Stallion Journal for Multidisciplinary Associated Research Studies
ISSN (Online): 2583-3340
Volume-3 Issue-5 || October 2024 || PP. 54-67 https://doi.org/10.55544/sjmars.3.5.7

Real-Time Disease Prediction Using Big Data and Human-Computer Interaction

Anita
Assistant Professor, Department of Computer Science & Engineering, Shri Ram College of Engineering and Management, Palwal, Faridabad, Haryana, INDIA.

Corresponding Author: anitasorout4@gmail.com

www.sjmars.com || Vol. 3 No. 5 (2024): October Issue

Date of Submission: 21-10-2024	**Date of Acceptance:** 31-10-2024	**Date of Publication:** 25-11-2024

ABSTRACT

Big data streaming involves managing the vast volumes of data generated continuously by wearable medical devices with sensors, healthcare cloud platforms, and mobile applications. Traditional methods for processing this data are often time- and resource-intensive. To address this challenge, there is a need for efficient and scalable real-time big data stream processing. This study introduces a novel architecture for a big data-driven real-time health status prediction and analytics system. In this architecture, we replace Hadoop MapReduce with Spark to enable a parallel, distributed, and scalable decision tree algorithm capable of handling real-time computations. This model is then applied to streaming data from various sources, supporting the prediction of health statuses across multiple diseases. Using distributed streaming data, the system predicts health conditions associated with different disorders. To evaluate the performance, we compare Spark's decision tree (Spark DT) with traditional machine learning tools such as Weka. Key performance metrics, including execution time and throughput, are analyzed to assess the effectiveness of the proposed architecture. Experimental results demonstrate that the proposed system can effectively manage and predict vast amounts of real-time IoT-enabled medical data related to various disorders, showcasing its potential for real-time healthcare applications.

Keywords- Healthcare, Stream processing, human-computer interaction, Big data, Apache Spark, 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]. 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”. The research community has undertaken numerous

Stallion Journal for Multidisciplinary Associated Research Studies
ISSN (Online): 2583-3340
Volume-3 Issue-5 || October 2024 || PP. 54-67 https://doi.org/10.55544/sjmars.3.5.7

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. 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].

The topic of study identified as "human computer interaction" is concerned with the boundaries that public consumption to interrelate with computers. HCI researchers study how public custom computers then create explanations that enable public to use them in innovative ways. “A Human-computer Interface is a tool that facilitates computer and human communication”.

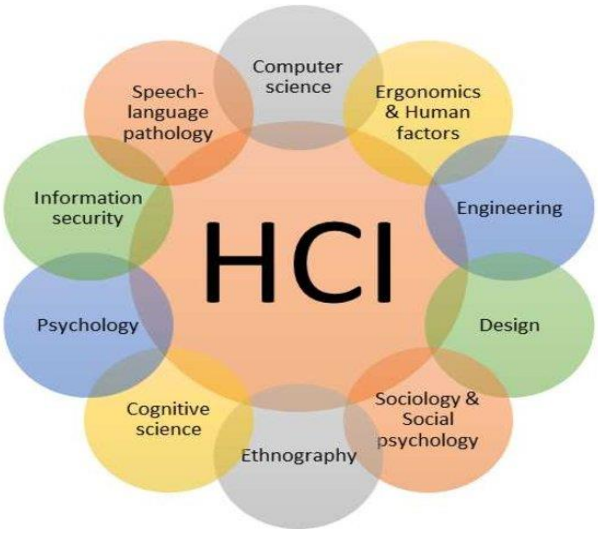

Figure 1: Different Research fields in HCI

II. BIG DATA CHALLENGES IN HEALTHCARE

“The 5Vs of big data, specifically Volume, Velocity, Variety, Veracity, as well as Value, can be used to explain the vast amount of the data that the healthcare sector produces today [7]”. The amount of healthcare data that must be gathered and analyzed is significant and constantly growing, and the diversity refers to the data that must be gathered from various sources. Healthcare data and domain knowledge, specifically velocity, should be current. The trustworthiness of the healthcare data is referred to as veracity. Finally, by carefully examining the enormous amounts of data in healthcare, significant information could be discovered. Distributed sources of healthcare data include electric medical records, medical photographs, analysis data, health claim data, streaming systems, as well as sensors affixed to patients' bedsides to continuously monitor their vital signs. They generate enormous amounts of data, which typical data processing systems are unable to handle adequately [8]. Figure [9] provides a concise summary of the big data difficulties. “Clinical records, health research records as well as organization operations records are the three main types of digital data used in the healthcare sector to collect data. A quick review of these sources is given in [10]”. In order to compare several methods of prediction and choose the most accurate one, an experiment was conducted in [11]. A genetically tuned neural network model is used to classify breast carcinoma [12]. Other information retrieval and data mining methods have been put forth in [13, 14].

Healthcare analytics have been examined in a variety of systems, including those for widespread prediction as well as inhibition, health approval systems, and medical decision-making to enhance care quality, save costs, and boost productivity. “A cloud based K means clustering using MapReduce that uses cloud based

Stallion Journal for Multidisciplinary Associated Research Studies
ISSN (Online): 2583-3340
Volume-3 Issue-5 || October 2024 || PP. 54-67 https://doi.org/10.55544/sjmars.3.5.7

healthcare data for clustering has been proposed in Hadoop and HBase are used to propose a web enabled disseminated electronic and personal health record management platform [15][16]". In [17], the predictive analytic algorithm as well as Hadoop MapReduce situation are used to forecast diabetes mellitus as well as the type of treatment to be used. "An example of IoT centered big data contextual sharing across all devices in a health system is provided in [18] by a Hadoop based intelligent care system". The suggested system uses the network architecture by means of improved processing abilities to gather data produced by various linked devices and the gathered data sent to intelligent buildings. [19] Provides a description of real time analysis targeted at electronic medical chronicles generated from a variety of sources, including medical devices and mobile applications. The proposed system, which incorporated Hadoop, MongoDB, and an unpremeditated treatment method, remained intended to enhance the outcomes of the processing of patient records.

Hadoop [20] is the main focus of the majority of healthcare analytics solutions because it can process a substantial number of data from a variety of sources when batch computing is used. Hadoop would have limitations for real time computing; nevertheless, Spark is quicker as well as performs improved than Hadoop, particularly for issues involving iterative machine learning [21]. The two most well-known tools in the big data environment, Hadoop and Spark, are both Apache projects, with Spark generating the most buzz. Some of the distinctions between these two platforms are shown in Table 1. The development of several scalable machine learning methods, on the other hand, aims to address the numerous problems in big data analytics. In [22], an ascendable Random Forest classification algorithm is used to produce a diabetes risk prediction model. [23] Discusses the convention of online logistic regression for phishing URL finding, with Mahout utilized for machine learning and Hadoop employed for data processing.

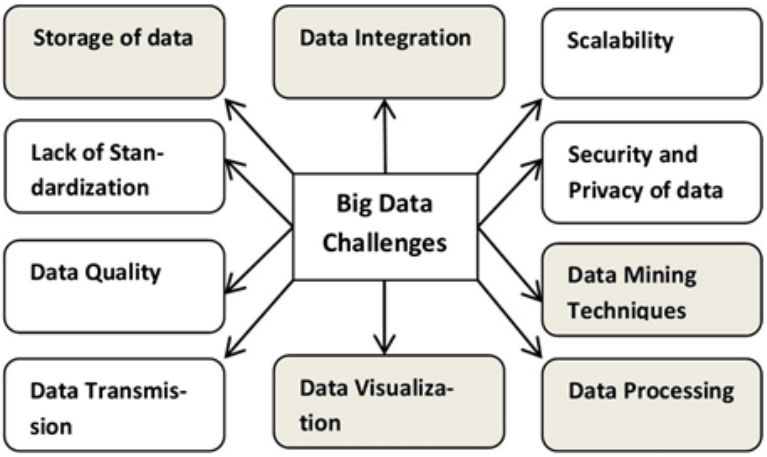

Figure 2: Big Data Challenges

The automated system based on a markov chain model that may identify anomalous patterns in the entrance and exiting behaviors of elderly people living alone [24]. The data are acquired through basic sensors installed in home-based settings [25]. "It discusses a real time medical emergency retort system that uses IoT centered medical sensors placed on the human body [26] which provides an overview of big data infrastructures and machine learning methods used in healthcare". Several research projects incorporate machine learning, but very few of them process streaming data. Numerous studies have been conducted to reveal important evidence in the analysis of social media data, particularly that from Twitter as well as other sources for efficient healthcare. For instance, [27] describes a real time flu as well as cancer surveillance system using Twitter data mining.

III. METHODOLOGY

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. 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". They are delivered to the Spark streaming programmer, which carries out the in-the-moment processing. Multiple health attributes are received through Kafka streaming by Spark streaming, which then applies a ML model to forecast health prestige as well as stores data in Cassandra, a NoSQL database. The information will

Stallion Journal for Multidisciplinary Associated Research Studies
ISSN (Online): 2583-3340
Volume-3 Issue-5 || October 2024 || PP. 54-67 https://doi.org/10.55544/sjmars.3.5.7

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.

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.

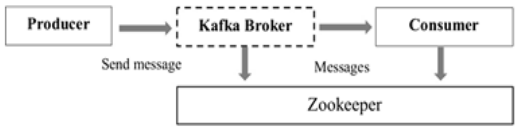

Figure 3: Kafka messaging system

Apache is publishing subscribe messaging distributed streaming system Kafka [28] is designed to be a disseminated, partitioned and replicated service. Through Kafka producer, the health monitoring devices feed real-time data. The Kafka communications system is depicted in Figure 3. Utilizing Zookeeper, distributed systems are coordinated and made easier. In this work, the data producers are binary connected device simulator programmers that use Apache Kafka to produce data events [29].

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

$Info(a, T) += \frac{Ta,v}{} Entropy(a)|$

$SplitInfo(a, T) += \frac{Ta,v}{} \log \frac{Ta,v}{}$

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

Stallion Journal for Multidisciplinary Associated Research Studies
ISSN (Online): 2583-3340
Volume-3 Issue-5 || October 2024 || PP. 54-67 https://doi.org/10.55544/sjmars.3.5.7

"The most well-known DT implementation is C4.5, which was created by J. Ross Quinlan, [30] 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:

$$Entropy = -\sum_{j=1} P(S, j) * logp(S, j) \quad (1)$$

"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".

$$Info(S, T) = -\sum_{v \in Values\ (T_s)} \frac{|T(S, v)|}{|T|_S} Entropy\ (S_V) \quad (2)$$

"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".

$$Gain(S, T) = Entropy(S) - Info(S, T) \quad (3)$$

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

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

SplitInfo is defined as follows:

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

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.

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 3.5: 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 reduce by Key function merges the data for each key using the supplied

Stallion Journal for Multidisciplinary Associated Research Studies
ISSN (Online): 2583-3340
Volume-3 Issue-5 || October 2024 || PP. 54-67 https://doi.org/10.55544/sjmars.3.5.7

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 2. The DT implementation in this work is carried out using MLlib, while Spark streaming handles the Kafka topic data streams.

ALGORITHM 2:

Step1: Start new Spark Context

Loading required package and APIs

spark Context (master, appName,

sparkHome)

step2: Load and parse the dataset into an RDD

Row Data (RDD) : sc.textFile (path) Data (RDD): Map (pareseFunction

(row Data)) 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)

Experimental Setup

Scala and Zeppelin were used as the development platforms to create a real time health position estimate system based on Spark, Kafka as well as Cassandra, which supports several interpreters including Scala, Spark, and Cassandra. The code and its dependencies do not need to be contained in an assembly package. "First, the suggested application is run on a solo node cluster built with a core i7 CPU, 8 GB of RAM, and Ubuntu 16.04 using the Spark platform.

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

The claim connects to Kafka streaming as shown in Figure as well as begins to receive data streams from various Kafka manufacturers. When it comes across streams that check health attributes, it excerpts the characteristic values from individually topic of disease events sent by Kafka streaming as well as uses the DT model to predict the health status. The Cassandra data base, on the other hand, uses the identification as the main key to record each projected state in a table, which is better suited for data redundancy. Later, a database query will be run to analyze past data.

A multi node Cluster was constructed after the program had been tested on a single node cluster. In this research were carried out using the existing computing power on a cluster that consisted of master node as well as two worker nodes. Ubuntu 16.04 uses VMware virtual nodes as its operating system.

Stallion Journal for Multidisciplinary Associated Research Studies
ISSN (Online): 2583-3340
Volume-3 Issue-5 || October 2024 || PP. 54-67 https://doi.org/10.55544/sjmars.3.5.7

ALGORITHM 3:

Step1: Create a group called spark, and user called spuser and add the spuser to sudoers list

addgroup spark adduser –ingroup spark spuservisudo

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.pubhome/.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/binStart Zookeeper Start Kafka

Create Kafka topic Cd home/usr/local/cassandra/binStart Cassandra

Create a keyspace and table Cd /usr/local/zeppelin/bin Start Zeppelin

First, to make communication between nodes simpler, a group called as the Spark then Spark user accounts have been developed. Scala and Java have been set up. Installing the Open SSH Server, creating key pairs, as well as configuring password less communication between nodes enable Spark master to connect, launch, stop, and run jobs across several workers. Spark, Kafka, and Cassandra have all been unpacked and installed on a single node. There are two themes and tables, one for diabetes disease and the other for heart illness. "We modify the user's home directory's .bashrc file as well as add environment variables like JAVA HOME also SPARK HOME and Add file slaves with the hostname of the workers in $SPARK HOME/conf". We duplicate the solitary node cluster setup folder 3 times in order to have identical copies of various frameworks, renaming 1 master and the other 2 worker:1 and worker:2. On all nodes, modify the hostname and hosts. The stages to build up the cluster are represented by Algorithm 3.

ALGORITHM 4:

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 saveToCassandramethod

Step 4: Start the computation

Start Spark streaming context using start method

Three situations were tested with stream intervals of 1, 2, and 3 s. The outcomes of our studies are displayed in Table 1.

Stallion Journal for Multidisciplinary Associated Research Studies
ISSN (Online): 2583-3340
Volume-3 Issue-5 || October 2024 || PP. 54-67 https://doi.org/10.55544/sjmars.3.5.7

IV. RESULTS AND DISCUSSION

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 1: 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 2 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

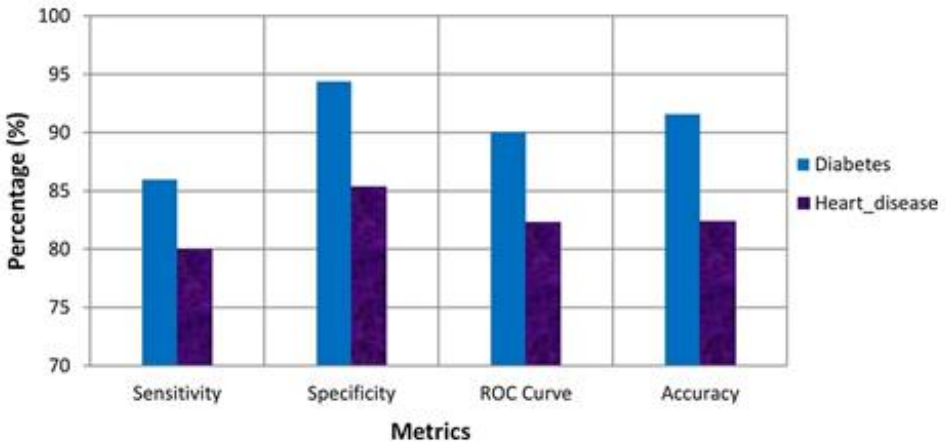

Figure. 5 Machine Learning Results

Stallion Journal for Multidisciplinary Associated Research Studies
ISSN (Online): 2583-3340
Volume-3 Issue-5 || October 2024 || PP. 54-67 https://doi.org/10.55544/sjmars.3.5.7

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

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.

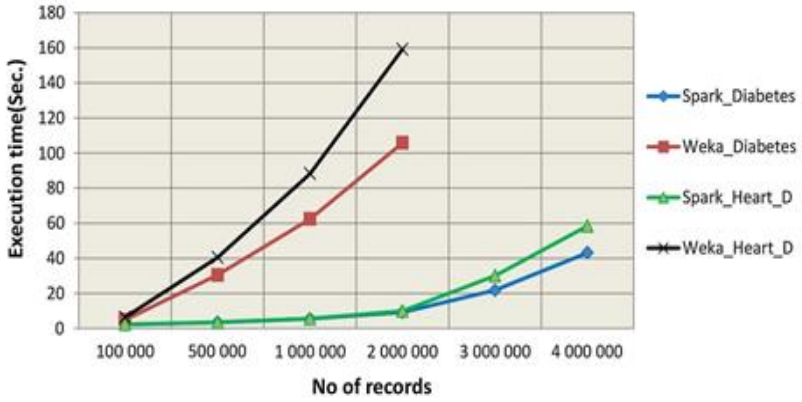

Figure. 6 Model building time is compared between DT using Spark and regular DT for execution times.

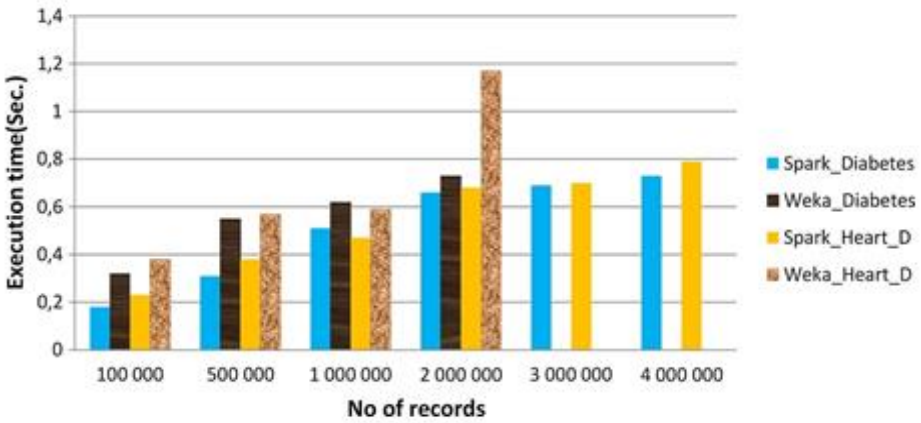

Figure. 7 Model testing time is compared between DT using Spark and traditional DT.

Stallion Journal for Multidisciplinary Associated Research Studies
ISSN (Online): 2583-3340
Volume-3 Issue-5 || October 2024 || PP. 54-67 https://doi.org/10.55544/sjmars.3.5.7

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.

Figure 8, 9, 10, and 11 show how long Spark based C4.5 algorithm takes to run through various numbers of nodes. 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.

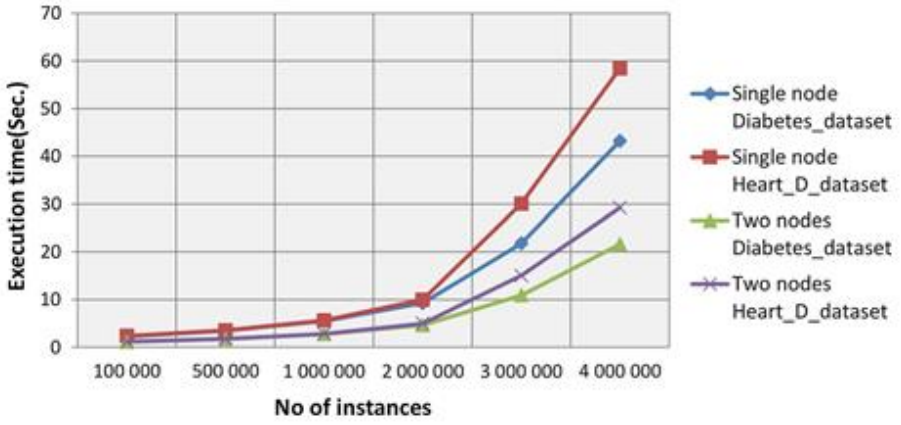

Figure. 8: Performance evaluation of DT with Spark on several nodes: duration of model construction

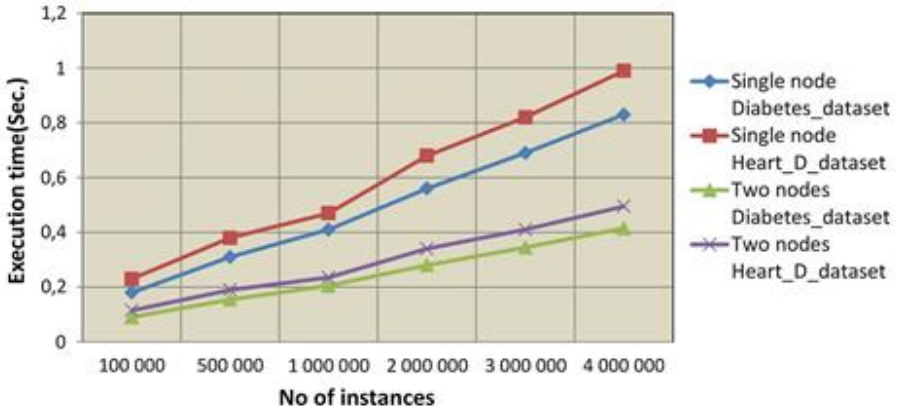

Figure. 9 Performance evaluation of DT with Spark on several nodes: duration of model testing

Stallion Journal for Multidisciplinary Associated Research Studies
ISSN (Online): 2583-3340
Volume-3 Issue-5 || October 2024 || PP. 54-67 https://doi.org/10.55544/sjmars.3.5.7

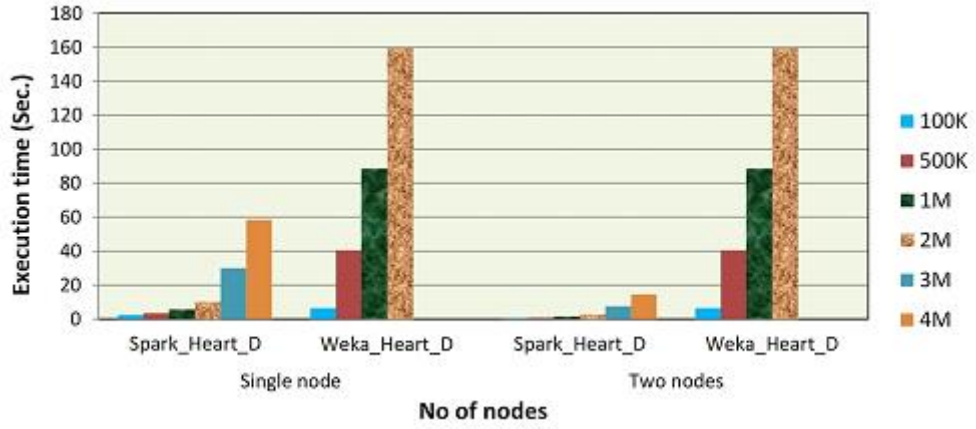

Figure. 10 Performance evaluation of DT for various nodes using Spark and Weka: construction time for the model. Thousand, Million

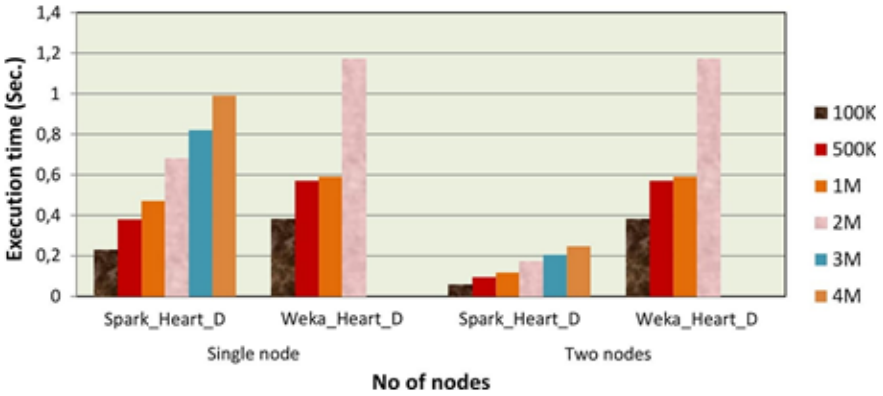

Figure. 11 Performance evaluation of DT for various nodes using Spark and Weka: elapsed time for model testing. Thousand, Million

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. Each of them transmits roughly 270,000 events/second/node, in a predetermined arrangement to the designated topic, which then feeds them to customers as shown in Fig. 12

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.

Using the Spark monitoring API, we calculated how long each task took to complete. We can observe that the system can handle about 550 000 records/second/node or about 30 MB/second/node.

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

Stallion Journal for Multidisciplinary Associated Research Studies
ISSN (Online): 2583-3340
Volume-3 Issue-5 || October 2024 || PP. 54-67 https://doi.org/10.55544/sjmars.3.5.7

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.

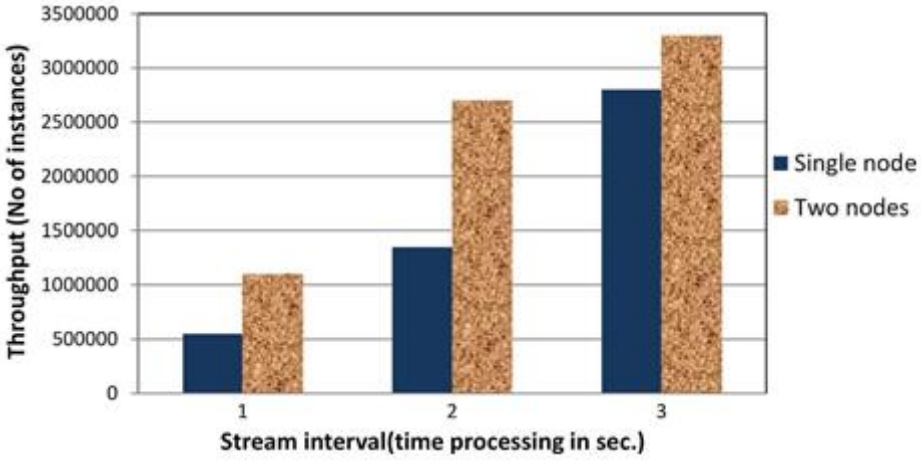

Figure. 12 Performance evaluation: processing time

V. 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 data events from different diseases that were streamed into Spark. The system processes important health data events using the Spark streaming API, applying DT to calculate health status, alerting caregivers, as well as storing the information in a distributed database. Healthcare data analytics plus stream reporting will be carried out by querying the stored result.

Using conventional analytical tools, building a disseminated as well as real time healthcare analytics system is exceedingly difficult and expensive. It takes a variety of expertise, expensive programs, a lot of time, and money. However, the same work may be easily accomplished by leveraging effective open source big data technology and data mining approaches. The same approach may predict additional diseases with little adjustments and can be expanded to other domains. In order to better respond to user requests, we intend to integrate actual data sources into our system in the future, including data from mobile devices, sensors, and social media.

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, char- acteristics 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 health- care 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 architec- ture 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 confer- ence 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 communi- cation 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 Io
[26] T and big data for public health. J Med Syst. 2016;40(12):283.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] Apache kafka. https://kafka.apache.org. Accessed 15 Dec 2017.
[29] 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.
[30] Quinlan JR. C4. 5: programs for machine learning. Amsterdam: Elsevier; 2014.