IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 16 June 2023, accepted 2 July 2023, date of publication 17 July 2023, date of current version 14 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3296260

APPLIED RESEARCH

Machine Learning Techniques for Prognosis Estimation and Knowledge Discovery From Lab Test Results With Application to the COVID-19 Emergency

ALFONSO EMILIO GEREVINI[1], ROBERTO MAROLDI[1,2,3], MATTEO OLIVATO[1], LUCA PUTELLI[1], AND IVAN SERINA[1]

[1]Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Brescia, 25121 Brescia, Italy
[2]Dipartimento di Specialitá Medico-Chirurgiche, Scienze Radiologiche e Sanitá Pubblica, Università degli Studi di Brescia, 25121 Brescia, Italy
[3]ASST Spedali Civili di Brescia, 25123 Brescia, Italy

Corresponding author: Alfonso Emilio Gerevini (alfonso.gerevini@unibs.it)

This work was supported in part by the Italian Ministry of Research and University, "Fondo Integrativo Speciale per la Ricerca (FISR)", Project AICovidPrognosis2020 under Grant FISR2020IP_04373; and in part by TAILOR, a Project funded by European Union (EU) Horizon 2020 Research and Innovation Program under Gant 952215. The work of Alfonso Emilio Gerevini was supported by Fondazione Garda Valley, Italy.

ABSTRACT AI and Machine Learning (ML) offer powerful tools to support clinical decision making in emergency situations such as the COVID-19 pandemic. In this context, the application of ML requires to design predictive systems that have adequate accuracy and can effectively deal with issues concerning data quality, sensitive errors, uncertainty, and interpretability of the predictions. We present a methodology that deals with all these problems and a concrete study of its application to estimate the prognosis of hospitalised patients with COVID-19. In particular, we address the task of predicting the outcome (alive or deceased) of a patient at different times of her/his hospitalisation minimising false negatives (wrong survival predictions). The proposed methodology builds different optimised ML models to select those that perform the best to recognise, at different times of hospitalisation, patients who will have an unfavourable prognosis (decease). These models exploit a new algorithm, presented in the paper, that identifies an uncertainty threshold to rule out uncertain predictions with the purpose of making a ML model both more performing and more reliable. Moreover, we propose a general method for automatically extracting multi-variable prognostic rules from the available data. Such rules can provide possible new useful knowledge on the considered disease. We also show how they can be used effectively to explain the predictions made by the ML models. All proposed methods and techniques are experimentally evaluated in the context of our application task.

INDEX TERMS Smart healtchare, machine learning (ML), learning systems.

I. INTRODUCTION

The application of machine learning (ML) techniques in a clinical environment can offer powerful tools to help face emergency situations, such as the COVID-19 pandemic, at various levels [1], [2], [3], [4]. In particular, predicting possible adverse events or even the decease of a patient can provide valuable support in clinical decision-making [5], as well as in the management of the limited critical resources available in a hospital.

In this paper, we investigate a methodology that exploits machine learning to build effective predictive models for the mortality outcome of a patient with COVID-19 at different times of her/his hospitalisation. Predictive models operate as binary classifiers for mortality risk. We address several important issues that arise when ML techniques are applied in the medical context and, in particular, during an emergency situation. In such a situation, the lack of time or scarcity of the

The associate editor coordinating the review of this manuscript and approving it for publication was Shadi Alawneh.

available resources may considerably limit the amount and quality of data on the patient's health that can be acquired and organised for training a ML model, while an urgent prognostic assessment could be very helpful. Most existing ML studies for COVID-19 use many features of a hospitalised patient to build a predictive model (e.g., [6], [7], [8], [9]). Differently from them, in our work we use a restricted number of features that are based on demographic information (sex and age) and the results of common laboratory tests, which are relatively easy to acquire from a hospitalised patient, and to collect from a hospital, in order to create training datasets.

Moreover, the amount of available data can sensibly differ from one patient to another. For instance, the number of exams performed strongly depends on the length of stay, and patients hospitalised in the most critical phases of a pandemic could have incomplete data. This makes generating datasets that are suitable for exploiting the ML techniques non-trivial.

Another important aspect to consider in a very sensitive context like clinical practise is that ML tools need to have high predictive performance. Therefore, it is necessary to perform an accurate training of the algorithms and a statistically sound evaluation of their performance. This evaluation should also include a mechanism to know which predictions are more reliable and which are not, in order to avoid costly and critical mistakes [10]. For this purpose, we investigated the degree of certainty of the predictions made by our ML models. Since standard approaches such as calibration [11] or NGBoost [12] cannot be effectively adopted in our context, mainly due to the limited size of the datasets [13], [14], [15], we designed a new algorithm to identify and rule out uncertain predictions. This is a general algorithm that can be used for domains other than COVID-19.

Furthermore, in ML healthcare applications, it is important to inform the user about the main criteria on which the provided prediction is based, to ensure that the prediction is well justified and not biased [16] We propose a method to address this explainability issue by a collection of (user-interpretable) if-then rules that are automatically extracted. The extraction of such rules uses simplified datasets where only the most important features, identified by SHAP [17], are considered. Our method belongs to the post-hoc approach to explainability [18], which deals with the extraction of explanatory information from an existing system, such as a complex ML model, that does not allow a user to directly understand its mechanisms in providing a result (e.g., a prediction), and can be seen as a ''black-box''. In relation to the conceptual framework for explainable AI in medicine recently proposed by Combi et al. [18], in which the *explainability* of a system is defined as the intersection of its interpretability, understandability, usability, and usefulness, our method supports and improves the interpretability and usefulness of the proposed ML system for the prediction of the prognosis.

In addition to being useful for explainability, the extracted rules can provide new knowledge that is useful, especially when the rules concern a scarcely known disease. The rules that were extracted from our models for the prognosis of COVID-19 have been analysed on the basis of the recent medical literature on COVID-19, as well as of the clinical practise in the hospital that provided the patients' data for building our ML models. This analysis confirmed their validity, and indicated that some of our extracted rules provide valuable clinical insights for the prognosis of COVID-19 patients.

Using raw data from more than 2000 patients collected in the period February–May 2020 of one of the hospitals that had more COVID-19 patients in Italy and Europe, Spedali Civili di Brescia (SCB), we built different datasets describing the ''clinical history'' of each patient during the hospitalisation. In particular, each dataset contains a ''snapshot'' of the infection conditions of each patient considered on a certain day after the start of hospitalisation. (As snapshot times for a patient, in the experimental analysis presented in the paper, we consider the 2nd, 4th, 6th, 8th, and 10th hospitalisation days, and the day before the end of the hospitalisation.) For each dataset, we built a different prediction model, allowing one to make progressive predictions over time that take into account the evolution of the disease severity in a patient, which helps to formulate a personalised estimation of the prognosis.

For generating the prediction models, we consider and analyse several ML algorithms. A preliminary experimental first comparison of their performance on our datasets showed that the methods based on Bagging and Boosting of decision trees were more promising than other methods [19]. A more in-depth analysis (in this paper) about the use of Feed-forward Neural Networks for irregular and complex datasets like those we have in our application context confirmed this initial observation. The resulting best-performing models showed good predictive performance on a randomly chosen test set of more than 350 patients. In particular, the overall system makes very few errors in predicting patient survival, i.e., the specificity of the prediction is very high.

Our datasets are designed and engineered to cope with some important issues about the available data, partially determined by the emergency, that include missing values and irregularities in performing the lab tests for the patients. We also propose a simple but effective technique to address a ''concept drift'' issue [20], [21], since on average the outcome of a hospitalised patient was significantly worse (many more deceased patients) during the earlier period of the emergency (March 2020) when in Northern Italy many people were hospitalised.

Summarising, in addition to proposing a new effective prediction system for the prognosis of COVID-19 patients, and experimentally analysing several machine learning methods for the considered task, from a methodological point of view the paper contains the following main contributions:

- a method for building several models at different days of hospitalisation (for our experiments, we consider some specific days, but other days could be chosen, as decided by the user);

- a method and an algorithm for identifying and handling uncertain predictions through a threshold that is automatically tuned at training time, and that we experimentally show to be effective at test time (using instances different from the training ones);
- a method for automatically discovering multi-variable prognostic rules from the patients' data that provide potentially useful clinical knowledge for better understanding a disease like COVID-19;
- a method for using the discovered rules to interpret the model predictions and provide explanations to the user.

The remainder of the paper is organised as follows. After an overview of the methodology, the main proposed techniques, and the issues addressed (Section II), we present our methods and system in four sections: Section III describes the techniques for creating the training and test sets; Section IV concerns the techniques for training and selecting the best performing ML models; Section V introduces our algorithm for handling uncertain predictions; Section VI regards the proposed model interpretation process and the procedure for automatically extracting prognostic knowledge in the form of multi-variable rules. Then, in Section VII we describe in detail our clinical data, their characteristics, and the data quality issues we encountered. Next, in Section VIII, we provide an experimental analysis aimed at (i) evaluating the performance of the various ML models considered in our system in order to identify the best performing ones, (ii) showing the effectiveness of our methods to handle uncertain predictions and to extract prognostic rules, (iii) providing a set of prognostic rules for hospitalised COVID-19 patients, of which we discuss clinical validity, and (iv) evaluating our method for predicting explanation through the rules discovered. Finally, in Section IX we discuss the related work and in Section X we give the conclusions.

II. OVERVIEW OF THE METHODOLOGY, PROPOSED TECHNIQUES AND ADDRESSED ISSUES

At a high level, our methodology for the problem of prognosis estimation through ML models consists of five main components, organised as described in Figure 1. The first component (C1) generates a sequence of datasets from the raw data of the patients about lab tests findings; each dataset is indexed by a length of stay of the patient (hospitalisation day), that we call the *Snapshot Dataset* for that time. Then, for each of these datasets, the second component (C2) builds a number of ML models by training different ML algorithms and configuring their hyperparameters. The third component (C3) evaluates such generated models to identify the best performing one for the considered hospitalisation day. The fourth component (C4) refines the selected model by including an (automatically generated) uncertainty threshold in order to make the prognostic prediction (either survival or decease) more reliable. The resulting final model can then be evaluated by a testing module using a test dataset. Finally, the purpose of the fifth component (C5) is explaining the system prediction for a patient, as well as to generate clinical knowledge in the form of prognostic rules.

In the remainder of this section, we first highlight the main issues we encountered in the design of these components, and then outline our solutions to deal with them, while in the next sections we provide a detailed description.

When applying machine learning to raw real-world data, there are a number of challenging issues to deal with. This is especially the case in the context of biomedical applications [22] and in emergency situations that hospitals have to face, such as the pandemic of COVID-19. These issues have a methodological impact on the design choices for building a ML-based system that is effective and reliable. The main issues that we address are:

- handling length-of-stay dependent data (in component C1);
- dealing with incomplete and changing data about the patient's health condition (in component C1);
- handling the evolution of the data distribution (in component C1);
- performing accurate training and selection of the learning method (in components C2 and C3);
- designing an effective algorithm for handling uncertain and unreliable predictions (in component C4);
- designing a method for providing user-interpretable explanations of the system predictions and discovery of clinical knowledge from them (in component C5).

A. LENGTH-OF-STAY DEPENDENT DATA

Estimating the prognosis of a patient can be done at different times during her/his hospitalisation using different information about the patient's health condition. Moreover, the length of stay can sensibly differ from one patient to another, depending on a number of factors such as the patient's age, comorbidities, and overall health conditions; thus, the total number of performed lab tests and the relative findings significantly vary among the patients. However, typically machine learning algorithms require a fixed amount of features for each sample in the training/test dataset, and there is the need to even each patient's sample in a standard format that can be processed by the machine learning algorithms. In order to deal with this, we introduce the concept of *patient snapshots* at different days of hospitalisation (e.g., at the sixth day after the admission), each of which represents the patient's health conditions as indicated by the latest lab tests performed with respect to the corresponding day.

B. INCOMPLETE AND CHANGING DATA ABOUT THE PATIENT'S CONDITION

While patient snapshots allow to standardise the lab tests and possibly other exams considered for each patient, it introduces other issues. Lab tests and exams are not performed at a regular frequency due, e.g., to the different kinds and timing of the relative procedures, the availability of the required resources (X-Ray machines, laboratory equipment, technical staff, etc.), or to the different severity of the health conditions

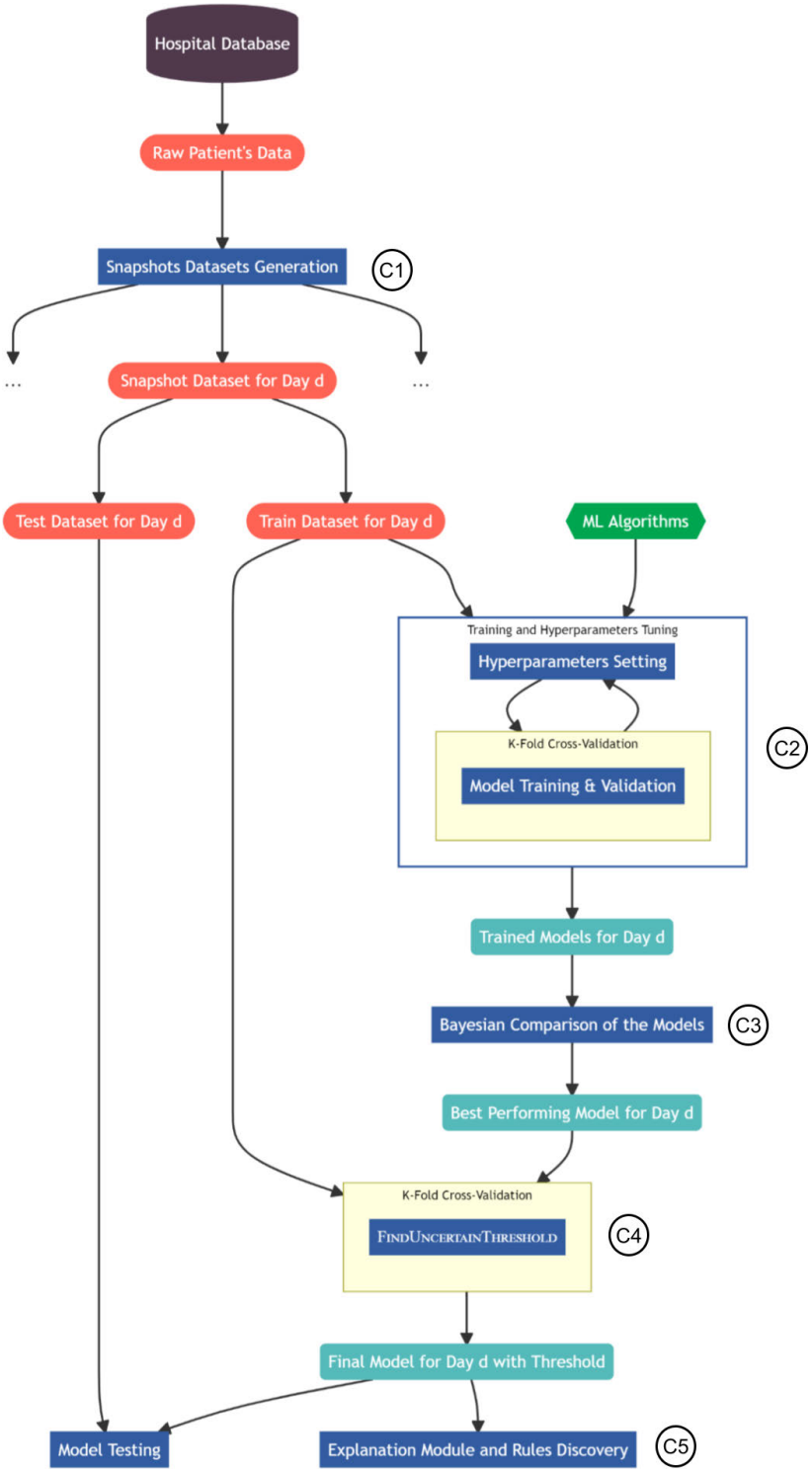

FIGURE 1. Diagram representing the main steps of our methodology for generating the final model. Starting from raw data and the Hospitalisation Day, we create a Snapshot Dataset (*Snapshot Dataset Generation*) representing the patients' conditions in that day. Then, we perform the *Training and Hyperparameter Tuning* of several ML algorithms. We select the best performing one in terms of performance via a Bayesian approach (*Bayesian Comparison of the Models*). Finally, we compute a threshold under which a prediction made by the ML model has to be considered uncertain (*FindUncertainThreshold*), generating the final model. The same methodology is applied at different hospitalisation days, therefore we have several snapshot datasets and we generate multiple models.

of the patients. This leads to the need of handling *missing values* and *outdated values*. For a snapshot of a patient at a certain day, we have a missing value for a lab test feature if that test has never been performed since the patient admission; while we have an outdated value for a feature if the corresponding lab test was performed several days earlier than the snapshot day. The outdated findings of the lab test could be inconsistent with the current conditions of the patient, and so they could mislead the predictive system. Consequently, we should introduce additional data in the snapshots to fill in the missing values through effective imputation techniques, and deal with the outdated values. Another important aspect that cannot be captured by the snapshots is the *evolution of the patient's condition*, from the admission day to the snapshot day, which we handle by introducing two additional features derived from the original data for representing the *trend of the disease*.

C. DATA DISTRIBUTION EVOLUTION

In emergency conditions such as an epidemic, an unfavourable prognosis (decease) of a patient can be influenced by multiple external factors, including the high number of patients currently hospitalised, the limited availability of ICU beds or other critical resources, the experimentation of new therapies, and the progressive increase of clinical knowledge. These factors can vary over time, changing the data distribution. In machine learning, this change is generally known as *concept drift* [20], [21], and a classical method to deal with it is training the algorithm using only a subset of samples, depending on the data distribution considered [20], [23]. This method has the drawback of significantly reducing training data, and in biomedical applications (which are often based on a limited number of patients) it can lead to poor predictive results. Thus, we follow a different approach, using the full dataset with an additional feature that helps the learning algorithm to discriminate if a patient is hospitalised during a highly critical period or not. This allows to exploit more training data, and achieve better predictive performance.

D. TRAINING AND SELECTION OF THE LEARNING METHOD

The use of real-world data poses crucial challenges also to the application of standard machine learning methods. While machine learning has been proven to be very effective in biomedical domains, typical issues of real-world data such as noise, missing values, data inconsistency, limited amount of available data, and data evolution have a negative impact on the performance of the generated ML models. This can be exacerbated when data are collected in emergency situations. Consequently, choosing an effective method among the variety of available learning algorithms and deriving, after adequate pre-processing of the data, an accurate ML model is a task that requires careful training of the algorithms and a thorough comparative evaluation of the generated models. Moreover, in our medical context, it is important to *minimise false negatives*, i.e., those patients in the training set whose adverse prognosis is not identified by the algorithm; these mistakes are the most critical and costly ones. Thus, we build our ML models and evaluate them with particular attention to this kind of classification errors.

In selecting the most adequate learning algorithm, it is particularly important to properly *configure the hyperparameters* of the algorithm, since this can have a huge impact on the resulting model. To do this, we automatically generate many alternative models, using different hyperparameter configurations, which are trained and validated in k-fold cross validation. The best performing models generated by the different configured algorithms are then evaluated through a *statistically sound comparison* based on a Bayesian approach [24].

E. UNCERTAINTY AND UNRELIABLE PREDICTIONS

In application domains where sensitive and risky decisions need to be taken, recognising when the suggestions provided by a decision support system are not confident enough is important to avoid costly mistakes and increase the reliability of the system. Especially in the first days of the hospitalisation stay, when there can be a scarcity of data indicating the patient's condition, the prognosis prediction obtained by a ML model can have a low degree of confidence. Therefore, we designed a method (Component C4 in Figure 1) to identify an *uncertainly threshold for recognising unreliable predictions*, which can be integrated in the prediction/classification model (in our case the one resulting from the analysis of Component C3 in Figure 1). This method is based on a new algorithm, called FindUncertainThreshold, that examines the quality of the predictions at training time, and automatically identifies a threshold under which the system should consider the prediction not sufficiently reliable; in such a case, the system considers the patient's prognosis *unpredictable* and classifies it as "uncertain".

The proposed method is general in the sense that it can be used in other clinical applications and in different domains.

F. EXPLAINABILITY AND DISCOVERY OF PROGNOSTIC KNOWLEDGE

Finally, our methodology includes a new method for extracting useful clinical knowledge from the generated machine learning model, that can be applied to any domain. Starting from the most important features for the prediction, as identified by the standard SHAP algorithm [25], we propose a new *technique for automatically extracting and evaluating multi-variable rules for prognosis classification* that involve such features. These rules combine the results of lab tests that are quite common for estimating the prognosis of a hospitalised patient, and can provide new knowledge that, especially for a novel disease, is clinically interesting and useful. Moreover, we devised a *method that exploits such derived rules to provide user-interpretable explanations* of (most of) the predictions made by the ML models, which is

particularly important especially for applications in sensitive domains.

Overall, the full methodology starts from raw data, generates a sequence of snapshot datasets indexed by increasing length of stays of the patients, produces a ML model optimised by an uncertainty threshold for each of the snapshots, and generates prognostic rules for explaining the system prediction and providing new clinical knowledge. In the following sections, all the modules and techniques introduced above are described in detail.

III. GENERATION OF THE DATASETS FOR TRAINING AND TESTING

In this section, we describe the process of generating the training and test datasets (Component C1 of Figure 1) at different days of a patient hospitalisation, according to the current conditions of the patient that are reflected by the available lab findings and other features. The basic features that we consider for the datasets are:

- the age and sex of the patients;
- the values and dates of throat swab exams for COVID-19;
- the values and dates of several common lab tests, such as PCR, LDH, Ferritin, Troponin-T, White Blood Cells, D-Dimer, Lymphocytes or Neutrophils;
- the final prognosis of hospitalisation at the end of the stay, which is the classification value of our application (either in-hospital death, released survivor, or transferred to another hospital or rehabilitation centre).

From the data of these features for a patient we generate some additional temporal features that are described below. Other information, such as symptoms, comorbidities, generic health conditions, admission in ICU or clinical treatments were not available from the hospital that provided the data during the considered emergency period.

We remark that the general methodology that we follow in this work and the specific techniques that we propose are independent from the specific basic features that we use, in the sense that they can be exploited in other biomedical contexts, with more or less demographic information, different lab tests or other clinical exams.

A. TEMPORAL FEATURES AND PATIENT SNAPSHOTS

To provide a prediction for a patient at different hospitalisation times, we introduced the concept of *patient snapshot* to represent the patient health conditions at a given day. In this snapshot, for each lab test of Table 1, we consider its most recent value. In the ideal case, we should know the lab test findings at every day. However, as explained in Section II, in a real-world context the situation is often quite different. For example, in our data, if we consider taking a snapshot of a patient 14 days after hospital admission, we have cases with very recent values of PCR, LDH, or WBC (one or a few days before), very old values for Fibrinogen or Troponin-T (obtained at the first day of the hospitalisation) and even no value for Ferritin.

Given the difficulty to set a predefined threshold that separates recent and old values of the lab tests, we chose to always use the most recent value, even if it could be outdated. To allow the learning algorithm to capture that a value may not be significant for representing the current status of the patient (because too old), we introduced a feature called **ageing** for each test finding. If a lab test has been performed at day d_0, and the snapshot of a patient is taken at day d_1, the ageing is defined as the number of days between d_1 and d_0. If there is no available value for a lab test, its ageing is considered as a missing value.

Monitoring the conditions of a patient means knowing not only the patient status at a specific time, but also how her/his conditions evolve during the hospitalisation. For this purpose, we introduced two features called **short trend** and **long trend** that are defined as follows.

For each lab test, if there is no available value for a lab test, or if the patient has not performed the lab test at least two times, then the trend features have missing values. Otherwise, we consider a set of pairs (d, v), where d is the date of a lab test and v is its value, as points in the Cartesian space of dates and lab-test values. From such points we calculate a simple linear regression $y = ax + b$ whose angular coefficient a indicates the trend of the lab-test values. To ensure the reliability of such a coefficient for representing the trend, we use only lab-test points that are strongly correlated according to the *Pearson Correlation Coefficient*, which we constrained to be above 0.7, in the case of a positive trend, and less than -0.7 in the case of a negative trend. If the Pearson coefficient does not indicate a sufficiently strong correlation according to all available values, then we iteratively omit some points until it does so. We perform this in two different ways as shown in Figure 2, obtaining two coefficients that define the values of the short/long features:

- **short trend**: the coefficient of the linear regression is computed starting from the last lab-test point (most recent lab-test findings) and incrementally adding less-recent points, going backwards in time, until one of the above correlation thresholds holds;
- **long trend**: the coefficient of the linear regression is computed by first considering all lab-test points, and then progressively removing the oldest ones, until the remaining points lead to a linear regression satisfying one of the above correlation thresholds.

Finally, in order to deal with the concept drift issue mentioned in the previous section, we introduce a new feature, that we call **death rate**, which provides an indicator of the status of the epidemic emergency at a given day d (the considered patient's snapshot), and is defined as the ratio of all patients who died over all those discharged (dead or alive) over the t days preceding d (in our experiments we set t to 7 days).

Summarising, for each dataset representing a patient snapshot we have: for each considered lab test, the most recent value, the relative ageing, and two trend features; two static

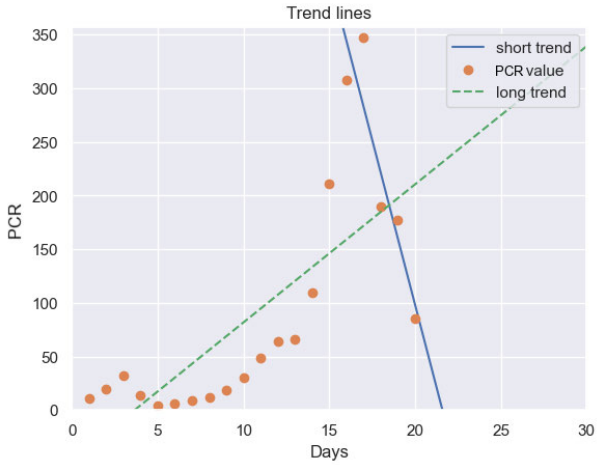

FIGURE 2. Plot of the short and long trend lines computed for the C-Reactive Protein (PCR) lab-test exam at 20th day of hospitalisation.

features for patient age and sex; the average death rate over the previous 7 days.

Figure 3 shows the feature extraction process for generating the hospitalization snapshot for a single patient. Given a specific hospitalisation day, we consider the hospitalisation data of the patient until that day, and then we transform the features (different exams) for the snapshot representation. For each feature, we consider the most recent value, and we compute its ageing and trends; then we combine these four features to derive the snapshot representation of the original feature. In addition, we add the death rate of the hospitalisation day chosen as input. The snapshot of the patient's hospitalisation for the given day is made by the union of all the previous features. The final snapshot dataset is obtained by repeating the same process for each patient who has a hospitalisation period longer than or equal to the number of days chosen for the snapshot generation, and merging all the resulting snapshots together.

B. CONSTRUCTION OF THE PATIENT SNAPSHOT DATASETS FOR TRAINING AND TESTING

While it is certainly useful to provide an initial indication of the prognosis based on the first lab tests, in the following days of hospitalisation more data are available, and they can be used to improve the prediction. Therefore, we can automatically create a training set and a test set for each day in a sequence of hospitalisation days during which we intend to progressively perform the prognosis prediction for a patient. In particular, we distinguish three configurations of these datasets:

- **Start of the hospitalisation**. This training/test dataset includes all patients' snapshots for the first days after admission.[1] In this snapshot, the ageing and trend features are not included. The purpose of the machine learning model trained using this dataset is to predict the outcome of the patient as soon as possible, with the earliest available information.
- **Different days of the hospitalisation**. In these datasets, the corresponding snapshots also contain the ageing and trend features (both short and long), and the lab-test values are the *most recent ones* in the available data. The purpose of each of these datasets (a pair of training and test sets for each considered day) is to capture the conditions of the patient at the corresponding hospitalisation day in order to estimate his/her prognosis at that time. We can consider all consecutive days of the patient hospitalisation or simply a selection of them; in our implementation and experiments, we assumed that the prognosis estimation is made every two days.
- **End of the hospitalisation**. This training/test dataset includes all patients' snapshots for a day near the end of the hospitalisation, capturing the patient conditions slightly before her/his release or death (in our implementation and experiments, we chose the day before the release/death).

It is important to observe that while the datasets of the latter days contain more information about the single patients (more lab tests findings, less missing values), the overall number of patients in the datasets decreases with the increase of the prediction day. This is due to the fact that more patients are released or died within longer periods of hospitalisation, and therefore such patients are not included in the corresponding datasets.

Moreover, each dataset can have missing values. In order to deal with this issue, we use an IterativeImputer with BayesianRidge as regressor [26], [27], [28], [29] for replacing missing values with automatically generated values.[2] A description of this and other imputing techniques that we tried is given in Appendix A-A.

Finally, we used stratified sampling for selecting 80% *of the patients for training* the models and 20% *for testing* them. This splitting of the data is done only once considering all patients, and not for each dataset. For instance, if a patient belongs to the training set at the start of the hospitalisation, then she/he does not belong to the test set of the following days.

IV. GENERATION OF THE PROGNOSIS MODELS

Our goal is to design and develop machine learning classifiers for recognising those patients who will have a fatal outcome *minimising false negatives* (i.e., incorrect classifications of patients with favourable prognosis). Therefore, in all phases of the project (preprocessing and dataset generation, model training, and testing of the trained models), when we evaluate alternative choices in the system design and the predictive performance of the resulting system, we use the F-β score

[1] In our implementation, as described in Section VII-A, we chose the first two days of hospitalisation. If a patient performed a lab test more than once during these initial days, the snapshot only considers the oldest value.

[2] In general, other imputing techniques can be used; if the dataset is particularly small, replacing the missing values with a constant or with the mean or the median value could improve the performance.

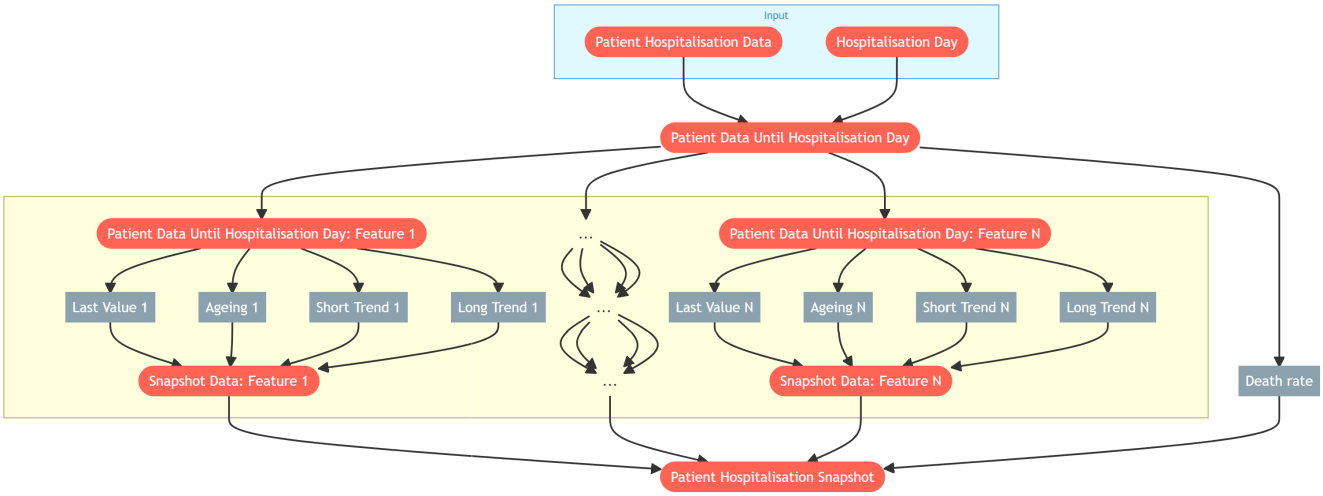

FIGURE 3. Diagram of the patient snapshot generation for a single patient. Starting from the *Patient Hospitalisation Data*, we select the data until a particular *Hospitalisation Day* (*d*). Then for each feature (numbered from 1 to *N*), we include the *Last Value* of that lab test, its *Ageing* and the two trend features (*Short Trend* and *Long Trend*), obtaining the snapshot data of that feature. In addition, we calculate the *Death Rate* until the day *d* considering the previous seven days of the Hospitalisation of all patients in the dataset. All these features are combined together to form the *Patient hospitalisation Snapshot* on day *d*.

with $\beta = 2$ as performance metric. The F-β score is the weighted harmonic mean of the *precision* and *recall* measures. Specifically, it is defined as follows [30]:

$$F\text{-}\beta = (1+\beta^2)\cdot\frac{Precision\cdot Recall}{\beta^2\cdot Precision + Recall}$$

where the parameter β indicates how many times the recall is more important with respect to the precision. We chose $\beta = 2$ in order to give particular importance to false negatives.

Another common issue in the generation of ML models for prognosis (outcome) estimation is *class imbalance*, since generally the deceased patients are much less than those released alive. Imbalanced classes lead the learning method to prefer the majority class, possibly causing a lack of performance in terms of F-2 Score. We address this problem through the *class weights* [31], [32] technique provided in the Scikit-Learn implementation.[3] This technique enables class balancing implicitly, improving performance without deleting original samples or adding synthetic ones. More details are given in A-B. Other popular solutions for imbalanced data are *sub-sampling* and *super-sampling* methods, which both alter the number of samples in the classes: the former removes samples from classes having more samples than the class with the fewest samples; the latter adds synthetic samples to the classes having fewer samples.

A. TRAINING AND HYPERPARAMETER TUNING

For each generated dataset of patients' snapshots, we trained and evaluated the following ML algorithms: Random Forests, Extra Trees, XGBoost, LightGBM (all based on ensemble of Decision Trees) and Feed-Forward Neural Networks. A description of these algorithms can be found in B.

It is well known that the performance of most machine learning algorithms strongly depends on the settings of their hyperparameters. Therefore, to obtain the best performance, the hyperparameters are (automatically) tuned through experiments that evaluate different hyperparameter configurations. The configuration obtaining the best performance is then used by the ML algorithm to build the prediction model. It is important to note that such models are then tested on an entirely separate test set, composed by instances used neither for the training of the algorithm nor for tuning the values of its hyperparameters. This process can be quite expensive especially if the machine learning algorithm has several hyperparameters with a large number of different possible configurations to consider. Therefore, we need a proper balance among the quality of the results, the strength of the evaluation for the hyperparamenter configurations, and its execution time. We perform a k-fold cross evaluation as follows:

1) The training set of each dataset is partitioned into k folds with $k = 10$.
2) For each considered configuration Σ of the hyperparameter values, the learning algorithm is run in k-fold cross validation.
3) For each fold of the k folds, the performance of the algorithm using Σ is evaluated in terms of F-2 score.
4) For each configuration Σ, the overall evaluation score of the k-fold cross validation is computed by averaging the scores obtained for each fold.
5) The hyperparameter configuration with the best overall score is selected.

[3]For more details on the computation of the Balanced Heuristic the reader can see the documentation page of the `class_weight` function at https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html

More specifically, for the algorithms based on ensemble of decision trees, we use a Random Search [33] optimising process with 4096 hyperparameter configurations randomly selected among all the possible configurations. These ensemble methods share some hyperparameters (e.g., the purity measure for node splitting, the number of decision trees to create, and their maximum depth). For Random Forest and Extra Trees, we considered four other hyperparameters provided by the implementation framework that we used, which are the same for both methods. For XGBoost, we considered ten more hyperparameters, and for LightGBM twelve more; such hyperparameters are related to the internal regularisation behaviours, feature selection, and the way the weak learners are built and assembled.[4] The full set of hyperparameters, the relative range of considered values, and the selected values are in B-E.

Random Forest and XGBoost algorithms are capable of exploiting parallel execution. Therefore, the Random Search approach, in combination with fast training algorithms, allows us to maximise parallel execution, obtaining good hyperparameters settings in a limited time, despite a high number of considered hyperparameter configurations (4096). On the contrary, an optimisation approach for the hyperparameters tuning, which adapts to the already evaluated configurations (i.e., Bayesian Optimisation, Tree-structured Parzen Estimator, etc.), puts some limitations in terms of highly parallel execution. Therefore, for Random Forest and XGBoost, Bayesian Optimisation generally achieves worse results in the same amount of time (i.e., far fewer considered configurations) compared to Random Search.

On the other hand, in neural networks approaches, the training time for a single hyperparameters configuration of a neural network and its initialisation overheads (weights initialisation, data transfer) is significantly higher than for models based on decision trees, and a smarter approach like Bayesian Optimisation allows us to obtain better hyperparameter settings with less trials, requiring nearly the same amount of time as the Random Search. Therefore, for optimising the neural network architecture, we implemented the Bayesian-optimisation approach via the Optuna framework [34] with a limit of 2048 search iterations. As previously described, the hyperparameters search is performed using 10-fold cross-validation on the training set of each dataset at different days of hospitalisation.

In addition, as proposed by Pastor-Pellicer et al. [35], the F-measure can be used in Neural Networks and other loss-based models as loss function, instead of the typical Log-Loss or Binary Cross Entropy, to improve performance (especially in terms of reducing false negatives) in highly unbalanced problems. In fact, other works [36], [37] use $F\text{-}\beta$ losses with different β values to specifically improve the performance for unbalanced and difficult tasks in the medical domain. Therefore, for our neural network models, we adopted a loss function based on the F-2 score metric defined as follows:

$$F\text{-}\beta_{loss} = 1 - \frac{(1+\beta^2)\cdot Precision \cdot Recall}{\beta^2 \cdot Precision + Recall + \varepsilon}$$

with $\beta = 2$, and $\varepsilon = 10^{-7}$ which is used to avoid zero-division errors.

B. MODEL SELECTION

Given that several datasets (with different number of instances, attributes, and missing values) are considered, there is no guarantee to find a single algorithm suitable for all of them. Therefore, after the hyperparameters tuning for each considered algorithm, we need to perform a comparison of the performance of all the generated models. For selecting the best performing one, instead of evaluating the relative performance of the models on a single validation dataset, we follow a Bayesian approach because it is statistically more robust. The Bayesian pairwise comparison was performed as suggested in [24]:

1) For each considered algorithm tuned using the best hyperparameter configuration, we compute a 10-fold cross validation repeated 10 times, for a total of 100 train-validation combinations.
2) Then, for each pair of tuned algorithms, we compute the mean $\hat{x}$ and the corrected variance $\hat{\sigma}^2$ of the differences between the performance scores (F-2 measures) of the models generated by the algorithms. The correction of the variance, suggested in [38], is needed because the scores of the models are not independent from each other, and so the relative performances could be underestimated.
3) For all pairs of models, we compute the posterior distribution of the performance difference as a Student's t distribution, as suggested in [24]. More formally, we compute

$$St\left(\mu; n-1, \hat{x}, \left(\frac{1}{n} + \frac{n_{test}}{n_{train}}\right)\cdot \hat{\sigma}^2\right)$$

where nn is the total number of samples, $\hat{x}$ is the mean of the performance score differences, n_{test} is the number of samples used for testing, n_{train} is the number of samples used for training, and $\hat{\sigma}^2$ is the variance of the observed differences.
4) Given the posterior distribution, we then compute the "worse probability" (i.e., the first model performs significantly worse than the second one), the "better probability" (i.e., the first mode performs significantly better than the second one), and the "Region Of Practical Equivalence" (ROPE), which denotes the probability of the two models having no significant differences in their performances. If two model performances differ by less than 1%, then they are considered practically equivalent. Therefore, our ROPE is the probability of x (i.e., the performance difference) being

[4]For more details the interested reader is referred to the documentation of the APIs of XGBoost and LightGBM available from https://xgboost.readthedocs.io/en/latest/ and https://lightgbm.readthedocs.io/en/latest/

in $[-0.01, 0.01]$, the worse probability is the probability of $x \in [-\infty, -0.01)$, and the better probability is the probability of $x \in (0.01, +\infty]$.

For each patient snapshot dataset, the model with the highest sum of ROPE probability and better probability is selected.

V. DECIDING UNCERTAIN PROGNOSIS THROUGH OPTIMISED CLASSIFICATION THRESHOLDS

As pointed out in the study by Brajer et al. [39] about the evaluation of mortality prediction models and their implementation in the real world to assist clinical decision-making, it is important that the predictions are associated with a degree of confidence evaluating their uncertainty [10]. This is particularly important in clinical applications where sensitive and difficult decisions are taken. Doctors have the need of discriminating the quality of the predictions suggested to them through an associated probability or degree of confidence [40].

Therefore, a classifier should not only predict the class of a given test sample, but also estimate the probability that this sample is a member of the predicted class. However, the most used ML models (Naive Bayes, Decision Tree, Random Forest, XGBoost, Artificial Neural Networks, etc.) give output probabilities that do not reflect the real probability of the sample to be of the predicted class [14]. On the other hand, it is possible to transform a machine learning model into a "good" probabilistic classifier as defined by [41] and [42] by a technique called *calibration* [11], [43], [44]. Calibrating a classifier allows to obtain better overall performances of the predictions by discharging the low probability predictions of the calibrated classifier [13], but it needs an high number of samples for avoiding overfitting issues and providing good results [13], [14], [15], which often are not available in emergency situations.

Given that the calibration approach seems infeasible in our context (a more detailed explanation is provided in Section IX), we propose a new technique to identify a threshold value under which the prediction of a classifier should be considered too uncertain, and so the prognosis of the patient should be classified as *unpredictable*. Such a technique is feasible for applications with complex and scarce data like ours. Instead of training a regressor, we consider the simpler task of finding a single value τ (the threshold) so that instances that have class-prediction (classification) probabilities below τ are classified as uncertain (unpredictable). Note that if τ is too high, many samples (patients) could be classified as uncertain, and our model would be much less useful in the clinical practise. To avoid this, during the search for an adequate τ value, we can impose a maximum fraction of (training) samples that can be considered uncertain. This is a parameter, called *max_u*, that in our experimental analysis was set to 0.25 (this value was considered adequate by the physicians collaborating in our project).

The threshold τ for a model A generated by a specific machine learning algorithm (in our case, the model selected as described in Section IV-B) is calculated *at training time* as follows. We evaluate A in k-cross validation, and we derive τ as the *average of the best thresholds* τ^i *found for each validation fold* i ($i = 1, \ldots, k$), where τ^i is computed by a new algorithm called FindUncertainThreshold.

Figure 4 shows the pseudocode of FindUncertainThreshold. Given the original classification labels L of the validation samples and their prediction probabilities P derived by the learning algorithm, FindUncertainThreshold performs the following main steps:

- The predicted labels L_{pred}, i.e., the output class values with the highest probabilities, and the relative probabilities P_{max} are calculated (lines 1-2).
- The algorithm evaluates the original performance score v calculated using the input score function (in our implementation, F-2 score) on the samples of the validation fold (line 3).
- The initial value of the threshold (τ) is set to the minimum probability in P_{max} (line 4).
- The loop of lines 6–18 finds an optimal threshold value τ and computes the score function for the validation set *reduced to the validation samples with predicted labels that have probabilities above* τ. The threshold values considered are obtained using the δ-increments defined at lines 5 and 7. First, we calculate the new threshold τ' by increasing the current threshold by δ, and then we derive the set S of sample ids with prediction probabilities higher than τ'. Next, we calculate the percentage u of samples that are labelled uncertain using the threshold τ'. If $u \geq max\_u$, we can terminate returning the current best new score v, and the corresponding threshold value τ (a higher threshold value cannot lead to label as uncertain fewer samples than the returned value τ). Otherwise, ($u < max\_u$), we set L' to the correct labels for the samples in S and L'_{pred} on the predicted labels for such samples, and we compute the new score value v' using L' and L'_{preds}. If v' is a better score than v, then we update both the threshold and the score values.

VI. TECHNIQUES FOR MODEL INTERPRETATION AND KNOWLEDGE DISCOVERY

Often, ML algorithms are seen as black boxes that provide no explanation to the user about their results or internal behaviour. However, the use of ML techniques in clinical practice requires that the provided predictions are supported by information that helps the user to interpret the results, explain how the prediction is made, and provide the main criteria used for it [45]. In addition, when a ML model for the prognosis estimation of a novel disease like COVID-19 performs well, explaining how the model works to the user could reveal new knowledge that is useful from the clinical point of view.

In this section, we propose a method for interpreting the predictions made by a machine learning model and for deriving prognostic knowledge in the form of multi-variable rules. Figure 5 gives a schematic overview of our method. First, the most important features for each snapshot dataset

FindUncertainThreshold 1: Algorithm for computing, during the training phase, an optimised prediction threshold under which the model classifies an instance as uncertain.

Input: label=–
- L: array of labels (alive or dead) with L[i] = correct label of sample i in the validation data (fold);
- P: array $[p_i = (p_{alive}, p_{dead})_i \mid$ i is the sample index in validation set], where p_{alive} is the probability of class alive returned by the classifier for sample i (analogously for p_{dead});
- max_u: maximum percentage of the samples in the validation set that can be labelled as uncertain (unpredictable);
- n: maximum number of thresholds to try;
- EvaluateScore: score function to maximise by dropping the uncertain samples;

Output: A pair (v, τ) where v is the score function value after dropping the uncertain samples and τ the optimized threshold value.

$L_{pred} \leftarrow$ array of labels such that L_{pred}[i] is the label predicted by the model (i.e., the label with the highest output probability) of the validation sample i;
$P_{max} \leftarrow$ array $[\max(p_{alive}, p_{dead})_i \mid (p_{alive}, p_{dead})_i \in P]$;
$v \leftarrow$ EvaluateScore(L, L_{pred});
$\tau \leftarrow$ min value in P_{max};
$\delta \leftarrow$ ((max value in P_{max}) − (min value in P_{max}))/n;
for i $\leftarrow$ 0 to n − 1 do
  $\tau' \leftarrow$ min value in $P_{max} + i \cdot \delta$;
  $S \leftarrow \{i \mid i$ is the id sample such that $P_{max}[i] > \tau'\}$
  $u \leftarrow 1 - (|S|/|P_{max}|)$;
  if $u \geq$ max_u then return (v, τ);
  $L' \leftarrow$ array of labels such that L[i] is the label of the validation sample i and $i \in S$;
  $L'_{pred} \leftarrow$ array of labels such that L_{pred}[i] is the predicted label of the validation sample i and $i \in S$;
  $v' \leftarrow$ EvaluateScore(L', L'_{pred});
  if $v' > v$ then
    $\tau \leftarrow \tau'$;
    $v \leftarrow v'$;
  end
end

FIGURE 4. Pseudocode of algorithm *FindUncertainThreshold*.

and corresponding best-performing prognosis-classification model (as described in Section IV) are identified and selected (Component C1 of Figure 5). Then a *Simplified Snapshot Dataset* containing only such selected features is generated, and a single Decision Tree is trained on this dataset, using the same training and hyperparameter tuning procedure presented in Section IV-A (Component C2 of Figure 5). Such decision tree is then processed to generate a collection of predictive rules (Component C3 of Figure 5). Finally, these rules are evaluated in terms of accuracy and agreement with the original model constructed using the full set of the features (Component C4 of Figure 5). In the following, we explain our methodology more in detail.

A. ANALYSIS OF THE MOST RELEVANT FEATURES AND SIMPLIFIED SNAPSHOT DATASETS

We analyse the contribution of the features used by our models through SHAP (SHapley Additive exPlanations), one of the most important methods to explain the prediction for an instance that a machine learning model makes [25]. This method is based on assigning to each feature a value, called *Shapley value*, which summarises the importance of the corresponding feature in the prediction made. Intuitively, this value is obtained by computing the contribution of the feature to the prediction of the model (a probability distribution over the possible class values) obtained with and without it in the dataset, for any possible combination of all the features [25].

The shapley values can be used not only for explaining a single prediction, but also for identifying which features are the most important for the model in general. Given a test set with N instances (patients), the overall Shapley value S_i for a feature i is the average contribution for each of the N instances, i.e.,

$$S_i = \frac{\sum_{j=1}^{N} s_{ij}}{N}$$

where s_i^j is the SHAP value of the feature i for the test instance j.

We calculated the Shapley values of our features for the best performing models in each dataset, and from these values we identified the features that are most important for making the predictions. To give an overall assessment of the relative

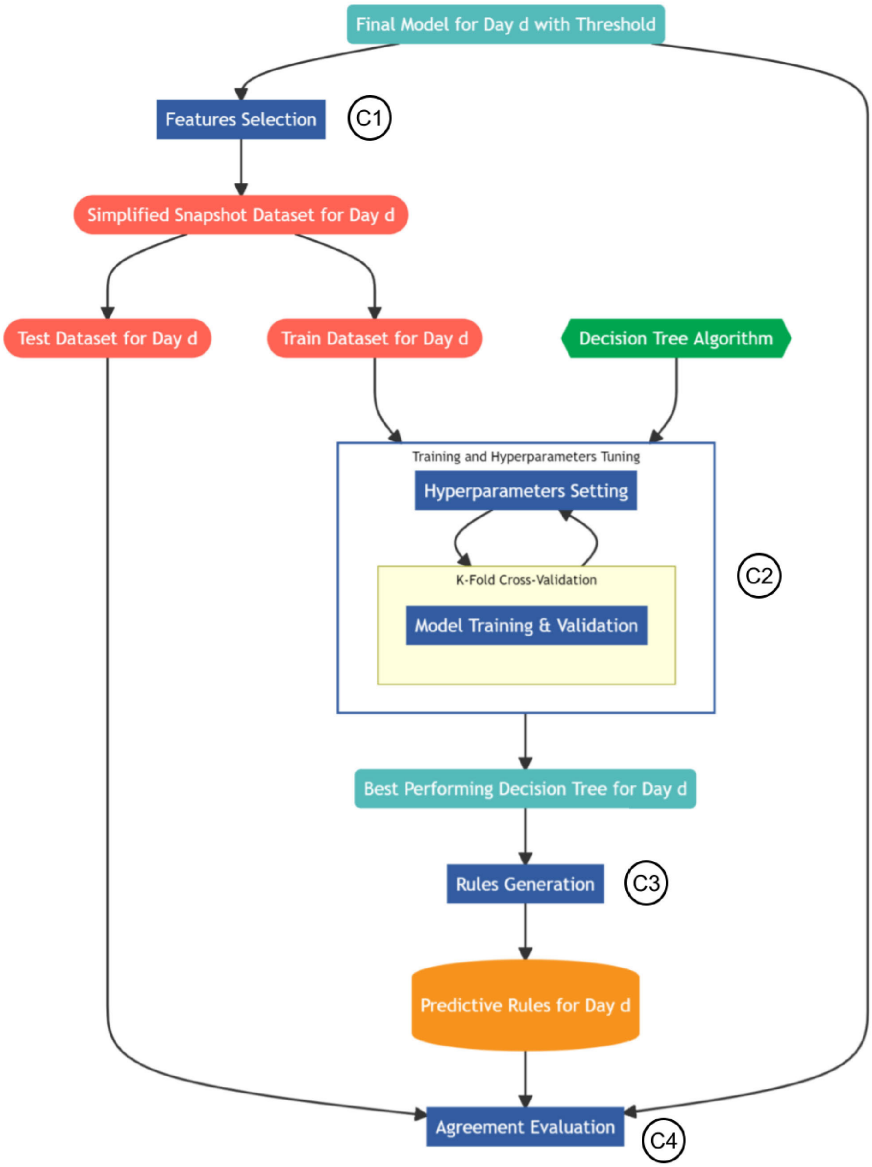

FIGURE 5. Methodology for deriving a collection of prediction rules for model intepretation and knowledge discovery. After computing the *Simplified Snapshot Dataset* with the most important features extracted by the *Features Selection* procedure, we train the *Best Performing Decision Tree*. This model is the result of the training and tuning many Decision Tree models (*Training and Hyperparameters Tuning*) with the aim of finding the best hyperparameters combination for the considered task. From the best performing decision tree, we extract the *Predictive Rules* via the *Rule Generation* procedure. Finally, such rules are evaluated in terms of their agreement (*Agreement Evaluation*) with the original model (*Final Generated Model*).

importance of our features, here we consider *an average Shapley value over all datasets*. First, for each feature i and each dataset d, we compute the sum S_i^d of the Shapley values of i for all tests instanced from d. Then we normalise each S_i^d to a value between 0 and 1. Finally, we calculate the arithmetic mean of the normalised values on all datasets: $\hat{S}_i = \sum_{d \in D} S_i^d / M$, where M is the number of features and D the set of the datasets considered.

Once we have computed a measure assessing the overall importance of the dataset features, defining a threshold on the Shapley values to discharge the less important features is not trivial and depends on the specific application context. In our context, we performed several trials executing the training and tuning procedure on simplified datasets, progressively excluding more features and evaluating the difference in terms of performance with respect to the original models built using all the features. The obtained final versions of our Simplified Snapshot Datasets include the demographic information, the death rate and 7 lab tests. The models built using such datasets perform

between 1 and 3 points worse than the original ones in terms of F-2 score.

B. A METHOD FOR GENERATING PREDICTION RULES

In a clinical application, it is important to provide not only high-accuracy predictions (and possibly minimising false negatives) but also explanations about the criteria the predictions are based on. We devised a technique to provide prediction explanations through implicative (if-then) rules extracted from the data of the form

$$c_1 \wedge \ldots \wedge c_n \Rightarrow \textit{class-value}$$

where $c_1 \wedge \ldots \wedge c_n$ is the *rule condition* defined as a conjunction of feature constraints, each of which is an inequality that bounds the value of a single feature (e.g., $c_i = age \leq 69$), and *class-value* is a prognostic prediction (either survival or decease). A rule can be evaluated in terms of its *support*, which is the percentage (or number) of instances in the dataset that satisfy the rule condition, and of its *accuracy*, that is, the percentage of instances in the dataset that are of the class predicted by the rule.

From a single decision tree we can (automatically) derive a prediction rule from each path connecting the root to a leaf node that classifies the instance. However, from an ensemble of decision trees, which is the best-performing method in our application, extracting this kind of rules can be much more complicated. For this reason, partially following the approach of [6], we propose to use the Simplified Snapshot Dataset described in the previous section to train a single Decision Tree, following the hyperparameter search described in Section IV-A. In general, such a simple prediction model does not perform as well as the model built with the complete Snapshot Dataset, especially in terms of the F-2 score. However, from the paths from the root to the leaf nodes of the decision tree, we can automatically extract a collection of prognostic rules that have remarkable accuracy and significant support for a large percentage of patients. For example, our rule generation approach derives the following rule for the second day of the hospitalisation of a COVID-19 patient

$$(age \leq 69) \wedge (LDH \leq 502) \wedge (neu/lym \leq 5.73) \Rightarrow survival$$

which is valid for 490 of the considered patients and has 97% accuracy in the test set (neu/lym stands for the ratio between neutrophils and lymphocytes). A generated rule can be effectively used for explaining the prediction made for a patient by a complex model (e.g., ensemble-based) built from a full dataset containing more features than the simplified dataset from which the rule was extracted (e.g., the snapshot dataset for a particular day) provided that the rule condition matches the patient's features. This explanation is effective only if three conditions hold:

- the rule has very high accuracy,
- the rule has adequate support, and
- the rule has a very high agreement with the class predicted by the complex model.

Given a rule that predicts class y, we define the *agreement of the rule* with the prediction model as the percentages of test instances that satisfy the rule condition and that the model classifies as y. For instance, if 90% of the test patients that satisfy the three conditions of the above rule are classified as ''survival'' by the model, then the agreement of the rule is 90%.

The set of rules discovered for our prognosis prediction task, together with their analysis in terms of accuracy, support and model agreement are given in the next section.

VII. DATA DESCRIPTION

In this section, we describe the raw data about our cohort of patients, from which the snapshot datasets were generated. Next, we analyse the data quality, and we report the pre-processing techniques that were employed to solve the most relevant issues.

A. AVAILABLE DATA

During the COVID-19 outbreak, from February to April 2020, at the Spedali Civili Hospital in Brescia, one of the largest hospitals in northern Italy, more than 2, 000 patients were hospitalised. From this hospital we had data for a total of 2015 hospitalised patients. For each of these patients, in addition to the information described in Section III, we include a score (an integer between 0 and 18) that evaluates the severity of pulmonary conditions resulting from radiographs according to the method described in [46]. While in our data such a RX score was manually assigned by the physicians, it can also be automatically calculated by a Convolutional Neural Network that analyses an X-ray image as described in [47].

The median age of our cohort of patients is 68 years (77 for the deceased patients, 65 for those released alive), and 65% of them are male. 81.6% of our patients were released alive, while the remaining 18.4% died during the hospitalisation. Please note that for the 4days, 6days, 8days and 10days datasets this percentage varies. This is because these datasets contain a subset of the overall cohort, excluding those patients who were released before the day selected for the dataset. For instance, the 8days dataset does not contain patients released or died before their 8th day of hospitalisation. The percentages of alive and decesead patients for these datasets are: 83.5% and 16.5% for 4days, 85.0% and 15.0% for 6days, 85.2% and 14.8% for 8days, 85.2% and 14.8% for 10days. In terms of mortality, there are no significant differences between male and female patients. As reported in Section IV, we addressed this class imbalance problem through *class weights* [31], [32]. For more information about this technique please see Appendix A-B.

Table 1 specifies the lab tests and the X-ray exam considered, their normal range of values (which is provided by the medical literature as a reference for healthy people), their median values in our set of patients, and the value ranges (first and third quartiles) for patients who were released alive and for the deceased ones. As we can observe, most of the

TABLE 1. Lab tests and X-ray exam performed during the hospitalisation. The second column shows the range of values considered clinically normal for each specific exam. The third column shows the median values extracted considering the lab test findings for our set of 2015 patients. The fourth and the fifth columns (each with 2 subcolumns) show, for the patients released alive and deceased respectively, the median and the range of values between the first and third quartile (in square brackets) of the distribution for the lab test findings.

		All	Alive		Deceased	
Lab test	Normal Range	Median	Median	Range	Median	Range
C-Reactive Protein (PCR)	≤ 10	39.5	34.3	[8.2, 86.8]	77.1	[26.0, 152.7]
Lactate dehydrogenase (LDH)	[80, 300]	295	280	[219, 363]	441	[328, 585]
Ferritin (Male)	[30, 400]	1093	1030	[565, 1684]	1646	[835, 2700]
Ferritin (Female)	[13, 150]	540	497	[243, 886]	1055	[452, 1987]
Troponin-T	≤ 14	22	19	[9, 73]	36	[19, 94]
White blood cell (WBC)	[4, 11]	7.4	7.1	[5.0, 10.1]	9.8	[6.6, 14.5]
D-dimer	≤ 250	649	553	[290, 1555]	1797	[671, 3817]
Fibrinogen	[180, 430]	442	442	[312, 582]	451	[289, 614]
Lymphocytes	[0.90, 4]	0.95	1.0	[0.7, 1.5]	0.6	[0.4, 0.9]
% of lymphocytes	[20, 45]	13.9	15.4	[8.8, 24.1]	6.5	[3.6, 11.6]
Neutrophils over lymphocytes	[0.8, 3.5]	5.3	4.8	[2.6, 9.4]	13.4	[6.9, 25.3]
Chest XRay-Score (RX)	< 7	8	8	[5, 11]	10	[7, 13]

median values (with the notable exceptions of WBC and LDH) of the lab test results of our cohort of patients are not in the normal range. In particular, we can see that the median values of PCR, Ferritin (for both male and female patients) and D-Dimer exceed the double of the maximum value in the normal range. If we analyze the values obtained from patients deceased and those who were released alive, we can notice some significant differences for most of the lab tests. For instance, the median value of the D-Dimer for deceased patients (1797) is more than three times the value for the alive patients (553). Similar results can be seen for PCR (77.1 versus 34.3) and Neutrophils over lymphocytes (13.4 versus 4.8). Even more worrying conditions can be seen if we observe the third quartile of the lab test results distribution. For instance, D-Dimer for a quarter of the deceased patients is higher than 3817, more than ten times the normal limit (250). Please note that for the values of Lymphocytes and their percentage, the most worrying conditions happen when the patient has a value below the normal range. In fact, deceased patients have a median of 6.5% of lymphocites (with a first quartile of 3.6) versus a median of 15.4% for the alive patients, while the minimum value of the normal range is 20%.

Although we observed that some lab tests have values in the normal range (almost exclusively for the patients released alive), we decided to include them in our datasets anyway. These values can provide useful information to the model, especially if they are at the extremes of the normal range, or if they have a different distribution for alive and deceased patients. In fact, for deceased patients, there are always at least some features outside their normal range. Note that the lowest/highest values of each normal range are defined by the first/third quartile of the distribution; therefore, the minimum and maximum observed values are not included in the normal range. Moreover, particular combinations of some values that are all in the normal range (and so are individually non-problematic) may, in spite of their normality, be informative. For example, four exams having values close to the minimum in their normal range could indicate a bad condition that is really informative. Furthermore, when considered in combination with other lab tests and trend features, they could indicate a favourable (or unfavourable) progression of the disease.

In accordance with the methodology described in Section III-B, we trained and tested ML models for the following hospitalisation times: start of the hospitalisation, considering the first lab tests performed during the first 2 days of hospitalisation; 4, 6, 8 and 10 days after the admission, considering the latest laboratory tests and also including ageing and trend features; the day before release or death of the patient, that we indicate with *End* of the hospitalisation.

B. DATA QUALITY ISSUES AND PRE-PROCESSING

As we discussed in Section II, medical data collected during emergency situations can present several issues in terms of data quality. In particular, it is often not possible to guarantee a regular frequency of performing the lab tests. The frequency can significantly vary for different lab tests and patients. In Table 2, we report the frequency to which the lab tests are performed in our dataset in terms of average and standard deviation. We can observe that there are significant differences among the different lab tests. For instance, the chest X-Ray score is provided on average every 7.34 days, while lab tests such as PCR and WBC are performed more frequently (respectively, every 2.38 and 2.06 days). Note that the different features related to Neutrophils and Lymphocytes have the same frequency because these are different results of the same lab test. Considering the standard deviation, there can be significant differences in terms of lab test frequency among different patients. Although most regular lab tests (such as PCR or Lymphocytes) have very low standard deviation values, which means that the frequency is almost the same for all patients, less frequent exams, such as Troponin-T and Fibrinogen, have a higher standard deviation (7.25 and 4.94 days, respectively).

During the pre-processing phases of our system, as reported in Section III-A, we extend our snapshot datasets with the additional *ageing* features (excluding the 2days dataset). Such features count the number of days passed since the lab test was performed, and allow the learning algorithms

TABLE 2. Frequency and ageing for the considered lab tests. The frequency columns (Freq.) report the frequency of the lab tests in terms of average and standard deviation. The ageing columns (Age) report the average ageing of the lab tests for each snapshot.

Lab test	Avg. Freq.	St.Dev. Freq.	Age 4days	Age 6days	Age 8days	Age 10days	Age end
C-Reactive Protein (PCR)	2.38	1.56	1.45	1.48	1.71	1.84	2.06
Lactate dehydrogenase (LDH)	5.1	4.51	2.12	2.98	3.66	4.09	4.45
Ferritin	4.46	5.15	1.83	2.41	2.80	2.91	3.62
Troponin-T	4.88	7.25	2.57	4.07	5.27	6.38	7.54
White blood cell (WBC)	2.06	1.38	1.32	1.35	1.62	1.61	1.69
D-dimer	3.17	4.74	1.76	2.21	2.51	2.78	3.46
Fibrinogen	4.35	4.94	2.12	2.79	3.40	3.87	4.35
Lymphocytes	2.68	1.81	1.54	1.62	1.93	1.99	1.92
% of lymphocytes	2.68	1.81	1.54	1.62	1.93	1.99	1.92
Neutrophils over lymphocytes	2.68	1.81	1.54	1.62	1.93	1.99	1.92
Chest XRay-Score (RX)	7.34	6.17	2.07	2.62	3.39	3.88	4.88

to consider whether a lab test result represents the current condition of a patient, or if it is too old to be valid. In Table 2, we report the average ageing of each lab test in our snapshot datasets. We can observe that some exams, such as Troponin-T or LDH can have a significant ageing, such as 6.38 and 4.09 days, respectively, on the 10*days* dataset. On the contrary, Neutrophils, Lymphocytes and PCR have an ageing that almost never exceeds two days, thus providing a very recent account of the patient's current conditions.

From the machine learning perspective the most important issue related to the irregular frequency of the lab tests concerns the missing values. In fact, for every considered day, several patients of our cohort did not perform one or more specific lab test. Therefore, the values of the corresponding features in the related snapshot datasets are considered missing. Table 3 reports the percentage of missing values for each lab test in each of our snapshot dataset. D-Dimer, Ferritin, Troponin-T, Fibrinogen and Chest XRay-Score are particularly problematic since these exams have a significant percentage of missing values especially in the first days of hospitalisation. For instance, 60.43% of the patients did not have their D-Dimer tested in the first two days of hospitalisation. There are significant issues also for the 8days and 10days snapshots: for almost 30% of the patients, we do not have a value of Ferritin even after 10 days of hospitalisation. Although exams such as PCR, WBC or lymphocytes are performed more frequently, the datasets related to the first days of hospitalisation can still contain some missing values (for instance, 3.19% of the patients did not have their PCR tested in the first 4 days).

To solve the missing-values problem, we tested the use of standard imputation techniques [48], [49], such as:

- substituting each missing value with a constant, with the mean or the median of the values in the snapshot datasets, and
- estimating the most probable value given the remaining values by implementing an Iterative Imputer with Bayesian Ridge Regression [26], [27], [28], [29], as already reported in Section III-B. This technique obtained the best results.

More information on these techniques is given in Appendix A-A.

VIII. EXPERIMENTAL ANALYSIS

In this section, we describe the context in which we evaluate the proposed methods, we present the results of the overall machine learning process, as produced by the components of Figure 1, and we provide an experimental evaluation concerning prediction performance and discovered knowledge.[5] Next we analyse the products of our methodology, showing the results of the Bayesian comparison assessing the generated models, the features that turned out to be most important in the considered application context, and the prognostic rules that were discovered for COVID-19 patients; such rules are discussed also comparing them with the medical literature. Finally, we experimentally evaluate our models and prognostic rules using a test set made of patients totally different from those used for the training and tuning processes.

A. LEARNED MODELS AND RULES

For each snapshot dataset that was generated, we performed the training and hyperparameter tuning, and we selected the best performing model using the Bayesian Approach (see Figure 1). The learning algorithms considered are as follows: Random Forests, Extra Trees, XGBoost, LightGBM, and Feed-forward Neural Networks. The hyperparameters and their spaces of values that we considered for the tuning of Ensembles of Decision Trees (Table 9) and Neural Networks (Table 8) are specified in B-E.

The results of the Bayesian analysis comparing the learning algorithms are given in Table 4. In the second column (Model 1) we have the best-performing model, and in the third column (Model 2) the other compared models performing worse than the first one; in the other columns, we have the probabilities (better, worse, and ROPE) for the first model versus the second one. For the 2days, 4days and 6days datasets, the XGBoost and LightGBM models perform quite similarly; indeed, their machine learning approach is very similar (both are based on advanced gradient boosting). For datasets 8days and 10days, the ExtraTrees model becomes the second (probabilistically) best model in terms of performance behind the XGBoost model, instead of the LGBM model.

[5]Our system is implemented using the Scikit-Learn [50] library for Python; all the experiments were carried out using an Intel (R) Xeon (R) Gold 6140M CPU @ 2.30GHz.

TABLE 3. Percentage of missing values for every lab test considered in each snapshot.

Lab test	2days	4 days	6days	8days	10days	end
C-Reactive Protein (PCR)	7.88	3.19	1.68	1.04	0.7	3.09
Lactate dehydrogenase (LDH)	16.31	10.84	8.48	7.8	6.39	9.87
Ferritin	47.21	40.57	35.7	31.75	29.59	31.77
Troponin-T	56.18	52.18	50.42	49.56	49.52	48.1
White blood cell (WBC)	2.68	0.65	0.07	0.04	0.02	1.97
D-dimer	60.43	55.2	51.98	48.32	46.66	47.04
Fibrinogen	33.7	27.87	25.02	23.73	21.33	25.21
Lymphocytes	9.34	3.47	1.87	0.6	0.12	3.79
% of lymphocytes	9.34	3.47	1.87	0.6	0.12	3.79
Neutrophils over lymphocytes	9.34	3.47	1.87	0.6	0.12	3.79
Chest XRay-Score (RX)	61.11	40.41	28.52	23.44	17.54	19.76

TABLE 4. Statistical comparison between pairs of considered models via the Bayesian method.

Dataset	Model 1	Model 2	Worse prob.	Better prob.	ROPE prob.
2days	XGB	LGBM	11.8	41.4	46.8
	XGB	DNN	9.5	49.9	40.6
	XGB	RF	11.2	50.3	38.5
	XGB	ET	3.8	71.9	24.3
4days	LGBM	XGB	23.0	33.2	43.8
	LGBM	ET	10.3	57.0	32.7
	LGBM	RF	1.2	87.1	11.7
	LGBM	DNN	0.0	99.2	0.7
6days	XGB	LGBM	10.8	51.3	37.9
	XGB	DNN	2.7	84.3	12.9
	XGB	ET	1.2	89.3	9.5
	XGB	RF	0.6	93.9	5.5
8days	XGB	ET	23.1	46.8	30.1
	XGB	LGBM	18.3	47.2	34.6
	XGB	DNN	3.0	85.9	11.1
	XGB	RF	1.7	89.3	9.0
10days	XGB	ET	24.2	38.3	37.6
	XGB	LGBM	15.8	56.0	28.2
	XGB	DNN	15.2	63.8	21.0
	XGB	RF	4.3	80.6	15.2
end	ET	DNN	13.2	31.1	55.8
	ET	LGBM	10.1	43.0	46.9
	ET	XGB	2.2	76.9	20.9
	ET	RF	0.6	90.0	9.4

Finally, the ExtraTrees model is the best for the End dataset, surprisingly followed by the DeepNeuralNetworks (DNN) model. It is worth noting that the End dataset differs from the other datasets because it has a low number of missing values for every patient, and furthermore it contains all the available patients (with the data regarding the last day of their hospitalisation). The larger amount of data and their better quality help the DNN model, increasing its performance for this dataset, and making it comparable with the models based on decision trees. In general, however, the best performing algorithms are the ones based on boosting techniques.

We believe that the main problems with using neural networks in our context are the scarcity of data, the high number of missing values in the data sources, the unbalanced classes, and the structured nature of our data. With these data issues, other standard ML methods obtain better results. Our conjecture is consistent with the work of Shaikhina and Khovanova [51], who showed that one of the main problems of medical datasets is the scarcity of data, and this negatively impacts the performance of ML models based on neural networks. When the dataset is small, neural networks are influenced by many factors such as the initialisation step, the train-test splitting procedure, and the order of the data used in the mini-batched optimisation procedure. These elements form a random bias that is usually irrelevant if the datasets are large, but that could be a crucial issue when they are small like ours. Several other works in the literature report that, for classification tasks in the medical field with relatively small datasets, neural networks tend to perform worse than ensembles of decision trees [52], [53].

After the hyperparameter tuning and the Bayesian comparison, we obtain the best performing models according to a sound statistical measure. As we explained in Section VI, we can use these models to extract the most important features and predictive rules. Following the approach described in Section VI-A, we observed that the most important feature in our models is the age of the patient, which obtains the highest value for each dataset. On the contrary, the patient's sex is not as important as one could have expected, since, according to

our datasets, this feature often is not in the ten most important features.

In Figure 6 we show the average Shapley values for the most important features based on the lab tests considered. We can observe that neutrophils and lymphocytes are very relevant indicators of the status of the disease; they are involved in four different features: the ratio between them (average Shapley value 0.97), which is the most important lab-test feature, the percentage of lymphocytes (average Shapley value 0.83), and their absolute values (average Shapley values 0.22 and 0.23, respectively). Furthermore, the learnt models also take into account how the percentage of lymphocytes changes over time, as indicated by the feature evaluating the ''long trend'' of the percentage of lymphocytes (average Shapley value 0.26).

The features of LDH and PCR are also valuable for monitoring an infection, and according to our datasets, these laboratory tests are performed almost every day. As shown in recent studies [54], D-Dimer is an important feature to assess COVID-19 severity, and our analysis confirms that it is one of the most relevant laboratory tests. However, in our datasets D-Dimer is not measured using the same frequency as PCR or LDH, and therefore we can have ''outdated'' or even missing values for this lab test. Therefore, our learnt models cannot rely much on the D-Dimer feature, which obtains an average Shapley 0.39. Slight importance is given to the swab test, while Ferritin and Troponin-T are not very relevant, possibly because according to our datasets, they are performed with a very low frequency (once a week or even less).

Regarding the Chest X-Ray Score (RX), despite its effectiveness for assessing the severity of the pulmonary conditions [46], the SHAP analysis shows that our models do not consider this information very useful, given the information provided by the other features. In our opinion, this is mainly due to the fact that, in the considered datasets, these exams were conducted too rarely, leading to several missing or outdated values.

We discussed the most important features that we identified with the medical staff of the SCB hospital that provided the data for our datasets. The validity of this information was also confirmed from a clinical point of view. Each of the features that we found to be the most relevant for our prediction models is significant in the clinical practice of COVID-19, and in particular PCR, LDH, D-Dimer, and the scarcity of lymphocytes. However, while in the hospital the lymphocytes are considered in their absolute value, our algorithms tend to consider their percentage or the ratio between them and the neutrophils. The latter is consistent with the clinical studies in [55], which show the remarkable effectiveness of using this ratio to evaluate the severity and mortality of COVID-19 patients.

We created the Simplified Snapshot Datasets considering the seven most important lab tests, the patient's age and sex and the death rate. As explained in Section VI-B, from each of these datasets we can build a Decison Tree from which

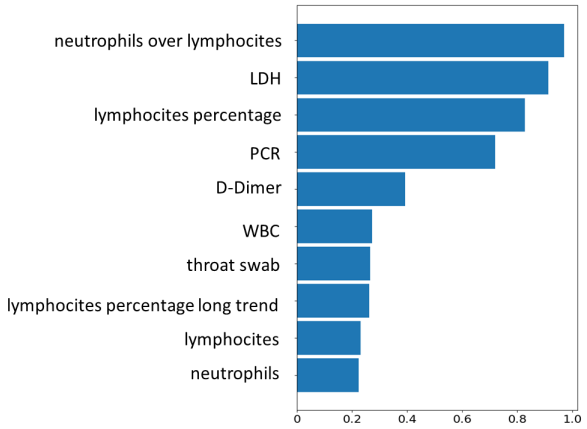

FIGURE 6. Most relevant lab-test features according to our SHAP-based analysis.

we can derive useful prognostic knowledge in the form of predictive rules. For our experimental analysis, we selected the rules discovered with the support of at least 50 training instances, training accuracy above 75%, and agreement above 85%. Then we applied them to the test sets and evaluated them in terms of accuracy and agreement with the test set. The extracted rules are presented in the left-hand side of Table 5, where we also report their (high) accuracy and agreement that will be described in Section VIII-B. (The meaning of the different colours used for the rule conditions is described in the caption of the table.)

Comparing our results with the medical literature, in many clinical studies examined in [55], such as the one presented in [56], the reported cut-off values are remarkably similar to those we (automatically) found for our rules. For example, considering the cohorts of patients analysed in [55] which are mostly similar to ours in terms of age, the reported cut-off values are between 5 and 6, which are consistent with the values 5.7 and 5.2 in the rules discovered from the 2days and 6days datasets, respectively. Furthermore, values around 3 (such as 2.7 in our rules for the 4days dataset) are used in several studies to discriminate the least severe patients. After 8 days, the threshold for neutrophils over lymphocytes in our rules is 9.1; according to clinicians, this is consistent with the administration of corticosteroids in severe patients for the treatment of respiratory failures. In fact, this treatment has the tendency to diminish lymphocytes.

In our opinion, the most interesting rules are those that in the left hand side of the rule include a positive feature condition (colored green) and a very negative feature condition (colored red), as well as those with some worrying conditions for the patient (in light green and orange). For example, the second rule for the 8days dataset regards patients with a very dangerous value of the neutrophils/lymphocytes ratio (higher than 9.1), but who are relatively young (less than 64 years old) and with a PCR value that is not considered dangerous. Despite this combination of conditions, the patient's

TABLE 5. Prognostic rules extracted from the simplified datasets, i.e. from data about the patient features that are most important (according to our SHAP-based analysis), and that are available at the 2nd/4th/6th/8th/10th day and before the last day of hospitalisation (End). Each rule states that, if the rule condition is satisfied by the patient (column 'Predition-rule condition"), then a certain prognosis is expected (column "Predicted Prognosis"); Each rule condition is the logic product ($\wedge$) of certain constraints on some feature (such as age $\leq$ 69). Column "Support" indicates the percentage and (in brackets) the number of patients who match the rule condition in the corresponding dataset. Columns "Acc." and "Agree" indicate the accuracy and the agreement rates of the rule w.r.t. the test set and our prediction models, respectively. The colours of the rule conditions have the following meaning: green for very positive conditions in terms of survival and which regards almost only alive patients; light green for conditions who regard a majority of alive patients but also some deceased; orange for negative conditions who regard also some alive patients; red for strongly negative conditions which regard a vast majority of deceased patients. The black conditions are not particularly significant. *neu*/*lym* is the ratio between neutrophils and lymphocytes; %*lym* is the percentage of lymphocytes.

Day	Prediction-rule condition	Predicted prognosis	Support	Acc.	Agree
2nd	(age $\leq$ 69) $\wedge$ (LDH $\leq$ 502) $\wedge$ (neu/lym $\leq$ 5.7)	survival	26% (490)	97%	100%
	(age $\leq$ 69)$\wedge$(LDH $\leq$ 502)$\wedge$(neu/lym $>$ 5.7)$\wedge$(WBC $>$ 2.89)	survival	11% (210)	94%	87%
	(age $>$ 69) $\wedge$ (LDH $>$ 402)	decease	11% (203)	57%	100%
4th	(%lym $\leq$ 11.8) $\wedge$ (age $\leq$ 64) $\wedge$ (LDH $\leq$ 559)	survival	10% (162)	100%	96%
	(%lym $\leq$ 11.8) $\wedge$ (age $>$ 64)	decease	24% (409)	47%	94%
	(%lym $>$ 11.8) $\wedge$ (neu/lym $\leq$ 2.7)	survival	22% (366)	97%	100%
	(%lym $>$ 11.8) $\wedge$ (neu/lym $>$ 2.7) $\wedge$ (LDH $\leq$ 392) $\wedge$ (age $\leq$ 71)	survival	16% (269)	97%	95%
6th	(LDH $\leq$ 373) $\wedge$ (neu/lym $\leq$ 5.2) $\wedge$ (age $\leq$ 61)	survival	14% (213)	100%	100%
	(LDH $\leq$ 301) $\wedge$ (neu/lym $\leq$ 5.2) $\wedge$ (age $>$ 61)	survival	18% (262)	100%	98%
	(LDH $\leq$ 373) $\wedge$ (neu/lym $>$ 5.2) $\wedge$ (D_dimer $\leq$ 1631) $\wedge$ (PCR $\leq$ 103) $\wedge$ (death_rate $>$ 0.2)	survival	4% (65)	86%	79%
8th	(neu/lym $\leq$ 9.1) $\wedge$ (age $\leq$ 72)	survival	39% (472)	98%	99%
	(neu/lym $>$ 9.1) $\wedge$ (age $\leq$ 64) $\wedge$ (PCR $\leq$ 104)	survival	13% (157)	91%	79%
	(neu/lym $>$ 9.1) $\wedge$ (age $>$ 64) $\wedge$ (LDH $>$ 421)	decease	8% (102)	71%	100%
10th	(%lym $\leq$ 12.1) $\wedge$ (age $\leq$ 65) $\wedge$ (D_dimer $\leq$ 449)	survival	7% (65)	100%	100%
	(%lym $>$ 12.1) $\wedge$ (PCR $\leq$ 267) $\wedge$ (LDH $>$ 107) $\wedge$ (neu/lym $<$ 5.2) $\wedge$ (swab $\neq$ Positive)	survival	21% (214)	100%	97%
	(%lym $\leq$ 12.1) $\wedge$ (age $>$ 65) $\wedge$ (neu/lym $>$ 13.4)	decease	14% (140)	78%	95%
	(%lym $>$ 12.1) $\wedge$ (PCR $\leq$ 267) $\wedge$ (LDH $>$ 107) $\wedge$ (swab = Positive) $\wedge$ (age $\leq$ 72)	survival	11% (111)	95%	100%
End	(%lym $\leq$ 12.5) $\wedge$ (age $\leq$ 64) $\wedge$ (LDH $<$ 503) $\wedge$ (death_rate $\leq$ 0.3) $\wedge$ (PCR $\leq$ 36)	survival	6% (114)	100%	100%
	(%lym $\leq$ 12.5) $\wedge$ (age $>$ 64) $\wedge$ (PCR $>$ 42)	decease	13% (246)	72%	91%
	(%lym $>$ 12.5) $\wedge$ (PCR $\leq$ 55)	survival	5% (98)	95%	92%
	(%lym $>$ 12.5) $\wedge$ (PCR $>$ 55) $\wedge$ (LDH $\leq$ 327)	survival	51% (953)	96%	92%

prognosis is "survival" with high accuracy. Another interesting rule in Table 5 is the third for the 6days dataset, which predicts survival for patients with potentially alarming values (neutrophils/lymphocytes > 5.2 and possibly high values of D-Dimer and LDH), and when the hospital was in a critical emergency phase (death rate > 0.2). Although the support for this rule is lower than for other rules, we think it can be helpful to consider it, especially in emergency situations.

A remarkable correlation between lab test findings can be seen in the first two rules for the 6days dataset. In these rules, for those patients who have a neutrophils/lymphocytes ratio below 5.2, the range of LDH values for predicting that a patient will be released alive differs. For the older patients ($age > 61$), LDH must not exceed 301, for the younger patients, *LDH* must not exceed 373. Given that according to UpToDate (a reliable online resource for medical knowledge used by clinicians),[6] severe conditions of COVID-19 are associated with LDH values higher than 245, the first two rules of the 6days dataset (which, combined, refer to more than 30% of our patients) provide interesting information about survival conditions of severe patients. Another useful example is the first rule for the End dataset. This rule highlights that relatively young patients ($age < 64$) can survive even with a very high value of LDH and a low percentage of lymphocytes, if the PCR is not particularly high (according to UpToDate, patients with severe COVID-19 are associated with PCR higher than 100 while the normal range is between 0 and 8).

The severity of the conditions in our rules (highlighted using different colours) is evaluated using the distribution of lab test findings in our cohort of patients. However, a similar evaluation can be obtained using the distribution derived from other cohorts, such as the ones described in [57] and [58]. Although we cannot evaluate the support and accuracy of our rules for these cohorts (because their data are not available), we think that this similarity provides promising insights about the generality of our rules. As a future work, we intend to evaluate their validity in another cohort of patients obtained from another hospital in Norther Italy, *ASST Papa Giovanni XXIII* (Bergamo).

B. TESTING OF THE LEARNED MODELS, RULES GENERATION AND EXPLANATION

As explained in Section III, 80% of the 2015 hospitalised patients considered in our study were used for the model training the hyperparameters tuning, which led to a set of trained learning models, one for each snapshot dataset. The remaining 20% of the patients were used as a test set to evaluate the performance of such models and the extracted prognosis-prediction rules.

1) PERFORMANCE OF THE LEARNED MODELS

The performance of the learned models was evaluated using the test set in terms of the F-2 score and the ROC-AUC score. The second metric is defined as the area under the Receiver Operating Characteristic (ROC) curve, which plots the true positive rate against the false positive rate, and it also takes into account the probability that the predictive system produces false positives (i.e., false alarms). This metric is a standard method for evaluating medical tests and predictive models [59], [60].

[6]https://www.uptodate.com/landing/covid19

TABLE 6. Predictive performance in terms of F-2 and ROC-AUC scores, considering all instances in the test set (columns F-2 and ROC) and omitting the instances classified as uncertain (columns F-2U and ROC-U). For each dataset, the column "Model" indicates the method chosen for generating the predictions i.e., is the best performing one for that dataset (see Table 4). The percentages of instances that the system classifies uncertain are in column "% Unc."; ET stands for Extra Trees, RF for Random Forest, XGB for XGBoost, and LGBM for light gradient boosting machine.

Dataset	Model	F-2	ROC	F-2U	ROC-U	% Unc
2days	XGB	67.8	75.7	74.9	80.9	19.4
4days	LGBM	70.7	81.9	74.4	84.4	16.8
6days	XGB	67.1	80.7	73.1	84.6	14.0
8days	XGB	76.5	86.3	78.8	87.9	8.1
10days	XGB	74.2	84.4	80.0	88.9	8.8
End	ET	83.8	89.6	91.8	94.9	23.2

Overall, our experimental analysis shows that models based on ensembles of decision trees perform better than the models based on neural networks. The training phase for the former models is also relatively fast: including the hyperparameter tuning over the 4096 random configurations considered and the optimisation of the uncertainty threshold, for every dataset at the considered hospitalisation days, the overall training time was between 30 and 40 minutes. Therefore, we can build the four most promising models generated by Random Forests, ExtraTrees, Light Gradient Boosting, and eXtreme Gradient Boosting in about two hours, and then select the best performing model among them. It is also worth noting that in our system, the models for predicting the patient's prognosis on different days are completely independent from each other, and so we can consider the prediction tasks on different days as different tasks.

Table 6 shows the performance of our system on each considered day (dataset). In terms of the F-2 score, the system obtains good results even during the early stages of hospitalisation, with scores 67.8 on the 2nd day of hospitalisation (considering only the first exams available) and 70.7 on the 4th day. On the 6th day, in terms of the F-2 score, the performance is inferior to what one could expect. This is due to the fact that in the 6days dataset the overall number of patients decreases by almost 20%, since some of them are deceased or have been released, and there is no significant information on the remaining patients during the previous days. In terms of ROC-AUC, the observed performances are generally good also for the 6th day. After a week, the patient's conditions are generally clearer, with more lab tests performed and more information about the progression of the disease through our trend features. This leads to a significant improvement in terms of the F-2 score, with a maximum of 76.5 on the 8th day, as well as in terms of the ROC-AUC with

values between 84 and 86 for both the 8th and 10th days. Finally, the performance for the End dataset is particularly good in terms of both the F-2 score (83.8) and the ROC-AUC (almost 90).

In all cases, extending the models with the use of the threshold computed by algorithm FindUncertaintyThreshold increases the performance in terms of F-2 and ROC-AUC scores. In particular, for the most problematic case, which is for the 6days dataset, we obtain a significant improvement of more than 6 points in terms of F-2, and almost 4 points in terms of ROC-AUC. It is important to note that both F-2U and ROC-U scores improve over time, confirming the intuition that, with days passing, the richer available data allow the learner to better evaluate the patient's condition.

While the threshold value under which the system labels an instance (patient prognosis) as uncertain is derived at training time, imposing a maximum percentage of uncertain samples (we used 25%), there is no formal guarantee that this percentage limit is satisfied for the test set. However, we observed that in most cases the percentage of uncertain test instances (indicated with "% Unc" in Table 6) is significantly below the limit imposed during training. Particularly interesting is the result for the 10days dataset, where we observe an improvement of more than 4 points in terms of ROC-AUC, and only 8.8% of the test instances are labelled as uncertain. The highest uncertainty is for the 2days dataset (19.4%), which considers less data for each patient, and for the End dataset (23.2%).

Figure 7 shows the confusion matrices for the test sets generated by our predictive models. The lowest performance is obtained at the 4th day, i.e., with very little information about the patient, where only nine patients in the test set are false negatives using the model without uncertainty threshold (see the second "complete" matrix in Figure 7). False negatives tend to decrease with the duration of hospitalisation, and they consistently decrease (for every considered dataset) if we use the uncertainty threshold ("No Unc" matrices in Figure 7).

On the other hand, we observe a high number of false positives (false alarms), especially on the second day of hospitalisation (more than 100 incorrect classifications). This is mainly due to the difficulty in evaluating the prognosis of a patient given only the first exams and no information on how the disease is progressing. Therefore, it is not surprising that our algorithm for dealing with uncertainty labels 19.4% of patients as uncertain, which has the consequence of avoiding 41 false positives (identified as uncertain cases).

False positives decrease markedly with hospitalisation time, with 23 errors at the 8th day and 16 errors at the 10th day. Furthermore, as also observed for false negatives, false positives decrease consistently when using the computed uncertainty threshold, reaching only 9 false positives for the 10days dataset.

Finally, note that all these results are obtained after addressing the class imbalance problem as described in Section IV. Without such a balance, the best performance achieved by our models was about 30% lower.

2) EVALUATION OF THE RULES GENERATION AND EXPLANATION

Tables 5 and 7 show the rules extacted as described in Section VI-B, their support over the entire dataset, their accuracy, and their agreement. In general, the agreement between the rules and the best performing model is often very high, and never lower than 79%. For each dataset except 2days and 6days, the proposed approach is capable of providing an explanation to more than 50% of all test instances in the dataset. While for the dataset End we have a remarkable result of 77.9% test instances explained, for the dataset of the 6th day we have a much lower percentage of explained instances (40.3%); notice that this is also the dataset with the worst performance in mortality prediction (see Table 6 and Section VIII-B). On the other hand, for the explained instances of dataset 4days, on average the selected rules have a high accuracy (81.9).

Moreover, if we consider applying these rules only to the test instances that are predicted alive (which are the most critical predictions if we need to minimise false negatives), both the percentage of explained instances and the accuracy of the rules significantly increases (see the 3rd and 5th columns of Table 7).

The average accuracy of the rules is generally very good. Only two rules on the first days of hospitalisation have inadequate accuracy; this is when it can be more difficult to predict the prognosis, as there are less available data on the patient at the first days. In particular, for the 4th day, the relatively low average accuracy of 81.9% is due to a single rule that predicts the *dead* class if the following condition holds:

$$(lymphocytes\_percentage < 11.8) \wedge (age > 64).$$

Given that the elderly patients are more at risk, and that the normal range of the percentage of lymphocytes for an adult is between 20 and 45, this rule captures potential critical patients. However, its condition is not sufficient to properly identify an unfavorable prognosis (decease), which can lead to many false positive errors. In fact, the accuracy of this rule is slightly below 50%. This is consistent with the behaviour of the prediction model using an ensemble of decision trees for the 4th day, which also has a significant number of false positive errors, as shown in Figure 7.[7] On the other hand, for the datasets on the latter days, the average accuracy is always higher than 90%.

IX. RELATED WORK

A. PROGNOSIS PREDICTION

A first study on prognosis prediction for COVID-19 is presented in [6]. This work uses data on lab tests, symptoms,

[7] As described and discussed above, our system attempts at optimising false negatives more than false alarms, since the first type of errors are more important for the kind of medical application we are considering.

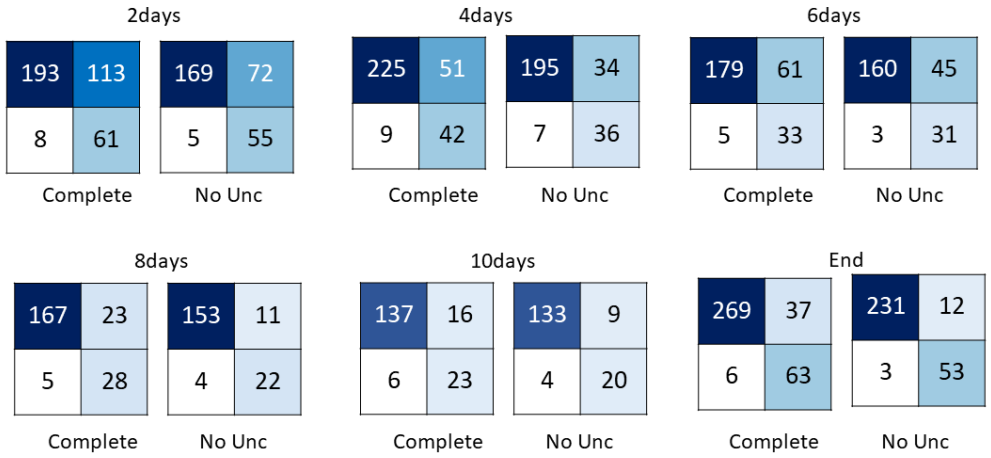

FIGURE 7. Confusion matrices at different hospitalisation days for the *dead-alive* predictions of (i) all test patients ("Complete" matrices) and (ii) only the test patients that are not classified as uncertain/unpredicatble ("No Unc" matrices). For each matrix of 4 numbers, on the main diagonal we have the correct predictions (*alive* class on the top-left corner and *dead* class on the bottom-right corner); on the anti-diagonal, we have the incorrect predictions (false positives and on the top-right corner and the false negatives on the bottom-left corner).

TABLE 7. Summary of the overall performance of the extracted rules in each dataset in terms of: percentage of *test* instances to which the rules can be applied (have their conditions satisfied) to explain the relative predictions (2nd column for all instances; 3rd column for only the instances predicted of class *alive*), weighted average accuracy of the class predicted by the rules (4th column for all instances; 5th column for only the instances predicted of class *alive*), and weighted average agreement (considering all instances) of the rules with the models based on ensembles of decision trees (6th column).

Dataset	Explained (all)	Explained (alive)	Accuracy (all)	Accuracy (alive)	Agreement
2days	47.7%	65.0%	87.2	97.8	96.5
4days	69.4%	75.7%	81.9	98.1	97.8
6days	40.3%	54.3%	98.2	99.1	96.4
8days	60.1%	64.4%	93.2	97.3	94.8
10days	56.6%	58.4%	93.2	96.2	97.8
End	77.9%	87.6%	94.2	97.1	96.6

and some epidemiological information to predict if a patient will die in the next 10 days The authors use an XGBoost algorithm [61] to build a prediction model considering a cohort of only 375 patients (2/3 of these for training and 1/3 for validation) from the Wuhan region, China. Then they select the most important features for the learned model, and train a single small decision tree that is intended to provide an operable decision schema to quickly predict patients at the highest risk. Our work significantly differs in many aspects: we study the prediction of the patient's mortality without a restricted temporal window (10 days in [6]); our models are learnt and tested using much larger datasets (overall 2015 patients); we use different information (features) about patients regarding lab test exams and lung damage information from X-ray exams; our models are trained considering the importance of minimising false negatives; we study the most important features for prediction and model interpretation using the standard approach SHAP [25], instead of a custom algorithm; the techniques in [6] do not deal with uncertain predictions, while we do it through a new method; we propose general methods to extract from data knowledge in the form of prediction rules, and to exploit them for providing explanation to the predictions made by a complex ML model. The extracted rules for the prognosis of COVID-19 are also analysed in depth.

The work by Cornelius et al. [62] proposes a Clustered Random Forest to predict the mortality of patients with COVID-19, and to examine the hidden heterogeneity of the patient frailty. They considered 10 features mainly related to demographic information for 10, 000 individuals, with 5% of the individuals who died, randomly selected from the CDC's case surveillance data from January 2020, to March 2021. They compare Clustered Random Forests and AdaBoost for mortality prediction, with the best model achieving 95% accuracy, but only 47% of F1 score. They also use Neural Networks and K-Means clustering for follow-up analysis to provide information on the type and magnitude of mortality risks associated with COVID-19. Despite the high accuracy, the authors report a low percentage of the F1 score and, moreover, they do not consider using more proper metrics for evaluation, such as the F2 score and ROC-AUC scores. In addition, they do not consider more advanced methods, such as boosting algorithms or neural networks, and an attempt to model explanation is missing.

The study in [63] proposes machine learning methods to predict COVID-19 mortality using data from 370 patients

(1766 datapoints). They use mainly blood tests as input features, such as Neutrophils, Lymphocytes, Lactate dehydrogenase (LDH), high-sensitivity C-reactive Protein (hs-CRP), and Age. The best-performing method, which uses XGBoost for feature importance and neural network for classification, predicts with an accuracy of 90% as early as 16 days before the prognosis. On the contrary, in our approach we predict prognosis at different days of hospitalisation starting from the second day. The reduced number of patients data, the small number of features considered, no information on the trend of the features over time, and the lack of explainability of the best model are some of the main differences from our work.

The work in [8] uses information on 1500 patients from the Ayatollah Taleghani Hospital registry (Iran) and several ML algorithms to predict COVID-19 mortality. The 38 features considered in [8] include risk factors (comorbidities), clinical manifestations, and oxygen therapy information, which can be very helpful features for mortality prediction. Although the data for such features were not available for our study, our approach reaches high predictive performance even without using them. The work in [8] treats unbalanced class labels (more alive than deceased) using the SMOTE technique, and proposes a best model based on the Random Forest algorithm, but without a method for automatically extracting prognostic rules and using them for the explanation, as we have proposed and experimentally analysed. Moreover, this approach lacks an evaluation of the uncertainty associated with the prediction made by the ML model.

The work in [9] proposes a prediction model of in-hospital mortality for COVID-19 patients treated with steroids and remdesivir. They considered 1571 patients (21% deceased) from the Mount Sinai Health System (New York City) hospitalised in the period from March 2020 to March 2021. Similarly to our work, this work proposes to use SHAP and a boosting model (LGBM) for the feature importance detection and the prognosis prediction, respectively. However, our approach appears to obtain good performance with less information about the hospitalised patients (especially their comorbidities and treatments), and furthermore it includes modules for the automatic generation of prognostic rules and their use to explain the prediction made by the learned models, that are not present in [9]. Another important difference is that our approach is able to identify those patients for whom the prediction made by the ML algorithm is unreliable, labeling them as uncertain.

In a short paper that we presented in 2020 at a workshop [19], we gave a first description of our work and preliminary results on predicting the patient's prognosis at different times during patient hospitalisation. Here, we substantially extend and revise that preliminary work with new results and several improvements in key aspects. In particular, additional learning algorithms are considered, better prediction results are obtained, and data quality issues such as trends, missing values, and concept drift are better analysed and handled. Moreover, in this paper we also analyse the most important features of the prediction task and address the problem of interpreting the predictions through a set of simple rules extracted from a simplified version of our model, which can be seen as a ''surrogate model'' [64].

Several AI and machine learning techniques have been developed for estimating the prognosis of diseases different from COVID-19. The survey in [65] presents a review of statistical and ML systems to predict the patient's prognosis, the need of beds in intensive care units [66], or the duration of patient hospitalisation. An overview of the issues and challenges of applying ML in a critical care context is available in [67]. This work stresses the need to deal with corrupted data, such as missing values, imprecision, and errors, which can increase the complexity of the prediction tasks. Lab test results and their variation over time are the main focus of the work presented in [68], where the authors describe a system that processes these data to generate an alarm predicting if a patient will have a circulatory failure 2 hours in advance.

B. UNCERTAINTY, EXPLANATION AND RULE EXTRACTION

The most important technique for making a machine learning model a probabilistic classifier [41], [42] is *calibration* [11], [43], [44]. With probabilistic classifiers, it is possible to discharge the low probability predictions and therefore to remove possible mistakes [13]. In our context, performing calibration onto our models lead to radically worse results, losing from 20 to 30 percent in terms of F-2 score without having a good *Brier score* [69], which is the common metric to evaluate calibration quality. Furthermore, the calibration curves are jagged and rough, showing that the calibration mechanism failed due to overfitting and insufficient samples.[8]

Another approach to obtain reliable probabilities as output predictions of a machine learning model consists in integrating probabilistic properties directly into the machine learning algorithm. NGBoost [12] is a recent example of this approach, which combines gradient boosting methods and probabilistic prediction, using Natural Gradient [70] to correct the training dynamics of the multiparameter boosting method, instead of the standard gradient approach. In [12], the authors of NGBoost show better performance compared to existing methods for probabilistic prediction (using mainly regression tasks), while maintaining flexibility and scalability. In our experiments, we used NGBoost with the settings supported for binary classification tasks (''Bernoulli'' as the classification distribution and ''LogScore'' as the prediction score).[9] The results we obtained using NGBoost in terms of the F-2 score are significantly worse than the results obtained using the other methods; we observed a performance drop of

[8]The calibrator was implemented using the Scikit-Learn package [50] which offers a probability calibration with either Isotonic [43], [44] or Logistic [11] regression in a cross validation fashion. We chose logistic regression due to the low number of available samples, and the same number of cross-validation folds as for the training (10), to reduce overfitting as suggested in [15]. We also performed calibration with other settings of the hyperparameters obtaining worse results.

[9]The official implementation of NGBoost is available at: http://proceedings.mlr.press/v119/duan20a.html

about 20%. We speculate that the complexity of the task, the class imbalance, and the presence of many missing values in our datasets compromise the learning of the correct probability distribution.

Recenlty, Combi et al. proposed a novel conceptual framework providing a foundational definition of explainable AI (XAI) in medicine in which the explainability of a system is definable as the intersection of four characteristics of the system: interpretability, understandability, usability, and usefulness [18]. In relation to this definition, our method supports explainability of our prognosis prediction models by improving their interpretability and uselfulness. In addition, the high performance of our models represents another element that fosters the usefulness of the system, in accordance with the discussion in [18].

While there are several approaches for explaining a single prediction (''local explanation'') such as [25] and [71], in our work we also focus on extracting knowledge in the form of rules that can be studied and applied by physicians. CHIRPS [72] and Action Rules [73] are typical methods for rule mining, but, applied to our scenario, they were unable to provide rules with sufficient accuracy and support. Similarly, TreeASP [74] was only partially applicable in our context, given that it extracts rules only from LightGBM and Random Forests, excluding XGBoost which provides the overall best results in our context. Moreover, for the only model into which TreeASP was applicabile (the LightGBM classifier in the 4days model) it didn't provide meaningful results. In particular, the extracted rules with a support over more than 50 patients had an average accuracy of 30%. Given the complexity of our data in terms of missing values, errors, and outdated values, approaches that directly mine rules without any relation to a pre-trained classifier like the Bayesian Rule List [75] are not suitable for our context.

X. CONCLUSIONS AND FUTURE WORK

We have proposed a collection of methods and a concrete system for the estimation of the prognosis of hospitalised patients in emergency situations, such as during the first wave of the COVID-19 pandemic in Northern Italy, in which the scarce knowledge of the disease and the limited availability of clinical data make the estimation of the prognosis of patients much harder. We focused the prognosis estimation on the worst-case scenario (decease), and we addressed it as a binary classification task. Although our approach, methods and techniques are proposed and evaluated in the context of COVID-19, they could be useful also for other diseases (or possible future harmful variants of COVID-19), especially when there is still limited clinical and prognostic knowledge.

Starting from raw data, our methods (i) generate a sequence of snapshot datasets indexed by the increasing length of stays of the patients, (ii) identify a best-performing ML model for each of the snapshot datasets, (iii) produce a set of multi-variable prognostic rules, and (iv) exploit such rules for providing user-interpretable explanations (in simple clinical terms) to the prognosis prediction of the ML models, as well as to suggest possible new clinical knowledge on the disease that is discovered from the patients' data. Furthermore, the ML models are optimised by a new technique for automatically identifying, at training time, an uncertain threshold that helps avoid potential prediction errors and improve the overall performance of the system.

All our techniques are implemented in a system that was experimentally analysed considering several ML algorithms. Overall, we observed that the predictive performance of our models is high, especially in terms of ROC-AUC scores and number of false negatives (patients erroneously predicted survivors), which are very few. This gives a predictive test for patient survival that has very good specificity, in particular when the system can exclude patients who are below the computed uncertainty threshold. On the other hand, in terms of false positives, there is still room for significant improvement. We are confident that the availability of additional information about patients, such as comorbidities or clinical treatments, can help improve performance, reducing the number of false positives and (very few) false negatives. The best-performing models were built using the ensemble methods XGBoost and LightGBM, while our experimental analysis shows that feed-forward neural networks do not reach the same level of performance.

We are currently studying the use of Recurrent Neural Networks (RNN) in which we consider the sequence of lab test findings, instead of the snapshot of the patient's health conditions defined by the most recent available lab test findings and their trend. However, our first results with an RNN model are not promising [76], since it does not improve the performances of the best models that we have presented in this paper. This is probably due to the limited size of the dataset that can be detrimental to the application of deep learning techniques.

In future work, we intend to study to what extent the predictive performance that we obtained can be improved if additional relevant information about the patients is available, as well as to address other prediction tasks, such as the duration of hospitalisation or the need for specialised equipments in intensive-case units or other critical resources. Furthermore, we intend to extend and validate our system with additional data on patients with COVID-19 from at least another hospital in Northern Italy, such as the Papa Giovanni XXIII hospital in Bergamo, and make our datasets available.

Finally, given that the proposed approach can potentially be applied to any disease other than COVID-19, further studies can be conducted to evaluate the generalisability of our techniques using different datasets and populations.

APPENDIX A
DATA PREPROCESSING TECHNIQUES

In this appendix we describe the techniques we investigated for handling missing values and class imbalance in the datasets built from the available raw data.

A. HANDLING MISSING VALUES

After the generation of training and test sets for different days according to the methodology that we have described, an important issue to address is handling the missing values. One of the most commonly used techniques for doing this is *imputation* [48], [49]. We considered and experimentally compared four different types of imputers, of which the fourth one showed to be the most effective for our ML models:

- *Constant*: it replaces a missing value with a constant value, typically 0 or -1. Given that our features are only positive, we use -1 as an indicator of a missing value.
- *Mean*: given a feature, it replaces a missing value with the mean of the values of that feature. This strategy is fast and widespread, but can be affected by *outliers*, and could amplify the biases already present in the training distribution, creating a strong correlation among the mean of the features having a particular output value.
- *Median*: it is similar to the previous one except that it uses the median instead of the mean, mitigating the effect of outliers, but with the same bias issues as the previous one.
- *Iterative with Bayesian Ridge Regression* [26], [27], [28], [29]: this type of imputer is a multivariate one that estimates each feature from all the others. Given a set of column features F, at each step, a feature column $f_i \in F$ is designated as the output and the other feature columns $F' = F \setminus \{f_i\}$ as the input of a regressor model r_i; the regressor model is trained using F' as input and f_i as output.

B. ADDRESSING CLASS IMBALANCE

Another important issue in ML tasks is class imbalance. Imbalanced classes lead the learning method to prefer the majority class, which can decrease performance in terms of the F-2 score. We address this problem through *class weights*. This technique concerns the creation and the use of a weight for each class to give different importance to samples from different classes. The weight w_i for a class i is defined as follows:

$$w_i = \frac{M}{N \cdot m_i}$$

where M is the size of the dataset, N is the number of classes (in our case $N = 2$) and m_i is the number of samples of class i [31]. The weights are used in the error function computed for training and for evaluating the trained model/classifier. In gradient-based methods and artificial neural networks, weights are used in the loss function to increase error values for samples of classes that have fewer samples, and to decrease error values for samples of classes with many samples. During the creation of a Decision Tree, the weights are used to increase the impurity value of a tree node that misclassifies samples from a class with limited samples in the training set.

Other popular solutions for imbalanced data are *sub-sampling* and *super-sampling* methods. The former method removes samples from classes that have more samples than the class with the fewest samples. The latter method adds synthetic samples to classes that have fewer samples. Both methods are used to obtain the same number of samples in each class. Unlike these two methods, the class-weighting technique enables class balancing implicitly, improving performance without deleting samples or adding synthetic ones.

APPENDIX B
MACHINE LEARNING ALGORITHMS

In this appendix we briefly describe the machine learning algorithms that we used for building the prognosis prediction models.

A. SINGLE DECISION TREE

Decision Trees are one of the most popular learning methods to solve classification tasks [77]. In a decision tree, the root and each internal node provide a condition for splitting the training samples into two subsets depending on whether the condition holds for a sample or not. In our context, for each numerical feature f, a candidate splitting condition is $f \leq C$, where C is called *cut point*. The final splitting condition is chosen by finding f and C providing the best split according to an impurity measure such as Entropy index, Gini index, or Information Gain. A subset of samples at a tree node can be split again by additional feature conditions that form a new *internal node* or, if the algorithm cannot find a split that improves the current measure, it can become a *leaf node* labelled with a specific classification (prediction). In our application domain, the classification label is patient *alive* or *dead*.

Let us consider a decision tree with a leaf node l and a subset S of associated *training* samples (those patients that meet all conditions on the path of the tree from the root to the leaf node). A *test* instance X that reaches l from the root tree is classified (predicted) y with probability:

$$P(y|X) = \frac{TP}{TP + FP} \tag{1}$$

where TP (True Positives) is the number of training samples in S that have a class value y, and FP (False Positives) is the number of samples in S that do not have a class value y [78]. Given that in our task there are only two classes ($y = alive$ and $\overline{y} = dead$), we have $P(\overline{y}|X) = 1 - P(y|X)$. The result of a classification of the decision tree for X is the class value with the highest probability.

B. RANDOM FORESTS AND EXTRA TREES

Random Forests (RF) [79] is an ensemble learning method [80] that builds a number of decision trees at training time. To build each individual tree of the random forest, randomly chosen subsets (with replacement) of the data features and of the training samples are used. While in the standard implementation of random forests the final classification label is provided using the statistical mode of the class values predicted by each individual tree, in our implementation

TABLE 8. Hyperparameters and relative space of values used for building the Neural Network models.

Hyperparameter	Type	Interval
Scaler	Categorical	{Standard, MinMax, MaxAbs, Robust}
Batch Size	Categorical	{32, 40, 48, 56, 64}
Hidden units	Integer	[1, 1000]
Hidden layers	Integer	[1, 10]
Hidden activations	Categorical	{Tanh, ReLU [90], ELU [91], SELU [92], Swish [93]}
Dropout rate	Real	[0.2, 0.6]
Batch Normalisation	Categorical	{True, False}
Activity Regularisation	Categorical	{True, False}

TABLE 9. Hyperparameters and relative space of values used for building the models based on Decision Trees (Random Forest, Extra Trees, XGBoost, LightGBM).

Hyperparameter	Type	Interval	ML Model
N Estimators	Integer	randint(16, 160)	ET, RF, XGB, LGBM
Criterion	Categorical	{gini, entropy}	ET, RF
Max Depth	Integer	randint(2, 16)	ET, RF, XGB, LGBM
Max Features	Categorical	{None, sqrt, log2}	ET, RF
Class Weight	Categorical	{None, balanced, balanced_subsample}	ET, RF, LGBM
CCP Alpha	Real	uniform(0.0, 0.035)	ET, RF
Scale Pos Weight	Real	uniform(5.0, 3.0)	XGB
Max Delta Step	Real	uniform(0, 10)	XGB
Learning Rate	Real	loguniform(0.001, 0.5)	XGB, LGBM
Gamma	Real	uniform(0.0, 50)	XGB
Reg Lambda	Real	loguniform(0.01, 10.0)	XGB, LGBM
Col Sample By Tree	Real	uniform(0.2, 0.8)	XGB, LGBM
Col Sample By Level	Real	uniform(0.2, 0.8)	XGB
Col Sample By Node	Real	uniform(0.2, 0.8)	XGB
Subsample	Real	uniform(0.2, 0.8)	XGB, LGBM
Min Child Weight	Integer	randint(1, 5), loguniform(1e-5, 1e-1)	XGB, LGBM
Num leaves	Integer	randint(32, 4096)	LGBM
Min Split Gain	Real	uniform(0.0, 50)	LGBM
Min Child Samples	Integer	randint(1, 40)	LGBM
Reg Alpha	Real	loguniform(0.01, 10.0)	LGBM
Importance Type	Categorical	{split, gain}	LGBM

the probability of the classification output is obtained by averaging the probabilities provided by all trees. Hence, given a random forest with n decision trees, a class (prediction) value y is assigned to an instance X with the following probability:

$$P(y|X) = \frac{\sum_{i=1}^{n} P_i(y|X)}{n} \tag{2}$$

where $P_i(y|X)$ is the probability given by the i-th decision tree of the ensemble (Equation 1).

Extremely Randomised Trees (Extra Trees or ET) are another ensemble learning method based on decision trees [81]. The main differences between Extra Trees and Random Forests are:

- In the original description of Extra Trees [81], each tree is built using the entire training dataset. However, in most implementations of Extra Trees the decision trees are built using a random subset of the training data, as in Random Forests.
- In standard decision trees and Random Forests, the cut point is chosen by first computing the optimal cut point for each feature and then choosing the best feature for branching the tree; while in Extra Trees, we first randomly choose k features and then, for each chosen feature f, the algorithm randomly selects a cut point C_f in the range of possible values of f. This generates a set of k couples $\{(f_i, C_i) \mid i = 1, \ldots, k\}$. The algorithm then compares the splits generated by each couple (e.g., under the split test $f_i \leq C_i$) to select the best using a split quality measure such as the Gini Index or others.

The probability $P(y|X)$ that instance X is of class y is calculated as in Random Forest (Equation 2).

C. GRADIENT BOOSTING ALGORITHMS

A Gradient Boosting Method (GBM) incrementally assembles several *weak learners*, such as small decision trees, to produce a single predictive model. GBM uses a gradient descent procedure to assign a weight to each weak learner, whose value is related to the ability of the learner to reduce errors. XGBoost (eXtreme Gradient Boosting) [61] and LightGBM (Light Gradient Boosting Machines) [82] are two of the most popular GBM-based algorithms that use decision trees as weak learners in the default setting. An important aspect of XGBoost and LightGBM is how they control overfitting, which is a known issue in gradient boosting algorithms. XGBoost and LightGBM adopt a more regularised model formalisation, which allows these algorithms to obtain better results than GBM.

The main differences between XGBoost and LightGBM are related to the methods for building trees, error calculation, and feature selection. While XGBoost builds each weak learner according to the distribution of the overall data, LightGBM focusses on the samples that actually have more

TABLE 10. Feed-forward neural network architectures and hyperparameter settings found by the tuning process. "B.S." stands for Batch size. "Hidden L." stands for Hidden layers.

Dataset	Scaler	B.S.	Units per level	Hidden L.	Activations	Dropout
2days	MinMax	40	[547, 275, 184, 139]	4	Swish	[0.29, 0.22, 0.15, 0.07]
4days	Standard	40	[700, 351, 235, 177, 142, 119, 102]	7	SeLU	[0.28, 0.14, 0.10, 0.07, 0.06, 0.05, 0.04]
6days	MinMax	48	[669, 336]	2	SeLU	[0.22, 0.11]
8days	MinMax	40	[997]	1	Swish	[0.28]
10days	Standard	40	[698]	1	ReLU	[0.59]
End	MinMax	32	[406, 271, 137]	3	ELU	[0.23, 0.23, 0.23]

impact on the loss function. In addition, LightGBM adopts several approaches to reduce memory usage, such as grouping categorical features that are mutually exclusive to each other and replacing continuous values with discrete bins.

D. NEURAL NETWORKS

Feed-forward Neural Networks are one of the most common architectures of artificial neural networks. They consist of layers of neurons that form a directed and weighted acyclic graph [83]. In these machine learning models, all input features (which form the *input layer)* are connected to a first *hidden layer* composed of a series of neurons. Each neuron computes the weighted sum of the input and applies an *activation function* that provides the output of the neuron. For computing the weighted sum, a feed-forward neural network has two sets of parameters: the weight matrix (which is multiplied by the input) and the bias vector, which is summed after this multiplication. The output calculated by the hidden layer can be connected to another hidden layer, which performs the same operations using a different weight matrix and a different bias vector. The last layer of a feed-forward neural network, called *output layer*, provides the predicted class for the input training instance. The learning algorithm compares the predictions with the target values and evaluates the *loss function* of the model. During the training phase, the feed-forward neural networks learn the parameters of each layer by minimising the loss function through an optimisation algorithm that is usually *Back-propagation*. In our case, given that as output we want probability values for two possible classes, the output layer of the network is composed of two neurons with activation *softmax* [84].

Feed-forward neural networks are the most widely used ANN models to address learning tasks with structured data in the medical and biological fields [85], [86], especially for small datasets [51]. In fact, the use of more complex architectures (e Deep Convolutional Neural Networks, for instance) requires a substantially larger amount of training data to provide adequate performance [87], [88]. Moreover, given the novelty of our application context, it is difficult to find a pre-trained model that can be effectively exploited for transfer learning [89].

E. HYPERPARAMETERS OF THE MACHINE LEARNING ALGORITHMS

Tables 8 and 9 show the hyperparameters and relative space of values that were used to build the ML models considered in our experimental analysis. Figure 8 shows the (automatically) tuned values of the hyperparameters of the selected (best performing) ML model for each of our datasets. Table 10 reports the tuned values of the hyperparameters of the models based on Neural Networks; we remind that these models are not chosen by our model selection procedure because they perform worse than others, and here we include them just for the sake of completeness.

XGB Hyperparameter	2days	6days	8days	10days
Col Sample By Level	0.84	0.903	0.5	0.55
Col Sample By Node	0.974	0.846	0.436	0.808
Col Sample By Tree	0.419	0.806	0.761	0.336
Gamma	29	41.41	29.56	13.887
Learning Rate	0.426	0.086	0.03	0.032
Max Delta Step	0.487	0.38	2.78	0.691
Max Depth	15	2	14	7
Min Child Weight	4	1	4	4
N Estimators	143	205	150	142
Reg Lambda	0.078	0.01	0.515	0.023
Scale Pos Weight	6.178	7.278	7.788	7.589
Subsample	0.793	0.865	0.565	0.727

LGBM Hyperparameter	4days
Class Weight	balanced
Col Sample By Tree	0.988
Importance Type	split
Learning Rate	0.086
Max Depth	8
Min Child Samples	37
Min Child Weight	0.045
Min Split Gain	4
N Estimators	225
Num Leaves	1674
Reg Alpha	6.248
Reg Lambda	6.394
Subsample	0.631

ET Hyperparameter	End
CCP Alpha	0.00365
Class Weight	balanced
Criterion	entropy
Max Depth	10
Max Features	sqrt
N Estimators	88

FIGURE 8. Hyperparameter values of the selected ML models. The table at the top left reports the values of the hyperparameters found for the best XGBoost (XGB) models selected for the 2days, 6days, 8days and 10days datasets. The table at the top right reports the values of the hyperparameters found for the best LightGBM (LGBM) model selected for the 4days dataset. The table at the bottom reports the values of the hyperparameters found for the best ExtraTrees (ET) model selected for the End dataset.

ACKNOWLEDGMENT

The authors would like to thank M.D. Mara Rossi from the Department of General Medicine, ASST Spedali Civili di Brescia, for several very helpful discussions on the importance of the results in this article for the clinical practise of patients with COVID-19. They also like to thank Prof. Federico Cerutti for a fruitful discussion of our model interpretation techniques.

REFERENCES

[1] S. Mohanty, M. H. A. Rashid, M. Mridul, C. Mohanty, and S. Swayamsiddha, "Application of artificial intelligence in COVID-19 drug repurposing," *Diabetes Metabolic Syndrome, Clin. Res. Rev.*, vol. 14, no. 5, pp. 1027–1031, Sep. 2020.

[2] M. Jamshidi, A. Lalbakhsh, J. Talla, Z. Peroutka, F. Hadjilooei, P. Lalbakhsh, M. Jamshidi, L. L. Spada, M. Mirmozafari, M. Dehghani, A. Sabet, S. Roshani, S. Roshani, N. Bayat-Makou, B. Mohamadzade, Z. Malek, A. Jamshidi, S. Kiani, H. Hashemi-Dezaki, and W. Mohyuddin, "Artificial intelligence and COVID-19: Deep learning approaches for diagnosis and treatment," *IEEE Access*, vol. 8, pp. 109581–109595, 2020.

[3] J. Bullock, A. Luccioni, K. H. Pham, C. S. N. Lam, and M. Luengo-Oroz, "Mapping the landscape of artificial intelligence applications against COVID-19," *J. Artif. Intell. Res.*, vol. 69, pp. 807–845, Nov. 2020.

[4] M. Olivato, N. Rossetti, A. E. Gerevini, M. Chiari, L. Putelli, and I. Serina, "Machine learning models for predicting short-long length of stay of COVID-19 patients," *Proc. Comput. Sci.*, vol. 207, pp. 1232–1241, 2022.

[5] S. Debnath, D. P. Barnaby, K. Coppa, A. Makhnevich, E. J. Kim, S. Chatterjee, V. Tóth, T. J. Levy, M. D. Paradis, S. L. Cohen, J. S. Hirsch, T. P. Zanos, L. B. Becker, J. Cookingham, K. W. Davidson, A. J. Dominello, L. Falzon, T. McGinn, J. N. Mogavero, and G. A. Osorio, "Machine learning to assist clinical decision-making during the COVID-19 pandemic," *Bioelectronic Med.*, vol. 6, no. 1, pp. 1–8, Dec. 2020.

[6] L. Yan, H.-T. Zhang, J. Goncalves, Y. Xiao, M. Wang, Y. Guo, C. Sun, X. Tang, L. Jing, and M. Zhang, "An interpretable mortality prediction model for COVID-19 patients," *Nature Mach. Intell.*, vol. 2, pp. 283–288, May 2020.

[7] I. Girardi, P. Vagenas, D. Arcos-D, L. Bessa, A. Bu, L. Furlan, R. Furlan, M. Gatti, A. Giovannini, and E. Hoeven, "On the explainability of hospitalization prediction on a large COVID-19 patient dataset," in *Proc. AMIA Annu. Symp.* Bethesda, MD, USA: American Medical Informatics Association, 2021, p. 526.

[8] K. Moulaei, M. Shanbehzadeh, Z. Mohammadi-Taghiabad, and H. Kazemi-Arpanahi, "Comparing machine learning algorithms for predicting COVID-19 mortality," *BMC Med. Informat. Decis. Making*, vol. 22, no. 1, pp. 1–12, Jan. 2022.

[9] T. Kuno, Y. Sahashi, S. Kawahito, M. Takahashi, M. Iwagami, and N. N. Egorova, "Prediction of in-hospital mortality with machine learning for COVID-19 patients treated with steroid and remdesivir," *J. Med. Virology*, vol. 94, no. 3, pp. 958–964, Mar. 2022.

[10] H. Papadopoulos, A. Gammerman, and V. Vovk, "Confidence predictions for the diagnosis of acute abdominal pain," in *Artificial Intelligence Applications and Innovations III*, M. Iliadis, V. Tsoumakasis, and Bramer, Eds. Boston, MA, USA: Springer, 2009, pp. 175–184.

[11] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Adv. Large Margin Classifiers*, vol. 10, no. 3 pp. 61–74, Mar. 1999.

[12] T. Duan, A. Anand, D. Y. Ding, K. K. Thai, S. Basu, A. Ng, and A. Schuler, "NGBoost: Natural gradient boosting for probabilistic prediction," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 2690–2700.

[13] I. Cohen and M. Goldszmidt, "Properties and benefits of calibrated classifiers," in *Proc. Eur. Conf. Princ. Data Mining Knowl. Discovery*. Cham, Switzerland: Springer, 2004, pp. 125–136.

[14] A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning," in *Proc. 22nd Int. Conf. Mach. Learn.*, 2005, pp. 625–632.

[15] M. Kull, T. M. Silva Filho, and P. Flach, "Beyond sigmoids: How to obtain well-calibrated probabilities from binary classifiers with beta calibration," *Electron. J. Statist.*, vol. 11, no. 2, pp. 5052–5080, Jan. 2017.

[16] M. A. Ahmad, A. Teredesai, and C. Eckert, "Interpretable machine learning in healthcare," in *Proc. IEEE Int. Conf. Healthcare Informat. (ICHI)*, Jun. 2018, pp. 559–560.

[17] S. M. Lundberg, G. G. Erion, and S. Lee, "Consistent individualized feature attribution for tree ensembles," 2018, *arXiv:1802.03888*.

[18] C. Combi, B. Amico, R. Bellazzi, A. Holzinger, J. H. Moore, M. Zitnik, and J. H. Holmes, "A manifesto on explainability for artificial intelligence in medicine," *Artif. Intell. Med.*, vol. 133, Nov. 2022, Art. no. 102423.

[19] A. E. Gerevini, R. Maroldi, M. Olivato, L. Putelli, and I. Serina, "Prognosis prediction in COVID-19 patients from lab tests and X-ray data through randomized decision trees," in *Proc. 5th Int. Workshop Knowl. Discovery Healthcare Data Co-Located 24th Eur. Conf. Artif. Intell. (KDH@ECAI)*, Santiago de Compostela, Spain, K. Bach, R. C. Bunescu, C. Marling, and N. Wiratunga, Eds. vol. 2675, 2020, pp. 27–34.

[20] J. Gama, I. Žliobaitundefined, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surveys*, vol. 46, no. 4, pp. 1–37, Mar. 2014.

[21] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset Shift in Machine Learning*. Cambridge, MA, USA: MIT Press, 2009.

[22] S. Hasan and R. Padman, "Analyzing the effect of data quality on the accuracy of clinical decision support systems: A computer simulation approach," in *Proc. AMIA Annu. Symp.* Bethesda, MD, USA: American Medical Informatics Association, 2006, p. 324.

[23] A. S. Rakitianskaia and A. P. Engelbrecht, "Training feedforward neural networks with dynamic particle swarm optimisation," *Swarm Intell.*, vol. 6, no. 3, pp. 233–270, Sep. 2012.

[24] A. Benavoli, G. Corani, J. Demšar, and M. Zaffalon, "Time for a change: A tutorial for comparing multiple classifiers through Bayesian analysis," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 2653–2688, Jan. 2017.

[25] S. M. Lundberg and S. Lee, "A unified approach to interpreting model predictions," in *Proc. Adv. Neural Inf. Process. Syst. Annu. Conf. Neural Inf. Process. Syst.*, Long Beach, CA, USA, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds. 2017, pp. 4765–4774.

[26] S. F. Buck, "A method of estimation of missing values in multivariate data suitable for use with an electronic computer," *J. Roy. Stat. Society, Ser. B Methodological*, vol. 22, no. 2, pp. 302–306, Jul. 1960.

[27] S. V. Buuren and K. Groothuis-Oudshoorn, "Mice: Multivariate imputation by chained equations in R," *J. Stat. Softw.*, vol. 45, no. 3, pp. 1–68, 2011.

[28] D. J. C. MacKay, "Bayesian interpolation," *Neural Comput.*, vol. 4, no. 3, pp. 415–447, May 1992.

[29] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *J. Mach. Learn. Res.*, vol. 1, pp. 211–244, Jun. 2001.

[30] C. J. van Rijsbergen, *Information Retrieval*. Oxford, U.K.: Butterworth-Heinemann, 1979.

[31] G. King and L. Zeng, "Logistic regression in rare events data," *Political Anal.*, vol. 9, no. 2, pp. 137–163, 2001.

[32] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9260–9269.

[33] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.

[34] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 2623–2631.

[35] J. Pastor-Pellicer, F. Zamora-Martínez, S. España-Boquera, and M. J. Castro-Bleda, "F-measure as the error function to train neural networks," in *Proc. Int. Work-Conf. Artif. Neural Netw.* Cham, Switzerland: Springer, 2013, pp. 376–384.

[36] H. He and X. Sun, "F-score driven max margin neural network for named entity recognition in Chinese social media," in *Proc. 15th Conf. Eur. Chapter Assoc. Comput. Linguistics (EACL)*, vol. 2, M. Lapata, P. Blunsom, and A. Koller, Eds. Valencia, Spain: Association for Computational Linguistics, 2017, pp. 713–718.

[37] J. Kawahara and G. Hamarneh, "Fully convolutional networks to detect clinical dermoscopic features," in *Proc. Int. Skin Imag. Collaboration (ISIC) Challenge Int. Symp. Biomed. Imag. (ISBI)*, 2017, pp. 1–8. [Online]. Available: https://arxiv.org/abs/1703.04559

[38] C. Nadeau and Y. Bengio, "Inference for the generalization error," *Mach. Learn.*, vol. 52, no. 3, pp. 239–281, Sep. 2003.

[39] N. Brajer, B. Cozzi, M. Gao, M. Nichols, M. Revoir, S. Balu, J. Futoma, J. Bae, N. Setji, A. Hernandez, and M. Sendak, "Prospective and external evaluation of a machine learning model to predict in-hospital mortality of adults at time of admission," *JAMA Netw. Open*, vol. 3, no. 2, Feb. 2020, Art. no. e1920733.

[40] L. Meijerink, G. Cina, and M. Tonutti, "Uncertainty estimation for classification and risk prediction on medical tabular data," *Stat*, vol. 1050, p. 23, May 2020.

[41] A. P. Dawid, "The well-calibrated Bayesian," *J. Amer. Stat. Assoc.*, vol. 77, no. 379, pp. 605–610, Sep. 1982.

[42] M. H. DeGroot and S. E. Fienberg, "The comparison and evaluation of forecasters," *J. Roy. Stat. Soc., Ser. D Statistician*, vol. 32, nos. 1–2, pp. 12–22, 1983.

[43] B. Zadrozny and C. Elkan, "Obtaining calibrated probability estimates from decision trees and Naive Bayesian classifiers," in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 1, 2001, pp. 609–616.
[44] B. Zadrozny and C. Elkan, "Transforming classifier scores into accurate multiclass probability estimates," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2002, pp. 694–699.
[45] A. Rajkomar, J. Dean, and I. Kohane, "Machine learning in medicine," *New England J. Med.*, vol. 380, no. 14, pp. 1347–1358, 2019.
[46] A. Borghesi and R. Maroldi, "COVID-19 outbreak in Italy: Experimental chest X-ray scoring system for quantifying and monitoring disease progression," *La Radiologia Medica*, vol. 125, no. 5, pp. 509–513, May 2020.
[47] A. Signoroni, M. Savardi, S. Benini, N. Adami, R. Leonardi, P. Gibellini, F. Vaccher, M. Ravanelli, A. Borghesi, R. Maroldi, and D. Farina, "BS-Net: Learning COVID-19 pneumonia severity on a large chest X-ray dataset," *Med. Image Anal.*, vol. 71, Jul. 2021, Art. no. 102046.
[48] B. Efron, "Missing data, imputation, and the bootstrap," *J. Amer. Stat. Assoc.*, vol. 89, no. 426, pp. 463–475, Jun. 1994.
[49] J. A. C. Sterne, I. R. White, J. B. Carlin, M. Spratt, P. Royston, M. G. Kenward, A. M. Wood, and J. R. Carpenter, "Multiple imputation for missing data in epidemiological and clinical research: Potential and pitfalls," *BMJ*, vol. 338, no. 291, p. b2393, Sep. 2009.
[50] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12 no. 10, pp. 2825–2830, 2012.
[51] T. Shaikhina and N. A. Khovanova, "Handling limited datasets with neural networks in medical applications: A small-data approach," *Artif. Intell. Med.*, vol. 75, pp. 51–63, Jan. 2017.
[52] H. Y. Chang, C. K. Jung, J. I. Woo, S. Lee, J. Cho, S. W. Kim, and T.-Y. Kwak, "Artificial intelligence in pathology," *J. Pathol. Translational Med.*, vol. 53, no. 1, p. 1, 2019.
[53] Y. Kim, "Comparison of the decision tree, artificial neural network, and linear regression methods based on the number and types of independent variables and sample size," *Exp. Syst. Appl.*, vol. 34, no. 2, pp. 1227–1234, Feb. 2008.
[54] Y. Yao, J. Cao, Q. Wang, Q. Shi, K. Liu, Z. Luo, X. Chen, S. Chen, K. Yu, Z. Huang, and B. Hu, "D-dimer as a biomarker for disease severity and mortality in COVID-19 patients: A case control study," *J. Intensive Care*, vol. 8, no. 1, pp. 1–11, Dec. 2020.
[55] X. Li, C. Liu, Z. Mao, M. Xiao, L. Wang, S. Qi, and F. Zhou, "Predictive values of neutrophil-to-lymphocyte ratio on disease severity and mortality in COVID-19 patients: A systematic review and meta-analysis," *Crit. Care*, vol. 24, no. 1, pp. 1–10, Dec. 2020.
[56] F.-F. Chen, M. Zhong, Y. Liu, Y. Zhang, K. Zhang, D.-Z. Su, X. Meng, and Y. Zhang, "The characteristics and outcomes of 681 severe cases with COVID-19 in China," *J. Crit. Care*, vol. 60, pp. 32–37, Dec. 2020.
[57] X. Guan, B. Zhang, M. Fu, M. Li, X. Yuan, Y. Zhu, J. Peng, H. Guo, and Y. Lu, "Clinical and inflammatory features based machine learning model for fatal risk prediction of hospitalized COVID-19 patients: Results from a retrospective cohort study," *Ann. Med.*, vol. 53, no. 1, pp. 257–266, Jan. 2021.
[58] W. Xu, N.-N. Sun, H.-N. Gao, Z.-Y. Chen, Y. Yang, B. Ju, and L.-L. Tang, "Risk factors analysis of COVID-19 patients with ARDS and prediction based on machine learning," *Sci. Rep.*, vol. 11, no. 1, pp. 1–12, Feb. 2021.
[59] G. L. Grunkemeier and R. Jin, "Receiver operating characteristic curve analysis of clinical risk models," *Ann. Thoracic Surgery*, vol. 72, no. 2, pp. 323–326, Aug. 2001.
[60] K. Hajian-Tilaki, "Receiver operating characteristic (ROC) curve analysis for medical diagnostic test evaluation," *Caspian J. Internal Med.*, vol. 4, no. 2, p. 627, 2013.
[61] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.
[62] E. Cornelius, O. Akman, and D. Hrozencik, "COVID-19 mortality prediction using machine learning-integrated random forest algorithm under varying patient frailty," *Mathematics*, vol. 9, no. 17, p. 2043, Aug. 2021.
[63] A. Karthikeyan, A. Garg, P. K. Vinod, and U. D. Priyakumar, "Machine learning based clinical decision support system for early COVID-19 mortality prediction," *Frontiers Public Health*, vol. 9, May 2021, Art. no. 626697.
[64] C. Molnar. (2019). *Interpretable Machine Learning*. [Online]. Available: https://christophm.github.io/interpretable-ml-book/
[65] A. Awad, M. Bader-El–Den, and J. McNicholas, "Patient length of stay and mortality prediction: A survey," *Health Services Manage. Res.*, vol. 30, no. 2, pp. 105–120, May 2017.
[66] J. Yoon, A. Alaa, S. Hu, and M. Schaar, "ForecastICU: A prognostic decision support system for timely prediction of intensive care unit admission," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1680–1689.
[67] A. E. W. Johnson, M. M. Ghassemi, S. Nemati, K. E. Niehaus, D. A. Clifton, and G. D. Clifford, "Machine learning and decision support in critical care," *Proc. IEEE*, vol. 104, no. 2, pp. 444–466, Feb. 2016.
[68] S. Hyland, M. Faltys, M. Hüser, X. Lyu, T. Gumbsch, C. Esteban, C. Bock, M. Horn, M. Moor, B. Rieck, M. Zimmermann, D. Bodenham, K. Borgwardt, G. Rätsch, and T. Merz, "Early prediction of circulatory failure in the intensive care unit using machine learning," *Nature Med.*, vol. 26, pp. 364–373, Mar. 2020.
[69] G. W. Brier, "Verification of forecasts expressed in terms of probability," *Monthly Weather Rev.*, vol. 78, no. 1, pp. 1–3, Jan. 1950.
[70] S.-i. Amari, "Natural gradient works efficiently in learning," *Neural Comput.*, vol. 10, no. 2, pp. 251–276, Feb. 1998.
[71] M. T. Ribeiro, S. Singh, and C. Guestrin, "Anchors: High-precision model-agnostic explanations," in *Proc. AAAI*, 2018, pp. 1–9.
[72] J. Hatwell, M. M. Gaber, and R. M. A. Azad, "CHIRPS: Explaining random forest classification," *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5747–5788, Dec. 2020.
[73] L. Sykora and T. Kliegr, "Action rules: Counterfactual explanations in Python," in *Proc. RuleML+RR*, 2020, pp. 1–13.
[74] A. Takemura and K. Inoue, "Generating explainable rule sets from tree-ensemble learning methods by answer set programming," in *Proc. 37th Int. Conf. Log. Program. Tech. Commun. (ICLP)*, Porto, Portugal, A. Formisano, Y. A. Liu, B. Bogaerts, A. Brik, V. Dahl, C. Dodaro, P. Fodor, G. L. Pozzato, J. Vennekens, and N. Zhou, Eds. vol. 345, Sep. 2021, pp. 127–140.
[75] B. Letham, C. Rudin, T. H. McCormick, and D. Madigan, "Interpretable classifiers using rules and Bayesian analysis: Building a better stroke prediction model," *Ann. Appl. Statist.*, vol. 9, no. 3, pp. 1350–1371, Sep. 2015.
[76] M. Chiari, A. E. Gerevini, M. Olivato, L. Putelli, N. Rossetti, and I. Serina, "An application of recurrent neural networks for estimating the prognosis of COVID-19 patients in northern Italy," in *Artificial Intelligence in Medicine* (Lecture Notes in Computer Science), vol. 12721, A. Tucker, P. H. Abreu, J. S. Cardoso, P. P. Rodrigues, and D. Riaño, Eds. Cham, Switzerland: Springer, Jun. 2021, pp. 318–328.
[77] L. Rokach and O. Maimon, *Data Mining With Decision Trees: Theory and Applications*. River Edge, NJ, USA: World Scientific, 2008.
[78] N. Chawla, "Evaluating probability estimates from decision trees," in *Proc. AAAI Workshop Eval. Methods Mach. Learn.*, Boston, MA, USA, 2006, pp. 18–23.
[79] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[80] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*, 1st ed. Boca Raton, FL, USA: Chapman & Hall/CRC, 2012.
[81] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, Apr. 2006.
[82] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3146–3154.
[83] S. Haykin, *Neural Networks: A Comprehensive Foundation*.Upper Saddle River, NJ, USA: Prentice-Hall, 1994.
[84] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, vol. 1. Cambridge, U.K.: MIT Press, 2016.
[85] D. Lin, A. V. Vasilakos, Y. Tang, and Y. Yao, "Neural networks for computer-aided diagnosis in medicine: A review," *Neurocomputing*, vol. 216, pp. 700–708, Dec. 2016.
[86] N. Shahid, T. Rappon, and W. Berta, "Applications of artificial neural networks in health care organizational decision-making: A scoping review," *PLoS ONE*, vol. 14, Feb. 2019, Art. no. e0212356.
[87] R. N. D'souza, P.-Y. Huang, and F.-C. Yeh, "Structural analysis and optimization of convolutional neural networks with a small sample size," *Sci. Rep.*, vol. 10, no. 1, pp. 1–13, Jan. 2020.
[88] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, "Deep learning for healthcare: Review, opportunities and challenges," *Briefings Bioinf.*, vol. 19, no. 6, pp. 1236–1246, Nov. 2018.
[89] W. Zhao, "Research on the deep learning of the small sample data based on transfer learning," in *Proc. AIP Conf.*, vol. 1864, 2017, pp. 1–9.

[90] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Statist. (AISTATS)*, Fort Lauderdale, FL, USA, G. J. Gordon, D. B. Dunson, and M. Dudík, Eds. vol. 15, Jun. 2011, pp. 315–323.
[91] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUS)," in *Proc. 4th Int. Conf. Learn. Represent. (ICLR)*, San Juan, Puerto Rico, Y. Bengio and Y. LeCun, Eds., 2016, pp. 1–9.
[92] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Proc. Adv. Neural Inf. Process. Syst. Annu. Conf. Neural Inf. Process. Syst.*, Long Beach, CA, USA, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds. 2017, pp. 971–980.
[93] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, May 2018, pp. 1–13.

ALFONSO EMILIO GEREVINI was a Research Scientist with IRST (Now FBK), Trento, Italy, from 1989 to 1995. In 1995, he joined the Università degli Studi di Brescia, Italy, where he is currently a Full Professor of information processing systems with the Department of Information Engineering. He is the author or coauthor of more than 150 published articles on various aspects of *Artificial Intelligence*. He is also involved in a number of research projects in AI, two of which are funded by the EU. His research interests include the fundamental and applied issues of automated planning, knowledge representation and reasoning, machine learning, deep learning, and data mining. He served as a program committee member of the most prestigious AI conferences for many years. He is a fellow of the European Association for Artificial Intelligence (EurAI). He was the Editorial Board Member and an Associate Editor of *Artificial Intelligence journal* (Elsevier), for several years. He was the Editorial Board Member of *Journal of Artificial Intelligence Research* and *Intelligenza Artificiale* (IOS Press).

ROBERTO MAROLDI was a retired Full Professor of radiology with the Medical School and the School of Dentistry, Università degli Studi di Brescia, and the former Head of the Department of Radiology, where he is currently a Professor and also the Dean of the radiography undergraduate course. He is the author or coauthor of more than 140 publications. His research interests include the oncological aspects of head and neck lesions in several anatomic regions and the analysis and assessment of the diagnostic roles of CT and MR. He was a member of the board of the Italian Society of Radiology, from 1994 to 1998, and the President of the Head and Neck Radiology Section, from 1996 to 2000. He was a member of the board of the European Society of Head and Neck Radiology, from 1998 to 2008. He chaired the European Congresses of Head and Neck Radiology, in 2001 and 2009.

MATTEO OLIVATO received the Ph.D. degree from the Department of Information Engineering, Università degli Studi di Brescia, in 2022. He is currently a Research Fellow with the Department of Information Engineering, Università degli Studi di Brescia. His research interests include the application of machine learning and deep learning for clinical data, deep learning for prognosis estimation and health applications, and machine learning in the domain of cybersecurity.

LUCA PUTELLI received the Ph.D. degree from the Department of Information Engineering, Università degli Studi di Brescia, in 2021. He is currently a Researcher with the Department of Information Engineering, Università degli Studi di Brescia. His research interests include the application of natural language processing in the biomedical domain, machine learning for prognosis estimation and health applications, the use of transformer-based architectures for the Italian language, and the application of deep learning techniques in the planning and scheduling domain.

IVAN SERINA received the Ph.D. degree in computer science engineering from the Università degli Studi di Brescia, Italy,in 2000, and a Marie Curie Fellowship in the field of planning and scheduling from the University of Strathclyde, Glasgow, in 2003. He was with the Faculty of Education, Free University of Bozen-Bolzano, from 2008 to 2012. He is currently an Associate Professor of information processing systems with the Department of Information Engineering, Università degli Studi di Brescia. His research interests include the development and the experimental analysis of machine learning, deep learning, efficient automatic domain independent AI planning techniques with innovative applications in several areas, including medicine and health care; predictive maintenance and intelligent manufacturing for Industry 4.0; and AI for e-learning.

Open Access funding provided by 'Università degli Studi di Brescia' within the CRUI CARE Agreement