Open challenges for Machine Learning based Early Decision-Making research

Alexis Bondu[1], Youssef Achenchabe[1,2], Albert Bifet[3,5], Fabrice Clérot[1], Antoine Cornuéjols[2], João Gama[4], Georges Hébrail[3], Vincent Lemaire[1], and Pierre-François Marteau[6]

[1] Orange Labs, France
[2] Paris-Saclay University, Agroparistech, France
[3] IPP, Télécom Paristech, France
[4] University of Porto, Portugal
[5] University of Waikato, New Zealand
[6] Bretagne-Sud University, France

ABSTRACT

More and more applications require *early* decisions, i.e. taken as soon as possible from partially observed data. However, the *later* a decision is made, the more its accuracy tends to improve, since the description of the problem to hand is enriched over time. Such a compromise between the *earliness* and the *accuracy* of decisions has been particularly studied in the field of Early Time Series Classification. This paper introduces a more general problem, called Machine Learning based Early Decision Making (ML-EDM), which consists in optimizing the decision times of models in a wide range of settings where data is collected over time. After defining the ML-EDM problem, ten challenges are identified and proposed to the scientific community to further research in this area. These challenges open important application perspectives, discussed in this paper.

1. INTRODUCTION

In numerous real situations, we have to make *early* decisions in the absence of *complete knowledge* of the problem at hand. For example, such decisions are necessary in medicine [54] when a physician must make a diagnosis, possibly leading to an urgent surgical operation, before having the results of all medical tests. In such situations, the issue facing the decision makers is that, most of the time, the longer the decision is delayed, the clearer is the likely outcome (e.g. the critical or not critical state of the patient) but, also, the higher the cost that will be incurred if only because decisions taken earlier allow one to be better prepared. We thus seek to make decisions at times that seem to be the best compromises between the *earliness* and the *accuracy* of our decisions.

Similarly in Machine Learning, when the input data is acquired over time, there can be situations with a trade-off between the earliness and accuracy of decisions. For instance, this is the case for anomaly detection, predictive maintenance, patient health monitoring, self-driving vehicles (see Section 10). In each case, the decisions are *time-sensitive* (e.g. in an autonomous car, it is critical to detect obstacles on the road as early as possible and at the same time as reliably as possible, in order to plan safe avoidance trajectories if needed). In general, it is assumed that there is a *gain of information* over time, i.e. delaying decisions tends to make them more reliable (e.g. the certainty about the existence or absence of an obstacle on the road becomes more and more accurate as the car gets closer).

This earliness *vs.* accuracy dilemma is part of many decision making scenarios, and is particularly involved in the problem of Early Classification of Time Series (ECTS). But, as we will see, it takes place in a larger perspective.

Early Classification of Time Series: a particular case

The ECTS problem consists in finding the *optimal time* to trigger the class prediction of an input time series observed over time. As successive measurements provide more and more information about the incoming time series, ECTS algorithms aim to optimize online the trade-off between the *earliness* and the *accuracy* of their decisions.

More formally, the individuals[1] considered are time series of *finite length* T, during which a decision must be made. At testing time, the measurements of the incoming time series are received over time, and the history of measurements available at time t is denoted by $\mathbf{x}_t = \langle x_1, \ldots, x_t \rangle$. It is assumed that each time series can be ascribed to some class $y \in \mathcal{Y}$, and the task is to make a prediction about the class of each incoming time series as early as possible, because a time increasing cost must be paid when the decision is triggered. In the ECTS problem, a single decision is triggered for each incoming time series, which is irrevocable and final. An ECTS approach is generally made of two main components: (i) an *hypothesis*[2] $h \in \mathcal{H}$ capable of predicting the class $y \in \mathbb{Y}$ of the incoming series at any time, such that $h(\mathbf{x}_t) = \hat{y}$ with $t \in [1, T]$; (ii) a *triggering strategy* capable of making decisions at the right moments, denoted by $Trigger$. Both the hypothesis and the triggering strategy are learned in batch mode (i.e. offline), by using a training set made of complete time series with their associated labels.

[1]The term *individual* refers to any type of statistical unit studied.

[2]An hypothesis is a candidate predictor which approximates the concept $P(y|\mathbf{x}_t)$.

A short overview of Early Classification of Time Series

This paragraph provides an overview of the ECTS approaches. For a recent and more complete survey, the reader can refer to [2, 34]. Doubts about the relevance of the ECTS framework and questions about the definition of the problem have been raised in [82]. The present paper sets up a formal framework and explores new directions to enrich this area and underlines its practical significance.

The pioneering approaches were based on some form of *confidence* criterion and waited until a predefined threshold is reached before triggering their decisions. For instance, in [32, 36, 62], a classifier is learned for each time step and various stopping rules are used (e.g. threshold on confidence level). In [83], such a threshold is indirectly set since the best time step to trigger the decision is estimated by determining the earliest time step for which the predicted label does not change, based on a 1NN classifier. Similarly, [58] proposes a method where the accuracy of a set of probabilistic classifiers is monitored over time, which allows the identification of time steps from whence it seems safe to make predictions.

Then, more informed approaches appeared which explicitly take into account the cost of delaying the decisions. A notable example is [59] where the conflict between earliness and accuracy is explicitly addressed. Moreover, instead of setting the trade-off in a single objective optimization criterion as in [57], the authors keep it as a multi-objective criterion and to explore the Pareto front of the multiple dominating trade-offs.

The Economy approach [2, 18] goes one step further by casting the ECTS problem as searching to optimize a loss function which combines the expected cost of misclassification at the time of decision, plus the cost of having delayed the decision thus far. This well-founded approach is *non-myopic*, as it is able to anticipate measurements which are not yet visible at decision time by estimating the expected costs for future time steps. This approach leads to the best performances observed to date, and [2] shows that the non-myopic feature of this approach explains its strong performances through an ablation study.

Limitations of the ECTS problem

While ECTS covers a wide range of applications, it does not exhaust all cases where a *Machine Learning* model can be applied on data acquired over time, and where the trade-off between the *earliness* and the *accuracy* of decisions must be optimized. Indeed, ECTS, as defined above, is limited to:

- a classification problem ;
- an available training set which contains completely and properly labeled time series ;
- a decision deadline that is finite, fixed and known ;
- unique decisions for each incoming time series ;
- decisions that once made can never be reconsidered ;
- fixed decision costs which do not depend on the triggering time and the decisions made.

All of these assumptions might be questioned and point to research issues. The purpose of this paper is to propose research directions for extending ECTS toward a more generic problem, that we call *Machine Learning based Early Decision-Making* (ML-EDM).

This position paper is organised as follows. Section 2 first defines the ML-EDM problem, shows how a triggering strategy can be learned, and positions ML-EDM with respect to Reinforcement Learning. A series of ten challenges is then proposed in order to develop ML-EDM approaches for a wide range of problems. Section 3 explains the deep origin of the delay costs involved in the ML-EDM problems. Section 4 considers a variety of *learning tasks*, and Section 5, a variety of *data types*. Section 6 gives some leads to address the problem of *online* ML-EDM. Section 7 extends ML-EDM to *revocable decisions*. Section 8 specifies the other decision costs involved, and shows how they can vary depending on the triggering time and the decisions made. Section 9 gives an overview on the proposed challenges, and makes a synthesis of long and short term application perspectives. Then, Section 10 provides some examples of *applications* of the ML-EDM techniques and Section 11 illustrates how the loss function can be defined and how to use it for evaluation purposes. At last, Section 12 concludes with perspectives for the development of the ML-EDM field in the coming years.

2. DEFINITION OF ML-EDM

This section defines what ML-EDM is by answering the following questions:
A- What is an early decision? **B-** How to learn a triggering strategy from training data? **C-** Can a triggering strategy be learned by Reinforcement Learning?

Question A - *What is an early decision?*

Two types of problems can be distinguished [35]. Decision-making *under ignorance* refers to a category of problems where the set of possible outcomes is known, but no information about their probabilities is available. By contrast, decision-making *under uncertainty* deals with problems where the probabilities of the possible outcomes are known, or partially known.

Basically, Early Decision Making consists in: (i) observing pieces of information over time ; (ii) deciding when to make a decision ; and (iii) making the decision itself. In the following, increasingly complex decision-making problems are considered, numbered from $(\boldsymbol{I})$ to $(\boldsymbol{IV})$, in order to progressively lead to a general definition of ML-EDM.

$(\boldsymbol{I})$ - ***Optimal Stopping Problem*** [73] is a canonical case of interest, where the decision to make is simply to stop receiving new pieces of information. More formally, $\{X_i\}$ is a sequence of random variables observed successively, whose joint distribution is known. Let $\{r_i\}$ be a sequence of reward functions, such that r_i is a function of the observed values $x_1, \ldots, x_i$. The objective is to maximize the reward, deciding after observing the value of the random variable X_i, either to stop and accept the reward r_i, or to observe the value of the next random variable X_{i+1}. A number of optimal stopping problems have been extensively studied in the literature, such as:

- The *Shepp's urn* [73] which is filled with a known number of \$1 bills, and a known number of anti-bills of -\$1.

Here, the reward is the sum of the bills gathered until the end of the game. The objective is to maximize our payoff by stopping to draw objects in this urn at the best time.

- The *secretary problem* [24] consists in selecting the largest possible value (which is unknown), among a sequence of values of known size observed in a uniform random order. At each step the choice is, either to stop and keep the last observed value, or to continue.

These two problems involve decision making *under uncertainty*, since the system under study is perfectly known and the probability of the possible outcomes can be estimated. For instance, in the Shepp's urn the probability of getting a bill or an anti-bill in the next draw is available, since the content of the urn is known at any time. In the secretary problem, the rank of the last value among the previously observed values approximates the rank in the entire set of values, since the observed values constitute a uniform sample of all values.

As in Early Decision Making problem, Shepp's urn and the secretary problem imply a trade-off between *early* and *accurate* decisions.

On the one hand, there is a *time pressure* which pushes to trigger early decisions. In a Shepp's urn, the number of objects is finite and if all of them are drawn, our payoff is bad, i.e. equal to the number of bills minus the number of anti-bills. In the secretary problem, the number of values is known. The more values are observed, the less future opportunity remains to select a high value.

On the other hand, there is a *gain of information* (about what's left in the urn) over time which tends to delay the decisions. In the Shepp's urn problem, the sample of already drawn objects grows over time, which provides useful information to be compared to the known quantities of bills and anti-bills. For the secretary's problem, the sample of already drawn values grows over time, and the last observed value can be compared to this sample.

Note: from here on, the decision-making problems presented in the following are part of *supervised learning.* A set of labeled examples, which takes different forms depending on the problem, is assumed to be available.

(II)* - *The ECTS problem can be considered as a particular instance of optimal stopping, where the decision to be made consists in: (*i*) stopping receiving new measurements ; and (*ii*) predicting the class of the incoming time series. The hypothesis $h \in \mathcal{H}$ is assumed to be available, allowing to predict the class $y \in \mathbb{Y}$ of the incoming series at any time, such that $h(\mathbf{x}_t) = \hat{y}$. In this case, the reward function $r(\mathbf{x}_t, t, \hat{y}, y)$ depends on the observed measurements $\mathbf{x}_t = \langle x_1, \dots, x_t \rangle$; the decision time t ; the predicted class $\hat{y}$; and the true class y. The following loss function can be defined:

$$\mathcal{L}(h(\mathbf{x}_t), t, y) = \mathcal{L}_{prediction}\,(h(\mathbf{x}_t), y) \;+ \mathcal{L}_{delay}(t) \qquad (1)$$

where $\mathcal{L}_{prediction}(.)$ is the cost of making a potentially bad prediction which can be expressed as a cost matrix, and $\mathcal{L}_{delay}(t)$ is a monotonically increasing function of t representing the cost of delaying the decision until t[3]. The best decision time t^* is given by the optimal triggering strategy $Trigger^*$ defined as:

$$Trigger^*\,(h(\mathbf{x}_t)) = \begin{cases} 1 & \text{if } t = t^* = \arg\min_{t\in[1,T]} \mathcal{L}(h(\mathbf{x}_t), t, y) \text{ or } t = T \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

where the decision is forced at $t = T$ if it was not taken before.

Here, the trade-off between *early* and *accurate* decisions takes the following form. On the one hand, the delay cost $\mathcal{L}_{delay}(t)$ incurred in making a decision urges to make an early decision. On the other hand, the cost of making a bad prediction $\mathcal{L}_{prediction}$ is assumed to decrease over time, as the description of the incoming time series becomes richer. This decision making problem is *under uncertainty*, since the hypothesis h is capable of estimating the distribution of the possible outcomes $P(y|\mathbf{x}_t)$, at any time.

In practice, ECTS approaches trigger decisions at $\hat{t}$, hopefully the closest as possible to the optimal time t^*, at least in terms of cost: $\mathcal{L}(h(\mathbf{x}_{\hat{t}}), \hat{t}, y) - \mathcal{L}(h(\mathbf{x}_{t^\star}), t^\star, y)$ must be small. Triggering such a decision is an *online* optimization problem, since $\hat{t}$ must be chosen based on a partial description $\mathbf{x}_t$ of the incoming time series $\mathbf{x}_T$ (with $t \leq T$), and the reward function can be defined as:

$$r(\mathbf{x}_t, t, h(\mathbf{x}_t), y) = \begin{cases} -\mathcal{L}(h(\mathbf{x}_t), t, y) & \text{if } t = \hat{t} \text{ or } t = T \\ 0 & \text{otherwise} \end{cases} \qquad (3)$$

where the risk equals to 0 when no decision is made, given that the decision is forced at $t = T$ resulting in an important risk due to the delay cost.

Note: in the rest of this section, and for readability reasons, T (the end of the considered time period) is still considered as finite and known, as in the ECTS problem. In Section 7, another setting is studied where T is indeterminate, i.e. where the successive measurements are observed as a data stream.

(III)* - *Early decisions to be located in time constitute a more challenging problem, which consists of both making a decision for each incoming time series, but also predicting a *time period* associated with the decision. For example, maintenance operations on hydroelectric dam turbines can only be performed when the electricity demand is at a low enough level. There are therefore periods where maintenance is possible and periods where this is not desirable. The objective here is to determine as early as possible *whether*

[3]Note that the delay cost $\mathcal{L}_{delay}(t)$ could depend on the class y of the time series (see Section 3). For instance, in the emergency department in a hospital, the cost of delaying a decision when there is internal bleeding is not the same as the one in case of gastroenteritis, where the early symptoms could look the same. Here, for reasons of readability, we make $\mathcal{L}_{delay}$ depend only on t.

and during *which period* it will be possible to shut down the turbines, within the day (if $[1, T]$ corresponds to one day). In this case, the ground truth $(y, (s, e))$ consists of a class $y \in \mathbb{Y}$, associated with a certain time period $[s, e]$, defined by a *start* timestamp $s \in [1, T]$ and a *end* timestamp $e \in [s, T]$. At testing time, the objective is twofold: triggering the decision as early as possible, while also predicting the associated time period $[s, e]$. Let us consider a decision denoted by $(h(\mathbf{x}_{\hat{t}}), (\hat{s}, \hat{e}))$, where $h(\mathbf{x}_{\hat{t}})$ is the class predicted at $\hat{t}$ (the triggering time), and $[\hat{s}, \hat{e}]$ is the associated predicted time period. The loss function $\mathcal{L}$ has to be redefined as a function of the following parameters:

$$\mathcal{L}\left(\underbrace{(h(\mathbf{x}_{\hat{t}}), (\hat{s}, \hat{e}))}_{predictions}, \underbrace{\hat{t}}_{triggering\ time}, \underbrace{(y, (s, e))}_{ground\ truth}\right) \quad (4)$$

The loss function $\mathcal{L}$ needs to be specified further, depending on the considered application. In general, this loss function should account for two aspects: (i) the *quality* of the predictions ; (ii) the *time overlap* between the decisions made and the true decisions. For example, Figure 1 shows a situation where the decision made is correct, since the predicted class (see the second line) matches the ground truth (see the first line). But these two decisions do not coincide exactly in time, as the predicted time period is earlier than the ground truth.

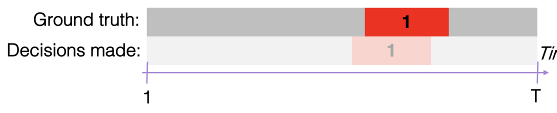


Figure 1: Example of a time-lagged decision.

(IV) - **ML-EDM**[4] considers, by extension, *multiple early decisions to be located in time* (i.e. in the time period $[1, T]$). , which is necessary in numerous applications. For example, consider a set of servers used to trade on a stock exchange platform (where $[1, T]$ corresponds to the platform's hours of operation during the day). For each server, key performance indices (e.g., CPU, RAM, network) are recorded over time. The ground truth consists of a sequence of states (e.g., overload or nominal) associated with the corresponding time periods. In this application, the task is to detect overload periods as early as possible.

Thus, in this problem, the true decisions $\{y_i, (s_i, e_i)\}_{i=1}^{k_\mathbf{x}}$ consists of a sequence of varying length $k_\mathbf{x}$, which is specific to each individual $\mathbf{x}$. Each element of this sequence is a decision to be *located in time*, which consists of a class $y_i \in \mathbb{Y}$ associated with a certain time period $[s_i, e_i]$. For a given individual $\mathbf{x}$, the time periods $\{(s_i, e_i)\}_{i=1}^{k_\mathbf{x}}$ constitute a *time partition*, each interval $[s_i, e_i]$ being associated with the true class y_i (e.g. in predictive monitoring, this time partition would correspond to the successive states, up or down, of a given device).

Here, the online optimization problem to be addressed is more complex than the previous one, since it consists in triggering a sequence of decisions as soon as possible, without knowing the number of true decisions $k_\mathbf{x}$, and also ignoring the time periods associated with each true decision. Let us consider that an ML-EDM approach triggers a sequence of decisions $\{h(\mathbf{x}_{\hat{t}_{i'}}), (\hat{s}_{i'}, \hat{e}_{i'})\}_{i'=1}^{\hat{k}_\mathbf{x}}$; where $\hat{k}_\mathbf{x}$ is the number of decisions made ; where $\{\hat{t}_{i'}\}_{i'=1}^{\hat{k}_\mathbf{x}}$ represents the associated triggering times ; and where $\{(\hat{s}_{i'}, \hat{e}_{i'})\}_{i'=1}^{\hat{k}_\mathbf{x}}$ represents the predicted time periods associated to the decisions which forms a partition of the time period $[1, T]$. In the scenario of *multiple early decisions to be located in time*, a loss function $\boldsymbol{\mathcal{L}}$ needs to be defined as a function of the following parameters:

$$\boldsymbol{\mathcal{L}}\left(\underbrace{\{h(\mathbf{x}_{\hat{t}_{i'}}), (\hat{s}_{i'}, \hat{e}_{i'})\}_{i'=1}^{\hat{k}_\mathbf{x}}}_{predictions}, \underbrace{\{\hat{t}_{i'}\}_{i'=1}^{\hat{k}_\mathbf{x}}}_{triggering\ times}, \underbrace{\{y_i, (s_i, e_i)\}_{i=1}^{k_\mathbf{x}}}_{ground\ truth}\right) \quad (5)$$

This equation shows the loss function used to evaluate an approach after the end timestamp T, when predictions have been taken for all instants in the time period $[1, T]$ (see Section 11.2 for more details). The loss function $\boldsymbol{\mathcal{L}}$ can be expressed in many different ways, depending on the application considered. In practice, *mapping rules* need to be defined to match the decisions made to the true ones (see Section 11.1).
Note that the problem of making predictions for all instants in $[1, T]$ points to the issue as whether decisions made can be revoked, or not, before T. In case decisions are irrevocable, once a decision has been made, let us say $(y, (s, e))$, then it is no longer possible to change the prediction of the class for all times $t \in (s, e)$. This renders the optimization problem dependent upon previous decisions, and it becomes more constraining for application cases. Revocable decision are studied in Section 7.

- The ***deadline***, denoted by $\mathcal{T}$, after which decisions are forced is an important component, that takes different forms depending on the problem. In the simple case of ECTS, only one decision needs to be made before the incoming time series is complete. Thus, the deadline is defined as the *maximum size* of the input series ($\mathcal{T} = T$), which is known in advance during training. By contrast, in the more complex case of ML-EDM where multiple decisions to be located in time must be taken before the end timestamp T, the deadline $\mathcal{T}$ is defined as a *maximum delay* allowed to detect the start of a true decision. In practice, two situations can be distinguished:

- Some applications do *not support the absence of decision*, and the entire considered time period must be partitioned by the successive decisions. This is the case for instance when moderating content on social networks, where discussions are continuously going on between users and where each part of these discussions must be classified as *appropriate* or *not* (see Section 10.3). In this case, no decision is not allowed, and all decisions are subject to the cost $\mathcal{L}_{delay}$ and thus constrained by the deadline $\mathcal{T}$.

- By contrast, in some applications, a *nominal operating state* exists which is almost permanent, and for which there is no decision deadline. This is for instance the case in predictive maintenance applications, where

[4]ML-EDM is a supervised problem, its extension to unsupervised problems is discussed in challenge #1, Section 4.

there is no urgency or even a deadline to detect the absence of failure. In this case, the delay cost $\mathcal{L}_{delay}$ and also the deadline $\mathcal{T}$ apply only to the other decisions (e.g. failures categorized by severity level) excluding the nominal state.

At the end, ML-EDM aims to develop approaches which allow for easy adaptation to all cases, whether the deadline $\mathcal{T}$ is applicable to all decisions, or whether there exists a nominal operation state which bypasses this deadline.

Question B - *How to learn a triggering strategy from data?*

In summary: this section shows that learning a triggering strategy follows the usual general principles of Machine Learning approach, with the particularity to consider *time-sensitive* loss functions (i.e. which depend on when decisions are triggered, as in Equations 1, 4 and 5).

In practice, the optimal triggering strategy is not available and it must be approximated by a learned function, such as $Trigger^{\gamma} \approx Trigger^{*}$, where $\gamma \in \Gamma$ is a set of parameters to be optimized within the space of parameters Γ of a chosen family of triggering strategies.
In addition, the hypothesis h is supposed to be learned previously during the training phase, making the system capable of predicting y at any time $t \in [1, T]$. This hypothesis is defined by a set of parameters $\theta \in \Theta$.
To illustrate what a triggering strategy is, let us consider an example from the ECTS literature. The SR approach, described in [57], involves 3 parameters $(\gamma_1, \gamma_2, \gamma_3)$ to decide if the current prediction $h(\mathbf{x}_t)$ must be chosen (output 1) or if it is preferable to wait for more data (output 0):

$$Trigger^{\gamma}\left(h(\mathbf{x}_t)\right) = \begin{cases} 0 & \text{if } \gamma_1 p_1 + \gamma_2 p_2 + \gamma_3 \frac{t}{T} \leq 0 \\ 1 & \text{otherwise} \end{cases} \quad (6)$$

where p_1 is the largest posterior probability estimated by h, p_2 is the difference between the two largest posterior probabilities, and the last term $\frac{t}{T}$ represents the proportion of the incoming time series that is visible at time t. The parameters $\gamma_1, \gamma_2, \gamma_3$ are real values in $[-1, 1]$ to be optimized, as described more generally in the following.

In the simple case of ECTS, a single decision has to be made for each time series $\mathbf{x} \in \mathbb{X}$ (see Equation 1). Thus, the *risk* associated with any triggering strategy $Trigger^{\gamma}$ belonging to any family Γ, is defined as follows, given the previously learned hypothesis h^{θ} within the family Θ:

$$R(Trigger^{\gamma}|h^{\theta}) = \mathop{\mathbb{E}}_{\mathbb{X},\mathbb{Y}}\left[\mathcal{L}(h^{\theta}(\mathbf{x}_{\hat{t}}), \hat{t}, y)\right] \quad (7)$$

where $\hat{t}$ is determined by γ, the parameters of the triggering strategy.

Similarly, the risk can be defined in the more complex case of ***multiple early decisions to be located in time***. Let T_{part} be the set of all possible partitions of the time domain $[1, T]$, having a varying number of time intervals k. The risk can be defined as:

$$R(Trigger^{\gamma}|h^{\theta}) = \mathop{\mathbb{E}}_{\mathbb{X},T_{part},\mathbb{Y}^k}\left[\boldsymbol{\mathcal{L}}(\{h^{\theta}(\mathbf{x}_{\hat{t}_{i'}}), (\hat{s}_{i'}, \hat{e}_{i'})\}, \{\hat{t}_{i'}\}, \{y_i, (s_i, e_i)\})\right] \quad (8)$$

where $\{\hat{t}_{i'}\}$ and $\{(\hat{s}_{i'}, \hat{e}_{i'})\}$ are determined by γ, and given h^{θ}. In Equation 8, the risk is an expectancy on three random variables, drawing triplets from the join distribution $P(\mathbf{x}, \{(s_i, e_i)\}, \{y_i\})$. The first element corresponds to the input data[5], which is an individual $\mathbf{x} \in \mathbb{X}$. The two other consist of the ground truth, which is composed of: (i) a partition of the time domain $\{(s_i, e_i)\} \in T_{part}$ with a particular number of time intervals, denoted by $k \in [1, T]$; (ii) and a set of class labels $\{y_i\} \in \mathbb{Y}^k$ for each time interval.
Now, the objective is to approximate the optimal triggering strategy $Trigger^{*}$ by finding $\gamma^{*} \in \Gamma$ which minimizes the risk, such that:

$$\gamma^{*} = \mathop{\arg\min}_{\gamma \in \Gamma} R(Trigger^{\gamma}|h^{\theta}) \quad (9)$$

The joint distribution $P(\mathbf{x}, \{(s_i, e_i)\}, \{y_i\})$ is unknown, thus Equation 8 can not be calculated ; however a training set $\mathcal{S}$ which samples this distribution is supposed to be available. The risk can be approximated by the *empirical risk* calculated on the training set $\mathcal{S} = \{\mathbf{x}^j, \{y_i^j, (s_i^j, e_i^j)\}\}_{j \in [1,n], i \in k_{\mathbf{x}_j}}$, as follows:

$$R_{emp}(Trigger^{\gamma}|h^{\theta}) = \frac{1}{n}\sum_{j=1}^{n} \boldsymbol{\mathcal{L}}\left(\{h(\mathbf{x}^j_{\hat{t}_{i'}}), (\hat{s}^j_{i'}, \hat{e}^j_{i'})\}, \{\hat{t}_{i'j}\}, \{y_i^j, (s_i^j, e_i^j)\}\right) \quad (10)$$

where $\hat{t}_{i'j}$ is the triggering time of the *i-th* made decision of the *j-th* individual.
At the end, training an ML-EDM approach can be viewed as a *two-step* Machine Learning problem: (i) *first*, the hypothesis h^{θ} must be learned in order to predict the most appropriate decision $h^{\theta}(\mathbf{x}_t)$, at any time $t \in [1, T]$; (ii) *second*, the best triggering strategy defined by γ^{*} must be learned, given the hypothesis h^{θ} and given the family Γ, such that:

$$\gamma^{*} = \mathop{\arg\min}_{\gamma \in \Gamma} R_{emp}(Trigger^{\gamma}|h^{\theta}) \quad (11)$$

Question C - *Can a triggering strategy be learned by Reinforcement Learning?*

In summary: this section shows that learning a triggering strategy of an ECTS approach can be cast as a Reinforcement Learning (RL) problem, with rewards well chosen, and it might be expected that provided with sufficient training, RL may end up with a good approximation of an efficient decision function.

[5]Notice that the notation $\mathbf{x} \in \mathbb{X}$ in Equations 7 and 8 is an abuse that we use use to simplify our purpose. In all mathematical rigor, the measurements observed successively constitute a family of time-indexed random variables $\mathbf{x} = (\mathbf{x}_t)_{t \in [1,T]}$. This stochastic process $\mathbf{x}$ is not generated as commonly by a distribution, but by a filtration $\mathbb{F} = (\mathcal{F}_t)_{t \in [1,T]}$ which is defined as a collection of nested σ-algebras [43] allowing to consider time dependencies. Therefore, the distribution $P(\mathbf{x}, \{(s_i, e_i)\}, \{y_i\})$ should also be rewritten as a filtration.

Reinforcement learning [75] aims at learning a function, called a policy π, from states to actions: $\pi : \mathcal{S} \rightarrow \mathcal{A}$. Rewards can be associated with transitions from states $s_t \in \mathcal{S}$ to states $s_{t+1} \in \mathcal{S}$ under an action $a \in \mathcal{A}$. Rewards are classically denoted $r(s_t, a, s_{t+1}) \in \mathbb{R}$. In all generality, the result of an action a in state s_t may be non deterministic and one among a set (or space) of states. The optimal policy $\pi^\star$ is the one that maximizes the expected gain from any state $s_t \in \mathcal{S}$. This gain, denoted R_t starting from the sate s_t, is defined as a function of the rewards from that state (e.g. a discounted sum of the rewards received). In order to learn a policy, value functions can be considered, such as the state-value function $v_\pi(s)$ classically defined as:

$$\begin{aligned} v_\pi(s_t) \;&\doteq\; \mathbb{E}_\pi[R_t \mid s_t] = \sum_{a \in \mathcal{A}} \pi(a|s_t) \sum_{s_{t+1}, r} \\ &p(s_{t+1}, r \mid s_t, a) \left[r(s_t, a, s_{t+1}) \;+\; \gamma \, v_\pi(s_{t+1}) \right] \end{aligned} \tag{12}$$

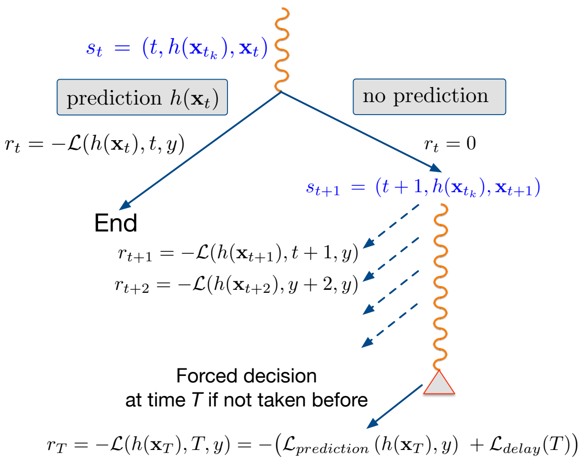
where $\mathbb{E}_\pi[\cdot]$ denotes the expected value of a random variable given that the agent follows the policy π and t is any time step. In the case of a non deterministic policy, $\pi(a|s_t)$ denotes the probability of choosing action a in state s_t and $p(s_{t+1}, r \mid s_t, a)$ the probability of reaching state s_{t+1} and receiving the reward r given that the action a has been chosen in state s_t. And γ is a discounting factor: $\gamma < 1$.
In our case, the agent aims to learn a triggering strategy given the previously learned classifier h^θ, and the state $s_t = (t, \mathbf{x}_t)$ is the current time t and the observed data at current time. The instantaneous reward $r(s_t, a)$ only depends on the current state s_t and the action taken a (i.e. prediction now, or postponed to a later time). Finally, the discounted factor γ, usually present in RL for reasons of convergence over infinite episodes, is equal to 1 in our case, since we always deal with finite episodes with forced decisions after a maximum delay. So that the equation (12) simplifies to:

$$v_\pi(s_t) \;=\; \sum_{a \in \mathcal{A}} \pi(a|s_t) \, r(s_t, a) \;+\; v_\pi(s_{t+1})$$

when, during learning, the agent takes a decision, it updates the value of the state s_t using:

$$v_\pi(s_t) \;=\; r(s_t, a) \;+\; v_\pi(s_{t+1})$$

where s_{t+1} is the state after having taken the action a in state s_t.
As the equation above shows, the core observation in RL is that the value function for a state s_t (i.e. an estimation of the expected gain from that state) is related to the value function of states s_{t+1} that may be reached from s_t. In that way, information gathered further down a followed path can be back-propagated to previous states thus allowing increasingly better decisions from those states to be made.
For instance, in game playing, rewards may happen both during play (e.g. the player just lost a pawn) and at the end of the game (e.g. the player is chess mate). Similarly, one could cast the ECTS problem as a RL problem where, at each time step, the "player" is in state $s_t = (t, h(\mathbf{x}_{t_k}), \mathbf{x}_t)$ and should choose between making a prediction (e.g. $h(\mathbf{x}_t)$) with an associated reward:
$r_t = -\mathcal{L}(h(\mathbf{x}_t), t, y) = -\mathcal{L}_{prediction}\,(h(\mathbf{x}_t), y) \;-\; \mathcal{L}_{delay}(t)$
or postpone the decision, with no immediate associated reward, that is $r_t = 0$. If no decision has been made before the term of the episode (e.g. when $t = T$) a decision is forced (see Figure 2). Provided with enough time series to train on,



$s_t \;=\; (t, h(\mathbf{x}_{t_k}), \mathbf{x}_t)$
prediction $h(\mathbf{x}_t)$
no prediction
$r_t = -\mathcal{L}(h(\mathbf{x}_t), t, y)$
$r_t = 0$
$s_{t+1} \;=\; (t+1, h(\mathbf{x}_{t_k}), \mathbf{x}_{t+1})$
End
$r_{t+1} = -\mathcal{L}(h(\mathbf{x}_{t+1}), t+1, y)$
$r_{t+2} = -\mathcal{L}(h(\mathbf{x}_{t+2}), y+2, y)$
Forced decision
at time T if not taken before
$r_T = -\mathcal{L}(h(\mathbf{x}_T), T, y) = -\big(\mathcal{L}_{prediction}\,(h(\mathbf{x}_T), y) \;+ \mathcal{L}_{delay}(T)\big)$

Figure 2: A part of an ECTS "game" when learning an optimal policy while "playing" a training time series. When a prediction is made, the game stops, otherwise it continues until a prediction is made or the term of the episode is reached.

and sufficient training in the form of "playing" these time series, a reinforcement learning agent may end up with a policy $\widehat{\pi}$ that approximates a good early triggering strategy, one that would converge over time, after a very large number of "plays" on the training time series, to the optimal decision function $\pi^\star$ (See Equations 2 and 11).

The RL framework is very general. It uses immediate and delayed rewards. As shown in this section, there is in principle no obstacle to apply RL to the learning of a good triggering strategy. However, if used directly, the generality of RL is paid for by a need for a large number of "experiments". In addition, the state space is continuous in the case of the ECTS problem, thus an interpolating functions must be used in order to represent the values such as $v_\pi(s)$ and this entails the choice of a family of functions and setting their associated parameters.
Another approach, the one favored in the current literature for ECTS [2], is to choose functions for representing the expected values of decision times, and thus providing a ground for the triggering strategy. This has the merit of incorporating prior knowledge of the trade-off between earliness and accuracy, at the cost of making modelling choices that may bias the method of estimating the expected future cost.
The respective performances, merits and limits of both approaches should be studied empirically by a comparison of RL based ECTS approaches, such as [53], with approaches that explicitly exploit the form of the optimization criterion designed for ECTS as in [2].

3. ORIGIN OF THE DELAY COST

ML-EDM approaches aim to trigger decisions at the right time, by reaching a good trade-off between the *earliness* and the *accuracy* of their decisions. To achieve this, a balance must be found between penalizing late decisions and penalizing prediction errors. *Decision costs* are key to make this antagonistic trade-off choice, as they allow us to evaluate the cost of waiting for new measures vs. the cost of making a decision now. In Section 2, decision costs are involved starting from Equation 2 in the loss function $\mathcal{L}$ and they have an important impact on the entire path of the description of the ML-EDM problem. The objective of this section is to understand the deep origin of the delay cost.

The *delay cost* represents the cost of postponing a decision (see the function $\mathcal{L}_{delay}$ in Equation 1). In the particular case of ECTS problems, the delay cost is present in all the works described in scientific literature. But it can be explicitly defined as in [2,56], or implicitly as in most approaches. For instance, the authors in [83] trigger all the decisions at the *minimum prediction length*, which correspond to the early moment such that no prediction differs from those applied to the full-length training time series (based on a KNN classifier). This approach thus *implicitly* assumes that the delay cost is very low, by favoring the accuracy of decisions at the expense of their earliness. In [59], the authors propose to model the trade-off between earliness and accuracy as a multi-objective criterion and explore the Pareto front of multiple dominant solutions. This approach is useful in applications where earliness and accuracy can not be evaluated in a commensurable way, and it provides a collection of optimal solutions each corresponding to a particular value of the delay cost.

For a better understanding, let us examine what happens once a decision is triggered in the simple ECTS problem. Figure 3 represents a *classifier* and a *triggering strategy*. At each time step $t \in [0, T]$, the classifier predicts the conditional distribution $P(y|\mathbf{x}_t)$ based on the input incomplete time series $\mathbf{x}_t = \langle x_0, x_1, \ldots, x_t \rangle$. Then, the triggering strategy either decides to *postpone* the decision until a new measurement x_{t+1} is available, or to *trigger* the decision by predicting the class value. In this first scenario, let us consider that triggering a decision at time t implies performing a given *task* (namely α or β) which depends on the predicted class (respectively A or B).

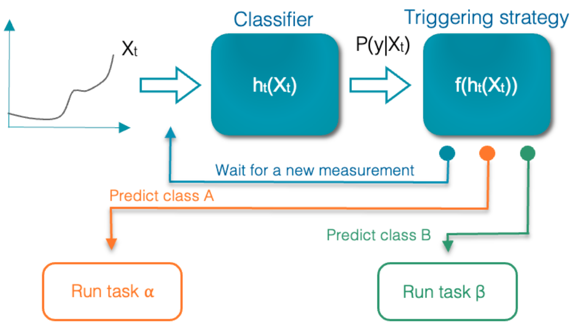


Figure 3: Tasks to be performed after the triggering of a decision.

Given that this task (α or β) must be completed *before* the deadline T, the problem is to determine how the cost of performing this task evolves depending on the trigger time t. In practice, the delay cost $\mathcal{L}_{delay}$ takes the form of a parametric function (e.g., a constant [83], linear [2] or exponential [9] function), whose form characterizes the additional cost to delay the execution of the tasks.

A *constant cost*, one where there is no penalty associated with delaying the decision, would mean that these tasks are achievable in an arbitrarily short time $T - t < \epsilon$. In practice, an irreducible amount of time is needed to perform the tasks using a single worker. To reduce this time, the tasks need to be parallelized using several workers, incurring an extra-cost when building the global result from sub-tasks. Formally, a constant delay cost would mean that the tasks are *infinitely parallelizable*, i.e. they can be divided into *independent* and *arbitrarily small* sub-tasks, and that there is no extra-cost in building the global result.

More generally in ML-EDM problems, the delay cost $\mathcal{L}_{delay}$ is necessarily an increasing function (monotonic or piecewise) depending on the time remaining before the decision deadline, and it may depend on the decision made (i.e. the predicted label). In addition, it should tend to $+\infty$ when the time remaining to perform these tasks $\mathcal{T} - t$ tends to zero [9]. For example, this delay cost may be modeled by $\mathcal{L}_{delay}(t) = 1/(\mathcal{T} - t)^{\alpha}$, with a single parameter α which influences the increase in cost when $(\mathcal{T} - t) \to 0$.

4. LEARNING TASKS

As in ECTS, the formal definition of ML-EDM provided in Section 2 is limited to classification problems, and involves ground truth. However, in many applications, it is extremely hard or costly to obtain, especially in the case of anomaly detection (e.g. fraud, cyber-attacks, predictive maintenance). In these application domains, there are several issues: (i) labels can be extremely expensive to obtain as they each require an examination from an expert ; (ii) the labels provided by experts can be uncertain ; and (iii) the class of anomalous observations is often poorly represented and drifts over time. For example, cyber-attack techniques are very diverse and change with time. Faced with these difficulties, anomaly detection is often addressed using unsupervised approaches, by assuming that the anomalies are outliers. In this case, the problem comes down to modeling the normal behavior of the system, if possible using historical data that are cleaned of anomalies. Then, it is necessary to define the notion of outlier to be able to assign an eccentricity score to the new observations. Note that this type of modeling can be considered as a *first step* to manage non-stationarity, since in this case the stationarity assumption only concerns the normal behavior of the system (this assumption could be removed in future work).

Challenge #1:
extending non-myopia to unsupervised approaches
A variety of unsupervised early decision problems could be studied, of which two examples are listed below: (i) the problem could be to decide, as soon as possible, whether a partially observed time series $\langle x_1, x_2, \ldots, x_t \rangle$ will be an outlier (or not) when fully observed at time T (i.e. with a single decision triggered for each incoming time series, as in the ECTS problem described in Section 2) ; (ii) an other unsupervised problem could be to detect, online and as early as possible, the chunks of the input data stream which deviate from the nominal learned behavior (i.e. with multiple early decisions to be located in time, as in the ML-EDM problem). In both cases, the *accuracy vs. earliness* trade-off still exists. On the one hand, an early detection is inaccurate by nature because the outlier series (*resp.* chunk) is unreliably detected, based on few (*resp.* poorly informative) observed measurements. On the other hand, delaying the detection of

anomalies can be very costly. For instance, a cyber-attack which is not detected immediately gives time to the hakers to exploit the security hole found. Designing ML-EDM approaches to tackle *unsupervised* learning tasks is challenging in several respects: (*i*) learning a triggering strategy with the goal of achieving a good trade-off between earliness and accuracy of its decisions cannot be achieved in the Machine Learning framework as described in the section 2 and should be formalized in another way without labels (in particular, the models evaluation described in Equation 5 should be reconsidered); (*ii*) developing unsupervised *non-myopic* approaches is very difficult, as the training set does not contain anomalous series, thus the triggering strategy cannot learn from their continuations.

The extension of ML-EDM both to *online* scenarios (see Section 6) and to *unsupervised* tasks is of particular interest, because combined they would enable a new generation of *monitoring systems* [1] to be developed. In this case, the learning task would consist in detecting online the start and end of the outlier chunks: (*i*) without requiring labels to learn the model ; (*ii*) by considering the trade-off between *accuracy* and *earliness* to trigger the decisions at the right time.

Challenge #2:
addressing other supervised learning tasks
The formal description of ML-EDM proposed in Section 2 is generic, in the sense that the type of the target variable y can easily be changed. By definition, the ECTS approaches in the literature are limited to *classification* problems, but they could naturally be extended to other supervised learning tasks. For instance, predicting a *numerical* target variable from a time series is a problem known as *Time Series Extrinsic Regression* (TSER) [76]. In some domains, TSER approaches are very useful and allow applications such as the prediction of the daily energy consumption of a house, based on the last week's consumption, temperature and humidity measurements. *Early* TSER would consist of predicting the value of the numerical target variable as soon as possible, while ensuring proper reliability. Another example of a supervised task for which ML-EDM approaches could be developed is time series *forecasting* [16]. Basically, a forecasting model aims to predict the next measurements of a time series up to an horizon ν, $Y = \langle x_{t+1}, x_{t+2}, \ldots, x_{t+\nu} \rangle$ from the recent past measurements $X = \langle x_{t-w}, \ldots, x_{t-1}, x_t \rangle$. Using a forecasting model, in a an online and *early* way, would consist of adapting the forecast horizon $t + \nu$ according to the observed values in X, by modeling the trade-off between the *accuracy* and the *earliness* of these predicted values.

The ML-EDM problem described in Section 2 should also be adapted to *semi-supervised* learning, which is of great help when the ground truth is only partially available. More generally, the collected ground truth may be imperfect for various practical reasons, such as the labeling cost, the availability of experts, the difficulty of defining each label with certainty, etc. This problem has recently gained attention in the literature through the field of *Weakly Supervised Learning* (WSL) [86] which aims to list these problems and provide solutions.

Challenge #3: early weakly-supervised learning
The extension of ML-EDM to weakly-supervised learning is an interesting challenge, as it would allow to better address applications where the ground truth has corruptions or is incomplete (which includes semi-supervised learning). However, the weakly-supervised learning is a very large domain with many types of supervision deficiencies to be studied. From a practical point of view, the priority is probably to extend ML-EDM to label noise, and more specifically to bi-quality learning [60], where the model is trained from two training sets: (*i*) one trusted with few labels ; (*ii*) the other, untrusted, with a large number of potentially corrupted labels. This would allow interesting applications, such as in cyber security where few labels are investigated by an expert, and the majority of labels are provided by rule-based systems. The major difficulty in designing *bi-quality learning* ML-EDM approaches is to learn a triggering strategy from these two training sets, which models the compromise between *accuracy* and *earliness* in a robust way to label noise. Another interesting avenue would be to adapt *Active Learning* [71] approaches to ML-EDM, with the goal of labeling examples which improve both *accuracy* and *earliness* of the decisions. Such approaches would be particularly helpful when early decisions have to be made, and when labeling examples is very costly as, again, it is the case in cyber security applications.

5. TYPES OF DATA

The ML-EDM definition proposed in Section 2 involves measurements (i.e. scalar values) acquired over time. However, this is only for reasons of simplicity of exposition. Ideally, ML-EDM approaches should be *data type agnostic*, i.e. they should operate for any data type as long as measurements are made over time and decisions are online.
Below, we outline data types that are present in applications where ML-EDM could be used.

i) *Multivariate time series* consist of successive measurements each containing more than one numerical value.

ii) More *complex signals* exist, such as video streams which involve higher dimension.

iii) *Data streams* is another type of data which can contain both numeric and categorical variables [10]. Successive measurements are received in an uncontrolled order and speed.

iv) Another type of data is *evolving graphs* which consist of graphs whose structure changes over time [46]. Several types of learning tasks can be considered, such as predicting the next changes in the graph structure, or the classification of parts of the graph (e.g. nodes, arcs, sub-graphs).

v) Successive snapshots of *relational data* [19] should be consider to design new ML-EDM approaches. More precisely, relational data consists of a collection of tables having logical connections between them. Like other types, relational data can evolve over time: (*i*) the connections between tables can change ; (*ii*) as well as the structure of the tables ; (*iii*) or even the values of the information stored in the tables.

vi) *Text* is another widespread type of data. An application example is the moderation of social networking platforms, with early deletion of inappropriate contents and automatic closure of fraudulent accounts (see Section 10.3).

Challenge #4: data type agnostic ML-EDM
Ideally, the new developed ML-EDM approaches should be *data type agnostic*, i.e. they should operate for any data type presented above. To do so, a pivotal format needs to be defined in order to learn the triggering strategies in a generic way. For instance, each learning example could be characterized by a series of T predictions indexed by time (corresponding to the output of the learned hypothesis $h(\mathbf{x}_t)$ for each time step $t \in [1, T]$), as well as by $\{y_i, (s_i, e_i)\}_{i=1}^{k_\mathbf{x}}$ the ground truth composed of the true decisions to be made over time for this individual. In the particular case of ECTS, some approaches can easily be adapted to become agnostic to data type [2, 58, 59]. In contrast, others have been designed to be very specific to time series [31, 37, 83, 84], especially with the search of features (e.g. shapelets) occurring early in the time series and helping to discriminate between classes. More generally, future work in ML-EDM should definitely promote data type agnostic approaches, to allow the use of these techniques in a wide range of application conditions.

6. ONLINE EARLY DECISION MAKING

In the specific case of Early Classification of Time Series (ECTS), an important *limitation* is that the training time series: (i) have the same length T ; (ii) correspond to different *i.i.d* individuals ; (iii) have a label which characterizes the whole time period of length T. There are obviously applications where this formulation of the problem is relevant [7, 17, 23, 34, 50, 67, 72, 78], especially in cases where the *start* and *end* of the time series are naturally defined (e.g. a day of trading takes place from 9:30am to 4pm, during the opening hours of the stock exchange).
The development of *online* ML-EDM approaches could overcome these limitations and enable a new range of applications. For this purpose, let us consider that the input measurements are observed without interruption, in the form of a *data stream* [29]. In the case of a *classification problem*, an online ML-EDM approach would consist in identifying *chunks* in the input data stream (i.e. fixed time-windows defined by their start and end timestamps) and *categorizing* them according to a predefined set of classes. For example, in a predictive maintenance scenario [64] such an approach would operate on a continuous basis to detect periods of system malfunction as soon as possible.

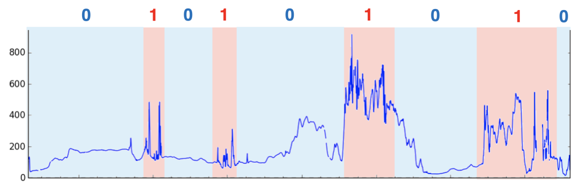


Figure 4: Example of a data stream labeled by chunks over a time period

Challenge #5:
online and early predictions to be located in time
In the case of a *classification* problem, the training data consist of the measurements observed from the stream during the training period, denoted by $\mathbf{x} = \langle x_1, x_2, \ldots, x_{|\mathbf{x}|}\rangle$, associated with their labels $\mathbf{y} = \langle y_1, y_2, \ldots, y_{|\mathbf{x}|}\rangle$. A *labeled chunk* is formed by the consecutive measurements, between the timestamps t_a and t_b, if their labels share the same value (i.e. if $\{y_i\}_{i \in [t_a, t_b]}$ is a singleton). As shown in Figure 4, the data stream defined over the training period is labeled by chunks of variable size. For example, these chunks could represent the periods of failure and nominal operation in a predictive maintenance scenario. During the deployment phase, the model is applied online on a data stream whose measurements are observed progressively over time. This model is expected to provide *predictions located in time*, since it needs to predict the *beginning* and the *end* of each chunk, associated with the predicted *class* which characterizes the state of the system during this chunk.

Challenge #6:
online accuracy vs. earliness trade-off
Designing *online* ML-EDM approaches requires redefining the accuracy vs. earliness trade-off for online decisions. The main issue is that a data stream is of indeterminate length: (i) its *beginning* may be too old to be considered explicitly, or can even be indeterminate ; (ii) its *end* is never reached, since it is constantly postponed by the new measurements which arrive. In the particular case of ECTS, it is precisely the fact that the input series has a maximum length T, known in advance, that leads to force triggering the decision when the current time t becomes close to the *deadline* T.

The rest of this paragraph presents an example of adapting the *accuracy vs. earliness* trade-off to online decisions developed in [4]. Let us consider a predictive maintenance problem for which a classifier has been trained in batch in order to detect the beginning and the end of abnormal chunks (see Figure 5). The prediction of the classifier focuses on a fixed timestamp s and the question is to determine if this timestamp corresponds (or not) to the beginning of an abnormal section. The input features used by the classifier are extracted from a sliding window $\mathbf{x}_t = \langle x_{t-w}, \ldots, x_{t-1}, x_t\rangle$ of length w. As shown in Figure 5, the sliding window $\mathbf{x}_t$ moves over time as it gets closer to s. At first, the timestamp s is located in the future ($s > t$). Making a good prediction is difficult since the potentially anomalous part of the stream is not yet visible in $\mathbf{x}_t$. In this case, the classifier have to detect the early signs of an anomaly. Then, the timestamp s enters the $\mathbf{x}_t$ window (at time $t = 4$). The prediction becomes easier to perform, since a part of the potentially abnormal chunk is visible in $\mathbf{x}_t$. The last possible moment to trigger the decision is reached when the timestamp s is getting ready to exit the sliding window $\mathbf{x}_t$.

Finally, the accuracy vs. earliness trade-off occurs as follows: (i) on the one hand, the accuracy of the decisions increases over time due to the classification task that becomes easier as the $\mathbf{x}_t$ window shifts ; (ii) on the other hand, predictive maintenance applications require early decisions which allow to anticipate breakdowns, or at least to detect it early. Ultimately, this proposal consists of changing the definition of *what is predicted* as normal or abnormal.

Here, the observation to be scored is no longer a time series of finite length, but a particular measurement of the input data stream identified by its timestamp. This proposal only partially addresses the problem, as the predictions for each timestamp would have to be consolidated in order to predict the start and end of each chunk. There are certainly other ways to adapt the accuracy vs. earliness trade-off to online decisions that would be valuable to investigate.

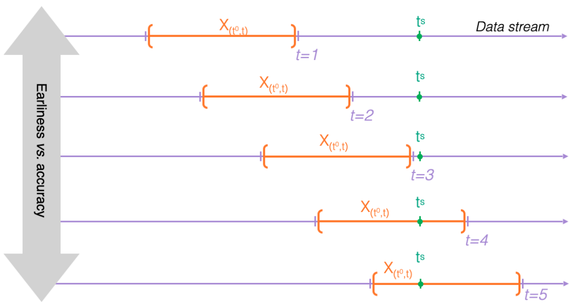


Figure 5: Illustration of the earliness vs. accuracy trade-off for online decisions

Challenge #7:
management of non-stationarity in ML-EDM

It is not always realistic to assume stationarity of the data. In practice, data collected from a stream may suffer from several types of drifts: (i) the distribution of the measurements within the sliding window $\mathbf{x}_t$ can vary over time, this is called *covariate-shift* [63]; (ii) the prior distribution of the classes $P(y)$ can be subject to such drifts; (iii) and the concept to be learned $P(y|\mathbf{x})$ can also change when *concept drift* occurs [28].

To manage these non-stationarities, a first family of approaches maintains a decision model trained using a sliding window of most recent examples. This is a blind approach, in the sense that there is no explicit drift detection. The main problem is deciding the appropriate window size.

A second family of approaches, explicitly *detects* the drifts [30, 48] and triggers actions when necessary, such as re-training the model from scratch, or using a collection of models in the case of ensembles. In this case, detecting concept drift can be considered as similar to the anomaly detection problem, and ML-EDM approaches could be used to tackle it in future work. A popular idea is to train the decision model using a growing window while data is stationary, and shrink the window when a drift is detected. These kinds of approaches can easily be adapted to *online* ML-EDM, since they decouple model training and non-stationarity management.

In the case of incremental concept drift, a third family of approaches consists in continuously adapting the model by training it online from recent data. This kind of *adaptive* approach is much more challenging to adapt to *online* ML-EDM. Indeed, as in ML-EDM problems (see Figure 3), two kinds of models are used: (i) the *predictive model(s)*, which can categorize the input data stream at any time ; (ii) the *triggering strategy* which makes the decisions at the appropriate time. The main challenge in developing adaptive drift management methods for the *online* ML-EDM problem is that the parameters of the *predictive models* and of the *triggering strategy* must be updated jointly. These two kinds of models are highly dependent: updating the parameters of one has an impact on the optimal parameters of the other.

By contrast, in standard ML-EDM approaches which operate in batch mode, the parameters of the predictive models are first optimized, and then the parameters of the triggering strategy are optimized in turn given the parameters of the classifiers (see paragraph B in Section 2). This two-step Machine Learning scheme is definitely not valid for managing drift online [45]. Adaptive drift management for the *online* ML-EDM problem has not yet been addressed in the literature and constitutes an interesting research direction. In drift detection systems, there is a trade-off between fast detection and the number of false alarms. Moreover, in problems where the target (e.g. the labels) is not always available or available with a delay requires unsupervised or semi-supervised drift detection mechanisms. The ML-EDM framework, improving the compromise between earliness and accuracy, can provide new approaches for drift detection.

7. REVOCABLE DECISIONS

In many situations, one can take a decision and then decide to change it after some new pieces of information become available. The change may be burdensome but nevertheless justified because it seems likely to lead to a much better outcome. This can be the case when a doctor revises what now seems a misdiagnosis.

Similarly, ML-EDM should be *extended* to consider such a revocation mechanism. In the classical ML-EDM problem as described in Section 2, a prediction $h(\mathbf{x}_{\hat{t}})$ cannot be changed once the decision is triggered at time $\hat{t} \leq T$. The cost of such an *irrevocable* decision is given by the loss function described by Equation 5. Whereas, the extension of ML-EDM to *revocable* decisions [3] allows a prediction to be modified several times before the end T of the considered time period. On the one hand, the revocation of a decision generates a higher delay cost $\mathcal{L}_{delay}$, as well as a cost of changing the decision $\mathcal{L}_{revoke}$. On the other hand, new data observed in the meantime provide information that makes the prediction more reliable, thus tending to decrease the misclassification cost $\mathcal{L}_{prediction}$. Ultimately, the main issue is to identify the appropriate decisions to revoke, in order to minimize the global cost, given by Equation 13.

Such an extension to revocable decisions could be of great interest: (i) in applications where the cost of changing decisions is low, i.e. the DAGs associated with each possible decision share reusable tasks (see Section 3) ; (ii) in applications involving online early decision making (see Section 6). There are many use cases where the need to *revoke* decisions appears clearly. For instance, the emergency stop system of an autonomous car brakes as soon as an obstacle is suspected on the highway, and releases the brake when it realizes, as it gets closer, that the suspected obstacle is a false positive (e.g. a dark spot on the road).

Developing ML-EDM approaches capable of appropriately revoking its decisions involves solving the two following challenges:

Challenge #8: reactivity *vs.* stability dilemma for revocable decisions
The first issue is to ensure that a decision change is driven by the information provided by the recently acquired measurements, and not caused by the inability of the system to produce a stable decision over time. This problem is not trivial. On the one hand, the system needs to be reactive by changing its decision promptly when necessary. On the other hand, the system is required to provide stable decisions over time by avoiding excessively frequent and undue changes. Thus, a trade-off exists between the *reactivity* of the system and its *stability* over time. One way to formalize this trade-off is to associate a cost to *decision changes*, as it is proposed in part (iii) of Equation 13. To our knowledge, only one approach uses such a cost of decision change [3], in order to penalize revocation of too many decisions. The *reactivity vs. stability* dilemma of revocable decisions is understudied in the literature, and it would be interesting for the scientific community to work on this question.

**Challenge #9:
extending non-myopia to revocation risk**
Non-myopic ML-EDM approaches are capable of estimating the information gain that will be provided by future measurements, based on the currently visible ones. In other words, these approaches are able to predict the reliability improvement of a decision in the future. Thus, a decision is triggered when the expected gain in miss-classification cost at the next time steps does not compensate the cost of delaying the decision [2]. In the case of revocable decisions, an important challenge is to estimate the future information gain by taking into account the *risk of revocation* itself. Specifically, a decision that will probably be revoked afterward should be delayed due to this risk. Conversely, a decision which promises to be sustainable should be anticipated. Designing *non-myopic to revocation risk* approaches could be an important step forward to (i) optimize the first trigger moment, and (ii) reduce the number of undue decision changes. The approach proposed in [3] constitutes a first step in this direction, by assigning a cost to decision changes and considering it in the expectation of future costs. To the best of our knowledge, this is the only approach which provides this interesting property. It is not clear whether alternative methods are possible. This is an interesting topic for further studies by the scientific community.

8. ORIGIN OF THE DECISION COSTS

The origin of the delay cost has been studied in Section 3, however it is necessary to further specify the operating scenario in order to understand the other decision costs involved in ML-EDM. Figure 6 describes a binary ECTS problem, where the actions to be performed depend on the predicted class and are described by two *Directed Acyclic Graphs* (DAG). These DAGs characterize the sequence and the relationships between the unit tasks which compose them (e.g. task 1 must be completed before starting task 2). Here, the DAGs of tasks are *fixed*, they do not depend on the decision time.
The total cost of a decision can be decomposed by:

(i) the *delay* cost, denoted by $\mathcal{L}_{delay}$, which reflects the need to execute the DAG of actions corresponding to the new decision in a constrained time, and in a parallel way (already detailed in Section 3);

(ii) the *decision* cost, which corresponds to the consequences of a bad decision, or the gains of a good decision (denoted by $\mathcal{L}_{prediction}$).

(iii) the *revocation* cost, which is the cumulative cost of the mistakenly performed tasks belonging to the DAG of previously made bad decisions, and which are not reusable for the new decision (denoted by $\mathcal{L}_{revoke}$) ;

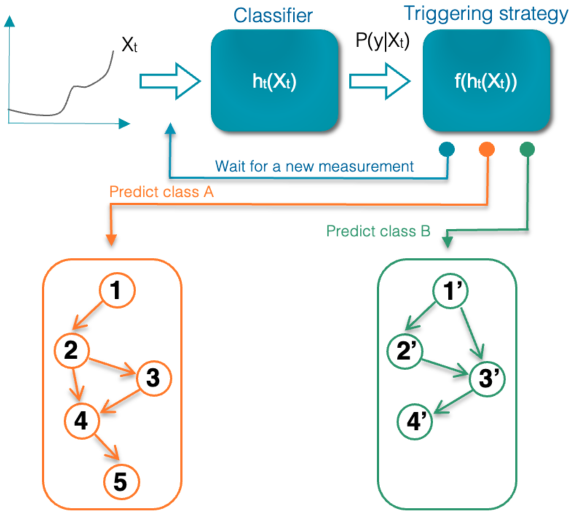


Figure 6: DAGs of tasks to be performed after the triggering of a decision.

When expressed in the same unit, these different types of costs can be summed up in order to reflect the quality of the decisions made, and their timing. Thus, Equation 1 becomes:

$$\begin{aligned}\mathcal{L}(h(\mathbf{x}_t), t, y) = \overbrace{\mathcal{L}_{delay}(t)}^{(i)} + \overbrace{\mathcal{L}_{prediction}\left(h(\mathbf{x}_t), y\right)}^{(ii)} \\ + \underbrace{\mathcal{L}_{revoke}\left(h(\mathbf{x}_t) | \{(h(\mathbf{x}_{\hat{t_i}}), \hat{t_i})\}_{i \in [1, D_t^{\mathbf{x}}]}\right)}_{(iii)}\end{aligned} \quad (13)$$

where $\{(h(\mathbf{x}_{\hat{t_i}}), \hat{t_i})\}_{i \in [1, D_t^{\mathbf{x}}]}$ represents the sequence of the previously made decisions and their associated triggering time, with $\hat{t_i} < t, \forall i \in [1, D_t^{\mathbf{x}}]$.

Term (ii): Taking into account the *decision* cost is a very common feature in the literature, particularly in the field of cost-sensitve learning [20]. These techniques take as input a function $\mathcal{L}_{prediction}(\hat{y}|y) : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ which defines the cost of predicting $\hat{y}$ when the true class is y. The aim is to learn a classifier which minimizes these costs on new data.

Term (iii): By contrast, the study of the *revocation* cost is very limited in the literature. To our knowledge, [3] is the only one article article that considers this problem, and this work shows that assigning a cost to decision changes is a first lead to manage the *reactivity vs. stability dilemma*, and to design *non-myopic to revocation risk* approaches (i.e. discussed later in challenges #8 and #9). The origin of this cost can be explained in the light of the tasks to be performed once a decision is triggered (see Figure 6). For instance, let us consider the first decision noted by $(A, \hat{t_1})$, in which the system predicts at time $\hat{t_1}$ that the input time series belongs to the class A. This decision is then revoked in favor of a new decision $(B, \hat{t_2})$. The cost of changing this decision, denoted by $\mathcal{L}_{revoke}((B, \hat{t_2})|(A, \hat{t_1}))$, can be defined as the cost of the actions already performed between $\hat{t_1}$ and $\hat{t_2}$ which turn out to be useless for the new decision, i.e. which cannot be reused in the DAG of tasks corresponding to the new predicted class B. In order to define the costs of decision changes, it is necessary to identify the *common tasks* between the DAGs of the different classes and to evaluate their execution time. In addition, the entire sequence of the past decisions must be taken into account to identify the already completed tasks which are now useful for the achievement of the current DAG of tasks. For instance, the cost $\mathcal{L}_{revoke}((A, \hat{t_3})|\{(A, \hat{t_1}), (B, \hat{t_2})\})$ can be reduced by the tasks executed between $\hat{t_1}$ and $\hat{t_2}$, if these tasks are *not perishable*, i.e. the results are identical to those that would be obtained by re-executing these tasks at $\hat{t_3}$.

Challenge #10:
scheduling strategy and time-dependent costs
In this paper, the DAGs of tasks are supposed to be *fixed*, i.e. not depending on the decision time. However, a more general problem could be considered (see Figure 7) where the DAGs of tasks are generated by a *scheduling strategy* depending on: (i) the decision made ; (ii) and the decision time. Such a scheduling strategy is useful in applications where the actions to be performed after a decision can be *adapted* to a time budget available to perform them. Two situations may occur: (i) ideally, a decision is triggered early enough to allow the scheduling strategy to generate a *complete* DAG of tasks which is optimal given the decision made (as in Figure 6) ; (ii) on the contrary, in the case of a too late decision, the scheduling strategy needs to build the DAG so that it can be achieved in the remaining time (e.g. by parallelizing some tasks, by changing or removing some of them). For instance, when flying an airplane, the tasks to be performed for an emergency landing are not the same as for a normal landing, and there is a range of situations with different emergency level, and therefore corresponding to different time budgets.
Such a time-dependent scheduling strategy radically transforms the ML-EDM problem and the way it can be formulated. In particular, the *triggering* and *scheduling* strategies become mutually dependent:

1. *Decision costs depend on the generated DAG of tasks:* all the previously discussed costs result from the structure of the DAG to be performed conditionally to the decision made: (i) the relationships between the tasks ; (ii) their execution time ; (iii) the conditions of their reuse when they are common to several DAGs. Since the structure of the DAG to be performed now depends on the decision time, the decision costs can no longer be considered as fixed, and they are available only after scheduling.

2. *The optimal decision time depends on the cost values:* on the other hand, the triggering strategy aims to optimize the decision time based on the cost values. As described in Equation 11, the triggering strategy is learned by minimizing the empirical risk, which is itself estimated using a loss function based on the costs.

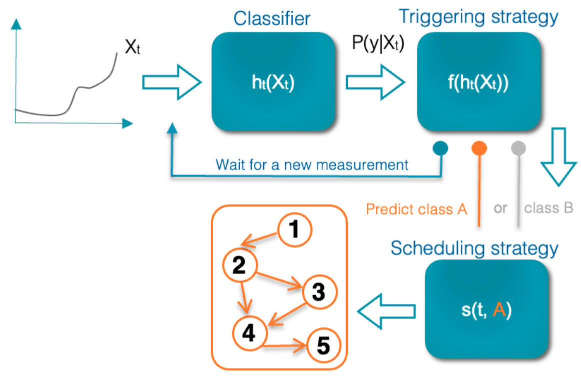


Figure 7: DAG of tasks to be performed after the triggering of a decision, generated by a scheduling strategy.

This mutual dependency between the triggering and the scheduling strategies has strong impacts on the ML-EDM problem. In particular, the optimal decision time t^* described in Equation 2 must be redefined as a *fixed point*, i.e. the function to be optimized takes the optimal solution as an input parameter, in such a way that $t^* = \arg\min_{t^* \in [1,T]} \mathcal{L}(h(\mathbf{x}_{t^*}), t^*, y)$. This leads to a much more difficult class of optimization problems, for which the simple existence of a solution is difficult to ensure.
Finding an optimal triggering strategy when the scheduling strategy is itself time-dependent makes ML-EDM a quite difficult challenge as the scheduling strategy is only known through its interactions with the triggering strategy. In this case, Reinforcement Learning seems to be a possible option to address the problem. The scheduling strategy could then be considered as part of the environment, and a contributor to the reward signal by determining the decision costs for each decision taken at a particular time. However, this line of attack remains to be investigated in order to assess its merit.
In many applications, fortunately, the implementation of a scheduling strategy is much simpler, especially when the variation of decision costs over time is known in advance (or modeled, and thus are partially known). The preceding remarks are reminders that if considered in all its complexity, ML-EDM becomes a very difficult problem. Addressing the case where the costs are assumed to be time dependent but with a known form, already offers interesting challenges and corresponds to a variety of applications.

9. OVERVIEW ON CHALLENGES

This section provides an overview of the previously presented challenges, indicating references which address part of these challenges (see the second column of Table 1), and summarizing the main prospects for applications in the short and long term (see the last column of Table 1). Table 1 organizes the proposed challenges by category, using colors to identify: (i) those related to changing the learning task ; (ii) those related to online ML-EDM ; (iii) and those related to revocable decisions.

ML-EDM challenges	SOTA	Main application perspectives
#1 (Section 4) **Extending non-myopia to unsupervised approaches**		In anomaly detection applications, anticipate the deviation of an observed individual from a normal behavior.
#2 (Section 4) **Addressing other supervised learning tasks**		Adapt ECTS approaches to extrinsic regression problems. Develop forecasting methods whose prediction horizon can adapt.
#3 (Section 4) **Early weakly supervised learning (WSL)**		Adapt ECTS approaches to the different WSL classification scenarios.
#4 (Section 5) **Data type agnostic ML-EDM**	[2, 18, 56, 57]	Identify agnostic approaches in the literature and promote this feature. Define a pivotal format allowing to develop an ML-EDM library.
#5 (Section 6) **Online predictions to be located in time**		Applications where the arrival of an event (e.g. a failure) must be predicted in advance, as well as its duration.
#6 (Section 6) **Online accuracy vs. earliness trade-off**	[4]	Optimize decision time in online predictive maintenance applications.
#7 (Section 6) **Management of non-stationarity in ML-EDM**		Properly manage the potentially long life of ML-EDM models.
#8 (Section 7) **Reactivity vs. stability dilemma for revocable decisions**	[3]	Applications where undue and excessive decision changes must be avoided.
#9 (Section 7) **Non-myopia to revocation risk**	[3]	Applications where it is necessary to delay decisions which are likely to be changed later.
#10 (Section 8) **scheduling strategy and time-dependent decision costs**		Applications where the variation of the decision costs over time is known or can be modeled. Applications where the scheduling strategy is only known through its interactions with the triggering strategy.

Table 1: Overview of the proposed challenges by category: in **blue** those related to the *learning task*, in **green** those related to *online* ML-EDM, in **purple** those related to *revoking decisions*, and in white the others.

10. USECASES

ML-EDM approaches can be applied to a wide range of applications, such as cyber security [87], medicine [41], surgery [69]. This section develops some key use cases and identifies possible advances in near future, if the proposed challenges are met.

10.1 Early classification of fetal heart rates

There are no precise figures on the number of deaths in childbirth due to poor oxygenation. According to the Portuguese Directorate-General for Health, the number of children who died due to hypoxia in 2013 was 192 fetuses. This is a critical example where making informed early decisions is critical.
Cardiotocography techniques are used to assess fetal well-being through continuous monitoring of fetal heart rate and uterine contractions [51]. Labor is a potentially threatening situation to fetal well-being, as strong uterine contractions stop the flow of maternal blood to the placenta, compromising fetal oxygenation [61].
In this field, ML-EDM techniques could be of great help to detect the early warning signs of complications during childbirth. This application can be addressed as an ECTS problem, as a fetal heart rate signal constitutes a time series. The extension of ECTS techniques to revocable decisions would be very relevant (see challenges #8 and #9) allowing for active monitoring of the children's well-being on a continuous basis, until delivery. In addition, two particular aspects need to be taken into account in developing an efficient approach: (i) the prediction cost $\mathcal{L}_{prediction}$ is highly asymmetrical since a false negative can mean the death of the baby or the mother; (ii) the deadline T which represents the moment of delivery is uncertain and varying. Thus, the deadline T corresponds to the occurrence of an event (i.e. the birth) which can be modeled as random variable as in [26, 44].

10.2 Digital twin in production systems

Digital Twin (DT) is an important active concept in the area of Industry 4.0. With the development of low cost sensors and efficient IoT communication facilities, almost all production systems are now equipped with several sensors enabling real time monitoring and helping in decisions about maintenance, or when failures occur. In this section, we consider digital twins (DT) of cyber-physical systems (CBS) which are in operation.

The main digital twin applications [27] are related to smart cities, manufacturing, healthcare and industry. The role of the DT is thus to use the data streams coming from the sensors of the CBS in order to constantly calibrate simulation models of different components of the system. Indeed, this offers several opportunities, namely (1) detection of anomalies when the system deviates from the simulation model ; (2) diagnostic of dysfunctions when they occur ; (3) exploration of different scenarios for system evolution in case of dysfunction ; (4) recommendation for repair actions.

Effective maintenance management methods are vital, and industries seek to minimize the number of operational failures. The availability of large volume of data coming from sensors of a CBS makes the use of Machine Learning techniques, supervised or unsupervised, very appealing. Typical unsupervised ML approaches are related to anomaly detection [66] where an alarm should be triggered when the behavior of the CBS differs from normal running. Typical supervised ML approaches in the context of manufacturing and industry are related to predictive maintenance [14, 64]. Predictive Maintenance (PdM) is a data-driven approach that emerged in Industry 4.0. It uses statistical analysis, Machine Learning (ML) models for modeling complex systems behavior, identifying trends and predicting failures.

We review below some challenges of the paper in light of this domain. Challenge #1 (extending non-myopia to unsupervised approaches) is relevant, since an efficient anomaly detection system requires unsupervised approaches which can be combined with physics-based simulations of the different components. Challenge #2 (other supervised tasks) is also appropriate since both classification and regression problems appear (e.g. breakdown occurrence, prediction of energy consumption). Challenge #4 (data type agnostic) is especially relevant for DT's, since a system is always composed of several heterogeneous components. In this situation, the update of one component or one or several sensors would be much easier and cheaper if ML-EDM were data type agnostic. DT's operating at a system level leads to complex prediction models and complex decisions since the different components operate differently but in interaction (cf. challenge #5). The ability to manage non-stationarity (cf. challenge #7) is obviously central in DT's: aging and wearing of equipment lead to covariate and concept drifts which must be taken into account.

10.3 Social networks: societal and psychological risks

Online social networking platforms are more popular than ever. They radically transform the way we communicate with each other. However, this transformation comes with many problems on both sides, for users and platforms.

For example, *Fake news* spread widely during the covid pandemic. [5] tackled this problem as a binary classification problem where classes are "fake" and "real" news. *Fake accounts* are also considered a major problem , as they are among the main culprits in spreading false information. For instance, [8, 21, 25, 74] use Machine Learning techniques to detect these fake account based on interactions between users. Fake accounts can also be used for harassment and can induce major psychological risks [81]. The detection of depression and risk of suicide has been addressed using Machine Learning techniques in [15, 39].

Decisions taken by Machine Learning models to prevent such risks on social networks are clearly *time-sensitive*:

- Fake news must be detected as early as possible to limit its spread. [85] focuses on early detection of fake news from the press before it is expressed on social media.
- The early detection of *fake users* has also been studied in recent work. [13] proposes a graph-based approach which uses network connectivity to detect fake users.
- Detecting as early as possible *depressed* is very critical for prevention. This problem has also been addressed under the perspective of early classification in [47].

The development of the ML-EDM domain is an opportunity to go further in these applications. In particular, it would be very useful to develop unsupervised and weakly supervised ML-EDM approaches (see challenges #1 and #3). In this application area, ground truth is often unavailable or corrupted. Training data is very complex and consists of multiple sources: streams of texts, a large graph evolving over time etc. Therefore, it would be particularly beneficial to develop ML-EDM approaches which are agnostic to data types (see challenge #4).

10.4 Autonomous vehicle

An autonomous vehicle is defined in [79] as capable of sensing its environment and navigating safely without human input. Five levels of vehicle automation have been defined [55] as intermediate goals toward full automation. The development of a fully autonomous vehicle (levels 4 or 5) requires a complex software architecture, which operates numerous functional components [70]. More precisely, three classes of components have been identified, corresponding to different levels of control:

i) *Operational* components, which implement basic vehicle control such that lateral and longitudinal vehicle motion, monitoring of the driving environment ;

ii) *Tactical* components, which plan and execute vehicle maneuvers and prepare appropriate responses to incoming events, e.g. trajectory control, lane change;

iii) *Strategic* components, which determine the general itinerary according to the driver's preferences.

Given the high complexity of the tasks to be automated, *Machine Learning* approaches have become an essential element in the design of autonomous vehicles [52]. Machine learning is therefore used in the development of different classes of components:

i) *Operational* components are the most developed in the literature, and can be classified as follows: (1) *mediated perception* approaches [22, 40] aim to detect a wide variety of objects, such as obstacles, road signs ; (2) *direct perception* [11, 12] aim to directly manage vehicle controls without explicitly dealing with location and mapping ; (3) *localization* approaches [6, 80] characterize similarities and discrepancies between the environment and a priori maps, to locate the vehicle and obstacles.

ii) *Tactical* components are mostly developed to automate vehicle maneuvers using Machine Learning techniques, such as : (1) advanced scenarios of *automated parking* [38] ; (2) *car-following* improvement by predicting the trajectories of human-drivers [33] [49] ; (3) *trajectory planning* including obstacle avoidance [27], self-driving in urban environment [68].

Considering the *earliness vs. accuracy* compromise is an emerging and important issue in research for autonomous vehicles. In particular, *cooperative perception* [42] has been developed to extend the perceptual range of a connected autonomous vehicle, by sharing real-time information with other surrounding vehicles. In some ways, cooperative perception improves both the *earliness* and *accuracy* of decisions by extending the vehicles' perceptual capability.
The development of the ML-EDM field could make it easier to design fully autonomous vehicles. Indeed, the ability of these approaches to make *non-myopic* decisions is an advantage to make self-driving more fluid and safe. A non-myopic ML-EDM approach would be able to identify probable continuations of an observed situation on the road, based on related situations encountered in the training data and their continuations.
Most of the challenges presented in this article are relevant for the autonomous vehicle. Indeed, training data consists of multiple sources: video, radar etc. In addition, training data may change over time, for example with the arrival of a new types of sensors on a next generation of car. It is therefore important to develop ML-EDM approaches which are *agnostic* to data types (see challenge #4). Ground truth contained in the training data includes both the actions to be performed and the timing of these actions . It would be very useful to develop ML-EDM methods which learn the evolution of *decision costs* over time (see challenge #10). In autonomous vehicles, sensor data is continuously observed, so it is essential to design *online* ML-EDM approaches (see challenges #5 and #6), and driving actions must definitely be *revocable* (see challenges #8 and #9).

11. LOSS FUNCTION AND EVALUATION

This section describes the practical implementation for real applications of ML-EDM, in particular by describing how a loss function should be developed and how it can be used for evaluation purposes.

11.1 How to define the loss function ?

The loss function $\mathcal{L}$ in Equation 5 can be expressed in many different ways, depending on the application considered. In practice, *mapping rules* need to be defined to match the decisions made to the true ones. In Equation 5, the purpose is to map the indices i' to i, considering that the number of decisions made may be different than it should be ($\hat{k}_{\mathbf{x}} \neq k_{\mathbf{x}}$). In particular, these rules address the following questions: (i) how long should we wait before considering that a true decision has been missed? (see a rule example in Figure 8) (ii) when the number of decisions made is too large, how to identify the undue decisions? (e.g. Figure 9) (iii) what is the minimum time overlap between a decision made and the corresponding true one? (e.g. Figure 10) And of course, these mapping rules are specific to each application.

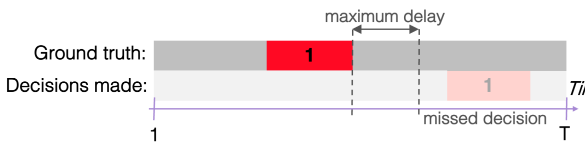


Figure 8: Maximum delay after which a decision is considered as missed.

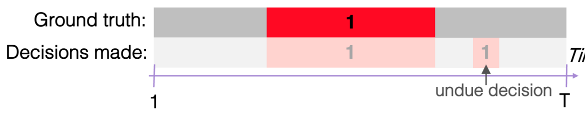


Figure 9: A decision is undue if no true decision exists in the time interval.

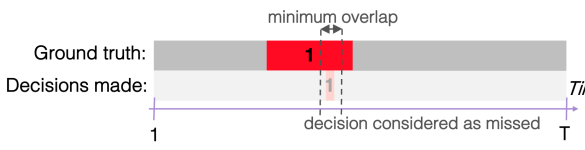


Figure 10: Minimum overlap to consider that a decision is not missed.

In its general form, the loss function $\mathcal{L}$ should involve several decision costs mentioned below:

- a prediction cost $\mathcal{L}_{prediction}$
- a delay cost $\mathcal{L}_{delay}$
- a time overlap cost $\mathcal{L}_{overlap}$
- a cost of missing the decision $\mathcal{L}_{missing}$
- a cost of an extra and undue decision $\mathcal{L}_{delete}$

The *prediction* cost $\mathcal{L}_{prediction}$ accounts for a potentially bad prediction and it can be expressed as a cost matrix. The delay cost $\mathcal{L}_{delay}$ depends on the trigger time $\hat{t}_{i'}$ and the time period associated to the i-th true decision $[s_i, e_i]$ (see Section 3). Figure 11 gives an example where a delay cost is paid since the triggering time (see the green vertical line) is located after the beginning of the period associated with the decision.
The last three decision costs have not been presented in the body of the article, and they can be modeled using an edit distance [65]. Namely, the *overlap* cost $\mathcal{L}_{overlap}$ accounts that the predicted periods $\{(\hat{s}_{i'}, \hat{e}_{i'})\}_{i'=1}^{\hat{k}_{\mathbf{x}}}$ might not coincide temporally with the periods of the true decisions $\{(s_i, e_i)\}_{i=1}^{k_{\mathbf{x}}}$. For instance in Figure 12, the decisions made (shown in the second line) are out of sync with the truth decisions (see the first line), which results in four overlapping

periods. The interested reader may refer to [77] which addresses the evaluation of models by considering such temporal overlap. Finally, the costs of missing a decision $\mathcal{L}_{missing}$ and making an additional undue one $\mathcal{L}_{delete}$ account that the number of decisions made can be different than it should be ($\hat{k}_{\mathbf{x}} \neq k_{\mathbf{x}}$). Figure 13 shows a situation where the first anomaly (represented by the class 1) is not detected, incurring a missing cost, and where a false detection occurs at the end leading to a delete cost.
Finally, to implement an ML-EDM application, it is necessary to define both the mapping rules and the decision costs mentioned above.

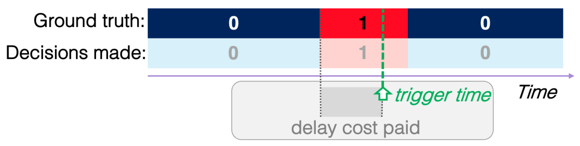


Figure 11: Example of paid delay cost.

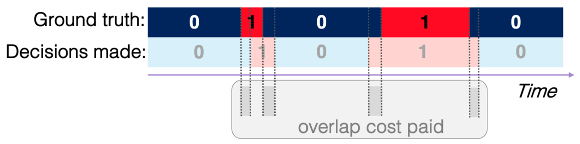


Figure 12: Example of paid overlap cost.

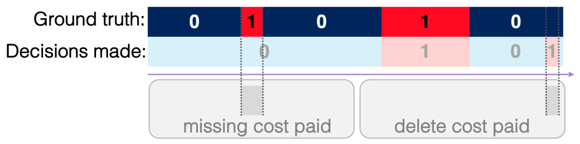


Figure 13: Example of a missing decision and an extra undue one.

11.2 How to evaluate an ML-EDM approach?

In some applications, decision costs are available as prior knowledge. It is the case for instance in [41], where the objective is to detect as early as possible patients suffering from septic shock, and where the cost of delaying decisions is perfectly known. When available, *decision costs* are of great help in evaluating ML-EDM approaches. Indeed, each decision made can be evaluated by the amount of costs actually incurred, considering: (i) the triggering moment ; (ii) the ground truth ; (iii) and the value of the decision costs (i.e. $\mathcal{L}_{delay}$, $\mathcal{L}_{prediction}$, and $\mathcal{L}_{revoke}$).
In the particular case of ECTS problem, a cost-based evaluation criterion is proposed in [2] called *AvgCost*, which simply corresponds to the *empirical risk* calculated on a set of test individuals $\mathcal{S}$ as following:

$$\begin{aligned} AvgCost(\mathcal{S}) &= \sum_{(\mathbf{x}_T, y) \in \mathcal{S}} \mathcal{L}(h(\mathbf{x}_{t^*}), t^*, y) \\ &= \sum_{(\mathbf{x}_T, y) \in \mathcal{S}} \mathcal{L}_{prediction}\,(h(\mathbf{x}_{t^*}), y) \; + \mathcal{L}_{delay}(t^*) \end{aligned} \tag{14}$$

where t^* is the triggering moment predicted for $\mathbf{x}$. *AvgCost* can also be interpreted as the average cost paid by the user on a particular set of examples, and this quantity should be minimized.
In the case of early and revocable time series classification [3], multiple decisions to classify the time series at hand are taken between timestamp 1 and T, the cost of revocation is added to Eq. 14 and becomes:

$$\begin{aligned} AvgCost(\mathcal{S}) = \sum_{(\mathbf{x}_T, y) \in \mathcal{S}} [\mathcal{L}_{prediction} \left(h(\mathbf{x}_{t^*_{\text{last}}}), y \right) \; + \\ \mathcal{L}_{delay}(t^*_{\text{last}}) + \mathcal{L}_{revoke} \left(\{(h(\mathbf{x}_{\hat{t_i}}), \hat{t_i})\}_{i \in [1, D^{\mathbf{x}}]} \right)] \end{aligned} \tag{15}$$

where $t^*_{\text{last}} = t_{D^{\mathbf{x}}}$ is the moment of the last decision. Furthermore, the loss of revocation is the sum of the multiple costs of decisions changes, the ideal situation is to have the less possible decision changes per time series, in order to guarantee a stable decision making algorithm.

$$\mathcal{L}_{revoke} \left(\{(h(\mathbf{x}_{\hat{t_i}}), \hat{t_i})\}_{i \in [1, D^{\mathbf{x}}]} \right) = \sum_{i=1}^{D^{\mathbf{x}}-1} C_{cd}(h(\mathbf{x}_{\hat{t}_{i+1}}) | h(\mathbf{x}_{\hat{t_i}}))$$

In the case of dealing with open time series (with infinite length), [4] proposed to evaluate decisions for each timestamp, the cost of delay becomes the cost of horizon.
In cases where decision costs are unavailable, a *multi-criteria* evaluation can be considered to take into account the different costs. For instance, in [59], the *earliness* and *accuracy* of decisions are evaluated separately, and the Pareto optimal front is made up of the dominant approaches considering both criteria.
In the more general case of ML-EDM where multiple early decisions have to be made, a cost-based evaluation requires more prior knowledge. Indeed, mapping rules would have to be defined in order to match the triggered decisions with the true ones, the cost of overlap $\mathcal{L}_{overlap}$ between predicted and true time periods, as well as the costs of missing a decision $\mathcal{L}_{missing}$ and triggering an undue one $\mathcal{L}_{delete}$ (see Equation 5 in Section 2 and Figures 8 to 13).

12. CONCLUSION AND PERSPECTIVES

More and more applications require to make time constrained decisions. On the one hand, an *early* decision is based on partially observed data, leaving time before the deadline which allows for a proper response by performing the right actions. On the other hand, a *late* decision based on nearly complete data tends to be more accurate, but leaves insufficient time to take appropriate action before the deadline. This compromise between the *earliness* and the *accuracy* of decisions has been particularly studied in the field of Early Time Series Classification (ECTS). In this paper a more general problem is introduced, called Machine Learning based Early Decision Making (ML-EDM), which consists in optimizing the decision times of models in a wide range of settings where data is collected over time.
This position paper aims to define the field of ML-EDM, and proposes ten challenges to the scientific community to further research in this area. In particular, ML-EDM has been defined and positioned with respect to related fields, such as machine learning and reinforcement learning. Three challenges have been presented in relation to the learning task at hand: extending ML-EDM to unsupervised learn-

ing (challenge #1), to regression tasks (challenge #2) and to weakly-supervised learning (challenge #3). The development of data type agnostic ML-EDM approaches has been singled out as an important direction of research to extend the domains of application, yielding challenge #4. Extending ML-EDM to the online scenario has also been recognized as important too which raises three challenges (challenge #5, #6 and #7). Being able to revoke decisions properly is significant as well in many applications and raises two challenges (#8 and #9). The origin of the different costs involved in the optimization of decision times has been discussed, leading to a last challenge (challenge #10) to extend the ML-EDM problem to cases where these costs vary over time. Finally, a range of application areas for which ML-EMD could lead to significant progress in the near future have been described.
The overall objective of this position paper is to define a new field of investigation, and to propose research avenues in order to generate interest from the scientific community. A Github page `https://github.com/ML-EDM/ML-EDM` has been created in order to gather all related material including research papers, datasets or source codes.

13. REFERENCES

[1] J. V. Abellan-Nebot and F. R. Subirón. A review of machining monitoring systems based on artificial intelligence process models. *The International Journal of Advanced Manufacturing Technology*, 47(1):237–257, 2010.

[2] Y. Achenchabe, A. Bondu, A. Cornuéjols, and A. Dachraoui. Early classification of time series. *Machine Learning*, 110(6):1481–1504, 2021.

[3] Y. Achenchabe, A. Bondu, A. Cornuéjols, and V. Lemaire. Early and revocable time series classification. In *In Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2022.

[4] Y. Achenchabe, A. Bondu, A. Cornuéjols, and V. Lemaire. Ecots: Early classification in open time series. *arXiv preprint arXiv:2204.00392*, 2022.

[5] O. Ajao, D. Bhowmik, and S. Zargari. Sentiment aware fake news detection on online social networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2507–2511. IEEE, 2019.

[6] P. F. Alcantarilla, S. Stent, G. Ros, R. Arroyo, and R. Gherardi. Street-view change detection with deconvolutional networks. *Autonomous Robots*, 42(7):1301–1322, 2018.

[7] A. Alipour-Fanid, M. Dabaghchian, N. Wang, P. Wang, L. Zhao, and K. Zeng. Machine learning-based delay-aware uav detection over encrypted wi-fi traffic. In *2019 IEEE Conference on Communications and Network Security (CNS)*, pages 1–7. IEEE, 2019.

[8] I. Aydin, S. Mehmet, and M. U. Salur. Detection of fake twitter accounts with machine learning algorithms. In *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, pages 1–4. IEEE, 2018.

[9] M. Beibel. A note on sequential detection with exponential penalty for the delay. *Annals of Statistics*, pages 1696–1701, 2000.

[10] A. Bifet, R. Gavaldà, G. Holmes, and B. Pfahringer. *Machine learning for data streams: with practical examples in MOA*. MIT press, 2018.

[11] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[12] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, and U. Muller. Explaining how a deep neural network trained with end-to-end learning steers a car. *arXiv preprint arXiv:1704.07911*, 2017.

[13] A. Breuer, R. Eilat, and U. Weinsberg. Friend or faux: Graph-based early detection of fake accounts on social networks. In *Proceedings of The Web Conference 2020*, pages 1287–1297, 2020.

[14] T. P. Carvalho, F. A. A. M. N. Soares, R. Vita, R. da P. Francisco, J. P. Basto, and S. G. S. Alcalá. A systematic literature review of machine learning methods applied to predictive maintenance. *Computers & Industrial Engineering*, 137:106024, 2019.

[15] G. Castillo-Sánchez, G. Marques, E. Dorronzoro, O. Rivera-Romero, M. Franco-Martín, and I. De la Torre-Díez. Suicide risk assessment using machine learning and social networks: A scoping review. *Journal of medical systems*, 44(12):1–15, 2020.

[16] C. Chatfield. *Time-series forecasting*. CRC press, 2000.

[17] A. Dachraoui, A. Bondu, and A. Cornuejols. Early classification of individual electricity consumptions. *RealStream2013 (ECML)*, pages 18–21, 2013.

[18] A. Dachraoui, A. Bondu, and A. Cornuéjols. Early classification of time series as a non myopic sequential decision making problem. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 433–447. Springer, 2015.

[19] S. Džeroski. Relational data mining. In *Data Mining and Knowledge Discovery Handbook*, pages 887–911. Springer, 2009.

[20] C. Elkan. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd, 2001.

[21] Y. Elyusufi, Z. Elyusufi, et al. Social networks fake profiles detection using machine learning algorithms. In *The Proceedings of the Third International Conference on Smart City Applications*, pages 30–40. Springer, 2019.

[22] D. J. Fagnant and K. Kockelman. Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations. *Transportation Research Part A: Policy and Practice*, 77:167–181, 2015.

[23] S. A. Fahrenkrog-Petersen, N. Tax, I. Teinemaa, M. Dumas, M. de Leoni, F. M. Maggi, and M. Weidlich. Fire now, fire later: alarm-based systems for prescriptive process monitoring. *arXiv preprint arXiv:1905.09568*, 2019.

[24] T. S. Ferguson. Who solved the secretary problem? *Statistical science*, 4(3):282–289, 1989.

[25] M. Fire, D. Kagan, A. Elyashar, and Y. Elovici. Friend or foe? fake profile identification in online social networks. *Social Network Analysis and Mining*, 4(1):194, 2014.

[26] P. I. Frazier and J. Y. Angela. Sequential hypothesis testing under stochastic deadlines. In *NIPS*, pages 465–472, 2007.

[27] A. Fuller, Z. Fan, C. Day, and C. Barlow. Digital twin: Enabling technologies, challenges and open research. *IEEE Access*, 8:108952–108971, 2020.

[28] J. Gama. *Knowledge discovery from data streams.* CRC Press, 2010.

[29] J. Gama. A survey on learning from data streams: current and future trends. *Progress in Artificial Intelligence*, 1(1):45–55, 2012.

[30] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37, 2014.

[31] M. F. Ghalwash, V. Radosavljevic, and Z. Obradovic. Utilizing temporal patterns for estimating uncertainty in interpretable early decision making. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 402–411. ACM, 2014.

[32] M. F. Ghalwash, D. Ramljak, and Z. Obradović. Early classification of multivariate time series using a hybrid hmm/svm model. In *2012 IEEE International Conference on Bioinformatics and Biomedicine*, pages 1–6. IEEE, 2012.

[33] S. Gong and L. Du. Cooperative platoon control for a mixed traffic flow including human drive vehicles and connected and autonomous vehicles. *Transportation research part B: methodological*, 116:25–61, 2018.

[34] A. Gupta, H. P. Gupta, B. Biswas, and T. Dutta. Approaches and applications of early classification of time series: A review. *IEEE Transactions on Artificial Intelligence*, 1(1):47–61, 2020.

[35] S. O. Hansson. Decision theory–a brief introduction. 1994.

[36] N. Hatami and C. Chira. Classifiers with a reject option for early time-series classification. In *Computational Intelligence and Ensemble Learning (CIEL), 2013 IEEE Symposium on*, pages 9–16. IEEE, 2013.

[37] G. He, Y. Duan, R. Peng, X. Jing, T. Qian, and L. Wang. Early classification on multivariate time series. *Neurocomputing*, 149:777–787, 2015.

[38] M. Heimberger, J. Horgan, C. Hughes, J. McDonald, and S. Yogamani. Computer vision in automated parking systems: Design, implementation and challenges. *Image and Vision Computing*, 68:88–101, 2017.

[39] M. R. Islam, M. A. Kabir, A. Ahmed, A. R. M. Kamal, H. Wang, and A. Ulhaq. Depression detection from social network data using machine learning techniques. *Health information science and systems*, 6(1):1–12, 2018.

[40] V. John, K. Yoneda, Z. Liu, and S. Mita. Saliency map generation by the convolutional neural network for real-time traffic light detection using template matching. *IEEE Transactions on Computational Imaging*, 1(3):159–173, 2015.

[41] F. Khoshnevisan and M. Chi. Unifying domain adaptation and domain generalization for robust prediction across minority racial groups. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 521–537. Springer, 2021.

[42] S.-W. Kim, W. Liu, M. H. Ang, E. Frazzoli, and D. Rus. The impact of cooperative perception on decision making and planning of autonomous vehicles. *IEEE Intelligent Transportation Systems Magazine*, 7(3):39–50, 2015.

[43] A. Klenke. *Probability theory: a comprehensive course.* Springer Science & Business Media, 2013.

[44] M. J. Kochenderfer. *Decision making under uncertainty: theory and application.* MIT press, 2015.

[45] G. Krempl, I. Žliobaite, D. Brzeziński, E. Hüllermeier, M. Last, V. Lemaire, T. Noack, A. Shaker, S. Sievi, M. Spiliopoulou, and J. Stefanowski. Open challenges for data stream mining research. *SIGKDD Explor. Newsl.*, 16(1):1–10, Sept. 2014.

[46] P. Latouche and F. Rossi. Graphs in machine learning: an introduction. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), Proceedings of the 23-th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2015)*, pages 207–218, 2015.

[47] V. Leiva and A. Freire. Towards suicide prevention: early detection of depression on social media. In *International Conference on Internet Science*, pages 428–436. Springer, 2017.

[48] V. Lemaire, C. Salperwyck, and A. Bondu. A survey on supervised classification on data streams. In *European Business Intelligence Summer School*, pages 88–125. Springer, 2014.

[49] L. Li, K. Ota, and M. Dong. Humanlike driving: Empirical decision-making system for autonomous vehicles. *IEEE Transactions on Vehicular Technology*, 67(8):6814–6823, 2018.

[50] J. M. Loyola, M. L. Errecalde, H. J. Escalante, and M. M. y Gomez. Learning when to classify for early text classification. In *Argentine Congress of Computer Science*, pages 24–34. Springer, 2017.

[51] R. Luzietti, R. Erkkola, U. Hasbargen, L. A. Mattsson, J. M. Thoulon, and K. G. Rosén. European community multi-center trial “fetal ecg analysis during labor”: St plus ctg analysis. 27(6):431–440, 1999.

[52] Y. Ma, Z. Wang, H. Yang, and L. Yang. Artificial intelligence applications in the development of autonomous vehicles: a survey. *IEEE/CAA Journal of Automatica Sinica*, 7(2):315–329, 2020.

[53] C. Martinez, E. Ramasso, G. Perrin, and M. Rombaut. Adaptive early classification of temporal sequences using deep reinforcement learning. *Knowledge-Based Systems*, 190:105290, 2020.

[54] C. Mathukia, W. Fan, K. Vadyak, C. Biege, and M. Krishnamurthy. Modified early warning system improves patient safety and clinical outcomes in an academic community hospital. *Journal of community hospital internal medicine perspectives*, 5(2):26716, 2015.

[55] D. Milakis, B. Van Arem, and B. Van Wee. Policy and society related implications of automated driving: A review of literature and directions for future research. *Journal of Intelligent Transportation Systems*, 21(4):324–348, 2017.

[56] U. Mori, A. Mendiburu, S. Dasgupta, and J. Lozano. Early classification of time series from a cost minimization point of view. In *Proceedings of the NIPS Time Series Workshop*, 2015.

[57] U. Mori, A. Mendiburu, S. Dasgupta, and J. A. Lozano. Early classification of time series by simultaneously optimizing the accuracy and earliness. *IEEE transactions on neural networks and learning systems*, 29(10):4569–4578, 2017.

[58] U. Mori, A. Mendiburu, E. Keogh, and J. A. Lozano. Reliable early classification of time series based on discriminating the classes over time. *Data mining and knowledge discovery*, 31(1):233–263, 2017.

[59] U. Mori, A. Mendiburu, I. M. Miranda, and J. A. Lozano. Early classification of time series using multi-objective optimization techniques. *Information Sciences*, 492:204–218, 2019.

[60] P. Nodet, V. Lemaire, A. Bondu, and A. Cornuéjols. Importance reweighting for biquality learning. *arXiv preprint arXiv:2010.09621*, 2020.

[61] N. J. P. Fetal electrocardiogram (ecg) for fetal monitoring during labour. *The Cochrane database of systematic reviews*, (12), 2015.

[62] N. Parrish, H. S. Anderson, M. R. Gupta, and D. Y. Hsiao. Classifying with confidence from incomplete information. *J. of Mach. Learning Research*, 14(1):3561–3589, 2013.

[63] J. Quiñonero-Candela, M. Sugiyama, N. D. Lawrence, and A. Schwaighofer. *Dataset shift in machine learning.* Mit Press, 2009.

[64] Y. Ran, X. Zhou, P. Lin, Y. Wen, and R. Deng. A survey of predictive maintenance: Systems, purposes and approaches. *arXiv preprint arXiv:1912.07383*, 2019.

[65] E. S. Ristad and P. N. Yianilos. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532, 1998.

[66] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K. Müller. A unifying review of deep and shallow anomaly detection. *Proc. IEEE*, 109(5):756–795, 2021.

[67] M. Rußwurm, R. Tavenard, S. Lefèvre, and M. Körner. Early classification for agricultural monitoring from satellite time series. *arXiv preprint arXiv:1908.10283*, 2019.

[68] D. O. Sales, D. O. Correa, L. C. Fernandes, D. F. Wolf, and F. S. Osório. Adaptive finite state machine based visual autonomous navigation system. *Engineering Applications of Artificial Intelligence*, 29:152–162, 2014.

[69] D. J. Samuel and F. Cuzzolin. Unsupervised anomaly detection for a smart autonomous robotic assistant surgeon (saras) using a deep residual autoencoder. *IEEE Robotics and Automation Letters*, 6(4):7256–7261, 2021.

[70] A. C. Serban, E. Poll, and J. Visser. A standard driven software architecture for fully autonomous vehicles. In *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 120–127. IEEE, 2018.

[71] B. Settles. Active learning literature survey. 2009.

[72] A. Sharma and S. Kumar Singh. A novel approach for early malware detection. *Transactions on Emerging Telecommunications Technologies*, page e3968, 2020.

[73] L. A. Shepp. Explicit solutions to some problems of optimal stopping. *The Annals of Mathematical Statistics*, 40(3):993, 1969.

[74] N. Singh, T. Sharma, A. Thakral, and T. Choudhury. Detection of fake profile in online social networks using machine learning. In *2018 International Conference on Advances in Computing and Communication Engineering (ICACCE)*, pages 231–234. IEEE, 2018.

[75] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[76] C. W. Tan, C. Bergmeir, F. Petitjean, and G. I. Webb. Time series extrinsic regression. *Data Mining and Knowledge Discovery*, 35(3):1032–1060, 2021.

[77] N. Tatbul, T. J. Lee, S. Zdonik, M. Alam, and J. Gottschlich. Precision and recall for time series. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[78] I. Teinemaa, N. Tax, M. de Leoni, M. Dumas, and F. M. Maggi. Alarm-based prescriptive process monitoring. In *International Conference on Business Process Management*, pages 91–107. Springer, 2018.

[79] S. Thrun. Toward robotic cars. *Communications of the ACM*, 53(4):99–106, 2010.

[80] H. J. Vishnukumar, B. Butting, C. Müller, and E. Sax. Machine learning and deep neural network—artificial intelligence core for lab and real-world test and validation for adas and autonomous vehicles: Ai for efficient and quality test and validation. In *2017 Intelligent Systems Conference (IntelliSys)*, pages 714–721. IEEE, 2017.

[81] H. Watanabe, M. Bouazizi, and T. Ohtsuki. Hate speech on twitter: A pragmatic approach to collect hateful and offensive expressions and perform hate speech detection. *IEEE access*, 6:13825–13835, 2018.

[82] R. Wu, A. Der, and E. J. Keogh. When is early classification of time series meaningful? *arXiv preprint arXiv:2102.11487*, 2021.

[83] Z. Xing, J. Pei, and S. Y. Philip. Early prediction on time series: A nearest neighbor approach. In *IJCAI*, pages 1297–1302. Citeseer, 2009.

[84] Z. Xing, J. Pei, S. Y. Philip, and K. Wang. Extracting interpretable features for early classification on time series. In *SDM*, volume 11, pages 247–258. SIAM, 2011.

[85] X. Zhou, A. Jain, V. V. Phoha, and R. Zafarani. Fake news early detection: A theory-driven model. *Digital Threats: Research and Practice*, 1(2):1–25, 2020.

[86] Z.-H. Zhou. A brief introduction to weakly supervised learning. *National science review*, 5(1):44–53, 2018.

[87] T. Zoppi, A. Ceccarelli, T. Capecchi, and A. Bondavalli. Unsupervised anomaly detectors to detect intrusions in the current threat landscape. *ACM/IMS Trans. Data Sci.*, 2(2), apr 2021.