```lua
local help= [[
NB:
(c)2022 Tim Menzies, timm@ieee.org

OPTIONS:
  -k  --k  handle rare classes    = 1
  -m  --m  handle rare attributes = 2
  -p  --p  distance coefficient   = 2
  -w  --wait wait before classifying = 10

OPTIONS (other):
  -h  --help  show help     = false
  -g  --go    start-up goal = nothing
  -s  --seed  seed          = 10019
  -f  --file  file          = ../../data/auto93.csv]]
--        ._    _.  ._ _    _    _  ----------------------------------------------
--        | |  (_|  | | |  (/_  _>

local lib = require"lib"
local cli,csv,demos,is,normpdf = lib.cli, lib.csv, lib.demos, lib.is, lib.normpdf
local oo,push,read,rnd,str     = lib.oo, lib.push, lib.read, lib.rnd, lib.str

local THE={}
help:gsub(" [-][-]([^%s]+)[^\n]*%s([^%s]+)",function(key,x) THE[key] = read(x) end)

local NB,NUM,SYM,COLS,ROW,ROWS= is"NB",is"NUM",is"SYM",is"COLS",is"ROW",is"ROWS"
--         _    _   |        ._ _    ._   ------------------------------------------
--        (_   (_)  |  |_|  | | |   | |

function NUM.new(i)        i.n,i.mu,i.m2,i.mu = 0,0,0,0 end
function NUM.mid(i,p)      return rnd(i.mu,p) end
function NUM.like(i,x,...) return normpdf(x, i.mu, i.sd) end
function NUM.add(i,v,   d)
  if v=="?" then return v end
  i.n = i.n + 1
  d    = v - i.mu
  i.mu = i.mu + d/i.n
  i.m2 = i.m2 + d*(v - i.mu)
  i.sd = i.n<2 and 0 or (i.m2/(i.n-1))^0.5 end

function SYM.new(i)          i.n,i.syms,i.most,i.mode = 0,{},0,nil end
function SYM.mid(i,...)      return i.mode end
function SYM.like(i,x,prior) return ((i.syms[x] or 0)+THE.m*prior)/(i.n+THE.m) end
function SYM.add(i,v)
  if v=="?" then return v end
  i.n = i.n + 1
  i.syms[v] = (inc or 1) + (i.syms[v] or 0)
  if i.syms[v] > i.most then i.most,i.mode = i.syms[v],v end end
--         _    _   |        ._ _    ._    _  --------------------------------------
--        (_   (_)  |  |_|  | | |   | |   _>

local function usep(x)   return not x:find":$" end
local function nump(x)   return x:find"^[A-Z]" end
local function goalp(x)  return x:find"[!+-]$" end
local function klassp(x) return x:find"!$"     end

local function new(at,txt)
  txt = txt or ""
  local i = (nump(txt) and NUM or SYM)()
  i.txt, i.usep, i.at, i.w = txt, usep(txt), at or 0, txt:find"-$" and -1 or 1
  return i  end

function COLS.new(i,t,     col)
  i.all, i.xs, i.ys, i.names = {},{},{},t
  for at,x in pairs(t) do
    col = push(i.all, new(at,x))
    if col.usep  then
      if klassp(col.txt) then i.klass=col end
      push(goalp(col.txt) and i.ys or i.xs, col) end end end

 function COLS.add(i,t)
   for _,cols in pairs{i.xs,i.ys} do
     for _,col in pairs(cols) do col:add(t[col.at]) end end
   return t end
--        ._    _          ---------------------------------------------------------
--        |    (_)  \/\/

function ROW.new(i,of,cells) i.of,i.cells,i.evaled=of,cells,false end
function ROW.klass(i)        return i.cells[i.of.cols.klass.at] end
--        ._    _        _  ------------------------------------------------------
--        |    (_)  \/\/  _>

local function load(src, fun)
  if type(src)~="string" then for _,t in pairs(src) do fun(t) end
                         else for   t in csv(src)   do fun(t) end end end

function ROWS.new(i,t) i.cols=COLS(t); i.rows={} end
function ROWS.add(i,t)
  t=t.cells and t or ROW(i,t)
  i.cols:add(t.cells)
  return push(i.rows, t) end
function ROWS.mid(i, cols, p,     t)
  t={};for _,col in pairs(cols or i.cols.ys) do t[col.txt]=col:mid(p) end;return t end

function ROWS.clone(i,t,   j)
  j= ROWS(i.cols.names);for _,row in pairs(t or {}) do j:add(row) end; return j end

function ROWS.like(i,t, nklasses, nrows,    prior,like,inc,has)
  prior = (i.n + THE.k) / (nrows + THE.k * nklasses)
  like  = math.log(prior)
  for _,col in pairs(i.cols.xs) do
    x = t[col.at]
    if x and x ~= "?" then
      like = like + math.log(col:like(x,prior)) end end
  return like end

function NB.new(i,src,   all,one,kl)
  i.all, i.one  = nil, {}
  load(src, function(t) if   i.all
                        then kl        = i.all:add(t):klass()
                             i.one[kl] = i.one[kl] or i.all:clone()
                             i.one[kl]:add(t)
                        else i.all = ROWS(t) end end) end
--       _|_   _    _  _|_   _  ------------------------------------------------
--        |_  (/_  _>   |_  _>

local no,go = {},{}
function go.the()  return type(THE.p)=="number" and THE.p==2 end
function go.num(n) n=NUM(); for i=1,100 do add(n,i) end; return n.mu==50.5 end

function go.sym(s)
  s=SYM(); add(s,{"a","a","a","a","b","b","c"}); return s.mode=="a" end

function go.csv(    n,s)
  n,s=0,0; for row in csv(THE.file)  do n=n+1; if n>1 then s=s+row[1] end end
  return rnd(s/n,3) == 5.441  end

function go.rows(    rows)
  load(THE.file,function(t) if rows then rows:add(t)  else rows=ROWS(t) end end)
  return rnd(rows.cols.ys[1].sd,0)==847 end

function go.nb()
  return 268 == #NB("../../data/diabetes.csv").one.positive.rows  end
--        _  _|_   _.  ._  _|_ -------------------------------------------------
--       _>   |_  (_|  |    |_

if    pcall(debug.getlocal, 4, 1)
then  return {ROW=ROW, ROWS=ROWS, NUM=NUM, SYM=SYM, THE=THE,lib=lib}
else  THE = cli(THE,help)
      demos(THE,go) end
```