Health data analytics using scalable logistic regression with stochastic gradient descent

Gunasekaran Manogaran* and Daphne Lopez

School of Information Technology and Engineering,
VIT University,
Vellore, India
Email: gunavit@gmail.com
Email: daphnelopez@vit.ac.in
*Corresponding author

Abstract: As wearable medical sensors continuously generate enormous data, it is difficult to process and analyse. This paper focuses on developing scalable sensor data processing architecture in cloud computing to store and process body sensor data for healthcare applications. Proposed architecture uses big data technologies such as Apache Flume, Apache Pig and Apache HBase to collect and store huge sensor data in the Amazon web service. Apache Mahout implementation of MapReduce-based online stochastic gradient descent algorithm is used in the logistic regression to develop the scalable diagnosis model. Cleveland heart disease database (CHDD) is used to train the logistic regression model. Wearable body sensors are used to get the blood pressure, blood sugar level and heart rate of the patient to predict the heart disease status. Proposed prediction model efficiently classifies the heart disease with the accuracy of training and validation sample is 81.99% and 81.52%, respectively.

Keywords: stochastic gradient descent; SGD; mapreduce logistic regression; wearable body sensor; sensor data; big data; heart disease; cloud computing; Cleveland heart disease database; CHDD; Amazon web service; AWS; scalable diagnosis model.

Reference to this paper should be made as follows: Manogaran, G. and Lopez, D. (2018) 'Health data analytics using scalable logistic regression with stochastic gradient descent', *Int. J. Advanced Intelligence Paradigms*, Vol. 10, Nos. 1/2, pp.118–132.

Biographical notes: Gunasekaran Manogaran is currently pursuing PhD in the School of Information Technology and Engineering, Vellore Institute of Technology University. He received his BE and MTech from Anna University and Vellore Institute of Technology University, respectively. He has worked as a research assistant for a project on spatial data mining funded by Indian Council of Medical Research, Government of India. His current research interests include data mining, big data analytics and soft computing. He is the author/co-author of papers in conferences, book chapters and journals. He got an award for being young investigator from India and Southeast Asia by Bill and Melinda Gates Foundation. He is a member of International Society for Infectious Diseases.

Daphne Lopez is a Professor in the School of Information Technology and Engineering, Vellore Institute of Technology University. Her research spans the fields of grid and cloud computing, spatial and temporal data mining and big data. She has a vast experience in teaching and industry. She is the

author/co-author of papers in conferences, book chapters and journals. She serves as a reviewer in journals and conference proceedings. She has worked in the software industry as a consultant in data warehouse and business intelligence. She is a member of International Society for Infectious Diseases.

1 Introduction

In ubiquitous computing environment sensors are playing a vital role in almost all applications such as environmental monitoring, transport, smart city development and healthcare and so on. Especially, wearable medical devices with sensors are essential for gathering of rich information about physical and mental health. More commonly, sensors are used for capturing and reporting some measures of the environment in which they are installed, such as the pressure, radiation, humidity, temperature, or gas levels. Generally, these measurements are stored in some database to process and find the useful information. However in personal healthcare system where sensors are installed and is produced data continuously, the amount of data to be stored and processed becomes a significant problem in real life. Relational database management system (RDBMS) is generally used to store the traditional data, but day by day the volume, velocity and variety of sensor data is growing towards Exabyte (Thilagavathi et al., 2014; Victor et al., 2016). This requires advanced tools and techniques to store, process and display such large amount of sensor data to the end users. Thus, storing and querying large amount of data requires database clusters and additional resources. However, storage and retrieval are not the only problem but also extract useful information from big data such as diagnostic information. In recent years more number of demanding applications is being developed for different environments. Sensors will play an essential role in critical applications for real or near future (Lopez and Sekaran, 2016).

For example, a number of wearable sensors and devices are developed for continuous monitoring of personal fitness, healthcare, and physical activity awareness (Jawbone Inc, 2016; FitBit Inc, 2016). Nowadays, researchers are interested to develop wearable clinical devices in remote health monitoring systems for continuous storage, management and clinical access to the patient's physiological information (Pantelopoulos and Bourbakis, 2010; Paradiso et al., 2005). Wearable clinical devices can give physical routine by a two- to three-day period of continuous physiological monitoring of patient. During this period sensors would continuously store the patient's physiological data to the database linked with your device. Doctors can give better results by not only using clinical test, but also patient's physiological data collected from sensors. Thus, sensor data is useful to take correct action for patient's health and treatment recommendation, early diagnosis, and life-style choices that are essential in improving the quality of human health (Chen et al., 2010; Latré et al., 2010). Traditional data storage platforms fail to perform well for above mentioned emerging sensor application domains where the volume and velocity of the data grow in enormous rates. In order to overcome this issue it is necessary to develop big data technologies and architectures for storing and processing such huge size of data. This paper discusses the architecture and implementation of scalable big data architecture for processing real time sensor data using cloud computing technologies.

2 Related work

Nowadays many health monitoring frameworks follow a three tier architecture that includes the following layers – data communication and networking, acquisition unit and the service layer (Pantelopoulos and Bourbakis, 2010). Fabric sensing elements also play an important role in capturing biomedical signals such as electrocardiogram, respiration, activity (Paradiso et al., 2005). Open geo-spatial consortium (OGC)-based health monitoring system is also identified to measure various physiological parameters such as blood pressure, body temperature, heart rate and respiration rate (Babu et al., 2013). Furthermore, sensors are fixed with tele-health monitoring system to transfer the vital signs of the patient through Bluetooth or Zigbee wireless networks (Corchado et al., 2010). Similar to above tele-health monitoring, Fengou et al. (2013) have proposed e-Health telemonitoring system architecture that consists of three blocks such as data collection, data management and data interpretation. Furthermore, it is important to maintain the communication between the devices in the e-health system. For example, Chehri et al. (2010) discussed about the communication between the body sensor hetwork (BSN), Zigbee, smart house and medical call centre in e-health system. They used UMTS, WiMax, and the internet to communicate between the devices in the e-health system. Table 1 depicts commonly used wearable sensors in human body (Lai et al., 2013).

Table 1 Commonly used wearable sensors in human body

S. no.	*Name of the sensor*	*Sensor use*	*Sensor measurement type*	*Sensor placement*
1	Accelerometer	Measuring the human energy expenditure	Continuous	Wearable
2	Carbon dioxide sensor	Measuring the carbon dioxide level from mixed gas	Discrete	Wearable
3	ECG/EEG/EMG sensor	Measuring the electrocardiograph signal	Continuous	Wearable
4	Gyroscope	Measuring the angular velocity with respect to the body axis	Continuous	Wearable
5	Humidity sensor	Measuring the sweating rate	Discrete	Wearable
6	Blood oxygen saturation sensor	Measuring the percentage of oxygen saturation in blood	Discrete	Wearable
7	Pressure sensor	Measuring the pressure changes of the underside of foot	Continuous	Wearable/ surrounding
8	Respiration sensor	Measuring the rate of breathing	Continuous	Wearable
9	Temperature sensor	Measuring the rate of body temperature	Discrete	Wearable
10	Visual sensor	Capturing the motion, length, location, and area.	Continuous/ discrete	Wearable/ surrounding

Table 1 Commonly used wearable sensors in human body (continued)

S. no.	*Name of the sensor*	*Sensor use*	*Sensor measurement type*	*Sensor placement*
11	Blood pressure sensor	Measuring the systolic and diastolic pressure	Continuous	Wearable
12	Heart rate sensor	Measuring the heart rate	Continuous	Wearable
13	Blood sugar sensor	Sensors record glucose levels continuously around the clock	Continuous	Wearable

The above mentioned sensor-based health monitoring systems continuously generate data and so it is difficult to process such huge size of data. This type of streaming sensor data is also called as big data. Traditional data processing frameworks and tools are not suitable for processing the big data. Nowadays healthcare organisations also store the clinical data continuously, which will rapidly increase records size that are available electronically. Simultaneously, noticeable growth has been made in clinical analytics. For example, new methodologies and processing platforms are identified in the healthcare. As a result, there are a number of techniques have been identified to use big data to reduce the costs of healthcare. For example, six use cases of big data are identified in a recent research to reduce the cost of patients. It includes readmissions, triage, decompensation, adverse events, and treatment optimisation (Bates et al., 2014). Furthermore, four big data use cases have been identified in healthcare that includes clinical decision, consumer behaviour, and support services (Hermon and Williams, 2014). In addition, recent studies describe how to reshape the healthcare system based on big data analytics to improve the patient treatment, disease control and diagnosis (Jee and Kim, 2013; Lopez et al., 2014; Lopez and Gunasekaran, 2015). Similarly a number of big data technologies have been developed to improve the healthcare system. The most familiar healthcare services include medical health record (MHR), personal health record (PHR), and electronic health record (EHR). EHR is the electronic record that stores history of the patient's medical information, accessible and managed by healthcare providers. Thus, healthcare providers use cloud technologies to store these EHR that enables users from anywhere in the world to use the data. Amazon S3 is used in the proposed architecture to store the EHR and made available online. In this proposed architecture sensors are fixed in the patient body that integrates with sensor server to send continuous data. Apache Flume is installed in the sensor servers to communicate and store the sensor data into Amazon S3. Finally, both sensor data and EHR are available in Amazon S3 that will be transferred into Apache HBase with the help of Apache Pig. Apache Mahout (machine learning algorithms) is used to process both EHR and sensor data to obtain the prediction model. In order to develop the efficient and scalable prediction model, stochastic gradient descent (SGD) (online learning) algorithm is used and the results are evaluated with real data.

3 Proposed framework

Electronic medical records are computerised medical information systems that collect, store and display patient information. High availability of healthcare information has led to increase the accuracy and overall quality of healthcare delivery. Nowadays size and structure of healthcare data is increasing dramatically. Hence, RDBMS is not suitable for storing such huge size of data. Researchers develop number of big data technlogies to solve such isues. NoSQL databases have significant advantages such as auto scaling, better performance and high availability which address the limitations of relational databases in distributed healthcare systems. Scalable sensor data processing architecture is proposed in this paper to store and process body sensor data for healthcare applications (Figure 1). In this proposed architecture electronic medical records are collected through clinical test and the results are stored into a cloud storage (Amazon S3). MapReduce implementation of online SGD algorithm is used in the logistic regression to develop the prediction model. Prior electronic medical records is used to train the logistic regression model. After completion of training process, the prediction model will use the current sensor data (blood pressure, blood sugar level and heart rate) of the patient to predict the heart disease status. Analytics workflow and sensor data flow of the proposed health monitoring system are depicted in Figure 2 and Figure 3 respectively.

Figure 1 Scalable sensor data processing architecture in the cloud

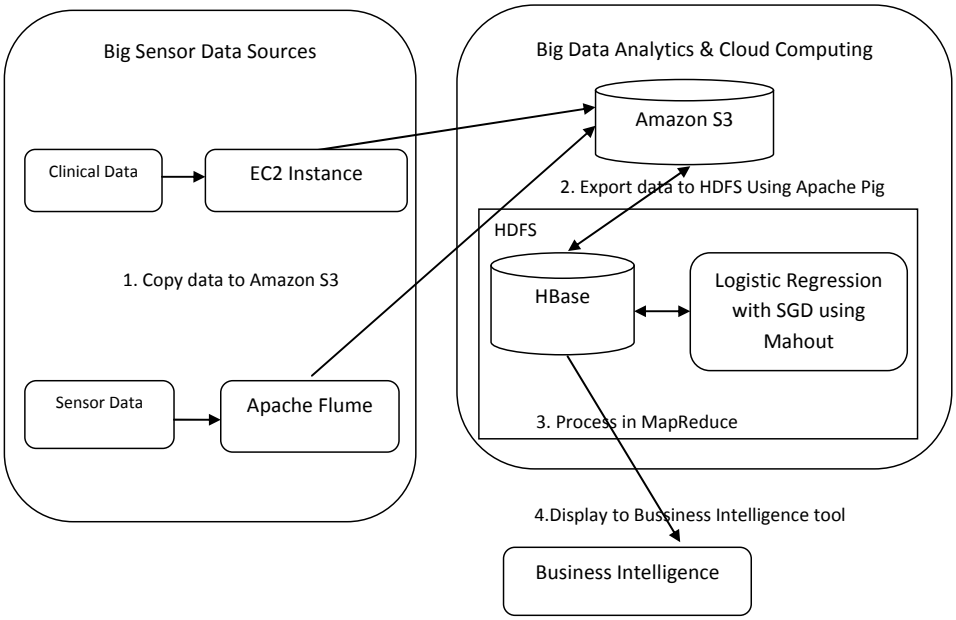


Figure 2 Workflow of the proposed system

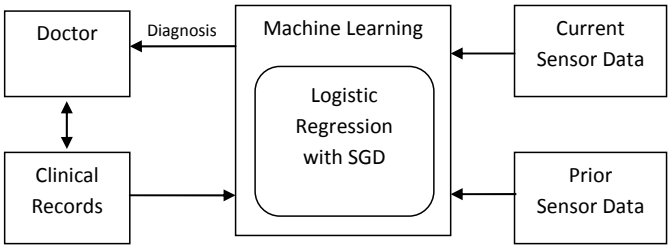
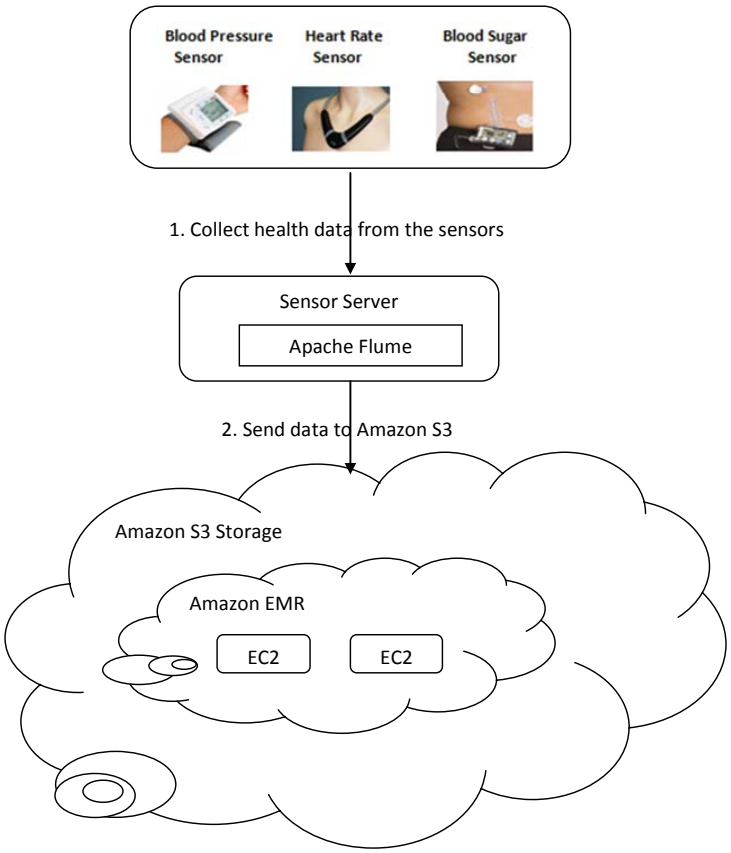


Figure 3 Sensor data flow of the proposed system (see online version for colours)

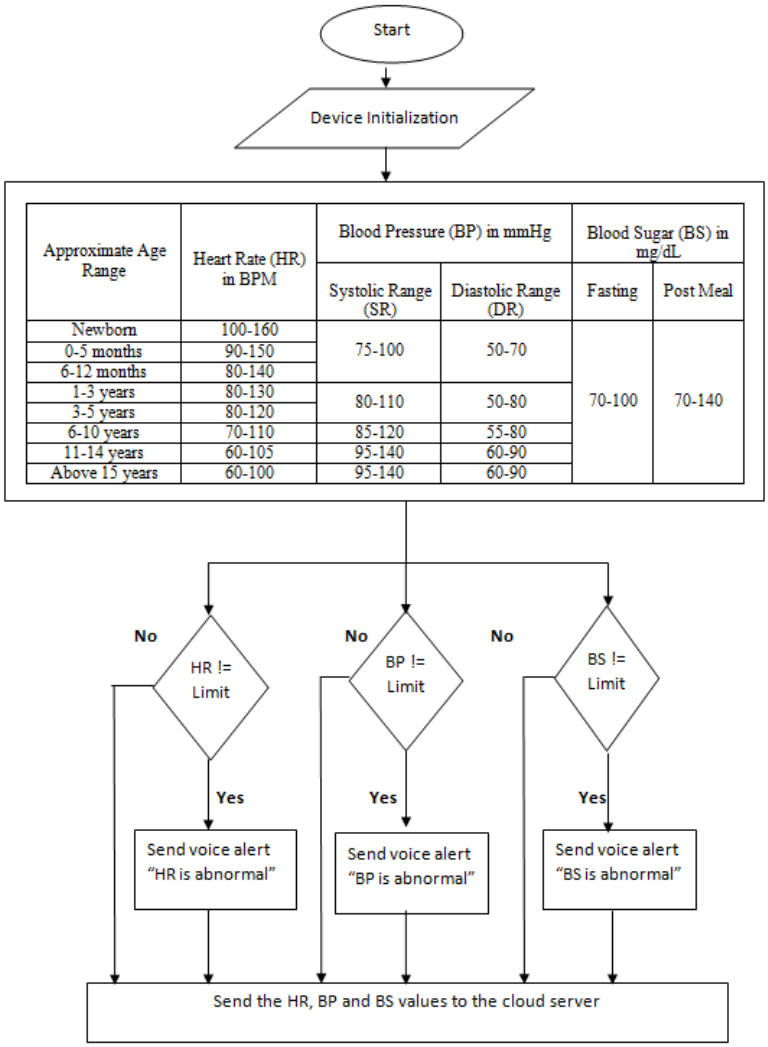


3.1 Wearable sensor data collection unit

The main challenge in Hadoop framework is moving data from the source to the final collection point. Especially, in cloud computing environment where devices and applications generate large volume of data in a short period of time, collecting data in a

scalable manner has an essential role in big-data architecture. Generally, data generated by various parts of architecture is stored in the form of text files and transferred to the destination by tools such as FTP, SSH, RYSNC, or other tools. Nowadays programmers write their own data collection tools with Ruby, PHP and so on. AWS users can use the frameworks that are already available to provide scalable and distributed data collection. This proposed architecture uses Apache Flume to transfer body sensor data to AWS.

Figure 4 Flow chart of the personal health monitoring system



3.1.1 Apache flume for moving body sensor data to Amazon simple storage service (S3)

Apache Flume (2016) is reliable and distributed software for collecting, aggregating, and moving huge size of data. Streaming data flows are used in the Apache Flume framework

to process the real time streaming data. Flume is robust and fault tolerant for data collection task. Apache Flume consists of three major components such as agents, collectors, and the master. Agents are used to collect the data such as API logs, web logs, syslogs, or periodically published data. Users can install Flume agents on the data sources to transfer data to the destination. Collectors are responsible for collecting the data from the agents and store them into Hadoop Distributed File System (HDFS) or Amazon S3. Flume follows simple and hierarchical data collection model that enables users to set up more number of agent nodes to collect data, so fewer collectors are needed (Deyhim, 2013). Proposed architecture uses Apache Flume to transfer sensor data to Amazon S3. As shown in Figure 4 whenever the heart rate, blood pressure, body temperature and blood sugar exceed threshold value then the devices would send an alert message with clinical value to the sensor server. Apache Flume is installed in the sensor server to transfer the body sensor measurements to the Amazon S3.

3.2 Big data analytics and cloud computing block

3.2.1 Amazon Elastic MapReduce

Amazon Elastic MapReduce (EMR) is an Amazon web service (AWS) for processing and analysing vast amounts of data. Amazon EMR is based on Hadoop, a Java-based programming framework that enables users to distribute the computational work across a cluster of virtual servers running in the Amazon cloud. Hadoop MapReduce is a sub-project of the Apache Hadoop project for distributed data processing of huge datasets on compute clusters of commodity hardware.

The framework takes care of node failure, task monitoring, and load balancing and scheduling them. The main objective of Map/Reduce is to split the input dataset into independent chunks that can be processed parallely. The primary objective of Hadoop MapReduce framework is to sort the output of the maps, which are then input to the reduce tasks. Typically, both the input and the output of the job are stored in a Hadoop file system. Amazon EMR processes huge data across a Hadoop cluster of virtual servers on the Amazon Elastic Compute Cloud (EC2). At any given time depending on the demand, Amazon EMR can ramp up or reduce the resource usage, so it is called as Elastic MapReduce. Amazon EMR is used for many applications such as log analysis, web indexing, financial analysis, data warehousing, scientific simulation, machine learning and so on.

Amazon EMR has enhanced number of features to work with Hadoop. It includes Amazon S3 for bulk storage of input and output data, CloudWatch to monitor cluster performance, and move data into and out of DynamoDB using Amazon EMR and Hive. Hadoop based open-source projects such as Hive, Pig, HBase, DistCp, and Ganglia also run on Amazon EMR. The following benefits of running Hadoop on Amazon EMR include the ability to scale the number of virtual servers based on the computation, only pay for what you use and integration with other AWS services.

As shown in Figure 2, EC2 instances in the Amazon EMR are broken up into master, core, and task clusters to execute the Job. The master group instance is responsible for job flow, maintains JARs and scripts, the master node collect the data from EC2 instances and store into Amazon S3. In addition, the master node also monitors the health and status of the core and task instances. Core group instances perform the map and reduce functions over the data and store intermediate results into the HDFS storage of Amazon

EMR cluster. Proposed architecture uses HBase to store the intermediate results of the `map` and reduce functions. Task group instances are optional for EMR and it does not have HDFS storage of the data and intermediate results. Hence, master nodes transfer the data to task group instances to do the work in the job flow. A loss of master or core group instance can cause failure of job and needs to be restarted (Schmidt, 2013).

3.2.2 Moving clinical data to Amazon simple storage service (S3)

Healthcare organisation uses EC2 instance in Amazon's cloud to store clinical records of the patients. However, this clinical dataset is not available in a location where it can be used in Amazon EMR, because clinical data stores only on the local disk of a running EC2 instance. In order to use clinical dataset in the cloud, proposed architecture moves the data into Amazon S3, where Amazon EMR can access it. Generally, Amazon EMR processes the data which is stored in Amazon S3 location or HDFS storage in the Amazon EMR cluster. It is necessary to have a globally unique name for Amazon S3 bucket to store the data. Each bucket has a unique URL naming constraint to interact with Amazon S3. As number of methods are available for loading data into Amazon S3. Proposed architecture uses 's3cmd utility' method to move data into Amazon S3. S3cmd is responsible for uploading, retrieving and managing data in cloud storage service providers such as Amazon S3 Google Cloud Storage or DreamHost DreamObjects. It is a command line program suitable for batch scripts and automated backup to S3 (Schmidt, 2013).

Pseudo code for load data into S3 bucket

```
system_name $ s3cmd mb s3://clinical-emr
Bucket 's3:// clinical-emr/' created
system_name $ s3cmd put clinical_ 1/3/2016.csv. s3://clinical-emr
sample-syslog.log -> s3:// clinical-emr/ clinical_1/3/2016.csv
```

The above pseudo code uses 'mb' option to create a new bucket called 'clinical-emr' and 's3cmd' put is used to move the clinical data clinical_1/3/2016.csv into the S3 bucket 'clinical-emr' (Manogaran et al., 2016).

3.2.3 Data transfer from Amazon S3 to HBase

Amazon EMR provides different approaches to get big data onto a cluster. The familiar approach is to upload the data to Amazon S3 and transfer into HBase cluster using built-in features of Amazon EMR. Proposed architecture uses Apache Pig to transfer clinical data from Amazon S3 to HBase. Apache HBase (2016) is a scalable and distributed database that can store huge size of rows and columns in distributed manner and it is greatly available to all nodes in the cluster. HBase follows columnar database storage, so it is organised by column families, which are sets of related columns. Apache Pig is a platform for analysing large amount of data by representing them as data flows. PigLatin scripting language is used in the Pig for extract, transform and load (ETL) process and adhoc data anlaysis on structured, semi-structured, or unstructured data. Pig is often used while joining new incremental data with the previous data results (Apache Pig, 2016). Proposed architecture uses Apache Pig to join a new incremental clinical data

with previous clinical data. The algorithm is used to transfer the incremental clinical data (1/3/2016) into HBase.

Pig Algorithm: Day_wise_ClinicalData_Table creation and loading Day_wise_ ClinicalData in to HBase

Data**:** *Day wise ClinicalData parameters collected from Amazon S3*

Input*: Load_Day_wise_ClinicalData_Table class object*

Output*: String object having Pig Load Table query.*

Step 1*:* ***Hadoop distcp s3n****:clinical-emr/clinical_1/3/2016.csv /user/myuser/mydirectory/*

Step 2*:* ***if*** *(Table_Name ≠ NULL)* ***then***

Step 3: ***load*** *'/user/myuser/mydirectory/clinical_1/3/2016.csv'* ***using PigStorage(',') as*** *(**add** field name, field data type),*
store a into *'hbase:// Day_wise_ ClinicalData_Table'*
using org.apache.pig.backend.hadoop.hbase.HBaseStorage
*(**add** column family name:field column family data type),*

Step 4*:* *Return the Pig Load Table query*

3.3 Big data analytics using Apache Mahout

3.3.1 Logistic regression

Logistic regression is a statistical method for analysing a dataset in which the dependent variable is dichotomous (binary). Logistic regression is used to find the relationship between one dependent binary variable and one or more independent variables. Each independent variable is multiplied with weights and summed up. This result will add into sigmoid function to find the result between 0 and 1. The values above 0.5 are considered as 1, and the values below 0.5 are considered as 0. It is important to find the best weights, or regression coefficients. Thus optimisation techniques are used to find the best regression coefficients and weights.

Logistic regression can be defined by

Assume that

$$V = \left\{\left(a_i \in R^d, b_i \in R\right)\right\}_{i=1}^{n}$$

where

$a_i \in R^d$ d dimensional independent variables

$b_i \in R$ dependent variables

A n by d matrix where a_i is the i^{th} row of the matrix A

b varies from –1 to 1.

Let the probability of b becoming 1 is

$$p(b = 1 \mid a, \theta) = \frac{1}{1 + e^{-\theta^T a}}$$

Let the probability of b becoming –1 is

$$p(b = 0 \mid a, \theta) = 1 - p(n = 1 \mid a, \theta)$$

where

$\theta \in R^d$ weights.

In order to reduce error function, the best weight vector θ is to be identified based on the negative log likelihood (Kang et al., 2014)

$$F(\theta) = \sum_{i=1}^{n} -\log\left(p\left(b_i \mid a_i, \theta\right) + \lambda \| \theta \|_1\right)$$

$$= \sum_{i=1}^{n} \log\left(1 + \exp\left(-b_i a_i^T \theta\right) + \lambda \| \theta \|_1\right)$$

where

λ regularisation term to penalise large weight parameters.

3.3.2 Stochastic gradient descent

Two optimisation functions are generally used to find the best regression weights in logistic regression namely gradient descent and SGD. Both algorithms update a set of parameters in an iterative manner to minimise an error function. Gradient descent algorithm train the entire sample in training set to do a single update for a parameter in a particular iteration. SGD uses only one training sample from training set to do the update for a parameter in a particular iteration. Hence, gradient descent algorithm is not suitable for big data analytics. For example, if the data points are billions and each point is containing thousands of features then traditional gradient descent is not suitable. Apache Mahout implementation of SGD algorithm is used in the logistic regression to update the weights. SGD is a type of online learning algorithm and it can incrementally update the classifier as new data comes in rather than all at once.

The standard gradient descent algorithm updates the parameters θ of the objective $F(\theta)$ as,

$$\theta = \theta - \alpha \nabla \theta E\left[F(\theta)\right]$$

where the expectation $E[F(\theta)]$ is defined by evaluating the cost and gradient over the full training set. SGD simply does away with the expectation in the update and computes the gradient of the parameters using only a single or a few training examples. The new update is given by,

$$\theta = \theta - \alpha \nabla F\left(\theta, a(i), b(i)\right)$$

with a pair $(a(i), b(i))$ from the training set.

3.3.3 Mahout implementation of SGD for logistic regression

Apache Mahout is a library of scalable machine-learning algorithms. Apache Mahout is implemented on top of Apache Hadoop and using the MapReduce paradigm. Machine

learning is a type of artificial intelligence focused on enabling machines to learn without being explicitly programmed, and it is commonly used to improve future performance based on previous outcomes. Big data is stored on the HDFS, Apache Mahout (2016) is used to execute machine learning algorithms that extract meaningful patterns from datasets. Mahout implementation of logistic regression using SGD supports the following command lines:

- *Training the model*

  Mahout org.apache.mahout.classifier.sgd.TrainLogistic – passes 1 – rate 1 – lambda 0.5 –input heart.csv – features 21 – output heart.model – target Num – categories 2 – predictors Thalach Trestbps Fbs – types n.

- *Testing the model*

  Mahout org.apache.mahout.classifier.sgd.RunLogistic – input heart.csv – model heart.model – auc – scores – confusion.

- *Dataset*

  Cleveland heart disease database (CHDD) is the de facto database for heart disease research. This database is used to train the proposed prediction model and it contains 76 variables, but all published experiments refer to using a subset of 14 of them. The Cleveland database is widely used by the machine learning (ML) researchers till date. Table 2 depicts the number of variables, description and its range.

Table 2 CHDD for training the logistic regression

S. no.	*Variable*	*Range*	*Description*
1	Age	Continuous	Age in years
2	Sex	0 to 1	Sex 0 = female, Sex 1 = male
3	Cp	1 to 4	Chest pain type (Cp 1 = typical angina, Cp 2 = atypical angina, Cp 3 = non-anginal pain, Cp 4 = asymptomatic)
4	Trestbps	Continuous	Resting blood pressure (in mm Hg)
5	Chol	Continuous	Serum cholesterol in mg/dl
6	Fbs	0 to 1	(Fasting blood sugar > 120 mg/dl) (Fbs 1 = true, Fbs 0 = false)
7	Restecg	0 to 2	Resting electrocardiographic results (Restecg 0 = normal, Restecg 1 = having ST-T wave abnormality, Restecg 2 = showing probable or define left ventricular hypertrophy by Estes' criteria)
8	Thalach	Continuous	Maximum heart rate achieved
9	Exang	0 to 1	Exercise induced angina (Exang 1 = yes, Exang 0 = no)
10	Oldpeak	Continuous	ST depression induced by exercise relative to rest
11	Slope	1 to 3	The slope of the peak exercise ST segment (Slope 1 = upsloping, Slope 2 = flat , Slope 3 = downsloping)
12	Ca	0 to 3	Number of major vessels (0–3) colored by fluoroscopy
13	Thal	3 to 7	(Thal 3 = normal, Thal 6 = fixed defect, Thal 7 = reversible defect)
14	Num	0 to 1	Diagnosis classes (Num 0 = healthy, Num 1 = patient who is subject to possible heart disease)

4 Results and discussions

Logistic regression with SGD algorithm is used in the proposed framework to develop the best prediction model. Logistic regression is trained using the prior clinical records and sensor data of the patients. The prediction model can uses current sensor data (blood pressure, blood sugar level and heart rate) of the patient to predict the heart disease status. In this analysis the prediction model use the current sensor data obtained from body sensor devices through cloud and big data technologies. Performance analysis of SGD with logistic regression is depicted in Table 4. It is observed that proposed prediction model efficiently classifies the heart disease with the accuracy of training and validation sample being 72.51% and 72.82% respectively. Performance evaluation metrics such as accuracy, sensitivity, specificity, precision and F-measure can be defined as,

$$Sensitivity = \frac{True\ Positive\ (TP)}{True\ Positive\ (TP) + False\ Negative\ (FN)}$$

$$Specificity = \frac{True\ Negative\ (TN)}{False\ Positive\ (FP) + True\ Negative\ (TN)}$$

$$Accuracy = \frac{True\ Positive\ (TP) + True\ Negative\ (TN)}{True\ Positive (TP) + False\ Neagtive (FN) + False\ Positive (FP) + True\ Negative (TN)}$$

$$Precision = \frac{True\ Positive\ (TP)}{True\ Positive\ (TP) + False\ Positive\ (FP)}$$

$$F\text{-}measure = 2 * \frac{Precision * Sensitivity}{Precision + Sensitivity}$$

CHDD is used by different models and the accuracies of different prediction models are tabulated in Table 5 and predicted classification table is depicted in Table 3.

Table 3 Predicted classification table

Training sample					*Validation sample*				
		Expected disease					*Expected disease*		
		Yes (1)	*No (0)*	*Total*			*Yes (1)*	*No (0)*	*Total*
Predicted disease	Yes (1)	99	23	122	Predicted disease	Yes (1)	40	7	47
	No (0)	15	74	89		No (0)	10	35	45

Table 4 Performance analysis of the prediction model

Accuracy	*Sensitivity*	*Specificity*	*Precision*	*F-measure*
		Training sample		
81.99%	86.84%	76.28%	81.14%	83.89%
		Validation sample		
81.52%	80%	83.33%	85.1%	82.47%

Table 5 Results of previous prediction model in CHDD

S. no.	*Technique*	*Accuracy*
1	Logistic regression with SGD algorithm	81.75%
2	Logistic regression	77%
3	Decision tree	78.9%
	Bagging algorithm	81.41%
4	Artificial neural networks ensembles	89.01%
5	Genetic algorithms with attribute weighted artificial immune system	87.0%
6	C4.5 decision tree	81.11%
	Naïve Bayes	81.48%
	Bayesian network with naive dependence	81.11%
	Bayesian network with naïve dependence and feature selection	80.96%
7	Artificial immune system	84.5%
8	Artificial immune recognition system with fuzzy and k-nearest neighbour	87.0%
9	Nine voting equal frequency discretisation gain ratio decision tree	84.1%
10	Cover learning using integer programming 3	84.0%
	Cover learning using integer programming 4	86.1%
	Cover learning using integer programming 4 ensemble	90.4%

5 Conclusions

Scalable sensor data processing architecture in cloud computing is proposed in this paper to store and process body sensor data for healthcare applications. Apache Flume, Apache Pig and Apache HBase are used in the proposed architecture to collect and store huge sensor data in the AWS. Online SGD algorithm with logistic regression is implemented using Apache Mahout to develop the best scalable diagnosis model. CHDD and wearable body sensors are used to train and test the scalable logistic regression model respectively. Proposed prediction model has achieved 81.99% accuracy for training sample and 81.52% accuracy for testing sample.

References

Apache Flume (2016) [online] https://flume.apache.org/ (accessed 11 May 2016).

Apache Hbase (2016) [online] https://hbase.apache.org/ (accessed 11 May 2016).

Apache Mahout (2016) [online] http://hortonworks.com/apache/mahout/ (accessed 11 May 2016).

Apache Pig (2016) [online] https://pig.apache.org/ (accessed 11 May 2016).

Babu, S., Chandini, M., Lavanya, P., Ganapathy, K. and Vaidehi, V. (2013) 'Cloud-enabled remote health monitoring system', *ICRTIT*: *Proceedings of the International Conference on Recent Trends in Information Technology*, Chennai, India, pp.702–707.

Bates, D., Saria, S., Ohno-Machado, L., Shah, A. and Escobar, G. (2014) 'Big data in health care: using analytics to identify and manage high-risk and high-cost patients', *Health Affairs*, Vol. 33, No. 7, pp.1123–1131.

Chehri, A., Mouftah, H. and Jeon, G. (2010) 'A smart network architecture for e-health applications', *Smart Innovation, Systems and Technologies*, Vol. 6, No. 1, pp.157–166.

Chen, M., Gonzalez, S., Vasilakos, A., Cao, H. and Leung, V. (2010) 'Body area networks: a survey', *Mobile Networks and Applications*, Vol. 16, No. 2, pp.171–193.

Corchado, J., Bajo, J., Tapia, D. and Abraham, A. (2010) 'Using heterogeneous wireless sensor networks in a telemonitoring system for healthcare', *IEEE Transactions on Information Technology in Biomedicine*, Vol. 14, No. 2, pp.234–240.

Deyhim, P. (2013) *Best Practices for Amazon EMR*, Technical Report.

Fengou, M., Mantas, G., Lymberopoulos, D., Komninos, N., Fengos, S. and Lazarou, N. (2013) 'A new framework architecture for next generation e-health services', *IEEE Journal of Biomedical and Health Informatics*, Vol. 17, No. 1, pp.9–18.

FitBit Inc. (2016) [online] https://www.fitbit.com/flex (accessed April 2016).

Hermon, R. and Williams, P.A. (2014) 'Big data in healthcare: what is it used for?', *Proceedings of Australian e-Health Informatics and Security Conference*, Perth, Australia, pp.40–49.

Jawbone Inc. (2016) [online] https://jawbone.com/up/trackers (accessed April 2016).

Jee, K. and Kim, G. (2013) 'Potentiality of big data in the medical sector: focus on how to reshape the healthcare system', *Healthc. Inform. Res.*, Vol. 19, No. 2, p.79.

Kang, D., Lim, W., Shin, K., Sael, L. and Kang, U. (2014) 'Data/feature distributed stochastic coordinate descent for logistic regression', *CIKM 2014: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, Shanghai, China, pp.1269–1278.

Lai, X., Liu, Q., Wei, X., Wang, W., Zhou, G. and Han, G. (2013) 'A survey of body sensor networks', *Sensors*, Vol. 13, No. 5, pp.5406–5447.

Latré, B., Braem, B., Moerman, I., Blondia, C. and Demeester, P. (2010) 'A survey on wireless body area networks', *Wireless Netw.*, Vol. 17, No. 1, pp.1–18.

Lopez, D. and Gunasekaran, M. (2015) 'Assessment of vaccination strategies using fuzzy multi-criteria decision making', *FANCCO 2015*: *Proceedings of the Fifth International Conference on Fuzzy and Neuro Computing*, Hyderabad, India, pp.195–208.

Lopez, D. and Sekaran, G. (2016) 'Climate change and disease dynamics-a big data perspective', *International Journal of Infectious Diseases*, Vol. 45, No. 1, pp.23–24.

Lopez, D., Gunasekaran, M., Murugan, B.S., Kaur, H. and Abbas, K.M. (2014) 'Spatial big data analytics of influenza epidemic in Vellore, India', *Big Data 2014: Proceedings of the IEEE International Conference on Big Data*, Washington, DC, pp.19–24.

Manogaran, G., Thota, C. and Kumar, M.V. (2016) 'MetaCloudDataStorage architecture for big data security in cloud computing', *Procedia Computer Science*, Vol. 87, pp.128–133.

Pantelopoulos, A. and Bourbakis, N. (2010) 'A survey on wearable sensor-based systems for health monitoring and prognosis', *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, Vol. 40, No. 1, pp.1–12.

Paradiso, R., Loriga, G. and Taccini, N. (2005) 'A wearable health care system based on knitted integrated sensors', *IEEE Transactions on Information Technology in Biomedicine*, Vol. 9, No. 3, pp.337–344.

Schmidt, K. (2013) *Programming elastic MapReduce: Using AWS services to Build an End-to-End Application*, O'Reilly Media, Inc., USA.

Thilagavathi, M., Lopez, D. and Murugan, B.S. (2014) 'Middleware for preserving privacy in big data', in Raj, P. et al. (Eds.): *Handbook of Research on Cloud Infrastructures for Big Data Analytics*, p.419, IGI Global Publishers, USA.

Victor, N., Lopez, D. and Abawajy, J.H. (2016) 'Privacy models for big data: a survey', *International Journal of Big Data Intelligence*, Vol. 3, No. 1, pp.61–75.