UNIVERSIDADE DE AVEIRO

Machine Learning - Lab work 7

Objectives: Introduction to Rapid Miner (RM). Implementation of Neural Networks and k-Nearest Neighbor (k-NN) learning algorithms in Rapid Miner.

Ex1. The Pima Indians Diabetes Dataset collects data about 768 patients with and without diabetes onset within 5 years. Dataset features are:

1) Pregnant: number of times to be pregnant
2) PlasmaGlucose: the plasma glucose concentration measured using a 2h oral glucose tolerance test.
3) DiastolicBP: diastolic blood pressure (mmHg)
4) TriceptsSFT: tricepts skin fold thickness (mm)
5) SerumInsulin: two-hours serum insulin (muU/mt)
6) BMI: body mass index (weight in kg; height in m)
7) DPF: diabetes pedigree function
8) Age: age of the patient in years
9) Class: diabetes onset within 5 years (0 or 1)

1.1. Install Rapid Miner (RM) https://rapidminer.com/ and start it. RM environment has two main views: Design (where you create your models) and Results (where you observe the results after running the model).

1.2. Import the data from *pima-indians-diabetes_excel.xlsx* file into the RM Local Repository (or create a new repository) with *Add Data* operator: At step 1 check the cell *Define header row*, at step 2 change the role of the last column (*Class*) as label (the color of column *Class* will change). At the next step choose a proper name, for example *DiabetesData.* You will observe the imported data into the Results View.

1.3. Switch to Design View and create a new process (for example Diabetes_process1):

a) Retrieve *DiabetesData* from your repository (drag *DiabetesData* icon into your process or use *Retrieve* operator to load it) and connect the output of the *Retrieve* block (out) by arrow to the process output *res.*

b) Run the process, switch to Results View and explore *Data, Statistics and Charts* menus. Observe the missing values in some features such as TricepsSFT and SerumInsulin which are given as 0 values. In *Charts*, choose *Quartile* option to analyze individual feature distributions. Choose *Scatter matrix* option, choose *class* for *Plots* to observe 2D distribution plots (the examples that belong to different classes have different colors).

How to find an operator and assign its parameters: In menu Operators, at the empty line on the top, insert the first few letters of the operator you are looking for until the block of the searched operator appears. Drag the block in your process and in menu Parameters choose the parameters of this operator.

c) Switch to Design View, add a new operator *Filter Examples* after the *DiabetesData* block and connect the output of the first block to the input of the second. *Filter Examples* operator can remove records with missing values in some features. Often missing values are expressed by 0 or ? in the data set. For example remove 0 values in feature TricepsSFT. Tip: use *Add filters (*from menu Parameters*)*=> TricepsSFT ~= 0. Add a second entry in the same filter to remove missing values in SerumInsulin (denoted as 0). Run the process to observe in the Results View the retained examples. (initially 769 => after filtering around 394).

d) Switch to Design View, add a new operator *Numerical to Polynomial* next to *Filter Examples* operator to transform the numerical value of the class (0,1..) to nominal one (zero, first). In menu Parameters: check the cell *include special attributes*; choose attribute filter type: *single*; attribute: *class* that will be transformed.

e) Save the modified data set adding a new operator *Store* next to *Numerical to Polynomial* operator, choose a name for the new data file (e.g. DiabetesData_classification) in menu Parameters of operator *Store*.

Note: The above procedure has to be applied always when the label (the class) is an integer value (1,2,3..) and the task is to classify the data.

1.4 Create a new process (Diabetes_NN) to classify the Diabetes data (see Fig. 1).
Retrieve the modified data file, add *Cross Validation* operator (connect its first 4 outputs to the process outputs *res*) to do cross-validation of the performance of the classification operator. This operator divides the process into training and testing sub-processes.

In the **training sub-process** add *Neural Net* operator as the classifier.
Choose the NN parameters: *Learning rate = 0.05; Momentum=0, training cycles =400.*
hidden layers=> hidden layer name = h1 (give a name); hidden layer size = 3 (number of nodes).

In the **testing sub-process** add *Apply Model* and *Performance (classification)* operators.
Run the process. In View Results observe the confusion matrix (accuracy, precision, recall).

1.5 Go back to Design View and explore the following options:

a) 10-fold cross validation versus leave one out options of operator *Cross Validation*.

b) Classification without data normalization versus data normalization (add operator Normalize).

c) Classification with all Attributes versus classification with selected Attributes (add operator Select Attributes to ignore some of the attributes and run again the process). First observe data into Results View - Charts and decide if there is a subset of attributes based on which the Diabetes data can be better classified.

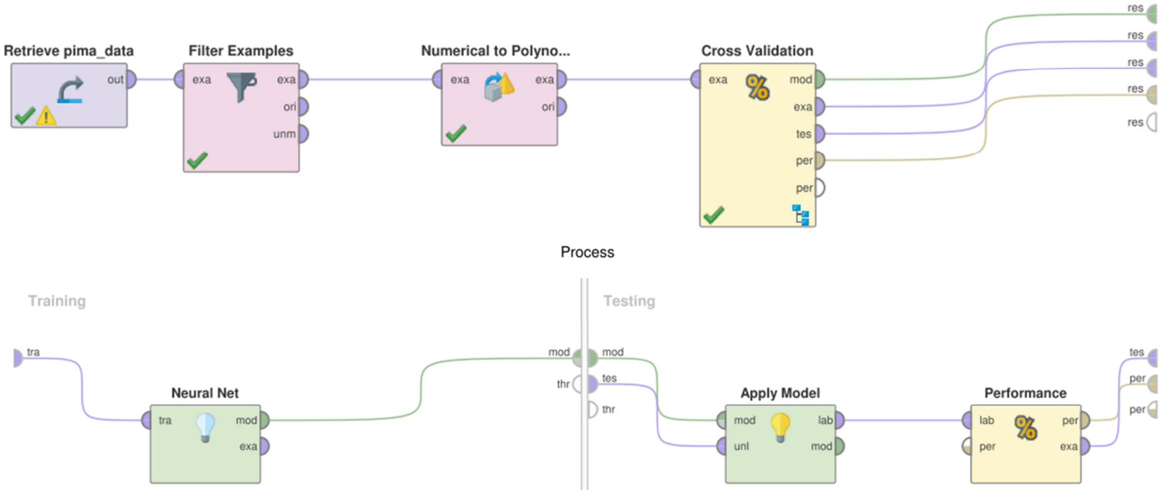

Fig. 1 RM classification model

Ex. 2. *tean_training.xlsx* file consists of data (the weight [kg] and the height [cm]) of teenage boys and girls.
You need to train a k-NN classifier to decide if a new example x =[60 kg 165 cm] is a boy or a girl.
Create a new process to solve this problem in RM.

a) Import in your repository the training data (file *tean_training.xlsx).* Create an excel file with the test example x and import it in RM.

b) Drag the imported data files into your process*.*

c) Drag the operator *k-NN.* Select its parameters (e.g. *k*=1 or 3), use as Numerical Measure -Euclidian distance.

d) Drag the operator *Apply Model*. Connect the test data to its second input.

e) Run the process for different *k*.

f) Make comparison with other classifiers.

Remarks: RM is an open-source free software, it is not perfect, bugs may occur !