FIREMAN

WP5 Deliverable 5.3a
Report on detection and prediction techniques: Part a

Project Title:	Framework for the Identification of Rare Events via MAchine learning and IoT Networks

Title of Deliverable:	Report on detection and prediction techniques: Part a
Status-Version:	Final-1.0
Delivery Date:	21/12/2021

Contributors:	Ioannis Christou, Pedro Juliano Nardelli, Daniel Gutierrez Rojas, Eslam Eldeeb, Hirley Alves
Reviewers:	Mehdi Rasti, Valentin Niclas
Approved by:	All Partners

Document Revision History

Version	Date	Description
v0.1	11 October 2021	Table of Contents
v0.2	05 November 2021	Main contents added
v0.3	12 November 2021	Additional contents added
v0.4	14 December 2021	First version delivered for external review
v1.0	21 December 2021	Approved form of Deliverable D5.4a

Summary

This document summarizes the progress made so far in the context of *Detection and Prediction Techniques*, which is the focus of the Task 5.1. Specifically, this report covers the initial results of machine-type of transmissions related to ON-OFF traffic, which is related to one of the test cases of WP6 (Oulu University smart campus measurements) and previous ideas presented in D3.1, D4.1 and D5.3a. The other part of this contribution covers the QARMA framework used to predict, detect and/or classify rare events (anomalies) based on quantitative association from data. In this case, we have indicated the success of QARMA in a case studied based on simulation results that was focus of WP2 (reported in D2.3). In more fundamental studies of QARMA, this deliverable extends the approach towards an "online" version; other reports on QARMA is available in Deliverables 2.3, 2.4, 4.2 and 5.1. The final report of this task will also deal with aspects related to the final deployments of WP6.

Table of Contents

1 Introduction

1.1 Objective of the document

In this deliverable, we highlight the main features of the methods being developed in the executions of Task 5.1. Specifically, we will discuss new aspects of QARMA, as well as its application in a fault selection task related to power lines. Besides, we have also analyzed traffic prediction in binary events (alarms) related to the test studied in WP6. We discuss an online version of the algorithm whereby, after an initial run of the algorithm on an initial training dataset, the produced rules are updated in real-time as new instances (labeled with ground truth values) are fed to the online version. Every rule's support and confidence is then updated in the database, and if either the support or confidence drops below the user-specified thresholds, the rule is removed altogether from the database. After a sufficient number of such new training data has been presented to the online version, the rule ensemble is tested on the (unseen) validation datasets, according to the algorithms presented in this deliverable for rule ensemble fusion. We present a number of results of this approach that demonstrate its effectiveness in practical real-world applications.

The practical development of this task is presented in two open access repositories:

- Machine learning scripts specifically from FIREMAN `https://github.com/5uperpalo/FIREMAN-project`
- Neural and MinExplainSet packages partly related to FIREMAN `https://github.com/ioannischristou/popt4jlib`

1.2 Structure of the document

The remainder of this document is structured as follows. Section 2 presents a novel approach to predict traffic in machine-type communications, considering hidden Markov models. Section 3 briefly presents the efficacy of the fault selection method using the FIREMAN framework empowered by QARMA in a real world dataset. Section 4 explains the ongoing work of extending QARMA, to develop an online QARMA. Section 5 provides a summary of all the findings presented in this document, indicating the upcoming activities of Task 5.1.

2 Traffic Prediction using Hidden Markov Models in Binary Events

The term Fast Uplink (FU) grant was introduced in [1] to overcome the limitations of the current random access (RA) such as latency and collisions in machine type communications (MTC). A key step to apply FU allocation schemes is traffic prediction, where the traffic correlation behaviour of MTC devices (MTDs) enables the existence of traffic prediction and forecasting algorithms to anticipate the set of active and silent MTDs. In this section, we present a traffic prediction algorithm based on hidden Markov models (HMM). This work is related to the test case of the Smart Campus at University of Oulu, which is currently studied in WP6.

2.1 System Model

We consider an uplink model as depicted in Fig. 1, with K IoT devices and a single base station (BS). The BS can schedule up to L devices for transmission in each time instant, as in LTE fast uplink. The scheduled devices use the granted resources and transmit only if they are active, i.e. if they have data to transmit.

We denote the activation of device k in discrete time slots $t = 1, 2, \ldots$ by the random variable $A_t^{(k)}$, which is equal to one if the device is active and zero otherwise. The IoT devices activation vector at time t is given by $\mathbf{A}_t = \left\{A_t^{(1)}, ..., A_t^{(K)}\right\}$. We assume that the activation pattern of the devices are affected by N independent binary events, which are model as Markov processes.

The activation pattern of the devices is controlled by N independent two-state Markov processes. Each Markov process is characterized by On and Off states, which can be described as temporal transition probabilities

$$\Pr\left(\mathcal{S}_{t+1}^{(n)} = 0 \middle| \mathcal{S}_t^{(n)} = 1\right) = \epsilon_0^{(n)}, \tag{1}$$

$$\Pr\left(\mathcal{S}_{t+1}^{(n)} = 1 \middle| \mathcal{S}_t^{(n)} = 0\right) = \epsilon_1^{(n)}, \tag{2}$$

$$\Pr\left(\mathcal{S}_{t+1}^{(n)} = 0 \middle| \mathcal{S}_t^{(n)} = 0\right) = 1 - \epsilon_1^{(n)}, \tag{3}$$

$$\Pr\left(\mathcal{S}_{t+1}^{(n)} = 1 \middle| \mathcal{S}_t^{(n)} = 1\right) = 1 - \epsilon_0^{(n)}. \tag{4}$$

A device is active if any of the N Markov processes activates the device, so that the probability that device k is active at time t is

$$\Pr\left(A_t^{(k)} = 1 \middle| \mathbf{S}_t\right) = 1 - \bigcap_{n=1}^{N} \Pr\left(A_t^{(k)} = 0 \middle| \mathcal{S}_t^{(n)}\right) \tag{5}$$

$$= 1 - \prod_{n=1}^{N} (1 - q_{nk})^{\mathcal{S}_t^{(n)}}, \tag{6}$$

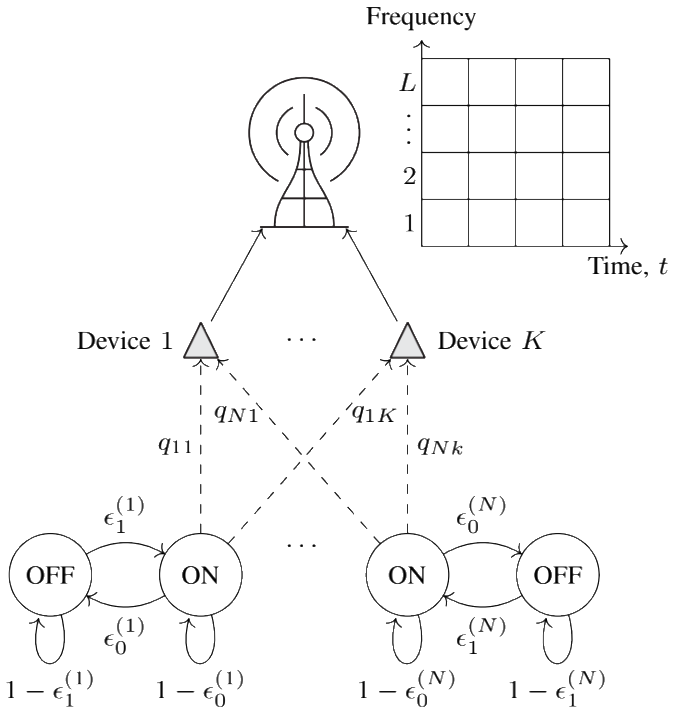

Figure 1: The considered activation model, in which N On-Off processes control the activation of K devices. If process n is in the On-state it activates device k with probability q_{nk}.

where we resorted to the fact that the activation is conditionally independent given the state vector $\mathbf{S}_t$ and q_{nk} refers to the probability that device k is active if the process n is in the On-state.

2.2 The Forward Algorithm

The BS needs to estimate the states of the Markov processes (On or Off) that affect the devices based on the observations. The activations observed by the base station can be described by an N-Hidden Markov Model (HMM) [2], where the total number of possible states at a certain time slot is 2^N. The BS exploits the forward algorithm to calculate the probability of the process being in a state and use the resulting state distribution to predict future device activation.

The forward algorithm computes the joint probability $p(\mathbf{S}_t, \mathbf{A}_t)$ efficiently by formulating the problem recursively [3]. In the notation of the given model, the forward algorithm can be formulated as

$$p(\mathbf{S}_t, \mathbf{A}_{1:t}) = p\left(\mathbf{A}_t|\mathbf{S}_t\right) \sum_{\mathbf{S}_{t-1}} p\left(\mathbf{S}_t|\mathbf{S}_{t-1}\right) p(\mathbf{S}_{t-1}, \mathbf{A}_{1:t-1}). \tag{7}$$

This joint probability distribution can be used for predicting the most likely hidden state as

$$\mathbf{S}_t^* = \arg\max_{\mathbf{S}_t} \; p(\mathbf{S}_t, \mathbf{A}_{1:t}). \tag{8}$$

The estimated hidden states at time instant t are used to predict the activation probabilities of each device using (5).

2.3 Simulation Results

In this section we present the simulation results of the proposed traffic prediction scheme using the forward algorithm. We compare the proposed scheme to random access (RA), where the devices request a transmission resources through a random selected resource (preamble), time division duplex (TDD), where the BS distributes the resources using round robin policy, and the genie-aided scheme, where the BS is assumed to perfectly knows the hidden states. The *regret* is one of the key metrics used to evaluate the performance of scheduling algorithms using learning schemes [1]. We define one unit of regret as wasting a resource on an inactive device, while there is one active device that did not receive a resource.

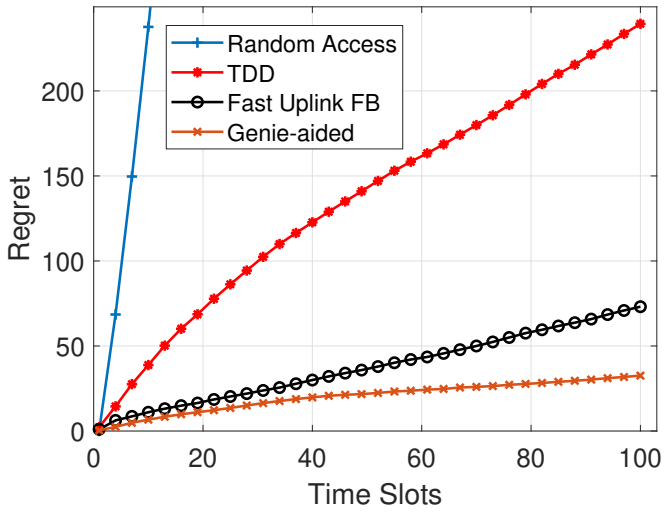

Figure 2: Regret evaluation for 5 events observed by 50 sensors competing for 10 transmission slots.

In Fig. 2, we compare the regret analysis of the proposed fast uplink scheme to other baseline schemes for 50 sensors competing for 10 transmission slots affected by 5 binary events. We can notice that the FU scheme significantly outperforms both RA and TDD. Specifically, when applying the proposed FU scheme, the regret function is reduced to 4 times less than the regret in case of TDD and 50 times less than the regret of RA due to the high number of collisions in RA. In addition, the proposed FU scheme results in a regret that is close to that of the genie-aided case, which reflects the high accuracy of the forward algorithm to estimate the hidden states.

3 Fault classification in power grids based on real occurrence

To test the procedure described in previous deliverable related to this task, namely D2.3[1], we used a real fault file from a transmission system located in Brazil, whose exact location cannot be disclosed. Real faults are usually gathered by fault recorders in $.cvg$ files, we used a algorithm to convert into matrices ($.mat$) for easier processing. Once the voltages and currents matrices are obtained, they can be injected in DM-DFT algorithm that yields the fault point and extracts the features as seen in Fig 3.

In real situations, faults can suddenly reappear for reasons as re-closure or reinsertion. This is the case on the CG-type fault we see in Fig 3. The algorithm detects successfully the first fault occurrence and also locate the exact sample where the phasors are extracted to perform selection. The NN and QARMA techniques were applied in the real fault data with a successful result: both correctly classify the fault as CG. Particularly with QARMA, it yielded 1100 rules that predicted the class of the fault, which resulted in the overall correct classification of the test case. One of the highest confidence rules for this test case was,

$$\begin{aligned} & [local\_voltage_A >= 235730.0266612848] \\ AND \;\; & [local\_voltage\_B >= 231130.0132737731] \\ AND \;\; & [remote\_ir >= 321.3212781340371] \\ \Rightarrow \; & [fault\_type = CG]. \end{aligned} \tag{9}$$

The support of this rule on the training set is 2.72% and it holds with confidence 100%.

Note that current implementations for real-time use cases, such as relay 21 in transmission lines, usually employ a full cycle of phasor estimation and around 4 ms of angle comparison between current/voltage components, as reported by some manufacturers. With either of the studied methods (DL or QARMA), once the model is generated based on historical data of the target system, the time taken to perform phase evaluation given a single-phase fault in the system is as small as 4 ms. Therefore, both methods can reliably select the faulty phase in the relay to make additional trip decisions. The algorithms described in Deliverable 2.3 that are commonly implemented in relays in operation today employ a full cycle Fourier phasor estimation for both the protection and the phase estimation. Because the phasor calculation is done in real time, it is based on the Fourier transform (or another filter with a similar output) that provides the root-mean-square (RMS) value required for the proposed phase estimation implying that only the phase selection itself has to be calculated.

Distance relays (ANSI 21) require 4 ms of angle comparison between current/voltage components, as reported by some manufacturers. The process to utilize the algorithms' outputs only requires multiplications and additional operations, making it more computationally efficient than most phasor operations and suitable for use in real-time applications. This scenario can be implemented in field following the schematic presented in Fig. 4, which indicates how the physical layer is mapped into the data layer to make a decision, in the way proposed in the deliverables related to WP2.

[1]For detaisl of the system model, please refer to Sec. 4.2 in Deliverable 2.3 `https://fireman-project.eu/attachments/article/10/FIREMAN_Deliverable_D2_3__FINAL_.pdf`

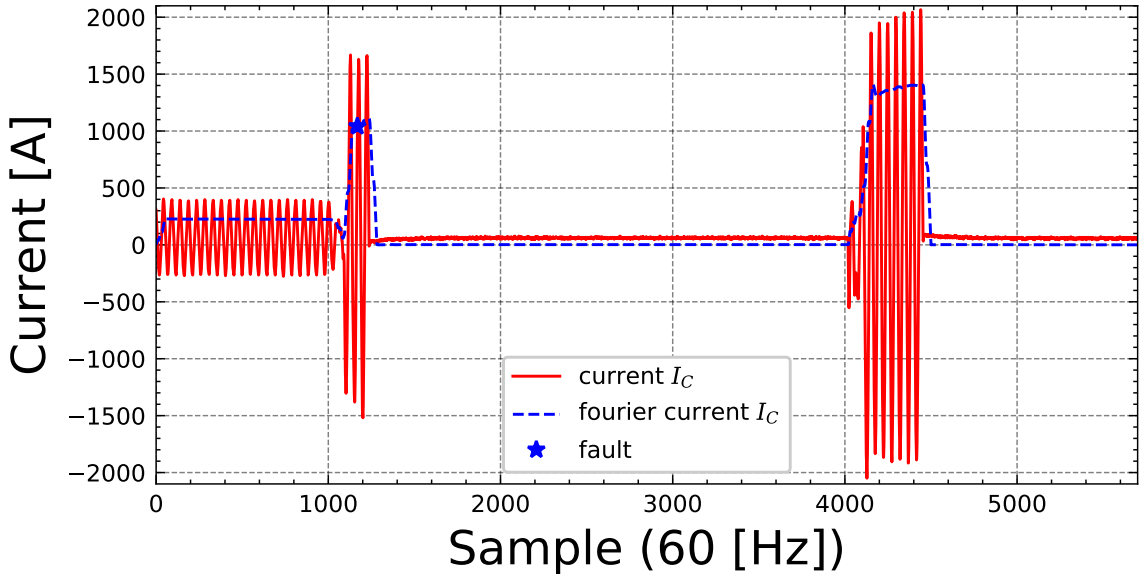

Figure 3: Phase C and DFT of real transmission system fault

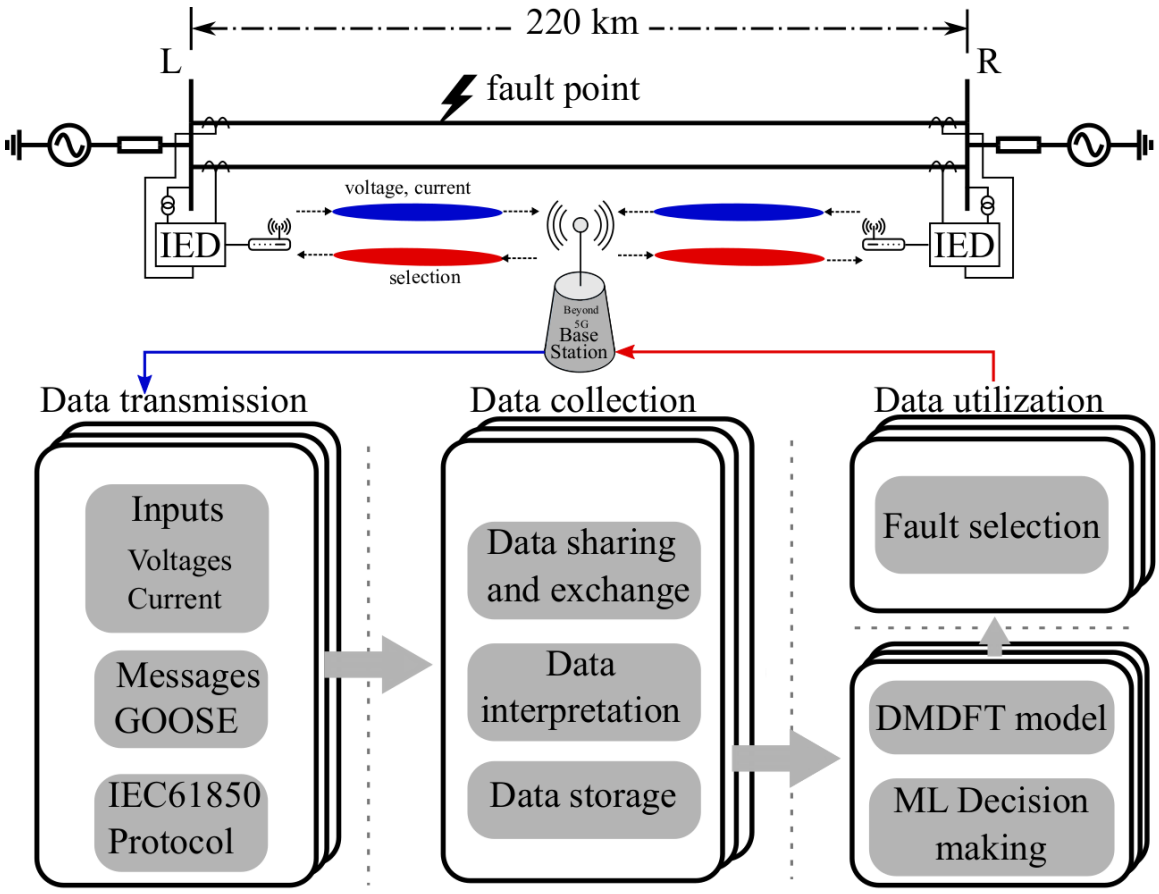

Figure 4: Schematic of the three layer model to be deployed for fault selection.

4 Fusing QARMA Results in Regression Problems

For classification problems, where we treat each different value that the target attribute T may take as a different label, we run the QARMA rule extraction algorithm with the request to produce rules of the form $I_1 \in [l_1, h_1] \wedge \ldots I_n \in [l_n, h_n] \rightarrow T = v$ and then, during decision-time, when a new instance is presented, we pick the top-performing rules and use the majority vote to decide the most likely value for the target attribute. However, for regression problems, where the target attribute t is considered a continuous variable that must be estimated from the data, we have found that a different procedure often yields much better results (notice that we use lower-case letter t to denote the target attribute for regression problems to make it clear that it is not a classification problem, where we denote the target attribute by upper-case letter T). The procedure first requires that we run the QARMA algorithm with the request to produce rules of the form:

- $I_1 \in [l_1, h_1] \wedge \ldots I_n \in [l_n, h_n] \rightarrow t \geq l$ and
- $I_1 \in [l_1, h_1] \wedge \ldots I_n \in [l_n, h_n] \rightarrow t \leq u$

The produced rules' consequent implications are visualized in Fig. 5 below.

Figure 5: QARMA produced rules' consequents visualization. Each rule predicts the target value to lie in a half-closed interval.

QARMA produces rules that upper- and lower-bound the value of our target variable t. Having all the valid, non-dominated upper-bounding and lower-bounding rules that hold on the dataset, we then fuse the results of the firing rules on a newly presented test-instance with the procedure that follows. Each firing rule r_i has associated weight (confidence or lift etc.) $w(r_i)$ and target value $v(r_i)$. The value decided by the rule ensemble is the average of the rule target values that maximize the sum of the weights of the rules that are satisfied by such target values.

Definition: Assuming that the target values $u_1, \ldots u_m$ and $l_1, \ldots l_n$ for the upper-bounding and lower-bounding firing rules are sorted in ascending order, the support of each such value is defined as follows:

- $\sigma(u_i) = \sum_{j=\min((k:u_k=u_i)}^{m} w(u_j) + \sum_{k=1}^{\max(j:l_j \leq u_i)} w(l_k)$

- $\sigma(l_i) = \sum_{j=1}^{\max((k:l_k=l_i)} w(l_j) + \sum_{k=\min(j:l_j \leq u_i)}^{m} w(u_k)$

Essentially therefore, the support value of a bound (lower or upper) computed by a rule, is the sum of weights of all rules in the ensemble for which the value is "supported", i.e. the value is within the half-closed interval they predict. Fig. 5 helps with a visualization of this concept: for any value x in the horizontal axis, the support of this value is the sum of the weights of the rules whose intervals (drawn as half-lines above the horizontal axis) cover the value x.

Having the value supports defined above, we define as the rule ensemble final regression estimate the quantity

$$\hat{T} = \frac{1}{N} \sum_{t \in \{l_1, \dots l_m\} \cup \{u_1, \dots u_m\}, \sigma(t) = \max(\sigma(u_i)), \sigma(l_i)} t$$

where $N = |\text{argmax}(\sigma(t) : t \in \{l_1, \dots l_n\} \cup \{u_1, \dots u_m\})|$, $i = 1, ..., m$ and $j = 1, ..., n$.

Computing Rule Target Value Support

The computation of all rule target value supports can be done in linear time in the size of the ruleset, exploiting the recursive nature of the equations defining rule target value support.

To do this, we first define the following quantities:

- $R_i = \sum_{j=\min(k:u_k=u_i}^{m} w(u_i), i = 1, ..., m$
- $Q_i = \sum_{k=1}^{\max(j:l_j \leq u_i)} w(l_k), i = 1, ..., m;$
- $r_i = \sum_{j=1}^{\max(k:l_k=l_i)} w(l_j), i = 1, ..., n;$
- $q_i = \sum_{\min(j:u_j \geq l_i}^{m} w(u_k), i = 1, ..., n;$

Having the quantities R_i, Q_i the quantities $\sigma(u_i)$ are instantly computed as $\sigma(u_i) = R_i + Q_i$ and having the quantities r_i, q_i the quantities $\sigma(l_i)$ are instantly computed as $\sigma(l_i) = r_i + q_i$

Computing the four quantities R_i, Q_i, r_i, q_i is easy using the following recursive relations:

- $R_j = R_{j+1} + \sum_{j=\min(k':u_{k'}=u_k)}^{m} w(u_k), j = m-1, ..., 1,\ R_m = \sum_{j=\min(k':u_{k'}=u_m}^{m} w(u_k)$
- $r_j = r_{j-1} + \sum_{k=j}^{\max(k':l_{k'}=l_i)} w(l_k), j = 2, ..., n;\ r_1 = \sum_{k=1}^{\max(k':l_{k'}=l_i)} w(l_k)$

- $Q_j = Q_{j-1} + \sum_{k=t_{j-1}+1}^{t_j} w(l_k), i = 1, ..., m$, $Q_1 = \sum_{1}^{t_1} w(l_k)$,
  where $t_j = \max(k : l_k \leq u_j), j = 1, ..., m$;

- $q_j = q_{j+1} + \sum_{k=p_j}^{p_{j+1}-1} w(u_k), i = 1, ..., n$; $q_n = \sum_{k=p_n}^{m} w(u_k)$,
  where $p_j = \min(j : u_k \geq l_j j = 1, ..., n$.

The formulae above can be implemented easily in single for-loops resulting in linear-time complexity.

Using the above scheme to estimate the target value in a number of regression problems encountered in industry over the past year, led to results that outperformed even deep learning models such as LSTM models applied to sequence data, at least for moderate dataset sizes that were provided (up to a few tens of thousands of rows, and a total of no more than 10-20 features).

4.1 Online QARMA

In addition to the above fusion algorithm for regression problems, we turn our our attention to an online- (also known as "continuous-") learning version of the QARMA algorithm, which is inspired from online versions of the classical Stochastic Gradient Descent method for neural network training. As has been shown, the QARMA family of algorithms is sound and complete in the sense that when it runs without any shortcuts, it is guaranteed to generate all non-dominated quantitative association rules that hold on the given training dataset. Updating these rules when new data instances arrive to reflect their support and confidence on the new augmented dataset can be done in constant time per rule per data instance (to verify this, see the code for the update of the rule measures below) and therefore for a rule-set of size $|R|$ and an update set of size $|O|$, the updates can be done in total time $O(|R||O|)$, but there is no longer any guarantee that the new rule-set is the same that would be produced if the QARMA algorithm runs on the entire new augmented dataset from scratch; in fact, the rule updates only guarantee the correctness of the support and confidence measures of the updated rules, but do not guarantee at all that the new rule-set contains no non-dominated rules, or that there aren't any other rules (not contained in the set) that are valid; these other rules could even dominate some of the updated rules (but shouldn't dominate any rules in the rule-set that remain unchanged after the online updates.)

Let D be an initial labeled training dataset, and O be a set of online data instances to be used to update the produced rules $R(D)$. Let the validation dataset be denoted by V. The online version of the QARMA algorithm is formally described in Figure 6.

Once the new data have been used to update the rule-set $R(D)$, performance can be measured on the validation set V using the fusion algorithm of section 4. It is worth noting that the main for-loop in the algorithm 6 can run fully parallel because the updates of one rule are not affected at all by the updates of any other rule; therefore the specific for-loop is "embarrassingly parallel". Even further, another fully (embarassingly) parallel for-loop after

step 4 that checks if any rule $r \in R(D \cup O)$ is dominated by any rule $r' \in R(D \cup O) - \{r\}$ and if so deletes r from $R(D \cup O)$ runs in time quadratic in the size of the rule-set $R(D \cup O)$ and guarantees that all remaining rules are consistent (meet minimum support and confidence threshold criteria on the new expanded dataset) and in addition are all non-dominated to each other.

4.2 Online QARMA Computational Results

In this section, we present a set of rather promising preliminary results of the approach described above for the online learning version of the QARMA algorithms and their corresponding ensemble fusion.

We first note that the grid-fault problem described in section 3 is rather amenable to the online-version of QARMA. We first ran QARMA with half the size of the original training set, then used the other half of the training set to update the extracted rules' support and confidence, and then tested the accuracy of the resulting rule-set on the same validation dataset as before. The accuracy only dropped from 98% to 95%. For reference, when testing before the online rule updates took place results in an accuracy of around 85%, so the online updates work as expected and are very useful in the decision making process.

The grid-fault problem described above is a classification problem. Below we describe three different problems (datasets coming from Open Call pilots of the QU4LITY H2020 project where one of the authors is an active researcher) that are regression problems where the ensemble fusion approach of section 4 applies perfectly.

4.2.1 Computing Body Measurements' Matching Scores to MorphoTypes in the Fashion Industry

We test on two real-world datasets (gardenia experiment) where the data represent body measurements of various people, and check their conformance on a particular "morphotype" to be used in the creation of a suit or dress (for males or females respectively) for the fashion industry. For the female dataset, when QARMA runs on the full training dataset, and then is tested on the validation dataset the test RMSE value is 3.76, the RAE% (Relative Absolute Error) is 5.69% and the Mean Absolute Error MAE value is 1.76; these values serve as baseline for the performance of online-QARMA. When QARMA is trained only on half the size of the initial training set, and then accuracy is measured again on the same validation dataset, the RMSE value is 7.15, RAE% value is 9.11% and MAE is 2.9; these values are higher than before, as we expect, since the rules were extracted using only half the original training set. However, when we update the previously found rules' support and confidence according to the data in the second half of the original training set, the results rules' ensemble fusion results in an RMSE value of 3.95, RAE% of 4.81% and MAE value of 1.53 (better than the fully trained version of QARMA!)

For the male dataset, when QARMA runs on the full training dataset, and is then tested on the validation dataset, the test RMSE value is 15.41, the RAE% value is 33.66%, and the MAE value is 10.03 (much higher than the female dataset values, probably due to less features available in the dataset). When QARMA is trained only on half the size of the initial training set, and then accuracy is measured again on the same validation dataset, the RMSE

value is 17.4, RAE% value is 38.63% and MAE is 11.62; again, these values are higher than before, as we expect, since the rules were extracted using only half the original training set. However, when we update the previously found rules' support and confidence according to the data in the second half of the original training set, the results rules' ensemble fusion results in an RMSE value of 15.96, RAE% of 34.64% and MAE value of 10.42 (much closer to the fully trained version of QARMA!)

4.2.2 Computing Wear-and-Tear of Machines

We test the ability of the online version of QARMA to predict a target variable (continuous real number) that represents the wear-and-tear of a machine in a production line, given data from sensors attached to the machine, including temperature readings and displacement values. For this dataset, when QARMA runs on the first 150 thousand lines of the training dataset, and is then tested on the validation dataset, the test RMSE value is 210.22, the RAE% value is 3.28%, and the MAE value is 27.46. After the online-version of QARMA has updated the previously found rules using the next 50 thousand rules of the training dataset, the resulting RMSE value (on the same validation set) is 210.21, RAE% becomes 3.27 and MAE becomes 27.45. All metrics improve (though slightly). However, when we update the previously found rules' support and confidence according to the data in the next 100 thousand lines of the original training set, the results rules' ensemble fusion results in an RMSE value of 209.62, RAE% becomes 1.83% and MAE becomes 16.69 which represents a serious reduction of error according to all metrics used.

4.3 Final remarks on QARMA

We have described a novel method for fusing the results of QARMA extracted rules, particularly suited to regression problems. We have also ran preliminary experiments on medium-scale to large-scale datasets that show that our proposed version of online-updating of QARMA extracted rules works and produces very good results, comparable to the ones obtained when the entire dataset is considered in the rule extraction phase. The speed-up that is achieved from this approach remains to be measured, but it seems to be very significant, without any serious loss of accuracy.

5 Conclusions

This document presented initial results related to the activities of Task 5.1. We have focused on three main aspects, namely: (i) traffic prediction related to Fast Uplink grant that used to enable machine-type communications to be employed in one of the testbeds (Smart Campus) in WP6, (ii) results of (traditional) QARMA of fault selection based on real data of power line fault, and (iii) further extension of QARMA and related results. For the next periods, we plan to extend (i) and (iii) towards the specifics of the tedstbed of WP6.

Algorithm Online QARMA

Inputs: initial dataset D, user specified minimum support and confidence thresholds mn_s, mn_c, online (labeled) dataset O, update function $update(r, O)$, support and confidence returning functions $supp(r, D), conf(r, D)$.

Begin

1. **set** $R(D) \leftarrow QARMA(D)$.

2. **for-each** $r \in R(D)$:

2.1. **set** $r \leftarrow update(r, O)$.

2.2. **if** $supp(r, D \cup O) < mn_s$ **OR** $conf(r, D \cup O) < mn_c$ **then set** $R(D) \leftarrow R(D) - \{r\}$.

3. **end-for**.

4. **set** $R(D \cup O) \leftarrow R(D)$

5. save $R(D \cup O)$ in the database.

End

Figure 6: QARMA Online Version

References

[1] S. Ali, N. Rajatheva, and W. Saad, "Fast uplink grant for machine type communications: Challenges and opportunities," *IEEE Communications Magazine*, vol. 57, no. 3, pp. 97–103, March 2019.

[2] O. Capp, E. Moulines, and T. Ryden, *Inference in Hidden Markov Models*. Springer Publishing Company, Incorporated, 2010.

[3] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, Feb 1989.