



UNIVERSIDAD
DE CÓRDOBA

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA

Implementación de una interfaz para el algoritmo
NSLVOrd en la biblioteca ORCA

Manual de Código

Autor

Federico García-Arévalo Calles

Directores

Pedro Antonio Gutiérrez Peña

Juan Carlos Gámez Granados

ESCUELA POLITÉCNICA
SUPERIOR DE CÓRDOBA
Universidad de Córdoba



ESCUELA POLITÉCNICA SUPERIOR

—
Córdoba, Julio de 2020

IMPLEMENTACIÓN DE UNA INTERFAZ PARA EL ALGORITMO NSLVORD EN LA BIBLIOTECA ORCA.

Firmado en Córdoba, Julio de 2020

Directores:

Fdo: Pedro Antonio Gutiérrez Peña

Fdo: Juan Carlos Gámez Granados

Autor:

Fdo: Federico García-Arévalo Calles

D./Dña. Pedro Antonio Gutiérrez Peña profesor/a del departamento
Informática y Análisis Numérico informa que el Trabajo Fin de Grado

Implementación de una interfaz para el algoritmo NSLVOrd en la biblioteca ORCA

realizado por el alumno/a D./Dña.

Federico Garcia-Arévalo Calles del Grado Ing. Informática

cumple con los requisitos y nivel exigido.

Córdoba, 30 de junio de 2020

Firmado por GUTIERREZ PEÑA
PEDRO ANTONIO - 45735457A el
día 30/06/2020 con un
certificado emitido por AC

Fdo. D./Dña. Pedro Antonio Gutiérrez Peña



UNIVERSIDAD DE CÓRDOBA

D./Dña. Juan Carlos Gámez Granados profesor/a del departamento
Ingeniería Electrónica y de Computadores informa que el Trabajo Fin de Grado
Implementación de una interfaz para el algoritmo NSLVOrd en la biblioteca ORCA realizado por el alumno/a D./Dña.
Federico León García-Arévalo Calles del Grado Ing. Informática
cumple con los requisitos y nivel exigido.

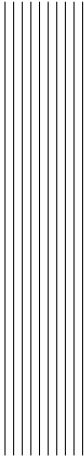
Córdoba, 30 de junio de 2020

Fdo. D./Dña. Juan Carlos Gámez Granados



UNIVERSIDAD DE CÓRDOBA

(+34) 957 21 83 15 / (+34) 957 21 83 20 - direccioneeps@uco.es
Campus Univ. de Rabanales, Ctra. Madrid-Cádiz Km. 396
14071 Córdoba - www.uco.es/eps



ÍNDICE GENERAL

Índice General	IX
Índice de Código	XI
1. Introducción	1
2. Módulo ORCA	3
2.1. Clase Utilities	3
2.2. Clase Dataset	19
2.3. Clase Experiment	24
3. Módulo Read File	31
3.1. Clase TFGFileReadClass	31
3.2. Clase ReadFileCommon	33
3.3. Clase matlab	35
3.4. Clase weka	36
4. Módulo Algorithms	41
4.1. Clase Algorithm	41

4.2. Clase NSLVOOrd	47
5. Módulo NSLVOOrd	57
5.1. Clase NSLVOOrdJava	57
5.2. Clase RuleSystem	67
6. Módulo Rule View	73
6.1. Clase RulesVisual	73
6.2. Función Visual	76
6.3. Clase VisualRules	78
6.4. Clase Rule	85
6.5. Clase ConditionCategoric	91
6.6. Clase ConditionFuzzyLogic	94
7. Módulo JFML	99
7.1. Clase RulesExport	99
7.2. Función JFML	102



ÍNDICE DE CÓDIGO

2.1.	Archivo <i>Utilities.m</i> correspondiente al módulo ORCA	3
2.2.	Archivo <i>Dataset.m</i> correspondiente al módulo ORCA	19
2.3.	Archivo <i>Dataset.m</i> correspondiente al módulo ORCA	24
3.1.	Archivo <i>TFGFileReadClass.m</i> correspondiente al módulo Read File	31
3.2.	Archivo <i>ReadFileCommon.m</i> correspondiente al módulo Read File	33
3.3.	Archivo <i>matlab.m</i> correspondiente al módulo Read File . . .	35
3.4.	Archivo <i>weka.m</i> correspondiente al módulo Read File	36
4.1.	Archivo <i>Algorithm.m</i> correspondiente al módulo Algorithms	41
4.2.	Archivo <i>NSLVOrd.m</i> correspondiente al módulo Algorithms .	47
5.1.	Archivo <i>NSLVOrdJava.java</i> correspondiente al módulo NSLVOrd	57
5.2.	Archivo <i>RuleSystem.java</i> correspondiente al módulo NSLVOrd	67
6.1.	Archivo <i>RulesVisual.m</i> correspondiente al módulo Rule View	73
6.2.	Archivo <i>Visual.java</i> correspondiente al módulo Rule View . .	76
6.3.	Archivo <i>VisualRules.java</i> correspondiente al módulo Rule View	78
6.4.	Archivo <i>Rule.java</i> correspondiente al módulo Rule View . . .	85

6.5.	Archivo <i>ConditionCategoric.java</i> correspondiente al módulo Rule View	91
6.6.	Archivo <i>ConditionFuzzyLogic.java</i> correspondiente al módulo Rule View	94
7.1.	Archivo <i>RulesExport.m</i> correspondiente al módulo JFML . .	99
7.2.	Archivo <i>JFML.m</i> correspondiente al módulo JFML	102



1 Introducción

Este documento corresponde al manual de código del Trabajo Fin de Grado “*Implementación de una interfaz para el algoritmo NSLVOrd en la biblioteca ORCA*”, en el que se muestran los códigos que se han creado y modificado para este trabajo. Los ficheros se agruparán según al módulo que pertenecen:

- **ORCA:** este es el módulo principal del sistema que se encarga configurar el experimento, calcular las métricas de los resultados y exportar toda la información. Está codificado en lenguaje Matlab.
- **Read File:** este módulo se encarga de leer los ficheros de entrenamiento y test; devolviendo los datos separando las entradas y salidas. Está codificado en lenguaje Matlab.
- **Algorithms:** este módulo se encarga de contener diferentes algoritmos y de la ejecución de estos. Está codificado en lenguaje Matlab.
- **NSLVOrd:** este módulo es el que se encarga de ejecutar el algoritmo NSLVOrd. Está codificado en Java.
- **Rule View:** este módulo es el que se encarga de mostrar las reglas generadas por el algoritmo en una ventana de tal forma que cualquier tipo de usuario pueda entenderlas. Está codificado en Java.

- **JFML:** este módulo es el encargado de exportar las reglas generadas por el algoritmo a ficheros XML en formato JFML y PMML en un directorio especificado por el usuario. Está codificado en Java.

2 Módulo ORCA

2.1. Clase Utilities

La clase *Utilities* es la principal del sistema y se encarga de la configuración del experimento como de guardar los resultados de este. En el Código 2.1 se muestra la implementación de esta clase.

```
1 classdef Utilities < handle
2     %UTILITIES Static class that contains several methods for
        configuring
3     % and running the experiments. It allows experiments CPU
        parallelization.
4     % Examples of integration with HTCondor are provided src/condor
        folder.
5     %
6     % UTILITIES methods:
7     %     runExperiments           - setting and running experiments
8     %     runExperimentFold        - Launchs a single experiment fold
9     %     configureExperiment      - sets configuration of the several
        experiments
10    %     results                   - creates experiments reports
11    %
12    % This file is part of ORCA: https://github.com/ayrna/orca
13    % Original authors: Pedro Antonio Guti  rrez, Mar  a P  rez Ortiz ,
        Javier S  nchez Monedero
14    % Citation: If you use this code, please cite the associated paper
        http://www.uco.es/grupos/ayrna/orreview
15    % Copyright:
16    %     This software is released under the The GNU General Public
        License v3.0 licence
```

```

17 %         available at http://www.gnu.org/licenses/gpl-3.0.html
18
19 properties
20
21 end
22
23
24 methods (Static = true)
25     function [logsDir] = runExperiments(expFile, varargin)
26         %RUNEXPERIMENTS Function for setting and running the
           experiments
27         % [LOGSDIR] = RUNEXPERIMENTS(EXPFILE) runs
28         % experiments described in EXPFILE and returns the folder
29         % name LOGSDIR that stores all the results. LOGSDIR is
30         % generated based on the date and time of the system.
31         %
32         % [LOGSDIR] = RUNEXPERIMENTS(EXPFILE, options) runs
33         % experiments described in EXPFILE and returns the folder
34         % name LOGSDIR that stores all the results. Options are:
35         %     - 'parallel': 'false' or 'true' to activate CPU
           parallel
36         %     processing of databases's folds. Default is 'false'
37         %     - 'numcores': default maximum number of cores or
           desired
38         %     number. If parallel = 1 and numcores <2 it sets the
           number
39         %     to maximum number of cores.
40         %     - 'closepool': whether to close or not the pool after
41         %     experiments. Default 'true'. Disabling it can speed
42         %     up consecutive calls to runExperiments.
43         %
44         % Examples:
45         %
46         % Runs parallel folds with 3 workers:
47         % Utilities.runExperiments('tests/cvtests-30-holdout/kdlor.
           ini', 'parallel', 1, 'numcores', 3)
48         % Runs parallel folds with max workers:
49         % Utilities.runExperiments('tests/cvtests-30-holdout/kdlor.
           ini', 'parallel', 1)
50         % Runs parallel folds with max workers and do not close the
51         % pool:
52         % Utilities.runExperiments('tests/cvtests-30-holdout/kdlor.
           ini', 'parallel', 1, 'closepool', false)
53         % Utilities.runExperiments('tests/cvtests-30-holdout/svorim.
           ini', 'parallel', 1, 'closepool', false)
54         %
55         addpath(fullfile(fileparts(which('Utilities.m')), 'Measures'));
56         addpath(fullfile(fileparts(which('Utilities.m')), 'Algorithms'))
           ;
57
58         disp('Setting up experiments...');
59
60         %TODO: move ID generation to configureExperiment?

```


2.1. Clase Utilities

```
61     c = clock;
62     dirSuffix = [num2str(c(1)) '-' num2str(c(2)) '-' num2str(c(3))
63                 '-' num2str(c(4)) '-' num2str(c(5)) '-' num2str(uint8(c(6))
64                 )]);
65     logsDir = Utilities.configureExperiment(expFile, dirSuffix);
66     expFiles = dir([logsDir '/' 'exp-*']);
67
68     % Parse options.
69     op = Utilities.parseParArgs(varargin);
70     myExperiment = Experiment;
71
72     if op.parallel
73         Utilities.preparePool(op.numcores)
74         if (exist('OCTAVE_VERSION', 'builtin') > 0)
75             logsCell = cell(numel(expFiles), 1);
76             logsCell(:) = logsDir;
77             parcellfun(op.numcores, @(varargin) Utilities.
78                 octaveParallelAuxFunction(varargin{:}), num2cell(
79                     expFiles), logsCell);
80
81         else
82             parfor i=1:numel(expFiles)
83                 if ~strcmp(expFiles(i).name(end), '~')
84                     disp(['Running experiment ', expFiles(i).name])
85                     ;
86                     myExperiment.launch([logsDir '/' expFiles(i).
87                         name]);
88             end
89         end
90     end
91
92     Utilities.closePool()
93 else
94     for i=1:numel(expFiles)
95         if ~strcmp(expFiles(i).name(end), '~')
96             disp(['Running experiment ', expFiles(i).name]);
97             myExperiment.launch([logsDir '/' expFiles(i).name])
98             ;
99         end
100     end
101
102     disp('Calculating results...');
103     % Train results (note last argument)
104
105     Utilities.results([logsDir '/' 'Results'], 'report_sum',
106         myExperiment.report_sum, 'train', true);
107
108     % Test results
109     Utilities.results([logsDir '/' 'Results'], 'report_sum',
110         myExperiment.report_sum);
111
112     %mpath('Measures');
113     %mpath('Algorithms');
114
115 end
```

```

105
106     function octaveParallelAuxFunction(experimentToRun, logsDir)
107         %OCTAVEPARALLELAUXFUNCTION Function for running one experiment
108         %   file
109         %   It is used in Octave because it Octave does not have parfor
110         %   OCTAVEPARALLELAUXFUNCTION(EXPERIMENT,LOGSDIR) run the
111         %   experiment
112         %   named EXPERIMENT and contained in the folder LOGSDIR
113         if ~strcmp(experimentToRun.name(end), '~')
114             myExperiment = Experiment;
115             disp(['Running experiment ', experimentToRun.name]);
116             myExperiment.launch([logsDir '/' experimentToRun.name]);
117         end
118     end
119
120     function results(experiment_folder, varargin)
121         %RESULTS Function for computing the results
122         %   RESULTS(EXPERIMENT_FOLDER) computes results of predictions
123         %   stored in EXPERIMENT_FOLDER. It generates CSV files with
124         %   several performance metrics of the testing (generalization)
125         %   predictions.
126         %   * |mean-results-test.csv|: CSV file with datasets in
127         %   files
128         %   and performance metrics in columns. For each metric two
129         %   columns
130         %   are created (mean and standard deviation considering
131         %   the _k_ folds
132         %   of the experiment).
133         %   * |mean-results-matrices-sum-test.csv|: CSV file with
134         %   performance metrics calculated using the sum of all the
135         %   confusion matrices of the _k_ experiments (as Weka
136         %   does). Each column
137         %   presents the performance of this single matrix.
138         %
139         %   RESULTS(EXPERIMENT_FOLDER, 'TRAIN', true) same as
140         %   RESULTS(EXPERIMENT_FOLDER) but calculates performance in
141         %   train
142         %   data. It can be usefull to evaluate overfitting.
143         %
144         %   See also MEASURES/MZE, MEASURES/MAE, MEASURES/AMAE,
145         %   MEASURES/CCR,
146         %   MEASURES/MMAE, MEASURES/GM, MEASURES/MS, MEASURES/Spearman,
147         %   MEASURES/Tkendall, MEASURES/Wkappa
148
149         addpath(fullfile(fileparts(which('Utilities.m')), 'Measures'));
150         addpath(fullfile(fileparts(which('Utilities.m')), 'Algorithms'));
151         ;
152
153         opt.train = false;
154         opt.report_sum = false;
155
156         opt = parsevarargs(opt, varargin);
157     
```

2.1. Clase Utilities

```
149         experiments = dir(experiment_folder);
150
151     for i=1:numel(experiments)
152         if ~(any(strcmp(experiments(i).name, {'.', '..'}))) &&
            experiments(i).isdir
153             disp([experiment_folder '/' experiments(i).name '/' '
                    dataset'])
154             fid = fopen([experiment_folder '/' experiments(i).name
                    '/' 'dataset'], 'r');
155             datasetPath = fgetl(fid);
156             fclose(fid);
157
158             if opt.train
159                 predicted_files = dir([experiment_folder '/'
                    experiments(i).name '/' 'Predictions' '/' '
                    train_*']);
160             else
161                 predicted_files = dir([experiment_folder '/'
                    experiments(i).name '/' 'Predictions' '/' '
                    test_*']);
162             end
163             % Check if we have a missing fold experiment.
164             %-2 is to compensate . and ..
165             predicted_files_train = dir([experiment_folder '/'
                    experiments(i).name '/' 'Predictions' '/' 'train_*'
                    ]);
166             predicted_files_test = dir([experiment_folder '/'
                    experiments(i).name '/' 'Predictions' '/' 'test_*'
                    ]);
167
168             if (numel(predicted_files_train)+numel(
                    predicted_files_test)) ~= numel(dir(datasetPath))
                    -2
169                 warning(sprintf(' \n ***** \n The execution of
                    some folds failed. Number of experiments
                    differs from number of train-test files. \n
                    ***** \n'))
170             end
171             time_files = dir([experiment_folder '/' experiments(i).
                    name '/' 'Times' '/' '.*']);
172             hyp_files = dir([experiment_folder '/' experiments(i).
                    name '/' 'OptHyperparams' '/' '.*']);
173
174             if opt.train
175                 guess_files = dir([experiment_folder '/'
                    experiments(i).name '/' 'Guess' '/' 'train_*'])
                    ;
176             else
177                 guess_files = dir([experiment_folder '/'
                    experiments(i).name '/' 'Guess' '/' 'test_*']);
178             end
179
180             % Discard "." and ".."
```

```

181         if ~(exist('OCTAVE_VERSION', 'builtin') > 0)
182             time_files = time_files(3:numel(time_files));
183             hyp_files = hyp_files(3:numel(hyp_files));
184         end
185
186         if opt.train
187             real_files = dir([datasetPath '/' 'train-*']);
188         else
189             real_files = dir([datasetPath '/' 'test-*']);
190         end
191
192         act = cell(1, numel(predicted_files));
193         pred = cell(1, numel(predicted_files));
194         proj = cell(1, numel(guess_files));
195
196         times = zeros(3,numel(predicted_files));
197         param = [];
198
199         for j=1:numel(predicted_files)
200             pred{j} = importdata([experiment_folder '/'
201                                 experiments(i).name '/' 'Predictions' '/'
202                                 predicted_files(j).name]);
203             times(:,j) = importdata([experiment_folder '/'
204                                     experiments(i).name '/' 'Times' '/' time_files(
205                                         j).name]);
206             proj{j} = importdata([experiment_folder '/'
207                                 experiments(i).name '/' 'Guess' '/' guess_files
208                                     (j).name]);
209
210             if ~isempty(hyp_files)
211                 struct_hyperparams(j) = importdata([
212                     experiment_folder '/' experiments(i).name '
213                     '/' 'OptHyperparams' '/' hyp_files(j).name],
214                     ',');
215                 for z = 1:numel(struct_hyperparams(j).data)
216                     param(z,j) = struct_hyperparams(j).data(z);
217                 end
218             end
219
220             actual = TFGFileReadClass().ReadFile(datasetPath,
221                                                     real_files(j).name,0);
222             act{j} = actual.targets;
223         end
224
225         names = {'Dataset', 'Acc', 'GM', 'MS', 'MAE', 'AMAE', '
226                 MMAE', 'RSpearman', 'Tkendall', 'Wkappa', 'TrainTime
227                 ', 'TestTime', 'CrossvalTime'};
228
229         if ~isempty(hyp_files)
230             for j=1:numel(struct_hyperparams(1).textdata)
231                 names{numel(names)+1} = struct_hyperparams(1).
232                     textdata{j};
233             end
234         end

```

2.1. Clase Utilities

```
221         end
222     end
223
224     if exist ('OCTAVE.VERSION', 'builtin') > 0
225         accs = cell2mat(cellfun(@(varargin) CCR.
226             calculateMetric(varargin{:}), act, pred, '
227             UniformOutput', false)) * 100;
228         gms = cell2mat(cellfun(@(varargin) GM.
229             calculateMetric(varargin{:}), act, pred, '
230             UniformOutput', false)) * 100;
231         mss = cell2mat(cellfun(@(varargin) MS.
232             calculateMetric(varargin{:}), act, pred, '
233             UniformOutput', false)) * 100;
234         maes = cell2mat(cellfun(@(varargin) MAE.
235             calculateMetric(varargin{:}), act, pred, '
236             UniformOutput', false));
237         amaes = cell2mat(cellfun(@(varargin) AMAE.
238             calculateMetric(varargin{:}), act, pred, '
239             UniformOutput', false));
240         maxmaes = cell2mat(cellfun(@(varargin) MMAE.
241             calculateMetric(varargin{:}), act, pred, '
242             UniformOutput', false));
243         spearmans = cell2mat(cellfun(@(varargin) Spearman.
244             calculateMetric(varargin{:}), act, pred, '
245             UniformOutput', false));
246         kendalls = cell2mat(cellfun(@(varargin) Tkendall.
247             calculateMetric(varargin{:}), act, pred, '
248             UniformOutput', false));
249         wkappas = cell2mat(cellfun(@(varargin) Wkappa.
250             calculateMetric(varargin{:}), act, pred, '
251             UniformOutput', false));
252     else
253         accs = cell2mat(cellfun(@CCR.calculateMetric, act,
254             pred, 'UniformOutput', false)) * 100;
255         gms = cell2mat(cellfun(@GM.calculateMetric, act,
256             pred, 'UniformOutput', false)) * 100;
257         mss = cell2mat(cellfun(@MS.calculateMetric, act,
258             pred, 'UniformOutput', false)) * 100;
259         maes = cell2mat(cellfun(@MAE.calculateMetric, act,
260             pred, 'UniformOutput', false));
261         amaes = cell2mat(cellfun(@MAE.calculateMetric, act,
262             pred, 'UniformOutput', false));
263         maxmaes = cell2mat(cellfun(@MMAE.calculateMetric,
264             act, pred, 'UniformOutput', false));
265         spearmans = cell2mat(cellfun(@Spearman.
266             calculateMetric, act, pred, 'UniformOutput',
267             false));
268         kendalls = cell2mat(cellfun(@Tkendall.
269             calculateMetric, act, pred, 'UniformOutput',
270             false));
271         wkappas = cell2mat(cellfun(@Wkappa.calculateMetric,
272             act, pred, 'UniformOutput', false));
273     end
274 end
```

```

245
246         results_matrix = [accs; gms; mss; maes; amaes; maxmaes;
                           spearmans; kendalls; wkappas; times(1,:); times
                           (2,:); times(3,:)];
247     if ~isempty(hyp_files)
248         for j=1:numel(struct_hyperparams(1).textdata)
249             results_matrix = [results_matrix ; param(j,:)
                               ];
250         end
251     end
252
253     results_matrix = results_matrix';
254
255     % Results for the independent dataset
256     if opt.train
257         fid = fopen([experiment_folder '/' experiments(i).
                     name '/' 'results_train.csv'], 'w');
258     else
259         fid = fopen([experiment_folder '/' experiments(i).
                     name '/' 'results_test.csv'], 'w');
260     end
261
262     for h = 1:numel(names)
263         fprintf(fid, '%s', names{h});
264     end
265     fprintf(fid, '\n');
266
267     for h = 1:size(results_matrix,1)
268         fprintf(fid, '%s', real_files(h).name);
269         for z = 1:size(results_matrix,2)
270             fprintf(fid, '%f', results_matrix(h,z));
271         end
272         fprintf(fid, '\n');
273     end
274     fclose(fid);
275
276     % Confusion matrices and sum of confusion matrices
277     if opt.report_sum
278         if opt.train
279             fid = fopen([experiment_folder '/' experiments(
280                         i).name '/' 'matrices_train.txt'], 'w');
281         else
282             fid = fopen([experiment_folder '/' experiments(
283                         i).name '/' 'matrices_test.txt'], 'w');
284         end
285
286         J = length(unique(act{1}));
287         cm_sum = zeros(J);
288         for h = 1:size(results_matrix,1)
289             fprintf(fid, '%s\n-----\n', real_files(h).
290                     name);
291             cm = confusionmat(act{h},pred{h});
292             cm_sum = cm_sum + cm;

```

2.1. Clase Utilities

```
290         for ii = 1:size(cm,1)
291             for jj = 1:size(cm,2)
292                 fprintf(fid, '%d ', cm(ii,jj));
293             end
294             fprintf(fid, '\n');
295         end
296     end
297     fclose(fid);
298
299     % Calculate metrics with the sum of confusion
300     % matrices
301     accs_sum = CCR.calculateMetric(cm_sum) * 100;
302     gms_sum = GM.calculateMetric(cm_sum) * 100;
303     mss_sum = MS.calculateMetric(cm_sum) * 100;
304     maes_sum = MAE.calculateMetric(cm_sum);
305     amaes_sum = AMAE.calculateMetric(cm_sum);
306     maxmaes_sum = MMAE.calculateMetric(cm_sum);
307     spearmans_sum = Spearman.calculateMetric(cm_sum);
308     kendalls_sum = Tkendall.calculateMetric(cm_sum);
309     wkappas_sum = Wkappa.calculateMetric(cm_sum);
310     results_matrix_sum = [accs_sum; gms_sum; mss_sum;
311                          maes_sum; amaes_sum; maxmaes_sum; spearmans_sum
312                          ; kendalls_sum; wkappas_sum; sum(times(1,:));
313                          sum(times(2,:)); sum(times(3,:))];
314
315     results_matrix_sum = results_matrix_sum';
316 end
317
318 means = mean(results_matrix,1);
319 stdev = std(results_matrix,0,1);
320
321 if opt.train
322     if ~exist([experiment_folder '/' 'mean-
323               results_train.csv'],'file')
324         add_head = 1;
325     else
326         add_head = 0;
327     end
328     fid = fopen([experiment_folder '/' 'mean-
329                 results_train.csv'],'at');
330 else
331     if ~exist([experiment_folder '/' 'mean-results_test
332               .csv'],'file')
333         add_head = 1;
334     else
335         add_head = 0;
336     end
337     fid = fopen([experiment_folder '/' 'mean-
338                 results_test.csv'],'at');
```

```

335         if add_head
336             fprintf(fid, 'Dataset-Experiment, ');
337
338             for h = 2:numel(names)
339                 fprintf(fid, 'Mean%, Std %, ', names{h}, names{h}
340                     });
341             end
342             fprintf(fid, '\n');
343         end
344
345
346         fprintf(fid, '%s', experiments(i).name);
347         for h = 1:numel(means)
348             fprintf(fid, '%f, ', means(h), stdev(h));
349         end
350         fprintf(fid, '\n');
351         fclose(fid);
352
353
354         %Confusion matrices and sum of confusion matrices
355         if opt.report_sum
356             if opt.train
357                 fid = fopen([experiment_folder '/' 'mean-
358                     results_matrices_sum_train.csv'], 'at');
359             else
360                 fid = fopen([experiment_folder '/' 'mean-
361                     results_matrices_sum_test.csv'], 'at');
362             end
363
364             if add_head
365                 fprintf(fid, 'Dataset-Experiment, ');
366
367                 for h = 2:numel(names)
368                     fprintf(fid, '%s', names{h});
369                 end
370                 fprintf(fid, '\n');
371             end
372
373             fprintf(fid, '%s', experiments(i).name);
374             for h = 1:numel(results_matrix_sum)
375                 fprintf(fid, '%f, ', results_matrix_sum(h));
376             end
377             fprintf(fid, '\n');
378             fclose(fid);
379         end
380
381     end
382
383     rmpath(fullfile(fileparts(which('Utilities.m')), 'Measures'));
384     rmpath(fullfile(fileparts(which('Utilities.m')), 'Algorithms'));

```


2.1. Class Utilities

```
385
386     end
387
388     function logsDir = configureExperiment(expFile, dirSuffix)
389         %CONFIGUREEXPERIMENT Function for setting the configuration of
390         the
391         % different experiments.
392         % LOGSDIR = CONFIGUREEXPERIMENT(EXPFILE,DIRSUFFIX) parses
393         EXPFILE and
394         % generates single experiment files describing individual
395         experiment
396         % of each fold. It also creates folders to store
397         predictions
398         % and models for all the partitions. All the resources
399         are
400         % created int exp-DIRSUFFIX folder.
401         if( ~(exist(expFile, 'file'))
402             error('The file %s does not exist\n', expFile);
403         end
404
405         logsDir = ['Experiments' '/' 'exp-' dirSuffix];
406         resultsDir = [logsDir '/' 'Results'];
407         if ~exist('Experiments', 'dir')
408             mkdir('Experiments');
409         end
410         mkdir(logsDir);
411         mkdir(resultsDir);
412
413         %Load and parse conf file
414         cObj = Config(expFile);
415
416         num_experiment = numel(cObj.exps);
417         for e = 1:num_experiment
418             expObj = cObj.exps{e};
419
420             id_experiment = expObj.expId;
421             directory = expObj.general('basedir');
422             if ~(exist(directory, 'dir'))
423                 error('Datasets directory "%s" does not exist',
424                     directory)
425             end
426
427             archive = 'matlab';
428             if isKey(expObj.general, 'archive')
429                 archive = lower(expObj.general('archive'));
430             end
431
432             TFGFileReadClass().Valid_archive(archive);
433
434             datasets = expObj.general('datasets');
435             conf_file = [logsDir '/' 'exp-' id_experiment];
436             [matchstart, matchend, tokenindices, matchstring, tokenstring,
437                 tokename, datasetsList] = regexpi(datasets, ',');
```

```

431         %Check that all datasets partitions are accesible
432         %The method checkDatasets calls error
433         Utilities.checkDatasets(directory, datasets, archive);
434
435         [train, test] = Utilities.processDirectory(directory,
436             datasetsList, archive);
437
438         % Generate one config file and corresponding directories
439         % for each fold.
440         for i=1:numel(train)
441             aux_directory = [resultsDir '/' datasetsList{i} '-'
442                 id_experiment];
443             mkdir(aux_directory);
444
445             mkdir([aux_directory '/' 'OptHyperparams']);
446             mkdir([aux_directory '/' 'Times']);
447             mkdir([aux_directory '/' 'Models']);
448             mkdir([aux_directory '/' 'Predictions']);
449             mkdir([aux_directory '/' 'Guess']);
450
451             file = [resultsDir '/' datasetsList{i} '-'
452                 id_experiment '/' 'dataset'];
453             fich = fopen(file, 'w');
454             fprintf(fich, [directory '/' datasetsList{i} '/'
455                 archive]);
456             fclose(fich);
457
458             runfolds = numel(train{i});
459             for j=1:runfolds
460                 iniFile = [conf_file '-' datasetsList{i} '-'
461                     num2str(j) '.ini'];
462
463                 expObj.general('directory') = [directory '/'
464                     datasetsList{i} '/' archive];
465                 expObj.general('train') = train{i}(j).name;
466                 expObj.general('test') = test{i}(j).name;
467                 expObj.general('results') = [resultsDir '/'
468                     datasetsList{i} '-' id_experiment];
469
470                 expObj.writeIni(iniFile);
471             end
472         end
473     end
474 end
475
476 function runExperimentFold(confFile)
477     %RUNEXPERIMENTFOLD(CONFFILE) launch a single experiment
478     % described in
479     % file CONFFILE
480     addpath(fullfile(fileparts(which('Utilities.m'))), 'Measures'));
481     addpath(fullfile(fileparts(which('Utilities.m'))), 'Algorithms'));
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

2.1. Clase Utilities

```
475         auxiliar = Experiment;
476         auxiliar.launch(confFile);
477
478         rmpath(fullfile(fileparts(which('Utilities.m')), 'Measures'));
479         rmpath(fullfile(fileparts(which('Utilities.m')), 'Algorithms'));
480
481     end
482 end
483
484 methods(Static = true, Access = private)
485
486     function [trainFileNames, testFileNames] = processDirectory(
487         directory, dataSetNames, archive)
488         %PROCESSDIRECTORY Function to get all the train and test pair
489         % of
490         % files of dataset's folds
491         % [TRAINFILENAMES, TESTFILENAMES] = PROCESSDIRECTORY(
492         % DIRECTORY, DATASETNAMES)
493         % process comma separated list of datasets names in
494         % DIRECTORY.
495         % All the dataset's folders need to be stored in DIRECTORY.
496         % Returns all the pairs of train-test files in TRAINFILENAMES
497         % and
498         % TESTFILENAMES.
499         % [TRAINFILENAMES, TESTFILENAMES] = PROCESSDIRECTORY(
500         % DIRECTORY,
501         % 'all') process all datasets in DIRECTORY.
502         dbs = dir(directory);
503         dbs(2) = [];
504         dbs(1) = [];
505         validDataSets = 1;
506
507         trainFileNames = cell(numel(dataSetNames), 1);
508         testFileNames = cell(numel(dataSetNames), 1);
509         for j=1:numel(dataSetNames)
510             dsdirectory = [directory '/' dataSetNames{j}];
511             if(isdir(dsdirectory))
512                 [trainFileNames{validDataSets}, testFileNames{
513                     validDataSets}] = ...
514                     TFGFileReadClass().TFGFileName(dsdirectory, archive
515                         , dataSetNames{j});
516
517                 validDataSets = validDataSets + 1;
518             end
519         end
520     end
521
522     function checkDatasets(basedir, datasets, archive)
523         %CHECKDATASETS Test datasets are accessible and with expected
524         %names. Launch error in case a dataset is not found.
525         % CHECKDATASETS(BASEDIR, DATASETS) tests all DATASETS (comma
526         % separated list of datasets) in directory BASEDIR.
```

```

520         if ~exist(basedir, 'dir')
521             error('Datasets directory "%s" does not exist', basedir)
522         end
523
524         dsdirsCell = regexp(datasets, '((\\w|-|_)+(\\w*))', 'tokens');
525         for i=1:length(dsdirsCell) %skip . and ..
526             dsName = dsdirsCell{i};
527             dsName = dsName{:};
528             if ~exist([basedir '/' dsName], 'dir')
529                 error('Dataset directory "%s" does not exist', [basedir
530                     '/' dsName])
531             end
532
533             datasetPath = [basedir '/' dsName '/' archive];
534             dsTrainFiles = dir([datasetPath '/train*']);
535
536             %Test every train file has a test file
537             for f=1:length(dsTrainFiles)
538                 trainName = dsTrainFiles(f).name;
539                 testName = strrep(trainName, 'train', 'test');
540
541                 cellData = TFGFileReadClass().ReadFile(datasetPath,
542                     trainName,0);
543                 trainData = [cellData.patterns cellData.targets];
544                 cellData = TFGFileReadClass().ReadFile(datasetPath,
545                     testName,0);
546                 testData = [cellData.patterns cellData.targets];
547
548                 if size(trainData,2) ~= size(testData,2)
549                     error('Train and test data dimensions do not agree
550                         for dataset "%s"', dsName)
551                 end
552             end
553         end
554     end
555
556     function preparePool(numcores)
557         %PREPAREPOOL(NUMCORES) creates a pool of workers. Function to
558         %abstract code from different matlab versions. Adapt the pool
559         %to the desired number of cores. If there is a current pool
560         %with
561         %desired number of cores do not open again to save time
562         if (exist('OCTAVE_VERSION', 'builtin') > 0)
563             maximum_ncores = nproc;
564         else
565             maximum_ncores = feature('numCores');
566         end
567
568         %Adjust number of cores
569         if numcores > maximum_ncores

```

2.1. Clase Utilities

```
568         disp(['Number of cores was too high and was set up to the
569               maximum available: ' num2str(feature('numCores')) ])
570     numcores = maximum_ncores;
571 end
572 %Check size of the pool
573 if (exist('OCTAVE_VERSION', 'builtin') > 0)
574     pkg load parallel;
575 else
576     if verLessThan('matlab', '8.3')
577         poolsize = matlabpool('size');
578         if poolsize > 0
579             if poolsize ~= numcores
580                 matlabpool close;
581                 matlabpool(numcores);
582             end
583         else
584             matlabpool(numcores);
585         end
586     else
587         poolobj = gcp('nocreate'); %If no pool, do not create
588                                     %new one.
589         if ~isempty(poolobj)
590             if poolobj.NumWorkers ~= numcores
591                 numcores = poolobj.NumWorkers;
592                 delete(gcp('nocreate'))
593                 parpool(numcores);
594             end
595         else
596             parpool(numcores);
597         end
598     end
599 end
600
601 function closePool()
602     if (exist('OCTAVE_VERSION', 'builtin') > 0)
603         pkg unload parallel;
604     else
605         if verLessThan('matlab', '8.3')
606             isOpen = matlabpool('size') > 0;
607             if isOpen
608                 matlabpool close;
609             end
610         else
611             delete(gcp('nocreate'))
612         end
613     end
614 end
615
616 function options = parseParArgs(varargin)
617     %OPTIONS = PARSEPARARGS(VARARGIN) parses parallelization
618     %options with are:
```

```

619         %- 'parallel': 'false' or 'true' to activate, default 'false'
620         %- 'numcores': default maximum number of cores or desired
621         %-   number. If parallel = 1 and numcores <2 it sets the number
622         %-   to maximum number of cores.
623         %- 'closepool': whether to close or not the pool after
624         %-   experiments. Default 'true'
625         % Solution adapted from https://stackoverflow.com/questions/2775263/how-to-deal-with-name-value-pairs-of-function-arguments-in-matlab#2776238
626
627         if (exist('OCTAVE_VERSION', 'builtin') > 0)
628             maximum_ncores = nproc;
629         else
630             maximum_ncores = feature('numCores');
631         end
632
633         options = struct('parallel', false, 'numcores', maximum_ncores, '
        closepool', true);
634
635         varargin = varargin{:};
636         if ~isempty(varargin)
637             options = parsevarargs(options, varargin);
638             if options.parallel && options.numcores <2
639                 disp('Number of cores to low, setting to default number
        of cores')
640                 options.numcores = maximum_ncores;
641             end
642         end
643     end
644
645 end
646 end

```

CÓDIGO 2.1: Archivo *Utilities.m* correspondiente al módulo ORCA

2.2. Clase Dataset

La clase *Dataset* se encarga de la lectura y el preprocesamiento de los datos. En el Código 2.2 se muestra la implementación de esta clase.

```
1 classdef DataSet < handle
2     %DATASET Class to specify the name of the datasets and perform data
        preprocessing
3     %
4     % This file is part of ORCA: https://github.com/ayrna/orca
5     % Original authors: Pedro Antonio Guti  rrez, Mar  a P  rez Ortiz ,
        Javier S  nchez Monedero
6     % Citation: If you use this code, please cite the associated paper
        http://www.uco.es/grupos/ayrna/orreview
7     % Copyright:
8     % This software is released under the The GNU General Public
        License v3.0 licence
9     % available at http://www.gnu.org/licenses/gpl-3.0.html
10
11 properties
12     directory = '';
13     train = '';
14     test = '';
15     standarize = true;
16     dataname = '';
17     nOfFolds = 5;
18 end
19
20 methods
21     function obj = dataSet(direct)
22         if (nargin ~= 0)
23             obj.directory = direct;
24         end
25     end
26
27     function obj = set.directory(obj,direc)
28         if isdir(direc)
29             obj.directory = direc;
30         else
31             error('%s --> Not a directory', direc);
32         end
33     end
34
35     function [trainSet, testSet] = preProcessData(obj,method)
36     %PREPROCESSDATA preprocess a data partition, i.e., deletes the
        constant
37     % and non numerical attributes and standarize the data. Test set
38     % is standardised using train mean and standard error.
```

```

40     % [TRAINSET, TESTSET] = PREPROCESSDATA() preprocess dataset and
41     % returns the preprocessed patterns in TRAINSET and TESTSET.
42
43     if(exist([obj.directory '/' obj.train], 'file') && exist([obj.
44         directory '/' obj.test], 'file'))
45         obj.dataname = strrep(obj.train, 'train_', '');
46         trainSet = TFGFileReadClass().ReadFile(obj.directory, obj.
47             train, method.categ);
48         testSet = TFGFileReadClass().ReadFile(obj.directory, obj.
49             test, method.categ);
50
51         if(obj.standarize)
52             % [trainSet, testSet] = obj.deleteNonNumericValues(
53                 trainSet, testSet);
54             [trainSet, testSet] = obj.deleteConstantAttributes(
55                 trainSet, testSet);
56             [trainSet, testSet] = obj.standarizeData(trainSet,
57                 testSet);
58             % [trainSet, testSet] = obj.scaleData(trainSet, testSet);
59         end
60
61         datasetname=[obj.directory '/' obj.train];
62         [matchstart, matchend] = regexpi(datasetname, '/');
63         trainSet.name = datasetname(matchend(end)+1:end);
64
65         datasetname=[obj.directory '/' obj.test];
66         [matchstart, matchend] = regexpi(datasetname, '/');
67         testSet.name = datasetname(matchend(end)+1:end);
68     else
69         error('Can not find the files');
70     end
71 end
72
73 end
74
75 methods (Static = true)
76
77     %%%%%%%%%%%%%%%
78     %
79     % Function: standarizeData (static)
80     % Description:
81     % Type: It returns the standardized patterns (train and test)
82     % Arguments:
83     %         trainSet—> Array of training patterns
84     %         testSet—> Array of testing patterns
85     %
86     %%%%%%%%%%%%%%%
87
88     function [trainSet, testSet] = standarizeData(trainSet, testSet)
89     %STANDARIZEDATA standarizes a set of training and testing patterns
90     .
91     % [TRAINSET, TESTSET] = STANDARIZEDATA(TRAINSET,TESTSET)

```


2.2. Clase Dataset

```
86         % standarizes TRAINSET and TESTSET with TRAINSET mean and std.
87         [trainSet.patterns, trainMeans, trainStds] = DataSet.
            standarizeFunction(trainSet.patterns);
88         testSet.patterns = DataSet.standarizeFunction(testSet.patterns,
            trainMeans, trainStds);
89     end
90
91     function [XN, XMeans, XStds] = standarizeFunction(X,XMeans,XStds)
92     %STANDARIZEFUNCTION standardises data with patterns stored in rows
93     .
94     % [XN, XMeans, XStds] = standarizeFunction(X) standardises X
95     % using X mean and std. Returns normalised data in XN and
96     % calculated mean and std in XMEANS and XSTDS respectively
97     % [XN, XMeans, XStds] = standarizeFunction(X,XMeans,XStds)
98     % standardises X
99     % using XMeans as mean and XStds as std.
100
101     if (nargin<3)
102         XStds = std(X);
103     end
104     if (nargin<2)
105         XMeans = mean(X);
106     end
107     XN = zeros(size(X));
108     for i=1:size(X,2)
109         XN(:,i) = (X(:,i) - XMeans(i)) / XStds(i);
110     end
111 end
112
113 function [trainSet, testSet] = scaleData(trainSet, testSet)
114 %SCALEDATA scales a set of training and testing patterns.
115 % [TRAINSET, TESTSET] = SCALEDATA(TRAINSET,TESTSET)
116 % scales TRAINSET and TESTSET.
117 for i = 1:size(trainSet.patterns,1)
118     for j = 1:size(trainSet.patterns,2)
119         trainSet.patterns(i,j) = 1/(1+exp(-trainSet.patterns(i,
120             j)));
121     end
122 end
123 for i = 1:size(testSet.patterns,1)
124     for j = 1:size(testSet.patterns,2)
125         testSet.patterns(i,j) = 1/(1+exp(-testSet.patterns(i,j)
126             ));
127     end
128 end
129
130 function [trainSet, testSet] = deleteNonNumericValues(trainSet,
131     testSet)
132 %DELETENONNUMERICVALUES This function deletes non numerical values
133 % in the data, as NaN or Inf.
134 % [TRAINSET, TESTSET] = DELETENONNUMERICVALUES(TRAINSET,TESTSET)
135 % performs data cleaning on arrays of patterns TRAINSET and
```

```

132     TESTSET. Returns
133     % processed matrices.
134     [ fils , cols]=find(isnan(trainSet.patterns) | isinf(trainSet.
135         patterns));
136     cols = unique(cols);
137     for a = size(cols):-1:1
138         trainSet.patterns(:,cols(a)) = [];
139     end
140     [ fils , cols]=find(isnan(trainSet.targets) | isinf(trainSet.
141         targets));
142     cols = unique(cols);
143     for a = size(cols):-1:1
144         trainSet.patterns(:,cols(a)) = [];
145     end
146     [ fils , cols]=find(isnan(testSet.patterns) | isinf(testSet.
147         patterns));
148     cols = unique(cols);
149     for a = size(cols):-1:1
150         testSet.patterns(:,cols(a)) = [];
151     end
152     [ fils , cols]=find(isnan(testSet.targets) | isinf(testSet.targets
153         ));
154     cols = unique(cols);
155     for a = size(cols):-1:1
156         testSet.patterns(:,cols(a)) = [];
157     end
158 end
159
160 function [trainSet , testSet] = deleteConstantAtributes(trainSet ,
161     testSet)
162 %DELETECONSTANTATRIBUTES This function deletes constant variables
163 % [TRAINSET, TESTSET] = DELETECONSTANTATRIBUTES(TRAINSET,TESTSET)
164 % performs data cleaning on arrays of patterns TRAINSET and
165 % TESTSET. Returns
166 % processed matrices.
167
168     all = [trainSet.patterns ; testSet.patterns];
169
170     minvals = min(all);
171     maxvals = max(all);
172
173     r = 0;
174     for k=1:size(trainSet.patterns,2)
175         if minvals(k) == maxvals(k)
176             r = r + 1;
177             index(r) = k;
178         end
179     end
180 end

```

2.2. Clase Dataset

```
178
179         if r > 0
180             r = 0;
181             for k=1:size(index,2)
182                 trainSet.patterns(:,index(k)-r) = [];
183                 testSet.patterns(:,index(k)-r) = [];
184                 r = r + 1;
185             end
186         end
187     end
188 end
189 end
```

CÓDIGO 2.2: Archivo *Dataset.m* correspondiente al módulo ORCA

2.3. Clase Experiment

La clase *Experiment* se encarga de la ejecución del experimento y guardar sus resultados. En el Código 2.3 se muestra la implementación de esta clase.

```

1 classdef Experiment < handle
2     %EXPERIMENT creates an experiment to run an ORCA's experiment which
3     % consist on optimising and running a method in fold (a pair of train
4     % dataset partition). Theexperiment is described by a configuration
5     % file.
6     % This class is used by Utilities to launch a set of experiments
7     %
8     % EXPERIMENT properties:
9     %     data          - DataSet object to store the train/test data
10    %     method         - Method to learn and classify data
11    %     cvCriteria     - Metric to guide the grid search for
12    %                     parameters optimisation
13    %     resultsDir     - Directory to store performance reports and
14    %                     learned models
15    %     seed           - Seed to be used for random number
16    %                     generation
17    %     crossvalide     - Activate corssvalidation
18    %
19    % EXPERIMENT methods:
20    %     launch         - Launch experiment described in file
21    %
22    % This file is part of ORCA: https://github.com/ayrna/orca
23    % Original authors: Pedro Antonio Guti  rrez, Mar  a P  rez Ortiz ,
24    % Javier S  nchez Monedero
25    % Citation: If you use this code, please cite the associated paper
26    % http://www.uco.es/grupos/ayrna/orreview
27    % Copyright:
28    %     This software is released under the The GNU General Public
29    %     License v3.0 licence
30    %     available at http://www.gnu.org/licenses/gpl-3.0.html
31    %
32    properties
33        data = DataSet;
34        method = Algorithm;
35        cvCriteria = MAE;
36        crossvalide = 0;
37        resultsDir = '';
38        %calculate metrics with the sum of matrices (only suitable for
39        %k-fold experimental design)
40        report_sum = 0;
41        seed = 1;
42        parameters; %parameters to optimize
43    end

```

2.3. Clase Experiment

```
37
38     properties (SetAccess = private)
39
40         logsDir
41     end
42
43     methods
44         function obj = launch(obj,expFile)
45             %LAUNCH Launch experiment described in file.
46             % EXPOBJ = LAUNCH(EXPFILE) parses EXPFILE and run the
               experiment
47             % described on it. It performs the following steps:
48             % # Preprocess data cleaning and standardization (option need
               to be activated in configuration file)
49             % # Optimize parameters by performing a grid search (if
               selected
50             % in configuration file)
51             % # Run algorithm with optimal parameters (if crossvalidation
               was
52             % selected)
53             % # Save experiment results for the fold
54             obj.process(expFile);
55             obj.run();
56         end
57     end
58
59     methods(Access = private)
60
61         function obj = run(obj)
62             %RUN do experiment steps: data cleaning and standardization ,
               parameters
63             % optimization and save results
64             [train, test] = obj.data.preProcessData(obj.method); %Modificado
               TFG añadido el parametro obj.method
65
66             if obj.crossvalide
67                 c1 = clock;
68                 Optimals = obj.crossValideParams(train);
69                 c2 = clock;
70                 crossvaltime = etime(c2,c1);
71                 totalResults = obj.method.runAlgorithm(train, test,
               Optimals);
72                 totalResults.crossvaltime = crossvaltime;
73             else
74                 totalResults = obj.method.runAlgorithm(train, test);
75             end
76
77             obj.saveResults(totalResults);
78         end
79
80         function obj = process(obj, fname)
81             %PROCESS parses experiment described in FNAME
82             cObj = Config(fname);
```

```

83     expObj = cObj.exps{:};
84     %Copy ini values to corresponding object properties
85
86     %General experiment properties
87     if expObj.general.isKey('num_folds')
88         obj.data.nOfFolds = str2num(expObj.general('num_folds'));
89     end
90     if expObj.general.isKey('standarize')
91         obj.data.standarize = str2num(expObj.general('standarize'));
92         ;
93     end
94     if expObj.general.isKey('cvmetric')
95         met = upper(expObj.general('cvmetric'));
96         eval(['obj.cvCriteria = ' met ';'']);
97     end
98     if expObj.general.isKey('seed')
99         obj.seed = str2num(expObj.general('seed'));
100     end
101     if expObj.general.isKey('report_sum')
102         obj.report_sum = str2num(expObj.general('report_sum'));
103     end
104
105     try
106         obj.data.directory = expObj.general('directory');
107         obj.data.train = expObj.general('train');
108         obj.data.test = expObj.general('test');
109         obj.resultsDir = expObj.general('results');
110     catch ME
111         error('Configuration file %s does not have minimum fields.
112             Exception %s', fname, ME.identifier)
113     end
114
115     %Algorithm properties are transformed to varargs ('key',value)
116     varargs = obj.mapsToCell(expObj.algorithm);
117     alg = expObj.algorithm('algorithm');
118     obj.method = feval(alg, varargs);
119
120     %Parameters to be optimized
121     if ~isempty(expObj.params)
122         pkeys = expObj.params.keys;
123         for p=1:cast(expObj.params.Count,'int32')
124             %sfield(obj.parameters,' pkeys{p})
125             eval(['obj.parameters.' pkeys{p} ' = [ ' expObj.params(
126                 pkeys{p}) '];']);
127             obj.crossvalide = 1;
128         end
129     end
130
131     end
132
133     function obj = saveResults(obj, TotalResults)
134         %SAVERESULTS saves the results of the experiment and
135         %the best hyperparameters.

```

2.3. Clase Experiment

```
133     par = obj.method.getParameterNames();
134     if ~isempty(par)
135         outputFile = [obj.resultsDir filesep 'OptHyperparams'
136                     filesep obj.data.dataname ];
137         fid = fopen(outputFile, 'w');
138         for i=1:(numel(par))
139             value = getfield(TotalResults.model.parameters, par{i});
140             fprintf(fid, '%f\n', par{i}, value);
141         end
142         fclose(fid);
143     end
144
145     outputFile = [obj.resultsDir filesep 'Times' filesep obj.data.
146                 dataname ];
147     fid = fopen(outputFile, 'w');
148     if obj.crossvalide
149         fprintf(fid, '%f\n%f\n%f', TotalResults.trainTime,
150                 TotalResults.testTime, TotalResults.crossvaltime);
151     else
152         fprintf(fid, '%f\n%f\n%f', TotalResults.trainTime,
153                 TotalResults.testTime, 0);
154     end
155     fclose(fid);
156
157     outputFile = [obj.resultsDir filesep 'Predictions' filesep obj.
158                 data.train ];
159     dlmwrite(outputFile, TotalResults.predictedTrain);
160     outputFile = [obj.resultsDir filesep 'Predictions' filesep obj.
161                 data.test ];
162     dlmwrite(outputFile, TotalResults.predictedTest);
163
164     model = TotalResults.model;
165     % Write complete model
166     outputFile = [obj.resultsDir filesep 'Models' filesep obj.data.
167                 dataname '.mat'];
168     save(outputFile, 'model');
169
170     outputFile = [obj.resultsDir filesep 'Guess' filesep obj.data.
171                 train ];
172     dlmwrite(outputFile, TotalResults.projectedTrain, 'precision',
173             '%.15f');
174
175     outputFile = [obj.resultsDir filesep 'Guess' filesep obj.data.
176                 test ];
177     dlmwrite(outputFile, TotalResults.projectedTest, 'precision',
178             '%.15f');
179
180     if obj.method.export
181         obj.method.export_rules([obj.resultsDir filesep 'Models']);
182     end
```

```

175
176         if obj.method.visual
177             obj.method.visual_rules();
178         end
179     end
180
181     function optimals = crossValideParams(obj,train)
182         %CROSSVALIDEPARAMS Function for performing the crossvalidation
183         %    in a specific train partition.
184
185         %
186         %    OPTIMALS = CROSSVALIDEPARAMS(TRAIN) Divides the data in k-
187         %    folds
188         %    (k defined by 'num fold' in configuration file). Returns
189         %    structure OPTIMALS with optimal parameter(s)
190         optimals = paramopt(obj.method,obj.parameters,train, 'metric',
191             obj.cvCriteria,...
192             'n folds', obj.data.nOfFolds, 'seed', obj.
193             seed);
194
195     end
196
197 end
198
199 methods (Static = true)
200
201     function varargs = mapsToCell(aObj)
202         %varargs = mapsToCell(mapObj) returns key value pairs in a
203         %    comma separated
204         %    string. Example: "'kernel', 'rbf', 'c', 0.1"
205
206         % If there are no parameters return empty cell
207         if aObj.Count == 1
208             varargs = cell(1,1);
209             return
210         end
211
212         mapObj = containers.Map(aObj.keys,aObj.values);
213         mapObj.remove('algorithm');
214         pkeys = mapObj.keys;
215         varargs = cell(1,cast(mapObj.Count,'int32')*2);
216
217         for p=1:2:(cast(mapObj.Count,'int32')*2)
218             p = cast(p,'int32');
219             keyasstr = pkeys(p/2);
220             keyasstr = keyasstr{:};
221             value = mapObj(keyasstr);
222             varargs{1,p} = sprintf('%s', pkeys{p/2});
223             % Check numerical values
224             valuenum = str2double(value);
225             if isnan(valuenum) %we have a string
226                 varargs{1,p+1} = sprintf('%s', value);
227             else %we have a number
228                 varargs{1,p+1} = valuenum;

```


2.3. Clase Experiment

```
223         end
224     end
225 end
226 end
227
228
229 end
```

CÓDIGO 2.3: Archivo *Dataset.m* correspondiente al módulo ORCA

3 Módulo Read File

3.1. Clase TFGFileReadClass

La clase *TFGFileReadClass* sirve de intermediario entre el módulo ORCA y el módulo Read File. En el Código 3.1 se muestra la implementación de esta clase.

```
1 classdef TFGFileReadClass
2     properties (Access = private)
3         path = fullfile(fileparts(which('TFGFileReadClass.m')),'
4             TFGReadFiles');
5     end
6     methods
7         function Valid_archive(obj, archive)
8             addpath(obj.path);
9
10            if exist(fullfile(obj.path,[lower(strtrim(archive)) '.m']), '
11                file ') ~= 2 || strcmpi(archive, 'ReadFileCommon')
12                error('"%s" unsupported file type', archive)
13            end
14
15            rmpath(obj.path);
16        end
17
18        function datas = ReadFile(obj, directory, file, cat)
19            addpath(obj.path);
20
21            folders = strsplit(directory, '/');
22            archive = char(folders(end));
```

```
22
23     TFGReadFiles = feval(archive);
24     raw = [directory '/' file];
25     datas = TFGReadFiles.ReadFileFunction(raw, cat);
26
27     rmpath(obj.path);
28 end
29
30 function [trainFileName, testFileName] = TFGFileName(obj, dsdirectory
    , archive, dataSetName)
31     addpath(obj.path);
32
33     format = feval(archive);
34     [trainFileName, testFileName] = format.FormatFile(dsdirectory,
        archive, dataSetName);
35
36     rmpath(obj.path);
37 end
38 end
39
40 methods (Static, Access = private)
41     function cols = SearchInvalidValue(datas)
42         [~, cols] = find(isnan(datas) | isinf(datas));
43     end
44 end
45 end
```

CÓDIGO 3.1: Archivo *TFGFileReadClass.m* correspondiente al módulo Read File

3.2. Clase ReadFileCommon

3.2. Clase ReadFileCommon

La clase *ReadFileCommon* es una clase abstracta que define los ajustes para los métodos de lectura de datos. En el Código 3.2 se muestra la implementación de esta clase.

```
1 classdef ReadFileCommon < handle
2     properties
3         info = [];
4         categ_att = [];
5     end
6
7     properties (Access = protected)
8         categ = 0;
9     end
10
11    methods
12        function [trainFileName, testFileName] = FormatFile(obj, dsdirectory,
13            archive, dataSetName)
14            [file_train_expr, file_test_expr] = obj.Format(dataSetName);
15
16            file_expr = [dsdirectory '/' archive '/' file_train_expr];
17            trainFileName = dir(file_expr);
18            file_expr = [dsdirectory '/' archive '/' file_test_expr];
19            testFileName = dir(file_expr);
20
21        end
22
23        function [file_train_expr, file_test_expr] = Format(obj, dataSetName
24            )
25            error('format should be implemented in all subclasses');
26        end
27
28        function datas = ReadFileFunction(obj, file, cat)
29            obj.categ = cat;
30            try
31                datas = obj.ReadFile(file);
32            catch ME
33                error('Cannot read file "%s" \n %s', file, ME.message)
34            end
35            datas = obj.deleteNonNumericValues(datas);
36
37            datas.info.personal = obj.info;
38            datas.info.utilities.type = class(obj);
39            datas.info.utilities.categ_att = obj.categ_att;
40        end
41
42        function datas = ReadFile(obj, file)
43            error('ReadFile method should be implemented in all subclasses'
44                );
```

```

41         end
42     end
43
44     methods (Access = private)
45         function datas = deleteNonNumericValues(obj,datas)
46             %Search invalid data on targets
47             [fils ,cols] = find(isnan(datas.targets) | isinf(datas.targets))
48             ;
49             del = fils;
50
51             if obj.categ == 0 || ~iscell(datas.patterns)
52                 %Search invalid data on patterns
53                 [fils ,cols] = find(isnan(datas.patterns) | isinf(datas.
54                     patterns));
55                 del = unique([del;fils]);
56
57             else
58                 for i = 1:size(datas.patterns,1)
59                     for j = 1:size(datas.patterns,2)
60                         if ~ischar(datas.patterns{i,j}) && isnan(datas.
61                             patterns{i,j})
62                             del = [del;i];
63                         end
64                     end
65                 end
66                 del = unique(del);
67
68             end
69
70             if isempty(del)
71                 return;
72             end
73
74             %Delete lines whit invalid data
75             datas.targets(del,:) = [];
76             datas.patterns(del,:) = [];
77         end
78     end
79 end

```

CÓDIGO 3.2: Archivo *ReadFileCommon.m* correspondiente al módulo Read File

3.3. Clase matlab

3.3. Clase matlab

La clase *matlab* es la que se encarga de la lectura de datos de los ficheros de tipo ‘matlab’. En el Código 3.3 se muestra la implementación de esta clase.

```
1 classdef matlab < ReadFileCommon
2     methods
3         function [file_train_expr , file_test_expr] = Format(obj ,dataSetName
4             )
5             file_train_expr = ['train_' dataSetName '.*'];
6             file_test_expr = ['test_' dataSetName '.*'];
7         end
8
9         function datas = ReadFile(obj , file )
10            raw = load(file);
11
12            datas.targets = raw(:,end);
13            datas.patterns = raw(:,1:end-1);
14        end
15 end
```

CÓDIGO 3.3: Archivo *matlab.m* correspondiente al módulo Read File

3.4. Clase weka

La clase *weka* es la que se encarga de la lectura de datos de los ficheros de tipo 'weka'. En el Código 3.4 se muestra la implementación de esta clase.

```

1 classdef weka < ReadFileCommon
2     properties
3         attrs = [];
4     end
5
6     methods
7         function [file_train_expr , file_test_expr] = Format(obj ,dataSetName
8             )
9             file_train_expr = [ 'train_' dataSetName '-*.arff' ];
10            file_test_expr = [ 'test_' dataSetName '-*.arff' ];
11        end
12
13        function datas = ReadFile(obj ,file)
14            [datas.patterns , datas.targets] = obj.ReadWekaFile( file );
15            obj.info.attrs = obj.attrs;
16        end
17
18    methods (Access = private)
19        function [patterns ,targets] = ReadWekaFile(obj ,file_name)
20            file = fopen(file_name , 'rt');
21
22            %Read header
23            obj.ReadHeader( file );
24
25            %Read datas
26            [patterns ,targets] = obj.ReadDatas( file );
27
28            fclose( file );
29        end
30
31        function ReadHeader(obj ,file)
32            while ~feof( file)
33                line = fgetl( file );
34
35                if ~isempty(line)
36                    vec = strsplit(line , ' ');
37                    if strcmpi(vec(1) , '@attribute')
38                        %Check if attribute have a name
39                        %and type
40                        if length(vec) < 3
41                            error( 'Attribute incorrect.' );
42                        end
43

```


3.4. Clase weka

```
44         %Read name and type
45         name = vec(2);
46         aux = strcat(name,{ ' '});
47         aux = aux{1};
48         ini = length(aux) + 12;
49         type = lower(line(ini:end));
50
51         %Add attribute
52         obj.NewAttribute(name,type);
53         elseif strcmpi(vec(1),'@data')
54             if length(obj.attrs) < 2
55                 error('Need more attributes.');
```

end

return;

end

end

end

error('Unrecognized as WEKA format.')

end

function NewAttribute(obj,name,type)

%Comprobar que no exista el nombre en otro atributo

if isempty(obj.attrs)

if ismember(name,[obj.attrs.name])

error('Attributes with same name.');

end

end

%Comprobar el tipo de atributo

info = [];

if ~strcmp(type,'numeric')

indexL = strfind(type,'{');

indexR = strfind(type,'}');

if length(indexL) == 1 && length(indexR) == 1 && indexL ==

1 && indexR == length(type)

type = strrep(type(2:length(type)-1),' ','');

[info,num] = strsplit(type,',');

for i = num

if length(i{1}) ~= 1

error('Categoric attribute without type.');

end

end

type = 'categoric';

else

error('Attributes should be numeric or categoric.');

end

end

% Guardar los datos

aux.name = name;

aux.type = type;

aux.info = info;

obj.attrs = [obj.attrs;aux];

```

96     end
97
98     function [patterns, targets] = ReadDats(obj, file)
99         %Leer los datos
100         datas = [];
101         while ~feof(file)
102             line = fgetl(file);
103             line = strrep(line, ' ', '');
104             if ~isempty(line)
105                 att_dats = strsplit(line, ',');
106                 datas = [datas; att_dats];
107             end
108         end
109         datas = lower(datas);
110
111         if isempty(datas)
112             error('No data found. ');
113         end
114
115         %Guardar las entradas
116         patterns_aux = datas(:, 1:end-1);
117         patterns = [];
118         att_aux = [];
119         for i = 1:size(patterns_aux, 2)
120             if strcmp(obj.attrs(i).type, 'categorical')
121                 if obj.categ == 0
122                     [patt, atti] = obj.ToOneHot(patterns_aux(:, i), obj.
123                         attrs(i));
124                     patterns = [patterns patt];
125                     att_aux = [att_aux; atti];
126                 else
127                     aux = patterns_aux(:, i);
128                     elements = obj.attrs(i).info;
129                     [k, ~] = obj.Categorical_to_Numeric(aux, elements);
130                     aux(find(isnan(k), 1)) = {NaN};
131
132                     patterns = [patterns aux];
133                     att_aux = [att_aux; obj.attrs(i)];
134                 end
135             elseif strcmp(obj.attrs(i).type, 'numeric')
136                 line_aux = zeros(length(patterns_aux(:, i)), 1);
137                 for j = 1:length(line_aux)
138                     line_aux(j) = str2double(patterns_aux(j, i));
139                 end
140                 if obj.categ == 0
141                     patterns = [patterns line_aux];
142                     att_aux = [att_aux; obj.attrs(i)];
143                 else
144                     aux = patterns_aux(:, i);
145                     aux(find(isnan(line_aux), 1)) = {NaN};
146                     patterns = [patterns aux];
147                     att_aux = [att_aux; obj.attrs(i)];
148                 end
149             end
150         end

```

3.4. Clase weka

```
148         else
149             error('Attribute type no valid. ');
150         end
151     end
152     obj.attrs = [att_aux;obj.attrs(end)];
153
154     % Gardar las salidas
155     datas = datas(:,end);
156     att = obj.attrs(end);
157     [targets,obj.attrs(end)] = obj.ToNumeric(datas,att);
158 end
159
160 function [datas,attnew] = ToOneHot(obj,patterns,att)
161     % Convert datas
162     datas = zeros(size(patterns,1),size(att.info,2));
163     for i = 1:size(datas,1)
164         for j = 1:size(datas,2)
165             datas(i,j) = double(strcmp(patterns{i},att.info{j}));
166         end
167     end
168
169     % Check all values are valids
170     ind = ~sum(datas,2);
171     datas(ind,:) = NaN;
172
173     % Update attribute
174     attnew = [];
175     for i = 1:size(att.info,2)
176         att_aux.type = 'categorical';
177         att_aux.name = strcat(att.name, '_',int2str(i));
178         att_aux.info = ['0' '1'];
179         attnew = [attnew;att_aux];
180     end
181 end
182
183 function [datas,att] = ToNumeric(obj,datas,att)
184     if strcmpi(att.type,'numeric')
185         line_aux = zeros(size(datas));
186         for j = 1:size(line_aux,1)
187             line_aux(j) = str2double(datas(j));
188         end
189         datas = line_aux;
190     elseif strcmpi(att.type,'categorical')
191         elements = att.info;
192         [datas,convert] = obj.Categoric_to_Numeric(datas,elements);
193         att.info = convert;
194     end
195 end
196
197 function [final_datas,targets_type] = Categoric_to_Numeric(obj,
198     datas,elements)
199     % Apuntar la conversion
200     targets_type.cat = elements;
```

3. Módulo Read File

```
200         targets_type.num = 1:length(elements);
201
202         %Convertir los datos
203         final_datas = zeros(size(datas,1),size(targets_type.cat,2));
204         for i = 1:size(final_datas,1)
205             for j = 1:size(final_datas,2)
206                 final_datas(i,j) = double(strcmp(datas{i},targets_type.
                    cat{j}));
207             end
208         end
209         final_datas = final_datas * targets_type.num';
210
211         %Comprobar que ninguno sea un valor no valido
212         ind = ~sum(final_datas,2);
213         final_datas(ind,:) = NaN;
214     end
215 end
216 end
```

CÓDIGO 3.4: Archivo *weka.m* correspondiente al módulo Read File

4 Módulo Algorithms

4.1. Clase Algorithm

La clase *Algorithm* es una clase abstracta que define los ajustes para los algoritmos que posee ORCA. En el Código 4.1 se muestra la implementación de esta clase.

```
1 classdef Algorithm < handle
2     %ALGORITHM abstract interface class. Abstract class which defines the
3     %settings for the algorithms (common methods and variables).
4     %
5     % This file is part of ORCA: https://github.com/ayrna/orca
6     % Original authors: Pedro Antonio Guti  rrez, Mar  a P  rez Ortiz ,
7     % Javier S  nchez Monedero
8     % Citation: If you use this code, please cite the associated paper
9     % Copyright:
10    % This software is released under the The GNU General Public
11    % License v3.0 licence
12    % available at http://www.gnu.org/licenses/gpl-3.0.html
13
14    properties
15        model = [];
16        categ = false;
17        export = false;
18        visual = false
19    end
20
21    methods
22        function mInf = runAlgorithm(obj,train , test , param)
```

```

21     %RUNALGORITHM runs the corresponding algorithm, fitting the
22     %model and testing it in a dataset.
23     % mInf = RUNALGORITHM(OBJ, TRAIN, TEST, PARAMETERS) learns a
24     % model with TRAIN data and PARAMETERS as hyper-parameter
25     % structure of values for the method. It tests the
26     % generalization performance with TRAIN and TEST data and
27     % returns predictions and model in mInf structure.
28     if nargin == 3
29         param = [];
30     else
31         % Mix parameters with default
32         obj.setParam(param)
33     end
34     param = obj.parameters;
35
36     c1 = clock;
37     [mInf.projectedTrain, mInf.predictedTrain] = obj.fit(train,
38         param);
39     % Save the model type
40     obj.model.algorithm = class(obj);
41     c2 = clock;
42     mInf.trainTime = etime(c2,c1);
43
44     c1 = clock;
45     [mInf.projectedTest, mInf.predictedTest] = obj.predict(test.
46         patterns);
47     c2 = clock;
48     mInf.testTime = etime(c2,c1);
49     mInf.model = obj.model;
50 end
51
52 function [projectedTrain, predictedTrain] = fit( obj,train,param)
53 %FIT trains the model for the Algorithm method with TRAIN data and
54 % vector of parameters PARAMETERS. Returns the projection of
55 % patterns (only valid for thresshold models) and the predicted
56 % labels.
57 %The model can be accessed thourgh getModel() method.
58 if nargin < 3
59     param = [];
60     if nargin < 2
61         error('Please provide training data')
62     end
63 end
64 if ~all(isfield(train, {'patterns','targets'}))
65     error('Please provide a structure with train patterns and
66         targets')
67 end
68 %check that dimensions agree
69 if ~size(train.patterns,1) == size(train.targets,1)
70     error('Number of train patterns and targets must agree')
71 end
72
73 [projectedTrain, predictedTrain] = obj.privfit(train, param);

```

4.1. Class Algorithm

```
70     end
71
72     function [projected, predicted]= predict(obj,test)
73     %PREDICT predicts labels of TEST patterns labels using fitted
74     MODEL.
75     % Check if there is a model
76     if isempty(obj.model)
77         error('The object does not have a fitted model')
78     end
79     % Avoid typicall error of passing a structure instead of
80     % the matrix of independent variables
81     if ~obj.categ && ~isa(test, 'double')
82         error('test parameter has to be a matrix')
83     end
84
85     [projected, predicted]= privpredict(obj,test);
86
87     % Abstract methods: they have been implemented in this way to
88     % ensure compatibility with Octave. An error is thrown if the
89     % method
90     % is not implemented in child class.
91
92     function [projectedTrain, predictedTrain] = privfit( obj,train,
93     param)
94     %PRIVFIT trains the model for the Algorithm method. It is
95     % called by
96     % super-class Algorithms's 'fit' function. This method is
97     % public, but
98     % should not be called by the user.
99     error('train method should be implemented in all subclasses');
100 end
101
102     function [projected, predicted]= privpredict(obj,test)
103     %PREDICT predicts labels of TEST patterns labels using fitted
104     MODEL.
105     % It is called by super-class Algorithms's 'predict' function.
106     % This method is public, but should not be called by the user.
107     error('test method should be implemented in all subclasses');
108 end
109
110     function parseArgs(obj,varargin)
111     %PARSEARGS(VARARGIN) parses a pair of keys-values in matlab
112     % style format. It throws different exceptions if the field
113     % does
114     % not exists on the class or if the type assignement is not
115     % consistent.
116     if ~isempty(varargin) && ~isempty(varargin{1})
117         while iscell(varargin{1})
118             varargin = varargin{1};
119             if isempty(varargin{1})
120                 return
121             end
122         end
123     end
```

```

115         end
116
117         %# read the acceptable names
118         optionNames = fieldnames(obj);
119
120         %# count arguments
121         nArgs = length(varargin);
122         if mod(nArgs,2)
123             error('parseParArgs needs propertyName/propertyValue
124                 pairs')
125         end
126
127         for pair = reshape(varargin,2,[]) % pair is {propName;
128             propName}
129             propName = pair{1}; %make case insensitive
130
131             if any(strcmp(inpName,optionNames))
132                 % overwrite properties.
133                 % check type
134                 if strcmp(class(obj.(inpName)), class(pair{2}))
135                     obj.(inpName) = pair{2};
136                 else
137                     % Check boolean
138                     if islogical(obj.(inpName)) && ...
139                         (strcmp(pair{2},'true') || strcmp(pair
140                             {2},'false'))
141                         obj.(inpName) = eval(pair{2});
142                     else
143                         msg = sprintf('Data type of property ''%s''
144                             (%s) not compatible with data type (%s
145                             ) of assigned value in configuration
146                             file', ...
147                             inpName, class(obj.(inpName)), class(
148                                 pair{2}));
149                         error(msg);
150                     end
151                 end
152             else
153                 error('Error ''%s'' is not a recognized class
154                     property name',inpName)
155             end
156         end
157     end
158 end
159
160 function setParam(obj,param)
161     %SETPARAM(PARAM) set parameters contained in param and keep
162     default
163     % values of class parameters field. It throws different
164     exceptions if
165     % the field does not exists on the class or if the type
166     assignement is not consistent.
167     % paramNames = fieldnames(obj.parameters);

```


4.1. Clase Algorithm

```
157
158     % Ignore empty argument
159     if isempty(param)
160         return
161     end
162     if ~isstruct(param)
163         error('parameters variable have to be a structure')
164     end
165     paramNames = fieldnames(param);
166
167     for i = 1:length(paramNames)
168         inpName = paramNames{i};
169         if isfield(obj.parameters, inpName)
170             % check type
171             if strcmp(class(obj.parameters.(inpName)), class(param
172                 .(inpName)))
173                 obj.parameters.(inpName) = param.(inpName);
174             else
175                 % Check boolean
176                 if islogical(obj.parameters.(inpName)) && ...
177                     (strcmp(param.(inpName), 'true') || strcmp(
178                         param.(inpName), 'false'))
179                     obj.parameters.(inpName) = eval(pair{2});
180                 else
181                     msg = sprintf('Data type of property ''%s'' (%s
182                         ) not compatible with data type (%s) of
183                         assigned value in configuration file ', ...
184                         inpName, class(obj.parameters.(inpName)),
185                         class(param.(inpName)));
186                     error(msg);
187                 end
188             end
189         else
190             error('Error ''%s'' is not a recognized class parameter
191                 name', inpName)
192         end
193     end
194
195     function m = getModel(obj)
196         m = obj.model;
197     end
198
199     function m = setModel(obj, m)
200         obj.model = m;
201     end
202
203     function name_parameters = getParameterNames(obj)
204         if ~isempty(obj.parameters)
205             name_parameters = sort(fieldnames(obj.parameters));
206         else
207             name_parameters = [];
208         end
209     end
```

```
204     end
205
206     function export_rules(obj,dir)
207         disp('Class ' + class(obj) + ' can not export rules.')
208     end
209
210     function visual_rules(obj)
211         disp('Class ' + class(obj) + ' can not see rules.')
212     end
213 end
214
215 end
```

CÓDIGO 4.1: Archivo *Algorithm.m* correspondiente al módulo Algorithms

4.2. Clase NSLVOOrd

4.2. Clase NSLVOOrd

La clase *NSLVOOrd* es la que contiene los métodos para ejecutar el algoritmo NSLVOOrd. En el Código 4.2 se muestra la implementación de esta clase.

```
1 classdef NSLVOOrd < Algorithm
2     properties
3         description = 'Inclusion del algoritmo NSLVOOrd como TFG de Federico
4             Garcia-Arevalo Calles';
5         %Parameters to optimize and default value
6         parameters = struct('Seed', 1286082570, 'LabelsInputs', 5, '
7             LabelsOutputs', 5,...
8             'Shift', 35, 'Alpha', 0.5, 'Population', -1, '
9             MaxIteration', 500,...
10            'IniProbBin', 0.9, 'CrosProbBin', 0.25, '
11            MutProbBin', 0.5, 'MutProbEachBin',
12            0.17,...
13            'IniProbInt', 0.5, 'CrosProbInt', 0.0, '
14            MutProbInt', 0.5, 'MutProbEachInt', 0.01,...
15            'IniProbReal', 0.0, 'CrosProbReal', 0.25, '
16            MutProbReal', 0.5, 'MutProbEachReal',
17            0.14,...
18            'SeeRules', 0, 'ExportRules', 0);
19
20 end
21
22 methods (Access = private, Static)
23     function param_java = initParameters(param)
24         param_java = {...
25             num2str(param.Seed), ...
26             num2str(param.LabelsInputs), ...
27             num2str(param.LabelsOutputs), ...
28             num2str(param.Shift), ...
29             num2str(param.Alpha), ...
30             num2str(param.Population), ...
31             num2str(param.MaxIteration), ...
32             num2str(param.IniProbBin), ...
33             num2str(param.CrosProbBin), ...
34             num2str(param.MutProbBin), ...
35             num2str(param.MutProbEachBin), ...
36             num2str(param.IniProbInt), ...
37             num2str(param.CrosProbInt), ...
38             num2str(param.MutProbInt), ...
39             num2str(param.MutProbEachInt), ...
40             num2str(param.IniProbReal), ...
41             num2str(param.CrosProbReal), ...
42             num2str(param.MutProbReal), ...
43             num2str(param.MutProbEachReal)};
44 end
```

```

37
38     function header = getHeader(datas)
39         header = '@relation NSLVOrd';
40         if strcmp(datas.info.utilities.type, 'weka')
41             for i = 1:length(datas.info.personal.attrs)-1
42                 line = strcat('@attribute',{ ' ' },datas.info.personal.
43                     attrs(i).name,{ ' ' });
44                 if strcmp(datas.info.personal.attrs(i).type, 'numeric')
45                     line = strcat(line,datas.info.personal.attrs(i).
46                         type);
47                 elseif strcmp(datas.info.personal.attrs(i).type, '
48                     categoric')
49                     line = strcat(line, '{',datas.info.personal.attrs(i)
50                         .info(1));
51                     for j = 2:length(datas.info.personal.attrs(i).info)
52                         line = strcat(line, ',',datas.info.personal.
53                             attrs(i).info(j));
54                     end
55                     line = strcat(line, '}');
56                 else
57                     error('error');
58                 end
59                 header = [header;line];
60             end
61
62             line = strcat('@attribute',{ ' ' },datas.info.personal.attrs(
63                 end).name,{ ' ' });
64             if strcmp(datas.info.personal.attrs(end).type, 'numeric')
65                 error('In NSLVOrd output should be categoric');
66             elseif strcmp(datas.info.personal.attrs(end).type, '
67                 categoric')
68                 line = strcat(line, '{',datas.info.personal.attrs(end).
69                     info.cat(1));
70                 for j = 2:length(datas.info.personal.attrs(end).info.
71                     cat)
72                     line = strcat(line, ',',datas.info.personal.attrs(
73                         end).info.cat(j));
74                 end
75                 line = strcat(line, '}');
76             else
77                 error('In NSLVOrd output should be categoric');
78             end
79             header = [header;line];
80         else
81             for i = 1:size(datas.patterns,2)
82                 line = strcat('@attribute x',{int2str(i)}, ' numeric');
83                 header = [header;line];
84             end
85             aux = unique(datas.targets);
86             line = strcat('@attribute y',{num2str(aux(1))},{ ' ' });
87             for i = 2:size(aux,1)
88                 line = strcat(line, ',',num2str(aux(i)));
89             end
90         end

```

4.2. Class NSLVOrd

```
80         line = strcat(line, '}');
81         header = [header; line];
82     end
83     header = [header; '@data'];
84 end
85
86 function datas_java = getDatas(datas)
87     [a,b] = size(datas);
88     datas_java = [];
89     for i = 1:a
90         aux = '';
91         for j = 1:b-1
92             dat = datas(i,j);
93             if strcmpi(class(dat), 'double')
94                 dat = num2str(dat);
95             end
96             aux = strcat(aux, dat, ',');
97         end
98         aux = strcat(aux, datas(i,b));
99         datas_java = [datas_java; aux];
100     end
101 end
102
103 function [targets, output] = ConvertTargetsToCategoric(train)
104     if strcmp(train.info.utilities.type, 'weka')
105         trans = train.info.personal.attrs(end).info;
106         targets_m = repmat(train.targets, 1, length(trans.num));
107         num_m = repmat(trans.num, length(train.targets), 1);
108         a = (targets_m == num_m) * [1: length(trans.num)]';
109         targets = trans.cat(a)';
110         output.cat = trans.cat;
111         output.num = trans.num;
112     else
113         aux = unique(train.targets);
114         aux_cat = [];
115         for i = 1:length(aux)
116             aux_cat = [aux_cat {num2str(aux(i))}];
117         end
118         output.cat = aux_cat;
119         output.num = unique(train.targets)';
120         targets = cellstr(num2str(train.targets));
121     end
122 end
123
124 function patterns = ConvertPatternsToChar(patterns)
125     patt_aux = [];
126     for i = 1:size(patterns, 2)
127         aux = patterns(:, i);
128         if strcmpi(class(aux), 'double')
129             patt_aux = [patt_aux cellstr(num2str(aux))];
130         else
131             patt_aux = [patt_aux aux];
132         end
133     end
134 end
```

```

133         end
134         patterns = patt_aux;
135     end
136
137     function targets = ConvertCategoricToTargets(result,trans)
138         a = zeros(size(result,1),size(trans.cat,2));
139         for i = 1:size(a,1)
140             for j = 1:size(a,2)
141                 a(i,j) = double(strcmp(result(i),trans.cat{j}));
142             end
143         end
144         a = a * [1:length(trans.cat)]';
145         targets = trans.num(a)';
146     end
147
148     function res = toChar(param)
149         res = [];
150         for i = 1:size(param,1)
151             a1 = param(i);
152             res1 = [];
153             for j = 1:size(a1,1)
154                 a2 = a1(j);
155                 res2 = [];
156                 for k = 1:size(a2,1)
157                     a3 = char(a2(k));
158                     res2 = [res2,{a3}];
159                 end
160                 res1 = [res1;{res2}];
161             end
162             res = [res;{res1}];
163         end
164     end
165
166     function res = toCell(param)
167         res = [];
168         for i = 1:size(param,1)
169             res = [res;{char(param(i))}];
170         end
171     end
172
173     function res = toJavaString(param)
174         res = javaArray('java.lang.String []',size(param,1));
175         for i = 1:size(param,1)
176             a1 = param{i};
177             res1 = [];
178             for j = 1:size(a1,1)
179                 a2 = a1{j};
180                 res2 = [];
181                 for k = 1:size(a2,2)
182                     a3 = java.lang.String(a2{k});
183                     res2 = [res2,a3];
184                 end
185                 res1 = [res1;res2];

```

4.2. Clase NSLVOrd

```
186         end
187         res(i) = [res1; []];
188     end
189 end
190 end
191
192 methods
193     function obj = NSLVOrd(~, varargin)
194         % Process key-values pairs of parameters
195         obj.parseArgs(varargin);
196         obj.categ = true;
197     end
198
199     function [projectedTrain, predictedTrain] = privfit(obj, train,
200         param)
201         % fit the model and return prediction of train set. It is
202         % called by
203         % super class Algorithm.fit() method.
204
205         % Convertir los datos a objetos Java
206         param_java = obj.initParameters(param);
207
208         header = obj.getHeader(train);
209         [targets, output] = obj.ConvertTargetsToCategoric(train);
210         patterns = obj.ConvertPatternsToChar(train.patterns);
211         datas = [patterns targets];
212         datas = obj.getDatas(datas);
213
214         % NSLVOrd Java
215         algorithmPath = fullfile(fileparts(which('Algorithm.m')),'
216             NSLVOrd');
217         jarfolder = fullfile(algorithmPath, 'NSLVOrdJava.jar');
218         javaaddpath(jarfolder);
219
220         nslvord = javaObject('NSLVOrdJava.NSLVOrdJava');
221         result = javaMethod('Train', nslvord, header, datas, param_java);
222         knowledgebase = javaMethod('get_knowledge_base', nslvord);
223         rulebase = javaMethod('get_rule_base', nslvord);
224         rules = javaMethod('get_rules', nslvord);
225
226         clear nslvord;
227         javarmpath(jarfolder);
228
229         % Process output
230         %f strcmpi(train.info.utilities.type, 'weka')
231         targets = obj.ConvertCategoricToTargets(result, output);
232         projectedTrain = targets;
233         predictedTrain = targets;
234
235         % Save the model
236         try
237             model.name = train.name;
238         catch
```

```

236         end
237         model.output = output;
238         model.knowledgebase = obj.toCell(knowledgebase);
239         model.rulebase = obj.toCell(rulebase);
240         model.rules = obj.toCell(rules);
241         model.type = train.info.utilities.type;
242         model.header = header;
243         model.parameters = param;
244         obj.model = model;
245
246         % Active export and see rules
247         obj.export = param.ExportRules;
248         obj.visual = param.SeeRules;
249     end
250
251     function [projected, predicted] = privpredict(obj, patterns)
252         % predict unseen patterns with 'obj.model' and return
253         % prediction and
254         % projection of patterns (for threshold models)
255         % It is called by super class Algorithm.predict() method.
256
257         % Convert inputs to java objects
258         patterns = obj.ConvertPatternsToChar(patterns);
259         datas = obj.getDatas(patterns);
260
261         % NSLVOrd Java
262         algorithmPath = fullfile(fileparts(which('Algorithm.m')), '
263             NSLVOrd');
264         jarfolder = fullfile(algorithmPath, 'NSLVOrdJava.jar');
265         javaaddpath(jarfolder);
266
267         nslvord = javaObject('NSLVOrdJava.NSLVOrdJava');
268         javaMethod('LoadModel', nslvord, obj.model.knowledgebase, obj.
269             model.rulebase);
270         result = javaMethod('Test', nslvord, obj.model.header, datas);
271
272         clear nslvord;
273         javarmpath(jarfolder);
274
275         % Process output
276         output = obj.model.output;
277         targets = obj.ConvertCategoricToTargets(result, output);
278         projected = targets;
279         predicted = targets;
280     end
281
282     function visual_rules(obj)
283         visual = RulesVisual;
284
285         % RuleBase
286         rb = obj.model.rules;
287         numr = str2num(rb{1});
288         finr = 1;

```


4.2. Clase NSLVOrd

```
286         % Rule
287         for i = 1:numr
288             inir = finr + 1;
289             finr = inir + 2 + str2num(rb{inir + 2});
290             r = rb(inir:finr);
291             rname = r{1};
292             rweight = str2double(r{2});
293             numant = str2num(r{4}) - 1;
294             con = r(end-1:end);
295             ant = getant(obj,r(5:end-2),numant);
296             visual.new_rule(rname,rweight);
297             for j = 1:size(ant,1)
298                 visual.add_antecedent(ant(j).name,ant(j).values);
299             end
300             visual.new_consequent(con{1},con{2});
301         end
302
303         % Visual
304         visual.visual_rules(obj.model.name);
305     end
306
307     function export_rules(obj,dir)
308         export = RulesExport;
309
310         % KnowledgeBase
311         kb = obj.model.knowledgebase;
312         numfv = str2num(kb{5});
313         finfv = 5;
314         % FuzzyVariable
315         for i = 1:numfv
316             inifv = finfv + 1;
317             numft = str2num(kb{inifv + 8});
318             finfv = inifv + 8 + 7 * numft;
319             fv = kb(inifv:finfv);
320             export.new_variable(fv{1},fv{5},fv{6});
321             finft = 9;
322             % FuzzyTerm
323             for j = 1:numft
324                 inift = finft + 1;
325                 finft = inift + 6;
326                 ft = fv(inift:finft);
327                 export.add_terms(ft{1},ft{2},ft{3},ft{4},ft{5});
328             end
329         end
330
331         % RuleBase
332         rb = obj.model.rules;
333         numr = str2num(rb{1});
334         finr = 1;
335         % Rule
336         for i = 1:numr
337             inir = finr + 1;
338             finr = inir + 2 + str2num(rb{inir + 2});
```

```

339         r = rb(inir:finr);
340         rname = r{1};
341         rweight = str2double(r{2});
342         numant = str2num(r{4}) - 1;
343         con = r(end-1:end);
344         ant = subrules(obj,r(5:end-2),numant);
345         for j = 1:size(ant,1)
346             acname = rname;
347             if size(ant,1) > 1
348                 acname = [rname '_' num2str(j)];
349             end
350             export.new_rule(acname,rweight);
351             antr = ant{j};
352             for k = 1:2:size(antr,2)
353                 export.add_antecedent(antr{k},antr{k+1});
354             end
355             export.new_consequent(con{1},con{2});
356         end
357     end
358
359     % Export
360     export.export_rules(dir,obj.model.name);
361 end
362
363 function ant = subrules(obj,r,numant)
364     [ant_aux,comb] = obj.getant(r,numant);
365     ant = [];
366     if ~isempty(ant_aux)
367         ant = cell(comb,1);
368         for i = 1:comb
369             start = comb;
370             index = i;
371             for j = 1:size(ant_aux,1)
372                 aux = ant_aux(j);
373                 name = aux.name;
374                 aux = aux.values;
375                 start = start / size(aux,1);
376                 value = aux(floor((index-1)/start) + 1);
377                 index = index - start * floor((index-1)/start);
378                 ant{i} = [ant{i},{name},{value}];
379             end
380         end
381     end
382 end
383
384 function [ant,comb] = getant(obj,r,numant)
385     ant = [];
386     fin = 0;
387     comb = 1;
388     for i = 1:numant
389         ini = fin + 1;
390         aux.name = r{ini};
391         numval = str2num(r{ini+1});

```

4.2. Clase NSLVOrd

```
392         fin = ini + 1 + numval * 7;
393         vr = r(ini:fin);
394         values = [];
395         finv = 2;
396         comb = comb * numval;
397         for j = 1:numval
398             iniv = finv + 1;
399             finv = iniv + 6;
400             tr = vr(iniv:finv);
401             aux2 = {tr{1},str2double(tr{2}),str2double(tr{3}),
                     str2double(tr{4}),str2double(tr{5}),str2num(tr{6}),
                     str2num(tr{7})};
402             values = [values;aux2];
403         end
404         aux.values = values;
405         ant = [ant;aux];
406     end
407 end
408 end
409 end
```

CÓDIGO 4.2: Archivo *NSLVOrd.m* correspondiente al módulo Algorithms



5 Módulo NSLVOOrd

5.1. Clase NSLVOOrdJava

La clase *NSLVOOrdJava* se encarga de la ejecución del algoritmo NSLVOOrd. En el Código 5.1 se muestra la implementación de esta clase.

```
1 package NSLVOOrdJava;
2
3 // para la integracion en keel
4 import keel.Dataset.*;
5
6 import java.util.*;
7
8 /**
9  * @file NSLVOOrd.java
10  * @brief main file of proyect
11  * @author Juan Carlos Gamez (original de Raul Perez)
12  * @version 1
13  * @date diciembre 2015
14  * @note Implement of NSLV algorithm for ordinal classification
15  */
16 public class NSLVOOrdJava {
17
18     // habra 3 valores: indice 0->izda, indice 1->centro, indice 2->dcha
19     // static int numDesplazamientos=3;
20     static int numDesplazamientos=1;
21     static double[] time;
22     static int[] iter;
23
24     static FuzzyProblemClass[] fuzzyProblem;
25     static ExampleSetProcess[] E_par, E_par_test;
```

```

26     static RuleSetClass [] R;
27
28     static InstanceSet iSet;
29     static InstanceSet tSet;
30
31     static Random [] randomNum;
32
33     static String fileResultDebug;
34     static double [][] costMatrix;
35     static int seed;
36     static int numLabelsInputs;
37     static int numLabelsOutput;
38     static int shift;
39
40     static int homogeneousLabel=0; // se elimina el parametro de cuda para
        introducir la creacion de etiquetas homogeneas
41 // parametros de ponderacion de la caracteristica ordinal o nominal de
        la funcion fitness
42 // alpha * CCR y (1-alpha) * MAE
43     static double alpha=0.5;
44
45     static String [] poblacionParam;
46
47     public static String [] Train(String [] _header,String [] _datas,String []
        args){
48         Attributes.clearAll();
49         initParameters(args);
50
51         // Aqui se inicializa Random
52         randomNum= new Random[numDesplazamientos];
53         iter= new int[numDesplazamientos];
54         time= new double[numDesplazamientos];
55         for (int i=0; i < numDesplazamientos; i++){
56             randomNum[i]= new Random(seed);
57         }
58
59         // obtener las instancias (ejemplos) de training y test y pasarlas
            a "los objetos de mis clases"
60         if(!ReadSet(_header,_datas,true)) return null;
61
62         if(!executeNSLVOrd(homogeneousLabel)) return null;
63
64         return Targets(E-par,1);
65
66     }
67
68     public static FuzzyProblemClass [] GetFuzzyProblem(){
69         return fuzzyProblem;
70     }
71
72     public static String [] get_knowledge_base(){
73         RuleSystem _exp = new RuleSystem(fuzzyProblem[0], R[0]);
74         String [] kb = _exp.Export_KnowledgeBase();

```

5.1. Clase NSLVOrdJava

```
75         return kb;
76     }
77
78     public static String[] get_rules(){
79         RuleSystem _exp = new RuleSystem(fuzzyProblem[0], R[0]);
80         String[] rules = _exp.Export_Rules();
81         return rules;
82     }
83
84     public static String[] get_rule_base(){
85         RuleSystem _exp = new RuleSystem(fuzzyProblem[0], R[0]);
86         String[] rb = _exp.Export_RuleBase();
87         return rb;
88     }
89
90     public static void LoadModel(String[] _fuzzyProblem, String[] _R){
91         fuzzyProblem = new FuzzyProblemClass[numDesplazamientos];
92         R= new RuleSetClass[numDesplazamientos];
93         E_par_test= new ExampleSetProcess[numDesplazamientos];
94
95         Load_KnowledgeBase(_fuzzyProblem);
96         Load_RuleBase(_R);
97     }
98
99     public static void Load_KnowledgeBase(String[] sfp){
100         // FUZZY PROBLEM
101         FuzzyProblemClass fp = new FuzzyProblemClass();
102         fp.setConsequentIndexOriginal(Integer.parseInt(sfp[0]));
103         fp.setShift(Integer.parseInt(sfp[1]));
104         fp.setDirection(Integer.parseInt(sfp[2]));
105         fp.setHomogeneousLabel(Integer.parseInt(sfp[3]));
106         fp.setFuzzyLinguisticVariableNum(Integer.parseInt(sfp[4]));
107
108         // FUZZY VARIABLE
109         FuzzyLinguisticVariableClass[] fv = new
            FuzzyLinguisticVariableClass[fp.getFuzzyLinguisticVariableNum()
            ];
110         int pos = 5;
111         for(int i = 0; i < fp.getFuzzyLinguisticVariableNum(); i++){
112             fv[i] = new FuzzyLinguisticVariableClass();
113             fv[i].setName(sfp[pos]); pos++;
114             fv[i].setUnit(Integer.parseInt(sfp[pos])); pos++;
115             fv[i].setNumTermAutomatic(Double.parseDouble(sfp[pos])); pos++;
116             fv[i].setVariableType(Integer.parseInt(sfp[pos])); pos++;
117             fv[i].setInfRange(Double.parseDouble(sfp[pos])); pos++;
118             fv[i].setSupRange(Double.parseDouble(sfp[pos])); pos++;
119             fv[i].setInfRangeIsInf(Integer.parseInt(sfp[pos])); pos++;
120             fv[i].setSupRangeIsInf(Integer.parseInt(sfp[pos])); pos++;
121             fv[i].setFuzzyLinguisticTermNum(Integer.parseInt(sfp[pos]));
                pos++;
122
123             // FUZZY TERM
124             FuzzyLinguisticTermClass[] ft = new FuzzyLinguisticTermClass[fv
```

```

125         [i].getFuzzyLinguisticTermNum();
126     for(int j = 0; j < fv[i].getFuzzyLinguisticTermNum(); j++){
127         ft[j] = new FuzzyLinguisticTermClass();
128         ft[j].setName(sfp[pos]); pos++;
129         ft[j].setA(Double.parseDouble(sfp[pos])); pos++;
130         ft[j].setB(Double.parseDouble(sfp[pos])); pos++;
131         ft[j].setC(Double.parseDouble(sfp[pos])); pos++;
132         ft[j].setD(Double.parseDouble(sfp[pos])); pos++;
133         ft[j].setAbInf(Integer.parseInt(sfp[pos])); pos++;
134         ft[j].setCdInf(Integer.parseInt(sfp[pos])); pos++;
135     }
136     fv[i].setFuzzyLinguisticTermList(ft);
137 }
138
139 fp.setFuzzyLinguisticVariableList(fv);
140
141 fuzzyProblem[0] = fp;
142 }
143
144 public static void Load_RuleBase(String[] srs){
145     // RULE SET
146     RuleSetClass rs = new RuleSetClass();
147     rs.setNumRules(Integer.parseInt(srs[0]));
148     rs.CCR = Double.parseDouble(srs[1]);
149     rs.SM = Double.parseDouble(srs[2]);
150     rs.TPR = Double.parseDouble(srs[3]);
151     rs.TNR = Double.parseDouble(srs[4]);
152     rs.FPR = Double.parseDouble(srs[5]);
153     rs.Kappa = Double.parseDouble(srs[6]);
154     rs.AUC = Double.parseDouble(srs[7]);
155     rs.MSE = Double.parseDouble(srs[8]);
156     rs.RMSE = Double.parseDouble(srs[9]);
157     rs.RMAE = Double.parseDouble(srs[10]);
158     rs.OMAE = Double.parseDouble(srs[11]);
159     rs.OMAENormalizado = Double.parseDouble(srs[12]);
160     rs.MMAE = Double.parseDouble(srs[13]);
161     rs.mMAE = Double.parseDouble(srs[14]);
162     rs.AMAE = Double.parseDouble(srs[15]);
163     rs.Spearman = Double.parseDouble(srs[16]);
164     rs.Kendall = Double.parseDouble(srs[17]);
165     rs.OC = Double.parseDouble(srs[18]);
166     rs.beta = Double.parseDouble(srs[19]);
167     rs.metric = Double.parseDouble(srs[20]);
168     rs.metricMedia = Double.parseDouble(srs[21]);
169     rs.Precision = Double.parseDouble(srs[22]);
170     rs.alphaMetric = Double.parseDouble(srs[23]);
171     rs.confusion = new double[Integer.parseInt(srs[24])][];
172     int pos = 25;
173     for(int i = 0; i < rs.confusion.length; i++){
174         rs.confusion[i] = new double[Integer.parseInt(srs[pos])]; pos
            ++;
175         for(int j = 0; j < rs.confusion[i].length; j++){

```


5.1. Class NSLVOrdJava

```
176         rs.confusion[i][j] = Double.parseDouble(srs[pos]); pos++;
177     }
178 }
179
180 // RULE
181 GenetCodeClass[] rul = new GenetCodeClass[rs.getNumRules()];
182 for(int i = 0; i < rul.length; i++){
183     // Binary elements
184     int binaryBlocs;
185     int[] sizeBinaryBlocs;
186     int[][] binaryMatrix;
187     binaryBlocs = Integer.parseInt(srs[pos]); pos++;
188     sizeBinaryBlocs = new int[binaryBlocs];
189     binaryMatrix = new int[binaryBlocs][];
190     for(int j = 0; j < binaryBlocs; j++){
191         sizeBinaryBlocs[j] = Integer.parseInt(srs[pos]); pos++;
192         binaryMatrix[j] = new int[sizeBinaryBlocs[j]];
193         for(int k = 0; k < sizeBinaryBlocs[j]; k++){
194             binaryMatrix[j][k] = Integer.parseInt(srs[pos]); pos++;
195         }
196     }
197
198     // Integer elements
199     int integerBlocs;
200     int[] sizeIntegerBlocs;
201     int[][] integerMatrix;
202     int[] integerRange;
203     integerBlocs = Integer.parseInt(srs[pos]); pos++;
204     sizeIntegerBlocs = new int[integerBlocs];
205     integerMatrix = new int[integerBlocs][];
206     for(int j = 0; j < integerBlocs; j++){
207         sizeIntegerBlocs[j] = Integer.parseInt(srs[pos]); pos++;
208         integerMatrix[j] = new int[sizeIntegerBlocs[j]];
209         for(int k = 0; k < sizeIntegerBlocs[j]; k++){
210             integerMatrix[j][k] = Integer.parseInt(srs[pos]); pos
211             ++;
212         }
213     }
214     integerRange = new int[Integer.parseInt(srs[pos])]; pos++;
215     for(int j = 0; j < integerRange.length; j++){
216         integerRange[j] = Integer.parseInt(srs[pos]); pos++;
217     }
218
219     // Real elements
220     int realBlocs;
221     int[] sizeRealBlocs;
222     double[][] realMatrix;
223     double[] realInfRange;
224     double[] realSupRange;
225     realBlocs = Integer.parseInt(srs[pos]); pos++;
226     sizeRealBlocs = new int[realBlocs];
227     realMatrix = new double[realBlocs][];
228     for(int j = 0; j < realBlocs; j++){
```

```

228         sizeRealBlocs[j] = Integer.parseInt(srs[pos]); pos++;
229         realMatrix[j] = new double[sizeRealBlocs[j]];
230         for(int k = 0; k < sizeRealBlocs[j]; k++){
231             realMatrix[j][k] = Double.parseDouble(srs[pos]); pos++;
232         }
233     }
234     realInfRange = new double[Integer.parseInt(srs[pos])]; pos++;
235     for(int j = 0; j < realInfRange.length; j++){
236         realInfRange[j] = Double.parseDouble(srs[pos]); pos++;
237     }
238     realSupRange = new double[Integer.parseInt(srs[pos])]; pos++;
239     for(int j = 0; j < realSupRange.length; j++){
240         realSupRange[j] = Double.parseDouble(srs[pos]); pos++;
241     }
242
243
244     rul[i] = new GenetCodeClass(binaryBlocs, integerBlocs, realBlocs,
245                               sizeBinaryBlocs, sizeIntegerBlocs, sizeRealBlocs,
246                               integerRange, realInfRange, realSupRange);
247     rul[i].setBinaryMatrix(binaryMatrix);
248     rul[i].setIntegerMatrix(integerMatrix);
249     rul[i].setRealMatrix(realMatrix);
250 }
251 rs.setRules(rul);
252
253 R[0] = rs;
254 }
255
256 public static String[] Test(String[] _header, String[] _datas){
257     Attributes.clearAll();
258     iter= new int[numDesplazamientos];
259     time= new double[numDesplazamientos];
260
261     // obtener las instancias (ejemplos) de training y test y pasarlas
262     // a "los objetos de mis clases"
263     if(!ReadSet(_header, _datas, false)) return null;
264
265     if(!executeNSLVOOrdPredict()) return null;
266
267     return Targets(E_par_test, 1);
268 }
269
270 public static void initParameters(String[] param){
271     fuzzyProblem = new FuzzyProblemClass[numDesplazamientos];
272     E_par= new ExampleSetProcess[numDesplazamientos];
273     E_par_test= new ExampleSetProcess[numDesplazamientos];
274     R= new RuleSetClass[numDesplazamientos];
275
276     seed= Integer.parseInt(param[0]);
277     homogeneousLabel = 0;
278     numLabelsInputs= Integer.parseInt(param[1]);
279     numLabelsOutput= Integer.parseInt(param[2]);
280     shift= Integer.parseInt(param[3]);

```

5.1. Clase NSLVOrdJava

```
280         alpha= Double.parseDouble(param[4]); // indica si realiza
           clasificacion(1) o regresion(0)
281         if ((alpha + (1-alpha)) != 1){
282             alpha= 0.5;
283         }
284
285         // el resto de parametros que corresponden a las probabilidades de
286         // inicializacion , cruce y mutacion de las subpoblaciones y de cada
287         // elemento de la subpoblacion
288         poblationParam = new String[14];
289         poblationParam[0] = param[5];
290         poblationParam[1] = param[6];
291         poblationParam[2] = param[7];
292         poblationParam[3] = param[8];
293         poblationParam[4] = param[9];
294         poblationParam[5] = param[10];
295         poblationParam[6] = param[11];
296         poblationParam[7] = param[12];
297         poblationParam[8] = param[13];
298         poblationParam[9] = param[14];
299         poblationParam[10] = param[15];
300         poblationParam[11] = param[16];
301         poblationParam[12] = param[17];
302         poblationParam[13] = param[18];
303     }
304
305     public static boolean ReadSet(String[] _header , String[] _datas ,
           boolean _train){
306         // obtener las instancias (ejemplos) de training y test y pasarlas
           a "los objetos de mis clases"
307         InstanceSet _Set= new InstanceSet();
308         _Set.readSetTFG(_header , _datas , _train);
309         _Set.setAttributesAsNonStatic();
310
311         //si no hay ejemplos sale directamente
312         if (_Set.getNumInstances() == 0){
313             return false;
314         }
315
316         if(_train){
317             iSet = _Set;
318         }else{
319             tSet = _Set;
320         }
321
322         return true;
323     }
324
325     public static boolean executeNSLVOrd(int homogeneousLabel){
326         // parte de ejecucion en serie
327         randomNum[0]= new Random(seed);
328         return executeLearning(0, 0, 0, homogeneousLabel); // para probar
           por ahora nada mas que con una ejecucion
```

```

329     // FIN – parte de ejecucion en serie
330 }
331
332 public static boolean executeNSLVOrdPredict() {
333     // parte de ejecucion en serie
334     return executePredict(0, 0, 0); // para probar por ahora nada mas
335         que con una ejecucion
336     // FIN – parte de ejecucion en serie
337 }
338
339 public static boolean executeLearning(int shift, int direction, int
index, int homogeneousLabel){
340     iter[index]=0;
341     if (numLabelsInputs == -1 || numLabelsOutput == -1){
342         // constructor para la creacion de etiquetas no homogeneas (en
343         funcion del numero de individuos por etiqueta)
344         numLabelsInputs = 11;
345         numLabelsOutput = 11;
346         fuzzyProblem[index]= new FuzzyProblemClass(iSet,
347             numLabelsInputs, numLabelsOutput, shift, direction,
348             homogeneousLabel);
349     }
350     else{
351         // constructor original para la creacion de etiquetas
352         homogeneas
353         fuzzyProblem[index]= new FuzzyProblemClass(iSet,
354             numLabelsInputs, numLabelsOutput, shift, direction,
355             homogeneousLabel);
356     }
357     // pasar los ejemplos a "mis objetos"
358     E_par[index]= new ExampleSetProcess(fuzzyProblem[index], iSet);
359     String result= E_par[index].calcAdaptExVarLabTFG();
360     if (result.compareTo("") != 0){
361         return false;
362     }
363
364     // calcular las medidas de informacion para agilizar los calculos
365     E_par[index].calcInformationMeasures();
366
367     // crear el objeto para el algoritmo genetico
368     R[index]= new RuleSetClass(alpha);
369
370     //creacion del objeto genetico e inicializarlo
371     GeneticAlgorithmClass GA= new GeneticAlgorithmClass(
372         poblacionParam, E_par[index]);
373     // inicializar la poblacion
374     GA.initPopulation(randomNum[index], E_par[index], costMatrix);
375
376     // BEGIN – aqui comenzaria el bloque de ejecuciones del algoritmo
377     genetico
378     Util.initStatisticalData(GA.getP(), fuzzyProblem[index]);
379

```

5.1. Clase NSLVOrdJava

```
372         // calcular la nueva regla
373         int ejemplosCubiertos=0, eliminadoReglas=0, newRule=1;
374         Util.numIterGenetic++;
375
376         //// AQUI PARA ANNADIR O NO LA REGLA POR DEFECTO AL COMIENZO ...
377         int addDefaultRule=0;
378         Util.classDefaultRule= GA.setDefaultRule(addDefaultRule,E-par[
            index],R[index]);
379         if (addDefaultRule == 1){ // Si se ha incluido la regla por
            defecto al principio.
380             ejemplosCubiertos= E-par[index].calcCoveredTFG(R[index],GA.getP
                (), fuzzyProblem[index]);
381         }
382
383         eliminadoReglas= 1;
384         while (eliminadoReglas == 1){
385             while (newRule == 1 && ejemplosCubiertos < E-par[index].
                numExamples){
386                 iter[index]++;
387                 newRule= GA.findNewRuleTFG(randomNum[index],0,E-par[index],R[
                    index]); // en la version de homogeneousLabel se ha
                    eliminado la opcion de cuda
388                 Util.numIterGenetic++;
389
390                 ejemplosCubiertos= E-par[index].calcCoveredTFG(R[index],GA.
                    getP(),fuzzyProblem[index]);
391
392
393                 //while (newRule == 1){
394                 eliminadoReglas= R[index].removeRulesForImproveMetricTFG(
                    E-par[index],GA, fuzzyProblem[index]); // probar a quitar
                    reglas y ver si mejora la precision
395                 if (eliminadoReglas == 1){
396                     newRule=1;
397                     ejemplosCubiertos= E-par[index].calcCoveredTFG(R[index], GA
                        .getP(), fuzzyProblem[index]);
398                 }
399                 // while (eliminadoReglas != 1){
400
401
402                 if (addDefaultRule == 0){ // No se ha incluido la regla por
                    defecto al principio. Se debe incluir al final
403                     R[index].addRule(Util.DefaultRule, Util.DefaultRule.
                        getRealMatrix(Util.classDefaultRule, 4), E-par[index]);
404                 }
405
406                 return true;
407             }
408
409         public static boolean executePredict(int shift, int direction, int
            index){
410             iter[index]=0;
411
```

```

412         // pasar los ejemplos a "mis objetos"
413         E_par_test[index]= new ExampleSetProcess(fuzzyProblem[index],
            tSet);
414         String result= E_par_test[index].calcAdaptExVarLabTFG();
415         if (result.compareTo("") != 0){
416             return false;
417         }
418
419         return true;
420     }
421
422     public static String [] Targets(ExampleSetProcess [] _par,int numShifts){
423         int indexRule=0;
424         int claseInference;
425         int varCons= _par[0].getProblemDefinition().consequentIndex();
426         double valueReglaCombinado;
427         int [] indRegla;
428         String [] _Resultado;
429
430         indRegla= new int[numShifts];
431         _Resultado = new String[_par[0].getNumExamples()];
432
433         for (int i=0; i < _par[0].getNumExamples(); i++){
434             for (int d=0; d < numShifts; d++){
435                 indexRule= R[d].inference(_par[d],i);
436                 indRegla[d]= indexRule;
437             }
438             claseInference= R[0].getRules(indexRule).getIntegerMatrix(0,0);
439             valueReglaCombinado= Util.getCentralValue(claseInference,
                varCons, _par[0]);
440
441             _Resultado[i] = _par[0].getProblemDefinition().
                getFuzzyLinguisticVariableList(varCons).
                getFuzzyLinguisticTermList((int)valueReglaCombinado).
                getName();
442         }
443
444         return _Resultado;
445     }
446 }

```

CÓDIGO 5.1: Archivo *NSLVOrdJava.java* correspondiente al módulo NSLVOrd

5.2. Clase RuleSystem

La clase *RuleSystem* se encarga de configurar la base de conocimiento y de reglas para ser exportadas a ORCA. En el Código 5.2 se muestra la implementación de esta clase.

```
1 package NSLVOrdJava;
2
3 import java.util.ArrayList;
4 import jfml.FuzzyInferenceSystem;
5
6 public class RuleSystem {
7     static private FuzzyInferenceSystem _f;
8     static private FuzzyProblemClass _fuzzyProblem;
9     static private RuleSetClass _R;
10
11     public RuleSystem(FuzzyProblemClass fuzzyProblem, RuleSetClass R){
12         _f = new FuzzyInferenceSystem();
13         _fuzzyProblem = fuzzyProblem;
14         _R = R;
15     }
16
17     public String[] Export_KnowledgeBase(){
18         ArrayList export_aux = new ArrayList();
19
20         // FUZZY PROBLEM
21         export_aux.add(String.valueOf(_fuzzyProblem.
22             getConsequentIndexOriginal()));
23         export_aux.add(String.valueOf(_fuzzyProblem.getShift()));
24         export_aux.add(String.valueOf(_fuzzyProblem.getDirection()));
25         export_aux.add(String.valueOf(_fuzzyProblem.getHomogeneousLabel()));
26
27         export_aux.add(String.valueOf(_fuzzyProblem.
28             getFuzzyLinguisticVariableNum()));
29
30         // FUZZY VARIABLE
31         for (FuzzyLinguisticVariableClass auxLinguisticVar : _fuzzyProblem.
32             getFuzzyLinguisticVariableList()){
33             export_aux.add(auxLinguisticVar.getName());
34             export_aux.add(String.valueOf(auxLinguisticVar.getUnit()));
35             export_aux.add(String.valueOf(auxLinguisticVar.
36                 getNumTermAutomatic()));
37             export_aux.add(String.valueOf(auxLinguisticVar.getVariableType()));
38             export_aux.add(String.valueOf(auxLinguisticVar.getInfRange()));
39             export_aux.add(String.valueOf(auxLinguisticVar.getSupRange()));
40             export_aux.add(String.valueOf(auxLinguisticVar.getInfRangeIsInf()));
41             export_aux.add(String.valueOf(auxLinguisticVar.getSupRangeIsInf()));
```

```

37         ));
38     export_aux.add(String.valueOf(auxLinguisticVar.
39         getFuzzyLinguisticTermNum()));
40     // FUZZY TERM
41     for (FuzzyLinguisticTermClass auxLinguisticTerm :
42         auxLinguisticVar.getFuzzyLinguisticTermList()){
43         export_aux.add(auxLinguisticTerm.getName());
44         export_aux.add(String.valueOf(auxLinguisticTerm.getA()));
45         export_aux.add(String.valueOf(auxLinguisticTerm.getB()));
46         export_aux.add(String.valueOf(auxLinguisticTerm.getC()));
47         export_aux.add(String.valueOf(auxLinguisticTerm.getD()));
48         export_aux.add(String.valueOf(auxLinguisticTerm.getAbInf()));
49         export_aux.add(String.valueOf(auxLinguisticTerm.getCdInf()));
50     }
51     // To String vector
52     String[] export = new String[export_aux.size()];
53     for(int i = 0; i < export_aux.size(); i++){
54         export[i] = (String) export_aux.get(i);
55     }
56     return export;
57 }
58
59 public String[] Export_RuleBase(){
60     ArrayList export_aux = new ArrayList();
61
62     // RULE SET
63     export_aux.add(String.valueOf(_R.getNumRules()));
64     export_aux.add(String.valueOf(_R.CCR));
65     export_aux.add(String.valueOf(_R.SM));
66     export_aux.add(String.valueOf(_R.TPR));
67     export_aux.add(String.valueOf(_R.TNR));
68     export_aux.add(String.valueOf(_R.FPR));
69     export_aux.add(String.valueOf(_R.Kappa));
70     export_aux.add(String.valueOf(_R.AUC));
71     export_aux.add(String.valueOf(_R.MSE));
72     export_aux.add(String.valueOf(_R.RMSE));
73     export_aux.add(String.valueOf(_R.RMAE));
74     export_aux.add(String.valueOf(_R.OMAE));
75     export_aux.add(String.valueOf(_R.OMAENormalizado));
76     export_aux.add(String.valueOf(_R.MMAE));
77     export_aux.add(String.valueOf(_R.mMAE));
78     export_aux.add(String.valueOf(_R.AMAE));
79     export_aux.add(String.valueOf(_R.Spearman));
80     export_aux.add(String.valueOf(_R.Kendall));
81     export_aux.add(String.valueOf(_R.OC));
82     export_aux.add(String.valueOf(_R.beta));
83     export_aux.add(String.valueOf(_R.metric));
84

```


5.2. Class RuleSystem

```
85     export_aux.add(String.valueOf(_R.metricMedia));
86     export_aux.add(String.valueOf(_R.Precision));
87     export_aux.add(String.valueOf(_R.alphaMetric));
88     export_aux.add(String.valueOf(_R.confusion.length));
89     for (double[] confusion : _R.confusion) {
90         export_aux.add(String.valueOf(confusion.length));
91         for (double val : confusion) {
92             export_aux.add(String.valueOf(val));
93         }
94     }
95
96     // RULE
97     for (GenetCodeClass auxGenetCode : _R.getRules()) {
98         // Binary elements
99         export_aux.add(String.valueOf(auxGenetCode.getBinaryBlocs()));
100        for (int i = 0; i < auxGenetCode.getBinaryBlocs(); i++) {
101            export_aux.add(String.valueOf(auxGenetCode.
102                getSizeBinaryBlocs(i)));
103            for (int j = 0; j < auxGenetCode.getSizeBinaryBlocs(i); j++) {
104                export_aux.add(String.valueOf(auxGenetCode.
105                    getBinaryMatrix(i, j)));
106            }
107        }
108        // Integer elements
109        export_aux.add(String.valueOf(auxGenetCode.getIntegerBlocs()));
110        for (int i = 0; i < auxGenetCode.getIntegerBlocs(); i++) {
111            export_aux.add(String.valueOf(auxGenetCode.
112                getSizeIntegerBlocs(i)));
113            for (int j = 0; j < auxGenetCode.getSizeIntegerBlocs(i); j++) {
114                export_aux.add(String.valueOf(auxGenetCode.
115                    getIntegerMatrix(i, j)));
116            }
117        }
118        export_aux.add(String.valueOf(auxGenetCode.getIntegerRange().
119            length));
120        for (int i : auxGenetCode.getIntegerRange()) {
121            export_aux.add(String.valueOf(i));
122        }
123        // Real elements
124        export_aux.add(String.valueOf(auxGenetCode.getRealBlocs()));
125        for (int i = 0; i < auxGenetCode.getRealBlocs(); i++) {
126            export_aux.add(String.valueOf(auxGenetCode.getSizeRealBlocs(i)));
127            for (int j = 0; j < auxGenetCode.getSizeRealBlocs(i); j++) {
128                export_aux.add(String.valueOf(auxGenetCode.
129                    getRealMatrix(i, j)));
130            }
131        }
132        export_aux.add(String.valueOf(auxGenetCode.getRealInfRange().
```

```

length));
129     for(double i : auxGenetCode.getRealInfRange()){
130         export_aux.add(String.valueOf(i));
131     }
132     export_aux.add(String.valueOf(auxGenetCode.getRealSupRange().
length));
133     for(double i : auxGenetCode.getRealSupRange()){
134         export_aux.add(String.valueOf(i));
135     }
136 }
137
138 // To String vector
139 String[] export = new String[export_aux.size()];
140 for(int i = 0; i < export_aux.size(); i++){
141     export[i] = (String) export_aux.get(i);
142 }
143
144 return export;
145 }
146
147 public String[] Export_Rules(){
148     ArrayList export_aux = new ArrayList();
149
150     // RULES
151     int numRules= _R.getNumRules();
152     export_aux.add(String.valueOf(numRules));
153     for (int i = 0; i < numRules; i++){
154         int classR= _R.getRules(i).getIntegerMatrix(0,0);
155
156         // DATA RULE
157         export_aux.add("R" + i);
158         export_aux.add(String.valueOf(_R.getRules(i).getRealMatrix(2+
classR,4)));
159
160         // ANTECEDENT
161         ArrayList ant = Export_Antecedents(i);
162         export_aux.add(String.valueOf(ant.size() + 2));
163         for (Object ant1 : ant) {
164             export_aux.add((String) ant1);
165         }
166
167         // CONSEQUENT
168         int conseqIndex = _fuzzyProblem.consequentIndex();
169         export_aux.add(_fuzzyProblem.getFuzzyLinguisticVariableList(
conseqIndex).getName());
170         export_aux.add(_fuzzyProblem.getFuzzyLinguisticVariableList(
conseqIndex).getFuzzyLinguisticTermList(_R.getRules(i).
getIntegerMatrix(0,0)).getName());
171     }
172
173     // To String vector
174     String[] export = new String[export_aux.size()];
175     for(int i = 0; i < export_aux.size(); i++){

```

5.2. Classe RuleSystem

```
176         export[i] = (String) export_aux.get(i);
177     }
178
179     return export;
180 }
181
182 public ArrayList Export_Antecedents(int rule){
183     // Get antecedents
184     ArrayList validTerm = new ArrayList();
185     int numVariables = _fuzzyProblem.getFuzzyLinguisticVariableNum();
186     int start = 0;
187     int tamBloc = _R.getRules(rule).getSizeRealBlocs(0);
188     int consequIndex = _fuzzyProblem.consequentIndex();
189     int numT = 0;
190     double infMeasureClass = _R.getRules(rule).getRealMatrix(0, tamBloc
191         -1);
192     for (int j=0; j < numVariables-1; j++){
193         ArrayList aux = new ArrayList();
194         int numLabels = _fuzzyProblem.getFuzzyLinguisticVariableList(j)
195             .getFuzzyLinguisticTermNum();
196         double actInfMeasure = _R.getRules(rule).getRealMatrix(0, j);
197         if ((_R.getRules(rule).binaryMatrix0AllToOne(start, numLabels)
198             != 1) &&
199             (j != consequIndex && actInfMeasure >= infMeasureClass)){
200             for (int k=0; k < numLabels; k++){
201                 int valueLabel = _R.getRules(rule).getBinaryMatrix(0,
202                     start+k);
203                 if (valueLabel == 1){
204                     aux.add(_fuzzyProblem.
205                         getFuzzyLinguisticVariableList(j).
206                         getFuzzyLinguisticTermList(k).getName());
207                     aux.add(String.valueOf(_fuzzyProblem.
208                         getFuzzyLinguisticVariableList(j).
209                         getFuzzyLinguisticTermList(k).getA()));
210                     aux.add(String.valueOf(_fuzzyProblem.
211                         getFuzzyLinguisticVariableList(j).
212                         getFuzzyLinguisticTermList(k).getB()));
213                     aux.add(String.valueOf(_fuzzyProblem.
214                         getFuzzyLinguisticVariableList(j).
215                         getFuzzyLinguisticTermList(k).getC()));
216                     aux.add(String.valueOf(_fuzzyProblem.
217                         getFuzzyLinguisticVariableList(j).
218                         getFuzzyLinguisticTermList(k).getD()));
219                     aux.add(String.valueOf(_fuzzyProblem.
220                         getFuzzyLinguisticVariableList(j).
221                         getFuzzyLinguisticTermList(k).getAbInf()));
222                     aux.add(String.valueOf(_fuzzyProblem.
223                         getFuzzyLinguisticVariableList(j).
224                         getFuzzyLinguisticTermList(k).getCdInf()));
225                 }
226             }
227         }
228         if (!aux.isEmpty()){
229             aux.add(_fuzzyProblem.getFuzzyLinguisticVariableList(j)
```

```

211         .getName());
212         validTerm.add(aux);
213         numT++;
214     }
215     start= start+numLabels;
216 }
217
218 // Save antecedentes
219 ArrayList ant = new ArrayList();
220 ant.add(String.valueOf(numT+1));
221 for (Object next : validTerm) {
222     ArrayList aux = (ArrayList) next;
223     ant.add((String) aux.get(aux.size()-1));
224     int numTerm = (aux.size() - 1) / 7;
225     ant.add(String.valueOf(numTerm));
226     for(int i = 0; i < numTerm; i++){
227         int num = i * 7;
228         ant.add((String) aux.get(num));
229         ant.add((String) aux.get(num + 1));
230         ant.add((String) aux.get(num + 2));
231         ant.add((String) aux.get(num + 3));
232         ant.add((String) aux.get(num + 4));
233         ant.add((String) aux.get(num + 5));
234         ant.add((String) aux.get(num + 6));
235     }
236 }
237
238 return ant;
239 }
240 }

```

CÓDIGO 5.2: Archivo *RuleSystem.java* correspondiente al módulo NSLVOrd

6 Módulo Rule View

6.1. Clase RulesVisual

La clase *RulesVisual* configura las reglas del algoritmo para su visualización. En el Código 6.1 se muestra la implementación de esta clase.

```
1 classdef RulesVisual < handle
2     properties
3         rules = [];
4     end
5
6     methods
7         function visual_rules(obj,name)
8             if isempty(obj.rules)
9                 error('Rules System is empty');
10            end
11
12            addpath(fullfile(fileparts(which('RulesVisual.m'))),'VisualRules
13                ');
14
15            Visual(name,obj.rules);
16
17            rmpath(fullfile(fileparts(which('RulesVisual.m'))),'VisualRules
18                ');
19        end
20
21        function num = detect_number(~,num)
22            if iscell(num)
23                if ~isempty(find(size(num) ~= 1,1))
24                    num = NaN;
25                else
```

```

24         num = num{1};
25     end
26 end
27
28     switch class(num)
29     case 'double'
30         if ~isempty(find(size(num) ~= 1,1))
31             num = NaN;
32         end
33     case 'char'
34         num = str2double(num);
35     otherwise
36         num = NaN;
37     end
38 end
39
40 function new_rule(obj,name,weight)
41     obj.rules = [obj.rules,struct('Name',name,'Weight',num2str(
42         weight),'Antecedent',[],'Consequent',[])]];
43 end
44
45 function add_antecedent(obj,variable,term)
46     if isempty(obj.rules)
47         error('Rules System is empty');
48     end
49
50     term_aux = [];
51     if ~iscell(term)
52         term = {term};
53     end
54     for i = 1:size(term,1)
55         try
56             for j = 1:7
57                 term_aux = [term_aux,{num2str(term{i,j})}];
58             end
59         catch
60             error('Terms no valids');
61         end
62     end
63     term = term_aux;
64
65     obj.rules(length(obj.rules)).Antecedent = [obj.rules(length(obj
66         .rules)).Antecedent;{variable term}];
67 end
68
69 function new_consequent(obj,variable,term)
70     if isempty(obj.rules)
71         error('Rules System is empty');
72     end
73
74     obj.rules(length(obj.rules)).Consequent = {variable term};
75 end

```

6.1. Clase RulesVisual

```
75 end
```

CÓDIGO 6.1: Archivo *RulesVisual.m* correspondiente al módulo Rule View

6.2. Función Visual

La función *Visual* es la que se encarga de ejecutar la visualización de las reglas. En el Código 6.2 se muestra la implementación de esta clase.

```

1 function Visual(name, rules)
2     %Load VisualRules.jar
3     algorithmPath = fileparts(which('Visual.m'));
4     jarfolder = fullfile(algorithmPath, 'VisualRules.jar');
5     javaaddpath(jarfolder);
6
7     %Initialize VisualRules
8     try
9         visual = javaObject('visualrules.VisualRules');
10    catch
11        disp('*****');
12        disp('See rules is not possible. ');
13        disp('*****');
14
15        %Clear Java
16        clear visual;
17        javarmpath(jarfolder);
18
19        return;
20    end
21
22    %Add rules
23    for i = 1:size(rules,2)
24        namer = rules(i).Name;
25        weight = rules(i).Weight;
26        ant = rules(i).Antecedent;
27        con = rules(i).Consequent;
28
29        %Rule
30        javaMethod('new_rule', visual, namer, weight);
31
32        %Antecedent
33        for j = 1:size(ant,1)
34            namea = ant(j,1);
35            values = ant(j,2);
36            javaMethod('new_antecedent', visual, namea{1}, values{1});
37        end
38
39        %Consequent
40        javaMethod('new_consequent', visual, con{1}, con{2});
41    end
42
43    %Activate panel
44    javaMethod('SeeRules', visual, name);

```


6.2. Función Visual

```
45
46     % Clear Java
47     clear visual;
48     javarmpath(jarfolder);
49 end
```

CÓDIGO 6.2: Archivo *Visual.java* correspondiente al módulo Rule View

6.3. Clase VisualRules

La clase *VisualRules* se encarga de crear la ventana en la que se visualizan las reglas. En el Código 6.3 se muestra la implementación de esta clase.

```

1 /*
2  * To change this license header, choose License Headers in Project
    Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package visualrules;
7
8 import java.util.ArrayList;
9 import java.util.Arrays;
10
11 /**
12  *
13  * @author Federico Garcia-Arevalo Calles
14  */
15 public class VisualRules extends javax.swing.JFrame {
16     /**
17      * Creates new form VisualRules
18      */
19     private final ArrayList rules;
20
21     public VisualRules() {
22         initComponents(0,0);
23         this.rules = new ArrayList();
24         pack();
25     }
26
27     public void new_rule(String name, String weight){
28         if(!this.rules.isEmpty()) CreateRules();
29         this.rules.add(name);
30         this.rules.add(weight);
31         this.rules.add("");
32         this.rules.add("");
33     }
34
35     public void new_antecedent(String name, String[] values){
36         if(this.rules.isEmpty()) return;
37         this.rules.add(name);
38         this.rules.add(String.valueOf(values.length/7));
39         this.rules.addAll(Arrays.asList(values));
40     }
41
42     public void new_consequent(String name, String value){
43         if(this.rules.isEmpty()) return;

```

6.3. Class VisualRules

```
44         this.rules.set(2,name);
45         this.rules.set(3,value);
46     }
47
48     /**
49      * This method is called from within the constructor to initialize the
50      * form.
51      * WARNING: Do NOT modify this code. The content of this method is
52      * always
53      * regenerated by the Form Editor.
54      */
55     @SuppressWarnings("unchecked")
56     // <editor-fold defaultstate="collapsed" desc="Generated Code">
57     private void initComponents(int w, int h) {
58
59         jPanel1 = new javax.swing.JPanel();
60         jLabel1 = new javax.swing.JLabel();
61         _actual_num_row = new javax.swing.JTextField();
62         _OK = new javax.swing.JButton();
63         _info = new javax.swing.JTextPane();
64         _cont = new javax.swing.JScrollPane();
65         _lista = new javax.swing.JPanel();
66
67         setDefaultCloseOperation(javax.swing.WindowConstants.
68             DISPOSE_ON_CLOSE);
69         setMinimumSize(new java.awt.Dimension(860, 642));
70
71         jLabel1.setText("Num of variable per row:");
72
73         _actual_num_row.setHorizontalAlignment(javax.swing.JTextField.
74             CENTER);
75         _actual_num_row.setMaximumSize(new java.awt.Dimension(45, 23));
76         _actual_num_row.setMinimumSize(new java.awt.Dimension(45, 23));
77         _actual_num_row.setPreferredSize(new java.awt.Dimension(45, 23));
78
79         _OK.setText("OK");
80         _OK.setFocusPainted(false);
81         _OK.setMaximumSize(new java.awt.Dimension(60, 23));
82         _OK.setMinimumSize(new java.awt.Dimension(60, 23));
83         _OK.setPreferredSize(new java.awt.Dimension(60, 23));
84         _OK.addActionListener(new java.awt.event.ActionListener() {
85             public void actionPerformed(java.awt.event.ActionEvent evt) {
86                 _OKActionPerformed(evt);
87             }
88         });
89
90         _info.setText("Zoom an area of graph: click left and mark the area
91             to right.\n" +
92             "See all graph: click left and move to left.");
93         _info.setDisabledTextColor(new java.awt.Color(0, 0, 0));
94         _info.setEnabled(false);
95         _info.setOpaque(false);
```

```

92     javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout
        (jPanel1);
93     jPanel1.setLayout(jPanel1Layout);
94     jPanel1Layout.setHorizontalGroup(
95         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.
            Alignment.LEADING)
96         .addGroup(jPanel1Layout.createSequentialGroup()
97             .addGap()
98             .addComponent(jLabel1, javax.swing.GroupLayout.
                PREFERRED_SIZE, 139, javax.swing.GroupLayout.
                PREFERRED_SIZE)
99             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                .UNRELATED)
100            .addComponent(_actual_num_row, javax.swing.GroupLayout.
                PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                javax.swing.GroupLayout.PREFERRED_SIZE)
101            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
                .RELATED)
102            .addComponent(_OK, javax.swing.GroupLayout.PREFERRED_SIZE,
                60, javax.swing.GroupLayout.PREFERRED_SIZE)
103            .addGap(18, 18, 18)
104            .addComponent(_info))
105     );
106     jPanel1Layout.setVerticalGroup(
107         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.
            Alignment.LEADING)
108         .addGroup(jPanel1Layout.createSequentialGroup()
109             .addGap()
110             .addGroup(jPanel1Layout.createParallelGroup(javax.swing.
                GroupLayout.Alignment.BASELINE)
111                 .addComponent(jLabel1, javax.swing.GroupLayout.
                    DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                    Short.MAX_VALUE)
112                 .addComponent(_actual_num_row, javax.swing.GroupLayout.
                    DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                    Short.MAX_VALUE)
113                 .addComponent(_OK, javax.swing.GroupLayout.DEFAULT_SIZE
                    , javax.swing.GroupLayout.DEFAULT_SIZE, Short.
                    MAX_VALUE))
114             .addGap())
115         .addGroup(jPanel1Layout.createSequentialGroup()
116             .addComponent(_info, javax.swing.GroupLayout.PREFERRED_SIZE
                , 43, javax.swing.GroupLayout.PREFERRED_SIZE)
117             .addGap(0, 0, Short.MAX_VALUE))
118     );
119
120     javax.swing.GroupLayout _listaLayout = new javax.swing.GroupLayout(
        _lista);
121     _lista.setLayout(_listaLayout);
122     _listaLayout.setHorizontalGroup(
123         _listaLayout.createParallelGroup(javax.swing.GroupLayout.
            Alignment.LEADING)
124         .addGap(0, w, Short.MAX_VALUE)

```

6.3. Class VisualRules

```
125         );
126         _listaLayout.setVerticalGroup(
127             _listaLayout.createParallelGroup(javax.swing.GroupLayout.
128                 Alignment.LEADING)
129             .addGap(0, h, Short.MAX_VALUE)
130         );
131         _cont.setViewportView(_lista);
132
133         javax.swing.GroupLayout layout = new javax.swing.GroupLayout(
134             getContentPane());
135         getContentPane().setLayout(layout);
136         layout.setHorizontalGroup(
137             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
138                 LEADING)
139             .addComponent(_cont)
140             .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
141                 javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
142         );
143         layout.setVerticalGroup(
144             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
145                 LEADING)
146             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.
147                 createSequentialGroup())
148             .addComponent(jPanel1, javax.swing.GroupLayout.
149                 PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
150                 javax.swing.GroupLayout.PREFERRED_SIZE)
151             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.
152                 RELATED)
153             .addComponent(_cont));
154
155     } // </editor-fold>
156
157     private void _OKActionPerformed(java.awt.event.ActionEvent evt) {
158         // TODO add your handling code here:
159         String text = _actual_num_row.getText();
160         if(text.length() < 1 || text.length() > 2) return;
161
162         try{
163             int num = Integer.parseInt(text);
164             if(num < 1 || num > 99) return;
165
166             change_num_rules_in_row(num);
167         }catch(NumberFormatException e){
168
169         }
170     }
171
172     public void SeeRules(String name){
173         this.setTitle(name);
174         CreateRules();
175         int num = 5;
```

```

169         change_num_rules_in_row(num);
170         _actual_num_row.setText(" " + num);
171         this.setVisible(true);
172     }
173
174     private void CreateRules(){
175         if(this.rules.isEmpty()) return;
176
177         Rule rule = new Rule();
178
179         if(this.rules.size() > 2){
180             // ANTECEDENT
181             rule = getAntecedent(this.rules, rule);
182
183             // CONSEQUENT
184             String consequent_variable = (String) this.rules.get(2);
185             String consequent_term = (String) this.rules.get(3);
186             rule.consequent(consequent_variable, consequent_term);
187         }
188
189         // ADD RULE
190         rule.weight(Float.parseFloat((String) this.rules.get(1)));
191         rule.number((String) this.rules.get(0));
192         _lista.add(rule);
193
194         this.rules.clear();
195     }
196
197     private void change_num_rules_in_row(int num){
198         int h = 6;
199         int w = 0;
200         Rule aux[] = new Rule[_lista.getComponentCount()];
201         for(int i = 0; i < _lista.getComponentCount(); i++){
202             aux[i] = (Rule) _lista.getComponent(i);
203             aux[i].regroup_components(num);
204             aux[i].setSize(aux[i].getPreferredSize());
205             aux[i].setLocation(6, h);
206             h += aux[i].getHeight() + 6;
207             if(aux[i].getWidth() > w) w = aux[i].getWidth();
208         }
209
210         this.remove(jPanel1);
211         this.remove(jLabel1);
212         this.remove(_actual_num_row);
213         this.remove(_OK);
214         this.remove(_info);
215         this.remove(_cont);
216         this.remove(_lista);
217
218         initComponents(w, h);
219
220         for (Rule rule : aux) {
221             _lista.add(rule);

```

6.3. Clase VisualRules

```
222     }
223 }
224
225 private static Rule getAntecedent(ArrayList aux, Rule rule) {
226     // Obtener las variables y terminos que van en la regla
227     int act = 4;
228     while(act < aux.size()){
229         String name = (String) aux.get(act); act++;
230         int numLabels = Integer.parseInt((String) aux.get(act)); act++;
231         int type-variable = 0;
232         ArrayList aux_2 = new ArrayList();
233         for(int k = 0; k < numLabels; k++){
234             String[] data = new String[7];
235             data[0] = (String) aux.get(act); act++;
236             data[1] = (String) aux.get(act); act++;
237             data[2] = (String) aux.get(act); act++;
238             data[3] = (String) aux.get(act); act++;
239             data[4] = (String) aux.get(act); act++;
240             data[5] = (String) aux.get(act); act++;
241             data[6] = (String) aux.get(act); act++;
242             if(data[1].equals(data[2]) && data[1].equals(data[3]) &&
                data[1].equals(data[4]) && type-variable != 2){
243                 type-variable = 1;
244             }else{
245                 type-variable = 2;
246             }
247             aux_2.add(data);
248         }
249         if(type-variable == 1){
250             String[] terms = new String[aux_2.size()];
251             for(int i = 0; i < terms.length; i++){
252                 String[] data = (String[]) aux_2.get(i);
253                 terms[i] = data[0];
254             }
255             rule.add_categorical_antecedent(name, terms);
256         }else if(type-variable == 2){
257             double[][][] series = new double[aux_2.size()][4][2];
258             for(int i = 0; i < series.length; i++){
259                 String[] data = (String[]) aux_2.get(i);
260                 series[i][0][0] = Double.parseDouble(data[1]);
261                 series[i][0][1] = Double.parseDouble(data[5]);
262                 series[i][1][0] = Double.parseDouble(data[2]);
263                 series[i][1][1] = 1;
264                 series[i][2][0] = Double.parseDouble(data[3]);
265                 series[i][2][1] = 1;
266                 series[i][3][0] = Double.parseDouble(data[4]);
267                 series[i][3][1] = Double.parseDouble(data[6]);
268             }
269             rule.add_fuzzy_antecedent(name, series);
270         }
271     }
272     return rule;
273 }
```

```
274
275 // Variables declaration — do not modify
276 private javax.swing.JButton _OK;
277 private javax.swing.JTextField _actual_num_row;
278 private javax.swing.JScrollPane _cont;
279 private javax.swing.JTextPane _info;
280 private javax.swing.JPanel _lista;
281 private javax.swing.JLabel jLabel1;
282 private javax.swing.JPanel jPanel1;
283 // End of variables declaration
284 }
```

CÓDIGO 6.3: Archivo *VisualRules.java* correspondiente al módulo Rule View

6.4. Clase Rule

La clase *Rule* sirve para representar una regla de forma individual. En el Código 6.4 se muestra la implementación de esta clase.

```
1 /*
2  * To change this license header, choose License Headers in Project
3  * Properties.
4  * To change this template file, choose Tools | Templates
5  * and open the template in the editor.
6  */
7
8 package visualrules;
9
10 import java.awt.Component;
11
12 /**
13  *
14  * @author Federico Garcia-Arevalo Calles
15  */
16 public class Rule extends javax.swing.JPanel {
17
18     /**
19      * Creates new form Rules
20      */
21     public Rule() {
22         initComponents();
23         _num_rules = 0;
24         _IF.setSize(_IF.getPreferredSize());
25         _num_row = -1;
26         _weight = 0;
27         _consequent = new ConditionCategoric();
28         _consequent.setLocation(46,0);
29         _thenpanel.add(_consequent);
30     }
31
32     public void weight(float weight) {
33         if(weight < 0) weight = 0;
34         _weight = weight;
35     }
36
37     public void number(String name) {
38         _num.setText(name + ": (WEIGHT = " + _weight + ")");
39     }
40
41     public void regroup_components(int num) {
42         if(_num_rules <= 1 || num == _num_row) return;
43         _num_row = num;
44         if(num < 1) num = 1;
45         Component a = _ifpanel.getComponent(1);
```

```

44     int h,w, line ;
45     w = a.getX() + a.getWidth() + 6;
46     java.awt.Dimension tam = new java.awt.Dimension(w,a.getHeight());
47     h = 0;
48     for(int i = 2; i < _num_rules*2; i = i + 2){
49         // Cambio de linea
50         if ((i/2) %num == 0){
51             w = 30;
52             h = tam.height + 6;
53         }
54
55         // Recolocacion AND
56         a = _ifpanel.getComponent(i);
57         a.setLocation(w,h + 11);
58         w += a.getWidth() + 6;
59
60         // Recolocacion antecedente
61         a = _ifpanel.getComponent(i+1);
62         a.setLocation(w,h);
63         w += a.getWidth() + 6;
64
65         // Cambiar tamanyo
66         if(tam.height < a.getHeight() + h) tam.height = a.getHeight() +
            h;
67         if(tam.width < w) tam.width = w;
68     }
69
70     // Aplicar cambios en los paneles
71     update_panel(tam);
72 }
73
74 private void update_panel(java.awt.Dimension tam){
75     // Actualizar el panel con las reglas
76     _ifpanel.setPreferredSize(tam);
77     _ifpanel.setSize(tam);
78
79     this.updateUI();
80 }
81
82 private void add_antecedent(Component antecedent) {
83     Component ult = _ifpanel.getComponent(_ifpanel.getComponent().
        length - 1);
84     int w = ult.getX() + ult.getWidth() + 6; // 6: el espacio entre
        componentes
85     int h = 0;
86     int lon = _ifpanel.getHeight();
87     int aux_lon;
88
89     // Anyadir un JLabel con el texto AND entre reglas
90     if( _num_rules != 0){
91         javax.swing.JLabel andText = new javax.swing.JLabel();
92         andText.setText("AND");
93         andText.setSize(andText.getPreferredSize());

```

6.4. Clase Rule

```
94         //if((float) (_num_rules + 1) % (_num_row + 1) == 0){ // Cambio
           de linea
95         //     w = 30;
96         //     h = lon + 6;
97         //}
98         andText.setLocation(w,h+11);
99         w += andText.getWidth() + 6; // 16: la altura del componente
           JLabel
100                                     // 6: el espacio entre componentes
101         _ifpanel.add(andText);
102     }
103
104     // Anyadir el antecedente
105     antecedent.setSize(antecedent.getPreferredSize());
106     antecedent.setLocation(w,h);
107     w += antecedent.getWidth() + 6;
108     _ifpanel.add(antecedent);
109
110     // Aplicar cambios en los paneles
111     aux_lon = antecedent.getHeight() + antecedent.getY();
112     if(aux_lon > lon) lon = aux_lon;
113     if(_ifpanel.getWidth() > w) w = _ifpanel.getWidth();
114     update_panel(new java.awt.Dimension(w,lon));
115
116     _num_rules++;
117 }
118
119 public void add_categoric_antecedent(String variable, String[] terms) {
120     ConditionCategoric antecedent = new ConditionCategoric();
121
122     antecedent.setVariable(variable);
123
124     for(String term : terms) {
125         antecedent.addLabel(term);
126     }
127
128     add_antecedent(antecedent);
129 }
130
131 public void add_fuzzy_antecedent(String variable, double[][][] series)
132 {
133     ConditionFuzzyLogic antecedent = new ConditionFuzzyLogic();
134
135     antecedent.setVariable(variable);
136
137     for(double[][] serie : series) {
138         antecedent.new_serie();
139         for(double[] point : serie) {
140             antecedent.add(point[0], point[1]);
141         }
142     }
143
144     antecedent.createGraph();
```

```

144
145     add_antecedent(antecedent);
146 }
147
148 public void consequent(String variable, String term) {
149     _consequent.setVariable(variable);
150     _consequent.addLabel(term);
151     _consequent.setSize(_consequent.getPreferredSize());
152     java.awt.Dimension tam = new java.awt.Dimension(46 + _consequent.
        getWidth(), _consequent.getHeight());
153     _thenpanel.setPreferredSize(tam);
154     _thenpanel.setSize(tam);
155 }
156
157 /**
158  * This method is called from within the constructor to initialize the
159  * form.
160  * WARNING: Do NOT modify this code. The content of this method is
161  * always
162  * regenerated by the Form Editor.
163  */
164 @SuppressWarnings("unchecked")
165 // <editor-fold defaultstate="collapsed" desc="Generated Code"> //GEN-
166 BEGIN: initComponents
167 private void initComponents() {
168
169     _num = new javax.swing.JLabel();
170     _ifpanel = new javax.swing.JPanel();
171     _IF = new javax.swing.JLabel();
172     _thenpanel = new javax.swing.JPanel();
173     _THEN = new javax.swing.JLabel();
174
175     _num.setText("RULE X: (WEIGHT = Y)");
176
177     _IF.setText("IF");
178
179     javax.swing.GroupLayout _ifpanelLayout = new javax.swing.
180         GroupLayout(_ifpanel);
181     _ifpanel.setLayout(_ifpanelLayout);
182     _ifpanelLayout.setHorizontalGroup(
183         _ifpanelLayout.createParallelGroup(javax.swing.GroupLayout.
184             Alignment.LEADING)
185         .addGroup(_ifpanelLayout.createSequentialGroup()
186             .addContainerGap()
187             .addComponent(_IF)
188             .addContainerGap(197, Short.MAX_VALUE))
189     );
190     _ifpanelLayout.setVerticalGroup(
191         _ifpanelLayout.createParallelGroup(javax.swing.GroupLayout.
192             Alignment.LEADING)
193         .addGroup(_ifpanelLayout.createSequentialGroup()
194             .addContainerGap()
195             .addComponent(_IF)

```

6.4. Clase Rule

```
190         .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
191                           Short.MAX_VALUE))
192     );
193     _THEN.setText("THEN");
194
195     javax.swing.GroupLayout _thenpanelLayout = new javax.swing.
196         GroupLayout(_thenpanel);
197     _thenpanel.setLayout(_thenpanelLayout);
198     _thenpanelLayout.setHorizontalGroup(
199         _thenpanelLayout.createParallelGroup(javax.swing.GroupLayout.
200             Alignment.LEADING)
201         .addGroup(_thenpanelLayout.createSequentialGroup()
202             .addContainerGap()
203             .addComponent(_THEN)
204             .addContainerGap(181, Short.MAX_VALUE))
205         );
206     _thenpanelLayout.setVerticalGroup(
207         _thenpanelLayout.createParallelGroup(javax.swing.GroupLayout.
208             Alignment.LEADING)
209         .addGroup(_thenpanelLayout.createSequentialGroup()
210             .addContainerGap()
211             .addComponent(_THEN)
212             .addGap(11, 11, 11))
213         );
214
215     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
216     this.setLayout(layout);
217     layout.setHorizontalGroup(
218         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
219             LEADING)
220         .addGroup(layout.createSequentialGroup()
221             .addContainerGap()
222             .addGroup(layout.createParallelGroup(javax.swing.
223                 GroupLayout.Alignment.LEADING)
224                 .addComponent(_num)
225                 .addGroup(layout.createSequentialGroup()
226                     .addGap(10, 10, 10)
227                     .addGroup(layout.createParallelGroup(javax.swing.
228                         GroupLayout.Alignment.LEADING)
229                         .addComponent(_ifpanel, javax.swing.GroupLayout.
230                             PREFERRED_SIZE, javax.swing.GroupLayout.
231                                 DEFAULT_SIZE, javax.swing.GroupLayout.
232                                     PREFERRED_SIZE)
233                         .addComponent(_thenpanel, javax.swing.
234                             GroupLayout.PREFERRED_SIZE, javax.swing.
235                                 GroupLayout.DEFAULT_SIZE, javax.swing.
236                                     GroupLayout.PREFERRED_SIZE)))
237                     .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
238                         Short.MAX_VALUE))
239             );
240     layout.setVerticalGroup(
241         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
```

```

229         LEADING)
230         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.
231             createSequentialGroup())
232         .addContainerGap()
233         .addComponent(_num)
234         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
235             .RELATED)
236         .addComponent(_ifpanel, javax.swing.GroupLayout.
237             PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
238             javax.swing.GroupLayout.PREFERRED_SIZE)
239         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
240             .RELATED)
241         .addComponent(_thenpanel, javax.swing.GroupLayout.
242             PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
243             javax.swing.GroupLayout.PREFERRED_SIZE)
244         .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
245             Short.MAX_VALUE))
246     );
247 }// </editor-fold>//GEN-END:initComponents
248
249 private int _num_row;
250 private int _num_rules;
251 private float _weight;
252 private ConditionCategoric _consequent;
253
254 // Variables declaration - do not modify//GEN-BEGIN:variables
255 private javax.swing.JLabel _IF;
256 private javax.swing.JLabel _THEN;
257 private javax.swing.JPanel _ifpanel;
258 private javax.swing.JLabel _num;
259 private javax.swing.JPanel _thenpanel;
260 // End of variables declaration//GEN-END:variables
261 }

```

CÓDIGO 6.4: Archivo *Rule.java* correspondiente al módulo Rule View

6.5. Clase ConditionCategoric

La clase *ConditionCategoric* sirve para representar un antecedente de tipo categórico. En el Código 6.5 se muestra la implementación de esta clase.

```
1 /*
2  * To change this license header, choose License Headers in Project
3    Properties.
4  * To change this template file, choose Tools | Templates
5  * and open the template in the editor.
6  */
7
8 package visualrules;
9
10 /**
11  *
12  * @author Federico Garcia-Arevalo Calles
13  */
14 public class ConditionCategoric extends javax.swing.JPanel {
15
16     /**
17      * Creates new form ConditionCategoric
18      */
19     public ConditionCategoric() {
20         initComponents();
21         _num_labels = 0;
22     }
23
24     /**
25      * This method is called from within the constructor to initialize the
26      * form.
27      * WARNING: Do NOT modify this code. The content of this method is
28      * always
29      * regenerated by the Form Editor.
30      */
31     @SuppressWarnings("unchecked")
32     // <editor-fold defaultstate="collapsed" desc="Generated Code"> //GEN-
33     BEGIN: initComponents
34     private void initComponents() {
35
36         _variable = new javax.swing.JLabel();
37         _is = new javax.swing.JLabel();
38         _label = new javax.swing.JPanel();
39         _parenthesis = new javax.swing.JLabel();
40
41         _variable.setForeground(new java.awt.Color(0, 0, 204));
42         _variable.setText("NameVariable");
43     }
44 }
```

```

40
41     _is.setText("IS");
42
43     _label.setLayout(new java.awt.FlowLayout(java.awt.FlowLayout.CENTER
44         , 0, 0));
45
46     _parenthesis.setText("(");
47     _parenthesis.setVisible(false);
48     _label.add(_parenthesis);
49
50     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
51     this.setLayout(layout);
52     layout.setHorizontalGroup(
53         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
54             LEADING)
55         .addGroup(layout.createSequentialGroup()
56             .addGap(0)
57             .addComponent(_variable)
58             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.
59                 UNRELATED)
60             .addComponent(_is)
61             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.
62                 RELATED)
63             .addComponent(_label, javax.swing.GroupLayout.
64                 PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
65                 javax.swing.GroupLayout.PREFERRED_SIZE)
66             .addGap(javax.swing.GroupLayout.DEFAULT_SIZE,
67                 Short.MAX_VALUE))
68         );
69     layout.setVerticalGroup(
70         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
71             LEADING)
72         .addGroup(layout.createSequentialGroup()
73             .addGap(0)
74             .addGroup(layout.createParallelGroup(javax.swing.
75                 GroupLayout.Alignment.LEADING)
76                 .addGroup(layout.createParallelGroup(javax.swing.
77                     GroupLayout.Alignment.BASELINE)
78                     .addComponent(_variable)
79                     .addComponent(_is))
80                 .addComponent(_label, javax.swing.GroupLayout.
81                     PREFERRED_SIZE, javax.swing.GroupLayout.
82                     DEFAULT_SIZE, javax.swing.GroupLayout.
83                     PREFERRED_SIZE))
84             .addGap(javax.swing.GroupLayout.DEFAULT_SIZE,
85                 Short.MAX_VALUE))
86         );
87 }
88
89 // </editor-fold> //GEN-END: initComponents
90
91 public void setVariable(String variable) {
92     _variable.setText(variable);
93 }
94

```


6.5. Class ConditionCategoric

```
79     public void addLabel(String label) {
80         javax.swing.JLabel new_label = new javax.swing.JLabel(label);
81         if(_num_labels > 1) {
82             _label.remove(_label.getComponentCount() - 1);
83         }
84         if(_num_labels > 0) {
85             _label.add(new javax.swing.JLabel(" OR "));
86             _parenthesis.setVisible(true);
87         }
88         new_label.setForeground(Color.BLUE);
89         _label.add(new_label);
90         if(_num_labels > 0) {
91             _label.add(new javax.swing.JLabel(")"));
92         }
93         _num_labels++;
94     }
95
96     int _num_labels;
97
98     // Variables declaration - do not modify//GEN-BEGIN:variables
99     private javax.swing.JLabel _is;
100    private javax.swing.JPanel _label;
101    private javax.swing.JLabel _parenthesis;
102    private javax.swing.JLabel _variable;
103    // End of variables declaration//GEN-END:variables
104 }
```

CÓDIGO 6.5: Archivo *ConditionCategoric.java* correspondiente al módulo Rule View

6.6. Clase ConditionFuzzyLogic

La clase *ConditionFuzzyLogic* sirve para representar un antecedente al que se le ha aplicado lógica difusa. En el Código 6.6 se muestra la implementación de esta clase.

```

1 /*
2  * To change this license header, choose License Headers in Project
    Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package visualrules;
7
8 import java.util.ArrayList;
9 import org.jfree.chart.ChartFactory;
10 import org.jfree.chart.ChartFrame;
11 import org.jfree.chart.ChartPanel;
12 import org.jfree.chart.JFreeChart;
13 import org.jfree.chart.plot.PlotOrientation;
14 import org.jfree.data.xy.XYSeries;
15 import org.jfree.data.xy.XYSeriesCollection;
16
17 /**
18  *
19  * @author Federico Garcia-Arevalo Calles
20  */
21 public class ConditionFuzzyLogic extends javax.swing.JPanel {
22
23     private static ArrayList _series;
24     private static int _num_series;
25
26     /**
27      * Creates new form ConditionFuzzyLogic
28      */
29     public ConditionFuzzyLogic() {
30         initComponents();
31         _series = new ArrayList();
32         _num_series = 0;
33     }
34
35     /**
36      * This method is called from within the constructor to initialize the
        form.
37      * WARNING: Do NOT modify this code. The content of this method is
        always
38      * regenerated by the Form Editor.
39      */
40     @SuppressWarnings("unchecked")

```

6.6. Class ConditionFuzzyLogic

```
41 // <editor-fold defaultstate="collapsed" desc="Generated Code"> //GEN-
    BEGIN: initComponents
42 private void initComponents() {
43
44     _variable = new javax.swing.JLabel();
45     _is = new javax.swing.JLabel();
46     _label = new javax.swing.JPanel();
47
48     _variable.setForeground(new java.awt.Color(0, 0, 204));
49     _variable.setText("NameVariable");
50
51     _is.setText("IS");
52
53     javax.swing.GroupLayout _labelLayout = new javax.swing.GroupLayout(
        _label);
54     _label.setLayout(_labelLayout);
55     _labelLayout.setHorizontalGroup(
56         _labelLayout.createParallelGroup(javax.swing.GroupLayout.
            Alignment.LEADING)
57         .addGroup(
58             .addGap(0, 393, Short.MAX_VALUE)
59         );
60     _labelLayout.setVerticalGroup(
61         _labelLayout.createParallelGroup(javax.swing.GroupLayout.
            Alignment.LEADING)
62         .addGroup(
63             .addGap(0, 263, Short.MAX_VALUE)
64         );
65
66     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
67     this.setLayout(layout);
68     layout.setHorizontalGroup(
69         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
            LEADING)
70         .addGroup(layout.createSequentialGroup()
71             .addGap(0, 0, Short.MAX_VALUE)
72             .addComponent(_variable)
73             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.
                UNRELATED)
74             .addComponent(_is)
75             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.
                UNRELATED)
76             .addComponent(_label, javax.swing.GroupLayout.PREFERRED.
                SIZE, javax.swing.GroupLayout.PREFERRED.SIZE,
                javax.swing.GroupLayout.PREFERRED.SIZE)
77             .addGap(0, 0, Short.MAX_VALUE)
78         );
79     layout.setVerticalGroup(
80         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
            LEADING)
81         .addGroup(layout.createSequentialGroup()
82             .addGap(0, 0, Short.MAX_VALUE)
83             .addGroup(layout.createParallelGroup(javax.swing.
                GroupLayout.Alignment.LEADING)
84                 .addComponent(_variable)
85                 .addComponent(_is)
86                 .addComponent(_label)
87             )
88         );
89 }
```

```

82         .addGroup(layout.createParallelGroup(javax.swing.
           GroupLayout.Alignment.BASELINE)
83         .addComponent(_variable)
84         .addComponent(_is))
85         .addComponent(_label, javax.swing.GroupLayout.
           PREFERRED_SIZE, javax.swing.GroupLayout.
           DEFAULT_SIZE, javax.swing.GroupLayout.
           PREFERRED_SIZE))
86     .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
           Short.MAX_VALUE))
87     );
88 }// </editor-fold>//GEN-END: initComponents
89
90 public void new_serie() {
91     _series.add(new XYSeries("" + _num_series));
92     _num_series++;
93 }
94
95 public void add(Number X, Number Y) {
96     ((XYSeries) _series.get(_num_series - 1)).add(X,Y);
97 }
98
99 public void setVariable(String variable) {
100     _variable.setText(variable);
101 }
102
103 public void createGraph() {
104     _label.setSize(_label.getPreferredSize());
105
106     XYSeriesCollection dataset = new XYSeriesCollection();
107     for(Object serie : _series){
108         dataset.addSeries((XYSeries) serie);
109     }
110
111     JFreeChart chart = ChartFactory.createXYAreaChart("", "", "", dataset,
112         PlotOrientation.VERTICAL, false, false, false)
113         ;
114
115     ChartFrame frame = new ChartFrame("", chart);
116     frame.pack();
117     frame.setSize(_label.getSize());
118     frame.setResizable(false);
119
120     ChartPanel panel = frame.getChartPanel();
121     panel.setPopupMenu(null);
122     //panel.setDomainZoomable(false);
123     //panel.setRangeZoomable(false);
124
125     _label.add(panel);
126 }
127
128 // Variables declaration - do not modify//GEN-BEGIN: variables
private javax.swing.JLabel _is;

```

6.6. Clase ConditionFuzzyLogic

```
129     private javax.swing.JPanel _label;  
130     private javax.swing.JLabel _variable;  
131     // End of variables declaration//GEN-END:variables  
132 }
```

CÓDIGO 6.6: Archivo *ConditionFuzzyLogic.java* correspondiente al módulo Rule View

7 Módulo JFML

7.1. Clase RulesExport

La clase *RulesExport* configura las reglas del algoritmo para su exportación. En el Código 7.1 se muestra la implementación de esta clase.

```
1 classdef RulesExport < handle
2     properties
3         rules = [];
4         knowledge_base = [];
5     end
6
7     methods
8         function export_rules(obj,dir,name)
9             if isempty(obj.rules) || isempty(obj.knowledge_base)
10                 error('Rules or Knowledge base is empty')
11             end
12
13             addpath(fullfile(fileparts(which('RulesExport.m'))),'JFML');
14
15             JFML(dir,name,obj.knowledge_base,obj.rules);
16
17             rmpath(fullfile(fileparts(which('RulesExport.m'))),'JFML');
18         end
19
20         function new_variable(obj,name, domain_left, domain_right)
21             domain_left = obj.detect_number(domain_left);
22             if isnan(domain_left)
23                 error('Domain left is not detected like a number.')
24             end
25
```

```

26         domain_right = obj.detect_number(domain_right);
27         if isnan(domain_right)
28             error('Domain right is not detected like a number.')
29         end
30
31         obj.knowledge_base = [obj.knowledge_base, struct('Name', name, '
DomainLeft', domain_left, 'DomainRight', domain_right, 'Terms'
, [])];
32     end
33
34     function add_terms(obj, name, p1, p2, p3, p4)
35         if isempty(obj.knowledge_base)
36             error('Knowledge base is empty');
37         end
38
39         p1 = obj.detect_number(p1);
40         p2 = obj.detect_number(p2);
41         p3 = obj.detect_number(p3);
42         p4 = obj.detect_number(p4);
43         if isnan(p1) || isnan(p2) || isnan(p3) || isnan(p4)
44             error('A value is not detected like a number.')
45         end
46
47         obj.knowledge_base(length(obj.knowledge_base)).Terms = [obj.
knowledge_base(length(obj.knowledge_base)).Terms; {name p1
p2 p3 p4}];
48     end
49
50     function new_rule(obj, name, weight)
51         weight = obj.detect_number(weight);
52         if isnan(weight)
53             error('Weight is not detected like a number.')
54         end
55
56         obj.rules = [obj.rules, struct('Name', name, 'Weight', weight, '
Antecedent', [], 'Consequent', [])];
57     end
58
59     function add_antecedent(obj, variable, term)
60         if isempty(obj.rules)
61             error('Rules base is empty');
62         end
63
64         obj.rules(length(obj.rules)).Antecedent = [obj.rules(length(obj
.rules)).Antecedent; {variable term}];
65     end
66
67     function new_consequent(obj, variable, term)
68         if isempty(obj.rules)
69             error('Rules base is empty');
70         end
71
72         obj.rules(length(obj.rules)).Consequent = {variable term};

```


7.1. Clase RulesExport

```
73     end
74 end
75
76 methods( Access = private , Static)
77     function num = detect_number(num)
78         if iscell(num)
79             if ~isempty(find(size(num) ~= 1,1))
80                 num = NaN;
81             else
82                 num = num{1};
83             end
84         end
85
86         switch class(num)
87             case 'double'
88                 if ~isempty(find(size(num) ~= 1,1))
89                     num = NaN;
90                 end
91             case 'char'
92                 num = str2double(num);
93             otherwise
94                 num = NaN;
95         end
96     end
97 end
98 end
```

CÓDIGO 7.1: Archivo *RulesExport.m* correspondiente al módulo JFML

7.2. Función JFML

La función *JFML* es la que se encarga de ejecutar la exportación de las reglas. En el Código 7.2 se muestra la implementación de esta clase.

```

1 function JFML(dir, name, knowledge_base, rules)
2     % Load JFML.jar
3     algorithmPath = fileparts(which('JFML.m'));
4     jarfolder = fullfile(algorithmPath, 'JFML.jar');
5     javaaddpath(jarfolder);
6
7     % Initialize FuzzyInferenceSystem
8     f = javaObject('jfml.FuzzyInferenceSystem');
9
10    %KNOWLEDGE BASE
11    KnowledgeBase(f, knowledge_base);
12
13    %RULE BASE
14    RuleBase(f, rules);
15
16    %WRITING INTO AN XML FILE
17    WriteFile(f, dir, name);
18
19    % Clear java
20    clear f;
21    javarmpath(jarfolder);
22 end
23
24 function KnowledgeBase(f, knowledge_base)
25     % Initialize KnowledgeBaseType
26     kb = javaObject('jfml.knowledgebase.KnowledgeBaseType');
27     javaMethod('setKnowledgeBase', f, kb);
28
29     %FUZZY VARIABLE
30     for i = 1:length(knowledge_base)-1
31         s = javaObject('jfml.knowledgebase.variable.FuzzyVariableType',
32             knowledge_base(i).Name, knowledge_base(i).DomainLeft,
33             knowledge_base(i).DomainRight);
34
35         %FUZZY TERM
36         for j = 1:size(knowledge_base(i).Terms, 1)
37             term = knowledge_base(i).Terms(j, :);
38             st = javaObject('jfml.term.FuzzyTermType', term{1}, 7, [term{2},
39                 term{3}, term{4}, term{5}]);
40
41             javaMethod('addFuzzyTerm', s, st);
42         end
43     end
44
45     javaMethod('addVariable', kb, s);

```

7.2. Función JFML

```
42     end
43
44     %OUTPUT CLASS
45     s = javaObject( 'jfml.knowledgebase.variable.FuzzyVariableType',
        knowledge_base(length(knowledge_base)).Name, knowledge_base(length(
            knowledge_base)).DomainLeft, knowledge_base(length(knowledge_base)).
            DomainRight);
46     javaMethod( 'setType', s, 'output' );
47
48     %FUZZY TERM OUTPUT CLASS
49     for j = 1:size(knowledge_base(length(knowledge_base)).Terms,1)
50         term = knowledge_base(length(knowledge_base)).Terms(j,:);
51
52         st = javaObject( 'jfml.term.FuzzyTermType', term{1},7,[term{2},term
            {3},term{4},term{5}]);
53
54         javaMethod( 'addFuzzyTerm', s, st);
55     end
56
57     javaMethod( 'addVariable', kb, s);
58 end
59
60 function RuleBase(f, rules)
61     % Initialize MamdaniRuleBaseType
62     rb = javaObject( 'jfml.rulebase.MamdaniRuleBaseType', '' );
63     kb = javaMethod( 'getKnowledgeBase', f);
64
65     % RULES
66     for i = 1:length(rules)
67         % ANTECEDENT
68         ant = javaObject( 'jfml.rule.AntecedentType' );
69         for j = 1:size(rules(i).Antecedent,1)
70             ant_aux = rules(i).Antecedent(j,:);
71             a = javaMethod( 'getVariable', kb, ant_aux{1});
72             b = javaMethod( 'getTerm', a, ant_aux{2});
73             javaMethod( 'addClause', ant, javaObject( 'jfml.rule.ClauseType', a,
                b));
74         end
75
76         % CONSEQUENT
77         con = javaObject( 'jfml.rule.ConsequentType' );
78         a = javaMethod( 'getVariable', kb, rules(i).Consequent{1});
79         b = javaMethod( 'getTerm', a, rules(i).Consequent{2});
80         javaMethod( 'addThenClause', con, a, b);
81
82         % ADD RULE
83         r = javaObject( 'jfml.rule.FuzzyRuleType', rules(i).Name, 'and', 'MIN',
            javaObject( 'java.lang.Float', rules(i).Weight));
84         javaMethod( 'setAntecedent', r, ant);
85         javaMethod( 'setConsequent', r, con);
86
87         javaMethod( 'addRule', rb, r);
88     end
```

```
89
90     javaMethod('addRuleBase',f,rb);
91 end
92
93 function WriteFile(f,dir,name)
94     % Configure directory
95     if ~exist(dir,'dir')
96         mkdir(dir);
97     end
98
99     % Export JFML
100    try
101        disp('Export JFML...');
102        a = javaObject('jfml.JFML');
103        file = javaObject('java.io.File',[dir '/' name '_JFML.xml']);
104        javaMethod('writeFSTtoXML',a,f,file);
105    catch
106        disp('JFML can not export');
107    end
108
109    % Export PMML
110    try
111        disp('Export PMML...');
112        b = javaObject('jfml.compatibility.ExportPMML');
113        file = [dir '/' name '_PMML.xml'];
114        javaMethod('exportFuzzySystem',b,f,file);
115    catch
116        disp('PMML can not export');
117    end
118 end
```

CÓDIGO 7.2: Archivo *JFML.m* correspondiente al módulo JFML