Journal of Big Data

RESEARCH Open Access

A new Internet of Things architecture for real-time prediction of various diseases using machine learning on big data environment

Abderrahmane Ed-daoudy* and Khalil Maalmi

*Correspondence: a.eddaoudy@gmail.com
LTTI Laboratory, Higher School of Technology, Sidi Mohamed Ben Abdellah University, Fez, Morocco

Abstract
A number of technologies enabled by Internet of Thing (IoT) have been used for the prevention of various chronic diseases, continuous and real-time tracking system is a particularly important one. Wearable medical devices with sensor, health cloud and mobile applications have continuously generating a huge amount of data which is often called as streaming big data. Due to the higher speed of the data generation, it is difficult to collect, process and analyze such massive data in real-time in order to perform real-time actions in case of emergencies and extracting hidden value. using traditional methods which are limited and time-consuming. Therefore, there is a significant need to real-time big data stream processing to ensure an effective and scalable solution. In order to overcome this issue, this work proposes a new architecture for real-time health status prediction and analytics system using big data technologies. The system focus on applying distributed machine learning model on streaming health data events ingested to Spark streaming through Kafka topics. Firstly, we transform the standard decision tree (DT) (C4.5) algorithm into a parallel, distributed, scalable and fast DT using Spark instead of Hadoop MapReduce which becomes limited for real-time computing. Secondly, this model is applied to streaming data coming from distributed sources of various diseases to predict health status. Based on several input attributes, the system predicts health status, send an alert message to care providers and store the details in a distributed database to perform health data analytics and stream reporting. We measure the performance of Spark DT against traditional machine learning tools including Weka. Finally, performance evaluation parameters such as throughput and execution time are calculated to show the effectiveness of the proposed architecture. The experimental results show that the proposed system is able to effectively process and predict real-time and massive amount of medical data enabled by IoT from distributed and various diseases.

Keywords: Healthcare, Stream processing, Big data, Apache Spark, Distributed machine learning, Internet of Things

Introduction

Over the past two decades our era can be described as big data era where digital data is becoming increasingly important in many domains like healthcare, science, technology and society. A large amount of data has been captured and generated from multiple areas, multiple sources such as streaming machines, high throughput instruments, sensor networks, mobile application and from every single field especially in healthcare, this high data volume represents big data [1]. Storing, processing, visualizing and knowledge extraction through this voluminous and varied data types has become a challenge using inadequate state of-the-art technologies tools. One of the most important technological challenges of big data analytics is exploring ways to effectively obtain valuable information for different types of users. Currently, the various forms of healthcare data sources are being collected in both clinical and non-clinical environments, where the digital copy of a patient's medical history are the most important data in healthcare analytics.

Therefore, designing a distributed data system to deal with big data faces three main challenges: First, due to the heterogeneous and huge volume of data, it is difficult to collect data from distributed locations. Second, storage is the main problem for heterogeneous and massive datasets. Big data system needs to store while providing performance guarantee. Last challenge is related to big data analytics, more precisely to mining massive datasets in real-time or near real-time that include modeling, visualization, prediction, and optimization [2]. These challenges require new processing paradigm as the current data management systems are not efficient in dealing with heterogeneous nature of data or the real-time. However, traditional relational database management systems (RDBMS) such as MySQL are mainly employed for management of structured data. These traditional systems do not provide any support for unstructured or semi-structured data. From a scalability perspective, when the data size grows, there are many standard RDBMS failures in scaling for parallel hardware management and fault tolerance, which is not suitable for managing growing data. To deal with the problems associated with massive and heterogeneous data storage, many research works have been proposed by the research community, such as NoSQL database management systems [3] which are useful when working with a huge quantity of data when the data's nature does not require a relational model [4].

MapReduce [5] is a parallel processing technique to process massive data distributed on a commodity cluster; it consists of the Map and Reduce operations. One of the major limitation of MapReduce is its inefficiency in running iterative algorithms. MapReduce is not designed for iterative processing. Hadoop (High-availability distributed object-oriented platform) is a batch processing system used for distributed storage and processing of big data using the MapReduce programming model. It offers a distributed storage system via its Hadoop Distributed File System (HDFS), it also highly fault tolerant. Hadoop supports batch processing only, it is not suitable for real-time stream processing and in-memory computation and it is not always easy to implement the MapReduce paradigm for all problems. Depending on the volume of the data being processed, the output can be delayed significantly. In contrast, stream computing involves continual input and outcome of data and it is emphasizes on the velocity of data. Big data streaming computing (BDSC) provides high throughput, distributed messages, real-time computing and low-latency processing. With it's massively parallel processing architectures, BDSC is a good choice to gain useful knowledge from big data which is the key requirement of big data analytics in healthcare.

The rapid expansion of large data analyzes has begun to play a pivotal role in the development of healthcare practices and research. It has provided tools for the collection, management, analysis and absorption of large amounts of disparate, structured and unstructured data produced by existing healthcare systems [6]. Nowadays, BDSC plays an important role in big data analytics to get the hidden value of big data in healthcare in real-time. However due the healthcare distributed data sources (the data are coming from the different sources), such as relational databases, Hadoop, search system and other analytics system. Applying machine learning on this big data stream is challenging as the traditional machine learning systems are not suitable to handle such massive volume or varied velocity. Other problem is related to the analytical data processing. Performing richer analytical data processing involves efficient data integration between systems. Most of the state of the art works involve machine learning, but in case of real-time machine learning applied to streaming big data is not handled. On the other hand most of the healthcare analytics solution mainly focused on Hadoop which is a batch oriented computing. Recently, the number of elderly and citizens suffering from chronic diseases is rising rapidly, disadvantages of conventional health services are becoming more and more important. Moreover, the use of medical IoT is increasing for continuous monitoring in order to perform real-time actions in case of emergencies especially for heart disease. Therefore, the millions of sensors generate massive volume of data. Processing these data and performing real-time actions in critical situations is a challenging task.

Based on the challenges facing the healthcare system we have proposed and developed a solution in healthcare with a real-time health status prediction use case. This solution based on the Kafka data streaming, Spark streaming, Spark MLlib, NoSQL Cassandra, and Apache Zeppelin. Multiple streams of messages that are generated from Kafka's producers are processed at Spark streaming with machine learning, then are stored in a distributed storage NoSQL for visualization and analytics. Efficient processing of data in healthcare increases the quality of patient monitoring.

The rest of this paper is illustrated over a few sections: In "Background", we present a brief introduction to big data challengers in healthcare with related work followed by detailed description of the proposed system in "Methods" section. Section "Experiments" presents the implementation process, while section "Results and discussion" presents results and discussion of proposed model. Finally, in section "Conclusion" we conclude the paper and present future work.

Background

Big data challenges in healthcare

The healthcare industry today generates large amount of data that can be described with the 5V's big data characteristics mainly Volume, Variety, Velocity, Veracity and Value [7]. The volume refers to the healthcare data to be collected and analyzed are considerable and constantly increasing, variety make reference to the healthcare data collected from multiple sources. The healthcare data and domain knowledge in health field should be up to date namely velocity. The veracity refers to the reliability of the healthcare data. Finally, valuable information could be found by carefully analyzing the massive data in healthcare. Healthcare data comes from distributed sources such as, electronic medical records, clinical images, diagnosis data and health claim data, streaming system, sensors attached to the patient's bedside to continually track patient vitals. They produce huge chunks of data where the traditional data processing system are inadequate to deal with them effectively [8]. The big data challenges can be summarized in Fig. 1 [9].

In this paper we focused on the first five major big data challenges such as data integration, data processing, data mining techniques, data storage and data visualization.

Related work

Nowadays, big data analytics especially healthcare analytics has become an important issue for a large number of research areas such as data mining, machine learning with the huge increasingly healthcare data as well as the potential information inside. The evolution of science and technology in healthcare has made a significant breakthrough in data collection. In healthcare sector, data are collected by three main types of digital data such as clinical records, health research records and organization operations records [10] provides a brief overview of healthcare data sources.

The extraction of knowledge from these distributed, large and various amount of data has become a challenging task using traditional techniques of data mining which is the process of extracting hidden interesting patterns from massive database. Techniques of data mining help to process the data and turn them into useful information. Many prediction and recommendation systems have been studied in healthcare. In [11] an experiment was performed for the prediction of heart attacks and comparison to find the best method of prediction. A breast cancer classification is performed by using genetically optimized neural network model [12]. Other data mining and information retrieval techniques have been proposed in [13, 14].

Healthcare analytic has been studied in many systems such as epidemic prediction and prevention, health recommendation system, medical decision making in order to improve quality of care, taking, reducing costs, and increasing efficiency. In [15] a cloud based K-means clustering running as a MapReduce job has been proposed which use healthcare data on cloud for clustering. A web enabled distributed electronic and

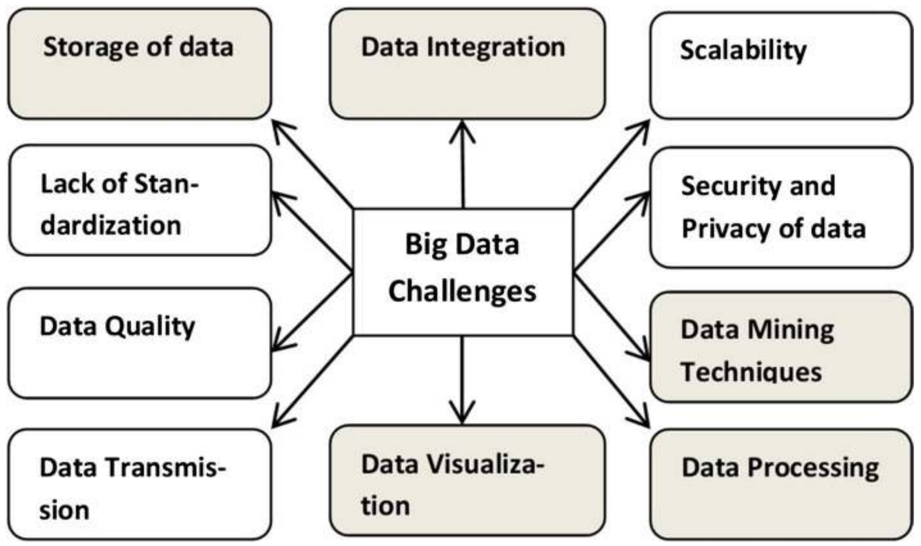


Fig. 1 Big data challenges

personal health record management framework is proposed using Hadoop and HBase [16].

In [17], predicting diabetes mellitus and type of treatment to be adopted is performed by using the predictive analysis algorithm and Hadoop MapReduce environment. A Hadoop based intelligent care system is proposed in [18] that illustrates Internet of Things (IoT) based big data contextual sharing across all devices in a health system. The proposed system adopts a network architecture with enhanced processing capabilities for collecting data generated by different connected devices. The collected data are forwarded to intelligent building. Real-time analysis focused on electronic medical records produced from many sources such as medical devices and mobile applications is described in [19]. The proposed framework combined Hadoop, MongoDB and improvised treatment technique which was meant for improving the results of the treatment of patient records.

Most of the healthcare analytics solution mainly focused on Hadoop [20], it can process a large volume and diverse data sources in case of batch oriented computing. Hadoop would be limited for real-time computing, which Spark is faster than Hadoop and has a better performance especially in problems involving iterative machine learning [21]. Hadoop and Spark are both Apache projects and most popular tools in the big data ecosystem, with great excitement around Spark. Table 1 cover some differences between these two platforms. On the other hand a number of scalable machine learning algorithms are developed to overcome the various issues in big data analytics. In [22] a predictive model related to the risk of diabetes is performed using a scalable Random Forest classification algorithm. Usage of online logistic regression for detection of phishing URL is discussed in [23] where Hadoop is used for data processing and Mahout for machine learning. An automated method that is able to detect abnormal patterns for the elderly living alone entering and exiting behaviors collected from simple sensors equipped in home-based setting is described in [24], the method is based on markov chain model. A real-time medical emergency response system that involves IoT based medical sensors deployed on the human body is discussed in [25]. An overview of big

Table 1 Spark and Hadoop MapReduce comparison

	Hadoop MapReduce	Apache Spark
Definition	Open source big data framework wich deals with structured and unstructured data that are stored in HDFS, Hadoop MapReduce is designed in a way to process a large amount of data on a cluster	Open source big data framework, it's a flexible in-memory framework that allows it to handle batch and real-time analytic and data processing workloads. Spark is basically designed for fast computation
Speed	Reading and writing from/to the file system and disk slows down the processing speed	100 times faster in memory and 10 times faster even when runing on disk than hadoop MapReduce. Because of run computation in memory
Easy of use	In Hadoop MapReduce, developers need to code each operation and require abstractions, so it is difficult to easily program each problem	Spark is easier to use than Hadoop, because it has whole of high-level operators with RDDs
Real-time analysis	No	Yes
Execution model	Batch	Batch, streaming
In-memory	No	Yes

data architectures and machine learning algorithms in healthcare is provided in [26]. Machine learning is involved in different research works, but streaming data is only processed in a few works. Various research works were done to expose useful information in analysis of the social media data especially twitter and other sources for effective healthcare. For example a real-time flu and cancer surveillance system by mining twitter data is described in [27]. A model for real-time analysis of medical big data is proposed in [28]. The approach is exemplified through Spark streaming and Apache Kafka using the processing of healthcare big data stream. In [29] a real-time health status prediction system is proposed, this work focuses on applying machine learning especially DT on data streams received from socket streams using Spark. In paper [30], authors propose a new heart disease monitoring system based on a new classification approach. It consists of the real-time distributed machine learning which uses the real-time predictive analysis algorithm in the Spark environment to predict heart disease.

Most of these works either consider a specific healthcare data sources or only focus on batch oriented computing. But in reality, healthcare data sources are divers and continuously generating various data with high rate. Furthermore, either consider power tools for data analytics such as machine learning and data mining or focus only on data storage and visualization. Hence, real-time analysis of healthcare that include stream data collection, real-time processing and power tools of machine learning, distributed data storage and real-time analytics is needed to build efficient system in dealing with distributed health data stream.

Over the last few decades, heart diseases and diabetes are the most common cause of global death. So early detection of these diseases and continuous monitoring can reduce the mortality rate. In addition, the availability of wearable health monitor, medical IoT technology adopted in the healthcare system and amount the growing patients diseases triggered the idea of taking benefit of big data technologies to predict health status in real-time. Real-time prediction can reduce physician attendance time, help doctors and patients react in advance to a probable disease. Another important feature of the proposed approach was that once the patient disease is not normal, the emergency service is notified at once through an alert technology to perform real-time actions in case of emergencies.

Methods

Proposed architecture for real-time health status prediction and analytics system

The proposed system is a data processing, monitoring application combining Kafka streaming and Spark streaming. This application will process real-time data sent by connected devices and store that data for real-time analytics. Figure 2 shows the architecture of proposed system. Firstly, Kafka producers continuously produce a stream of data messages, which are captured by Kafka streaming, a stream that is coming in the Kafka streaming is modeled by a topic which gives name to the multiple diseases. They are sent to the Spark streaming application, where the real-time processing is performed. The Spark streaming receives multiple health attributes from Kafka streaming and apply machine learning model to predict health status and store data in NoSQL Cassandra.

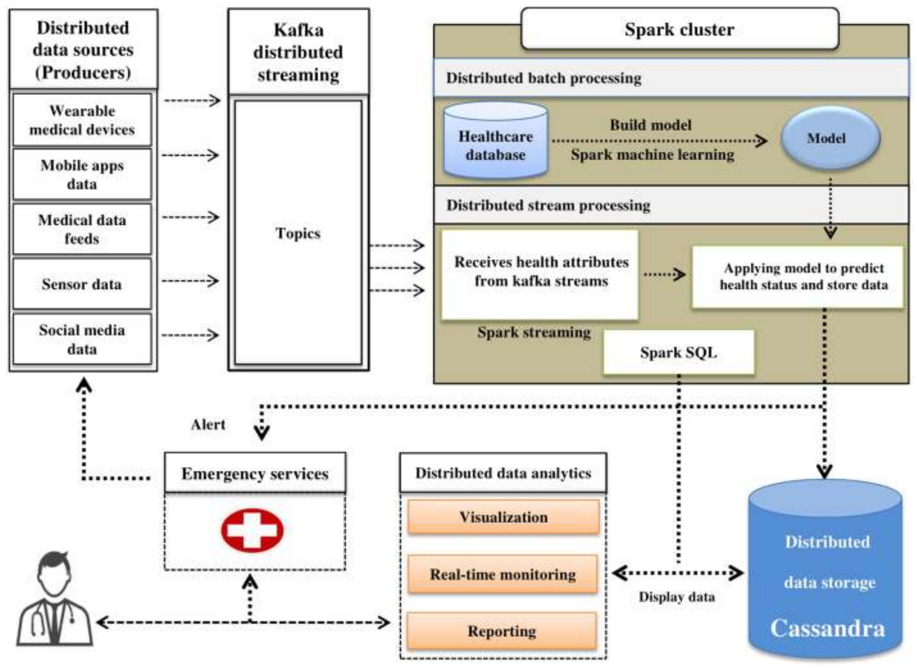


Fig. 2 Architecture of real-time health status prediction and analytics system

Using Apache Zeppelin the data will be retrieved from database and making dashboard that displays data in charts, lines and tables in real-time. Based on the proposed system architecture, data from monitors (IoT) can be analyzed in real-time and send an alert to care providers, so they know instantly about changes in a patient's condition. The data will be refreshed automatically by fixing times intervals. The following subsections give detailed flow.

Data sources

The Internet of Things (IoT) is a network of physical devices and other items, embedded with electronics, smart clothing, software and smart applications, sensors, and network connectivity, so they can collect and exchange data with each other or with data centers systems. With the availability of wearable health monitor at many homes, the data generated by these devices is large in volume and random in nature and needs to be analyzed using a big data analytics system in order to understand the user behavioral patterns or extract the critical information. By 2020, 40% of IoT-related technology will be health-related, more than any other domain [31]. The convergence of medicine and information technologies such as medical informatics will transform healthcare as we know it, reducing inefficiencies, curbing costs, and saving lives. Real-time monitoring via IoT can save lives in event of a medical emergency like heart disease, diabetes and in many other chronic diseases. Many sources related to health are now available which constantly monitor health indicators. Figure 3 shows the workflow for the proposed system with different data sources.

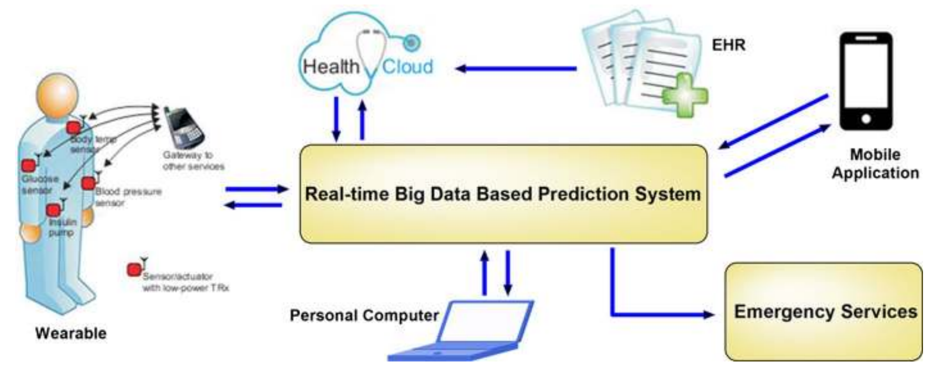


Fig. 3 Workflow for the proposed system

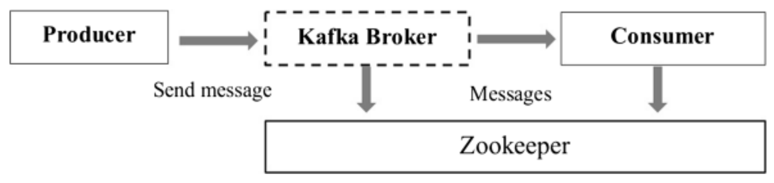


Fig. 4 Kafka messaging system

Kafka real-time data collection

As the data generated in healthcare field is growing at an exponential rate, managing this data with Spark itself becomes a challenge task, while Kafka is designed specifically to streaming data managing. Hence, it has been integrated in our system. In the proposed system architecture, data collection block is used for collecting the individual's health data from distributed sources and multiple diseases using different devices integrated with telemedicine and telehealth. This bloc collect, filter and manage the patient's clinical data in a continuous manner. It allows us to classify streaming data into corresponding topic (kind of disease) in which records are published.

Apache Kafka [32] is a distributed streaming system that uses publish-subscribe messaging and is developed to be a distributed, partitioned, replicated service. The real-time data is streamed from the health monitoring devices through Kafka producer. Kafka servers store all incoming messages from publishers for some period of time and publish them to a stream of data called topic which is a category name to which records are published, topics are the core abstraction which Kafka provides for a stream of records. Each of these topics is split into multiple partitions, each storing one or more of those partitions with ability to accept multiple formats. On the consumer side, Kafka consumers subscribe to one or more topics, and receive data as it's published. A stream or topic can have many different consumers like real-time consumer, all with their own position in the stream maintained. Figure 4 shows the Kafka messaging system. The coordination and facilitation of distributed system is performed by using Zookeeper [33]. In our case study, data producers are two simulator applications for connected devices and uses Apache Kafka to generate data events.

Spark streaming data processing

Apache Spark [34] is an open-source distributed processing engine, designed for fast computation. The major feature of Spark that makes it unique is its ability to perform in-memory computations, but it can also perform disk-based processing when data sets are too large to fit into the available memory, ease of use and complex analysis framework of large data processing. Spark uses the concept of Resilient Distributed Datasets (RDDs) [35] which is the immutable distributed collection of objects. Internally, Spark distributes the data in RDD to different nodes across the cluster to achieve parallelization. RDDs can cache both input data and intermediate data in memory which largely reduces the Input-Output cost for reading and writing from and to the file system allowing it to be reused efficiently especially for iterative machine learning algorithms. Once the data is loaded in a RDD, two basic types of operation can be performed:

- Transformations: that create a new RDD from the existing RDDs by applying processes such as mapping, filtering and more.
- Actions: compute a result based on RDD, and either returned or saved to an external storage system.

Spark provide a machine learning library MLlib, it consist of popular learning algorithms such as classification, regression, clustering etc.

Spark streaming is built on top of core Spark API for live processing of data from various sources like Twitter and Kafka. Incoming data stream is grouped into batches of interval less than a second and processed by the batch processing Spark engine integrating the powerful features to near real-time processing. Spark implements an extension through the Spark streaming module providing a high-level abstraction called discretized stream or DStream which is a sequence of mini-batches where each mini-batch is represented as a Spark RDD.

In this work the streaming data processing task uses Spark where Spark streaming handles the Kafka data stream using Spark streaming library, while the DT implementation is performed using the Spark machine learning library, MLlib.

Spark architecture

Spark is a distributed processing engine and it follows the master-worker architecture, so for every Spark application it will create one master process and multiple workers. In Spark terminology, the master is the driver and the workers are the executors. Since the driver is the master, it is responsible for analyzing, distributing, scheduling and monitoring work across the executors. The driver is also responsible for maintaining all the necessary information during the lifetime of the application. On the other side, executors are only responsible for executing the code assigned to them by driver and reporting the status back to the driver, Fig. 5.

Use case datasets

To implement the proposed model for this research, two datasets have been used. The data set we used for diabetic data analysis is taken from a website named Kaggle [36] which provides online datasets for data scientists and aims at discovering and seamlessly

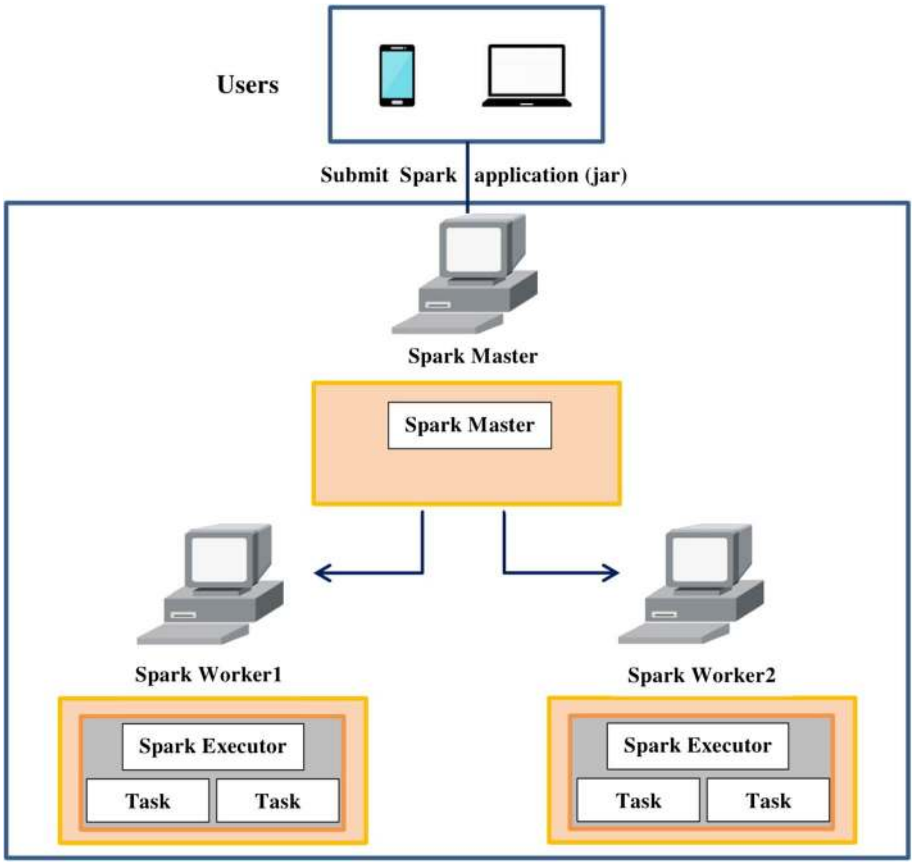


Fig. 5 Spark architecture

analyzing open data. The diabetes dataset consist of 15 000 records and nine attributes, each record has eight attributes which are pregnancies, glucose, blood pressure, skin thickness, insulin, bmi, diabetes pedigree function, age and one of the two possible outcomes, namely whether the patient is tested positive for diabetes indicated by 1 or not indicated by 0.

The second one is the processed.cleveland.data of Heart Disease (HD) database, it was used and analyzed. This is a labelled dataset which consist of 303 records and 14 attributes (Table 2). It was used in many machine learning research works. For each heart disease observation, we have constructed a labelled dataset with attributes, where class label attribute labelled with two classes, presence of heart disease and absence of heart disease. The class label attribute values modified to just 0 and 1, where value 1 indicates presence of heart disease replacing values 1, 2, 3 and 4 while value 0 indicates absence of heart disease, turning it to a binary class dataset.

In this module, datasets are analysed using the predictive analysis approach using Spark environment. The data is loaded from the csv file into an RDD of Strings. We use the map transformation on the RDD, which will apply the Parse RDD function to transform each String element in the RDD into an RDD of Labeled Point and use it for training and testing the machine learning model which predicts health status. As the focus of this work is primarily on real-time data collection, streaming data

Table 2 Heart disease dataset attributes description

No	Attributes	Description
1	Age	Age in years
2	Sex	Sex (1= male, 0= female)
3	Cp	Chest pain type
4	Restbpss	Resting blood pressure
5	Chol	Serum Cholesterol
6	Fbs	Fasting blood sugar
7	Restecg	Resting electrocardiographic results
8	Thalach	Maximum heart rate
9	Oldpeak	ST depression induced by exercise relative to rest
10	Exang	Exercise induced angina
11	Slope	Slope of peak exercise ST segment
12	Ca	Number of major vessels colored with fluoroscopy
13	Thal	3 (normal), 6 (fixed defect), 7 (reversible defect)
14	Num	Class (1 = presence of heart disease, 0 = absence of heart disease)

processing, distributed machine learning and distributed storage, the datasets used and the expected related health status is not very important since this datasets can be easily replaced by any other relevant dataset.

Spark implementation of parallel decision tree

After collecting the data from distributed sources of various diseases, the classification of these data needs to build a classification model which is capable to classify the attributes of a user in absence or presence of disease. A classification is one main technique of data mining useful to find hidden information. The classification consists of examining the characteristics of a newly introduced element in order to assign it to a class of a predefined set. DT are widely used for classification and regression problems. DT are popular methods for the machine learning tasks of classification, it used extensively in machine learning because they are easy to use, easy to operationalize, easy to interpret and extend to the multiclass classification setting. The prediction has been performed using DT based on Spark's machine MLlib which supports DT for binary and multiclass classification.

A DT is a machine learning model that partitions the data into subsets. The partitioning process starts with a binary split and continues until no further splits can be made. Recursive partitioning is the step-by-step process by which a DT is constructed by either splitting or not splitting each node, each partition is selected by finding the best among all possible splits. The split is based on a particular criterion such as Gini impurity and Entropy. The measure of the homogeneity of the label at the node level is based on the impurity of the node. Currently, the implementation provides two classification impurity measures which are Gini and Entropy.

The most popular representative of DT is C4.5 it was developed by J. Ross Quinlan [37], it was the standard algorithm for DT on Spark which has the same parallel idea with C4.5 on MapReduce (Algorithm 1).

Algorithm 1 DT algorithm description

```
Input: training dataset T; attributes S.
Output: decision tree Tree
  if $T$ is $NULL$ then
    return failure
  end if
  if $S$ is $NULL$ then
    return Tree as single node with most frequent class label in $T$
  end if
  if $|S| = 1$ then
    return Tree as single node $S$
  end if
  set Tree = {}
  for $a \in S$ do
    $SetInfo(a,T) = 0, andsplitInfo(a,T) = 0$
    Compute Entropy(a)
    for $v \in$ values($a$,$T$) do
      set $T_{a,v}$ as the subset of $T$ with attribute $a = v$
      $Info(a,T) += \frac{T_{a,v}}{T_a}$ Entropy($a_v$)
      $SplitInfo(a,T) += \frac{T_{a,v}}{T_a} \ log\frac{T_{a,v}}{T_a}$
    end for
    $Gain(a,T) = Entropy(a) - Info(a,T)$
    $GainRatio(a,T) = \frac{Gain(a,T)}{SplitInfo(a,T)}$
  end for
  set $a_{best} = argmax\{GainRatio(a,T)\}$
  attach $a_{best}$ into Tree
  for $v \in$ values($a_{best}$,$T$) do
    call C4.5($T_{a,v}$)
  end for
  return Tree
```

In this algorithm the entropy of attribute *S* is calculated as:

$$Entropy = -\sum_{j=1}^{C} p(S,j) * logp(S,j) \tag{1}$$

It represents the ratio of instances in S which has the j-th class label, C denote the number of classes and $p(S,j)$ is the proportion of instances in *S* that are assigned to j-th class.

$$Info(S,T) = -\sum_{v \in Values(T_s)} \frac{|T(S,v)|}{|T_S|} Entropy(S_v) \tag{2}$$

is the information needed after splitting by attribute S, where values T_s is the set of values of *S* in *T*, T_s is the subset of *T* induced by *S* and $T_{s,v}$ is the subset of *T* in which attribute *S* has a value of *v*. Accordingly information gain is defined as:

$$Gain(S,T) = Entropy(S) - Info(S,T) \tag{3}$$

which measures the information gain after splitting by attribute *S*. The information gain ratio of attributes *S* is defined as:

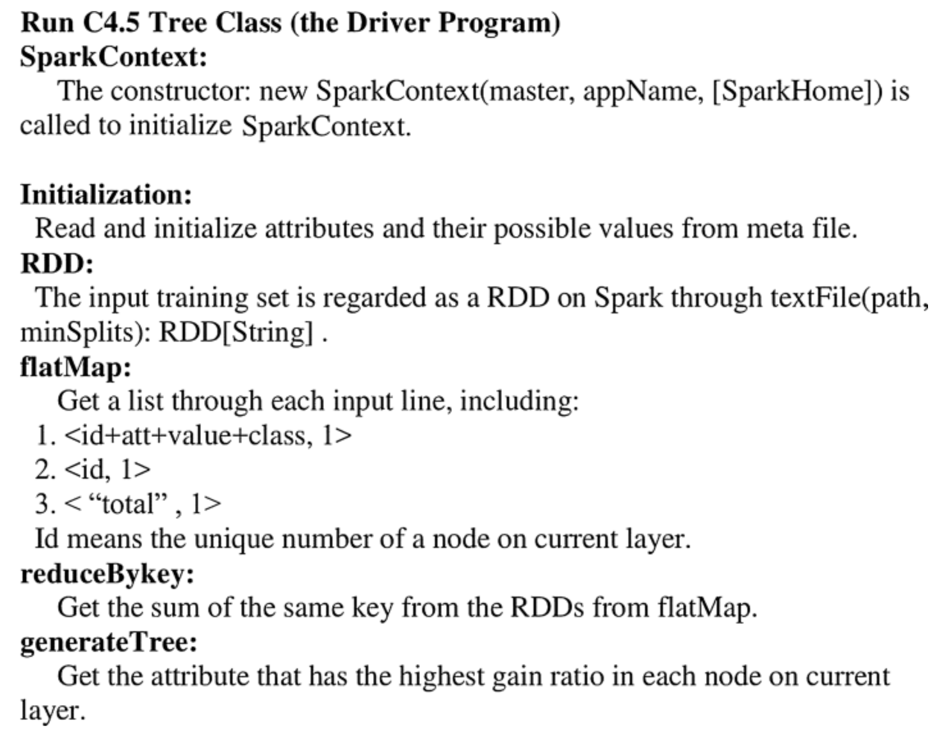
$$GainRatio(S,T) = \frac{Gain(S,T)}{SplitInfo(S,T)} \tag{4}$$

where SplitInfo is defined as :

$$SplitInfo(S,T) = -\sum_{v \in Values(T_s)} \frac{|T(S,v)|}{|T_S|} * log \frac{|T(S,v)|}{|T_S|} \tag{5}$$

Hence, an adequate and parallel model for predicting health status in big data context using Spark is needed. Based on this, a C4.5 model adaptation is more important. In this work the parallelization of C4.5 is performed using Spark. The pseudo-code of C4.5 on Spark is illustrated in Fig. 6.

Firstly, we use SparkContext to get access to the cluster. Loading data to an RDD using textFile() function. The input training dataset is regarded as a RDD on Spark through textFile(). The .cache() method is used, it caches an RDD reused without re-computing. A flatMap function is another transformation operation of Spark, it is almost similar to the map function in MapReduce framework. The reduceByKey function is the parallel version of reduce in MapReduce framework that merges the values for each key using the provided function and returns an RDD. Algorithm 2 represents the steps to train and test the DT on Spark based distributed environment. In this work, Spark streaming handles the Kafka topic data streams using Spark streaming library, while the DT implementation is performed using MLlib. The Machine learning process is given in Fig. 7.


Run C4.5 Tree Class (the Driver Program)
SparkContext:
The constructor: new SparkContext(master, appName, [SparkHome]) is called to initialize SparkContext.

Initialization:
Read and initialize attributes and their possible values from meta file.
RDD:
The input training set is regarded as a RDD on Spark through textFile(path, minSplits): RDD[String] .
flatMap:
Get a list through each input line, including:
1. <id+att+value+class, 1>
2. <id, 1>
3. < "total" , 1>
Id means the unique number of a node on current layer.
reduceBykey:
Get the sum of the same key from the RDDs from flatMap.
generateTree:
Get the attribute that has the highest gain ratio in each node on current layer.

Fig. 6 Implementation of C4.5 on Spark

Algorithm 2 Steps to train and test the DT on Spark

```
Step1: Start new SparkContext
        Laoding required package and APIs
        sparkContext(master,appName, sparkHome)
step2: Load and parse the dataset into an RDD
        rowData(RDD) : sc.textFile(path)
        Data(RDD) : Map(pareseFunction(rowData))
        parse each input line in parallel
Step3: Split the data into training and test sets
        Set related parameters
        trainData(RDD), testData(RDD): randomSplit(Data)
        trainData.cache(): cache the trainData in memory
        testData.cache(): cache the testData in memory
        train the model
step4: Test the model
        LabelAndPredict(RDD) : Map(predictFunction(testData)) parse and predict each input line
in parallel
        Save model : save(sc, path)
```

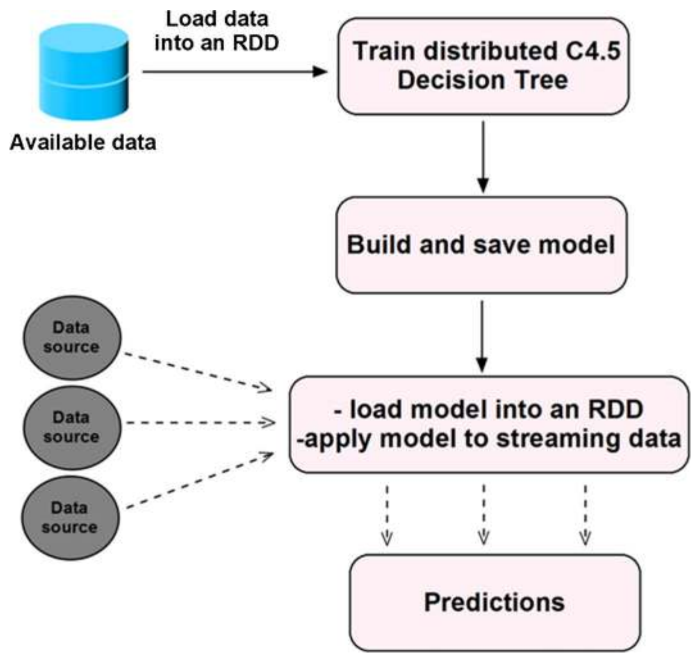


Fig. 7 Machine learning process

Data storage and visualization

The results as well as data streams generated by all the user needs to be stored in a distributed way to ensure the data availability with no single point of failure. Distributed databases are more scalable and provide better performance compared to traditional database systems. Apache Cassandra [38] is a free, open-source and distributed NoSQL database system designed for managing large amounts of structured, semi-structured and unstructured data across many commodity servers, it provides high availability with no single point of failure. The architecture of Cassandra greatly contributes to its being able to scale, perform and offer continuous availability. Also Cassandra provides extremely fast write and read speeds with Spark [39]. The distributed databases have several features that each bring value added to their use:

- The reasonable cost and ease of implementation.
- Partitioning and replication of data across multiple machines.
- Scalability by adding columns, allowing more data to be processed quickly, especially larger data.
- The speed of data transfer compared to conventional databases.
- Scalability by adding additional nodes to the cluster without the need to create a distribution. Here after data processing with Spark, the result data is stored in a table with a primary key through Cassandra. Data stored in database will be queried later for historical data analysis, visualizing, reporting and real-time monitoring.

Apache Zeppelin [40] is a web based and multipurpose notebook that enables interactive data analytics, it is an open-source data analysis environment that runs on top of Apache Spark. The notebook supports real-time interactive data exploration, visualization, and collaboration. Zeppelin supports a growing list of programming languages and interfaces, including Scala, Python, Hive, SparkSQL, AngularJS, shell, and markdown. It can make beautiful data driven, interactive and collaborative documents with scala and more. Apache Zeppelin is useful for working interactively with long workflows: developing, organizing, and running analytic code and visualizing results. Zeppelin can dynamically create input forms in your notebook and provide basic graphics to show results and the notebook URL can be shared among collaborators. Using Zeppelin, a real-time data dashboard has been created which will retrieve data from the Cassandra database and displays it in charts and tables and many others. This dashboard is refreshing data in every second. Dashboard can be shared with an authorized person, who could be a physician, doctor, a participating health firm or an external consultant to allow them to look at the collected data regardless of their patient and health status.

Experiments

Experiment setup

The real-time health status prediction system based on Spark, Kafka and Cassandra was written using Scala and Zeppelin as a development platform which support many interpreters like Scala, Spark and Cassandra. Indeed, we don't need to create an assembly package containing the code and its dependencies. Firstly, the proposed application is carried out on single node cluster created with core i7 processor and 8 GB RAM, having Ubuntu 16.04 operating system through Spark platform which integrates DT model with Kafka streaming data handling. The application after establishing connection to the Kafka streaming as detailed in Fig. 2, is continuously receiving the data streams from multiple Kafka producers and once it encounters the health attributes check streams, it extracts the attribute values from each topic of disease events sent by Kafka streaming and apply the DT model to predict the health status. On the other hand, each predicted status is stored in a table through Cassandra database, based on the identifier (ID) as a primary key which is more suitable for the data redundancy. Data stored in database will be queried later for historical data analysis.

After testing the application on single node cluster, a multi-node cluster was created. Table 3 shows the characteristics of our master and worker nodes.

Table 3 Cluster nodes characteristics

Parameter	Master	Worker
Processor	Core i7	Core i3
Cores	4	4
Memory	8 GB	4 GB
Operating system	Ubuntu 16.04	Ubuntu 16.04

In this step, all experiments were conducted with the available computing resources on a cluster of one master node and two nodes acting as workers. VMware virtual nodes are used in the Ubuntu 16.04 operating system. Firstly, a group called Spark and Spark user account has been created to simplify the communication between nodes. Java and Scala have been installed. We install Open SSH Server, generate key pairs and configure passwordless ssh between the nodes such that Spark master can connect, start, stop and execute jobs in different workers. We have unpacked and installed Spark, Kafka and Cassandra in single node. Two topics and tables have been created, one for heart disease and one other for diabetes disease. We edit .bashrc file located in user's home directory and add environment variables such as JAVA_HOME and SPARK_HOME. Add file slaves in $SPARK_HOME/conf which must include hostname of workers. To have the same copy of different frameworks, we copy the single node cluster setup folder three times, rename one as master and other as worker1 and worker2. Change the hostname and hosts on all nodes. Algorithm 3 represents the steps to setup the cluster.

Algorithm 3 The steps to setup the cluster

```
Step1: Create a group called spark, an user called spuser and add the spuser to sudoers list
    addgroup spark
    adduser –ingroup spark spuser
    visudo
    ALL= (ALL) ALL
Step2: Install ssh server and change permissions and disable IPV6
    apt-get install openssh-server
    ssh-keygen
    cat home/.ssh/id_rsa.pub ≫ home/.ssh/authorized_keys
    chmod 700 home/.ssh/authorized_keys
Step3: Install Java, Scala, Spark, Kafka, Cassandra and Zeppelin
    Download zip file of all installation
    Unpacked and move all these zip files to /usr/local/
    Change permissions of all files to have all permissions for spuser
Step4: Update home/.bashrc file
    Add all necessary environment variables
Step5: Setup multinode cluster
    Copy the single node cluster 3 times
    Rename one Master and other as worker1 and worker2
    Update hostname and hosts of all 3 nodes
Step6: Starting the cluster
    Open terminal (Ctr+Alt+T)
    Cd home/usr/local/kafka/bin
    Start Zookeeper
    Start Kafka
    Create Kafka topic
    Cd home/usr/local/cassandra/bin
    Start Cassandra
    Create a keyspace and table
    Cd  /usr/local/zeppelin/bin
    Start Zeppelin
```

Algorithm 4 describes the main steps to implement our Spark application in Zeppelin notebook.

Algorithm 4 Processing steps of Spark application

```
Step 1 : Spark context
    Create an instance of SparkContext (sc by default in Zeppelin notebook) and StreamingContext
to use all Spark streaming features
Step 2: Get Kafka streams
    Create the direct stream with the Kafka parameters and topic using createDirectStream method
of KafkaUtils
Step 3: Data processing
    Extract identifier and attributes from each stream and from each topic using foreachRDD method
    Apply the saved machine learning model to predict health status
    Save all attributes and predicted label to Cassandra keyspace and table using saveToCassandra
method
Step 4: Start the computation
    Start Spark streaming context using start method
```

We conducted three scenarios with a stream interval of 1 , 2 and 3 s. Our conducted experiments is shown in Table 4.

Results and discussion

Performnace evaluation of machine learning model

The two datasets have been randomly split into a training data set and a test data set, 70% of the data is used to train the model, and 30% will be used for testing. DT has been trained over this data. For large distributed datasets, sorting feature values is expensive, in this implementation, an approximate set of split candidates are calculated over a sampled fraction of data and the ordered splits create bins and maxBins parameter specify the maximum number of such bins, so maxDepth parameter specifies the maximum depth of the DT. Using the dataset below with varying parameters maxDepth, maxBins and indices of impurity, different DT models has been tested and the classification accuracy values are calculated in each case. Using the testing dataset and based on the model error analysis which avoids negative effects of both under fitting and over fitting, it has been discovered that the higher accuracy prediction stabilizes as the number of maxBins and maxDepth take the values indicated in Table 5. Figure 8 represents in bar chart the performance of the implementation of DT using Spark.

- Receiver operating characteristic (ROC) curve: is one of the most important and effective metrics of evaluating the quality or performance of diagnostic tests. The ROC curve is supported by MLlib.
- Classification accuracy: This is defined as the ratio of all correct predictions made to overall prediction data. In this work, classification accuracy for the datasets are measured using the equation:

Table 4 Conducted experiments

No of nodes	Events/s	Kafka	Topic partitions	Spark workers	Cassandra
1	~ 550,000	1	1	1	1
2	~ 1,300,000	2	2	2	2

Table 5 Classification results

Dataset	Diabetes	Heart disease
maxBins	250	100
maxDepth	8	6
Sensitivity (%)	85.97	80.00
Specificity (%)	94.38	85.36
ROC curve (%)	90.03	82.3
Accuracy (%)	91.57	82.40

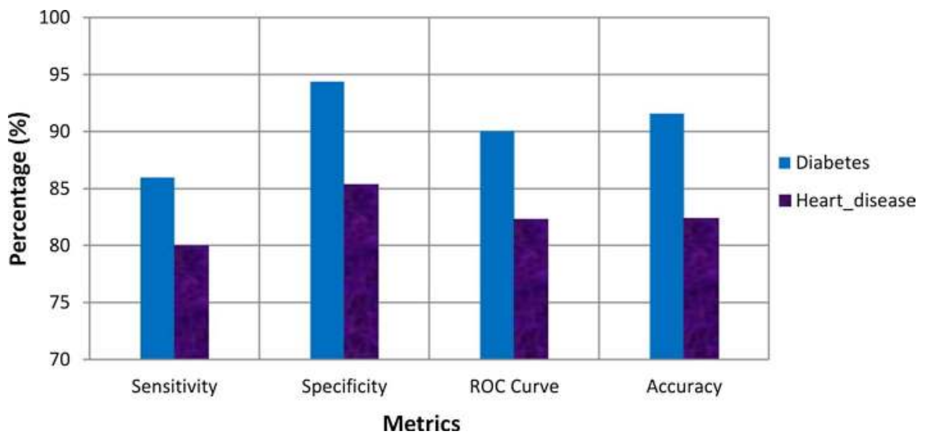


Fig. 8 Machine learning results

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{6}$$

where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively.

Sensitivity measures the proportion of actual positives which are correctly identified and specificity measures the proportion of negatives which are correctly identified. These are formulated by:

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{7}$$

$$\text{Specificity} = \frac{TN}{FP + TN} \tag{8}$$

The effectiveness of our machine learning model was conducted on two synthetic data sets. The empirical results indicate that our implementation of the DT algorithm with Spark is both effective and scalable. From the table above, the proposed model provides stable and high prediction quality.

Apache Spark vs Weka performance

With its capabilities like in-memory computation, Spark performance can be several times faster than other traditional technologies especially in iterative machine learning. In order to show the efficiency of Spark based prediction system in terms of time of training and testing the machine learning model on large data, other data records have been simulated. The simulation is performed using scikit-learn which is a Python library for machine learning that provides functions for generating a suite of test problems.

To test and demonstrate the scalability of our approach, a comparative study between execution time of DT based on Spark and Weka tool was performed. In this context, to have sufficient database size, we increase the number of records. Then, the speed of the DT algorithm using scala and MLlib library was measured and compared with the same algorithm in Weka. Indeed, we have compared the running time of DT in the cluster (Spark) against Weka. Figs. 9 and 10 represent in bar chart the performance comparison of the implementation of DT using Spark and Weka. In this simulation when the number of records is equal to or higher than 3 million, the model training is not supported by using simple C4.5 in Weka.

As we can see from the line and bar chart, running DT model with Spark is faster than Weka tool. It takes only 43.2 s to train the model in case of 4 million records with

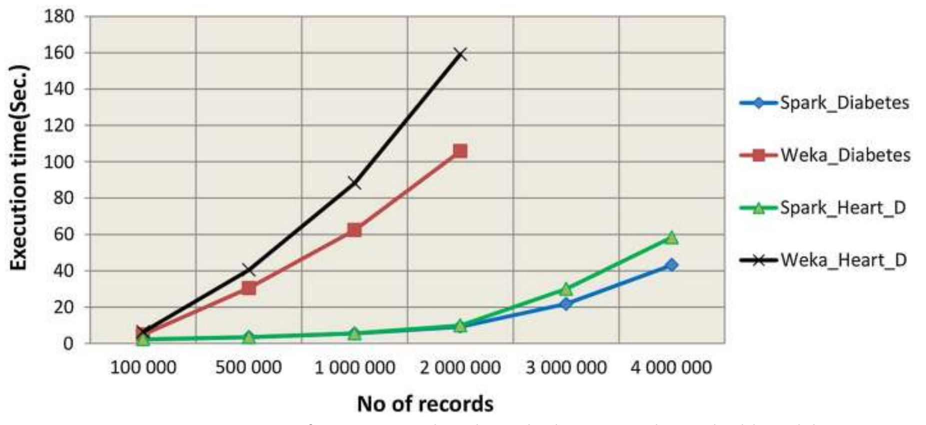


Fig. 9 Execution time comparison of DT using Spark and standard DT: time taken to build model

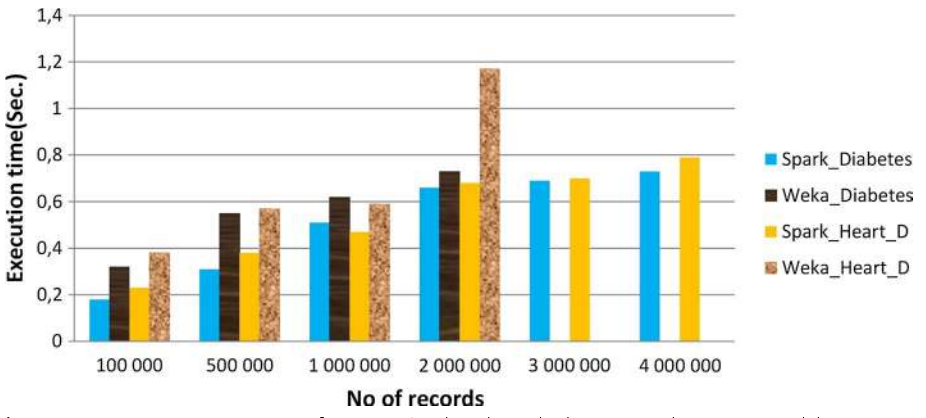


Fig. 10 Execution time comparison of DT using Spark and standard DT : time taken to test model

diabetes dataset when Weka takes 185 s. Also it takes only 58.43 s to train the model in case of 4 million records with heart disease dataset when Weka takes 274.36 s. The proposed Spark based DT takes less time to train and test the machine learning model. The parallel DT algorithm of Spark Mllib reaches best scalability owing to the distributed computing on cluster nodes and in-memory computation. Spark MLlib processes data in less time because it divides the job into several tasks which are executed on workers nodes. After an analysis of the results achieved has been made, it can be concluded that Spark provides the best way to implements the proposed system to predict health status in real-time.

Spark based DT scalability

In this step, performance evaluation of the proposed Spark-based C4.5 algorithm in a distributed parallel environment is performed. Different numbers of nodes and different sizes of training datasets are considered. As mentioned earlier, we have 2 nodes and our training dataset ranges from 100k to 4 million in terms of the number of records.

Figures 11, 12, 13 and 14 illustrate the execution time of our Spark-based C4.5 algorithm with a different number of nodes. When the number of instances is 2, 3 and 4

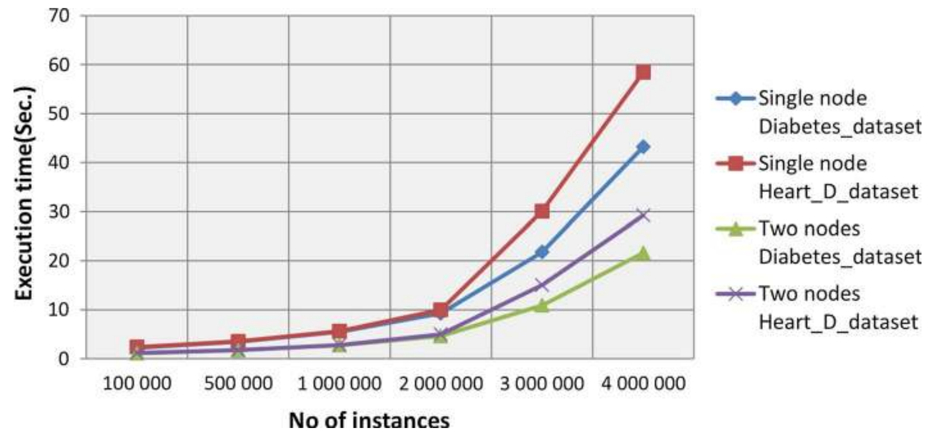


Fig. 11 Performance comparison of DT using Spark for different nodes: time taken to build model

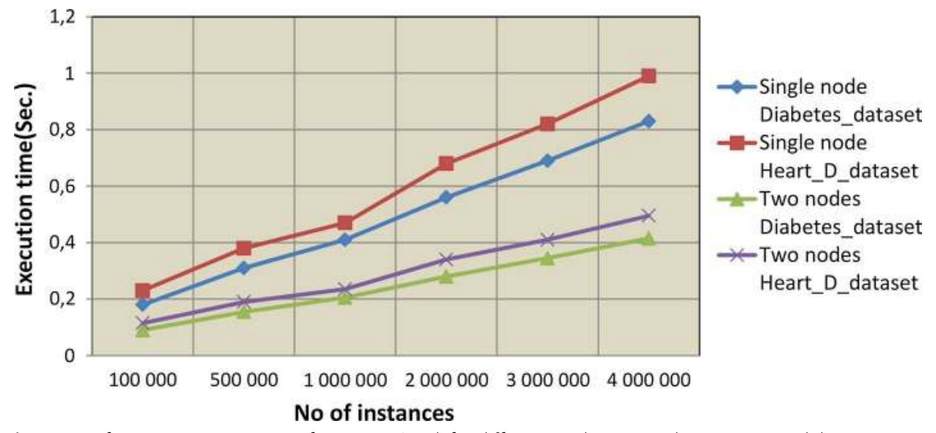


Fig. 12 Performance comparison of DT using Spark for different nodes: time taken to test model

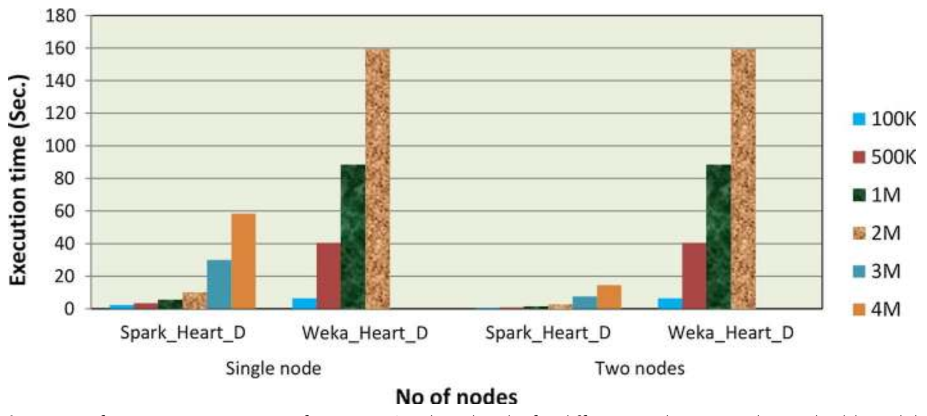


Fig. 13 Performance comparison of DT using Spark and Weka for different nodes: time taken to build model. K: thousand, M: Million

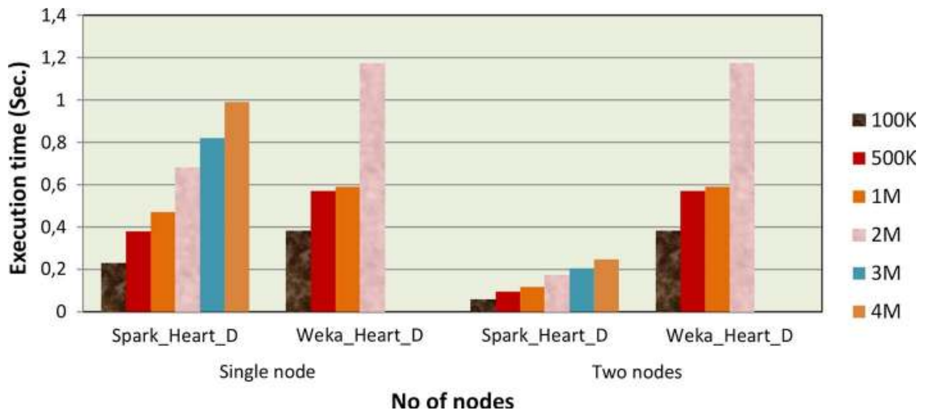


Fig. 14 Performance comparison of DT using Spark and Weka for different nodes: time taken to test model. K: thousand, M: Million

million, respectively, we can see that the total execution time decreases as the number of nodes increases. This indicates that the higher the number of nodes involved in computation, the faster the algorithm will be due to the distributed computing. In contrast, the execution time of standard DT using Weka with two nodes remains stable due to the undistributed computing.

Throughput

Firstly, the DT model was built and tested separately by varying parameters such as impurity, maxDepht and maxBins, the minimum model error is taken into account based on the classification accuracy of the model. An offline model has been created and saved in order to use it in real-time. In our case, data producers consist of two simulator applications, one for heart disease streams and other for diabetes. Each one sends approximately 270,000 events per second per node (it can be more) in predefined format to the specified topic and which in turn serves them to consumers (Spark streaming), Fig. 15.

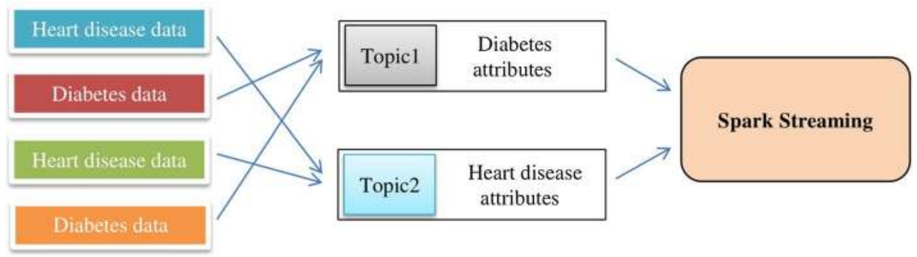


Fig. 15 Simulation topics

For sake of simplicity we are using two producers. All these data events will be captured by Kafka streaming in real-time and are sent to the Spark streaming application, Kafka data streaming module is responsible for managing the events streams. Streaming application consumes data streams and processes them for predicting the health status, series of transformation on Dstreams is performed using Spark streaming API. For each instance, the generated identifier and attribute values were extracted and applies machine learning model on extracted health attributes. The details of each instance were persisted in Cassandra database table for querying them later. We measured the processing time for all tasks using the Spark monitoring API. Figure 16 represents the performance evaluation of the proposed system. We see that the system can process roughly 550 k records per second/node, approximately 30 MB per second/node.

As we can see from Fig 16:

- the larger the throughput, the more cost of processing time.
- the more nodes we use, the less of processing time.
- if enough nodes are leveraged, even the size of throughput is big, the performance can be near to the optimal one. For example, given a throughput of 2.5 million records, if we use 2 nodes, the execution time is near to 2 s. While the execution time

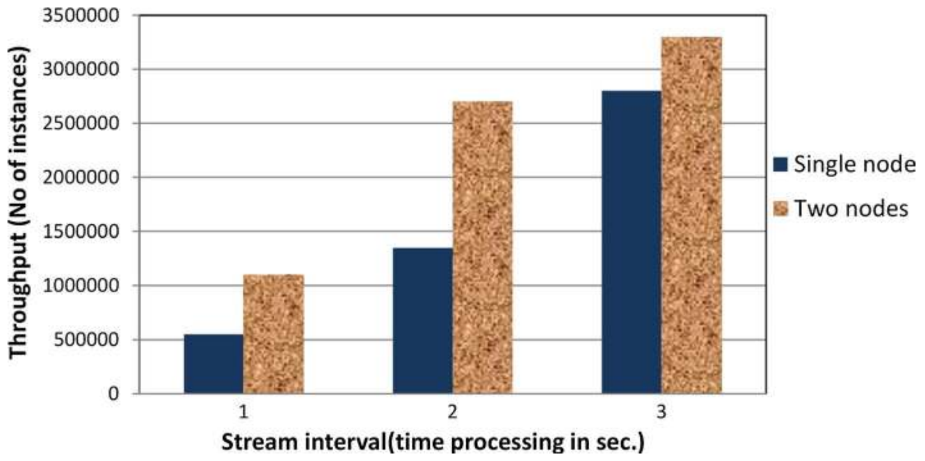


Fig. 16 Performance evaluation: processing time

is near to 3 s when we use a single node. The execution time of the same throughput decrease by adding other nodes.
- Spark streaming with distributed machine learning is a good choice to deal with real-time problems. Namely, by leveraging more nodes, we can resolve big data problems especially those related to real-time prediction in healthcare field.

Using Apache Zeppelin a data dashboard has been created which will retrieve data from the Cassandra database and displays it in charts and tables. This application uses Angularjs and Spark SQL to push the data to the web page in fixed intervals, so data will be refreshed automatically. It can be accessible on desktop as well as mobile devices. The database can be queried by different queries like, number of instance, number of positive and negative cases, some statistics, consult the state of a patient by his identifier in order to extract the critical and useful information or to understand the user behavioral patterns by the practitioners.

Our study focuses on application of machine learning model on streaming big data coming from various source of diseases, with Kafka serving for managing the event streams and transforming healthcare data into information. Through discussing the result of our experiment, it is concluded that Spark is especially fit for iterative algorithms that require multiple passes on data. It provides a faster execution engine for distributed and streaming processing. The usage of Spark in this system improves the data processing speed time more than other traditional tool of data mining.

The main difference between proposed system and traditional data analytics approaches is that the traditional methods analyze one instance at time and it depends of imported data volume. On the other hand, our system can process thousands of instances coming each second in real-time based on Dstream which is a collection of RDD and each RDD represents one or more instances. Also, the system supports big data processing, uses real-time and distributed machine learning, handles incoming streams using Spark streaming instead of MapReduce, predicts different type of diseases at the same time based on the concept of Kafka topic, provide high speed in real-time classification which becomes more meaningful since the volume of generated data from devices, cloud and many others is increasing at an observable rate.

Conclusion

The amount of healthcare data is constantly growing time by time at alarming rate in different and inconsistent data sources. Improving the patient outcome and making scalable real-time health status prediction system, streaming computing platform is needed. However, this vertiginous volume of data can no longer be collected, processed, stored, and exploited by traditional information technology solutions combining physical infrastructures and relational databases. Based on the challenges discussed previously, many flaws in traditional information technology in scaling with the hardware in parallel which is not suitable in dealing with growing data. In this paper a real-time health status prediction and analytics system is proposed and tested on cluster, it was built around open source big data technologies. The data events coming from various diseases ingested to Spark through Kafka streaming. Using the Spark streaming API, the system process received relevant health data events by applying DT to predict health status, send an alert to care providers and store the details in distributed database. The stored result will be queried to perform healthcare data analytics and stream reporting.

The creation of a distributed and real-time healthcare analytics system using traditional analytical tools is extremely complex, it requires a variety of skills, intensive and more expensive programs and considerable amount of time and money. However, using efficient open source big data technologies and data mining techniques can easily do the same job. With slight modification, the same system can predict others diseases, also it can be extended to other domain. As a future work, we aim to integrate real data sources, such as mobile devices, sensor data and social media data to our system for interacting user's requests.

Abbreviations

IoT: Internet of Thing; DT: decision tree; RDBMS: relational database management system; HDFS: Hadoop Distributed File System; BDSC: big data streaming computing; MLlib: machine learning library; RDD: Resilient Distributed Datasets; HD: heart disease; IT: information technology.

Acknowledgements

Not applicable.

Authors' contributions

AE proposed and developed the real-time prediction system presented in this article, prepared and analyzed the data, interpreted the results, and authored the manuscript. Comments, guidance, significant discussion that highly improved the quality of manuscript were received from KM. All mentioned authors contribute to the elaboration of the article. Both authors read and approved the final manuscript.

Funding

None.

Availability of data and materials

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Received: 12 July 2019 Accepted: 18 November 2019
Published online: 27 November 2019

References

1. Manogaran G, Lopez D. Health data analytics using scalable logistic regression with stochastic gradient descent. Int J Adv Intell Paradigms. 2018;10(1–2):118–32.
2. Hu H, Wen Y, Chua T-S, Li X. Toward scalable systems for big data analytics: a technology tutorial. IEEE Access. 2014;2:652–87.
3. Cattell R. Scalable sql and NoSQL data stores. ACM Sigmod Record. 2011;39(4):12–27.
4. Moniruzzaman A, Hossain SA. NoSQL database: New era of databases for big data analytics-classification, characteristics and comparison. 2013. arXiv preprint arXiv:1307.0191.
5. Dean J, Ghemawat S. Mapreduce: simplified data processing on large clusters. Commun ACM. 2008;51(1):107–13.
6. Belle A, Thiagarajan R, Soroushmehr S, Navidi F, Beard DA, Najarian K. Big data analytics in healthcare. BioMed Res Int. 2015; 2015.
7. Anuradha J, et al. A brief introduction on big data 5vs characteristics and hadoop technology. Procedia Comput Sci. 2015;48:319–24.
8. Banaee H, Ahmed MU, Loutfi A. Data mining for wearable sensors in health monitoring systems: a review of recent trends and challenges. Sensors. 2013;13(12):17472–500.
9. Mathew PS, Pillai AS. Big data challenges and solutions in healthcare: a survey. In: Snášel V, Abraham A, Krömer P, Pant M, Muda A, editors. Innovations in bio-inspired computing and applications. Berlin: Springer; 2016. p. 543–53.
10. Sun J, Reddy CK. Big data analytics for healthcare. In: Proceedings of the 19th ACM SIGKDD International Discovery and Data Mining. New York: ACM; 2013. p. 1525–1525.
11. Masethe HD, Masethe MA. Prediction of heart disease using classification algorithms. Proc World Congress Eng Comput Sci. 2014;2:22–4.
12. Bhardwaj A, Tiwari A. Breast cancer diagnosis using genetically optimized neural network model. Expert Syst Appl. 2015;42(10):4611–20.
13. Tomar D, Agarwal S. A survey on data mining approaches for healthcare. Int J Bio-Sci Bio-Technol. 2013;5(5):241–66.

14. Herland M, Khoshgoftaar TM, Wald R. A review of data mining using big data in health informatics. J Big Data. 2014;1(1):2.
15. Rallapalli S, Gondkar R, Rao GVM. Cloud based k-means clustering running as a Mapreduce job for big data healthcare analytics using Apache mahout. In: Satapathy S, Mandal J, S Udgata, Bhateja V, editors. Information systems design and intelligent applications. Berlin: Springer; 2016. p. 127–35.
16. Sarkar BB, Paul S, Cornel B, Rohatinovici N, Chaki N. Personal health record management system using Hadoop framework: An application for smarter health care. In: International Workshop Soft Computing Applications. Berlin: Springer; 2016. p. 385–93.
17. Sampath P, Tamilselvi S, Kumar NS, Lavanya S, Eswari T. Diabetic data analysis in healthcare using Hadoop architecture over big data. Int J Biomed Eng Technol. 2017;23(2–4):137–47.
18. Rathore MM, Paul A, Ahmad A, Anisetti M, Jeon G. Hadoop-based intelligent care system (HICS): analytical approach for big data in IoT. ACM Trans Internet Technol (TOIT). 2017;18(1):8.
19. Basco JA, Senthilkumar N. Real-time analysis of healthcare using big data analytics. Comput Inf Technol. 2017;263:042056.
20. Yadranjiaghdam B, Pool N, Tabrizi N. A survey on real-time big data analytics: Applications and tools. In: 2016 international conference On computational science and computational intelligence (CSCI). New York: IEEE; 2016. p. 404–9.
21. Hazarika AV, Ram GJSR, Jain E. Performance comparison of hadoop and spark engine. In: 2017 international conference on I-SMAC (IoT in social, mobile, analytics and cloud)(I-SMAC). New York: IEEE; 2017. p. 671–4.
22. Rallapalli S, Suryakanthi T. Predicting the risk of diabetes in big data electronic health records by using scalable random forest classification algorithm. In: 2016 international conference on advances in computing and communication engineering (ICACCE). New York: IEEE; 2016. p. 281–4.
23. Feroz MN, Mengel S. Examination of data, rule generation and detection of phishing urls using online logistic regression. In: 2014 IEEE international conference on big data (Big Data). New York: IEEE; 2014. p. 241–50.
24. Zhao T, Ni H, Zhou X, Qiang L, Zhang D, Yu Z. Detecting abnormal patterns of daily activities for the elderly living alone. In: International conference on health information science. Berlin: Springer; 2014. p. 95–108.
25. Rathore MM, Ahmad A, Paul A, Wan J, Zhang D. Real-time medical emergency response system: exploiting IoT and big data for public health. J Med Syst. 2016;40(12):283.
26. Manogaran G, Lopez D. A survey of big data architectures and machine learning algorithms in healthcare. Int J Biomed Eng Technol. 2017;25(2–4):182–211.
27. Lee K, Agrawal A, Choudhary A. Real-time disease surveillance using twitter data: demonstration on flu and cancer. In: Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining. New York: ACM; 2013. p. 1474–7.
28. Akhtar U, Khattak AM, Lee S. Challenges in managing real-time data in health information system (HIS). In: International conference on smart homes and health telematics. Berlin: Springer; 2016. p. 305–13.
29. Ed-daoudy A, Maalmi K. Application of machine learning model on streaming health data event in real-time to predict health status using spark. In: 2018 International symposium on advanced electrical and communication technologies (ISAECT). New York: IEEE; 2018. p. 1–4.
30. Ed-daoudy A, Maalmi K. Real-time machine learning for early detection of heart disease using big data approach. In: 2019 International conference on wireless technologies, embedded and intelligent systems (WITS). New York: IEEE; 2019. p. 1–5.
31. Bauer H, Patel M, Veira J. The Internet of Things: sizing up the opportunity. http://www.mckinsey.com/. Accessed 15 Dec 2017.
32. Apache kafka. https://kafka.apache.org. Accessed 15 Dec 2017.
33. Hunt P, Konar M, Junqueira FP, Reed B. Zookeeper: Wait-free coordination for internet-scale systems. In: USENIX Annual technical conference, vol. 8. Boston, MA, USA; 2010.
34. Apache Spark. https://spark.apache.org. Accessed 15 Dec 2017.
35. Zaharia M, Chowdhury M, Das T, Dave A, Ma J, McCauley M, Franklin MJ, Shenker S, Stoica I. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In: Proceedings of the 9th USENIX conference on networked systems design and implementation. Berkeley: USENIX Association; 2012. p. 2.
36. kaggle. https://www.kaggle.com/fmendes/diabetes-from-dat263x-lab01. Accessed 24 Dec 2018.
37. Quinlan JR. C4. 5: programs for machine learning. Amsterdam: Elsevier; 2014.
38. Apache cassandra. http://cassandra.apache.org. Accessed 15 Dec 2017.
39. Hassan M, Bansal SK. Semantic data querying over NoSQL databases with Apache Spark. In: 2018 IEEE international conference on information reuse and integration (IRI). New York: IEEE; 2018. p. 364–71.
40. Apache zeppelin. https://zeppelin.apache.org. Accessed 15 Dec 2017.