FOX: a neuro-Fuzzy model for process Outcome prediction and eXplanation

Vincenzo Pasquadibisceglie, Giovanna Castellano, Annalisa Appice, Donato Malerba
Department of Computer Science
University of Bari Aldo Moro
Bari, Italy
Email: [vincenzo.pasquadibisceglie,giovanna.castellano,annalisa.appice,donato.malerba]@uniba.it

***Abstract*—Predictive process monitoring (PPM) techniques have become a key element in both public and private organizations by enabling crucial operational support of their business processes. Thanks to the availability of large amounts of data, different solutions based on machine and deep learning have been proposed in the literature for the monitoring of process instances. These state-of-the-art approaches leverage accuracy as main objective of the predictive modeling, while they often neglect the interpretability of the model. Recent studies have addressed the problem of interpretability of predictive models leading to the emerging area of Explainable AI (XAI). In an attempt to bring XAI in PPM, in this paper we propose a fully interpretable model for outcome prediction. The proposed method is based on a set of fuzzy rules acquired from event data via the training of a neuro-fuzzy network. This solution provides a good trade-off between accuracy and interpretability of the predictive model. Experimental results on different benchmark event logs are encouraging and motivate the importance to develop explainable models for predictive process analytics.**

***Index Terms*—Predictive process monitoring, Neuro-fuzzy model, Outcome prediction, Explainable Artificial Intelligence**

I. Introduction

The ability to analyze event log data and proactively monitor business processes has recently matured as one of the main enablers of data insights in enterprises. Several predictive process monitoring (PPM) approaches have been formulated to predict the unfolding of running traces (e.g. next activity, trace outcome, completion time) based on the knowledge learned from historical event logs. These approaches promise to be essential to transform logs in smart decisions and boost business values of enterprise. In the recent times, machine learning models have been widely applied in the realm of PPM. One of the main tasks in PPM is predicting the outcome of a business process. This involves the development of models to predict as early as possible whether the outcome of a process will fall into a positive class or a negative class.

While a huge number of machine learning techniques and more recently, deep learning techniques have been used to introduce advanced predictive capabilities in process analytics, the derived models are in most cases 'black-box' models [1]–[6]. Indeed, the models learned from data using a (deep) neural network are implicitly represented in numerical form as synaptic weights in the network. In general it is difficult, if not impossible, to interpret these weights due to their complex nature. This lack of model transparency is acceptable as soon as accuracy is the major objective in PPM. Indeed, so far the dominant criterion for assessing the quality of models for outcome prediction has been their accuracy expressed in terms of standard evaluation metrics. In particular, in case of outcome-oriented PPM, accuracy is typically measured by the area under the ROC curve (AUC) metric. The higher the AUC, the better the model is at predicting the outcomes.

However, when predicting outcomes of a business process trace, the interpretability of the predictive model is of fundamental importance to understand the decision of the model and possibly turn predictions into actions for preventing undesired outcomes. While the priority remains on giving accurate predictions, users need to be provided with an explanation of the reason why a given process execution is predicted to behave in a certain way. Otherwise, users would not trust the model, and hence they would not adopt the predictive-monitoring technology. Hence an explainable model is especially desirable in predictive process analytics since it enables process stakeholders to better assess its suitability when predicting the process behavior as well as for understanding the importance and relevance of certain features used for prediction. Conversely, due to the lack of explainability, a process analysts cannot identify suitable interventions that might affect the decision making related to the business process.

To address this issue, explainable artificial intelligence (XAI) has raised as a sub-field of artificial intelligence that aims to enable humans to understand the decisions of artificial systems by producing more explainable models, while maintaining a good level of predictive accuracy [7]. Explainability, in part, refers to the human interpretability of the processes underlying decisions given by a predictive model. The more interpretable a certain model is, the simpler it will be for a human to understand or explain the underlying reasoning.

A significant research interest is recently observed in the development of post-hoc explanations, in which the XAI algorithm can be applied to already trained classification models. For example, a consistent collection of model-agnostic explainers for predictive black boxes is described in in [8]. Alternatively, an explainable method can be incorporated into the learning algorithm. A learning methodology oriented toward the design of interpretable predictive models is Fuzzy Modeling [9], which leverages the Fuzzy set theory to develop knowledge-based models capable of both representing highly

2021 3rd International Conference on Process Mining (ICPM 2021), available at https://doi.org/10.1109/ICPM53251.2021

non-linear input-output relations, and at the same time offering an interpretable view of such relations through the use of linguistic IF-THEN rules. As such, fuzzy models are white-box models that turn out to be fully interpretable [10]–[12] and have a huge potential for the development of XAI systems [13], [14].

Traditionally, fuzzy modeling approaches assume the availability of a-priori knowledge to be represented in form of fuzzy rules. In many real-world scenarios, like business process monitoring, it is often unlikely an expert can accurately describe the complete behavior of the phenomenon underlying the data. To circumvent this limitation, neuro-fuzzy modeling approaches have been introduced as an ideal technique for utilizing empirical data to learn fuzzy rules through the training of neural networks [15], [16]. Despite its old origin, neuro-fuzzy modeling is currently going through a new renaissance within the realm of XAI [17] since it provides *gray-box* predictive models which, once learned from data, can be easily interpreted as a set of linguistic rules.

This paper represents the first attempt to investigate the use of neuro-fuzzy models for explainable PPM and propose a fully interpretable model for outcome prediction. The proposed method is based on a set of fuzzy rules acquired from event data via the training of a neuro-fuzzy network. Experiments show that this solution provides a good trade-off between accuracy and interpretability of the predictive model also compared to related outcome prediction approaches.

The rest of the paper is structured as follows. Section II provides a brief overview of existing related works. Section III describes the proposed methodology for process outcome prediction based on neuro-fuzzy modeling. In Section IV we report the results of several experiments on different benchmark event logs. Section V concludes the paper.

II. Related works

A. Outcome-oriented PPM

Previous approaches to outcome prediction commonly extract features from traces seen as sequences of event labels, and use these features to construct a classifier for run-time prediction [18]–[20]. However, these approaches often ignore the data payload associated to each event. In [21], the payload of the last executed event is taken into account, but the evolution of data throughout the execution traces is ignored. To address this issue, traces are handled as sequences of activities in [22], so that each activity carries a data payload consisting of attribute-value pairs. In addition, the authors of [22] compare the performance of various feature engineering approaches formulated for encoding the event occurring in a position and the value of each data attribute in that position. In [23], unstructured (textual) information, contained in text messages exchanged during process executions, is coupled to control and data flow information. A crucial phase in these approaches is how to encode a complex symbolic sequence in terms of vectors of features representative of the data payload.

As concerns the construction of the outcome classification model, different machine learning algorithms have been tested. In general, Random Forests and eXtreme Boosted Trees have shown to outperform other classification algorithms in several outcome prediction problems [24]. On the other hand, with the recent boom of deep learning in machine learning, the use of deep neural networks has emerged as a valuable candidate solution for PPM problems. In particular, various recent studies have definitely shown that the non-linear activation layers of deep neural networks may facilitate the discovery of accurate predictive models for predicting the next activity of a running trace [1], [3], [5], [6], [25].

Following the research direction of deep learning, the pioneering study in [2] has adapted the LSTM neural architecture defined in [1] for problems of next activity prediction, in order to predict the outcome of ongoing traces. More recently, a CNN-based neural architecture has been trained to predict the outcome of each trace once a trace has been encoded as an image [4]. The authors of [26] have explored the facets of LSTMs, LSTMs with attention ad CNNs in various outcome prediction problems. Note that the recent deep learning studies reported above have achieved new milestones in terms of accuracy . However, all the deep neural networks experimented until now for the outcome prediction derive black-box models that make their decisions difficult to interpret.

B. Explainable PPM

A common approach to address model interpretability in PPM is via post hoc interpretation that allows us to derive explanations and visualisations from a learned black-box model [27]. For example, in [28] the authors use a representative local surrogate method, known as Local Interpretable Model-Agnostic Explanations (LIME) to derive explanations useful in interpreting the prediction for a particular trace learned by machine learning techniques. The work [29] proposes a local post-hoc explanation approach for a deep learning classifier to develop explainable business process prediction solutions. In [30] two local posthoc explanation approaches, Shapley Values and Individual Conditional Expectation (ICE) plots are applied to generate the relevant explanations from a deep neural network trained to predict the process outcomes. In [31] a framework for explainable process monitoring of generic KPIs is proposed. Explanations are derived from LSTM (Long Short-Term Memory) neural models using the Theory of Shapley Values. The derived explanations can be delivered to process stakeholders to understand the behavior of the model, but also at run-time, to explain the predictions of each single running case. Similarly, in [32] a layer-wise relevance propagation method is proposed to make predictions of LSTM model explainable by providing relevance values for activities. Hence another approach to make predictions more explainable is to estimate how much the different activities included in a process may influence the prediction, which is an important information for process stakeholders. One of the first studies in this direction is [33], where Gated Graph neural networks are employed to derive relevance scores for each activity of a process instance that indicate its impact on the outcome prediction. A similar idea is described in [34] where

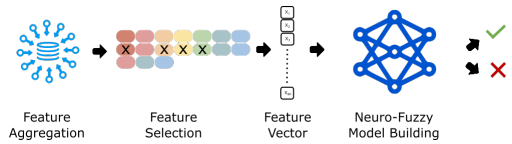


Fig. 1: FOX pipeline.

posthoc explanations in Belief Networks are derived through an evidence sensitivity analysis that differentiates between variables that cause the next event and variables that are the effect of an event. Also, in [35] the authors use post-hoc explainations and different encodings to identify the most common features that lead a predictive model for outcome-oriented predictions to provide wrong predictions.

All the above methods try to extract some form of interpretable knowledge after the predictive model has been generated, and hence they are model agnostic. A different XAI approach is to employ white-box or gray-box models that guarantee a certain level of interpretability already in the phase of the construction of the model. Hence, they are immediately interpretable by a human. Among white-box models, fuzzy models offer the advantage of a linguistic knowledge representation that can be cast into a conventional mathematical framework based on fuzzy logic concepts. This leads to a model structure made of linguistic rules, which can be viewed as a layered network having much in common with an ordinary feed-forward neural network. Thus neural training can be employed to fine-tune or learn from scratch a set of fuzzy rules using input-output data, leading to the so-called neuro-fuzzy models. Fuzzy models have been largely applied for outcome prediction in many fields, e.g. for clinical outcome prediction [36] or for prediction of student's academic performance [37]. However, to the best of our knowledge, no attempt has been made to apply fuzzy models to predict the outcome of business processes of public and private organizations. This work is the first contribution toward this research direction.

III. The FOX methodology

In this section we introduce FOX (a neuro-Fuzzy model for process Outcome prediction and eXplanation), a novel approach that combines running trace feature extraction and neuro-fuzzy modeling (see Figure 1) to address the outcome prediction problem. FOX resorts to a neuro-fuzzy modeling methodology by coupling interpretability to accuracy.

A. Problem statement

Let us assume the availability of an event log that records the execution of process traces of a specific business process. Every trace consists of a finite sequence of events $\sigma = \langle e_1, e_2, \ldots, e_n \rangle$ and is labeled with the outcome class (that is, the outcome achieved at the completion of the trace according to some business goal). An event measures a vector of attributes (comprising, for example, the executed activity, the resource triggering the event, the timestamp at which the event has been started). Optional trace attributes, that remain the same throughout the whole trace, may be also associated with each trace. From each trace in the event log, we can derive several prefix-traces σ^k with $1 \leq k \leq |\sigma|$. Hence, a trace is a complete process instance (started and ended), while a prefix trace is an instance in execution (running trace). Every prefix trace can, in principle, be labeled, with the same outcome class of the complete trace it belongs to. Based upon these premises, given an event log of completed traces with their known outcome class labels, FOX learns a set of IF-THEN rules from all the prefixes of all the labeled traces in the input event log. Through the inference of these rules, FOX is able to predict the outcome of any given running trace. In particular, in this work the predicted outcome is limited to be a regular or a deviant trace.

B. Feature extraction

Various trace encoding schemes are synthesized in the process mining literature. In [24], the authors have compared the performance of Last state, Aggregation and Index encoding schemes in various outcome prediction methods. Although no absolute "best encoding" exists, this study has shown that the Aggregation encoding schema can achieve high accuracy in various outcome prediction problems. The Aggregation schema been recently adopted in the CNN architecture described in [4] for outcome-oriented process monitoring. Based on these premises, we use the Aggregation encoding schema in this study to represent all prefix-traces as fixed length vectors of features. This schema considers all the events since the beginning of the prefix trace and applies aggregation operators to events' attributes. For numerical attributes of events (e.g. cost or duration attributes), it computes the average and standard deviation. For categorical attributes of events (e.g. activities and resource attributes), it uses the count aggregation function (e.g. how many times a given activity has been executed). Finally, it adds the trace features to the feature vector "as is" without any loss of past information. Trace categorical features are dealt as ordinal features, whose values are sorted according to the frequency of their occurrence in the original event log.

We apply a feature selection mechanism, in order to reduce the number of features by discarding the less relevant ones that have been computed through the Aggregation schema. We measure the relevance of each measure through the Mutual Info (MI)—a metric that is commonly used for feature scoring in classification problems [38]. The MI quantifies the amount of information obtained about the class through observing each feature to be analyzed. So, a feature that does not have much effect on the data classification has very small MI and it can be ignored without affecting the accuracy of a classifier. Based upon this consideration, features are ranked into the training set according to the MI measure so that the top-m features can be retained for the neuro-fuzzy modeling. Hence, each prefix trace is represented by a m-dimensional feature vector $\mathbf{x} = (x_1 .... x_m)$ that is associated with one of the two class labels ("regular" or "deviant") in order to create an annotated training set to perform neuro-fuzzy modeling.

C. Neuro-fuzzy modeling

The input-output data extracted from the prefix traces of a complete event log are used as training set to learn a classification model for predicting the outcome of a business process. To learn an interpretable model for outcome prediction we train a neuro-fuzzy network, i.e. a neural network that encodes in its structure a collection of fuzzy IF-THEN rules.

Each rule expresses a fuzzy relation between a prefix trace and the possible outcomes in the following form:

IF (AMOUNT_REQ is LOW) and
(Activity_O_SENT_BACK-COMPLETE is LOW) and
(Activity_W_Valideren aanvraag-SCHEDULE is LOW) and
(Activity_W_Valideren_aanvraag-START is LOW)
THEN trace is (**Regular** with degree 0.73)
(**Deviant** with degree 0.26)

Since fuzzy sets describe input features by means of linguistic terms (such as LOW, MEDIUM, HIGH,...) fuzzy rule-based models turn out to be white-box models with excellent capabilities to describe a given input-output mapping in an interpretable way. We consider zero-order Takagi-Sugeno (TS) fuzzy models [39] that use rules with fuzzy sets in the antecedents and singleton values in the consequent parts. The TS fuzzy rules for outcome prediction can be formalized as:

IF (x_1 is A_{k1}) AND ... AND (x_m is A_{km}) THEN (y_1 is b_{k1}) AND (y_2 is b_{k2})

for $k = 1, .., K$, where K is the number of rules, $A_{ki} (i = 1, ..., m)$ are fuzzy sets defined over the input features x_i and b_{kj} is a fuzzy singleton expressing the certainty degree of the outcome class $y_j, j = 1, 2$ (y_1 is the Regular class and y_2 is the Deviant class). Each fuzzy set is defined by a Gaussian membership function:

$$u_{ki} = \mu_{ki}(x_i) = \exp\left(\frac{(x_i - c_{ki})^2}{\sigma_{ki}^2}\right) \tag{1}$$

being c_{ki} and σ_{ki} the center and the width of the Gaussian function. The K rules are used to compute outcome certainty degrees for a trace $\mathbf{x} = (x_1, ..., x_m)$ by means of rule inference involving the following steps:

1) Compute membership values u_{ki} according to (1);
2) For $k = 1, .., K$ calculate the fulfillment degree of the k-th rule, by means of product operator:

$$u_k(\mathbf{x}) = \prod_{i=1}^{m} u_{ki} \tag{2}$$

3) Normalize fulfillment degrees:

$$\mu_k(\mathbf{x}) = \frac{u_k(\mathbf{x})}{\sum_{h=1}^{K} u_h(\mathbf{x})}; \tag{3}$$

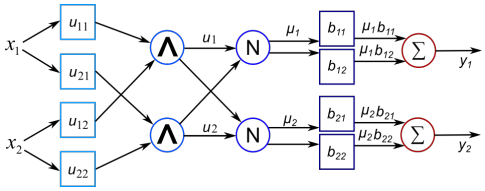


Fig. 2: ANFIS architecture.

4) For each outcome class, calculate the certainty degree:

$$y_j = \sum_{k=1}^{K} \mu_k(\mathbf{x}) \cdot b_{kj} \qquad j = 1, 2. \tag{4}$$

In presence of domain expert knowledge, the fuzzy rules can be manually defined by the expert who specifies the fuzzy terms in the antecedent and in the consequent of each rule. When the expert knowledge is missing (as in our task) the parameters of fuzzy rules can be easily learned from data exploiting supervised learning of neural networks. This leads to powerful neuro-fuzzy networks that are adaptive fuzzy system exploiting the similarities between fuzzy systems and Radial Basis Function neural networks [40]. Hence, the behavior of a model learned by a neuro-fuzzy network can either be represented by a set of humanly understandable fuzzy rules or by a combination of localized basis functions associated with local models, making them an ideal framework to perform nonlinear predictive modeling for tasks where model interpretability, as well as accuracy, is desirable.

To learn the parameters of fuzzy rules, namely the antecedent fuzzy set parameters c_{ik} and σ_{ik}, and the consequent parameters b_{jk}, we use the ANFIS (Adaptive-network-based fuzzy inference system) neuro-fuzzy architecture [15] which is a four-layer feed forward neural network which reflects the fuzzy rule base in its parameters and topology. The architecture of the network is depicted in Fig. 1. The four layers of the neuro-fuzzy network realize the inference of fuzzy rules by computing, respectively: membership degree to fuzzy sets, fulfillment degree for each fuzzy rule, normalized fulfillment degree for each fuzzy rule, and inferred output. Each computational unit u_{ki} in the first layer receives the input value x_i and computes its membership value to fuzzy set A_{ki} according to (1). The center c_{ki} and the width σ_{ki} of the Gaussian function are the adjustable parameters of unit u_{ki}. The second layer contains K units that compute the fulfillment degree of each rule according to (2). In this layer, no adjustable parameter is associated with the units. The third layer normalizes the rule fulfillment degrees. The fourth layer provides the outputs of the network, i.e. the certainty degree for each possible outcome (regular and deviant). Each output results from the inference of rules, according to (4). The ANFIS network is trained by means of a Backpropagation learning procedure based on gradient descent.

D. Implementation details

FOX, whose code is publicly available on the GitHub repository, is implemented in Python 3.6.9 – 64 bit version –using PyTorch 1.8.1 library. The source code of the proposed approach is available on the GitHub[1] repository. The code of the feature extraction stage is provided by [24] on https://github.com/irhete/predictive-monitoring-benchmark. In a conventional fuzzy inference system, the number of fuzzy sets and the number of rules is decided by an expert who knows the process to be modeled and can decide which level of granularity to apply. In our work, however, no domain expert is available. Hence the number of fuzzy sets assigned to each input variable is chosen empirically to be equal to three, so as to granulate each variable into three linguistic terms (i.e. LOW, MEDIUM and HIGH). Consequently, the number of rules is 3^m being m the number of input variables. This number defines the structure of the ANFIS network. The optimization of the hyper-parameters was conducted by considering the 20% of the training set as validation set using the Tree-structured Parzen Estimator (TPE). As the structure of the ANFIS is well defined, the optimised hyper-parameters are the learning rate $\{0.00001,\ 0.01\}$ and batch size [128, 256, 512]. The Backpropagation training is applied with early stopping to avoid overfitting. The training phase is stopped when there is no improvement in terms of validation loss for 10 consecutive epochs. Adam optimizer is choosen to minimize the loss function. We fix to 100 the number of epochs.

IV. Experimental results

A. Event logs

The experiments are performed on 11 outcome-prediction problems formulated in [24] and also tested in [4] from four real-life event logs:

- Sepsis that records trajectories of patients with symptoms of the life-threatening sepsis condition in a Dutch hospital. Three different labeling procedures are defined for this log: (i) sepsis_1 where a patient returns to the emergency room within 28 days from the discharge, (ii) sepsis_2 where a patient is (eventually) admitted to intensive care and (iii) sepsis_3 where a patient is discharged from the hospital on the basis of something other than Release A (i.e. the most common release type).
- BPIC2011 that contains data coming from a Dutch Academic Hospital. Each trace describes the medical history of a given patient so that the applied procedures and treatments are recorded as activities. Four constraint conditions are adopted to define four binary outcome prediction tasks accordingly (i.e. bpic2011_1, bpic2011_2, bpic2011_3 and bpic2011_4). In each binary task, a trace is assigned to outcome equal to 1 if the constraint is violated, to 0 otherwise. Details on the used constraint formulation are reported in [24].
- BPIC2012 that contains the execution history of a loan application process in a Dutch financial institution. The multi-class labeling for this log is based on the final outcome of a trace, that is, whether the application is accepted, rejected or canceled. Based on this multi-class labeling, three separate binary outcome prediction tasks are defined (bpic2012_1, bpic2012_2 and bpic2012_3).
- Production that contains data from a manufacturing process. The labeling (production) is based on whether or not the number of rejected work orders is larger than zero.

The selected logs are accessible from the 4TU Centre for Research Data.[2] The considered outcome prediction problems have been selected among those experimented in [24], where the Aggregation schema (that we have also considered in this paper for feature extraction) achieved the highest performance.

B. Experimental setup

We have reproduced the experimental setting adopted in both [4] and [24]. A temporal split is used to divide the event log into train and test traces. To this aim, the traces of a log are sorted by the the starting timestamp. The first 80% are selected for training the predictive model, while the remaining 20% are considered to evaluate the performance of the learned model on the unseen traces. This splitting procedure allows us to simulate the real-life situation, where prediction models are trained using historic data (started before a given data) and applied to ongoing traces. Whenever some training events overlap with the testing period, training traces with events that overlap with the testing period have been discarded.

For the evaluation, let us consider that a classifier usually outputs a real-valued score, reflecting how likely it is that the sample will end in one way or the other. Accounting for the considerations reported in [24], a good outcome classifier will give higher scores to ongoing traces that will end with a positive outcome, and lower values to those ending with a negative one. Therefore, similarly to [4] and [24], we also measure the area under the ROC curve (AUC) metric that expresses the probability that a given classifier will rank a positive case higher than a negative one. A major advantage of the AUC metric over the commonly used accuracy is that the AUC remains unbiased even in case of a highly imbalanced distribution of class labels. Furthermore, AUC is a threshold-independent measure, as it operates on the ranking of the scores rather than the binary class values. In this study, the AUC is computed for predictions yielded for all testing prefix traces with length greater than one.

C. Related methods

We compare the performance of FOX to that of several related methods formulated in the recent literature, namely SVM, LR, RF and XGB (introduced in [24]), LSTM [2] and ORANGE, that uses a CNN architecture [4]. SVM, LR, RF and XGB are run using the optimization described in [24] to select the hyperparameters of the machine learning algorithms. ORANGE is run with the architecture hyper-parameters automatically selected with tree-structured Parzen

[1]https://github.com/vinspdb/FOX

[2]https://data.4tu.nl/

estimator as described in [4]. In addition, SVM, LR, RF, XGB and ORANGE, similarly to FOX, use the Aggregation schema for the trace feature extraction. The decision of using the Aggregation schema with these competitors follows the conclusions drawn in the empirical study discussed in [24]. LSTM is run with the architecture hyper-parameters set as described in [2] and the trace feature extraction performed with the Index schema. The decision of coupling the LSTM architecture with the Index schema follows the formulation of the competitor reported in [2], as well as the consideration that, differently from the Aggregation schema, the Index schema is able to preserve the information on the order that the LSTM architecture can process.

D. Results and discussion

We start this validation comparing the performance of FOX to that of a baseline denoted as CN2. This baseline uses the same feature extraction step adopted in FOX and the traditional rule learning algorithm CN2 [41] for the outcome prediction. The algorithm CN2 induces an ordered list of classification rules from examples using entropy as its search heuristic. This experiment aims at analysing how the neuro-fuzzy model can improve the performance of a traditional rule learning algorithm given that both approaches learn interpretable predictive models. CN2 is run with a grid search conducted for the parameter optimization of the rule learning algorithm (beam_width and min_covered_examples are optimized between 1 and 100 with step 5, max_rule_length is optimized between 1 and the number of distinct attribute-values pairs, default_alpha and parent_alpha are optimized between 0 and 1).

Table I reports the AUC, the time spent in minutes completing the training process and the number of rules learned by both FOX and CN2. The computation times are collected running the experiments on Intel(R) Core(TM) i7-9700 CPU, GeForceRTX 2080 GPU, 32GB Ram Memory, Windows 10 Home. The AUC results highlight that FOX can actually take advantage of the neuro-fuzzy model to outperform CN2 in all the considered outcome prediction problems. However, as expected the improvement in the AUC performance is at the cost of the more time spent completing the learning stage. On the other hand, no general conclusion can be drawn on the complexity of the model (number of rules) as there are several datasets (sepsis_2, bpic2011_1, bpic2011_4 and bpic2012_1), where FOX learns a lower number of rules than CN2. On the other hand, there are problems (sepsis_1, sepsis_3, bpic2011_2, bpic2011_3, bpic2012_2, bpic2012_3 and production), where CN2 learns a lower number of rules than FOX.

We proceed with this validation by comparing the performance of FOX to that of the list of related methods. The AUC of both FOX and the related methods is reported in Table II. We use the Friedman's test followed by the post-hoc test—the Nemenyi test to compare and rank the AUC of these methods on multiple problems. According to the Friedman's test, the null hypothesis is rejected, since no approach is singled out

TABLE I: Comparison between FOX and CN2 in terms of AUC, computation time (in minutes) and number of rules. For each metric, the best results per event log are in bold.

Event Log	FOX			CN2		
	AUC	Time(m)	N. of rules	AUC	Time(m)	N. of rules
sepsis_1	**0.58**	12	243	0.41	**6**	**212**
sepsis_2	**0.73**	5	**81**	0.68	**2**	130
sepsis_3	**0.68**	24	729	0.64	**5**	**228**
bpic2011_1	**0.97**	16	**81**	0.96	**6**	274
bpic2011_2	**0.92**	208	2187	0.71	**25**	**212**
bpic2011_3	**0.98**	43	729	0.87	**10**	**309**
bpic2011_4	**0.89**	17	**9**	0.73	**4**	369
bpic2012_1	**0.64**	48	**81**	0.61	**10**	188
bpic2012_2	**0.60**	354	729	0.53	**87**	**424**
bpic2012_3	**0.69**	318	729	0.64	**23**	**148**
production	**0.74**	2	243	0.62	**0.4**	**52**

TABLE II: AUC of FOX and related methods.The best AUC per event log are in bold.

Event Log	SVM	LR	RF	XGB	ORANGE	LSTM	FOX
sepsis_1	0.49	0.57	0.41	0.33	0.57	0.52	**0.58**
sepsis_2	0.82	**0.86**	0.79	0.85	**0.86**	0.73	0.73
sepsis_3	0.72	**0.73**	0.66	0.72	**0.73**	0.61	0.68
bpic2011_1	0.87	0.92	0.94	0.94	0.96	0.93	**0.97**
bpic2011_2	0.95	0.94	**0.98**	**0.98**	**0.98**	0.91	0.92
bpic2011_3	0.96	0.96	**0.98**	**0.98**	**0.98**	0.93	**0.98**
bpic2011_4	0.87	0.87	0.89	0.86	**0.90**	0.89	0.89
bpic2012_1	0.63	0.65	0.69	**0.70**	0.67	0.62	0.64
bpic2012_2	0.55	0.59	**0.61**	0.57	**0.61**	0.60	0.60
bpic2012_3	**0.70**	0.69	**0.70**	0.69	**0.70**	**0.70**	0.69
production	0.66	0.67	0.63	0.70	0.71	0.67	**0.74**

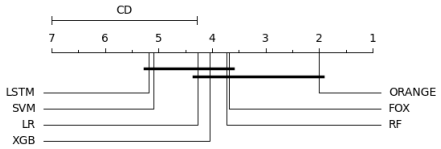


Fig. 3: Comparison among FOX and the related methods in terms of AUC with the Nemenyi test. Groups of methods that are not significantly different (at $p \leq 0.05$) are connected. CD denotes the critical difference area of the test.

with *p-value* ≤ 0.05. The results of pairwise comparisons of Nemenyi test reported in Figure 3 shows that ORANGE is ranked higher than the competitors as it takes advantage of a sophisticated CNN architecture trained to model possible correlation patterns among trace features. However, FOX is the runner-up of this comparison. In addition, ORANGE trains a black-box model (based on a CNN architecture) that allows us to predict the outcome of any running trace without providing any explanation of how the prediction is yielded. FOX learns a set of human interpretable rules that can always explain why a decision is taken. We conclude that FOX realizes a good trade-off between accuracy and explainability in the current overview of the PPM approaches for trace outcome prediction.

As an example of the explainability of a prediction yielded by FOX, Figure 4a (resp. Figure 4b) shows the top firing rule in case of a trace belonging to the deviant class (resp. regular class). Rule in fig. 4a explains that FOX predicts the trace as a deviant one (with certainty degree 0.15) and this outcome derives mainly from a medium value of Work_order_Qty, a

```
This trace is deviant with degree=0.40 because:
Work_Order_Qty is medium
Activity_Turning & Milling - Machine 4 is low
Resource_ID0998 is low
Resource_ID4794 is low
Resource.1_Machine 4 - Turning & Milling is low
```

(a)

```
This trace is regular with degree=0.15 because:
Work_Order_Qty is low
Activity_Turning & Milling - Machine 4 is low
Resource_ID0998 is medium
Resource_ID4794 is low
Resource.1_Machine 4 - Turning & Milling is low
```

(b)

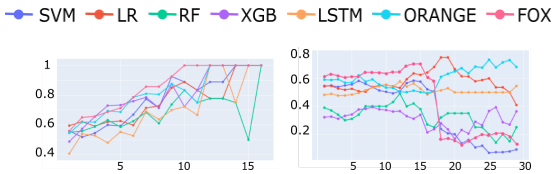
Fig. 4: An example of fuzzy rule learned for class Deviant (a) and for class Regular (b).

low value of the Activity_Turning & Milling - Machine 4, and a low frequency of occurrence in the trace for the resources ID0998, ID4794 and 1_Machine 4 - Turning & Milling. The rule in fig. 4b can be interpreted in a similar manner. Hence the derived rules describe the prediction in terms of fuzzy values that granulate input variables through linguistic labels (low, medium,high) representing high-level concepts. Since fuzzy rules describe the model behaviour in human-readable form, the model can be easily interpreted. Moreover each rule is very simple and hence easily readable. Indeed, regardless the number of features characterizing a dataset, the number m of variables that appear in the rule antecedent varies from 2 up to 7. This enhances explainability in terms of readability of each single fuzzy rule. FOX can actually single out rules that explain the prediction. However, the resulting number of rules (3^m) grows exponentially with the number of input variables, and this may hamper explainability, preventing experts to easily analyze the whole knowledge base of the model. One solution could be to prune off unnecessary rules, i.e. rules with low fulfillment degree, in order to achieve simpler models without losing accuracy. This is our current work in progress.

At the completion of this study, we evaluate the earliness of the predictions yielded with both FOX and the related methods. As an example, Figures 5a and 5b show the AUC computed along the different prefix lengths of Production and Sepsis_1, respectively. These results show that FOX outperforms related methods in almost all stages in Production, and in the early stages in Sepsis_1. This is a desirable result as it is more critical predicting the deviant outcome in the early stages of a trace to have time for prescribing corrections.

V. Conclusion

Boosted by the recent growing attention towards eXplainable Artificial Intelligence, we propose a novel approach, named FOX, to learn a fully interpretable model for trace outcome prediction, so that the model can be also used for explaining the predicted outcome. In particular, this approach



(a) (b)

Fig. 5: AUC (axis Y) on the outcome prediction problem of datasets production (a) and sepsis_1 (b) across different prefix lengths (axis X).

founds on the idea that neuro-fuzzy models can be trained as white-box PPM models that turn out to be fully able to explain outcome predictions. Experimental results on several real-world event logs show that the proposed approach compared to various related methods, comprising deep learning-based, achieves a good trade-off between accuracy and explainability.

Although the experiments have proved the effectiveness of FOX in various outcome prediction problems, the proposed method suffers from a few limitations. One limitation of FOX is the lack of prescription with the explanation of predictions. We plan to explore how we can take advantage of interpretability of the fuzzy rules to enrich the proposed PPM approach with guidelines that describe what to do to achieve specific outcomes. We also plan to expand the model by integrating possible reactions to prediction-based alerts. A further limitation is that FOX, similarly to competitors, does not implement any solution to deal with the imbalanced condition that commonly occurs in various outcome prediction problems. To overcome this limitation, we plan to extend the proposed method by implementing trace augmentation techniques, in order to achieve the balance in the learning stage. Another interesting research direction is that of extending the proposed approach in a streaming setting with the training performed continuously as new events are logged. This will require the use of a transfer learning approach to fine-tune the neuro-fuzzy model as new events are collected and deal with possible concept drift. Another future work concerns the exploration of encoding mechanisms illustrated in [42] as an alternative to the encoding schema adopted in this study.

Acknowledgment

The research of Vincenzo Pasquadibisceglie is funded by PON RI 2014-2020 - Big Data Analytics for Process Improvement in Organizational Development - CUP H94F18000260006. We acknowledge the support of the MIUR-Ministero dell'Istruzione dell'Università e della Ricerca through the project "TALIsMan -Tecnologie di Assistenza personALizzata per il Miglioramento della quAlità della vitA" (Grant ID: ARS01_01116), as well as the project "Modelli e tecniche di data science per la analisi di dati strutturati" funded by the University of Bari "Aldo Moro".

References

[1] N. Tax, I. Verenich, M. La Rosa, and M. Dumas, "Predictive business process monitoring with LSTM neural networks," in *International Con-*

ference on Advanced Information Systems Engineering, CAISE 2017, E. Dubois and K. Pohl, Eds. Springer, 2017, pp. 477–492.

[2] I. Teinemaa, M. Dumas, A. Leontjeva, and F. M. Maggi, "Temporal stability in predictive process monitoring," *Data Mining and Knowledge Discovery*, vol. 32, no. 5, pp. 1306–1338, 2018.

[3] V. Pasquadibisceglie, A. Appice, G. Castellano, and D. Malerba, "Using convolutional neural networks for predictive process analytics," in *Proc. of the 1st International Conference on Process Mining (ICPM2019)*. Aachen, Germany: IEEE, June 24-26 2019.

[4] V. Pasquadibisceglie, A. Appice, G. Castellano, D. Malerba, and G. Modugno, "ORANGE: Outcome-oriented predictive process monitoring based on image encoding and cnns," *IEEE Access*, pp. 184 073–184 086, 2020.

[5] M. Camargo, M. Dumas, and O. G. Rojas, "Learning accurate LSTM models of business processes," in *International Conference on Business Process Management, BPM 2019*, ser. LNCS, T. T. Hildebrandt, B. F. van Dongen, M. Röglinger, and J. Mendling, Eds., vol. 11675. Springer, 2019, pp. 286–302.

[6] V. Pasquadibisceglie, A. Appice, G. Castellano, and D. Malerba, "A multi-view deep learning approach for predictive business process monitoring," *IEEE Transactions on Services Computing*, pp. 1–1, 2021.

[7] H. Lakkaraju, E. Kamar, R. Caruana, and J. Leskovec, "Faithful and customizable explanations of black box models," in *Proc. of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, 2019, pp. 131–138.

[8] P. Biecek, "Dalex: Explainers for complex predictive models in r," *Journal of Machine Learning Research*, vol. 19, no. 84, pp. 1–5, 2018. [Online]. Available: http://jmlr.org/papers/v19/18-416.html

[9] J. Abonyi, "Fuzzy model identification," in *Fuzzy model identification for control*. Springer, 2003, pp. 87–164.

[10] J. M. Alonso, C. Castiello, and C. Mencar, "Interpretability of fuzzy systems: Current research trends and prospects," *Springer handbook of computational intelligence*, pp. 219–237, 2015.

[11] M. J. Gacto, R. Alcalá, and F. Herrera, "Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures," *Information Sciences*, vol. 181, no. 20, pp. 4340–4360, 2011.

[12] C. Mencar, G. Castellano, and A. Fanelli, "On the role of interpretability in fuzzy data mining," *International Journal of Uncertainty, Fuzziness and Knowlege-Based Systems*, vol. 15, no. 5, pp. 521–537, 2007.

[13] J. M. Alonso, C. Castiello, L. Magdalena, and C. Mencar, "Explainable fuzzy systems: Paving the way from interpretable fuzzy systems to explainable ai systems," *Studies in Computational Intelligence; Springer Nature: Cham, Switzerland*, 2021.

[14] A. Fernandez, F. Herrera, O. Cordon, M. J. del Jesus, and F. Marcelloni, "Evolutionary fuzzy systems for explainable artificial intelligence: Why, when, what for, and where to?" *IEEE Computational Intelligence Magazine*, vol. 14, no. 1, pp. 69–81, 2019.

[15] J.-S. Jang and C.-T. Sun, "Neuro-fuzzy modeling and control," *Proceedings of the IEEE*, vol. 83, no. 3, pp. 378–406, 1995.

[16] G. Castellano and A. Fanelli, "Self-organizing neural fuzzy inference network," *Proceedings of the International Joint Conference on Neural Networks*, vol. 5, pp. 14–19, 2000.

[17] T. Rojat, R. Puget, D. Filliat, J. Del Ser, R. Gelin, and N. Díaz-Rodríguez, "Explainable artificial intelligence (xai) on timeseries data: A survey," *arXiv preprint arXiv:2104.00950*, 2021.

[18] A. Pika, W. van der Aalst, M. Wynn, C. Fidge, and A. ter Hofstede, "Evaluating and predicting overall process risk using event logs," *Information Sciences*, vol. 352-353, pp. 98–120, 2016.

[19] C. Di Francescomarino, M. Dumas, F. M. Maggi, and I. Teinemaa, "Clustering-based predictive process monitoring," *IEEE Transactions on Services Computing*, vol. 12, no. 6, pp. 896–909, 2019.

[20] L. Genga, C. Di Francescomarino, C. Ghidini, and N. Zannone, "Predicting critical behaviors in business process executions: When evidence counts," in *Proc. of Business Process Management Forum (BPM Forum 2019)*, ser. Lecture Notes in Business Information Processing, T. Hildebrandt *et al.*, Eds., vol. 360. Vienna, Austria: Springer, 2019, pp. 72–90.

[21] F. M. Maggi, C. Di Francescomarino, M. Dumas, and C. Ghidini, "Predictive monitoring of business processes," in *Advanced Information Systems Engineering*, M. Jarke, J. Mylopoulos, C. Quix, C. Rolland, Y. Manolopoulos, H. Mouratidis, and J. Horkoff, Eds. Springer International Publishing, 2014, pp. 457–472.

[22] A. Leontjeva, R. Conforti, C. Di Francescomarino, M. Dumas, and F. M. Maggi, "Complex symbolic sequence encodings for predictive monitoring of business processes," in *International Conference on Business Process Management*. Springer, 2016, pp. 297–313.

[23] I. Teinemaa, M. Dumas, F. M. Maggi, and C. Di Francescomarino, "Predictive business process monitoring with structured and unstructured data," in *Proc. of International Conference on Business Process Management*. Springer, 2016, pp. 401–417.

[24] I. Teinemaa, M. Dumas, M. La Rosa, and F. M. Maggi, "Outcome-oriented predictive process monitoring: Review and benchmark," *ACM Transactions on Knowledge Discovery from Data*, vol. 13, no. 2, pp. 1–57, 2019.

[25] V. Pasquadibisceglie, A. Appice, G. Castellano, and D. Malerba, "Predictive process mining meets computer vision," in *Business Process Management Forum - BPM Forum 2020, Proceedings*. Springer, 2020.

[26] H. Weytjens and J. D. Weerdt, "Process outcome prediction: CNN vs. LSTM (with attention)," in *Business Process Management Workshops - BPM 2020 International Workshops, Seville, Spain, September 13-18, 2020, Revised Selected Papers*, ser. Lecture Notes in Business Information Processing, A. del-Río-Ortega, H. Leopold, and F. M. Santoro, Eds., vol. 397. Springer, 2020, pp. 321–333.

[27] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*. IEEE, 2018, pp. 80–89.

[28] R. Sindhgatta, C. Ouyang, and C. Moreira, "Exploring interpretability for predictive process analytics," in *International Conference on Service-Oriented Computing*. Springer, 2020, pp. 439–447.

[29] N. Mehdiyev and P. Fettke, "Explainable artificial intelligence for process mining: A general overview and application of a novel local explanation approach for predictive process monitoring," *arXiv preprint arXiv:2009.02098*, 2020.

[30] ——, "Local post-hoc explanations for predictive process monitoring in manufacturing," *CoRR*, vol. abs/2009.10513, 2020. [Online]. Available: https://arxiv.org/abs/2009.10513

[31] R. Galanti, B. Coma-Puig, M. de Leoni, J. Carmona, and N. Navarin, "Explainable predictive process monitoring," in *2020 2nd International Conference on Process Mining (ICPM)*. IEEE, 2020, pp. 1–8.

[32] S. Weinzierl, S. Zilker, J. Brunk, K. Revoredo, M. Matzner, and J. Becker, "Xnap: Making lstm-based next activity predictions explainable by using lrp," in *Business Process Management Workshops*, A. Del Río Ortega, H. Leopold, and F. M. Santoro, Eds. Cham: Springer International Publishing, 2020, pp. 129–141.

[33] M. Harl, S. Weinzierl, M. Stierle, and M. Matzner, "Explainable predictive business process monitoring using gated graph neural networks," *Journal of Decision Systems*, pp. 1–16, 2020.

[34] J. Brunk, M. Stierle, L. Papke, K. Revoredo, M. Matzner, and J. Becker, "Cause vs. effect in context-sensitive prediction of business process instances," *Information Systems*, vol. 95, p. 101635, 2021.

[35] W. Rizzi, C. Di Francescomarino, and F. M. Maggi, "Explainability in predictive process monitoring: When understanding helps improving," in *Business Process Management Forum*, D. Fahland, C. Ghidini, J. Becker, and M. Dumas, Eds. Cham: Springer International Publishing, 2020, pp. 141–158.

[36] G. Cosma, G. Acampora, D. Brown, R. C. Rees, M. Khan, and A. G. Pockley, "Prediction of pathological stage in patients with prostate cancer: a neuro-fuzzy model," *PLoS One*, vol. 11, no. 6, 2016.

[37] O. Taylan and B. Karagözoğlu, "An adaptive neuro-fuzzy model for prediction of student's academic performance," *Computers & Industrial Engineering*, vol. 57, no. 3, pp. 732–741, 2009.

[38] J. Vergara and P. Estevez, "A review of feature selection methods based on mutual information," *Neural Computing and Applications*, vol. 24, pp. 175–186, 01 2014.

[39] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE transactions on systems, man, and cybernetics*, no. 1, pp. 116–132, 1985.

[40] J.-S. Jang and C.-T. Sun, "Functional equivalence between radial basis function networks and fuzzy inference systems," *IEEE transactions on Neural Networks*, vol. 4, no. 1, pp. 156–159, 1993.

[41] P. Clark and R. Boswell, "Rule induction with cn2: Some recent improvements," in *Machine Learning — EWSL-91*, Y. Kodratoff, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 151–163.

[42] A. Leontjeva, R. Conforti, C. Di Francescomarino, M. Dumas, and F. M. Maggi, "Complex symbolic sequence encodings for predictive monitoring of business processes," in *Business Process Management*, H. R. Motahari-Nezhad, J. Recker, and M. Weidlich, Eds. Cham: Springer International Publishing, 2015, pp. 297–313.