PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
Ministry of Higher Education and Scientific Research

UNIVERSITY ECHAHID HAMMA LAKHDAR - EL OUED

FACULTY OF EXACT SCIENCES
Computer Science department

Thesis
submitted in partial fulfillment of the requirements for the degree of

ACADEMIC MASTER

Field: **Mathematics and Computer Science**
Option: **Computer Science**
Specialty: **Distributed Systems and Artificial Intelligence**

Presented by:

- Hicham lehouedj
- Ali chekima

IOT DATA ANALYTICS ARCHITECTURE FOR HEALTHCARE BASED ON CLOUD COMPUTING

Examination Commitee:

M. Bali Mouadh	MAA	Framer
M. Beggas Mounir	MAA	President
M. GHARBI Kaddour	MCA	Examinator

Academic year: 2021-2022

Thanks

First of all we thank our God who gave us the strength and the will to develop this work.

We extend our sincere thanks to our supervisor Mr. “Bali Mouadh”.

Who was responsible for providing advice and guidance For each young and old in this dissertation We extend our sincere thanks to all the teachers of the computer science department University Echahid Hamma LakhdarEl-Oued . Also to our colleagues from the 2021-2022 promotion. We also thank all those who participated directly or indirectly in the development of this work.

Abstract

The revolution of the Internet of Things has become in a few years the future of IT players such as: Smart Grid, home automation, connected health, industry, logistics, urban planning and agriculture. Because of this revolution, connected objects generate a large volume of data to collect, monitor and analyze to ensure their proper functioning and optimize the user experience as proactively as possible. The ability to analyze this data is a crucial differentiating point for all IoT solutions, the classic approaches (of which the most used today are Thresholds Algorithms, Data Mining and Metaheuristic Algorithms) aimed at traditional data processing result in failures because these tools are not developed to meet the challenges of the IoT. The use of technologies based on AI methods (such as: Machine Learning) is a new challenge in IoT systems and makes it possible to exploit huge amounts of data and involves constantly pushing the capabilities of treatment.

As part of this work, we propose a data analysis tool architecture for a monitoring system for the Internet of Things platform. This tool should use AI methods to supervise connected objects and provide intelligent prediction and automated evaluation of events and anomalies related to data flow. They used Apache Kafka to collect data flow and Apache Spark framework to process big data and build a predictive machine learning model based on the Random forest algorithm that works in spark.

Keywords: Internet of Things, Connected Object, Sensor Networks, IoT Analytics, Machine Learning, Apache Kafka, Apache Spark, Random forest algorithm.

Résumé

La révolution de l'internet des objets est devenue en quelques années le futur des acteurs de l'IT tels que : Smart Grid, domotique, santé connectée, industrie, logistique, urbanisme et l'agriculture. A cause de cette révolution, les objets connectés génèrent une masse importante de volume des données à collecter, surveiller et analyser pour s'assurer de leur bon fonctionnement et optimiser l'expérience de l'utilisateur de façon aussi proactive que possible. L'aptitude à analyser ces données est un point différenciateur crucial pour toutes solutions IoT, les approches classiques (dont les plus utilisées aujourd'hui sont Thresholds Algorithms, Data Mining et Algorithmes métaheuristiques) ayant pour objectif un traitement traditionnelle des données se traduisent par des échecs car ces outils ne sont pas développés afin de répondre aux enjeux de l'IoT. Le recours aux technologies basées sur les méthodes de l'IA (tels que : Machine Learning) est un nouvel enjeu dans les systèmes IoT et permet d'exploiter d'énormes quantités de données et implique de repousser sans cesse les capacités de traitement.

Dans le cadre de ce travail, nous proposons une architecture d'outil d'analyse de données pour un système de surveillance de la plateforme Internet des Objets. Cet outil devrait utiliser des méthodes d'IA pour superviser les objets connectés et fournir une prédiction intelligente et une évaluation automatisée des événements et anomalies liés au flux de données. Ils ont utilisé Apache Kafka pour collecter le flux de données et le framework Apache Spark pour traiter le Big Data et créer un modèle d'apprentissage automatique prédictif basé sur l'algorithme Random Forest qui fonctionne dans Spark.

Mots clés : internet des objets, objet connecté, Réseaux de capteurs, IoT Analytics, Machine Learning, Apache Kafka, Apache Spark, algorithme de forêt aléatoire.

الملخص

أصبحت ثورة إنترنت الأشياء في غضون سنوات قليلة مستقبل لاعبي تكنولوجيا المعلومات مثل: الشبكة الذكية ، والأتمتة المنزلية ، والصحة المتصلة ، والصناعة ، والخدمات اللوجستية ، والتخطيط الحضري والزراعة. بسبب هذه الثورة ، تولد الكائنات المتصلة حجمًا كبيرًا من البيانات لجمعها ومراقبتها وتحليلها لضمان عملها بشكل صحيح وتحسين تجربة المستخدم بشكل استباقي قدر الإمكان. تُعد القدرة على تحليل هذه البيانات نقطة تفاضل مهمة لجميع حلول إنترنت الأشياء ، والنهج الكلاسيكي (الأكثر استخدامًا اليوم هو خوارزميات العتبات ، وتعدين البيانات ، وخوارزميات الميتاهوريستية) التي تهدف إلى معالجة البيانات التقليدية تؤدي إلى الفشل لأن هذه الأدوات لم يتم تطويرها لمواجهة تحديات إنترنت الأشياء. يعد استخدام التقنيات القائمة على أساليب الذكاء الاصطناعي (مثل: التعلم الآلي) تحديًا جديدًا في أنظمة إنترنت الأشياء ويجعل من الممكن استغلال كميات هائلة من البيانات وينطوي على دفع قدرات العلاج باستمرار.

كجزء من هذا العمل ، نقترح بنية أداة تحليل البيانات لنظام مراقبة لمنصة إنترنت الأشياء. يجب أن تستخدم هذه الأداة أساليب الذكاء الاصطناعي للإشراف على الكائنات المتصلة وتوفير تنبؤ ذكي وتقييم آلي للأحداث والحالات الشاذة المتعلقة بتدفق البيانات. استخدموا Apache Kafka لجمع تدفق البيانات وإطار عمل Apache Spark لمعالجة البيانات الضخمة وبناء نموذج تعلم آلي تنبئي يعتمد على خوارزمية Random Forest التي تعمل في Spark.

الكلمات الرئيسية: إنترنت الأشياء ، الكائن المتصل ، شبكات الاستشعار ، تحليلات إنترنت الأشياء ، التعلم الآلي ، أباتشي كافكا ، أباتشي سبارك ، خوارزمية الغابة العشوائية.

Contents

List of Figures

General introduction

The past few years have been marked by a growing quantitative explosion in digital data, which has led to the creation of a new type of data called big data; A general term for a large set of data. This phenomenon can bring many advantages in all areas, namely: industry, commerce, marketing, etc. It may generate other problems, in particular, with the storage and processing of this data.

The Internet of Things has played an important role in increasing researchers' interest in the field of big data, especially as it is considered one of the most important sources in generating big data.

To design a big data processing system we will address four main challenges: First, data is generated very quickly by various medical devices. Second, due to the huge volume of heterogeneous data, it is difficult to collect data from distributed sites. Third, storage is the main problem for large and heterogeneous data sets. Big data system needs storage while providing performance guarantee. Fourth, it is about big data analytics, more precisely the real-time or near-real-time mining of large data sets that includes modeling, visualization, forecasting, and optimization. These challenges require a new processing model because current data management systems are not effective in dealing with the heterogeneous nature of data or in real time. These traditional systems do not provide any support for unstructured or semi-structured data. From the perspective of scalability, when the data volume grows, there are many failures of the traditional RDBMS in scaling for parallel device management and fault tolerance, which is not suitable for additional data management.

The content of this message is organized as follows:

Chapter 1: Presenting the concept of the Internet of Things, its characteristics, structure, challenges, applications, and the technology used in communication and its integration with the cloud and the data it generates.

Chapter 2: Presenting the concept of big data, the method of analyzing it, the algorithms used in the analysis, innovative solutions to its problems, and the frameworks used to treat it.

Chapter 3: Formulation of the problem to be addressed, related work and proposed solution to address the problem at hand.

Chapter 4: We review the different stages of implementing and achieving our work and the results obtained.

Chapter 1

Internet of Things (IoT)

1.1 Introduction

One of the most important words and terms that have spread recently in the world of technology is the term Internet of Things. It is also referred to as the Internet of Objects. It is a widely distributed network of things. These things can be different physical forms with unique identifiers (UIDs), and these things can communicate with each other without any direct human interaction. This not only gives us control over things around us, but also gives us insight into the state of things, as things or devices can transmit, share, and use data from the physical environment. The Internet of Things aims to unite everything in our world under a common infrastructure. So that real-world objects are converted into virtual smart objects. The Internet of Things offers a lot of Services for multiple sectors such as education, safety, health, industry, agriculture...etc. For example, with the help of the Internet of Things, remote patient monitoring can be improved. In addition, the Internet of Things is known to be used in our daily lives. Thus, we can say that the Internet of things has gradually brought a sea of technological changes in our daily life, which in turn helps in making our lives simpler and more comfortable, through various technologies and applications. We find that sensors are one of the applications of the Internet of Things that can be used in daily life such as safety and Home Improvement Housekeeping.

1.2 Architectures

The main problem with the Internet of Things is that it is too broad and a broad concept, and there is no proposed unified architecture. For the idea of the Internet of Things to work, it must consist of a variety of sensors, networks, communications, and computing technologies, different IoT architectures have been proposed. Depending on the type of application.

1.2.1 Three Layer architecture of IoT

- The perception layer is also called the physical layer. This layer contains sensors, RFID tags, and other essential components. This layer senses and collects necessary information from connected devices [13].

- The network layer acts as a gateway. This layer is intended to receive useful information in some form of the digital signal from the perception layer and send it to the processing systems at the middleware layer through transmission media such as WiFi, Bluetooth, WiMAX, Zigbee, GSM, and 3G etc with protocols like IPv4, IPv6, MQTT, DDS, etc [13].

- The application layer is the top layer. It is responsible for transmitting the data to the required destination. Figure 1.1 shows the 3-layer model of IoT [13].

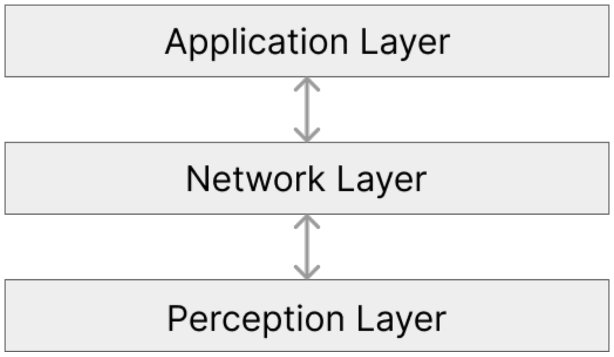


Figure 1.1: 3-layer model of IoT.

The main disadvantage of this architecture is more functions Scheduled in a single layer, since updating a single layer or multiple layers is a difficult task.

1.2.2 Cloud Based Architecture

The cloud-based IoT architecture has 4 layers, as shown in Figure 1.2 [28]

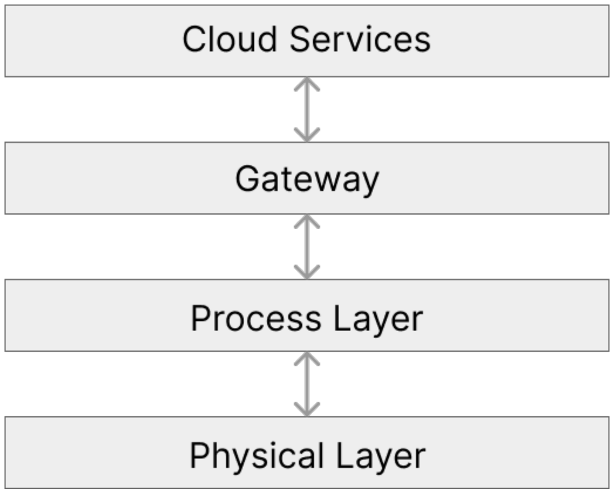


Figure 1.2: Cloud based architecture of IoT

- The physical layer contains technologies used, such as RFID, whose task is to collect data from connected devices
- The process layer whose task is to analyze the information received by it

- The gateway layer contains network information like LAN or WAN etc. You transfer data to cloud services.
- The core part of a cloud-based architecture is cloud services. It is responsible for processing data collected from various connected devices using data analysis algorithms. The main components of cloud services as shown in Figure 1.3 [28].

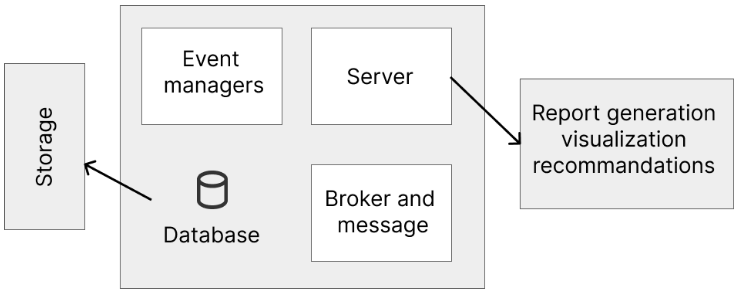


Figure 1.3: cloud services

1. The broker and message queue are responsible for managing incoming messages. It simplifies and handles it and helps increase the scalability of the network (for example, the number of devices can be increased.)
2. The database is used to store data.
3. The server helps in arranging the data, processing it, and preparing reports. It also helps the user to understand them.
4. Event managers handle the event. So that it implements high priority interrupts and takes certain instantaneous actions such as fire alarms. In case of emergency.

1.2.3 Fog based architecture

Fog computing is one of the latest technologies presented. Expands the features of cloud computing. As fog computing provides decentralized data processing and storage in contrast to cloud computing [28].

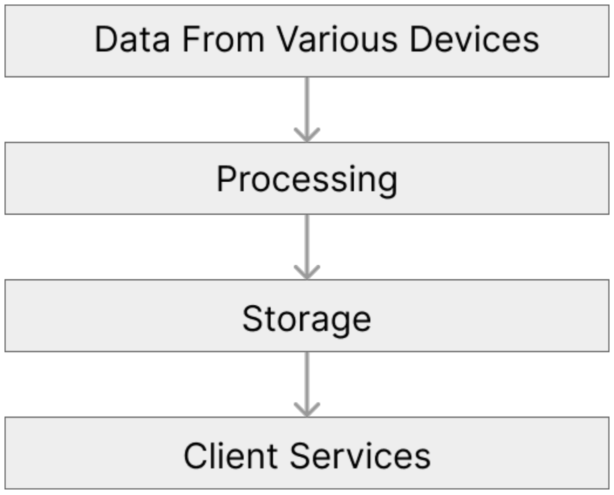


Figure 1.4: Fog based architecture

1.2.4 Service Oriented Architecture of IoT

This reference model is generally used in business applications [28].

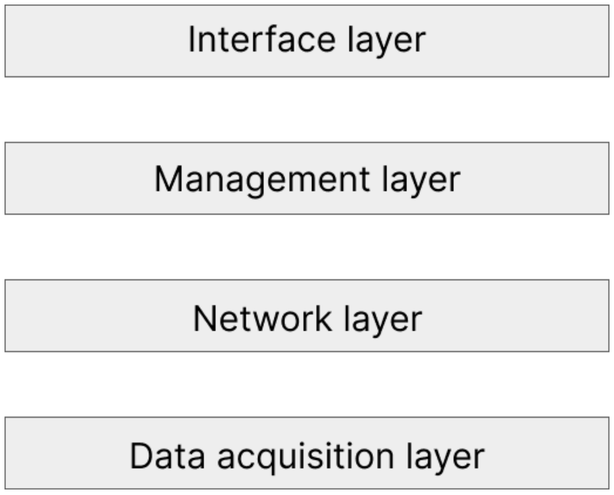


Figure 1.5: SOA for IoT

- The data acquisition layer is similar to the perception layer. It transmits sensing data to the network layer. It is also responsible for data processing (using data computing algorithms).
- The network layer as discussed earlier.

- The management layer or the so-called business layer, one of its tasks is to verify the transfer of data to the other party, and one of its tasks is also the integrity and security of the data and its transmission in the correct manner.
- The interface layer or the so-called application layer, one of its tasks is to coordinate data and present it to users, and it also allows them to interact with the data easily.

1.3 Technologies Of IoT

The main technologies of the Internet of Things are categorized into two parts [21].

1. identification technology
2. Communication technology.

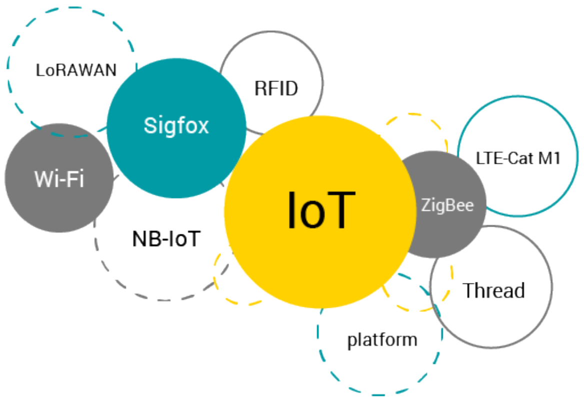


Figure 1.6: Technologies Of IoT

1.3.1 Identification Technology

These types of technologies are used to identify monitoring purposes. Examples: RFID, WSN, QR Code, Barcodes, Smart Sensors, etc [28].

1.3.1.1 Radio Frequency Identification (RFID)

Radio Frequency Identification (RFID) is a system that transmits the identity of an object or person wirelessly using radio waves in the form of a serial number. The main components of RFID are the tag, reader, antenna, access controller, software, and server. It is more reliable, efficient, safe, cheap, and accurate. RFID has a wide range of wireless applications such as distribution, tracking, patient monitoring, military applications, etc [52].

1.3.1.2 Wireless Sensor Networks (WSN)

A WSN is a wireless network consisting of independent, spatially distributed devices that use sensors to collaboratively monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, movement, or pollutants. There are different types of topologies for configuring a WSN, including stellar topologies, lattice topology, and mesh hybrid stars [57].

1.3.2 Communication Technology

These technologies are used to transmit data, examples include e Zig bee, Z wave, MQTT, Bluetooth, Li-fi, Wifi, Near Field Communication (NFC), HaLow, and power line area network, we will touch on some of them [28].

ZigBee is a protocol developed to improve the features of wireless sensor networks. ZigBee is characterized by low cost, low data rate, relatively short transmission range, scalability, reliability, and flexible protocol design. It is a low-power wireless network protocol based on the IEEE 802.15.4 standard. ZigBee has a range of about 100 meters and a bandwidth of 250 kbps. It is used in home automation, digital agriculture, industrial controls, medical monitoring, and power systems [52].

Bluetooth wireless technology is an inexpensive, short-range radio technology, generally used in our everyday applications. For example, in our smartphone using Bluetooth, we will transmit information to the paired device. It follows the IEEE 802.15.1 standard [52].

Wi-Fi follows the IEEE 802.11 standard. It is a networking technology that allows computers and other devices to communicate over a medium-range wireless signal, generally used in a local area network [52].

Near Field Communication (NFC) is a short-range (about 4 cm) networking protocol. It provides a point-to-point connection between communication devices. NFC technology makes life easier and more convenient for consumers by facilitating transactions, exchanging digital content, and connecting electronic devices with a single touch [52].

1.4 Applications of IoT

Most of the daily life applications that we usually see are smart but unable to communicate with each other and enabling them to communicate with each other and exchange useful information with each other will create a wide range of innovative applications. All this is due to the concept of the Internet of Things. There are many applications currently in the market and work is underway on future applications that could be of great use. We present some of these apps. (Smart Homes, Smart

City, Self-driven Cars, IoT Retail Shops, Farming, Wearables, Smart Grids, Industrial Internet ... etc.) for example :

- **Smart Homes :** The term smart home refers to the use of information and communication technology to control the home, from controlling appliances to automating home features (windows, lighting, etc.). Using a central hub generally, a smartphone (containing sensors such as an accelerometer) will be connected to the smartphone using NFC, Bluetooth, Zigbee, or other short-range low-energy protocols. One of the essential elements of a smart home is the use of smart energy scheduling algorithms, which will provide residents with the ability to make optimal advanced choices about how electricity is spent to reduce energy consumption. Another commonly used term is smart home or home automation [10].
- **Health care :** Nowadays, the concept of smart healthcare has become very important, as it can be through integrating sensors and actuators in patients and their medications for monitoring and tracking purposes. IoT also uses clinical care to monitor the physiological conditions of patients through sensors by collecting and analyzing their information and then sending remotely analyzed patient data to treatment centers to take appropriate action. For example, Masimo Radical-7 monitors the patient's condition remotely and informs the medical staff about it. Also, in case of accidents, someone has to head to the hospital to get an ambulance, but in the case of IoT whenever there are accidents, the wearables automatically give a signal to the nearest Wi-Fi router, and then the hospitals get the ambulance, depending On health conditions such as heart rate..etc [7].

1.5 Cloud of things

It is the integration of both cloud computing and the Internet of Things, now called ”CoT” which stands for ”Cloud of Things”. This term emerged as a result of the rise in the number of devices connected to the Internet of Things, with the high volume of data, and because local temporary storage is not possible. That's why the Internet of Things has been combined with cloud computing resulting in the Cloud of Things (CoT) or CloudIoT. Cloud of Things has helped the Internet of Things (IoT) a lot in data processing/analysis and has created more utility from data generated by IoT by allowing more advanced smart applications to be developed [21].

1. **Integration Engines** Since the Internet of Things and cloud computing can complement each other, there are three categories to generate this integration:
   - **Communication** The main advantage of the integration of cloud services and the Internet of Things is the process of sharing data on the Internet of Things, and the data

is transferred at the lowest cost, and this feature also provides access to anything and operates it whenever they want.

- **Storage** Day by day, the sources and amount of data in the Internet of Things are increasing, and this amount of data requires rapid processing and sharing with various other devices. The cloud is the most effective solution to provide unlimited, low-cost virtual storage on demand for data generated by the Internet of Things.
- **Computation** Data is processed in the Internet of Things in a complex and unlimited manner, and this is not provided by local processing, unlike the cloud.

2. **Architecture design for CoT** The CoT architecture allows the dynamic addition of IoT devices as shown in Figure 1.7. This architecture provides a framework that facilitates the integration of different components and layers to harness the power of cloud computing to process, manage, and transform information related to the IoT environment.

   - **sensing layer** This layer senses and collects necessary information from connected devices.
   - **communication layer** This layer is considered basic in this architecture as it represents the access network responsible for maintaining the integrity of communications between the various elements of the Internet of Things, including sensors, objects, and humans. which could be for example Wireless Sensor Networks (WSN), Body Sensor Network (BSN), or Net Generation Networks (NSG). The network layer consists of gates receiving and integrating unstructured data coming from the sensor layer and then passing it on to the control layer It can be stored and processed by cloud computing resources.
   - **control layer** This layer is responsible for storing and processing transmitted data. Where it provides a range of services for the Internet of Things. Such as hosting, monitoring, and device management.
   - **actuation layer** This layer is responsible for representing the information received from the control layer by making it in a form that can be used to perform actions in response to input from the sensor layer.

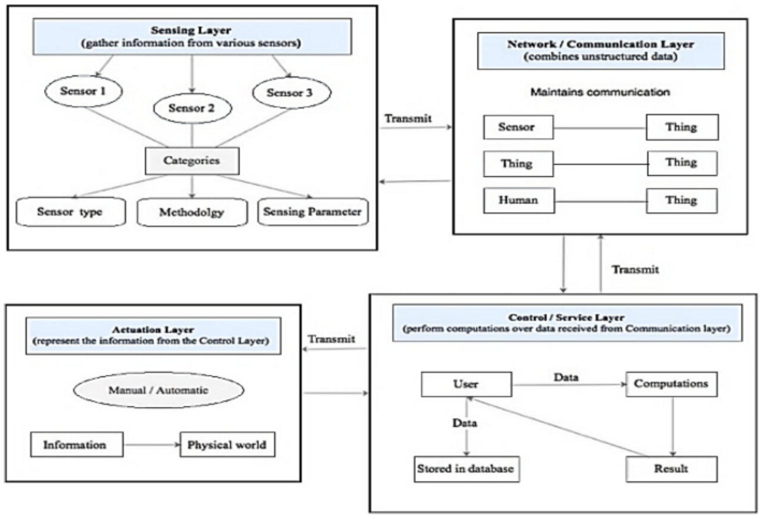


Figure 1.7: CoT architecture

1.6 Challenges

Despite the endless benefits that the Internet of Things brings us, it also makes our lives easier. However, there are many challenges and obstacles that we face while relying on these technologies nowadays. Since the Internet of Things includes objects connected to the Internet, this allows devices to be vulnerable to hacking. Devices can cause security risks. Similarly, in the case of hacking, the hacker may take control of the devices which could lead to disaster. For example, in the case of traffic lights and Other urban applications of the Internet of Things. In general, there are several security challenges to consider when deploying this technology. Examples of these challenges are as follows :

1.6.1 Scalability

Since there are a huge number of connected devices in IoT, these devices are usually connected in hierarchical subdomains rather than a network, the complexity of this architecture is an obstacle to scalability [38].

1.6.2 Security

Security is key in IoT as well as in the cloud if both IoT and the cloud are integrated. They are expected to be more vulnerable to various cyber attacks, such as session riding, SQL insertion, cross-site scripting, and side-channeling, as well as many vulnerabilities, such as session hijacking and virtual machine escape [21].

1.6.3 Privacy

Sensitive and important user data is collected on the Internet of Things. It is collected from the devices that we use in our daily life like phones, smart watches, home appliances, sensors...etc. This information describes the user in a great and accurate way and owning it puts us in front of a great challenge, which is to preserve it from any misuse [33].

1.6.4 Connectivity

Connecting multiple devices is a huge challenge in the Internet of Things. These devices must work together to function properly, even if these devices operate in different areas. The lack of common connectivity, standard data formats, and common software interfaces complicates IoT implementation. Another challenge is the cost of retrofitting or replacing traditional devices to work with the latest technology [38].

1.6.5 Identity and Access Management

Identity investigators face some issues when authenticating, managing, and maintaining IoT devices due to heterogeneity and the large number of IoT devices connected to the same IoT system. The identity validator is responsible for pre-registering IoT devices so that they can use the infrastructure authentication mechanism. IAM is about the unique definition of things [52].

1.7 Data in the Internet of things

Nowadays, data is generated by humans, and also this data is generated from various sources such as mobile phones, smart sensors, vehicles, and smart equipment. Together, these entities make up the Internet of Things. This data can be transmitted, aggregated, analyzed, and stored, which will facilitate user behavior and result in more sustainable and resilient applications.

1.7.1 Type of data

The data in the Internet of Things can be both low-level raw data and high-level general data. This difference and heterogeneity in the data and the dynamic change of the data leads to more uncertainty in the data. For example, the data in the Name field can be an acronym, a full name, or a name and last name. This is inconsistent [53].

1.7.2 Data Characteristics of IOT

1.7.2.1 Polymorphism heterogeneity

Applications in the Internet of Things include different types of data from different sources, and these data can be linked to each other. It may be physical, chemical, biological data, etc., for example: logistics, transportation control and monitoring systems, positioning and roadmap of things from the Global Positioning System (GPS), health monitoring systems evaluate, patient condition by taking biological data such as Blood pressure, heart rate, facial expressions, voice, chemical data on humidity, lighting, oxygen consumption, etc. This data can be structured or unstructured such as images, video, audio, or it can be digital data [63].

1.7.2.2 Massive scale

The more devices connected to the Internet, the greater the amount of data may exceed trillions of data in real-time. This data needs large storage spaces and huge systems for processing [53].

1.7.2.3 Rich semantic

The data in the Internet of Things depends on space and time, such as the data we take from home health care devices or geographic information systems. These applications generate different raw sensory data that are deployed on highly distributed, heterogeneous, resource-limited devices that are interconnected and connected in different scenarios independently. The principle of automated information communications and interactions in the Internet of Things is to make data descriptions reliable so that machines and software agents can accurately process and interpret the data [53].

1.7.2.4 Timeliness of Data

The data is sensed in real time because the state of things depends on changes that occur at different times regularly. Historical data is only to record the transaction development process, new data can only reflect the current state of things the system sees, and therefore response speed or response time system is the key to the reliability and operation of the system. This requires that the software IOT data processing system must have sufficient operating speed, otherwise, it may lead to wrong conclusions and even cause huge losses [63].

1.7.3 Sources of data

Data is collected from various sources in the Internet of Things [53].

1.7.3.1 RFIDs:

We have already touched on it in IoT technologies, and it is possible to associate these tags with everyday things.

1.7.3.2 Addresses/Unique Identifiers:

Every object on the Internet of Things has a unique IP address. There are two versions of it: ipv4 and ipv6, ipv6 is an extension of ipv4 that can accommodate a large number of addresses

1.7.3.3 Metadata about objects, processes, and systems:

Metadata should be structured or structured information about an object such as its source, scope, physical or digital properties, context, or any other details about the object itself. The data and metadata recorded in the objects are the driving force of the Internet of Things.

1.7.3.4 Positional Data and Pervasive Environmental Data:

We obtain location data from GPS or local positioning systems. These devices include satellites, wifi access points, etc., which gives us accurate tracking of both stationary and moving objects. For a widespread site, information about the environment is modestly available for interactions with surroundings to thrive in the so-called ”Internet of Places”

1.7.3.5 Sensor Data:

A large amount of data is quickly and accurately captured by the sensors. In catastrophic situations, such as temperature sensing, pressure data, etc., we often use wireless sensor networks.

1.7.3.6 Historical Data:

Over time the volume of recorded data increases so that the data is collected as history. This will undoubtedly help us in making decisions and planning business. Therefore, this data must be managed well [8].

1.7.4 Data Management Applications

The figure shows how to manage data in the Internet of Things in a simplified manner [53].

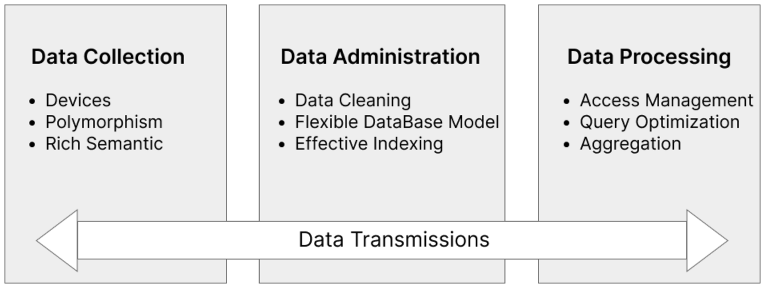


Figure 1.8: Basic Concept of IoT Data Management

1.7.4.1 Data Collection

We can rely on collecting data on the Internet of things from several different sources, the main ones being RFIDs, GPS and NFC, which we already touched on in the explanation. As long as the phone battery is alive. The phone's camera can also help us sense data in the form of images, we can also use the microphone, we can also geolocate...etc [45].

1.7.4.2 Data Administration

Data management is represented in the Internet of Things, whether this data is in a relational database, document-oriented, knowledge base, etc., so that it is considered as an intermediary store for managing and analyzing this big data, and this is done in the following form [45] :

- **Cleaning** The concept of data cleaning is to remove errors and inconsistencies from the data to improve its quality of this data. In the case of the Internet of Things, data is generated in real-time and from multiple sources, and this makes us need to clean this data to improve its quality [45].

- **Flexible** Database Model IoT data storage can be specified as SQL (Structured Query Language) or NoSQL (Unstructured Query Language). SQL: is a query language used in a relational model where data is organized into relationships so that each data record is represented by a table consisting of rows and columns. NoSQL: It differs from SQL in that it does not break data into relationships, and does not use SQL to communicate with the database. It is also dedicated to the large and heterogeneous volume of data, and this is what is focused on in the uses of the Internet of things for databases [45].

- **Effective Indexing**

  In databases, most transactions are queries. Thus, efficient indexing is required to query large data very quickly. Although index storage is very expensive, both in storage and retrieval, dynamic temporal and spatial indexing is an essential step. The suggested techniques in indexing are Service Discovery (DS) and the ”Query and Update Efficiency Index” (UQE Index). DS aims to improve the efficiency of rapid discovery because the search is done in local or central (although large) indexes. UQE aims for high input throughput capabilities and efficient multidimensional querying [45].

1.7.4.3 Data Processing

Processing, analyzing, and sorting data usually requires heavy and complex computations. This leads to excessive power consumption. Various technologies and components involved in data processing often use especially dedicated protocols to exchange data between different systems. The next section describes the basic techniques for dealing with data representation, interfaces, and interoperability [53].

- **Access Management** IoT applications rely on ”big data” where previously SQL was the de facto standard for accessing data. SQL is usually the base process for performing standard select/drop/join/group operations or nested operations for complex queries. TinySQL and TrikiDB 27 are examples of combinations to accommodate new forms of data access. So they represent query processing systems, which have been successfully used in wireless sensor networks (WSN). These query processing systems provide a SQL query interface for extracting data from sensors and have proven to be a convenient programming interface in these environments [53].

- **Query Optimization** One of the things of great concern in IoT is the number of incoming messages and the way large data is fetched, so query optimization in IoT is a goal, and this is done by defining an effective execution plan for query evaluation. . After the query is parsed, the parsed query is passed to the query optimizer, which generates various execution plans to evaluate the distributed query and determine the plan at the lowest estimated cost. Some comprehensive solutions attempt to improve query handling. It is a very efficient process for data deployment and query processing based on two scheduling mechanisms, wave scheduling, and tree scheduling. The first was a class of simple activation tables and associated routing protocols that achieved scalability and Energy is efficient with a modest delay penalty. The latter implements a tree structure to route messages from sensor nodes to a specific server. There is currently a new model for large-scale data analysis called MapReduce. MapReduce is bundled with rich features, high scalability, accurate fault tolerance, and easy programming [53].

- **Data Aggregation** The idea of data collection aims to eliminate duplicate data and reduce the number of transmissions by combining the most important data received from different sources. It also allows data to be made available in an energy-efficient manner with minimal latency. Other important measurements of aggregation algorithms include network age, data accuracy, and response time. Through the aggregation process, a potential loss of accuracy may occur because the underlying detailed data may be discarded. However, data aggregation and merging are still required for certain applications [53].

1.8 Conclusion

The Internet of Things has gradually brought a sea of technological changes into our daily lives, which in turn are helping to make our lives simpler and more convenient, through various technologies and applications. There is an infinite benefit to IoT applications in all fields including medical, manufacturing, industrial, transportation, education, governance, mining, etc. In this chapter, we have discussed the different structures of the Internet of Things, as well as its applications, fields, communication technology, and the challenges facing it. We also discussed how to integrate the Internet of Things with the cloud. And how the Internet of things deals with data from storage, processing, and management. In the next chapter, we will discuss ways to deal with big data, how to manage it, analyze it, and how to deal with it.

Chapter 2

Big data analytics

2.1 Introduction

Information technology has become an essential element of our lives and as such it generates a huge volume of data that is exchanged over the internet today. According to William [58], approximately 45% of the world population, or 3.5 billion people use social media, with the average user spending approximately 3 hours of their day generally on social media, and with this large amount of data, It becomes difficult to process it, unlike its creation, which is very easy, even with the development of computer systems, it is still very difficult to analyze this data.

In the presence of problems in the analysis of big data, many methods have appeared, including distributed computing. These methods are frequently used to improve the analysis process in record time and with higher efficiency.

Despite all this development in computing systems, the problems of dealing with big data still require more effective methods. This is what made Fisher et al. [22] point out that the data cannot be processed, because data mining or data analysis methods will not be able to deal with this huge data. Lani [19] also presented a definition that explains big data called (3Vs), which is: volume, speed, and diversity. This means that the data is large, generated quickly, and will be more diverse and different. Later, other studies [42, 32] indicate that the definition of 3Vs lacks honesty, validity, value, diversity, place, vocabulary, and ambiguity so that the explanation of big data is more accurate.

Due to the importance of big data, it is now being marketed in huge quantities, and these quantities are constantly increasing. The EMR Marketing website [24] indicates that the market will witness healthy growth in the forecast period 2022-2027 to reach 450 billion US dollars by 2026. All these numbers indicate that the field of big data analytics will grow unreasonably.

The field of big data was not limited to marketing, it has entered strongly in the field of disease control and prevention, the field of business intelligence and the smart city. This indicates the great importance researchers attach to developing effective methods and techniques for analyzing big data.

2.2 Definition of big data

In the last two decades, the world has witnessed an unprecedented interest in data and its production, as the year 2002 is considered the beginning of this boom. In fact, the transition from analog storage devices to digital storage devices and the Internet of Things has greatly expanded the capacity of data collection, which led to the era of big data [12].

The term ”big data” first appeared in the 1990s and commonly refers to data sets whose size exceeds the ability to process and analyze within reasonable time limits. However, the expression does not mean any specific storage size, but it has a deeper meaning than that. Big data includes a wide range of data sources including structured, semi-structured, and, for the most part, unstructured data. Although there have been multiple definitions over the years to the concept of big data, they share the definition of 5 Vs (Volume, Variety, Velocity, Value, Veracity) [12]:

- Volume: The actual amount of data generated is huge, within the size of terabytes and petabytes. In general, it refers to the volumes that are too large and complex to exploit traditional data storage and processing techniques.
- Variety: The data may come in many types of data and from a variety of sources. These include sources like sensors, social media, log files, and more, plus they include heterogeneous formats like text, images, audio, video, etc.
- Velocity: Data is generated and/or processed at high rates, almost in real time.
- Value: Data must carry valuable information that, if properly analyzed, brings business value and profitable insights. In the scientific context, this means information that contributes to the advancement of human knowledge.
- Veracity: Data sources must be reliable and generate high-quality data that can produce value.

However, the community has not reached complete agreement on the definition of big data, with some authors suggesting moving its description from intrinsic properties to approved technologies for data acquisition, storage, sharing and analysis [12].

2.3 Data analytics and data mining

The increase in the exponential time in the data made it difficult to obtain useful information from that data. Traditional methods showed a lot of performance; However, their predictive power is limited because traditional analysis deals only with primary analysis, while data analytics deals with secondary analysis. Data mining is the drilling or extraction of data from many dimensions or viewpoints through data analysis tools to find previously unknown patterns and relationships in the data that can be used as valid information; Moreover, it uses this extracted information to build a predictive model. It has been used extensively and widely by many organizations.[12]

Data mining is not a magic wand in fact it is a giant tool that does not discover solutions without guidance. Data mining is useful for the following purposes:

- Exploratory analysis: Examines data to summarize its main characteristics.
- Descriptive modeling: dividing data into subgroups based on their characteristics.

- Predictive modeling: forecast information from existing data.
- Pattern Detection: Discover patterns that occur most frequently.
- Retrieval by content: discover hidden patterns.

Big data and machine learning have great potential to use data and analytics systematically to discover an interesting pattern that was not previously known and reveal the shortcomings of big data stores in order to build predictive models of best practices that improve the quality of services, as well as reduce the cost. [12]

2.4 Big Data analytics

The size of the data is no longer the only problem in our time, but the different types of data, including the data flow. This is due to the characteristics of big data: massive, high dimensional, heterogeneous, complex, disorganized, incomplete, noisy, and erroneous. Thus, they call for the need to change the approach to statistical and data analysis. Although it seems to us at first that big data enables us to collect more data and thus find more important information, this does not necessarily mean more important information. It may lead to more ambiguous or abnormal information. [15]

Big data is generated by many sources, including: mobile devices, the Internet of things, social networks, and many other new applications and devices that share the characteristics of size, speed and diversity, and therefore data analytics must be re-examined from another point of view [15]:

- As for the size, the huge amount of data that is generated, because it is very large, the traditional data analysis method will not work with it. Baraniuk [49] indicated in the example of wireless sensor network data analysis that the bottleneck in big data analytics will be shifted from sensor to processing, communication and storage of sensor data, as shown in Figure 2.1. This is because sensors can collect more data, but when Loading such big data to the upper layer system, it can lead to bottlenecks everywhere.
- As for speed, real-time or streaming data is the problem of receiving a large number of data in a short time, and therefore the system may be unable to process this data.
- As for diversity, diversity is due to the different sources of data used, and this constitutes a problem in how to deal with this data.

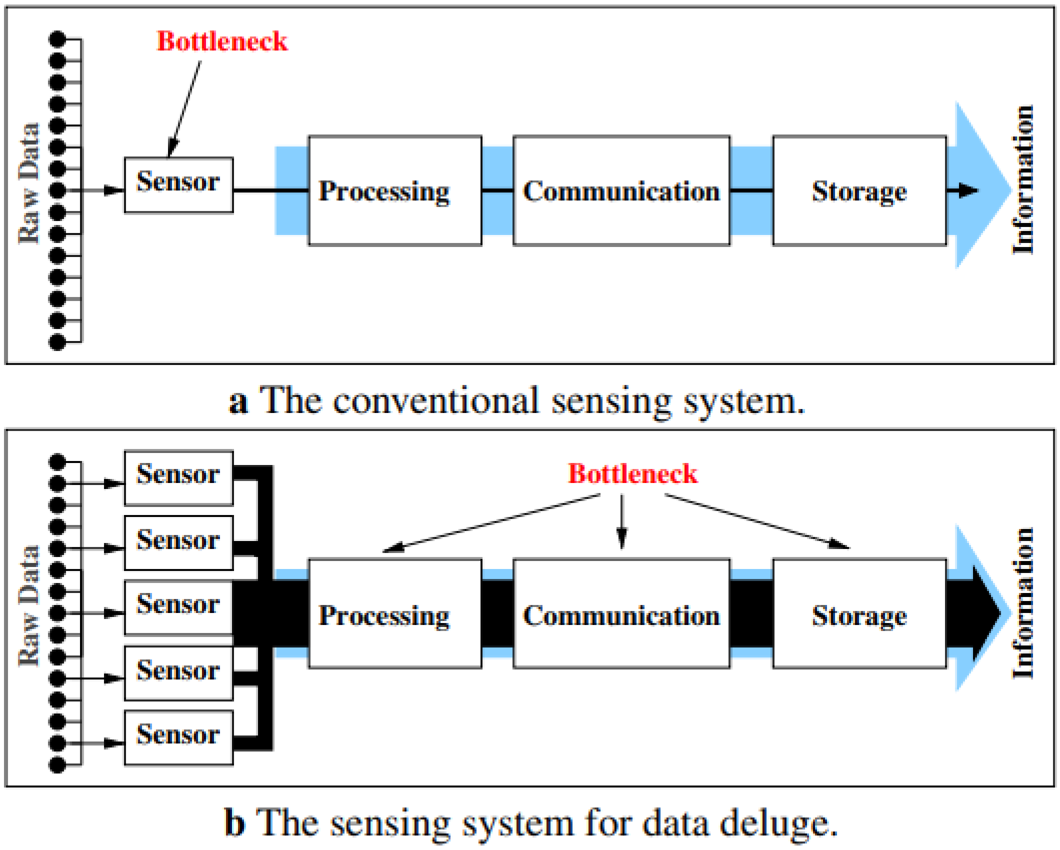


Figure 2.1: The comparison between traditional data analysis and big data analysis on wireless sensor network [49]

2.4.1 Big data input (pre-processing)

The problems of dealing with big data that the system is unable to address is not a new problem, but rather an old topic and led to the emergence of many early methods [60, 4, 40], and for example we have, marketing analysis, weather forecasts, and even astronomy analysis. Despite the development of technology, this problem still exists in big data analytics today. This is what calls for the importance of pre-processing the data so that it can be easily dealt with by the computer, platform algorithm and analysis. Despite the effectiveness of traditional data pre-processing methods with large data at the present time, there is still a study that focuses on reducing the complexity of the input data so that it can be processed more efficiently by the devices. Even with the development of devices, no single device can process the entire input data efficiently in most cases . Using domain knowledge to design a preprocessing operator is a possible solution to big data.[15]

It is logical that reducing the size of the data either by sampling or compressing the data is a solution to reduce the cost of analyzing the data from the computational point of view, and therefore less computation time, especially when the data reaches the system quickly. It is necessary to make the sampling data effectively represent the original data [17], because the number of possibilities for samples to be selected is a research problem in itself [2] and thus will affect the performance of the sampling method.[15]

To ensure the speed of the compression process, Jun et al in [31] attempted to use an FPGA to speed up the compression process. Improving I/O performance in and of itself is a solution to ensuring the speed of the compression process. This appears when Zhou et al. [67] used temporal selection and predictive dynamic selection and switched the appropriate compression method from two different strategies to improve compression performance. There is also the aggregation method, where we divide the input data into groups and then compress these input data according to the resulting groups. As it appears in [62] and [61] where they first collected the input data, it was compressed to improve the performance of the compression process.[15]

In addition to the problems of processing large and fast data entry, data analysis performance may be affected by heterogeneous data, incomplete data, and garbled data. This leads to the need to consider ways to clean the data.[15]

2.4.2 Big data analysis algorithms

2.4.2.1 Mining algorithms for big data

As big data problems have been emerging for nearly twenty years, in 1998 the terms “big data” and “big data mining” were first introduced. Big data and big data mining appearing almost simultaneously made it clear that finding something from big data would be one of the main tasks in this field of research. Data mining algorithms for data analysis also play a vital role in big data analysis, in terms of computation cost, memory requirements, and accuracy of the final results.[15]

- **Clustering algorithms :** In the age of big data, traditional clustering algorithms will become more limited than before because they usually require all data to be in the same format and loaded into the same machine to find some useful stuff from the whole data. Although the problem of analyzing large-scale and high-dimensional data sets has attracted many researchers from various disciplines in the past century, and many solutions have been presented in recent years, the characteristics of big data still exist until many new challenges to data aggregation issues. Among them, how reducing the complexity of data is one of the important issues of big data aggregation. Big data aggregation has been divided into two categories: single-machine aggregation (that is, sampling and dimensionality reduction solutions), and multi-device aggregation (parallel MapReduce solutions). This means that traditional down-sampling solutions can also be used in the era of big data since the complexity and memory space required for the data analysis process will be reduced by using sampling and dimensionality reduction methods.[15]

- **Classification Algorithms :** Similar to the clustering algorithm for big data mining, many studies have also tried to modify traditional classification algorithms to make them work in a parallel computing environment or to develop new classification algorithms that work normally in a parallel computing environment. The design of the classification algorithm took into account the input data collected by distributed data sources and will be processed by a heterogeneous group of learners. A study introduced a new classification algorithm called "Sort or Transmit for Classification" (CoS). They assumed that each learner could be used to process input data in two different ways in a distributed data classification system. One is to perform the classification function itself while the other is to forward the input data to another learner to be classified. Information will be exchanged between different learners. In short, this kind of solution can be considered as collaborative learning to improve accuracy in solving big data classification problem.[15]

- **Frequent pattern mining algorithms :** Recursive pattern mining algorithms Most of the research on recursive pattern mining (i.e. association rules and sequential pattern mining) focused on dealing with large-scale datasets initially because some of the early methods attempted to analyze data from transaction data from large shopping malls. Since the number of transactions is usually more than 'tens of thousands', problems with how to deal with large-scale data have been studied for several years, such as FP-tree using tree structure to include recurring patterns to reduce computational time for link base mining. In addition to traditional mining algorithms with recurring patterns, of course, parallel computing and cloud computing technologies have attracted researchers in this research field. Among them, a map reduction solution was used to improve the performance of the recursive pattern mining algorithm. Some studies not only used the map-reduction model, but also allowed users to express the constraints of their specific interests in the process of recursive pattern-mining. The performance of these methods using a map reduction model for big data analysis is undoubtedly better than traditional recursive mining algorithms running on a single machine.

2.4.2.2 Machine learning for big data

Designing machine learning algorithms for data analytics is different from designing a data mining algorithm for specific problems. We can use machine learning algorithms for different mining and analysis problems because they are usually used as a "search" algorithm for the desired solution. Because most machine learning algorithms can be used to find an approximate solution to the optimization problem, and therefore can be used for most data analysis problems if we can formulate the problem as an optimization problem. For example, the genetic algorithm, one of the machine learning algorithms, can not only be used to solve the clustering problem [37] but can also be used to solve the recursive pattern mining problem [35]. Machine learning capabilities are not only for solving various mining problems in KDD's data analysis engine; It also has the potential to improve the performance of other parts of KDD, such as reducing features for input operators [40].

The results of recent studies show that machine learning algorithms will be an essential part of big data analysis. Existing machine learning methods for big data analytics are similar to a problem found in most traditional data mining algorithms designed for sequential or centralized computing. However, the best possible solution is to make it work in a parallel computing environment. Some machine learning algorithms can be used primarily for parallel computing (for example, random forest algorithms).

In [36], Kiran and Babu show that the framework for distributed data mining algorithms needs to aggregate information from different computer nodes. As shown in Figure 2.2. The design of most distributed data mining algorithm is: Each algorithm will be executed on a computer node (worker) that has its own locally coherent data, and then the results of each computer node are aggregated and combined into a final model to represent the full knowledge. Kiran and Babo [36] also noted that connectivity will be problematic when using this distributed computing framework.

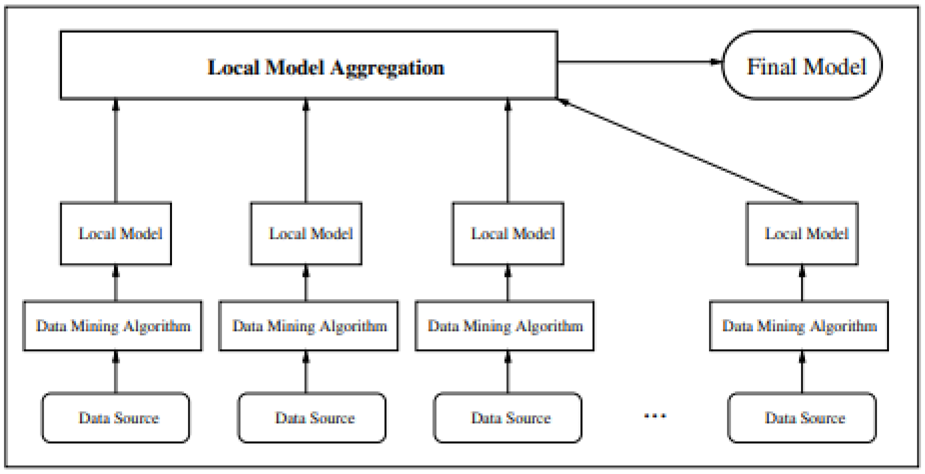


Figure 2.2: A simple example of distributed data mining framework

In [11], Poe et al., found some research problems when trying to apply machine learning algorithms to parallel computing platforms. An example is the first release of the Map Reduction framework that does not support "iteration" (ie recursion). In recent studies [5, 43] I paid more attention to this problem and tried to find a solution to it. Even with solutions to improve the performance of traditional data mining algorithms, one of the possible solutions to improve the performance of a machine learning algorithm is to use a graphics processing unit (GPU) called CUDA, to reduce the computing time for data analysis. In [29] Hassan et al. used CUDA to implement a self-organizing map (SOM) and multiple back propagation (MBP) for the classification problem. The simulation results show that GPU usage is faster than CPU usage. That is, GPU-powered SOM is three times faster than CPU-powered SOM, and GPU-powered MPB is twenty-seven times faster than CPU-powered MPB.

Starting from the above, we can divide machine learning studies for big data analytics into two types: The first type attempts to make machine learning algorithms run on parallel platforms with high efficiency such as Spark [6], Mahout [5], and PIMRU [11]. The second type attempts to redesign machine learning algorithms to make them suitable for parallel computing or parallel computing environments, such as GPU neural network algorithms [29].

2.4.3 Output the result

There are many criteria for evaluating the performance of cloud computing and big data analytics systems, including PigMix [47], GridMix [26], TeraSort and GraySort [54], TPC-C, TPC-H, TPC-DS [56], and yahoo cloud service standard. (YCSB) [16] and in [25] Ghazal et al. introduced another standard (called BigBench) to be a comprehensive big data standard that covers the 3V properties of big data and uses load time, query time, procedural processing query time, and time for residual queries as metrics. Computation time is one of the intuitive metrics for evaluating the performance of various big data analytics systems and algorithms. This made Cheptsov [1] make a comparison between high-performance computing (HPC) and the cloud using computational time measurement to understand their scalability for text file parsing. Zhao et al. said in [66], We believe that maximum data volume and a maximum number of jobs are the two important metrics for understanding the performance of a big data analytics platform. Most big data analytics performance standards provide only response time or computational cost. In fact, there are several factors that must be among the criteria to be reckoned with when building a big data analytics system. Among these factors mentioned in [34, 30], the hardware, data transmission bandwidth, fault tolerance, cost, and power consumption of these systems are all issues that need to be considered.

The results of data analysis are not complete until a method is prepared to present these results because the results will be useless if the user cannot easily understand them. The two popular methods for presenting analysis results are business intelligence and network monitoring because they have a user interface that makes analysis results easy to understand. Chang et al. in [65] note that the visual analytics tasks of business systems can be divided into four categories: exploration, dashboards, reporting, and alerting. Cloud UI [55, 9] is the recent trend of big data analytics.

2.5 Frameworks of big data

Most data analysis frameworks belong to one of the three types, which are Interactive analytics, Batch processing, and Stream analytics, which we will address next. [23].

2.5.1 Interactive Analytics

It is an interactive analysis process on data, which enables large data streams to be queried to meet terabyte-sized variable types of data interactively in response time, the user is directly connected to the computer and can link to the system, the data can be matched, reviewed and analyzed in a graphical or tabular diagram or both in a Same time, most tools used in this type are Apache drill.

2.5.1.1 Apache Drill

It is an open-source distributed query engine framework used to handle the interactive analysis of huge datasets, Drill is inspired by Google's Dremel and developed by the Apache software foundation, the main aim of Apache Drill is to process the ad-hoc queries in a very low latency mode, it is intended to handle up to petabytes of data extended across several thousands of servers, in very fast speed which required by the business intelligent analytic environment. A drill specifically focuses on non-relational databases but it can also work with relational ones it supports various NoSQL databases like Hbase and MongoDB, Amazon s3, HDFS, and others. It is compatible with BI/tools like Tableau and Microstage, it can run on Hadoop or on any distributed cluster.

2.5.1.2 Impala

impala is an open-source SQL engine, Impala was introduced in 2015, and runs on hundreds of machines as distributed architecture, it provides an interactive process. Impala supports a massively parallel processing (MPP)engine, when compared with Hive and Spark SQL it is much faster than both. Impala is providing suitable performance using scans, aggregations, and joins to give queries, it considers a low latency apache and faults tolerance, the stored data in impala is in Parquet files, this apache doesn't use Hadoop but it installs a set of daemons on each Data Node for local processing, this strategy is to avoid bottleneck problem. The run time of impala is the very short run time it is supported by HiveQL, although it is compatible with BI it is not a visualization of apache data. In comparison with apache Drill, both of them share a lot of similarities, both are MPP SQL engines and inspired by Google's Dremel, and both are powerful and support low latency processing and fault-tolerant, the most prominent difference is, that Drill is schemaless, it does not need predefined schema, while Impala required schema to be pre-defined.

2.5.2 Batch processing

Batch computing is the implementation of large sets of data already stored in a database, all at once - in batches at the same time. Blocks are stored during work time and then executed when the system becomes idle. Among the most popular frameworks that work with this type of analysis are Apache Hadoop, Apache Dryad, and Apache Mahout. [23].

2.5.2.1 Hadoop/MapRreduce

Hadoop is an open-source framework/platform. Developed by the Apache Software Foundation. Written in Java, launched in 2006, it is considered the most widespread tool for dealing with big data, as it is dedicated to processing and storing unstructured data, and it is inspired by the Google file system that used Map/Reduce that breaks the application into small blocks and runs them on multiple nodes across a group. Hadoop consists of two main components [23]:

1. Hadoop Distributed File System (HDFS), it is the storage layer of Hadoop that stores data in the form of small memory blocks and distributes them to nodes across the cluster, it is specially designed to handle huge amounts of data and it is specially designed for fault tolerance.

2. Map / Reduce is a programming paradigm, used to process huge amounts of data on distributed systems, that perform the task in parallel by distributing it across the node in the cluster (map phase), and then processing the result from the nodes and using that result as input in the reduce phase. Map/Reduce consists of two functions map and reduce function. The map function is used for filtering and sorting while the reduce function is used for grouping.

2.5.2.2 Apache Mahout

It is a project introduced by the apache software foundation in 2009 as a machine learning library which is run on top of Apache Hadoop via Map/Reduce, it is a distributed computing apache, Mahout written in java and scale, it is aimed to support machine learning technique and algorithms for scalable and intelligent big data application analysis, there are numbers of the algorithm in Mahout which are based on classification, clustering, and regression techniques, however, the number of available algorithms are in increase, but various of it are still missing. [23].

Table 2.1 presents a comparison of batch processing frameworks:

Framework	Hadoop / Mapreduce	Mahout
Main Functionality	Distrubuted programming system	Processing machine learning algorithm
Job processing	Process Map/Reduce Job (many input few output)	Process Map/Reduce Job
Complexity	Simple	Very Difficult
Advantages	support fault tolerant, data consistency, concurrency and very high processing performance (Faster than Dryad)	Provides wide collection of various machine learning algorithm.

Table 2.1: comparison of batch processing frameworks

2.5.3 Streams Analytics

Big data technology for data analysis. Which processes a series of data objects that are frequently generated at a high rate, and these sequences of data can be unlimited and need to be processed in a short period of time almost in real time with low latency. This should also happen in the process of data entry and output. [23].

2.5.3.1 Apache Storm

Is a distributed stream processing framework, used to process a enormous amount of structured or unstructured data with high velocity rate in near -real time response Storm cluster run master /slave architecture, the master node run a daemon called “Nimbus”, where slave nodes run a ”supervisor” daemon, “Nimbus ” manage and assigned the tasks - which named as topologies – between the worker nodes. Storm is designed to be a directed acyclic graph which consist of nodes and edges, the edges perform the data transfer between the nodes, where the nodes are categorized for two type, ”spouts” and “blots”, ”spouts” are perform the source of the data/stream where “blots” are the transformation/operation that performs on the stream of data it is best choice for big stream data, Storm have the fault tolerant feature in which if one of the nodes fails master node will reassign the topology for another one.

2.5.3.2 Apache Spark

Spark is an open source distributed big data processing framework widely used after Hadoop. Based on memory arithmetic, Spark improves the real-time performance of data processing in big data environments while ensuring high fault tolerance and scalability. It supports a range of advanced components, including Spark SQL for processing structured data, MLlib for machine learning, GraphX for image processing, and Spark Streaming. These components ensure that Spark performs well in a wide range of areas including machine learning, log file analysis, database management, and graph computing. Typically, one task class of these applications runs recursively with a similar input data set size, called periodic jobs. The modular community has demonstrated both in theory and in practice that most long-range applications can also be represented by some periodic function.[27]

Spark defines more than 180 configuration parameters to support the efficient operation of cyclical jobs. Setting these parameters has a strong impact on overall performance, which are set as default values when Spark is deployed. Choosing the right configuration can greatly improve Spark's performance. However, Spark's configuration for optimal performance is application-specific, where applying only one configuration or modifying one parameter results in suboptimal performance.[27]

Resilient Distributed Datasets (RDDs) are the main data abstraction used in Spark. An RDD is a read-only collection of objects that are partitioned across worker nodes in a cluster. The right side of Figure.2.3 shows an RDD named myRDD that has three partitions. Each partition references a subset of the original dataset and each partition is assigned to a different worker node. Since a partition is the basic unit of parallelism in Spark, it is a common practice to set the minimum number of partitions to be the same as the number of cores in the cluster.[44]

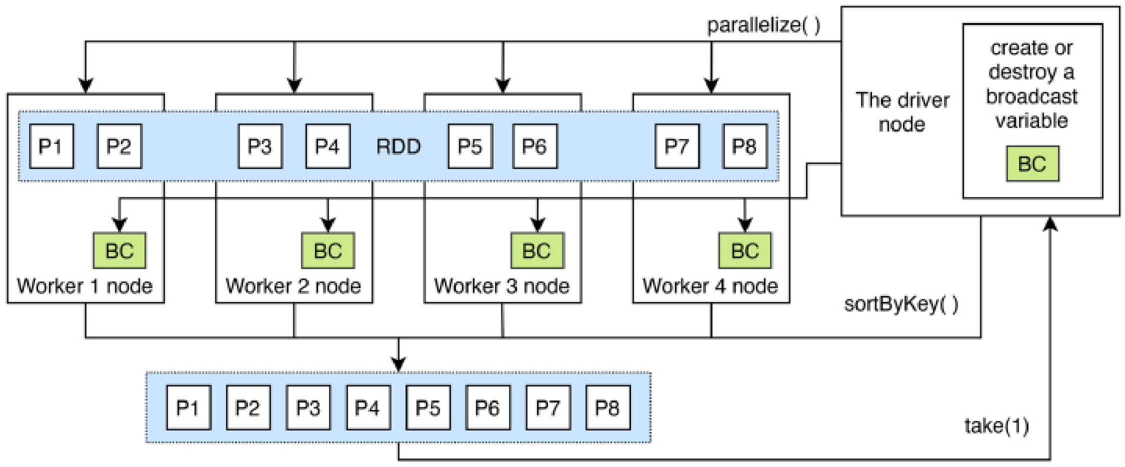


Figure 2.3: Apache Spark architecture. [44]

Table 2.2 presents a comparison of streams analytics frameworks:

2.5.4 frameworks comparison based on supported machine learning algorithms

Many studies have paid great attention to machine learning due to its great importance today. Thus, finding solutions to work with big data, such as predicting patients' conditions from sensors

Framework	Apache Spark/ Spark streaming	Apache Storm
Stream Type	DStream (Discretized streams)	Tuple
Streaming strategy	Process streams as micro batches	Process streams item by item
Latency	A few seconds	Sub-second
Processing Guarantee	Exactly-once	At least-once
Programming languages support	Java, Scala, Python, R	JVM languages (java, scala), python
Throughput	Very high	High
Source of streams	HDFS, Kafka, any DBMS	Spouts
storing state across steams	Stateful	Stateless
Batch processing support	Yes	NO
schdular	Mesos, Yarn Stand alone	ZooKeper
Advantages	• Support both batch and stream mode processing. • Guarantee arriving streams in order.	• provides near real time processing. • very low latency at high stream rate.
Disadvantages	• High memory usage. • Not suitable for sensitive data that require very low latency stream processing.	• Does not guarantee arriving stream in order. • process at least one stream this imply data duplicate.
Data aggregation	integration Kafka library	integration Kafka library

Table 2.2: Comparison of streams analytics frameworks

installed on the body. In the table below we will compare the most popular big data frameworks that support machine learning.

Table 2.3 provides a comparison of frameworks that support machine learning algorithms:

2.5.5 Approaches and solutions for big data

If we search for tools or platforms that provide solutions to big data problems, we will find many of them. Most of these platforms and tools use cloud computing technologies and this is due to the power of computing and the availability of huge storage spaces. As shown in Figure 2.4, all we need is to move the work on big data to the cloud. The obsession with high response time and

Criteria	Hadoop	Spark	Flink	Storm
ML library support	Mahout	Spark-MLib	FlinkML	Trident-m
Used Algorthims	• Clustering (K-Means, fuzzy k-Means, Canopy, Dirichlet, Mean-Shift) • Classification / LogisticRegression (NaiveBayes, SVM,Logistic and linear regression) • Callaborative filtering algorthim (ALS)	• Clustering (K-Means, streaming k-means, Gaussian matrix) • Classification / LogisticRegression (NaiveBayes, SVM, Logistic and linear regression, Random Forests, isotonic) • Callaborative filtering algorthim (ALS)	• Clustering (KNearest neighbors join) • Classification / LogisticRegression (SVM, linear regression) • Callaborative filtering algorthim (ALS)	• Clustering (K-Means) • Classification / LogisticRegression (Perceptron, winnow, Passive Aggressive, A ROW)
Coverage of Algorithms	High	Very High	Low(still growing)	low
Integrattion	Build on top of Hadoop	Build on top of spark core and compatible with others spark library	Build on Flink	Trident-ml build on Trident which is abstraction build on storm

Table 2.3: frameworks that support machine learning algorithms

limited storage memory is no longer an issue, thus data scientists are no longer required to take care of everything with current technologies. All that is currently required of data scientists is to pay more attention to discovering useful information from the data. Because of this, recent studies have paid great attention to providing more efficient and effective methods for analyzing big data and discovering important information.[15]

To improve performance, most models use one of two common methods: scale-up or scale-out. Scale-up (Vertical scaling) improves performance by increasing larger resources (processor, memory, ...) or replacing the device with a higher performance device as shown in Figure 2.5.a. where M1, M2, and M3 represent computer systems that have computing power. different. The computing

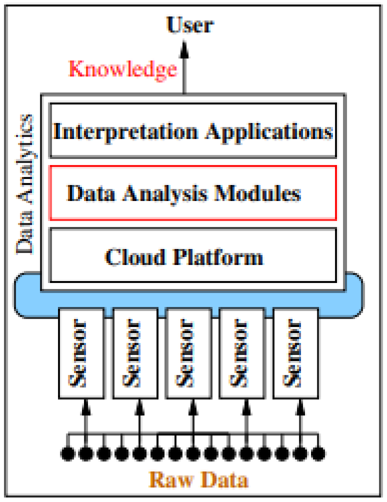


Figure 2.4: the basic idea of big data analytics on cloud system.

power of the three systems is in the range of M3 > M2 > M1. As for the scale-out (horizontal scaling) method, it works to add a new device every time the performance decreases without the need to abandon the old device and thus share processing (divide and conquer) and ensure high performance as shown in Figure 2.5. b. Although this method increases the complexity and anxiety more than maintenance and debugging.[15]

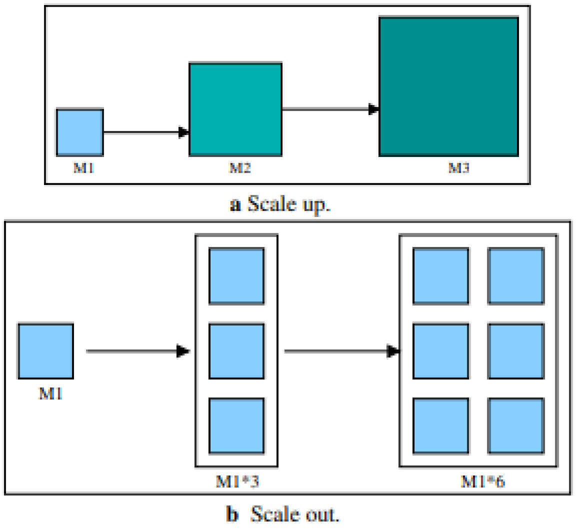


Figure 2.5: The comparisons between scale up and scale out.

Another powerful solution for big data analysis appeared in [50], called the Generalized Linear Aggregate Distributed Engine (GLADE). It is a multi-level tree system for big data analysis consisting of two types of coordinated nodes and operators. As it appears in the simulation results in [14] that GLADE offers better performance than Hadoop in terms of execution time. This is because Hadoop needs a lot of memory and storage to copy data.[15]

The results directed by Fisher et al. [22] Introduced the Big Data pipeline to show the workflow of big data analytics to extract valuable knowledge from big data, which consists of data acquired, structure selection, data modeling in architecture, coding/debugging, and reflection work. From the perspective of statistical computation and data mining.[15]

In [59] a theoretical study was presented explaining the characteristics of big data, called HACE: Among the characteristics of large, random, and distributed data, and starting from these data we try to extract useful information of importance, given this importance, Wu and colleagues [59] presented a data processing framework Big data which includes data access, computing layer, data privacy, domain knowledge level and big data mining algorithm layer. This work shows that the data mining algorithm will become more important and much more difficult than before, and from this, challenges will also arise in the design and implementation of big data analysis platforms. In addition, the issue of privacy for big data analytics has been a promising research in recent years. Explained by Laurella et al. [39]. Privacy is very important when we try to find important information from data collected from mobile devices. Therefore, data security and data anonymization must be ensured when analyzing this data.[15]

Talia noted in [20] that cloud-based data analysis services can be divided into data analysis software as a service, data analytics platform as a service, and data analytics infrastructure as a service. Other studies have emerged from a distance [41] that provide a general architecture for big data analytics that includes multi-source big data collection, distributed big data storage, and in/between big data processing. With the emergence of many frameworks and data analysis platforms, some studies have tried to compare them to help choose the most suitable and optimal solution among them. In [18], Cuzzocrea et al. First discuss how recent studies have responded to the issue of "computational contingency" to big data analyses. Some open issues, such as data source heterogeneity and filtering of unrelated data, and potential research directions were also presented in the same study. And in [64], Zhang and Huang use the 5Ws model to explain what kind of framework and method we need for different big data approaches. Zhang and Huang also explained that the 5Ws model represents the type of data, why we have this data, where the data comes from, when the data occurs, who receives the data, and how the data is transmitted.[15]

In [30], in addition to specifying that a big data system should include data generation, data acquisition, data warehousing, and data analysis units, Hu et al. He also mentioned that a big data system can be decomposed into infrastructure, computing, and application layers. In addition, in this study, the research of storage systems of type NoSQL, which is divided into databases with key value, column, document, and rows, was discussed. In light of the high cost of computing in big data analysis. The HPCC cluster system is an early stage solution for big data analysis. In [51] Sagiroglu and Sinanc compare characteristics between HPCC and Hadoop. They concluded that HPCC uses multivariate indexes and variables on distributed file systems while Hadoop uses column-oriented database. [15]

2.6 Conclusion

In the above, we discussed in detail Big Data, its definition, stages of analysis, and the most popular frameworks for its processing.
In the next chapter, we will provide a detailed explanation of the problem they have addressed and the proposed solution in all its stages of work, each stage separately.

Chapter 3

Internet of Things Model for streaming data analytics using machine learning on big data environment

3.1 Introduction

In this chapter, we will provide a detailed explanation of the problem that we will address, the general outline of the solution, and a detailed explanation of the actors in the system

3.2 Problem definition

The healthcare industry produces a large volume of healthcare data due to advances in technology and the digitization of medical records. In recent years, Health Information Technology (HIT) has developed the ability to electronically create, store, and transmit data worldwide in a matter of seconds and also has the potential to deliver significantly better productivity and quality of service to healthcare. Healthcare sectors have generated massive amounts of healthcare data through patient record keeping. All of this data together makes up the healthcare big data. To be more specific, healthcare big data can be defined as electronic medical records (EMRs) that include a patient's medical history, doctor's notes, clinical reports, vital data, and other health-related medical data. Importantly, the exponential growth of healthcare data is another major issue in current healthcare information systems (HISs). This is not just about the sheer volume of healthcare data; However, we are also seeing an exponential rise in the speed with which this data is generated, as well as a greater diversity of medical data and this issue presents major challenges that need to be solved and by whom: good data storage and analysis and ensuring all data is retrieved without loss.[48]

Designing a distributed data processing system to deal with the huge data generated by medical devices faces four major challenges: First, the data is generated very quickly by the various medical devices. Second, due to the sheer volume of heterogeneous data, it is difficult to collect data from distributed sites. Third, storage is the main problem for large and heterogeneous data sets. Big data system needs storage while providing performance guarantee. Fourth, it relates to big data analytics, more precisely the mining of large data sets in real time or near real time that includes modeling, visualization, prediction and optimization. These challenges require a new processing model because current data management systems are not effective in dealing with the heterogeneous nature of data or in real time. These traditional systems do not provide any support for unstructured or semi-structured data. From the perspective of scalability, when the data volume grows, there are many failures of traditional RDBMS in scaling to manage parallel devices and fault tolerance, which is not suitable for managing the incremental data.

Based on the above-mentioned problems and challenges, we will present a solution that limits the following problems:

- Data flow problems from several sources quickly and very large volume.

- Problems with processing big data that is distributed in multiple sources
- Real-time streaming big data processing problems

3.2.1 Related work

There are many studies that discuss the problems of big data resulting from the Internet of things, which use the field of machine science to predict risks or support decisions...etc. Among the most prominent of these studies, we will discuss in this section.

Abderrahmane Ed-daoudy and Khalil Maalmi [3] built a real-time health prediction system (heart disease and diabetes) based on data generated from wearable health monitors. Abderrahmane and Khalil say the real-time prediction can reduce doctor attendance and help clinicians and patients respond in advance to a potential illness. They used the Apache Spark framework to process the huge amount of data generated by wearable health monitors and to organize the flow of this huge data they used Spark streaming built into the Spark framework and Apache Kafka for data streaming and in the machine learning part they used Spark's built-in MLlib library which contains a collection of One of the most popular algorithms is classification, regression, clustering, etc.

In [46] Muhammad Syafrudin et al. proposed a real-time monitoring system that uses IoT-based sensors, big data processing, and a hybrid prediction model. Muhammad says: Monitoring systems have become important factors in management decision-making. Existing technologies such as sensors based on the Internet of Things (IoT) can be considered as a solution to provide effective monitoring of the manufacturing process. They used the Apache Storm framework to process big data generated by IoT-based sensors and to organize traffic and queues. They used Apache Kafka and MongoDB to store data generated by sensors, a density-based spatial clustering model for applications with noise (DBSCAN) and random forest classification to remove data Output sensor and fault detection during the manufacturing process, and this model is considered one of the hybrid models.

3.3 General architecture

To design a distributed data processing system to deal with huge data generated by medical care devices. Figure 3.1 explains the general architecture of the system

The system consists of five main sections, which are as follows: the data source section, then the data collection and organization section, and then the data processing section, then storing and displaying the results to the end user.

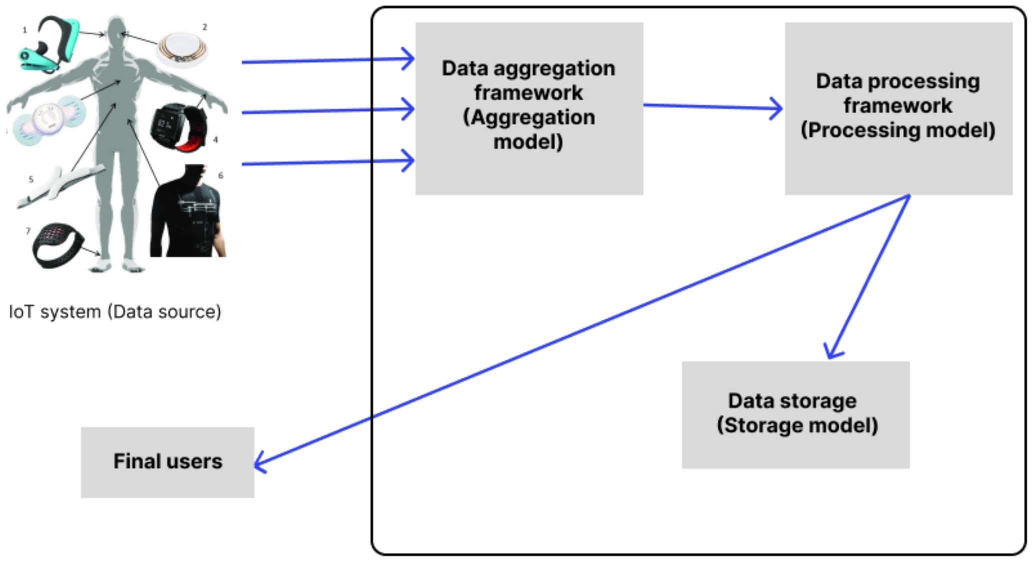

IoT system (Data source)
Data aggregation framework (Aggregation model)
Data processing framework (Processing model)
Data storage (Storage model)
Final users

Figure 3.1: General architecture

3.4 IoT system (Data source)

The Internet of Things (IoT) is a network of physical devices and other elements, integrated with electronics, smart clothing, software, smart applications, sensors, and network connectivity so that they can collect and exchange data with each other or with data center systems. With wearable health monitoring devices available in many homes, the data generated by these devices are large in size and random in nature and need to be analyzed using a big data analytics system in order to understand user behavior patterns or extract important information. The convergence of medicine and information technologies such as medical informatics will transform health care as we know it, reduce inefficiencies, reduce costs, and save lives. Real-time monitoring via the Internet of Things can save lives in case of medical emergencies such as heart disease, diabetes, and many other chronic diseases. Many health-related resources are now available that constantly monitor health indicators.

3.5 Data aggregation framework

As data generated in healthcare grows at an exponential rate, managing this data using big data frameworks becomes a challenging task, while a data aggregation framework is specifically designed for data management. Hence it was integrated into our system. In the proposed system architecture, a data aggregation block is used to collect individual health data from distributed sources and multiple diseases using different devices integrated with telemedicine and telehealth. This block collects, filters and manages patients' clinical data in an ongoing manner. The data collection framework allows us to categorize data flows into categories (type of disease) in which records are published.

The data collection framework is a distributed stream system that uses publish and subscribe messages and is developed as a distributed, segmented, and iterative service. The data is flowed in real time from the health monitoring devices to the framework. All data received from healthcare devices is stored in the form of multiple partitions within the data collection framework servers. The data arrives at the processing form at the same moment it is received by the health care services, and then it is processed using the processing framework.

3.6 Data processing framework

The architecture below shows how the data processing framework works, which consists of two parts: batch processing, which is at the level of data stored in databases, and streaming processing, which is real-time processing of data. Figure 3.2

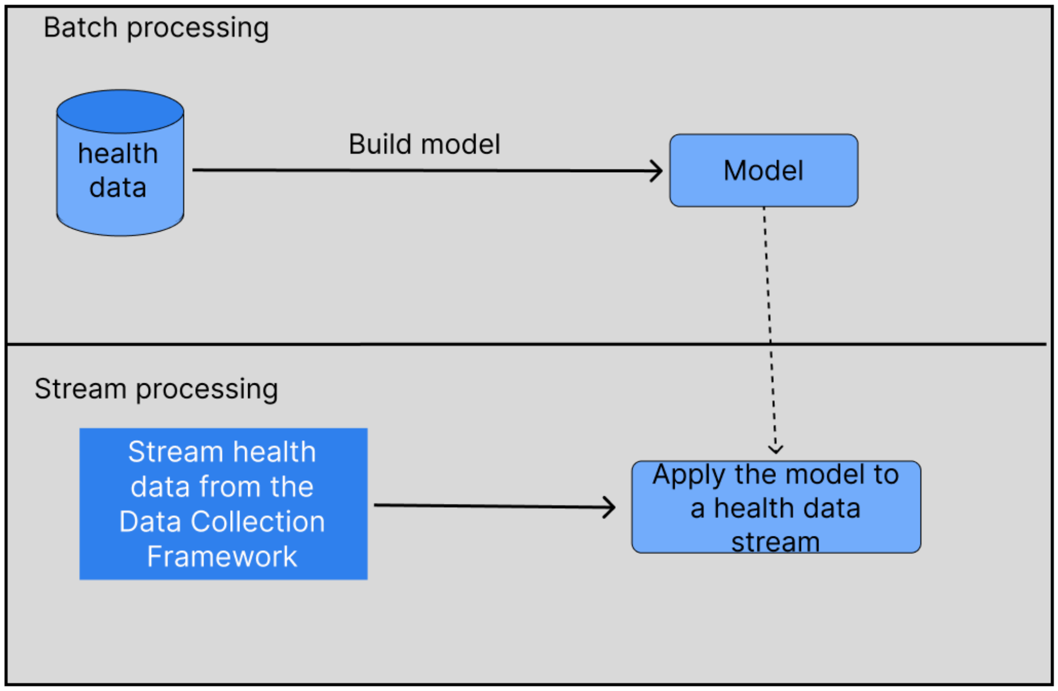

Batch processing
health data
Build model
Model
Stream processing
Stream health data from the Data Collection Framework
Apply the model to a health data stream

Figure 3.2: Data processing framework architecture

3.6.1 Healthcare data

After collecting the data from the distributed sources for different diseases, the classification of this data needs to build a classification model capable of classifying the user's traits in the absence or presence of the disease. Classification is one of the main methods of data mining that is useful for finding hidden information. Classification consists in examining the properties of a newly entered element in order to assign it to a class from a predetermined set. DT is widely used for classification and regression problems. DT is a popular methods for classification machine learning tasks, and it is widely used in machine learning because it is easy to use, easy to operate, easy to interpret, and extends to multi-layer classification setup. DT prediction was made using a big data processing framework that supports DT for binary and multi-category classification.

3.6.2 Batch processing (model training)

After storing the data received from healthcare devices, we can benefit from it to improve the accuracy of the results of the healthcare model based on the Random Forest algorithm, where we retrain the model in each period to reduce erroneous results.

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

Random forest (RF) is an ensemble learning classification and regression method suitable for handling problems involving grouping data into classes. The algorithm was developed by Breiman and Cutler [6]. In RF, prediction is achieved using decision trees. During the training phase, a number of decision trees are constructed which are then used for the class prediction; this is achieved by considering the voted classes of all the individual trees and the class with the highest vote is deemed to be the output.

A summary of how a forest (i.e., a collection of trees) is constructed is explained in Figure 3.2.

```
Begin RF Algorithm
  Input:   $N$: number of nodes
           $M$: number of features
           $D$: number of trees to be constructed
  Output: $V$: the class with the highest vote
  While stopping criteria is false do
    Randomly draw a bootstrap sample A from the training data $D$
    Use the steps below to construct tree $T_i$ from the drawn bootstrapped sample A:
      (I) Randomly select $m$ features from $M$; where $m \ll M$
      (II) For node d, calculate the best split point among the $m$ features
      (III) Split the node into two daughter nodes using the best split
      (IV) Repeat I, II and III until $n$ number of nodes has been reached
    Build your forest by repeating steps I–IV for $D$ number of times
  End While
    Output all the constructed trees $\{T_i\}_1^D$
    Apply a new sample to each of the constructed trees starting from the root node
    Assign the sample to the class corresponding to the leaf node.
    Combine the decisions (or votes) of all the trees
    Output $V$, that is, the class with the highest vote.
End RF Algorithm
```

Figure 3.3: Random forest (RF) Algorithm

3.6.3 Stream processing

When health data is received by the data collection framework, it is split between nodes so that the healthcare model is applied to them in real-time. The results are stored in databases and a copy is sent to the final users in the form of reports and graphs that give them a clearer view of the health status of patients.

3.7 Final users

After the data is processed, the results are sent to the end users in the form of interactive graphic curves and reports that are read by the doctors or the emergency department until the appropriate decision is taken.

3.8 Conclusion

In the above, we have discussed in detail the scheme of the distributed data processing system to deal with the big data generated by medical devices. In the next chapter, we will present a comprehensive overview of the work environment, the tools used in this system, and the results obtained.

Chapter 4

Implementation

4.1 Introduction

In this chapter, we introduce the implementation aspect of our application.

We started by introducing the hardware environment in which our system is developed, the programming languages and the tools used.

Finally, we present a description of our system based on experimental results on Apache Spark for the purpose of demonstrating the efficacy of our solution.

4.2 Development environment

In this part we will cite the hardware and software environment used.

4.2.1 Hardware environment

Processor	Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.71 GHz
RAM memory	16.0 GB (5.9 GB used)
System type	operating system 64-bit and x64-based processor
Operating system	Ubuntu 22.04 Lts

Table 4.1: Characteristics of the computer used for development

4.2.2 Software environment

4.2.2.1 The Python language

Python is the open source programming language most used by computer scientists. This language has propelled itself to the fore in infrastructure management, data analysis or in the field of software development. Indeed, among its qualities, Python allows developers to focus on what they do rather than how they do it. It freed developers from the form constraints that plagued their time with older languages. Thus, developing code with Python is faster than with other languages.

The main uses of Python by developers are:

- Application programming.
- The creation of web services.
- Code generation.
- Meta-programming.

Technically, this language will be used mainly for scripting and automation (interaction with web browsers).

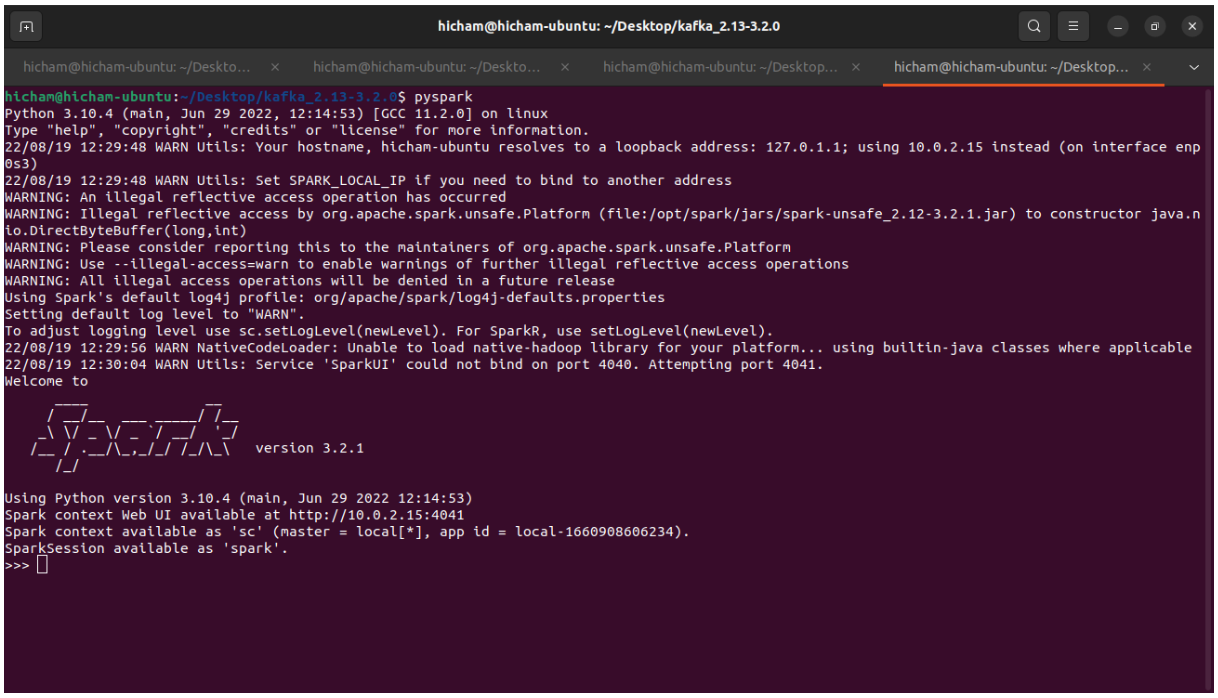
There are two versions: Python 2 and Python 3. Python 2, the old version offers updates until 2020. Python 3 is the current version. Its interpreter is more efficient, as well as its concurrency control.

4.2.2.2 Apache Spark

Apache Spark is an open source, multi-language, distributed processing system for big data workloads. It uses in-memory caching and optimized query execution for fast queries against data of any size. Simply put, Spark is a fast, generic engine for large scale data processing.

The quick part means that it's faster than previous methods for working with big data like the classic MapReduce. The secret to being faster is that Spark runs on memory (RAM), and this makes processing much faster than disk drives.

The generic part means that it can be used for various things like running distributed SQL, creating data pipelines, ingesting data in a database, running machine learning algorithms, and working with graphs or data flows.


hicham@hicham-ubuntu: ~/Desktop/kafka_2.13-3.2.0
hicham@hicham-ubuntu: ~/Deskto...
hicham@hicham-ubuntu: ~/Deskto...
hicham@hicham-ubuntu: ~/Desktop...
hicham@hicham-ubuntu: ~/Desktop...

```
hicham@hicham-ubuntu:~/Desktop/kafka_2.13-3.2.0$ pyspark
Python 3.10.4 (main, Jun 29 2022, 12:14:53) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
22/08/19 12:29:48 WARN Utils: Your hostname, hicham-ubuntu resolves to a loopback address: 127.0.1.1; using 10.0.2.15 instead (on interface enp
0s3)
22/08/19 12:29:48 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/opt/spark/jars/spark-unsafe_2.12-3.2.1.jar) to constructor java.n
io.DirectByteBuffer(long,int)
WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/08/19 12:29:56 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
22/08/19 12:30:04 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.2.1
      /_/

Using Python version 3.10.4 (main, Jun 29 2022 12:14:53)
Spark context Web UI available at http://10.0.2.15:4041
Spark context available as 'sc' (master = local[*], app id = local-1660908606234).
SparkSession available as 'spark'.
>>> 
```

Figure 4.1: PySpark terminal.

4.2.2.3 Apache Kafka

Apache Kafka is a distributed event store and stream-processing platform. It is an open-source system developed by the Apache Software Foundation written in Java and Scala. The project aims to provide a unified, high-throughput, low-latency platform for handling real-time data feeds. Kafka can connect to external systems (for data import/export) via Kafka Connect, and provides the Kafka Streams libraries for stream processing applications. Kafka uses a binary TCP-based protocol that is optimized for efficiency and relies on a ”message set” abstraction that naturally groups messages together to reduce the overhead of the network roundtrip. This ”leads to larger network packets, larger sequential disk operations, contiguous memory blocks which allows Kafka to turn a bursty stream of random message writes into linear writes.

4.2.2.4 Pycharm IDE

PyCharm Community Edition is a simplified code editor, which is free and developed in open source by JetBrains. It works on Windows, macOS, and Linux operating systems. It provides developers with an integrated development environment with tools to advance technical projects, from editing, to building, to debugging.

The features offered by PyCharm Community Edition are numerous. We find in particular:

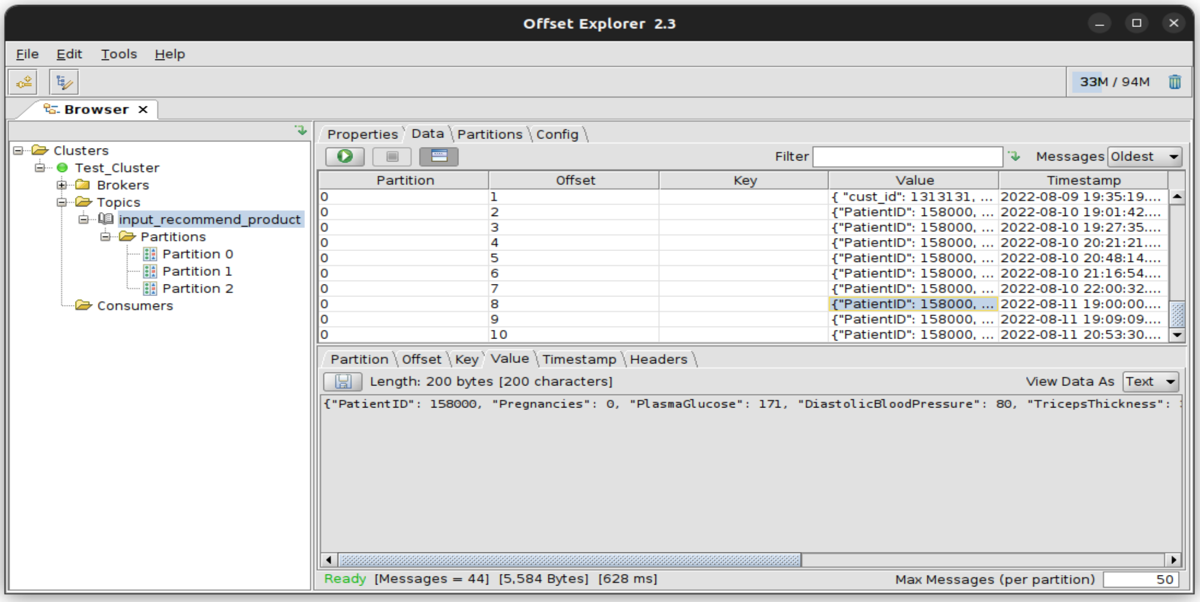
- Intelligent Coding Assistance: PyCharm provides smart code completion, code inspections, on-the-fly error highlighting and quick-fixes, along with automated code refactorings and rich navigation capabilities.
- Built-in Developer Tools: PyCharm's huge collection of tools out of the box includes an integrated debugger and test runner; Python profiler; a built-in terminal; integration with major VCS and built-in database tools; remote development capabilities with remote interpreters; an integrated ssh terminal; and integration with Docker and Vagrant.
- Web Development: In addition to Python, PyCharm provides first-class support for various Python web development frameworks, specific template languages, JavaScript, CoffeeScript, TypeScript, HTML/CSS, AngularJS, Node.js, and more.
- Scientific Tools: PyCharm integrates with IPython Notebook, has an interactive Python console, and supports Anaconda as well as multiple scientific packages including Matplotlib and NumPy.

4.2.2.5 Offset Explorer

Offset Explorer (formerly Kafka Tool) is a GUI application for managing and using Apache Kafka clusters. It provides an intuitive UI that allows one to quickly view objects within a Kafka cluster as well as the messages stored in the topics of the cluster. It contains features geared towards both developers and administrators. Some of the key features include [1]

- Quickly view all Kafka clusters, including their brokers, topics and consumers.
- View contents of messages partitions and add new messages.
- View offsets of the consumers, including Apache Storm Kafka spout consumers.
- Show JSON, XML and Avro messages in a pretty-printed format.
- Add and drop topics plus other management features.
- Save individual messages from partitions to local hard drive.
- Write plugins that allow to view custom data formats.
- Offset Explorer runs on Windows, Linux and Mac OS.

Offset Explorer is free for personal use only. Any non-personal use, including commercial, educational and non-profit work is not permitted without purchasing a license. Non-personal use is allowed for evaluation purposes for 30 days following the download of Offset Explorer, after which you must purchase a valid license or remove the software.



Partition	Offset	Key	Value	Timestamp
0	1		{ "cust_id": 1313131, ...	2022-08-09 19:35:19....
0	2		{"PatientID": 158000, ...	2022-08-10 19:01:42....
0	3		{"PatientID": 158000, ...	2022-08-10 19:27:35....
0	4		{"PatientID": 158000, ...	2022-08-10 20:21:21....
0	5		{"PatientID": 158000, ...	2022-08-10 20:48:14....
0	6		{"PatientID": 158000, ...	2022-08-10 21:16:54....
0	7		{"PatientID": 158000, ...	2022-08-10 22:00:32....
0	8		{"PatientID": 158000, ...	2022-08-11 19:00:00....
0	9		{"PatientID": 158000, ...	2022-08-11 19:09:09....
0	10		{"PatientID": 158000, ...	2022-08-11 20:53:30....

Length: 200 bytes [200 characters]

```
{"PatientID": 158000, "Pregnancies": 0, "PlasmaGlucose": 171, "DiastolicBloodPressure": 80, "TricepsThickness":
```

Ready [Messages = 44] [5,584 Bytes] [628 ms]

Figure 4.2: Offset Explorer.

4.3 Dataset

To implement the proposed model for this research, two datasets have been used. The data set we used for diabetic data analysis is taken from a website named Kaggle [2] which provides online datasets for data scientists and aims at discovering and seamlessly analyzing open data. The diabetes dataset consists of 15 000 records and nine attributes, each record has eight attributes which are pregnancies, glucose, blood pressure, skin thickness, insulin, BMI, diabetes pedigree function, age, and one of the two possible outcomes, namely whether the patient is tested positive for diabetes indicated by 1 or not indicated by 0.

No	Attributes	Description	min	max
1	Pregnancies	number of pregnancies	0	14
2	PlasmaGlucose	number of pregnancies	4	192
3	DiastolicBloodPressure	Blood Pressure	24	117
4	TricepsThickness	Triceps Thickness	7	93
5	SerumInsulin	Insulin level in the blood	14	799
6	BMI	The body mass index	18.2	56
7	DiabetesPedigree	Diabetes pedigree function	0.08	2.3
8	Age	Age in years	21	77
9	Diabetic	Indicators of diabetes (1= positive , 0= negative)	0	1

Table 4.2: The attributes of the dataset and the minimum and maximum values for each attribute.

Indicators show variance in characteristics:

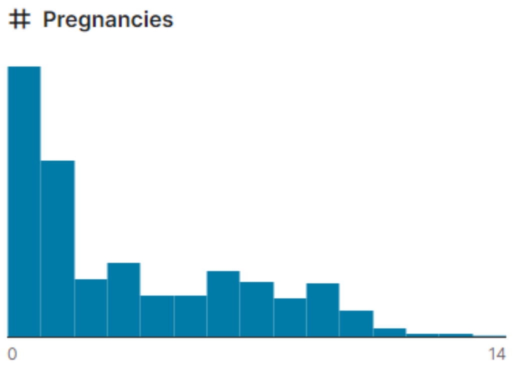


Figure 4.3: Graphic columns for Pregnancies.

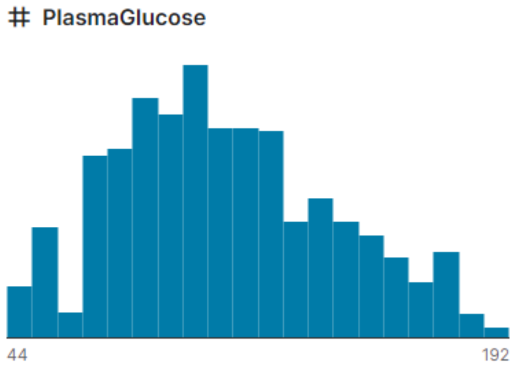


Figure 4.4: Graphic columns for PlasmaGlucose.

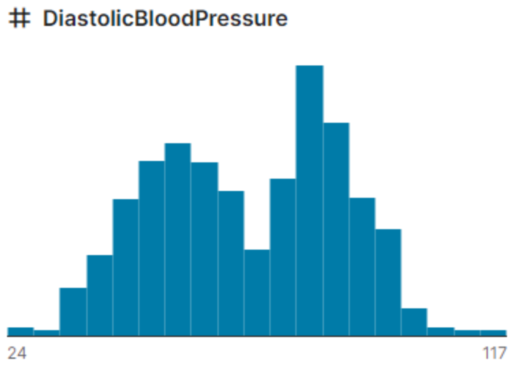


Figure 4.5: Graphic columns for DiastolicBloodPressure .

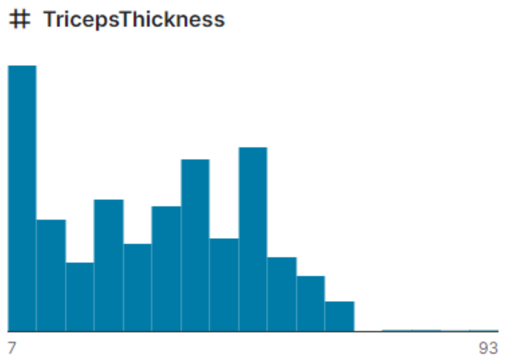


Figure 4.6: Graphic columns for TricepsThickness.

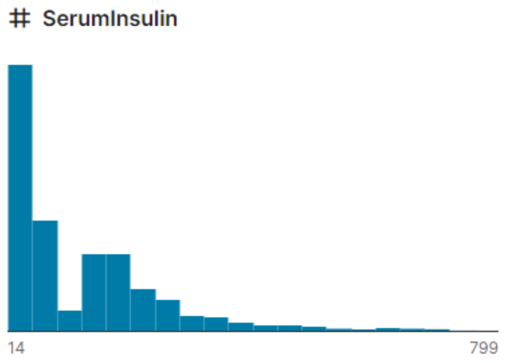


Figure 4.7: Graphic columns for SerumInsulin.

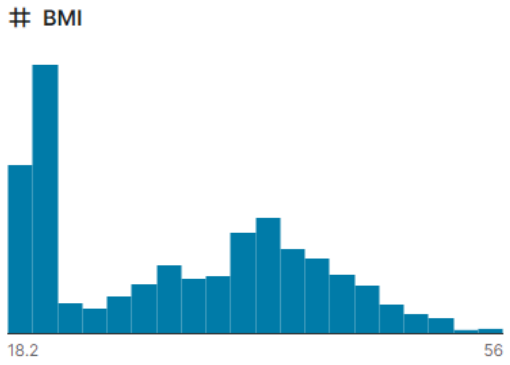


Figure 4.8: Graphic columns for BMI.

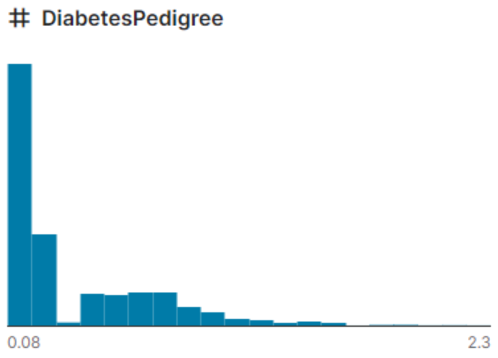


Figure 4.9: Graphic columns for DiabetesPedigree.

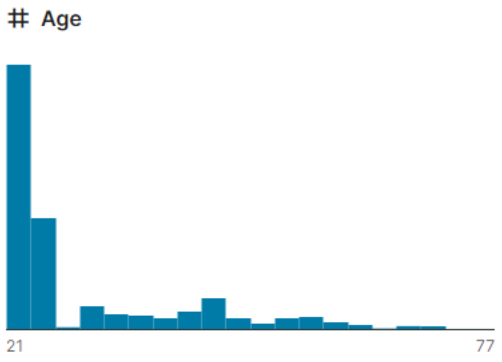


Figure 4.10: Graphic columns for Age.

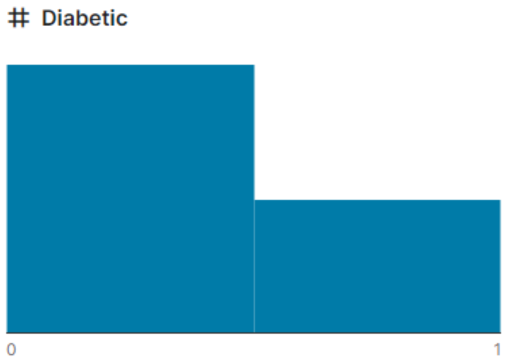
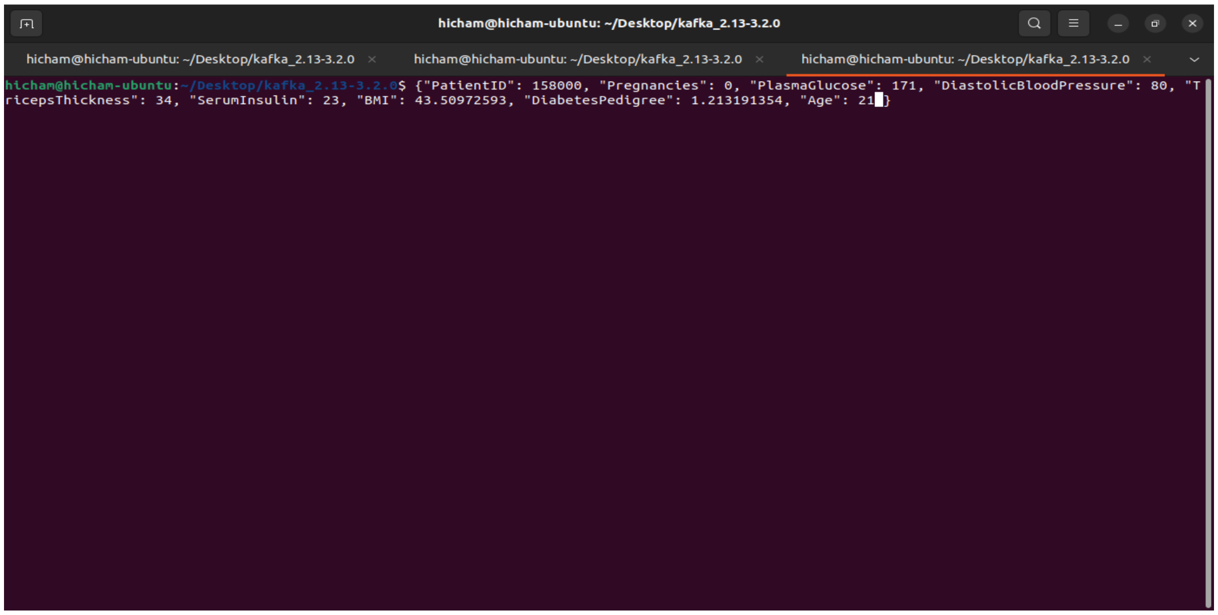


Figure 4.11: Graphic columns for Diabetic.

4.4 Implementation

4.4.1 Data sent from sensors

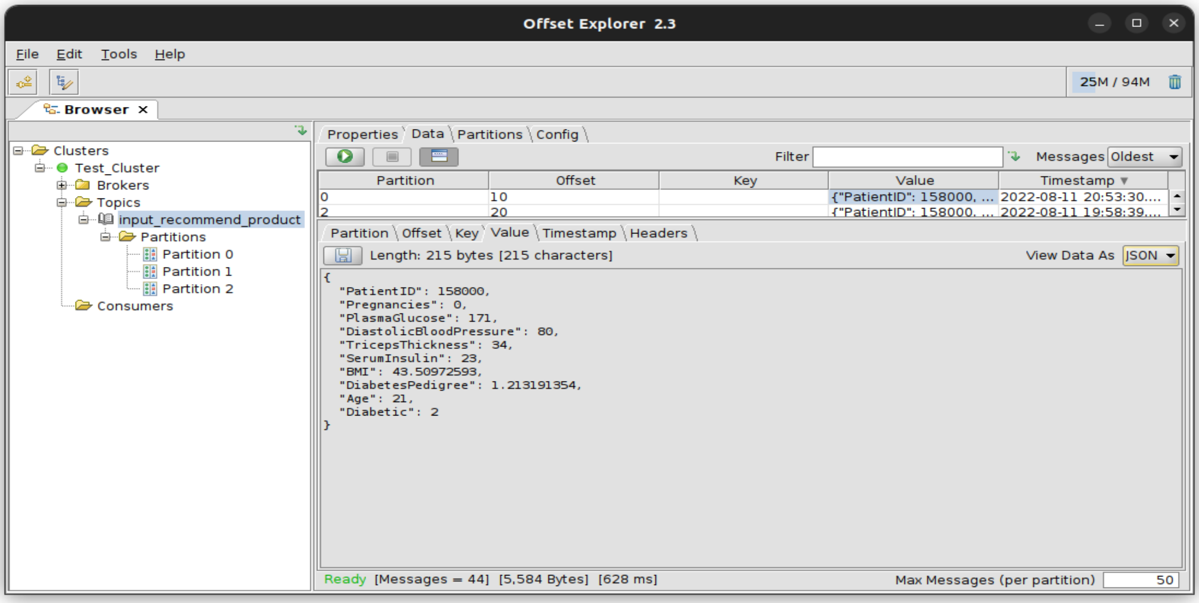
Simulate data sent from sensors (producer) in JSON format to Apache Kafka: 4.12


hicham@hicham-ubuntu: ~/Desktop/kafka_2.13-3.2.0
hicham@hicham-ubuntu: ~/Desktop/kafka_2.13-3.2.0
hicham@hicham-ubuntu: ~/Desktop/kafka_2.13-3.2.0
hicham@hicham-ubuntu: ~/Desktop/kafka_2.13-3.2.0

```
hicham@hicham-ubuntu:~/Desktop/kafka_2.13-3.2.0$ {"PatientID": 158000, "Pregnancies": 0, "PlasmaGlucose": 171, "DiastolicBloodPressure": 80, "T
ricepsThickness": 34, "SerumInsulin": 23, "BMI": 43.50972593, "DiabetesPedigree": 1.213191354, "Age": 21}
```

Figure 4.12: Graphic columns for Simulate data sent from sensors.

Data inside Apache Kafka using Offset Explorer:



Partition	Offset	Key	Value	Timestamp
0	10		{"PatientID": 158000, ...	2022-08-11 20:53:30....
2	20		{"PatientID": 158000, ...	2022-08-11 19:58:39....

Length: 215 bytes [215 characters]

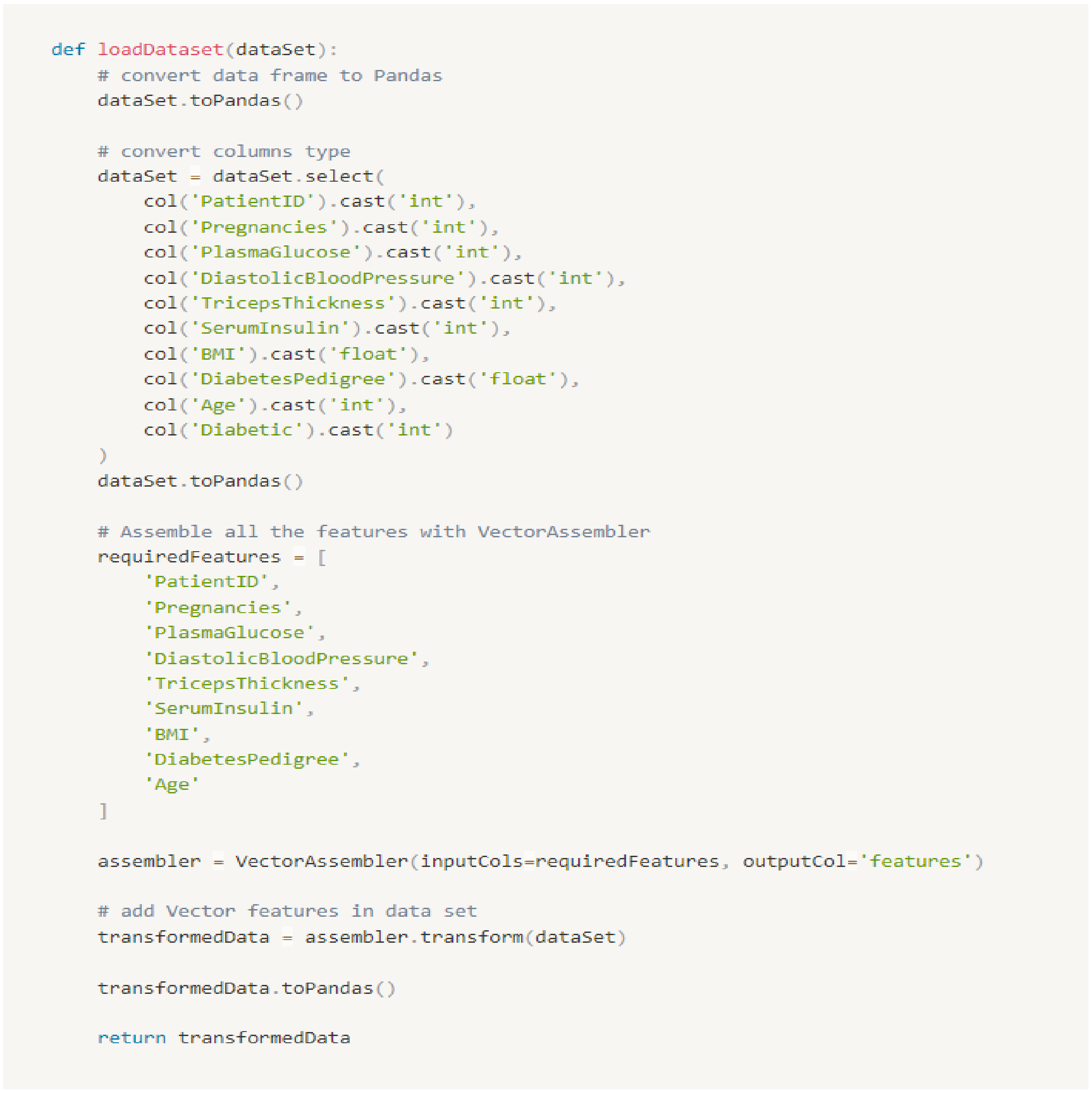
```
{
  "PatientID": 158000,
  "Pregnancies": 0,
  "PlasmaGlucose": 171,
  "DiastolicBloodPressure": 80,
  "TricepsThickness": 34,
  "SerumInsulin": 23,
  "BMI": 43.50972593,
  "DiabetesPedigree": 1.213191354,
  "Age": 21,
  "Diabetic": 2
}
```

Ready [Messages = 44] [5,584 Bytes] [628 ms]

Figure 4.13: Graphic columns for Diabetic.

4.4.2 Data reception, analysis and processing by Apache Spark

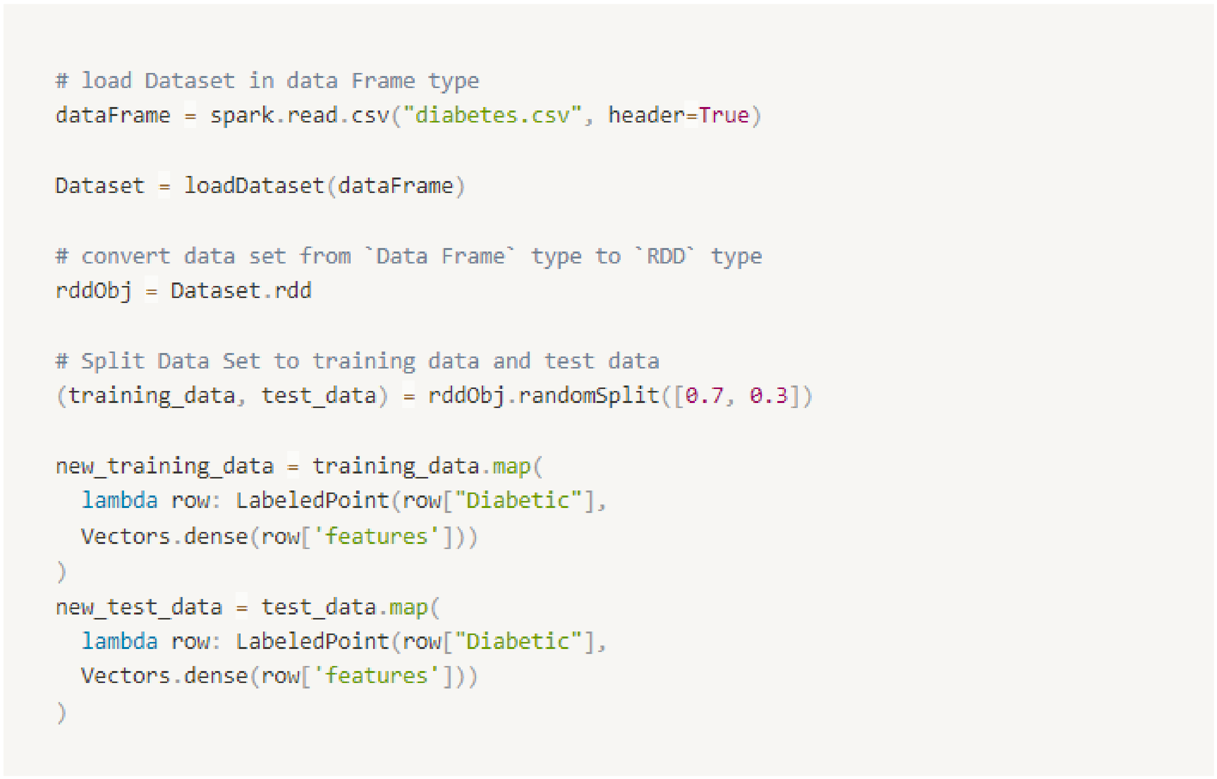
Download the dataset and collect all the properties into a single vector called features so that they are ready to be passed to the classification algorithm. Figure 4.14



```
def loadDataset(dataSet):
    # convert data frame to Pandas
    dataSet.toPandas()

    # convert columns type
    dataSet = dataSet.select(
        col('PatientID').cast('int'),
        col('Pregnancies').cast('int'),
        col('PlasmaGlucose').cast('int'),
        col('DiastolicBloodPressure').cast('int'),
        col('TricepsThickness').cast('int'),
        col('SerumInsulin').cast('int'),
        col('BMI').cast('float'),
        col('DiabetesPedigree').cast('float'),
        col('Age').cast('int'),
        col('Diabetic').cast('int')
    )
    dataSet.toPandas()

    # Assemble all the features with VectorAssembler
    requiredFeatures = [
        'PatientID',
        'Pregnancies',
        'PlasmaGlucose',
        'DiastolicBloodPressure',
        'TricepsThickness',
        'SerumInsulin',
        'BMI',
        'DiabetesPedigree',
        'Age'
    ]

    assembler = VectorAssembler(inputCols=requiredFeatures, outputCol='features')

    # add Vector features in data set
    transformedData = assembler.transform(dataSet)

    transformedData.toPandas()

    return transformedData
```

Figure 4.14: Code snippet load the dataset and collect.

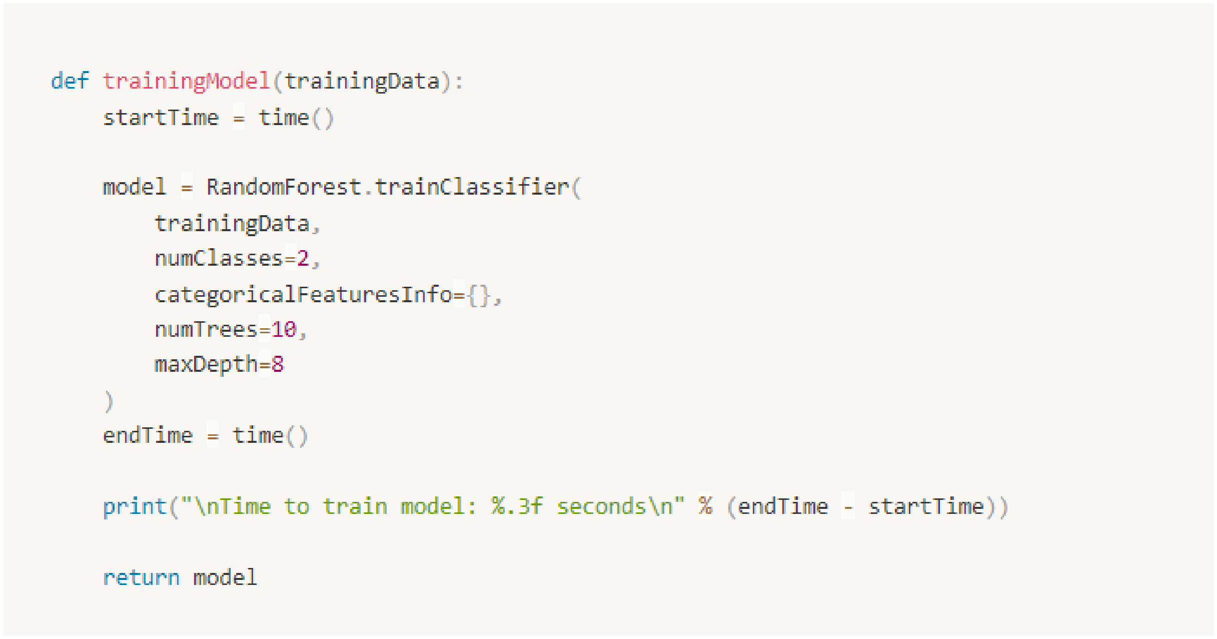
Convert the data to RDD format and divide it into 70% training data and 30% testing data. Figure 4.15



```
# load Dataset in data Frame type
dataFrame = spark.read.csv("diabetes.csv", header=True)

Dataset = loadDataset(dataFrame)

# convert data set from `Data Frame` type to `RDD` type
rddObj = Dataset.rdd

# Split Data Set to training data and test data
(training_data, test_data) = rddObj.randomSplit([0.7, 0.3])

new_training_data = training_data.map(
  lambda row: LabeledPoint(row["Diabetic"],
  Vectors.dense(row['features']))
)
new_test_data = test_data.map(
  lambda row: LabeledPoint(row["Diabetic"],
  Vectors.dense(row['features']))
)
```

Figure 4.15: Code snippet convert the data to RDD.

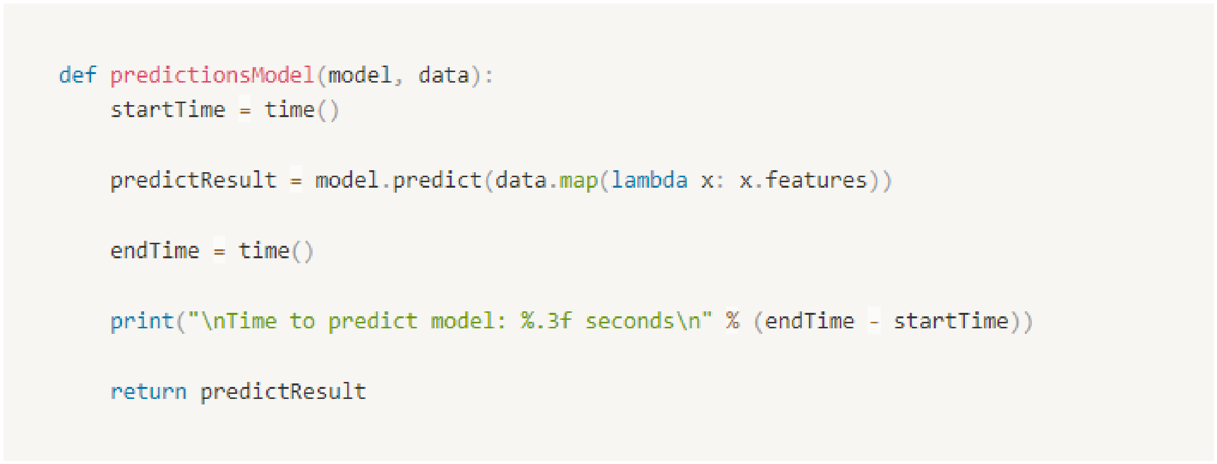
Training the model based on the Random Forest algorithm and displaying the training time. Figure 4.16



```
def trainingModel(trainingData):
    startTime = time()

    model = RandomForest.trainClassifier(
        trainingData,
        numClasses=2,
        categoricalFeaturesInfo={},
        numTrees=10,
        maxDepth=8
    )
    endTime = time()

    print("\nTime to train model: %.3f seconds\n" % (endTime - startTime))

    return model
```

Figure 4.16: Code snippet training the model based on the Random Forest algorithm.

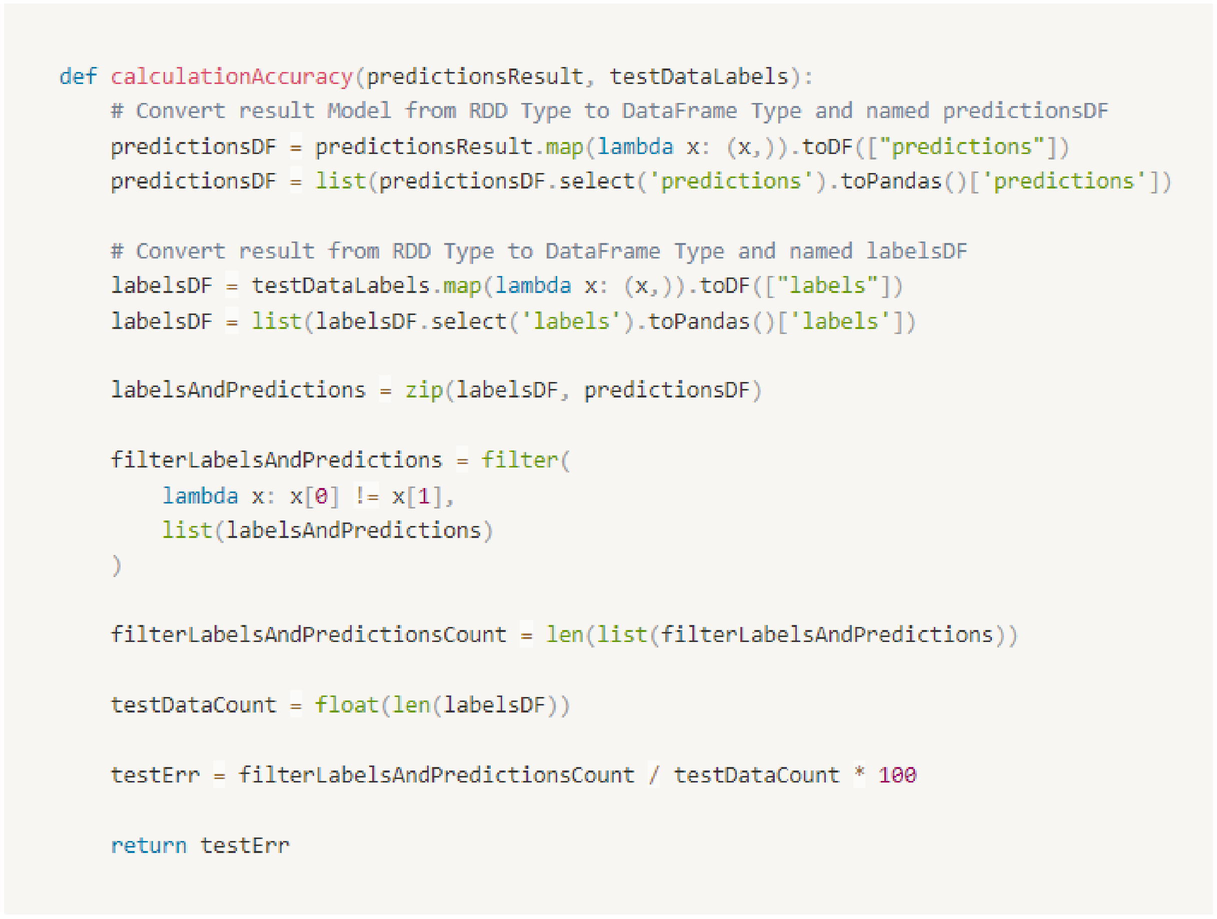
Test the model and print execution time. Figure 4.17



```
def predictionsModel(model, data):
    startTime = time()

    predictResult = model.predict(data.map(lambda x: x.features))

    endTime = time()

    print("\nTime to predict model: %.3f seconds\n" % (endTime - startTime))

    return predictResult
```

Figure 4.17: Code snippet test the model.

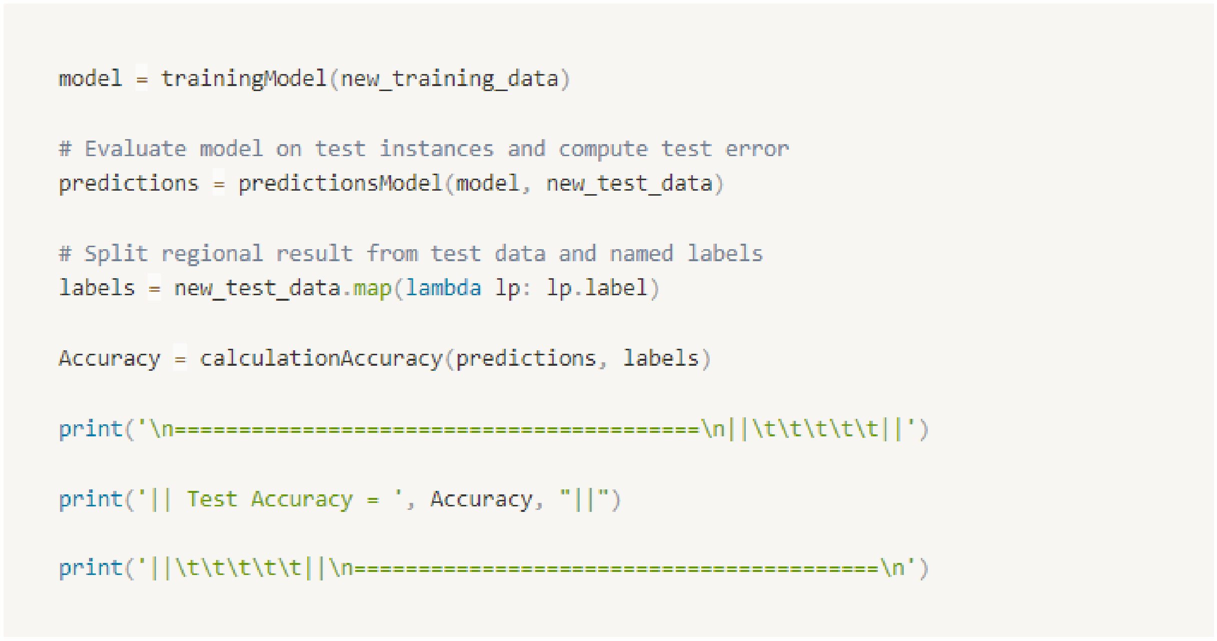
Calculate the model's accuracy percentage by dividing the number of false results from the experiment by the total number of experimental data. Figure 4.18



```python
def calculationAccuracy(predictionsResult, testDataLabels):
    # Convert result Model from RDD Type to DataFrame Type and named predictionsDF
    predictionsDF = predictionsResult.map(lambda x: (x,)).toDF(["predictions"])
    predictionsDF = list(predictionsDF.select('predictions').toPandas()['predictions'])

    # Convert result from RDD Type to DataFrame Type and named labelsDF
    labelsDF = testDataLabels.map(lambda x: (x,)).toDF(["labels"])
    labelsDF = list(labelsDF.select('labels').toPandas()['labels'])

    labelsAndPredictions = zip(labelsDF, predictionsDF)

    filterLabelsAndPredictions = filter(
        lambda x: x[0] != x[1],
        list(labelsAndPredictions)
    )

    filterLabelsAndPredictionsCount = len(list(filterLabelsAndPredictions))

    testDataCount = float(len(labelsDF))

    testErr = filterLabelsAndPredictionsCount / testDataCount * 100

    return testErr
```

Figure 4.18: Code snippet calculate the model's accuracy.

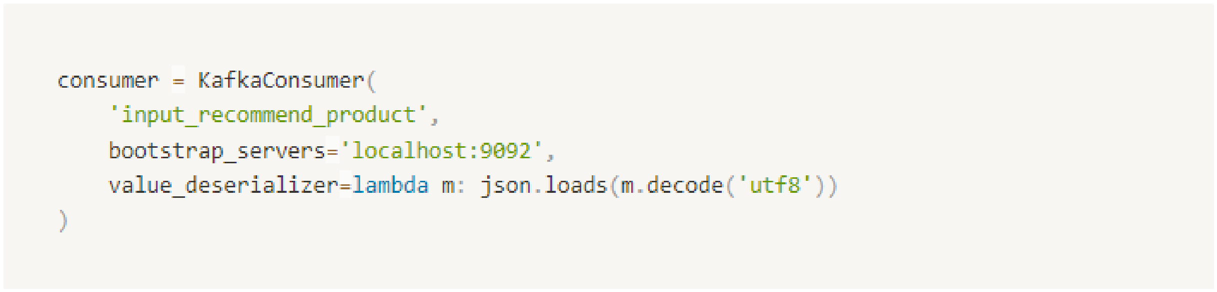
Call the trainingModel function and the predictionsModel function and calculationAccuracy function. Figure 4.19



```
model = trainingModel(new_training_data)

# Evaluate model on test instances and compute test error
predictions = predictionsModel(model, new_test_data)

# Split regional result from test data and named labels
labels = new_test_data.map(lambda lp: lp.label)

Accuracy = calculationAccuracy(predictions, labels)

print('\n=========================================\n||\t\t\t\t\t||')

print('|| Test Accuracy = ', Accuracy, "||")

print('||\t\t\t\t\t||\n=========================================\n')
```

Figure 4.19: Call the trainingModel function and the predictionsModel function and calculationAccuracy function.

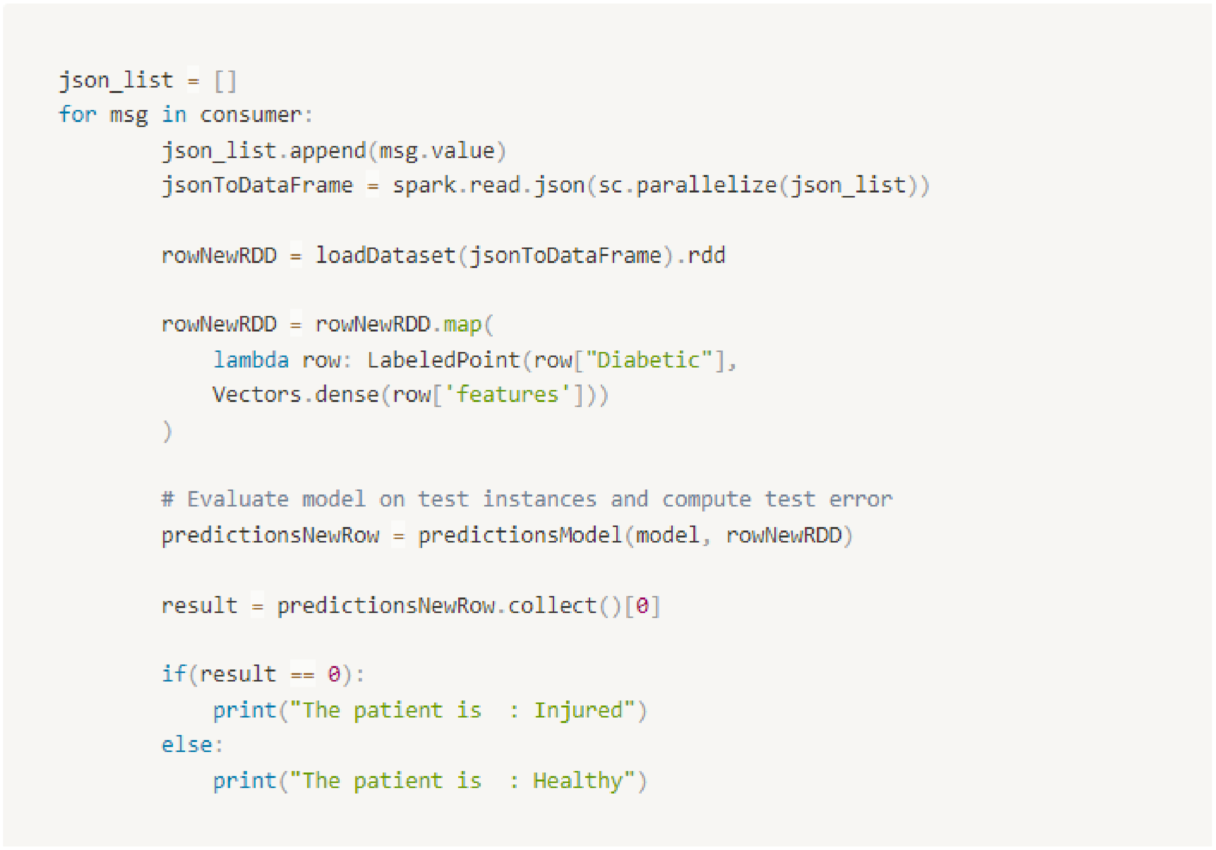
Contact with Apache Kafka. Figure 4.20



```
consumer = KafkaConsumer(
    'input_recommend_product',
    bootstrap_servers='localhost:9092',
    value_deserializer=lambda m: json.loads(m.decode('utf8'))
)
```

Figure 4.20: Code snippet Contact Spark with Apache Kafka.

Receiving data in JSON format and converting it to RDD format and sending it to the form to return to us the patient's condition, either injured or healthy with the time of implementation. Figure 4.21



```
json_list = []
for msg in consumer:
        json_list.append(msg.value)
        jsonToDataFrame = spark.read.json(sc.parallelize(json_list))

        rowNewRDD = loadDataset(jsonToDataFrame).rdd

        rowNewRDD = rowNewRDD.map(
            lambda row: LabeledPoint(row["Diabetic"],
            Vectors.dense(row['features']))
        )

        # Evaluate model on test instances and compute test error
        predictionsNewRow = predictionsModel(model, rowNewRDD)

        result = predictionsNewRow.collect()[0]

        if(result == 0):
            print("The patient is  : Injured")
        else:
            print("The patient is  : Healthy")
```

Figure 4.21: Code snippet converting data from JSON to RDD.

4.5 experimentation

The table 4.3 below shows the typical training time on about 15,000 lines of data with the working environment settings used and how long the result can be returned in real-time.

Lines of data	Kafka nodes	Topic partitions	Spark workers	time training	Accuracy	Time predict
15000	1	3	1	9.149	92.13%	0.024s

Table 4.3: typical training time on about 15,000 lines of data with the working environment settings used

General Conclusion

The amount of healthcare data is constantly increasing over time at alarming rates in different and inconsistent data sources. To improve patient outcomes and create a scalable real-time health prediction system. A streaming computing platform is needed. However, this large volume of data can no longer be collected, processed, stored and exploited by traditional IT solutions that combine physical infrastructures with relational databases.

Based on the previously discussed challenges, there are many errors in traditional IT in scaling with devices in parallel which is not suitable for dealing with the growing data (dataset used 15000 records). In this paper, real-time health analytics and prediction system are proposed and tested, built around open source big data technologies. Data events come from healthcare devices through Kafka Stream. Using the Spark Streaming API, the processed system received relevant health data events through the DT (Random forest) health forecast app, sent an alert to caregivers, and stored the details in a distributed database. The stored result will be queried for healthcare data analytics and streaming reports.

Building a real-time, distributed health analytics system using traditional analytics tools is very complex, requires a variety of skills, intensive and more expensive software, and a significant amount of time and money. However, using open source big data techniques and efficient data mining techniques can easily perform the same task. With a slight modification, the system itself can predict other diseases, and it can also extend to other areas.

prospects: In future work, we aim to:

- Integrate real data sources, such as mobile devices, sensor data, and social media data into our system to interact with user requests.
- Applying our idea on the ground in one of the state hospitals.
- Operating the system in a cloud computing environment with a larger number of devices.

Bibliography

[1] Cheptsov A. "Hpc in big data age: An evaluation report for java-based data-intensive applications implemented with hadoop and openmpi." In: (2014), 175:180.

[2] Satyanarayana A. "Intelligent sampling for big data using bootstrap sampling and chebyshev inequality." In: (2014), pp. 1–6.

[3] Khalil M Abderrahmane Ed. "A new Internet of Things architecture for real-time prediction of various diseases using machine learning on big data environment." In: (2019). URL: https://sci-hub.se/10.1186/s40537-019-0271-7.

[4] Swami A Agrawal R Imieliński T. "Mining association rules between sets of items in large databases." In: (1993), pp. 207–16.

[5] "Apache Mahout." In: (Sept. 2022). URL: http://mahout.apache.org.

[6] "Apache Spark." In: (Sept. 2022). URL: https://spark.apache.org.

[7] T. Ramanjaneyulu3 B. Sobhan Babu1 K. Srikanth2 and I. Lakshmi Narayana4. "IoT for Healthcare." In: (2014). URL: https://www.researchgate.net/profile/Sobhanbabu-Badugu-2/publication/348929916_IoT_for_Healthcare/links/6017b7d145851517ef2ead71/IoT-for-Healthcare.pdf.

[8] Joel Rodrigues Bassirou Diène a, Ousmane Diallo, EL Hadji Malick Ndoye a, and Valery V. Korotaev d. "Data management techniques for Internet of Things." In: (2020). URL: https://www.sciencedirect.com/science/article/abs/pii/S088832701930785X.

[9] de Lima BSLP Beckmann M Ebecken NFF and Costa MA. "A user interface for big data with rapidminer." In: (2014), pp. 173–182. URL: http://www.slideshare.net/RapidMiner/a-user-interface-for-big-data-with-rapidminer-marcelo-beckmann.

[10] StojkoskaKire V.Trivodaliev Biljana L.Risteska. "A review of Internet of Things for smart home: Challenges and solutions." In: (2016). URL: https://www.sciencedirect.com/science/article/abs/pii/S095965261631589X.

[11] Carey MJ Bu Y Borkar VR, Rosen J, Polyzotis N, Condie T, Weimer M, and Ramakrishnan R. "Scaling datalog for machine learning on big data." In: (2012). URL: https://dblp.uni-trier.de/rec/journals/corr/abs-1203-0160.html.

[12] Luca C. "Survey of Big Data sizes in 2021." In: (2022). URL: https://arxiv.org/pdf/2202.07659.pdf.

[13] Ren-gen Huang Chang-le Zhong Zhen Zhu. "Study on the IOT Architecture and Access Technology." In: (2017). URL: https://ieeexplore.ieee.org/abstract/document/8253048.

[14] Rusu F Cheng Y Qin C. "GLADE: big data analytics made easy." In: (2012), pp. 697–700.

[15] Han-Chieh Chao Chun-Wei Tsai Chin-Feng Lai and Athanasios V. Vasilakos. "Big data analytics: a survey." In: (2015). URL: https://journalofbigdata.springeropen.com/counter/pdf/10.1186/s40537-015-0030-3.pdf.

[16] Tam E Cooper BF Silberstein A, Ramakrishnan R, and Sears R. "Benchmarking cloud serving systems with ycsb." In: (2010), pp. 143–154.

[17] Duffield N Cormode G. "Sampling for big data: a tutorial." In: (2014), pp. 1975–1975.

[18] Davis KC Cuzzocrea A Song IY. "Analytics over large-scale multidimensional data: The big data revolution!" In: (2011), pp. 101–104.

[19] Laney D. "3D data management: controlling data volume, velocity, and variety." In: (2001). URL: http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf.

[20] Talia D. "Clouds for scalable big data analytics." In: (2013), pp. 98–101.

[21] Rachid Zagrouba1 Fahd Alhaidari1 Atta Rahman1. "Cloud of Things: architecture, applications and challenges." In: (2020).

[22] Czerwinski M Fisher D DeLine R and Drucker S. "Interactions with big data analytics." In: (2012), pp. 50–9.

[23] Jaber Alwidian Flasteen Abuqabita Razan Al-Omoush. "A Comparative Study on Big Data Analytics Frameworks, Data Resources and Challenges." In: (2019). URL: https://pdfs.semanticscholar.org/6398/f4b05069651c7eb59062964e7451a09f017b.pdf.

[24] Press G. "$16.1 billion big data market: 2014 predictions from IDC and IIA, Forbes." In: (2013). URL: http://www.forbes.com/sites/gilpress/2013/12/12/16-1-billion-big-data-market-2014-predictions-from-idc-and-iia.

[25] Hu M Ghazal A Rabl T, Raab F, Poess M, Crolotte A, and Jacobsen HA. "BigBench: Towards an industry standard benchmark for big data analytics." In: (2013), pp. 1197–1208.

[26] "GridMix." In: (Sept. 2022). URL: http://hadoop.apache.org/docs/r1.2.1/gridmix.html.

[27] Bingming Wang Guoli Cheng Shi Ying and Yuhang Li. "Efficient Performance Prediction for Apache Spark." In: (2020). URL: https://sci-hub.se/10.1016/j.jpdc.2020.10.010.

[28] V Mnssvkr Gupta Harika Devi Kotha1*. "IoT Application, A Survey." In: (2018). URL: https://www.researchgate.net/profile/Harika-Kotha/publication/325116647_IoT_Application_A_Survey/links/5b0cea0ba6fdcc8c25366a29/IoT-Application-A-Survey.pdf.

[29] Lopes N Hasan S Shamsuddin S. "Soft computing methods for big data problems." In: (2013), pp. 235–247.

[30] Chua T-S Hu H Wen Y and Li X. "Toward scalable systems for big data analytics: a technology tutorial." In: (2014), pp. 652–87.

[31] Adler M Jun SW Fleming K and Emer JS. "Zip-io: architecture for application-specific compression of big data." In: (2012), pp. 343–351.

[32] Borne K. "Top 10 big data challenges a serious look at 10 big data v's." In: (2014). URL: https://www.mapr.com/blog/top-10-big-data-challenges-look-10-big-data-v.

[33] Ashwin Karale. "The Challenges of IoT Addressing Security, Ethics, Privacy, and Laws." In: (2021). URL: https://www.sciencedirect.com/science/article/abs/pii/S2542660521000640.

[34] Goudar R Katal A Wazid M. "Big data: issues, challenges, tools and good practices." In: (2013), pp. 404–409.

[35] Alhajj R Kaya M. "Genetic algorithm based framework for mining fuzzy association rules." In: (2005).

[36] Babu AV kranthi Kiran B. "A comparative study of issues in big data clustering algorithm with constraint based genetic algorithm for associative clustering." In: (2014).

[37] Murty MN Krishna K. "Genetic k-means algorithm." In: (1999).

[38] Marta Biegańska 1ORCID Krzysztof Wójcicki 1 ORCID and Beata Paliwoda 2ORCID and-Justyna Górna 2. "Internet of Things in Industry: Research Profiling, Application, Challenges and Opportunities—A Review." In: (2022). URL: https://www.mdpi.com/1996-1073/15/5/1806/htm#overview.

[39] Aad I Laurila JK Gatica-Perez D, Blom J, Bornet O, Dousse O Do T, Eberle J, and Miettinen M. "The mobile data challenge: big data for mobile computing research." In: (2012), pp. 1–8.

[40] Lee JH Lee J Hong S. "An efficient prediction for heavy rain from big weather data using genetic algorithm." In: (2014), pp. 1–25.

[41] Liu X Lu R Zhu H, Liu JK, and Shao J. "Toward efficient and privacy-preserving computing in big data era." In: (2014), pp. 46–50.

[42] Van Rijmenam M. “Why the 3v’s are not sufficient to describe big data.” In: (2013). URL: http://www.bigdata-startups.com/3vs-sufficient-describe-big-data.

[43] Bik AJ Malewicz G Austern MH, Dehnert JC, Horn I, Leiser N, and Czajkowski G. “Pregel: A system for large-scale graph processing. In: Proceedings of the ACM SIGMOD International Conference on Management of Data.” In: (2010), pp. 135–146.

[44] Mohammad Alfailakawi Maryam AlJame Imtiaz Ahmad. “Apache Spark Implementation of Whale Optimization Algorithm.” In: (2020). URL: https://sci-hub.se/10.1007/s10586-020-03162-7.

[45] andNajah Abu Ali 2 Mervat Abu-Elkheir 1 Mohammad Hayajneh. “Data Management for the Internet of Things: Design Primitives and Solution.” In: (2013). URL: https://www.mdpi.com/1424-8220/13/11/15582/htm.

[46] Norma L F Muhammad S Ganjar A and Jongtae R. “Performance Analysis of IoT-Based Sensor, Big Data Processing, and Machine Learning Model for Real-Time Monitoring System in Automotive Manufacturing.” In: (2018). URL: https://www.mdpi.com/1424-8220/18/9/2946/htm.

[47] “PigMix.” In: (Sept. 2022). URL: https://cwiki.apache.org/confluence/display/PIG/PigMix.

[48] Indrajit Mukherjee Rakesh Raja and Bikash Kanti Sarkar. “Review Article: A Systematic Review of Healthcare Big Data.” In: (2020). URL: https://downloads.hindawi.com/journals/sp/2020/5471849.pdf.

[49] Baraniuk RG. “More is less: signal processing and the data deluge.” In: (2011), pp. 717–9.

[50] Dobra A Rusu F. “GLADE: a scalable framework for efficient analytics.” In: (2012), pp. 1–6.

[51] Sinanc D Sagiroglu S. “Big data: a review.” In: (2013), pp. 42–47.

[52] Siddharth Tripathi Somayya Madakam R. Ramaswamy. “Internet of Things (IoT): A Literature Review.” In: (2015). URL: https://www.scirp.org/html/56616_56616.htm?pagespeed=noscript.

[53] Sucha Smanchat2 Suwimon Vongsingthong1. “A Review of Data Management in Internet of Things.” In: (2015). URL: https://rtt.kku.ac.th/ejournal/pa_upload_pdf/562694.pdf.

[54] “TeraSoft.” In: (Sept. 2022). URL: http://sortbenchmark.org.

[55] Jain N Thusoo A Sarma JS, Shao Z, Chakka P, Anthony S, Liu H, Wyckoff P, and Murthy R. “Hive: a warehousing solution over a map-reduce framework.” In: (2009).

[56] “TPC, transaction processing performance council.” In: (Sept. 2022). URL: http://www.tpc.org.

[57] SamiKara WenLi. "Methodology for Monitoring Manufacturing Environment by Using Wireless Sensor Networks (WSN) and the Internet of Things (IoT)." In: (2017). URL: https://www.sciencedirect.com/science/article/pii/S2212827116313427.

[58] David B William D Lorien C, Melissa A, Jeanie A, Megumi I, and Sohail A. "Digital Media for Behavior Change: Review of an Emerging Field of Study." In: (2022). URL: https://www.mdpi.com/1660-4601/19/15/9129/htm.

[59] Wu G-Q Wu X Zhu X and Ding W. "Data mining with big data." In: (2014), pp. 97–107.

[60] Wunsch D Xu R. "Clustering." In: (2009), pp. 717–9.

[61] Li J Xue Z Shen G, Xu Q, Zhang Y, and Shao J. "Compression-aware I/O performance analysis for big data clustering." In: (2012), pp. 45–52.

[62] Zhong C Yang C Zhang X, Liu C, Pei J, Ramamohanarao K, and Chen J. "A spatiotemporal compression based approach for efficient big data processing on cloud." In: (2014).

[63] Peiying Liang Yangqing Zhu. "Research on key technologies of data processing in internet of things." In: (2017). URL: https://iopscience.iop.org/article/10.1088/1742-6596/887/1/012047/pdf.

[64] Huang ML Zhang J. "5Ws model for big data analysis and visualization." In: (2013), pp. 1021–1028.

[65] Behrisch M Zhang L Stoffel A, Mittelstadt S, Schreck T, Pompl R, Weber S, Last H, and Keim D. "Visual analytics for the big data era—a comparative review of state-of-the-art commercial systems." In: (2012), pp. 173–182.

[66] Liu X Zhao JM Wang WS and Chen YF. "Big data benchmark - big DS." In: (2014), pp. 49–57.

[67] Tang W Zou H Yu Y and Chen HM. "Improving I/O performance with adaptive data compression for big data applications." In: (2014), pp. 1228–1237.