The Journal of Supercomputing (2023) 79:11078–11100
https://doi.org/10.1007/s11227-023-05100-x

A new Apache Spark-based framework for big data streaming forecasting in IoT networks

Antonio M. Fernández-Gómez[1] · David Gutiérrez-Avilés[2] · Alicia Troncoso[1] · Francisco Martínez-Álvarez[1]

Accepted: 1 February 2023 / Published online: 21 February 2023

Abstract

Analyzing time-dependent data acquired in a continuous flow is a major challenge for various fields, such as big data and machine learning. Being able to analyze a large volume of data from various sources, such as sensors, networks, and the internet, is essential for improving the efficiency of our society's production processes. Additionally, this vast amount of data is collected dynamically in a continuous stream. The goal of this research is to provide a comprehensive framework for forecasting big data streams from Internet of Things networks and serve as a guide for designing and deploying other third-party solutions. Hence, a new framework for time series forecasting in a big data streaming scenario, using data collected from Internet of Things networks, is presented. This framework comprises of five main modules: Internet of Things network design and deployment, big data streaming architecture, stream data modeling method, big data forecasting method, and a comprehensive real-world application scenario, consisting of a physical Internet of Things network feeding the big data streaming architecture, being the linear regression the algorithm used for illustrative purposes. Comparison with other frameworks reveals that this is the first framework that incorporates and integrates all the aforementioned modules.

✉ Francisco Martínez-Álvarez
fmaralv@upo.es

Antonio M. Fernández-Gómez
amfergom@alu.upo.es

David Gutiérrez-Avilés
dgutierrez3@us.es

Alicia Troncoso
atrolor@upo.es

[1] Data Science and Big Data Lab, Pablo de Olavide University of Seville, Ctra. de Utrera, km. 1, ES-41013 Seville, Seville, Spain

[2] Department of Computer Science, University of Seville, Avda. Reina Mercedes s/n, ES-41012 Seville, Spain

Keywords IoT · Big data · Streaming analysis · Time series

Notation

IoT	Internet of things
LoRa	Long range wide area
LoRaWAN	Long range wide area networks
NB-IoT	Narrowband internet of things
MOA	Massive online analysis
JSON	JavaScript object notation
MQTT	Message queuing telemetry transport
HDFS	Hadoop distributed file system
RDD	Resilient distributed dataset
MAPE	Mean absolute percentage error
GB	Gigabytes

1 Introduction

In recent years, computer science has made significant advancements in all aspects related to data [1], including data processing, transformation, and management. Machine learning methods, which group, classify, and forecast reality, have also seen rapid growth in the field of data analysis.

Time series forecasting in large data streaming is a critical task in various industries such as finance, weather forecasting, and transportation. The goal of this process is to predict future values of a time series, such as stock prices, temperature, or traffic flow, by utilizing a stream of incoming data. The challenge of this task lies in the high volume, velocity, and variety of data that is typically involved. For example, in finance, stock prices fluctuate rapidly and require real-time prediction to make informed decisions. In weather forecasting, the data coming from various weather stations need to be analyzed in real-time to make accurate predictions. In transportation, the data coming from various traffic sensors need to be analyzed in real-time to optimize traffic flow. This requires the use of efficient algorithms and architectures that can process and analyze data as it is received, without the need for data to be stored and processed offline. Additionally, it is also important to consider the temporal dependencies and patterns in the data, as well as the ability to handle missing or incomplete data.

One of the key challenges in time series forecasting in large data streaming is the ability to handle data in real-time. This requires the use of efficient algorithms and architectures that can process and analyze data as it is received, without the need for data to be stored and processed offline. Additionally, it is also important to consider the temporal dependencies and patterns in the data, as well as the ability to handle missing or incomplete data. It is also important to consider the scalability and computational efficiency of these models as the data volume increases.

In order to effectively perform time series forecasting in large data streaming, it is important to have a robust and scalable architecture in place. One approach

is to use a distributed computing framework such as Apache Storm or Apache Kafka to handle the incoming data stream. These frameworks allow for parallel processing of the data, which can greatly improve the efficiency and scalability of the forecasting process. Additionally, using a distributed database such as Apache Cassandra or MongoDB can also help with handling the high volume of data.

The big data streaming environment requires developing new data models and machine learning algorithms to handle the aforementioned scenario. Also, new system architectures must be set up to support these functionalities [2]. Apache Hadoop is a commonly used platform for building big data streaming architectures, offering features like the Hadoop Distributed File System for storing large amounts of data using multiple machines, and MapReduce [3] for developing applications in a distributed infrastructure. To manage data streams, Apache Spark framework [4] is a popular solution, providing a variety of tools for processing and analyzing big data streams, such as data manipulation, machine learning, and graph-parallel computation.

The rapid development of Internet of Things (IoT) technologies was a major catalyst for big data streaming and architecture [5]. As IoT architectures are being deployed in various contexts, the demand for real-time processing of sensor data is dramatically increasing [6]. Furthermore, real-time prediction of sensor values can aid in enhancing production methods and decision-making [7].

This work proposes a complete framework to forecast time series coming from IoT networks. These time series are analyzed in a big data streaming scenario where the data is presented in a continuous flow and a vast amount of it. Moreover, the response of the prediction model is in near-real-time mode, and it has an adaptive functioning to be suitable for continuous data flow analysis.

The development of the framework has been divided into five main modules, which are analyzed in this paper, to meet the previously mentioned requirements:

1. An IoT network architecture has been designed and deployed, utilizing compatible sensors to measure atmospheric pressure and temperature in a real-world environment.
2. Big data streaming architecture. A system for collecting and consuming data from IoT sensors has been developed.
3. Streaming-data modeling for time series forecasting. A methodology to model data streams as time series forecasting is presented.
4. Big data streaming forecasting method. A general workflow to forecast streams of data, responding in near-real-time.
5. System validation. A complete experimentation batch has been conducted to test the suitability of the framework in a real scenario using a linear regression algorithm based on the gradient descent method. The framework can incorporate other algorithms; however, linear regression was chosen for illustrative purposes due to its simplicity.

This framework integrates a solution for the generalized IoT domain, allowing for the analysis of all time series data sources, regardless of their origin (industrial,

smart home, healthcare, etc.). It also creates a method for modeling data streams for time series forecasting and a general workflow for consuming and feeding machine learning data. These goals were achieved after conducting a thorough state-of-the-art analysis (Sect. 2). Despite the presence of proposals related to individual modules or subsets, few solutions bring together all the aspects presented in this framework.

Furthermore, the framework was designed to provide a complete system for forecasting big data streams from IoT networks and serve as a guide for designing and deploying other solutions. It combines the time series forecasting modeling for streaming data (Module (3)) and the big data streaming forecasting method (Module (4)). As a result, other contributors can use the streaming data modeling to implement their own big data streaming forecasting system following the workflow in Module (4).

The remainder of the paper is structured as follows. Section 2 performs an analysis of previous and state-of-the-art research related to the modules. Section 3 provides a complete explanation of the framework. The experimentation environment and results are presented and discussed in Sect. 4. Finally, the conclusions and future work are outlined in Sect. 5.

2 Related work

This research includes several modules and technologies as discussed in Sect. 1. The state-of-the-art of this study is analyzed in parts. First, the technologies and solutions for IoT networks are evaluated in Sect. 2.1. Sections 2.2 and 2.3 examine the technologies related to big data streaming architecture and infrastructure, and a review of the literature on streaming analysis, respectively. Lastly, Sect. 2.4 compares solutions that integrate the previous modules, focusing on the advantages of the current proposal.

2.1 IoT networks

IoT networks play a key role in Industry 4.0 [8]. Three widely used IoT technologies for machine-to-machine communication are LoRa [9], Sigfox [10], and Narrowband IoT (NB-IoT) [11]; a comparison of these technologies can be found in [12]. This research chose LoRa due to its high availability, low cost, and open-source characteristic [13].

IoT technologies have various application fields, such as smart cities, automotive, precision agriculture, and healthcare. [14] reviews IoT and cloud computing in smart cities. [15] analyzes IoT technology in the automotive industry and its future trends. [16] studies the performance of IoT networks in precision agriculture using LoRa technology. [17] presents a study of IoT-based devices in healthcare and related machine learning methods.

Data transfer security [18] and user privacy are important in IoT and are currently open research areas [19]. Machine learning approaches are seen as a solution for both, such as in [20] for IoT access control [21] and for user privacy [22].

2.2 Big data streaming infrastructure and architecture

As mentioned in Sect. 1, Apache Hadoop [23] is a platform for deploying big data infrastructures, with applications developed and deployed using the MapReduce paradigm [3]. It serves as the base for Apache Spark [4], a big data environment that facilitates programming and deployment of big data applications. Spark offers a range of tools for processing and analyzing big data, including data manipulation, machine learning, graph computation, and streaming analysis.

Scheduling is important in data processing because it enables the efficient and effective use of resources by determining the order and timing of tasks. It also helps to ensure that data is processed in a timely manner and that there are no delays in the data processing pipeline. Thus, the work in [24] presents a technique for efficient scheduling of real-time tasks on multicore systems, saving energy and reducing temperature by 2.52% and 9.59 °C compared to existing methods. Analogously, RESET [25] improves resource utilization, reduces energy consumption and core temperature. TEFRED [26] saves energy and lowers peak temperatures while being fault-resistant. Finally, CETAS [27] is a strategy for efficient scheduling of periodic tasks for energy and temperature, improving success ratios, energy savings, and temperature reduction.

Three state-of-the-art technologies for data stream management and analysis are Apache Spark Streaming, Apache Storm and Apache Flink [28]. Spark Streaming processes data streams and provides a set of tools including online ML algorithms [29]. Storm is a real-time processing system [30], while Flink provides a native in-memory streaming processing framework [31]. Comparisons of the frameworks are in [32] (comparing Spark and Flink), [33] (benchmarking Spark and Storm), and [34] (comparing all three frameworks).

In [35], a Hadoop-Spark proposal was implemented with a scalability study showing feasibility and good performance for deploying big data apps in a real cluster. A comparison of the streaming platforms is in [36] highlighting performance, resource usage, and scalability.

2.3 Streaming analysis

Time series forecasting from data streams is a main aspect covered in this proposal. Regression algorithms play a crucial role in modeling the problem as a regression problem in machine learning. The regression model must adapt to new data in a streaming environment.

Linear regression is widely used to solve forecasting problems in multiple domains. For example, in [37], it is used to predict future sales of an online market, in [38], to forecast the electronic work function of metal elements, and in [39], to predict new cases of COVID-19. In [40], linear regression is compared to other

machine learning approaches, and in [41] linear regression is compared to support vector machine. A current literature review of linear regression and its applications can be found in [42].

With the big data scenario, the collection of regression algorithms has been extended. For example, a new nearest neighbors algorithm for big data is presented in [43]. Ensemble models like in [44] combine several models to solve regression problems. In [44], decision trees, gradient-boosted trees, and random forests are used to forecast big data time series. Data modeling is crucial in the regression task; the authors in [45] propose a data modeling approach for big data time series forecasting. In [46], the approach is applied to a deep neural network for power consumption forecasting. Marine technology in the form of side-scan sonar images is analyzed in [47].

Focusing on streaming analysis, the adaptation to new data and prompt response is essential factors. In [48], a comparative study of Apache Spark MLlib and MOA is presented. In [49], the MORStreaming algorithm predicts two or more values in a row and quickly adapts to changes.

In [50], a nearest neighbors-based forecasting method is proposed and tested with electricity demand data, yielding promising results. [51] presents a three-dimensional streaming algorithm for real-time anomaly detection in a sensor field and tests it with data from several sensors in Malaga (Spain) with good performance. The previous method was used for biological image screening in [52] to detect behavior patterns in cell structures over time.

2.4 Big data streaming frameworks

This research involves several modules and technologies, as introduced in Sect. 1. The state-of-the-art in this field is analyzed in this paper by combining all previously discussed features into a framework to address a general problem. In Sect. 2, the authors compare current proposals in the literature with the main features of the proposed framework.

In [53], the authors present SCDAP, a novel big data analytics framework for smart cities, with a focus on the big data streaming architecture. In [54], the authors present BiDeL, a complete, scalable, and adaptable big data framework for e-learning that covers the big data streaming architecture and system validation features. In [55], the authors present a scalable and distributed framework for real-time time series forecasting, focusing on streaming data modeling and forecasting and system validation.

Each of these proposals covers some aspects of the present research, but the authors aim to provide a comprehensive framework that integrates all the necessary components for a general solution. Also in this field, the authors in [56] present a framework for time series forecasting combining both historical and real-time data for predictions. The framework, called AESTSF, addresses adaptability and scalability challenges in time series forecasting. A distributed vector autoregression (VAR) algorithm was implemented using Apache Spark. The results of the experiments are evaluated and compared for different types of streaming time series data.

Healthcare is an area where research has been applied, and the authors in [57] implement a health monitoring framework to analyze and predict COVID-19. They conduct descriptive, diagnostic, predictive, and prescriptive analyses using big data analytics and validate the system by solving a classification problem to predict COVID-19 infection using a novel dataset with different pandemic symptom measures. This proposal covers the big data streaming architecture and system validation aspects of the present research.

To address the scalability issue of storing data generated by IoT networks, the authors in [58] propose an adaptive multi-attribute index framework for big IoT data. Based on four main attributes and their queries (spatial, temporal, keyword, and value), they present an adaptive method to determine the most efficient index to store data in a key-value system. The system is validated using simulated and real data from Taiwan's Open IoT platform, covering data modeling and providing a complete system validation. In [59], the author proposes a generic model for IoT Streaming Data Integration (ISDI) from multiple sources and presents several algorithms for ISDI including a generic window-based algorithm for processing IoT streaming data, a timing alignment algorithm for aligning timing conflicts, a de-duplication algorithm for resolving data redundancy, and experiments to demonstrate the practicality of the proposed ISDI approach. Finally, the authors in [60] present a framework for quick access and retrieval of IoT streaming data from multiple sources using indexing and optimizing data access. The framework includes a data compression approach for numerical data in IoT applications. The framework is evaluated through experiments with different indexing schemes for user query responses.

Confirming the previously discussed line of research based on frameworks, a systematic review of big data and IoT-based applications in smart environments can be found in [61]. The authors of [62] present a survey that focuses on the challenges and solutions for processing real-time big data streams. In conclusion, as observed in Table 1, this research covers a set of features that other proposals do not, making it a key contribution to the state-of-the-art.

Table 1 Comparative study of big data streaming frameworks

Framework	IoT network	Big data streaming	Streaming-data modeling	Forecasting	Validation
Proposal	✓	✓	✓	✓	✓
[53]	×	✓	×	×	×
[54]	×	✓	×	×	✓
[55]	×	×	✓	✓	✓
[57]	×	✓	×	×	✓
[58]	×	×	✓	×	✓
[56]	×	✓	×	✓	✓
[59]	×	×	✓	×	✓
[60]	×	×	✓	×	✓

3 Methodology for Apache Spark-based framework

This proposal, as mentioned in Sect. 1, aims to establish a comprehensive methodology for near-real-time forecasting of big data streams in an IoT environment and provide a complete framework for its implementation. The ultimate goal is to make near-real-time predictions of time series from data streams obtained from IoT sensors.

In Fig. 1, a global overview of the methodology is shown, which encompasses the first four main modules outlined in Sect. 1. The input of the system is an IoT network that collects data from sensors measuring various variables, which are streamed in a continuous flow and processed by the data stream modeling system. This module converts the data streams into inputs for a regression problem and makes them suitable for a machine learning algorithm to solve. The modeled data is then fed into the big data streaming architecture and used by the big data streaming forecasting method to train a regression model and generate predictions. The forecasting results are stored in the big data streaming architecture for access by authorized clients or third-party applications.

Therefore, a comprehensive description of this research proposal is outlined in this section. To begin with, the IoT network architecture is explained in detail at the hardware level (Sect. 3.1). Additionally, an examination of the big data streaming architectures applied in the proposal is presented (Sect. 3.3). Finally, the proposed modeling for time series forecasting based on streaming data (Sect. 3.2) and the big data streaming forecasting method (Sect. 3.4) are introduced.

The framework can be accessed at the following URL: https://github.com/DataLabUPO/IOT_framework.

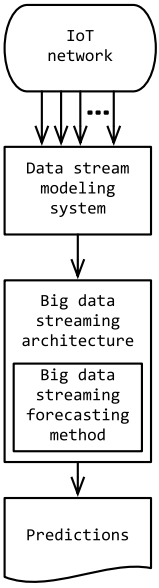

Fig. 1 Methodology workflow

3.1 IoT network architecture

The designed and deployed architecture is illustrated in Fig. 2. The infrastructure is based on LoRa technology and the LoRaWAN standard [63] that defines a communication protocol and system architecture for IoT networks. Its purpose is to collect the data that will later be processed and analyzed using the big data streaming forecasting method. LoRaWAN is an open standard and there are several open-source software solutions available for managing the network, in this case, the ChirpStack software [64].

Two LoRa-based devices have been developed to measure atmospheric pressure (in kilopascals, kPa) and temperature (in Celsius, °C). The data from these devices is transmitted through the LoRa network and managed by a LoRa gateway, which serves as the access point for the IoT network and transforms the binary packets from the devices into JSON files called payloads. These payloads are then transmitted to ChirpStack for management and control. The embedded MQTT broker in ChirpStack allows authorized applications to consume sensor data from the LoRa network as a data source.

3.2 Data stream modeling system for time series forecasting

This method aims to transform a data stream, or, in other words, a continuous flow of values for a particular variable (e.g., temperature, pressure, etc.), into a dataset, which can then be used to train, test, and validate a machine learning algorithm. To meet the requirements of this proposal, the machine learning algorithm must have three key characteristics: it must be compatible with regression problems, since the ultimate goal is time series forecasting; it must be capable of performing training and testing in an online, incrementally manner, meaning the regression model must be able to adapt as new data becomes available from the stream; and finally, its implementation must meet the interface requirements of Apache Spark for streaming algorithms

As shown in Fig. 3, the inputs of the process are the data flow coming from the IoT network, defined as streaming variables SV, formalized as a set of values V_i measured at the time point i, and received continuously. These streaming variables can come from various sources, such as sensors, cameras, or other devices that are connected to the IoT network. The data can be in the form of numerical values, images, or other types of data that are relevant to the problem at hand.

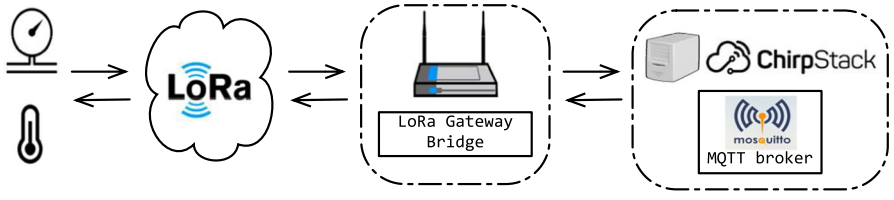

Fig. 2 IoT network architecture

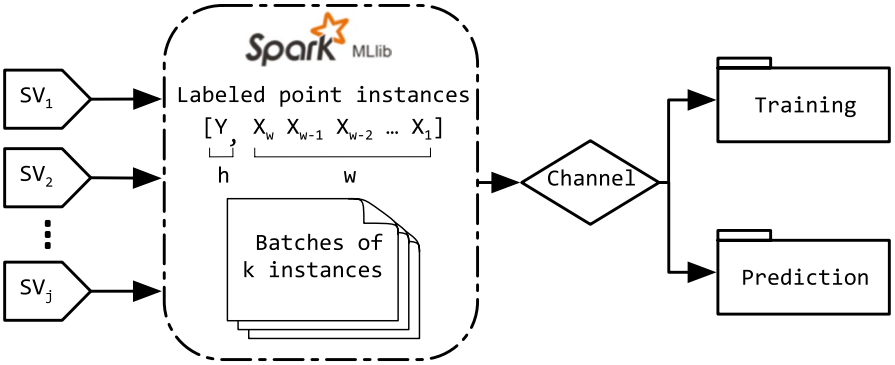

Fig. 3 Data stream modeling system

The number of arriving values N keeps increasing while the streaming channel is *ON* (e.g., a sensor is working), so it is considered that tending to infinity (Eq. (1b)). This means that the data stream is continuous and never-ending, making it essential to have a robust and scalable architecture in place to handle the high volume of data. Each time point i is a tuple defined by a value v, a unit u (seconds, minutes, or hours), and a frequency f (Eq (1c)). This allows for a clear definition and understanding of the data stream and how it is being captured. As an example, a temperature sensor SV_T streaming variable is considered. It generates a nearly infinite stream of values, acquired every 10 s ($u = s, f = 10$). Therefore, the value of SV_T at the time point 4 (40 s from the sensor is ON) is represented by V_4. This example illustrates how the data stream is defined and captured and how it can be used for time series forecasting.

$$SV = \{V_1, V_2, V_3, \ldots, V_i, \ldots V_N\} \tag{1a}$$

$$N \in \mathbb{N}, N \to \infty \tag{1b}$$

$$\langle i \in \mathbb{N}, u \in \{s, m, h\}, f \in \mathbb{N}\rangle \tag{1c}$$

The streaming variables SV are transformed into a lagged dataset comprehensible for machine learning algorithms [65]. The process depends on the learning window parameter, denoted by w, which represents the number of past values used to predict the following value (h, the prediction horizon). Instances are built using Apache Spark's Labeled Point format after $w + 1$ values have been received.

Each time k instances (Labeled Points) are generated, they are stored in a batch file and transferred to the big data streaming architecture. In production environments, k is set to 1, meaning that each instance is stored in a separate batch file. In development environments, k can be set to higher values to store larger batch files and test the other modules independently. The batches are stored in the big data streaming architecture in two channels (folders): training and prediction.

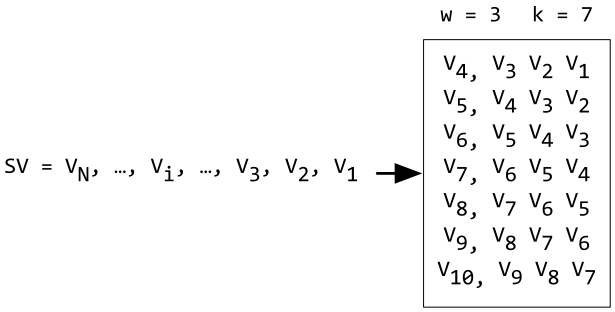

Fig. 4 Example of instances batch file building

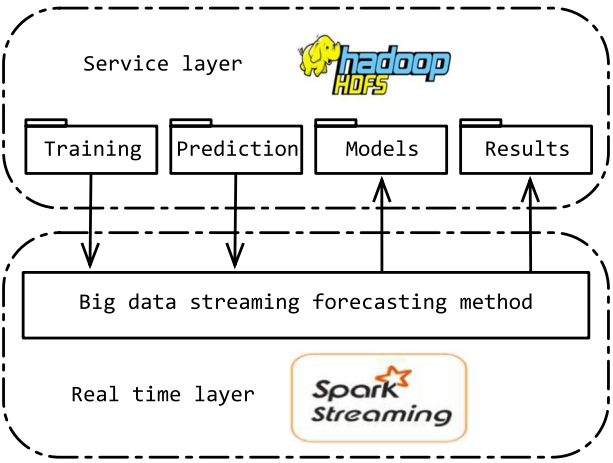

Fig. 5 Big data streaming architecture

An example of this method for $w = 3$ and $k = 7$ is shown in Fig. 4. For every four values, three correspond to the learning window (V_{N-1}, V_{N-2}, V_{N-3}) and V_N is the value to be predicted (h). This module accumulates values in a buffer, and constructs an instance after receiving w values. In the example, where $w = 3$, the module accumulates four values (V_4, V_3, V_2, or V_1) to construct an instance.

3.3 Big data streaming architecture

The training and prediction channels, referred to in Sect. 3.2, are part of the Kappa architecture [66] of the system. They are responsible for coordinating the data flow between the channels and the big data streaming forecasting method, making it scalable and fault-tolerant.

As can be seen in Fig. 5, the Service Layer is implemented by the Hadoop HDFS infrastructure. This layer acts as a foundation for the entire system, providing a robust and scalable platform to store and process the large amounts of data that are

received from the IoT network. The Hadoop HDFS infrastructure allows for distributed storage and processing of data, which is crucial for handling the high volume of data that is generated by the IoT network. Both channels feed the real-time layer composed of the implementation of the proposed big data streaming forecasting method based on Apache Spark Streaming technology. The real-time layer is where the data is processed and analyzed in near-real-time. The Apache Spark Streaming technology allows for efficient and effective processing of the data stream, enabling the system to quickly identify patterns and make predictions. Furthermore, the named Service Layer contains another two channels where the yielded regression models and prediction results are stored. These channels provide storage for the models and predictions that are generated by the system. This is important for future reference and for monitoring the performance of the models over time. This also allows for the data to be easily accessible and retrievable for further analysis and monitoring.

3.4 Big data streaming forecasting method

The proposed methodology aims to forecast streaming variables (SV) in near-real-time using the Apache Spark Streaming framework (as described in Sect. 3.2). The machine learning algorithm must have the capability to handle regression problems, employ auto-incremental learning, and be compatible with the Apache Spark Streaming framework [66].

For an auto-incremental regression algorithm, a first prediction model, M_t, is generated from a continuous stream of time-dependent data during the first training epoch. As new data arrive, the model is updated incrementally, resulting in models M_{t+1}, M_{t+2}, ..., M_{t+n}. To predict a future value at a specific moment, the most recent model is used. This process of generating prediction models for a data stream and making predictions is shown in Fig. 6. The M_t model is used to predict the P_q streaming variable that will arrive at time q. The prediction of the P_{q+1} streaming variable, which will arrive later but not necessarily consecutively, is made using the M_{t+1} model. This prediction cycle is repeated. Note that the model generation is time-dependent, while predictions are made asynchronously on demand, independent of time.

To train the regression algorithm and make real-time predictions over the streaming variable, the methodology in Fig. 7 is presented. It involves consuming two channels of batches of Labeled Points as described in Sect. 3.2:

1. The training channel is the input for the training process and where batches $B_t, \ldots, B_{t+i}$ are received. Each new batch generates a new model.
2. The Prediction channel receives batches $B_{q+j}, \ldots, B_q$ with instances to be predicted by the last model. Each new batch results in as many predictions as instances.

The streams are collected and managed by Apache Spark's DStream structure [67] in the Spark Streaming module, which is composed of a sequence of Spark's RDD structures [4]. The chosen data source is Apache Hadoop's HDFS [68]. The

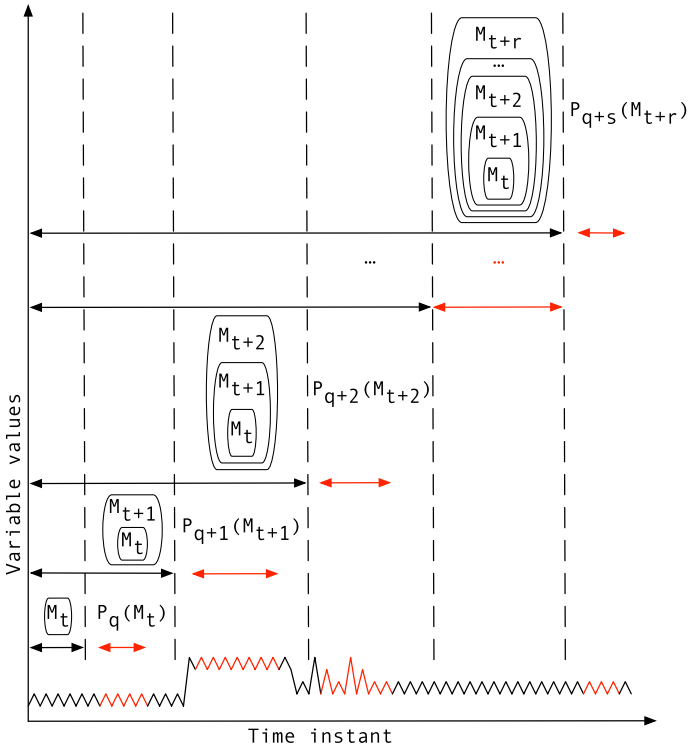

Fig. 6 Auto-incremental machine learning methodology

streaming context listener listens to the DStream and waits for new data. When new data arrive, the following tasks can be performed depending on the channel:

1. Training task. The current regression model M_t is updated using the new data. If it is the first model, it will be generated as an offline linear regression model. In other cases, the update depends on the regression algorithm. In this research, Spark Streaming's linear regression was used, updating the coefficients of M_t by applying the stochastic gradient descent method [69].
2. Prediction task. M_t is used to make predictions, considering historical data of length defined by the learning window w. The predictions are stored in HDFS for further analysis or consumption by third-party clients.

4 Case study

A comprehensive case study is presented to validate the effectiveness of the entire system in a real-world scenario. This case study includes three experimental tasks:

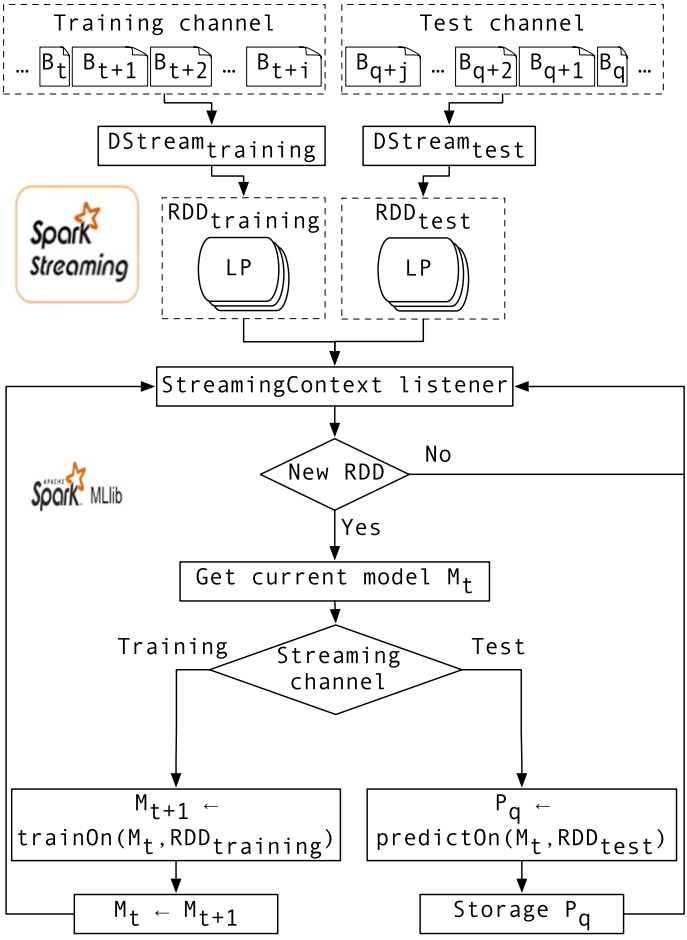

Fig. 7 Streaming analysis workflow

1. The collection of data from sensors connected to the IoT network was tested to verify its proper functioning.
2. Streaming variables were predicted to evaluate the performance of the data stream modeling system and the big data streaming forecasting method in a near-real-time environment.
3. Different batch sizes were simulated to assess the scalability of the big data streaming architecture.

The case study is organized as follows: the data collection task (Module (1)) is described in Sect. 4.1, followed by an analysis of the forecasting task (Module (2)) in Sect. 4.2, and finally, the scalability of the big data architecture (Module (3)) is presented in Sect. 4.3.

4.1 Data collection

Temperature and air pressure sensors were connected to the IoT network (as detailed in Sect. 3.1), collecting data from various areas of the Pablo de Olavide University of Seville campus. Data was collected every 10 s for one month. During this period, the sensors experienced two main types of malfunctions: blackouts, caused by power faults, and saturation, where the sensor continuously measures the same value and provides false readings. There were also internal software lock issues.

After a preprocessing task that involved checking the data and removing erroneous and repetitive measurements, approximately fifty thousand records (out of seventy thousand collected) remained. From these records, two sets of 5,726 records (one set for each variable type) were selected to perform streaming variables forecasting, resulting in the following two streaming variables:

1. SV_T. The temperature values, measured in Celsius degrees (°C), captured every 10 s.
2. SV_P. The air pressure values, measured in kilopascals (kPa), captured every 10 s.

4.2 Streaming variables forecasting

The next step after defining the streaming variables (as outlined in Sect. 4.1) is to create the experimental datasets using the data stream modeling system (as described in Sect. 3.2), and then perform a forecasting experiment using the big data streaming forecasting method (outlined in Sect. 3.4).

A collection of datasets has been created from the streaming variables SV_T and SV_P acquired from the IoT network (see Sect. 4.1) to simulate a near-real-time scenario and conduct a comprehensive experimental task. To achieve this goal, the data stream modeling system described in Sect. 3.2 was applied by following these steps:

1. Definition of ten prediction scenarios with ten values of the learning window, $w = \{3, 4, 6, 12, 24, 90, 180, 360, 720, 1080\}$. As previously mentioned, each w represents the number of values to be taken into account to predict the next one; that is, if $w = 3$, the historical values V_1, V_2, and V_3 will be used to predict the V_4 value. The window sizes were selected based on the authors' research experience and on the extensive studies performed in the literature. Additionally, ten w values were set to provide a representative sampling of experimental scenarios and identify possible overfitting situations.
2. Creation of labeled point instances file for each w.
3. Splitting each lagged stream file into thirty batches, B_i with $i = \{1, \ldots, 30\}$, for training composed of 150 instances and one for testing, B_{test}, with 226 instances.

The methodology in this study aims to test the performance of a big data streaming forecasting method in a near-real-time environment by predicting the

SV_T and SV_P streaming variables using previous datasets. The study computes the mean absolute percentage error (MAPE) by feeding the auto-incremental learning model with new learning data through the streaming channel.

For each streaming variable (SV_T and SV_P) and learning window ($w = 3, 4, 6, 12, 24, 90, 180, 360, 720, 1080$), the following steps are performed:

1. Activate the big data streaming forecasting method in near-real-time.
2. Inject the training datasets (B_i) into the training channel with a time-lapse of 5 s between them, resulting in 30 generated models (M_i).
3. Use each model (M_i) to make a prediction on the test dataset (B_{test}) and calculate the MAPE.

This results in a MAPE for each of the 30 generated models, and allows for analysis of how the MAPE changes as the learning algorithm is fed with more data. The observed values are compared with the predicted values, showing the best and worst predictions for each learning window (w).

The methodology uses a linear regression algorithm based on gradient descent in streaming [70]. The optimization of two parameters (α and σ) is necessary to generate the models. An ad hoc comprehensive search algorithm has been developed to find the optimal values of these parameters. The algorithm explores different values within specified intervals and steps. For example, the explored values of α are obtained using the interval [10, 15] and step $S = 1$.

15,000 different instances (10,000 for training and 5000 for testing) were chosen to perform the training and evaluate the models. The process was applied to temperature and pressure experiments with all w values. The optimal parameters are shown in Table 2.

The thirty models, tested with ten different learning windows, were generated for the streaming variables SV_P and SV_T. The results showed that the MAPE for the pressure streaming variable was 0.43%, while the maximum reached 100%, with an average of 0.82% and a standard deviation of 5.74%. The best learning window for this variable was $w = 24$, with an average MAPE of 0.44%. For the temperature streaming variable, the minimum MAPE was 6.93% and the maximum was 9.22%, with an average of 7.79% and a standard deviation of 7.79%. The best learning window for this variable was $w = 360$, with an average MAPE of 7%.

An incremental study of the big data streaming forecasting method was performed to analyze the evolution of the models in terms of the mean absolute percentage error (MAPE) as data arrives through the system. The results of this study are shown in Figs. 8 and 9 for the pressure streaming variable (SV_P) and temperature streaming variable (SV_T), respectively.

Table 2 Optimum parameters obtained for linear regression

Data	σ	α	MAPE
Pressure	10	$3.33E-11$	$1.27E-05$
Temperature	15	$3.61E-05$	$4.18E-05$

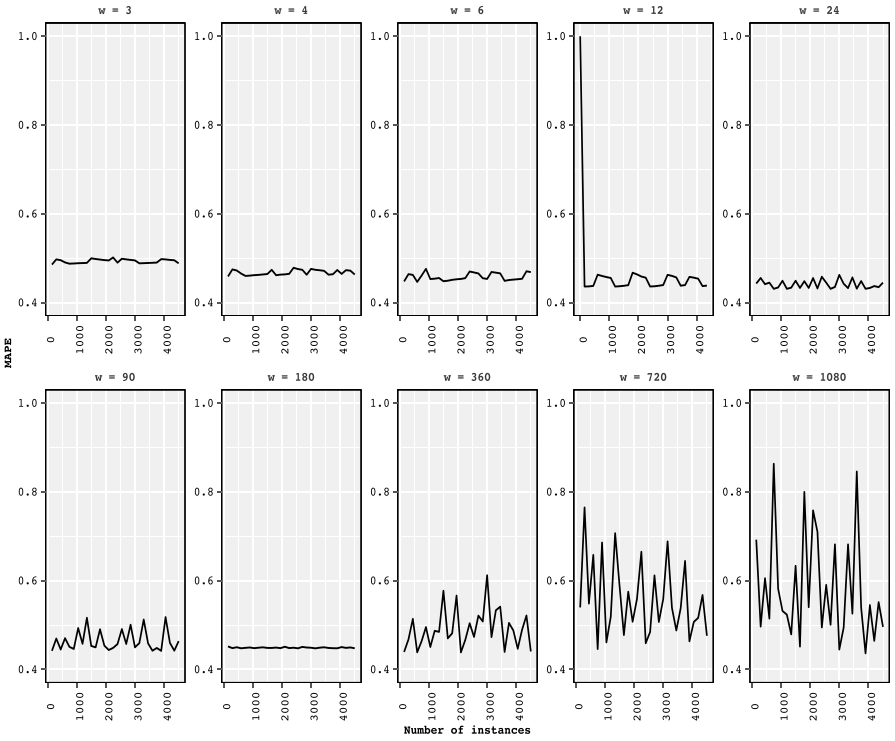

Fig. 8 Incremental study for SV_P streaming variable (pressure)

Each figure shows ten plots, one for each of the learning windows considered in the experimental setup. The *X*-axis represents the number of instances (labeled points) that have arrived at the system, while the *Y*-axis represents the MAPE obtained by the auto-incremental linear regression model.

In the incremental study of SV_P (Fig. 8), an unstable behavior of MAPE is predominant for most of the learning windows. Significant fluctuations are observed for learning windows in $\{90, 360, 720, 1080\}$, while smooth fluctuations are observed for learning windows in $\{3, 4, 6, 24\}$. A nearly constant value for MAPE is observed for learning window $w = 180$. Finally, for learning window $w = 12$, a combination of stabilization from the highest value and smooth fluctuation is observed.

In the incremental study of SV_T (Fig. 9), three behaviors of MAPE are observed. For learning windows in $\{3, 4, 6\}$, MAPE reaches its highest values proportionally with low values of the number of instances and is reduced until it stabilizes around a specific value. Nearly constant values of MAPE are observed for learning windows $w = 12$ and $w = 24$. Finally, an unstable behavior of MAPE with pronounced fluctuations (for learning windows in $\{720, 1080\}$) and smooth fluctuations (for learning windows in $\{90, 180, 360\}$) is observed.

These experimental results confirm that the big data streaming forecasting method produces better models in terms of MAPE as more data arrives from the streaming channel. The study also shows the influence of the learning window, w,

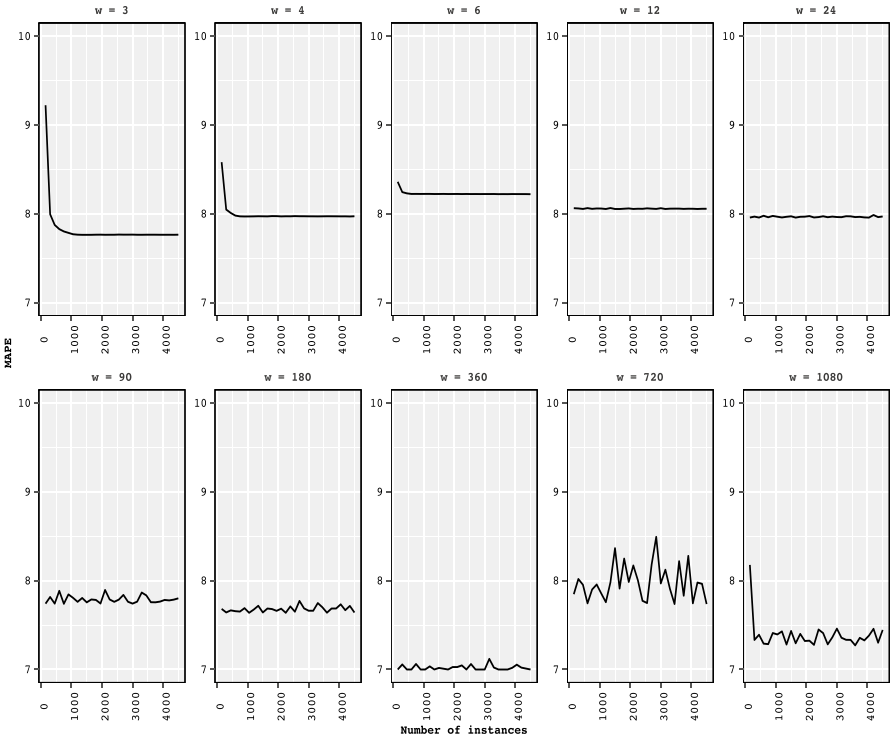

Fig. 9 Incremental study for SV_T variable (temperature)

on the stabilization of MAPE as new data arrives. Figures 8 and 9 demonstrate that for learning windows in $\{3, 4, 6, 24\}$, the values of MAPE are stable while new data arrives, whereas for learning windows in $\{90, 180, 360\}$, MAPE values fluctuate smoothly, and for learning windows in $\{720, 1080\}$, fluctuations are pronounced. The overfitting of the models when w increases is the reason for this effect. A high value of w implies a high number of coefficients for the linear equation that fits the data in the training process of a linear regression algorithm. As a result, the equation will be closely adapted to newly arrived data in the training stage, leading to predicted values that are close to the previous training data, resulting in a higher value of MAPE as the observed values have a different pattern than the training ones.

4.3 Big data scalability

The final part of the case study focuses on evaluating the scalability of the big data streaming architecture. Specifically, the efficiency of the system in terms of execution time is tested.

The big data streaming architecture was implemented on a Spark cluster of four machines, each having an Intel Core i7-5820K 3.3 GHz CPU with six cores and 12 execution threads, 15 MB of cache memory, and 50 GB of RAM. The machines

run Ubuntu 18.04 (64-bit version). The big data streaming forecasting method was applied to train a streaming linear regression algorithm for ten different learning windows ($w = \{3, 4, 6, 12, 24, 90, 180, 360, 720, 1080\}$) using batch sizes of 1 GB, 5 GB, 10 GB, 50 GB, 100 GB, 150 GB, and 200 GB. The algorithm was executed using 2, 4, 8, 24, and 48 cores, resulting in 200 experiments (ten learning windows per four batch sizes per five core configurations).

The results of this experimentation are displayed in Fig. 10. The figure presents ten plots, one for each w value, where the x-axis represents the batch size in GB, and the y-axis represents the execution time in seconds. Hence, a point (s, t) indicates that the big data streaming forecasting method took t seconds to process a batch of s GB.

The results obtained for all w values show similar patterns and depict a linear increase in execution time proportional to the batch size for each core configuration. Furthermore, the use of a greater number of cores leads to greater scalability, as evidenced by the lower slope of the linear equation.

In conclusion, this study highlights the competitiveness of the proposed framework in terms of execution time in big data environments. This is apparent in Fig. 10, where using a large number of cores in the cluster for each w value improves the efficiency of the system when processing large data sizes in big data environments.

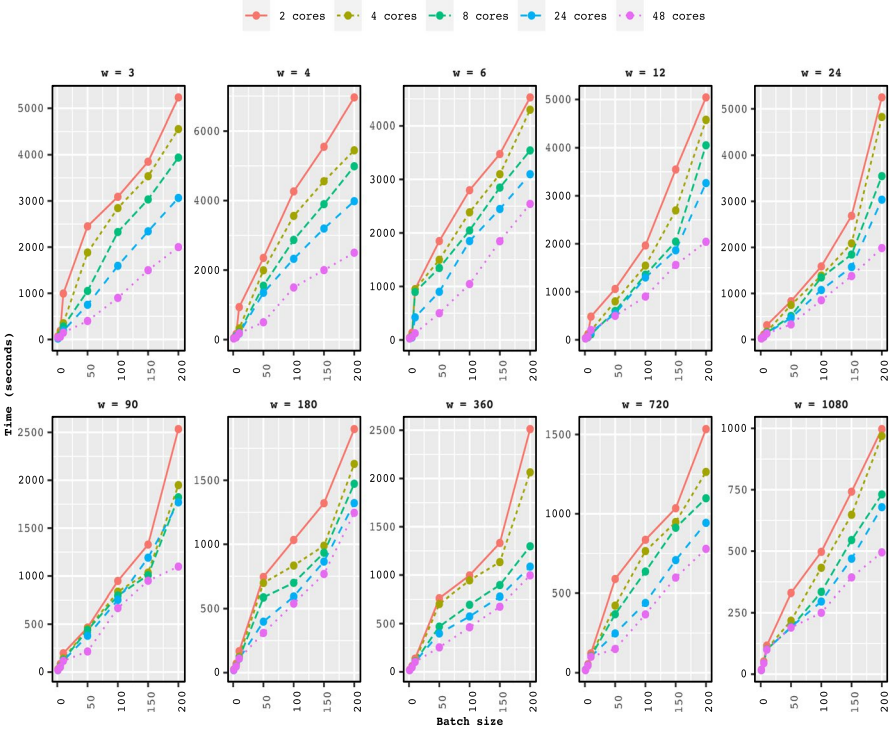

Fig. 10 Big data efficiency study

5 Conclusions and future work

This work has introduced a new Apache Spark-based framework for big data streaming forecasting in IoT networks. The framework comprises four main components: an IoT network architecture, a big data streaming architecture, a time series forecasting model for streaming data, and a big data streaming forecasting method. The entire proposed system was tested using a case study that involved conducting three experimental tasks: data collection, streaming variable forecasting, and big data scalability. The results demonstrate the effectiveness of the framework in collecting data from IoT sensors, modeling, and forecasting streaming variables in a near-real-time environment, and efficiently handling big data streams. The proposed system faces a significant challenge in terms of deployment. It requires the integration of multiple third-party software and a Hadoop cluster to utilize HDFS. Currently, the system only supports linear regression as the integrated stream algorithm. Moreover, the preprocessing of input data batches must be in LabelPoint format, as required by the algorithms implemented in Apache Spark. To address these limitations, future work will focus on developing scripts for the automated deployment of the entire framework. Additionally, the system will be enhanced by integrating more algorithms for streaming data flows, using the MLlib library provided by Apache Spark-streaming. This will increase the system's versatility and ability to handle a wider range of data processing tasks. Moreover, the architecture could be expanded to include new machine learning tasks such as classification and clustering. Other parameters than efficiency, like memory usage or energy consumption, will be considered to assess the quality of the framework. Finally, additional parameters should be considered for fine-tuning the algorithms in order to optimize their performance. The researchers are also developing an online adaptive subsystem for tuning the hyperparameters of the streaming machine learning algorithm to improve performance based on the incoming data.

Acknowledgements The authors would like to thank the Spanish Ministry of Science and Innovation and the Junta de Andalucía for their support within the projects PID2020-117954RB-C21 and TED2021-131311B-C22, PY20-00870 and UPO-138516, respectively.

Declarations

Conflict of interest The authors declare no conflict of interest regarding the publication of this paper.

References

1. Han J, Pei J, Kamber M (2011) Data mining: concepts and techniques. Elsevier, Amsterdam
2. Marz N, Warren J (2015) Big data: principles and best practices of scalable realtime data systems. Manning Publications Co., Greenwich, CT, USA
3. Dean J, Ghemawat S (2010) MapReduce: a flexible data processing tool. Commun ACM 53(1):72–77

4. Zaharia M, Xin RS, Wendell P, Das T, Armbrust M, Dave A, Meng X, Rosen J, Venkataraman S, Franklin MJ, Ghodsi A, Gonzalez J, Shenker S, Stoica I (2016) Apache spark: a unified engine for big data processing. Commun ACM 59(11):56–65
5. Omoniwa B, Hussain R, Javed MA, Bouk SH, Malik SA (2019) Fog/Edge computing-based IoT (FECIoT): architecture, applications, and research issues. IEEE Internet of Things J 6(3):4118–4149
6. Navani D, Jain S, Nehra MS (2017) The Internet of Things (IoT): A study of architectural elements. In: Proceedings of the International Conference on Signal-Image Technology Internet-Based Systems, pp. 473–478
7. Larrañaga P, Atienza D, Rozo JD, Ogbechie A, Puerto-Santana C, Bielza C (2018) Industrial applications of machine learning. CRC Press, United States of America
8. Kassab W, Darabkh KA (2020) A-Z survey of internet of things: architectures, protocols, applications, recent advances, future directions and recommendations. J Netw Comput Appl 163:102663
9. LoRaWAN standard. https://www.lora-alliance.org/ (2022)
10. Official SigFox website. https://www.sigfox.com (2022)
11. NB-IoT specification website. https://www.3gpp.org/DynaReport/WiVsSpec--700012.htm (2022)
12. Iqbal M, Abdullah AYM, Shabnam F (2020) An application based comparative study of LPWAN technologies for IoT environment. In: Proceedings of the IEEE Region 10 Symposium, pp. 1857–1860
13. Liya ML, Aswathy M (2020) LoRa technology for Internet of Things (IoT): a brief survey. In: Proceedings of the International Conference on IoT in Social, Mobile, Analytics and Cloud, pp. 8–13
14. Rani R, Kashyap V, Khurana M (2022) Role of IoT-cloud ecosystem in smart cities: review and challenges. Mater Today Proc 49(8):2994–2998
15. Rahim MA, Rahman MA, Rahman MM, Asyhari AT, Bhuiyan MZA, Ramasamy D (2021) Evolution of IoT-enabled connectivity and applications in automotive industry: a review. Vehicular Commun 27:100285
16. Miles B, Bourennane E-B, Boucherkha S, Chikhi S (2020) A study of LoRaWAN protocol performance for IoT applications in smart agriculture. Comput Commun 164:148–157
17. Farrokhi A, Farahbakhsh R, Rezazadeh J, Minerva R (2021) Application of Internet of Things and artificial intelligence for smart fitness: a survey. Comput Netw 189:107859
18. Sarma R, Kumar C, Barbhuiya FA (2022) MACFI: a multi-authority access control scheme with efficient ciphertext and secret key size for fog-enhanced IoT. J Syst Architect 123:102347
19. Sarma R, Barbhuiya FA (2021) MOFIT: An efficient access control scheme with attribute merging and outsourcing capability for fog-enhanced IoT. In: Proceedings of the Parallel and Distributed Computing, Applications and Technologies, pp. 523–535
20. Tahsien SM, Karimipour H, Spachos P (2020) Machine learning based solutions for security of Internet of Things (IoT): a survey. J Netw Comput Appl 161:102630
21. Sarma R, Kumar C, Barbhuiya FA (2020) ACS-FIT: A secure and efficient access control scheme for fog-enabled IoT. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, pp. 1–8
22. Sarma R, Kumar C, Barbhuiya FA (2022) Sustainable computing: informatics and systems PAC-FIT: an efficient privacy preserving access control scheme for fog-enabled IoT. Sustain Comput Inf Syst 30:100527
23. Shvachko K, Kuang H, Radia S, Chansler R (2010) The hadoop distributed file system. In: Proceedings of the IEEE Symposium on Mass Storage Systems and Technologies, pp. 1–10
24. Sharma Y, Chakraborty S, Moulik S (2022) ETA-HP: an energy and temperature-aware real-time scheduler for heterogeneous platforms. J Supercomput 78:1–25
25. Moulik S (2021) RESET: a real-time scheduler for energy and temperature aware heterogeneous multi-core systems author links open overlay panel. Integration 77:59–69
26. Sharma Y, Das Z, Moulik S (2022) TEFRED: A temperature and energy cognizant fault-tolerant real-time scheduler based on deadline partitioning for heterogeneous platforms. In: Proceedings of the International Conference on Parallel and Distributed Computing: Applications and Technologies, pp. 358–366
27. Sharma Y, Moulik S (2022) CETAS: a cluster based energy and temperature efficient real-time scheduler for heterogeneous platforms. In: Proceedings of the ACM/SIGAPP Symposium on Applied Computing, pp. 501–509
28. Jaiswal A, Dwivedi VK, Yadav OP (2020) Big data and its analyzing tools : a perspective. In: Proceedings of the International Conference on Advanced Computing and Communication Systems, pp. 560–565

29. Apache Spark Streaming. https://spark.apache.org/streaming/ (2022)
30. Apache Storm. https://storm.apache.org/ (2022)
31. Apache Flink. https://flink.apache.org/ (2022)
32. García-Gil D, Ramírez-Gallego S, García S, Herrera F (2017) A comparison on scalability for batch big data processing on Apache Spark and Apache Flink. Big Data Anal 2(1):1–11
33. Bang J, Choi M-J (2020) Docker environment based apache sand spark benchmark test. In: Proceedings of the Asia-Pacific Network Operations and Management Symposium, pp. 322–325
34. Chintapalli S, Dagit D, Evans B, Farivar R, Graves T, Holderbaugh M, Liu Z, Nusbaum K, Patil K, Peng BJ, Poulosky P (2016) Benchmarking streaming computation engines: storm, flink and spark streaming. In: Proceedings of the IEEE International Parallel and Distributed Processing Symposium Workshops, pp. 1789–1792
35. Fernández AM, Gutiérrez-Avilés D, Troncoso A, Martínez-Álvarez F (2020) Automated deployment of a spark cluster with machine learning algorithm integration. Big Data Res 19–20:100135
36. Isah H, Abughofa T, Mahfuz S, Ajerla D, Zulkernine F, Khan S (2019) A survey of distributed data stream processing frameworks. IEEE Access 7:154300–154316
37. Gopalakrishnan T, Choudhary R, Prasad S (2018) Prediction of sales value in online shopping using linear regression. In: Proceedings of the International Conference on Computing Communication and Automation, pp. 1–6
38. Li N, Zong T, Zhang Z (2021) Prediction of the electronic work function by regression algorithm in machine learning. In: Proceedings of the IEEE International Conference on Big Data Analytics, pp. 87–91
39. Rath S, Tripathy A, Tripathy AR (2020) Prediction of new active cases of coronavirus disease (COVID-19) pandemic using multiple linear regression model. Diabetes Metab Syndr Clin Res Rev 14(5):1467–1474
40. Ray S (2019) A quick review of machine learning algorithms. In: Proceedings of the International Conference on Machine Learning, Big Data, Cloud and Parallel Computing, pp. 35–39
41. Kavitha S, Varuna S, Ramya R (2016) A comparative analysis on linear regression and support vector regression. In: Proceedings of the International Conference on Green Engineering and Technologies, pp. 1–5
42. Maulud D, Abdulazeez AM (2020) A review on linear regression comprehensive in machine learning. J Appl Sci Technol Trends 1(4):140–147
43. Talavera-Llames R, Pérez-Chacón R, Troncoso A, Martínez-Álvarez F (2019) MV-kWNN: a novel multivariate and multi-output weighted nearest neighbours algorithm for big data time series forecasting. Neurocomputing 353:56–73
44. Galicia A, Talavera-Llames R, Troncoso A, Koprinska I, Martínez-Álvarez F (2019) Multi-step forecasting for big data time series based on ensemble learning. Knowl Based Syst 163:830–841
45. Torres JF, Galicia A, Troncoso A, Martínez-Álvarez F (2018) A scalable approach based on deep learning for big data time series forecasting. Integr Comput Aid Eng 25(4):335–348
46. Torres JF, Gutiérrez-Avilés D, Troncoso A, Martínez-Álvarez F (2019) Random hyper-parameter search-based deep neural network for power consumption forecasting. In: Proceedings of the International Work-Conference on Artificial Neural Networks, pp. 259–269
47. Połap D, Wawrzyniak N, Włodarczyk-Sielicka M (2022) Side-scan sonar analysis using ROI analysis and deep neural networks. IEEE Trans Geosci Remote Sens 60:4206108
48. Akgun B, Oguducu SG (2015) Streaming linear regression on spark MLlib and MOA. In: Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 1244–1247
49. Yu H, Lu J, Zhang G (2021) Morstreaming: a multioutput regression system for streaming data. IEEE Trans Syst Man Cybern Syst 52(8):4862–4874
50. Melgar-García L, Gutiérrez-Avilés D, Rubio-Escudero C, Troncoso A (2021) Nearest neighbors-based forecasting for electricity demand time series in streaming. In: Proceedings of the Conference of the Spanish Association for Artificial Intelligence, pp. 185–195
51. Melgar-García L, Gutiérrez-Avilés D, Rubio-Escudero C, Troncoso A (2021) Discovering three-dimensional patterns in real-time from data streams: an online triclustering approach. Inf Sci 558:174–193
52. Melgar-García L, Gutiérrez-Avilés D, Rubio-Escudero C, Troncoso A (2020) High-content screening images streaming analysis using the strigen methodology. In: Proceedings of the ACM Symposium on Applied Computing, pp. 537–539

53. Osman AMS (2019) A novel big data analytics framework for smart cities. Future Gener Comput Syst 91:620–633
54. Otoo-Arthur D, van Zyl TL (2020) A scalable heterogeneous big data framework for e-learning systems. In: Proceedings of the International Conference on Artificial Intelligence, Big data, Computing and Data Communication Systems, pp. 1–15
55. Ferreira D, Senna C, Sargento S (2020) Distributed real-time forecasting framework for IoT network and service management. In: Proceedings of the IEEE/IFIP Network Operations and Management Symposium, pp. 1–4
56. Pandya A, Odunsi O, Liu C, Cuzzocrea A, Wang J (2020) Adaptive and efficient streaming time series forecasting with lambda architecture and spark. In: Proceedings of the IEEE International Conference on Big Data, pp. 5182–5190
57. Ahmed I, Ahmad M, Jeon G, Piccialli F (2021) A framework for pandemic prediction using big data analytics. Big Data Res 25:100190
58. Huang C-Y, Chang Y-J (2021) An adaptively multi-attribute index framework for big IoT data. Comput Geosci 155:104841
59. Tu DQ, Kayes ASM, Rahayu W, Nguyen K (2020) ISDI: a new window-based framework for integrating IoT streaming data from multiple sources. In: Proceedings of the Advanced Information Networking and Applications, pp. 498–511
60. Doan Q-T, Kayes ASM, Rahayu W, Nguyen K (2022) A framework for iot streaming data indexing and query optimization. IEEE Sens J 22(14):14436–14447
61. Hajjaji Y, Boulila W, Farah IR, Romdhani I, Hussain A (2021) Big data and IoT-based applications in smart environments: a systematic review. Comput Sci Rev 39:100318
62. Mehmood E, Anees T (2020) Challenges and solutions for processing real-time big data stream: a systematic literature review. IEEE Access 8:119123–119143
63. Rizzi M, Ferrari P, Flammini A, Sisinni E (2017) Evaluation of the IoT LoRaWAN solution for distributed measurement applications. IEEE Trans Instrum Meas 66(12):3340–3349
64. ChirpStack. https://www.chirpstack.io (2022)
65. Hyndman RJ, Athanasopoulos G (2018) Forecasting: principles and practice. OTexts, Australia
66. Sanla A, Numnonda T (2019) A comparative performance of real-time big data analytic architectures. In: Proceedings of the IEEE International Conference on Electronics Information and Emergency Communication, pp. 1–5
67. Zaharia M, Das T, Li H, Shenker S, Stoica I (2012) Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters. In: Proceedings of the USENIX Conference on Hot Topics in Cloud Computing, 10–11
68. Apache Hadoop. http://hadoop.apache.org/ (2022)
69. Akgün B, Oguducu SG (2015) Streaming linear regression on Spark MLlib and MOA. In: Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 1244–1247
70. Hu T, Wu Q, Zhou DX (2016) Convergence of gradient descent for minimum error entropy principle in linear regression. IEEE Trans Signal Process 64(24):6571–6579