New Regressor Evaluator

```
%% ¡header!
NNRegressorEvaluator < NNEvaluator (nne,
   evaluator for a neural network
   regressor) evaluates the performance
   of a neural network regressor with a
   specific dataset.

%% ¡description!
This evaluator evaluates the performance
   of a neural network regressor
on root mean square error (RMSE).

%% ¡props!

%%% ¡prop!
RMSE (result, scalar) is the root mean
   squared error between targets and
   predictions for validation set.
%%%% ¡calculate!
if nne.get('GR_PREDICTION').get('SUB_DICT
   ').length() == 0
    value = 0;
else
    preds = cellfun(@(x) cell2mat(x.get('
       PREDICTION'))', nne.memorize('
       GR_PREDICTION').get('SUB_DICT').
       getItems(), 'UniformOutput', false
       );
    preds = cell2mat(preds);
    targets = cellfun(@(x) cell2mat(x.get
       ('TARGET')), nne.get('
       GR_PREDICTION').get('SUB_DICT').
       getItems(), 'UniformOutput', false
```

```
        );
    targets = cell2mat(targets);
    value = double(sqrt(mean((preds -
        targets).^2)));
end

%%% ¡prop!
SCATTER_CHART (result, matrix) creates a
    scatter chart with circular markers at
     the locations specified by
    predictions and targets.
%%%% ¡calculate!
if nne.get('GR_PREDICTION').get('SUB_DICT
    ').length() == 0
    value = 0;
else
    preds = cellfun(@(x) cell2mat(x.get('
        PREDICTION'))', nne.memorize('
        GR_PREDICTION').get('SUB_DICT').
        getItems(), 'UniformOutput', false
        );
    preds = cell2mat(preds);
    targets = cellfun(@(x) cell2mat(x.get
        ('TARGET')), nne.get('
        GR_PREDICTION').get('SUB_DICT').
        getItems(), 'UniformOutput', false
        );
    targets = cell2mat(targets);
    value = double([preds' targets']);
end
%%%% ¡gui!
pr = PanelPropMatrix('EL', nne, 'PROP',
    NNRegressorEvaluator.SCATTER_CHART,
    ...
    'ROWNAME', char("cellfun(@(x) x.get('
        ID'), pr.get('EL').memorize('GR').
        get('SUB_DICT').getItems(), '
        UniformOutput', false)"), ...
    'COLUMNNAME', char("{'Prediction', '
        Target'}"), ...
    varargin{:});
```

```
40
41 %%% ¡prop!
42 PFSP (gui, item) contains the panel
       figure of the scatter plot.
43 %%%% ¡settings!
44 'PFScatterPlot'
45 %%%% ¡postprocessing!
46 if ~braph2_testing % to avoid problems
       with isqual when the element is
       recursive
47     nne.memorize('PFSP').set('NNE', nne)
48 end
49 %%%% ¡gui!
50 pr = PanelPropItem('EL', nne, 'PROP',
       NNRegressorEvaluator.PFSP, ...
51     'GUICLASS', 'GUIFig', ...
52     varargin{:});
53
54 %% ¡props_update!
55
56 %%% ¡prop!
57 NN (data, item) is a neural network model
       that needs to be evaluated.
58 %%%% ¡settings!
59 'NNRegressorDNN'
60 %%%% ¡default!
61 NNRegressorDNN()
62
63 %%% ¡prop!
64 GR_PREDICTION (result, item) is a group
       of NN subjects containing the
       prediction from the neural network.
65 %%%% ¡settings!
66 'NNGroup'
67 %%%% ¡calculate!
68 if nne.get('GR').get('SUB_DICT').length()
       == 0
69     value = NNGroup();
70 else
71     nn = nne.memorize('NN');
72     nn_gr = nne.get('GR');
```

```
73      inputs = nn.reconstruct_inputs(nn_gr)
            ;
74      net = nn.get('MODEL');
75      if isa(net, 'NoValue') || ~BRAPH2.
            installed('NN', 'msgbox')
76          predictions = zeros(nn_gr.get('
                SUB_DICT').length(), 1);
77      else
78          predictions = net.predict(inputs)
                ;
79      end
80      nn_gr_pred = NNGroup( ...
81          'SUB_CLASS', nn_gr.get('SUB_CLASS
                '), ...
82          'SUB_DICT', IndexedDictionary('
                IT_CLASS', 'Subject') ...
83          );
84
85      nn_gr_pred.set( ...
86          'ID', nn_gr.get('ID'), ...
87          'LABEL', nn_gr.get('LABEL'), ...
88          'NOTES', nn_gr.get('NOTES'), ...
89          'FEATURE_SELECTION_MASK', nn_gr.
                get('FEATURE_SELECTION_MASK')
                ...
90          );
91
92      % add subejcts, it has to be created
            as new subjects
93      sub_dict = nn_gr_pred.get('SUB_DICT')
            ;
94      subs = nn_gr.get('SUB_DICT').getItems
            ();
95      for i = 1:1:length(subs)
96          sub = NNSubject( ...
97              'ID', [subs{i}.get('ID') ' in
                    ' nn_gr.get('ID')], ...
98              'BA', subs{i}.get('BA'), ...
99              'age', subs{i}.get('age'),
                    ...
100             'sex', subs{i}.get('sex'),
```

```
                ...
101             'input', subs{i}.get('input')
                    , ...
102             'INPUT_TYPE', subs{i}.get('
                    INPUT_TYPE'), ...
103             'INPUT_LABEL', subs{i}.get('
                    INPUT_LABEL'), ...
104             'PREDICTION', {predictions(i,
                    :)}, ...
105             'TARGET', subs{i}.get('TARGET
                    '), ...
106             'TARGET_NAME', subs{i}.get('
                    TARGET_NAME') ...
107             );
108         sub_dict.add(sub);
109     end
110     nn_gr_pred.set('SUB_DICT', sub_dict);
111
112     value = nn_gr_pred;
113 end
114
115 %% ¡staticmethods!
116 function lbls = measure_types()
117     lbls = {'Global', 'Nodal', 'Binodal'
            };
118 end
```