```
------------------------------------------------------------------------------
---   __                __                    __                    __
---  /\ \              /\ \                  /\ \                  /\ \
---  \ \ \____   _ __   \ \ \/'\       ___    \ \ \____     __      \_\ \
---   \ \ '__`\ /\`'__\ \ \ , <     /' _ `\  \ \ '__`\   /'__`\    /'_` \
---    \ \ \L\ \\ \ \/   \ \ \\`\   /\ \/\ \  \ \ \L\ \/\ \L\.\_ /\ \L\ \
---     \ \_,__/ \ \_\    \ \_\ \_\ \ \_\ \_\  \ \_,__/\ \__/.\_\\ \___,_\
---      \/___/   \/_/     \/_/\/_/  \/_/\/_/   \/___/  \/__/\/_/ \/__,_ /

---     .-------.
---     | Ba    | Bad <----.  planning= (better - bad)
---     |    56 |          |  monitor = (bad - better)
---     .-------.------.   |
---             | B    |   v
---             |    5 | Better
---             .------.
---
------------------------------------------------------------------------------

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

  -bins  -b   number of bins              = 16
  -cohen -c   cohen                       = .35
  -file  -f   file name                   = ../etc/data/breastcancer.csv
  -goal  -g   goal                        = recurrence-events
  -K     -K   manage low class counts     = 1
  -M     -M   manage low evidence counts = 2
  -seed  -S   seed                        = 10019
  -todo  -t   start up action             = nothing
  -wait  -w   wait                        = 10
]]

local max,min,ent,per
local push,map,sort,up1,upx,down1,slots,up1,down1
local words,thing, things, lines
local cli
local fmt,o,oo
local inc,inc2,inc3,has,has2,has3
local rogues
local classify,test,train,score,nb1,nb2,abcd
local bins,nb3
local eg,the,ako={},{},{}

---     _ _ |    _ _  _   _|_    _  _  _
---    (_(_)||_|| | || |   |  \/|_)(/__\
---                           /  |

local ako={}
ako.num    = function(x) return x:find"^[A-Z]" end
ako.goal   = function(x) return x:find"[-+!]"  end
ako.klass  = function(x) return x:find"!$"     end
ako.ignore = function(x) return x:find":$"     end
ako.less   = function(x) return x:find"-$"     end

------------------------------------------------------------------------------
-- BSD 2-Clause License
-- Copyright (c) 2022, Tim Menzies
--
-- Redistribution and use in source and binary forms, with or without
-- modification, are permitted provided that the following conditions are met:

-- 1. Redistributions of source code must retain the above copyright notice,this
--    list of conditions and the following disclaimer.
--
-- 2. Redistributions in binary form must reproduce the above copyright notice,
--    this list of conditions and the following disclaimer in the documentation
--    and/or other materials provided with the distribution.

-- THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
-- AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
-- IMPLIED WARRANTIES OF MERCHANTABILITY & FITNESS FOR A PARTICULAR PURPOSE ARE
-- DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
-- FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
-- DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
-- SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
-- CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
-- OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
-- OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

```

```
------------------------------------------------------------------------------
---    ___  ____ ____ _ ____
---    |__] |__| [__  | |
---    |__] |  | ___] | |___

function classify(i,t)
  local hi,out = -1
  for h,_ in pairs(i.h) do
    local prior = ((i.h[h] or 0) + the.K)/(i.n + the.K*i.nh)
    local l = prior
    for col,x in pairs(t) do
      if x ~= "?" and col ~= #t then
        l=l*(has3(i.e,col,x,h) + the.M*prior)/((i.h[h] or 0) + the.M) end end
    if l>hi then hi,out=l,h end end
  return out end

function test(i,t)
  if i.n > i.wait then push(i.log,{want=t[#t], got=classify(i,t)}) end  end

function train(i,t)
  local more, kl = false, t[#t]
  for col,x in pairs(t) do
    if x ~="?" then
      more = true
      inc3(i.e, col, x, kl)
      if col ~= #t then
        inc2(kl==the.goal and i.best or i.rest, col,x) end end end
  if more then
    i.n = i.n + 1
    if not i.h[kl] then i.nh = i.nh + 1 end
    inc(i.h, kl)
    if kl==the.goal then i.bests=i.bests+1 else i.rests=i.rests+1 end end end

function score(i)
  local acc,out=0,{}
  for _,x in pairs(i.log) do if x.want==x.got then acc=acc+1/#i.log end end
  for col,xns in pairs(i.best) do
    for x,b in pairs(xns) do
      local r1 = has2(i.rest,col,x)/i.rests
      local b1 = b/i.bests
      push(out, {100*(b1^2/(b1+r1))//1, col,x,b}) end end
  return acc, sort(out,down1) end

function nb1(file, log)
  local i = {h={}, nh=0,e={}, names=nil, n=0, wait=the.wait,
            bests=0,rests=0,best={}, rest={},log=log or {}}
  for row in lines(file) do
    if not i.names then i.names=row else
      test(i,row); train(i,row) end end
  return i end

---       . _|_ |_     _      _|
---    vv |  |  | |  (/_ vv (_|

function nb2(file,  log)
  local tmp, i, create, update, discretize = {}
  i = {h={}, nh=0,e={}, names=nil, n=0, wait=the.wait,
       bests=0,rests=0,best={}, rest={},log=log or {},
       hi={},lo={}, nums={}}

  function create(t)
    for j,txt in pairs(t) do
      if ako.num(txt) then i.nums[j] = {lo=1E32, hi=-1E32} end end; return t end

  function update(t,    x)
    for j,n in pairs(i.nums) do
      x=t[j]
      if x~="?" then n.lo=min(x,n.lo); n.hi=max(x,n.hi) end end; return t end

  function discretize(t, x)
   for j,n in pairs(i.nums) do
     x=t[j]
     t[j]=x=="?" and x or (x - n.lo) // ((n.hi - n.lo+1E-32) / the.bins) end end

  tmp={}
  for row in lines(file) do
    if not i.names then i.names = create(row) else push(tmp,update(row)) end end
  for _,row in pairs(tmp) do
    discretize(row); test(i,row); train(i,row) end
  return i end

---     _ _  _ _|_ _ ._ _
---    | | |(/_ | | |(__\

function abcd(gotwants, show)
  local i, exists, add, report, pretty = {
    data=data or "data", rx= rx or "rx",known={},a={},b={},c={},d={},yes=0,no=0}

  function exists(x,   new)
    new = not i.known[x]
    inc(i.known,x)
    if new then
      i.a[x]=i.yes + i.no; i.b[x]=0; i.c[x]=0; i.d[x]=0 end end

  function report(    p,out,a,b,c,d,pd,pf,pn,f,acc,g,prec)
    p = function (z) return math.floor(100*z + 0.5) end
    out= {}
    for x,_ in pairs( i.known ) do
      pd,pf,pn,prec,g,f,acc = 0,0,0,0,0,0,0
      a= (i.a[x] or 0); b= (i.b[x] or 0); c= (i.c[x] or 0); d= (i.d[x] or 0);
      if b+d > 0     then pd   = d     / (b+d)        end
      if a+c > 0     then pf   = c     / (a+c)        end
      if a+c > 0     then pn   = (b+d) / (a+c)        end
      if c+d > 0     then prec = d     / (c+d)        end
      if 1-pf+pd > 0 then g=2*(1-pf) * pd / (1-pf+pd) end
      if prec+pd > 0 then f=2*prec*pd / (prec + pd)   end
      if i.yes + i.no > 0 then
         acc= i.yes / (i.yes + i.no) end
      out[x] = {data=i.data,rx=i.rx,num=i.yes+i.no,a=a,b=b,c=c,d=d,acc=p(acc),
                prec=p(prec), pd=p(pd), pf=p(pf),f=p(f), g=p(g), class=x} end
    return out end

  function pretty(t)
    print""
    local s1  = "%10s | %10s | %4s | %4s | %4s | %4s "
    local s2  = "| %3s | %3s| %3s | %4s | %3s | %3s |"
    local d,s = "---", (s1 .. s2)
    print(fmt(s,"db","rx","a","b","c","d","acc","pd","pf","prec","f","g"))
    print(fmt(s,d,d,d,d,d,d,d,d,d,d,d,d))
    for _,x in pairs(slots(t)) do
      local u = t[x]
      print(fmt(s.." %s", u.data,u.rx,u.a, u.b, u.c, u.d,
                          u.acc, u.pd, u.pf, u.prec, u.f, u.g, x)) end end

  for _,one in pairs(gotwants) do
    exists(one.want)
    exists(one.got)
    if one.want == one.got then i.yes=i.yes+1 else i.no=i.no+1 end
    for x,_ in pairs(i.known) do
      if   one.want == x
      then inc(one.want == one.got and i.d or i.b, x)
      else inc(one.got  == x       and i.c or i.a, x) end end end
  return show and pretty(report()) or report() end
```

```
-------------------------------------------------------------------------------
---   ____ _  _ ____ ____ ____    ____ ____ _  _ ____ ____ ____
---   [__  |  | |__] |___ |__/    |__/ |__| |\ | | __ |___ [__
---   ___] |__| |    |___ |  \    |  \ |  | | \| |__] |___ ___]

function nb3(file,  log)
  local tmp, i, create, update, discretize1, discretize = {}
  i = {h={}, nh=0,e={}, names=nil, n=0, wait=the.wait,
       bests=0,rests=0,best={}, rest={},log=log or {},
       nums={}}

  function create(t)
    for j,txt in pairs(t) do
      if ako.num(txt) then i.nums[j] = {} end end; return t end

  function update(t,    x)
    for j,n in pairs(i.nums) do
      x=t[j]
      if x~="?" then push(n, {x=x, y= t[#t]}) end end; return t end

  function discretize1(t,x)
    if x == "?" then return x end
    for j,b in pairs(t) do if b.lo <= x and x < b.hi then return j end end end

  function discretize(t, x)
    for j,bins in pairs(i.nums) do t[j] = discretize1(bins,t[j]) end end

  tmp={}
  for row in lines(file) do
    if not i.names then i.names = create(row) else push(tmp,update(row)) end end
  for j,xys in pairs(i.nums) do i.nums[j] = bins(xys) end
  for _,row in pairs(tmp) do
    discretize(row);
    test(i,row); train(i,row) end
  return i end

---     |`. _  _|  |_ . _  _
---    ~|~|| |(_|  |_)|| |_\

function bins(xys)
  xys  = sort(xys, upx)
  local cohen    = the.cohen * (per(xys,.9).x - per(xys, .1).x) / 2.54
  local minItems = #xys / the.bins
  local out, b4  = {}, -math.huge
  local function add(f,z) f[z] = (f[z] or 0) + 1 end
  local function sub(f,z) f[z] =  f[z] - 1       end
  local function argmin(lo,hi)
    local lhs, rhs, cut, div, xpect, xy = {},{}
    for j=lo,hi do add(rhs, xys[j].y) end
    div = ent(rhs)
    if hi-lo+1 > 2*minItems
    then
      for j=lo,hi - minItems do
        add(lhs, xys[j].y)
        sub(rhs, xys[j].y)
        local n1,n2 = j - lo +1, hi-j
        if   n1        > minItems and           -- enough items (on left)
             xys[j].x ~= xys[j+1].x and         -- there is a break here
             xys[j].x  - xys[lo].x > cohen and  -- not trivially small (on left)
             xys[hi].x - xys[j].x  > cohen      -- not trivially small (on right)
        then xpect = (n1*ent(lhs) + n2*ent(rhs)) / (n1+n2)
             if xpect < div then                -- cutting here simplifies things
               cut, div = j, xpect end end end --end for
    end -- end if
    if   cut
    then argmin(lo,    cut)
         argmin(cut+1, hi )
    else b4 = push(out, {lo=b4, hi=xys[hi].x, n=hi-lo+1, div=div}).hi end
  end ------------------------------------------------
  argmin(1,#xys)
  out[#out].hi =  math.huge
  return out end
-------------------------------------------------------------------------------
---    _  _ _ ____ ____
---    |\/| | [__  |
---    |  | | ___] |___

---    _ _  _ _|_|_  _
---    | | |(_| | | |_\

min = math.min
max = math.max

function per(t,p) return t[ (p or .5)*#t//1 ] end

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

---     _ |_  _  _ |
---    (_ | |(/_(_ |<

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

---     _ _      _ _|_
---    (_(_)|_|| | |

function inc(f,a,n)       f=f or{};f[a]=(f[a] or 0) + (n or 1) return f end
function inc2(f,a,b,n)    f=f or{};f[a]=inc( f[a] or {},b,n);  return f end
function inc3(f,a,b,c,n)  f=f or{};f[a]=inc2(f[a] or {},b,c,n);return f end

function has(f,a)       return f[a]                     or 0 end
function has2(f,a,b)    return f[a] and has( f[a],b)    or 0 end
function has3(f,a,b,c)  return f[a] and has2(f[a],b,c)  or 0 end

---    |. __|_ _
---    ||_\ | _\

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

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

function sort(t,f) table.sort(t,f); return t end

function upx(a,b)   return a.x < b.x end
function up1(a,b)   return a[1] < b[1] end
function down1(a,b) return a[1] > b[1] end


function slots(t, u)
  local function public(k) return tostring(k):sub(1,1) ~= "_" end
  u={};for k,v in pairs(t) do if public(k) then u[1+#u]=k end end
  return sort(u) end

---     __|_ _ . _  _   '~)  _|_|_ . _  _  _
---    _\ | | || |(_|   /_   | | |||(_|_\
---               _|                  _|

function words(s,sep,   t)
  sep="([^" .. (sep or ",")  .. "]+)"
  t={}; for y in s:gmatch(sep) do t[1+#t] = y end; return t end

function things(s) return map(words(s), thing) end

function thing(x)
  x = x:match"^%s*(.-)%s*$"
  if x=="true" then return true elseif x=="false" then return false end
  return tonumber(x) or x end

function lines(file,f,      x)
  file = io.input(file)
  f    = f or things
  return function() x=io.read(); if x then return f(x) else io.close(file) end e
nd end

---    _|_|_ . _  _  _   '~)  __|_ _ . _  _
---     | | |||(_|_\    /_  _\ | | || |(_|
---              _|                      _|

fmt = string.format

function oo(t) print(o(t)) end

function o(t,  seen, u)
  if type(t)~="table" then return tostring(t) end
  seen = seen or {}
  if seen[t] then return "..." end
  seen[t] = t
  local function show1(x) return o(x, seen) end
  local function show2(k) return fmt(":%s %s",k, o(t[k],seen)) end
  u = #t>0 and map(t,show1) or map(slots(t),show2)
  return (t.s or "").."{"..table.concat(u," ").."}" end

---     _|.
---    (_||

function cli(help)
  local d,used = {},{}
  help:gsub("\n ([-]([^%s]+))[%s]+(-[^%s]+)[^\n]*%s([^%s]+)",
    function(long,key,short,x)
      assert(not used[short], "repeated short flag ["..short.."]")
      used[short]=short
      for n,flag in ipairs(arg) do
        if flag==short or flag==long then
          x = x=="false" and true or x=="true" and "false" or arg[n+1] end end
       d[key] = x==true and true or thing(x) end)
  if d.help then os.exit(print(help)) end
  return d end

```

```
-------------------------------------------------------------------------------
---     ___  ____ _  _ ____ ____
---     |  \ |___ |\/| |  | [__
---     |__/ |___ |  | |__| ___]

function eg.ent()
  print(ent{a=9,b=7}) end

function eg.nb1()
  local i = nb1(the.file);
  local acc, out = score(i); print(acc); map(out,oo) end

function eg.nb2()
  local i = nb2(the.file);
  local acc, out = score(i); print(acc); map(out,oo) end

function eg.nb2a()
  local i = nb2(the.file);
  local acc, out = score(i)
  abcd(i.log, true)
  map(out,oo) end

function eg.bins(   t)
  local t,n = {},30
  for j=1,n do push(t, {x=j, y=j<.6*n and 1 or j<.8*n and 2 or 3}) end
  map(bins(t),oo)
end

function eg.nb3(  i)
  print(20)
  i=nb3("../etc/data/diabetes.csv")
  for n,bins in pairs(i.nums) do
    print(n,#bins) end
  local acc, out = score(i)  -- XXX
  print(#out)
  print(acc)
  map(out,oo)
  end

-------------------------------------------------------------------------------
---     ____ ___ ____ ____ ___
---     [__   |  |__| |__/  |
---     ___]  |  |  | |  \  |

the=cli(help)
math.randomseed( the.seed or 10019 )
if eg[the.todo] then eg[the.todo]() end
rogues()
```