The Journal of Supercomputing (2022) 78:7078–7105
https://doi.org/10.1007/s11227-021-04166-9

Hybrid-based framework for COVID-19 prediction via federated machine learning models

Ameni Kallel[3,4] · **Molka Rekik**[1,3] · **Mahdi Khemakhem**[2,3]

Accepted: 19 October 2021 / Published online: 5 November 2021

Abstract

The COronaVIrus Disease 2019 (COVID-19) pandemic is unfortunately highly transmissible across the people. In order to detect and track the suspected COVID-19 infected people and consequently limit the pandemic spread, this paper entails a framework integrating the machine learning (ML), cloud, fog, and Internet of Things (IoT) technologies to propose a novel smart COVID-19 disease monitoring and prognosis system. The proposal leverages the IoT devices that collect streaming data from both medical (e.g., X-ray machine, lung ultrasound machine, etc.) and non-medical (e.g., bracelet, smartwatch, etc.) devices. Moreover, the proposed hybrid fog-cloud framework provides two kinds of federated ML as a service (federated MLaaS); (i) the distributed batch MLaaS that is implemented on the cloud environment for a long-term decision-making, and (ii) the distributed stream MLaaS, which is installed into a hybrid fog-cloud environment for a short-term decision-making. The stream MLaaS uses a shared federated prediction model stored into the cloud, whereas the real-time symptom data processing and COVID-19 prediction are done into the fog. The federated ML models are determined after evaluating a set of both batch and stream ML algorithms from the Python's libraries. The evaluation considers both the quantitative (i.e., performance in terms of accuracy, precision, root mean squared error, and $F1$ score) and qualitative (i.e., quality of service in terms of server latency, response time, and network latency) metrics to assess these algorithms. This evaluation shows that the stream ML algorithms have the potential to be integrated into the COVID-19 prognosis allowing the early predictions of the suspected COVID-19 cases.

Keywords COVID-19 pandemic · Machine learning · IoT devices · Batch/streaming data · Hybrid fog-cloud federation · Federated MLaaS · Real-time prediction · Decision-making · Quantitative and qualitative evaluation

✉ Ameni Kallel
kallel.ameni@gmail.com

Extended author information available on the last page of the article

1 Introduction

Healthcare is one of the important application fields that require real-time and accurate results. Technologies including big data, Internet of Things (IoT), cloud and fog computing [1] have gained significance due to their available abilities to provide diverse services based on latency-sensitive or real-time applications [2–4]. Since the manual processing has not been effective, the use of the artificial intelligence (AI) in healthcare [5] has become prominent for monitoring, prognosis and diagnosis purposes [6–8]. Regarding the continuous COronaVIrus Disease 2019 (COVID-19) [9] pandemic growth across the world, several researchers are attempting to find solutions for exploring accurately the infected persons and isolate them to reduce the pandemic spread. The machine learning (ML)-based systems can improve the decision-making for clinicians and consequently limit the spread of this pandemic. The literature survey focused on discussing the learning-based COVID-19 combating proposals. In [10–12], the authors proposed learning-based approaches for detecting the facemask wearing state based on integrated monitoring cameras and ML techniques. Other works [13–17] dealt with the COVID-19 prediction and the early prognosis of this disease. They suggested COVID-19 medical image recognition models based on clinical and chest radiological imaging such as computed tomography (CT) and X-ray. Some other studies [18–20] presented ML-based framework for predicting whether a suspected person is COVID-19 infected or not. Their frameworks adopted a set of IoT devices such as temperature, heartbeat, oxygen saturation monitor, etc., which might collect health data and help to monitor the users in real time. However, such frameworks, were based on traditional batch ML models, treated only the historical existing data. Although our approach considers both the historical and streaming data collected from different medical devices like X-ray machine, lung ultrasound (LUS) machine, etc., as well as non-medical devices (e.g., drones, wearable devices like smartwatch, bracelet, etc.), in this work, we propose a new framework for a smart monitoring system detecting the suspected COVID-19 infected people as well as other critical diseases' patients such as pneumocystis, legionella, streptococcus, etc. As the data stream learning differs from the traditional batch learning (i.e., the environment settings are mainly different) [21], the proposed framework enables collaborative learning into a hybrid (i.e., fog/cloud) federation system, whereas we adopt three kinds of federated ML as a service (federated MLaaS). In fact, we provide (i) a batch MLaaS implemented on the cloud environment for the long-term decision making, (ii) a stream MLaaS installed on the fog environment for the short-term decision making, and (iii) the both ML types simultaneously based on their dependencies and their connections. Indeed, this paper is an extension of a previous publication [22]. It extends the original work in various points:

- The framework is adapted to be customized for the COVID-19 disease monitoring and prognosis. In fact, to customize the previous framework, we were based on the expert (i.e., physicians and medical staff) experiences through sharing a survey with them.

- According to the survey's results, the framework enables the data preprocessing and features selection in order to extract relevant indicators for the COVID-19 disease classification. It is worth mentioning that the data are collected from the medical as well as the non-medical devices.
- The learning-based framework maintains an accurate decision-making model shared between the two federated stream MLs (i.e., the same learning model is adopted by the stream ML existing in the cloud layer as well as by the one installed on the fog layer). Moreover, the proposal ensures an accurate long-term decision-making using the federated batch ML, which is installed on cloud layer.
- The proposed framework is able to learn and train the collected data incrementally over time as well as historical stored data. In the best of our knowledge, our proposal is the first one handling both batch and streaming data.
- The evaluation of the proposed framework demonstrates its performance in terms of accuracy, response time, precision, etc., for COVID-19 monitoring and prognosis.

In this work, we propose twofold evaluation. The first focuses on selecting the best-performing algorithm/model for the batch ML among the *Scikit-learn*'s [23] widely used ones. For doing so, we adopt the two different data splitting strategies (i) K-fold cross-validation (KCV) and (ii) no-shuffle train/test split with a train/test split of *80/20*. The experiment results demonstrate that the most accurate predictions are those obtained while adopting *no-shuffle train/test split* strategy and *MLP Classifier*. In fact, this quantitatively outperforms other strategies and algorithms by providing the best values in terms of performance metrics such as accuracy, precision, root-mean-squared error (RMSE), and F1 score. The second evaluation deals with quantitative and qualitative assessment of the *River*'s [24] stream ML models to find the best algorithms in terms of performance and quality of services (QoS) parameters, respectively. According to [25], the QoS metrics includes the server latency, response time, and network latency. The results are given by the following for both the multi- and binary classification. It is worthy mentioned that the multi-class classification might predict normal, COVID-19, and non-COVID-19 pneumonia cases, while the binary classification might categorize the patients into COVID-19 vs. non-COVID-19 cases. The results demonstrate that by considering the multi-classification: (i) *Hoeffding Adaptive Tree Classifier* is the most performing in terms of accuracy, RMSE, precision, and response time with 90.12%, 38.39%, 77.6%, 48 milliseconds, respectively and (ii) *Logistic Regression* algorithm is the best-performing algorithm in term of server latency with 0.055 milliseconds. While taking into account the binary classification, the best values of accuracy, RMSE, precision and training time are given as 28.79%, 77.75%, 91.71% and 36.5561 seconds, respectively. The remainder of the paper is structured as follows: the next section, on the first hand, defines the basic concepts related to the learning approach. On the second hand, it discusses some relevant works dealing with adopting the fog, cloud, IoT and deep learning in the battle against the COVID-19 pandemic. Section 3 presents the survey, which we are based in order to propose a customized framework for COVID-19 monitoring and prognosis. The proposed framework is given in Sect. 4. The implementation of our framework and its evaluation are given

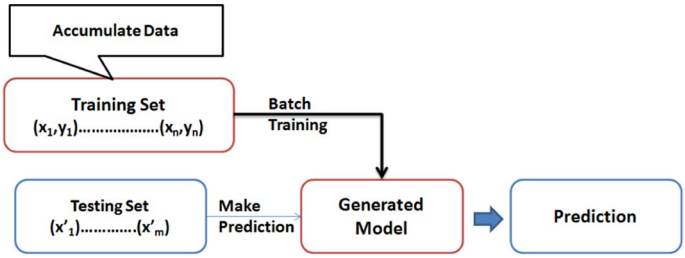


Fig. 1 Traditional batch learning technique

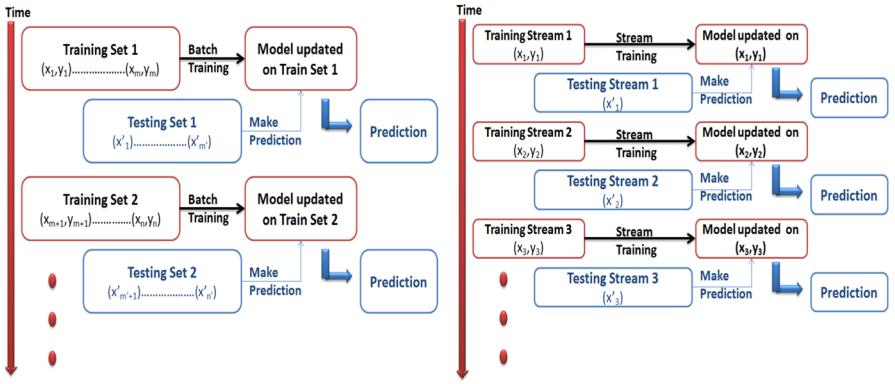
in Sect. 5. Finally, Sect. 6 concludes the paper and provides some perspectives for future work.

2 Background and related work

2.1 Background

Nowadays, the ML has acquired a favorable position within the healthcare arena. The ML is beneficial since its capability to detect patterns and run predictive analysis. In data mining, data are divided into three sets of sequences [26], such as (i) training set for training the ML model by using one or more algorithm(s), (ii) validation set for predicting the ML model and finding the best one, and (iii) testing set for predicting final model performance. Indeed, each sequence contains (x_i,y_i), where $x_i = x_{i,1}, x_{i,2}, \ldots x_{i,n}$ and $y_i = y_{i,1}, y_{i,2}, \ldots y_{i,n}$. In fact, the ML adoption serves to build a classifier for predicting a label y_{pred} for a given x. In the literature, researchers have analyzed three ML approaches: traditional batch learning, incremental batch learning, and streaming learning [26].

1. Traditional batch learning: It is characterized by an offline phase of training the classifier/algorithm once the complete training set of data is used (see Fig. 1). Afterward, such model is deployed online. In the second phase, the online testing phase, the predictions are made while testing the set of data. Therefore, the traditional batch model is treated as a static object. In order to learn from new data, the model has to be retrained from scratch. Moreover, training the full dataset requires resources with high computing capabilities in terms of CPU, memory size, disk size, network I/O, etc. [27]. It is worth mentioning that adopting a traditional batch learning approach is not the best choice to do if the amount of data is huge (i.e., in the case of big data). In addition, the traditional batch algorithms cannot ensure a dynamic ML model adaptation (i.e., according to new data arrival).
2. Incremental batch learning: This approach is able to override above-mentioned cons. Thus, the ML algorithm loads a subset of the data forming a batch, and when the batch contains all the dataset, the algorithm loops over the batch in order to run the training step on that batch (see Fig. 2a). Consequently, the testing step



(a) Incremental batch learning technique **(b)** Streaming learning technique

Fig. 2 Incremental learning technique

occurs between a set of batches [28]. However, it should define a parameter to specify the size of the batch for this learning approach; in fact, the inference is only possible after a batch has been completed [28] as well as the model cannot be learned from the arrival of the new data but until a new batch has been completed [21]. Moreover, trained models should be removed afterward to make space for new ones that may affect the ability of these algorithms to immediately learn from new data arrival.

3. Streaming learning: Here, the ML algorithm use the instance-incremental models [21]. In fact, the algorithms learn from each training stream upon the data arrival in order to quickly auto-adapt. In contrast with the traditional batch learning models, the streaming ones are appropriate for adopting when we have a huge dataset. Indeed, the ML does not access to dataset at once but the data is continuously loaded as a stream. Therefore, the ML must learn the flow stream one by one, and update the model each time a new stream arrives (see Fig. 2b). Moreover, this does not require important computing resources capabilities; since once the ML algorithm learns from the new stream, the ML deletes it and keeps free its associated space. Consequently, the streaming learning might optimize the resources utilization.

2.2 Related work

The COVID-19 pandemic becomes one of the biggest threats facing humanity. Since its emergence in December 2019, several research studies have addressed the benefits, threats and challenges of new technologies such as fog, cloud, IoT and deep learning in the battle against the COVID-19 pandemic. Some researches focused on the identification of facemask wearing state. In this context, Loey et al. [10] proposed a facemask detection model based on integrated monitoring cameras. The

model detected the people who did not wear facemasks by applying a deep transfer learning. The authors defined three classical ML classifiers such as SVM, Decision Tree, and Ensemble. In [11], the authors utilized deep learning model for automatic facemask wearing detection based on three categories: correct facemask wearing, incorrect facemask wearing, and no facemask wearing. The proposed model achieved 98.70% accuracy. Similarly, Chowdary et al. [12] introduced a facemask detection model based on deep transfer and ML techniques. The model ensured an accuracy obtained by 99.9% in training and 100% during testing. Other research works proposed COVID-19 medical image recognition models. Nora El-Rashidy [13] suggested a neural network-based deep learning model. The proposed model classified the patients' X-ray scan images to two classes, COVID-19 and normal classes. The model was trained on COVID-19 dataset, and the evaluation performance carried out was 97.78% accuracy, 97.2% F1 score, 97.4% precision and 97.5% recall. Similarly, Ozturk et al. [14] proposed a DarkNet model trained on 125 chest X-ray. The proposed model provided predictions with 98.08% accuracy for a binary class classification (COVID-19 vs. No-Findings classes) and 87.02% for a multi-class classification (COVID-19 vs. No-Findings vs. Pneumonia). Along with chest X-rays, computerized tomography (CT) scans play a crucial role in effectively discriminating against COVID-19 [19, 29, 30]. In this context, several research studies [15, 16] proposed a deep learning model to classify COVID-19 cases by considering their chest CT images. In [15], the proposed model achieved 99.68% accuracy with tenfold cross-validation using a SVM classifier. Despite all the above proposals demonstrated optimal results regarding the combat against COVID-19 pandemic by training models applicable with clinical trials, they adopt the classical ML classifiers (i.e., batch ML classifiers) having a set of drawbacks that we mentioned above (see previous subsection). To the best of our knowledge, our work is the first one that considers both the batch and stream ML classifiers. The batch one is for long-term predictions and analytical dashboards, while the second ML is for short-term (i.e., real-time) predictions regarding COVID-19 prognosis. Besides, it is so important to consider (i) the Internet of Medical Things (IoMT) allowing interconnection between medical devices (e.g., CT scans and/or X-ray imaging devices found in all hospitals), patients and medical systems [20], and (ii) the integration of fog/cloud environments to support the stream ML models for a real-time predictions in the hospitals. Other literature studies explored the relationship between the risk of to be COVID-19 infected and other factors, such as diabetic disease, smoking, respiratory/lung disease, cardiovascular disease and hypertension. [31, 32]. Moreover, the high-risk patients should be frequently monitored even if they are outside hospitals by checking their vital signs such as temperature and oxygen saturation. [13]. Several studies suggested IoT frameworks for determining suspected COVID-19 cases [18, 19]. Thus, the end-users might use a set of sensors on the user's body to collect the real-time symptom data such as temperature, heart rate and cough quality. Kumar et al. [20] proposed an artificial intelligence-based framework, which enables the data collection not only from wearable sensors and drones. The proposed framework enables, in addition, the exchange between edge, fog, and cloud servers in order to share computing resources, data, and analytics. The authors integrated a drone-level federated learning for analyzing changing COVID-19 symptoms and strategies. This

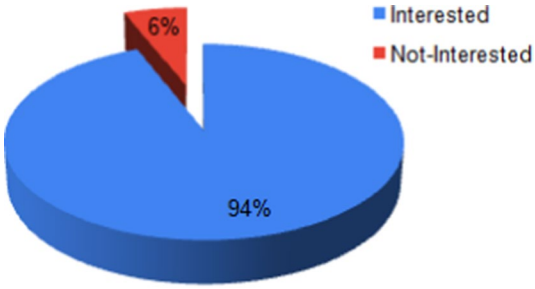


Fig. 3 Experts feedback regarding adopting a COVID-19 automatic monitoring and detecting system

algorithm consists of detecting the patient's body temperature, making decision when critical value is detected, and sharing the training data with the edge, fog, or cloud computing environments. However, these works did not detail how the decision were made using the ML models, how the MLs were deployed in the three layers, how their frameworks supported the connectivity between the edge, fog and cloud computing, and how they evaluated their frameworks' performances in terms of accuracy, precision, etc. To the best of our knowledge, all the existing studies focused on the batch learning approaches to predict if whether a person suspected COVID-19 infected or not. However, our approach is the first one that considers both the batch and streaming learning approaches for more accurate COVID-19 disease predictions.

3 Survey

The IoT devices are the primary sources of collecting the real-time symptom data from COVID-19 suspected infected users. The collected data must be preprocessed to filter the relevant information that can help in COVID-19 monitoring and prognosis. In a previous work [22], the authors proposed an IoT-fog-cloud-based architecture for smart systems supporting the distributed communication, real-time stream processing, and multi-application execution. In this work, we focus on customizing such architecture and proposing a framework tailored for COVID-19 monitoring and prognosis. For doing so, we conducted a survey of 19 questions in which 50 experts of the medical staff were engaged. The survey aims to find out the experts' requirements and recommendations to combat the COVID-19 pandemic by limiting its spread. Thus, the first question is about whether they are interested by adopting an automatic system for detecting suspected COVID-19 cases. As shown in Fig. 3, 94% of the medical staff are interested by such system.

The second question is what about connecting an equipped object through sensors (e.g., smartwatch, bracelet) that the suspected infected person may wear? Figure 4 depicts that the doctors find that this idea is more effective than sharing a questionnaire with the COVID-19 suspected infected people. Moreover, they suggest integrating (i) a Global Positioning System (GPS) device to locate individuals who may be contaminated by transmission and (ii) a set of sensors helping to detect whether the disease symptoms appear at the individuals. As shown

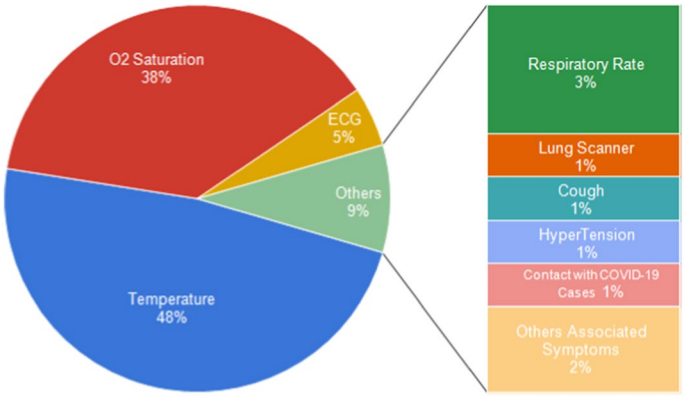


Fig. 4 Experts feedback regarding integrating sensors to detect COVID-19 symptoms

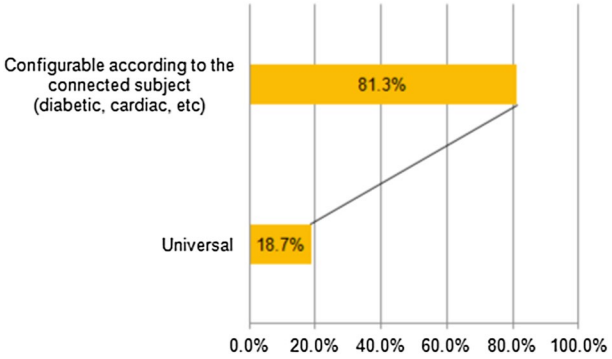


Fig. 5 Experts feedback regarding adopting a customized monitoring system based on the patient's health state

in Fig. 4, about 48% of the experts go to integrate a temperature sensor, 38% of them desire to use a saturation sensor, 5% require to utilize an electrocardiogram (ECG) sensor, and 9% of them propose other types of sensors such as respiratory rate, lung scanner and cough sensor.

In addition, physicians recommend remotely monitoring the health state of their registered patients through an alert system that may send, in the case of COVID-19 symptom detection, a real-time notification either to them or to the COVID-19 pandemic center. As well, they require that the system can enable to launch a call to the connected person through their bracelet in order to assign with him/her an appointment to do a rapid test. As shown in Fig. 5, about 81.3% of the medical staff recommend the alert system and suggest that it should be configurable and customized according to the patient's health state (i.e., if the patient has one or more chronic disease(s) such as diabetic, renal failure, heart, etc.).

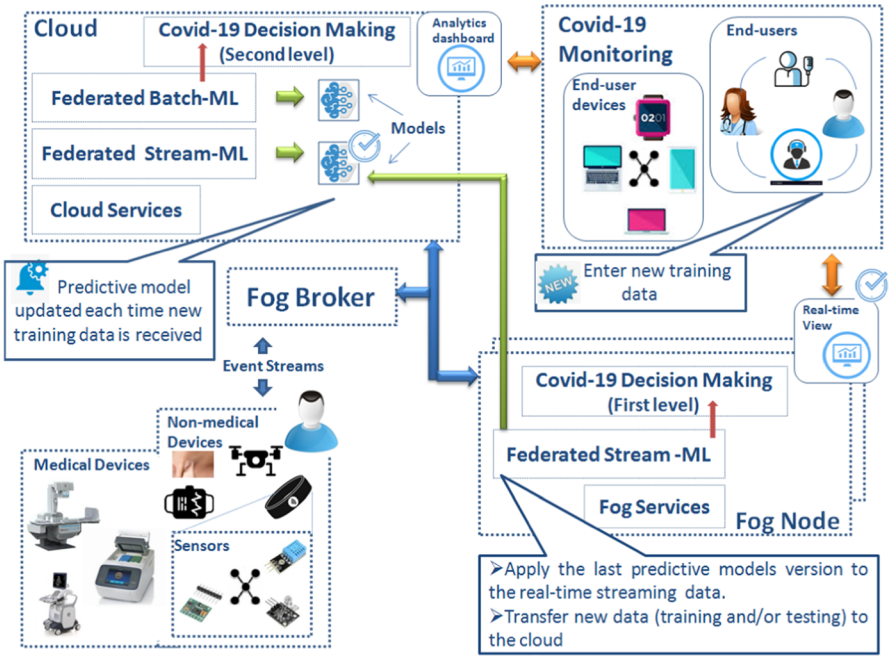


Fig. 6 Customized learning-based framework for COVID-19 monitoring and prognosis

Obviously, the survey results help us in the data preprocessing phase, whereas we take into consideration the different sensor values as well as the health state of the patient, whether the patient was traveling or not, and the people who have been in contact with the suspected infected person.

4 Customized framework for COVID-19 monitoring and prognosis

As mentioned above, we adopt the previous architecture [22] in order to propose a customized framework for COVID-19 monitoring and prognosis. In fact, we define a hybrid framework dealing with several hardware and software components described as follows.

4.1 Hardware components

The hardware components include the IoT devices, end-user devices, fog broker, fog node, and cloud datacenters.

1. IoT device: The IoT device may be either a medical or a non-medical device (see Fig. 6). A medical device such as (i) a real-time polymerase chain reaction (RT-PCR) machine that detects the COVID-19 virus in upper and lower respiratory

specimens, or (ii) X-ray machine like a CT scan, or (iii) LUS machine, which provides X-ray images depicting a possible aspect of COVID-19 disease. It is worth mentioning that the medical devices are currently widely used by researchers since they achieve accurate results of COVID-19 prediction [13–16]. Therefore, we adopted the medical devices as key sources of training data. The non-medical device like a bracelet, smartwatch, drone, etc., may be connected by several sensors and equipped with a rechargeable battery. Indeed, the non-medical devices help a COVID-19 suspected person in storing the sensor values and transferring them to the fog layer for real-time data processing and decision making. Hence, an AI module classify whether these values are normal or require more treatment (i.e., It shall send a notification to the connected person to do a rapid test). It is worth mentioning that multiple sensors can be integrated into a non-medical device such as (i) temperature sensor, (ii) passive GPS location tracking, (iii) microphone may be used to facilitate the communication between the person and the medical staff, (iv) ring-type pulse oximeter sensor might monitor both heart rate and oxygen saturation in the body, (v) cough sensor to detect the nature, duration and time of the cough, (vi) blood pressure sensor might track changes in blood pressure, and *(vii)* respiratory rate sensor enables breathing rate monitoring. All of these devices expose their services and data over the standard REST and/or Pub/Sub Application Programming Interfaces (APIs) and this complies with the Web of Things (WoT) architecture.

2. Fog broker: The fog broker represents a gateway between the IoT devices, fog, and cloud computing environments (see Fig. 6). Its main role is (i) collect real-time streams from connected IoT devices, (ii) choose the appropriate node for the real-time and/or batch processing, and (iii) stream data to the appropriate node (usually to the fog node if the streams contain testing data while to the cloud node whether the streams move training data).
3. Fog node: It is a worker or a computer node, which performs the tasks received from the fog broker. As shown in Fig. 6, it serves for running the stream MLaaS for the real-time decision-making. It should mention that the federated stream ML wherever it is installed at the fog or cloud environment, it behaves as follows:

    - If the streams move training data (i.e., the data collected from the medical devices or staff), the stream ML updates the prediction model, which is stored in the cloud.
    - If the streams transmit testing data (i.e., the data collected from the non-medical devices), the stream ML applies the last generated prediction model on such data for real-time diagnosis and prognosis.
    - The stream ML transfers the training and/or testing data to the cloud in order to store it for a later usage by the batch ML.

4. Cloud: As shown in Fig. 6, the cloud system provides for the end users as on demand services; a stream ML having the same functionalities as one exists in the fog, a batch ML, and both. The batch MLaaS reacts with the historical dataset for a long-term decision making (e.g., inferring the importance of COVID-19

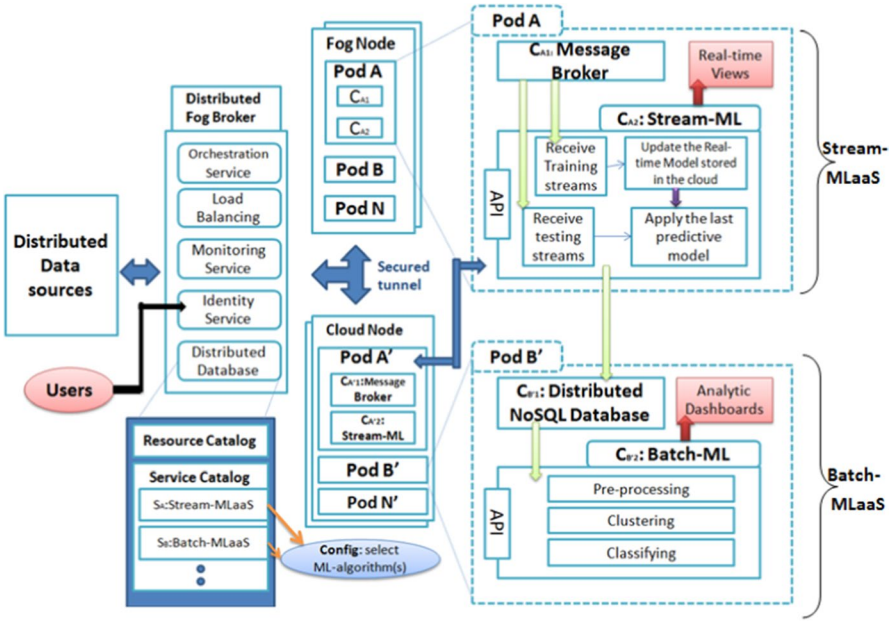


Fig. 7 Software components view of the proposed framework

symptoms, determining the most infected regions, etc.). In addition, the cloud stores all predictive models generated by these MLs.

5. End-user devices: It might be a personal computer, mobile phone, tablet, etc. used by the patient, the medical staff, the emergency medical aid service, COVID-19 rapid test center, etc. Such devices facilitate the interaction between the end-users and our system (e.g., display in real-time the analytic dashboards, send SMS/Email notifications, etc.). In this context, it should be noted that the end-users may enter new training data through their devices, which will be taken into account by stream ML and/or update existing data processed by batch ML (see Fig. 6).

4.2 Software components

The proposed framework includes a set of interrelated software components ensuring the (i) WoT-based interaction between the IoT devices, the fog and cloud layers that serves to solve the interoperability issues (see Fig. 7), (ii) resources management for running the federated MLs as a Service (MLaaS). Indeed, the computational tasks are distributed on the compute instances (i.e., virtual machine or docker container), (iii) orchestration service according to the load balancing, and (iv) reliability and security constraints. In the following, we detail the set of software components, which is offered as cloud and fog services.

Obviously, each service has its own API permitting the end users managing and adopting the service according to their requirements.

1. Security service: To improve the security level, we adopt the transport layer security (TLS) to encrypt the streams moving from the IoT devices to upstream servers running the ML. In fact, encryption is an effective technique for protecting both the training and testing data as it moves the public and/or private networks. Indeed, our framework consists of establishing a secured tunnel between the hardware components via a virtual private network (VPN) protocol like internet protocol security (IPSec) or a secure general purpose protocol like hypertext transfer protocol secure (HTTPS) or secure shell (SSH). In this regard, we think for creating a tunnel over SSH for the message queuing telemetry transport (MQTT). First, we carried out an IoT device agent and an SSH daemon on the remote device. Next, we configure the remote device with the MQTT topic subscription and the authentication by using public/private key pair in order to connect to the MQTT server via SSH without asking for a password. This ensures unique use of the keys in order to minimize additional server-side checks and also maximize the security level.
2. Distributed database service: Its main role is to store the two catalogs: resource catalog and service or application catalog. The resource catalog is maintained at the orchestration service, and it represents the holistic and abstracted view of the computing resources [22], while the service catalog illustrates information about our framework provided services and/or applications like their operations, their requirements and dependencies. The main services of the proposed framework are the stream MLaaS and batch MLaaS. Here, it should be mentioned that our work is the first one that aims at offering to the end-users a distributed stream ML, a distributed batch ML or both at the same time by basing on their dependencies and their connections, respectively. These federated MLaaS services include, obviously, a set of services such as predictive analytics, data mining, data visualization, APIs, etc. That's why, the end-users can get started with an ML-based system without having to install any software, or develop algorithms. (They can select one or more classification algorithm(s) according to the chosen ML type.) In addition, they may choose the most appropriate configuration fitting their requirements.
3. Identity service: It enables end-users to remain connected to several applications while ensuring the single sign-on (SSO) between the applications. That's why, the identity server federates identities between multiple heterogeneous and distributed systems (at the edge, fog and cloud computing layers). Such component allows the user authentication before accessing to any on demand provided service. For instance, users can use identity as a service (IDaaS) [33] by specifying connection with SQL database (e.g., Mysql), NoSQL database (e.g., Cassandra), lightweight directory access protocol (LDAP) or active directory (AD) user stores. Additionally, they can use MLaaS and link it with IDaaS in order to analyze user activity, assign risk to each access request, and create policies to be triggered when an abnormal behavior is detected.

4. Container orchestration service: It allows the management of the application services across multiple containers, the automation of the containers deployment within clusters, scaling the containers, and the real-time management of the end-user health thanks to the provided federated MLaaS. In the following, we detail further the federated MLaaS services:

   - Batch MLaaS: It is based on the micro-services architecture, and it is deployed in a Pod. The micro-services are deployed in containers; the first container includes a NoSQL database representing the dataset treated by the batch ML. In fact, the batch ML is deployed in a second container that can access to the dataset for the model generation enabling the pre-processing, clustering, classifying and building analytical dashboards.
   - Stream MLaaS: Similarly, it adopts a micro-services architecture. It is deployed by using two Pods into a hybrid fog/cloud environment. Each Pod contains a micro-service involving the message broker that receives real-time streams from distributed data-sources and then sends them to another micro-service representing by stream ML. The stream ML allows to receive training streams for updating the model in order to (i) apply the latest predictive model in real time to testing streams and (ii) display the real time analytics. The two Pods use a shared federated prediction model, which is stored into the cloud, whereas the real-time symptom data processing and COVID-19 prediction are done into the fog.

   The end-user is able to (i) deploy the stream MLaaS into a fog/cloud hybrid environment and the batch MLaaS into a public, private or hybrid cloud and (ii) define the interaction between them; so that the testing and training data generated by stream MLaaS should be stored in the database offered by batch MLaaS. The provided federated MLaaS are scalable services ensuring a high-performance level for building/updating batch/real-time predictive models. In this context, it should be noted that the container orchestration service mainly serves to (i) scale up/down the active fog/cloud nodes number according to the system requirements, (ii) improve the portability of workloads (i.e., services/applications), and (iii) efficiently balance between the workloads.
5. Load Balancing (LB): The adoption of the micro-services architecture requires an effective infrastructure services management in order to ensure the resource availability and scalability, and consequently to meet the end-users requirements. Precisely, when the containers are deployed on a cluster, the role of the LB is to balance the workload (especially, in the case if there are multiple containers on a single node being accessed on the same port), monitor the cluster and its containers, and continuously upgrade the micro-services inside the containers without services disruption.
6. Monitoring service: We should monitor the model generation and its performance in term of predictions. So, monitoring as a service should be a key service offered to the end users into the hybrid fog/cloud environment. That's why, our framework allows the end users to (i) ensure that the data is trust to be used in training, (ii) track the accuracy of their predictions at run time, and (iii) adjust the model and the input data while updating, adding, or removing features and/or metrics.

For example, the end-users, in particularly the medical staff, can specify that the predictive model shall use temperature, O2 saturation, and cough level features as the sensors detected COVID-19 symptoms. Over time, new COVID-19 symptoms may be discovered and old symptoms that may affect model performance degradation may be ignored. Therefore, in addition to the monitoring service, users can identify a new set of metrics and key performance indicators (KPIs) that are more directly correlated to the overall performance.

5 Proposed COVID-19 monitoring and prognosis system

5.1 Dataset preparation

Based on the above presented framework, we propose a COVID-19 monitoring and prognosis prototype. The first phase consists in collecting real-time symptom data. Therefore, we adopt the open COVID-19 dataset.[1] The dataset addresses COVID-19 and other respiratory viruses. It covers symptoms data, after a RT-PCR test, and classifies the prediction result as a binary 0 in case of normal case, 1 in case of COVID-19 case, or 2 in case of non-COVID-19 pneumonia case. In the given dataset, we select eight binary features such as sex, age, known contact with an infected individual, cough, fever, sore throat, shortness of breath, and headache. After the feature selection, we do the data preprocessing while encoding and normalization to specific range [0, 1] using min–max normalization. The preproceeded dataset includes 108,549 records of confirmed COVID-19 cases and 940,026 records of non-confirmed cases (i.e., it contains 922,017 normal patient and 18009 non-COVID-19 pneumonia patient).

5.2 Learning algorithms application for data classification

In this phase, we implement an algorithm written in *Python* language that will be running in the Raspberry Pi simulator in order to convert the dataset into event streams transmitted to the broker node (see Listing 1).

[1] https://github.com/nshomron/covidpred.

Listing 1: Pseudo code for data classification written in Python language

```
#Non-Medical Device(Raspberry Pi simulator)
...
client = mqtt.Client()
rc=client.connect("@IP_Brocker", 1883)
EventStreams=stream.iter_csv('testingData.csv',
target='Class',
converters={'Class':int, 'temperature':float,
        'O2_saturation':float, 'diabetes':int, .....})
for x, y in EventStreams:
es={"x":x,"y":y}
client.publish("topic/TestingData",
json.dumps(es));
#End-user or Medical Device(Raspberry Pi
simulator)
...
rc=client.connect("@IP_Brocker", 1883)
EventStreams=stream.iter_csv('trainingData.csv',
...)
for x, y in EventStreams:
es={"x":x,"y":y}
client.publish("topic/TrainingData",
json.dumps(es) );
#Stream-ML (Fog Node)
def on_connect(client, userdata, flags, rc):
client.subscribe("topic/TestingData")
def on_message(client, userdata, msg):
global metric
eventStream=json.loads(msg.payload)
x= eventStream["x"]
y= eventStream["y"]
....
model=pickle.load(open("model.csv", 'rb'))
# make a prediction
y_pred=model.predict_one(x)
# update the metric
metric=metric.update(y,y_pred)
...
client = mqtt.Client()
rc=client.connect("@IP_Brocker", 1883)
client.on_connect=on_connect
client.on_message=on_message
try:
client.loop_forever()
except KeyboardInterrupt:
...
#Stream-ML (Cloud Node)
def on_connect(client, userdata, flags, rc):
client.subscribe("topic/TrainingData")
def on_message (client, userdata, msg):
eventStream=json.loads(msg.payload)
...
model=pickle.load(open("model.csv", 'rb'))
# update the model
model=model.learn_one(x, y) )
pickle.dump(model, open("model.csv", 'wb'))
...
client = mqtt.Client()
rc=client.connect("@IP_Brocker", 1883)
...
```

Then, the fog broker receives the collected event streams and transfers them to the appropriate node (i.e., fog or cloud node) through MQTT protocol. For doing so,

we install Docker on ubuntu 18.04 with Intel Core i5 3210M@2.50 GHz processor and 6.00 GB RAM and then instantiate a container for running Mosquitto as MQTT broker.

As illustrated in Listing 1, the experiments are implemented using ML algorithms of python's libraries, whereas the first stream ML, deployed in a fog node, is running on other Docker container, while the second one and the batch ML, deployed on OpenStack private cloud, is running on ubuntu 18.04 Docker machine with 6.00GB RAM. At the first time, we evaluate divers classifiers of Python's *Scikit-learn* library in order to choose the best performing for our batch ML aiming at a long-term decision making. The existing ML algorithms might be classified into three major classes: (i) Conventional machine learners including *Adaptive Boosting (AdaBoost) Classifier* [34], *Linear Support Vector Classification* (LinearSVC) [35], *Decision Tree Classifier*[36], *Random Forest Classifier*[37], *Gaussian Naive Bayes*[38] and *Gradient Boosting Classifier* [39]], (ii) Simple neural network learners (i.e., Single layer), which is a *Multi-Layer Perceptrons (MLP) Classifier*[40] with a single layer of training, and (iii) Deep learning with multiple layers, which is an MLP-based algorithm with multiple layers. In fact, we should select the algorithm displaying an analytic dashboard for the medical staff in order to help them in determining the critical symptom to detect a COVID-19 infected case. Moreover, the algorithm should allow our system to exclude the irrelevant indicators that may decrease the model accuracy as well as to find out new symptoms, which may be relevant and useful for our model retraining. Obviously for each new streaming event, the stream ML model is updated and makes a real-time prediction. Such new data is then transmitted to the cloud environment for the batch processing. At the second time, we evaluate the Python's *river* library including several stream ML classifiers, such as *Logistic Regression*, *Adaptive Random Forest Classifier*, *Hoeffding Adaptive Tree Classifier*, *Extremely Fast Decision Tree Classifier*, *Gaussian Naive Bayes*, and *MLP Classifier*. It is worth mentioning that all these algorithms might be used to build a stream ML model for a real-time COVID-19 prediction.

5.3 Federated MLs performance analysis

5.3.1 Performance metrics

To evaluate the performance of each federated ML model, four performance metrics were carried out: accuracy, root-mean-square error, precision and F1 score. These metrics can be further detailed through the confusion matrix concept [41]. In fact, the confusion matrix is a table that illustrates the relations between the real and predicted values within a classification problem (see Table 1). There are: (i) **True Positive (TP)** quarter representing the number of instances correctly identified, while the prediction indicates COVID-19 disease and it is true, (ii) **True Negative (TN)** quarter indicating the number of instances correctly identified, while the prediction indicates non-COVID-19 disease and it is true, (iii) **False Positive (FP)** quarter, which is the number of incorrect instances, while the COVID-19 prognosis is negative and it is false, and (iv) **False Negative**

Table 1 Confusion matrix-based performance metrics

	Non-COVID	COVID	
Non-COVID	True Negative (TN)	False Positive (FP)	Specificity TN/(TN + FP)
COVID	False Negative (FN)	True Positive (TP)	Recall TP/(TP + FN)
	Negative Predictive Value TN/(TN + FN)	Precision TP/(TP + FP)	Accuracy (TP + TN)/(TP + TN + FP + FN)

(FN) quarter illustrating the number of the opposite error where the prediction instances are incorrect (i.e., fail to indicate the presence of a COVID-19 disease when it is present). Obviously, the four performance metrics are computed as follows:

(a) **The accuracy** is computed as the number of correctly classified instances (i.e., true and false positive) to the total number of instances (see Equation (1)).

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

(b) **The RMSE**, as shown in Equation (2), it is computed as the square root of the average of squared differences between the predicted and actual values.

$$RMSE = \sqrt{\left(\frac{1}{n}\right)\sum_{i=1}^{n}(Y_i - Y_i')^2} \tag{2}$$

Where Y_i is the real value and Y_i' is the predicted value.

(c) **The precision** is the fraction of relevant instances to the retrieved instances (see Equation (3)).

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

(d) **The F1 score**, as illustrated by Equation (4), is computed by multiplying 2 by the precision and recall measures and then dividing the result by the sum of the precision and recall measures.

$$F1_{score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

where

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

For stream ML model evaluation, we consider the training time metric in addition to the four above-mentioned ones. Obviously, the response time is a key metric for the real-time predictions.

5.3.2 Batch ML model evaluation

The batch ML model evaluation aims at selecting the best performing algorithm for our multi-class classification problem (i.e., classify the collected symptoms data as normal, COVID-19, or non-COVID-19 pneumonia patient). This second evaluation is two-goal. The first one is to compare the different ML classifiers based on two different data splitting strategies:

(a) The *no-shuffle train/test split* strategy consisting of splitting without shuffling the dataset into two parts as 80% of the dataset for training and 20% for testing the performance of the batch ML algorithm deployed into the cloud.
(b) The *KCV* strategy splits the dataset into k number of subsets; some of them are randomly selected to be used to learn the model, while the rest are used to assess the model performance [6, 42]. Thanks to such strategy, one can test the entire dataset. In this work, we adopt threefold cross-validation to split our dataset to train and test the used models.

The obtained results are given in Table 2. We analyze these experiment results in order to select the best strategy, algorithms, and parameters. In fact, we remark that:

(i) The most accurate result for the majority of classifiers belongs to no-shuffle train/test split strategy.
(ii) Decision Tree, Random Forest, Gradient Boosting and the Simple Neural Networks with 5, 10, 50 or 100 neurons might achieve the higher accuracy while adopting no-shuffle train/test split. Thus, they outperform other algorithms by providing prediction obtained up with 87.54% precision, 90.14% accuracy, 14.52% RMSE, and 87.9% F1 Score (see the bold values in Table 2).
(iii) Decision Tree Classifier achieves the higher training time with 0.53 seconds by using no-shuffle train/test split, and 2.06 seconds while adopting KVC strategy with $k = 3$ (see the italic values in Table 2).
(iv) MLP classifier outperforms other algorithms by always providing the best values in terms of accuracy, precision, RMSE and F1 score regardless of the adopted strategy. Therefore, we suggest to use the MLP in order to ensure better performance for our proposed batch ML model (see the underlined values in Table 2).
(v) On average, the MLP is the best classifier with outstanding performance. The number of neurons on the singleton hidden layer seems to have some light impacts on accuracy. For instance, an MLP model with only one hidden layer and three neurons achieved 90% accuracy, whereas whenever we increase the number of neurons to 5 and 10, the accuracy grow to be 90.04% and 90.06%, respectively. Consequently, the augmentation of the neurons number per layer

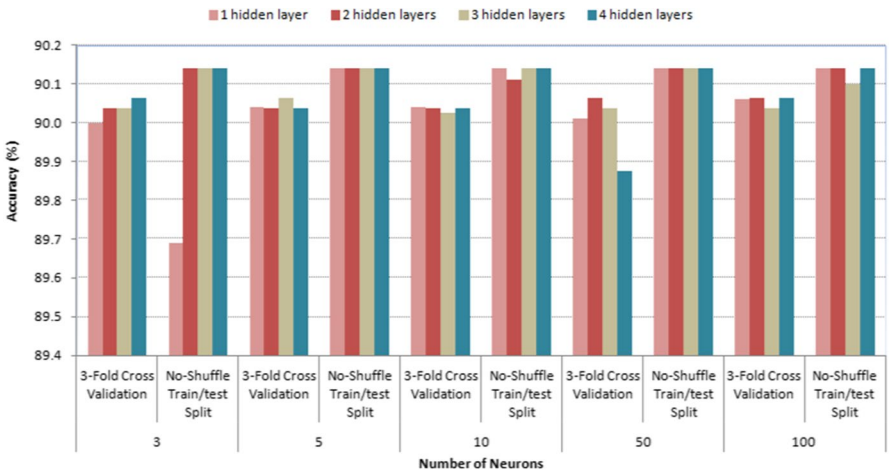


Fig. 8 Result of layer variation (1–4)

causes an additional computational cost and a slight improvement in accuracy simultaneously. Accordingly, the question is whether a more complex model deserves to be considered or a simpler model with slightly lower accuracy would be sufficient for prediction.

Figure 8 depicts the experiments in details, including parameters values selected for each model (i.e., number of layers used in each ML model and the number of hidden layer neurons in artificial neural networks (ANN)). It is worth mentioning that we used the MLP with adam activation function like ANN type. Indeed, Fig. 8 shows that (i) by using no-shuffle train/test split, a simple deep learning model (e.g., MLP(5)) might give similar values such a more deeper and larger classifier (e.g., MLP(100,100,100,100)), and (ii) while using threefold CV, a simple deep learning (e.g., MLP(3) with accuracy of 90%) could perform a closer similar value from a larger classifier (e.g., MLP(3,3,3,3) with accuracy of 90.06%). Hence, the choice of the both hidden layers and neurons numbers depends on the appropriate level of accuracy.

Furthermore, we are interested in evaluating the features' importance to extract those that significantly help determine the predictive capability of the COVID-19 symptoms within the large dataset. In this context, we evaluate *AdaBoost*, *Decision Tree*, *Random Forest*, and *Gradient Boosting* Classifiers. Figure 9 shows that the head ache, fever, cough, and sore throat are the key features that may mainly contribute in COVID-19 predictions and consequently improve our federated model.

5.3.3 Stream ML model evaluation

The second goal of the evaluation is to find the best algorithms for our stream ML models in terms of (i) performance parameters such as accuracy, RMSE, precision,

Table 2 Batch ML model evaluation for multi-class classification

ML Algorithm	Performance metrics									
	Threefold cross-validation					No-shuffle train/test split				
	Accuracy (%)	RMSE (%)	Precision (%)	F1 Score (%)	Training time (s)	Accuracy (%)	RMSE (%)	Precision (%)	F1 Score (%)	Training time (s)
Conventional machine learning										
AdaBoost (n = 50)	89.35	15.6	86.73	86.11	74.26	89.67	15.13	87.22	86.66	89.67
AdaBoost (n = 100)	89.35	15.6	86.73	86.11	254.6	89.67	15.13	87.22	86.66	69.37
LinearSVC (SVM)	89.67	15.25	87.12	86.69	3110.03	89.69	15.1	87.25	86.7	685.96
DecisionTree	90.04	14.76	87.32	87.73	*2.06*	**90.14**	**14.52**	**87.54**	**87.9**	*0.53*
Random Forest (n = 50)	90.04	14.76	87.32	87.73	53.93	**90.14**	**14.52**	**87.54**	**87.9**	18.93
Random Forest (n = 100)	90.04	14.76	87.32	87.73	128.12	**90.14**	**14.52**	**87.54**	**87.9**	36.22
Gaussian Naive Bayes	89.81	14.91	86.98	87.84	3.61	89.86	14.71	87.11	87.94	0.77
Gradient Boosting	90.04	14.76	87.32	87.73	630.59	**90.14**	**14.52**	**87.54**	**87.9**	239.51
Average	89.79	15.05	87.11	87.21	532.15	89.93	14.77	87.37	87.45	142.62
Simple neural network (one layer)										
MLP (3)	90	14.84	87.32	87.48	149.34	89.69	15.1	87.24	86.7	74.2
MLP (5)	90.04	14.76	87.32	87.73	193.47	**90.14**	**14.52**	**87.54**	**87.9**	67.62
MLP (10)	90.06	14.73	87.34	87.78	325.32	**90.14**	**14.52**	**87.54**	**87.9**	126.75
MLP (50)	90.01	14.81	87.33	87.56	275.03	**90.14**	**14.52**	**87.54**	**87.9**	102.35
MLP (100)	90.06	14.73	87.34	87.78	454.42	**90.14**	**14.52**	**87.54**	**87.9**	163.77
Average	90.03	14.77	87.33	87.67	279.52	90.05	14.64	87.48	87.66	106.94

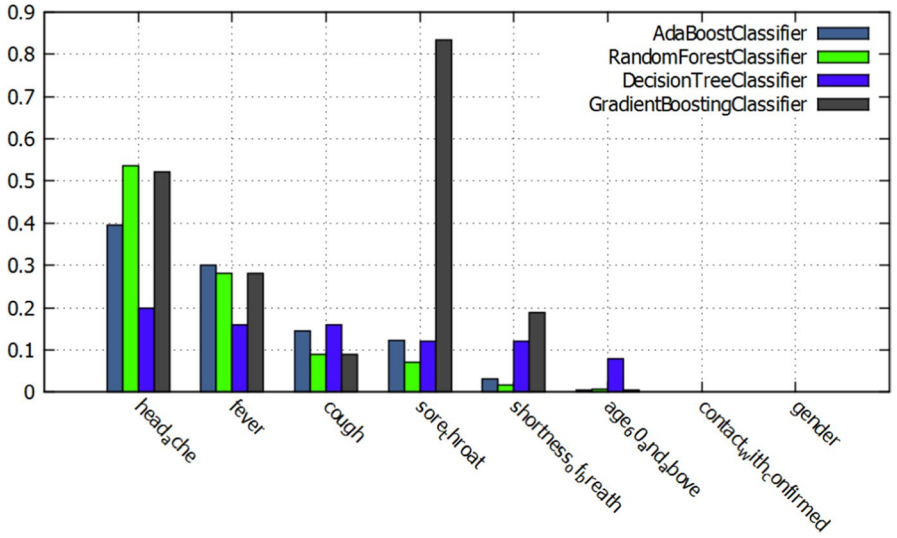


Fig. 9 Feature evaluation for COVID-19 disease classification

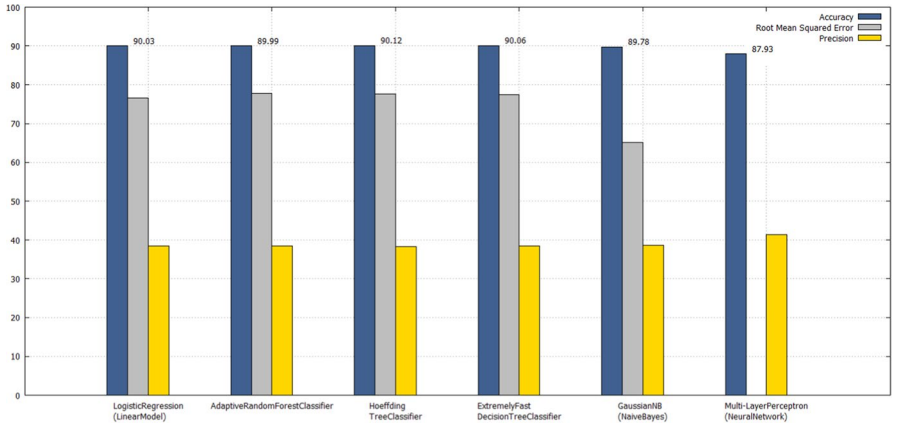
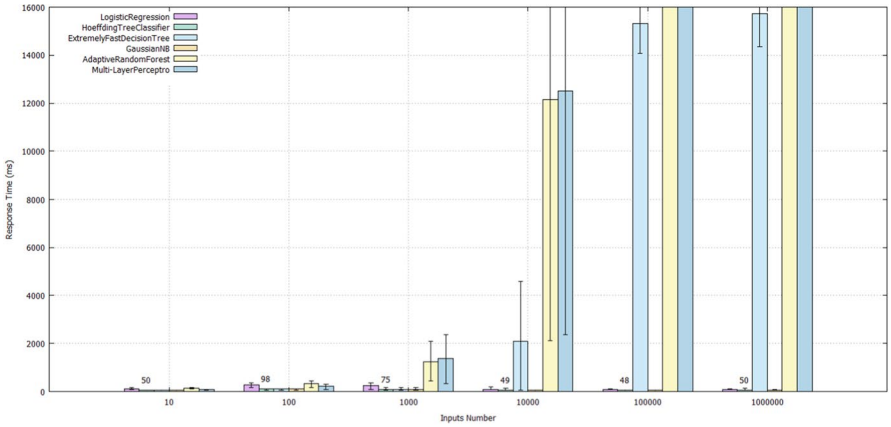


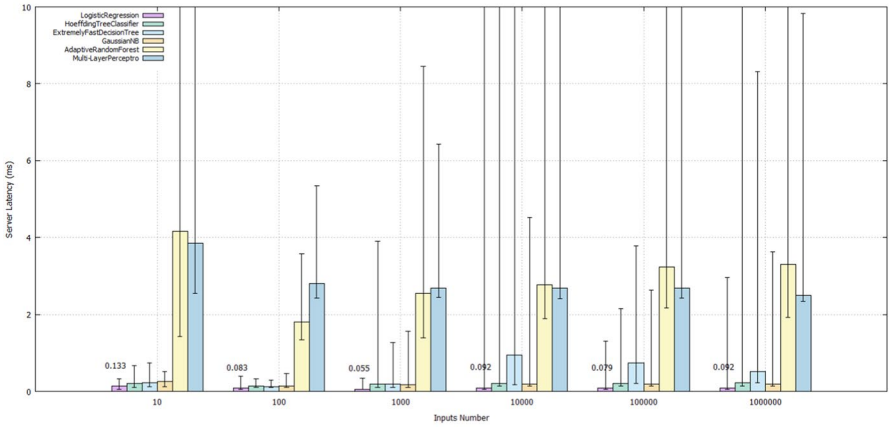
Fig. 10 Performance parameters evaluation results

and F1 score (shown in Fig. 10) and (ii) QoS parameters such as server latency, response time, and network latency (shown in Fig. 11).

(a) **Quantitative evaluation for multi-class classification**: Figure 10 depicts performance-based comparisons between the different stream ML algorithms in terms of accuracy, RMSE, precision and F1 score while considering a multi-class problem (i.e., the problem of classifying instances into one of three classes, normal, COVID-19, non-COVID-19 pneumonia). Indeed, *Hoeffding Adaptive Tree Classifier* outperforms other algorithms in providing prediction obtained up with



(a) Response time evaluation of COVID-19 disease classification

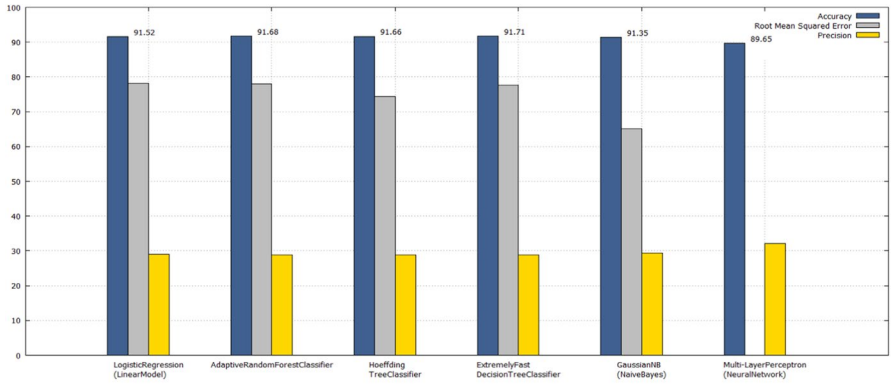


(b) Server latency evaluation of COVID-19 disease classification

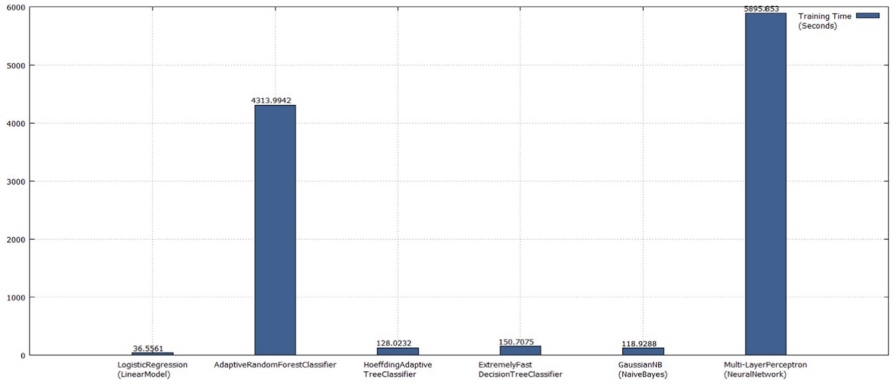
Fig. 11 QoS parameters evaluation results

77.6% precision, 90.12% accuracy and 38.39% RMSE. Moreover, as shown in Fig. 10, the MLP with a simple layer having a limited number of neurons (i.e., 3) is the most performing in terms of accuracy, RMSE, precision and F1 score with 90.14%, 14.52%, 87.54%, 87.9%, respectively. By either increasing the number of neurons or the number of layers, MLP exceeds the maximum time (i.e., 48 hours) without providing a prediction result.

(b) **Qualitative evaluation for multi-class classification**: Our proposed framework is also assessed using three QoS parameters including server latency, response time, and network latency. In fact, the minimum, maximum, and average values of these parameters are calculated after sending all data (i.e., 1048575 records of data) continuously as *JSON* objects. Figure 11a, b shows the QoS evaluation results between the different stream ML algorithms while receiving 10, 10^2, 10^3, 10^4, 10^5 and 10^6 records, respectively. The experiment results demonstrate



(a) Performance evaluation of COVID-19 disease classification



(b) Training time evaluation of COVID-19 disease classification

Fig. 12 Stream ML model evaluation for binary class classification

that both (*i*) *Hoeffding Adaptive Tree Classifier* and *Logistic Regression* provide the best performance average, whereas the *Hoeffding Adaptive Tree Classifier* achieves the best response time with 48 milliseconds, *Logistic Regression* spends the minimum server latency with 0.055 milliseconds, (*ii*) Whatever the number of received records, *Hoeffding Adaptive* and *Logistic Regression* remain the best algorithms that always offer the best results, (*iii*) *Adaptive Random Forest* and *MLP Classifier* are the worst QoS parameters in the evaluation (i.e., they are very slow while continuously processing streaming data), and (*vi*) the average of the network latency per transferred record across the network is 58 ms.

It should be noted that all the above-mentioned performance results correspond to a multi-class problem while both considering the stream and batch ML models evaluation. In the following, we will present a binary class (i.e., COVID vs. non-COVID classes) performance evaluation. The suitable stream ML model is chosen according to the confusion matrices shown in Fig. 13 as well as on the

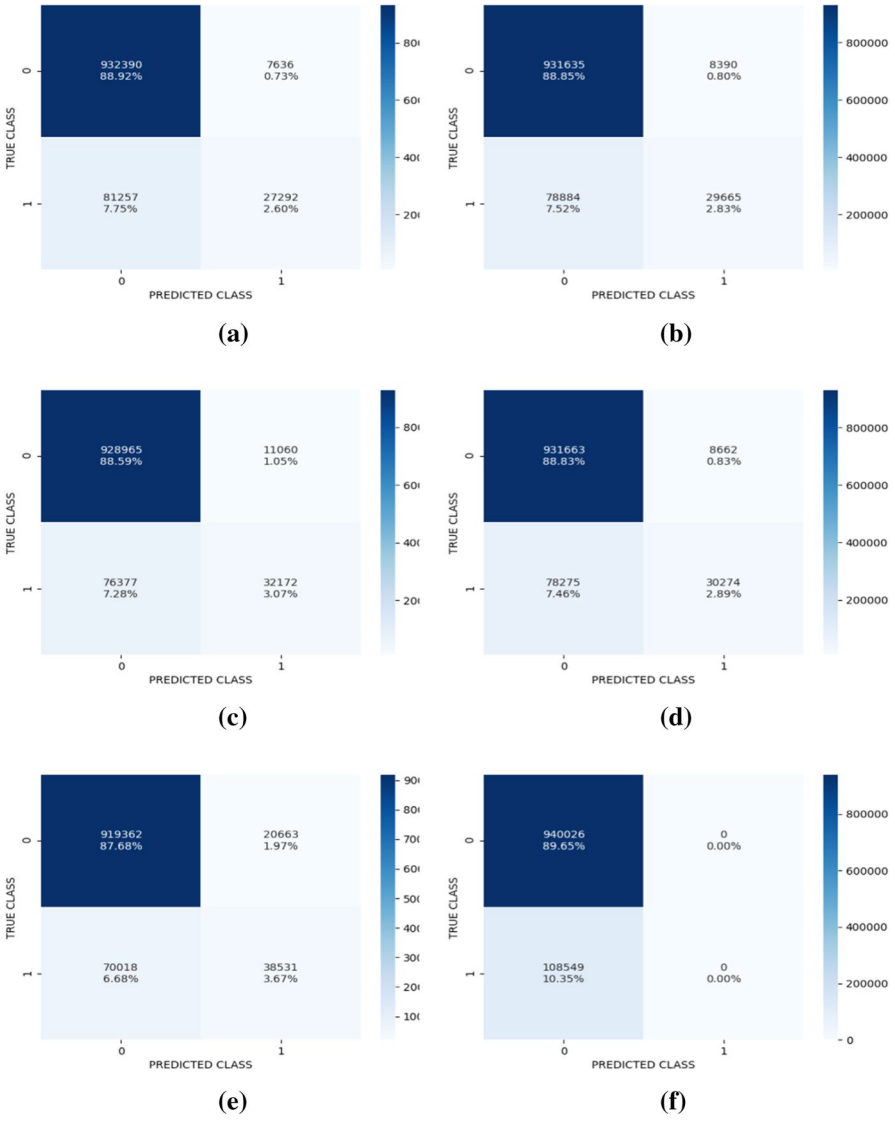


Fig. 13 Confusion matrix-based performance metrics for a binary classification by using **a** Logistic Regression, **b** Adaptive Random Forest, **c** Hoeffding Adaptive Tree, **d** Extremely Fast Decision Tree, **e** Gaussian NB, **f** MLP

accuracy, precision, RMSE, and training time metrics shown in Fig. 12a, b. The evaluation results demonstrate that whether considering binary classification: (i) Extremely Fast Decision Tree Classifier algorithm outperforms other algorithms by providing predictions obtained up with 77.75% precision, 91.71% accuracy and 28.79% RMSE, and (ii) Logistic Regression is the best-performing algorithm

in term of the training time, which is a key metric for real-time predictions. Moreover, it should notice that by adopting some ML algorithms, the binary classification is more performing than the multi-classification to predict the COVID-19 prognosis. In contrast, as shown in Fig. 12a, b, we obtain worse values by using a binary-classification when adopting certain other algorithms.

According to these results, we conclude that the stream ML algorithms have the potential to be integrated into the COVID-19 prognosis best practices since they allow the early predictions of the suspected COVID-19 cases.

6 Conclusion

The proposed work presented a COVID-19 monitoring and prognosis system. The system followed a hybrid framework integrating the IoT, fog, and cloud technologies. Our framework enabled the data collection from medical and non-medical devices, preprocessing, classification models, and training using a set of federated MLs (i.e., batch and stream MLs) provided as services. The batch MLaaS was implemented on the cloud environment for a long-term decision making, and the stream MLaaS was installed into a hybrid fog/cloud environment for a short-term decision making. The proposed federated batch and stream models were determined after a series of quantitative and qualitative evaluation based on Python's libraries algorithms and the widely used IEEE COVID-19 dataset. Both the performance and QoS metrics of the federated MLs evaluation demonstrated that the proposed framework promise efficient results for the detection of COVID-19 disease. In the future work, we plan to involve other deep learning algorithms to improve the prediction of COVID-19 cases as well as the determination of the key features impacting the predictions. In addition, we intend to make our federated models more robust and accurate by testing and training more real data (i.e., consider COVID-19 cases as much as possible). Finally, we aim at providing the proposed federated MLaaS services on the large scale demand.

Acknowledgements We thank the editor and the anonymous referees who have provided valuable comments on an earlier version of this paper. We would also like to show our gratitude to Dr. Nizar Elleuch (Professor of vascular surgery, Habib Bourguiba University Hospital, Sfax, Tunisia) and Dr. Mariem Ben Hmida (Specialist in preventive medicine and epidemiology, Hedi Chaker University Hospital, Sfax, Tunisia) for their helps by broadcasting the conducted survey throw engaged medical staff in order to find out the experts' requirements and recommendations to combat the COVID-19 pandemic.

References

1. Stojmenovic I (2014) Fog computing: A cloud to the ground support for smart things and machine-to-machine networks. In: 2014 Australasian telecommunication networks and applications conference (ATNAC). IEEE, pp 117–122
2. Tuli S, Mahmud R, Tuli S, Buyya R (2019) Fogbus: a blockchain-based lightweight framework for edge and fog computing. J Syst Softw 154:22–36

3. Gia TN, Jiang M, Sarker VK, Rahmani AM, Westerlund T, Liljeberg P, Tenhunen H (2017) Low-cost fog-assisted health-care IoT system with energy-efficient sensor nodes. In: 2017 13th international wireless communications and mobile computing conference (IWCMC). IEEE, pp 1765–1770
4. Debauche O, Mahmoudi S, Manneback P, Assila A (2019) Fog IoT for health: a new architecture for patients and elderly monitoring. Procedia Comput Sci 160:289–297
5. Davenport T, Kalakota R (2019) The potential for artificial intelligence in healthcare. Future Healthc J 6(2):94
6. Nematzadeh Z, Ibrahim R, Selamat A (2015) Comparative studies on breast cancer classifications with k-fold cross validations using machine learning techniques. In: 2015 10th Asian Control Conference (ASCC). IEEE, pp 1–6
7. Tuli S, Basumatary N, Gill SS, Kahani M, Arya RC, Wander GS, Buyya R (2020) Healthfog: an ensemble deep learning based smart healthcare system for automatic diagnosis of heart diseases in integrated IoT and fog computing environments. Futur Gener Comput Syst 104:187–200
8. Munir M, Siddiqui SA, Chattha MA, Dengel A, Ahmed S (2019) Fusead: unsupervised anomaly detection in streaming sensors data by fusing statistical and deep learning models. Sensors 19(11):2451
9. Hageman JR (2020) The coronavirus disease 2019 (covid-19). Pediatr Ann 49(3):e99–e100
10. Loey M, Manogaran G, Taha MHN, Khalifa NEM (2020) A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the covid-19 pandemic. Measurement 167:108288
11. Qin B, Li D (2020) Identifying facemask-wearing condition using image super-resolution with classification network to prevent covid-19. Res Sensors 20(18):5236
12. Chowdary GJ, Punn NS, Sonbhadra SK, Agarwal S (2020) Face mask detection using transfer learning of inceptionv3. arXiv preprint arXiv:2009.08369
13. El-Rashidy N, El-Sappagh S, Islam S, El-Bakry HM, Abdelrazek S (2020) End-to-end deep learning framework for coronavirus (covid-19) detection and monitoring. Electronics 9(9):1439
14. Ozturk T, Talo M, Yildirim EA, Baloglu UB, Yildirim O, Acharya UR (2020) Automated detection of covid-19 cases using deep neural networks with x-ray images. Comput Biol Med 121:103792
15. Barstugan M, Ozkaya U, Ozturk S (2020) Coronavirus (covid-19) classification using CT images by machine learning methods. arXiv preprint arXiv:2003.09424
16. Pathak Y, Shukla PK, Tiwari A, Stalin S, Singh S, Shukla PK (2020) Deep transfer learning based classification model for covid-19 disease. IRBM
17. Tsiknakis N, Trivizakis E, Vassalou EE, Papadakis GZ, Spandidos DA, Tsatsakis A, Sánchez-García J, López-González R, Papanikolaou N, Karantanas AH et al (2020) Interpretable artificial intelligence framework for covid-19 screening on chest X-rays. Exp Ther Med 20(2):727–735
18. Otoom M, Otoum N, Alzubaidi MA, Etoom Y, Banihani R (2020) An IoT-based framework for early identification and monitoring of covid-19 cases. Biomed Signal Process Control 62:102149
19. Hossain MS, Muhammad G, Guizani N (2020) Explainable AI and mass surveillance system-based healthcare framework to combat covid-i9 like pandemics. IEEE Netw 34(4):126–132
20. Kumar A, Sharma K, Singh H, Naugriya SG, Gill SS, Buyya R (2020) A drone-based networked system and methods for combating coronavirus disease (covid-19) pandemic. Future Gener Comput Syst 115:1–19
21. Read J, Bifet A, Pfahringer B, Holmes G (2012) Batch-incremental versus instance-incremental learning in dynamic and evolving data. In: International symposium on intelligent data analysis. Springer, pp 313–323
22. Kallel A, Rekik M, Khemakhem M (2020) IoT-fog-cloud based architecture for smart systems: prototypes of autism and covid-19 monitoring systems. Softw Pract Exp 51(1):91–116
23. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V et al (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12:2825–2830
24. Montiel J, Halford M, Mastelini SM, Bolmier G, Sourty R, Vaysse R, Zouitine A, Gomes HM, Read J, Abdessalem T et al (2020) River: machine learning for streaming data in python. arXiv preprint arXiv:2012.04740
25. Singh A, Kaur A, Dhillon A, Ahuja S, Vohra H (2021) Software system to predict the infection in covid-19 patients using deep learning and web of things. Softw Pract Exp https://doi.org/10.1002/spe.3011

26. Jagirdar NM (2018) Online machine learning algorithms review and comparison in healthcare, Ph.D. dissertation, University of Tennessee, The address of the publisher, 12, an optional note
27. Mitchell TM et al (1997) Machine learning. McGraw-Hill, New York, Tech. Rep.
28. Hayes TL, Kanan C (2020) Lifelong machine learning with deep streaming linear discriminant analysis. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, pp 220–221
29. Tartaglione E, Barbano CA, Berzovini C, Calandri M, Grangetto M (2020) Unveiling covid-19 from chest X-ray with deep learning: a hurdles race with small data. arXiv preprint arXiv:2004.05405
30. Yasser I, Twakol A, El-Khalek A, Samrah A, Salama A et al (2020) Covid-x: novel health-fog framework based on neutrosophic classifier for confrontation covid-19. Neutrosophic Sets Syst 35(1):1
31. Huang C, Wang Y, Li X, Ren L, Zhao J, Hu Y, Zhang L, Fan G, Xu J, Gu X et al (2020) Clinical features of patients infected with 2019 novel coronavirus in Wuhan, China. Lancet 395(10223):497–506
32. Wu C, Chen X, Cai Y, Zhou X, Xu S, Huang H, Zhang L, Zhou X, Du C, Zhang Y et al (2020) Risk factors associated with acute respiratory distress syndrome and death in patients with coronavirus disease 2019 pneumonia in Wuhan, China. JAMA Intern Med
33. Zwattendorfer B, Stranacher K, Tauber A (2013) Towards a federated identity as a service model. In: International conference on electronic government and the information systems perspective. Springer, pp 43–57
34. Rojas R et al (2009) Adaboost and the super bowl of classifiers a tutorial introduction to adaptive boosting, Freie University, Berlin, Tech. Rep
35. Tang Y (2013) Deep learning using linear support vector machines. arXiv preprint arXiv:1306.0239
36. Safavian SR, Landgrebe D (1991) A survey of decision tree classifier methodology. IEEE Trans Syst Man Cybern 21(3):660–674
37. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
38. Murphy KP et al (2006) Naive bayes classifiers. Univ B C 18:60
39. Friedman JH (2001) Greedy function approximation: a gradient boosting machine. Ann Stat 29:1189–1232
40. Egmont-Petersen M, Talmon JL, Hasman A, Ambergen AW (1998) Assessing the importance of features for multi-layer perceptrons. Neural Netw 11(4):623–635
41. Yildirim M, Cinar A (2020) A deep learning based hybrid approach for covid-19 disease detections. Traitement du Signal 37(3):461–468
42. Anguita D, Ghelardoni L, Ghio A, Oneto L, Ridella S (2012) The 'k' in k-fold cross validation. In: ESANN

Authors and Affiliations

Ameni Kallel[3,4] · Molka Rekik[1,3] · Mahdi Khemakhem[2,3]

Molka Rekik
m.rekik@psau.edu.sa

Mahdi Khemakhem
mahdi.khemakhem@enetcom.usf.tn

[1] Department of Information Systems, College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, AlKharj 11942, Saudi Arabia

[2] Department of Computer Science, College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, AlKharj 11942, Saudi Arabia

[3] Data Engineering and Semantics Research Unit, Faculty of Sciences of Sfax, University of Sfax, Sfax, Tunisia

[4] Département Technologies de l'Informatique, Higher Institute of Technological Studies (ISET), Sidi Bouzid, Tunisia