COMS 3101 - Fall 2013
Perl Homework 5 (Extra)

- Extra HW grade can replace lowest grade.
- See submission instructions.

1. Data Processing Tool:

Write a module that implements some useful functionality for processing data in standard format (each row can be separated into fields using a delimiter). Typical data processing steps include reading/writing to files, removing examples (rows), removes features (columns), replacing missing features in some rows, changing format of features (e.g. from categorical to binary), choosing subset of the data, and reporting statistics on parts of the data.

a) The module (**DataSet.pm**) defines a class with several attributes (some of them not used in this assignment), and methods to set these attributes. It also provides methods to do various processing tasks (mentioned above). The DataSet object should be implemented using a hash reference to store the data.

Specifically, the DataSet class should contain the following features:

Attributes:

- **name** - name of dataset
- **dataTxt** – reference (ref) to array of strings which holds a row of text in each element
- **dataMtx** – ref to array of arrays (matrix) which holds the data
- **input** – ref to hash which stores the specifications (properties) of the input data as follows (key-value pairs):
    - "file" => name of input data file
    - "sep" => separator of fields in input file (e.g. "," or "\t")
    - "dfmt" => string specifying the input data format
- **output** – ref to hash which stores the specifications (properties) of the output (processed) data as follows (key-value pairs):

- "file" => name of output data file
- "sep" => separator of fields in output file (e.g. "," or "\t")
- "dfmt" => string specifying the output data format

Constructor:

- Creates the hash reference to store the object data. Initialize it with 'name' of dataset, input-file-name, and output-file-name, passed as arguments. By default, *dataTxt, dataMtx* should be references to empty arrays, and other properties can be set to values of your choice. The constructor takes arguments and should be called as:
  DataSet->new("UCI", "yeast.data", "yeast.data.txt");

Methods:

- **setInSep** – sets the input file separator to the argument passed
- **setOutSep** – sets the output file separator to the argument passed
- **printDetails** - prints the name, and each key-value pair of input & output, all on separate lines.

- **readDataTxt** – opens the file specified by *input{file}* (check for failure), reads the data from the file line by line, and stores the data in the array referred to by *dataTxt.*
- **readDataMtx** – opens the file specified by *input{file}* (check for failure), reads the data from the file (data is separated by *input{sep}*), and stores the data in the AoA referred to by *dataMtx*, where each line is separated into fields using the separator and stored as a row of the AoA.
- **writeData** – opens the file specified by *output{file}*, writes the data from the array ref *dataTxt* by default (or AoA ref *dataMtx* if *dataTxt* is empty), where the data is separated by output{*sep}*).

- **removeRow** – given a row number as argument, removes that row completely from the data.
- **removeRows** – given an array with row numbers as argument, removes all the rows completely from the data.
- **removeCol**– given a column number as argument, removes that column completely from the data.

- **exchangeCols** – given two column numbers as arguments, exchanges the columns (switch the order).

- **subPattern** – given a pair of strings as arguments, where the first string is the pattern in the data to be replaced by the second string (e.g. replace A by 1), does the substitution globally (for the entire data).
- **subPatterns** – given a hash as argument, where each key-value pair are old/new patterns, for each pair, does the substitution globally (for the entire data).

- **countCat** – given a column number as argument, returns a hash which counts the number of rows for each distinct category in the column (we assume that column values are either strings or integers). Hash key,value = distinct category, #rows with that category.
- **cat2bin** – given a column number as argument, for every row, replaces each categorical value with a set of binary values. For example, if the categorical values are {A;B;C} these are replaced with {1,0,0; 0,1,0; 0,0,1}. The replacement must be consistent for all rows (A is replaced by 1,0,0 everywhere). Observe that the number of columns may increase (which can affect future calls which process columns).

Notes:

1. By default all relevant functions above first act on the dataTxt array ref. Only if dataTxt refers to an empty array, then the same function is applied to the dataMtx AoA ref.
2. Although arrays in Perl are indexed from 0, you can still let the user index column/rows from 1 (this is more intuitive) by decrementing where necessary. However you must be consistent throughout.
3. Feel free to add your own subroutines (outside the class) if you find that this approach simplifies your code

b) Download a dataset from the UCI repository:
archive.ics.uci.edu/ml/

You can choose any dataset you wish (but specify in your comments how it was used). Each dataset has documentation provided and some are “cleaner” than others. Write a script **testdata.pl** to test your modules on the chosen dataset. The script should be similar in style to the **testbank.pl** script from HW4.