Efficient Processing of Large-Scale Medical Data in IoT: A Hybrid Hadoop-Spark Approach for Health Status Prediction

Yu Lina[1]*, Su Wenlong[2]
Hebei College of Industry and Technology, Hebei Shijiazhuang, 050091, China[1]
Liaoning University, Liaoning Shenyang, 110036, China[2]

***Abstract*—In the realm of Internet of Things (IoT)-driven healthcare, diverse technologies, including wearable medical devices, mobile applications, and cloud-based health systems, generate substantial data streams, posing challenges in real-time operations, especially during emergencies. This study recommends a hybrid architecture utilizing Hadoop for real-time processing of extensive medical data within the IoT framework. By employing distributed machine learning models, the system analyzes health-related data streams ingested into Spark streams via Kafka threads, aiming to transform conventional machine learning methodologies within Spark's real-time processing, crafting scalable and efficient distributed approaches for predicting health statuses related to diabetes and heart disease while navigating the landscape of big data. Furthermore, the system provides real-time health status forecasts based on a multitude of input features, disseminates alert messages to caregivers, and stores this valuable information within a distributed database, which is instrumental in health data analysis and the production of flow reports. We compute a range of evaluation parameters to evaluate the proposed methods' efficacy. This assessment phase encompasses measuring the performance of the Spark-based machine learning algorithm in a distributed parallel computing environment.**

Keywords—Internet of Things; big data; hadoop; spark-based machine learning

I. Introduction

Over the past two decades, our epoch has come to be recognized as the era of big data, wherein digital data has assumed a pivotal role across various domains, encompassing society, research endeavors, and, particularly, the medical domain [1]. Big data denotes the characterization of copious data amassed from diverse sources, such as sensor networks, high-throughput apparatus, mobile applications, streaming devices, and data reservoirs spanning numerous industries, with a pronounced emphasis on the healthcare sector [2, 3]. Effectively managing, processing, presenting, and deriving insights from this diverse and voluminous data spectrum has posed substantial challenges using the extant technological toolset [4]. Efficiently deriving meaningful insights from this multitude of data, tailored to various user profiles, ranks among the paramount technological quandaries facing the domain of big data analytics [5, 6]. Presently, numerous data sources within healthcare, both clinical and non-clinical, are converging, with the digital medical history of patients being of paramount importance in healthcare analytics [7]. Consequently, three primary challenges surface in creating a distributed data system designed to handle extensive data volumes [8].

The initial challenge stems from the complexity of collecting data from disparate sources due to its heterogeneous and vast nature. Second, the fundamental predicament revolves around storage, as big data systems must effectively store data while maintaining optimal performance. The final challenge pertains to big data analytics, especially real-time or near-real-time analysis of vast datasets, incorporating forecasting, optimization, visualization, and modeling [9]. In light of the shortcomings of current data management systems in addressing real-time and heterogeneous data, a need emerges for a new processing paradigm [10]. Conventional relational database management systems, exemplified by MySQL, predominantly cater to structured data management, with limited support for unstructured or partially structured data. Furthermore, traditional RDBMS scaling strategies for parallel hardware management and fault tolerance often prove inadequate as data volumes expand [11].

To tackle these challenges, the research community has introduced a variety of projects to address large-scale and diverse data storage, including NoSQL database management systems suitable for scenarios where a relational model is not requisite. MapReduce, an amalgamation of Map and Reduce operations, serves as a parallel processing technique for handling vast distributed datasets in commodity clusters [12]. Yet, it is marred by its sluggishness when dealing with iterative algorithms. The Hadoop framework, a batch processing system, is employed for distributed data processing and storage, relying on the MapReduce model for programming [13]. The Hadoop Distributed File System (HDFS) offers a distributed storage solution that is highly resilient [14]. However, Hadoop is ill-suited for in-memory computing and real-time stream processing and does not uniformly apply the MapReduce paradigm to all challenges. The volume of processed data is a determinant of the speed of results. Conversely, stream computing prioritizes data velocity and involves continuous input and output. Big data streaming computing (BDSC) comprises real-time computing, distributed messaging, high throughput, and minimized processing latency. It is essential for extracting meaningful information from vast datasets, particularly in the healthcare realm.

The swift advancement of large data analytics holds significant implications for advancing medical practices and academic research. Data collection, management, analysis, and assimilation tools designed to handle heterogeneous, unstructured, and structured data within contemporary healthcare systems have become accessible. BDSC is now integral to the landscape of big data analytics, facilitating the rapid exploration of the latent value of extensive healthcare data. Nevertheless, challenges persist due to the diverse data sources within the healthcare sector, necessitating the integration of data originating from relational databases, Hadoop, search engines, and other analytical systems. The application of machine learning to such extensive and high-velocity data streams presents considerable challenges, as conventional machine learning algorithms are not well-suited for such massive data volumes and variable velocities. Furthermore, efficient analytical data processing is a pressing concern, necessitating effective data integration. While contemporary research predominantly relies on machine learning, real-time machine learning applications are absent for streaming big data. Moreover, most healthcare analytics solutions predominantly focus on Hadoop, a batch-oriented computational platform.

The growing elderly population and the rising prevalence of chronic illnesses have exacerbated the inadequacies of conventional healthcare practices. In tandem, medical IoT has increased, enabling continuous monitoring and real-time emergency interventions, especially in cardiac conditions. This proliferation has led to the generation of vast datasets by millions of sensors, challenging the capacity to process and respond to this data under critical conditions. To address these challenges, we have developed a healthcare framework exemplified by a real-time health status forecasting case study. NoSQL Cassandra, Spark streaming, Spark MLlib, Kafka data streaming, and Apache Zeppelin technologies underpin this system. Kafka's producers generate multiple message streams, which are filtered using Spark streaming, enriched through machine learning, and stored in NoSQL repositories, facilitating analytics and visualization. This endeavor has substantially improved the quality of patient monitoring within healthcare.

The remaining portion of the paper is organized in the following fashion. Section II provides a comprehensive analysis of previous research in the field, which serves as a foundation for our suggested approach. Section III provides a detailed explanation of the hybrid architecture, with a focus on the incorporation of Hadoop, Spark, and distributed machine learning models. Section IV provides detailed explanations of the specific scenarios or use cases relevant to our proposed architecture. Subsequently, Section V provides the discussion of comprehensive examination, evaluating the architecture's advantages, constraints, possible uses, and comparative observations. Section VI explores the collected data, demonstrating the effects of adopting our architecture for predicting health status. Section VII ultimately ends by providing a concise overview of significant discoveries and proposing potential avenues for further study.

II. RELATED WORKS

A. *Medical Big Data Challenges*

The concept of the 5Vs in big data, comprising Volume, Variety, Velocity, Veracity, and Value, aptly elucidates the sheer magnitude of data generated within the contemporary healthcare sector. The healthcare domain is burdened by a substantial and ever-expanding volume of data that necessitates comprehensive collection and analysis [15]. The notion of variety underscores the diverse range of data sources that must be tapped into within healthcare. Pertinently, healthcare data and the domain's knowledge demand real-time acquisition, encapsulated by the concept of velocity. The integrity and trustworthiness of healthcare data are encapsulated in the dimension of veracity. Ultimately, valuable insights can be gleaned through meticulous examination of the colossal healthcare dataset. Distributed sources of healthcare data encompass medical electronic records, health claims, diagnosis data, clinical imagery, streaming systems, and sensors affixed to patients' bedsides for continuous vital sign monitoring. These sources collectively generate vast amounts of data, surpassing the processing capabilities of conventional data handling systems. The myriad challenges associated with big data are illustrated in Fig. 1. In this research, our focus has been dedicated to the initial five pivotal challenges within big data, encompassing data integration, storage, analysis, and representation.

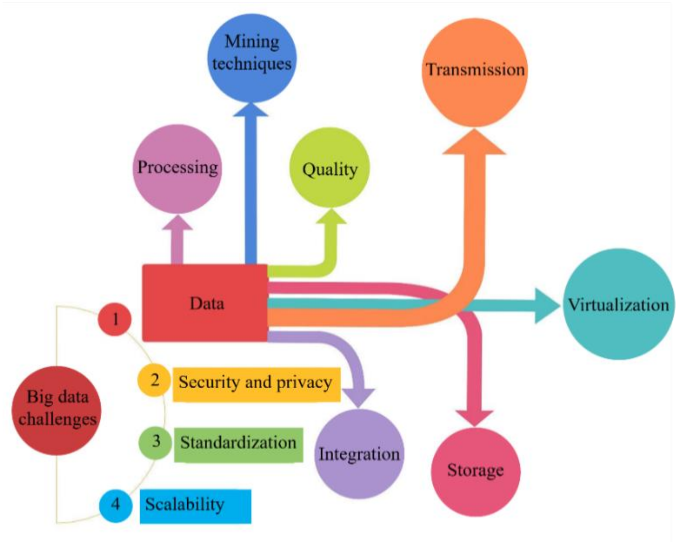

Fig. 1. Big data challenges.

B. *Literature Study*

The amalgamation of Machine Learning (ML), Deep Learning (DL), Neural Networks (NN), Fuzzy Logic Systems (FLSs), Wireless Sensor Networks (WSNs), and Temporal Graphs (TGs) holds pivotal significance in the processing of large-scale medical data within the IoT landscape. ML and DL techniques empower healthcare systems to discern intricate patterns within vast datasets, enabling predictive analytics for disease diagnosis, treatment planning, and health status forecasting [16-18]. NN, a subset of ML, simulates the human brain's learning process, aiding in complex data analysis, especially in image recognition and signal processing tasks within medical imaging and diagnostics [19, 20]. FLSs supplements decision-making processes by handling uncertain or imprecise data, crucial in medical scenarios where data might exhibit variability [21]. WSNs, integrated with IoT devices, facilitate real-time health monitoring, efficiently collecting and transmitting patient data for timely analysis [22]. Meanwhile, TGs provide an intricate understanding of dynamic patient interactions over time, aiding in disease progression modeling and personalized treatment plans [23]. The convergence of these technologies optimizes medical data processing, fostering precision medicine, remote patient monitoring, and efficient healthcare delivery, thereby revolutionizing patient care and augmenting medical research endeavors within the IoT-driven healthcare domain [24].

The exponential proliferation of healthcare data, coupled with its profound insights, has positioned big data analytics, particularly within the healthcare domain, as a formidable challenge spanning multiple academic disciplines, including data mining and machine learning. The advancement of data collection in healthcare can be primarily attributed to strides in scientific and technological innovation. The healthcare sector primarily leverages three fundamental categories of digital data for data collection: health research, operational processes within healthcare organizations, and clinical records. Traditional data mining techniques, which involve identifying valuable patterns within vast databases, struggle to unearth insights from the dispersed, extensive, and diverse datasets prevalent in healthcare. Data mining techniques are pivotal in transforming this data into actionable information. Numerous studies in the medical field focus on prediction and recommendation systems. These studies include experiments on heart attack prediction and a comparative assessment of different approaches. Breast carcinoma classification employs a genetically tuned neural network model. Other research endeavors encompass information retrieval and data mining methodologies.

Healthcare analytics encompass various applications, such as epidemic forecasting, health decision support, and recommendation systems, geared toward enhancing care quality, reducing costs, and augmenting productivity. One notable approach is utilizing a K-means clustering algorithm operating in the cloud as a MapReduce task, utilizing healthcare data for clustering. An alternative proposal suggests a decentralized platform for managing electronic health records personally, employing Hadoop and HBase. Predictive analysis in healthcare involves forecasting diabetes and determining the most suitable therapy using algorithms and the Hadoop MapReduce environment. Big data contextual exchange among healthcare systems through the Internet of Things (IoT) is demonstrated through an intelligent care system built on Hadoop. This system leverages an architecture with advanced data processing capabilities to collect data from diverse linked devices and transmit it to intelligent buildings. Real-time analysis of electronically generated medical records and data from medical equipment and mobile applications is described. This system, incorporating Hadoop, MongoDB, and an innovative treatment method, aims to enhance patient information processing outcomes. The predominant focus in most healthcare analytics solutions lies in Hadoop, which can handle substantial data volumes from diverse sources in batch-oriented processing. However, Hadoop's real-time processing capabilities are limited, and Spark emerges as a swifter and more efficient alternative, particularly for iterative machine learning tasks. Both Hadoop and Spark, being Apache projects, are integral to the big data landscape, with Spark generating significant interest.

Several scalable machine learning algorithms aim to address the diverse challenges within big data analytics. These algorithms include a scalable Random Forest classification model for diabetes risk prediction, logistic regression for phishing URL detection, and a Markov chain-based system for identifying abnormal patterns in the behaviors of elderly individuals. Real-time management of medical emergencies using IoT-based medical sensors is presented, along with a paradigm for real-time analysis of extensive medical data using Spark Streaming and Apache Kafka. A real-time health forecasting system focusing on machine learning, particularly Decision Trees, is developed to process data streams obtained via socket streams. A novel strategy for cardiac disease monitoring, centered on real-time decentralized machine learning within the Spark environment, is proposed in one study. Most of these studies either center on specific healthcare data sources or predominantly deal with batch-oriented computation. Healthcare generates a myriad of rapidly accumulating data from diverse sources. Moreover, some studies prioritize data storage and visualization, while others emphasize powerful data analytics tools like data mining and machine learning. Thus, the creation of an effective system for managing remote health data streams necessitates real-time healthcare analysis, which encompasses data collection, real-time processing, and robust machine learning capabilities.

The two leading causes of global mortality in recent times have been heart disease and diabetes. Continuous monitoring and early detection of these ailments can significantly reduce mortality rates. The availability of wearable health monitors, the adoption of IoT medical technology within healthcare systems, and the surge in patient conditions further underscore the potential of big data technologies for real-time health condition prediction. Real-time prediction can streamline healthcare visits and empower patients and healthcare providers to anticipate potential illnesses. Furthermore, the proposed system includes an alert mechanism, ensuring that emergency services are promptly notified when a patient's condition deviates from the norm, facilitating rapid interventions during emergencies.

III. Proposed Architecture

Within the scope of this research, a system for data processing and monitoring is introduced, amalgamating Kafka and Spark streams. This system operates by first processing data received from connected devices and subsequently storing this data for real-time analysis. The architectural layout of this proposed system is elucidated in Fig. 2. The system commences with the continuous generation of data messages from Kafka generators. These data messages encompass diverse disease names and are subsequently conveyed to a Spark streaming application for immediate processing. Spark Stream harnesses machine learning models to analyze various health attributes acquired from the Kafka Stream, thereby predicting health status. The results of this analysis are stored in a NoSQL Cassandra database. In the proposed architecture, Apache Zeppelin is instrumental in retrieving data from the database and presenting it in a real-time dashboard featuring data visualization in the form of graphs, charts, and data tables. By leveraging this real-time data in an Internet of Things (IoT) context, it becomes feasible to promptly scrutinize it, enabling the timely dispatch of alert notifications to caregivers when significant changes in a patient's condition arise. This real-time monitoring capability facilitates immediate action and intervention when necessary, ensuring patients receive timely and responsive care.

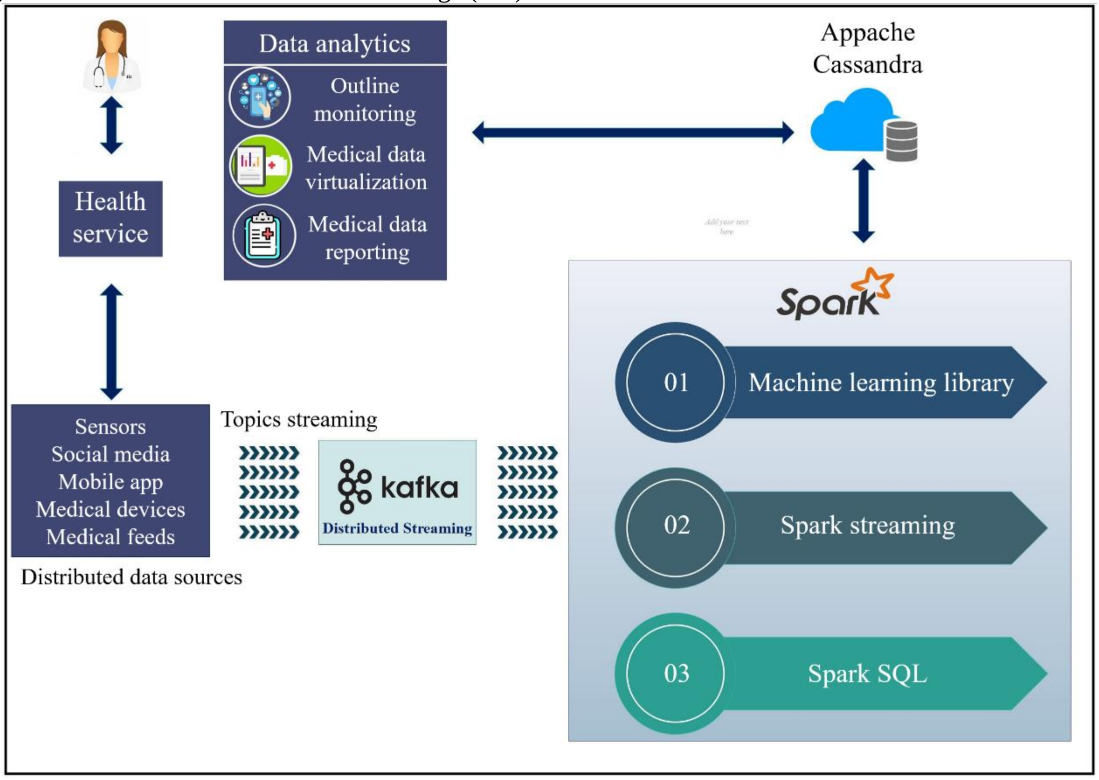

Fig. 2. Proposed architecture.

The IoT encompasses a network of physical and virtual entities equipped with electronics, intelligent wearables, software, applications, sensors, and network connectivity, all designed to collect and exchange data among themselves and with data center systems. The data generated by ubiquitous wearable health monitors, commonly found in households, is characterized by its substantial volume and random nature. Stimulating user activity trends or gathering essential data necessitates analysis through a robust big data analytics system. Forecasts indicate that, by 2020, IoT-related technologies within the healthcare sector will constitute a significant portion, accounting for forty percent of all IoT-related technologies. The integration of information technology in healthcare, particularly health informatics, is poised to bring about a paradigm shift, significantly reducing inefficiencies, containing costs, and, ultimately, saving lives. Real-time monitoring facilitated by the IoT can be a lifesaver in medical emergencies, encompassing conditions such as diabetes, heart disease, and various chronic disorders. Numerous sources are presently accessible for the continuous monitoring of health indicators. The workflow of the proposed system, involving multiple data sources, is outlined in Fig. 3. This system aims to harness the power of IoT to provide real-time monitoring and timely interventions in healthcare, thereby enhancing the quality of care and potentially saving lives in critical situations.

The escalating volume of data generated within healthcare systems has surpassed the capabilities of Spark alone for data management. In response to this challenge, Kafka, designed explicitly for managing streaming data, has been seamlessly integrated into our system. The data collection component in the proposed system architecture plays a pivotal role in gathering health-related data from various sources and multiple medical conditions, employing a range of devices coupled with telemedicine and telehealth services. This data collection group continually gathers, organizes, and manages clinical data related to patients. It facilitates categorizing streaming data according to the relevant domain (e.g., specific medical conditions), where records are subsequently published. Apache Kafka, operating as publishes-subscribe messaging system designed for distributed streaming, is a central component in this data management strategy. It is built to be a replicated, distributed, and partitioned service. Health monitoring devices feed real-time data into Kafka via Kafka producers. The fundamental concept that Kafka introduces for a stream of records is termed a "topic." Kafka servers use these topics to store incoming messages from publishers for a defined period before releasing them to the relevant data stream. Each topic is subdivided into multiple partitions, each capable of storing data in diverse formats.

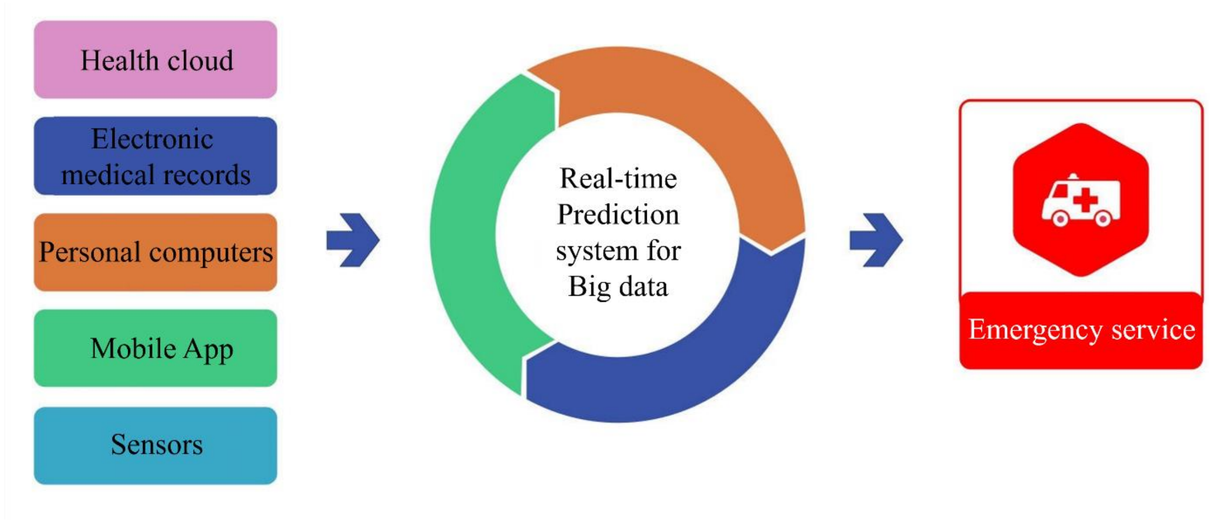
Health cloud
Electronic medical records
Personal computers
Mobile App
Sensors
Real-time Prediction system for Big data
Emergency service

Fig. 3. The workflow of the proposed architecture.

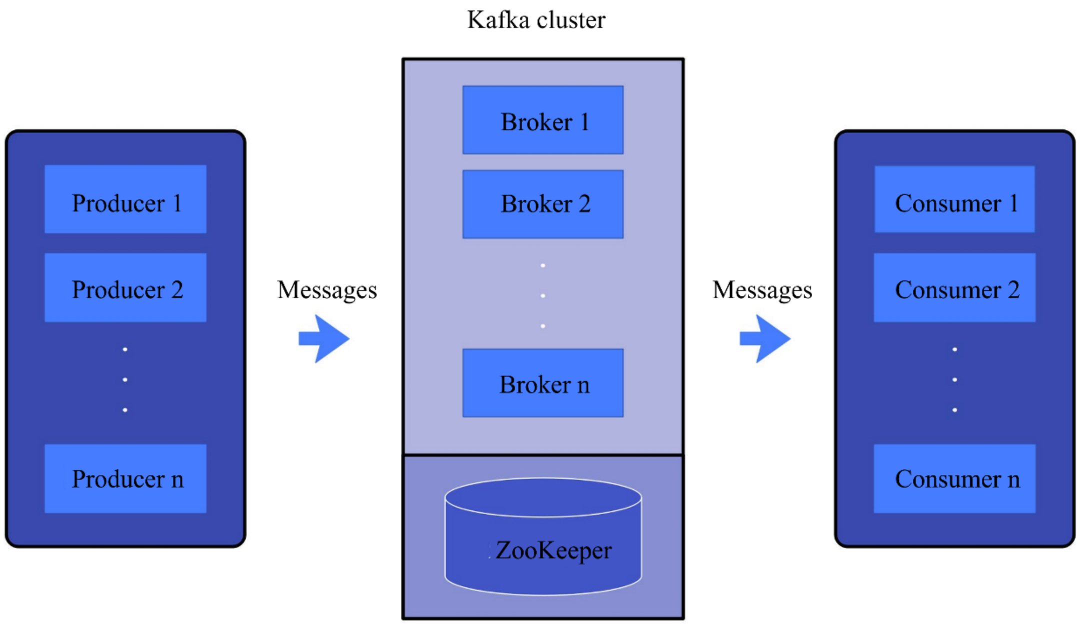

Fig. 4. The Kafka communications system.

Consumers of Kafka access information as it becomes available by subscribing to one or more topics. The Kafka communication infrastructure is depicted in Fig. 4. To ensure the efficient operation of Kafka, ZooKeeper, a centralized service, plays a vital role by providing group services, distributed synchronization, configuration information maintenance, and naming services. Distributed applications use these services extensively, although implementing and maintaining them comes with inherent challenges, such as dealing with recurring defects and race conditions. Typically, applications initially underinvest in these services due to the complexity of their implementation, rendering them fragile in the face of change and difficult to manage. Consequently, resolving these issues and enhancing the robustness of distributed applications is an ongoing endeavor within distributed systems.

This case study involves two data producer programs that simulate connected devices, utilizing Apache Kafka to generate data events. Apache Spark, an open-source, high-speed distributed processing engine, plays a central role in this system. Spark's most notable feature is its capability for in-memory calculations, significantly enhancing its processing speed. Furthermore, Spark offers user-friendly features, an advanced framework for large-scale analysis, and the ability to execute disk-based computing when dealing with datasets that exceed available memory. A key concept employed by Spark is Resilient Distributed Datasets (RDDs), which are distributed, immutable collections of items. To achieve parallelization, Spark internally spreads the RDD data across multiple nodes within the cluster. RDDs can store input and intermediate data in memory, reducing the cost of input-output operations associated with reading from or writing to system files. This feature enables efficient data reuse, which is particularly beneficial for iterative machine learning algorithms. Once data is transformed into an RDD, two fundamental types of operations can be performed:

- Transformations: These operations involve applying mapping, filtering, and more to existing RDDs to generate new RDDs.
- Actions: These operations compute a result using an RDD, which is then returned or saved to an external storage system.

Spark also includes an ML library, MLlib, which encompasses popular machine learning techniques such as classification, regression, clustering, and more. To handle real-time data from sources like Kafka and Twitter, Spark streaming builds upon the Spark API. The batch-processing Spark engine divides incoming data streams into less than one-second segments, creating discretized streams (DStreams) as high-level abstractions. Each mini-batch within the DStream collection is patterned after a Spark RDD. In this study, Spark is employed for streaming data processing, with Spark streaming managing the Kafka data stream, and MLlib is used to implement machine learning algorithms. Spark adheres to a master-worker architecture for distributed processing. Each Spark application can establish one master process, the executor in Spark, and several worker processes referred to as drivers. These drivers, like the master, are responsible for evaluating, allocating, scheduling, and supervising the tasks among the executors. The driver also maintains the necessary data consistency throughout the program. In contrast, the executors are solely responsible for executing the code assigned by the driver and transmitting the results back to the driver, as depicted in Fig. 5. This architecture ensures the efficient distribution of tasks and data processing within the Spark application.

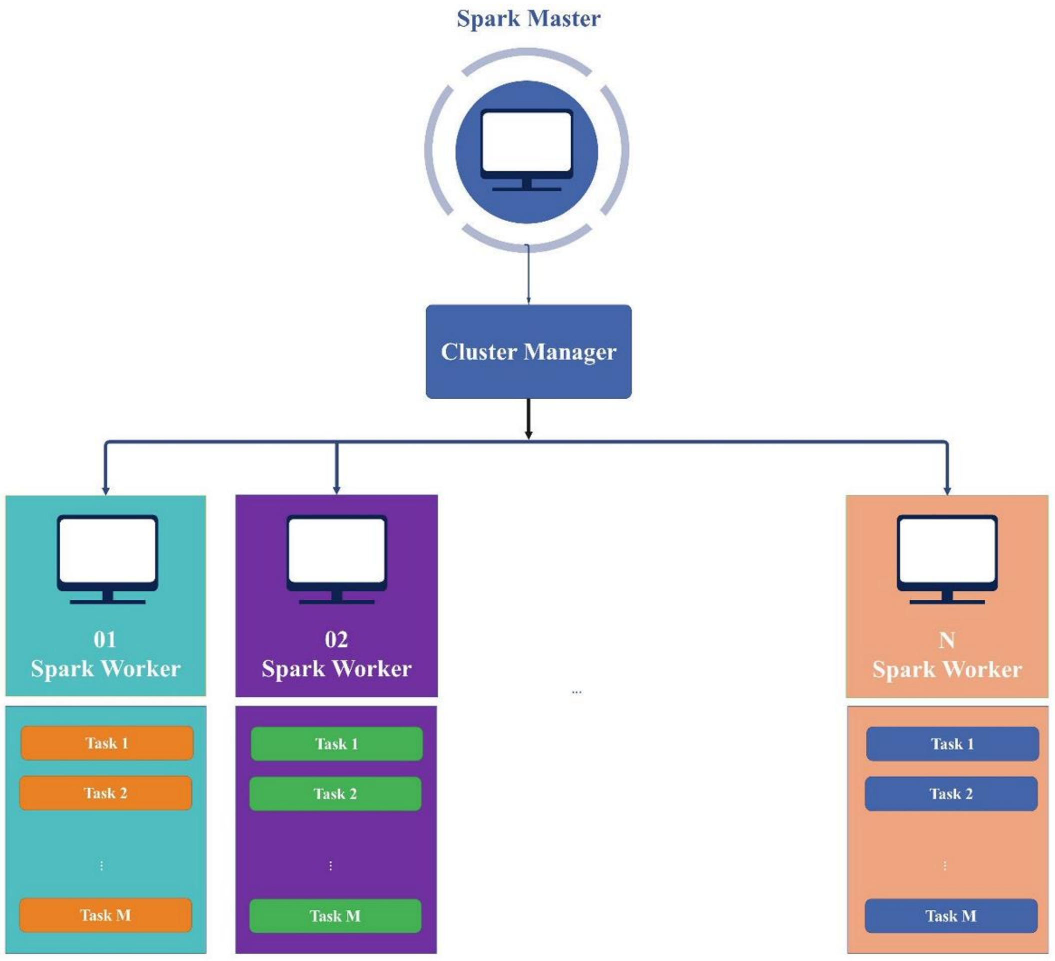

Fig. 5. Master-worker architecture.

The classification of data collected from diverse sources for various diseases necessitates using classification models capable of discerning user characteristics in the presence or absence of a disease. In this research, two classification models have been employed, each briefly introduced below:

1) K-Nearest Neighbor (KNN): KNN is a versatile supervised learning method that can serve as a classification and regression algorithm. It determines the distance between the test data point and all training data points and selects the K training data points closest to the test data. Based on their distances, the test data point is then assigned to the class that most of these K neighbors belong to. This method, represented in Method 1, can be briefly described by Algorithm 1.

Algorithm 1 KNN Algorithm

```
1: procedure KNN(Instance, TestData, K)
2:     C ← Size(TestData)
3:     Dist[C][2] ← 0
4:     for i in TestData do
5:     d ← EclideanDistance(i, Instance)
6:     Dist[i][1] ← d
7:     Dist[i][2] ← Class(i)
8:     end for
9:     Srt ← Sort(dist[:][1]) ▷ Sort 2nd column based on that
10:    Sel ← Srt[1 : K][2]
11:    Cls ← Mode(Sel)
12:    return Cls
13: end procedure
```

2) Support Vector Machine (SVM): SVM is another supervised machine learning method primarily used for classification, although it can also handle regression tasks. In SVM, each data point is represented as a point in an n-dimensional space, where "n" represents the number of features available for classification. Each feature corresponds to a specific coordinate within this space. SVM aims to identify the hyperplane that optimally separates the two classes in the data. Support vectors represent individual data points within this multi-dimensional space, and the SVM classifier seeks to identify the hyperplane or line that maximally divides the two classes. To achieve this, the SVM algorithm considers certain assumptions about the data, aiming to find the best hyperplane:

- Maximizing margin: SVM strives to find the hyperplane that maximizes the margin or the distance between the hyperplane and the nearest data points of both classes. This maximized margin ensures robust separation.
- Support vectors: The data points closest to the hyperplane, known as support vectors, significantly influence the determination of the optimal hyperplane.
- Kernel functions: SVM can employ kernel functions to map the data into higher-dimensional spaces when a linear separation is not feasible. These functions allow SVM to perform non-linear classification effectively. As a classification algorithm, SVM provides the means to efficiently distinguish between different classes within a dataset by defining the most appropriate hyperplane or decision boundary.

The suggested architecture addresses the complex issues involved in forecasting real-time health status in healthcare scenarios powered by the IoT. The applicability of this is emphasized by numerous essential features designed to tackle these particular challenges. The architecture's scalability is a fundamental aspect that allows it to easily handle large and growing datasets often encountered in healthcare. The ability to effortlessly increase resources with data expansion guarantees consistent performance. The ability to analyze data in real-time is another important aspect, allowing for quick intake, analysis, and understanding of streaming healthcare data.

Furthermore, the architecture's distinctive advantage resides in its implementation of distributed machine learning models specifically created to handle the vastness and complexities of medical data. This enables the simultaneous execution of tasks to enhance the efficiency of training models, hence improving the accuracy and speed of health status forecasts. Moreover, the architecture has exceptional proficiency in incorporating various IoT devices and dissimilar data sources, merging distinct data streams for thorough analysis. By prioritizing security and privacy safeguards, adapting to different data speeds from IoT sensors, and maximizing resource efficiency, it effectively tackles the complex difficulties often seen in healthcare situations powered by IoT. In conclusion, these architectural characteristics together enable the system to effectively negotiate the intricacies of real-time health status prediction, establishing it as an optimal framework for handling the distinct requirements of healthcare data analysis in IoT contexts.

IV. Scenario Descriptions

In Scenario 1, Fig. 6 illustrates three hyperplanes labeled A, B, and C. The key principle to selecting the appropriate hyperplane is to choose the one that best separates the two classes. In this scenario, hyperplane B does an excellent job of achieving this separation.

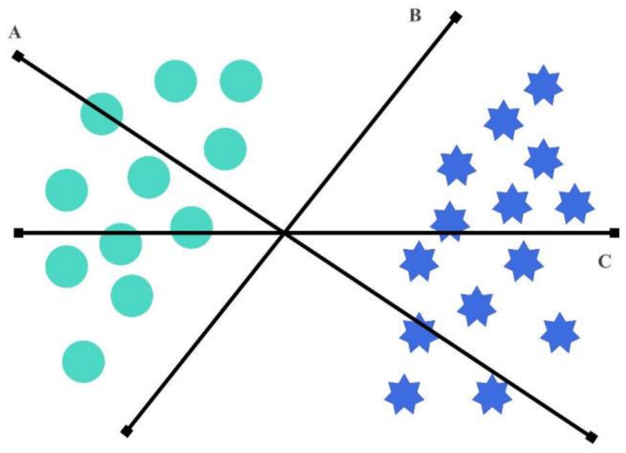

Fig. 6. Three sample hyper-planes.

Scenario 2 presents three hyperplanes (A, B, and C) in Fig. 7. The goal is to choose the hyperplane that maximizes the distance between the closest data point of any class and the hyperplane. This distance is referred to as the margin, as shown in Fig. 8. Hyperplane A has a larger margin than B and C, making it the right choice. Opting for a hyperplane with a larger margin enhances robustness and minimizes the chances of misclassification.

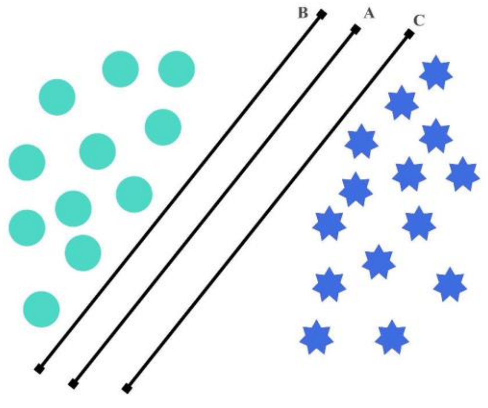

Fig. 7. Three hyper-planes that could separate two classes.

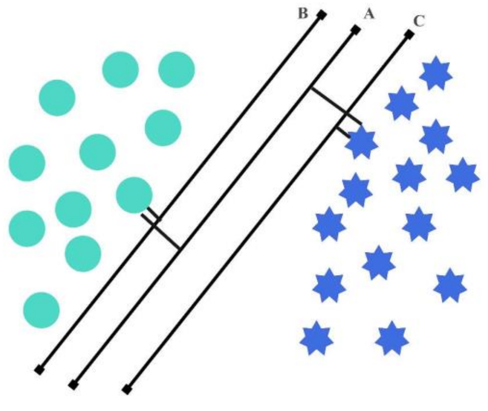

Fig. 8. Comparison of three hyper-planes with margins in scenario 2.

In Scenario 3, although hyperplane B has a larger margin than A, SVM prioritizes proper classification of the classes before maximizing the margin. Hyperplane B makes a classification error, whereas A correctly categorizes everything. Therefore, hyperplane A is selected as the appropriate choice (see Fig. 9).

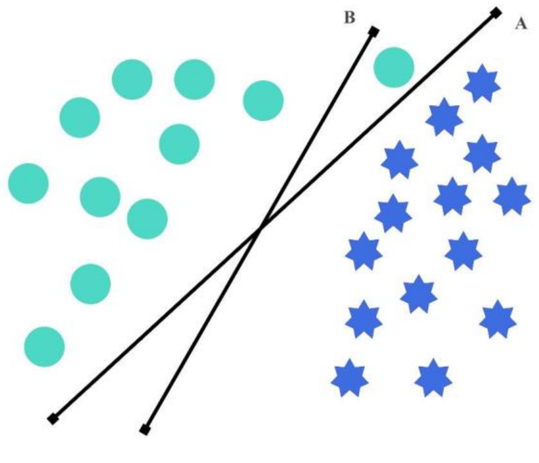

Fig. 9. Evaluation of hyperplanes A and B in scenario 3.

Scenario 4 involves an outlier, represented by the star, residing in the region of the circle class, making it impossible to separate the two classes using a straight line. However, the SVM algorithm can disregard outliers and identify the hyperplane with the maximum margin. As a result, SVM classification is robust against outliers (see Fig. 10).

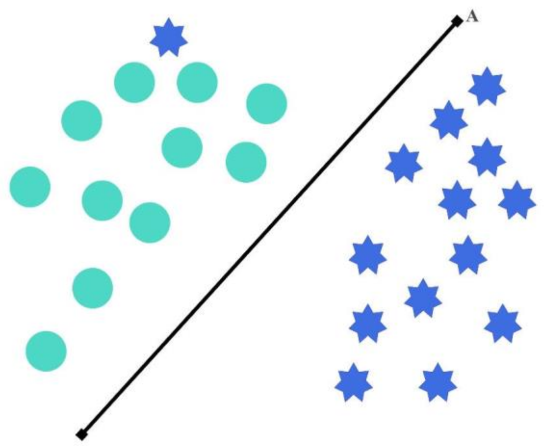

Fig. 10. Robustness of SVM against outliers in scenario 4.

In Scenario 5, when a linear hyperplane is insufficient to categorize two classes, a new feature, z = x^2 + y^2, is introduced to create a three-dimensional representation of the data points, as shown in Fig. 11 and Fig. 12. This new feature, z, is a mathematical construct that enables the creation of a linear hyperplane, making it possible for SVM to classify the two classes effectively. The SVM algorithm employs the "kernel trick" to automatically find this hyperplane. The SVM kernel is a function that transforms non-separable problems into separable ones by projecting data from a low-dimensional input space into a higher-dimensional space. This is particularly valuable for addressing problems with non-linear separations. It performs intricate data transformations before determining how to split the data based on the provided labels or outputs. The hyperplane appears as a circle in the original input space, as depicted in Fig. 13. The kernel trick allows SVM to handle complex, non-linear separations and enables the classification of data that cannot be linearly separated in the original feature space.

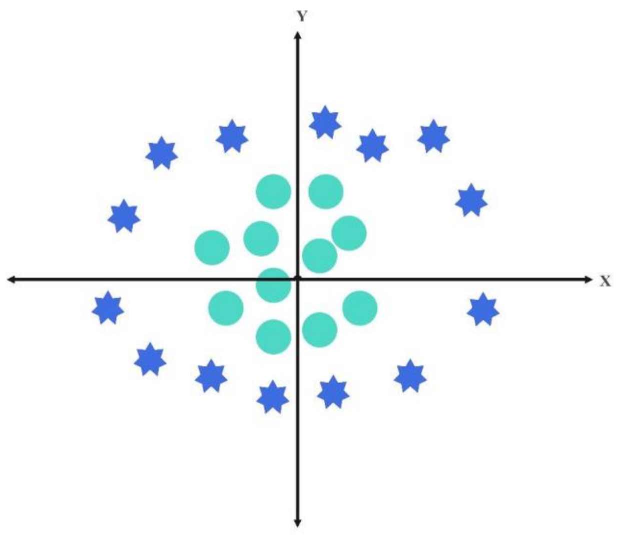

Fig. 11. Introduction of a new feature in scenario 5.

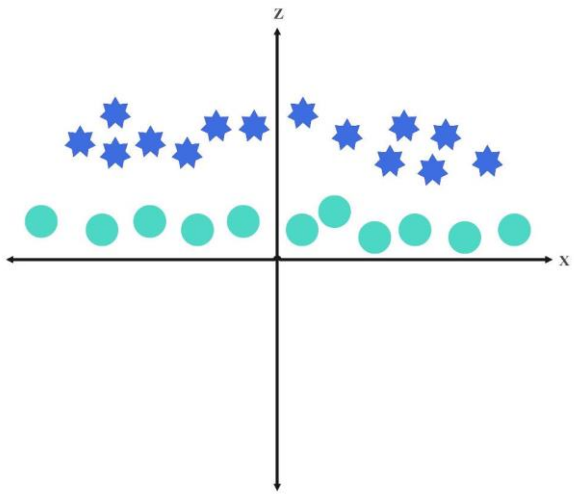

Fig. 12. Three-dimensional representation with additional feature (z) in scenario 5.

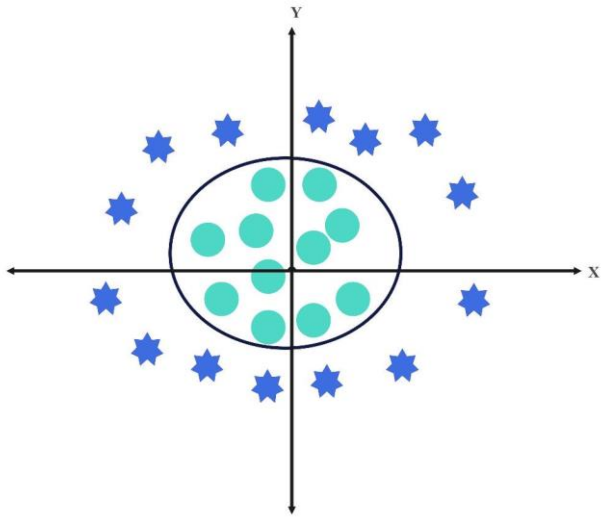

Fig. 13. Application of SVM kernel trick for non-linear separation in scenario 5.

V. Discussion

To ensure high data availability and avoid a single point of failure, it is essential to store the results and data streams generated by each user in a distributed manner. Distributed databases outperform traditional database systems in terms of performance and scalability. Apache Cassandra is an open-source, distributed, and free NoSQL database system designed to handle massive volumes of data, whether structured, unstructured, or semi-structured, across multiple computers. Cassandra's architecture greatly enhances its scalability, operational capabilities, and continuous accessibility. It also offers rapid write and read rates when used with Spark. Distributed databases provide several valuable features:

- Affordability and ease of use: Distributed databases are cost-effective and straightforward.
- Data transfer speed: They offer significantly faster data transport than traditional databases.
- Scalability: Distributed databases can be scaled easily by adding columns, accelerating the processing of larger and more data.
- Cluster scalability: Distributed databases can expand their cluster capacity by adding more nodes without a specific distribution. After processing data with Spark, the output data is stored in a table using Cassandra and a primary key. This database can be accessed later for real-time monitoring, reporting, and analysis of historical data.
- Data replication and partitioning: Data is replicated across various computers to enhance data availability and fault tolerance.

TABLE I. UCI Heart Disease Dataset

No	Attribute No	Attribute Name	Description
1	3	Age	Age of Patients
2	4	Sex	0/1(M/F)
3	9	CP	Type of Chest Pain
4	10	TRestBPS	Blood Pressure when the Patient is on
			Rest
5	12	Chol	Blood Cholesterol
6	16	FBS	Fasting Blood Sugar
7	19	RestECG	ElectroCardioGraphic when Patient
			is on Rest
8	32	Thalach	Heart Rate(Max)
9	38	Exang	Exercise Causes Angina (Y/N =
			1/0)
10	40	OldPeak	Exercise-induced ST depression in
			comparison to rest
11	41	Slope	The Peak Exercises in cline ST section. (UpSloping/Flat/DownSloping = 1/2/3
12	44	CA	Main Vessels Colored with
			Fluoroscopy in Number (0–3)
13	51	Thal	Normal/ Fixed defect/ Reversible
			Defect = 3/6/7
14	58	Num(Class)	Heart Disease Diagnosis (Status of
			Angiographic Disease) if Diameter
			Narrowing¡= 50% =0 Otherwise =1

Apache Zeppelin is an open-source data analysis environment that works with Apache Spark. It is a web-based, versatile notebook that facilitates interactive data analysis, real-time data exploration, visualization, and collaboration. Zeppelin supports an expanding list of programming languages and interfaces, including SparkSQL, Hive, AngularJS, Scala, Python, markdown, and Shell. Using Scala, it can create dynamic, data-driven, and collaborative documents, among other capabilities. Apache Zeppelin is valuable for writing, organizing, and executing analytical code and visualizing results across extensive workflows. Zeppelin can automatically generate input forms in your notebook, provide simple visualizations to present results, and allow colleagues to share the notebook's URL. In real-time data retrieval from the Cassandra database, a Zeppelin dashboard is developed to display data in charts, tables, and other formats. This dashboard updates its data every second, allowing authorized individuals, such as doctors, healthcare companies, or external consultants, to access the data regardless of their patient's or client's health status.

In this research, two datasets obtained from well-known data sources, Kaggle and UCI, were utilized. These datasets pertain to medical conditions, specifically diabetes and heart diseases. Table I provides an overview of the information related to these datasets. It is worth noting that although the Cleveland dataset contains 76 attributes, previous studies have primarily focused on using only a subset of fourteen attributes. Among the various datasets, the Cleveland dataset has been the primary focus of machine learning researchers. The "Class" field in these datasets indicates the presence or absence of a particular medical condition, such as heart disease. The values in the "Class" field range from zero (indicating no presence of the condition) to four, with the Cleveland dataset primarily concentrating on discriminating between the presence (values 1, 2, 3, 4) and absence (value 0) of heart disease.

The dataset used in this research was sourced from Kaggle's Diabetes Dataset. Kaggle is a well-known platform for data science competitions and provides a freely available dataset that numerous authors have used in previous studies. This dataset consists of ten features and 15,000 observations, and it is employed to predict whether a patient has diabetes. Table II offers an overview of the features included in this dataset.

TABLE II. KAGGLE DIABETES DATASET DESCRIPTIONS

No	Attribute Name	Description
1	Patient ID	Patient Identification Number
2	Pregnancies	A patient gets diabetes after Pregnancy
3	Plasma Glucose	Glucose amount in Blood
4	Diastolic Blood Pressure	Blood Pressure when Patient is on Rest
5	Triceps Thickness	Body Fat
6 7 8	Serum Insulin Body Mass Index(BMI) Diabetes Pedigree	Insulin amount in Blood *W eightinKG Height*2 *inM* 2) Diabetes History in Family
9	Age	Patient Age
10	Class	Diabetic = 1, NonDibaetic = 0

VI. RESULTS

The proposed real-time health status forecasting system is driven by a single-node cluster featuring a Core i7 CPU, 16 GB of RAM, and the Ubuntu 20.04 operating system. This system seamlessly integrates the trained model with Kafka streaming data processing and runs on the Spark platform. As depicted in Fig. 14, the application establishes a connection to Kafka streaming and commences receiving data streams from various Kafka producers. When it encounters streams related to health characteristics, it retrieves the attribute values from each topic within the illness events sent via Kafka streaming. Subsequently, it employs the trained model to predict the health state of the individuals. In parallel, the Cassandra database records each forecasted health state in a table, employing the identification (ID) as the primary key, which is ideal for ensuring data redundancy and reliability. This stored data can later be queried to examine historical information.

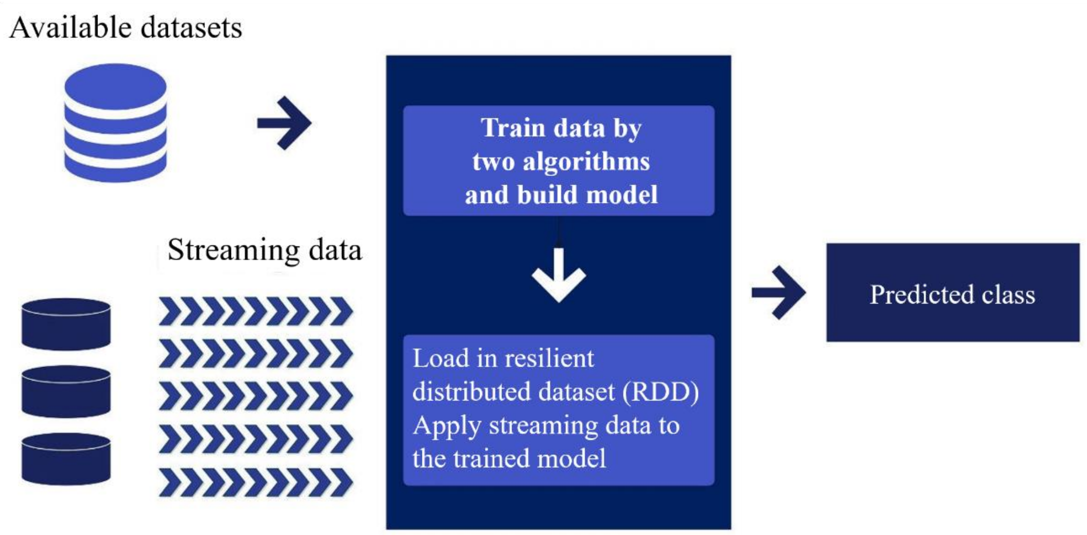
Available datasets
Streaming data
Train data by two algorithms and build model
Load in resilient distributed dataset (RDD) Apply streaming data to the trained model
Predicted class

Fig. 14. Apply classification algorithms.

All the tests were conducted using a cluster configuration consisting of one primary node and two worker nodes, each running the Ubuntu 20.4 operating system within VMware virtual environments. Several steps were undertaken to facilitate communication between the nodes and ensure the proper functioning of the Spark application:

- User accounts and development environment setup: Spark user accounts were created to simplify inter-node communication. Scala and Java were installed. Open SSH Server was set up. Key pairs were generated to enable passwordless SSH configuration across the nodes, ensuring that the Spark master can effectively connect, launch, pause, and run tasks on multiple worker nodes.
- Software installation: Spark, Kafka, and Cassandra were unpacked and installed on a single node. Two themes and corresponding tables were created, one for diabetic disease and the other for heart illness.
- Environment variables: The bashrc file was modified to include essential environment variables like SPARK and JAVA_HOME in the home directory.
- Node replication: To ensure uniformity and consistency across various nodes, the setup folder of the single-node cluster was duplicated multiple times, with one node designated as the master and the others as workers.
- Hostname and host configuration: The hostname and hosts were modified on all nodes to facilitate proper inter-node communication.

The primary stages for implementing the Spark application in a Zeppelin notebook are as follows:

- Spark context and streaming context creation: An instance of Spark contexts and streaming context was created to access all Spark streaming functionalities.
- Direct stream creation: A direct stream was created using the specified Kafka parameters and topics.
- Data extraction: The identifiers and characteristics of each topic and stream were extracted.
- ML model utilization: The pre-trained ML model was used to predict the health status.
- Data storage: All attributes and the predicted labels were saved to the Cassandra keyspace and table.
- Streaming start: The Spark streaming context was initiated using the start method, allowing real-time health data processing.

The research allocated 25% of the data for testing purposes, while the remaining 75% was used to train the machine learning models. The datasets were divided into training and test datasets randomly. To address the issue of the computational cost of sorting feature values across large distributed datasets, an approximate set of candidate splits was identified over a sampled portion of the data. This method has been shown to enable more accurate predictions by analyzing the model error and the test data, effectively mitigating the negative impacts of both underfitting and overfitting. One of the most crucial and valuable measures for assessing the performance of testing and treatment is the Receiver Operating Characteristic (ROC) curve. The MLlib provides support for ROC curve evaluation. On the other hand, classification accuracy is determined by the ratio of all correct predictions to all the prediction data. The classification accuracy for the datasets in this study was assessed using the following equation:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{1}$$

Sensitivity and specificity are two critical metrics used to assess classification models' performance, particularly in medical diagnoses and other fields where accurate predictions are crucial. These metrics are calculated as follows:

Sensitivity (True Positive Rate or Recall): Sensitivity is the percentage of actual positives (e.g., patients with a specific condition) that are correctly identified by the model. It indicates the model's ability to detect true positive cases.

Specificity: Specificity is the percentage of actual negatives (e.g., patients without the condition) that are correctly identified by the model. It measures the model's ability to avoid false positive predictions.

In these equations, true positives (TPs) are the cases correctly classified as positive, true negatives (TNs) are the cases correctly classified as negative, false positives (FPs) are cases incorrectly classified as positive (when they are actually negative), and false negatives (FNs) are cases incorrectly classified as negative (when they are actually positive).

Sensitivity and specificity provide insights into the model's performance in terms of both correctly identifying individuals with the condition and correctly identifying individuals without the condition. Balancing these two measures is important, especially in situations, where missing a true positive (e.g., a medical condition) or incorrectly identifying a false positive (unnecessary treatment or diagnosis) has significant consequences.

$$Sensitivity = \frac{TP}{TP+FN} \tag{2}$$

$$Specifity = \frac{TN}{TN+FP} \tag{3}$$

Our machine learning model's effectiveness was assessed on two established datasets. The empirical results indicate that our utilization of Spark for the execution of the proposed methodology demonstrates notable efficiency and scalability (see Fig. 14 and Fig. 15). Fig. 16 shows the specificity, sensitivity, ROC curve and accuracy obtained in heart disease. The findings further underscore that the proposed model consistently delivers dependable and superior predictive outcomes.

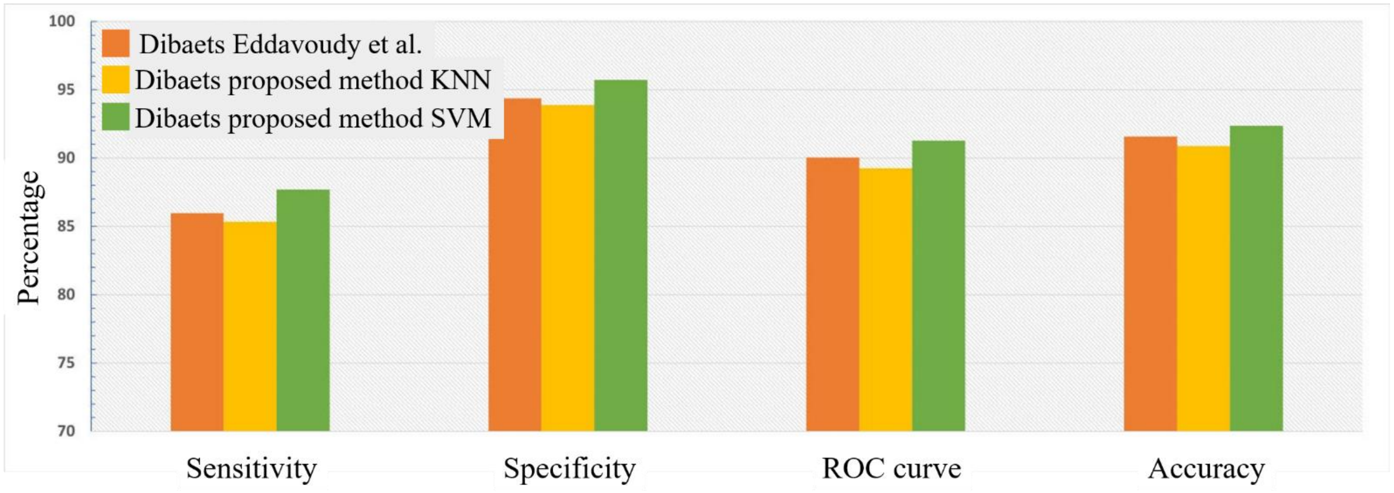

Fig. 15. Specificity, sensitivity, ROC Curve, and accuracy obtained in diabetes datasets in comparison to the other algorithms.

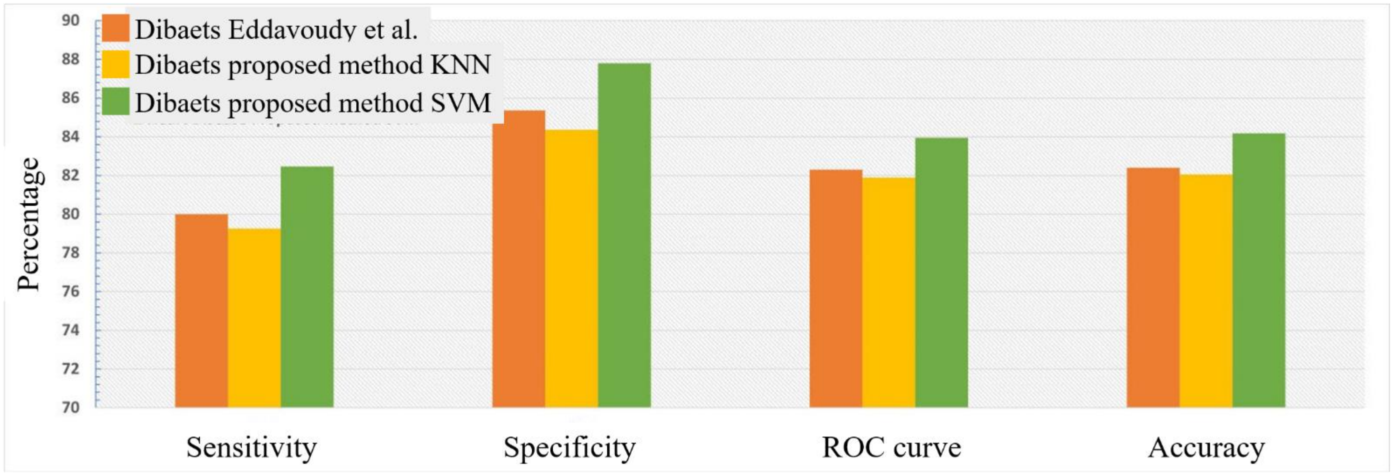

Fig. 16. Specificity, sensitivity, ROC Curve, and accuracy obtained in heart disease datasets in comparison to other algorithms.

VII. Conclusion

This study has demonstrated the successful application of a machine learning model for real-time health status prediction in the healthcare domain. By employing Apache Spark in conjunction with Kafka streaming and Cassandra, we have created an efficient and scalable system for processing and analyzing healthcare data streams. The results of our empirical tests on two distinct datasets reveal that our model consistently provides reliable and high-quality predictions. However, the current design also reveals inherent constraints in the scalability of traditional data storage systems like Cassandra when handling exponentially growing healthcare data. While effective for many use cases, these systems might face challenges in handling future data volume surges, potentially leading to performance bottlenecks and increased resource requirements. The ability to monitor and predict health conditions in real-time is of paramount importance, particularly in the context of chronic illnesses and emergencies. Our proposed system offers a promising solution for continuous health monitoring and early detection, potentially saving lives and reducing healthcare costs. The key takeaway from our research is the effectiveness of combining advanced technologies like Spark, Kafka, and Cassandra to process and analyze healthcare data streams. This approach opens up new possibilities for healthcare analytics and real-time monitoring, benefiting patients, healthcare providers, and the broader medical community. In the future, we envision further refinements and enhancements to our system, including the integration of additional data sources and the development of more sophisticated machine-learning algorithms. As the healthcare sector continues to generate vast amounts of data, the need for innovative solutions like the one presented in this study will only grow, ushering in a new era of data-driven healthcare.

References

[1] M. Asch et al., "Big data and extreme-scale computing: Pathways to convergence-toward a shaping strategy for a future software and data ecosystem for scientific inquiry," The International Journal of High Performance Computing Applications, vol. 32, no. 4, pp. 435-479, 2018.

[2] M. El Samad, S. El Nemar, G. Sakka, and H. El-Chaarani, "An innovative big data framework for exploring the impact on decision-making in the European Mediterranean healthcare sector," EuroMed Journal of Business, vol. 17, no. 3, pp. 312-332, 2022.

[3] M. Karatas, L. Eriskin, M. Deveci, D. Pamucar, and H. Garg, "Big Data for Healthcare Industry 4.0: Applications, challenges and future

perspectives," Expert Systems with Applications, vol. 200, p. 116912, 2022.

[4] S. S. Ghahfarrokhi and H. Khodadadi, "Human brain tumor diagnosis using the combination of the complexity measures and texture features through magnetic resonance image," Biomedical Signal Processing and Control, vol. 61, p. 102025, 2020.

[5] K. Batko and A. Ślęzak, "The use of Big Data Analytics in healthcare," Journal of big Data, vol. 9, no. 1, p. 3, 2022.

[6] A. A. Zein, S. Dowaji, and M. I. Al-Khayatt, "Clustering-based method for big spatial data partitioning," Measurement: Sensors, vol. 27, p. 100731, 2023.

[7] M. Mohtasebi et al., "Detection of low-frequency oscillations in neonatal piglets with speckle contrast diffuse correlation tomography," Journal of Biomedical Optics, vol. 28, no. 12, pp. 121204-121204, 2023.

[8] R. Nathan et al., "Big-data approaches lead to an increased understanding of the ecology of animal movement," Science, vol. 375, no. 6582, p. eabg1780, 2022.

[9] C. Acciarini, F. Cappa, P. Boccardelli, and R. Oriani, "How can organizations leverage big data to innovate their business models? A systematic literature review," Technovation, vol. 123, p. 102713, 2023.

[10] M. Andronie et al., "Big Data Management Algorithms, Deep Learning-Based Object Detection Technologies, and Geospatial Simulation and Sensor Fusion Tools in the Internet of Robotic Things," ISPRS International Journal of Geo-Information, vol. 12, no. 2, p. 35, 2023.

[11] W. Li, "Big Data precision marketing approach under IoT cloud platform information mining," Computational intelligence and neuroscience, vol. 2022, 2022.

[12] M. Q. Bashabsheh, L. Abualigah, and M. Alshinwan, "Big data analysis using hybrid meta-heuristic optimization algorithm and MapReduce framework," in Integrating meta-heuristics and machine learning for real-world optimization problems: Springer, 2022, pp. 181-223.

[13] G. S. Bhathal and A. Singh, "Big Data: Hadoop framework vulnerabilities, security issues and attacks," Array, vol. 1, p. 100002, 2019.

[14] A. Adnan, Z. Tahir, and M. A. Asis, "Performance evaluation of single board computer for hadoop distributed file system (hdfs)," in 2019 International Conference on Information and Communications Technology (ICOIACT), 2019: IEEE, pp. 624-627.

[15] S. Vairachilai, A. Bostani, A. Mehbodniya, J. L. Webber, O. Hemakesavulu, and P. Vijayakumar, "Body Sensor 5 G Networks Utilising Deep Learning Architectures for Emotion Detection Based On EEG Signal Processing," Optik, p. 170469, 2022.

[16] M. Bolhassani and I. Oksuz, "Semi-Supervised Segmentation of Multi-vendor and Multi-center Cardiac MRI," in 2021 29th Signal Processing and Communications Applications Conference (SIU), 2021: IEEE, pp. 1-4.

[17] S. P. Rajput et al., "Using machine learning architecture to optimize and model the treatment process for saline water level analysis," Journal of Water Reuse and Desalination, 2022.

[18] S. R. Abdul Samad et al., "Analysis of the Performance Impact of Fine-Tuned Machine Learning Model for Phishing URL Detection," Electronics, vol. 12, no. 7, p. 1642, 2023.

[19] V. Monjezi, A. Trivedi, G. Tan, and S. Tizpaz-Niari, "Information-Theoretic Testing and Debugging of Fairness Defects in Deep Neural Networks," arXiv preprint arXiv:2304.04199, pp. 1571-1582, 2023 2023, doi: 10.1109/ICSE48619.2023.00136.

[20] W. Anupong et al., "Deep learning algorithms were used to generate photovoltaic renewable energy in saline water analysis via an oxidation process," Water Reuse, vol. 13, no. 1, pp. 68-81, 2023.

[21] M. Khodayari, J. Razmi, and R. Babazadeh, "An integrated fuzzy analytical network process for prioritisation of new technology-based firms in Iran," International Journal of Industrial and Systems Engineering, vol. 32, no. 4, pp. 424-442, 2019.

[22] J. Zandi, A. N. Afooshteh, and M. Ghassemian, "Implementation and analysis of a novel low power and portable energy measurement tool for wireless sensor nodes," in Electrical Engineering (ICEE), Iranian Conference on, 2018: IEEE, pp. 1517-1522, doi: 10.1109/ICEE.2018.8472439.

[23] Y. Lu, Z. Miao, P. Sahraeian, and B. Balasundaram, "On atomic cliques in temporal graphs," Optimization Letters, vol. 17, no. 4, pp. 813-828, 2023.

[24] M. R. Moradi, S. R. N. Kalhori, M. G. Saeedi, M. R. Zarkesh, A. Habibelahi, and A. H. Panahi, "Designing a Remote Closed-Loop Automatic Oxygen Control in Preterm Infants," Iranian Journal of Pediatrics, vol. 30, no. 4, 2020.