```lua
#!/usr/bin/env lua
-- vim : filetype=lua ts=2 sw=2 et :
local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
local function cli(flag,x)
  for n,word in ipairs(arg) do if word==flag then
    return x==false and true or tonumber(arg[n+1]) or arg[n+1] end end
  return x end

local THE={
  seed    = cli("-s", 10019),
  debug   = cli("-d", ""),
  epsilon = cli("-e", .35),
  file    = cli("-f", "../../data/auto93.csv"),
  p       = cli("-p", 2),
  todo    = cli("-t", "")
}
-----------------------------------------------------------------------------------
local function sort(t,f)    table.sort(t,f);   return t end
local function keys(t,   u) u={};for k,_ in pairs(t) do u[1+#u]=k     end; return u end
local function map(t,f,  u) u={};for k,v in pairs(t) do u[1+#u] =f(v) end; return u end

local function csv(file,   x,line)
  function line(x,  t)
    t={}; for y in x:gmatch"([^,]+)" do t[1+#t]=tonumber(y) or y end; return t end
   file = io.input(file)
   return function() x=io.read(); if x then return line(x) else io.close(file) end end end

local function o(t,   u,key)
  function key(k) return string.format(":%s %s", k, o(t[k])) end
  if type(t) ~= "table" then return tostring(t) end
  return "{"..table.concat(#t>0 and map(t,o) or map(sort(keys(t)),key)," ")..."}" end

local function rand(lo,hi)
  THE.seed = (16807 * THE.seed) % 2147483647
  return (lo or 0) + ((hi or 1) - (lo or 0)) * THE.seed / 2147483647 end

local function randi(lo,hi) return math.floor(0.5 + rand(lo,hi)) end
local function any(t) return t[randi(1,#t)] end
local function shuffle(t,   j)
  for i=#t,2,-1 do j=randi(1,i); t[i],t[j]=t[j],t[i] end; return t end

local function norm(i,x)
  return math.abs(i.lo - i.hi) < 1E-32 and 0 or (x - i.lo) / (i.hi - i.lo) end

local function new(x,y) x.__index=x; return setmetatable(y,x) end
-----------------------------------------------------------------------------------
local EG={}
function EG.new(t) return new(EG, {has=t}) end

function EG.better(eg1,eg2,smpl,    e,n,a,b,s1,s2)
  s1,s2,e,n = 0,0, 2.71828,#smpl.nys
  for col,w in pairs(smpl.ys) do
    num = smpl.num[col]
    a  = norm(num, eg1.has[col])
    b  = norm(num, eg2.has[col])
    s1 = s1 - e^(num.w * (a-b)/n)
    s2 = s2 - e^(num.w * (b-a)/n) end
  return s1/n < s2/n end

function EG.dist(i,j,smpl,   a,b,d,n,inc,dist1)
  function dist1(num,a,b)
    if   num
    then if     a=="?" then b=norm(num,b); a=b>.5 and 0 or 1
         elseif b=="?" then a=norm(num,a); b=a>.5 and 0 or 1
         else   a,b = norm(num,a), norm(num,b) end
         return math.abs(a-b)
    else return a==b and 0 or 1
  end
  d, n = 0, 1E-31
  for col,_ in pairs(smpl.xs) do
    a,b = i.has[col], j.has[col]
    inc = a=="?" and b=="?" and 1 or dist1(smpl.num[col],a,b)
    d   = d + inc^THE.p
    n   = n + 1 end
  return (d/n)^(1/THE.p) end
-----------------------------------------------------------------------------------
local SAMPLE={}
function SAMPLE.new()
  return new(SAMPLE, {head=nil,w={},egs={},num={},sym={},xs={},ys={},nys=0}) end

function SAMPLE.add(i,eg)
  local nump,goalp,head,num1,data,skip
  function skip(x)  return x:find":" end
  function nump(x)  return not skip(x) and x:find"^[A-Z].*" end
  function goalp(x) return not skip(x) and (x:find"-" or x:find"+") end
  function top(names)
    for n,s in pairs(names) do
      if nump(s)  then i.num[n] = {txt=s, at=n,lo=math.huge, hi=-math.huge,
                                   w=s:find"-" and -1 or 1}
                  else i.sym[n] = {txt=s} end
      if goalp(s) then i.ys[n]  = s:find"-" and -1 or 1; i.nys = 1+i.nys
                  else i.xs[n]  = n end end end
  function num1(num,x)
    if x=="?" then return end
    if x<num.lo then num.lo=x elseif x>num.hi then num.hi=x end end
  function data(eg)
    for n,num in pairs(i.num) do num1(num, eg[n]) end end
  if not i.head then i.head = eg;                  top(eg)
                else i.egs[1+#i.egs]=Row.new(eg); data(eg) end
  return i end

function SAMPLE.far(i,eg1,egs,    dist,tmp)
  dist= function(eg2) return {eg2,eg1:dist(eg2,i)} end
  tmp = sort(map(egs, dist), function(a,b) return a[2] < b[2] end)
  return tmp[#tmp*THE.far//1] end

function SAMPLE.label(i, egs,n,one,    egs,n,one,two,a,b,c,lo,hi)
  egs,n = egs or i.egs, 1
  for _,eg in pairs(egs) do eg.klass= n end
  if #egs < 2*#i.egs^THE.epsilon then return end
  one   = one or i:far(any(egs),egs)
  two,c = i:far(one, egs)
  for _eg in pairs(egs) do
    a = eg:dist(one, i)
    b = eg:dist(two, i)
    eg.x = (a^2 + c^2 - b^2)/(2*c)
  end
  lo,hi = {},{}
  for n,eg in pairs(sort(egs,function(a,b) return a.x < b.x end)) do
    table.insert(n <= .5*#egs and lo or hi, eg) end
  i:label(one:better(two,i) and lo or hi, n*2, two) end
-----------------------------------------------------------------------------------
for k,v in pairs(_ENV) do if not b4[k] then print("Rogue?",k,type(v)) end end
```