Università degli Studi di Cagliari

PHD DEGREE

in Mathematics and Computer Science

Cycle XXXIII

TITLE OF THE PHD THESIS

Machine Learning Models for Sports Remote Coaching Platforms

Scientific Disciplinary Sector

INF/01

PhD Student:	Walid Iguider
Supervisors:	Prof. Salvatore Carta Prof. Ludovico Boratto

Final exam. Academic Year 2020/2021
Thesis defence: January 2022 Session

Statement of Authorship

I declare that this thesis entitled "Machine Learning Models for Sports Remote Coaching Platforms" and the work presented in it are my own. I confirm that:

- this work was done while in candidature for this PhD degree;
- when I consulted the work published by others, this is always clearly attributed;
- when I quoted the work of others, the source is always given;
- I have acknowledged all main sources of help;
- with the exception of the above references, this thesis is entirely my own work;
- appropriate ethics guidelines were followed to conduct this research;
- for work done jointly with others, my contribution is clearly specified.

Walid Iguider

Abstract

Offering timely support to users in eCoaching systems is a crucial factor to keep them engaged. However, coaches usually follow many users, so it is hard to prioritize those they should interact with first. Timeliness is especially needed when health implications might be the consequence of a lack of support.

Thanks to the data provided by U4FIT (an eCoaching platform for runners we will describe in Chapter 1) and the rise of high-performance computing, Artificial Intelligence can turn such challenges into unparalleled opportunities. One of its sub-fields, namely Machine Learning, enables machines to receive data and learn for themselves without being programmed with rules. Bringing this intelligent support to the coaching domain has many advantages, such as reducing coaches' workload and fostering sportspeople to keep their exercise routine.

This thesis's main focus consists of the design, implementation, and evaluation of Machine Learning models in the context of online coaching platforms. On the one hand, our goal is to provide coaches with dashboards that summarize the training behavior of the sportspeople they follow and with a ranked list of the sportspeople according to the support they need to interact with them timely. On the other hand, we want to guarantee a fair exposure in the ranking to ensure that sportspeople of different genres have equal opportunities to get supported. Past research in this field often relied on statistical processes hardly applicable at a large scale.

Our studies explore opportunities and challenges introduced by Machine Learning for the above goals, a relevant and timely topic in literature. Extensive experiments support our work, revealing a clear opportunity to combine human and machine sensing for researchers interested in online coaching. Our findings illustrate the feasibility of designing, assessing, and deploying Machine Learning models for workout quality prediction and sportspeople dropout prevention, in addition to the design and implementation of dashboards providing trainers with actionable knowledge about the sportspeople they follow.

Our results provide guidelines on model motivation, model design, data collection, and analysis techniques concerning the applicable scenarios above. Researchers can use our findings to improve data collection on eCoaching platforms, reduce bias in rankings, increase model effectiveness, and increase the reliability of their models, among others.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

With the rise of mobile technologies and the spread of Internet access over the world, people's lifestyle and habits changed. People nowadays pursue a more sedentary lifestyle which has become an increasing problem for people's health. A sedentary lifestyle has been reported to be the cause of several serious illnesses such as weight gain, high blood pressure, diabetes, high levels of cholesterol, depression, and several other chronic diseases. In 2020, WHO updated the previous recommendations they released in 2010. They reaffirm messages that some physical activity is better than none, that more physical activity is better for optimal health outcomes and provide a new recommendation on reducing sedentary behaviors. These guidelines highlight the importance of regularly undertaking both aerobic and muscle-strengthening activities.

Researchers from different domains put together their knowledge to tackle this issue from different perspectives to foster people towards a healthy and active lifestyle. Computer scientists and technologists exploited the development of mobile applications and wearable technologies in the last few years to build solutions that help people keep motivated to exercise and to perform a regular physical activity known as e-health persuasive technologies (eHPTs) [SF20]. Several studies put in evidence that the usage of these kinds of support has to be monitored by high-qualified figures.

In this context, as a researcher I am contributing to the development of U4FIT (`https://www.u4fit.com`), a remote coaching ecosystem that connects sportspeople to personal trainers, who can then provide tailored, supervised training programs. This way, U4FIT offers a persuasive technology that encourages people to pursue an active lifestyle under the supervision of specialized coaches. It is made up of a web application and a mobile client. The mobile application uses the devices' sensors to record training statistics, while the web application provides users with an area where they can manage their workout settings and find workout session statistics; it also serves as a dashboard for the coaches, so that they can find all the tools needed

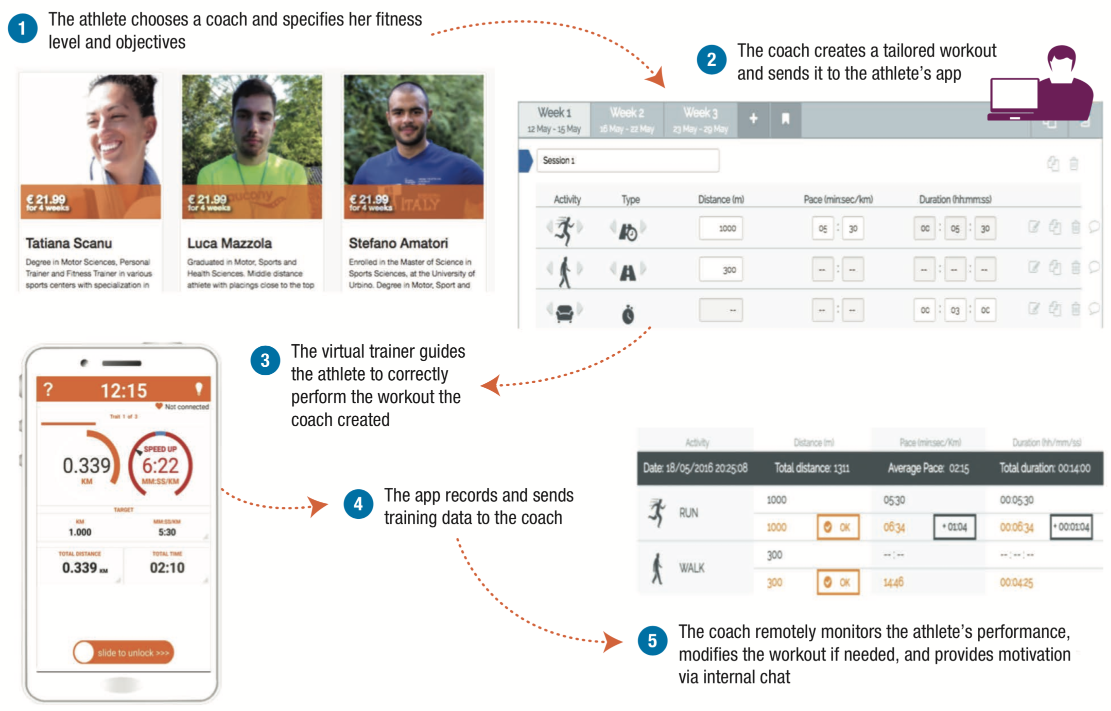

1 The athlete chooses a coach and specifies her fitness level and objectives
2 The coach creates a tailored workout and sends it to the athlete's app
3 The virtual trainer guides the athlete to correctly perform the workout the coach created
4 The app records and sends training data to the coach
5 The coach remotely monitors the athlete's performance, modifies the workout if needed, and provides motivation via internal chat
Tatiana Scanu
Luca Mazzola
Stefano Amatori
Week 1
Week 2
Week 3
Session 1
RUN
WALK

Figure 1.1: User-trainer interaction flow in the *U4FIT* platform

to handle requests of tailored workout plans. After the user chooses a coach and specifies her objectives and current physical skills, the coach receives the user's data and creates a tailored workout plan and sends it to the sportsman's app. When the user receives the workout plan, the virtual personal trainer functionality of the mobile app guides her to correctly complete the workout and the mobile app records training data. At the end of the workout, the coach receives training statistics and remotely monitors the user's performance, modifies the workout (if needed), and motivates her by means of the messaging system.

The goal of my research is to provide coaches with tools that optimize their work and reduce their work load to follow sportspeople in an efficient and effective way.

1.2 Challenges

Much recent research in the machine learning field emphasized the importance of applying learning algorithms to a real-world context [JM15, Sar21]. Nevertheless, the same studies highlighted how challenging it is to use machine learning models for real-world applications. They spotted many challenges that researchers and ML professionals could encounter when dealing with real-world data. Building a successful machine learning-based system mainly depends on two things: the learning algorithm and the data. To choose a learning model that is suitable for the tar-

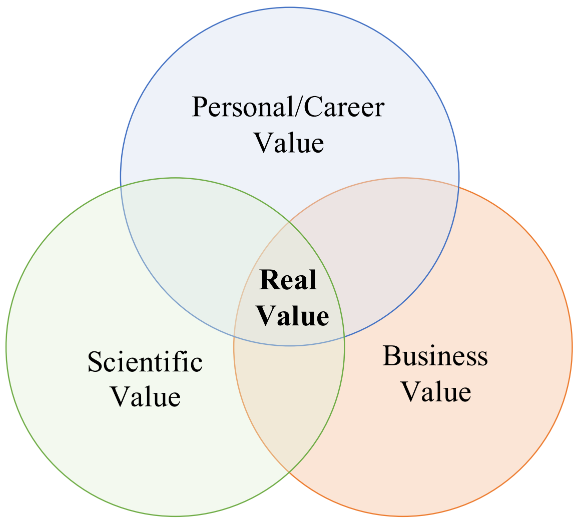


Figure 1.2: An industrial Ph.D. student should find a trade-off between scientific value, business value, and their personal/career value.

get application, one should take many considerations; If the data are inadequate to learn, such as non-representative, poor-quality, irrelevant features, or insufficient quantity for training, then the machine learning models may become useless or will produce lower accuracy. For this reason, during the Ph.D. journey, I have spent much time developing parts of the U4FIT platform that were crucial to performing the research presented in this thesis and which may be helpful for future works. As an industrial Ph.D. student, I faced many challenges; The biggest challenge was creating value for different stakeholders. Namely, the industrial Ph.D. student's job is to create value for science, business, and themselves by working on things that let them grow from a professional perspective depending on their aspirations. Besides this challenge, several others were also highlighted by state-of-the-art. This thesis will focus on research around the following key challenges:

- Many data gathered through wearable sensors have erred values or do not correspond directly to the workout activities in workout plans;
- The data raw made available by the U4FIT platform, without any pre-processing, is insufficient to reasonably build and validate machine and deep learning models for specific tasks, which is needed in the design of meaningful remote coaching interventions;
- Real-world data suffer from huge imbalance when it comes to some machine and deep learning classification tasks;
- In order to deploy the machine learning algorithms we developed, we need servers that have a minimum performance to be able to handle data pipelines

that could be needed at inference time; This could be costly for small and medium-sized companies such as U4FIT;

- Coaches need to know different factors about their followees such as their workout history, the training load, and intensity, among others, to provide them with suitable workout plans that fit their needs and that are not likely to cause them injuries or over-training syndromes; For this reason, it is challenging for coaches to follow several sportspeople,
- Many sportspeople stop training regularly or drop out working out after a few weeks using the platform. This is a common behavior among sportspeople as it was reported by Fletcher et al. [FBB+92] only the 50% of people that start an exercise routine will continue to keep the habit for more than six months.

As AI-based eCoaching will play a consistent role in everyday coaches' and sportspeople's lives, it becomes compelling to approach such critical challenges and provide appropriate support for solving them.

1.3 Thesis goal

Thesis goal. *Support coaches in their daily activities, by developing tools that summarize their work and facilitate their decision-making processes.*

As we highlighted in Section 1.2, the coaches follow a lot of sportspeople on a daily basis. Thus, given the complexity of the results uploaded by the sportspeople, it is challenging for coaches to monitor and analyze the history of the users they follow. This thesis aims to support coaches by providing them with actionable knowledge about the behavior of their sportspeople. To this end, we present the approaches we designed and built to help coaches in different aspects of their daily activities. Some of these approaches exploit Machine Learning based algorithms; meanwhile, others consist of dashboards that summarize the results, workout frequency, and the evolution in time of the performance achieved by sportspeople during workouts sessions.

1.4 Outline

The rest of this thesis is organized as follows: Chapter 2 provides a brief introduction to the most representative machine-learning concepts underlying this thesis.

Chapter 3 presents an overview of the leading research works related to the strategies, techniques, and technologies developed to help sportspeople in pursuing their physical goals. These works are regarding recommender systems for health and wellness, dropout prediction techniques in remote coaching/learning platforms, and algorithmic fairness.

After describing the related works, this thesis is composed of three main parts; Part I is composed of Chapter 4 and Chapter 5; it describes the machine learning algorithms we developed to help coaches interact effectively and efficiently with the sportspeople they follow. Namely, in Chapter 4, we present a machine-learning algorithm that predicts the workout quality of sportspeople to recommend them to their coach using a personalized learning-to-rank approach that takes into account algorithmic fairness constraints. The goal of this algorithm is to provide coaches with a ranked list of sportspeople that need timely support due to a loss in the quality of their workout performance; in the other hand, we reduce the gender unfairness in the ranked list such that the sportspeople are supported fairly regardless of their gender.

Chapter 5 instead describes a deep learning-based approach to help coaches spot sportspeople that are likely to drop out of their exercise routine. The goal of this algorithm is to spare coaches the time to analyze the whole workout history of the sportspeople they follow by providing them with a list of users that are likely to give up training and recommend them to their coaches so that they can interact with them and motivate them to exercise.

Part II is composed of Chapter 6 that describes the tools that are actually deployed in production. These tools consist of different dashboards that provide coaches with actionable knowledge about the training behavior of the sportspeople they follow. Particularly, one of these dashboards is based on the application of the machine learning algorithm described in Chapter 5.
Another goal of these tools is to gather meaningful data for future research work.

Finally, Chapter 7 offers concluding remarks on the implications of our research and provides some opportunities for future work in this field.

Chapter 2

Machine Learning Fundamentals

This chapter provides essential context around artificial intelligence, machine learning, and deep learning concepts leveraged by this thesis.

2.1 What is Machine Learning?

Over the last decades, computer science and engineering researchers investigated strategies and techniques to make computers perform tasks that usually require human intelligence [Nil14]. This field, known as Artificial Intelligence (AI), includes Machine Learning (ML) and Deep Learning (DL), in addition to other rule-based algorithms (Figure 2.1).

When we talk about Machine Learning, we are talking about a particular branch of AI that uses data and algorithms to mimic the way humans learn. ML mainly focuses on building systems that are given in input data and answers about a particular task and learn, on their own, patterns that can be applied to unseen data (Figure 2.2) [CMM84]. Examples that are relevant to the target task are given as an input to the machine learning system that learns patterns from these data, making

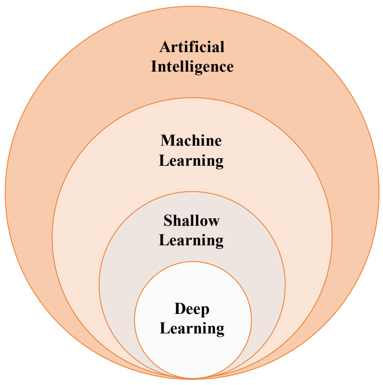


Figure 2.1: Hierarchy in artificial intelligence, machine learning, deep learning fields.

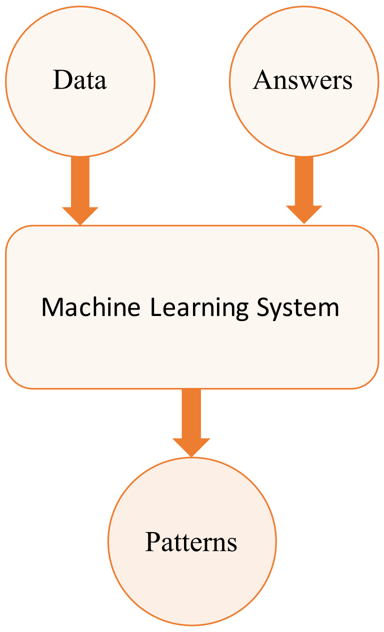


Figure 2.2: Machine Learning systems.

it possible to get complex patterns for solving the task. For instance, a machine-learning model for object classification is fed with human-labeled images from which a set of rules for associating pictures to object labels are learned.

Deep learning (DL) is an ML branch wherein patterns are learned from data through consecutive manipulation through a sequence of stacked layers [LBH+15]. It differs from traditional machine learning, namely shallow learning, which learns only one or two layers of data representations. The transformation implemented by a deep neural layer is parameterized by its weights. Hence, learning means optimizing the weights of all layers, such that the network correctly maps inputs to expected targets. Given the predicted and actual targets, the system computes a score through a loss function that captures how well the network maps the current samples. The score is then used by the optimizer that arranges the weight values through a Back-propagation algorithm so that the loss score will be lower in the next iteration. Repeating the loop a sufficient number of times makes it possible to learn weight values that minimize the loss, obtaining a trained model.

2.2 Types of Machine Learning

Several approaches can be suitable for defining problems faced by a machine-learning system (Figure 2.3).

The machine learning problems can be classified into four broad categories [Lis15], namely:

- *Human-Supervised learning* is based on learning how to map data to known an-

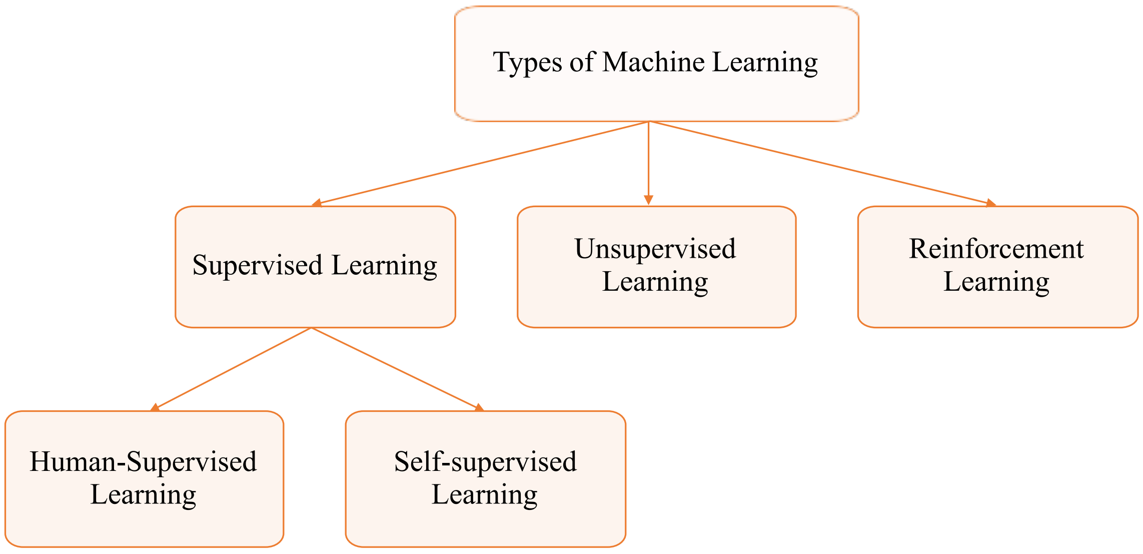


Figure 2.3: Types of machine learning algorithms.

notations (i.e., labels). Most applications, such as object recognition, speaker verification, sentiment prediction, and language understanding, fall into this category;

- *Self-Supervised learning* implies supervised learning with labels generated from the input data, typically using a heuristic algorithm. For instance, auto-encoders, where the inputs are also the generated targets, take full advantage of self-supervised learning;

- *Unsupervised learning* aims to find interesting transformations of the input data without knowing any or only a subset of targets. Sample applications are survival analysis, data denoising, dimensionality reduction, and clustering;

- *Reinforcement learning* is based on an agent which receives information about its environment and learns to choose actions that will maximize some reward. For instance, a neural network that outputs game actions to maximize its score can leverage it.

Over this thesis, we mainly focus on supervised learning problems. Therefore, the subsequent sections provide information tailored to this type of problem.

2.3 Experimental Workflow

Common pipelines solving a machine-learning problem include problem definition, data pre-processing, model development and model evaluation (Figure 2.4).

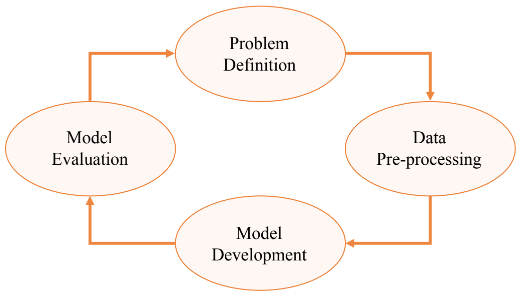


Figure 2.4: Common pipeline to solve a ML problem.

2.3.1 Problem Definition

This step serves to define the type of problem (e.g., binary classification, multi-class classification, multi-label classification, scalar regression). Identifying the problem type guides the choices made at the following steps, such as the model architecture, the loss function, etc. In addition, inputs and outputs need to be defined and, based on that design choice, good training data should be retrieved. For instance, learning to classify the sentiment of reviews implies having both reviews and sentiment annotations. The hypothesis is usually that the learning algorithm can predict the outputs given the inputs; hence, the retrieved data should be sufficiently informative to learn the relationship between inputs and outputs. Therefore, ML can only be used to get the patterns present in the training data and recognize what it has seen there.

2.3.2 Data Pre-Processing

This step aims to make data more manageable by algorithms through vectorization, normalization, missing values handling, and/or feature extraction:

- Vectorization. Inputs and outputs should be numerical vectors, irrespective of the data (e.g., images, text). Turning data into vectors is called vectorization. For instance, text can be represented as a list of integers standing for sequences of words. On the other hand, this step is not needed when data is already in numerical form.

- Normalization. It should be noted that feeding into an ML model data that takes large values or is heterogeneous could prevent the model from converging. To make learning easier, data should have values ranging between 0 and 1. For instance, image data encoded as integers ranging between 0 and 255 is usually cast to float and divided by 255 so that they become float values ranging between 0 and 1. Similarly, when predicting users' identities, each feature could be normalized to have a standard deviation of 1 and a mean of 0.

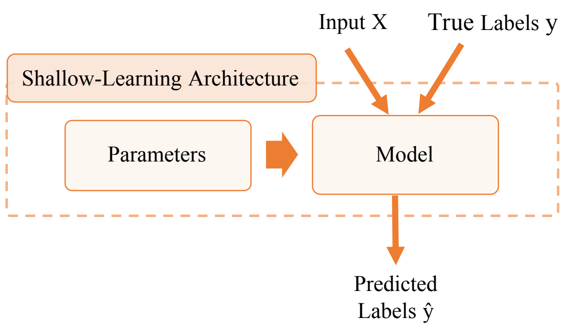


Figure 2.5: The common components of a shallow-learning model.

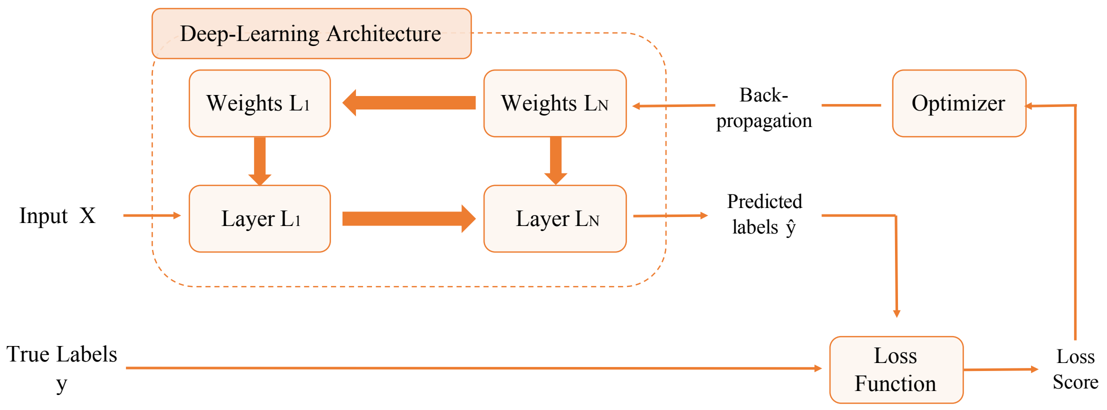


Figure 2.6: The common components of a deep-learning model.

- Missing Values Handling. If there could be missing values when predicting through an ML model, it is generally a good practice to simulate such a situation also during model training. To this end, while training, missing values as 0 could be introduced by copying some training samples and dropping some features that may become missing while predicting. In this way, the model can learn that the value 0 means missing data and starts ignoring the value.

- Feature Extraction. Using human knowledge about the data and the ML algorithm can make the algorithm work better. Feature extraction is usually adopted by shallow algorithms not having hypothesis spaces rich enough to learn valuable features by themselves. For instance, in speaker verification, inputs for neural networks are typically based on pre-processed data, such as spectrograms and filterbanks extracted from the raw audio. Modern DL makes it possible to be fed raw data and let neural networks extract useful patterns.

2.3.3 Model Development

The goal at this step is to develop an ML model able to solve the original task [Cho17]. As we mentioned in Chapter 2.2, in this thesis we mainly focus on supervised learning problems. That said, three key choices to build the model should be considered: (i) model architecture that should learn meaningful data representations, (ii) differentiable optimization function that should match the type of problem, (iii) optimization configuration that should support the model in minimizing the objective function. Some shallow-learning models structured as depicted in Figure 2.5 are described below.

- Decision Trees (DTs) [BFOS17] predict the value of a target variable by learning decision rules from input features. The model has a root node containing all data features of the training set. Then, the root node is split into several children according to a given criterion. This process recursively continues on children until no nodes to be split exist.

- Support Vector Machines (SVMs) [DLPS14] map each training data sample to a point in an N-dimensional space, where N is the number of features and the value of each feature is the value of a particular coordinate. Then, it finds the set of hyper-planes that better differentiate the points based on the targets. A linear combination of vectors determines the location of the decision boundaries producing the best separation.

- Random Forest (RF) [Bre01] is a meta estimator that (i) fits several decision tree classifiers on various random data sub-samples and (ii) uses averaging to improve the predictive accuracy and to control over-fitting. Each decision tree is a weak classifier, while all the decision trees combined aim to be a more robust classifier.

- Gradient Boosting (GB) [NK13] is also an ensemble algorithm that improves the accuracy of a predictive function through incremental minimization of the error term. After the initial base learner (almost always a tree) is grown, each tree in the series is fit to the so-called ”pseudo residuals” of the prediction from the earlier trees to reduce the error.

Some deep-learning models structured as depicted in Figure 2.6 are described below [Sch15].

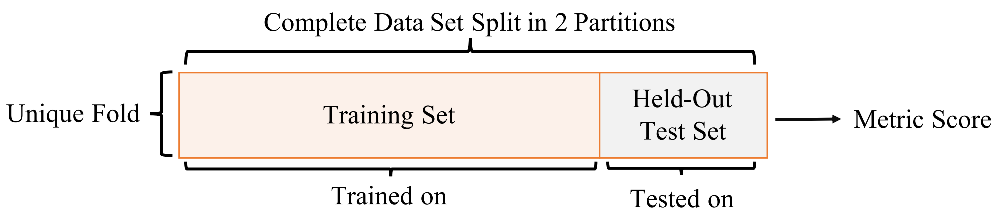
- Feed-forward Neural Networks(FNN) were one of the first components applied to learn from data using DL [HLR06, ZZ06]. One or more levels of nodes, namely perceptrons, are randomly joined by weighted connections in a many-to-many fashion. These networks were historically thought of in order to simulate a biological model where nodes are neurons, and links between them represent synapses. Based on the input values fed into the network, nodes of a certain level can be activated, and their signal is broadcasted to the subsequent level. In order to activate nodes of a subsequent level, the signal generated at a level is weighted and must be greater than a given threshold.

- Recurrent Neural Networks (RNNs) are tailored for processing data as a sequence [VXD$^+$14, GMH13]. In contrast to FNNs, RNNs have cyclic connections among nodes of distinct levels. Recurrent connections connect past data with the one that is currently being processed, simulating a state memory. The forward pass is similar to FNN forward pass. The difference is that the activation of a node depends on both the current input and the previous status of the hidden layers. This workflow is useful when data presents patterns from the past to the future. As an extension, Bidirectional RNNs (BiRNNs) walk through the training data forward and backward using two hidden RNNs combined into a common output layer, making it possible to find patterns from both past and future data [SP97].

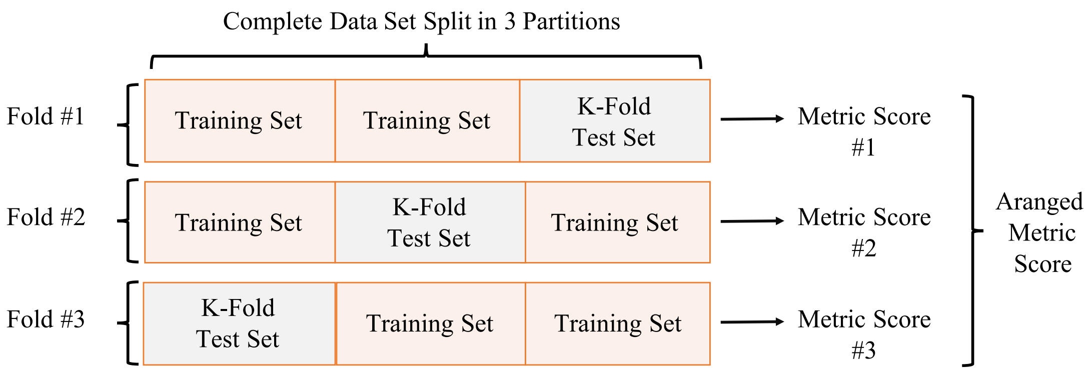
- Long Short-Term Memory (LSTM) extends RNNs by employing recurrent connections and adding memory blocks in their recurrent hidden layers [SSB14, HS97]. These memory blocks save the current temporal state and make it possible to learn temporal observations hidden in the data. Using memory blocks allows relating the current data being processed with the data processed long before, solving the problem experienced by common RNNs. For this reason, LSTMs have been proved to have a positive impact on sequence prediction tasks. Bidirectional layers using two hidden LSTMs can be leveraged to process data both forward and backward.

- Convolutional Neural Networks (CNNs) perform filtering operations on the nodes of a layer, abstracting and selecting only meaningful nodes. Such networks have been historically applied in Computer Vision [SVSS15, DSG14]. Hence, they are not directly applicable to texts, as the text should be vectorized before applying convolutional filters to them. Each filter is composed of a kernel that slides on the vector representation and repeats the same function on each element until all vectors are covered.

As part of the optimization for DL, the error for the current state of the model must be estimated repeatedly. This requires the choice of an error function, namely a loss function. Such a function is used to estimate the loss of the model so that the weights can be updated to reduce the loss on the next evaluation. Based on the type of ML problem, there are several adequate loss functions. For instance, for regression problems, common loss functions are Mean Squared Error, Mean Squared Logarithmic Error, and Mean Absolute Error; for binary supervised classification, Binary Cross-Entropy, Hinge Loss, and Squared Hinge Loss are usually adopted; for multi-class classification, common solutions are Multi-Class Cross-Entropy, Sparse Multi-Class Cross-Entropy, and Kullback Leibler Divergence. Please refer to [JC17]



(a) Held-out ML evaluation protocol.



(b) K-fold ML evaluation protocol.

Figure 2.7: Common ML evaluation protocols.

for further details on each function. Finally, within DL, an optimizer is integrated to update the weights and minimize the loss function. The optimizer uses the loss function to move in the right direction to reach the global minimum. Common optimizers include Root Mean Square Propagation (RMSProp) [TH17] and Adaptive Moment Estimation (ADAM) [KB14].

2.3.4 Model Evaluation

Evaluating a model always needs to subdivide the available data into (i) a training set, (ii) a validation set, and (iii) a test set. During training, the model is fed with the training set and evaluated on the validation set. The latter data is used because developing a model usually involves tuning its configuration (e.g., choosing the number of layers or the size of the layers). Such tuning can be achieved by using as feedback the performance of the model on the validation set. To test performance on unseen data, a completely different data set is used to evaluate the model: the test set. The common procedures for splitting data are provided in what follows (Figure 2.7) [Mar16]:

- Held-Out Validation. A subset of the data is set apart for testing, typically around 10% and 20% of the whole dataset. The model is trained on the rest of the data, and its performance is evaluated on the test set. However, if little data is available, then the validation and test sets may contain too few samples to be statistically representative, preventing the validity of the experimental results.

- k-Fold Validation. The data is split into K equal-sized partitions. An independent instance of the model is trained on K–1 partitions and evaluated on partition i. The process is repeated K times, with a different partition i as a test set. The final metrics are averaged to obtain the final score. This might solve issues related to significant variance on final metrics over different train-test splits.

After splitting the data, the metrics for measuring success should be defined. The metric should be directly aligned with the high-level goals, such as the success of the business. For balanced-classification problems, where every class is equally likely, Accuracy and area under the Receiver Operating Characteristic curve (ROC) are standard metrics. For class-imbalanced problems, Precision and Recall are usually used. For ranking problems or multi-label classification, Mean Average Precision is generally adopted. In some cases, custom metrics are defined. Such metrics represent the output of the protocol. As the universal tension in ML is between optimization and generalization, the ideal model is the one that stands right at the border between under-fitting and over-fitting. Hence, all the pipeline steps should be repeated until the model closely reaches this goal.

Chapter 3

Related Work

3.1 Recommender Systems for Health and Wellness

Several studies emphasized the importance of providing users with personalized recommendations, to support them in having a healthy and active lifestyle, and to design effective interventions [Smy19, KWB06, YTFK+17].

In this context, some studies focused on recommending physical activities tailored to the user profile. Donciu et al. [DIDT11] bring together the social dimension acquired from a growing community and expert knowledge defined within an ontology to provide users with diet and workout recommendations based on their profile information, preferences, and declared purpose. In [HAST14], He et al. suggest recommending physical activities to users based on the context (e.g., risk tolerance, budget, location, weather). Ahire et al. [AK15] use semantic web technology to analyze users' preferences, build a user profile based on this knowledge, then recommend food and exercise inquiries to users based on their profile. Khwaja et al. suggest recommending physical activities to users by considering the type of personality [KFI+19]. Finally, in [NodAV14], Nassabi et al. propose tailoring the recommendations according to the user's health status, goals, and preferences.

Other approaches, instead, have focused on making recommendations to users with specific characteristics. Tseng et al. provide people suffering from chronic diseases (e.g., metabolic syndrome) with diet and exercise guideline recommendations [TLL+15]. Dobrican and Zampunieris [DZ16] focus on cardiac patients with the goal of rehabilitation and thereby aims to provide the optimum of both automated and manual interventions [MAA+87]. Santos-Gago et al. suggest making personalized recommendations to sportswomen by considering their menstrual cycle [SSG+19]. Berndsen et al. and Smyth and Cunningham [BSL19, SC18b, SC18a] propose supporting users in marathon preparation using recommender systems that suggest to runners a challenging but achievable goal-time in addition to a tailored plan based on a pace.

Even though the use of recommender systems for health and wellbeing is an emerging trend, recent studies put in evidence that having a health care expert-based intervention is necessary when using these kinds of support [PKB18, MGA$^+$16]. However, only a few works on technology-based physical activity promotion have included expert knowledge in their recommendation process.

3.2 Dropout Prediction in eCoaching Platforms

Since there are no works that study dropout prediction in sports eCoaching platforms to the best of our knowledge, we retain the closest domain to our research to be student dropout prediction. *Manhães et al. proposes a student dropout prediction system to support academic administrators spot the ones who are in danger of dropout in a public Brazilian University [MdCZ14]. According to the experiments, the classifier Naïve Bayes achieved the best performance.* Li et al. presents a method based on behavior features and multi-view semi-supervised learning for dropout prediction [LGL$^+$16]. The study by *Liu et al. [LWBT18] describes a time series-based approach for disengagement prediction and illustrates the potential of the method by applying the methodology to the Open University learning analytics dataset.* Fei et al. also presents an approach based on time series to solve the problem of dropout prediction based on extensive experiments conducted on two MOOCs offered on Coursera and edX [FY15]. While *Wang et al. and* Qiu et al. propose approaches based on Convolutional Neural Networks for solving dropout prediction problem in MOOCs, resectively in [WYM17] and [QLHL19].

Although other work has been carried out on supporting users to keep an active lifestyle, to the best of our knowledge, a lot of work has been done in the student dropout prediction. Still, none of them predicts *if* users will train on a weekly basis to spot and recommend sportspeople likely to lose motivation to their coach, based on their recent behavior and their adherence to the objectives set by the coaches following them.

3.3 Fairness in Rankings

Across time, there have been many debates on fairness and justice in moral philosophy that led to many and different points of view and thus to different definitions of fairness that are not well-agreed [Bin18]. Hence, in the Machine Learning field, it is common to evaluate the fairness of an algorithm using measures that assess how much this algorithm is discriminating against a protected group. Fairness in the field of Information Retrieval and, more precisely in ranking problems, has been approached from different perspectives.

Yang and Stoyanovich [YS17] suggest assessing fairness in rankings by adopting measures based on statistical parity that compute the difference in the distribution of different groups for different prefixes of the ranking (top-10, top-20, and so on).

Zehlike et al. [ZBC+17], face the challenge of generating a trade-off between fairness and utility in "Top-k ranking" by satisfying two levels of constraints. The first level consists of making sure that the more relevant items are above less relevant ones within the same group. In contrast, the second introduces a fairness constraint that ensures that the proportion of protected group items in every prefix of the top-k ranking is above a minimum threshold.

Several other works proposed different fairness constraints that mainly present parity constraints restricting the fraction of items with each attribute in the ranking [SJ18]. However, Biega et al. [BGW18] go beyond such parity constraints and present a framework that ensures amortized fairness in rankings, based on the equity of attention, by focusing on individual fairness while making exposure proportional to relevance for all subjects, using an integer linear program to generate a series of rankings.

In parallel with this work, Singh and Joachims in [SJ18] tackle the challenge of the fairness of exposure in rankings by suggesting a more generic framework for finding rankings that maximize the utility for the user while satisfying a specifiable notion of fairness.The authors propose three fairness constraints:

1. **Demographic Parity** enforces that the average exposure of the documents in the protected and non-protected groups are equal;

2. **Disparate Treatment** enforces that exposure of the protected and non-protected groups to be proportional to their average utility;

3. **Disparate Impact** assures that the click-through rates for the groups as determined by the exposure and relevance are proportional to their average utility.

As mentioned in [SJ18], there is no single definition of a fair ranking, but fairness constraints depend on context and application. Indeed, in some works that presented real-world applications of user recommendation under fairness constraints, the authors have chosen measures that best fit their domain and context.

In [HTBL18], Hutson et al. highlighted the issue of bias, discrimination, and exclusion w.r.t. race during the matchmaking process in the study and design of intimate platforms. Also, in the people recommendation domain, Geyik et al. [GAK19] proposed a framework for ensuring fairness in the hiring domain. More precisely, they exploited the concepts of equality of opportunity [HPS16] and fairness through awareness [DHP+12] to create fair opportunities for all users seeking a job in the LinkedIn Talent Search platform. In the context of educational recommender systems, Marras et al. introduced a novel fairness metric that monitors the equality of learning opportunity according to a novel set of educational principles and proposed a re-ranking approach to mitigate unfairness in online educational platforms [MBRF20].

Part I

Algorithms

Chapter 4

Machine Learning Models for Workout Quality Prediction

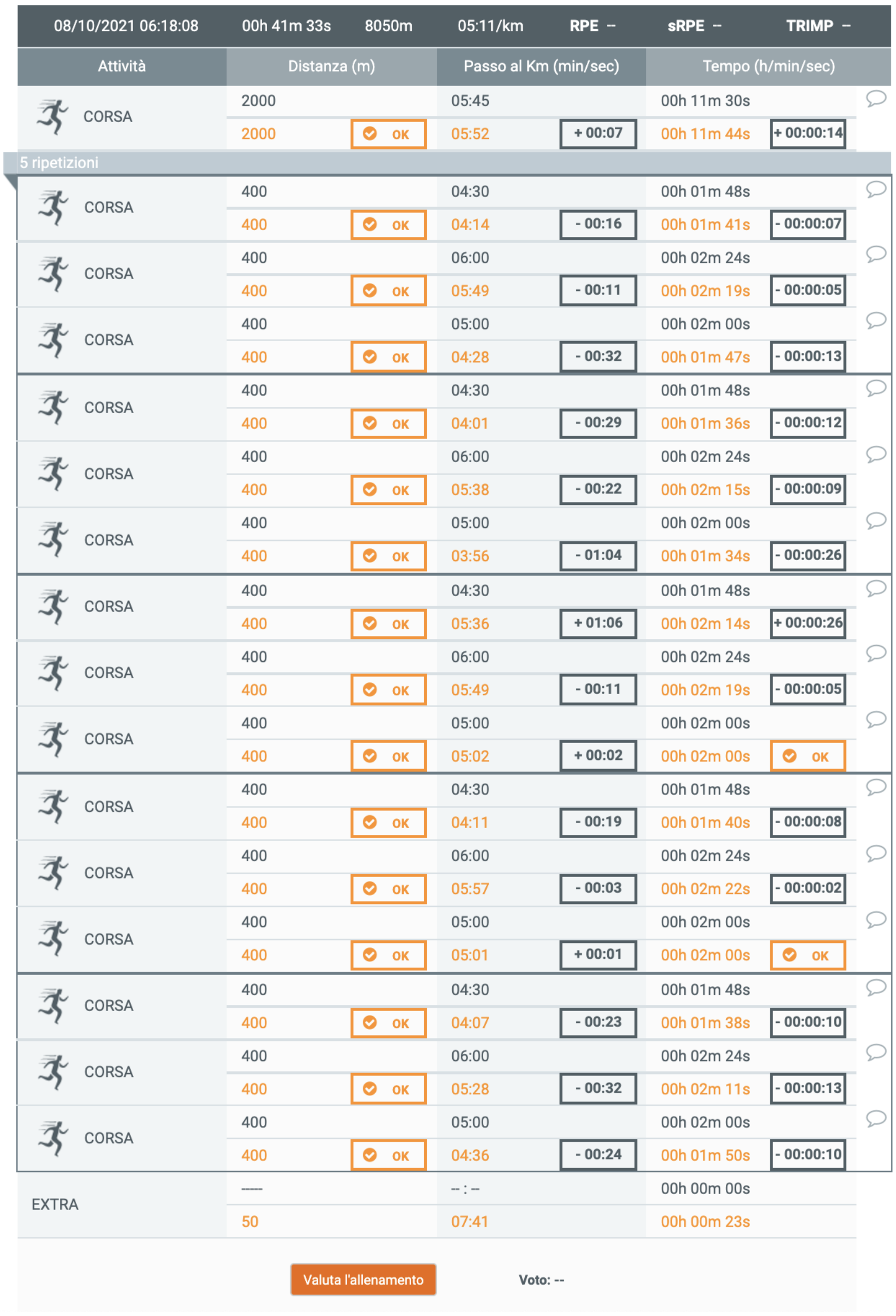
4.1 Introduction

eCoaching systems support users at achieving their personal goals [Kam17]. In the context of health, they assist users in their self-care, sometimes through the promotion of physical activity routines [KMM+15]. Human coaches have a key role in keeping users engaged [BCMP17]. However, keeping users engaged on the long-term is a challenging task, since a coach usually supports a lot of people[1]. In the physical activity domain, this means that after a workout session a coach should get in touch with the people they support (e.g., via a chat). In this sense, prioritizing users after their workout is key, in order to get in touch first with those who need more support (e.g., because they completed a workout with bad performances). A lack of prioritization might have consequences that go beyond engagement and might have direct implications on the health and well-being of users, since those with the worst performances would have a delayed support.

Our contributions. In this chapter, we propose a recommender system that suggests to a coach the sportspeople who performed a workout, according to their performance. User recommendation (a.k.a. contact recommendation) is usually intended as the task of suggesting one user to another, in order for them to connect [SCC19]. In our domain, a sportsperson can be recommended to a coach multiple times (even very frequently, according to the sportsperson's training schedule).

Our approach first models users according to their workout performance, then ranks them in ascending order of workout quality, thus suggesting first those with the worst performances. The choice of introducing a recommender system between the end of a workout and the support offered by the coach is not only motivated by the large number of users that a coach follows, but also by the complexity of

[1]In the platform considered in this thesis, a coach follows on average 21.3 users.



08/10/2021 06:18:08	00h 41m 33s	8050m	05:11/km	RPE --	sRPE --	TRIMP --
Attività	**Distanza (m)**		**Passo al Km (min/sec)**		**Tempo (h/min/sec)**	
CORSA	2000		05:45		00h 11m 30s	
	2000	✓ OK	05:52	+ 00:07	00h 11m 44s	+ 00:00:14
5 ripetizioni						
CORSA	400		04:30		00h 01m 48s	
	400	✓ OK	04:14	- 00:16	00h 01m 41s	- 00:00:07
CORSA	400		06:00		00h 02m 24s	
	400	✓ OK	05:49	- 00:11	00h 02m 19s	- 00:00:05
CORSA	400		05:00		00h 02m 00s	
	400	✓ OK	04:28	- 00:32	00h 01m 47s	- 00:00:13
CORSA	400		04:30		00h 01m 48s	
	400	✓ OK	04:01	- 00:29	00h 01m 36s	- 00:00:12
CORSA	400		06:00		00h 02m 24s	
	400	✓ OK	05:38	- 00:22	00h 02m 15s	- 00:00:09
CORSA	400		05:00		00h 02m 00s	
	400	✓ OK	03:56	- 01:04	00h 01m 34s	- 00:00:26
CORSA	400		04:30		00h 01m 48s	
	400	✓ OK	05:36	+ 01:06	00h 02m 14s	+ 00:00:26
CORSA	400		06:00		00h 02m 24s	
	400	✓ OK	05:49	- 00:11	00h 02m 19s	- 00:00:05
CORSA	400		05:00		00h 02m 00s	
	400	✓ OK	05:02	+ 00:02	00h 02m 00s	✓ OK
CORSA	400		04:30		00h 01m 48s	
	400	✓ OK	04:11	- 00:19	00h 01m 40s	- 00:00:08
CORSA	400		06:00		00h 02m 24s	
	400	✓ OK	05:57	- 00:03	00h 02m 22s	- 00:00:02
CORSA	400		05:00		00h 02m 00s	
	400	✓ OK	05:01	+ 00:01	00h 02m 00s	✓ OK
CORSA	400		04:30		00h 01m 48s	
	400	✓ OK	04:07	- 00:23	00h 01m 38s	- 00:00:10
CORSA	400		06:00		00h 02m 24s	
	400	✓ OK	05:28	- 00:32	00h 02m 11s	- 00:00:13
CORSA	400		05:00		00h 02m 00s	
	400	✓ OK	04:36	- 00:24	00h 01m 50s	- 00:00:10
EXTRA	-----		-- : --		00h 00m 00s	
	50		07:41		00h 00m 23s	

Figure 4.1: Workout results for a sportsperson

workout results (a workout is usually composed by different activities, such as running, walking, and resting, and each activity is in turn made up of several statistics, like the speed and covered distance, as illustrated in Figure 4.1), which need to be contextualized with the characteristics of the users (e.g., gender, age, and workout objective). With our proposal, we are offering coaches an initial filtering of the workout results, to facilitate their work. Hence, in the flow presented in Figure 1.1, our solution enriches step 4. Concretely, when the app returns the workout results, it would not only provide the coach with the fine-grained results of each user, but the coach also sees a ranking of the users, thus being able to analyze first the results of those who are supposed to be more in need.

In this work, we model the recommendation problem as a ranking problem, since our goal is not to predict the quality of a workout with a score (rating), but to provide the coach with an effective ranking of the users to support, in so-called "Personalized Learning to Rank" approaches [AB15]. Indeed, predicting the rating of a workout is not enough to provide a coach with effective information about the user, since the performance in the last workout should be contextualized with the usual behavior of the user; e.g., it would be much more urgent to support a user who does a poor workout but usually does well, than to support a user who performed a poor workout, but always does so (in this second case, a coach expects that the performance of that user would not be optimal). In a nutshell, *we model users' workout performance by contextualizing it to their recent behavior and use this modeling to provide a personalized ranking of these users to the coaches.*

As previously mentioned, sensitive attributes of the users, such as gender, are used by our ranking algorithm. Hence, there might be the risk for the users who belong to a certain gender to receive a *disparate treatment*, i.e., to receive a less timely support, because of an attribute that should not affect their ranking position. Hence, it is important that users receive a *fair exposure*, i.e., that their ranking positions are not affected by their gender. However, relevance estimation by itself does not guarantee fairness of exposure [BGW18, SJ18]. In order to deal with this issue, we provide metrics to assess fairness of exposure and an efficient algorithm to re-rank the unfair lists.

To the best of our knowledge, in the athletic field, no one has ever developed a system able to support users by ranking them in a fair way, helping their human coaches prioritizing who needs the most immediate support.

Although contact recommenders have been widely studied in the literature [GP16, SCC19], our novelty goes much beyond the application domain. Indeed, classic contact recommenders are not necessary anymore after two people connect, while a user connects to a coach *through a recommendation* multiple times (i.e., each time they perform a workout). Later in the chapter, we will also highlight differences at an algorithmic level, which make our problem fundamentally new.

Specifically, our contributions can be summarized as follows:

- We present an approach to model the performance of the users in a running workout session;

- We introduce an algorithm to rank the users according to the support they need and recommend them to the coach;
- We provide, for the first time in the literature of athletic-related user recommendation, algorithms to provide fairness of exposure in the results;
- We validate our proposal on a real-world dataset collected from an eCoaching platform on standard metrics to assess ranking quality.

Roadmap. The rest of the chapter is structured as follows: Section 4.2 presents the preliminaries, to provide foundations to our work. In Section 4.3, we introduce the dataset and our approach to workout modeling. Section 4.4 describes the user recommendation algorithm, and in Section 4.5 we present the experimental framework and results. We conclude the chapter in Section 4.6.

4.2 Preliminaries

Here, we present the preliminaries, to provide foundations to our work.

4.2.1 Recommendation scenario

Let U be a set of users, and C be a set of coaches, both belonging to the eCoaching platform. The subscriptions of users to the services of the coaches is a binary relation $S \subseteq U \times C$; we denote as S_c the users that are followed by a coach $c \in C$. Moreover, we denote as R_w the set of raw features captured by eCoaching platform during a workout $w \in W$.

Our first goal is to build a model of each workout, denoted as M_w, which captures information about the workout performance of a user, and contextualizes this performance with the previous behavior of the user. More formally, we will build a function $f : R \rightarrow M$, which takes the raw features R, to build a new set of features M. Given the set of workout plans prepared by a coach, which is a binary relation $P \subseteq C \times M$, we denote as P_c the plans prepared by a coach $c \in C$. Given a coach $c \in C$ our final goal is to build a functions, $g : P_c \times S_c \rightarrow S_c$, considers the set of workouts of the users followed by a coach and ranks those users according to their performance; The users will be ranked from the poorly performing to the better performing one.

4.3 Dataset and Workout Modeling

In this section, we provide the details of the dataset we employed in this study, and we provide a first characterization of the data. Later, we present the pre-processing

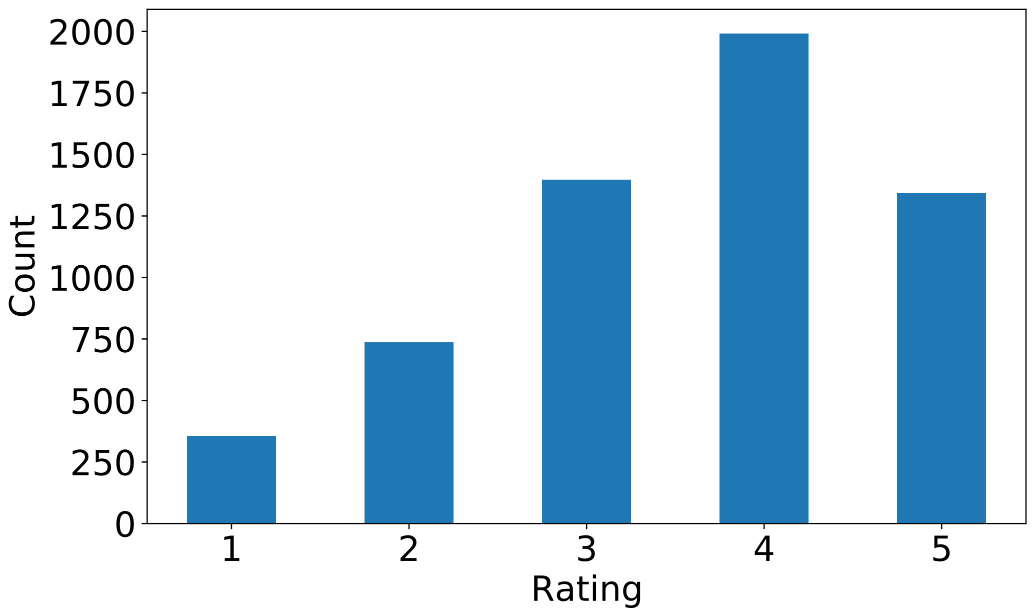


Figure 4.2: **Ratings distribution in the dataset.** The x axis (Rating) reports each rating that could be assigned by a trainer, and y axis (Count) reports the number of workout that received that rating.

steps we performed on the obtained workouts and our approach to model workouts.

Our research is based on a real-world dataset, containing 47,555 activities that compose 8,486 workouts (our set W). This means that each workout is composed by several activities. The workouts were performed by the set U of 412 users. Users have a different running experience and the coach is aware of the background of the users she follows.

The coaches in the platform evaluated these workouts by assigning a rating (denoted as r_w, where w is the workout who received that rating) ranging between 1 and 5. As we are dealing with real-world data, we encountered the problem of class imbalance. Figure 4.2 represents graphically the distribution of ratings, where "Count" indicates the number of samples having the corresponding rating. We will deal with this phenomena before the classification process, as described in Section 4.5.1.

Table 4.1 describes the raw features of each workout in the original dataset (our set R) and Figure 4.3 presents the distribution of activities and workouts.

4.3.1 Dataset Characterization

In this section, we delve into our data, to understand how it is distributed. This characterization also serves as a motivation to our problem, since we provide in-

Table 4.1: **Description of the raw features.** We use four columns to characterize each feature. Concretely, we report the feature's ID (the u prefix denotes a *user* feature, the w prefix a *workout* feature, and a an *activity* feature), the feature's name, the type with which its values can be represented, and a textual description of it.

ID	Feature	Type	Description
$u1$	User ID	int	ID of the user
$u2$	User Birth Date	Date	Date of birth of the user
$u3$	User Gender	string	Gender of the user (M for male, and F for female)
$u4$	User Height	int	Height of the user (in meters)
$u5$	User Weight	int	Weight of the user (in kg)
$w1$	Workout ID	int	ID of the workout
$w2$	Burnt Calories	float	Amount of calories burnt during the workout session.
$w3$	Workout Date	date	The date when the workout was performed
$a1$	Activity ID	int	ID of the activity
$a2$	Distance Objective	int	The distance goal given by the coach to the sportsperson for that activity (in meters)
$a3$	Covered Distance	float	The distance covered by the sportsperson when performing that activity
$a4$	Speed Objective	int	The speed goal given by the coach to the sportsperson for that activity (in km/h)
$a5$	Average Speed	float	The average speed performed by the sportsperson for that activity (in km/h)
$a6$	Time Objective	int	The time goal given by the coach to the sportsperson for that activity (in seconds)
$a7$	Time Elapsed	float	The time performed by the sportsperson for that activity (in seconds)
$a8$	Pace Objective	int	The pace goal given by the coach to the sportsperson for that activity (in min/km)
$a9$	Average Pace	float	The average pace performed by the sportsperson for that activity (in min/km)
$a10$	Activity Type	string	The type of that activity (either *walking*, *running*, or *resting*)
$a11$	Activity Label	string	The label of that activity (either, *pace*, *distance*, *time*, or *unknown*, indicating the type of objective the activity has; the *unknown* label is taken by those activities that do not have an objective)

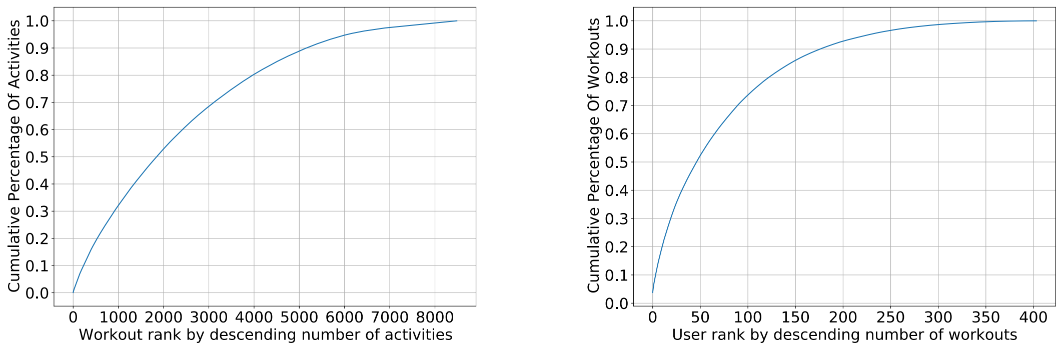


Figure 4.3: **Distributions of activities and workouts**. Cumulative distribution of activities per workouts (left). Cumulative distribution of workouts per users (Right).

sights on data imbalance from multiple perspectives, and conjecture on the possible implications it can have when ranking workout results.

From the left part of Figure 4.3, we can see that almost 70% of the activities in the dataset compose only the first 3000 workouts (hence, less than one third of the workouts comprise 70% of the activities). This means that those workouts are composed of a lot of activities, which makes it very challenging for coaches to analyze and evaluate in a short time. For this reason, it would be helpful to provide coaches with a ranking of users in order to spot immediately the sportspeople that need timely support. Thus, by optimizing the coaches' workload, our system will certainly help increase the efficiency and effectiveness of eCoaching. Observing the right part of the figure, we can remark that almost 70% of the workouts in the dataset where performed by the first 100 users (hence, by around one fifth of the users). This means that the first 100 users performed a considerable number of workouts, which makes it interesting to contextualize our modeling also with the workout history of users.

From Figure 4.4, instead, we notice that the percentage of workouts performed by male sportspeople is mostly twice the percentage of workouts performed by their female counterpart. Hence, in our dataset, the male users represent the majority group. The different in the number of workouts performed by different genders in our dataset may lead our system to be biased w.r.t. the workouts performed by the users of the gender that performed more workouts (i.e., males). We would like to remark once again that the gender should not impact the ratings, since the coaches that created the workout plans and rated the performance of users take into consideration the gender of users, their experience, and their health conditions.

Table 4.2: **Workout Modeling Features.** We use five columns to model each workout. Concretely, we report the feature's category, its ID, the feature's name, the type with which its values can be represented, and a textual description of it.

Category	ID	Feature	Type	Description
	$f1$	Workout ID	int	ID of the given workout, directly derived from $w1$
Distance-based features	$f2$	Distance Objective	float	Sum of the distance objectives of the activities of the considered workout (feature $a2$ in Table 4.1)
	$f3$	Covered Distance	float	Sum of all the covered distances of the activities of the considered workout (feature $a3$ in Table 4.1)
	$f4$	Distance Gap	float	Obtained by first calculating the difference between the distance objective (feature $a2$ in Table 4.1) and covered distance (feature $a3$ in Table 4.1) for each activity in the workout, and then averaging the obtained values (this feature indicates how well the users respected their distance objective)
	$f5$	Distance Gap Variance	float	Variance of the distance gaps in each activity considered to compute feature $f4$ (this feature indicates how far are the individual values from the average)
	$f6$	Distance Gap Standard Deviation	float	Standard deviation of the distance gaps in each activity considered to compute feature $f4$ (this feature also indicates how far are the individual values from the average, but it is expressed in the same units as the data)
Temporal features	$f7$	Time Objective	int	Sum of the time objectives of the activities of the considered workout (feature $a6$ in Table 4.1)
	$f8$	Workout Duration	float	Sum of all the time the user has taken to complete the activities of the considered workout (feature $a7$ in Table 4.1)
	$f9$	Temporal Gap	float	First create the difference between the time objective (feature $a6$ in Table 4.1) and elapsed time (feature $a7$ in Table 4.1) for each activity in the workout, and then average the obtained values (this feature indicates how well the user respected her time objective)
	$f10$	Temporal Gap Variance	float	Variance of the temporal gaps in each activity considered to compute feature $f9$
	$f11$	Temporal Gap Standard Deviation	float	Standard deviation of the temporal gaps in each activity considered to compute feature $f9$
Pace-based features	$f12$	Pace Objective	int	Average of the pace objectives of the activities of the considered workout (feature $a8$ in Table 4.1)
	$f13$	Average Pace	float	Average of the paces of the activities of the considered workout (feature $a9$ in Table 4.1)
	$f14$	Pace Gap	float	Obtained by first calculating the difference between the pace objective (feature $a8$ in Table 4.1) and average pace (feature $a9$ in Table 4.1) for each activity in the workout, and then averaging the obtained values (this feature indicates how well the user respected her pace objective)
	$f15$	Pace Gap Variance	float	Variance of the pace gaps in each activity considered to compute feature $f14$
	$f16$	Pace Gap Standard Deviation	float	Standard deviation of the pace gaps in each activity considered to compute feature $f14$
Workout characteristics	$f17$	Walking Activities' Percentage	float	Percentage of activities in a workout where feature $a10$ is equal to *walking*
	$f18$	Running Activities' Percentage	float	Percentage of activities in a workout where feature $a10$ is equal to *running*
	$f19$	Percentage of Activities with an Objective	float	Percentage of activities in a workout where feature $a11$ is not equal to *unknown*
	$f20$	Percentage of Well-performed Activities	float	Percentage of activities in a workout that have any gap equal to 0
	$f21$	Week Day	int	The day of week when the workout was performed; this feature takes values from 1 to 7, and is obtained from the feature $w3$ in Table 4.1
	$f22$	Week Number	int	The week of year when the workout was performed; this feature takes values from 1 to 53, to account for years who have 53 weeks, , and is obtained from the feature $w3$ in Table 4.1
	$f23$	Month	int	The month when the workout was performed; this feature takes values from 1 to 12, and is obtained from the feature $w3$ in Table 4.1
	$f24$	Days From Previous Workout	int	The number of days from the previous workouts session.
User characteristics and behavior	$f25$	User Age	int	Created using feature $u2$ described in Table 4.1, in order to contextualize the workout performance with the age of the user
	$f26$	User Gender	categorical	Directly computed from feature $u3$ described in Table 4.1 (0 for female, 1 for male)
	$f27$	User Height	int	Directly computed from feature $u4$
	$f28$	User Weight	int	Directly computed from feature $u5$
	$f29$	User BMI	float	Computed using features $f27$ and $f28$
	$f30$	User Fidelity	int	Number of workout sessions the user has performed from the first time they used the platform.
	$f31$	Mean Rating	float	Decaying average of the ratings r_w obtained by the user in previous workouts. This feature allows us to monitor the evolution in the performance of a user, by giving more importance to the last sessions without neglecting the past ones. The decaying average of the element X at the position N is: $\overline{X}_N = \frac{3}{4} \cdot X_{N-1} + \frac{1}{4} \cdot \overline{X}_{N-2}$.

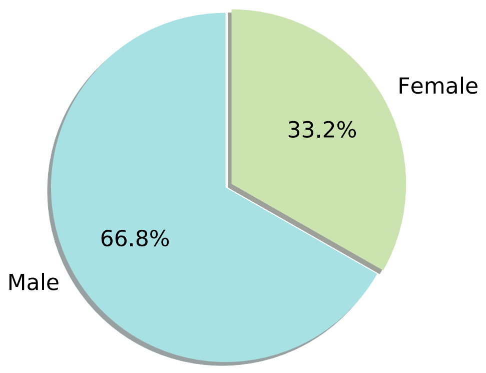


Figure 4.4: **Percentage of workouts performed by each gender.** For each gender of the dataset, we report the percentage of workout performed by the users that recognize themselves as belonging to that gender.

4.3.2 Data Pre-processing and Feature Extraction

Data Pre-processing. From all the workouts in the dataset, we removed all those that are not reliable. A workout is not reliable when at least one of the following conditions is met: (*i*) $covered\ distance > 43,000\ meters$, (*ii*) $workout\ duration > 5\ hours$, (*iii*) $rest\ time > 1\ hour$, (*iv*) $average\ speed > 16\ km/h$. We also removed the workouts that were not performed under the supervision of a coach. After removing the irrelevant workouts, the final dataset consists of 5,823 workouts performed by 291 users.

Feature Extraction. Given the raw features available in our dataset and presented in Table 4.1, the next goal is to model each workout, by doing some feature engineering. We regrouped all the activities that belong to each workout and excluded the activities that have *resting* as *activity type* (feature $a10$) since, according to coaches, they are not considered when evaluating workout quality; for this reason, they should not be part of our user modeling and recommendation algorithm.

In Table 4.2, we describe the features we created, and how they are derived from the original ones.

4.4 Fair User Recommendation

In this section, we describe the algorithm we implemented to recommend users who need support of the coach.

4.4.1 Motivation

Before we go into the detailed steps of our approach, it is important to highlight why our approach departs from the main classes of recommender systems (collaborative filtering and content-based approaches) and from classic people recommender systems:

- **Classic people recommenders** exploit the topology of the social network ("since you are connected to these users, you might connect to these"); this would not fit our work, since in this work we are not recommending sportspeople to coaches that might suit them, but we recommend to coaches those who need support after a workout;

- **Collaborative-filtering approaches** do not consider item features, which are essential to predict if a user needs support or not (we are basing support on a prediction of workout quality). Moreover, collaborative filtering approaches consider *static items* (e.g., a movie does not change over time), while in our domain there is no such thing as two identical workouts. Hence, collaborative algorithms would not fit our approach either;

- **Content-based approaches** match two users based on the content they post. While training results are a form of content exploited by our algorithm, the matching between the coach and the sportsperson is not what triggers our recommendations.

Workout quality and its relation to previous users' behavior and their objectives are what drive the recommendation of a user to a coach, thus making our problem new from a recommendation point of view. Hence, no direct comparison of our work to existing people recommenders is possible.

Continuing, we motivate our choice to provide fairness via a re-ranking approach and how our method departs from the existing ones. Mitigation methods for unfairness in rankings can be categorized into pre-processing, in-processing, and post-processing methods, as illustrated in Figure 4.5;

- **Pre-processing methods** aim to mitigate disparities in user ranks by intervening at the level of training data, either before these candidates are processed by a ranking algorithm or during the ranking process;

- **In-processing methods** intervene on the ranking algorithm such that it produces a ranked outcome that meets the specified fairness criteria;

- **Post-processing methods** intervene on the output ranking in such a way that it meets the specified fairness criteria.

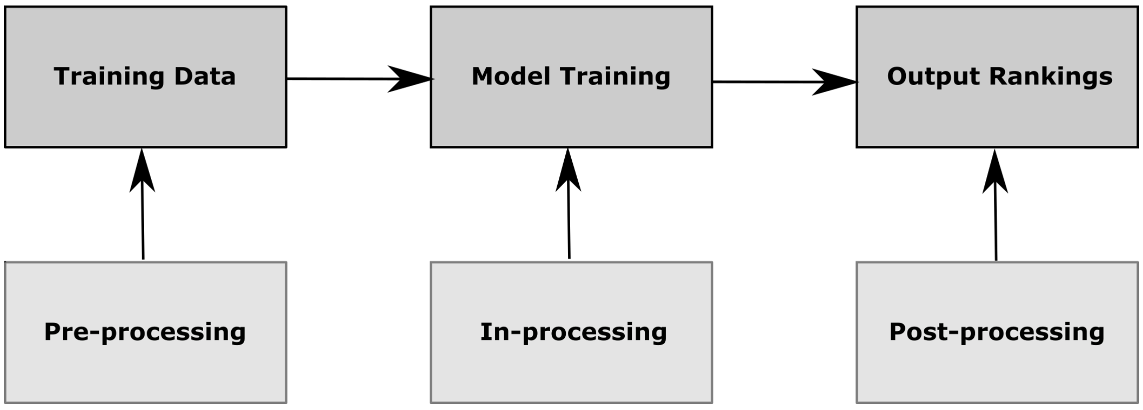


Figure 4.5: **Unfairness mitigation at different stages of ranking**. The bottom boxes (Pre-processing, In-processing, and Post-processing) indicate the three forms of mitigation strategy. Each bottom box points to another box (Training Data, Model Training, and Output Rankings), which indicates at what stage of the pipeline the strategy makes an intervention.

From the perspective of fairness, we opted for a post-processing method by making a classic assessment of *the exposure given to the different genders in the ranking*. Here, the application domain is new, by providing fairness to users in need of support in eCoaching platforms.

A re-ranking algorithm is the only option when optimizing ranking-based metrics, such as visibility and exposure. An in-processing regularization, such as those that have been presented in [KAAS18, BCD+19], would not be possible, since at prediction stage the algorithm does not predict *if and where* an item will be ranked in a recommendation list; hence, no direct comparison with these approaches is possible. This is not due to the specific choice of algorithms, since this consideration would also hold for list-wise approaches. Re-rankings have been introduced to reduce disparities, both in the context of non-personalized rankings [ZBC+17, SJ18, BGW18, CSV18, ZC20, PBG+20] and of recommender systems [MMB+18, BSO18], with approaches such as Maximal Marginal Relevance [CG98].

However, all these algorithms optimize only one property (either utility or exposure). As we will show later in our ablation study, optimizing for one metric is not enough, so no direct comparison with these approaches is possible.

4.4.2 Our Approach

The user recommendation process is divided into two main steps:

1. **Performance-based ranking**: we rank the sportspeople based on the performance in the last workout, contextualized to their recent behavior.

2. **Fair re-ranking**: we assess how fair is the ranking algorithm in terms of exposure of the sportspeople and provide a re-ranking algorithm for the cases in which sportspeople of a given gender are affected by disparate exposure.

The steps are now described in detail.

4.4.3 Performance-based Ranking

The intuition behind this algorithm is that predicting the quality of a workout is a central element in order to provide a recommendation to a coach. For this reason, we initially predict the rating that the coach would assign to a given workout. The input received by the classifier is the workout model composed of the 30 features we engineered in Section 4.3.2. Different classes of classification algorithms can be employed for the purpose of predicting workout quality, from ordinal to multi-class approaches. As we will show in Section 4.5.2, the chosen class of algorithms implies treating the ground truth as a continuous or disjoint set of classes (ordinal and multi-class classification, respectively); in our evaluation, we explore the effectiveness of the two classification strategies in our context. The output of a classifier is a predicted rating, denoted as $\hat{r}_w$.

Finally, we rank the users based on the predicted rating $\hat{r}_w$. The "urgency" with which they will get support depends on their performance during their last workout session. In general, a high $\hat{r}_w$ leads to a high rank. Instead, if $\hat{r}_w$ is low, the user will get a more timely support.

A recommendation list $\mathbf{R}$ for a coach is represented by the list of users followed by them, ranked by ascending $\hat{r}_w$.

Since coaches and sportspeople have a continuous relationship, we simulate the recommendation scenario of the real-world application. Under this scenario, we assume that the coach will check who might need support by checking the u4fit application at regular intervals. To simulate these intervals, we start by ranking the users that performed the first 5 workouts for each coach, then we update the ranking for each coach whenever the sportspeople followed by this coach perform 5 new workouts.

4.4.4 Fair Re-ranking

Every output generated by the previous step is a list of users to be recommended to a coach, based on their likelihood of needing support, according to their performance.

The classification algorithm uses the gender of the users as a feature used in the classification process (feature $f24$), systematically under-exposing the users of a given gender would mean that the ranking is affected by the so-called *disparate treatment*. Disparate treatment means that users belonging to a given gender might be ranked lower w.r.t. to their counterpart, even though they might need the same (or more) support.

Hence, the first step is to assess *how fair* is the ranking, in terms of the exposure given to the users [SJ18]. The exposure that a user gets in a ranking is given by:

$$\text{Exposure}\left(u|\mathbf{R}\right) = \frac{1}{\log(1+j)} \tag{4.1}$$

where j is the position the user covers in $\mathbf{R}$.

In order to measure how "deserving" is that user to cover position j in the ranking, we measure their utility according to:

$$\text{Utility}\left(u|\mathbf{R}\right) = \frac{2^{\text{rel(u)}} - 1}{\log(1+j)} \tag{4.2}$$

where $rel(u) = max(r_w) - \hat{r}_w$.

It should be trivial to note that the utility of a user corresponds to their DGC, which is a common practice in the literature [SJ18].

Let G_i denote the subgroup of users having the same gender. The Exposure and Utility for that group are calculated as follows:

$$\text{Exposure}\left(G_i|\mathbf{R}\right) = \frac{1}{|G_i|}\sum_{u\in G_i} Exposure(u) \tag{4.3}$$

and

$$\text{Utility}\left(G_i|\mathbf{R}\right) = \frac{1}{|G_i|}\sum_{u\in G_i} Utility(u). \tag{4.4}$$

We first assume a recommendation list (ranking) to be fair if the two groups get the same *Exposure*, defined as follows:

$$\text{Exposure}\left(G_0|\mathbf{R}\right) = \text{Exposure}\left(G_1|\mathbf{R}\right). \tag{4.5}$$

In order to assess if a recommendation list is fair, we measure Demographic Parity Ratio (DPR) as follows:

$$\text{DPR}\left(G_0, G_1|\mathbf{R}\right) = \frac{\text{Exposure}\left(G_0|\mathbf{R}\right)}{\text{Exposure}\left(G_1|\mathbf{R}\right)} \tag{4.6}$$

A DPR equal to 1 indicates the users of a given gender get a fair exposure, while a value lower or greater than 1 tells us which group is disadvantaged in terms of disparate exposure.

The DPR metric only accounts for the position in which users are ranked, without accounting for their utility, in demographic parity fashion. To account also for the *Utility* of the users of a given group, we introduce another constraint that considers it, to balance *Exposure* of the two groups while preserving ranking quality:

$$\frac{\text{Exposure}\left(G_0|\mathbf{R}\right)}{\text{Utility}\left(G_0|\mathbf{R}\right)} = \frac{\text{Exposure}\left(G_1|\mathbf{R}\right)}{\text{Utility}\left(G_1|\mathbf{R}\right)}. \tag{4.7}$$

In order to assess if a recommendation list is fair, we measure Disparate Treatment Ratio (DTR) as follows:

$$\mathrm{DTR}\left(G_0, G_1|\mathbf{R}\right) = \frac{\mathrm{Exposure}\left(G_0|\mathbf{R}\right) / \mathrm{Utility}\left(G_0|\mathbf{R}\right)}{\mathrm{Exposure}\left(G_1|\mathbf{R}\right) / \mathrm{Utility}\left(G_1|\mathbf{R}\right)} \tag{4.8}$$

A DTR equal to 1 indicates fair exposure for the users, while a value lower or greater than 1 tells us which group is disadvantaged in terms of disparate exposure.

In case our two metrics, DPR and DTR, report scores different from 1, we developed a re-ranking approach to generate a fair exposure. The intuition behind our algorithm is that each pair of users that have a different gender and appear consequently in a ranking is a candidate for a swap, so that the disadvantaged gender can be given more exposure. Our approach is summarized in Algorithm 1.

ALGORITHM 1: Order-Based Re-ranking

```
input  : $X$: users sorted by rank, $D$: fairness metric (either $DTR$ or $DPR$)
output: $R$: ranked list of users that respects group fairness constraints
1  $d \leftarrow$ empty dictionary;
2  $s \leftarrow$ empty dictionary;
3  $d[X] \leftarrow D$;
4  $s[X] \leftarrow$ getAllSwappableRows($X$);
5  while $s[X]$ is not empty do
6      $p \leftarrow getNextPair(s[X])$;
7      remove $p$ from $s[X]$;
8      $X\_temp \leftarrow$ swapPair($X, p$);
9      $D\_temp \leftarrow$ calculateD($X\_temp$);
10     if $D\_HasImproved(D, D\_temp)$ then
11         $X \leftarrow X\_temp$;
12         $D \leftarrow D\_temp$;
13         $d[X] \leftarrow D$;
14         $s[X] \leftarrow$ getAllSwappableRows($X$);
15     end
16 end
17 $R \leftarrow$ the ranking in $d$ that have the best $D$ value;
18 return $R$;
```

The algorithm takes as input the list of users in a ranking update and a metric D that measures either Disparate Treatment Ratio (DTR) or Demographic Parity Ratio (DPR). First, the algorithm creates two empty dictionaries; in the first, we save the rankings as keys, with the metric D associated to that ranking stored as value and, in the second, we save the ranking as key and the list of possible pairs of users to swap as value. In line 3, we save in the first dictionary (d) the original ranking as key and the respective D as value. Then, in line 4, the function *getAllSwappableUsers* looks for the disadvantaged gender (if $D < 1$, then G_0 is disadvantaged, while if $D > 1$, then G_1 is disadvantaged) and returns a list containing the pairs of indexes of the users that could be swapped, ordered by their occurrence, such that the disadvantaged gender may get more attention. Then, we save the ranking and the users to swap respectively as key, value in the second dictionary (s).

In line 6, we take the first pair of users to swap and check if D has improved (i.e., $abs(1 - D) > abs(1 - D\_temp)$). If this is the case, we save the new ranking and the respective D to d, update the users to swap given this new ranking, and repeat this process until we make sure there are now users that we can swap and that can improve the value of D for the ranking (lines 5-16). Finally, from d we take the ranking that has the best D value.

4.5 Experimental Framework

This section describes the experiments performed to validate our proposal.

4.5.1 Experimental Setup

The experimental framework exploits the Python scikit-learn 0.19.1 library. The experiments were executed on a computer equipped with a 3.1 GHz Intel Core i7 processor and 16 GB of RAM.

The learning phase and consequently the prediction of most Machine Learning classifiers may be biased towards the occurrences that are frequently present in the dataset [RK17, KWMM09].

Researchers have suggested two main approaches to deal with data imbalance: the first approach consists of tuning the data by performing a sampling, and the other is to tweak the learning algorithm [KWMM09]. Due to its effectiveness in our data, we employed the first approach.

More specifically, we have considered the oversampling approach, since it is more effective for small dimension datasets [SKW16]. We opted for *Synthetic Minority Over-sampling Technique Tomek* (SMOTETomek), since it creates completely new samples and eliminates only examples belonging to the majority class instead of replicating the existing ones, which offers more examples to the classifier to learn from. This means that the minority class examples are over-sampled, whereas the majority class examples are under-sampled [CBHK02, BPM04].

In our framework, we applied SMOTETomek using *imbalanced-learn*, which is a package that provides with a bunch of sampling approaches used in datasets showing high class imbalance [LNA17].

4.5.2 Evaluation Strategy

In this section we present our strategy to evaluate our proposal.

Workout Quality Prediction

In order to rank the users that need timely support, we first predict the quality of their performance during the last workout that the coach assigned to them. To this end, we compare two kinds of classification, the first is Ordinal classification (which takes into account the order of ratings) and the second is Multi-class classification (which does not take into account the order of ratings).

Ordinal Classification. In this study, we compared four ordinal classifiers, which consider as classes the ordered set of ratings.

1. *Ordinal Ridge (OR).* This classifier overwrites the Ridge classifier in scikit-learn, so that it uses the (minus) absolute error as score function;

2. *Least Absolute Deviation (LAD).* This classifier optimizes the sum of the absolute errors;

3. *Logistic Immediate-Threshold (LIT).* This classifier implements the ordinal logistic model, considering the Immediate-Threshold variant;

4. *Logistic All-Threshold (LAT).* This classifier implements the ordinal logistic model, considering the All-Threshold variant.

Multi-class Classification. To treat the workout-quality prediction problem as a multi-class classification, we compared four tree-based classifiers, as these perform better compared to those that are not tree-based when it comes to low-dimensional data [RK17].

Gradient Boosting (GB) is an ensemble algorithm that improves the accuracy of a predictive function through incremental minimization of the error term. After the initial base learner (almost always a tree) is grown, each tree in the series is fit to the so-called "pseudo residuals" of the prediction from the earlier trees with the purpose of reducing the error [NK13].

Random Forest (RF) is a meta-estimator of the family of the ensemble methods. It fits a number of decision tree classifiers, such that each tree depends on the values of a random vector sampled independently and with the same distribution for all the trees in the forest [Bre01].

Extra Trees (ET) is another ensemble method. Similarly to Random Forest, it uses a random subset of candidate features while splitting a tree node; however, instead of looking for the most discriminative thresholds, thresholds are drawn at random for each candidate feature and the best of these randomly-generated thresholds is picked as the splitting rule [GEW06].

Decision Tree (DT) is a non-parametric supervised learning method used for classification and regression. One of the main advantages of decision trees with respect to other classifiers is that they are easy to inspect, interpret, and visualize, given they are less complex than the trees generated by other algorithms addressing non-linear needs [BCI+18].

Strategy. To validate our proposal, we performed five sets of experiments:

1. **Classifiers comparison**. We evaluated the ordinal and multi-class classifiers, by running them on all the features. We compared the accuracy metrics they obtained, in order to determine the most effective one;

2. **Feature sets importance evaluation**. After choosing the most effective ordinal and multi-class classifiers, we evaluated the importance of the used features by measuring the correlation between the value of each feature and the values predicted using the best performing classifier, to understand how each feature impacts the quality of workouts;

3. **Ablation study**. We took away the least important features one by one, and evaluated the classification accuracy, to check how the less relevant features affected the effectiveness of the classifiers;

4. **Re-training simulation**. To simulate the real-world scenario, we re-train and monitor the performance of the best ordinal and multi-class classifiers each 100 workouts (i.e., we first train the classifier on the first 100 workouts and evaluate its performance on the following 100 on then we train on the first 200 and evaluate its performance on the following 100).

Ranking Under Fairness Constraints

To rank the users we sort them according to the rating predicted by the best classifier. Then, we compare the effectiveness and the fairness of the resulting ranking, before and after applying Algorithm 1 described in Section 4.4.4.

To simulate the real-world scenario, we re-rank the users for each coach every n new workouts.

4.5.3 Metrics

Workout Quality Prediction. Our approach ranks users on the basis of their workout performance. For this reason, the first set of evaluation metrics should be capable of capturing *how effective* is a classification approach at predicting workout quality. Given that the ground truth is represented by 5-star ratings, we had to choose metrics that are most suitable for multi-class datasets. Nevertheless, the majority of the performance measures present in the literature are designed only for two-class problems [GFB+11].

However, several performance metrics for two-class problems have been adapted to multi-class. Some measures that fit well our needs, give us relevant information about the performance of our classifier, and are successfully applied for multi-class problems are Accuracy, Recall, Precision, F2-score, and Informedness [GFB+11]. In what follows, we present these metrics in detail.

Accuracy is defined as:

$$Accuracy = \frac{TP + TN}{P + N} \tag{4.9}$$

where P represents positively labeled instances, whereas N represents negatively labeled ones. TP represents the true positives (i.e., instances of the positive class that are correctly labeled as positive by a classifier), TN represents the true negatives (i.e., instances of the negative class that are correctly labeled as negative by a classifier). It represents the fraction of all instances that are correctly classified.

Recall is defined as:

$$Recall = \frac{TP}{P} \tag{4.10}$$

and it measures the completeness of a classifier.

Precision is defined as:

$$Precision = \frac{TP}{TP + FP} \tag{4.11}$$

and it measures the exactness of a classifier.

F2-score is defined as:

$$F2 = 5 \cdot \frac{Precision \cdot Recall}{4 \cdot Precision + Recall} \tag{4.12}$$

and it is a metric that considers both recall and precision.

None of the metrics presented so far takes into account the true negative rate (defined as TN/N) and this is an issue when dealing with imbalanced datasets [Pow11]. Considered this, we decided to measure *Informedness*, which is the clearest measure of the predictive value of a system [Pow12]. *Informedness* is defined as:

$$Informedness = Recall + true\_negative\_rate - 1 \tag{4.13}$$

where *true_negative_rate* is TN/N. It ranges between -1 and 1, where 1 represents a perfect prediction, 0 no better than random prediction, and -1 indicates total disagreement between prediction and observation. This metric is particularly effective for multi-class problems as opposite to the accuracy [GFB+11].

Table 4.3: **Ordinal classifiers comparison.** In each line, we report the results obtained by each classifier. The first column indicates the classifier's name, while the following columns are associated with each considered evaluation metric. The values in bold represent the best results for each metric.

Classifier	Accuracy	F2-Score	Recall	Precision	Informedness
OR	**0.917**	**0.917**	**0.917**	**0.919**	**0.883**
LAD	0.575	0.561	0.575	0.601	0.487
LIT	0.803	0.797	0.803	0.827	0.730
LAT	0.790	0.782	0.790	0.815	0.710

Ranking Under Fairness Constraints. To evaluate the ranking quality, we compare the ranking lists generated as output by the model and those given as the ground truth (i.e., the user rankings shaped based on the ratings assigned to each coach for the workouts in the test set). The most suitable metric for this purpose is the Normalized Discounted Cumulative Gain (NDCG).

We compared our rankings effectiveness using an exponential gain and logarithmic decay based on the graded relevance judgments. In our case, NDCG at position k is defined as:

$$NDCG@k(R) = \frac{1}{N} \sum_{j=1}^{k} \frac{2^{rel(u_j)} - 1}{\log(j+1)} \tag{4.14}$$

where N is the maximum possible DCG given the known relevant users, u_j is the u^{th}-ranked user returned by R, and $rel\,(u_j)$ is the binarized relevance assessment of this user [RC10]. NDCG values range between 0 and 1; the higher the value, the better.

4.5.4 Experimental results

In this section, we present our results.

Classifiers comparison

- **Ordinal classification.** Table 4.3 (visually presented in Figure 4.6) shows that LIT, LAT, and OR achieved a good performance, where LAD achieved the worst results. The ordinal classifier that gets the best scores for all the metrics is OR. It achieves an F2-Score of almost 92% and an Informedness of 0.88, which means that we are correctly predicting the rating of a workout in 92% or more of the cases. Based on these results, OR is the ordinal classifier chosen for the subsequent analyses.

- **Multi-class classification.** Table 4.4 (visually presented in Figure 4.7) shows that almost all the classifiers have a good performance, but RF is the one that

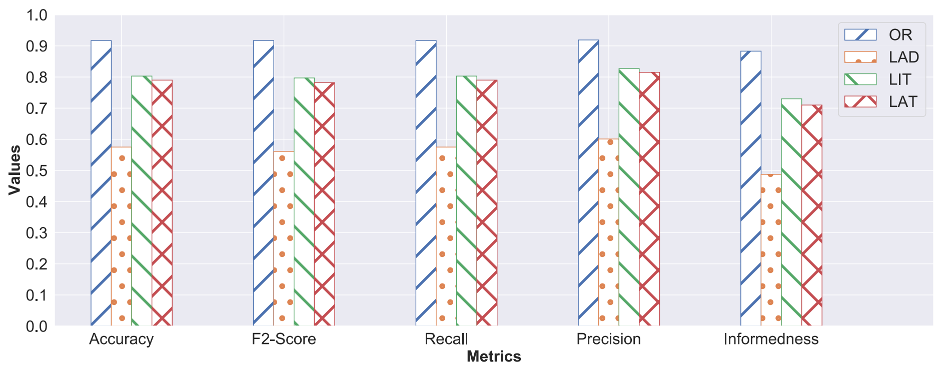


Figure 4.6: **Ordinal classifiers comparison.** Each block of columns reports the results obtained for each metric. Each column denotes an ordinal classifier. The higher the value, the better is the classifier.

Table 4.4: **Multi-class classifiers comparison.** In each line, we report the results obtained by each classifier. The first column indicates the classifier's name, while the following columns are associated with each considered evaluation metric. The values in bold represent the best results for each metric.

Classifier	Accuracy	F2-Score	Recall	Precision	Informedness
GB	0.925	0.925	0.925	0.927	0.899
RF	**0.929**	**0.929**	**0.929**	**0.930**	**0.912**
ET	0.900	0.900	0.900	0.901	0.879
DT	0.916	0.916	0.916	0.916	0.901

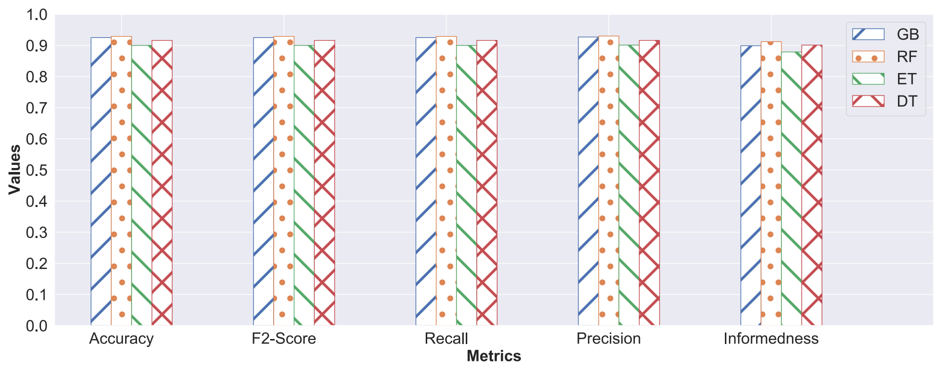


Figure 4.7: **Multi-class classifiers comparison.** Each block of columns reports the results obtained for each metric. Each column denotes a multi-class classifier. The higher the value, the better is the classifier.

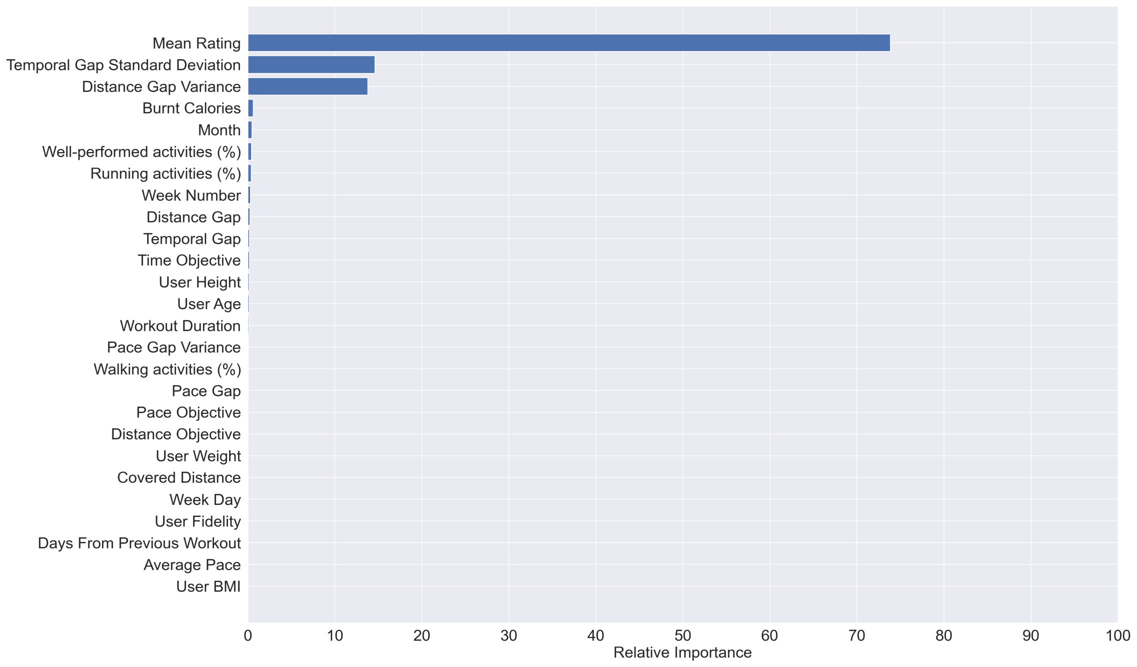


Figure 4.8: **Features' importance for the OR classifier.** Each line reports the relative importance of a feature, in a score between 0 and 100. The higher is the score, the more important is the feature.

gets the best scores for all the metrics. Concretely, RF achieves an F2-Score of almost 93% and an Informedness of 0.91, which means that we are correctly predicting the rating of a workout in 93% or more of the cases. Based on these results, RF is the multi-class classifier chosen for the subsequent analyses.

- **Ordinal vs. Multi-class classification.** The best ordinal classifier and the best multi-class classifier achieve a similar performance for the ratings prediction task, nevertheless, RF performs slightly better than OR for all the metrics. This is true for all the metrics we consider to evaluate classification quality. This leads us to our first observation.

Observation 1. *The ratings that the coaches use to assess workout quality are in a continuous scale and, conceptually, an ordinal classifier would better suit this task. However, the multi-class classifiers outperform the ordinal ones. Hence, we conjecture that coaches might have a more schematic way of evaluating workouts, better captured by multi-class approaches.*

Feature sets importance evaluation.

Fig. 4.8 illustrates the impact of each feature on the performance of OR, using a scale ranging from 0 (no importance) to 100 (very important). We can see that

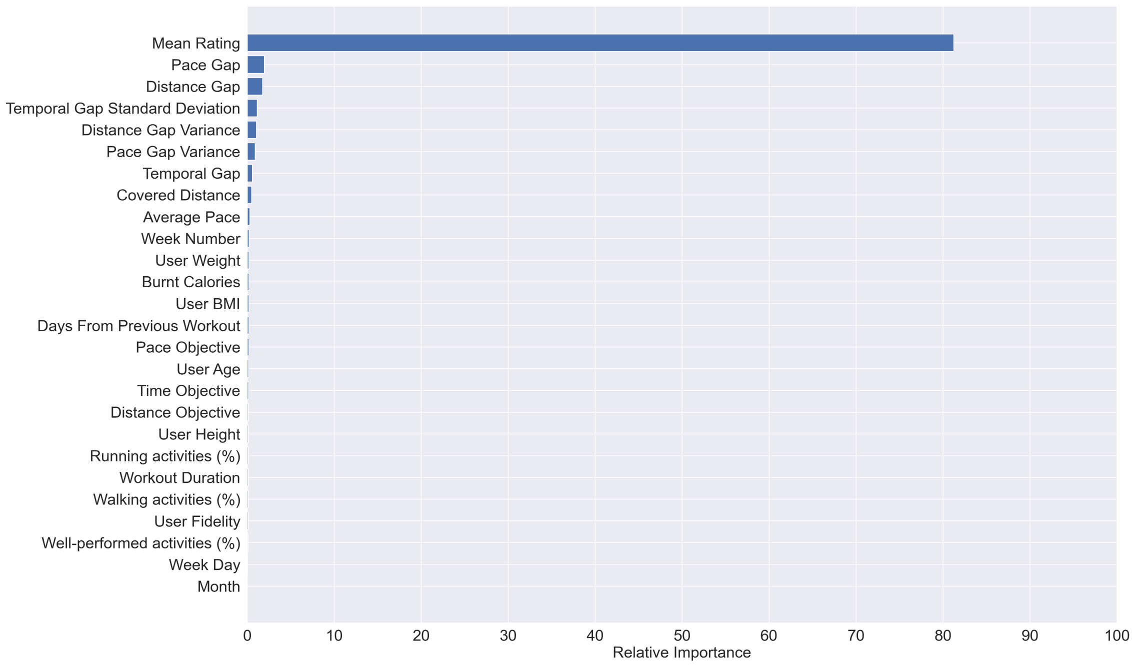


Figure 4.9: **Features' importance for the RF classifier.** Each line reports the relative importance of a feature, in a score between 0 and 100. The higher is the score, the more important is the feature.

the features that have more impact on the classification process are mainly those that model the recent behavior of the users and their adherence to their workout objectives. The mean rating is the most important feature, and we assume that this is due to the fact that it represents the decaying average of the recent ratings achieved by the users, and since users usually tend to change their behavior gradually their performance is very correlated with their recent ratings. We can see also that the effort (Burnt Calories), the month when the workout sessions were planned, and the percentage of well-performed activities have a significant impact on the workout quality prediction. However, user characteristics were not very relevant to the classifier.

Fig. 4.9 illustrates the impact of each feature on the performance of RF, using a scale ranging from 0 (no importance) to 100 (very important). We can see that the features that have more impact on the classification process are mainly those that model the recent behavior of the users and their adherence to their workout objectives. The mean rating is the most important feature also according to RF. We can see also that the covered distance, average pace, and the week number have a significant impact on the workout quality prediction. However, user characteristics and workout characteristics were not very relevant to the classifier.

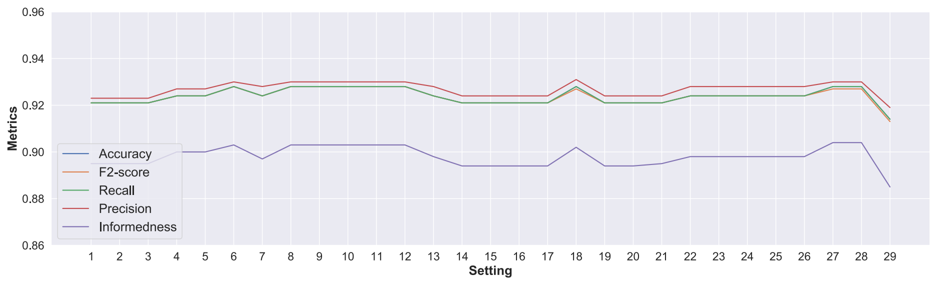


Figure 4.10: **Results returned by training OR with different sets of features.** For each set of features, denoted in the x axis (Setting), we report the value obtained by each metric.

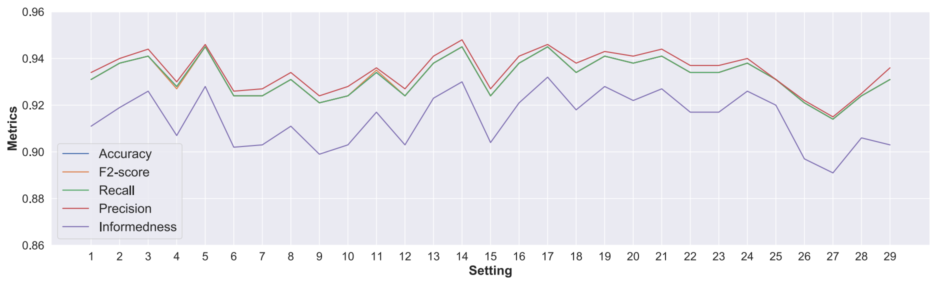


Figure 4.11: **Results returned by training RF with different sets of features.** For each set of features, denoted in the x axis (Setting), we report the value obtained by each metric.

Ablation study.

During the ablation study, we train the classifier on different feature settings by removing features one by one, starting from the least important (i.e., for OR, the first setting runs the classifier without the User BMI, while in the second setting we removed User BMI and Average Pace, and so on).

Training OR on fewer features showed that it achieves a better performance using the feature set 18 (i.e., when we do not consider the first 18 less important features while training the classifier), as reported in Figure 4.10.

Training RF on fewer features showed that it achieves a better performance using the feature set 14 (i.e., when we do not consider the first 18 less important features while training the classifier), as reported in Figure 4.11.

Table 4.5 shows the best performance of OR and RF after training them on fewer features. Both classifiers achieve a very good performance, though RF outperforms OR for all the metrics we considered.

Table 4.5: **Performance of OR and RF when trained on the best feature sets.** Each line reports the results of a metric and each column the classifier associated with the reported results.

Classifier	OR	RF
Accuracy	0.928	0.945
F2	0.927	0.945
Recall	0.928	0.945
Precision	0.931	0.948
Informedness	0.902	0.930

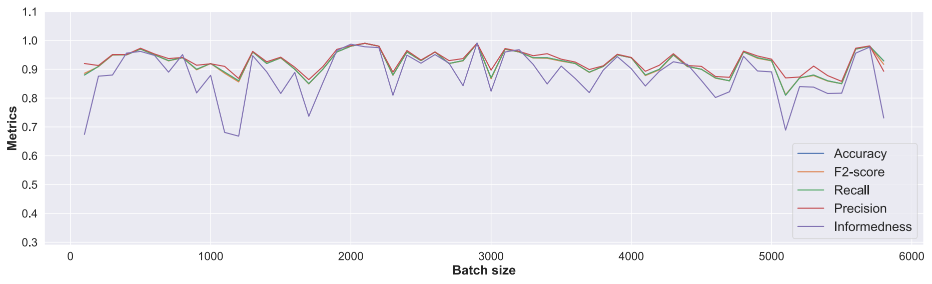


Figure 4.12: **Evolution of the performance of OR.** The x axis (Batch size) contains a point every 100 workouts, that is when a classifier gets retrained. The y axis (Metrics) reports the value obtained by each metric with the associated batch size.

Observation 2. *Regardless of the users' characteristics and how a workout is made up, the workout quality depends above all on how much the runners stick to their workout objectives and how much effort they are putting during workouts. Apart from being adherent to the goals set by the coach, the period of the year when the workouts are planned can also influence the performance of runners; we conjecture that this last phenomenon means that good weather positively influences workout quality.*

Re-training simulation.

In this evaluation, we re-train and assess the effectiveness of the classifiers every 100 workouts.

Considering ordinal classification, Figure 4.12 shows that OR maintains a good performance over time, with the F2-score values ranging between 81% and 99%. A peculiarity of this classifier is that it can predict effectively even when training on a subset of workouts.

Considering the most effective multi-class classifier, Figure 4.13 shows that when training on fewer workouts (less than 300) the performance of RF is low but, when

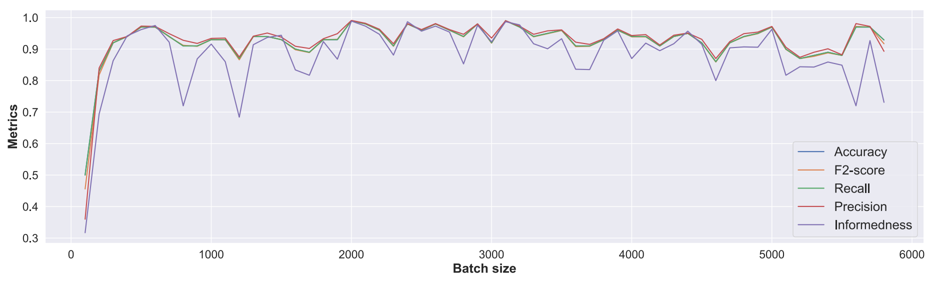


Figure 4.13: **Evolution of the performance of RF.** The x axis (Batch size) contains a point every 100 workouts, that is when a classifier gets retrained. The y axis (Metrics) reports the value obtained by each metric with the associated batch size.

training on 300 workouts or more, the classifier maintains a good performance over time (F2-score ranges between 86% and 99%).

Ranking

For each coach, we started by ranking the users that performed the first 5 workouts, and we updated the rankings each new 5 workouts. We do this for the last 50 workouts performed by the users followed by all the coaches. Then, we mitigated unfairness for each ranking update w.r.t. our disparate treatment metrics, DPR and DTR.

- **Ranking quality.** Figure 4.14 shows the evolution of the average NDCG@10 over time before and after mitigating unfairness in ordinal and multi-class based rankings. We notice that before mitigating unfairness, all the rankings achieved an NDCG@10 of 1 using both multi-class based and ordinal based rankings. This means that the ratings predicted by the classifier reflect both workout performance and the timeliness with which sportspeople should be contacted. After mitigating unfairness, we remark that the values of NDCG@10 get lower as the number of ranking updates grows. This could be explained by the fact that, while we are mitigating unfairness, we reorder the users such that they get assisted in a fair way and this influences the quality of rankings. However, we can see that after mitigating unfairness, the ordinal based rankings maintain a slightly higher NDCG@10 than the multi-class based ones.

- **Global evolution of DPR and DTR over time before and after mitigating unfairness.** Figure 4.15 illustrates the evolution of the average $|DPR - 1|$ (how far is DPR w.r.t. its perfect value) before and after mitigating unfairness in ordinal and multi-class based rankings for each ranking

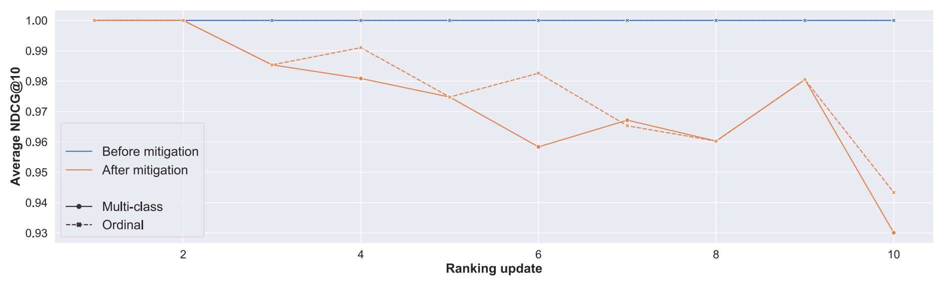


Figure 4.14: **Ranking accuracy results.** The x axis (Ranking update) contains a point every 5 workouts, that is when a classifier gets retrained. The y axis (Average NDCG@10) reports the NDCG@10 obtained by each classifier in the associated ranking update. We report these results before and after mitigation (blue and orange line, respectively), for multi-class and ordinal classifiers (continuous and dashed lines, respectively).

update. From Figure 4.15, we see that over time the average $|DPR - 1|$ gets closer to 0 after applying Algorithm 1 to mitigate unfairness, but its values get higher as the number of ranking updates grows. The values of $|DPR - 1|$ for ordinal and multi-class based rankings are very similar, though, ordinal based rankings achieved a slightly better DTR compared to multi-class based rankings.

Figure 4.16 illustrates the evolution of the average $|DTR - 1|$ (how far is DTR w.r.t. its perfect value) before and after mitigating unfairness in ordinal and multi-class based rankings for each ranking update. From Figure 4.15, we see that, like $|DPR - 1|$, over time the average $|DTR - 1|$ gets closer to 0 after applying Algorithm 1 to mitigate unfairness, but its values get higher as the number of ranking updates grows. Nevertheless, the values of $|DTR - 1|$ are closer to 0 comparing to the values of $|DPR - 1|$ before and after mitigating unfairness.

In contrast with what we have seen in Figure 4.15, according to DTR the multi-class based strategy is the one that generates more fair rankings.

Figure 4.17 illustrates the evolution of DPR before and after mitigating unfairness in ordinal and multi-class based rankings for each ranking update. From Figure 4.17, we see that the average DPR mostly ranges between 0.95 an 1.09 for all the ranking updates in ordinal and multi-class based rankings. Before mitigating unfairness, the values of DPR are ranging between 1 and 1.09 and after unfairness mitigation the values are ranging between 0.95 and 1.06. We can deduce that the values of DPR after mitigation vary more, but are closer to 1.

In addition, we can notice that in majority of cases the discriminated gender

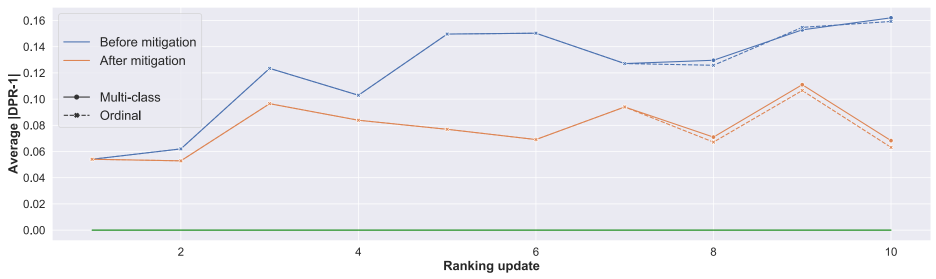


Figure 4.15: **Fairness in terms of demographic parity.** The x axis (Ranking update) contains a point every 5 workouts, that is when a classifier gets retrained. The y axis (Average $|DPR - 1|$) reports the distance of each classifier with respect to the expected DPR score in a ranking update. We report these results before and after mitigation (blue and orange line, respectively), for multi-class and ordinal classifiers (continuous and dashed lines, respectively).

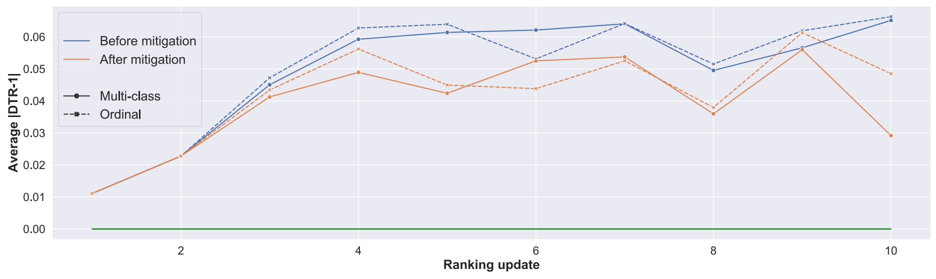


Figure 4.16: **Fairness in terms of disparate treatment.** The x axis (Ranking update) contains a point every 5 workouts, that is when a classifier gets retrained. The y axis (Average $|DTR - 1|$) reports the distance of each classifier with respect to the expected DTR score in a ranking update. We report these results before and after mitigation (blue and orange line, respectively), for multi-class and ordinal classifiers (continuous and dashed lines, respectively).

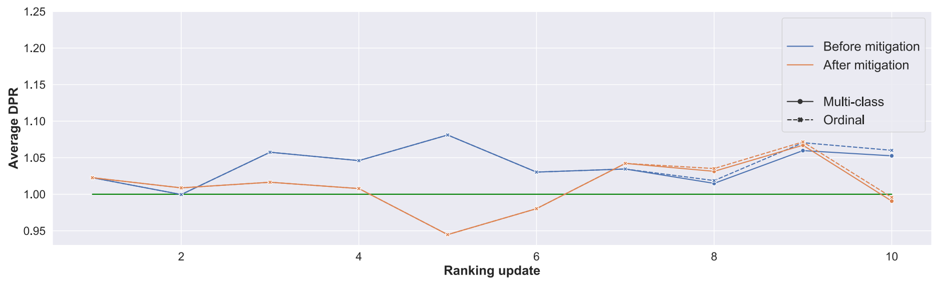


Figure 4.17: **Demographic parity scores.** The x axis (Ranking update) contains a point every 5 workouts, that is when a classifier gets retrained. The y axis (Average DPR) reports raw DPR score returned in a ranking update. We report these results before and after mitigation (blue and orange line, respectively), for multi-class and ordinal classifiers (continuous and dashed lines, respectively).

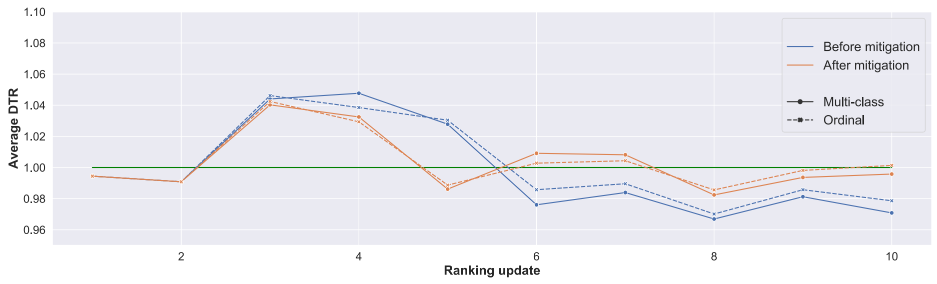


Figure 4.18: **Disparate treatment scores.** The x axis (Ranking update) contains a point every 5 workouts, that is when a classifier gets retrained. The y axis (Average DTR) reports raw DTR score returned in a ranking update. We report these results before and after mitigation (blue and orange line, respectively), for multi-class and ordinal classifiers (continuous and dashed lines, respectively).

in terms of demographic parity is the male gender ($DPR > 1$).

Figure 4.18 illustrates the evolution of DTR before and after mitigating unfairness in ordinal and multi-class based rankings for each ranking update. Figure 4.18 shows that the average DTR values mostly range between 0.97 and 1.05 for all the ranking updates in ordinal and multi-class based rankings. Before mitigating unfairness the values of DTR are ranging between 0.97 and 1.05, meanwhile after the mitigation the values are ranging between 0.98 and 1.04. We can deduce that the values of DTR after mitigation are less variate comparing to the values of DPR and closer to 1. Furthermore, we notice that for almost all the ranking updates, the ordinal classification based rankings are achieving better results compared to multi-class classification based rankings with respect to DTR.

Moreover, we notice that, in terms of disparate treatment, the discriminated gender is the female gender ($DTR < 1$), unlike what we have observed earlier when assessing fairness using DPR.

At this point, one may pose the question: *Which metric is telling the truth about the discriminated gender?* Both metrics are somehow right about the discriminated group, except that DPR is not considering the performance of sportspeople when measuring unfairness, while DTR also includes the utility of the rankings instead, and thus considers also the performance of sportspeople when assessing unfairness. For this reason, we may consider that DTR is more suited to our application's context, especially for the fact that not considering the utility of rankings when mitigating unfairness could influence negatively the quality of the users' experience by attributing sportspeople an exposure that is not proportional to their performance during their last workout session.

To explore more in depth this phenomena, we represented graphically the evolution of DPR and DTR before and after mitigating unfairness in all the rankings where females are more than males and the ones where females are more than males (Figures 4.19, 4.20, 4.21, and 4.22). We analyze our results in our following analysis.

- **Evolution of DPR and DTR over time before and after mitigating unfairness when females are more than males.** Figure 4.19 illustrates the evolution of DPR in the rankings where females are more than males, before and after mitigating unfairness in ordinal and multi-class based rankings for each ranking update. We observe that in most cases the discriminated group is the male group, which is the minority group in this case, and that the minority group gets ranked in a more unfair way as the number of ranking updates increases. This could be explained by the fact that, as the rankings are updated, the number of ranked users gets larger and more diverse which makes the original rankings more unfair. After mitigating unfairness the average DPR gets closer to its perfect value, and the discriminated group could change for

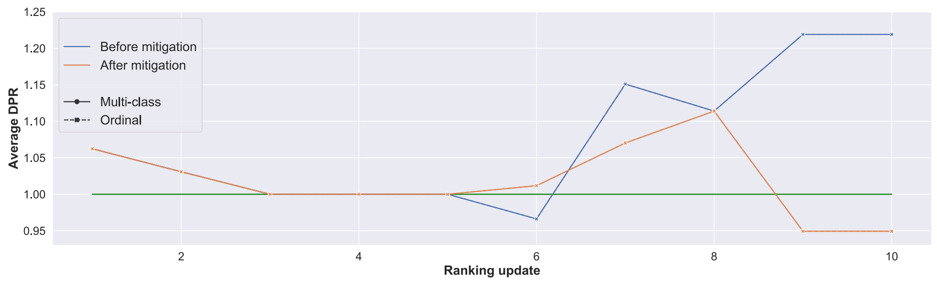


Figure 4.19: **Demographic parity scores, when females are more than males.** The x axis (Ranking update) contains a point every 5 workouts, that is when a classifier gets retrained. The y axis (Average DPR) reports raw DPR score returned in a ranking update. We report these results before and after mitigation (blue and orange line, respectively), for multi-class and ordinal classifiers (continuous and dashed lines, respectively).

some ranking updates. Hence, we do not observe any difference in terms of DPR values when comparing ordinal and multi-class based ranking strategies.

Figure 4.20 illustrates the evolution of DTR in the rankings where females are more than males, before and after mitigating unfairness in ordinal and multi-class based rankings for each ranking update. According to DTR the discriminated group is mostly the one of males, and after the mitigation of unfairness the average DTR got closer to 1 in all the cases for the ordinal classification based ranking strategy. Instead, for the multi-class classification based rankings, we can notice that in one case the average DTR in higher than the average DTR after mitigating unfairness, and this could be explained by the fact that the values of DTR for that ranking update are more variate than before mitigating unfairness.

- **Evolution of DPR and DTR over time before and after mitigating unfairness when females are less than males.** Figure 4.21 illustrates the evolution of DPR in the rankings where females are less than males, before and after mitigating unfairness in ordinal and multi-class classification based rankings for each ranking update. This figure shows that the discriminated group according to DPR before mitigating unfairness is the male group, when it appears that after mitigating unfairness the discriminated group is mostly the one of females. Since, the values of average DPR before and after mitigating unfairness are very close for the multi-class and the ordinal classification based ranking strategies. Figure 4.22 illustrates the evolution of DTR in the rankings where females are less than males, before and after mitigating unfairness in ordinal and multi-class classification based rankings for each ranking update. According to the average DTR, the discriminated group for both ranking strategies is mostly the one of females before and after mitigating

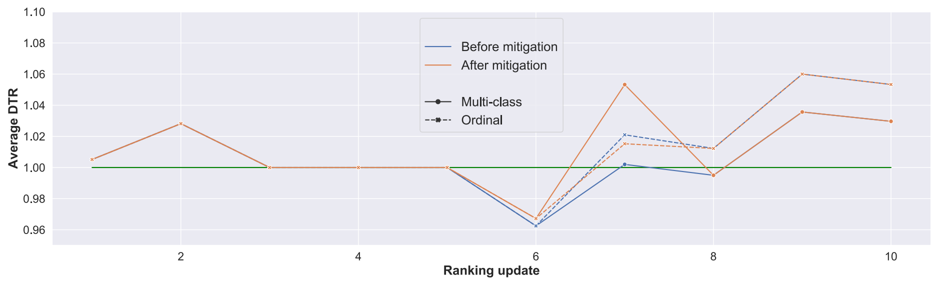


Figure 4.20: **Disparate treatment scores, when females are more than males.** The x axis (Ranking update) contains a point every 5 workouts, that is when a classifier gets retrained. The y axis (Average DTR) reports raw DTR score returned in a ranking update. We report these results before and after mitigation (blue and orange line, respectively), for multi-class and ordinal classifiers (continuous and dashed lines, respectively).

unfairness.

Observation 3. *The discriminated gender when assessing fairness using DTR coincides with the gender of the minority group. This phenomenon aligns our work with what is usually observed in the fairness literature, where the demographic group representing the minority in the training data is the discriminated one [BFM20].*

4.6 Conclusions and Future Work

In this chapter, we proposed and validated an approach to identify and rank sportspeople that need timely support due to low performance in workouts and recommend them to their coaches so that they can be contacted with a higher priority. Furthermore, we guarantee a fair exposure in the ranking, to make sure that users of different genres have equal opportunities to get supported. Our approach models the performance and running behavior of the users, in order to apply a ranking algorithm to recommend users to coaches, according to their performance in the last running session and the quality of the previous ones. Then, we presented a re-ranking algorithm to provide fair exposure to users.

As future work, we look to introduce explainability and coach-in-the-loop insights to improve the recommendations. Furthermore, we are currently preparing a live user-evaluation, to see how coaches perceive our ranking and the fairness dimensions we introduced in this work.

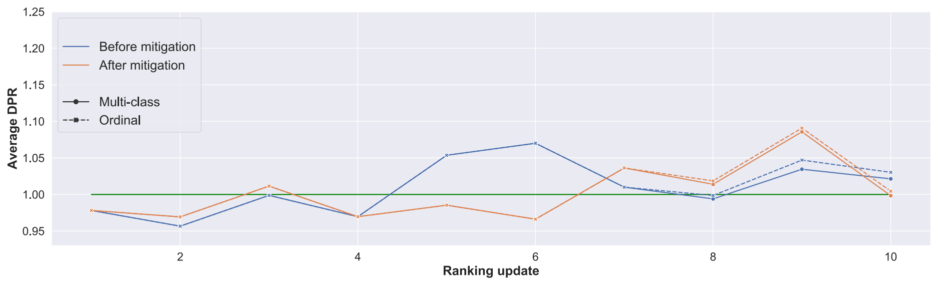


Figure 4.21: **Demographic parity scores, when females are less than males.** The x axis (Ranking update) contains a point every 5 workouts, that is when a classifier gets retrained. The y axis (Average DPR) reports raw DPR score returned in a ranking update. We report these results before and after mitigation (blue and orange line, respectively), for multi-class and ordinal classifiers (continuous and dashed lines, respectively).

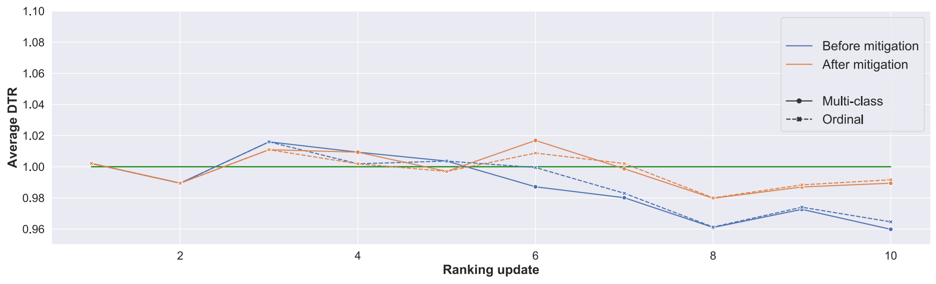


Figure 4.22: **Disparate treatment scores, when females are less than males.** The x axis (Ranking update) contains a point every 5 workouts, that is when a classifier gets retrained. The y axis (Average DTR) reports raw DTR score returned in a ranking update. We report these results before and after mitigation (blue and orange line, respectively), for multi-class and ordinal classifiers (continuous and dashed lines, respectively).

Chapter 5

Machine Learning Models for Sportspeople Dropout Prediction

5.1 Introduction

Engagement is a key aspect in platforms that aim to support the users, such as eCoaching platforms [BCMP17]. Besides motivational aspects, when eCoaching is associated to health-related issues, such as keeping a healthy lifestyle, keeping users engaged assumes extra importance. Sportspeople engagement can take several shapes, from follow-up messages after workouts, through ensuring that sportspeople work out at the right pace, to avoiding sportspeople dropout of the platform.

In this chapter, we focus on this last scenario, to *predict sportspeople's dropout in an eCoaching platform*. Specifically, we consider a real-world eCoaching platform, where users receive from coaches weekly training plans. Our goal is to predict if users will continue working out in the week following the current one, by analyzing their behavior. Current approaches, deal with this task with a *workout-centric approach*; considering the results of a workout, they predict if a user will continue working out in the week following the workout, with classic classification algorithms that are trained with 2-dimensional vectors [PPB+17]. This approach is clearly limited from multiple perspectives as: (*i*) it does not consider the *temporal evolution* of the workouts, which limits the training of classification algorithms, as they are not learning the evolution in the way users work out; (*ii*) it cannot model complex scenarios with tensors that go beyond two dimensions; this means that, also at an algorithmic level, to model the interplay between sportpeople, their workout, and time, we need ad-hoc approaches to predict the dropout; (*iii*) it cannot keep track, during the training, of the whole history of the users, since 2-dimensional vectors usually represent a single workout of a user.

Our contributions In this work, we tackle the under-explored perspective of the support that coaches provide to users. A coach usually does a daily monitoring of the sportspeople they follow and of their performance, to track their adherence to the workout plan they prepared. However, the role of a coach should be to go beyond this day-to-day job, to be able to track possible losses in motivation. Given the high amount of users a coach follows,[1] it is challenging to track these phenomena, which usually involve the contextualization of the last/recent workout(s) of a user with their history and with what the coaches expects from them.

To address this problem, in this chapter we propose an approach to *predict if a user will workout in the week following the last workout.* We propose a neural architecture to predict sportspeople's dropout from eCoaching platforms, capable of modeling users' workout behavior and its evolution over time. Specifically, our modeling and prediction are based on a novel a Tri-branch convolutional neural network that captures workout persistence, changes in terms of workout performance, and the combination of both, to predict if a user will work out in the week following the considered one or not. This week-based prediction is related to the platform we consider in this study, where workout plans are given to users on a weekly basis.

We evaluate the effectiveness of this approach in offline fashion, considering real-world data collected from the aforementioned eCoaching platform for running activities. Results shows its effectiveness against different baselines architectures. This classification for dropout prediction has also been integrated in the platform and is currently being evaluated by the coaches, in applied research fashion.

Specifically, our work provides the following contributions:

- We model users' workout behavior in running, considering the temporal perspective of the workouts;
- We present a novel neural architecture to predict sportspeople workout;
- We evaluate our approach on real-world data and against state-of-the-art classification approaches and alternative architecture, showing the effectiveness of our approach.

Roadmap The rest of the chapter is structured as follows. Section 5.2 describes the dataset we used in this study. Then, we present our approach in Section 5.3, which is validated in Section 5.5. In Section 5.4 we present our experimental framework. Finally, we provide concluding remarks in Section 5.6.

5.2 Data Description

Dataset. This research work is based on data collected by means of a real-world eCoaching platform. The dataset contains 31,833 activities that compose 6,315 workouts. This means that each workout is composed by several activities. Each

[1]In the platform considered in this thesis, a coach follows on average 21.3 users.

activity has a type (Running, walking, resting, extra), an objective (in terms of pace, or in terms of distance or time) and a result (in terms of pace, distance, and time). To the best of our knowledge, this is the only dataset containing historical information about users and their workouts.

Each workout result is represented by the following aggregate statistics: (i) Objective distance; (ii) Objective time; (iii) Objective pace; (iv) Covered distance; (v) Elapsed time; (vi) Average pace; (vii) Burnt calories; (viii) Percentage of running activities; (ix) Percentage of walking activities; (x) Percentage of activities that have pace as objective; (xi) Percentage of activities that have distance as objective; (xii) Percentage of activities that have time as objective; (xiii) Percentage of activities that does not have a specific objective; (xiv) Percentage of activities that are extra; (xv) Device type (0 for the results uploaded via smartphone and 1 for the results uploaded via other wearable devices); (xvi) Month in which a sportsperson performed the workout; (xvii) Week day in which a sportsperson performed the workout; (xviii) Week of year in which a sportsperson performed the workout; (xix) Year in which a sportsperson performed the workout; (xx) User gender; (xxi) User age; (xxii) Number of days from the first day in which the user started using the platform; (xxiii) Number of weeks from the first day in which the user started using the platform; (xxiv) Number of workouts performed by the user from the first day they started using the platform; and (xxv) Number of workouts performed by the user during the same week.

Data pre-processing. From all the workouts in the dataset, we removed all those that are not reliable. A workout is not reliable when at least one of the following conditions is met: (i) $covered\ distance > 43,000\ meters$, (ii) $workout\ duration > 5\ hours$, (iii) $rest\ time > 1\ hour$, (iv) $average\ speed > 16\ km/h$, and (v) $burnt\ calories < 3000$. We also removed the workouts that were not performed under the supervision of a coach. The final dataset contains 5,166 workouts performed by 73 users.

Data split. As described in Section 5.3.1, we relied on these data to model the users' history. For each user, we selected the sets of workouts they performed each week from the first time in which they used the platform, and exploiting all the workouts they performed before the current week we predict whether they will train at least once during the week that follows. For each week, 70% of the data was used for training, 20% of the data was used for testing, and the rest for validation.

Data analysis. Figure 5.1 shows the number of users that worked out, at least once, during the first four weeks in which they used the platform. We chose this time span, since it allowed us to show an entire subscription period to the workout plans, and it covers a relevant part of our user base. In the figure, we show the behavior of the users considering the entire user base and only those that will (not) dropout in a certain week.

As the analysis shows, the number of users working out reduces as the number of weeks progresses and this is true for the scenarios we considered, minus the users

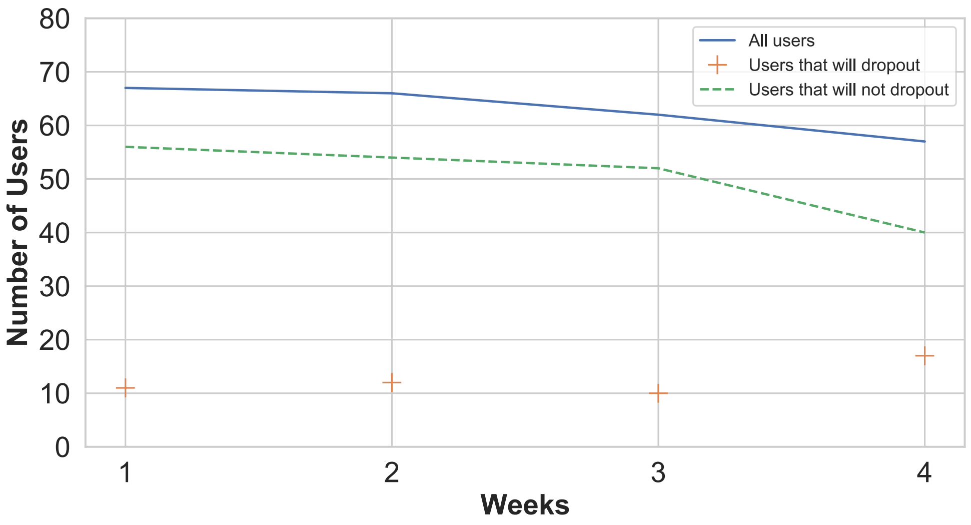


Figure 5.1: Distribution of users over weeks.

who will dropout, who remain more or less constant over time. Hence, we can observe that the users start loosing motivation as the number of weeks gets larger, since the number of sportspeople working out shrinks from a week to another.

This drop in the amount of users over the weeks is especially true after the third week. Considering that, after the fourth week, users are expected to renew their subscription plan, this phenomenon shows how important it is to spot possible losses in motivation. We can see that the most severe drop in the amount of users after the third week can be observed for the users who will not drop out during the subscription plan (green line). These are the most engaged users, since they consistently workout during the subscription period, so it is important to keep them engaged also when they have to renew their plan.

This observational analysis strengthens the motivation to our approach and leads us to our first observation.

Observation 1. *Users lose motivation over time. This is true during the subscription period (first 4 weeks), but becomes more severe as the subscription renewal gets closer. Hence, it is crucial for coaches to be notified timely about sportspeople losing motivation, so that they can provide an adequate support and engage with them to train.*

5.3 Prediction Strategy

In this section, we present our approach and compare it to different prediction strategies. In Section 5.3.1, we formulate the problem of sportspeople weekly dropout prediction. Then, in Section 5.3.2, we present a classic machine learning based approach to sportspeople weekly dropout prediction. Finally, we detail the architecture of the proposed Tri-branch Convolutional Neural Network (Tri-branch CNN) for sportspeople weekly dropout prediction in Section 5.3.3.

5.3.1 Problem Formulation

To address the problem of predicting sportspeople dropping out due to losing motivation to exercise, we model users to capture their behavior change over the weeks. Specifically, for each sportsperson, we collect the workout statistics for each week in which they used the platform. To model the users that trained in the i-th week and predict if they will workout during the next week, each sportsperson is represented as a third-order tensor $X : \mathbb{R}^{|F|\times|W|\times i}$, where $|F|$ denotes the number of features collected by the mobile application combined with data from other sensing devices, $|W|$ denotes the maximum number of workouts performed in one week by all the considered sportspeople from their first week using the platform until the i-th week (i.e. if a certain user performed a maximum number of workouts per week equals to n before the i-th week and another user performed a maximum number of workouts per week equals to m, if $m > n$ then $|W| = m$), while i denotes the number of weeks in which the sportsperson trained using the platform. If the sportsperson performed at least a workout during the i-th week, we set $X(i-1, w, f) = 0$, otherwise $X(i-1, w, f)$ equals one. For the next-week training dropout prediction, we seek to obtain a mapping function $f_\theta : X \to \mathbb{R}$, which yields a real value $f_\theta(\boldsymbol{X})$ indicating the dropout risk of each sportsperson.

5.3.2 Classic Machine Learning Prediction Strategies

Since classic machine learning algorithms does not support third-order tensors, we first flatten the data from each week of training such that each user is represented by a simple feature vector, then, we train and compare the performance of the classifiers over the weeks. In this approach, we compared three tree-based classifiers, as these perform better compared to others, which are not tree-based when it comes to low dimensionality data [RK17].

Gradient Boosting (GB) is an ensemble algorithm that improves the accuracy of a predictive function through incremental minimization of the error term. After the initial base learner (almost always a tree) is grown, each tree in the series is fit to the so-called "pseudo residuals" of the prediction from the earlier trees with the purpose of reducing the error [BM12].

Random Forest (RF) is a meta-estimator of the family of the ensemble methods. It fits a number of decision tree classifiers, such that each tree depends on the values of a random vector sampled independently and with the same distribution for all the trees in the forest.

Decision Tree (DT) is a non-parametric supervised learning method used for classification and regression. One of the main advantages of decision trees with respect to other classifiers is that they are easy to inspect, interpret, and visualize, given they are less complex than the trees generated by other algorithms addressing non-linear needs [PPC+18].

5.3.3 Deep Learning Prediction Strategies

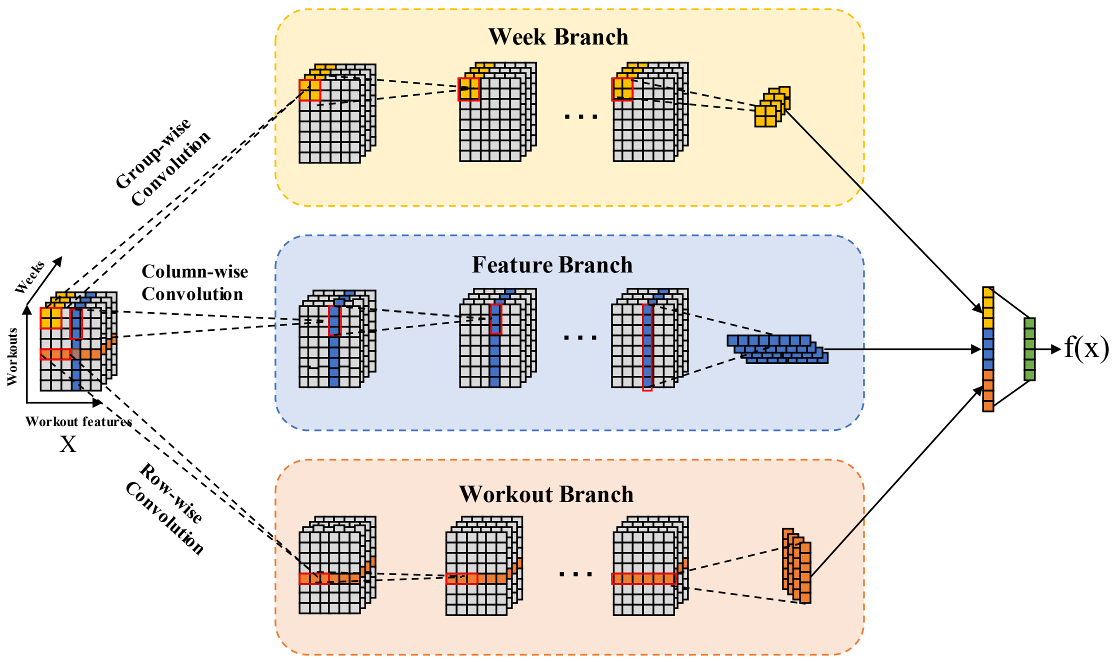


Figure 5.2: The structure of the proposed Tri-branch CNN.

Tri-branch CNN

The intuition behind the choice of this architecture (Tri-branch CNN) comes from the scenario in which human coaches would analyze the behavior of the sportspeople they follow in the real world; Concretely, the coaches are likely to investigate the workout history of their coachees from different perspectives:

- Their periodicity with which sportspeople workout during each week;
- Their performance change from a workout to another;
- Their behavior changes over the weeks.

For this reason, we based our approach on an end-to-end Convolutional Neural Network, namely Tri-branch CNN, which takes as input the workout history of the sportspeople from their first week using the platform to the current week and predicts if they will train or not during the next week. The network accepts the tensor representation X_i of each spostsperson at the i-th week, and is made up of three branches, i.e., workout branch, feature branch, and week branch. The three branches are equipped with row-wise convolution, column-wise convolution, and group-wise convolution, to capture the workout frequency, performance, and week-aware characteristic of sportspeople behavior, respectively. Finally, we merge the results of the three branches to obtain the output $f_\theta(\boldsymbol{X_i})$ via two fully-connected layers.

Figure 5.2 displays the architecture of the proposed Tri-branch CNN. In what follows, we detail how we built each branch.

Workout branch: The way a sportsperson workout frequency changes each week may influence their workout behavior during the next week. We adopt a row-wise convolution of size $1 \times |F|$ in the workout branch, which models the changes of the workouts on each week. At the end of the network branch, we flatten each feature map, to obtain a vector of size $|W| \times 1$, which encodes the sportsperson's workout behavior on each week.

Feature branch: The way the sportsperson behaves during each workout session may influence their motivation to workout. As we did for the week branch, we use a column-wise convolution of size $|W| \times 1$ in the feature branch, which captures the changes of a certain feature during all the workouts performed during the week. In the last layer of the branch, we shrink each feature map through the workout dimension, resulting a vector of size $1 \times |F|$ that represents the periodicity of a sportsperson's behavior for each workout feature.

Week branch: Through the week branch, we capture how the overall behavioral changes of the sportspeople from a week to another. We introduce to the week branch a group-wise convolution of size $|W| \times |W|$ and flatten the result of the convolution to a vector that lets us observe the behavioral changes on each week, and thus model the characteristic of a sportsperson's weekly behavior over the entire range of features and workouts.

In the three branches, we added after each convolution layer a batch normalization layer to reduce the internal covariance shift issue caused by the change in the distribution of network activations due to the change in network parameters during training, which contributes also to accelerating the training of our neural networks, as highlighted by *Sergey Ioffe and Christian Szegedy* in [IS15].

Note that the shape of the input layer depends on the week for which we are predicting, thus, the kernel size in each convolution layer varies from a week to another. But, the number of filters of the first layer was set to 32 which is the closest power of 2 to the number of features (26 features), then the number of filters in the other convolution layers was set respectively to 16 and 8. We adopted the area under curve (**AUC**) as the performance metric.

5.4 Experimental Framework

This section describes the experiments performed to validate our proposal. Section 5.4.1 described the experiments performed using classic machine learning based approaches, and Section 5.4.2 describes the experiments performed using deep learning based approaches. The experimental framework exploits the Python Tensorflow 2.5.0 and scikit-learn 0.22.1 libraries. The experiments were executed on a computer equipped with a 3.1 GHz Intel Core i7 processor and 16 GB of RAM. Each classification was repeated 10 times with a 10-fold cross-validation.

5.4.1 Classic Machine Learning Prediction Strategies

We trained each model on default parameters. The learning phase and consequently the prediction of most Machine Learning classifiers may be biased towards the occurrences that are frequently present in the dataset [RK17, KWMM09].

Researchers have suggested two main approaches to deal with data imbalance: the first approach consists of tuning the data by performing a sampling, and the other is to tweak the learning algorithm [KWMM09]. Due to its effectiveness in our data, we employed the first approach.

More specifically, we have considered the oversampling approach, since it is more effective for small dimension datasets [SKW16]. We opted for *Synthetic Minority Over-sampling Technique Tomek* (SMOTETomek), since it creates completely new samples and eliminates only examples belonging to the majority class instead of replicating the existing ones, which offers more examples to the classifier to learn from. This means that the minority class examples are over-sampled, whereas the majority class examples are under-sampled [CBHK02, BPM04].

In our framework, we applied SMOTETomek using *imbalanced-learn*, which is a package that provides with a bunch of sampling approaches used in datasets showing high class imbalance [LNA17].

Baselines. In our study, we compare the performance of Gradient Boosting and Random Forest classifiers, taking as a baseline a Decision Tree classifier.

Evaluation strategy. To validate our proposal, we perform two sets of experiments:

1. **Performance evaluation.** We evaluate the performance of our approach and compare it to the baseline to investigate the behavior of our classifiers against a less complex model by the progress of weeks.

2. **Hyper-parameter Tuning.** During the hyper-parameter tuning phase, we used the grid search algorithm to find the best setting for the best performing classifier. Our choice to do this tuning for the most performing algorithm was made because the search of the best hyper-parameters is very computationally consuming.

5.4.2 Deep Learning Prediction Strategies

We trained each neural network for 50 epochs with the mini-batch size of 16 and an early stopping callback with patience set to 10. The learning rate was set to 10^{-3}. To deal with class-imbalance we opted for an over-sampling technique exploiting tf.data API [MSKI21].

Baselines. In our study, we compare our Tri-branch CNN (described in 5.3.3) to three neural network architectures:

- Vanilla CNN: a Convolutional Neural Network with a convolution of size 3×3 and composed of one 2D convolutional layer and a dense output layer with a sigmoid activation function;
- Bidirectional LSTM: a neural network composed of a TimeDistributed input layer, a Bidirectional LSTM layer and a dense output layer with a sigmoid activation function;
- Feed-Forward: a feed-forward Neural Network composed of a dense input layer, a dense layer with 32 units and a dense output layer with a sigmoid activation function.

Evaluation strategy. To validate our proposal, we perform three sets of experiments:

1. **Performance evaluation.** We evaluate the performance of our approach and compare it to the two baselines to investigate the behavior of our Tri-branch CNN against less complex networks by the progress of weeks.
2. **Ablation study.** We performed an ablation study to our Tri-branch CNN to analyse the impact of network branches by using a single branch or the combinations of any two of them, for the first four weeks of each user.
3. **Tri-Branch CNN vs the best Tree-Based classifier** We compare the performance of Tri-Branch CNN approach with the best performing three based classifier.

5.5 Experimental Results

Here, we present in detail the results obtained from each set of experiments. Section 5.5.1 described the results obtained from each set of experiments performed using classic machine learning based approaches, and Section 5.5.2 describes the results obtained from each set of experiments performed using deep learning based approaches.

Table 5.1: Performance evaluation of different methods over weeks (The values that are in bold represent the best results for each week)

Week	RF	GB	DT
1	**0.96**	**0.96**	0.88
2	**0.64**	0.55	0.50
3	0.75	**0.77**	0.61
4	**0.88**	**0.88**	0.56

Table 5.2: Best setting and performance of RF after Grid Search

Week	max_depth	n_estimators	AUC
1	8	32	0.96
2	14	200	0.68
3	4	16	0.73
4	5	100	0.88

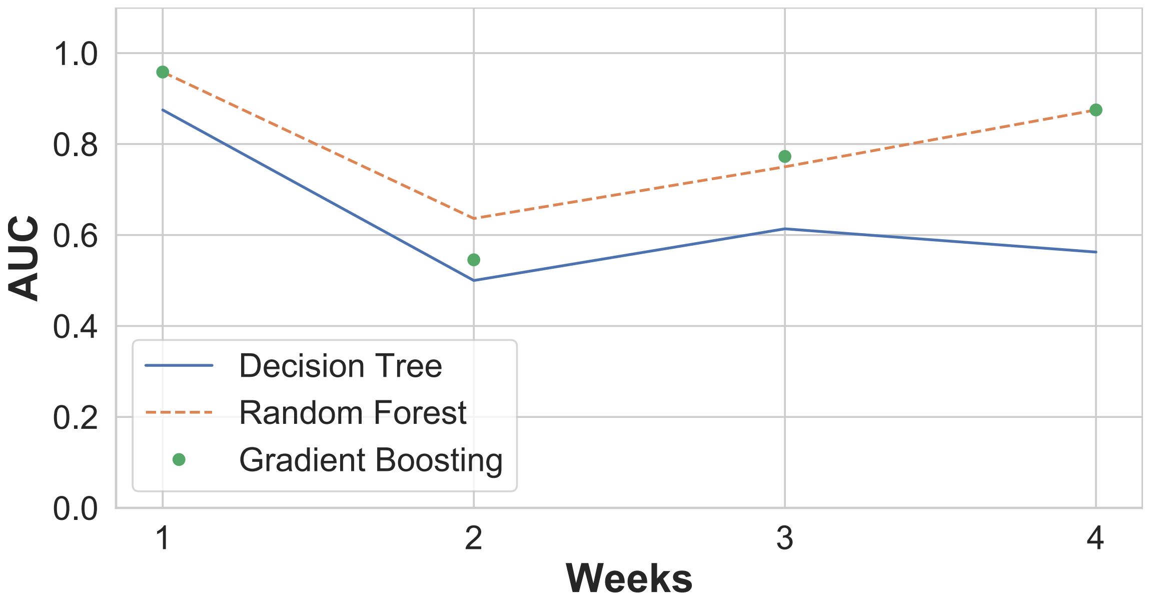


Figure 5.3: Performance evaluation of different tree-based classifiers over weeks

Table 5.3: Performance evaluation of different methods over weeks (The values that are in bold represent the best results for each week)

Week	Tri-branch CNN	Vanilla CNN	LSTM	Feed-Forward
1	**0.78**	0.69	0.68	0.61
2	**0.82**	0.75	0.65	0.73
3	**0.78**	0.59	0.60	0.70
4	**0.88**	0.77	0.57	0.77

5.5.1 Classic Machine Learning Prediction Strategies

Performance Evaluation.

Table 5.1 shows the performance, in terms of AUC of the classifiers we considered for this study. All the classifiers achieved a great performance in the first and the last week, meanwhile the performance tends to lower for the second week and starts to raise in the third week (this may be caused by the high class imbalance).

Hyper-parameter Tuning.

We estimated the best parameters for Random Forest using Grid Search. The classifier was run with the default parameters, except for the number of trees in the forest (*n_estimators* parameter) and the max number of levels in each decision tree (*max_depth* parameter). This is because a larger number of trees in the forest (*n_estimators*) could improve the performance of Random Forest and *max_depth* limits the number of nodes in each decision tree. The best parameters and performance of RF after applying a Grid Search are presented in Table 5.2. After performing a Grid Search, we managed to ameliorate the performance of RF for almost all the weeks. For the third week training the classifier on the default setting had a slightly better performance in terms of AUC, but after applying Grid Search it still achieved a good performance using less trees and a lower number of levels in each decision tree.

5.5.2 Deep Learning Prediction Strategies

Performance Evaluation

Table 5.3 shows the accuracy, in terms of AUC of our approach, Tri-branch CNN, and of the three baselines on the data. As in our observational analysis, we consider as time span first four weeks of training of each user. These results are also graphically illustrated in Figure 5.4.

From Table 5.3 and Figure 5.4 we notice that all the deep learning based approaches achieve a good performance. We can see clearly that Tri-branch CNN is the one that had the best performance compared to the deep learning baselines,

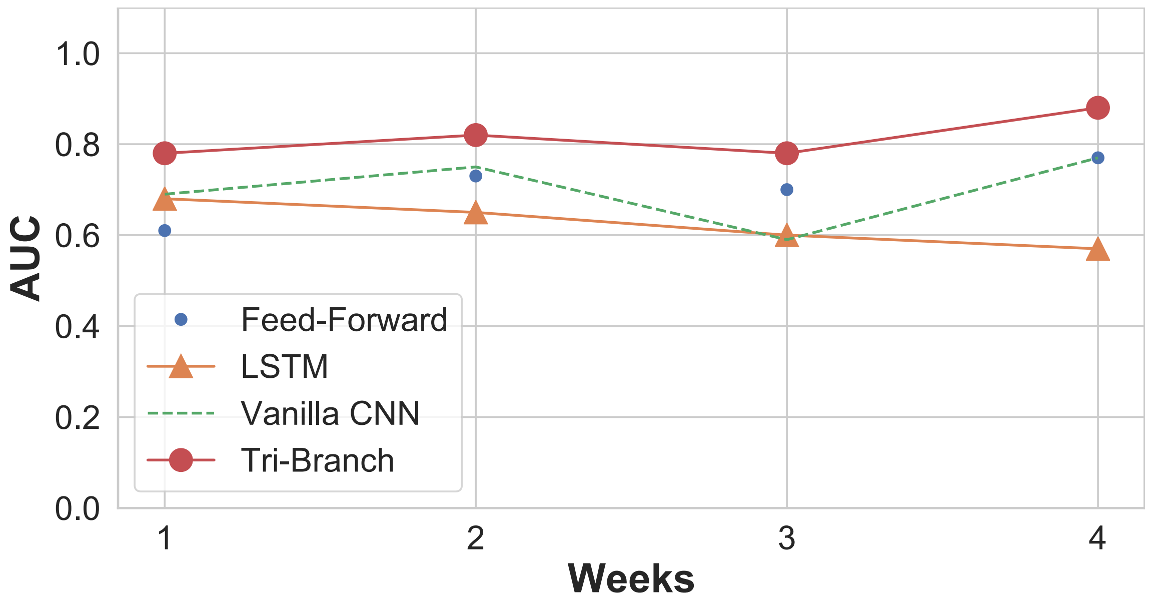


Figure 5.4: Performance evaluation of different neural architectures over weeks

thus showing that modeling users also considering (i) the temporal aspects of their workout behavior (Week branch) and (ii) how the features evolve (Feature branch), can provide important benefits in terms of accuracy. The Feed-Forward and Vanilla CNN have almost the same behavior along the weeks, where the LSTM model is very influenced by the volume of data, as the number of users shrinks from a week to another the performance of LSTM tends to lower.

Observation 2. *To predict possible dropouts of the users, it is important to model their behavior beyond workout results. The temporal evolution of the performance and the focus on the specific features that compose a workout, increase prediction accuracy.*

Ablation Study (Impact of network branches)

To investigate the impact of the network branches in providing accurate predictions, we train networks using a single branch or the combinations of any two of them.

In Table 5.4, we resume the performance of all the combinations of network branches in terms of AUC for the first four weeks and the overall performance of each network. Figure 5.5 illustrates the performance of our Tri-branch CNN compared to networks composed of a single branch and combination of any two of them.

From Table 5.4 and Figure 5.5, we observe that the best performing network is not the same for all the weeks. For the first and the fourth week, the best performing networks are those composed of the combination of all the workout branches. Hence, the branches that had a significant contribution were the one composed by the

Table 5.4: Performance evaluation of different settings over weeks (The values that are in bold represent the best results for each week. In the setting column, w: represents a CNN composed only of the workout branch, f: represents a CNN composed only of the feature branch, g: represents a CNN composed only of the week branch, and all stands for the Tre-branch CNN)

setting	Week 1	Week 2	Week 3	Week 4	Mean
w	**0.79**	0.77	0.65	0.77	0.75
f	0.67	0.79	0.78	0.78	0.76
g	0.76	0.83	0.69	**0.88**	0.79
w+f	0.63	0.83	**0.80**	0.87	0.78
w+g	0.69	0.77	0.75	0.82	0.76
f+g	0.70	**0.87**	0.72	0.87	0.79
all	**0.79**	0.82	0.78	**0.88**	**0.83**

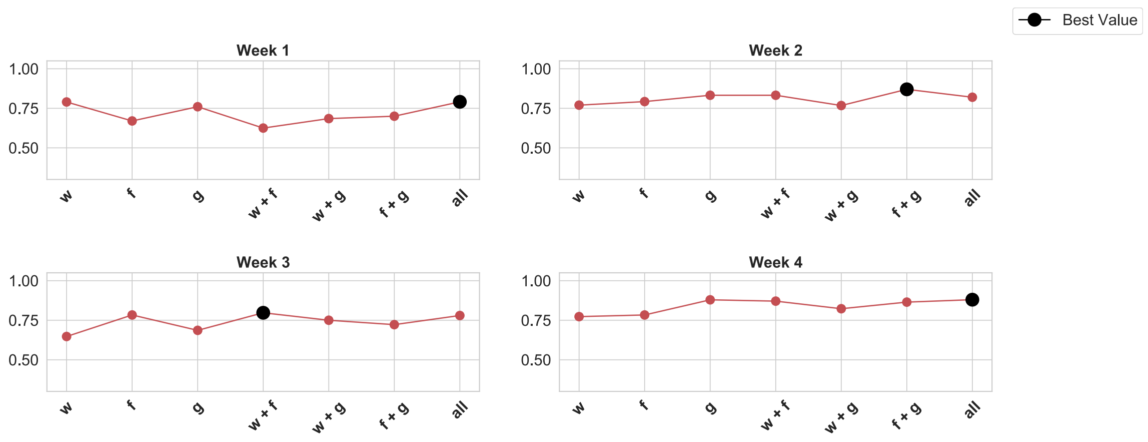


Figure 5.5: Performance evaluation of different settings over weeks (w: represents a CNN composed only of the workout branch, f: represents a CNN composed only of the feature branch, g: represents a CNN composed only of the week branch, and all stands for the Tre-branch CNN)

feature branch and week branch (f+g) for the second week, and the one composed of workout branch and the feature branch (w+f) for the third week.

From Table 5.4, we can also see that regardless of the week for which we are predicting, the Tri-branch CNN achieves the best average performance.

Observation 3. *When the amount of information about each user is scarce (first few weeks working out), the temporal evolution of their performance and the workout results are the drivers towards an accurate classification, confirming the need for a user-centered approach. As the weeks progress, we know more about the users and their behavior, but these users become less, the feature branch modeling how the values of each feature evolve, plays a significant role in the classification process. Each perspective (technically, each branch of our CNN) plays a different role over time, and acts in synergy with the others.*

Deep Learning vs the Classic Machine Learning Prediction Strategies

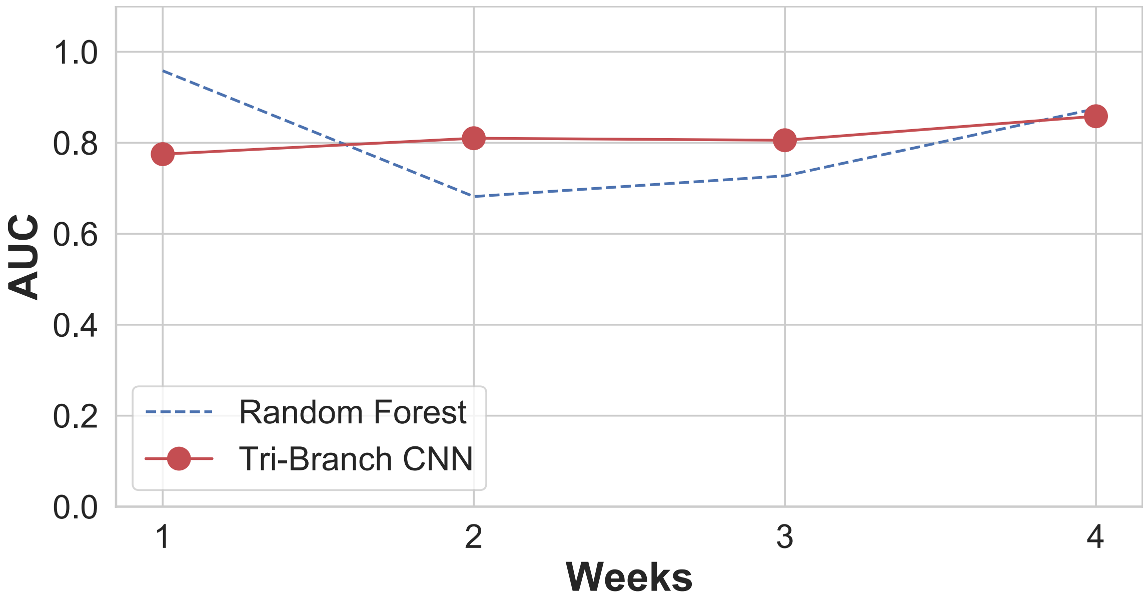


Figure 5.6: Performance evaluation of different approaches over weeks

Figure 5.6, illustrates the performance over weeks of Random Forest (the shallow learning model that achieved the best performance) and Tri-Branch CNN (the deep learning model that achieved the best performance). When predicting for the first week, Random Forest achieves a better performance than Tri-Branch CNN. This behavior is in line with what has been highlighted earlier in the literature by Rathore et al. [RK17]. When predicting in the fourth week, Random Forest and Tri-branch achieved almost the same performance, meanwhile Random Forest performed slightly better. Despite Random Forest outperforms Tri-Branch CNN in the first and fourth week, Tri-Branch CNN maintains a good and stable performance allover the weeks. Thus, we consider that Tri-branch CNN is more reliable w.r.t. Random Forest. Indeed, when deployed into a real-world platform, our approach, Tri-branch CNN, would provide constantly effective predictions over time, while Random Forest has a more unpredictable behavior according to the weeks in which it is trained.

In parallel with this work, Gattermann et al. [GIT21] showed the effect of training on multiple time slices on the classification performance in churn prediction. The results of this work emphasized that training on samples from multiple time slices improves prediction performance, and that multi-slicing makes models more generalizable [GIT21]. The observations made by Gattermann et al. are also confirmed by the results of our experiments, since we see the benefits of training on multiple time slices on quality of predictions and on the generalizability of the models.

5.6 Conclusions and Future Work

In this chapter, we proposed and validated an approach to identify sportspeople that need timely support due to losses in their motivation to work out. To tackle this problem, we proposed an end-to-end model, named Tri-branch CNN, that applies convolutions along different dimensions (namely, the workouts in each week, the features, and the weeks dimensions) to identify the users that are not likely to work out in the next week. Results show the effectiveness of our approach and the role of each dimension at different prediction times.

We have recently completed the integration of our approach in the eCoaching platform to let coaches interact with the users they follow in an effective way. Hence, this integration will allow us to perform an A/B testing and ameliorate the effectiveness of our system based on coaches' feedback.

As future work, we look forward to introducing explainability and fairness insights, to make sure that different groups of users are getting the same level of attention. Furthermore, we will also study the feasibility of recommending tailored workout plans that will be suggested to coaches and sent to users after being approved or modified (if necessary) by the coach.

Part II

Applications

Chapter 6

Tools in support of the coaches

6.1 Introduction

An essential aspect of coaching is to monitor the progress of coachees over time. To support sportspeople effectively, coaches need to have an idea of the progress in the performance of each sportsperson. Athletes could be exposed to many risk factors such as injuries and over-exertion syndromes. The role of the coaches is to early prevent these factors by monitoring the training load and progress. As we mentioned in Section 1.2, one of the biggest challenges an industrial Ph.D. student could face is to build products that create value for themselves, for the business, and also for the scientific community. To this end, we developed a bunch of dashboards that monitor the sportspeople's progress w.r.t different aspects. These dashboards are also crucial to collect meaningful data that we could use for further research works and create value to the business by optimizing the work of coaches, so they can follow their sportspeople effectively and keep them motivated to train regularly.

As we mentioned in Chapter 5, a significant phenomenon that eCoaching platforms suffer from is the problem of sportspeople dropping out of training. Different factors could cause this. Injuries, loss of motivation, going on holidays, or switching to another platform may lead people to drop out exercising regularly. Nevertheless, these factors have different impacts on people's health. For this reason, we decided to build and deploy a machine learning algorithm able to predict sportspeople drop out before it occurs and notify their coaches in order to get in touch with them and take action at an early stage. By helping coaches spot, at an early stage, the people that will give up training regularly, we spare them the time of going through the entire training history of their clients and analyzing the trends of their training behavior and performance by directly providing them with a list of users who are very likely not to train during the following week.

Thus, the developed products directly impact sportspeoples' lifestyles and coaches' ability to follow more users effectively and in less time. Moreover, by optimizing the workload of coaches, U4FIT could generate more sales since a coach could follow more people than they usually follow without any loss in the quality of their service. The value for the Ph.D. researcher is to learn cutting-edge technologies that are significant for their future career and the data gathered using these systems that could be useful for further research.

6.2 Sportspeople requests and Statistics Dashboard

The first dashboard (illustrated in Figure 6.1) provides the trainers with two tables. The first table shows the new user requests, and the second shows the statistics about the number of workouts performed by each sportsperson during the last three weeks and the dates on which the workout plans and user subscriptions end. From this dashboard, the trainers could take different actions such as:

- Contacting a certain sportsperson;
- Creating a new workout-plan for a certain sportsperson;
- Consulting the profile of a certain sportsperson;
- Analyzing the results of the last workout session of certain sportsperson;
- Consulting other dashboards that monitor the sportspeople's behavior.

6.3 Sportspeople Analytics Dashboard

To provide the coaches with a global vision about their sportspeople's behavior, we developed a dashboard composed of multiple charts showing the evolution in time of the performance of each sportsperson they follow w.r.t. different metrics (as illustrated in Figure 6.2).

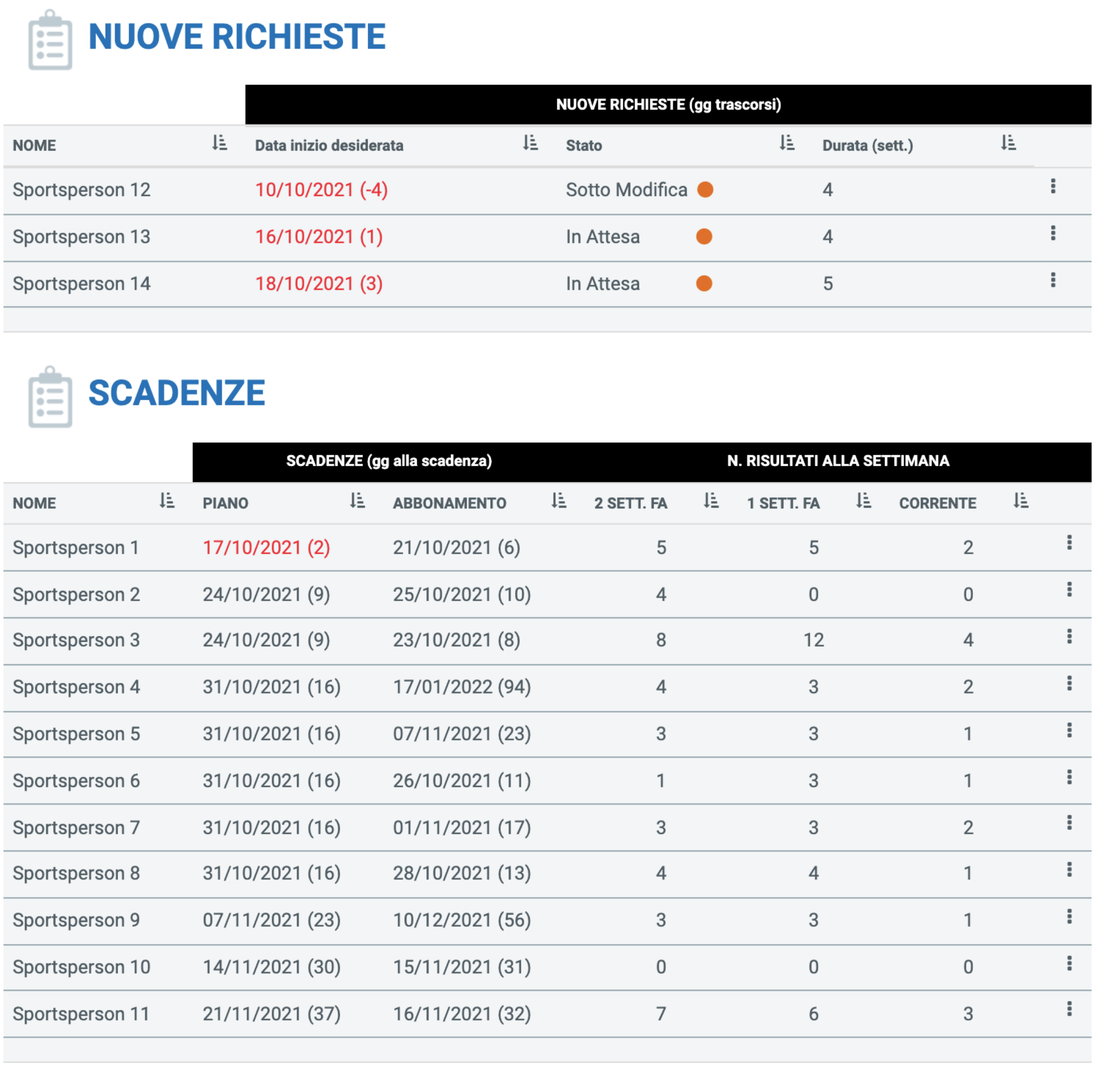
In Figure 6.2, the first chart (top left) monitors the session-RPE against the intensity levels of exercise; The second one (top right) monitors the distribution of the distance covered by the sportsperson by the variation of time; The third chart (bottom left) monitors the evolution of the Training Impulse (aka Trimp) over time; The last graph shows the development of the average pace in each intensity zone over time.

The click on each of the charts leads the trainer to another page showing a large version of the chart provided with a menu the permits them to aggregate the data in the chart by months, by weeks, or by days, as illustrated in Figure 6.3.

The metrics monitored by these charts are:

- **Intensity Zones**: Exercise is categorized into three different intensity zones. These zones include low, moderate, and vigorous and could be based on heart

Figure 6.1: Sportspeople requests and Statistics



NUOVE RICHIESTE

	NUOVE RICHIESTE (gg trascorsi)		
NOME	Data inizio desiderata	Stato	Durata (sett.)
Sportsperson 12	10/10/2021 (-4)	Sotto Modifica ●	4
Sportsperson 13	16/10/2021 (1)	In Attesa ●	4
Sportsperson 14	18/10/2021 (3)	In Attesa ●	5

SCADENZE

	SCADENZE (gg alla scadenza)		N. RISULTATI ALLA SETTIMANA		
NOME	PIANO	ABBONAMENTO	2 SETT. FA	1 SETT. FA	CORRENTE
Sportsperson 1	17/10/2021 (2)	21/10/2021 (6)	5	5	2
Sportsperson 2	24/10/2021 (9)	25/10/2021 (10)	4	0	0
Sportsperson 3	24/10/2021 (9)	23/10/2021 (8)	8	12	4
Sportsperson 4	31/10/2021 (16)	17/01/2022 (94)	4	3	2
Sportsperson 5	31/10/2021 (16)	07/11/2021 (23)	3	3	1
Sportsperson 6	31/10/2021 (16)	26/10/2021 (11)	1	3	1
Sportsperson 7	31/10/2021 (16)	01/11/2021 (17)	3	3	2
Sportsperson 8	31/10/2021 (16)	28/10/2021 (13)	4	4	1
Sportsperson 9	07/11/2021 (23)	10/12/2021 (56)	3	3	1
Sportsperson 10	14/11/2021 (30)	15/11/2021 (31)	0	0	0
Sportsperson 11	21/11/2021 (37)	16/11/2021 (32)	7	6	3

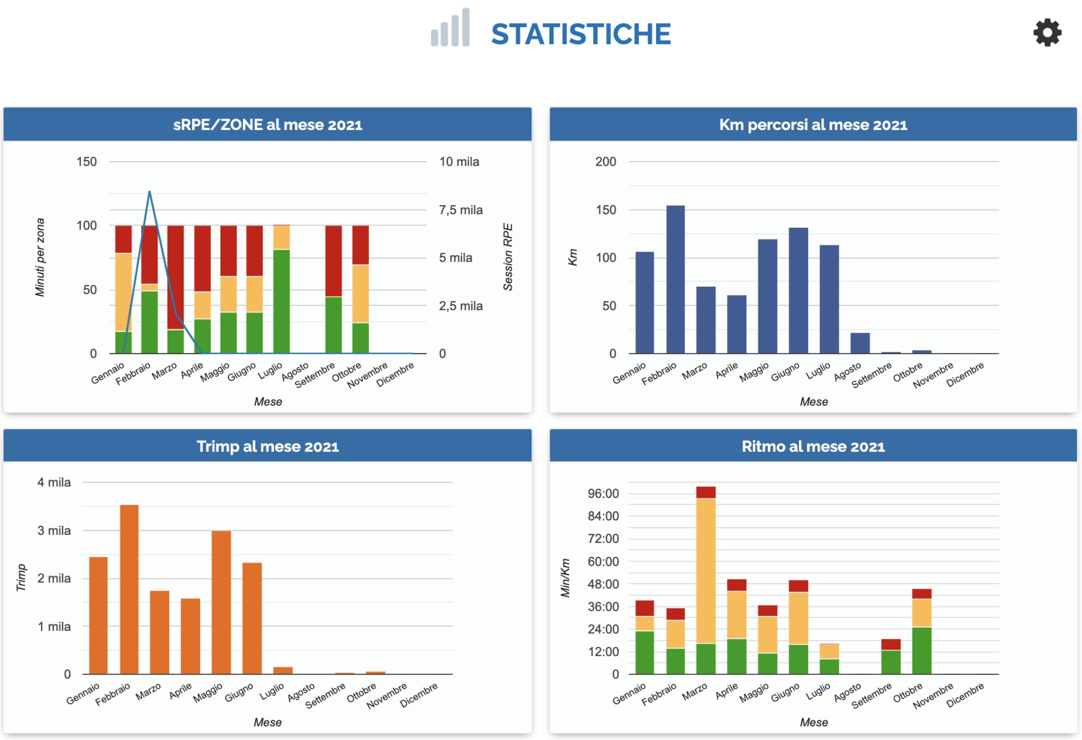


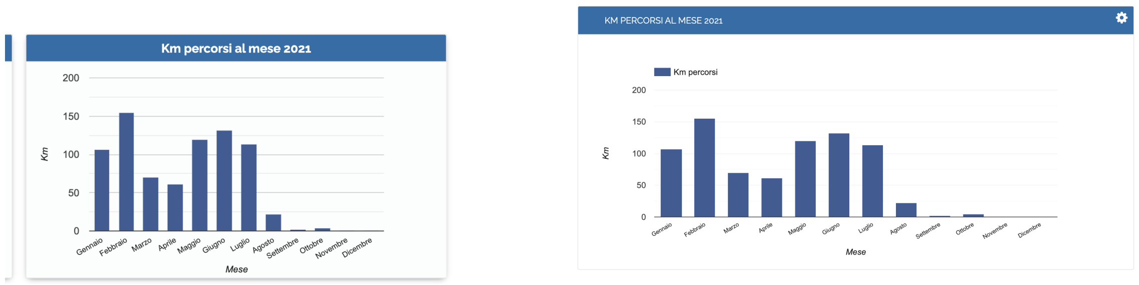
Figure 6.2: Charts monitoring the workout analytics of a sportsperson.

rates associated with 2 and 4 mmol/L of blood lactate as determined during an incremental exercise test. The effects of exercise are different at each intensity zone [Gil96].

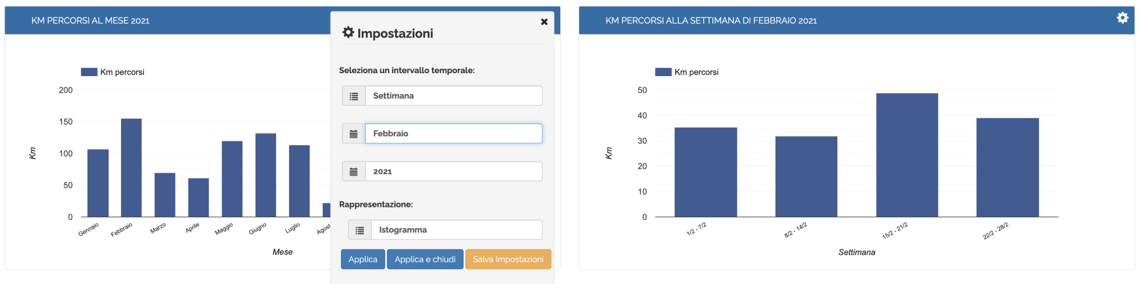
- **Session RPE**: Session-rating of perceived exertion (RPE) is a simple modification of the perceived exertion rating scale (RPE) initially developed by Borg [BHL87], in which the participant is asked to rate the overall intensity of the entire training session [FHW$^+$95]. When this intensity score is multiplied by the duration of the training session, a single number is obtained that represents the magnitude of that training session [Fos98]. The session RPE is a very important measure because it allows us to estimate the effort perceived by the athlete during the training session. Through it, we can therefore monitor their level of fatigue to avoid the risk of overexertion syndromes.

- **TRIMP**: TRIMP is an abbreviation of TRaining IMPulse. It is computed from training intensity, measured as the mean exercise heart rate, and training duration, measured in minutes. It is considered an integrative marker of exercise load during training and competition [STVS07].

- **Average Pace**: In running, the pace is usually defined as the amount of time required to run a fixed distance. It is generally calculated as the number of



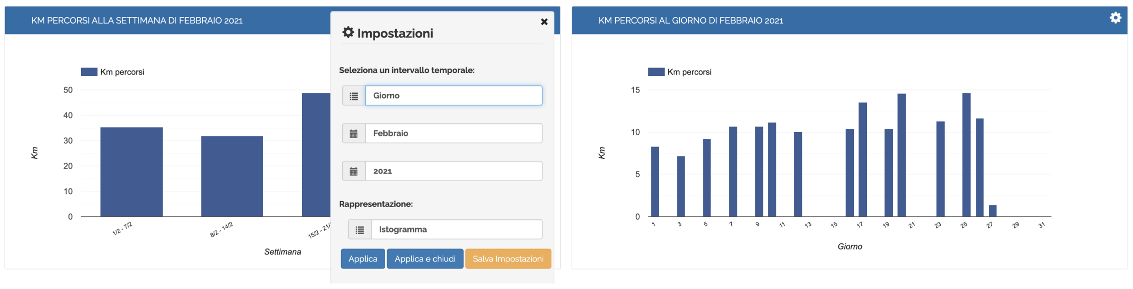
(a) Left: Covered distance chart in the analytics dashboard. Right: Large view of the covered distance chart



Seleziona un intervallo temporale: Settimana, Febbraio, 2021

Rappresentazione: Istogramma

(b) Left: Menu of the covered distance per month chart. Right: Large view of covered distance per week chart

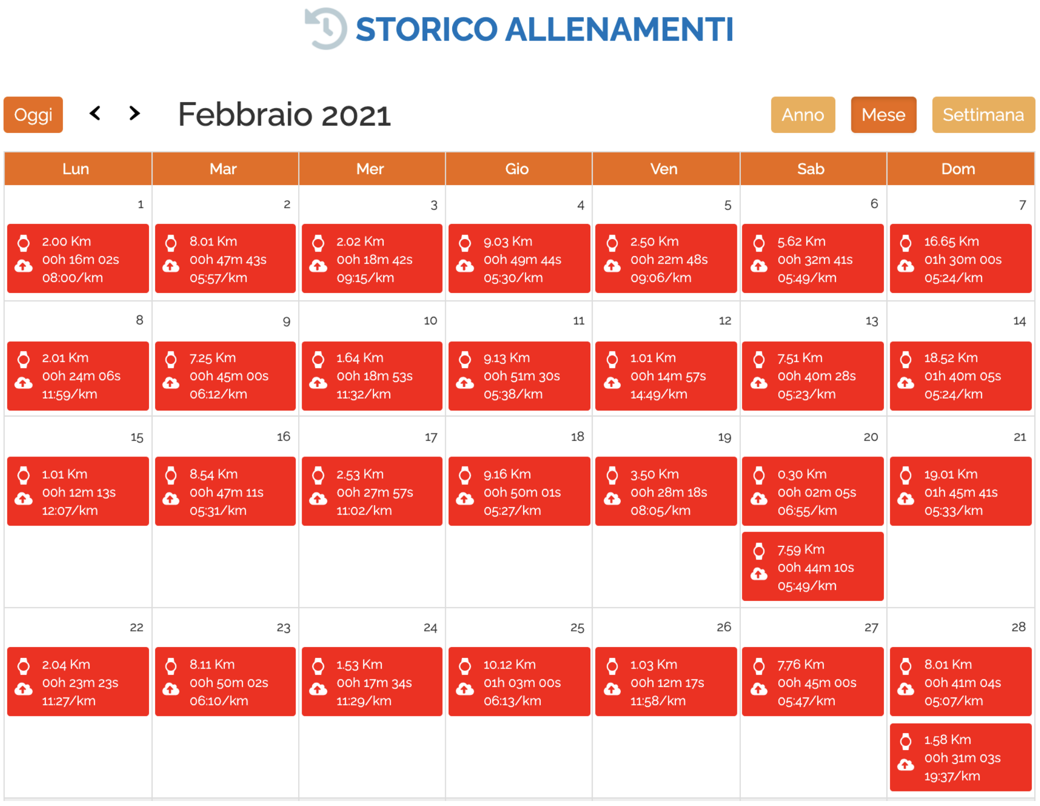

KM PERCORSI ALLA SETTIMANA DI FEBBRAIO 2021
Km percorsi
Km
Settimana
Impostazioni
Applica
Applica e chiudi
Salva Impostazioni
KM PERCORSI AL GIORNO DI FEBBRAIO 2021
Giorno

Seleziona un intervallo temporale: Giorno, Febbraio, 2021

Rappresentazione: Istogramma

(c) Left: Menu of the covered distance per week chart. Right: Large view of covered distance per day chart

Figure 6.3: Aggregations of the covered distance chart.


STORICO ALLENAMENTI

Oggi < > Febbraio 2021 Anno Mese Settimana

Lun	Mar	Mer	Gio	Ven	Sab	Dom
1 2.00 Km 00h 16m 02s 08:00/km	2 8.01 Km 00h 47m 43s 05:57/km	3 2.02 Km 00h 18m 42s 09:15/km	4 9.03 Km 00h 49m 44s 05:30/km	5 2.50 Km 00h 22m 48s 09:06/km	6 5.62 Km 00h 32m 41s 05:49/km	7 16.65 Km 01h 30m 00s 05:24/km
8 2.01 Km 00h 24m 06s 11:59/km	9 7.25 Km 00h 45m 00s 06:12/km	10 1.64 Km 00h 18m 53s 11:32/km	11 9.13 Km 00h 51m 30s 05:38/km	12 1.01 Km 00h 14m 57s 14:49/km	13 7.51 Km 00h 40m 28s 05:23/km	14 18.52 Km 01h 40m 05s 05:24/km
15 1.01 Km 00h 12m 13s 12:07/km	16 8.54 Km 00h 47m 11s 05:31/km	17 2.53 Km 00h 27m 57s 11:02/km	18 9.16 Km 00h 50m 01s 05:27/km	19 3.50 Km 00h 28m 18s 08:05/km	20 0.30 Km 00h 02m 05s 06:55/km 7.59 Km 00h 44m 10s 05:49/km	21 19.01 Km 01h 45m 41s 05:33/km
22 2.04 Km 00h 23m 23s 11:27/km	23 8.11 Km 00h 50m 02s 06:10/km	24 1.53 Km 00h 17m 34s 11:29/km	25 10.12 Km 01h 03m 00s 06:13/km	26 1.03 Km 00h 12m 17s 11:58/km	27 7.76 Km 00h 45m 00s 05:47/km	28 8.01 Km 00h 41m 04s 05:07/km 1.58 Km 00h 31m 03s 19:37/km

Figure 6.4: Calendar showing the workout history of a sportsperson.

minutes it takes to cover a kilometer or a mile. Researchers retain that the pace is a much more significant quantity to a runner [Sas99]. Pacing is often a crucial aspect of endurance events. Some coaches recommend training with a combination of specific paces related to one's fitness to stimulate various physiological improvements.

6.4 Sportspeople Training History

To give the trainers an overview about the training load of the sportspeople thy follow, we also provide them with a calendar that shows the workouts they performed each day, and a summary of the results of each workout, as illustrated in Figure 6.4. The calendar in Figure 6.4 can show the coaches the workout performed by day, by month, or by year. The click on each workout in the calendar redirects the coach to the workout plan where they can compare the training objectives to the results achieved by the sportspeople to accurately evaluate their performance.

6.5 Online dropout prediction

To help coaches spot the sportspeople willing to drop out, we developed a framework composed of a front-end and a back-end. The front-end consists of a web interface that provides coaches with the list of sportspeople predicted to drop out of exercise next week. Instead, the back-end consists of an API that models and analyses the sportspeople's training history to predict through a machine learning algorithm if they will perform at least a workout during the following week. In Section 6.5, we present the technologies we exploited to build the back-end API. In Section 6.5.1, we describe the technologies we used to build the front-end client, and in Section 6.5.2 we present the technical architecture of our framework.

Back-end (API)

The predictions API was developed using Python programming language, the scikit-learn library for machine learning. We used Flask, Celery to handle multitasking, to deliver our software as a container to build the API, and finally Docker to deploy our software as a container.

This technologies are defined as follows:

- **Python** Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create various programs and isn't specialized for any specific problems.

- **Scikit-learn** Scikit-learn is a key library for the Python programming language that is typically used in machine learning projects. Scikit-learn focuses on machine learning tools, including mathematical, statistical, and general-purpose algorithms that form the basis for many machine learning technologies.

- **Flask** Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies, and several standard framework-related tools.

- **Celery** As mentioned on the official website, Celery is a distributed task queue with which you can handle millions or even billions of tasks in a short time. Celery requires a messaging agent to handle requests from an external source; usually, this comes in the form of a separate service called a message broker. There are many options for brokers available to choose from, including relational databases, NoSQL databases, key-value stores, and messaging systems.

Here, we choose RabbitMQ as a messaging system. RabbitMQ is feature-complete, stable, durable, and easy to install. It's an excellent choice for a production environment.

- **Docker** Docker is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries, and configuration files; they can communicate with each other through well-defined channels.

6.5.1 Front-end (Client)

The web client was built using PHP programming language exploiting CodeIgniter framework and JavaScript.

Bellow there are the definitions of each of these technologies:

- **PHP** PHP (recursive acronym for PHP: Hypertext Preprocessor) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.

- **CodeIgniter** CodeIgniter is a PHP MVC framework used for developing web applications rapidly. CodeIgniter provides out-of-the-box libraries to connect to the database and perform various operations like sending emails, uploading files, managing sessions, etc.

- **JavaScript** JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side scripts to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

6.5.2 Dropout Prediction Framework Architecture

This section presents the technical architecture of our framework.

Figure 6.5 illustrates the architecture of our framework. The workflow of our framework is described as follows:

- The client sends a request to the back-end by means of a REST API;

- The back-end receives the request and send it to the celery producer;

- The producer adds a job to the broker's queue;

- Celery worker takes jobs from the broker and executes them;

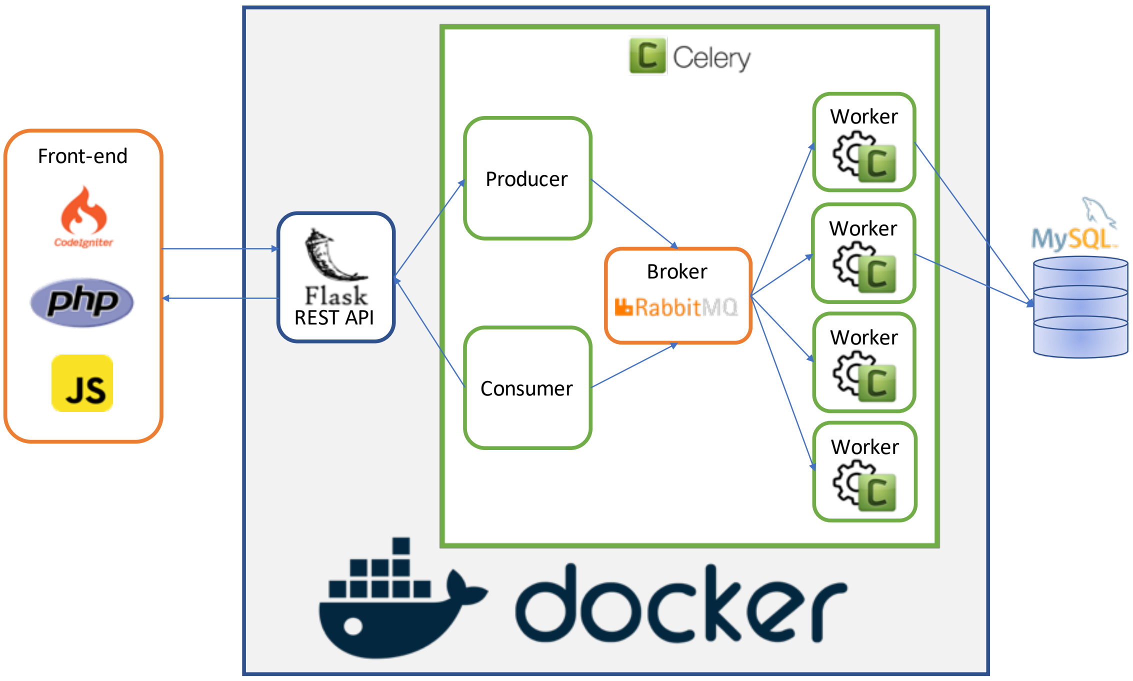
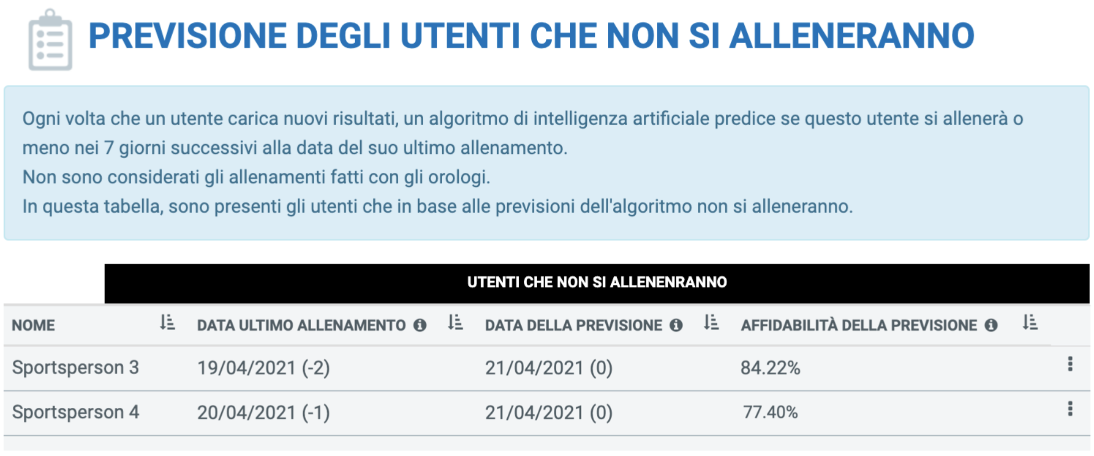


Figure 6.5: Dropout Prediction Framework Architecture.



PREVISIONE DEGLI UTENTI CHE NON SI ALLENERANNO

Ogni volta che un utente carica nuovi risultati, un algoritmo di intelligenza artificiale predice se questo utente si allenerà o meno nei 7 giorni successivi alla data del suo ultimo allenamento.
Non sono considerati gli allenamenti fatti con gli orologi.
In questa tabella, sono presenti gli utenti che in base alle previsioni dell'algoritmo non si alleneranno.

UTENTI CHE NON SI ALLENENRANNO

NOME	DATA ULTIMO ALLENAMENTO	DATA DELLA PREVISIONE	AFFIDABILITÀ DELLA PREVISIONE
Sportsperson 3	19/04/2021 (-2)	21/04/2021 (0)	84.22%
Sportsperson 4	20/04/2021 (-1)	21/04/2021 (0)	77.40%

Figure 6.6: Users predicted not to workout next week

- the consumer receives the results of the executed jobs and returns them to the Flask controller;
- Then, the modeling of users and prediction results are saved to a MySql database;
- The API returns the response to the front-end;
- Finally, the front-end processes the response and renders the final result graphically, namely a list of users that are not likely to train during the following week, the date of their last workout, the date when the prediction was made, and the predictions' accuracy (Figure 6.6).

The back-end system is running on an Amazon EC2 t2.micro machine. The main challenge here was to optimize the data pipeline such a way that it dose not create memory leak problems since our system models the full users' workout history since they began training using the U4FIT platform. A simple solution could be to increase the server's memory, but this could be very costly w.r.t. the budget of SMEs. For this reason, the proposed solution consisted of modifying the data processing pipeline, without causing major changes to the original modeling, by saving the last weeks modelings in the DB, and contextualizing the new performed workouts with the last modeling in the DB.

6.6 Conclusions and Future Work

In this chapter, we presented the tools we developed to support coaches to reduce their workload, giving them the possibility to monitor different aspects and metrics that measure sportspeople's performance and workout frequency. In addition to that, we provided coaches with an online sportspeople dropout prediction system. The system was enabled for a bunch of trainers in order to perform A/B testing. On the other hand, the modeling of users' data and prediction results were saved to the DB for our system's online validation and future research work.

Part III

Concluding Remarks

Chapter 7

Conclusions and future work

This industrial Ph.D. thesis focuses on exploiting machine learning techniques to help personal trainers in Online Coaching Platforms to be efficient yet effective in supporting sportspeople.

7.1 Contributions

Our research on machine learning in eCoaching provides the following contributions:

Offline contributions:

- **User/Workout Modeling.** We presented different approaches to model users' workout behavior in the running, considering the temporal perspective of the workouts;
- **Machine Learning.** We presented different shallow and deep learning-based architectures to predict sportspeople dropout and workout quality;
- **Learning To Rank.** We introduced an algorithm to rank the users according to the support they need and recommend them to the coach;
- **Algorithmic Fairness.** We provided, for the first time in the literature of athletic-related user recommendation, algorithms to provide fairness of exposure in the results;
- **Dataset.** We make the datasets used in this thesis available in order to allow the community to advance the research on this topic.

Online contributions:

- **Tools in support of the coaches.** We provided trainers in the eCoaching platform with dashboards that optimize their workload to guarantee high-quality support for the sportspeople;

- **Online Dropout Prediction.** We designed, developed, and deployed a Machine Learning algorithm to help coaches spot sportspeople likely to give up training.

Publications:

- Chapter 4 is based on the results of two papers: "Predicting Workout Quality to Help Coaches Support Sportspeople" published in the Proceedings of the Third International Workshop on Health Recommender Systems co-located with Twelfth ACM Conference on Recommender Systems (HealthRecSys'18), and "Fair Performance-based User Recommendation in eCoaching Systems" under review to be published in the Special Issue on Recommender Systems for Health and Wellbeing of User Modeling and User-Adapted Interaction (UMUAI) journal;

- Chapter 5 is based on the paper "Sportspeople Dropout Prediction in eCoaching Platforms" under review to be published in the proceedings of Special Track on Health Informatics and Bioinformatics of the SIGAPP SAC 2022 conference.

7.2 Future Research Directions

Our research on machine learning for sports remote coaching platforms has produced a variety of methods but still poses some interesting challenges that require further investigation:

- **Explainability and Interpretability.** ML models embedded in eCoaching artificial intelligence systems might suffer from low explainability (e.g., on the reason those particular recommendations are provided to the user). Hence, it becomes crucial to understand how we can explain the output of a model and how it varies based on changes in input or algorithmic parameters. Moreover, it requires attention to how internal mechanics can be explained in human terms.

- **Fairness, Transparency, and Accountability.** With the advent of AI-based coaching, addressing bias within ML models will be a core priority due to several reasons. Some biases can be introduced by using training data which is not an accurate sample of the population (e.g., more men than women) or is influenced by socio-cultural stereotypes (e.g., popularity). Moreover, advanced properties built on top of biases, e.g., fairness, transparency, and accountability, require attention. Future research should control these properties in the developed models.

- **Online evaluation.** ML models embedded in eCoaching artificial intelligence systems might have a different behavior when deployed in production. A model that has a high accuracy on a test dataset might not necessary do well in production. That can be due mainly to differences between data used in training and testing the model and the data in the phase of production. For example, collecting the data used for the training in a certain season of the year, and the deploying of the model was done in a different season could influence the accuracy of the model, another example could be the pandemic or other external factors that could influence on the performance of our model. For this reason it is crucial to track the models' online accuracy and to perform A/B testing in order to ameliorate the effectiveness of our system based on coaches' feedback.

Bibliography

[AB15] Xavier Amatriain and Justin Basilico. Recommender systems in industry: A netflix case study. In Francesco Ricci, Lior Rokach, and Bracha Shapira, editors, *Recommender Systems Handbook*, pages 385–419. Springer, 2015.

[AK15] Shreya B Ahire and Harmeet Kaur Khanuja. A personalized framework for health care recommendation. In *2015 International Conference on Computing Communication Control and Automation*, pages 442–445. IEEE, 2015.

[BCD+19] Alex Beutel, Jilin Chen, Tulsee Doshi, Hai Qian, Li Wei, Yi Wu, Lukasz Heldt, Zhe Zhao, Lichan Hong, Ed H. Chi, and Cristos Goodrow. Fairness in recommendation ranking through pairwise comparisons. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019.*, pages 2212–2220. ACM, 2019.

[BCI+18] Ludovico Boratto, Salvatore Carta, Walid Iguider, Fabrizio Mulas, and Paolo Pilloni. Predicting workout quality to help coaches support sportspeople. In David Elsweiler, Bernd Ludwig, Alan Said, Hanna Schäfer, Helma Torkamaan, and Christoph Trattner, editors, *Proceedings of the 3rd International Workshop on Health Recommender Systems, HealthRecSys 2018, co-located with the 12th ACM Conference on Recommender Systems (ACM RecSys 2018), Vancouver, BC, Canada, October 6, 2018*, volume 2216 of *CEUR Workshop Proceedings*, pages 8–12. CEUR-WS.org, 2018.

[BCMP17] Ludovico Boratto, Salvatore Carta, Fabrizio Mulas, and Paolo Pilloni. An e-coaching ecosystem: design and effectiveness analysis of the engagement of remote coaching on athletes. *Personal and Ubiquitous Computing*, 21(4):689–704, 2017.

[BFM20] Ludovico Boratto, Gianni Fenu, and Mirko Marras. Interplay between upsampling and regularization for provider fairness in recommender systems. *CoRR*, abs/2006.04279, 2020.

[BFOS17] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees*. Routledge, 2017.

[BGW18] Asia J. Biega, Krishna P. Gummadi, and Gerhard Weikum. Equity of attention: Amortizing individual fairness in rankings. In Kevyn Collins-Thompson, Qiaozhu Mei, Brian D. Davison, Yiqun Liu, and Emine Yilmaz, editors, *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 405–414. ACM, 2018.

[BHL87] Gunnar Borg, Peter Hassmén, and Monica Lagerström. Perceived exertion related to heart rate and blood lactate during arm and leg exercise. *European journal of applied physiology and occupational physiology*, 56(6):679–685, 1987.

[Bin18] Reuben Binns. Fairness in machine learning: Lessons from political philosophy. In Sorelle A. Friedler and Christo Wilson, editors, *Conference on Fairness, Accountability and Transparency, FAT 2018, 23-24 February 2018, New York, NY, USA*, volume 81 of *Proceedings of Machine Learning Research*, pages 149–159. PMLR, 2018.

[BM12] Iain Brown and Christophe Mues. An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Systems with Applications*, 39(3):3446–3453, 2012.

[BPM04] Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, 6(1):20–29, 2004.

[Bre01] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[BSL19] Jakim Berndsen, Barry Smyth, and Aonghus Lawlor. Pace my race: recommendations for marathon running. In Toine Bogers, Alan Said, Peter Brusilovsky, and Domonkos Tikk, editors, *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019*, pages 246–250. ACM, 2019.

[BSO18] Robin Burke, Nasim Sonboli, and Aldo Ordonez-Gauger. Balanced neighborhoods for multi-sided fairness in recommendation. In *Conference on Fairness, Accountability and Transparency, FAT 2018*, volume 81 of *Proceedings of Machine Learning Research*, pages 202–214. PMLR, 2018.

[CBHK02] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[CG98] Jaime G. Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 335–336. ACM, 1998.

[Cho17] Francois Chollet. *Deep learning with Python.* Simon and Schuster, 2017.

[CMM84] JG Carbonell, RS Michalski, and TM Mitchell. An overview of machine learning. chapter 1. *Machine Learning dan Artificial Intelligence Approach (Eds. Michaliski, RS, Carbonell, JG, Mitchell. TM): springer-Verlag, New York*, 1984.

[CSV18] L. Elisa Celis, Damian Straszak, and Nisheeth K. Vishnoi. Ranking with fairness constraints. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018*, volume 107 of *LIPIcs*, pages 28:1–28:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[DHP$^+$12] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S. Zemel. Fairness through awareness. In Shafi Goldwasser, editor, *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 214–226. ACM, 2012.

[DIDT11] Mihnea Donciu, Madalina Ionita, Mihai Dascalu, and Stefan Trausan-Matu. The runner - recommender system of workout and nutrition for runners. In Dongming Wang, Viorel Negru, Tetsuo Ida, Tudor Jebelean, Dana Petcu, Stephen M. Watt, and Daniela Zaharie, editors, *13th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2011, Timisoara, Romania, September 26-29, 2011*, pages 230–238. IEEE Computer Society, 2011.

[DLPS14] Vito D'Orazio, Steven T Landis, Glenn Palmer, and Philip Schrodt. Separating the wheat from the chaff: Applications of automated document classification using support vector machines. *Political analysis*, 22(2):224–242, 2014.

[DSG14] Cicero Dos Santos and Maira Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, 2014.

[DZ16] Remus-Alexandru Dobrican and Denis Zampuniéris. A proactive solution, using wearable and mobile applications, for closing the gap between the rehabilitation team and cardiac patients. In *2016 IEEE International Conference on Healthcare Informatics, ICHI 2016, Chicago, IL, USA, October 4-7, 2016*, pages 146–155. IEEE Computer Society, 2016.

[FBB+92] Gerald F Fletcher, Steven N Blair, James Blumenthal, Carl Caspersen, Bernard Chaitman, Stephen Epstein, Harold Falls, ES Froelicher, Victor F Froelicher, and Ileana L Piña. Statement on exercise. benefits and recommendations for physical activity programs for all americans. a statement for health professionals by the committee on exercise and cardiac rehabilitation of the council on clinical cardiology, american heart association. *Circulation*, 86(1):340–344, 1992.

[FHW+95] Carl Foster, Lisa L Hector, Ralph Welsh, Mathew Schrager, Megan A Green, and Ann C Snyder. Effects of specific versus cross-training on running performance. *European journal of applied physiology and occupational physiology*, 70(4):367–372, 1995.

[Fos98] CARL Foster. Monitoring training in athletes with reference to overtraining syndrome. *Medicine and science in sports and exercise*, 30(7):1164–1168, 1998.

[FY15] Mi Fei and Dit-Yan Yeung. Temporal models for predicting student dropout in massive open online courses. In *ICDM Workshops*, pages 256–263. IEEE Computer Society, 2015.

[GAK19] Sahin Cem Geyik, Stuart Ambler, and Krishnaram Kenthapadi. Fairness-aware ranking in search & recommendation systems with application to linkedin talent search. In Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis, editors, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 2221–2231. ACM, 2019.

[GEW06] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[GFB+11] Mikel Galar, Alberto Fernández, Edurne Barrenechea, Humberto Bustince, and Francisco 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 Recognition*, 44(8):1761–1776, 2011.

[Gil96] Muriel B Gilman. The use of heart rate to monitor the intensity of endurance training. *Sports Medicine*, 21(2):73–79, 1996.

[GIT21] Theresa Gattermann-Itschert and Ulrich W Thonemann. How training on multiple time slices improves performance in churn prediction. *European Journal of Operational Research*, 2021.

[GMH13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee, 2013.

[GP16] Ido Guy and Luiz Pizzato. People recommendation tutorial. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 431–432. ACM, 2016.

[HAST14] Qian He, Emmanuel Agu, Diane M. Strong, and Bengisu Tulu. Recfit: a context-aware system for recommending physical activities. In Sandeep K. S. Gupta and Ayan Banerjee, editors, *Proceedings of the 1st Workshop on Mobile Medical Applications, MMA '14, Memphis, Tennessee, USA, November 3-6, 2014*, pages 34–39. ACM, 2014.

[HLR06] H Gholam Hosseini, Dehan Luo, and Karen Jane Reynolds. The comparison of different feed forward neural network architectures for ecg signal diagnosis. *Medical engineering & physics*, 28(4):372–378, 2006.

[HPS16] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3315–3323, 2016.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HTBL18] Jevan A. Hutson, Jessie G. Taft, Solon Barocas, and Karen Levy. Debiasing desire: Addressing bias & discrimination on intimate platforms. *Proc. ACM Hum. Comput. Interact.*, 2(CSCW):73:1–73:18, 2018.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015.

[JC17] Katarzyna Janocha and Wojciech Marian Czarnecki. On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659*, 2017.

[JM15] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[KAAS18] Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. Recommendation independence. In *Conference on Fairness, Accountability and Transparency, FAT 2018*, volume 81 of *Proceedings of Machine Learning Research*, pages 187–201. PMLR, 2018.

[Kam17] Bart A. Kamphorst. E-coaching systems - what they are, and what they aren't. *Personal and Ubiquitous Computing*, 21(4):625–632, 2017.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[KFI+19] Mohammed Khwaja, Miquel Ferrer, Jesus Omana Iglesias, A. Aldo Faisal, and Aleksandar Matic. Aligning daily activities with personality: towards a recommender system for improving wellbeing. In Toine Bogers, Alan Said, Peter Brusilovsky, and Domonkos Tikk, editors, *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019*, pages 368–372. ACM, 2019.

[KMM+15] Michel C. A. Klein, Adnan R. Manzoor, Anouk Middelweerd, Julia S. Mollee, and Saskia J. te Velde. Encouraging physical activity via a personalized mobile system. *IEEE Internet Computing*, 19(4):20–27, 2015.

[KWB06] Willemieke Kroeze, Andrea Werkman, and Johannes Brug. A systematic review of randomized trials on the effectiveness of computer-tailored education on physical activity and dietary behaviors. *Annals of behavioral medicine*, 31(3):205–223, 2006.

[KWMM09] William Klement, Szymon Wilk, Wojtek Michaowski, and Stan Matwin. Dealing with severely imbalanced data. In *Proc. of the PAKDD Conference*, page 14. Citeseer, 2009.

[LBH+15] Yann LeCun, Yoshua Bengio, Geoffrey Hinton, et al. Deep learning. nature 521 (7553), 436-444. *Google Scholar Google Scholar Cross Ref Cross Ref*, 2015.

[LGL+16] Wentao Li, Min Gao, Hua Li, Qingyu Xiong, Junhao Wen, and Zhongfu Wu. Dropout prediction in moocs using behavior features and multi-view semi-supervised learning. In *IJCNN*, pages 3130–3137. IEEE, 2016.

[Lis15] Pierre Lison. An introduction to machine learning. *Language Technology Group (LTG)*, 1(35):1–35, 2015.

[LNA17] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.

[LWBT18] Haiyang Liu, Zhihai Wang, Phillip Benachour, and Philip Tubman. A time series classification method for behaviour-based dropout prediction. In *ICALT*, pages 191–195. IEEE Computer Society, 2018.

[MAA+87] JJ McMurray, S Adamopoulos, SD Anker, A Auricchio, and M Bohm. Dickstein k et al. esc guidelines for the diagnosis and treatment of acute and chronic heart failure 2012: The task force for the diagnosis and treatment of acute and chronic heart failure 2012 of the european society of cardiology. developed in collaboration with the heart failure association (hfa) of the esc. 1787.

[Mar16] A Mark. Hall eibe frank and ian h. witten. data mining: Practical machine learning tools and techniques, 2016.

[MBRF20] Mirko Marras, Ludovico Boratto, Guilherme Ramos, and Gianni Fenu. Equality of learning opportunity in personalized recommendations. *arXiv preprint arXiv:2006.04282*, 2020.

[MdCZ14] Laci Mary Barbosa Manhães, Sérgio Manuel Serra da Cruz, and Geraldo Zimbrão. WAVE: an architecture for predicting dropout in undergraduate courses using EDM. In Yookun Cho, Sung Y. Shin, Sang-Wook Kim, Chih-Cheng Hung, and Jiman Hong, editors, *Symposium on Applied Computing, SAC 2014, Gyeongju, Republic of Korea - March 24 - 28, 2014*, pages 243–247. ACM, 2014.

[MGA+16] Corby K Martin, L Anne Gilmore, John W Apolzan, Candice A Myers, Diana M Thomas, and Leanne M Redman. Smartloss: a personalized mobile health intervention for weight management and health promotion. *JMIR mHealth and uHealth*, 4(1):e18, 2016.

[MMB+18] Rishabh Mehrotra, James McInerney, Hugues Bouchard, Mounia Lalmas, and Fernando Diaz. Towards a fair marketplace: Counterfactual evaluation of the trade-off between relevance, fairness & satisfaction in recommendation systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018*, pages 2243–2251. ACM, 2018.

[MSKI21] Derek G Murray, Jiri Simsa, Ana Klimovic, and Ihor Indyk. tf. data: A machine learning data processing framework. *arXiv preprint arXiv:2101.12127*, 2021.

[Nil14] NJ Nilsson. Principles of artificial intelligence. burlington, ma, 2014.

[NK13] Alexey Natekin and Alois Knoll. Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7:21, 2013.

[NodAV14] Mohammad Hossein Nassabi, Harm op den Akker, and Miriam M. R. Vollenbroek-Hutten. An ontology-based recommender system to promote physical activity for pre-frail elderly. In Andreas Butz, Michael Koch, and Johann H. Schlichter, editors, *Mensch & Computer 2014 - Workshopband, 14. Fachübergreifende Konferenz für Interaktive und Kooperative Medien - Interaktiv unterwegs - Freiräume gestalten, 31. August - 3. September 2014, München, Germany*, pages 181–184. De Gruyter Oldenbourg, 2014.

[PBG$^+$20] Gourab K. Patro, Arpita Biswas, Niloy Ganguly, Krishna P. Gummadi, and Abhijnan Chakraborty. Fairrec: Two-sided fairness for personalized recommendations in two-sided platforms. In *WWW '20: The Web Conference 2020*, pages 1194–1204. ACM / IW3C2, 2020.

[PKB18] Despoina Petsani, Evdokimos I. Konstantinidis, and Panagiotis D. Bamidis. Designing an e-coaching system for older people to increase adherence to exergame-based physical activity. In Panagiotis D. Bamidis, Martina Ziefle, and Leszek A. Maciaszek, editors, *Proceedings of the 4th International Conference on Information and Communication Technologies for Ageing Well and e-Health, ICT4AWE 2018, Funchal, Madeira, Portugal, March 22-23, 2018*, pages 258–263. SciTePress, 2018.

[Pow11] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

[Pow12] David MW Powers. The problem with kappa. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 345–355. Association for Computational Linguistics, 2012.

[PPB$^+$17] Paolo Pilloni, Luca Piras, Ludovico Boratto, Salvatore Carta, Gianni Fenu, and Fabrizio Mulas. Recommendation in persuasive ehealth systems: an effective strategy to spot users' losing motivation to exercise. In David Elsweiler, Santiago Hors-Fraile, Bernd Ludwig, Alan Said, Hanna Schäfer, Christoph Trattner, Helma Torkamaan, and André Calero Valdez, editors, *Proceedings of the 2nd International Workshop on Health Recommender Systems co-located with the 11th International Conference on Recommender Systems (RecSys 2017), Como, Italy, August 31, 2017*, volume 1953 of *CEUR Workshop Proceedings*, pages 6–9. CEUR-WS.org, 2017.

[PPC$^+$18] Paolo Pilloni, Luca Piras, Salvatore Carta, Gianni Fenu, Fabrizio Mulas, and Ludovico Boratto. Recommender system lets coaches identify and help athletes who begin losing motivation. *Computer*, 51(3):36–42, 2018.

[QLHL19] Lin Qiu, Yanshen Liu, Quan Hu, and Yi Liu. Student dropout prediction in massive open online courses by convolutional neural networks. *Soft Comput.*, 23(20):10287–10301, 2019.

[RC10] Filip Radlinski and Nick Craswell. Comparing the sensitivity of information retrieval metrics. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 667–674. ACM, 2010.

[RK17] Santosh S Rathore and Sandeep Kumar. A decision tree logic based recommendation system to select software fault prediction techniques. *Computing*, 99(3):255–285, 2017.

[Sar21] Iqbal H Sarker. Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, 2(3):1–21, 2021.

[Sas99] Ravindra Vadali Sastry. *A Need for Speed: A New Speedometer for Runners.* PhD thesis, Massachusetts Institute of Technology, 1999.

[SC18a] Barry Smyth and Pádraig Cunningham. An analysis of case representations for marathon race prediction and planning. In Michael T. Cox, Peter Funk, and Shahina Begum, editors, *Case-Based Reasoning Research and Development - 26th International Conference, ICCBR 2018, Stockholm, Sweden, July 9-12, 2018, Proceedings*, volume 11156 of *Lecture Notes in Computer Science*, pages 369–384. Springer, 2018.

[SC18b] Barry Smyth and Padraig Cunningham. Marathon race planning: A case-based reasoning approach. In Jérôme Lang, editor, *Proceedings* *of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 5364–5368. ijcai.org, 2018.

[SCC19] Javier Sanz-Cruzado and Pablo Castells. *Contact Recommendations in Social Networks*, chapter Chapter 16, pages 519–569. 2019.

[Sch15] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[SF20] Scott Sittig and Amy L Franklin. Persuasive technology: Designing mobile health triggers to impact health behavior. 2020.

[SJ18] Ashudeep Singh and Thorsten Joachims. Fairness of exposure in rankings. In Yike Guo and Faisal Farooq, editors, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 2219–2228. ACM, 2018.

[SKW16] José A Sáez, Bartosz Krawczyk, and Michał Woźniak. Analyzing the oversampling of different classes and types of examples in multi-class imbalanced datasets. *Pattern Recognition*, 57:164–178, 2016.

[Smy19] Barry Smyth. Recommender systems: A healthy obsession. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 9790–9794. AAAI Press, 2019.

[SP97] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[SSB14] Hasim Sak, Andrew W Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. 2014.

[SSG$^+$19] Juan M. Santos-Gago, Luis Álvarez Sabucedo, Roberto González-Maciel, Víctor M. Alonso Rorís, José L. García-Soidán, Carmina Wanden-Berghe, and Javier Sanz-Valero. Towards a personalised recommender platform for sportswomen. In Álvaro Rocha, Hojjat Adeli, Luís Paulo Reis, and Sandra Costanzo, editors, *New Knowledge in Information Systems and Technologies - Volume 1, World Conference on Information Systems and Technologies, WorldCIST 2019, Galicia,* *Spain, 16-19 April, 2019*, volume 930 of *Advances in Intelligent Systems and Computing*, pages 504–514. Springer, 2019.

[STVS07] Karl M Stagno, Rhys Thatcher, and Ken A Van Someren. A modified trimp to quantify the in-season training load of team sport players. *Journal of sports sciences*, 25(6):629–634, 2007.

[SVSS15] Tara N Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak. Convolutional, long short-term memory, fully connected deep neural networks. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4580–4584. IEEE, 2015.

[TH17] T Tieleman and G Hinton. Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning. *Technical Report*, 2017.

[TLL$^+$15] Jerry C. C. Tseng, Bo-Hau Lin, Yu-Feng Lin, Vincent S. Tseng, Miin-Luen Day, Shyh-Chyi Wang, Kuen-Rong Lo, and Yi-Ching Yang. An interactive healthcare system with personalized diet and exercise guideline recommendation. In *Conference on Technologies and Applications of Artificial Intelligence, TAAI 2015, Tainan, Taiwan, November 20-22, 2015*, pages 525–532. IEEE, 2015.

[VXD$^+$14] Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. Translating videos to natural language using deep recurrent neural networks. *arXiv preprint arXiv:1412.4729*, 2014.

[WYM17] Wei Wang, Han Yu, and Chunyan Miao. Deep model for dropout prediction in moocs. In *ICCSE*, pages 26–32. ACM, 2017.

[YS17] Ke Yang and Julia Stoyanovich. Measuring fairness in ranked outputs. In *Proceedings of the 29th International Conference on Scientific and Statistical Database Management, Chicago, IL, USA, June 27-29, 2017*, pages 22:1–22:6. ACM, 2017.

[YTFK$^+$17] Elad Yom-Tov, Guy Feraru, Mark Kozdoba, Shie Mannor, Moshe Tennenholtz, and Irit Hochberg. Encouraging physical activity in patients with diabetes: intervention using a reinforcement learning system. *Journal of medical Internet research*, 19(10):e338, 2017.

[ZBC$^+$17] Meike Zehlike, Francesco Bonchi, Carlos Castillo, Sara Hajian, Mohamed Megahed, and Ricardo Baeza-Yates. Fa*ir: A fair top-k ranking algorithm. In Ee-Peng Lim, Marianne Winslett, Mark Sanderson, Ada Wai-Chee Fu, Jimeng Sun, J. Shane Culpepper, Eric Lo, Joyce C. Ho, Debora Donato, Rakesh Agrawal, Yu Zheng, Carlos Castillo, Aixin

Sun, Vincent S. Tseng, and Chenliang Li, editors, *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*, pages 1569–1578. ACM, 2017.

[ZC20] Meike Zehlike and Carlos Castillo. Reducing disparate exposure in ranking: A learning to rank approach. In *WWW '20: The Web Conference 2020*, pages 2849–2855. ACM / IW3C2, 2020.

[ZZ06] Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 2006.

La borsa di dottorato è stata cofinanziata con risorse del
Programma Operativo Nazionale Ricerca e Innovazione 2014-2020 (CCI 2014IT16M2OP005),
Fondo Sociale Europeo, Azione I.1 "Dottorati Innovativi con caratterizzazione Industriale"

UNIONE EUROPEA
Fondo Sociale Europeo

Ministero dell'Università
e della Ricerca

PON
RICERCA
E INNOVAZIONE
2014 - 2020