```lua
local b4={}; for k,v in pairs(_ENV) do b4[k]=k end
local function cli(flag,x)
  for n,word in ipairs(arg) do if word==flag then
    x = x and (tonumber(arg[n+1]) or arg[n+1]) or true end end
  return x end

local the= {BEST=  cli("-b", .1),
            FILE=  cli("-f","../../data/auto93.csv"),
            SEED=  cli("-s",  10119),
            TINY = cli("-t", .1),
            TRIVIAL = cli("-T", .35)}

------------------------------------
local same,push,sort,map,csv,norm,ent,copy
function same(x)    return x end
function push(t,x)  t[1+#t]=x; return x end
function sort(t,f)  table.sort(t,f);   return t end
function map(t,f,u) u={};for k,v in pairs(t) do push(u,f(k,v)) end; return u end
function norm(lo,hi,x) return math.abs(lo-hi)<1E-32 and 0 or (x-lo)/(hi-lo) end
function copy(t,  u) u={}; for k,v in pairs(t) do u[k]=v end; return u end

function csv(file,   x)
  file = io.input(file)
  x    = io.read()
  return function(   t,tmp)
    if x then
      t={}
      for y in x:gsub("[\t ]*",""):gmatch"([^,]+)" do push(t,tonumber(y) or y) end
      x = io.read()
      if #t>0 then return t end
    else io.close(file) end end end

local shout,out
function shout(x) print(out(x)) end
function out(t,    u,key,keys,value)
  function keys(t,u)
    u={}; for k,_ in pairs(t) do if tostring(k):sub(1,1)~="_" then u[k]=k end end
    return sort(u) end
  function key(_,k)   return string.format(":%s %s", k, out(t[k])) end
  function value(_,v) return out(v) end
  if type(t) ~= "table" then return tostring(t) end
  u = #t>0 and map(t, value) or map(keys(t), key)
  return "{"..table.concat(u," ").."}" end

function inc3(t, x, y, z, n)
    if not t[x] then t[x] = {} end; t = t[x]
    if not t[y] then t[y] = {} end; t = t[y]
    t[z] = (t[z] or 0) + (n or 10)
end

function get3(t, x, y, z, n)
    if not t[x] then t[x] = {} end; t = t[x]
    if not t[y] then t[y] = {} end; t = t[y]
    return t[z] or 0
end

local Seed,randi,rand,ent
Seed = the.SEED
function randi(lo,hi) return math.floor(0.5 + rand(lo,hi)) end
function rand(lo,hi)
  lo, hi = lo or 0, hi or 1
  Seed = (16807 * Seed) % 2147483647
  return lo + (hi-lo) * Seed / 2147483647 end

function ent(t,    n,e)
  n=0; for _,n1 in pairs(t) do n = n + n1 end
  e=0; for _,n1 in pairs(t) do e = e - n1/n*math.log(n1/n,2) end
  return e end

--------------------------
local slurp,sample,ordered,clone
function slurp(out)
  for eg in csv(the.FILE) do out=sample(eg,out) end
  return out end

function clone(i, inits,     out)
  out = sample(i.heads)
  for _,eg in pairs(inits or {}) do out = sample(eg,out) end
  return out end

function sample(eg,i)
  local numeric,independent,dependent,head,data,datum
  i = i or {n=0,xs={},nys=0,ys={},lo={},hi={},w={},egs={},heads={}}
  function head(n,x)
    function numeric()     i.lo[n]= math.huge; i.hi[n]= -i.lo[n] end
    function independent() i.xs[n]= x end
    function dependent()
      i.w[n]  = x:find"-" and -1 or 1
      i.ys[n] = x
      i.nys   = i.nys+1 end
    if not x:find":" then
      if x:match"^[A-Z]" then numeric() end
      if x:find"-" or x:find"+" then dependent() else independent() end end
    return x end
  function data(eg) return {raw=eg, cooked=copy(eg)} end
  function datum(n,x)
    if x ~= "?" then
      if i.lo[n] then
        i.lo[n] = math.min(i.lo[n],x)
        i.hi[n] = math.max(i.hi[n],x) end end
    return x end
  eg = eg.raw and eg.raw or eg
  if #i.heads==0 then i.heads=map(eg,head) else push(i.egs,data(map(eg,datum))) end
  i.n = i.n + 1
  return i end

function ordered(i)
  local function best(eg1,eg2,     a,b,s1,s2)
    s1,s2=0,0
    for n,_ in pairs(i.ys) do
      a  = norm(i.lo[n], i.hi[n], eg1.raw[n])
      b  = norm(i.lo[n], i.hi[n], eg2.raw[n])
      s1 = s1 - 2.71828^(i.w[n] * (a-b)/i.nys)
      s2 = s2 - 2.71828^(i.w[n] * (b-a)/i.nys) end
    return s1/i.nys < s2/i.nys end
  for j,eg in pairs(sort(i.egs,best)) do
    if j < the.BEST*#i.egs then eg.klass=true else eg.klass=false end end
  return i end

local discretize,div
function discretize(i,          bin,xys,p,bins,divs)
  function bin(z,divs)
    if z=="?" then return "?" end
    for n,x in pairs(divs) do if x.lo<= z and z<= x.hi then return string.char(96+n) end end
  end ------------------------
  for col,_ in pairs(i.xs) do
    if i.lo[col] then
      xys={}
      for _,eg in pairs(i.egs) do
        local x=eg.raw[col]
        if x~="?" then push(xys, {x=x,  y=eg.klass}) end end
      xys  = sort(xys, function(a,b) return a.x < b.x end)
      p    = function (z) return xys[z*#xys//10].x end
      divs = div(xys, the.TINY*#xys, the.TRIVIAL*math.abs(p(.9) - p(.1))/2.56)
      for _,eg in pairs(i.egs) do
        eg.cooked[col]= bin(eg.raw[col], divs) end end end
  return i end

function div(xys,tiny,trivial,     one,all,merged,merge)
  function merged(a,b,an,bn,      c)
    c={}
    for x,v in pairs(a) do c[x] = v end
    for x,v in pairs(b) do c[x] = v + (c[x] or 0) end
    if ent(c)*.99 <= (an*ent(a) + bn*ent(b))/(an+bn) then return c end
  end ------------------------
  function merge(b4)
    local j,tmp = 0,{}
    while j < #b4 do
      j = j + 1
```

```lua
      local now, after = b4[j], b4[j+1]
      if after then
        local simpler = merged(now.has,after.has, now.n,after.n)
        if simpler then
          now = {lo=now.lo, hi=after.hi, n=now.n+after.n, has=simpler}
          j = j + 1 end end
      push(tmp,now) end
    return #tmp==#b4 and b4 or merge(tmp) -- recurse until nothing merged
  end ------------------------
  one = {lo=xys[1].x, hi=xys[1].x, n=0, has={}}
  all = {one}
  for j,xy in pairs(xys) do
    local x,y = xy.x, xy.y
    if  j< #xys-tiny and x~= xys[j+1].x and one.n> tiny and one.hi-one.lo> trivial
    then  one = push(all, {lo=one.hi, hi=x, n=0, has={}})
    end
    one.n  = 1 + one.n
    one.hi = x
    one.has[y] = 1 + (one.has[y] or 0); end
  return merge(all) end

local s=discretize(ordered(slurp()))

for k,v in pairs(_ENV) do if not b4[k] then print("?rogue: ",k,type(v)) end end
```