```lua
--  __                                          ___
-- /\ \__                                      /\_ \
-- \ \ ,_\  /\_\    ___    __  __             \//\ \    __  __     __
--  \ \ \/  \/\ \ /' _ `\ /\ \/\ \              \ \ \  /\ \/\ \  /'__`\
--   \ \ \_  \ \ \/\ \/\ \\ \ \_\ \      __      \_\ \_\ \ \_\ \/\ \L\.\_
--    \ \__\  \ \_\ \_\ \_\\/`____ \    /\_\     /\____\\ \____/\ \__/.\_\
--     \/__/   \/_/\/_/\/_/ `/___/> \   \/_/     \/____/ \/___/  \/__/\/_/
--                             /\___/
--                             \/__/
--
local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
local THE,help= {},[[
TINY:
(c)2022 Tim Menzies, timm@ieee.org

OPTIONS:
  --bins  -b  bins                    = 10
  --k     -k  handle rare classes     = 1
  --m     -m  handle rare attributes  = 2
  --p     -p  distance coefficient    = 2

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

local big,cli,copy,csv,demos, fmt,fmtp,map,normpdf
local oo,pop,push,rand,read,rnd,shuffle,splice,str
local function is(name,    t,new,x)
  function new(kl,...) x=setmetatable({},kl); kl.new(x,...); return x end
  t = {__tostring=str, is=name}; t.__index=t
  return setmetatable(t, {__call=new}) end

local ROW, ROWS, NUM, SYM = is"ROW", is"ROWS", is"NUM", is"SYM"

--------------------------------------------------------------------------------
--        _
--   __  | |  __ _   ___  ___  ___   ___
--  / _| | | / _` | (_-< (_-< / -_) (_-<
--  \__| |_| \__,_| /__/ /__/ \___| /__/
--
function SYM.new( i, at,txt)
  i.n, i.at, i.txt = 0, at or 0, txt or ""
  i.has, i.most, i.mode = {}, 0, nil end

function SYM.add(i,x,inc)
  if x=="?" then return x end
  i.n = i.n + 1
  i.has[x] = (inc or 1) + (i.has[x] or 0)
  if i.has[x] > i.most then i.most,i.mode = i.has[x],x end end

function SYM.like(i,x,prior)
  return ((i.has[x] or 0) + THE.m*prior) / (i.n + THE.m) end

function SYM.mid(i) return i.mode end
function SYM.dist(i,x,y) return x=="?" and y=="?" and 1 or x==y and 0 or 1 end

-------------------------------------------------------------------------------
function ROW.new(i,of,cells) i.of, i.cells, i.evaluated = of, cells, true end

function ROW.__lt(i,j,        n,s1,s2,v1,v2)
  i.evaluated = true
  j.evaluated = true
  s1, s2, n = 0, 0, #i.of.ys
  for _,col in pairs(i.of.ys) do
    v1,v2 = col:norm(i.cells[col.at]), col:norm(j.cells[col.at])
    s1    = s1 - 2.7183^(col.w * (v1 - v2) / n)
    s2    = s2 - 2.7183^(col.w * (v2 - v1) / n) end
  return s1/n < s2/n end

function ROW.klass(i) return i.cells[i.of.klass.at] end

function ROW:__sub(other,    cols,d,inc)
  d, cols = 0, self.of.xs
  for _,col in pairs(cols) do
    inc = col:dist(self.cells[col.pos], other.cells[col.pos])
    d   = d + inc^THE.p end
  return (d / #cols) ^ (1/THE.p) end

-------------------------------------------------------------------------------
function NUM.new(i, at,txt)
  i.n, i.at, i.txt = 0, at or 0, txt or ""
  i.w = i.txt:find"-$" and -1 or 1
  i.mu, i.m2, i.sd, i.lo, i.hi = 0, 0, 0, big, -big end

function NUM.add(i,x,     d)
  if x=="?" then return x end
  i.n  = i.n+1
  d    = x-i.mu
  i.mu = i.mu + d/i.n
  i.m2 = i.m2 + d*(x - i.mu)
  i.sd = i.n<2 and 0 or (i.m2/(i.n-1))^0.5
  i.lo = math.min(x,i.lo)
  i.hi = math.max(x,i.hi) end

function NUM.like(i,x,...) return  normpdf(x, i.mu, i.sd) end

function NUM.mid(i,p) return rnd(i.mu,p) end

function NUM.norm(i,x)
  return i.hi - i.lo < 1E-9 and 0 or (x-i.lo)/(i.hi - i.lo + 1/big) end

function NUM.dist(i,x,y)
  if     x=="?" and y=="?" then return 1 end
  if     x=="?"            then y = self:norm(y); x = y<.5 and 1 or 0
  elseif y=="?"            then x = self:norm(x); y = x<.5 and 1 or 0
  else x,y = self:norm(x), self:norm(y) end
  return math.abs(x - y) end

-------------------------------------------------------------------------------
function ROWS.new(i,src)
  i.has={}; i.cols={}; i.xs={}; i.ys={}; i.names={}
  if type(src)=="string" then for    row in csv(  src) do i:add(row) end
                         else for _,row in pairs(src) do i:add(row) end end end

function ROWS.add(i,row,   col)
  if #i.cols==0 then
    i.names = row
    for at,s in pairs(row) do
      col = push(i.cols, (s:find"^[A-Z]" and NUM or SYM)(at,s))
      col.goal = s:find"[!+-]$"
      if not s:find":$" then
        if s:find"!$" then i.klass = col end
        push(col.goal and i.ys or i.xs, col) end end
  else
    row = push(i.has, row.cells and row or ROW(i,row))
    for _,col in pairs(i.cols) do col:add(row.cells[col.at]) end end end

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


function ROWS.like(i,t,klasses, all,     prior,like,x)
  prior = (#i.has + THE.k) / (all + THE.k * klasses)
  like  = math.log(prior)
  t     = t.cells and t.cells or t
  for _,col in pairs(i.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 ROWS.mid(i,p,   u)
  u={}; for _,col in pairs(i.ys) do u[col.txt] = col:mid() end; return u end

-------------------------------------------------------------------------------
--    __                      _     _
--   / _|  _  _   _ _    __  | |_  (_)  ___   _ _    ___
--  |  _| | || | | ' \  / _| |  _| | | / _ \ | ' \  (_-<
--  |_|    \_,_| |_||_| \__|  \__| |_| \___/ |_||_| /__/
--
big  = math.huge
fmt  = string.format
fmtp = function(...) print(fmt(...)) end
rand = math.random

function cli(t)
  for key,x in pairs(t) do
    x=str(x)
    for n,flag in ipairs(arg) do
      if flag==("-"..key:sub(1,1)) or flag==("--"..key) then
         x= x=="false" and"true" or x=="true" and"false" or arg[n+1] end end
    t[key] = read(x) end
  return t end

function csv(csvfile)
  csvfile = io.input(csvfile)
  return function(s, t)
    s=io.read()
    if not s then io.close(csvfile) else
      t={}; for x in s:gmatch("([^,]+)") do t[1+#t] = read(x) end
      return t end end end

function demos(THE,go)
  local fails,backup = 0,{}
  for k,v in pairs(THE) do backup[k]=v end
  for txt,fun in pairs(go[THE.go] and {go[THE.go]} or go) do
    for k,v in pairs(backup) do THE[k]=v end
    math.randomseed(THE.seed)
    io.write(".")
    local result = fun()
    if result ~= true then
      fails = fails + 1
      print("--Error",s,status) end end
  for k,v in pairs(_ENV) do  if not b4[k] then print("?",k,type(v)) end end
  os.exit(fails)  end

function copy(t,   u)
  if type(t) ~= "table" then return t end
  u={};for k,v in pairs(t) do u[copy(k)]=copy(v) end
  return setmetatable(u, getmetatable(t)) end

function map(t,f,  u)
  u={}; for k,v in pairs(t) do u[1+#u]=(f and f(v) or v) end return u end

function normpdf(x, mu, sd,      denom,nom)
 return sd==0 and (x==mu and 1 or 0 ) or
   math.exp(-1*(x - mu)^2/(2*sd^2)) * 1 / (sd * ((2*math.pi)^0.5)) end

oo = function(i) print(str(i)) end

function pop(t)  return table.remove(t) end

function push(t,x) t[1+#t] = x ; return x end

function read(str)
  str = str:match"^%s*(.-)%s*$"
  if str=="true" then return true elseif str=="false" then return false end
  return math.tointeger(str) or tonumber(str) or str  end

function rnd(n, p) local m=10^(p or 2); return math.floor(n*m+0.5)/m  end

function shuffle(t,    j)
  for i = #t, 2, -1 do j=math.random(i); t[i], t[j] = t[j], t[i]; end;
  return t end

function splice( t, i, j, k,    u)
  u={};for n=(i or 1)//1,(j or #t)//1,(k or 1)//1 do u[1+#u]=t[n] end;return u end

function str(i,    j)
  if type(i)~="table" then return tostring(i) end
  if #i> 0 then j= map(i,tostring)
  else j={}; for k,v in pairs(i) do j[1+#j] = string.format(":%s %s",k,v) end
       table.sort(j) end
  return (i.is or "").."{"..table.concat(j," ").."}" end

-------------------------------------------------------------------------------
--      _
--   __| |  ___   _ __    ___   ___
--  / _` | / -_) | '  \  / _ \ (_-<
--  \__,_| \___| |_|_|_| \___/ /__/
--
local go,no = {},{}

function go.num(  n)
  n=NUM(); for i=1,100 do n:add(i) end; oo(n); return true end

function go.sym(  s)
  s=SYM(); for i=1,100 do s:add(i) end; oo(s); return true end

function go.read(  rows,n)
  rows = ROWS(THE.file)
  table.sort(rows.has)
  n= #rows.has
  print("all",  str(rows:mid()))
  print("best", str(rows:clone(splice(rows.has,1,30)):mid()))
  print("rest", str(rows:clone(splice(rows.has,n-30)):mid()))
  return true end

function go.smo(rows,n,    all,kl,it,most,tmp)
  rows = ROWS("../../data/auto93.csv")
  table.sort(rows.has)
  for n,row in pairs(rows.has) do row.rank = 100*n/#rows.has//1 end
  all  = shuffle(map(rows.has))
  local seen = {pop(all), pop(all), pop(all)}
  while #seen < 20 and #all > 10 do
    local n,bests,rests,maybe
    table.sort(seen)
    print""
    for _,row in pairs(seen) do io.write(row.rank," ") end; print("")
    n     = math.floor(.5 + math.log(#seen,2))
    bests = rows:clone(splice(seen,1,n))
    rests = rows:clone(splice(seen,n+1))
    print(#seen, n,       "all", str(rows:mid(2)))
    print(#seen, n,       "bests", str(bests:mid(2)))
    print(#seen, #seen-n,"rests", str(rests:mid(2)))
    good = function(row1,row2)
             return bests:like(row1,2,#all) >  bests:like(row2,2,#all) end
    maybe = function(row1,row2,  b1,b2,r1,r2)
              b1= bests:like(row1,2,#all); r1= rests:like(row1,2,#all)
              b2= bests:like(row2,2,#all); r2= rests:like(row2,2,#all)
              --return b1^2/(b1+r1) > b2^2/(b2+r2) end
              return b1/math.abs(b1-r1)/b1 < b2/math.abs(b2-r2) end
    table.sort(all, maybe)
    push(seen,pop(all))
    table.sort(all, good)
    all = splice(all,1,.66*#all//1)
  end end

-------------------------------------------------------------------------------
--                _
--   _ __    __ _  (_)  _ _
--  | '  \  / _` | | | | ' \
--  |_|_|_| \__,_| |_| |_||_|

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

if   pcall(debug.getlocal, 4, 1) then
  return {ROW=ROW, ROWS=ROWS, NUM=NUM, SYM=SYM, THE=THE}
else
  THE = cli(THE)
  if THE.help then os.exit(print(help:gsub("[%u][%u%d]+","\27[1;31m%1\27[0m"))) end
  demos(THE,go) end
```