2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)
July 6-11, 2014, Beijing, China

The Experimenter Environment of the NIP Imperfection Processor

Raquel Martínez, José M. Cadenas, M. Carmen Garrido
Dpto. Ingeniería de la Información y las Comunicaciones
Facultad de Informática. University of Murcia
3071 Espinardo (Murcia). Spain
Email: raquel.m.e@um.es, jcadenas@um.es, carmengarrido@um.es

***Abstract*—Currently, most datasets from real-world problems contain low-quality data. In particular, within soft computing and data mining areas, the research and development of techniques that can deal with this type of data has been increased recently. In order to facilitate the design of experiments in this field and with these data, an experimenter environment in NIP imperfection processor software tool has been developed. This environment allows the generation of datasets with low-quality data, allowing the researcher to design experiments that analyze the robustness of different techniques which utilize this type of information in an easy and intuitive way.**

***Index Terms*—Software tool, Data mining, Low quality data, Softcomputing**

I. Introduction

At present, different software tools in the field of Soft Computing are available. Although most of them are commercial, the number of open source software tools has been increased considerably. In particular, "NIP Imperfection Processor" (hereafter NIP) is an open source software tool [1] which was developed with the aim of supporting the work in data mining, with specific emphasis on data preprocessing in environments with Low-Quality Data (LQD). NIP allows the management and generation of datasets with LQD. Some of the main advantages of this tool are:

- It facilitates the work in the field of data mining allowing the generation of datasets with LQD. The generation of datasets with NIP tool can serve as common framework to carry out a comparison of the different data mining techniques developed.
- Allows the generation of the same dataset with the information expressed by different theories of formalizing imprecision/uncertainty, allowing a comparison between them.

The aim of this paper is to present a new module of the NIP tool which includes an experimenter environment so that researchers can design, create and execute experiments with datasets with LQD of a more appropiate form in a oriented-research of the LQD management.

This work is structured as follows. In Section II the experimenter environment (called ExpNIPip) of the NIP tool is described. In Section III a case study is used to show the definition process of an experiment using ExpNIPip environment and finally, in Section IV the main conclusions of this work are presented.

II. ExpNIPip Environment

The experimenter environment ExpNIPip is composed of two general components: a framework for the construction of an experiment or group of experiments and, a simple and intuitive graphic interface developed using Java technology and the Swing API which controls the construction of such experiments. It is a multiplatform software. It is important to note that throughout this document the term experiment in ExpNIPip environment refers to the preparation of all datasets with LQD. These datasets will be subsequently used by the researcher to carry out their research in the field of data mining with LQD.

ExpNIPip allows graphically designing an experiment or group of experiments with datasets with LQD. The interface of the experimenter environment ExpNIPip is based on dataflow where the researcher can select functional elements of NIP (denoted nodes) from a toolbar. Also these elements must be connected each other (with links) in order to reflect the sequence of them. This structure forms a graph representing the flow of knowledge for data processing. The software is robust detecting wrong graphs through semantic control module whose operation will be discussed throughout the work.

ExpNIPip facilitates the use of the NIP tool at the research level and preparation of large-scale experiments. Its main advantages are:

- It is a software tool developed for research purposes.
- Permits the functioning of NIP tool on large sets of datasets in one step.
- Includes new functions which were not available in previous NIP tool such as the imputation process, new partition/discretization methods, etc.
- The experiments are defined graphically based on dataflow.
- Allows the definition of several functional sequences of datasets (branches of the same graph). Therefore, it permits the ability to carry out similar experiments on many datasets easily and quickly.
- Once an experiment is designed, it may be executed in on-line mode. ExpNIPip allows parallel execution in on-line mode of several experiments, while the user can continue with the definition of another experiment. A log

978-1-4799-2072-3/14/$31.00 ©2014 IEEE

indicating the execution status of each experiment which is in progress can be visualized.
- An experiment can be saved and executed in any other time in batch mode using the script generated. Also, a log of the experiment execution is generated.
- A saved experiment can be subsequently retrieved from ExpNIPip to be modified. This allows green the user the ability to obtain similar versions of the same experiment in an quick and efficient way.

A. Main Actions and Components in the ExpNIPip Environment

When accessing the ExpNIPip environment, the user is prompted with the screen shown in Figure 1.

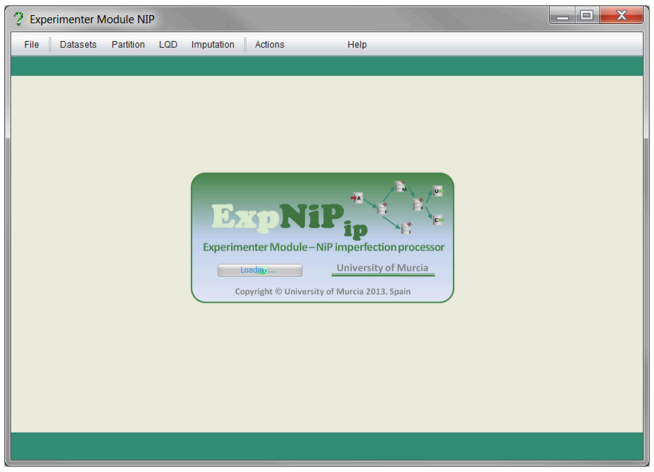

Fig. 1. Interface of the ExpNIPip environment

At the top of the screen shown in Figure 1, we can find options that allow us to incorporate the main components of an experiment. These options display a toolbar that allows us to set up the experiment graphically and interactively. The central panel of the screen is the drawing panel and shows the components of the current experiment with the flow and relations between them.

The main actions used to carry out the creation of experiments are grouped under *File* and *Actions*:

- *New*: Creates a new experiment.
- *Open*: Selects an experiment previously created and load it into ExpNIPip for editing.
- *Save*: Saves the current experiment with the present name or another (*Save as*). Saving the experiment, files ".exp" and ".scr" are generated. The experiment saved may be executed in batch mode at any time using the generated script (file ".scr") or retrieved later to complete or modify its structure.
- *Run*: Executes the current experiment in on-line mode and generates the corresponding datasets. ExpNIPip allows the execution of several experiments in on-line mode while the user continues with the definition of another experiment.
- *Links*: Adds links to define the relations between the components of the experiment at any time.
- *Visualize*: Displays the status of execution of the experiments that are running in on-line mode. The result will also be saved in a file *".log"*.

Possible components of a graph of an experiment are:

- *Datasets-Import*: Configures the set of experiment's input datasets.
- *Partition*: Applies different partitioning methods of numerical attributes on datasets.
- *LQD-Add*: Adds different kinds of LQD into the datasets.
- *Imputation*: Applies different imputation methods on LQD.
- *LQD-formats*: Sets the output format of LQD.
- *Datasets-Export*: Sets the output format of the experiment datasets.

Figure 2 shows an example of the dynamic work with ExpNIPip and its main elements. When the user introduces a component of any type in the graph, by clicking the right mouse button the user can activate any component of the graph and sets each particular properties.

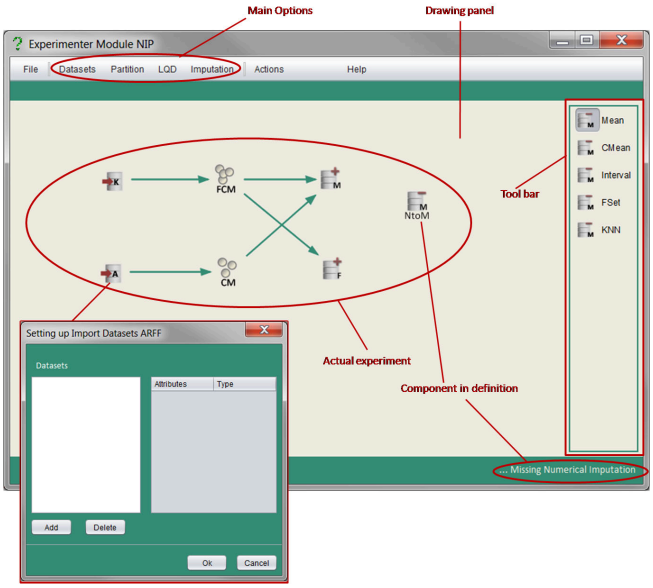

Fig. 2. ExpNIPip: An illustrative window

The semantic verification of the built graph is carried out by a semantic control module. Once we have completed the definition of an experiment and during the execution, the semantic control module checks that there are no inconsistencies in the definition of the experiment, showing if an error is found. Throughout this paper we will comment on the main situations of incoherence that this semantic control module can detects.

In general, a graph of the experiment will contain a number of branches that start at a root node (components *Datasets-Import*) and end in leaf nodes that may be components

Partition or *Datasets-Export*. These nodes correspond to the generation of partitions or defining output formats respectively. This graph structure is shown in the classes diagram of Figure 3. This figure shows the simplified classes diagram on which ExpNIPip is constructed. The diagram only shows the main classes (omitting attributes and methods) for viewing the branch structure of a graph of experiment.

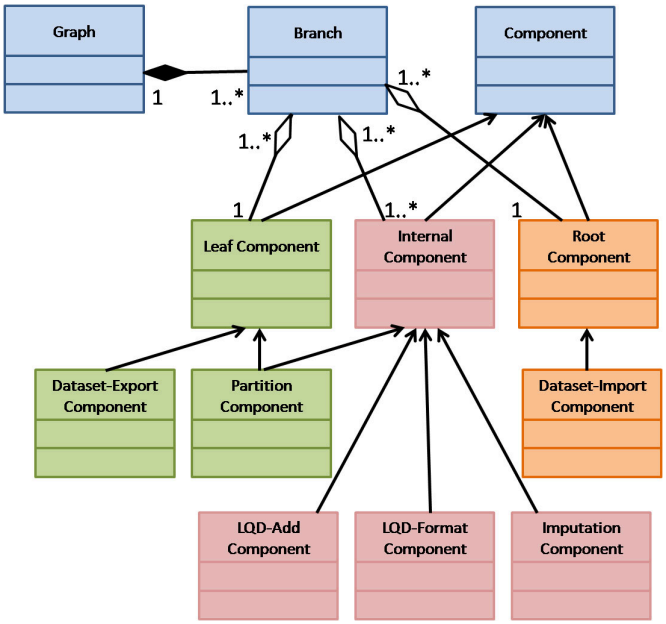

Fig. 3. Simplified classes diagram of ExpNIPip

B. Import Datasets

A component *Datasets-Import* allows the incorporation of datasets into several standard formats as well as predefined formats determined by the user.

The available input formats are ARFF, KEEL, UCI, CSV and Custom[1].

The user can incorporate as many components *Datasets-Import* as desired in an experiment. When the user adds a component *Dataset-Import* to the experiment, it may be added to that component several datasets in the format specified by the component. Datasets with different custom formats can be incorporated into the experiments by adding a component for each of them.

At any time during the definition of the experiment, we can modify the configuration of the component. Such changes are transferred to other components of the experiment connected to it. The semantic control module is responsible for checking both that components *Datasets-Import* are always root nodes of the various branches of the graph, and that there is at least one node in each branch.

[1]To be interpreted a dataset with custom input format by ExpNIPip, a file ".in" is needed. This file can be obtained by NIP tool.

C. Partition Methods

A component *Partition* allows the incorporation of the discretization of numerical attributes within a group of datasets. This discretization or partitioning can be carried out through a set of methods in order to provide the researcher the automatic generation of such partitions. These partitions can be used in data mining techniques which require a partitioning of numerical attributes as well as the replacement of numerical values by their corresponding linguistic labels in the datasets. Because of this, ExpNIPip improves the task of checking what type of partitioning is more suitable for a particular technique.

Discretization is a process in which the numerical values of attributes in a dataset are converted into discrete or nominal values with a finite number of intervals, establishing a correspondence between each numerical value and an interval. Once the discretization is performed, the attribute may be considered nominal. This process is very important in certain data mining techniques because there are techniques that exhibit better performance when attributes are discretized since discretization reduces the number of continuous attribute values, enabling faster and more accurate learning [2]. Also, some data mining techniques are designed to deal only with nominal attributes although many real-world problems involve information expressed with numerical attributes. All these numerical attributes, in this case, have to be discretized.

In the literature there are a large number of discretization techniques and diverse taxonomies of them [3]. Given the main purpose of ExpNIPip, the feature we wish to emphasize about discretization methods is the type of partitions generated by distinguishing between fuzzy and crisp partitions. We consider that the partition generated by a specific method is fuzzy when a set of discrete or nominal values with a certain degree is assigned to a numerical value. On the other hand, we consider that the partition is crisp when a discrete or nominal value is assigned to a numerical value. The first type of partition produces a set of intervals with some overlap, while the latter generates a set of non-overlapping intervals.

Tables I and II show the different partitioning methods available in the tool and are classified according to the type of partition generated. For each method the acronym used in the tool, a reference and a brief description is shown.

TABLE I
CRISP PARTITION METHODS

Acronym	Ref.	Description
DT	[4]	Based on a Decision Tree Algorithm
CM	[5]	Based on C Means Algorithm
Extern	–	Any external crisp partition in appropiate format

In addition to automatically generating partitions through the methods described above, the tool provides the possibility of using a partition generated with any external method (components *Partition-Crisp-Extern* or *Partition-Fuzzy-Extern*) if it is provided in the appropriate format. The fuzzy partitions allowed in this tool consist of trapezoidal fuzzy sets.

TABLE II
FUZZY PARTITION METHODS

Acronym	Ref.	Description
OFP	[4]	Based on a Decision Tree and a Genetic Algorithms
BAGOFP	[6]	Based on a Decision Tree, Genetic Algorithm and Bagging
GA	[7]	Based on a Genetic Algorithm
FCM	[8]	Fuzzy C Means
Extern	–	Any external fuzzy partition in appropiate format

The format of the file that contains a fuzzy partition for a given dataset is shown in Figure 4. This file, for each numerical attribute, includes different linguistic labels corresponding to its discretization with the four values that define the trapezoidal membership function. In the case of a crisp partition, the file format is the same however the first two and the last two values of each linguistic label are equal.

```
AGE 6
  AGE1,7.0,7.0,24.8849,28.1154
  AGE2,24.8849,28.1154,37.4022,37.593903
  AGE3,37.4022,37.593903,40.3984,40.6043
  AGE4,40.3984,40.6043,41.9675,44.0265
  AGE5,41.9675,44.0265,52.9370,70.062195
  AGE6,52.937,70.062195,78.0,78.0
BILIRUBIN 1
  BILIRUBIN0,0.3,0.3,8.0,8.0
ALK_PHOSPHATE 2
  ALK_PHOSPHATE1,26.0,26.0,37.3249,201.657
  ALK_PHOSPHATE2,37.3249,201.657,295.0,295.0
SGOT 1
  SGOT0.14.0.14.0.648.0.648.0
```

Fig. 4. File with a fuzzy partition

The semantic control module of ExpNIPip is responsible for detecting whether at least one partition is specified when carrying out the execution of the experiment. This occurs when the option of adding numerical imprecision is selected (fuzzy, interval or fuzzy subsets) or if we want to express the fuzzy/interval values using a label in the output datasets. When there are several components *Partition* in a sequential flow, the last of them will be used in components which need a partition.

D. Add Low Quality Data

The component *LQD-Add* of ExpNIPip encapsulates the functionality of introducing certain percentages of LQD in datasets. This functionality allows the generation of the same dataset with different types and different percentages of LQD. This facilitates the task of generating datasets to evaluate the robustness of those techniques with LQD treatment according to the different types of data.

The possibilities provided by ExpNIPip are:

- Introduction of certain percentages of missing values both into nominal and into numerical attributes. Introduction of these values is carried out randomly, so that the missing is of the type completely at random [9]. In this case the distribution of the examples with missing values in an attribute does not depend on the observed data and the missing values. Therefore the distribution of the complete data and the missing data is the same.
- A certain percentage of fuzzy values in numerical attributes. In this case it is necessary to have a partitioning of numerical attributes that may have been generated automatically by the tool or be provided by the user in the correct format. Given the partition, when a crisp numerical value must be transformed into a fuzzy value, that value of the fuzzy partition to which it belongs with the highest degree is selected.
- A certain percentage of interval values in numerical attributes. In this case, there are three options: 1) use a crisp partition automatically generated by the tool or provided by the user, 2) add to and subtract from a fixed amount specified by the user to the numerical values of the attribute, and 3) add to and subtract from a random amount within a range specified by the user, to the numerical values of the attribute.
- Crisp subsets in nominal attributes. If the nominal value is replaced by a crisp subset, all other possible domain values may be added to the current value with a probability equal to the proportion of each value in the dataset.
- Fuzzy subsets in both nominal and numerical attributes. In the case of numerical attributes, given a fuzzy partition of that attribute, the numerical value is replaced by the subset of the partition labels to which this value belongs with degree greater than 0 along with their membership degrees. For instance, a crisp value of 45 for the numerical attribute "temperature" could be replaced by the fuzzy subset {0.1/warm,1/hot} if the value 45 belongs to "warm" label with degree 0.1 and to "hot" label with degree 1. In the case of nominal attributes, if the value is replaced by a fuzzy subset, as membership degree of the current value its proportion in the dataset is added. Also other values are added in the same way as in the case of crisp subsets including in this case together with the value, its proportion in the dataset as membership degree.
- Certain percentage of noise in nominal and numerical attributes. In the case of the numerical attributes, the noise that we add is gaussian noise. In this kind of noise, it is necessary to indicate in the attribute, which we want to modify, the mean and the standard deviation. This kind of noise is useful for simulating the existence of an error with known distribution in data (for example, a sensor of which the measurement accuracy is known). In the case of the nominal attributes the percentage of specified values will be changed randomly by another value of the domain.

When defining the properties of these components, a list of all datasets associated by links to that component is shown. If we select one of them, the list of attributes in that dataset is expanded allowing the selection of: a set of attributes, all attributes, all numerical attributes or all nominal attributes, depending on the type of LQD we are adding (Figure 5). The

same can be done by selecting a set of datasets.

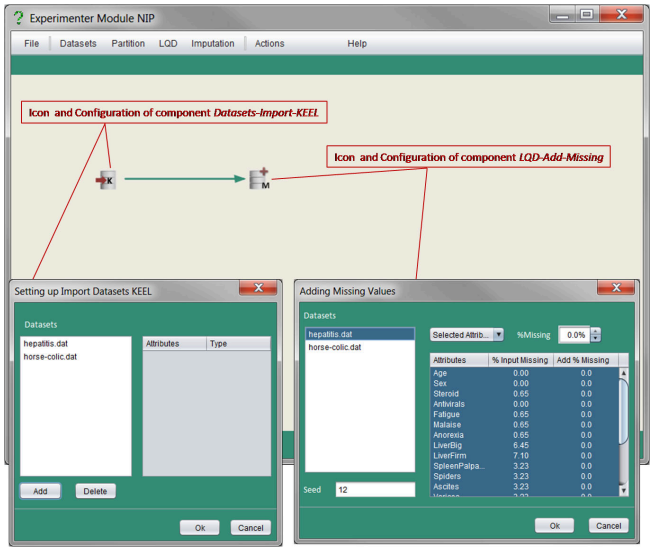

Fig. 5. Configuration of components *LQD-Add*

Also, the semantic control module detects that in the flow of the experiment, there is a partition selected when interval values, fuzzy values, or fuzzy subsets in numerical attributes are added.

E. Imputation methods

In general, the treatment of LQD can be carried out from two different approaches:

- Removing all those examples with some value of low quality in some of its attributes or remove those attributes with low quality values in a considerable percentage of examples.
- Impute values of low quality by estimated ones. Since in most cases the attributes are not independent of each other, we can identify relationships between them using predictive models to carry out the imputation of values.

NIP tool focuses on the use of imputation methods which replace the values by other ones obtained from the dataset, such as the attribute mean, mode, etc. Although in the literature, most imputation methods have focused on the imputation of missing values, NIP tool facilitates the carrying out of imputation of other values of low quality. Therefore, the imputed values by the tool will not always be missing values in the same way that the resulting values of the imputation will not always be crisp values. The choice, therefore, of the imputation method will be conditioned by the type of LQD with which the technique of data mining is able to treat.

An important advantage of imputation methods is that they are independent of the subsequent data mining technique applied, but some of them may be more appropriate than others when working with a particular technique. This is the point where ExpNIPip provides the functionality of generating quickly and efficiently the same dataset with the low quality values imputed by several imputation methods.

ExpNIPip includes the replacement methods of NIP tool and also provides new functionality incorporating methods that use predictive models to impute values. The imputation methods available in ExpNIPip and the type of provided value are listed below (in parentheses, the acronym used in the tool for each method is shown):

- To nominal missing values:
  - (NotoM) The missing value is imputed by the mode or most common value of the attribute. The provided value is crisp.
  - (NotoCM) The missing value is imputed by the conceptual mode, i.e., the most common value of the attribute considering only examples belonging to the same class of example to impute. The provided value is crisp.
  - (NotoCS) The missing value is imputed by the full set of possible values. The generated value is a set containing all possible values of the attribute. The set is classic because it does not assign a weight or probability to the values.
  - (NotoFS) The missing value is imputed by a fuzzy subset that assigns to each possible value of the attribute the proportion in which it appears in the dataset. In this case, the value returned is a set consisting of the possible values of the attribute, associating a weight to each value based on its proportion of occurrence in the dataset.
  - (NoKNN) The missing value is imputed by a crisp value, a crisp or fuzzy set depending on the value obtained by the predictive imputation method K_{M-LQD}-NN [10].
- To numerical missing values:
  - (NtoM) The missing value is imputed by the mean of the attribute. The provided value is crisp.
  - (NtoCM) The missing value is imputed by the conceptual mean, i.e., by the mean of the attribute considering only examples belonging to the same class of example to impute. The provided value is crisp.
  - (NtoI) The missing value is imputed by an interval corresponding to the entire domain of the attribute. The value generated is a classic interval defined between the minimum and maximum value of the attribute in the dataset.
  - (NtoF) The missing value is imputed by the fuzzy set $(\mu-2\sigma, \mu-\sigma, \mu+\sigma, \mu+2\sigma)$ where μ and σ are the mean and standard deviation of the attribute. In this case, the value generated is a trapezoidal fuzzy set.
  - (NKNN) The missing value is imputed by a crisp, fuzzy or interval value depending on the value obtained by the predictive imputation method K_{M-LQD}-NN [10].

- To interval values:
  - (ItoC) Impute the interval by its intermediate value. In this case, the provided value is crisp.
  - (ItoMin) Impute the interval by its minimum value. The provided value is crisp.
  - (ItoMax) Impute the interval by its maximum value. The provided value is crisp.
- To fuzzy values:
  - (FtoI) Impute the fuzzy value by the interval associated to α-cut 1. The provided value is an interval.
  - (FtoCG) Impute the fuzzy value by the crisp value corresponding to its center of gravity. The provided value is crisp.

The possibility of imputing the same type of value for a set of possibilities facilitates research in data mining in two directions: 1) facilitating the analysis of which kind of value provides the best alternative to the nature of the imperfection that appears in the data. This is important when working with data mining techniques that perform LQD processing in different forms and 2) facilitating the analysis about what imputation method generates crisp values in a more appropriate way for a given data mining technique, in the case of techniques that do not perform any LQD processing.

According to connections that a user does of imputation methods in the graph of the experiment, the imputations can be chained. The semantic control module does not verify the absence of a particular kind of low quality values, which the researcher wishes to impute, since in this case the imputation is not performed.

F. Low Quality Data Formats and Export Datasets

Components *LQD-Format* and *Datasets-Export* allow the incorporation in the experiment of the output configuration of datasets generated in various formats of an easy and quick way. Each branch of the graph of the experiment generates an output which can be: 1) a partition of the attributes when the leaf node of the branch is a component *Partition* or 2) a set of datasets with a particular format if the leaf node of the branch is a component *Datasets-Export*.

A component *Datasets-Export* sets the output format for the datasets generated by the experiment. Output formats can be ARFF, KEEL, UCI, CSV and Custom[2].

You can specify as many output formats for datasets as desired, indicating for each one the following options:

- Seed to be used (Seed).
- If the experiment is a cross validation, the number of folds and repetitions must be indicated.
- If the output is required as train and test files, the percentage of train and test examples and the number of repetitions must be specified.

Furthermore, each one specifies the default syntax for the different low quality values. If the user wants to modify the default format of low quality values, they can do it by components *LQD-Format*.

Components *LQD-Format* allow the specification of each output format of low quality values present in the different datasets. In particular, if the output with label is selected in the case of interval or fuzzy values, the semantic control module checks that there is a fuzzy partition previously defined and only the fuzzy values or intervals that are part of that partition are printed as a label. The remaining values are printed according to the default format of each output format. If we want to leave the default formats, this component can be omitted.

[2]To be interpreted a dataset with custom output format by ExpNIPip, a file ".out" is needed. This file can be obtained with NIP tool.

III. A CASE STUDY

In this section, a case study is presented as an example of the functionality and the process necessary used when creating an experiment using ExpNIPip enviroment.

Suppose we are performing an experimental analysis of the classification technique FRF [11] with the following objectives 1) Analyze the robustness of the FRF technique to the existence of different types of LQD, in particular the existence of missing data in the datasets, the existence of the interval values, and the existence of the fuzzy values. To do that, we want to analyze whether this technique performs better with a dataset with a certain percentage of missing values, or the same percentage of interval values, or the same percentage of fuzzy values and 2) Analyze the behavior of FRF against different imputation methods of missing data. That is, we impute the missing values added in numerical attributes using different imputation methods to see which method produces better performance in classification of the technique. With the ExpNIPip tool, we expect to help to ease the researchers' task of preparing the datasets required for their researches.

We want to obtain ten datasets in KEEL format [12] and in UCI format [13] the following datasets: 1) datasets containing missing values, 2) datasets with missing values imputed with different methods, 3) datasets with interval values and finally, 4) datasets with fuzzy values. These datasets must be adequate to perform a 2×3-fold cross-validation. The output format must be the specific one that uses FRF for all datasets, in particular, we have the output format "frf.out" (a custom format).

Table III shows the datasets used in the experiment, indicating for each one the name, the number of attributes, the number of examples, the number of class values, if the datasets contain LQD and the input format.

A. Designing the experiment with ExpNIPip

We start by building the graph of the experiment adding two components *Datasets-Import*, one for each input format of the datasets used in the experiment (*Datasets-Import-KEEL* and *Datasets-Import-UCI*). To add a percentage of interval values to the datasets, we incorporate in the graph a component *Partition-Crisp*. This component should appear in the flow graph before adding therein, the component *LQD-Add-Interval*. In the same way, we create a fuzzy partition to add a

TABLE III
DATASETS EMPLOYED IN THE CASE STUDY

Name	#Ats	#Ins	#Cla	lqd	#Format
appendicitis	7	106	2	No	KEEL
pima	8	768	2	No	KEEL
glass	9	214	7	No	KEEL
hepatitis	19	155	2	Yes	KEEL
horse-colic	2	368	2	Yes	KEEL
wine	13	178	3	No	UCI
BCW	32	569	2	No	UCI
iris	4	150	3	No	UCI
ionosphere	34	351	2	No	UCI
zoo	17	101	7	No	UCI

percentage of fuzzy values to datasets. Next, we incorporate a component to add a percentage of missing values (component *LQD-Add-Missing*). Later, the missing values will be imputed using three different imputation methods for which we add the components *Imputation-Missing-NtoM*, *Imputation-Missing-NtoCM* and *Imputation-Missing-NKNN*. Finally, we add a component *Datasets-Export-Custom* indicating the number of repetitions and size of cross-validation to generate, in the format defined by the user "frf.out".

The graph of Figure 6 shows the data-flow and links which form the experiment.

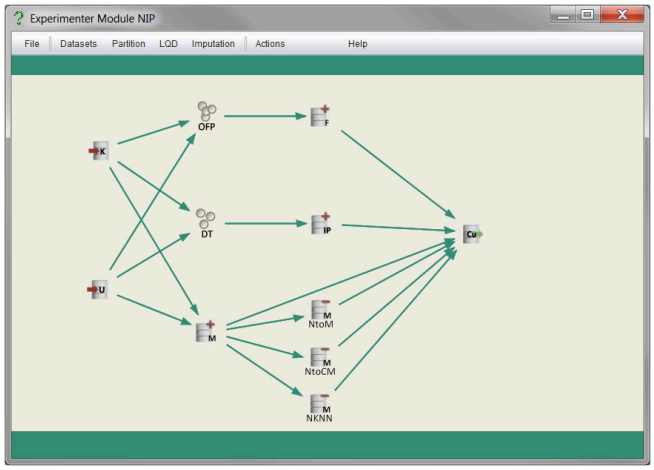

Fig. 6. Graphical representation of the experiment

Nodes represent components *Datasets-Import* containing sets of datasets, components *Partition* corresponding to obtain crisp and fuzzy partitioning of the datasets, components *LQD-Add* to add missing, interval and fuzzy values to datasets, components *Imputation* corresponding to imputation methods of missing values and finally, components *Datasets-Export* for the generation datasets in an output format predefined by user. The nodes are distinguished because they have different icons.

All parameters of the different components are set by clicking the right mouse button when we are located on the corresponding icon. Table IV shows the parameters used in the configuration of each component of experiment.

TABLE IV
PARAMETERS USED IN COMPONENTS

Component	Parameters
Import datasets UCI	–
Import datasets KEEL	–
DT Partition	Seed-59, MinEx-1, %Pure-100, Thread-2, Class attribute for all datasets
OFP Partition	Seed-59, MinEx-1, %Pure-100, Thread-2, Gene-100, Popu-20, PCross-0.9, PMut-0.2, Class attribute for all datasets
Add missing	Seed-0, 5% in all attributes
Add interval	Seed-0, 10% in all numerical attributes
Add fuzzy	Seed-0, 5% in all numerical attributes
Imputation missing NtoM	–
Imputation missing NtoCM	–
Imputation missing NKNN	K_value-sqrt, Thresdhold-0, Distance-F1
Export datasets Custom	"frf.out", N. folds-3, Repetitions-2, Seed-0

When we have configured the experiment, we can save it (files ".exp" and ".scr" are generated). Then we can retrieved later or run in batch mode. From the experiment definition screen, we can run it in on-line mode (*Actions-Run*) following its execution through a log that we can access by option *Actions-Visualize*. This log will be saved in a file ".log" describing the sequence of actions that have been carried out in the execution of the experiment and actions that have not been carried out by an error.

For the case study we are analyzing, Figure 7 shows a part of the directory structure and the files generated after running the experiment.

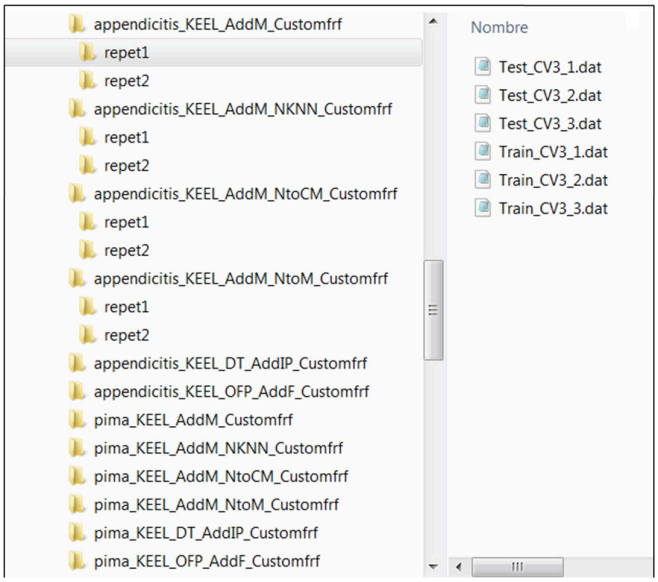

Fig. 7. Partial view of output generated by the experiment

Links to component *Datasets-Export* define the set of datasets to generate. Thus, for each dataset *DS* in the experiment were generated six directories in the directory specified by the user in component *Datasets-*

Export: "*DS*_KEEL_OFP_AddF_Customfrf" contains datasets with fuzzy values of a fuzzy partition generated by the *OFP* method, "*DS*_KEEL_DT_AddIP_Customfrf" contains datasets with interval values of a partition generated by the *DT* method, "*DS*_KEEL_AddM_Customfrf" contains datasets with missing values, "*DS*_KEEL_AddM_NtoM_Customfrf" contains datasets with missing values imputed using the *NtoM* method, "*DS*_KEEL_AddM_NtoCM_Customfrf" contains datasets with missing values imputed using the *NtoCM* method and "*DS*_KEEL_AddM_NKNN_Customfrf" contains datasets with missing values imputed using the *NKNN* method. Each of these six directories contain the respective datasets as an experiment 2×3-fold cross-validation. The name of each directory describes the branch in the graph that generates it.

In addition, the crisp and fuzzy partitions that are part of the experiment are generated in the directory specified by the user.

A part of file ".scr" generated by experiment is shown in Figure 8.

```
------------------------------------------------

Import Datasets Format KEEL
     glass.dat: Path-H:\keel
     hepatitis.dat: Path-H:\keel
     pima.dat: Path-H:\keel
     appendicitis.dat: Path-H:\keel
     horse-colic.dat: Path-H:\keel

Discretization Method-OFP_CLASS Path-H:\partitionf Seed-59 MinEx-1 %
Pure-100 Thread-2 Gene-100 Popu-20 PCross-0.9 PMut-0.2
     glass.dat TypeGlass
     hepatitis.dat Class
     pima.dat Class
     appendicitis.dat Class
     horse-colic.dat Surgical_lesion?

Add Fuzzy Seed-0
     glass.dat: (RI-4.9999995% Na-4.9999995% Mg-4.9999995% Al-
4.9999995% Si-4.9999995% K-4.9999995% Ca-4.9999995% Ba-4.9999995% Fe-
4.9999995%)
     hepatitis.dat: (Age-4.9999995% Sex-4.9999995% Steroid-4.9999995%
Antivirals-4.9999995% Fatigue-4.9999995% Malaise-4.9999995% Anorexia-
4.9999995% LiverBig-4.9999995% LiverFirm-4.9999995% SpleenPalpable-
4.9999995% Spiders-4.9999995% Ascites-4.9999995% Varices-4.9999995%
Bilirubin-4.9999995% AlkPhosphate-4.9999995% Sgot-4.9999995% AlbuMin-
4.9999995% ProTime-4.9999995% Histology-4.9999995%)
     pima.dat: (Preg-4.9999995% Plas-4.9999995% Pres-4.9999995% Skin-
4.9999995% Insu-4.9999995% Mass-4.9999995% Pedi-4.9999995% Age-
4.9999995%)
     appendicitis.dat: (At1-4.9999995% At2-4.9999995% At3-4.9999995%
At4-4.9999995% At5-4.9999995% At6-4.9999995% At7-4.9999995%)
     horse-colic.dat: (Hospital_Number-4.9999995% Rectal_temperature-
4.9999995% Pulse-4.9999995% Respiratory_rate-4.9999995%
Nasogastric_reflux_PH-4.9999995%  Packed_cell_volume-4.9999995%
Total_protein-4.9999995% Abdomcentesis_total_protein-4.9999995%)

Export Dataset Format Custom: FileFormat-"frf.out" Path-H:\results
FoldCv-3 Repe-2 Seed-10.0
```

Fig. 8. Part of ".scr" file of the experiment

IV. Remarks and conclusions

Due to the nature of the datasets from real-world problems, it is important to include into data mining techniques the LQD processing or develop new techniques that support this type of data. In the same way that there are currently open source software tools to support data mining with datasets without LQD, in this work the functionality of the NIP imperfection processor software tool is completed with an experimenter environment in order to facilitate the design and generation of large-scale datasets to conduct experiments in environments with LQD. We have presented through a case study how an extensive set of datasets can be generated easily and intuitively using this module. Finally, it is important to take into account that EXpNIPip can be easily extended adding new components in any functionality of the tool.

Acknowledgment

Supported by the project TIN2011-27696-C02-02 of the Ministry of Economy and Competitiveness of Spain. Thanks also to "Fundación Séneca - Agencia de Ciencia y Tecnología de la Región de Murcia" (Spain) for the support given to Raquel Martínez by the scholarship program FPI.

References

[1] J.M. Cadenas, M.C. Garrido, R. Martínez, *NIP - An Imperfection Processor to Data Mining datasets*, International Journal of Computational Intelligence Systems, 6(1): 3-17, 2013.

[2] M. Antonelli , P. Ducange, B. Lazzerini, F. Marcelloni, *Learning knowledge bases of multi-objective evolutionary fuzzy systems by simultaneously optimizing accuracy, complexity and partition integrity*, Soft Computing, 15(12):2335-2354, 2010.

[3] S. García, J. Luengo, J.A. Sáez, V. López, F. Herrera, *A Survey of Discretization Techniques: Taxonomy and Empirical Analysis in Supervised Learning*, IEEE Transaction on Knowledge and Data Engineering, 25(4): 734-750, 2013.

[4] J.M. Cadenas, M.C. Garrido, R. Martínez, P.P. Bonissone, OFP_{CLASS}*: a hybrid method to generate optimized fuzzy partitions for classification*, Soft Computing, 16(4): 667-682, 2012.

[5] E.W. Forgy, *Cluster analysis of multivariate data: efficiency versus interpretability of classifications*, Biometrics 21: 768-769, 1965.

[6] J.M. Cadenas, M.C. Garrido, R. Martínez, *Improving a Fuzzy Discretization Process by Bagging*, International Conference on Fuzzy Computation Theory and Application, 201-213, 2013.

[7] Y.S. Choi, B.R. Moon, *Feature Selection in Genetic Fuzzy Discretization for the Pattern Classification Problems*, IEICE Trans. Inf. and Syst., E90-D(7): 1047-1054, 2007.

[8] J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum, New York, 1981.

[9] R.J.A. Little, D.B. Rubin, *Statistical analysis with missing data*, Wiley Series in Probability and Statistics, 1st edn. Wiley, New York, 1987.

[10] J.M. Cadenas, M.C. Garrido, R. Martínez, A. Martínez, *Imputing missing values from Low Quality Data by NIP tool*, IEEE International Conference on Fuzzy Systems, 1166-1173, 2013.

[11] P.P. Bonissone, J.M. Cadenas, M.C. Garrido, R. A. Díaz-Valladares, *A fuzzy random forest*, International Journal of Approximate Reasoning 51(7):729-747, 2010.

[12] J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, *KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework*, Journal of Multiple-Valued Logic and Soft Computing 17(2-3): 255-287, 2011.

[13] A. Frank and A. Asuncion, *UCI Machine Learning Repository*, University of California, Irvine, School of Information and Computer Sciences, http://archive.ics.uci.edu/ml, 2010.