EECS 837
Spring 2016

EECS 837 Programming Project

due: midnight between April 25 and April 26, 2016

Implement an enhanced version of the LEM2 algorithm for rule induction, ready for *k*-fold cross-validation. **Your project should be implemented on one of the departmental computers, e.g., cycle 1, under Linux**. You may use any programming language you wish. Your program should be able to deal with symbolic and numerical attributes.

Input data files for the LEM2 algorithm will be in the LERS format. Input data files will start from the list of variable declarations. This list starts form "<", then a space, then a sequence of the following symbols: "a", "x", or "d", separated by spaces; then another space, the last symbol of the list is ">". You may ignore this line. The second list of the input data file starts from "["; then a space, then comes a list of attribute and decision names, separated by spaces; then another space, and then "]". The decision will be always the last variable, all remaining variables are attributes. The following part contains values of the attributes and decisions. The input data file may contain comments. A comment starts from "!"; everything that follows "!", until the end of the line, is the comment. Obviously, comments should be ignored during reading of data. Attribute values are separated by spaces. Spaces should be understood not only as ordinary spaces but also as white space characters, such as the end of line, tab, etc. Any line of the input data file may start from one or more spaces and one or more spaces may end it. Note that a "line" does not need to be a physical line (it is rather a paragraph). Examples of symbolic values are: *medium*, 12..14, 1.25..2.37, etc. Examples of numerical values are 42, –12.45, etc. Use *all cutpoints strategy* to deal with numerical attributes.

Your program should induce a rule set from the entire data sets. Then it should randomly change the order of all cases in the input data file, divide the input data file into approximately equal *k* parts, induce rules from the second to *k*-th piece, and check the first part against the rule set using *strength, support and partial matching factor*. Then the same procedure should be done by replacing the first part by the second part, etc. Each case will participate *k* times in training and once in testing.

The first question of your project should be for the name of the input data file. The expected response of the user is the name followed by pressing the <RETURN> key. The next question should be what is the value of the parameter *k*. The next question should be: what kind of approximation should be used: lower or upper. Then the program should ask for a name of the output data file (with rules).

Your program should tell what is the error rate and create the output data file with rules, induced from the entire data sets, one rule per line, e.g.

```
(2, 2, 2)
(Eyes, blue) & (Hair, blond) -> (Attractiveness, plus)
(1, 1, 1)
(Hair, red) -> (Attractiveness, plus)
(1, 3, 3)
(Eyes, brown) -> (Attractiveness, minus)
(1, 3, 3)
(Hair, dark) -> (Attractiveness, minus)
```

General Remarks. Your program should be able to deal with unexpected answers of the user and not crush but rather repeat the question. Use of recursion is not encouraged. You may assume that the input data file does not contain errors. You should expect input data files of any size, with thousands examples and hundreds attributes. **Extra credit.** If your program will be able to process very large input files (using interactive processing and in reasonable time), you will get extra credit: an additional **10 pts**.

Include all comments, including instructions about compiling and linking in a single file called **read.me**. Do not forget to include your name and KUID#. When you are ready to submit the project, send ALL necessary source files, makefile (if any), and the read.me file by e-mail to the TA. **Do not send object files, executable files, and test data files.** Time stamp of the e-mail will be used to decide late penalty, if applicable. Late projects will be accepted with 10% penalty per day up to five days.