i Gimy.lwa

local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
local THE, help= {}, [[

(c)2022 Tim Menzies, timm@ieee.org

```
--bins -b bins
--k -k handl
--m -m handl
                  -b bins = 10
-k handle rare classes = 1
-m handle rare attributes = 2
-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 lib={}
lib.lo= | lib-| |
lib.big = math.huge
lib.fmt = string.format
lib.fmtp = function(...) print(fmt(...)) end
lib.rand = math.random
function lib.cli(t)
  for key,x in pairs(t) do
  x = lib.str(x)
  x = lib.str(x)
for n,flag in ipairs(arg) do
if flag==("-".key:sub(1,1)) or flag==("--".key) then
    x = x=="false" and "rue" or x=="true" and "false" or arg[n+1] end end
t[key] = lib.read(x) end
return t end
function lib.csv(csvfile)
   csvfile = io.input(csvfile)
return function(s, t)
       seturn tunector(s, c)
s=io.read()
if not s then io.close(csvfile) else
t=(); for x in s:gmatch("([^]+)") do t[1+#t] = lib.read(x) end
           return t end end end
function lib.demos(THE,go)
   unction lib.demos(THE,go)
local fails,backup = 0,{}
for k,v in pairs(THE) do backup[k]=v end
for txt,todo 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 = todo()
if result == true then
fails = fails + 1
print(""-Enor",txt,status) end end
for k,v in pairs(FRV) do if pot h4[k] then print("?",k,type
    for k,v in pairs (_ENV) do if not b4[k] then print("?",k,type(v)) end end os.exit(fails) end
 \begin{array}{lll} \textbf{function lib.copy}(t, & u) \\ \textbf{if type}(t) & \sim & \text{"table" then return t end} \\ \textbf{u=(); for } k, v \text{ in pairs}(t) \text{ do } u(\text{lib.copy}(k)) \text{=lib.copy}(v) \text{ end} \\ \textbf{return setmetatable}(u, \text{ getmetatable}(t)) \text{ end} \\ \end{array} 
function lib.is(name, t,new,x)
function new(kl,...) x=setmetatable({}(},kl); kl.new(x,...); return x end
t = (_tostring=lib.str, is=name); t.__index=t
return setmetatable(t, (_call=new)) end
function lib.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
function lib.oo(i) print(lib.str(i)) end
function lib.pop(t) return table.remove(t) end
function lib.push(t,x) t[1+#t] = x ; return x end
function lib.read(str) str = str:match^\%s^(-)\%s^\$" if str==\"function" if str==\"function" return true elseif str==\"false\" then return false end
    return math.tointeger(str) or tonumber(str) or str end
function lib.rnd(n, p) local m=10^(p or 2); return math.floor(n*m+0.5)/m end
function lib.shuffle(t, j) for i = tt, 2, -1 do j=math.random(i); t[i], t[j] = t[j], t[i]; end; return t end
```

```
s local big, cli, copy, csv, demos = lib.big, lib.cli, lib.copy, lib.csv, lib.demos s local fmt, is, map, normpdf, oo = lib.fmt, lib.is, lib.map, lib.normpdf, lib.oo, lib. rand s local read, rnd, shuffle, splice = lib.read, lib.rnd, lib.shuffle, lib.splice
                                                        = lib.str
     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
        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,
   i.evaluated = true
   j.evaluated = true
        j.evaluatea = tro, 0, #i.of.ys
for _,col in pairs(i.of.ys) do
v1,v2 = col:norm(i.celis[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)
       function NUM.new(i, at,txt)
i.n, i.at, i.txt = 0, at or 0, txt or ""
i.w = i.txt:find"-5" 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</pre>
        i.lo = math.min(x,i.lo)
i.hi = math.max(x,i.hi) end
     function NUM.like(i,x,...) return normodf(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
     else x,y = self:norm(x), self:norm(y) end
return math.abs(x - y) end
     function ROWS.new(i,src)
        inha={|; 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)
       unction ROWS.add(i,row, col)
if #i.coles=0 them
   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*"s* then
        if s:find*"s* then i.klass = col end
        push(col.goal and i.ys or i.xs, col) end end
   alse
            see 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
223 function ROWS.mid(i,p, u)
224 u={}; for _,col in pairs(i.ys) do u[col.txt] = col:mid() end; return u 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.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("Best", str(rows:clone(splice(rows.has,1,30)):mid()))
    print("best", str(rows:clone(splice(rows.has,n-30)):mid()))
    print("best", str(rows:clone(splice(rows.has,n-30)):mid()))
    print("best", str(rows:clone(splice(rows.has,n-30)):mid()))
    print("rest", str(rows:clone(splice(rows.has,n-30)):mid()))
    print("set", str(rows:clone(splice(rows.has,n-30)):mid()))
    return true end

function go.smo(rows,n, all,kl,it,most,tmp)
    rows = ROWS("././dan/auu09/s.cv")
    table.sort(rows.has)
    for n,row in pairs(rows.has)
    do local sen = {pop(all), pop(all), pop(all)}
    while sen( < 20 and fall > 10 do
    local seen = {pop(all), pop(all), pop(all)}
    while seen < 20 and fall > 10 do
    local n,bests,rests,maybe,good
    tunction
    tunction go.smo(rows.has)
    return true end

    print("seen, n, "all", str(rows:mid(2)))
    print(#seen, n, "all", str(rows:mid(2)))
    print(#seen, n, "ens", str(rests:mid(2)))
    print(#seen, n, "ens",
```

```
help:gsub("[-||-||(1/%s|+)|/m]|*%s[/%s|+)", function(key,x) THE[key]=read(x) end)

if pcall(debug.getlocal, 4, 1) then
return (ROW=ROM, ROWS=ROWS, NUM=NUM, SYM=SYM, THE=THE,lib=lib)

if EHE = cli(THE)

if THE.help then os.exit(print(help:gsub("[%u][%u\%d]+","\27[1;3lm\%l\27[0m"))) end

demos(THE,go) end
```