ISEL

MERCM – ADEETC

AMD / MLDM (Machine Learning and Data Mining) **– Module of Practice**

Paulo Trigo Silva

1. Orange Data Mining – past, present and future

Take a look ay: “`http://videolectures.net/is2012_demsar_orange/`” and get the perspective of professor Janez Demšar (from Faculty of Computer and Information Science; University of Ljubljana) one of the co-founders of “Orange Data Mining”. That same information is also available in “`p01_Demsar-Orange-DataMining_perspective.pdf`”.

2. Contingency matrix and data distribution – graphic mode

Consider the dataset: “`.\_dataset\lenses.tab`”. Execute “Orange”.

a) Select “Data” separator, choose “File” load “_dataset\lenses.tab” and build workflow “File” >> “Data Table” >> “Select Rows” >> “Select Columns” >> “Data Table (1)”. Configure the workflow such that “Data Table (1)” only exhibits the “lenses” and “tear_rate” columns where “tear_rate=normal”.

b) Select “Visualize” separator and build “Data Table” >> “Distributions”. Open the “Distributions” widget, set “Split by” to “(None)” and select each “Variable” to analyze its values’ frequencies.

c) Now set “Split by” to an attribute and combine it with a “Variable”; check “Show probabilities”, select an attribute’s value (the attribute is the one chosen for the “Split by”). Analyze the graph and click on the “colored bars”; notice the dashed line around the bars; wait until you see a (yellow) table; in that table, the column named “(in group)” represents a conditional probability.

d) Finally, link “Data Table (1)” to “Distributions (1)”, and make the necessary calculations to better understand how the (yellow) table may represent P[lenses=none | tear_rate=normal]=0.25).

e) Also connect “File” to “Box Plot”, execute and analyze the results. You can check a workflow similar to the one you just built in “`_exercise_01.ows`”

f) Choose some other datasets, (in the “File” icon select “Browse documentation datasets”” or look at “(…)\Orange\Lib\site-packages\Orange\datasets”) and repeat the above experiments.

Paulo Trigo Silva

3. Contingency and "error-matrix" – programmatic mode

Consider the file: "`a01_dataset_analysis.py`"; use also dataset: "._dataset\lenses.tab".

a) Go to function "`test()`" and comment everything except for the computation of missing values. Also make sure that you are using the "._dataset\lenses.tab". Execute and analyze the results. Copy the dataset and change the data (of the copy) in order to increase the frequency of the missing values. Execute again and validate the results. Repeat the above test using the dataset from "._dataset\lenses_with_missingValues.tab". Analyze and validate the results.

b) Eliminate the comments on the construction of the "contingency matrix" (or "frequency matrix") of the nominal (discrete) attributes; make sure that you are now using the dataset from "._dataset\lenses_fromLecture.tab". Execute the code and compare the result with the info that you have from the lecture where 1R was exposed.

c) Eliminate comments in order to execute the first call to the "`show_conditionalProbability`" function (keep using the "._dataset\lenses_fromLecture.tab). Analyze the code of the "`get_conditionalProbability`" function and open an Excel file. Use the Excel file to "replicate" exactly the same computations that are done in the function. Make sure that you get exactly the same results.

d) Eliminate comments to also execute the second call to the "`show_conditionalProbability`". Explore these calls freely and use different datasets (e.g., those provided in the "`test`" function or any others that you may consider.

e) Now, analyze the "`get_errorMatrix`" function. Use the same previous Excel file and "replicate" exactly the same computations that are done in the function (use that same dataset!).

f) Uncomment the call to "`get_errorMatrix`" function and all the remaining code. Execute the code and compare the result with the lecture where 1R was exposed (cf., slide #20, first table).

Note: you may use (or extend) the above code to support your implementation of 1R algorithm.

4. … oneR and integrate the "command-line" design pattern

a) Consider the "oneR_classifier.pyc" implementation of OneR algorithm. It only uses filename "`.\_dataset\lenses.tab`". You can use this implementation to validate your own solution.

b) At this point you may consider the "command-line" design pattern (developed in the previous module of practice; cf., "`e_Config.py`") so that, in your own implementation, parameters such as the dataset file name are passed in the command line.