Hybrid-based Framework for COVID-19 Prediction via Federated Machine Learning Models

Ameni Kallel, *Laboratoire OLID,LR19ES21, École Nationale d'Électronique et des Telécommunications, University of Sfax, 3021, Tunisia*
Molka Rekik, *Department of Information System, College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, Alkharj, 11942, Saudi Arabia*
Mahdi Khemakhem, *Department of Information System College of Computer Engineering and Sciences Prince Sattam Bin Abdulaziz University Alkharj, 11942, Saudi Arabia*

Abstract—The COronaVIrus Disease 2019 (COVID-19) pandemic is unfortunately highly transmissible across the people. Therefore, a smart monitoring system that detects and tracks the suspected COVID-19 infected persons may improve the clinicians decision-making and consequently limit the pandemic spread. This paper entails a new framework integrating the Machine Learning (ML), cloud, fog, and Internet of Things (IoT) technologies to propose a 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, which is implemented on the cloud environment for a long-term decision-making, and *(ii)* the distributed stream-MLaaS installed into a hybrid fog/cloud environment for a short-term decision-making. Stream-MLaaS use 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.

Index Terms—COVID-19 pandemic, machine learning, IoT devices, batch/streaming data, hybrid fog-cloud federation, federated MLaaS, real-time prediction, decision-making

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 peopole 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

• *Corresponding author: M. Rekik was with the Department of Information System, College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, Alkharj, 11942, Saudi Arabia.*
E-mail: molka.rekik@gmail.com, m.rekik@psau.edu.sa

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 pre-processing and features selection in order to extract relevant indicators for the COVID-19 disease classification. It is worth mentioning that the data is 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, the federated-MLs evaluation is two-fold; the first focuses on selecting the best-performing algorithm/-model for the batch-ML from the widely used Python's library called *Scikit-learn* [23]. For do so, we follow three data splitting strategies *no-shuffle train/test split*, *K-fold Cross Validation* (KCV), and *shuffle Split Cross Validation (SCV) with a train/test split of 70/30*. The obtained results demonstrate that *(i)* the most accurate predictions are those of *shuffle split* strategy while using 10 random samples and *(ii) Random Forest Classifier* outperforms other algorithms by providing prediction obtained up with 71,4% precision, 75% accuracy, 25% Root Mean Square Error (RMSE), and 71,7% F1 Score. The second evaluation considers the stream-ML models while taking into account of Python's *River* library [24], which offers a set of streaming classification algorithms. The obtained results involve two classification kinds, the multi- and binary-class. It is worthy mentioned that with the multi-class kind, the prediction model can distinguish normal, COVID-19, and non COVID-19 pneumonia patients. While with the binary classification kind, the patients are categorized to COVID-19 Vs. non COVID-19 cases. In fact, the evaluation results demonstrate that, when considering multi-class, (i) *AdaptiveRandomForestClassifier* algorithm outperforms others by providing predictions obtained up with 76% precision, 75% accuracy, 49% RMSE, and 85% F1 Score and (ii) *LogisticRegression* algorithm is the best-performing algorithm in term of the execution time, which is a key metric for real-time predictions. While taking into account a binary classification, we obtain the best values of accuracy, RMSE, precision, F1 score, and execution (i.e., training) time, which are 89.26%, 32.76% , 92.48%, 92.48%, and 0.22 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 presented in Section 4. The implementation of our framework and its evaluation are discussed in Section 5. Finally, Section 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 3 sets of sequences [25], 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 $x_i = x_{i,1}, x_{i,2}, ...x_{i,n}$ and $y_i = y_{i,1}, y_{i,2}, ...y_{i,n}$. 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 [25].

1) Traditional batch learning: is characterized by an offline phase of training the classifier/algorithm once the complete training set of data is used (see Figure 1). Afterwards, such model is deployed online. In the second phase, the on-line 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. [26]. 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 Figure 2a). Consequently, the testing step occurs between a set of batchs [27]. However, it

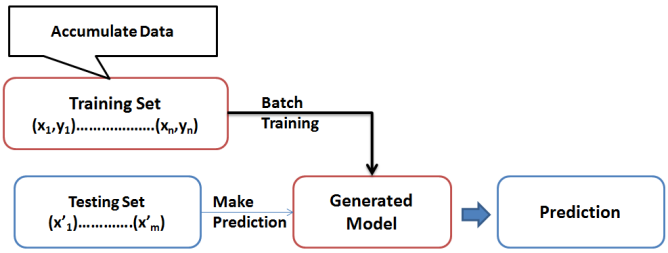


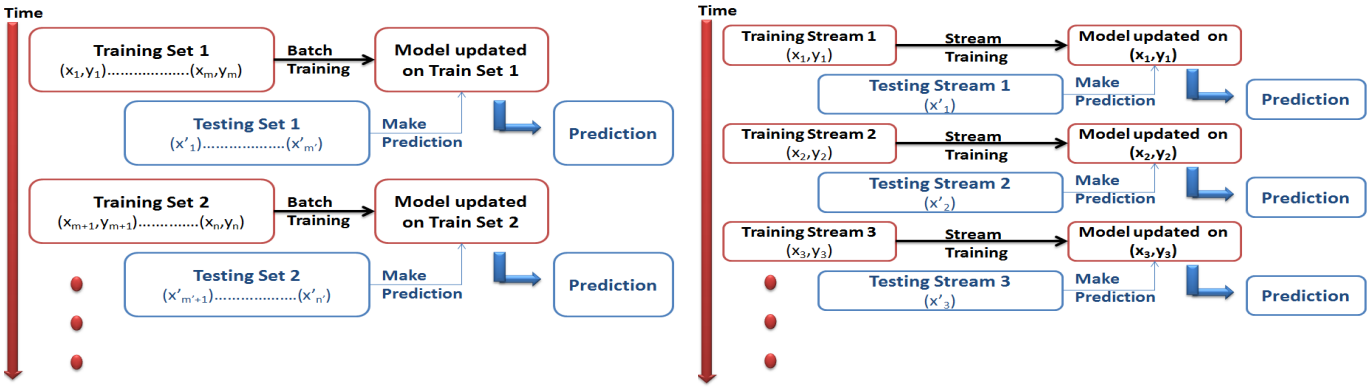
Fig. 1: Traditional batch learning technique

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 [27] as well as the model can not be learned from the arrival of the new data but until a new batch has been completed [21]. Moreover, trained models should be removed afterwards 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 Figure 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 optimizing the resources utilisation.

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 were 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], [28], [29]. 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 10-fold 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, hypertension, etc. [30], [31]. Moreover, the high-risk patients should be frequently monitored even if they are outside hospitals by checking their vital signs such as temperature, oxygen saturation, etc. [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 temperature, heart rate, cough quality, etc. 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 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



(a) Incremental batch learning technique

(b) Streaming learning technique

Fig. 2: Incremental learning technique

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 pre-processed 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 Figure 3, 94% of the medical staff are interested by such system. The second question is what about connecting an equipped

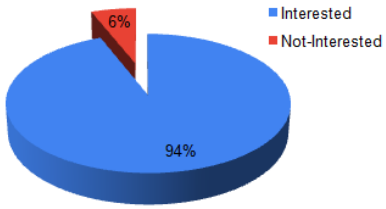


Fig. 3: Experts feedback regarding adopting a COVID-19 automatic monitoring and detecting system

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 in Figure 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, cough sensor, etc. In addition, physicians recommend

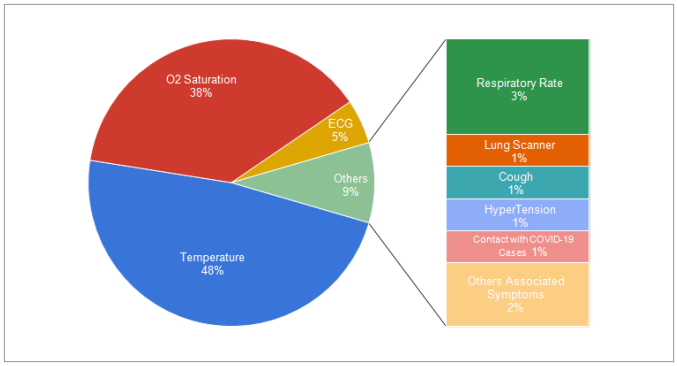


Fig. 4: Experts feedback regarding integrating sensors to detect COVID-19 symptoms

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 depicted in Figure 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.). Obviously, the survey results help us in the data pre-processing 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.

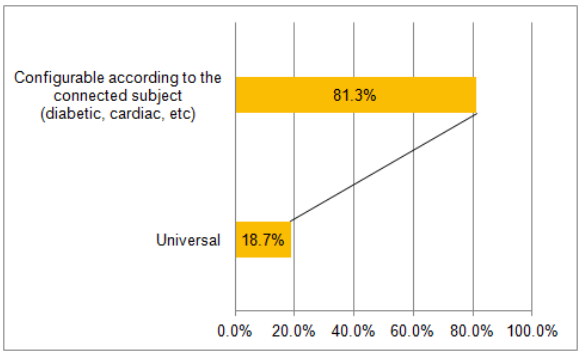


Fig. 5: Experts feedback regarding adopting a customized monitoring system based on the patient's health state

4 Customized Framework for COVID-19 Monitoring and Prognosis

As above-mentioned, 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 Figure 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.
2) Fog broker: The fog broker represents a gateway between the IoT devices, fog, and cloud computing environments (see Figure 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 whilst 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 Figure 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 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 Figure 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 symptoms, determining the most infected regions, etc.). In addition, the cloud stores all predictive models generated by these MLs.
5) End-user devices: 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 Figure 6).

4.2 Software components

The proposed framework includes a set of interrelated software components ensuring the *(i)* interaction between the IoT devices, the fog and cloud layers (see Figure 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 Application Programming Interface

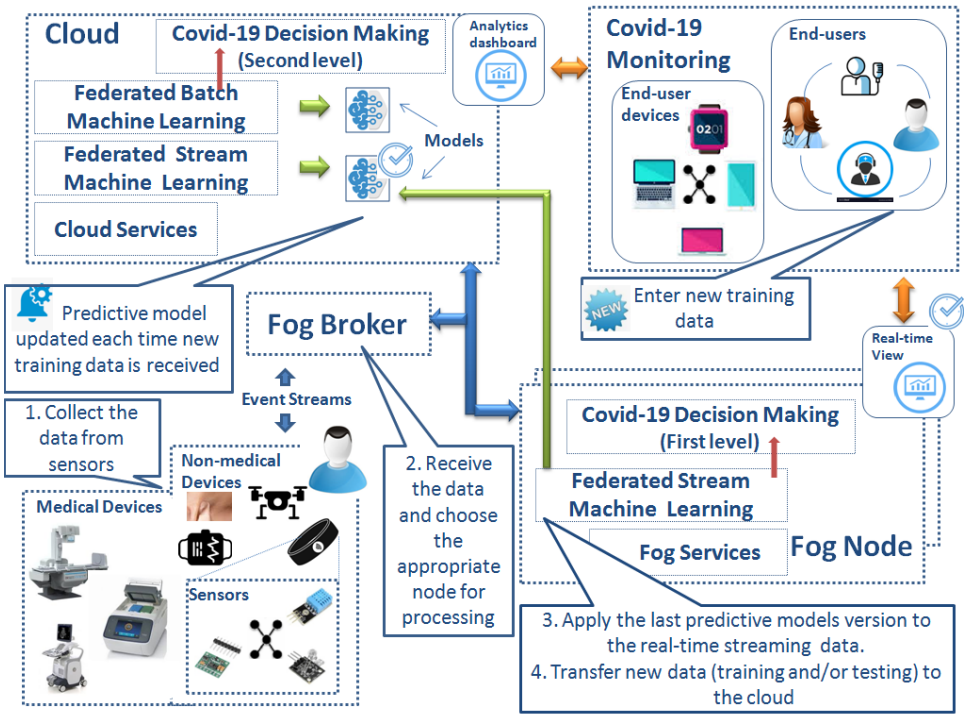


Fig. 6: Customized learning-based framework for COVID-19 monitoring and prognosis

(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 mainly 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]. Whilst, 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 mention 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) [32] 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

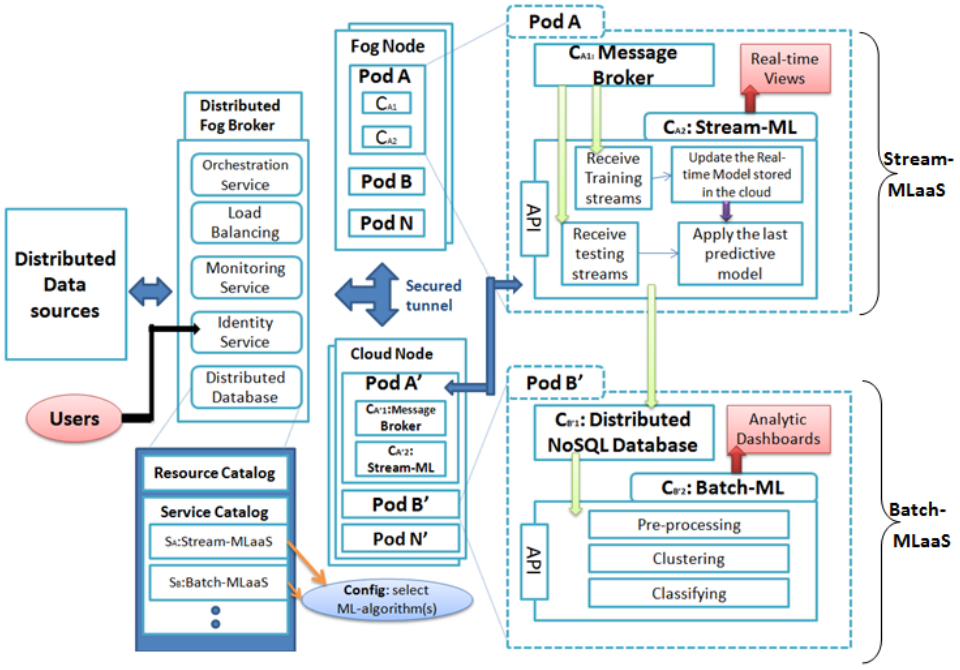


Fig. 7: Software components view of the proposed framework

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 IEEE COVID-19 dataset[1], which is open and widely used. It takes into consideration different types of pneumonias, such COVID-19, SARS, Streptococcus, Pneumocystis, etc. Indeed, the raw data can be gathered from smart medical and non-medical based IoT sensors. The dataset needs to be accumulated before it can be put to use. It has missing data values, that may be produced from faulty sensors or lack of communication among the components in the system. These missing values affect the system performance and they need to be addressed appropriately. In this work, missing values are replaced with their normal values (e.g., 37 for the temperature). Thus, we prepare the data of effective environment indicators relating to COVID-19 prediction (i.e., before using the ML models). Therefore, we apply a pre-processing module written in python extracts various COVID-19 symptoms from the "clinical-notes" column based on the survey results. In addition, our module promotes the normalizing, converting and structuring all the obtained data in order to better suited for ML. We select 10 features, such as temperature, O2-saturation, diabetes, heart-disease, smoking, hyper-tension, travel, obesity and cough. Additionally, we categorize 3 classes: (0) normal patient, (1) COVID-19 patient, and (2) non-COVID-19 pneumonia patient. After pre-processing, the dataset includes 307 records of confirmed COVID-19 cases and 139 records of non-confirmed cases (containing 17 normal patient and 122 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).

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@2.50GHz processor and 6.00GB RAM and then instantiate a container for running Mosquitto as MQTT broker.

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)
...
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
...
#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()
...
```

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 the eight classifiers of Python's *Scikit-learn* library, named *Adaptive Boosting (AdaBoost) Classifier* [33], *Linear Support Vector Classification* (LinearSVC) [34], *KNeighbors Classifier* [35], *Decision Tree Classifier* [36], *Random Forest Classifier* [37], *Multi-Layer Perceptrons (MLP) Classifier* [38], *Gaussian Naive Bayes* [39], and *Gradient Boosting Classifier* [40] in order to choose the best-performing as our batch-ML for a long-term decision making. 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 Apadtive Tree Classifier*, *Extremely Fast Decision Tree Classifier*, *Gaussian Naive Bayes*, and *KNearest Neighbors 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 illus-

1. https://github.com/ieee8023/covid-chestxray-dataset

trates 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's true, *(ii)* **True Negative (TN)** quarter indicating the number of instances correctly identified while the prediction indicates non COVID-19 disease and it's true, *(iii)* **False Positive (FP)** quarter, which is the number of incorrectly instances while the COVID-19 prognosis is negative and it's false, and *(iv)* **False Negative (FN)** quarter illustrating the number of the opposite error where the prediction instances are incorrectly (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.

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

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), it 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,

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

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)**

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 three data splitting strategies:

a) The *no-shuffle train/test split* strategy consisting of splitting without shuffling the dataset into two parts as 70% of the dataset for training and 30% 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 [42]. Thanks to such strategy, one can test the entire dataset [6]. In this work, we adopt 3-fold, 5-fold and 10-fold cross-validation to split our dataset to train and test the used models.

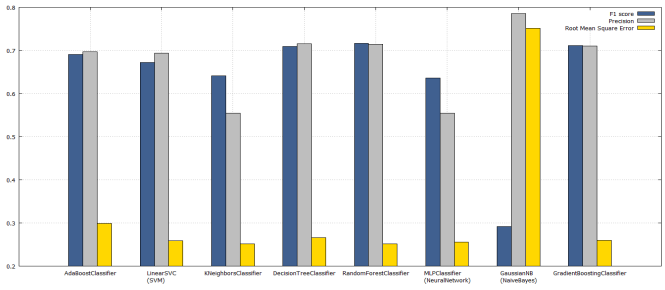
c) The *shuffle SCV* or *repeated learning-testing* [43], [44] strategy is similar to the KCV one; whereas the data is split into k-folds where one of the folds will be the test set and the rest $(k-1)$ folds will be the training set. In contrast in the shuffle split cross-validation, the data is not shuffled after each iteration [45]. Like the previous experiment, we adopt 3, 5, and 10 iterations of shuffle SCV with a train/test split rate by 70/30.

The obtained results are illustrated in Table 2. Based on the results analysis, we select the best strategy and parameters. In fact, we remark that: *(i) Random Forest Classifier* achieves the higher accuracy using shuffle SCV data splitting strategy after 10 iterations (see the circled value in Table 2) and *(ii)* the most accurate result for *LinearSVC*, *Decision Tree Classifier*, *Random Forest Classifier*, and *Gradient Boosting Classifier* belongs to shuffle SCV strategy using 10 random samples with 70% of data. Likewise, the shuffle SCV strategy has the most accurate result for *KNeighbors Classifier* and *MLP Classifier* while randomly splitting 3 folds. Regarding *AdaBoost Classifier* and *Guassian Naive Bayes*, the more accurate result is achieved by using KVC with $k = 10$ (see the bold values in Table 2). Moreover, we conclude that the most accurate result for the majority of classifiers belongs to shuffle SCV strategy using 10 random samples. Based on these results, we choose a *Shuffle SCV* strategy while using 10 random folds with 70% of data in order to assess the performance of the ML-algorithms in terms of precision, RMSE and F1 score. As shown in Figure 8a, *Random Forest Classifier* outperforms other algorithms by providing prediction obtained up with 71,4% precision, 75% accuracy, 25% RMSE, and 71,7% F1 Score. Therefore, we suggest to use *Random Forest Classifier* in order to ensure the better accuracy and RMSE for our proposed batch-ML model.

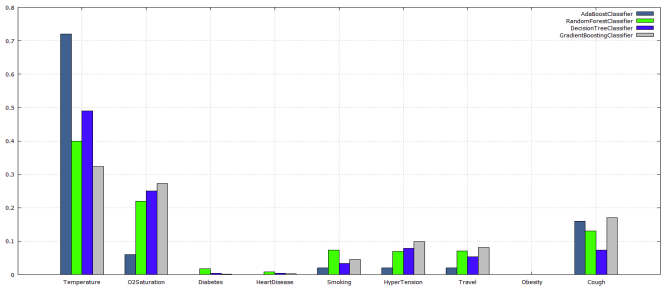
Furthermore, we are interested in assessing the features importance, which may help us determine the predictive capability of the COVID-19 symptoms within the dataset. In this context, we evaluate *AdaBoost*, *Decision Tree*, *Random Forest*, and *Gradient Boosting* Classifiers. Figure 8b shows that the temperature, O2-saturation and cough are the key features may mainly contribute in COVID-19 predictions and consequently improve our federated model.

5.3.3 Stream-ML model evaluation

The evaluation's second goal consists in finding out the best-performing algorithms for our stream-ML models. Figures 9a and 9b depict the performance-based comparisons

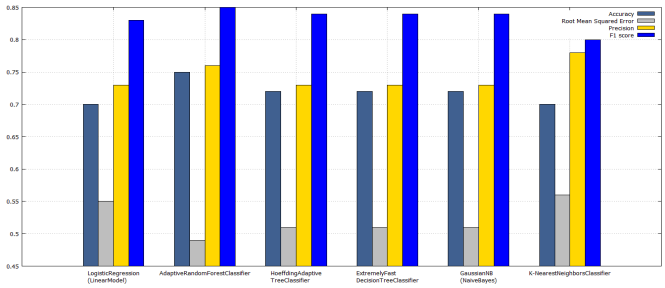


(a) Performance evaluation of COVID-19 disease classification

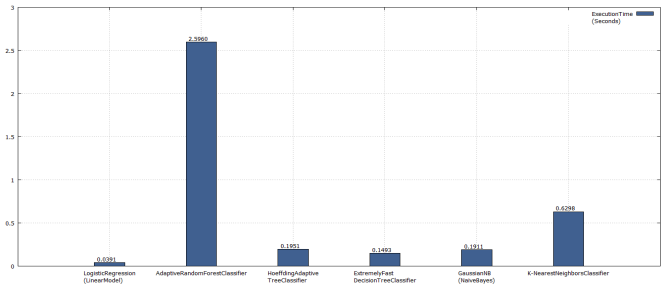


(b) Feature evaluation for COVID-19 disease classification

Fig. 8: Batch-ML model evaluation for multi-class classification



(a) Performance evaluation of COVID-19 disease classification



(b) Time training evaluation of COVID-19 disease classification

Fig. 9: Stream-ML model evaluation for multi-class classification

TABLE 2: Accuracy Average (AA) of the eight classifiers using *shuffle SCV*, *KCV*, and *no-shuffle train/test split* strategies

AA% with / ML-Algo	**Shuffle Split 10 splits**	**Shuffle Split 5 splits**	**Shuffle Split 3 splits**	**10-fold Cross Validation**	**5-fold Cross Validation**	**3-fold Cross Validation**	**No-shuffle train/test split**
AdaBoost Classifier	70.96	70.37	67.9	**71.47**	69.52	66.44	69.62
LinearSVC (SVM)	**74.51**	74.07	74.07	72.10	71.09	70.46	71.11
KNeighors Classifier	74.22	74.66	**74.81**	73.19	71.32	72.7	71.11
DecisionTree Classifier	**74.44**	73.77	73.33	71.2	69.52	70.69	71.1
RandomForest Classifier	**75.55**	74.22	74.32	73.25	70.19	70.69	70.37
MLPClassifier (NeuralNetwork)	74.44	74.37	**74.56**	72.96	71.53	71.58	71.11
GaussianNaiveBayes	33.58	32	31.6	**35.23**	30.68	30.64	34.07
GradientBoosting Classifier	**74.66**	73.33	72.09	70.77	70.42	70.91	69.62

TABLE 3: Stream-ML model evaluation for binary classification (COVID vs. No-COVID)

ML-Algorithm	Performance Metrics				
	Accuracy (%)	RMSE (%)	Precision (%)	F1 Score (%)	Execution Time (seconds)
LogisticRegression (LinearModel)	89.26	32.76	92.48	92.48	0.22
AdaptiveRandomForest	74.05	50.77	75.13	83.30	4.45
HoeffdingAdaptiveTree	68.01	56.42	68.79	80.96	0.45
ExtremelyFastDecisionTree	69.8	54.81	70.54	81.59	0.24
GaussianNB (NaiveBayes)	68.68	55.82	68.83	81.54	0.35
K-NearestNeighbors	68.23	56.22	76.43	77.29	1.1

between the different stream-ML algorithms in terms of accuracy, RMSE, precision, F1 score, and training time while considering a multi-class (i.e., the three classes, normal, COVID-19, non COVID-19 pneumonia) problem. Indeed,

Adaptive Random Forest Classifier and *Logistic Regression* can better perform, but each one of them do so with its own manner. In fact, *Adaptive Random Forest Classifier* outperforms other algorithms in providing prediction obtained up with 76% precision, 75% accuracy, 49% RMSE, and 85% F1 Score. Whilst *Logistic Regression* spends the minimum training time, which is a key metric for the real-time predictions. It should mention that all the above-mentioned performance results correspond to both stream and batch-ML models evaluation while considering a multi-class problem. In the following, we will present a binary-class (i.e., COVID Vs. Non-COVID classes) performance evaluation. The proposed stream-ML model is chosen based on the confusion matrices shown in Figure 10 as well as on the accuracy, precision, F1 score, RMSE, and execution time metrics illustrated in Table 3. Thus, for a binary classification solution for the COVID-19 prognosis cases, we have the best obtained values of accuracy, RMSE, precision, F1 score, and execution time are 89.26%, 32.76%, 92.48%, 92.48%, and 0.22 seconds, respectively. Here, it should be noted that we decide to adopt *Logistic Regression* for our stream-ML model. Moreover, it should notice that, by adopting certain ML algorithms, a binary-classification is more performing than a multi-classification regarding the COVID-19 prognosis problem. In contrast, as depicted in Table 3, 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 aims to propose a COVID-19 monitoring and prognosis system. The system follows a hybrid framework integrating the IoT, fog, and cloud technologies. Our framework enables the data collection from medical and non-medical devices, pre-processing, 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 evaluation based on Python's libraries algorithms and the widely used IEEE COVID-19 dataset. The performance metrics of the federated MLs evaluation demonstrated that the proposed framework promising 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.

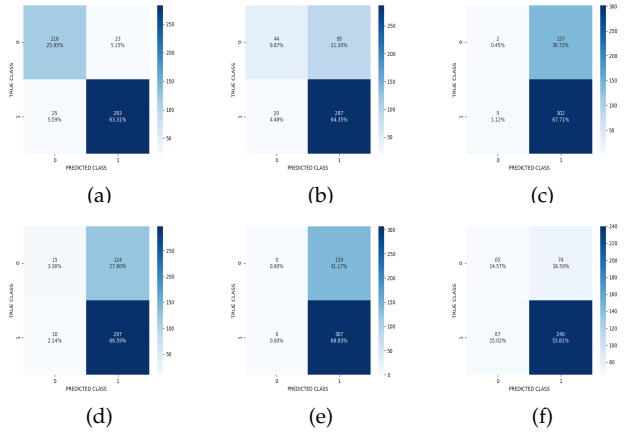


Fig. 10: Confusion matrix-based performance metrics for a binary classification by using (a) LogisticRegression (b) AdaptiveRandomForest (c) HoeffdingAdaptiveTree (d) ExtremelyFastDecisionTree (e) GaussianNB (f) K-NearestNeighbors

References

[1] I. Stojmenovic, "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, 2014, pp. 117–122.

[2] S. Tuli, R. Mahmud, S. Tuli, and R. Buyya, "Fogbus: A blockchain-based lightweight framework for edge and fog computing," *Journal of Systems and Software*, vol. 154, pp. 22–36, 2019.

[3] T. N. Gia, M. Jiang, V. K. Sarker, A. M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen, "Low-cost fog-assisted healthcare iot system with energy-efficient sensor nodes," in *2017 13th international wireless communications and mobile computing conference (IWCMC)*. IEEE, 2017, pp. 1765–1770.

[4] O. Debauche, S. Mahmoudi, P. Manneback, and A. Assila, "Fog iot for health: A new architecture for patients and elderly monitoring." *Procedia Computer Science*, vol. 160, pp. 289–297, 2019.

[5] T. Davenport and R. Kalakota, "The potential for artificial intelligence in healthcare," *Future healthcare journal*, vol. 6, no. 2, p. 94, 2019.

[6] Z. Nematzadeh, R. Ibrahim, and A. Selamat, "Comparative studies on breast cancer classifications with k-fold cross validations using machine learning techniques," in *2015 10th Asian Control Conference (ASCC)*. IEEE, 2015, pp. 1–6.

[7] S. Tuli, N. Basumatary, S. S. Gill, M. Kahani, R. C. Arya, G. S. Wander, and R. Buyya, "Healthfog: An ensemble deep learning based smart healthcare system for automatic diagnosis of heart diseases in integrated iot and fog computing environments," *Future Generation Computer Systems*, vol. 104, pp. 187–200, 2020.

[8] M. Munir, S. A. Siddiqui, M. A. Chattha, A. Dengel, and S. Ahmed, "Fusead: unsupervised anomaly detection in streaming sensors data by fusing statistical and deep learning models," *Sensors*, vol. 19, no. 11, p. 2451, 2019.

[9] J. R. Hageman, "The coronavirus disease 2019 (covid-19)," *Pediatric annals*, vol. 49, no. 3, pp. e99–e100, 2020.

[10] M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, "A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the covid-19 pandemic," *Measurement*, vol. 167, p. 108288, 2020.

[11] B. QIN and D. LI, "Identifying facemask-wearing condition using image super-resolution with classification network to prevent covid-19," *Research Square*, 2020.

[12] G. J. Chowdary, N. S. Punn, S. K. Sonbhadra, and S. Agarwal, "Face mask detection using transfer learning of inceptionv3," *arXiv preprint arXiv:2009.08369*, 2020.

[13] N. El-Rashidy, S. El-Sappagh, S. Islam, H. M. El-Bakry, and S. Abdelrazek, "End-to-end deep learning framework for coronavirus (covid-19) detection and monitoring," *Electronics*, vol. 9, no. 9, p. 1439, 2020.

[14] T. Ozturk, M. Talo, E. A. Yildirim, U. B. Baloglu, O. Yildirim, and U. R. Acharya, "Automated detection of covid-19 cases using deep neural networks with x-ray images," *Computers in Biology and Medicine*, p. 103792, 2020.

[15] M. Barstugan, U. Ozkaya, and S. Ozturk, "Coronavirus (covid-19) classification using ct images by machine learning methods," *arXiv preprint arXiv:2003.09424*, 2020.

[16] Y. Pathak, P. K. Shukla, A. Tiwari, S. Stalin, S. Singh, and P. K. Shukla, "Deep transfer learning based classification model for covid-19 disease," *IRBM*, 2020.

[17] N. Tsiknakis, E. Trivizakis, E. E. Vassalou, G. Z. Papadakis, D. A. Spandidos, A. Tsatsakis, J. Sánchez-García, R. López-González, N. Papanikolaou, A. H. Karantanas *et al.*, "Interpretable artificial intelligence framework for covid-19 screening on chest x-rays," *Experimental and Therapeutic Medicine*, vol. 20, no. 2, pp. 727–735, 2020.

[18] M. Otoom, N. Otoum, M. A. Alzubaidi, Y. Etoom, and R. Banihani, "An iot-based framework for early identification and monitoring of covid-19 cases," *Biomedical Signal Processing and Control*, vol. 62, p. 102149, 2020.

[19] M. S. Hossain, G. Muhammad, and N. Guizani, "Explainable ai and mass surveillance system-based healthcare framework to combat covid-i9 like pandemics," *IEEE Network*, vol. 34, no. 4, pp. 126–132, 2020.

[20] A. Kumar, K. Sharma, H. Singh, S. G. Naugriya, S. S. Gill, and R. Buyya, "A drone-based networked system and methods for combating coronavirus disease (covid-19) pandemic," *Future Generation Computer Systems*, vol. 115, pp. 1–19, 2020.

[21] J. Read, A. Bifet, B. Pfahringer, and G. Holmes, "Batch-incremental versus instance-incremental learning in dynamic and evolving data," in *International symposium on intelligent data analysis*. Springer, 2012, pp. 313–323.

[22] A. Kallel, M. Rekik, and M. Khemakhem, "Iot-fog-cloud based architecture for smart systems: Prototypes of autism and covid-19 monitoring systems," *Software: Practice and Experience*, 2020.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[24] J. Montiel, M. Halford, S. M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H. M. Gomes, J. Read, T. Abdessalem *et al.*, "River: machine learning for streaming data in python," *arXiv preprint arXiv:2012.04740*, 2020.

[25] N. M. Jagirdar, "Online machine learning algorithms review and comparison in healthcare," Ph.D. dissertation, University of Tennessee, The address of the publisher, 12 2018, an optional note.

[26] T. M. Mitchell *et al.*, "Machine learning," McGraw-hill New York, Tech. Rep., 1997.

[27] T. L. Hayes and C. Kanan, "Lifelong machine learning with deep streaming linear discriminant analysis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 220–221.

[28] E. Tartaglione, C. A. Barbano, C. Berzovini, M. Calandri, and M. Grangetto, "Unveiling covid-19 from chest x-ray with deep learning: a hurdles race with small data," *arXiv preprint arXiv:2004.05405*, 2020.

[29] I. Yasser, A. Twakol, A. El-Khalek, A. Samrah, A. Salama *et al.*, "Covid-x: Novel health-fog framework based on neutrosophic classifier for confrontation covid-19," *Neutrosophic Sets and Systems*, vol. 35, no. 1, p. 1, 2020.

[30] C. Huang, Y. Wang, X. Li, L. Ren, J. Zhao, Y. Hu, L. Zhang, G. Fan, J. Xu, X. Gu *et al.*, "Clinical features of patients infected with 2019 novel coronavirus in wuhan, china," *The lancet*, vol. 395, no. 10223, pp. 497–506, 2020.

[31] C. Wu, X. Chen, Y. Cai, X. Zhou, S. Xu, H. Huang, L. Zhang, X. Zhou, C. Du, Y. Zhang *et al.*, "Risk factors associated with acute respiratory distress syndrome and death in patients with coronavirus disease 2019 pneumonia in wuhan, china," *JAMA internal medicine*, 2020.

[32] B. Zwattendorfer, K. Stranacher, and A. Tauber, "Towards a federated identity as a service model," in *International Conference on Electronic Government and the Information Systems Perspective*. Springer, 2013, pp. 43–57.

[33] R. Rojas *et al.*, "Adaboost and the super bowl of classifiers a tutorial introduction to adaptive boosting," *Freie University, Berlin, Tech. Rep*, 2009.

[34] Y. Tang, "Deep learning using linear support vector machines," *arXiv preprint arXiv:1306.0239*, 2013.

[35] Q. Hu, D. Yu, and Z. Xie, "Neighborhood classifiers," *Expert systems with applications*, vol. 34, no. 2, pp. 866–876, 2008.

[36] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE transactions on systems, man, and cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.

[37] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[38] M. Egmont-Petersen, J. L. Talmon, A. Hasman, and A. W. Ambergen, "Assessing the importance of features for multi-layer perceptrons," *Neural networks*, vol. 11, no. 4, pp. 623–635, 1998.

[39] K. P. Murphy *et al.*, "Naive bayes classifiers," *University of British Columbia*, vol. 18, p. 60, 2006.

[40] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[41] M. Yildirim and A. Cinar, "A deep learning based hybrid approach for covid-19 disease detections," *Traitement du Signal*, vol. 37, no. 3, pp. 461–468, 2020.

[42] D. Anguita, L. Ghelardoni, A. Ghio, L. Oneto, and S. Ridella, "The'k'in k-fold cross validation." in *ESANN*, 2012.

[43] S. Arlot, A. Celisse *et al.*, "A survey of cross-validation procedures for model selection," *Statistics surveys*, vol. 4, pp. 40–79, 2010.

[44] G. Varoquaux, P. R. Raamana, D. A. Engemann, A. Hoyos-Idrobo, Y. Schwartz, and B. Thirion, "Assessing and tuning brain decoders: cross-validation, caveats, and guidelines," *NeuroImage*, vol. 145, pp. 166–179, 2017.

[45] D. Ologunagba and S. Kattel, "Machine learning prediction of surface segregation energies on low index bimetallic surfaces," *Energies*, vol. 13, no. 9, p. 2182, 2020.

Ameni Kallel received her Diploma of Engineer in Computer Science from the National School of Engineering of Sfax in 2012. Between 2012 and 2016, She worked in the Profesional Cloud and Network entreprise in Sfax as an responsible for engineering and technical operations for Infrastructure-as-as-Service (IaaS) platform. She joined the Higher Institute of Technological Studies in Sidi Bouzid as an Technologiste of Computer in 2016. Her main fields of interest include Virtualization, Cloud Computing, Internet of Things, with a current focus on dynamic allocation and management of virtualized compute and network resources.

Dr. Molka Rekik received the PhD in Computer Science from the Faculty of Economics and Management of Sfax (FSEGS)-University of Sfax (Tunisia) in 2017. Since 2018, she has been Associate professor at Information Systems department of Prince Sattam Bin Abdulaziz University, Kingdom of Saudi Arabia. She is a member of the Multimedia, Information systems and Advanced Computing Laboratory (Miracl), University of Sfax. Her research interests are in the Business Intelligence area with focus on business process variability management, IoT-aware BPM, fog and cloud computing and semantic cloud services description. Contact her m.rekik@psau.edu.sa or molka.rekik@gmail.com

Dr. Mahdi Khemakhem has got his M.Sc. in computer science from the University of Valenciennes UVHC (France) in 2003 and his Ph.D. from the same university conjunctly with the University of Sfax (Tunisia) in 2008. He has worked as lecturer and assistant professor at the University of Gabes (Tunisia) from 2003 to 2011. He then joined the department of Mathematics and Business Intelligence at the National School of Electronics and Telecommunications (ENET'COM) of the University of Sfax (Tunisia). Since 2017 he occupied the position of Associate Professor in the same university. His research interests include combinatorial optimization, complex systems modeling, heuristics, meta-heuristics and exact algorithms for combinatorial optimization problems in transportation and networks, resources management, cloud computing, etc.