Applied Science & Engineering Journal for Advanced Research *Peer Reviewed and Refereed Journal*
ISSN (Online): 2583-2468
Volume-3 Issue-1 || January 2024 || PP. 20-31 DOI: 10.5281/zenodo.10578916

Big Data and Computer-Human Interaction for Real-Time Illness Diagnosis

Sonam Goyal*
Assistant Professor, Department of Computer Science, Rawal Institution, Faridabad, Haryana, India

*Corresponding Author: **

Received: 10-12-2023 Revised: 28-12-2023 Accepted: 27-01-2024

ABSTRACT
One particularly significant technology for deterrence of many chronic diseases is the constant plus real time tracking system, which is made possible through IoT and human computer interaction. Big data streaming stays the enormous volume of data that wearable medical procedures with sensors, healthcare clouds as well as mobile applications constantly produce. The increased pace of data collecting makes it challenging to gather, process as well as analyses such massive data sets in actual time in order to respond quickly in an emergency situation and unearth the hidden value. To offer an effective and scalable solution, real-time large data stream processing is therefore significantly needed. This work suggests a novel architecture for a big data based real time health prestige prediction as well as analytics system to address this problem. The system focuses on using a disseminated ML model to analyze health data events that are streamed into Spark via Kafka topics. First, we replace Hadoop MapReduce with Spark to produce a parallel, distributed, scalable, and rapid decision tree algorithm, which develops constrained for the real time computation. Second, this model is utilized to stream data from various sources that deal with numerous diseases in order to forecast health status. It is used to forecast health status using streaming data commencing distributed sources that represent various disorders.

Keywords: *healthcare, stream processing, human computer interaction, big data, apache spark, distributed machine learning, internet of things*

I. INTRODUCTION

Our time, which spans the former two decades, can be categorized as the "age of big data," in which digital data stands playing an ever-increasing role in a variety of industries, including society, research, health, and technology. Numerous domains and sources, including streaming machines, extraordinary throughput equipment, sensor networks, mobile applications, and each domain, particularly the healthcare industry, have generated and collected a significant amount of the data. Big data remains represented by this large size of data [1]. When employing inadequate tools with sophisticated technology, storing, processing, displaying, and extracting knowledge across these huge and various data kinds has become difficult. "One of the most important technological challenges facing big data analytics is finding effective ways to obtain relevant information for varied user types". The digital record of a patient's medical past is the most important data for healthcare analytics, and it is now being collected from a variety of clinical and non-clinical venues.

Consequently, there are three primary difficulties in developing a scattered data system to handle massive data: The first issue is that it is challenging to gather data from dispersed areas outstanding to the heterogeneous and enormous number of data. Second, the fundamental issue with heterogeneous and large datasets is storage. Big data systems must store data while guaranteeing performance. "The third issue is related to big data analytics, more precisely the mining of massive datasets in real-time or almost real time that encompasses modelling, visualization, prediction, and optimization [2]". These issues necessitate an innovative processing paradigm because the existing data management solutions are incapable of handling the diverse or real-time nature of the data. "However, traditional relational database management systems (RDBMS), such as MySQL, are largely responsible for organized data administration". These conventional systems don't support unstructured or partially organized data in any way. From the standpoint of scalability, characteristic RDBMS ascending for parallel hardware management as well as fault tolerance frequently fails when the data size increases, making it unsuitable for managing rising data. The research community has undertaken numerous efforts to address the problems related with enormous besides diverse data storing, such that NoSQL database management systems, which stay convenient after working with a lot of data when the data's nature organizes not need a relational model[3] [4].

MapReduce [5] remains the parallel dispensation procedure that combines the Map in addition to Reduce operations to handle large amounts of disseminated data. An inability of MapReduce to run iterative algorithms effectively is one of its main drawbacks. Iterative processing is not intended for MapReduce. Hadoop is a batch processing system that uses the MapReduce programming paradigm for the disseminated storage and processing of enormous quantities of data. It proposals a distributed storage solution with the Hadoop Distributed File System, which is as well precise fault tolerant. Hadoop only enables batch processing; this one cannot be meant for in memory computing or real time stream processing, nor is it always simple to apply the MapReduce paradigm to all issues. The amount of data being processed will determine how quickly the outcome will appear. In contrast, stream computing stresses data velocity and involves continuous data input and output. “Real time computing, distributed messaging, high throughput, and low latency processing are all features of Big Data Streaming Computing”. The main necessity of big data analytics in healthcare is the ability to extract facts from huge amounts of data, and BDSC is a promising option due to its massively parallel processing architectures.

Medical procedures and scholarly research have started to advance significantly as a result of the quick growth of massive data analysis. Huge amounts of heterogeneous, structured, in addition unstructured data produced by the present healthcare schemes can now be collected, managed, analyzed, and absorbed with the use of tools [6].

We have built an explanation in the healthcare with real time health eminence estimate use case established on the issues the healthcare system is facing. This system is built on the NoSQL Cassandra, Spark streaming, Spark MLlib, Kafka data streaming, and Apache Zeppelin technologies. Kafka's producers generate many message streams, which are subsequently processed at Spark streaming using machine learning and stored in a distributed storage NoSQL for analytics and visualization. The quality of patient monitoring is improved in healthcare by efficient data processing

II. PROPOSED HEALTH STATUS PREDICTION AND ANALYTICS SYSTEM ARCHITECTURE

The suggested solution combines Spark streaming and Kafka streaming to provide a data processing and monitoring application. Real time data received by linked devices will be processed by this application and stored for real-time analytics. “The proposed system's architecture is shown in Figure 1 First, Kafka manufacturers continually generate a stream of data messages that are collected by Kafka streaming; this stream is modelled by a subject that stretches names to various diseases”.

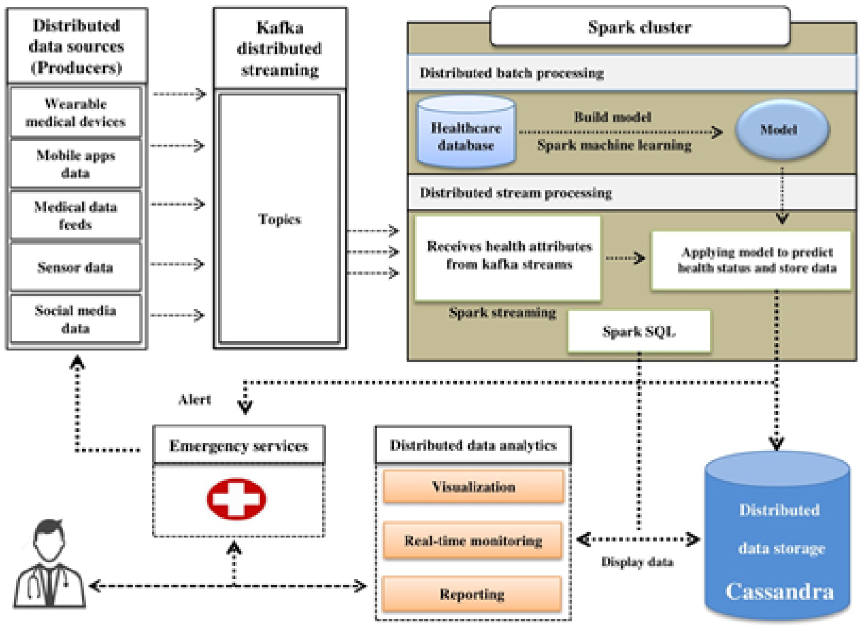


Figure1: Real-time health status prediction and analytics system architecture

The information will be regained from the database using Apache Zeppelin, as well as a dashboard will be created that shows the data in real time charts, lines, as well as tables. According to the suggested system design, it is possible to analyses monitor data in real-time and notify healthcare professionals of changes in a patient's status. By setting times between updates, the data will be automatically renewed. The subsections that follow provide more specific flow.

III. SOURCES OF DATA

The IoT is a network of physical objects as well as other things that are entrenched with electronics, intelligent clothes, software and apps, sensors, as well as network connectivity in order to gather and share data with one another or with data center systems. The data produced by wearable health monitors, which are widely available in households, is of a great volume and random character, necessitating analysis with the Big Data Analytic system in directive to comprehend user communication patterns or remove the necessary information. Health related IoT technology will account for 40% of all IoT connected technology by 2020 [7].

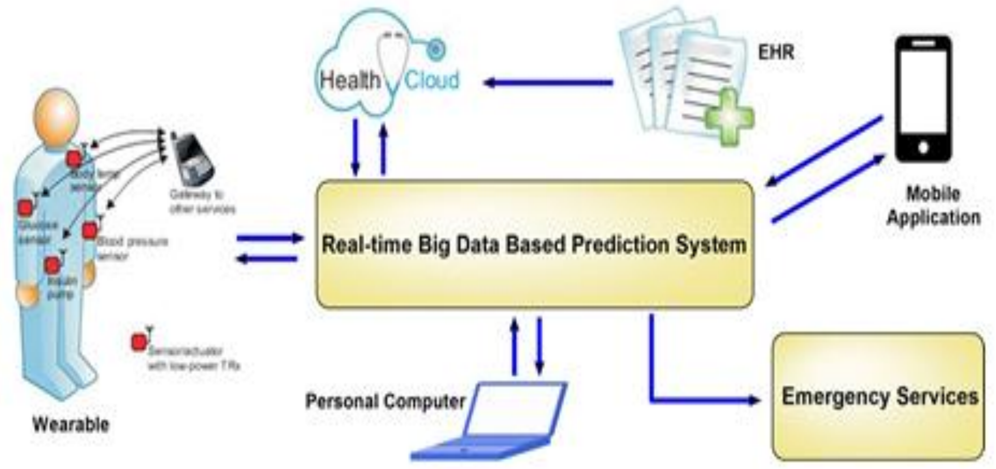


Figure2: Activity flow for the suggested system

By dropping inefficiencies, containing costs, and saving lives, the fusion of medicine as well as information technology, like the medical informatics, will revolutionize healthcare as we currently know it. “In the event of a medical emergency like heart disease, diabetes, or many other chronic conditions, real-time monitoring via IoT can save lives”. There are several sources accessible today that track health indicators regularly. The workflow for the suggested system with various data sources is shown in Figure 2.

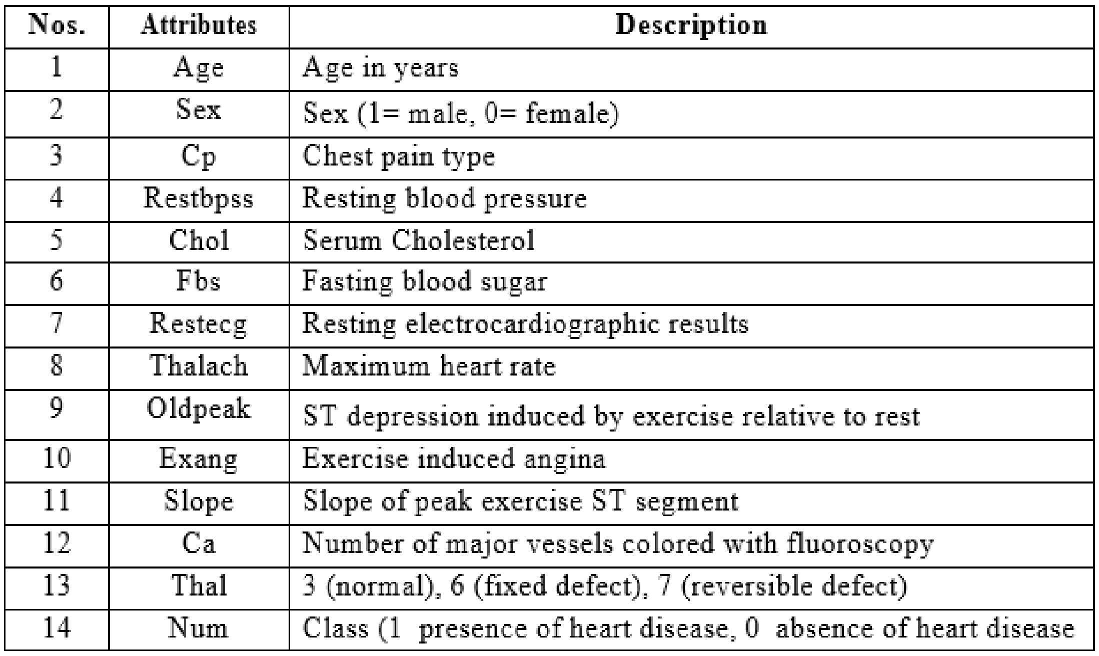
IV. KAFKA REAL TIME DATA COLLECTION

Since amount of data engendered in the healthcare industry is increasing exponentially, management of this data through Spark alone converts a difficult effort, whereas Kafka is made expressly for managing streaming data. It has therefore been incorporated into our system. The data collection block in the suggested system architecture is used to gather data on a person's health from dispersed sources and many diseases utilizing various devices combined with telemedicine and telehealth. This group continuously gathers, sorts, and manages clinical data about the patient. It enables us to group streaming data according to the relevant topic (kind of ailment) where records are released. Kafka's real-time data collection offers an unmatched solution for efficient and reliable data processing

V. CASE DATASETS UTILIZATION

Two datasets have been used to put the proposed model for this study into practice. The dataset we secondhand for the diabetic data analysis was obtained since the Kaggle [8] website, which offers data scientist’s online datasets with the goal of identifying and easily examining exposed data. “The diabetes dataset has 15,000 records as well as nine attributes, with 8 of those being pregnancies, blood pressure, skin thickness, insulin, bmi, diabetes pedigree function, age, plus one of two outcomes: whether the patient is diagnosed with diabetes, which is denoted by a 1 or a 0, or not, which is denoted by a 0”.

Table 1: Description of the heart disease dataset attributes



Nos.	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

The processed one is the second. It was used and evaluated to use Cleveland data from the Heart Disease (HD) database. There are 303 records in this tagged dataset, and there are 14 attributes shown in Table. Numerous machine learning research projects used it. We consume built a categorized dataset with qualities for each observation of heart illness, where the class label quality is marked with 2 classes. “A binary class dataset was created by changing the values of the class label attribute to simply 0 and 1, where value 1 denotes the existence of heart disease in place of values 1, 2, 3, and 4, and value 0 denotes the absence of heart illness”. Datasets are evaluated in this module utilizing the Spark environment and a predictive analysis technique.

VI. SPARK IMPLEMENTATION OF PARALLEL DECISION TREE

Afterward gathering data on different diseases after many disseminated sources, a classification model that can categories a user's qualities in the absence or presence of sickness must be built. One key method of data mining that can be used to uncover hidden information is categorization. A newly presented element is classified by looking at its qualities and being grouped into one of a specified set of classes. For classification and regression issues, DT are frequently utilized. Due to its ease of use, operationalization, interpretation, and extension to multiclass classification settings, DT are attractive methods for machine learning applications involving classification. The prediction was made using DT, which is supported for binary and multiclass classification by Spark's machine MLlib.

The ML model termed a DT splits the data into groups. A dual split recruits the partitioning process, which lasts until no more splits are likely. A DT is built using recursive partitioning, which involves dividing or not splitting each node in turn. The best split among all potential splits is chosen for each partition. “A specific criterion, such as Gini impurity and entropy, forms the basis for the divide. Based on the node's impurity, the homogeneity of the label is measured at the node level”. The implementation currently offers Gini and Entropy as two classification impurity measures.

Algorithm 1:

```
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 belongs to S do
SetInfo(a, T) = 0, andsplitInfo(a, T) = 0
Compute Entropy(a)
for v belongs to values (a,T) do
set Ta,v as the subset of T with attribute a = v
Info (a, T)+ = Ta,v Entropy(a )|
SplitInfo(a, T)+ = Ta,v log Ta,v
end for          GainRatio
Gain(a, T) = Entropy(a) − Info(a, T) SplitInfo(a,T )
end for set a_best = argmax GainRatio(a, T)
attach a_best into Tree
for values(a_best, T) do call C4.5(Ta,v)
end for
return tree
```

"The most well-known DT implementation is C4.5, which was created by J. Ross Quinlan, [9] and served as the default algorithm for DT on the Spark, sharing a parallel concept with C4.5 on MapReduce as shown in Algorithm 1". The entropy of feature S in this algorithm is determined as follows:

$$Entropy = -\sum_{j=1} P(S,j) * \log p(S,j) \tag{1}$$

"P (S, j) is nothing but the proportion of examples in S which are allocated to the jth class, and it indicates the proportion of instances in S that have the jth class label. C stands for the numeral of classes".

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

"A data required following attribute S splitting, where values Ts is the set of S values in T, Ts is the subset of T caused by S, and Ts, v is the subset of T where attribute S has a value of v. Information gain".

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

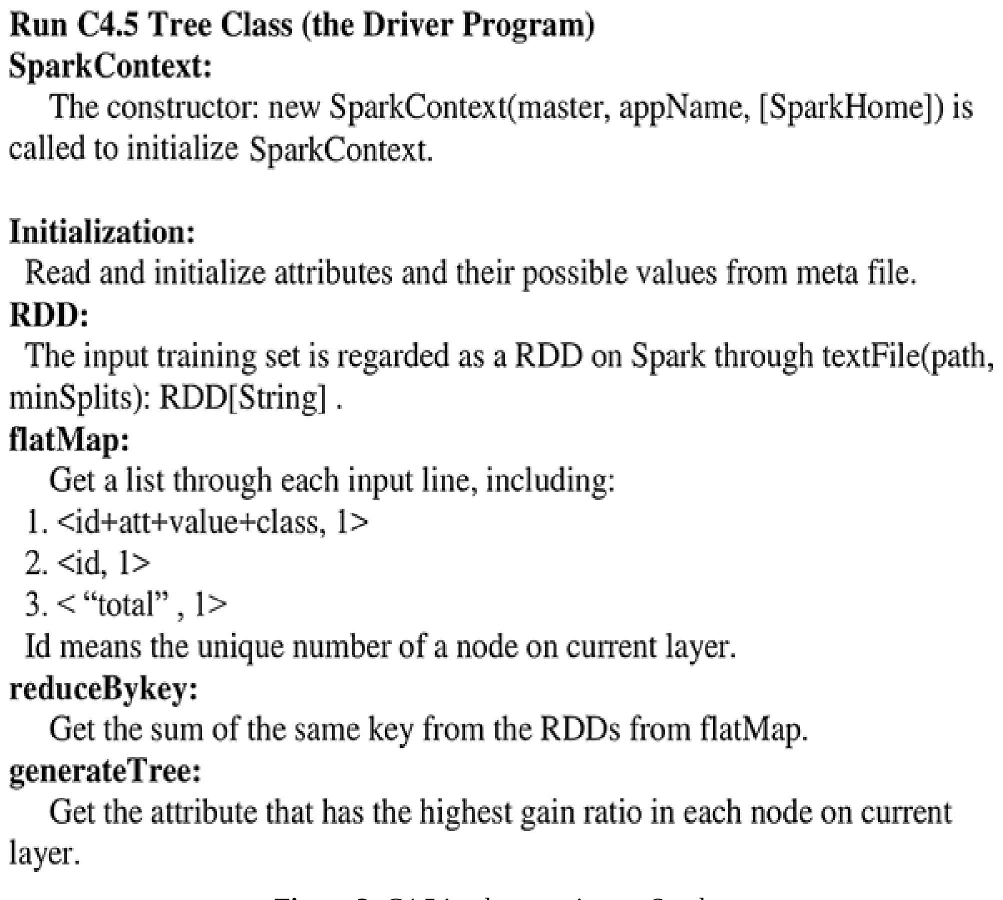
It calculates the information gain following attribute S-based segmentation. According to this definition, the qualities S's information gain ratio is:

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

SplitInfo is defined as follows:

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

Therefore, “a suitable as well as parallel model for health status prediction in a big data scenario utilizing Spark is required and this makes a C4.5 model edition additional crucial. Using Spark, C4.5 parallelization is carried out in this work”. Figure shows the pseudo code for C4.5 on Spark.



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.

Figure 3: C4.5 implementation on Spark

First, we access the cluster using SparkContext. Using the text File () function, data is loaded into an RDD. The contribution training dataset is treated by way of a text file RDD on Spark (). The RDD is cached using the cache () method, preventing the need for additional computation. Another transformation function in Spark is the flatMap function, which is just about identical to this map purpose in the MapReduce outline. “In the MapReduce architecture, the reduceByKey function merges the data for each key using the supplied function and returns an RDD”. It is a parallel variant of reduce. The procedures to train as well as test the DT in a disseminated atmosphere based on Spark are represented by Algorithm. The DT implementation in this work is carried out using MLlib, while Spark streaming handles the Kafka topic data streams.

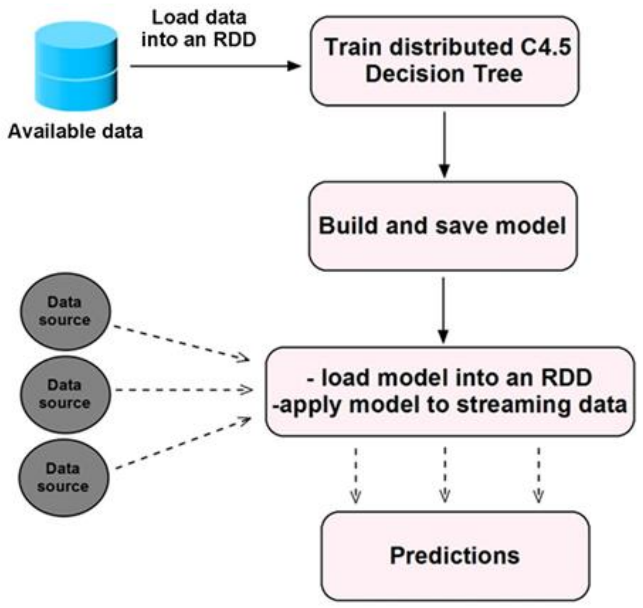


Figure 4: Machine learning process

VII. RESULT AND ANALYSIS

Performance Evaluation of Machine Learning Model

30% data will be used for the testing, and the left over seventy percent will be secondhand to train the model from the two datasets. These data have been used to train DT. "In this application, an estimated set of split candidates are intended over a sampled portion of data, as well as the ordered splits create bins, the maxBins parameter specifies the extreme number of such bins, and the maxDepth parameter specifies the maximum depth of the DT". Sorting feature values is expensive for large distributed datasets.

Different DT models consume been weighed by means of the dataset underneath with variable parameters for maxDepth, maxBins, as well as impurity indices, also the sorting exactness values are intended in apiece case. The greater accuracy prediction stabilizes when the number of maxBins and maxDepth take the values shown in Table 4 thanks to the testing dataset and model error analysis, which avoids the detrimental impacts of both under fitting and overfitting. Figure 8 displays in a bar graph how well the DT implementation utilizing Spark performed.

- Receiver Operating Characteristic Curve: is unique of tfle most significant as well as effective metrics of evaluating tfle excellence or presentation of diagnostic tests. The ROC curve is reinforced by MLlib.
- Classification Accuracy: This is measured by the proportion of accurate predictions to all other predictions. The classification accuracy for tfle datasets is measured in this work by means of the tfle equation:

Table 2: carried out 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 3: 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

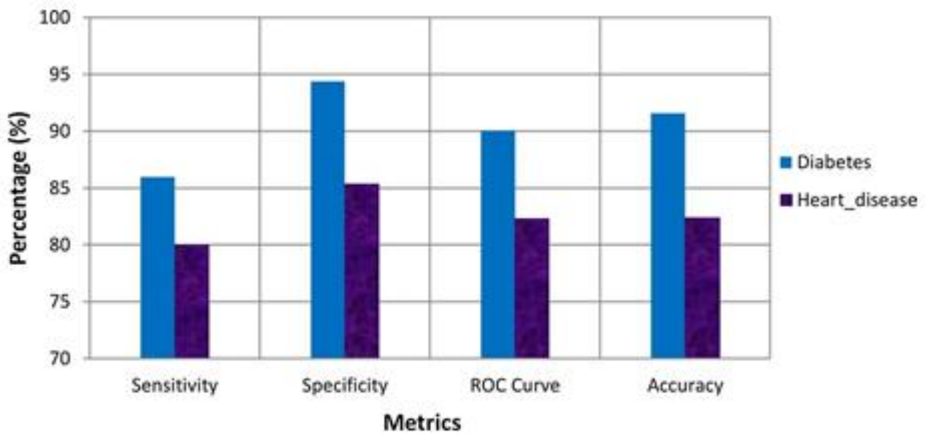


Figure 5: Machine Learning Results

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (6)$$

In such a way that TP, TN, FP, FN as well as FP represent the values of true positives, true negatives, and false positives respectively. Sensitivity is the percentage of positives that are actually properly recognized, whereas specificity measures the percentage of true negatives that are correctly detected. These were produced by:

$$\text{Sensitivity} = \frac{TP}{TP + FN} \quad (7)$$

$$\text{Specificity} = \frac{TN}{FP + TN} \quad (8)$$

We evaluated the performance of our ML model on dual fictional data sets. Actual results demonstrate how effective and scalable our use of Spark to implement the DT algorithm is rendering to the table above, the proposed model provides consistent and excellent predictions.

Apache Spark vs Weka Performance

Spark speed can be significantly quicker than additional older technologies, particularly in iterative ML, with to features like in memory processing. Additional data records consume been replicated in order to demonstrate the effectiveness of the Spark based prediction system in expressions of the time required to train and test the machine learning model on huge datasets. Scikitlearn, a Python package for ML that offers purposes for creating a set of test problems, is secondhand to run the simulation.

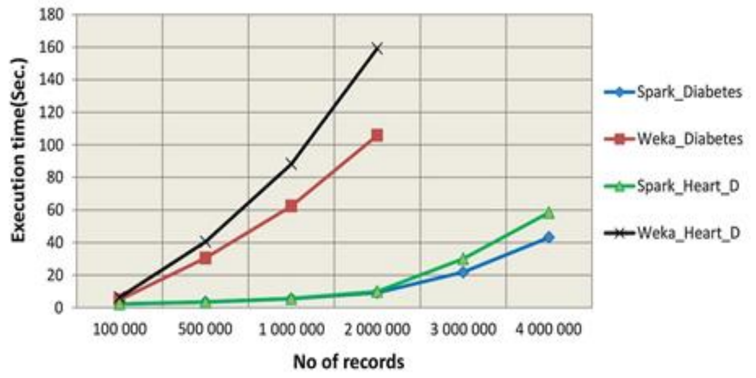


Figure 6: Model building time is compared between DT using Spark and regular DT for execution times.

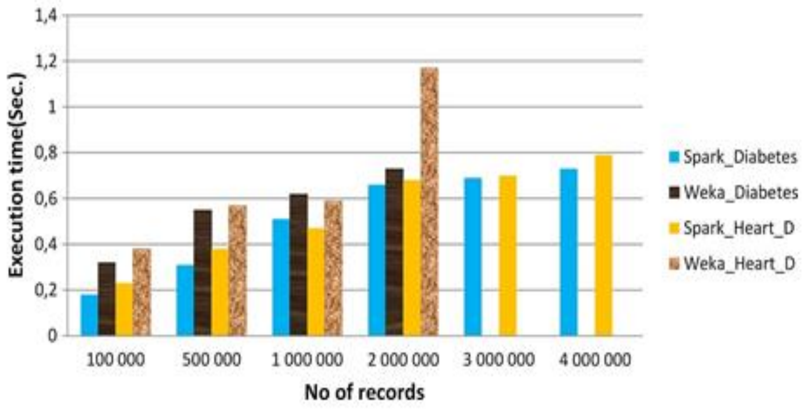


Figure 7: Model testing time is compared between DT using Spark and traditional DT.

DT execution times using the Spark and Weka tools were compared in instruction to assess and show the scalability of this technique. In this situation, we increase the amount of records in order to have a large enough database. Then, using the identical technique in Weka and Scala, the rapidity of the DT algorithm was measured as well as compared. In fact, Weka and DT's operating times in the cluster (Spark) were compared. The presentation judgement of the implementation of DT utilizing Spark as well as Weka is shown in a bar chart in Figure 6 and 7. In this simulation, employing Weka's straightforward C4.5 does not enable model training when the amount of records is equivalent to or more than three million.

The line as well as bar charts show that running the DT model using Spark is quicker than using the Weka tool. In contrast to Weka, which needs 185 seconds to train a model with four million records from a diabetic dataset, it only requires 43.2 seconds. Additionally, it trains the model in 58.43 seconds as opposed to 274.36 seconds for Weka when using a dataset of 4 million records with heart disease. The proposed Spark based DT makes machine learning model testing and training faster. Due to disseminated computing on the cluster nodes and in memory computation, the parallel DT method of Spark Mllib achieves the finest scalability. Data processing with Spark MLlib takes less time since the workload is broken up into smaller jobs that are carried out on workers nodes. Spark offers the best means of implementing the suggested method to calculate health condition in real time, according to an examination of the findings obtained.

Spark Based DT Scalability

In this step it involves measuring the proposed Spark based C4.5 algorithm's performance in a disseminated parallel atmosphere. Dissimilar training dataset sizes as well as node counts are taken into consideration. We have two nodes, as was already indicated, and our training dataset contains between 100k and four million entries.

We can perceive that the overall implementation time lowers as the number of nodes grows when the number of instances is two, three, and four million respectively. This suggests that the algorithm will run more quickly due to distribute computing the more nodes engaged in the computation. In contrast, because to the undistributed computing, in the conventional DT execution time utilizing Weka with two nodes stays steady.

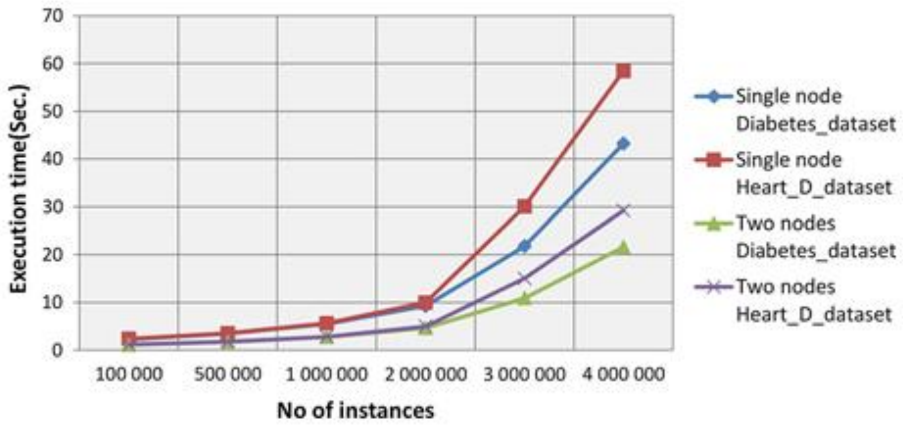


Figure 8: Performance evaluation of DT with Spark on several nodes: duration of model construction

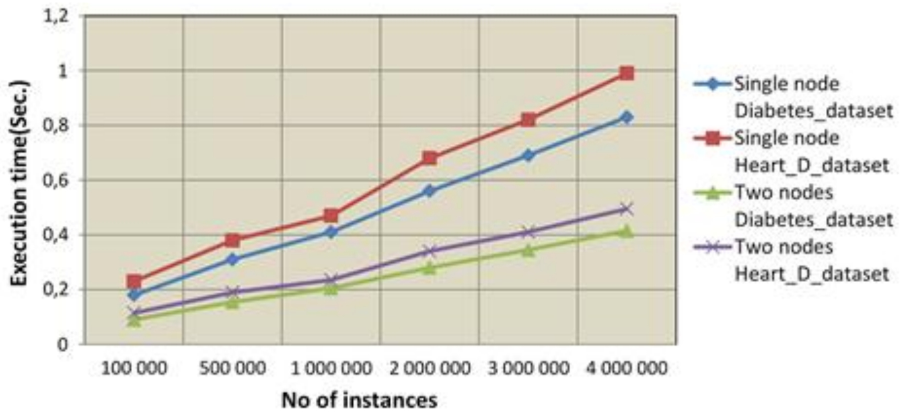


Figure 9: Performance evaluation of DT with Spark on several nodes: duration of model testing

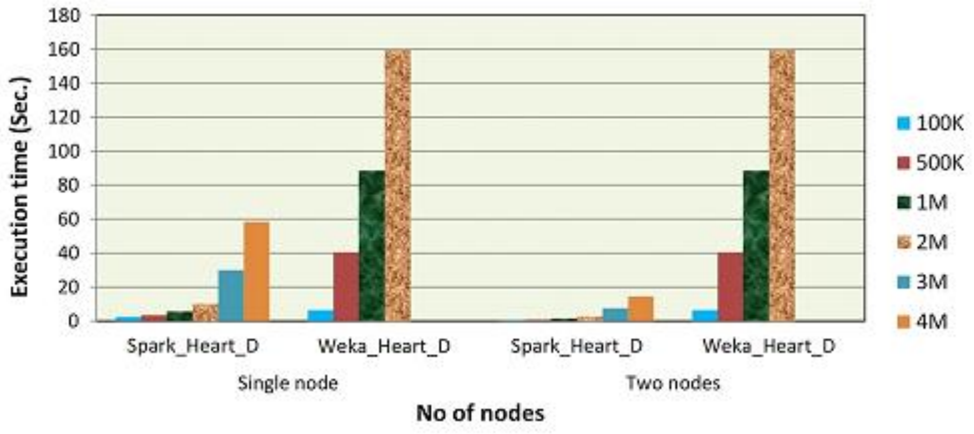


Figure 10: Performance evaluation of DT for various nodes using Spark and Weka: construction time for the model. Thousand, Million

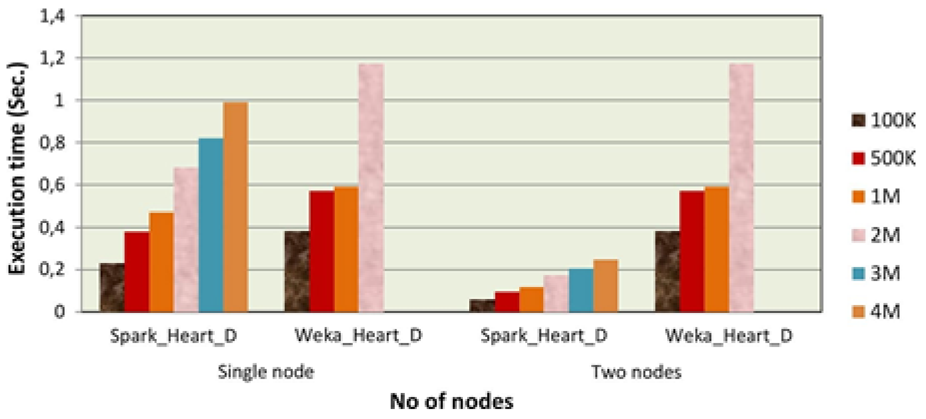


Figure 11: Performance evaluation of DT for various nodes using Spark and Weka: elapsed time for model testing. Thousand, Million

Throughput

First, the DT model was constructed and evaluated independently using different parameters including impurity, maxDepht, as well as maxBins; the smallest model fault is occupied into consideration based on the model's ability to accurately classify data. To use it in real time, an offline model has remained built as well as saved. In our instance, the data generators are two simulator programs, one for diabetes streams and the other for heart disease streams. We are only employing two producers for the purpose of simplicity. The Kafka data streaming module is in charge of handling the events streams while Kafka streaming captures all of these data events in real time as well as sends them to the Spark streaming application. The Spark streaming API is used by streaming applications to perform a series of transformations on data streams in order to anticipate a user's health state. The produced identity and attribute values for each instance were retrieved, and the extracted health attributes were subjected to a machine learning model. Each instance's specifics were saved in a Cassandra database table so they could be later queried.

According to Fig

- The cost of processing time increases as throughput increases.
- The processing time decreases as we add more nodes.
- If sufficient nodes are used, even if the volume of data is large, presentation may be close to the ideal level. For instance, if we use 2 nodes and have a throughput of 2.5 million records, the time it takes to complete the task is close to 2 s. When we custom a single node, an execution duration is close to three seconds. By adding more nodes, the execution time of the same output decreases.
- Distributed streaming using Spark is a solid option for dealing with real-time issues. In particular, we handle big data issues, notably those connected to real-time prediction in the field of health care, by utilizing additional nodes.

A data dashboard that retrieves information after the Cassandra database as well as presents it in charts as well as tables has been developed using Apache Zeppelin. The data is pushed to the web page in set intervals by this application using Angularjs as well as Spark SQL, ensuring that data is automatically refreshed. It is available from both desktop and mobile devices. In command to excerpt the crucial and useful information or to help practitioners understand user behavioral patterns, the database can be queried using a variety of different queries, such as the number of instances, the number of positive as well as negative cases, approximately statistics, and the status of a patient by his identifier.

Our research focuses on using ML models with HCI to analyze streaming big data generated from a variety of illness sources. Kafka is used to manage the event streams then convert the healthcare data into information. Discussion of the experiment's findings leads to the conclusion that Spark is particularly suited for iterative algorithms that call for repeated data passes. It offers a quicker execution engine for processing that is distributed and streaming. The use of Spark in this system accelerates data processing more quickly than other conventional data mining tools.

The primary distinction concerning the proposed system and conventional approaches to data analytics is that the latter analyze one instance at a time and are dependent on the volume of input data. On the other hand, this system, which is constructed on Dstream, which is a collection of RDD and each RDD represents one or additional instances, can procedure thousands of instances coming in each second in real-time. Additionally, the system supports big data processing, uses real time as well as distributed machine learning, handles incoming streams using Spark streaming rather than MapReduce, predicts multiple diseases at once based on the idea of Kafka topics, and offers quick real time classification, which is more significant as the amount of produced data from devices, the cloud, as well as other sources is growing at an apparent rate.

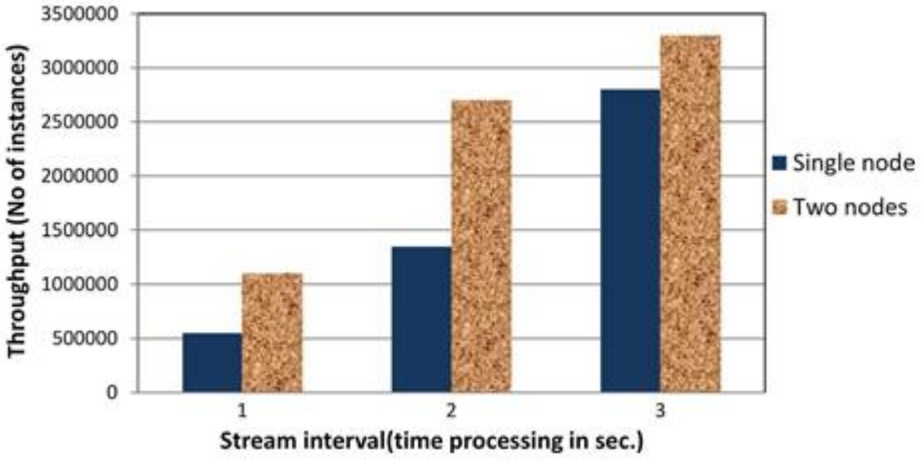


Figure 12: Performance evaluation: processing time

VIII. CONCLUSION

Volume of the healthcare data is expanding alarmingly quickly over time from numerous conflicting data sources. A streaming computing platform is required to enhance patient outcomes and commercialize real-time health status prediction systems. However, outdated information technology solutions integrating physical infrastructure as well as relational databases are no longer able to collect, process, store, and utilize this dizzying volume of data. Based on the difficulties previously described, classical information technology has numerous problems in scaling with parallel hardware, making it unsuitable for handling expanding data. In our research, a real time health status estimate and analytics system that was created using open source big data technologies is suggested and evaluated on a cluster.
The integration of big data and human computer interaction holds immense potential for real-time disease prediction; however, its effectiveness must be balanced against concerns of privacy invasion and the ethical implications surrounding data collection and usage.

REFERENCES

1. Manogaran G, & Lopez D. (2018). Health data analytics using scalable logistic regression with stochastic gradient descent. *Int J Adv Intell Paradigms*, *10*(1–2), 118–32.
2. Hu H, Wen Y, Chua T‑S, & Li X. (2014). Toward scalable systems for big data analytics: A technology tutorial. *IEEE Access*, *2*, 652–87.
3. Cattell R. (2011). Scalable sql and NoSQL data stores. *ACM Sigmod Record*, *39*(4), 12–27.
4. Moniruzzaman A, & Hossain SA. (2013). *NoSQL database: New era of databases for big data analytics ‑classification, char ‑ acteristics and comparison*. arXiv preprint arXiv:1307.0191.
5. Dean J, & Ghemawat S. Mapreduce. (2008). Simplified data processing on large clusters. *Commun ACM*, *51*(1), 107–13.
6. Belle A, Thiagarajan R, Soroushmehr S, Navidi F, Beard DA, & Najarian K. (2015). Big data analytics in healthcare. *BioMed Res Int*.
7. Bauer H, Patel M, & Veira J. (2017). *The internet of things: Sizing up the opportunity*. Available at: http://www.mckinsey.com/.
8. Zaharia M, Chowdhury M, Das T, Dave A, Ma J, McCauley M, Franklin MJ, Shenker S, & Stoica I. (2012). Resilient distributed datasets: A fault‑tolerant abstraction for in‑memory cluster computing. *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*.
9. https://www.kaggle.com/fmendes/diabetes‑from‑dat263x‑lab01.
10. Quinlan JR. (2014). *C4. 5: Programs for machine learning*. Amsterdam: Elsevier.