**CLUSTERING BASED PREDICTIVE MONITORING FRAMEWORK:**

**-Shailesh Nanisetty**

The goal of predictive monitoring is to determine if a current running trace will reach a given outcome based on historical traces.

So, we have built a framework called as the “PM Framework” which requires a labelling function that, given a trace tells us if it is normal or deviant. Accordingly, the PM Framework takes as input both an event log and a labelling function fc. The labelling function can be defined, for example, using Linear Temporal Logic (LTL) rules, which we’ll discuss later.

**CONTENTS:**

**2.1. What is an event log?**

**2.2. Running example.**

**2.3. Clustering-Based Predictive Monitoring.**

*2.3.1. Control Flow Encoding.*

*2.3.2. Clustering.*

*2.3.3. Encoding.*

*2.3.4. Supervised Learning.*

*2.3.5. Predictive Monitoring.*

**3. Evaluation of the built PM Framework.**

*3.1. Dataset.*

*3.2. Process.*

*3.3. Research Questions and Metrics.*

*3.4. Results.*

*3.5. Discussion.*

*3.6. Limitations of the model.*

*3.7. Alternative Models which can be tried out.*

**2.1.** In the context of business process analytics, an event log is a collection of time-stamped event records produced by the execution of a business process. Each event record tells us something about the execution of a work item (a task) of the process.

For example, an event record may tell us that a given task started or completed at a given point in time, or it tells us that a given message event, escalation event, or other relevant event has occurred in a given case in the process.

**2.2**. P: Payment

PAYLOAD

Sym: painA

Dia: d1

Pre: p1

Treat: ?

T1:<M,A,C,D,A,A,P,D,C,R> M:MessageRequest

T2:<A,A,A,C,D,D,A,A,S,P,D,C,S> A:Analysis

T3:<M,A,C,D,D,A,P,D,C> C:Confirmation

T4:<P,A,A,C,D,D,A,A,S,P,D,C,S> V:Visit

. D: Diagnosis

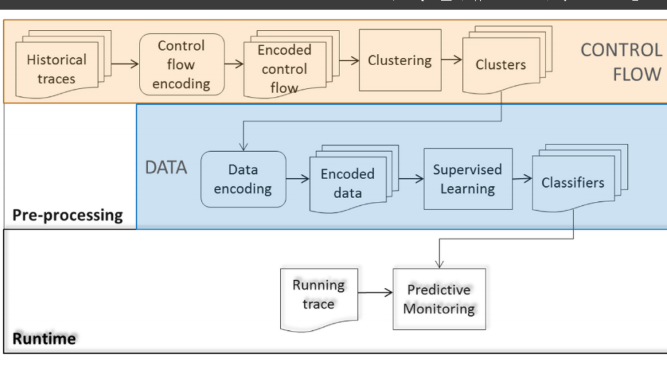
FIG.1. . R: Recovery

In Fig. 1, we show examples of executions pertaining to a patient diagnosis process. In trace t1, a message with a diagnosis request (M) arrives. The request contains a list of patient’s symptoms, e.g., painA .The patient is required to do some clinical analysis (A) and, once the results of the analysis are received, the reception is confirmed (C). Then, a diagnosis is made by the doctor (D), e.g., d1, and again two times some new clinical analysis is required. After that, the hospital fee is paid by the patient (P), a new diagnosis is made by the doctor and the reception of the analysis is confirmed. Finally the patient recovers from the disease (R).

Data consumed and produced in the process is globally visible throughout the whole process execution in the form of attribute: value pairs carried by event payloads. The payload of an event, signalling the execution of an activity A, contains the values of each attribute after the execution of A as well as the values of attributes for each activity that occurred before activity A in the trace. For example, in Fig. 1, the payload associated to the doctor diagnosis activity (D) contains the values associated to attributes of the diagnosis activity, e.g., the diagnosis (dia) and possibly the prescription (pre) of the doctor, as well as those associated to past activities, e.g., patient’s symptoms (sym). We write P(D)= {sym: painA; dia: d1; pre: p1; ...} to refer to the payload of activity D.

**2.3.**

So, we’ll be talking and trying to understand about our proposed framework .i.e. Clustering-Based Predictive Monitoring.



Fig, 2.

In such a framework, state-of-the-art approaches for clustering and classification are applied to the historical data in order to (i) identify and group historical trace prefixes with a similar control flow (clustering from a control flow perspective); and (ii) get a precise classification based on the data of traces with similar control flow (data-based classification).

At runtime, the classification of the historical trace prefixes is used to classify new traces during their execution and predict how they will behave in the future. The overall picture of the framework is illustrated in Fig. 2. In the following, we describe each framework component in detail.

**2.3.1.**

Before applying state-of-the-art techniques for clustering and classification, two propaedeutic steps are applied: (i) the selection of the historical trace prefixes to consider; and (ii) their encoding. In particular, prefixes of past execution traces are selected (rather than the entire trace or all the prefixes for a trace). The reason behind this choice is twofold: on the one side, taking all the prefixes could become very expensive in terms of efficiency. On the other side, we are interested in early predictions, when still reparative actions can be undertaken to prevent violations. In this light, considering only the initial parts of the historical traces seems to be a reasonable choice.

For example, given the six traces t1; ...; t6 of Fig. 1, only a selection of k prefixes for each trace will be considered. Different approaches can be used for the selection of these k prefixes. For example, the first k prefixes of each historical trace can be selected or alternatively k prefixes, one every g events. In the latter case, two prefixes differ one from another for a gap of g events.2 Different approaches can also be taken to perform the encoding of trace prefixes for clustering. Just to name a few, a trace (prefix) can be encoded as a sequence of events or in terms of the frequency of the occurrence of sequence patterns in the trace. The simplest case is the one related to the occurrence of unary patterns, i.e., patterns composed of a single log event. For example, in the scenario in Fig. 1, we can represent the alphabet of the events as an ordered vector L = <A; C; D; M; P; R; S; V >. In this case, trace t1 will be encoded as a vector of frequencies <3; 2; 2; 1; 1; 1; 0; 0>, obtained by replacing each symbol of the alphabet in vector L by its frequency in trace t1. Trace prefixes encoded in this way are used as input of the clustering phase.

**2.3.2.**

In the clustering phase, a selection of prefixes of the historical traces with the same (control flow) characteristics is grouped together based on some distance notion. Examples of distances are Euclidean distance and the string-edit distance.

The historical traces contained in each cluster are then used to generate a classifier that is exploited, in turn, to make predictions on running traces, once identified their matching cluster. For example, the execution traces in Fig. 1 could be grouped by a clustering algorithm in two clusters c1 and c2, according to the similarities in their control flow, so that c1 contains traces t1 and t3 (which have a very similar control flow), and c2 contains the remaining four traces.

**2.3.3.**

Trace prefixes in each cluster are used as input for supervised learning. In this case, the data perspective is taken into consideration. Historical execution traces are encoded using the available data attributes in the event payloads, i.e., prefixes clustered based on control flow are now analysed from a data perspective.

Each prefix is encoded as a feature vector that includes elements corresponding to the data assignments contained in the payload associated to the last event of the prefix. In addition, each prefix in a cluster is classified based on whether the corresponding completed trace is “normal” or “deviant” with respect to the input labelling function fc.

**2.3.4.**

Each cluster is used as training set of a supervised learning technique (e.g., decision tree learning, random forest) to generate a classifier that allows for discriminating between deviant and normal behaviours. For example, given two clusters c1 and c2, for each of them a classifier is built.

**2.3.5.**

At runtime, the set of classifiers generated during the pre-processing phase is used to make predictions about how the behaviour of a current running trace will develop in the future. At any point in time, the current prefix of the running trace is classified as part of one of the clusters identified during the pre-processing phase. This is done by considering the cluster containing the prefix with the minimum distance from the current prefix.

Based on the selected cluster (and, therefore, based on the control flow characteristics of the current prefix) the corresponding classifier is selected. This classifier is queried using the payload of the last event of the current prefix (exploiting the data perspective of the current prefix).

For example, given a partial execution trace tp: <M; A; C; D> and the predicate “the patient will recover eventually”, we first identify the cluster to which the partial trace belongs, e.g., c1, and then the classifier associated to the cluster (e.g., the decision tree in Fig. 3) is exploited in order to predict whether the predicate will be verified or not.

**3**.

So, We implemented the proposed PM Framework as a so-called “Operational Support provider” on top of the ProM framework. In this way, the framework can be used in a “streaming” mode, meaning that it can take as input a stream of events coming from an external system.

Specifically, the PM Framework uses the Weka implementation of the clustering and classification algorithms.

Using such an implementation, we conducted an evaluation of different configurations of the proposed framework on a real-life dataset as reported below.

**3.1.**

We conducted the analysis using the BPI challenge 2011 event log. This log records the execution of a cancer treatment process in a Dutch academic hospital over a three years period.

The log contains 1,143 traces and 150,291 events. Each trace in the log refers to the treatment of one patient. Each event represents an execution of one among 623 activities.

Each event contains a timestamp, an event type (i.e., an activity lifecycle state like start or complete), a case (i.e., patient) identifier, and a number of domain-specific attributes (e.g., Age, Diagnosis, and Treatment code). There are 15 domain-specific attributes in total.

Since the goal of predictive monitoring is to classify a case as normal or deviant, we need to define a notion of deviance (i.e., a labelling function). We experimented with 4 labelling functions corresponding to the following LTL rules:

F1=F (“tumour marker CA-19.9”) V F (“ca-125 using meia”);

F2=G (“CEA-tumor marker using meia”-> F (“squamous cell carcinoma using eia”));

F3= (-,”histological examination-biopsies nno”) U (“squamous cell carcinoma using eia”);

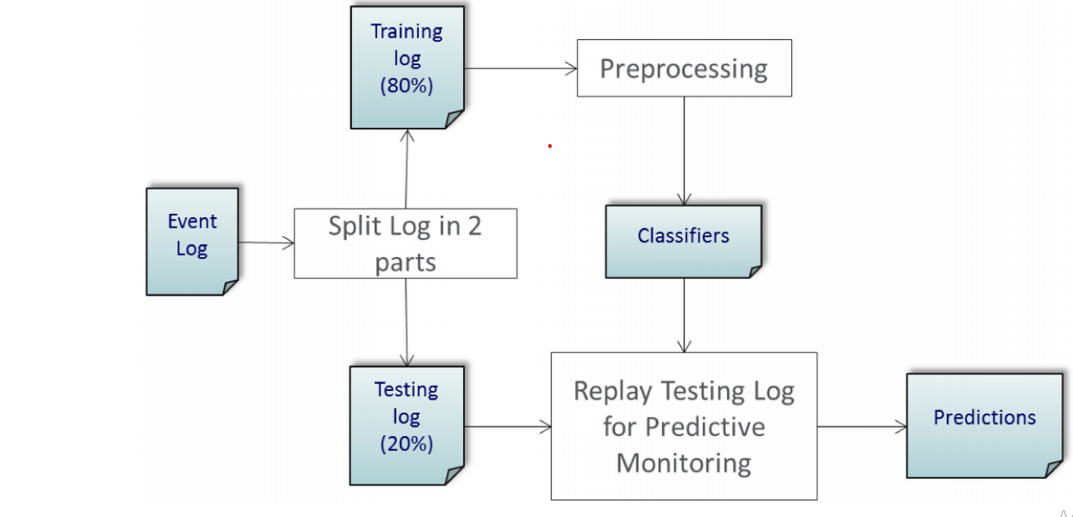
F4=F (“histological examination –big resectiep”);

For a given LTL rule, if a case violates the LTL rule, it is labelled as deviant (herein called a “positive” case); otherwise it is labelled as normal (herein called a “negative” case).

Each of these labelling functions captures a business rule. Specifically, F1 assesses that either the diagnostic test for the tumor marker CA-19.9 or for the tumor marker ca-125 has to be performed.

F2 states that every time the diagnostic test for the CEA tumor marker is performed, then the eia test for the squamous cell cancer has also to be performed eventually. F3 assesses that no histological examination can be performed until the eia test for the squamous cell cancer is performed and, finally, F4 states that the resection for the histological examination has to be performed eventually.

The distribution of positive and negative cases in the event log is: 458 negative versus 682 positive for F1, 893 negative versus 247 positive for F2, 259 negative versus 881 positive for F3, and 319 negative versus 821 positive for F4.



**3.2.**

We’ll divide the process of building the Predictive Monitoring Framework into 4 steps.

**A**. Data Pre-processing.

**B**. Construction of Cluster.

**C**. Building of the Classifier.

**D**. Prediction of Testing log.

A.

First, we order the traces in the log based on the time at which the first event of each trace has occurred. Then, we split the log temporally into two parts: 80-20 percent. We used the first part as training log, i.e., we used these traces as historical data to construct clusters and build the classification models for prediction. Then, we implemented a log replayer to simulate the execution of the remaining traces (the testing log) by pushing them as an event stream to the implementation of the PM Framework and making predictions for each case during this replay.

This step involves transformation of Event Log Train data (in our example is a dataset which is highly sparse with traces as identities) into a clean and useful form of data.

So, we first order the traces in the log based on the time at which the first event of each trace has occurred because the data is identified by time stamped events.

After all the necessary data pre-processing steps, we proceed to step 2.

**B.**

The second step in the framework is to construct clusters of prefixes. For this, we used two popular clustering methods, namely model-based clustering [7] and DBSCAN [8]. In model-based clustering, we need to calculate the centre point and covariance matrix of clusters. . These parameters can only be computed if we use an Euclidean distance.

The string-edit distance-which is otherwise a natural distance in the context of traces-is not an Euclidean distance. Hence, we applied model—based clustering using the Euclidean distance over the frequency—encoded prefixes. On the other hand, in DBSCAN, we just need to calculate the distance between two points and this can be done using the edit distance.

Accordingly, for DBSCAN, we used edit distance over sequence encoded prefixes.

For the model-based clustering method, it is necessary to set the number of clusters to be created (parameter k). Meanwhile, DBSCAN requires the minimum number of points in a cluster (parameter minPoints) and the minimum radius of a cluster (parameter).

In the case of model-based clustering, we applied model-based clustering with k = 3 to 10 clusters and chose the value of k that achieved the highest Bayesian Information Criteria (BIC).

Meanwhile, the DBSCAN optimal parameters were estimated by using the sorted k-dist graph. This led us to set minPoints = 4 and epsilon= 0.125.

**C**.

The third step in the framework is to build a classifier for each cluster of trace prefixes. Specifically, each cluster is used as training set of a supervised learning technique to generate a classifier that discriminates between deviant and normal cases.

In this paper, we use decision trees, which are known for the interpretability of the models they generate, and random forests, which apply similar principles as decision trees but are designed to maximize accuracy rather than interpretability.

Also, we’ll use Artificial Neural Networks which are well known when dealing with very sparse and high-dimensional data. These are also very robust to outliers which makes it very compatible to use.

Combining the two clustering and the first two classification techniques, we obtain the following four PM Framework instances:

**A.) mbased\_dt:** model-based clustering and decision trees;

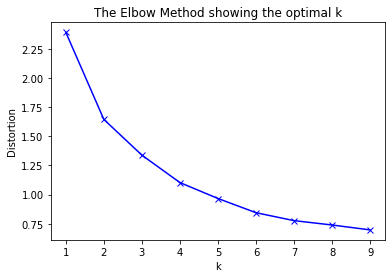
**B.) dbscan\_dt:** DBSCAN clustering and decision trees;

**C.) mbased\_rf:** model based clustering and Random Forests;

**D.)** **dbscan\_rf:** DBSCAN clustering and Random Forests;

We replayed each trace in the testing set and produced a prediction of the outcome of the case every five events (starting from the first event in each trace). Each prefix of each running trace is encoded in the same way as the historical traces and assigned to the closest cluster.

In case of model-based clustering, the closest cluster is the one with the minimum Euclidean distance from the current prefix, while for DBSCAN the closest cluster is the cluster containing the prefix with the minimum edit distance from the current prefix. We use the classifier associated to the closest cluster to predict the label for the current running case.



From the above fig, we have 8 as the optimal number of clusters obtained for Model-Based Clustering.

**3.3.**

The goal of the evaluation is focused on two aspects: the performance of the approach in terms of the quality of the results and the performance of the approach in terms of the time required to provide predictions.

In particular, we are interested in answering the following two important questions:

Q1. How effective is the PM Framework in providing accurate results as early as possible?

Q2. How efficient is the PM Framework in providing results?

Overall, we have used three metrics in our experiment to answer the above questions.

1. **F1-score:** for Q2.
2. **Earliness:** for Q1.
3. **Accuracy:** for Q2.

In the following pages, we’ll go through the discussion of each one of these three parameters.

**F1-Score:**

This measure is defined with respect to a gold standard that indicates the correct labelling of each trace. In our experiments, we extracted the gold standard (**Check Section 3.1**) by evaluating the input predicate on each completed trace in the testing set.

Given the gold standard, we classify predictions made at runtime into four categories: i) true-positive (**TP: positive outcomes correctly predicted**); ii) false-positive (**FP: negative outcomes predicted as positive**); iii) true-negative (**TN: negative outcomes correctly predicted**); iv) false-negative (**FN: positive outcomes predicted as negative**). F1-score, which intuitively represents the proportion of correctly classified positive results with respect to all the possible cases, is defined as:

F1-Score = 2\*TP/2\*TP+FP+FN - (1)

**Earliness:**

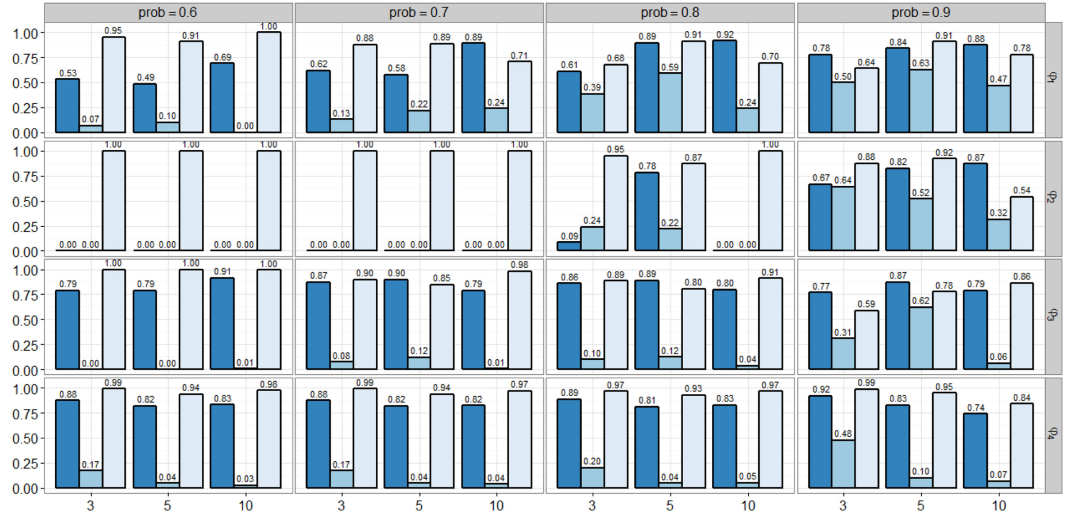
The earliness of the prediction is defined as 1 minus the ratio between the index indicating the position of the last evaluation point (the one corresponding to the reliable prediction) and the size of the trace under examination. Earliness is a crucial measure since, during the execution of a business process, the stakeholders must be provided with predictions as soon as possible to apply possible reparative actions in case there is high probability of deviance in the future.

**Accuracy:**

It gives you the overall **accuracy** of the model, meaning the fraction of the total samples that were correctly **classified** by the **classifier.** To calculate **accuracy**, use the following formula: (TP+TN)/ (TP+TN+FP+FN).

**3.4.**

1. **Model-Based clustering and decision tree classification**:

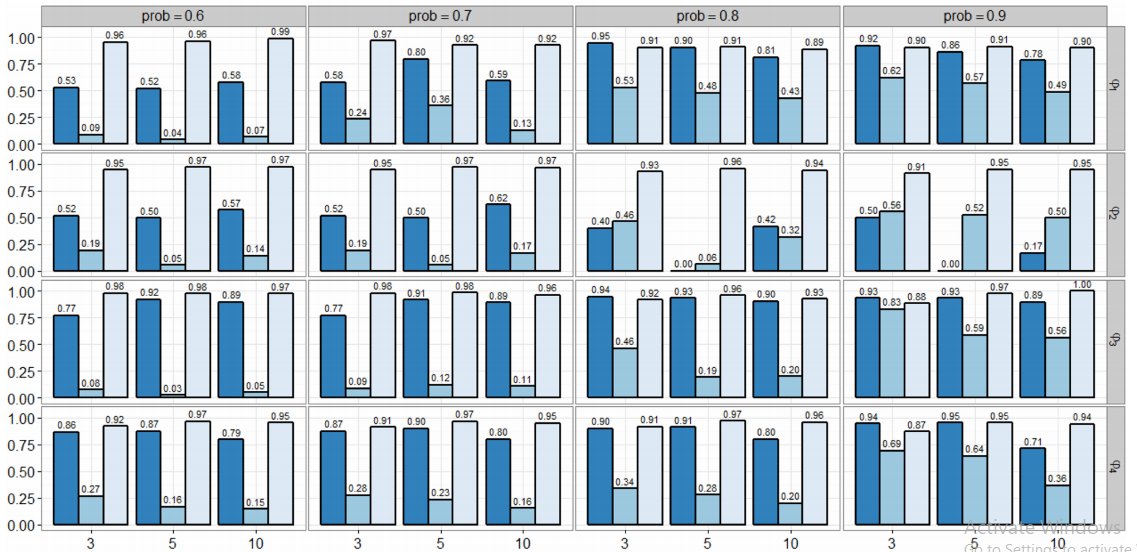


- Failure Rate, - 1-Accuracy, - Earliness

For different minimum class probability thresholds from 0.6 to 0.9 and trace lengths 3,5,10... ; The metrics have been evaluated for each one corresponding to four gold standards.

The plots show that mbased\_dt reaches peaks of F1-score of 0.92 for phi1 with a threshold for minimum class probability (prob = 0.8). A high Accuracy influences the F1-score since, in its computation, we can rely on a lower number of predictions. On the other hand, a high earliness can also negatively affect the F1-score. Indeed, a too high earliness results into very little information carried by the running trace in terms of control flow. Focusing only on the results with a reasonably good accuracy (e.g., with failure-rate greater than 0.6) and an earliness not excessively high (e.g., lower than 0.9), mbased\_dt still guarantees to find, for each predicate, a parameter configuration resulting in a good F1-score. The F1-score values range, indeed, between 0.58 and 0.92 for the four predicates.

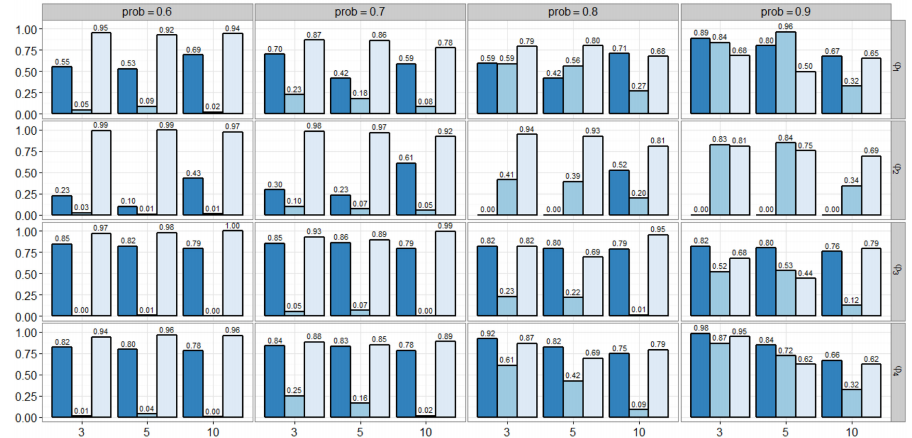
2. **DBSCAN clustering and decision tree classification:**

****

- Failure Rate, - 1-Accuracy, - Earliness

In this case (somewhat similar to previous figure), the earliness is always very high and this leads to low values for F1-score for the two predicates with a low number of deviant cases, i.e., for phi1 and especially for phi2. For class probability thresholds 0.8 and 0.9, the F1-score reaches values higher than 0.9 for three of the four predicates under examination (phi1, phi3 and phi4), whereas for phi2 it is not able to perform better than 0.5.

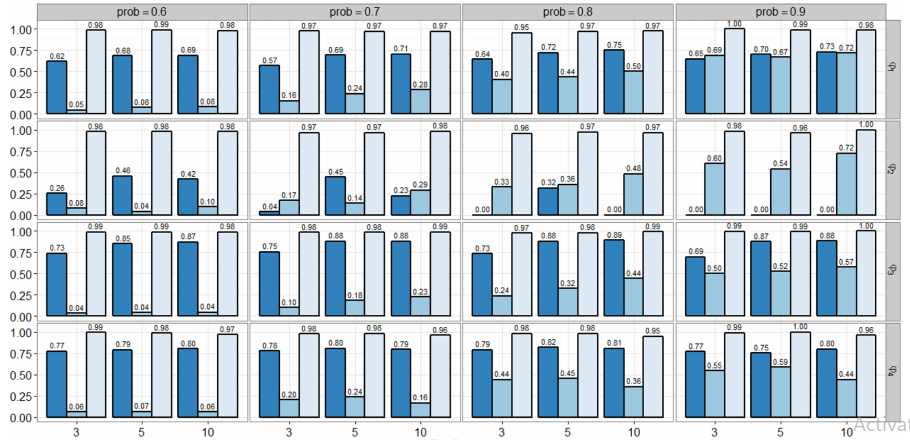
1. **Model Based Clustering and Random Forest Classification:**

****

- Failure Rate, - 1-Accuracy, - Earliness

For prob = 0.9, the F1-score reaches values higher than 0.8 for three of the four predicates under examination (phi1, phi3 and phi4), whereas for phi2 is equal to 0 for all the prefix gaps. By looking at the only results with Accuracy greater than 0.75 and earliness lower than 0.9, the F1-score values for all predicates lie in the range 0.42-0.86.

1. **DBSCAN Clustering and Random Forest Classification:**



- Failure Rate, - 1-Accuracy, - Earliness

We note that, also in this case, like in dbscan\_dt , the earliness is extremely high (for all the considered configurations it is always higher than 0.95). This leads to low values for F1- score for the two predicates with a low number of deviant cases, i.e., phi1 and phi2. For prob= 0.9, the F1-score reaches values higher than 0.7 for three of the four predicates under examination (phi1, phi3 and phi4), whereas for phi2 is equal to 0 for all the prefix gaps.

**3.5.**

The observations and the analysis performed so far allow us to draw some conclusions and guidelines. The solutions provided by the different instances of the Clustering based PM Framework offer the possibility to meet different types of needs, by opportunely setting the available configuration parameters.

For instance, in settings in which users are more interested in getting predictions at an early stage of a trace execution, low minimum class probability thresholds should be preferred. The same type of thresholds should also be preferred to have a prediction even if not always correct rather than a non-prediction. Indeed, low minimum class probability thresholds would allow users to get an almost null failure-rate with an acceptable F1-score in many cases.

For the choice of the clustering and the classification technique to use for instantiating the PM Framework, mbased\_rf presents an average F1-score lower than the other instances. In general, the instances based on random forests seem to perform slightly worse than the ones based on decision trees. Furthermore, mbased\_dt outperforms dbscan\_dt. The instances based on DBSCAN present a very high earliness. In all the cases, the failure-rate increases with the minimum class probability threshold.

The choice of the configuration values also depends on the predicate under consideration. Predicates with a lower number of positive cases in the historical dataset lead to a lower F1-score. For instance, in the investigated settings, the F1-scores derived for phi1 and phi2 are generally worse than the ones derived for phi3 and phi4.

**3.6**.

One of the main threats to the external validity of the evaluation is the application of the PM Framework to a single event log. The use of more logs would clearly allow for more general results. However, such a threat is mitigated by the fact that the considered log is a real-life log with real data chronologically ordered so as to simulate a realistic scenario. A second threat is the choice of the predicates used for the evaluation. Also in this case, we limited ourselves to 4 predicates. However, they are realistic business rules covering all the LTL constructs.

**3.7.**

Other model which can be tried out to solve this problem is using **Artificial Neural Networks. ANN** may give better results than the models we built because of the sparseness of data and presence of large number of features.