MODULAR FRAMEWORK TO MODEL CRITICAL EVENTS IN STROKE PATIENTS

Kevin Henares
José L. Risco-Martín
Román Hermida

Dept. of Computer Architecture and Automation
Complutense University of Madrid
Calle Prof. José García Santesmases, 9
28040 Madrid, Spain
{khenares, jlrisco, jayala, rhermida}@ucm.es

Gemma Reig Roselló

Stroke Care Unit
Hospital Universitario de La Princesa Madrid
Calle de Diego de León, 62
28006 Madrid, Spain
gemma.reig@salud.madrid.org

Román Cárdenas

Laboratorio de Sistemas Integrados (LSI)
Universidad Politécnica de Madrid
ETSI Telecomunicación, Avenida Complutense 30
28040 Madrid, Spain
roman.cardenas.rodriguez@alumnos.upm.es

ABSTRACT

Several of the major causes of death in the world are related to neurological diseases, like strokes, sclerosis, or Parkinson's. As a consequence, extraordinary amounts of clinical information are collected. With proper Modeling and Simulation (MS) techniques, predictive models can be defined to help physicians in their diagnoses. We have come to the conclusion that the implementation of an abstract MS methodology to facilitate the processing and modeling of the collected data would be of great help. In this paper, we focus on the development of such an abstract framework, mainly aimed at facilitating and automating the process of data collection and predictive and diagnose models. As a use case, we show how this methodology is applied to determine the stroke type and exitus (i.e. probability of death) of stroke patients in the early stages of their episodes. The best models are evaluated and constantly updated to generate these predictions.

Keywords: Health, Diagnosis, Prediction, DEVS, Methodology

1 INTRODUCTION AND RELATED WORK

Neurological disorders are an important cause of mortality and constitute 12% of total deaths, reaching 16.8% in low-middle-income countries (WHO 2018). Among them, cerebrovascular diseases represent the 85% of the deaths.

Data related to the diagnosis, treatment and consequences of these disorders are collected in many different ways. Some of them are the diagnoses performed by medical staff, patients monitoring, surveys and receipts records. This information can be combined and processed to generate models that reproduce the trends and behaviors of these diseases. In this way, models can be generated to determine the most effective treatments,

to predict critical events or to estimate the outcomes of a certain disease. Hence, these models can be used by the medical staff as a support tool in decision-making. For instance, (Khosla et al. 2010) generates and compares predictions of stroke attacks using several machine learning algorithms. In (Thirumalai et al. 2017) a system for decision making on heart attacks using Pearson models is presented. (Pagán et al. 2015) generates migraine per-patient prediction models using the N4SID algorithm over ambulatory data.

However, the format of the collected data usually lacks normalization. As a consequence, it is common to find databases composed of multiple and heterogeneous datasets. Therefore, to correctly analyze and study these datasets, processes of standardization and normalization are required. This situation has been pointed out by several authors in the literature, proposing standardization methods and normalizing and grouping specific heterogeneous data sources (Kimura et al. 2010) (Low et al. 2015).

Once all the heterogeneous data sources are processed and unified, the resulting information can be adapted and used to generate specific models and systems. For that, different machine learning techniques can be applied, including kernel methods, classification, regression, and neural networks. Despite the number of variants, there is a well-known and widely used subset of machine learning techniques. They are usually present in the literature and implemented in a variety of machine learning software and libraries (Witten et al. 1999) (Pedregosa et al. 2011). When applying these techniques to a specific use case, several machine learning algorithms are chosen based on the data types and characteristics of the problem to select the ones with better performance. This process of training, validating, evaluating and comparing different sets of models is usually a repetitive and time-consuming task. By applying appropriate techniques, this whole process can be automated, allowing better control and analysis of the parameters involved in each model.

Our research group has recently achieved several advances in the M&S of neurological diseases. Some examples are the modeling and prediction of migraine pain episodes (Pagán et al. 2015), as well as the classification of primary progressive aphasia patients based on the study of FDG-PET imaging (Matias-Guiu et al. 2018). Nevertheless, these predictive models are created ad-hoc, without automatic control mechanisms to prevent common mistakes in data formatting and model generation. Different sets of models were independently generated, validated and evaluated. Subsequently, the best models were selected comparing their scores. This complex modeling work-flow can be simplified with the creation of an specific, abstract, and automatic M&S methodology. This would simplify and facilitate the processing and modeling of the data and reduce the time spent on these tasks.

This paper presents a methodology that encapsulates the main filtering, processing and modeling operations. This methodology has been tackled from an abstract perspective, in order to be used against several neurological disorders. In its current implementation, specific modules are coded to encapsulate these operations. In this way, the specific processing and modeling tasks are achieved by instantiating and parameterizing the suitable models and linking them adequately. This modular behavior, in addition to the predefined modules, allows an straight-forward specification of all the subsystems involved in the process. Due to this modular methodology, the resulting systems can easily be adapted by simply altering their parameters.

For the definition of such abstract M&S methodology, it is highly convenient the use of an M&S formalism. This facilitates the creation of abstract models and the automatic control of the actions performed by the user. There exist several validated and widely used M&S formalisms, such as Petri Nets, Timed Automata and Discrete Event System Specification (DEVS) (Zeigler et al. 2000). All that formalisms are used as a way to model, verify and validate systems and support the specification of distributed systems. However, both Petri Nets and Timed Automata models are not able to deal with complex data types. In the implementation of the methodology, DEVS is used, as it has a modular nature, allows a complete integration to the physical part of a cyber physical system, and the combination of discrete and continuous subsystems can be performed on a straightforward way.

This paper is organized as follows: the methodology is presented in Section 2. A use case that follows this methodology and generates initial diagnosis for stroke crises is presented in Section 3. Finally, conclusions are drawn in Section 4.

2 METHODOLOGY

In this section the presented M&S methodology is explained. Its main ideas are presented, and some advantages are discussed.

The goal of the methodology is the automatic creation of predictive models. Figure 1 shows the common phases followed in this process. First, data from one or several data sources are collected (`Data sources` element in Figure 1). These data are initially filtered following some common criteria, to assure their quality and discard wrong or incomplete records. Also, feature selection procedures can be performed in this stage if necessary. Both tasks are performed in the `Filter` module. Second, normalization operations are applied into the `Normalize` module according to the specific needs of the use case. This stage is specially important when heterogeneous data sources are taken into account. It includes unifying the units of the variables, categorization tasks and adjustments in the implied values or distributions. At the end of this stage, all the sources have to be unified in a single format. Hence, each data source may need different processing to convert their data to this common view. Third, some transformations may have to be applied in the pre-processing stage to adapt the data format to the use case (`Pre-Process` module in Figure 1). For example, in the health domain, single samples coming from a patient monitoring device can be too granular to be directly related with the event to predict. In this situation it is common to split clinical data into time-windows of a predefined size so that a certain number of consecutive samples matches with the class to predict. Next, the `Pre-Process` module separates the input data in several datasets: train datasets, that are used to fit the model, and validation/test datasets, that are used to adjust the models and select the best ones. Once the separation is done, different sets of models are generated with the train dataset using the suitable machine learning algorithms in the `Train` module. The set of algorithms to use depends as much on the problem characteristics as on the data types. They can include classification and regression algorithms, neural networks and kernel methods, among others. Once trained, models are verified and compared using the previously separated validation/test datasets in the `Validation/Test` module. The selected models will be subsequently evaluated to generate valuable information of the area to be modeled. New samples can be used both to evaluate the models and to feed the data sources. This allows us to generate new models periodically, so that its accuracy can be increased over time.

In the presented M&S methodology, the main operations performed in each one of the stages are encapsulated and implemented in the set of modules described above. Each of these modules have a set of parameters that can be modified to adapt the transformations performed according to he nature of both the input data and the predictive models. In this way, the processing of the data goes through a chain of modules, starting from the data sources and finishing with the generation and evaluation of the models, as Figure 1 depicts. Hence, the design and implementation of the proposed methodology includes a set of instances of these modules (with the suitable parameters) and a set of couplings. Within a DEVS context, these couplings link output and input ports. These ports represent the input/output points of the modules.

All the stages involved in the data processing and model definition depicted in Figure 1 are configured with metadata related to the corresponding dataset, represented at the top of the Figure. This includes datasets and modules identifiers, specific configurations established for each of the modules to adjust the transformations performed in them and custom tags reflected in the specification.Through these metadata, which is global information, all the modules of the system are aware of all the transformations made in each dataset. Following this idea, modules can auto-adapt their operations based on the transformations performed by previous modules and other metadata (or even request configuration changes in other modules).

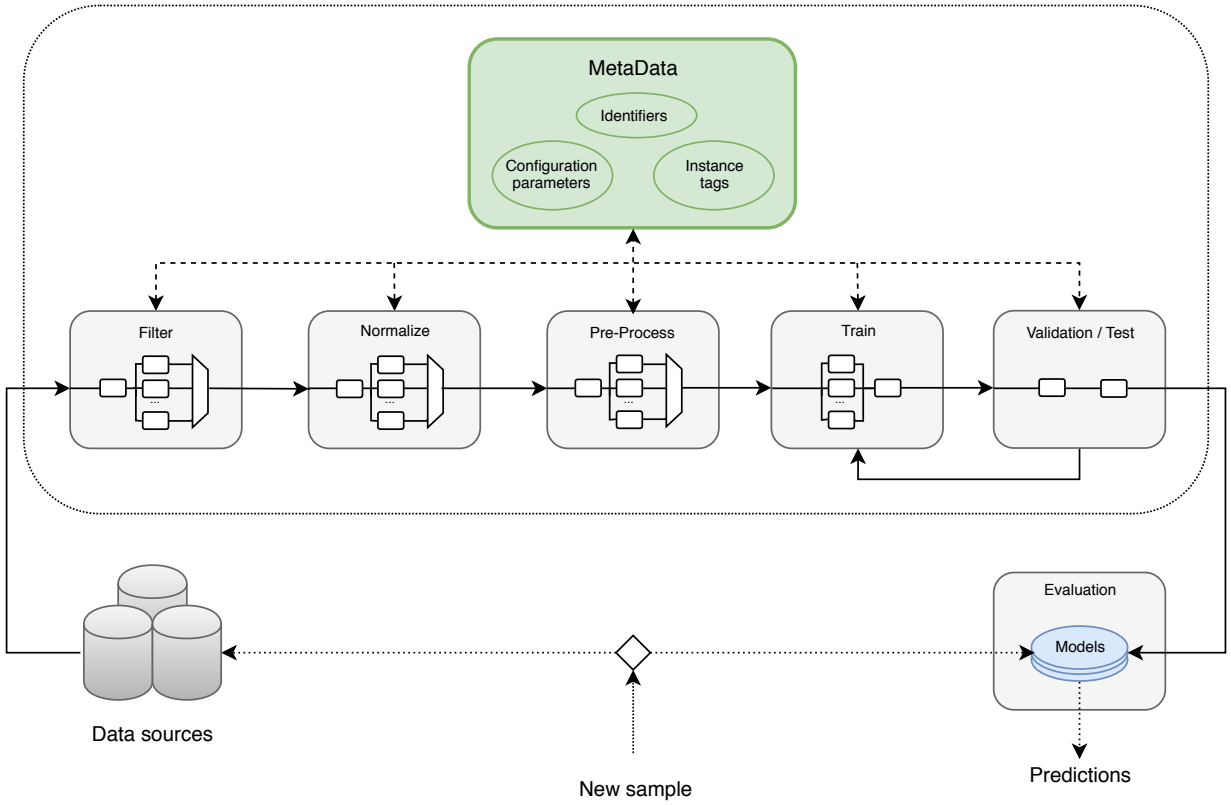

Figure 1: Global view of the methodology.

As a result, this modular methodology favors the reusability of the systems and modules. It also facilitates the creation of subsystems and, therefore, the generation of distributed systems, for scalability purposes. Furthermore, the main techniques and procedures are encapsulated in predefined modules, allowing us to specify using configuration files the structure and operation of the system, instead of directly coding its behavior.

In particular, the definition of this M&S framework for a given neurological disease is performed through the support of XML files. These files include the structure and behavior of the abstract M&S method illustrated in Figure 1, but focused on a particular disease, with its features and objectives. We have developed a framework that parses these XML files, and automatically generates the set of DEVS models that describes the behavior depicted in Figure 1, but for the specific disease. All the information propagated through the whole system is encapsulated into a special custom structure called *MetaFrame*. It contains both a table of data and metadata accumulated by all the modules included in the datapath.

3 USE CASE

In this section the methodology previously explained is used to address the specification of a stroke prediction system. First, a description of the data sources and flows is presented. Next, the architecture of the different subsystems that compose the case use is described.

3.1 General Description

For this prediction system, a dataset obtained from the Stroke Care Unit of the Hospital Universitario de la Princesa (Madrid, Spain) is used. It includes information about 118 stroke patients, collected from March 2017 to July 2017. Of them, 104 correspond to ischemic strokes and the remaining 14 to hemorrhagic strokes (which accomplishes with the usual proportion of ischemic and hemorrhagic strokes). Following the general scheme provided in Figure 1, two models will be automatically generated: one to diagnose the stroke type and the other one to predict the *exitus* risk, both in the early stages after the episode. Therefore, these predictions are intended to be generated when monitoring new stroke attacks, before conclusive tests have been done. In this way, these predictive models could be used in ambulances, during the patient transportation from an rural area to his reference hospital, or upon arrival at the health center, and serve as a first diagnosis to facilitate the treatment of patients.

Stroke is the third most common cause of death in developed countries, exceeded only by Coronary Heart Disease (CHD) and cancer (WHO 2018). Annually, fifteen million people worldwide suffer a stroke. Survivors can experience loss of vision and/or speech, paralysis, and confusion, among others (WHO 2018). There exist two main types of strokes: ischemic (87%) and hemorrhagic. The Stroke Alliance For Europe (SAFE) estimated the total stroke cost in 2015 at €45 billion. Multiple techniques are used to diagnose stroke. However, they are usually complex and are conducted by the medical staff. Hence, they are not accessible everywhere. This is specially important in non-urban areas. Moreover, for stroke diseases time is critical (Hill, Michael D 2005). These special situations make necessary to find alternatives that allow the generation of accurate diagnoses without incurring a significant delay in the treatment. We propose a system that generate predictions based on different sets of models, generated using several Machine Learning algorithms.

In the following, we explain how data are collected in the Stroke Care Unit of the Hospital Universitario de la Princesa. Next, we define the particular specification of the framework provided in Figure 1. With these specifications, the framework will be able to automatically generate the predictive models mentioned above.

A specific view of the general methodology workflow (Figure 1) adapted to this use case is shown in Figure 2. This system centralizes the information in a database. It is used both to store the data generated by the data sources (Monitoring Units (MU) and diagnoses of the medical staff) and to support the intermediate data and the models generated by the subsystems. These MUs are associated with each one of the hospital stretchers and includes a Philips IntelliVue Information Center iX (PIIC iX). This device collects information related to ECG signal, perfusion, respiratory rate, and oxygen saturation, and sends it to the database (labeled as DB in Figure 2). When the diagnosis and the resulting status of the patients are clear, the medical staff introduce additional information into the database. These data are associated with each stroke event and include variables like the recurrence, the stroke type, and the *exitus*. This categorization is used to train predictive models that will be used as references in the early stages of stroke events suffered by other patients.

Following the general view illustrated in Figure 1, this particular system needs five of the six subsystems to separate the different stages of the processing. In this use case, the module of data normalization techniques was not considered necessary due to the use of a single data source and the intrinsic characteristics of the input data.

First, both the Filter & Pre-Process systems (labeled with the same names but as a single block in Figure 2) discards the episodes that do not accomplish several fixed criteria to assure the quality of input data. Some corrections and transformations are also applied in this stage to adapt the original data to the next procedures. Second, the Training System generates different models that can be used to generate predictions. It considers different variations in the input parameters and uses different well-known classification algorithms. These

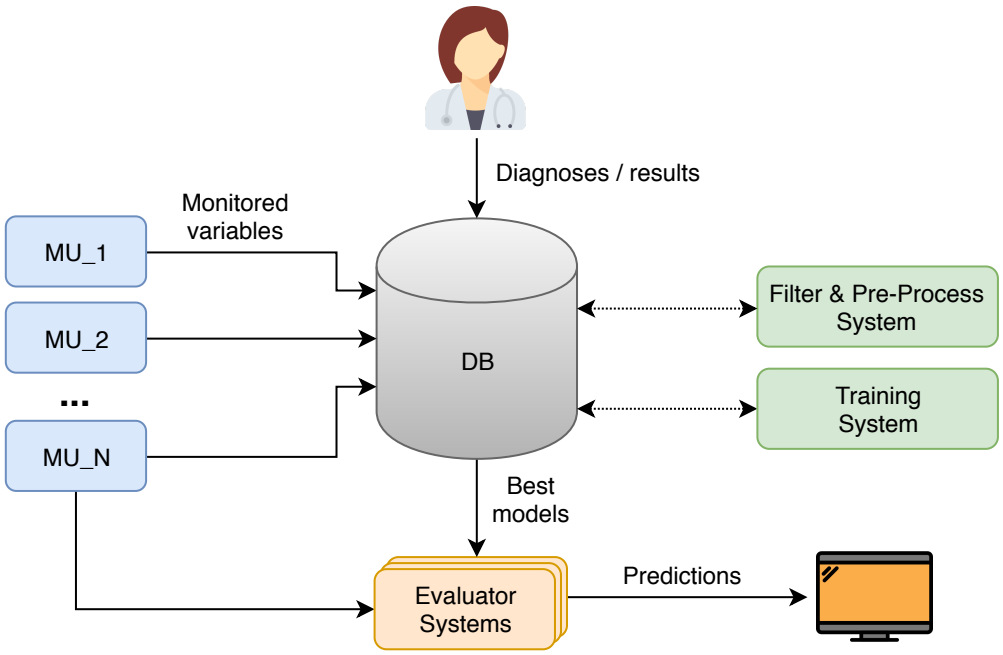

Figure 2: Top-View of the prediction system. Both the monitoring units and the medical staff add the information to the system. The DEVS-based subsystems preprocess it, generate the models and the predictions, according to the general scheme depicted in Figure 1.

models are uploaded to the database, including information about their fitness and error. Finally, the Evaluator Systems load the models with the best accuracy from the common database and generate predictions. An Evaluator System can be deployed for instance into an ambulance monitoring equipment or in a patient stretcher. The resulting predictions are sent to the medical staff and may be used to facilitate the diagnosis and treatment of the stroke events.

In order to manage all the information, several intermediate database tables are automatically created (represented in Figure 3). First, the Filter & Pre-Process system loads the *Raw data* table, that contains the data gathered by the Information Centers. It adequately formats them and recovers damaged signals (if any). Then, it saves the results in several intermediate tables (*Processed data*). Finally, the Training System combines these tables with the diagnoses of the medical staff, also stored in the database (*Diagnoses* table), to generate both stroke type and *exitus* sets of predictive models.

Figure 4 shows the structure of the Filter & Pre-Process system more in detail. This module loads the original data generated by the MUs and splits them so that a *MetaFrame* is generated for each patient. Then, invalid patients are discarded using the `Filter` module. A patient is considered to be invalid when it does not have at least 45 minutes of monitoring data or when the percentage of null values in some of the variables of the first 45 minutes exceeds the 20% of the total. On the used dataset, 14 of the 118 patients are discarded in accordance with the filtering rules. After the filter, the first 45 minutes of each valid patient's data are selected and they are re-directed to the `Filler` module of Figure 4. In it, null values that appear in the monitoring variables are filled based on previous and next valid data.

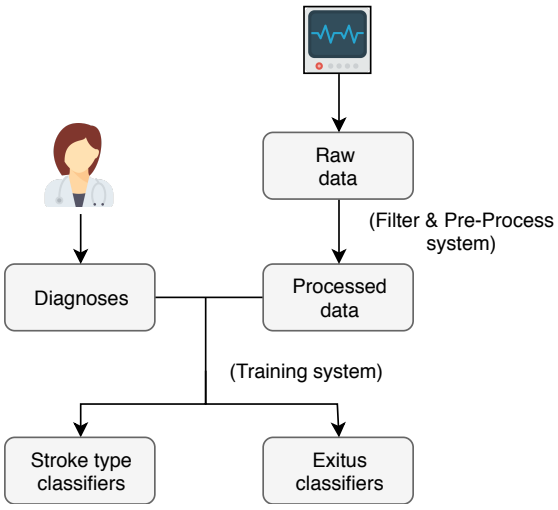

Figure 3: Tables implied in the database of the system.

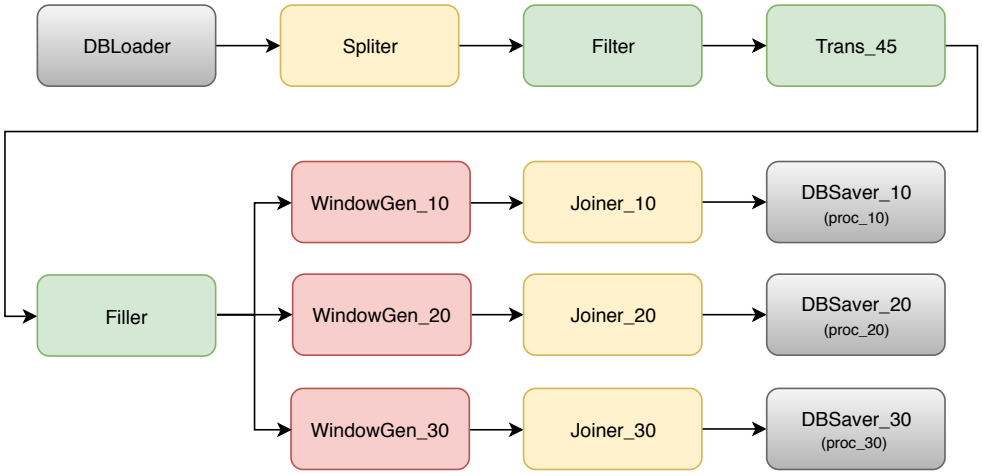

Figure 4: Filter & Pre-Process system. It pre-process the monitoring data and repairs possible failures.

3.2 Technical Process

Next, as Figure 4 shows, each resulting *MetaFrame* goes through three `WindowGen` modules. These modules group the patient's data using windows. An example of this process is illustrated in Figure 5, where a generic window structure of size 3 is created. That means that each row of the resulting structure is composed of three samples of the original one and the class related to the last one. In the case of the presented system, the variables on the left are the periodic measurements of a specific patient and the class (in the right) is the stroke type or the *exitus* value. Each one of the `WindowGen` modules is configured to group the packet with a different window length (10, 20 and 30 samples per row). The use of these structures is due to the data nature. Each monitoring sample is too granular to perform predictions separately, but a consecutive set of samples can represent a significant trend that can be used to recognize patterns. After that, the resulting structures are joined in the `Joiner` module to group the data relative to each patient in new *MetaFrames*.

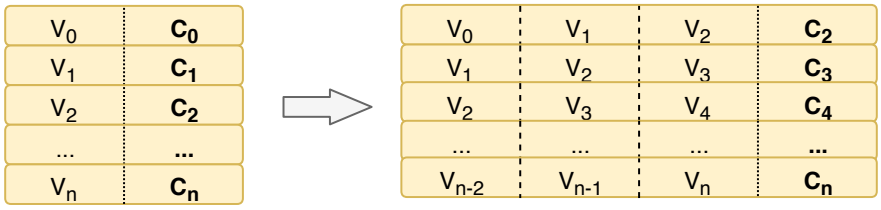

Figure 5: Example of grouping information using windows.

Finally, that information is saved in three different tables of the central database (one for each window length). It is worth to mention that our framework generates several predictive models in parallel (using different algorithms, different window sizes, cross-validation type, etc). Thus, all data are automatically pre-processed in all the needed formats to avoid the execution of this procedure each time that new models are being created. That subsystem is executed every 12 hours, to check if there are new available data buffered to process. After several tests, we have concluded that 12 hours is enough to gather new relevant data and update existing models or generate new ones.

Details of the Training System are depicted in Figure 6. This module is in charge of generating updated models periodically. Is executed every 24 hours and replace all the models stored in the database if new data are available (i.e. when the Filter & Pre-Process system adds new data to the intermediate tables or the medical staff tag previous episodes in the *Diagnoses* table). It also sends pulses to all the active Evaluator Systems to update their models if needed.

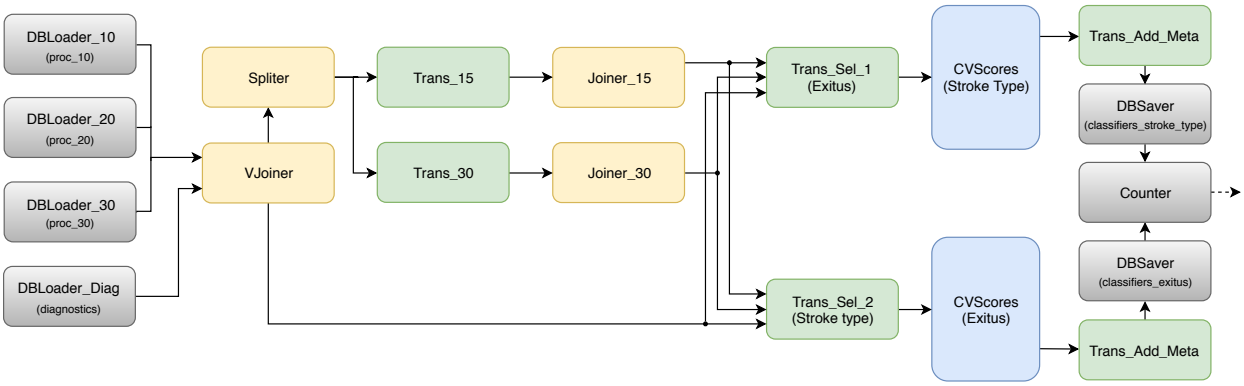

Figure 6: Training System. It generates models based on the processed patients info generated by the Filter & Pre-Process system.

This system starts loading the pre-processed data generated for the Filter & Pre-Process system. Each one of the three resulting *MetaFrames* is joined with the information present in the *Diagnoses* table, using a `VJoiner` module. With that, all the tagged episodes are recovered. The patient's data that have not been still tagged by the medical staff are discarded in this module.

Next, each one of the *MetaFrames* is used to generate three new ones: one taking into account the first 15 minutes of monitoring, a second that uses the first 30 minutes and a third one that uses the first 45 minutes. To this end, a `Splitter` module is used to separate the patients and two `Transformers` are used to cut the information in pieces of 15 and 30 minutes. The group of 45 minutes does not need to be cut because it was already saved with this length in the Filter & Pre-Process system.

After that, the resulting structures are joined in the `Joiner` module to group the data relative to each patient in new *MetaFrames*. Finally, that information is saved in three different tables of the central database (one

for each window length). As above, it is saved in this pre-processed format to avoid executing this procedure each time that new models are being created. That subsystem is executed once each 12 hours, to check if there are new available valid data to process.

At this point, nine packets are generated combining three different window sizes and three sample lengths. Each one of them is used to create several custom prediction models, using different well-known classification algorithms. For doing that, they are injected into two models in charge of selecting the classes of the data (one of them selects the stroke type, and the other the *exitus* risk). Each output goes to the corresponding `CvScores` module. These coupled components generate sets of models using several classifiers included in the *sklearn* Python library. Specifically, the following Machine Learning algorithms are used: *Ada Boost*, *Bagging*, *Extra Tree*, *Decision Tree*, *Gradient Boosting* and *Random Forest*. As output, a *MetaFrame* is obtained containing the name of each classifier, the generated models, its score and its error. The score corresponds to the average fit values obtained in a 5-fold cross validation process. Next, this information is tagged with the window size and the monitoring length used in each combination (using a `Transformer` module and getting this information from the metadata of the packets). Finally, the models are saved in two tables of the central database (*classifiers_stroke_type* and *classifiers_exitus*) to be used by the Evaluator System. The `Counter` is used to notify the Evaluator Systems of the existence of updated models. This is done by generating and sending a pulse after the 9 sets of models have been uploaded.

The Evaluator Systems are the ones in charge of loading the most suitable models from the database and use them to generate predictions about the stroke type and the *exitus* of new stroke attacks, evaluating the models with the real-time monitoring of the patient. The structure of this system can be seen in Figure 7. It is centered in two `DBClassifiers` modules, that start loading the best models among the ones trained with samples of 15 minutes. This model will be updated when one of these two conditions happen: (i) it arrives a pulse from the Training system, indicating that the models have been updated, or (ii) the `Counter` modules detects that enough data have been collected to use another set of models. This occurs when the monitoring device collects data for 30 and 45 minutes. When the model has been updated, the `DBClassifiers` modules generate a *MetaFrame* with the information of the new model. This packet is modified with a `Transformer` module to generate a special packet that alters the window length used in the corresponding `WindowBuffer` module. These modules are responsible of generating the appropriate packets to be evaluated in the models, grouping the input data using windows of the same length of the ones used in training.

The input of that system is generated by a `StreamLoader` module, that simulates the operation of the monitoring units. In this way, each output packet contains only one value for each measured variable (as they were measured in real-time). In the following `Transformer` the suitable variables are selected (the ones that are used in the training phase). The `Logger` module receives the predictions generated by the two `DBClassifiers` and show them so that it can be seen by the medical staff.

As a result of the execution of the system several models are generated and stored in the database. Specifically, 54 models are generated to diagnose the stroke type (Figure 8) and other 54 models correspond to *exitus* predictions (Figure 9). The fit score of each one of them, generated with different combinations of sample size in minutes (m) and window lengths (s), can be seen in both Figures. With regard to stroke type predictions, the best fit (0.824 ± 0.001060) is achieved by a *Gradient Boosting* model with a sample size of 30 minutes and a window length of 30 samples. The best *exitus* prediction model corresponds to a *Random Forest* with a sample size of 45 minutes and a window length of 30 samples, and has a fit of 0.936 ± 0.000342.

It is worth to mention that the whole framework is automatically generated by properly filling the `MetaData` information shown in Figure 1, leading to the M&S framework described through Figures 2, 4, 6 and 7. Our hypothesis is that the presented abstract methodology can be used to provide support to predict

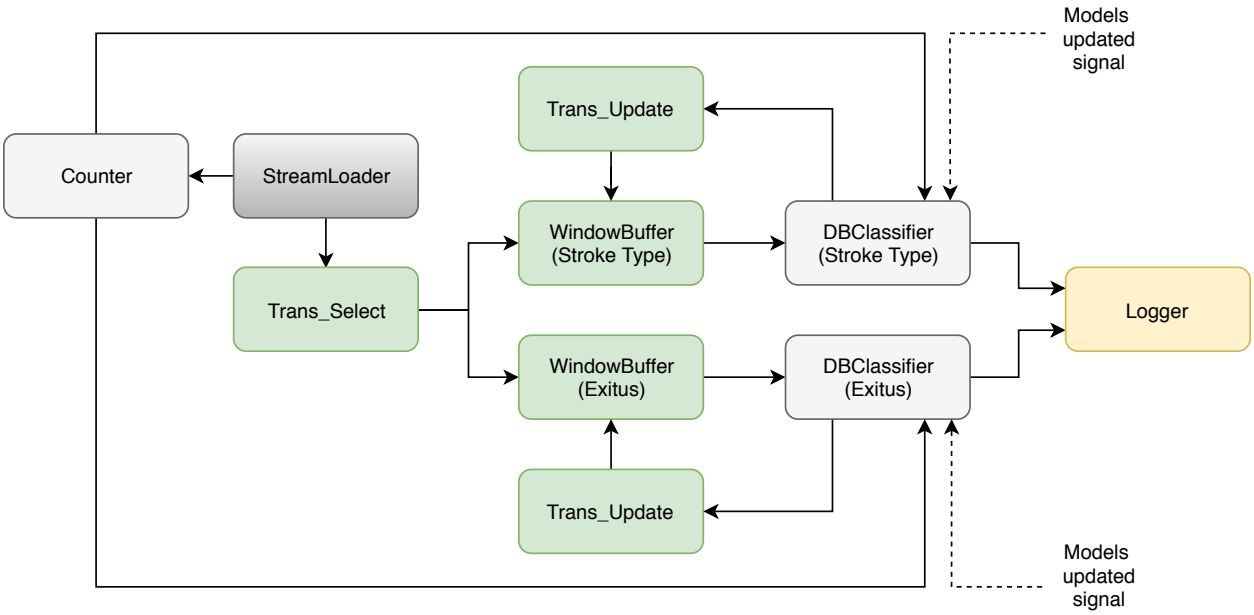

Figure 7: Evaluator system. It downloads and evaluates the best models and display the predictions.

future risks in several neurological diseases, or other clinical scenarios where the patient is continuously being monitored.

4 CONCLUSIONS

This paper presents a methodology to facilitate the generation of predictive models in clinical diseases. It is based on the specification of an abstract set of modules and the couplings between them. These modules are predefined and encapsulate the main operations needed for the automatic generation of predictive models. Moreover, each one of them contains a set of parameters that can be tailored to adapt its operation to a particular use case. The resulting systems can provide the medical staff with additional tools able to predict outcomes in particular crises or events. This information can be used as a reference prior to the completion of definitive tests or be part of more complex Decision Support Systems.

This methodology has been applied in the creation of a prediction system able to categorize the stroke type and the *exitus* risk in early stages of the monitoring of stroke events. The system initially creates several models with the clinical data stored in a database, as well as the diagnoses of the medical staff. Once the models are defined, they are periodically updated and improved when new information is stored in the database, i.e., new clinical data is loaded through the monitoring devices or additional diagnoses are added. From this point of view, these models are *alive*, since they are permanently and automatically upgraded. As long as the monitoring devices and the diagnosis protocols are connected to a database, the whole loop (model definition and upgrade system) are automatically maintained by the generated framework.

The current implementation of this methodology makes it suitable only for the processing of signals measured over time, i.e., the scenario is considered as discrete time system. As future work, its features will be expanded to deal with non-temporary data. Because of our collaboration with the Neurological Unit of the Hospital Universitario de La Princesa, our approach is currently focused on neurological diseases. However, we will test the methodology on other clinical trials. Our future work also include the analysis of several

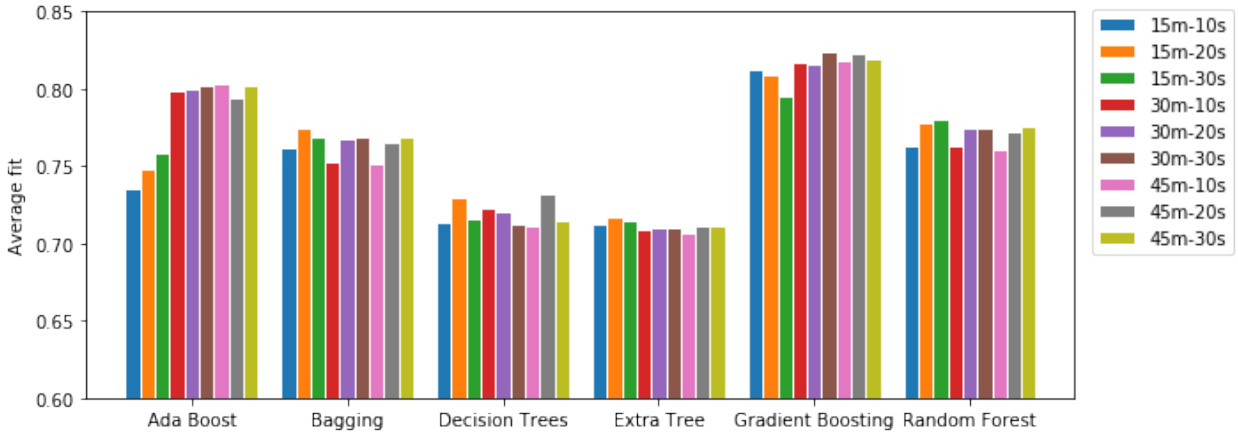

Figure 8: Fit of the stroke type prediction models, generated with different combinations of sample size in minutes (m) and number of samples used in the size of the window (s).

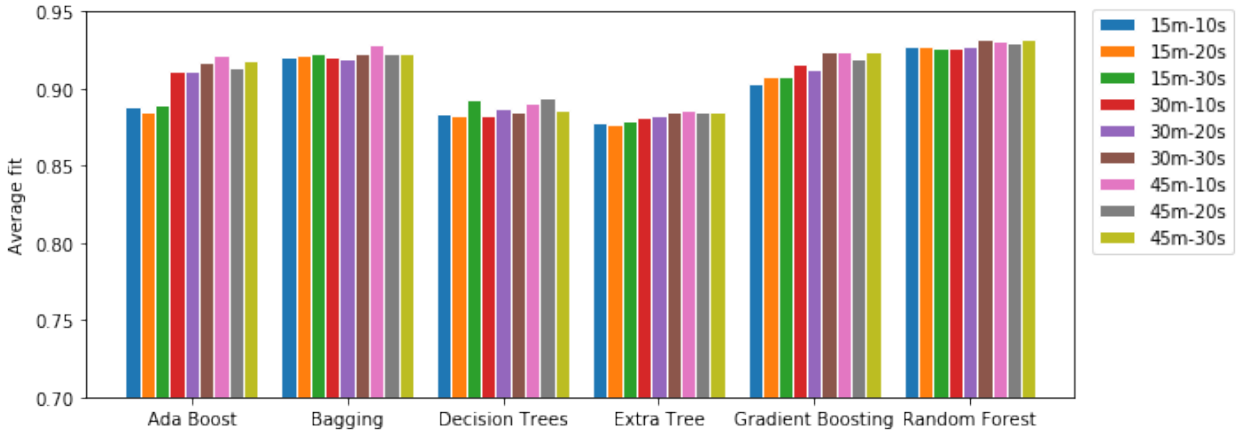

Figure 9: Fit of the *exitus* prediction models, generated with different combinations of sample size in minutes (m) and number of samples used in the size of the window (s).

scalability issues, since this methodology demands high computational resources as the number of patients increases.

REFERENCES

Hill, Michael D 2005. "Diagnostic biomarkers for stroke: a stroke neurologist's perspective".

Khosla, A., Y. Cao, C. C.-Y. Lin, H.-K. Chiu, J. Hu, and H. Lee. 2010. "An Integrated Machine Learning Approach to Stroke Prediction". In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pp. 183–192. New York, NY, USA, ACM.

Kimura, S., T. Sato, S. Ikeda, M. Noda, and T. Nakayama. 2010. "Development of a database of health insurance claims: standardization of disease classifications and anonymous record linkage". *Journal of epidemiology* vol. 20 (5), pp. 413–419.

Low, D. E., D. Alderson, I. Cecconello, A. C. Chang, G. E. Darling, X. B. D'journo, S. M. Griffin, A. H. Hölscher, W. L. Hofstetter, B. A. Jobe et al. 2015. "International consensus on standardization of data

collection for complications associated with esophagectomy". *Annals of surgery* vol. 262 (2), pp. 286–294.

Matias-Guiu, J. A., J. Díaz-Álvarez, J. L. Ayala, J. L. Risco-Martín, T. Moreno-Ramos, V. Pytel, J. L. Carreras, J. Matias-Guiu, and M. N. Cabrera-Martín. 2018. "Clustering analysis of FDG-PET imaging in primary progressive aphasia". *Frontiers in aging neuroscience* vol. 10, pp. 230.

Pagán, J., D. Orbe, M. Irene, A. Gago, M. Sobrado, J. L. Risco-Martín, J. V. Mora, J. M. Moya, and J. L. Ayala. 2015. "Robust and accurate modeling approaches for migraine per-patient prediction from ambulatory data". *Sensors* vol. 15 (7), pp. 15419–15442.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg et al. 2011. "Scikit-learn: Machine learning in Python". *Journal of machine learning research* vol. 12 (Oct), pp. 2825–2830.

Thirumalai, C., A. Duba, and R. Reddy. 2017. "Decision making system using machine learning and Pearson for heart attack". In *Electronics, Communication and Aerospace Technology (ICECA), 2017 International conference of*, Volume 2, pp. 206–210. IEEE.

WHO 2018. "The Atlas of Heart Disease and Stroke". https://www.who.int/cardiovascular_diseases/resources/atlas/en/ (accessed 13-June-2019).

Witten, I. H., E. Frank, L. E. Trigg, M. A. Hall, G. Holmes, and S. J. Cunningham. 1999. "Weka: Practical machine learning tools and techniques with Java implementations".

Zeigler, B. P., H. Praehofer, and T. G. Kim. 2000. *Theory of Modeling and Simulation. Integrating Discrete Event and Continuous Complex Dynamic Systems*. 2 ed. Academic Press.

AUTHOR BIOGRAPHIES

KEVIN HENARES is a Ph.D. candidate at the Complutense University of Madrid (UCM). His work focuses on the development of robust modeling and simulation methodologies to study the behavior of complex systems. His email address is khenares@ucm.es.

JOSÉ L. RISCO-MARTÍN received his Ph.D. from Complutense University of Madrid, and currently is Associate Professor in the Department of Computer Architecture and Automation at Complutense University of Madrid. His research interests include computer aided design, and modeling, simulation and optimization of complex systems. He can be reached at jlrisco@ucm.es.

ROMÁN HERMIDA received his Ph.D. from Complutense University of Madrid, and is currently Full Professor in the Department of Computer Architecture and Automation at the same university. His research interests include design automation, computer architecture and embedded systems. He can be reached at rhermida@ucm.es.

GEMMA REIG ROSELLÓ is a Ph.D. candidate at the Stoke Care Unit of the Hospital Universitario de La Princesa, where she is Area Specialist in Neurology since 2009. She can be reached at gemma.reig@salud.madrid.org.

ROMÁN CÁRDENAS is a MSc candidate at the Technical University of Madrid (UPM). He received his BSc in Telecommunication Engineering in 2017 from UPM. His research interests include modeling and simulation with applications in the IoT domain. His email address is roman.cardenas.rodriguez@alumnos.upm.es.