 7, no. 4, pp. 3960–3963, Apr. 2019, https://doi.org/10.22214/ijraset.2019.4662.

[9] K. A. B. Hamou, Z. Jarir, and S. Elfirdoussi, "Design of a Machine Learning-based Decision Support System for Product Scheduling on Non Identical Parallel Machines," *Engineering, Technology & Applied Science Research*, vol. 14, no. 5, pp. 16317–16325, Oct. 2024, https://doi.org/10.48084/etasr.7934.

[10] H. Cai, S. Shen, Q. Lin, X. Li, and H. Xiao, "Predicting the Energy Consumption of Residential Buildings for Regional Electricity Supply-Side and Demand-Side Management," *IEEE Access*, vol. 7, pp. 30386–30397, 2019, https://doi.org/10.1109/ACCESS.2019.2901257.

[11] O. Etzion and P. Niblett, *Event processing in action*. Manning Publications, 2011.

[12] J. Dunkel, A. Fernández, R. Ortiz, and S. Ossowski, "Event-driven architecture for decision support in traffic management systems," *Expert Systems with Applications*, vol. 38, no. 6, pp. 6530–6539, Jun. 2011, https://doi.org/10.1016/j.eswa.2010.11.087.

[13] A. Khan, B. Lazzerini, G. Calabrese, and L. Serafini, "Soccer Event Detection," in *Computer Science & Information Technology*, Apr. 2018, pp. 119–129, https://doi.org/10.5121/csit.2018.80509.

[14] M. Lima, R. Lima, F. Lins, and M. Bonfim, "Beholder – A CEP-based intrusion detection and prevention systems for IoT environments," *Computers & Security*, vol. 120, Sep. 2022, Art. no. 102824, https://doi.org/10.1016/j.cose.2022.102824.

[15] Y. Wang and K. Cao, "A Proactive Complex Event Processing Method for Large-Scale Transportation Internet of Things," *International Journal of Distributed Sensor Networks*, vol. 10, no. 3, Mar. 2014, Art. no. 159052, https://doi.org/10.1155/2014/159052.

[16] G. Lee, U. Yun, and K. H. Ryu, "Sliding window based weighted maximal frequent pattern mining over data streams," *Expert Systems with Applications*, vol. 41, no. 2, pp. 694–708, Feb. 2014, https://doi.org/10.1016/j.eswa.2013.07.094.

[17] N. F. Omran, S. F. Abd-el Ghany, H. Saleh, A. A. Ali, A. Gumaei, and M. Al-Rakhami, "Applying Deep Learning Methods on Time-Series Data for Forecasting COVID-19 in Egypt, Kuwait, and Saudi Arabia," *Complexity*, vol. 2021, no. 1, 2021, Art. no. 6686745, https://doi.org/10.1155/2021/6686745.

[18] P. Arumainayagam, E. Y. A. Charles, and S. R. Kodituwakku, "A Review on processing of data streams," *Research Journal of Computer Systems Engineering - An International Journal*, vol. 2, no. 2, pp. 73–77, Jun. 2011.

[19] Q. Wang, B. Yan, H. Su, and H. Zheng, "Anomaly Detection for Time Series Data Stream," in *2021 IEEE 6th International Conference on Big Data Analytics (ICBDA)*, Xiamen, China, Mar. 2021, pp. 118–122, https://doi.org/10.1109/ICBDA51983.2021.9402957.

[20] A. Fais, G. Lettieri, G. Procissi, S. Giordano, and F. Oppedisano, "Data Stream Processing for Packet-Level Analytics," *Sensors*, vol. 21, no. 5, Jan. 2021, Art. no. 1735, https://doi.org/10.3390/s21051735.

[21] Q. Zhang, C. Pang, S. Mcbride, D. Hansen, Charles Cheung, and M. Steyn, "Towards Health Data Stream Analytics," in *IEEE/ICME International Conference on Complex Medical Engineering*, Gold Coast, Australia, Jul. 2010, pp. 282–287, https://doi.org/10.1109/ICCME.2010.5558827.

[22] C. Q. Jin, W. N. Quan, and A. Y. Zhou, "Analysis and management of streaming data: A survey," *Journal of Software*, vol. 15, no. 8, pp. 1172–1181, Aug. 2004.

[23] J. C. Hardin, "Introduction to Time Series Analysis," National Aeronautics and Space Administration, NAS 1.61:1145, Mar. 1986. [Online]. Available: https://ntrs.nasa.gov/citations/19860014920.

[24] W. F. Velicer and J. L. Fava, "Time series analysis," *Research methods in psychology*, vol. 2, 2003.

[25] A. S. Singh and M. B. Masuku, "Sampling techniques & determination of sample size in applied statistics research: An overview," *International Journal of Economics, Commerce and Management*, vol. 2, no. 11, 2014.

[26] S. Ulas and U. M. Diwekar, "Role of sampling in process design, optimization and control," presented at the AIChE Annual Meeting, Nov. 2006.

[27] J. S. Vitter, "Random sampling with a reservoir," *ACM Transactions on Mathematical Software*, vol. 11, no. 1, pp. 37–57, Mar. 1985, https://doi.org/10.1145/3147.3165.

[28] Y. Tillé, "A general result for selecting balanced unequal probability samples from a stream," *Information Processing Letters*, vol. 152, Dec. 2019, Art. no. 105840, https://doi.org/10.1016/j.ipl.2019.105840.

[29] Z. Wen, D. L. Quoc, P. Bhatotia, R. Chen, and M. Lee, "ApproxIoT: Approximate Analytics for Edge Computing," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, Vienna, Jul. 2018, pp. 411–421, https://doi.org/10.1109/ICDCS.2018.00048.

[30] "Traffic Monitoring XML Data," *informo.madrid.es*. https://informo.madrid.es/informo/tmadrid/pm.xml.

[31] S. Mehrmolaei and M. R. Keyvanpour, "Time series forecasting using improved ARIMA," in *2016 Artificial Intelligence and Robotics (IRANOPEN)*, Qazvin, Iran, Apr. 2016, pp. 92–97, https://doi.org/10.1109/RIOS.2016.7529496.

[32] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, Aug. 2004, https://doi.org/10.1023/B:STCO.0000035301.49549.88.

[33] M. J. Sax, "Apache Kafka," in *Encyclopedia of Big Data Technologies*, S. Sakr and A. Zomaya, Eds. Springer International Publishing, 2018, pp. 1–8.

[34] A. de Myttenaere, B. Golden, B. Le Grand, and F. Rossi, "Mean Absolute Percentage Error for regression models," *Neurocomputing*, vol. 192, pp. 38–48, Jun. 2016, https://doi.org/10.1016/j.neucom.2015.12.114.

[35] A. Y. Sun, Z. Zhong, H. Jeong, and Q. Yang, "Building complex event processing capability for intelligent environmental monitoring," *Environmental Modelling & Software*, vol. 116, pp. 1–6, Jun. 2019, https://doi.org/10.1016/j.envsoft.2019.02.015.

[36] B. Khazael, H. T. Malazi, and S. Clarke, "Complex Event Processing in Smart City Monitoring Applications," *IEEE Access*, vol. 9, pp. 143150–143165, 2021, https://doi.org/10.1109/ACCESS.2021.3119975.

[37] T. Beckers, "An Introduction to Gaussian Process Models." arXiv, Feb. 10, 2021, https://doi.org/10.48550/arXiv.2102.05497.

[38] L. Bottou, "Stochastic Gradient Descent Tricks," in *Neural Networks: Tricks of the Trade*, vol. 7700, G. Montavon, G. B. Orr, and K. R. Müller, Eds. Springer Berlin Heidelberg, 2012, pp. 421–436.

[39] R. G. Brereton and G. R. Lloyd, "Support Vector Machines for classification and regression," *The Analyst*, vol. 135, no. 2, pp. 230–267, 2010, https://doi.org/10.1039/B918972F.

[40] K. Thurnhofer-Hemsi, E. López-Rubio, M. A. Molina-Cabello, and K. Najarian, "Radial basis function kernel optimization for Support Vector Machine classifiers." arXiv, Jul. 16, 2020, https://doi.org/10.48550/arXiv.2007.08233.