IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 6, 2020, accepted January 31, 2020, date of publication February 4, 2020, date of current version February 12, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2971576

Human Digital Twin for Fitness Management

BARBARA RITA BARRICELLI[1], (Member, IEEE), ELENA CASIRAGHI[2], (Member, IEEE), JESSICA GLIOZZO[3], ALESSANDRO PETRINI[2], AND STEFANO VALTOLINA[2]

[1]Department of Information Engineering, Università degli Studi di Brescia, 25123 Brescia, Italy
[2]Department of Computer Science, Università degli Studi di Milano, 20133 Milan, Italy
[3]Department of Dermatology, Fondazione IRCCS Ca'Granda, Ospedale Maggiore Policlinico, 20122 Milan, Italy

Corresponding author: Elena Casiraghi (casiraghi@di.unimi.it)

This work was supported by the Bando Sostegno alla Ricerca 2019 through the University of Milan.

ABSTRACT Our research work describes a team of human Digital Twins (DTs), each tracking fitness-related measurements describing an athlete's behavior in consecutive days (e.g. food income, activity, sleep). After collecting enough measurements, the DT firstly predicts the physical twin performance during training and, in case of non-optimal result, it suggests modifications in the athlete's behavior. The athlete's team is integrated into SmartFit, a software framework for supporting trainers and coaches in monitoring and manage athletes' fitness activity and results. Through IoT sensors embedded in wearable devices and applications for manual logging (e.g. mood, food income), SmartFit continuously captures measurements, initially treated as the dynamic data describing the current physical twins' status. Dynamic data allows adapting each DT's status and triggering the DT's predictions and suggestions. The analyzed measurements are stored as the historical data, further processed by the DT to update (increase) its knowledge and ability to provide reliable predictions. Results show that, thanks to the team of DTs, SmartFit computes trustable predictions of the physical twins' conditions and produces understandable suggestions which can be used by trainers to trigger optimization actions in the athletes' behavior. Though applied in the sport context, SmartFit can be easily adapted to other monitoring tasks.

INDEX TERMS Counterfactual explanations, digital twins, Internet of Things, machine learning, smart health, sociotechnical design, wearables.

I. INTRODUCTION

Nowadays, the extension of Internet connectivity into physical devices and everyday objects has radically transformed interactions and communications happening in all the aspects of human life. Devices can now communicate and interact with each other over the Internet, and the data they generate can be remotely monitored and controlled. This condition, referred to as the Internet of Things (IoT) is having a profound impact on our daily lives, and it is fundamentally changing how people interact with physical objects and the environment.

Some of the latest IoT developments include technologies for home, health, transportation, and environment monitoring. In particular, health and wellness applications have emerged as a fast-growing category of smart applications based on the use of wearable devices. This increasing trend is considered as a prompt and useful resource for collecting users' data that are exploited for generating recommendations for a healthy lifestyle.

Together with these advances, the synergic action of ubiquitous connectivity, spread sensor technologies, advances in artificial intelligence (AI), big data analytics, cloud computing, and shared databases (distributed in the cloud) has motivated the development of the industrially diffused Digital Twin (DT) technology [1]. In the fields of aviation and manufacturing, where DTs are nowadays being massively developed and used, a DT is defined as the virtual (digital) counterpart of a physical system. The DT starts living as the physical is being prototyped, dies when it is disassembled, and follows its physical twin during its whole life, resembling its status and being continuously connected and synchronized with it. Thanks to AI, the DT continuously controls and monitors its physical twin status, with the aim of optimizing its performance by triggering self-optimization and self-healing mechanisms. The digital-physical twins interaction is based on a "closed-loop" [2], [3], which refers to the continuous exchange of data between the cyber and physical worlds in order to continuously optimize the physical side. The DT

The associate editor coordinating the review of this manuscript and approving it for publication was Mostafa M. Fouda.

receives data from its physical twin, reconfigures itself to be synchronized with it, applies AI algorithms to detect anomalies and then suggests self-healing or optimization actions.

Thanks to the DT, all stakeholders have a quick and easy access to the physical object status, to monitor and control it and eventually trigger required actions.

The success of DT technology in manufacturing, documented in the extensive survey reported in [4], has motivated several studies aimed at extending it to humans, by designing human DTs, that is computer models of humans tailored to any patient to allow researchers and clinicians to monitor the patient's health, for providing and test treatment protocols [4]. Human DTs differ from DTs developed and used in Industry 4.0 [5] because they are not continuously connected to their physical twin. In this case, users and experts are meant to continuously update the DT with the input of medical digital data describing the health condition of the physical twin.

A specific domain of application of human DTs is described in this paper; it is a software framework, called SmartFit that allows to monitor and manage the fitness activity of teams of athletes, by exploiting a team of DTs, each linked to a team member.

SmartFit [6]–[8] has been designed and developed for allowing trainers to monitor a team of athletes by capturing, for each athlete, a set of measurements describing the athlete's behavior over a period, typically a number of days (D). Such measurements are recorded by physical sensors embedded in wearable devices (e.g. heartbeat, number of steps, physical activity, sleep), and by applications aimed for example at food or mood logging.

After observing the athletes' measurements and their performance during the training sessions, coaches and trainers should be able to identify critical behaviors that need to be avoided or regulated. For doing this, SmartFit allows the coaches to define some rules that are automatically triggered when specific situations take place. SmartFit aims at providing non-professional sport team members with an environment for creating rules using an easy and visual language, familiar to them. Those rules are used for monitoring events related to athletes' habits and are automatically triggered to notify the athlete whose behavior needs to be corrected. Despite the simplicity and efficacy of use of SmartFit interface, during its use it appears that very often the coaches and trainers were not sure about the rules to define, especially when they needed to be complex. To provide help to the trainers in their rules creation task, we have extended the SmartFit application with the artificial intelligence required to make predictions and to compute suggestions helpful to improve athletes' performance. This is exactly the artificial intelligence characterizing DTs. Precisely, the extended SmartFit exploits a team of DTs, each linked to one of the athletes of the team. When receiving novel measurements from one of the athletes, SmartFit sends the measurements to the DT of that athlete, which predicts the fitness of the athlete and provides the suggestions needed to improve the athlete's status. The trainer can exploit the proposed suggestions.

To form the DTs' artificial intelligence, we have initially used SmartFit to collect, for each athlete, t=10 vectors, each containing the measurements recorded for D=3 days. Each vector represents the athlete's status during the D days. To form the training (historical) dataset, a trainer has given a vote (label) after each training session, which has followed the D days of measurement. The collected training (historical) set has been used by SmartFit to build the artificial intelligence of each athlete's DT. After training, SmartFit is ready to: collect novel unlabeled vectors describing the status of each athlete; exploit each athletes' DTs to predict their status and to compute suggestions for improving the athlete's outcome; output the computed suggestions to the trainer, who can trigger changes in the athletes' behavior.

The novelty of this paper is the introduction of the concept of *team of human DTs*. The main strength and peculiarity of such artificial intelligent DTs is the capability of describing the reasons of the computed predictions, by computing the smallest change to the feature values that increases the prediction. Such explanations are important since they not only allow improving the physical twins' conditions, but also make experts, in this case the trainers, to trust the output of the automatic learner, which would be otherwise viewed as a ''black-box''. Thanks to SmartFit, the trainer can then trigger all or some of the modifications suggested by the team of DTs.

Therefore, the main contribution of this work is to introduce a sixth element in the ''Human-In-The-Loop'' ecosystem [9]: a team of humans' digital twins interacting with the end users (at the center of the ecosystem) to predict their future statuses and provide suggestions for improving their fitness experience and their wellness in general.

Moreover, thanks to the generalization capability of the machine learning applications used to develop the team of DTs, this work describes a human monitoring framework that could be exploited by generic experts to monitor the conditions of patients whose health status may be described by measurements collected through medical IoT connected devices.

This paper is organized as follows: in Section II the literature review and research background are reported; Section III is devoted to the detailed description of the artificial intelligence of the athletes' DTs; Section IV presents and discusses the experimental results of the team of DTs; Section V reports conclusions and future works.

II. RESEARCH BACKGROUND

A. LITERATURE REVIEW

1) DIGITAL TWIN: FROM AVIATION TO HEALTHCARE

In the 1970 NASA started creating mirrored systems to monitor unreachable physical spaces (e.g. spacecrafts in mission), with the aim of finding out solutions to unexpected problems, by using the mirror to test possible solutions. One of the most famous examples of NASA mirrored system is the simulated environment developed by engineers in Houston and Kennedy Space Center which was used to rescue

astronauts during Apollo 13 mission. Indeed, when the air tanks exploded, engineers exploited the simulated environment to model and test possible solutions, and successfully found a way out, which was an improvised air purifier [10]. From earth, engineers instructed the astronauts how to build it with materials available in the spacecraft. At the same time, by using simulating different scenarios, engineers on earth finally found the way to get the crew of Apollo 13 back to earth alive.

Mirrored systems were useful for they bridged physical and virtual spaces; however, they still lacked the abilities required for allowing a smart interaction between the two spaces. Such abilities are those characterizing the DTs, virtual twins living together, and being synchronized with, their physical twin (PT). Thanks to seamless connection and continuous interaction with their PT and with the external environment, DTs are able to continuously simulate the conditions of the PT. Simultaneously, they analyze the received data, which describes both the PT's condition and the external environment, in order to predict future statuses and trigger optimizing and/or preventive actions in case of predicted failures [1], [4].

In the field of aviation, DT models are principally developed for improving the aircrafts performance and reduce costs of failures, by predicting damages and triggering self-healing actions to avoid them [11]–[13].

In the manufacturing fields, DTs [14]–[17] are used not only for allowing damage forecast and self-healing mechanisms through virtualization of manufacturing machines, but also for optimizing manufacturing processes or even factories [18]–[22].

Due to their success in the aviation and manufacturing fields, DTs are spreading also in the healthcare and medical fields.

In healthcare, they have been firstly implemented for allowing predictive maintenance of medical devices and for optimizing their performance in terms of examination speed and energy consumption, while other applications have been successfully developed to optimize hospital lifecycle [23]–[25].

In the medical and clinical fields, DT technology is studied with the aim of building the human DT, a virtual human linked to its physical human. Such a DT would allow a detailed and continuous inspection of the human health status, thus allowing to predict the occurrence of an illness and its best prevention and/or treatment by also considering the PT's personal history and the current context such as location, time, and activity [26]. Such human DT is important because it would promote the shift from the common way treatments are delivered in medicine, which is the ''one-size-fits-all'' method, where patients are treated according to some ''norm'' or ''Standard of Care'', to the so-called ''personalized medicine'' [27], [28], where treatments are tailor-made, based on the individual ''physical asset'', defined by all the structural, physical, biological, historical characteristics of the individual.

The research effort devoted to the development of human DTs [29] has brought to the development of computational models such as the ''AnyBody Modeling System''[1] that allows to simulate the human body working in concert with its environment. With the AnyBody model users can run advanced simulations to calculate: 1) individual muscle forces; 2) joint contact-forces and moments; 3) metabolism; 4) elastic energy in tendons; and 5) antagonistic muscle action.

A physiological model virtualized by a DT would allow physicians to make *in silico* predictions of how the real organ might behave in any given situation; this motivates the great deal of research devoted to the development of DTs for monitoring the conditions of organs or of their functions in patients; examples are the DT of the heart [30]–[32], or the DT of the hearth specifically developed for monitoring myocardial conditions [33], or the DT of the airway system [34], specifically developed for monitoring the inhalation of patients receiving aerosol-delivered chemotherapeutic drugs. This has been possible thanks to the availability of economical storage and convenient access and exchange of medical examinations (mainly images) from multiple modalities (source machine types).

Even if DTs have encountered great success in many fields, the DTs developed in healthcare differ enormously from the one designed in industry, mainly for two reasons. The first, human DTs should be built upon AI applications and should make a strong usage of them. The second, and most important, humans are not equipped with embedded sensors, and medical data describing their status can be extracted only from medical examinations; therefore, the seamless connection among humans and their DT cannot be guaranteed.

In this regard, a recent and interesting research work [35], strongly related to our, presents a conceptual framework for allowing a continuous communication between IoT-connected wearables devices transmitting PTs' health-related data to their human DTs. Such conceptual human DT model would allow improving the healthcare of elderly people by allowing a continuous monitoring of their conditions.

A similar theoretical work [36] describes and defines a conceptual DT model to be applied for continuous monitoring and personalized data-driven medical treatments.

2) MISSING DATA IMPUTATION METHODS

Being connected through their physical twins by also (manually) recorded data, human DTs must generally deal with datasets containing missing data. In this case the application of techniques for data imputation, which ''fill in'' the missing data based on the structure of the underlying dataset, have been proven to be an effective technique when dealing with biological data.

Among them, the k-nearest neighbors imputation (knn-imputation) algorithm [37]–[39] has proven to be

[1] https://www.anybodytech.com.

effective when applied in the field of bioinformatics to impute missing values for gene expression arrays [37]–[38].

The schema of knn-imputation is simple; to impute the missing value of the i^{th} missing feature value x_i of sample x, the algorithm searches for the k_{impute} feature vectors that are the most similar (closest or nearest) to the i^{th} feature vector F_i. In the following such "nearest to F_i" feature vectors are referred to as: $F_{NN(i,h)}$ (h=1,..., k_{impute}), where NN(i,h) is the index of the feature vector which is the h^{th} nearest to F_i. The imputed value x_i for sample x(n) is then computed as the weighted average of the k_{impute} nearest feature values (of x(n) itself), that is:

$$x_i(n) = \sum_{h=1,\ldots,k_{impute}} \frac{w_{NN(i,h)} x_{NN(i,h)}(n)}{k_{impute}} \quad (1)$$

where the weights $w_{NN(i,h)}$ are directly proportional to the similarity between the feature vectors F_i and $F_{NN(i,h)}$. If one of the values $x_{NN(i,h)}$ from one of the nearest neighbor features is also missing, the $(k_{impute} + 1)^{th}$ nearest feature value is used. It is worth noting that the point label is never used in the computation of the knn-imputation, thus avoiding any information leakage.

The work reported in [40] interestingly concentrates on the data imputation when the amount of missing data is high, that is when up to 60% of data is missing. Precisely, after inserting different percentages of missing data in different attributes of three datasets (Bupa, Cmc and Pima) belonging to the UCI repository [41], they compared results obtained by knn-imputation to those achieved by the internal methods used by the C4.5 (tree generator algorithm [42]) and the CN2 (rule induction algorithm [43]) to treat missing values. The study showed that, not only knn-imputation allows obtaining higher performances, but also that it is superior over other imputations methods, such as "case substitution", "mean and mode", "hot deck and cold deck", or those using predictive models.

In [44] authors pointed out the most notable characteristics of knn-imputation, which are:

- knn can predict both discrete attributes (the most frequent value among the k_{impute} nearest neighbors) and continuous attributes (the mean among the k_{impute} nearest neighbors);
- imputed values are inferred values which belong to the dataset distribution;
- knn makes use of auxiliary information provided by the values in the dataset, thus having a major chance of preserving the original data structure;
- once the k_{impute} value is established, knn is non-parametric and uses the data set as a "lazy" model, thus not requiring the creation of predictive models for the missing measurements, and therefore being less prone to model misspecification;
- since no explicit models (like a decision tree or a set of rules) are needed for imputation, knn-imputation can be easily adapted to work with any feature set, by just modifying which attributes will be considered in the similarity metric;
- knn can easily treat examples with multiple missing values.

However, authors also highlighted the following notable drawbacks of any method based on knn searches:

1) results may be dependent from the choice of the metric used to gauge the similarity (or the distance) between observations [45].
2) due to the knn search, when the dataset has many samples, the algorithm might be time consuming.
3) when unbalanced data are used, knn-imputation may create a bias toward the most represented class, causing the nearest neighbor to be always chosen from that class [46].
4) results may be dependent from the choice of the number of neighbors to be considered [44].

To deal with the first problem, several state-of-the-art works have been proposed which essentially modify the metric used to compute the distance between samples. In particular, in [38], while imputing missing data in genes, authors compare the usage of Pearson correlation, Euclidean distance and variance minimization, and conclude that, though being sensitive to outliers, Euclidean distance appears to be the most accurate norm when dealing with DNA microarray data. In [47] authors present a comparative work to evaluate the effect of knn-imputation in the performance of a knn-classifier, a Bayesian tree, and an svm classifier. The task is the outcome prediction of breast cancer patients described by clinical data. The distance measure used to find the k_{impute} nearest neighbors is the Heterogeneous Euclidean-Overlap Metric [48]. Interestingly, authors show that the application of knn-imputation allows improving the performance of all the classifiers.

In [49] authors propose a knn-imputation method which exploits a novel "hybrid" distance measure integrating the Pearson correlation and the Euclidean distance, while in [50] authors design an iterative knn-imputation scheme for filling missing data in a trash pickup logistics management system. The iteration is devoted to the recomputation of an ad-hoc distance, namely the Grey Relational Grade [51], between samples after each missing attribute is imputed. Finally, in [52] the authors use knn-imputation to fill missing values in miRNA expression profiles of patients, but they substitute the measure between expression profiles with the Euclidean distance between the clinical patient data.

Other studies concentrate on the choice of the best k_{impute} value. Preliminary works in the field of scene analysis, when the data is sparse and its variability is high [53], suggested to use the square root of the number of complete cases, rounded to the nearest odd integer, and such proposal was confirmed by the study reported in [54], where knn-imputation was applied to questionnaires using Likert scales from 0 to 5 for collecting answers. However, as the k_{impute} value gets bigger the computational time required for

searching the neighbors increases. Moreover, the neighbors used for computing the missing value are much more different, which implies that the imputed value could be less accurate [52]. To finally propose trustable values for k_{impute}, in [44] authors performed exhaustive experiments on both simulated and real datasets. The achieved results showed that, whatever the experimental framework, 1NN was the only method capable of preserving the data structure, and data distortion was neglectable only when relatively small values of k_{impute} ($k_{impute} \leq 5$) neighbors were considered. However, in case of noisy data, 1NN means that noise will have a higher influence on the computed results. For this reason, and given the results of their experimental evaluation, authors suggest that the value of $k_{impute} = 3$ is a reasonable choice for several classification problems.

Other works propose iterative versions of the knn imputation scheme. In particular, a notable iterative knn-imputation method has been proposed in [55]. The proposed knn-imputation framework initially split the input data (genes) into an ''incomplete'' and ''complete'' set, which contain, respectively, missing and not missing values. The genes in the incomplete set are sorted according to the percentage of missing values and are imputed by the order of missing rate. After the missing values on genes with less missing values are imputed through 1NN imputation, such imputed gene are moved into the complete set and used for the imputation of the rest of genes in incomplete set.

Since this method works on subsampled complete set, its time consumption is reduced in case of datasets composed by many samples. Indeed, subsampling is the sometimes the preferred choice both when the knn search is performed in datasets with high cardinality and when unbalanced datasets must be treated [46], [56].

3) knn CLASSIFIERS AND SUPPORT VECTOR MACHINES FOR PREDICTION

When missing data have been imputed different DTs must generally perform prediction of future conditions, to eventually propose repairing actions.

In the particular case of human DTs, most of the times such prediction regards the classification of health conditions. In this context, several state-of-the art works [57], [58], perform classification by exploiting ''network-based'' approaches, which construct graphs of patients and exploit their relationships to perform classification. Among such methods, a simple and effective approach for performing classification in a constructed ''patient space'' is the knn algorithm [59], which is essentially based on the exploration of the graphs constructed on the basis of some similarity between patients.

knn [59], [60] is a non-parametric instance-based learning method used for classification and regression. Due to its simplicity and efficacy, it is one of the most relevant methods in the machine learning and data mining fields [61].

Briefly, given a set of training samples and an unlabeled (test) instance, whose prediction must be computed, knn builds the patient graphs based on the patient similarity, and assigns the class to the test instance as the most frequent label in the group of the k nearest training instances. Essentially, to classify an unlabeled sample, knn simply evaluates its distance to every training samples, finds the k closest ones and uses their labeling to select the class to be assigned to the test sample.

Due to the knn search, two major factors influence the prediction performance (see the drawback list mentioned for knn-imputation): the choice of an appropriate metric to evaluate the distance between samples and the choice of the number of nearest neighbors (k). For the former, a few out of the possible choices are Euclidean distance, Minkowsky distance or Mahalanobis distance and their variants. Unfortunately, the choice of an appropriate metrics is highly dependent from the problem [64]. For the latter, the parameter k is usually fixed and does not take into account the actual distribution of the dataset. For this reason, some approaches have proposed to dynamically assign an appropriate value of k for each test sample based on the training set distribution [65].

Anyhow, at the state of the art, some techniques have been using simple knn classifiers, e.g. to classify EEG signals for predicting depression [62] or Paroxysmal Atrial Fibrillation [63], while some other authors have combined several knn classifiers, each characterized by a different value of parameter k, into an ensemble classifier, whose prediction is generally performed by aggregating the classification of all the classifiers in the ensemble through a majority vote [66], [67].

Other approaches have been using knn techniques as the final judges at the end of cascading systems, e.g. for the classification of diabetic patients [68] or for the classification of marker pixels [69] from histochemical images.

Other classification approaches achieved promising results in classification of patient data, are those exploiting support vector machines (svm) [70], [71]. Svms [72] are supervised machine learning methods developed to solve binary classification problems. Briefly, the algorithm tries to identify the optimal hyperplane that separates data points in the correct classes by maximizing the margins between the vectors of the two classes. The so-called support vectors are the data points that are closer to the separating hyperplane and influence the position and orientation of the hyperplane.

Since not all problems are linearly separable, the application of a suitable kernel (e.g. Gaussian, polynomial, or linear) is generally employed to implicitly map the input data into a higher dimensional feature space where the dataset may be separable [73].

In their original formulation, svms were meant to be applied to balanced datasets, which means that the distributions of the classes in the dataset are similar. However, when dealing with real datasets, the negative class is often the mostly represented, and, as we already mentioned, this may introduce a bias in the accuracy of the classifier [46] and may particularly decrease the performance of svm classifiers [74]. To tackle this problem, some authors oversample

the under-represented class before training the classical svm classifiers by generating synthetic samples [75], other works subsample the over-represented class [46], [56], and some other authors apply both approaches at the same time[76] before training the svms. Though some promising results have been reported, it is well-known that the generation of synthetic samples must address the critical choice of a data model driving the sample synthetization [77]. Models which are too close to the original data may cause overtraining, while too general models introduce distortion in the underlying data distribution. On the other hand, the simple random under sampling of the majority class may cause the loss of potentially useful information to build the classifier and the choice of meaningful ''informed under sampling'' techniques is required to overcome this issue [78].

For these reasons, most approaches [70], [79], [80] obtain promising results by exploiting cost-sensitive svm approaches.

Precisely, cost-sensitive learning methods introduce a cost of misclassification to weigh differently the classification errors in different classes [78]. In particular, in this paper we applied the cost-sensitive svm, also known as ''biased penalties svm'' [81]. More in detail, in the framework of the svm optimization problem, two regularization parameters, θ_+ and θ_-, were introduced to be able to adjust the cost of misclassification of negative and positive samples.

Since svm machines are binary classifiers, before applying them to perform multiclass classification, they are generally combined into ensemble techniques. In particular, several state-of-the-art works have shown that the best results are computed by ensemble methods using ''binarization'' strategies [82]. They divide the original data set into two-class subsets, learn a different binary model for each new subset, and then compose the predictions of the binary classifiers, according to different aggregation schemes.

In this work, we have chosen to use error-correcting output code (ECOC) models, which are ensemble methods specifically designed to handle multiclass classification problems. The basis of the ECOC framework is to decompose a multiclass problem into a larger number of n binary problems, according to a coding design[2]. In this way, each classifier is trained on a binary meta-class problem, where each meta-class consists of some combinations of the original classes, according to some coding design. Since each binary classifier has output 0 or 1, each class is then represented by the expected n-dimensional codeword (string) which contains the expected output of each binary classifier. When a novel point must be classified, the n binary classifiers are evaluated to obtain an n-dimensional string representing the point, and the class whose codeword is the closest to x's output string is chosen as the predicted label. In our case, the distance among two codewords is the Hamming distance.

[2]For a list of all the possible coding designs see the MATLAB documentation at: https://it.mathworks.com/help/stats/fitcecoc.html

ECOC improves the generalization performance of the base binary classifiers [83], and reduces their bias and variance errors [84]–[86].

4) EXPLAINING PREDICTION AND PROVIDING SUGGESTIONS WITH COUNTERFACTUAL EXPLANATIONS

One of the main features desirable especially for human DTs regards the ability of explaining the computed predictions. With the aim of opening the so-called ''black-boxes'', a great deal of research work have been recently devoted to the development of automatic techniques for explaining the predictions computed by machine learning methods, and the growing literature about this subject has indeed generated a novel field of research called ''interpretable machine learning'' [87]–[91].

Among the different techniques for explaining the classifiers' decision, counterfactual explanations are sometimes preferred because they explain the output of the classifier for the specific input point.

Counterfactual Explanations are defined as statements taking the form [92]: ''Score p was returned because variables V had values (v1,v2,...) associated with them. If V instead had values (v1', v2',...), and all other variables had remained constant, score p' would have been returned''.

In the field of pattern classification, counterfactual explanations explain the trained model by showing how the input parameters allow obtaining specific classification results, or, in other words, how to change the values of the input data to obtain a desired classification result. Therefore, by computing counterfactual explanations we may provide prescriptive suggestions. Essentially, thinking in counterfactuals requires imagining a hypothetical reality that contradicts the observed facts, hence the name ''counterfactual''.

The ability to derive counterfactual explanations is nowadays becoming a problem of paramount importance, especially in the clinical field, where classifications from ''black boxes'' are not useful at all.

In [93], authors present a theoretical approach for finding counterfactual explanations. Essentially, if the trained classifier model computes function $\hat{f}()$ and y' is the desired class, they suggest finding the counterfactual $\tilde{x}$ that minimizes the following loss:

$$L(x,x', y', \lambda) = \lambda(\hat{f}(x') - y')^2 + d(x,x') \quad (2)$$

where the first term is the quadratic distance between the model prediction for the counterfactual x' and the desired outcome y', d(x,x') is a distance function measuring the difference between the analyzed point x and the counterfactual x', λ is a user-set parameter balancing the distance in prediction (first term) against the distance in feature values (second term).

Given an input point x, a desired outcome y', a trained classifier model (that is, a function $\hat{f}()$), a value for the

parameter λ, and a distance function d(), the loss function in Equation 6 becomes a function of the counterfactual x'. Therefore, the best counterfactual $\tilde{x}$ is found as: $\tilde{x} = \text{argmin}_{x' \in X}(L(x'))$, where X is the input space and function argmin(L) returns the argument that minimizes the loss L.

Though this theoretical approach is sound and convincing, function minimizations are often misled by local minimums and might require high computational costs, which are not acceptable when the need is to develop real time applications.

Therefore, some authors find counterfactuals by applying ''smart'' iterative procedures in the input space [94] or search the counterfactual in a (growing) neighborhood of the starting point x; the neighborhood is enlarged until a counterfactual that allows to obtain the desired output is obtained [95]. Of note, some authors provide classifier interpretations that, though opposite to counterfactual explanations, allow anyway to interpret the classifier and provide counterfactuals. As an example, in [96], authors find anchor explanations. An anchor explanation is a rule that sufficiently ''anchors'' the prediction locally, such that changes to the rest of the feature values of the instance do not matter. In other words, for instances on which the anchor holds, the prediction is (almost) always the same. Though the definition of anchors is opposite to counterfactuals, once anchors are found, the counterfactuals can be found by identifying the anchor values that allow obtaining the desired predictions, without looking at the not-anchor features.

5) THE ''HUMAN-IN-THE-LOOP'' ECOSYSTEM

As stated by Shneiderman [97], the so-called ''old computing'' was the one focused on what computers can do for their users; the ''new computing'', instead, is meant to be about the users' activities and what they can achieve by using computers. The continuous evolution of users from being passive consumers of data into active producers of information and even of software is deeply changing the way interaction and systems in general are designed [98], [99].

Especially in IoT design, the possibility of connecting devices over the Internet offers great opportunities and does not only enrich the way the users interact with technology and manage their personal data, but also gives the chance of sharing data with other people who can be family members, friends, colleagues, or others.

However, data sharing in the long term leads to the creation of a large quantity of data; this calls for integration of recommending, intelligent, and distributed systems to support domain experts, who are not necessarily technical experts, in dealing with such large amount of data and to help them in making sense of it. Success in designing and developing tools and services based on IoT – and DTs are the most complete example – requires a broad approach that includes expertise in sensing and hardware, networked systems, human-computer interaction, usability, and data management.

Essentially a DT, or a generic IoT environment, can be viewed as a ''Human-in-the-Loop'' [9] ecosystem of elements (hardware and software) that exchange data through the Internet and act and react in a semi-automatic or automatic way according to events, and/or to preferences, rules, or decision of domain experts' (users') which are at the center of the ecosystem (the Loop) [100], [101]. Especially in case of a human DT, at the center of this ecosystem stands the user, the one who generates (or contributes to) the data, manages the IoT elements in the ecosystem, and unwittingly develops in the IoT environment defining the interactions among the elements and the elements' behavior [100], [101]. In this scenario, the users find themselves at the center of a complex ecosystem that they need to manage in efficient, effective, satisfactory, and aware manner.

While in [102] the elements of the ecosystem have been categorized into five groups, which are sensors, applications, social media, recommendation systems, and other users, the contribution of this work is to introduce a sixth element in the ecosystem: a team of humans' digital twins that interact with the end users (at the center of the eco-system) to predict their future statuses and provide suggestions for improving their fitness experience and their wellness in general.

B. SMARTFIT

SmartFit [6]–[8] is a framework helping coaches and athletic trainers of non-professional sports teams to monitor and analyze data regarding the fitness and well-being of the athletes.

Through the use of specific rules, coaches and athletic trainers can detect relevant and significant events that may affect their athletes' performances and that depend on data such as the number of calories taken and burned or the number of hours of sleep.

Unfortunately, the problem with this system is that the coach/trainer is often ''lost in the data sea'' and does not know which parameter each athlete must modify to improve its fitness.

To help coaches and trainers in the monitoring task, we have extended SmartFit with machine learning techniques that enable predictions and we have developed a method, based on counterfactual explanations, to compute suggestions for improving athletes' performance. Essentially, suggestions are provided as instructions, or rules, that describe a change in behavior.

SmartFit is therefore able to compute predictions and the consequent rules, by extracting knowledge from the collected measurements describing both the athletes' performances and the coachers/trainers' feedbacks. In other words, SmartFit provides an interpretable predicting and prescriptive system, often said a ''white-box'' model [103], [104].

In particular, the developed machine learning approach produces a set of readable and understandable IF-THEN rules, which are easily interpretable and follow a similar pattern to human thinking.

For example, if coaches/trainers/nutritionists, or any other expert, want to trigger actions on athletes showing bad performance, they could use the rules produced by the athletes' DTs integrated in SmartFit to check if, e.g., too much intake

calories or too few hours of sleep are correlated to bad performances during the training sessions. On the other side, if coaches had no idea about the reasons of bad performance during training, they could use the rules provided by the DTs in SmartFit to trigger ''optimization'' rules, to modify the athletes' behavior. Thanks to SmartFit, while creating these rules, the coaches/trainers must specify what happens when a particular set of conditions is met/not met by defining a list of actions to be performed, such as by specifying that a warning needs to be sent via direct messages.

By using direct messaging system, coaches and trainers have access to suggestions that can be shared among other colleagues at a later time. Moreover, adjustable parameters in the rules produced by the DTs allow to fine-tune them if coaches/trainers are not satisfied with the recommendations.

To test SmartFit, we have used data collected in 2016, obtained by monitoring for one month a non-professional soccer team composed by 11 teenager athletes (all males and all 19 years old).

The measurements related to each athlete have been labeled by the trainer according to the performance of each athlete in the training session following the D=3 days of measurement.

Precisely, each athlete's day is described by t = 22 features, which record:

- The food income, described by 7 features: grams of carbohydrates, fat, proteins, cholesterol, sodium, sugar and fibers. These data were collected by inviting the athletes to use the mobile application *MyFitnessPal*[3] to log their meals (breakfast, lunch, dinner and snacks).
- The calories intake relative to the day, which are: the calories eaten, the basal calories, and the calories for activities burnt in one day. All these data were produced by the app *MyFitnessPal* on the basis of the meals logged by the athletes.
- The activity of the day, described by: number of steps, number of floors, walked distance, minutes of seating activity, minutes of light activity, minutes of moderate activity, and minutes of intense activity. These data were collected using *Fitbit Charge HR*[4], a wristband tracker that detects physical activity, rest, sleep and heartbeat.
- The quality of the athlete's rest, described by: minutes of rest, minutes of sleep, minutes of wake, number of awakenings. These data are collected by *Fitbit Charge HR* too.
- The athletes' mood, represented by an integer number ranging from 0 (bad day) to 5 (happy). These data are collected by inviting the athletes to log their mood on a daily basis by using the mobile application for Android *Moodtrack Social Diary*[5]. It has to be clarified that this app has deeply changed over time and its interaction has become way richer than it was at the time of our data collection. In fact, in 2016, through the app the users could just select one out of six moods that were graphically represented by an emoticon hence the use of the range 0-5 for our application.

[3] https://www.myfitnesspal.com/
[4] https://www.fitbit.com/
[5] https://play.google.com/store/apps/details?id=com.moodtrak.diary

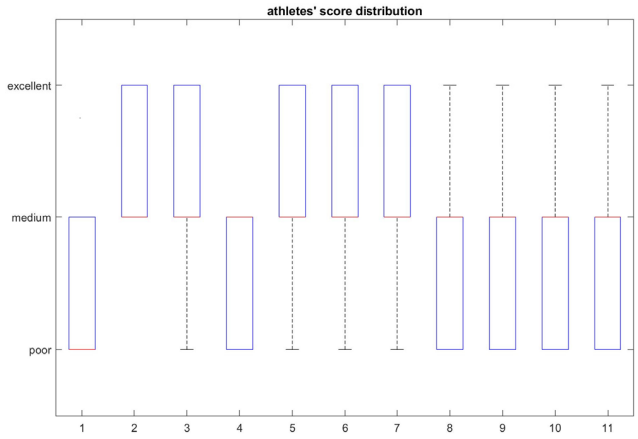

FIGURE 1. Athletes' scores distribution. Note that the intra-patient score distribution is quite variable.

Since the measurements are recorded for D=3 days, each athlete's condition is described by t=66 measurements.

After the three days, each athlete has a training session, after which an expert in the field (e.g. the trainer or the physiotherapist), evaluates both the quality of the athlete's training and their health with a number: 1 (poor), 2 (medium), 3 (excellent).

Since the athletes have been monitored for one month (30 days), for each athlete, the process of 3-days measurements plus training end evaluation has been repeated for 10 times. In this way, we obtain a set containing N=110 points (10 measurements, or sample points, per A=11 athletes).

Precisely, the (quite unbalanced) dataset is composed of $N_{poor} = 19$ points with fitness score ''poor'', $N_{medium} = 71$ with fitness score ''medium'', $N_{good} = 20$ with fitness score ''good''.

The boxplots in FIGURE 1 show, for each athlete, the minimum, the median, and the maximum score. This allows observing that each athlete has a quite variable fitness score distribution. Indeed, only two athletes have never been assigned score excellent, one athlete has never been assigned score poor, while others have been assigned each score at least ones.

III. THE TEAM OF DTS

In this Section, we describe the system developed to create the artificial intelligence (AI) of a DT, which monitors the health status of a generic person (i.e. a physical twin, which is an athlete in this case), described by a set of (fitness related) parameters, and provides predictions about the fitness outcome and suggestions to improve it.

More precisely, in Subsection III.A we firstly describe the algorithms developed for analyzing the historical (training) data, which contains the measurements describing N fitness

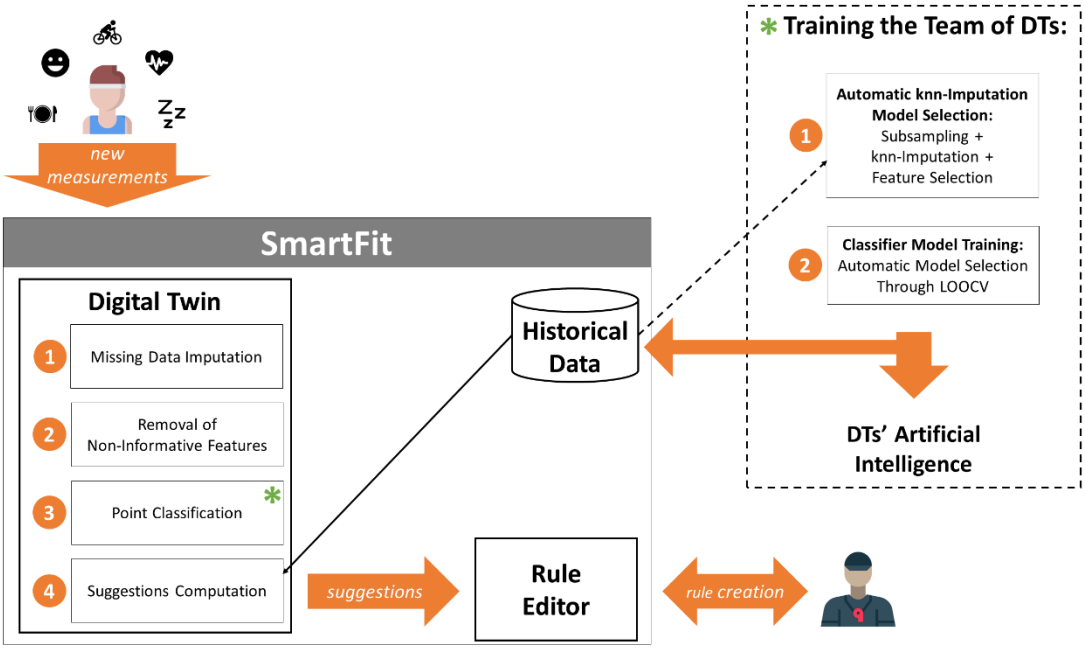

FIGURE 2. Architecture of the developed model.

conditions, each represented by a set of heath measurements, and by a "known" fitness/health score provided by experts (e.g. fitness trainers, coaches, medical doctors, or nutritionists). After processing such data, and therefore forming the memory, knowledge, and prediction capabilities of the DT (III.A), when a novel set of measurements describing the PT is provided (by the PT themselves, or by any other expert monitoring the PT), the DT estimates the current PT's conditions, and then provides suggestions for improving the PT's health status (see Subsection III.B).

In our explanation, the training set X is composed by N training points (measurements), that is $X = \{x(1), \ldots, x(N)\}$. The training points will be also referred to as samples, sample points, or training points in the following. Each $x(m)=[x_1,\ldots x_i, \ldots, x_\tau]$ is represented by a feature set containing t features describing the health status of one PT.

The label (ground truth) of the points in the training set is a numeric, categorical fitness score, $c \in C$ assigned by an expert (e.g. a fitness trainer, a medical expert, or a nutritionist).

Note that, when real datasets are used, the class-subsets have quite different cardinalities, thus resulting in unbalanced datasets, which may pose problems when dealing with both binary and multiclass classification problems [46], [105]–[107]. Moreover, some measurements might be missing for some samples, thus introducing the need of appropriate approaches for handling not available data.

In Figure 2 the developed model architecture is depicted.

At first the SmartFit framework collects historical data composed by the athletes' measurements and by the trainers' evaluation (ground truth), and stores it in the data storage.

When enough historical data has been collected, the training phase starts (dashed box in Figure 2, described in detail in Subsection III.A) to generate the DTs' artificial intelligence.

Particularly, during training, the following tasks are executed. At first, the missing data imputation model is automatically selected and the missing historical data are imputed (as explained in Subsection III.A.1). Simultaneously, the most discriminative features are selected (see Subsection III.A.2), the (imputed) historical data is coded by solely using them. Next, by using the coded historical data, the optimal classifier models are automatically selected and trained (as illustrated in Subsection III.A.3). Finally, the resulting coded historical data is re-sent to the data storage, to be stored for processing novel data.

When the training phase ends, the team of DTs are ready to process novel measurements, whose fitness score is not yet known (SmartFit box in Figure 2, as described in Subsection III.B). Precisely, when novel measurements are provided, they are coded by eventually imputing missing data and by selecting only the most informative features (identified during training). At this stage, they are input to the trained classifier that provides a fitness score prediction and then retrieves the coded historical data from the data storage to compute suggestions for improving the predicted score.

The suggestions are then sent to the Rule Editor environment in SmartFit, to let trainers and coaches compose rules as extensively explained in other publications (e.g. [6], [102]).

Note that, whenever some measurements with their ground truth label are provided by the expert trainers, they are added to the historical data and the training phase is repeated to update the DTs knowledge with the novel labeled data.

A. TRAINING THE DT

In this Section, we describe the steps aimed at training the DT, to form its AI (dashed box in Figure 2). Precisely, after estimating the missing values with the knn-imputation algorithm (see Subsection III.A.1), the most informative characteristics, appropriately called the ''right data'' in [108], are chosen through feature selection (see Subsection III.A.2), and four different multiclass classifiers are then trained (Subsection III.A.3).

1) MISSING DATA IMPUTATION

Since the measurements describing each person's health are manually recorded, they are sometimes forgotten, thus resulting in missing data. To fill the gap, the DT applies knn-imputation [37] (see Section II.A).

Though being effective for several classification tasks where datasets have a high percentage of missing values [40], it must be noted that knn-imputation works well when the feature vectors have the same type and take values in the same range. However, in our problem, we treat health features whose types are quite different from one another; they might be both categorical variables (expressed by discrete number), or continuous variables, and their ranges of variation are very different between each other. For this reason, imputing missing values by searching for nearest features is not meaningful. Based on this consideration, we estimate the missing values by searching for the k_{impute} nearest samples, rather than the k_{impute} nearest features. In practice, if the i^{th} feature, $x_i(n)$, for the n^{th} point is missing, the imputed value is computed as the weighted mean of the i^{th} feature values of the k nearest neighbors of x(n). In practice, we compute:

$$x_i(n) = \sum_{h=1,\ldots,k_{impute}} \frac{wP_{NN(i,h)} x_i(NN(n,h))}{k_{impute}} \quad (3)$$

where $wP_{NN(i,h)}$ is the weight, directly proportional to the normalized similarity $Sim_{Norm}(x(n), x(NN(i,h)))$ (described later in this Subsection) between sample x(n) and its h^{th} nearest neighbor whose index is NN(n,h).

As mentioned in Section II.A knn-imputation has four main drawbacks which regard:

a) the choice of the metric used to gauge the similarity between observations [45]. In our case we use the normalized similarity measure between samples (described in the following Subsection a)), which has been defined to work on clinical patient data where each feature has a different meaning.
b) the high computational cost required by the knn search. To cope with this issue we subsample the training set as explained in the following Subsection b).
c) the data unbalancing. To cope with this problem we test two different techniques; the first one is the training set subsampling described in Subsection b), while the second is the use of cost sensitive support vector machines [81] (see Subsection III.A.3).
d) the choice of the number of neighbors to be considered. To tackle this problem, we developed an automatic knn-imputation model selection that automatically selects the value of k maximizing the informativeness of the imputed dataset.

In the following, we describe the similarity measure between point samples (Subsection a), the training subsampling method (Subsection b), and the automatic knn-imputation model selection technique we developed (Subsection c).

a: NORMALIZED SIMILARITY BETWEEN POINT SAMPLES

To answer point a) in the previous drawbacks' list, we use the weighted normalized similarity measure (defined in [109]), that has been specifically designed to evaluate the similarity between patients described by discrete, categorical, clinical features with differing semantic meaning and characterized by different data types.

Precisely, as it is defined in [109], the normalized similarity, $Sim_{Norm}(x(1), x(3))$, between points x(1) and x(3) is computed as:

$$\begin{aligned} &Sim_{Norm}(x(1), x(2)) \\ &= 1 - Dist_{Norm} \\ &= 1 - \frac{\sum_{f=1}^{t} \frac{|x_f(1) - x_f(3)|}{max(f) - min(f)}}{t}, \; x(i) = [x_1(i), \ldots, \; x_t(i)] \end{aligned} \quad (4)$$

where $Dist_{Norm}$ is the normalized distance between x(1) and x(3), and max(f) and min(f) are, respectively, the maximum and the minimum values of feature vector F_f. Note that the difference $|x_f(1)-x_f(3)|$ is neglected when one of, or both, the two values $x_f(*)$ is missing for feature f (in this case, the mean is obviously computed over t-1 values).

The $Sim_{Norm}(x(1), x(3))$ distance works well when distances between points have the same importance. However, to improve the DT's generalization capabilities, it must receive and process historical data from both its PT and from entities in the so-called ''external environment''. In this case, such entities are the other team members, and their data are their measurements and the achieved performance. Based on the consideration that each individual is characterized by personal conditions, attitudes, and reactions to events, the DT must give more importance to data from its physical twin. To this aim, we have introduced a weight (w_{Ind}) that gives more importance to similarities between measurements from the same individual, by lowering the normalized distance between them:

$$\begin{aligned} &SimW_{Norm}(x(1), x(3)) \\ &= 1 - Dist_{WNorm} \\ &= 1 - \frac{w_{Ind} * \sum_{f=1}^{t} \frac{|x_f(1) - x_f(3)|}{max(f) - min(f)}}{t} \end{aligned} \quad (5)$$

where $Dist_{WNorm}$ is the weighted normalized distance, and the weight $w_{Ind} = w_{same}$, $0 < w_{same} < 1$, if x(1) and x(3) are two measurements from the same individual, while $w_{Ind} = 1$, otherwise (we recognize measurements from the same person thanks to a unique identifier). Though we believe that a good setting for w_{same} is $w_{same} = 0.75$, to analyze the influence

of its value on the computed results, in Section IV we show results obtained by varying the value of w_{same} in the set {0.5, 0.75, 1}. Note that this set comprises both an extreme value ($w_{same} = 0.5$), which essentially neglects similarities between different individuals, and the value $w_{same} = 1$, which essentially deletes the effect of w_{Ind} for all the similarities are weighted in the same way.

The normalized similarity measure is simple and effective on clinical data [109]. Moreover, it allows to avoid any dataset normalization/standardization, since the resulting similarity measure takes values in the continuous interval [0, 1].

b: SUBSAMPLING THE TRAINING WHEN HAVING UNBALANCED DATA

To cope with problem (b) and (c), that is to avoid the bias caused by data unbalancing and to reduce the high computational times and costs caused by searches in datasets with high cardinality, before applying knn-imputation we subsample the training set where the k_{impute} nearest neighbors must be searched for. The subsampled set is formed by sampling each class to select a smaller set of most representative points [46]. When opportunely designed, this procedure not only reduces computational costs, but it could also increase the classifier performances decreasing the risk of overfitting due to the preferential prediction towards the majority class in unbalanced datasets [56].

Precisely, we find the class C_{min} with the lowest cardinality ($|C_{min}|$, where $|S|$ is the cardinality of set S), and, for every other class, we sample n points, where $n=unbF * |C_{min}|$, and unbF is the maximum unbalancing factor, which limits the total number of points and the unbalancing in the training set.

The sampling method is one the following:

- nearestToMin: sample the n points that are the nearest to the points in C_{min}. Points are sampled by computing the pairwise normalized similarities between the points in C_{big} and the points in C_{min} and by taking the n points that are the most similar to some point of C_{min}.
- farthestFromMin: works in the opposite way than nearestToMin; it samples points that are the farthest from the points in C_{min}.

The afore-mentioned sampling methods have been described and tested in the work of [46]. Precisely, they have been used to form balanced training sets with limited cardinality for training a knn classifier for information extraction from the biomedical literature. Experimental results showed that the best performing sampling method is nearestToMin.

In this work (see Section III.A.3) we tested values of parameter unbF in the set {2, 3, 4} (see Section IV) to show its effect on the computed results.

c: Knn-IMPUTATION MODEL: AUTOMATIC MODEL SELECTION

As mentioned in Section II.A, the experimental results proposed in [44] showed that, when performing knn-imputation, the value $k_{impute} = 1$ should be avoided to guarantee robustness to noise; at the same time, k_{impute} should be set to low values to avoid biasing the data structure. Based on the afore-mentioned considerations and given that some parameters in our data are manually recorded and might therefore be quite noisy, our algorithm automatically chooses the best k_{impute} in the range [2, ..., 5].

Briefly, to perform automatic knn-imputation model selection, which regards the selection of both the training subsampling method and the k_{impute} value, our algorithm performs a grid search, which tries each of the eight combinations of training sampling method $sub \in$ {nearestToMin, farthestFromMin} and value of k_{impute}, $k_{imp} \in [2,\ldots,5]$, and chooses the one that maximizes the discriminative power of the imputed set.

Precisely, for each couple of values (sub, k_{imp}), the training set is subsampled by using method sub, and knn-imputation is performed by imputing each missing value as the average of the corresponding values in the k_{imp} nearest neighbors found in the subsampled training set.

After imputing all the values, the feature selection method presented in detail in the following Section III.A.2 is applied. This method selects the most representative subset of features and returns a mean p-value, P_{SelF}, which is inversely related to the discriminative power of the selected feature set SelF (the lower the value of P_{SelF}, the highest the discriminative power of the feature set, or, in other words, the inter-class separation provided by the feature set).

The automatically chosen training sampling method and value of k_{impute} are those that allow obtaining a selected feature set minimizing P_{SelF} (that is, maximizing the inter-class separation).

2) FEATURE SELECTION

Feature selection has the aim of choosing the so-called "right data" [108] by selecting the most informative (discriminating) features while discarding redundant information. This problem has been extensively studied in literature [110], [111] since it allows reducing the computational costs of the following algorithms, and it often improves the performance of classifiers, which are not misled by the (removed) uninformative or redundant information.

Informative features can be identified by applying statistical tests for measuring the difference in the class distribution of each feature.

Precisely, each feature is firstly analyzed through the Anderson-Darling's test [112] for detecting whether the feature is normally distributed against the alternative hypothesis that it is not normally distributed [113]. Features for which the computed p-value is $p<0.05$ (95% confidence interval) are normally distributed, while the others are considered as not normally distributed.

After this step, the Bartlett's test ([114], [115]) is applied on normally distributed features to check whether they are homoscedastic against the alternative hypothesis that they are heteroscedastic. Features for which the Bartlett's p-value is $p<0.05$ (95% confidence interval) are homoscedastic.

In case of not normal distributions or heteroscedastic normal distributions, the multivariate Kruskal-Wallis's test [116] is applied to check if the feature distribution in different classes have the same mean, against the alternative hypothesis that they have different means. On the other side, when normal homoscedastic distributions are treated, ANOVA [117] is the best statistical test for performing the same check.

After computing the p-values with Kruskal-Wallis or ANOVA, we keep only features for which the feature distribution in different classes have different means, that is, we keep the t_{SelF} features for which the p-value allows to discard the null hypothesis at 5% significance level, that is $p<0.05$ (95% confidence). The p-values of the retained (most discriminative) features are stored in a vector $P_{val} = [p_i]$, $i=1,\ldots,t_{SelF}$, where p_i is the p-value ($p_i < 0.05$) of the i^{th} selected feature, computed either by Kruskal-Wallis or ANOVA test.

The afore-mentioned statistical tests are effective in selective discriminative information but do not consider the inter-features relationships. Consequently, if two discriminative features are highly correlated, the redundant information is anyway included in the selected feature set. To remove such redundant information, we therefore compute the pairwise Pearson linear correlation coefficient between each pair of feature vectors and, for each couple of features obtaining an absolute, statistically significant ($p_{corr} < 0.05$ [118]) Pearson correlation coefficient, that is Pearson ≥ 0.5, we remove the less discriminative feature, that is the feature for which the corresponding p-value in P_{val} is higher.

In this way, we obtain the set of selected features, SelF, with cardinality $L = |SelF|$ and a vector of corresponding p-values extracted from P_{val}. In the following, we will refer to the mean of such p-values, as P_{SelF}, and we will consider it as the p-value of the (selected) feature set, which is inversely related to the discriminative power of the selected feature set.

3) THE MULTICLASS CLASSIFIERS

To deal with the multiclass classification problem we use ECOC models [83], which have been shown to improve the performance of multiclass classifiers by combining them into an ensemble their binary versions [84]–[86].

To confirm this thesis, given the (imputed) training set (built by using the selected feature set SelF, see Subsection III.A.2), we firstly subsample it by using the training subsampling method (nearestToMin or farthestFromMin, see Subsection III.A.1) automatically selected during the knn-imputation phase (see Subsection III.A.1). This allows to work with a ''more balanced'' and smaller training set T_{RED}, containing, for each class, $unbF*|C_{min}|$ training points.

Using T_{RED}, we compared the performance obtained by a simple multiclass knn classifier [59], [60] referred to as *knn* in the following, to those obtained by three ECOC models differing for the employed training set and binary learners. Precisely, two ECOC models, referred to as *knn-ECOC* and *svm-ECOC* in the following, exploit, respectively, knn and support vector machines (svms) as base classifiers, and are trained on T_{RED}.

The third ECOC model, referred to as svm_{COST}*-ECOC* in the following, uses cost-sensitive svms [81] as binary classifiers, and is trained on the whole unbalanced training set. In particular, following the approach showed in [81], in the experiments presented here, if the positive class is c_+ and the negative class is c_-, when a point of class c_+ is wrongly assigned to class c_-, we set the misclassification cost $\theta_+ = (|c_-|)/(|c_+|)$. On the other side, the cost of negative misclassifications is $\theta_- = (|c_+|)/(|c_-|)$. Essentially θ_+ and θ_- weight more misclassifications of points belonging to the smallest class.

Note that all the classifiers are based on the computation of the distance between points. We therefore defined a feature-weighted normalized distance, $DistW_{WNorm}$, which extends the weighted normalized distance $Dist_{WNorm}$ defined in Section III.A.1 (equation 4). Precisely, when computing $DistW_{WNorm}$ each feature vector F_f ($f=1,\ldots,|SelF|$) is weighted by a weight $w_f = 1\text{-}p_f$, where p_f is the p-value computed for feature f (see Section III.A.2); the weight w_f is directly proportional to its discriminative power.

Essentially, the distance between two points x(1) and x(3) is computed as:

$$DistW_{WNorm}(x(1), x(2)) = w_{Ind} * \frac{\left(\sum_{f=1}^{L} \frac{|x_f(1)-x_f(2)|*w_f}{\max(f)-\min(f)}\right)}{L} \quad (6)$$

where each point x(i) is an L-dimensional vector ($L = |SelF|$) composed by the values of the L selected features, and we recall that w_{Ind} is the weight used to lower the distances between the same individual. Essentially, the f^{th} element w_f ($f=1,\ldots,|SelF|$) of the weight vector influences the difference between the corresponding f^{th} elements in the two points. The efficacy of this distance measure, e.g. when compared to the classical Euclidean distance, is shown by the tsne decomposition plots [119] reported in Appendix A.

A final note to be done is that each classification model is defined by a set of parameters, which are automatically selected by the Bayesian Optimization procedure [120], [121], which attempts to minimize the classification error computed by assessing the trained classifier with an internal Leave-One-Out cross-validation (LOOCV). Precisely, the parameters defining each classifier model are the following:

- the knn-based classifiers are defined by the number of nearest neighbors (k);
- the svm-based classifiers are defined by the employed kernel function, K(x,y), and by its parameters;
- the ECOC models are defined by the coding design.

Moreover, though the feature set SelF has been composed by selecting discriminative and not redundant features, we decided to perform a greedy search to choose, among features in SelF, the minimal set of features that obtains the best classification performance. To this aim, we employ a

TABLE 1. Top: manual parameters of our model. Bottom: hyper-parameters automatically chosen by bayesian optimization. On the left we report the (hyper-)parameter name, while the hyper-parameter space is reported on the right. The bottom table is divided into blocks, according to the automatic model selection criteria.

Manual Parameters	
Parameter	Parameter Space
w_{same}	{0.5, 0.75, 1}
unbF	{2, 3, 4}
Automatically selected Hyper-Parameters	
Parameter	Parameter Space
Hyper-parameters defining the knn-imputation model; automatic model selection by maximizing the informativeness of the imputed (selected) feature set	
sampleM (training set subsampling method, see Section III.A.1)	{nearestToMin, farthestFromMin}
k_{impute} (neighborhood size)	[2, …,5]
Hyper-parameters defining classifier models; automatic model selection performed by Bayesian optimization minimizing classification loss by LOOCV	
Coding scheme for ECOC models	{'onevsone', 'allpairs', 'binarycomplete', 'denserandom', 'onevsall', 'ordinal', 'sparserandom', 'ternarycomplete' }
k (knn neighborhood size)	[1, …, N/2]
K(x,y) (svm kernel)	{linear (lin), radial basis function (rbf), polynomial (poly)}
if K(x,y) == poly poly order	[2, …, 6]
if K(x,y) == rbf kernel scale	[$1e^{-3}$, …, $1e^{3}$]
Automatic selection by greedy search	
$\hat{t}$ (number of most discriminative features used for classification)	[1, …, L] (L = |SelF|)

greedy method which sorts the selected features according to their p-values (from the smallest to the highest), and, for each classifier, chooses the first $\hat{t}$ features which allow to minimize the classification error computed through LOOCV. In our multiclass problem, the classification error on the training set is simply computed as the ratio of the number of wrongly classified points divided by the number of points, regardless of their label.

For clarity of explanation, Table 1 lists all the parameters used in our algorithm, their optimizing set. Note that, all the parameters but two (w_{same} and unbF) are automatically set in order to achieve the best classification performance.

Regarding the two manually set parameters, in the experimental results of Section IV we show their influence on the results achieved by either knn-based classifiers or svm-based classifiers. In this way we may provide our suggestions for setting their values.

B. TREATING NOVEL MEASUREMENTS TO PROVIDE SUGGESTIONS BASED ON COUNTERFACTUAL EXPLANATIONS

After using the historical (training) data to form its artificial intelligence, the DTs are ready to process new data coming from their physical twin to estimate their conditions and provide suggestions for improvement (SmartFit Box in Figure 2).

Precisely, when the physical twin (or an expert) provides a novel vector of measurements, x_{novel}, that is a set of t-dimensional points whose fitness score is unknown, the first step regards data imputation, if some data is missing. In this case, missing data are imputed by using knn-imputation, which works on the subsampled training data by searching for k_{impute} neighbors (where the method for training set subsampling and the value of k_{impute} are chosen as described in the previous Section).

Next, x_{novel} is described by using only the minimal set of features chosen while training, and its fitness score is estimated by one of the multiclass classifiers trained as described in Section III.A.3 (the experimental evaluation reported in Section IV allows choosing the best performing learner).

Finally, the training data is used to compute prescriptive suggestions for improving the fitness status of the physical twin, based on the computation of the so-called counterfactual explanations.

In our work, after predicting the fitness score for a novel measurement x, the DT computes counterfactuals in a way similar to that used in [95].

Precisely, let $\{sc_{min}, \ldots, sc_{max}\}$ be the sorted fitness scores, where sc_{min} corresponds to bad performance, while sc_{max} corresponds to good performance, and suppose that the predicted score for x_{novel} is sc.

The DT provides suggestions to get higher fitness scores, that is, to get fitness scores $sc_{desired} = \{sc+1, \ldots, sc_{max}\}$.

To this aim, for each desired score $sc_{desired}$, the DT find the nearest point, x_{Wish}, whose fitness score is $sc_{desired}$, where the distance is the $DistW_{Norm}$ used by the multiclass classifiers (see Section III.A.3). Using x_{Wish} as the starting point, a dichotomic procedure is applied to find the point, x_{DESIRE}, which is the nearest to x_{novel} in the segment joining x_{novel} and x_{Wish} and whose predicted fitness score is $sc_{desired}$.

Precisely, setting $x1{=}x_{novel}$ and $x_{DESIRE} = x_{Wish}$, the dichotomic procedure computes the point $x_{Med} = (x_1{+}x_{DESIRE})/2$, which is the point at half distance from x_1 and x_{DESIRE}. The id of x_{Med} is set to be equal to that of x_{novel} (we recall that the id is used in $DistW_{Norm}$) and the score sc_{Med} of x_{Med} is then predicted. If $sc_{Med} = sc_{desired}$, then $x_{DESIRE} = x_{Med}$, otherwise $x_1 = x_{Med}$. The iterative process stops when the distance between x_1 and x_{DESIRE} is less than 10^{-10}, that is when $DistW_{Norm}(x_{Desire}, x_1) < 10^{-10}$,

meaning that the points are similar, or when a maximum of 10000 iterations has been performed. Note that, in all our experiments, the dichotomic procedure always stopped because the first condition was met (the number of iterations was never bigger that 2051).

Once x_{DESIRE} has been found, suggestions for improving the outcome are easily computed as the difference between x_{DESIRE} and x_{novel} and are shown to the DT user (which might be the physical twin or an expert), as text. An example of output of the prescriptive algorithm is shown at the end of Section IV.

IV. EXPERIMENTAL RESULTS

In this Section we describe the experimental evaluation of a team of DTs used by trainers for supervising the activity and fitness level of a team of 11 athletes.

A. MATERIALS

The athletes' data we treat in this work are described at the end of Section II.B.

For each of the 11 athletes, we have a set of 10 labeled vectors, each of which composed by $t = 66$ measurements describing the athlete's behavior for three days. Each vector has been labeled by the trainer according to the athlete's performance during the training session following the three days of measurements.

To choose the best input-data representation, in this Section we show the experimental comparison we performed by coding the 66-dimensional vectors in three different ways, thus obtaining three different input datasets.

Precisely, the used datasets are listed in the following:

1) Dataset Y1 contains only the 22 measurements recorded in the last day before training. This dataset has been created to check whether it is the last day before training that influences the athlete's outcome.
2) Dataset $Y3_{AVG}$ is created by averaging each measurement over the three days. This dataset contains 22 features, each being the average of the same measurement over the three days (e.g. average number of steps, average calories, average activity, and so on). This dataset is based on the consideration that the average behavior in the preceding days is the factor influencing the outcome.
3) Dataset Y3 is composed by all the feature vectors, and it is therefore a 66-dimensional dataset containing, for each athlete, the detailed measurements of the three days.

We recall that our data may contain missing values. Though we tackle this problem by using knn-imputation, before processing each dataset, we decided to remove those points which have a number of missing values bigger than 1/3 of the dataset dimensionality.

In this way, we deleted points from dataset Y1 and dataset Y3, while from dataset $Y3_{AVG}$ we removed no points because the average over the three days is computed by neglecting missing values; in other words, only the not-missing data is used to compute each mean.

TABLE 2. Dataset Cardinality and Class Distribution After Points Deletion. N_* refers to the cardinality of the whole dataset.

Dataset	N_{poor}	N_{medium}	N_{good}	N_*
Y1	18	70	19	N_{Y1}=107
$Y3_{AVG}$	19	71	20	N_{Y3AVG}=110
Y3	18	70	20	N_{Y3}=108

Precisely, in Table 2, for each dataset, we show the dataset cardinality and the class distribution after deleting points with too many missing values. Figures 3-A, 3-B, and 4 respectively show the histogram of missing values in dataset Y1, $Y3_{AVG}$ and Y3, before (purple color) and after (yellow) deletion (obviously for dataset $Y3_{AVG}$ the two histograms overlap).

B. ANALYSIS OF COMPUTED RESULTS

Given one of the afore-mentioned datasets, to test the algorithm, we apply a LOOCV procedure. Precisely, we consider each sample point as a novel (test) sample, for which prediction and suggestions must be computed. All the other samples are considered as historical (training) samples, which are used to form the DT's artificial intelligence as described in Section III.A. For each test sample, the DT is trained on (learns from) the historical (training) data, and it is then used to classify the test sample (with one of the classifiers described in Section III.A.3), and to compute advices aimed at obtaining higher scores as detailed in Section III.B. Note that, during each training process, Bayesian optimization, which is used to perform automatic model selection (see Section III.A.3), minimizes the classification loss computed by applied an internal LOOCV on the training set.

When the LOOCV has iterated over all the points in the dataset, the classifiers are assessed in terms of classification Loss. Precisely, the Loss is defined as:

$$\text{Loss} = \frac{\sum_{n=1,..,N} \left|\hat{y}(n) - y(n)\right|}{N} \tag{7}$$

where $\hat{y}(n)$ is the label predicted for the n^{th} point and $y(n)$ is its ground truth label. Note that this loss function weighs the error by considering the L1 distance (amount of error in classification) between the predicted class and the actual class; in this way, errors where measurements with excellent fitness are wrongly classified as poor (or viceversa) are weighted more than errors where measurements with excellent fitness are wrongly classified as average (or viceversa).

The classifier performance provides an indirect evaluation of the advices, which are computed after classification. Indeed, the changes needed for improving the outcome are computed on the basis of the classifier's predictions. For this reason, the lowest the classification error is, the more trustable the computed advices are.

Before analyzing the DT performance, we recall that, though the proposed DT needs some parameters to be set, only two of them must be manually chosen, while the others

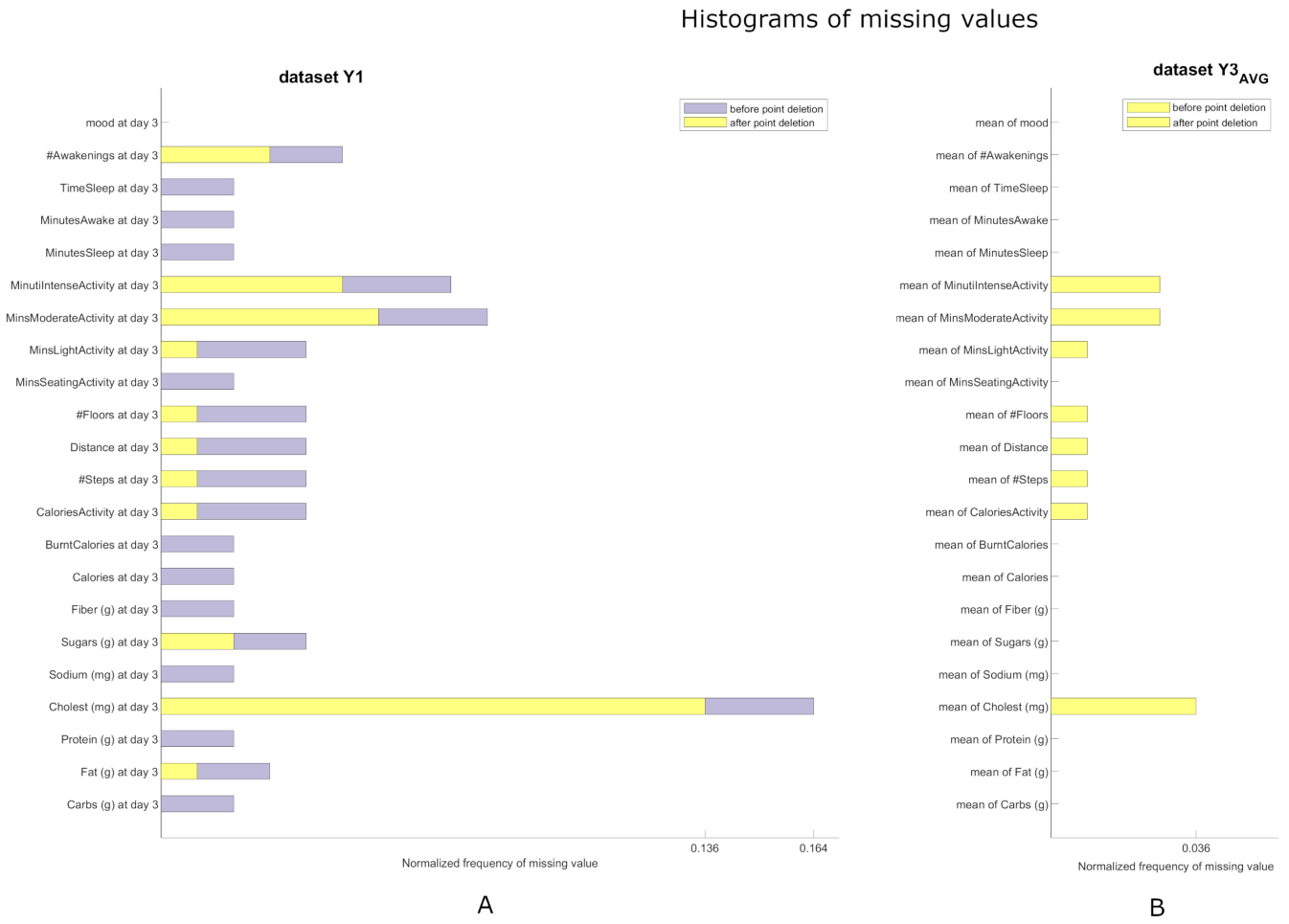

FIGURE 3. 3-A, 3-B. Histograms of missing values before (purple) and after (yellow) deletion of points with more than 1/3 of missing values. a: deletion of missing samples is performed on dataset y1. b: no points are deleted from dataset $Y3_{AVG}$. Note that the features in $Y3_{AVG}$ are mean values, which have been computed by neglecting missing values. For this reason, $Y3_{AVG}$ contains less missing values when compared to Y1 and Y3 datasets.

are automatically selected in order to maximize the classification accuracy on the training set. The two manually chosen parameters are:

- the ratio unbF, which is used for the training point sampling;
- the value of weight w_{same}, which is used to lower the distance between points describing measurements from the same athlete.

Moreover, in Section III.B.3 we have described 4 different classifiers that may be used by the DT to compute predictions and subsequent advices, while in Subsection IV.A we have described three different Y1, $Y3_{AVG}$, Y3 datasets, which may be used to code the input-data.

In this Section, for each of the three input-datasets (Y1, $Y3_{AVG}$, Y3), we initially report the knn-imputation results achieved on the three datasets after running 3x3 different LOOCV experiments, by varying the values of parameter unbF in the set {2, 3, 4}, the value of parameter w_{same} in the set {0.5, 0.75, 1.0}. The analysis of such results, not only provides hints about the discriminative power of the imputed sets, but also shows that the system is stable with respect to different settings of parameters w_{same} and unbF, whose choice is therefore not critical.

Note that, to check whether knn-imputation is relevant to increase performance, while running the knn-imputation tests we also tested the case $k_{impute} = 0$, which means that knn-imputation should not be performed and the algorithm should deal with missing data. To this aim, we simply modified the normalized similarity measures between points, which are $Dist_{WNorm}$ (Section III.A.1, equation 4) and $DistW_{WNorm}$(Section III.A.3, equation 5), in order to neglect the differences among corresponding feature values when one or both of them are missing.

Next, we compare the classification performance achieved by the four multiclass classifiers and show examples of computed advices. All the comparative evaluations reported in this Section are supported by the Wilcoxon signed rank test at the 0.01 significance level.

During knn-imputation, the DT chooses the training subsampling method and the value of parameter k_{impute} that

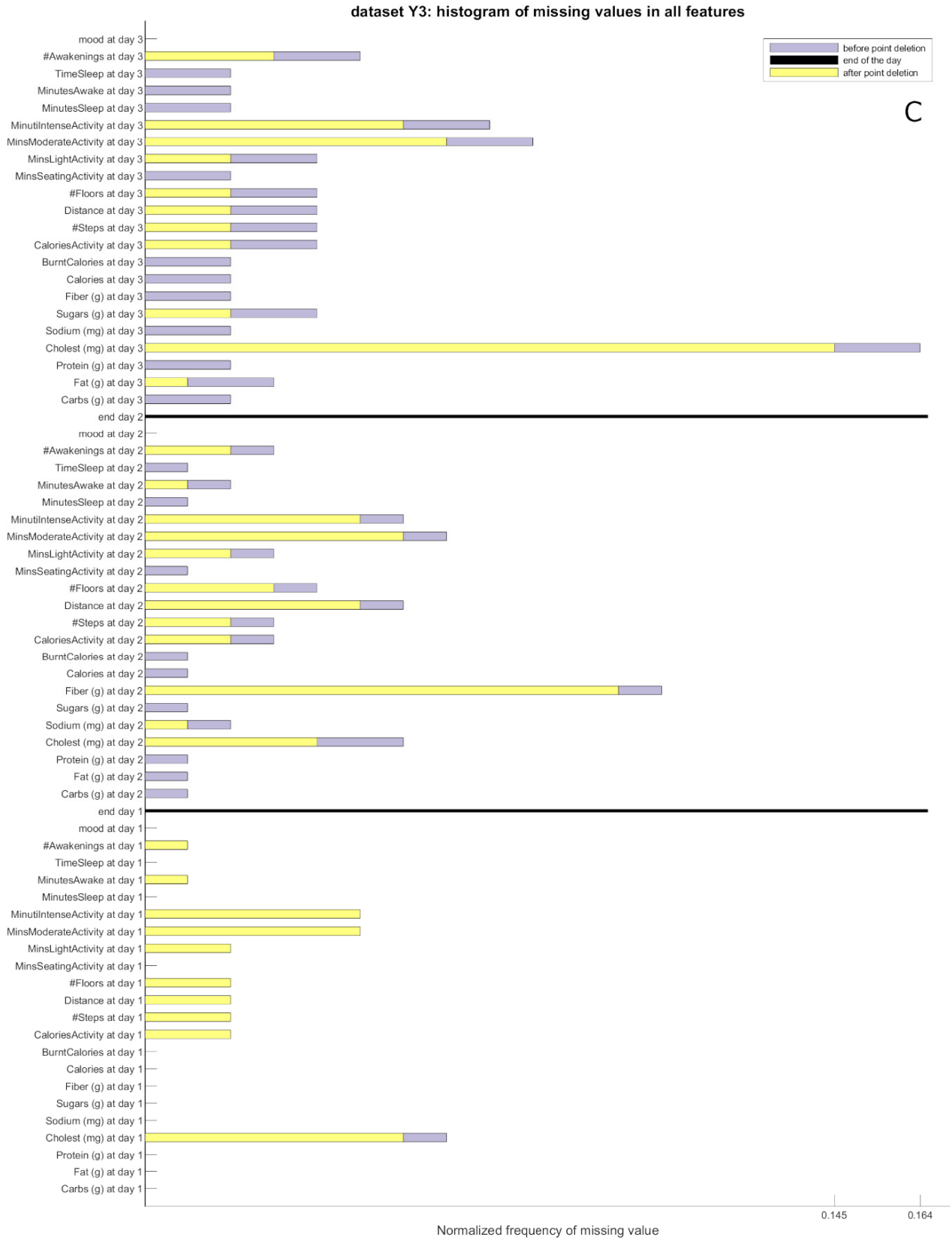

FIGURE 4. Deletion of points with too many missing data is performed on dataset y3. in this case, the black line identifies the end of one day.

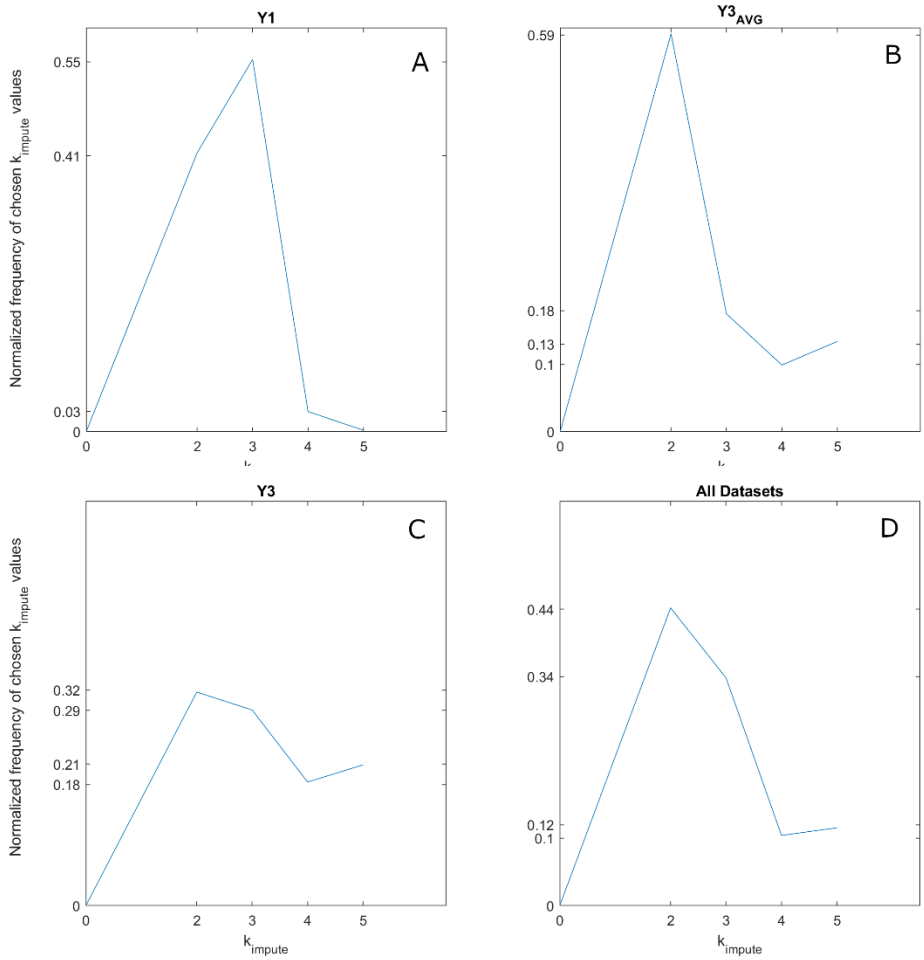

FIGURE 5. The first three plots show, for each dataset, the histogram (e.g. the normalized frequency) of chosen k_{impute} values. The last plot shows the histogram of all the k_{impute} values. When k=0, knn-imputation is not performed.

allows creating the most informative imputed dataset, composed by a set of most discriminative features.

Regarding the training subsampling methods (nearestToMin or farthestFromMin, see III.A.1.b), our results are different from those reported in [46]. Indeed, in our experiments, for the 31.4% of the times nearestToMin and farthestFromMin achieved the same P_{SelF} value (that is the mean p-value of the selected features set), on the 22.4% of the times nearestToMin was chosen for being the subsampling method which minimize the P_{SelF} value, while the remaining 46.2% of the times farthestFromMin was the optimal choice. This suggests that, in our problem, farthestFromMin is the best performing subsampling method.

Before analyzing the distributions of the k_{impute} values chosen during the LOOCV, we note that the value of k_{impute} obviously affects the discriminative power of the imputed set. Indeed, if the value of k_{impute} is such that all the k_{impute} neighbors (or at least the majority of them) belong to the class of the point to be imputed, the resulting imputed point may increase the class separation since the imputation of its missing values would ''move'' the points towards points in its same class. On the contrary, if the majority of the k_{impute} neighbors belong to classes that differ from that of the point to be imputed, the class separation after imputation could be diminished because that point would be ''moved'' towards other classes. Therefore, the automatically chosen k_{impute} values is related to the size of the neighborhoods so that the majority of neighborhoods are composed of points belonging to the same class.

In Figure 5, we show the histograms of the chosen k_{impute} values. The A, B, C plots show, for each dataset, the histogram of the k_{impute} values chosen during all the LOOCV repetitions, that is by all the couples (w_{same}, unbF). The D plot summarizes the preceding (per-dataset) histograms by showing the histogram of all k_{impute} values automatically chosen over all the datasets and all the LOOCV iterations. We recall that the value $k_{impute} = 0$ refers to the case where no knn-imputation is performed. In all the cases, the mostly chosen k_{impute} values are $k_{impute} > 0$ and $k_{impute} \leq 3$, thus confirming the results reported in [44]. Moreover, we note that the

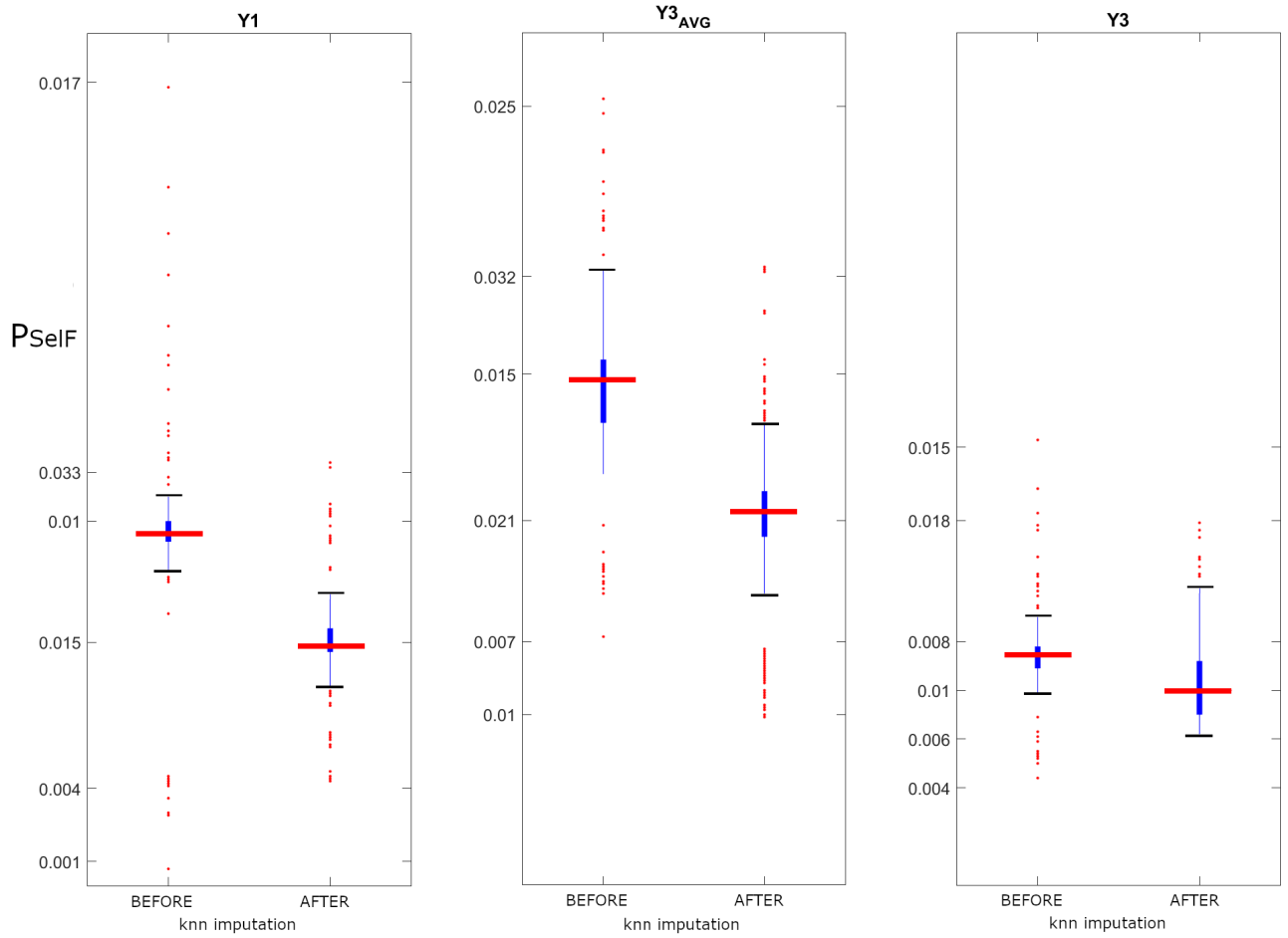

FIGURE 6. Comparison between the discriminative power of each dataset, when feature selection is applied before and after knn-imputation. Each plot shows, on the left (BEFORE), a boxplot visualizing the distributions of the P_{SelF} values achieved when applying feature selection before knn-imputation. The boxplot on the right (AFTER) shows the distribution of the P_{SelF} values computed by applying feature selection after knn-imputation.

value $k_{impute} = 0$ has never been chosen, thus suggesting that knn-imputation effectively helps the following prediction task by increasing the discriminative power of the selected feature set. To confirm this hypothesis and to investigate the influence of knn-imputation on the discriminative power of the input dataset, we observed how the discriminative power of each dataset changes after imputation. To this aim, for each iteration of the LOOCV, we have run feature selection (on the training set) before and after knn-imputation. Recalling that each application of feature selection results in a P_{SelF} value inversely related to the discriminative power of the dataset under analysis (see Section III.A.2), for each dataset, we used the Wilcoxon signed rank test at the 0.01 significance level to compare the two distributions of the P_{SelF} values computed before and after knn-imputation[6].

The before (BEFORE) and after (AFTER) P_{SelF} distributions computed over the three datasets are visualized by the boxplots shown in Figure 6. In particular, each boxplot shows: the median P_{SelF} value of the distribution with a tick red horizontal line. The 25^{th} percentile (q1) and the 75^{th} percentile (q2) are the limits of the blue tick rectangle, while the horizontal black segments delimit the distribution spread. Precisely, the distribution spread is composed by points p such that $q1 - 1.5 \times (q3 - q1) \leq p \leq q3 + 1.5 \times (q3 - q1)$. All the other points, visualized through red dots, are considered as outliers.

The visual comparison provided by the boxplots highlight that, on all the three datasets, knn-imputation effectively fills missing values by increasing the inter-class separations. For all the three datasets, the Wilcoxon test confirms the differences in the distributions off the P_{SelF} values computed before and after knn-imputation. From the boxplots in Figure 6 it can also be noted that, both before and after knn-imputation, dataset $Y3_{AVG}$ seems to be the less discriminative among the three datasets. Applying the Wilcoxon signed rank test (at the 0.01 significance level) to compare the means of the P_{SelF} distributions characterizing the three datasets

[6]Each distribution contains all the P_{SelF} values computed by all the LOOCV procedures, obtained by varying w_{same} in the set $\{0.5, 0.75, 1\}$ and unbF in the set $\{2, 3, 5\}$.

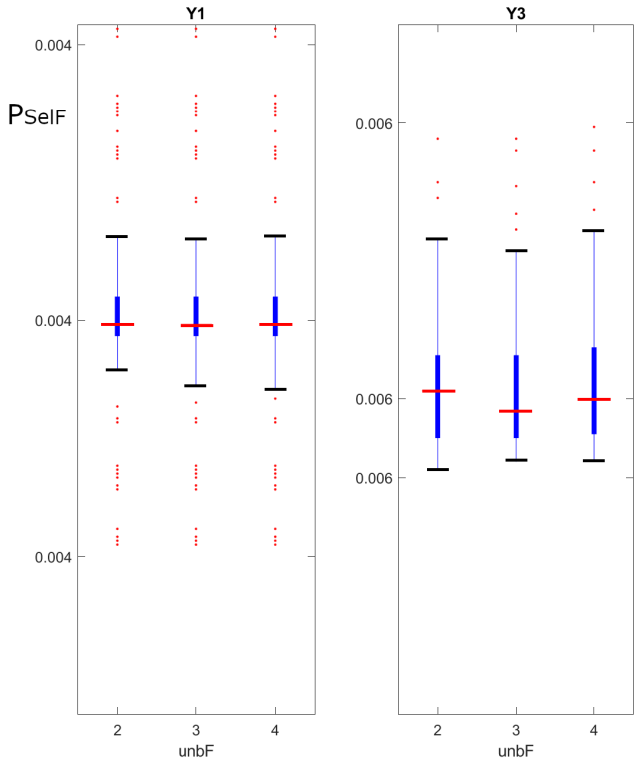

FIGURE 7. Distributions of P_{SelF} for varying values of parameter unbF.

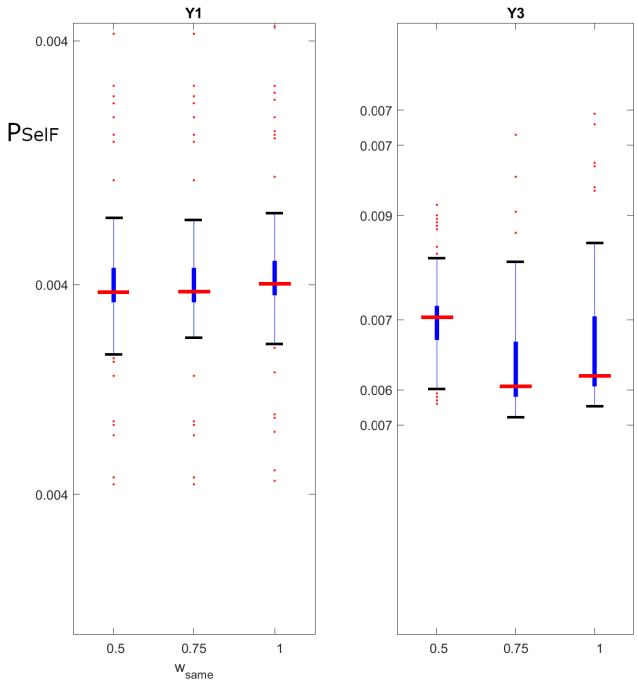

FIGURE 8. Distributions of the P_{SelF} values for varying values of parameter w_{same}.

before and after knn-imputation, we could see indeed that the visible differences are statistically significant. Precisely, dataset $Y3_{AVG}$ is characterized by the highest mean of the P_{SelF} distribution, both before and after knn-imputation. Precisely, the mean of the P_{SelF} values computed over dataset $Y3_{AVG}$ before (0.0208) and after (0.0154) knn-imputation is 1.43 and 1.55 times bigger than the mean P_{SelF} achieved over Y1 before (0.0145) and after (0.0099) knn-imputation, and 2.19 and 1.93 times bigger than the mean P_{SelF} achieved over Y3 before (0.0095) and after (0.008) knn-imputation. This low discriminative power may be due to the fact that the average over the three days has neglected the missing values.

Essentially, some points have been coded by averaging over three values, while some other points are computed as the average of one or two values. This variability may be the cause of disturbance.

Given its lower discriminative power even after imputing its missing values, we have discarded dataset $Y3_{AVG}$ and, in the following, we report only results achieved with the Y1 and the Y3 datasets.

Next, to check whether differing values of parameters unbF and w_{same} produce a great change in the distribution of the P_{SelF} values, we also analyzed the distributions of P_{SelF} values computed when the two parameters change their values. Again, we used Wilcoxon signed rank test (at the 0.01 significance) to detect statistically significant differences in the means of the distributions.

In Figure 7 and Figure 8, the distributions of P_{SelF} values for varying values of parameters unbF and w_{same} are shown.

When considering parameter unbF (see Figure 7), in both datasets the Wilcoxon test detected no significant differences in the means of the P_{SelF} distributions. Therefore, the automatic selection of the knn-imputation model is robust with respect to differing values of parameter unbF. Anyhow, we decided to set unbF=3, which is a good tradeoff between the requirement of having a limited unbalancing factor and the need of having a training set with a number of samples able to capture the population variability.

When considering parameter w_{same} (Figure 8), in dataset Y1 the Wilcoxon test found no statistically significant differences. In Y3, as expected, the Wilcoxon test found statistically significant differences when comparing the distribution obtained with $w_{same} = 0.5$ to the other two distributions, while it found no statistically significant differences when comparing the distributions achieved when $w_{same} = 0.75$ and $w_{same} = 1$. We expected these results since we believe that the setting $w_{same} = 0.5$ is extreme. This motivates our choice of setting, $w_{same} = 0.75$. Anyhow, in Appendix B we report the classification results we obtained with different values of parameter w_{same}.

To finally choose the best input dataset and the best performing classifier, we have compared the classification losses achieved by the four classifiers when applying LOOCV to the Y1 dataset and the Y3 dataset.

We recall that, through Bayesian optimization, the team of DTs choose the classifiers' models, while the number of most discriminative features to be used for classification is defined by a greedy search. Since we perform LOOCV testing, we have $N*$ chosen classifier models and feature set dimensionalities (where $N*$ is the number of samples

TABLE 3. TOP: classifiers loss computed through loocv on Y1 (top table) and Y3 (bottom table) datasets.

Dataset Y1. On all the LOOCV iterations, most of the times the selected feature set, SelF, has dimensionality equals 16 (-2, +3)				
classifier	**Loss**	coding	parameters	# Feat
knn	**0.08**	-	k = 2	11 (-2, +4)
knn-Ecoc	**0.04**	onevsone	k = 2	8 (-3, +3)
svm-Ecoc	**0.03**	onevsall	Poly (order 4)	16 (-2,+4)
svm_{COST}-Ecoc	**0.02**	onevsone	Gaussian (scale 2.23)	12 (-4,+3)

Dataset Y3. On all the LOOCV iterations, most of the times the selected feature set, SelF, has dimensionality equals 39 (-1, +2)				
knn	**0.07**	-	k = 2	10 (-1, +8)
knn-Ecoc	**0.04**	onevsall	k = 2	28 (-7, +5)
svm-Ecoc	**0**	onevsone	Poly (order 4)	15 (-1, +5)
svm_{COST}-Ecoc	**0**	onevsone	Poly (order 2)	13 (-1, +7)

per dataset, listed in Table 2). Table 3 reports for dataset Y1 (top) and dataset Y3 (bottom): the mean classification losses achieved by each classifier (highlighted with bold text), the mostly chosen coding scheme used by the ECOC models (column coding), the mostly chosen (over all the LOOCV repetitions) neighborhood size in case of knn-based classifiers or the mostly chosen svm-kernels in case of svm-based classifiers (column parameters), the mostly chosen number of features used for classification (# Feat) and in round brackets the minimum and maximum offset.

When analyzing and comparing the results achieved by knn classifiers (knn vs knn-ECOC), we firstly note that the statistically significant difference in performance on both the Y1 and the Y3 dataset, confirm results reported in in [83]–[86], where authors showed that ECOC models are generally better performing models when dealing with multiclass classification tasks. Regarding the automatic model selection performed on knn-based models we note that, on all the $N_{Y1}+N_{Y3} = 107 + 108$ LOOCV repetitions, the chosen neighborhood size has been: $k = 1$ for the 0.02 (5 times on 107+108), k=3 for the 0.1 of the times (21/(107+108)), and $k = 2$ for the remaining of the times 0.88 of the 107+108 times. Note that such results are coherent with the neighborhood size mostly chosen by the automatic knn-imputation model selection.

Regarding svm-classifiers, we firstly note that the cost-sensitive svm classifier is the best performing classifier in Y1, while in Y3 there is no difference between its performance and that of svm-Ecoc (both of them committed no error). However, on all the LOOCV repetitions svm-Ecoc generally chooses an higher number of features, thus requiring higher computational time for training. Moreover, svm_{COST}-Ecoc has the advantage of viewing all the training set and therefore covering all the training set variability. Given such considerations, when using svm models in our unbalanced problem, we believe that weighting the classification cost might be a better choice than subsampling the training set [70]. Regarding the chosen kernel, the Bayesian search has almost never chosen the linear kernel (on the 0.8% of the cases), while, in the remaining cases, the Bayesian kernel has selected the Gaussian kernel in the 6% of the cases, and the polynomial kernel for the 93.2% of the cases. Precisely, in both the datasets, the polynomial kernels of order 4 are the mostly selected for svm-Ecoc, while svm_{COST}-Ecoc has favoured the polynomial kernel of order 2.

TABLE 4. Results achieved by the classifiers when knn-imputation is avoided and the data contains missing data. TOP: classifiers loss computed through loocv on Y1 (top table), which contains the 10.57% of missing data and Y3 (bottom table) datasets, which contains the 25.51% of missing data.

Dataset Y1. On all the LOOCV iterations, most of the times the selected feature set, SelF, has dimensionality equals 6 (-1, +5)				
classifier	**Loss**	coding	parameters	# Feat
knn	**0.33**	-	k = 43	5 (-1, +1)
knn-Ecoc	**0.25**	onevsone	k = 31	4 (-0, +2)
svm-Ecoc	**0.45**	onevsone	Gaussian (scale 13.44)	6 (-1,+0)
svm_{COST}-Ecoc	**0.41**	onevsone	Poly (order 6)	6 (-1,+0)

Dataset Y3. On all the LOOCV iterations, most of the times the selected feature set, SelF, has dimensionality equals 17 (-3, +6)				
knn	**0.25**	-	k = 49	11 (-1, +4)
knn-Ecoc	**0.21**	onevsone	k = 47	15 (-5, +2)
svm-Ecoc	**0.39**	onevsone	Poly (order 6)	17 (-3, +0)
svm_{COST}-Ecoc	**0.37**	onevsone	Poly (order 6)	11 (-1, +3)

One final note is that the only coding schemes chosen by Bayesian optimization for the Ecoc models have been the 'onevsall' or 'onevsone' coding schemes.

Appendix B shows results achieved for all the values of $w_{same} = \{0.5, 0.75, 1\}$. The reported results show that, while knn based classifiers achieve better performance when $w_{same} \geq 0.75$, both the svm-based classifiers, and particularly the svm_{COST}-Ecoc classifier, are robust with respect to different values of parameter w_{same}.

Note that, to objectively assess whether knn-imputation is effectively helpful in improving the classifier performance, we run the classification tests by avoiding the imputation of missing data. In Table 4 for dataset Y1 (top) and dataset Y3 (bottom): the mean classification losses achieved by each classifier (highlighted with bold text), the mostly chosen coding scheme used by the ECOC models (column coding), the mostly chosen (over all the LOOCV repetitions) neighborhood size in case of knn-based classifiers or the mostly chosen svm-kernels in case of svm-based classifiers (column parameters), the mostly chosen number of features used for classification (# Feat) and in round brackets the minimum and maximum offset.

The achieved results confirm that knn-imputation has the effect of improving performance by decreasing the

classifier loss. We believe that these results are a direct consequence of the fact that the selected feature set not only has a lower cardinality, but it is also composed by less discriminative features (that is, their P_{SelF} is lower, see Figure 6).

Moreover, the observation of the selected classifier models evidences the increased model complexity with respect to the classifier models selected for the imputed data (see Table 3). Indeed, the number of neighbors needed by the knn-based classifiers is near to N/2, while the kernels chosen by the svm-based classifiers are either the polynomial kernel with order 6, which is the maximum admissible polynomial order, or the Gaussian kernel with a scale which is ten times bigger than that chosen for the svm-based kernels working on imputed data. This behavior is generally happening when dealing with classification problems where the classes to be discriminated are highly overlapping.

For what regards the comparison between different classifiers' models, we note that, the knn-based classifiers are the better performing classifiers, which suggest that such classifier models are more robust with respect to missing data. Again, the knn-Ecoc models perform better that the knn model, thus providing a further confirmation of what reported in [83]–[86]. On the other side, when comparing the performance achieved by the svm-based classifiers we note that the cost-sensitive models confirm their superior ability to deal with unbalanced data.

Regarding the coding scheme chosen for ECOC models, we noted that the 'onevsone' scheme has been preferred most of the times.

Regarding the computed advices, in the following we show examples of suggestions computed for improving an athlete's (predicted) outcome.

In this first example, the athlete's data belong to dataset Y1 (which contains 22 dimensional vectors), and the two parameters w_{same} and unbF are set to $w_{same} = 0.75$ and unbF = 3. The DT exploits the best performing classifiers knn-ECOC and svm_{COST}-ECOC. The athete's data, in this case, has no missing value.

After performing knn-imputation on the historical data, the selected feature set contains 18 features, which are (in increasing order of P_{SelF}): 'mood at day 3', 'Minutes of Moderate Activity at day 3', 'Carbohydrates (g) at day 3', 'Minutes of Rest at day 3', 'Minutes Intense Activity at day 3', 'Burnt Calories at day 3', 'Number of Awakenings at day 3', 'Proteins (g) at day 3', 'Distance (km) at day 3', 'Minutes of Sleep at day 3', 'Minutes Awake at day 3', 'Fats (g) at day 3', 'Sugars (g) at day 3', '# Floors at day 3', 'Calories burnt with Activity at day 3', 'Number of Steps at day 3', 'Minutes Light Activity at day 3', 'Fibers (g) at day 3'.

During learning with knn-ECOC, Bayesian optimization has chosen a neighborhood size k = 2, while the greedy algorithm has chosen to use the first five most discriminative features for classification.

During learning with svm_{COST}-ECOC, Bayesian optimization has chosen the linear kernel, while the greedy algorithm has chosen to use the first most discriminative 12 features for classification.

Since the athlete's score is (correctly) classified as poor by both the classifiers, required changes are computed to increase the athlete's score from poor to medium, and then from medium to good.

Suggestions are visualized in the text format shown in Figure 9.

---

compute suggestions with classifier: knn Ecoc

Athlete 7, measurement 9: you will get score poor

To get vote medium you must:
decrease Minutes of Moderate Activity at day 3 of 14 units
increase Carbohydrates (g) at day 3 of 3 units
increase Minutes of Rest at day 3 of 16 units
decrease Minutes of Intense Activity at day 3 of 107 units

To get vote good you must:
increase Minutes of Moderate Activity at day 3 of 12 units
decrease Carbohydrates (g) at day 3 of 4 units
increase Minutes of Rest at day 3 of 229 units
decrease Minutes of Intense Activity at day 3 of 92 units

compute suggestion with classifier: svm_{COST}-Ecoc

Athlete 7, measurement 9: you will get score poor

To get vote medium you must:
decrease Minutes of Moderate Activity at day 3 of 14 units
increase Carbohydrates (g) at day 3 of 3 units
increase Minutes of Rest at day 3 of 16 units
decrease Minutes of Intense Activity at day 3 of 107 units
decrease Burnt Calories at day 3 of 1376 units
increase # Awakenings at day 3 of 4 units
increase Proteins (g) at day 3 of 1 units
decrease Distance (km) at day 3 of 6 units
increase Minutes of Sleep at day 3 of 118 units
increase Minutes Awake at day 3 of 17 units

To get vote good you must:
increase Minutes of Moderate Activity at day 3 of 24 units
increase Carbohydrates (g) at day 3 of 6 units
increase Minutes of Rest at day 3 of 269 units
decrease Minutes of Intense Activity at day 3 of 143 units
decrease Burnt Calories at day 3 of 919 units
increase # Awakenings at day 3 of 11 units
decrease Proteins (g) at day 3 of 7 units
decrease Distance (km) at day 3 of 7 units
increase Minutes of Sleep at day 3 of 141 units
increase Minutes Awake at day 3 of 11 units
decrease Fats (g) at day 3 of 2 units

FIGURE 9. Suggestions for athlete with score poor. The computed suggestions are shown in text format.

When the same athlete is coded by using data in dataset Y3, which is 66 dimensional, the number of required changes is greater. This is because the selected feature set after knn-imputation contains 40 features and the greedy search chooses to perform classification by either using 28 features, when the learner is knn-ECOC (where the neighborhood size chosen by Bayesian optimization is still k=2), or 14 features when the learner is an svm_{COST}-ECOC model (where Bayesian optimization has chosen the polynomial kernel of order 2). To improve visibility, the computed suggestions for dataset Y1 and dataset Y3 are reported in Appendix C.

Note that the procedure described in this Section is general and might be applied to any data whose status must be classified, independent from the label meaning. In other words, there is no binding to the health applications.

The described software has been implemented in MATLAB (R2019b). The code is highly parameterized to allow easy changes in the parameter settings.

V. CONCLUSION

In this paper we have presented an extension to SmartFit, a computational framework exploiting wearable sensors and internet applications to allow monitoring a team of athletes, by collecting measurements describing their behavior (activity, sleep, food income, mood, etc…).

The extension regards a team of human DTs which are aimed at mirroring the athletes' conditions (physical twins) and behaviors, with the aim of predicting their condition during training and then suggest changes in behavior to increase performance and health conditions.

Usage of SmartFit has highlighted one the drawback related to human DTs, which is the fact that not all the (human related) data describing the PT status can be continuously collected via Wi-Fi connected sensors, but it must be manually signed by experts or by the PT themselves. Due to lack of attention or distractions, the human related data, such as that treated in our problem, often contains missing and/or noisy data. Each human DT must therefore be designed in order to cope with such problems. In this paper, missing data have been imputed by knn-imputation data, where the value of k (neighborhood size) has been automatically set in order to guarantee robustness to noisy data and maximize the data informativeness.

After data imputation the DT exploits machine learning methods to provide trustable predictions and prescriptions. To choose the most appropriate parameter settings and the best classifier models among four different proposals, which we considered as the most appropriate given the multiclass problem under investigation and the unbalanced dataset we are treating, we have conducted tests by using three different datasets and by performing experiments to evaluate how different parameter settings influence the achieved performance. Our results show that the only manual parameter having some influence on the achieved results is parameter w_{same} (see Section III.A.1). Regarding such parameter, svm-based classifiers are more robust w.r.t. the differing values of w_{same}, while knn-based classifiers achieve the best performance when $w_{same} = 0.75$. A strength of our approach relies in the usage of automatic model selection techniques for both the knn-imputation method and the employed classifiers, which allow achieving promising performance without requiring the tuning of critical parameters.

Moreover, we highlight that all the algorithms integrated in the team of DTs have been developed in order to process any dataset, without any specific requirement. Moreover, the generalization capabilities of the machine learning applications used by the DTs make the proposed work a human monitoring framework that could be exploited by any expert wishing to monitor the conditions of persons being described by measurements collected through IoT connected devices. As an example, SmartFit could be used by cardiologists to monitor the electrical activity of the hearts of patients with cardiac problems, which may be collected by wireless Holter machines.

We highlight that, considering the limited cardinality of the dataset we have used in our experiments, in future works, we aim at using smartFit for collecting an historical dataset with a larger cardinality from a professional team of athletes. This will allow developing more complex, and probably more stable, classifier models (e.g. by using deep learning models).

Moreover, our system has one drawback; whenever some novel ground truth (historical) data is available, all the ground truth data is processed again by the machine learning algorithm (svm or knn) to increase knowledge. This means that, when the training historical data will have a huge cardinality (e.g. after years of data collection), the training process will take too much time. For this reason, our future works will be aimed at investigating/developing an incremental learning method. Incremental learning refers to situations as the one we are facing, where learning is performed on ''streaming data'' arriving over time. Incremental learning techniques are desirable because they are specifically developed to account for limited memory resources and real-time processing, without sacrificing model accuracy [122].

Moreover, we will continue our research investigations for extending the method currently used to compute counterfactual explanations.

Finally, we are now working to provide suggestions in a visual form, e.g. by developing a web application where PTs can view their historical data, their performance, and the suggestions their DTs compute to improve their performance.

APPENDIXES

APPENDIX A

TSNE DECOMPOSITIONS AND VISUALIZATIONS

In this Appendix, for dataset Y1 (Figure 10) and dataset Y3 (Figure 11), we plot the dataset points after projecting them on the first two components computed by the t-sne decomposition [119]. In particular, in the left, the t-sne decomposition exploits the Euclidean distance, while, in the right, t-sne decomposition uses our $DistW_{WNorm}$ distance. Note that, while both the distances cannot well separate the points, when

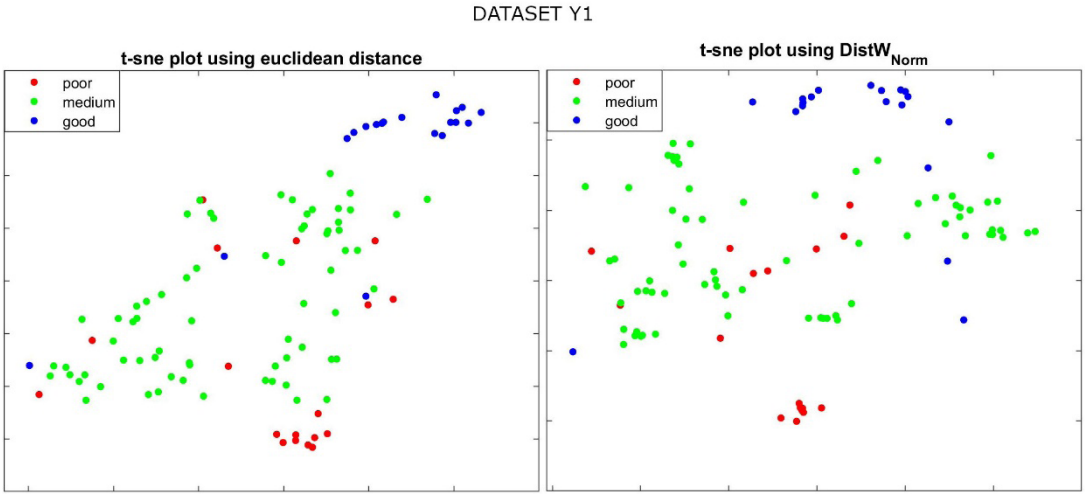

FIGURE 10. t-sne decomposition of points in dataset Y1. Left: t-sne is computed by using the euclidean distance. Right: t-sne is computed by using the DidtW$_{WNorm}$ distance.

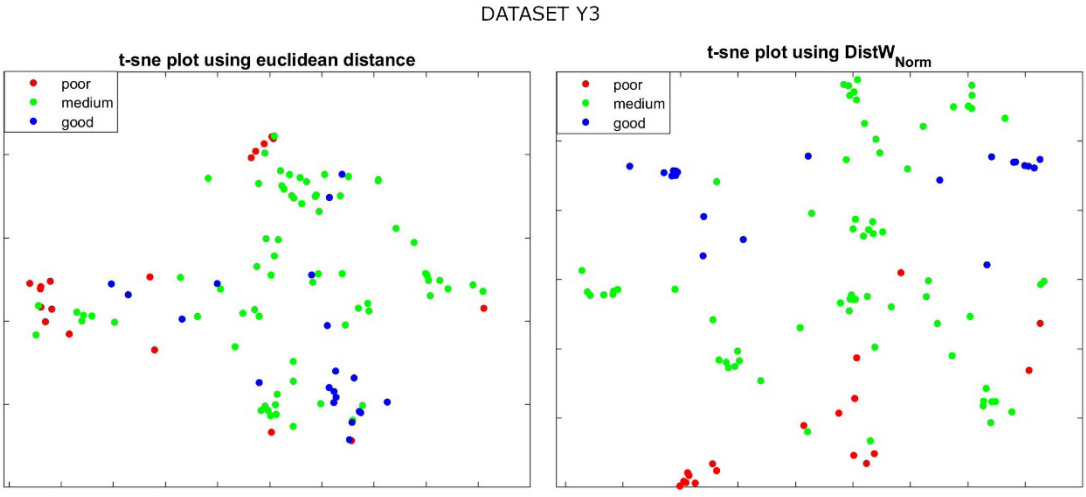

FIGURE 11. t-sne decomposition of points in dataset Y3. Left: t-sne is computed by using the euclidean distance. Right: t-sne is computed by using the DidtW$_{WNorm}$ distance.

using DistW$_{WNorm}$, the separation between points of different classes is increased and, if two points belonging to different classes are near, also their classes are near. In other words, while with the Euclidean distance it happens that some points of class ''poor'' are the nearest to points of the (farthest) class ''good'' (and viceversa). On the contrary, when using DistW$_{WNorm}$ it never happens that points of class ''poor'' are the nearest to points of class ''good'' (and viceversa)

APPENDIX B
CLASSIFIERS PERFORMANCE FOR DIFFERENT VALUES OF PARAMETER W$_{same}$

In this appendix we show all the results we achieved by varying the value of parameter w$_{same}$ in the set $= \{0.5, 0.75, 1\}$. In Table 5, for dataset Y1 (top table) and dataset Y3 (bottom table) we show, for each classifier, the classification loss achieved by LOOCV (column Loss), the neighborhood size chosen the majority of the times in case of knn based classifiers, the mostly chosen svm kernels and their parameters, and the mostly chosen feature dimensionalities (with their offsets).

Results show that, while in Y1 the value of w$_{same}$ clearly influences the classifier performance, which (as expected) are maximized when w$_{same} = 0.75$, on dataset Y3 results differ when the base classifier model are knns or svms. Precisely, the performance of the knn-based classifiers increase together with the value of w$_{same}$, while the classification losses of svm based classifier always achieve a null, or anyway very low (0.01), classification loss. Therefore, svm-based classifiers seem robust with respect to differing values of w$_{same}$.

APPENDIX C
EXAMPLES OF DT'S SUGGESTIONS FOR IMPROVING ATHLETES' PERFORMANCE

In this Section, we firstly show the required changes suggested by the DT, computed by the knn-Ecoc and the svmCOST-Ecoc models, on an athlete's represented by the

TABLE 5. Results achieved for different values of parameters w_{same}.

classifier	Loss	w_{same}	parameters	# Feat
Dataset Y1. On all the LOOCV iterations, most of the times the selected feature set, SelF, has dimensionality equals 16 (-2, +3)				
knn	**0.09**	0.5	k = 3	15 (-2, +3)
	0.08	0.75	k = 2	11 (-2, +4)
	0.09	1	k = 2	9 (-0, +3)
knn-Ecoc	**0.07**	0.5	k = 3	3 (-1, +5)
	0.04	0.75	k = 2	8 (-3, +3)
	0.05	1	k = 2	9 (-2, +4)
svm-Ecoc	**0.05**	0.5	linear	16 (-1, +3)
	0.03	0.75	Poly (order 4)	16 (-2, +4)
	0.06	1	Poly (order 3)	10 (-1, +5)
svm_{COST}-Ecoc	**0.08**	0.5	Poly (order 4)	13 (-3, +5)
	0.02	0.75	Gaussian (scale 2.23)	12 (-4, +3)
	0.03	1	Poly (order 2)	5 (-0, +6)
Dataset Y3. On all the LOOCV iterations, most of the times the selected feature set, SelF, has dimensionality equals 39 (-1, +2)				
	0.07	0.5	k = 2	11 (-2, +7)
knn	**0.07**	0.75	k = 2	10 (-1, +8)
	0.05	1	k = 2	24 (-4, +3)
	0.05	0.5	k = 2	5 (-0, +7)
knn-Ecoc	**0.04**	0.75	k = 2	28 (-7, +5)
	0	1	k = 1	18 (-1, +6)
	0	0.5	Poly (order 4)	17 (-5, +3)
svm-Ecoc	**0**	0.75	Poly (order 4)	15 (-1, +5)
	0.01	1	Gaussian (scale 1.37)	14 (-6, +6)
svm_{COST}-Ecoc	**0**	0.5	Poly (order 4)	13 (-7, +3)
	0	0.75	Poly (order 2)	13 (-1, +7)
	0	1	Poly (order 2)	27 (-4, +1)

22 features in dataset Y1 (these are the suggestions visualized in Figure 9). The athlete has an initial outcome which is ''poor'', and suggestions are therefore computed to reach class ''medium'' and then ''good''.

- -

compute suggestions with classifier: knn Ecoc
Athlete 7, measurement 9: you will get score poor
To get vote medium you must:
decrease Minutes of Moderate Activity at day 3 of 14 units
increase Carbohydrates (g) at day 3 of 3 units
increase Minutes of Rest at day 3 of 16 units
decrease Minutes of Intense Activity at day 3 of 107 units

To get vote good you must:
increase Minutes of Moderate Activity at day 3 of 12 units
decrease Carbohydrates (g) at day 3 of 4 units
increase Minutes of Rest at day 3 of 229 units
decrease Minutes of Intense Activity at day 3 of 92 units

compute suggestion with classifier: svmCOST-Ecoc
Athlete 7, measurement 9: you will get score poor
To get vote medium you must:
decrease Minutes of Moderate Activity at day 3 of 14 units
increase Carbohydrates (g) at day 3 of 3 units
increase Minutes of Rest at day 3 of 16 units
decrease Minutes of Intense Activity at day 3 of 107 units
decrease Burnt Calories at day 3 of 1376 units
increase # Awakenings at day 3 of 4 units
increase Proteins (g) at day 3 of 1 units
decrease Distance (km) at day 3 of 6 units
increase Minutes of Sleep at day 3 of 118 units
increase Minutes Awake at day 3 of 17 units

To get vote good you must:
increase Minutes of Moderate Activity at day 3 of 24 units
increase Carbohydrates (g) at day 3 of 6 units
increase Minutes of Rest at day 3 of 269 units
decrease Minutes of Intense Activity at day 3 of 143 units
decrease Burnt Calories at day 3 of 919 units
increase # Awakenings at day 3 of 11 units
decrease Proteins (g) at day 3 of 7 units
decrease Distance (km) at day 3 of 7 units
increase Minutes of Sleep at day 3 of 141 units
increase Minutes Awake at day 3 of 11 units
decrease Fats (g) at day 3 of 2 units

The following are the required changes suggested when the athlete is represented by all the 66 measurements collected in the three days (dataset Y3).

- -

compute suggestion with classifier: knn Ecoc
Athlete 7, measurement 9: you will get score poor
To get vote medium you must:
decrease Minutes Awake at day 2 of 6.2569 units
increase Minutes Seating Activity at day 3 of 179.7969 units
decrease Minutes Moderate Activity at day 3 of 8.5039 units
decrease Proteins (g) at day 1 of 1.2148 units
increase Minutes Intense Activity at day 2 of 10.3262 units
increase Carbohydrates (g) at day 3 of 1.8223 units
increase Minutes of Rest at day 3 of 9.7188 units
increase Proteins (g) at day 2 of 0.60742 units
decrease Burnt Calories at day 1 of 686.9941 units
decrease Minutes Seating Activity at day 2 of 51.0234 units
increase #Awakenings at day 3 of 2.4297 units
decrease #Awakenings at day 1 of 2.558 units
increase mood at day 1 of 0.60742 units
decrease Minutes of Intense Activity at day 3 of 64.9941 units
decrease Burnt Calories at day 3 of 835.8125 units
decrease Calories burnt with Activity at day 1 of 403.9355 units
increase Minutes of Sleep at day 3 of 71.6758 units
increase Calories burnt with Activity at day 2 of 142.7441 units
increase Minutes Seating Activity at day 1 of 2.4297 units
decrease Distance (km) at day 3 of 3.6445 units

decrease Fibers (g) at day 2 of 1.8223 units
increase Minutes Moderate Activity at day 2 of 13.3633 units
decrease Fats (g) at day 2 of 0.60742 units
decrease # Floors at day 1 of 1.8223 units
increase Minutes Awake at day 3 of 10.3262 units

To get vote good you must:
increase mood at day 3 of 0.84766 units
increase Minutes Awake at day 2 of 19.9832 units
increase Minutes Seating Activity at day 3 of 479.1917 units
increase Minutes Moderate Activity at day 3 of 38.5443 units
decrease Proteins (g) at day 1 of 4.4234 units
increase Minutes Intense Activity at day 2 of 6.6591 units
increase Carbs (g) at day 3 of 7.0589 units
increase Minutes of Rest at day 3 of 282.0548 units
decrease Proteins (g) at day 2 of 4.9934 units
increase Cholesterol (mg) at day 1 of 20.3438 units
decrease Burnt Calories at day 1 of 595.4522 units
increase mood at day 2 of 1.6953 units
decrease Minutes Seating Activity at day 2 of 287.4997 units
increase #Awakenings at day 3 of 10.542 units
increase #Awakenings at day 1 of 1.1266 units
increase mood at day 1 of 0.94019 units
decrease Minutes Intense Activity at day 3 of 120.0968 units
decrease Burnt Calories at day 3 of 751.2058 units
decrease Calories burnt with Activity at day 1 of 700.6699 units
increase Minutes Sleep at day 3 of 139.7631 units
increase Calories burnt with Activity at day 2 of 189.5821 units
decrease Minutes Seating Activity at day 1 of 42.0127 units
decrease Distance at day 3 of 7.3365 units
decrease Fibers (g) at day 2 of 2.8206 units
increase Minutes Moderate Activity at day 2 of 69.0007 units
decrease Fats (g) at day 2 of 1.7878 units
decrease # Floors at day 1 of 4.5159 units
increase Minutes Awake at day 3 of 20.2216 units
- -
compute suggestion with classifier: svmCOST-Ecoc
Athlete 7, measurement 9: you will get score poor
To get vote medium you must:
decrease Minutes Awake at day 2 of 0.020119 units
increase Minutes SeatingActivity at day 3 of 0.57813 units
decrease Minutes ModerateActivity at day 3 of 0.027344 units
decrease Protein (g) at day 1 of 0.0039063 units
increase Minutes IntenseActivity at day 2 of 0.033203 units
increase Carbs (g) at day 3 of 0.0058594 units
increase Minutes of Sleep at day 3 of 0.03125 units
increase Protein (g) at day 2 of 0.0019531 units
decrease Burnt Calories at day 1 of 2.209 units
decrease Minutes Seating Activity at day 2 of 0.16406 units
increase #Awakenings at day 3 of 0.0078125 units

To get vote good you must:
increase mood at day 3 of 0.84668 units
increase Minutes Awake at day 2 of 11.5958 units
increase Minutes Seating Activity at day 3 of 563.1306 units
decrease Minutes Moderate Activity at day 3 of 12.7044 units
decrease Proteins (g) at day 1 of 5.0807 units
increase Minutes Intense Activity at day 2 of 7.6252 units
increase Carbohydrates (g) at day 3 of 5.081 units
increase Minutes of Rest at day 3 of 269.2489 units
decrease Proteins (g) at day 2 of 5.0798 units
decrease Cholesterol (mg) at day 1 of 0.84668 units
decrease Burnt Calories at day 1 of 1616.6502 units
decrease Minutes Seating Activity at day 2 of 60.9861 units
increase #Awakenings at day 3 of 8.468 units
- -

ACKNOWLEDGMENT

The author would like to thank Prof. D. Fogli, Prof. A. Rizzi, and Prof. G. Valentini for their invaluable support.

REFERENCES

[1] M. Grieves. (2015). *Digital Twin: Manufacturing Excellence Through Virtual Factory Replication*. [Online]. Available: https://research.fit.edu/media/site-specific/researchfitedu/camid/documents/1411.0_Digital_Twin_White_Paper_Dr_Grieves.pdf

[2] Q. Qi and F. Tao, "Digital twin and big data towards smart manufacturing and industry 4.0: 360 degree comparison," *IEEE Access*, vol. 6, pp. 3585–3593, 2018, doi: 10.1109/access.2018.2793265.

[3] F. Tao and M. Zhang, "Digital twin shop-floor: A new shop-floor paradigm towards smart manufacturing," *IEEE Access*, vol. 5, pp. 20418–20427, 2017, doi: 10.1109/access.2017.2756069.

[4] B. R. Barricelli, E. Casiraghi, and D. Fogli, "A survey on digital twin: definitions, characteristics, applications, and design implications," *IEEE Access*, vol. 7, pp. 167653–167671, 2019, doi: 10.1109/access.2019.2953499.

[5] J. Vachàlek, L. Bartalsky, O. Rovny, D. Sìsmìsovà, M. Morhàç, and M. Loksk, "The digital twin of an industrial production line within the industry 4.0 concept," in *Proc. 21st Int. Conf. Process Control (PC)*, Bratislava, Slovakia, 2017, pp. 258–262.

[6] B. R. Barricelli and S. Valtolina, "A visual language and interactive system for end-user development of Internet of Things ecosystems," *J. Vis. Lang. Comput.*, vol. 40, pp. 1–19, Jun. 2017, doi: 10.1016/j.jvlc.2017.01.004.

[7] L. Zhu, P. Mussio, B. R. Barricelli, and C. Iacob, "A habitable space for supporting creative collaboration," in *Proc. Int. Symp. Collaborative Technol. Syst. (CTS)*, 2010, pp. 617–622, doi: 10.1109/CTS.2010.5478455.

[8] L. Zhu, B. R. Barricelli, and C. Iacob, "A meta-design model for creative distributed collaborative design," *Int. J. Distrib. Syst. Technol.*, vol. 2, no. 4, pp. 1–16, 2011, doi: 10.4018/jdst.2011100101.

[9] G. Shirner, D. Erdogmus, K. Chowdhury, and T. Padir, "The future of human-in-the-loop cyber-physical systems," *Computer*, vol. 46, no. 1, pp. 36–45, 2013, doi: 10.1109/MC.2013.31.

[10] NASA. *The Ill-Fated Space Odyssey of Apollo 13*. Accessed: Feb. 4, 2020. [Online]. Available: https://er.jsc.nasa.gov/seh/pg13.htm

[11] C. Mandolla, A. Messeni Petruzzelli, G. Percoco, and A. Urbinati, "Building a digital twin for additive manufacturing through the exploitation of blockchain: A case analysis of the aircraft industry," *Comput. Ind.*, vol. 109, pp. 134–152, 2019, doi: 10.1016/j.compind.2019.04.011.
[12] S. H. Chowdhury, F. Ali, and I. K. Jennions, "A methodology for the experimental validation of an aircraft ECS digital twin targeting system level diagnostics," in *Proc. Annu. Conf. PHM Soc.*, vol. 2019, vol. 11, no. 1. [Online]. Available: https://www.phmpapers.org/index.php/phmconf/article/view/888, doi: 10.36001/phmconf.2019.v11i1.888.
[13] O. Hazbon, L. Gutierrez, C. Bil, M. Napolitano, and M. Fravolini, "Digital twin concept for aircraft system failure detection and correction," in *Proc. AIAA Aviation Forum*, 2019, p. 2887, doi: 10.2514/6.2019-2887.
[14] C. Zhang, W. Xu, J. Liu, Z. Liu, Z. Zhou, and D. T. Pham, "A reconfigurable modeling approach for digital twin-based manufacturing system," *Procedia CIRP*, vol. 83, pp. 118–125, 2019, doi: 10.1016/j.procir.2019.03.141.
[15] Q. Qi, F. Tao, T. Hu, N. Anwer, A. Liu, Y. Wei, L. Wang, and A. Y. C. Nee, "Enabling technologies and tools for digital twin," *J. Manuf. Syst.*, to be published, doi: 10.1016/j.jmsy.2019.10.001.
[16] Q. Liu, H. Zhang, J. Leng, and X. Chen, "Digital twin-driven rapid individualised designing of automated flow-shop manufacturing system," *Int. J. Prod. Res.*, vol. 57, no. 12, pp. 3903–3919, Jun. 2019, doi: 10.1080/00207543.2018.1471243.
[17] Y. Lu, C. Liu, K. I.-K. Wang, H. Huang, and X. Xu, "Digital twin-driven smart manufacturing: Connotation, reference model, applications and research issues," *Robot. Comput.-Integr. Manuf.*, vol. 61, Feb. 2020, Art. no. 101837, doi: 10.1016/j.rcim.2019.101837.
[18] J. Guo, N. Zhao, L. Sun, and S. Zhang, "Modular based flexible digital twin for factory design," *J. Ambient Intell. Humanized Comput.*, vol. 10, no. 3, pp. 1189–1200, Mar. 2019, doi: 10.1007/s12652-018-0953-6.
[19] R. Kharat, V. Bavane, S. Jadhao, and R. Marode, "Digital twin: Manufacturing excellence through virtual factory replication," *Global J. Eng. Sci. Res.*, vol. 18, pp. 6–15, Nov. 2018, doi: 10.5281/zenodo.1493930.
[20] R. Stark, C. Fresemann, and K. Lindow, "Development and operation of digital twins for technical systems and services," *CIRP Ann.*, vol. 68, no. 1, pp. 129–132, 2019, doi: 10.1016/j.cirp.2019.04.024.
[21] J. Guo, N. Zhao, L. Sun, and S. Zhang, "Modular based flexible digital twin for factory design," *J. Ambient Intell. Humanized Comput.*, vol. 10, no. 3, pp. 1189–1200, Mar. 2019, doi: 10.1007/s12652-018-0953-6.
[22] G. E. Modoni, E. G. Caldarola, M. Sacco, and W. Terkaj, "Synchronizing physical and digital factory: Benefits and technical challenges," *Procedia CIRP*, vol. 79, pp. 472–477, 2019, doi: 10.1016/j.procir.2019.02.125.
[23] K. Polyniak and J. Matthews. (2016). The Johns Hopkins Hospital Launches Capacity Command Center to Enhance Hospital Operations. John Hopkins Medicine. [Online]. Available: https://www.hopkinsmedicine.org/news/media/releases/the_johns_hopkins_hospital_launches_capacity_command_center_to_enhance_hospital_operations
[24] S. Scharff. (2010). From Digital Twin to Improved Patient Experience. Siemens Healthineers. [Online]. Available: https://www.siemens-healthineers.com/news/mso-digital-twin-mater.html
[25] A. Karakra, F. Fontanili, E. Lamine, and J. Lamothe, "HospiT'Win: A predictive simulation-based digital twin for patients pathways in hospital," in *Proc. IEEE EMBS Int. Conf. Biomed. Health Inform. (BHI)*, Chicago, IL, USA, May 2019, pp. 1–4, doi: 10.1109/BHI.2019.8834534.
[26] K. Bruynseels, F. S. di Sio, and J. van den Hoven, "Digital twins in health care: Ethical implications of an emerging engineering paradigm," *Frontiers Genet.*, vol. 9, p. 31, Feb. 2018, doi: 10.3389/fgene.2018.00031.
[27] Genetics Home Reference. (2019). *What is Precision Medicine?* [Online]. Available: https://ghr.nlm.nih.gov/primer/precisionmedicine/definition
[28] The Precision Medicine Initiative. (2015). *Data-Driven Treatments as Unique as Your Own Body*. [Online]. Available: https://obamawhitehouse.archives.gov/blog/2015/01/30/precision-medicine-initiative-data-driven-treatments-unique-your-own-body
[29] G. Clapworthy, M. Viceconti, P. V. Coveney, and P. Kohl, "The virtual physiological human: Building a framework for computational biomedicine," *Philos. Trans. Roy. Soc. A*, vol. 366, no. 1878, pp. 2975–2978, 2008, doi: 10.1098/rsta.2008.0103.
[30] M. Holtmannspotter, M. Martinez-Galdamez, M. Isokangas, R. Ferrara, and M. Sanchez, "Simulation in clinical practice: First experience with Sim&Cure before implantation of flow diverter (pipeline) or Web-device for the treatment of intracranial aneurysm," in *Proc. ABC/WIN*, 2017. [Online]. Available: https://sim-and-cure.com/wp-content/uploads/2018/01/91aa70_01fd87e567ec49e9907a061bd780c76b.pdf
[31] Dassault Systèmes. (2019). *The Living Heart Project*. [Online]. Available: https://www.3ds.com/products-services/simulia/solutions/life-sciences/the-living-heart-project.
[32] J. Ospel, G. Gascou, V. Costalat, L. Piergallini, K. Blackham, and D. Zumofen, "Comparison of Pipeline embolization device sizing based on conventional 2D measurements and virtual simulation using the Sim&Size software: An agreement study," *Amer. J. Neuroradiol.*, vol. 40, no. 3, pp. 524–530, 2019, doi: 10.3174/ajnr.A5973.
[33] R. Martinez-Velazquez, R. Gamez, and A. E. Saddik, "Cardio twin: A digital twin of the human heart running on the edge," in *Proc. IEEE Int. Symp. Med. Meas. Appl. (MeMeA)*, Istanbul, Turkey, Jun. 2019, pp. 1–6, doi: 10.1109/MeMeA.2019.8802162.
[34] Y. Feng, X. Chen, and J. Zhao, "Create the individualized digital twin for noninvasive precise pulmonary healthcare," *Significances Bioeng. Biosci.*, to be published.
[35] Y. Liu, L. Zhang, Y. Yang, L. Zhou, L. Ren, F. Wang, R. Liu, Z. Pang, and M. J. Deen, "A novel cloud-based framework for the elderly healthcare services using digital twin," *IEEE Access*, vol. 7, pp. 49088–49101, 2019, doi: 10.1109/access.2019.2909828.
[36] L. F. Rivera, M. Jiménez, P. Angara, N. M. Villegas, G. Tamura, and H. A. Müller, "Towards continuous monitoring in personalized healthcare through digital twins," in *Proc. 29th Annu. Int. Conf. Comput. Sci. Softw. Eng.*, 2019, pp. 329–335.
[37] T. Hastie, R. Tibshirani, G. Sherlock, M. Eisen, P. Brown, and D. Botstein, "Imputing missing data for gene expression arrays," Division Biostatistics, Stanford Univ., Stanford, CA, USA, Tech. Rep., 1999.
[38] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman, "Missing value estimation methods for DNA microarrays," *Bioinformatics*, vol. 17, no. 6, pp. 520–525, Jun. 2001.
[39] T. Speed, *Statistical Analysis of Gene Expression Microarray Data*. London, U.K.: Chapman & Hall, 2003.
[40] G. Batista and M. C. Monard, "A study of K-nearest neighbour as an imputation method," *Hybrid Intell. Syst., Frontiers Artif. Intell. Appl.*, vol. 87, no. 46, pp. 251–260, 2002.
[41] C. J. Merz and P. M. Murphy. (1998). *UCI Repository of Machine Learning Datasets*. [Online]. Available: http://www.ics.uci.edu/mlearn/MLRepository.html
[42] J. R. Quinlan, *C4.5 Programs for Machine Learning*. San Mateo, CA, USA: Morgan Kaufmann, 1988.
[43] P. Clark and T. Niblett, "The CN2 induction algorithm," *Mach Learn*, vol. 3, no. 4, pp. 261–283, Mar. 1989.
[44] L. Beretta and A. Santaniello, "Nearest neighbor imputation algorithms: A critical evaluation," *BMC Med. Informat. Decis. Making*, vol. 16, no. 3, pp. 197–208, 2016, doi: 10.1186/s12911-016-0318-z.
[45] S. Yenduri and S. S. Iyengar, "Performance of imputation methods for incomplete datasets," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 17, no. 1, pp. 127–152, 2007.
[46] J. Zhang and I. Mani, "KNN approach to unbalanced data distributions: A case study involving information extraction," in *Proc. Workshop Learn. Imbalanced Datasets II*, 2003. [Online]. Available: http://www.site.uottawa.ca/~nat/Workshop2003/jzhang.pdf?attredirects=0
[47] P. J. García-Laencina, P. H. Abreu, M. H. Abreu, and N. Afonoso, "Missing data imputation on the 5-year survival prediction of breast cancer patients with unknown discrete values," *Comput. Biol. Med.*, vol. 59, pp. 125–133, Apr. 2015, doi: 10.1016/j.compbiomed.2015.02.006.
[48] G. E. A. P. A. Batista and M. C. Monard, "An analysis of four missing data treatment methods for supervised learning," *Appl. Artif. Intell.*, vol. 17, nos. 5–6, pp. 519–533, May 2003, doi: 10.1080/713827181.
[49] M. Chattopadhyay, S. Chattopadhyay, C. Das, and S. Bose, "A novel distance-based iterative sequential KNN algorithm for estimation of missing values in microarray gene expression data," *Int. J. Bioinf. Res. Appl.*, vol. 12, no. 4, pp. 312–342, 2016, doi: 10.1504/ijbra.2016.10001720.
[50] M. Zhu and X. Cheng, "Iterative KNN imputation based on GRA for missing values in TPLMS," in *Proc. 4th Int. Conf. Comput. Sci. Netw. Technol. (ICCSNT)*, Harbin, China, 2015, pp. 94–99, doi: 10.1109/ICCSNT.2015.7490714.
[51] C. Zhang, X. Zhu, J. Zhang, Y. Qin, and S. Zhang, "GBKII: An imputation method for missing values," in *Advances in Knowledge Discovery and Data Mining* (Lecture Notes in Computer Science), vol. 4426, Z. H. Zhou, H. Li, and Q. Yang, Eds. Berlin, Germany: Springer, 2007.

[52] A. Suyundikov, J. R. Stevens, C. Corcoran, J. Herrick, R. K. Wolff, and M. L. Slattery, "Accounting for dependence induced by weighted KNN imputation in paired samples, motivated by a colorectal cancer study," *PLoS ONE*, vol. 10, no. 4, Apr. 2015, Art. no. e0119876, doi: 10.1371/journal.pone.0119876.

[53] P. E. Duda and R. O. Hart, *Pattern Classi cation and Scene Analysis*. New York, NY, USA: Wiley, 1973.

[54] P. Jonsson and C. Wohlin, "An evaluation of k-nearest neighbour imputation using likert data," in *Proc. 10th Int. Symp. Softw. Metrics*, Chicago, IL, USA, 2004, pp. 108–118, doi: 10.1109/METRIC.2004.1357895.

[55] K. Kim, B. Kim, and G. Yi, "Reuse of imputed data in microarray analysis increases imputation efficiency," *BMC Bioinf.*, vol. 5, no. 1, p. 160, 2004, doi: 10.1186/1471-2105-5-160.

[56] J. L. Leevy, "A survey on addressing high-class imbalance in big data," *J. Big Data*, vol. 42, no. 5, p. 42, 2018, doi: 10.1186/s40537-018-0151-6.

[57] K. Graim, "Revealing cancer subtypes with higher-order correlations applied to imaging and omics data," *BMC Med. Genomics*, vol. 10, no. 1, 2017, Art. no. 20, doi: 10.1186/s12920-017-0256-3.

[58] Gliozzo, "Network modeling of patients' biomolecular profiles for clinical phenotype/outcome prediction," *Sci. Rep.*, to be published.

[59] E. Fix and J. L. Hodges, "Discriminatory analysis, nonparametric discrimination," USAF School Aviation Med., Randolph Field, Tex, Project 21-49-004, Tech. Rep. 4, 1951.

[60] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *Amer. Statistician*, vol. 46, no. 3, pp. 175–185, Aug. 1992, doi: 10.1080/00031305.1992.10475879.

[61] X. Wu, "Top 10 algorithms in data mining," *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 1–37, 2008, doi: 10.1007/s10115-007-0114-2.

[62] B. Hosseinifard, M. H. Moradi, and R. Rostami, "Classifying depression patients and normal subjects using machine learning techniques and nonlinear features from EEG signal," *Comput. Methods Programs Biomed.*, vol. 109, no. 3, pp. 339–345, Mar. 2013, doi: 10.1016/j.cmpb.2012.10.008.

[63] B. Pourbabaee, M. J. Roshtkhari, and K. Khorasani, "Deep convolutional neural networks and learning ECG features for screening paroxysmal atrial fibrillation patients," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 12, pp. 2095–2104, Dec. 2018, doi: 10.1109/tsmc.2017.2705582.

[64] Y. Qin, S. Zhang, X. Zhu, J. Zhang, and C. Zhang, "Semi-parametric optimization for missing data imputation," *Appl. Intell.*, vol. 27, no. 1, pp. 79–88, Jun. 2007, doi: 10.1007/s10489-006-0032-0.

[65] S. Zhang, X. Li, M. Zong, X. Zhu, and D. Cheng, "Learning k for kNN Classification," *ACM Trans. Intell. Syst. Technol. (TIST)*, vol. 8, no. 3, 2017, Art. no. 43, doi: 10.1145/2990508.

[66] R. J. Samworth, "Optimal weighted nearest neighbour classifiers," *Ann. Statist.*, vol. 40, no. 5, pp. 2733–2763, 2012, doi: 10.1214/12-AOS1049.

[67] A. Gul, A. Perperoglou, Z. Khan, O. Mahmoud, M. Miftahuddin, W. Adler, and B. Lausen, "Ensemble of a subset of kNN classifiers," *Adv. Data Anal. Classification*, vol. 12, no. 4, pp. 827–840, 2018, doi: 10.1007/s11634-015-0227-5.

[68] A. G. Karegowda and A. S. Manjunath, "Cascading K-means clustering and K-nearest neighbor classifier for categorization of diabetic patients," *Int. J. Eng. Adv. Technol.*, vol. 1, no. 3, pp. 147–151, 2012.

[69] E. Casiraghi, V. Huber, and M. Frasca, "A novel computational method for automatic segmentation, quantification and comparative analysis of immunohistochemically labeled tissue sections," *BMC Bioinf.*, vol. 19, no. 10, 2018, Art. no. 357, doi: 10.1186/s12859-018-2302-3.

[70] P. Campadelli, E. Casiraghi, and D. Artioli, "A fully automated method for lung nodule detection from postero-anterior chest radiographs," *IEEE Trans. Med. Imag.*, vol. 25, no. 12, pp. 1588–1603, Dec. 2006, doi: 10.1109/tmi.2006.884198.

[71] L. Demidova, I. Klyueva, Y. Sokolova, N. Stepanov, and N. Tyart, "Intellectual approaches to improvement of the classification decisions quality on the base of the SVM classifier," *Procedia Comput. Sci.*, vol. 103, pp. 222–230, 2017, doi: 10.1016/j.procs.2017.01.070.

[72] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[73] M. N. Murty and R. Raghava, "Kernel-Based SVM," in *Support Vector Machines and Perceptrons* (SpringerBriefs in Computer Science). Cham, Switzerland: Springer, 2016, pp. 57–67, doi: 10.1007/978-3-319-41063-0_5.

[74] Y. Tang, Y.-Q. Zhang, N. Chawla, and S. Krasser, "SVMs modeling for highly imbalanced classification," *IEEE Trans. Syst., Man, Cybern. B*, vol. 39, no. 1, pp. 281–288, Feb. 2009.

[75] Z. Cai, Z. Yu, H. Zhou, and Z. Gu, "The early stage lung cancer prognosis prediction model based on support vector machine," in *Proc. IEEE 23rd Int. Conf. Digit. Signal Process. (DSP)*, Shanghai, China, Nov. 2018, pp. 1–4, doi: 10.1109/ICDSP.2018.8631657.

[76] D. C. Li, C. W. Liu, and S. C. Hu, "A learning method for the class imbalance problem with medical data sets," *Comput. Biol. Med.*, vol. 40, no. 5, pp. 509–518, 2010, doi: 10.1016/j.compbiomed.2010.03.005.

[77] M. Kächele, P. Thiam, G. Palm, and F. Schwenker, "Majority-class aware support vector domain oversampling for imbalanced classification problems," in *Artificial Neural Networks in Pattern Recognition* (Lecture Notes in Computer Science), vol. 8774. Cham, Switzerland: Springer, 2014.

[78] H. Haibo and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, 2009.

[79] H.-Y. Lo, C.-C. Yen, S.-D. Lin, C.-M. Chang, T.-H. Chiang, C.-Y. Hsiao, A. Huang, T.-T. Kuo, W.-C. Lai, M.-H. Yang, and J.-J. Yeh, "Learning to improve area-under-FROC for imbalanced medical data classification using an ensemble method," *SIGKDD Explor. Newsl.*, vol. 10, no. 2, p. 43, Dec. 2008.

[80] T. Razzaghi, O. Roderick, I. Safro, and N. Marko, "Multilevel weighted support vector machine for classification on healthcare data with missing values," *PLoS ONE*, vol. 11, no. 5, May 2016, Art. no. e0155119, doi: 10.1371/journal.pone.0155119.

[81] A. Iranmehr, H. Masnadi-Shirazi, and N. Vasconcelos, "Cost-sensitive support vector machines," *Neurocomputing*, vol. 343, pp. 50–64, May 2019, doi: 10.1016/j.neucom.2018.11.099.

[82] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes," *Pattern Recognit.*, vol. 44, no. 8, pp. 1761–1776, Aug. 2011, doi: 10.1016/j.patcog.2011.01.017.

[83] T. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *J. Artif. Intell. Res.*, vol. 2, pp. 263–286, Jul. 1995.

[84] E. Kong and T. Dietterich, "Error-correcting output coding corrects bias and variance," in *Proc. 12th Int. Conf. Int. Conf. Mach. Learn.*, 1995, pp. 313–321.

[85] E. B. Kong and T. G. Dietterich, "Why error-correcting output coding works with decision trees," Dept. Comput. Sci., Oregon State Univ., Corvallis, OR, USA, Tech. Rep., 1995. [Online]. Available: http://web.engr.oregonstate.edu/~tgd/publications/ecc-why-draft-1994.pdf

[86] T. Windeatt and R. Ghaderi, "Coding and decoding strategies for multi-class learning problems," *Inf. Fusion*, vol. 4, no. 1, pp. 11–21, Mar. 2003.

[87] C. Molnar. (2019). *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. [Online]. Available: https://christophm.github.io/interpretable-ml-book/

[88] H. Lakkaraju, S. H. Bach, and J. Leskovec, "Interpretable decision sets: A joint framework for description and prediction," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, vol. 16, 2016, pp. 1675–1684.

[89] B. Ustun and C. Rudin, "Supersparse linear integer models for optimized medical scoring systems," *Mach. Learn.*, vol. 102, no. 3, pp. 349–391, Mar. 2016, doi: 10.1007/s10994-015-5528-6.

[90] F. Wang and C. Rudin, "Falling rule lists," in *Proc. 18th Int. Conf. Artif. Intell. Statist. (AISTATS)*, vol. 38, 2016, pp. 1013–1022,.

[91] T. Wang, C. Rudin, F. Doshi-Velez, Y. Liu, E. Klampfl, and P. MacNeille, "Or's of and's for interpretable classification, with application to context-aware recommender systems," 2015, *arXiv:1504.07614*. [Online]. Available: https://arxiv.org/abs/1504.07614

[92] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the GDPR," *Harvard J. Law Technol.*, vol. 31, p. 841, 2017.

[93] S. Watcher, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the GDPR," 2017, *arXiv:1711.00399*. [Online]. Available: https://arxiv.org/abs/1711.00399

[94] D. Martens and F. Provost, "Explaining data-driven document classifications," *MIS Quart.*, vol. 38, no. 1, pp. 73–100, 2014, doi: 10.25300/MISQ/2014/38.1.04.
[95] T. Laugel, M. J. Lesot, C. Marsala, X. Renard, and M. Detyniecki, "Inverse classification for comparison-based interpretability in machine learning," 2017, *arXiv:1712.08443*. [Online]. Available: https://arxiv.org/abs/1712.08443
[96] M. T. Ribeiro, S. Singh, C. Guestrin, C, "High-precision model-agnostic explanations," in *Proc. 32nd AAAI Conf. Artif. Intell. (AAAI)*, 2018. [Online]. Available: https://pdfs.semanticscholar.org/1d8f/4f76ac6534627ef8a1c24b9937d8ab2a5c5f.pdf?_ga=2.106644992.2083133979.1580812718-395322051.1573561033
[97] B. Shneiderman, *Leonardo's Laptop: Human Needs and the New Computing Technologies*. Cambridge, MA, USA: MIT Press, 2002.
[98] G. Fischer. (2002). *Beyond 'Couch Potatoes': From Consumers to Designers and Active Contributors*. [Online]. Available: http://firstmonday.org/issues/issue7_12/fischer/
[99] B. R. Barricelli, G. Fischer, D. Fogli, A. Mørch, A. Piccinno, and S. Valtolina, "Cultures of participation in the digital age: From 'have to' to 'want to' participate," in *Proc. ACM Int. Conf. Ser.*, Oct. 2016, Art. no. 128, doi: 10.1145/2971485.2987668.
[100] M. F. Costabile, P. Mussio, L. Parasiliti Provenza, and A. Piccinno, "End users as unwitting software developers," in *Proc. WEUSE*, 2008, pp. 6–10.
[101] B. R. Barricelli, F. Cassano, D. Fogli, and A. Piccinno, "End-user development, end-user programming and end-user software engineering: A systematic mapping study," *J. Syst. Softw.*, vol. 149, pp. 101–137, Mar. 2019, doi: 10.1016/j.jss.2018.11.041.
[102] B. R. Barricelli and S. Valtolina, "Designing for end-user development in the Internet of things," *End-User Development* (Lecture Notes in Computer Science: Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9083. Cham, Switzerland: Springer, 2015, pp. 9–24, doi: 10.1007/978-3-319-18425-8_2.
[103] L. Rokach and O. Z. Maimon, "Data mining with decision trees: Theory and applications," in *Series in Machine Perception and Artificial Intelligence*, vol. 81. Singapore: World Scientific, 2015.
[104] M. Mitchell, *Machine Learning*. Burr Ridge, IL, USA: McGraw-Hill, 1997.
[105] P. Campadelli, E. Casiraghi, and G. Valentini, "Support vector machines for candidate nodules classification," *Neurocomputing*, vol. 68, pp. 281–288, Oct. 2005.
[106] Y. Sun, A. K. Wong, and M. S. Kamel, "Classification of imbalanced data: A review," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 23, no. 4, pp. 687–719, 2009.
[107] A. Rozza, G. Lombardi, E. Casiraghi, and P. Campadelli, "Novel Fisher discriminant classifiers," *Pattern Recognit.*, vol. 45, no. 10, pp. 3725–3737, Oct. 2012.
[108] K. Warmefjord, R. Soderberg, L. Lindkvist, B. Lindau, and J. Carlson, "Inspection data to support a digital twin for geometry assurance," in *Proc. ASME*, 2017. [Online]. Available: https://asmedigitalcollection.asme.org/IMECE/proceedings-abstract/IMECE2017/58356/V002T02A101/265583, doi: 10.1115/IMECE2017-70398.
[109] S. Pai, S. Hui, R. Isserlin, M. A. Shah, H. Kaka, and G. D. Bader, "netDx: Interpretable patient classification using integrated patient similarity networks," *Mol. Syst. Biol.*, vol. 15, no. 3, 2019, Art. no. e8497, doi: 10.15252/msb.20188497.
[110] A. Rozza, G. Lombardi, and E. Casiraghi, "Novel IPCA-based classifiers and their application to spam filtering," in *Proc. 9th Int. Conf. Intell. Syst. Design Appl.*, 2009, pp. 797–802, doi: 10.1109/ISDA.2009.21.
[111] P. Campadelli, E. Casiraghi, C. Ceruti, and A. Rozza, "Intrinsic dimension estimation: Relevant techniques and a benchmark framework," *Math. Problems Eng.*, vol. 2015, Oct. 2015, Art. no. 759567, doi: 10.1155/2015/759567.
[112] T. W. Anderson and D. A. Darling, "Asymptotic theory of certain 'goodness of fit' criteria based on stochastic processes," *Annals Math. Statist.*, vol. 23, pp. 193–212, Jun. 1952, doi: 10.1214/aoms/1177729437.
[113] M. A. Stephens, "EDF statistics for goodness of fit and some comparisons," *J. Amer. Stat. Assoc.*, vol. 69, no. 347, pp. 730–737, Sep. 1974, doi: 10.1080/01621459.1974.10480196.
[114] M. S. Bartlett, "Properties of sufficiency and statistical tests," in *Proc. Roy. Stat. Soc.*, vol. 160, pp. 268–282, May 1937.
[115] G. W. Snedecor and W. G. Cochran, *Statistical Methods*. Iowa State Univ. Press, 1989. [Online]. Available: https://lib.dr.iastate.edu/press/
[116] W. H. Kruskal and W. A. Wallis, "Use of ranks in one-criterion variance analysis," *J. Amer. Stat. Assoc.*, vol. 47, no. 260, pp. 583–621, Dec. 1952, doi: 10.1080/01621459.1952.10483441.
[117] R. A. Fisher, "The correlation between relatives on the supposition of mendelian inheritance," *Philos. Trans. Roy. Soc. Edinburgh*, vol. 52, no. 2, pp. 399–433 and 2337, 1918.
[118] J. D. Gibbons, *Nonparametric Statistical Inference*, 2nd ed. New York, NY, USA: Marcel Dekker, 1985.
[119] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.
[120] J. Snoek, H. Larochelle, and R. P. Adams,"Practical Bayesian optimization of machine learning algorithms," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, vol. 2, 2012, pp. 2951–2959.
[121] M. A. Gelbart, J. Snoek, and R. P. Adams, "Bayesian optimization with unknown constraints," in *Proc. 13th Conf. Uncertainty Artif. Intell. (UAI)*, 2014, pp. 250–259.
[122] A. Gepperth and B. Hammer, "Incremental learning algorithms and applications," in *Proc. Eur. Symp. Artif. Neural Netw. (ESANN)*, Bruges, Belgium, 2016, Art. no. hal-01418129.