```
1  --------------------------------------------------------------------------------
2  ---    __        __
3  ---   /\ \      /\ \           __         __              __          __
4  ---   \ \ \     \ \ \        /'__`\     /'__`\          /'__`\      /'__`\
5  ---    \ \ \  __ \ \ \___   /\ \/\ \   /\ \/\ \        /\ \/\ \    /\ \/\ \
6  ---     \ \ \L\ \ \ \  _ `\ \ \ \ \ \  \ \ \ \ \       \ \ \ \ \   \ \ \ \ \
7  ---      \ \___,_\ \ \_\ \_\ \ \_\ \ \  \ \_\ \ \       \ \_\ \ \   \ \_\ \ \
8  ---       \/__,_ /  \/_/\/_/  \/_/\/_/   \/_/\/_/        \/_/\/_/    \/_/\/_/
9  ---
10 ---      .------.
11 ---      | Ba   | Bad <----.   planning= (better - bad)
12 ---      |   56 |          |   monitor = (bad - better)
13 ---      .------.------.    |
14 ---             | B    |    v
15 ---             |    5 | Better
16 ---             .------.
17 ---
18 --------------------------------------------------------------------------------
19
20 local b4={}; for k,_ in pairs(_ENV) do b4[k]=k end
21 local help=[[
22
23   -bins  -b   number of bins              = 16
24   -cohen -c   cohen                        = .35
25   -file  -f   file name                    = ../etc/data/breastcancer.csv
26   -goal  -g   goal                         = recurrence-events
27   -K     -K   manage low class counts      = 1
28   -M     -M   manage low evidence counts   = 2
29   -seed  -S   seed                         = 10019
30   -todo  -t   start up action              = nothing
31   -wait  -w   wait                         = 10
32 ]]
33
34 local max,min,ent,per
35 local push,map,sort,up1,upx,down1,slots,up1,down1
36 local words,thing, things, lines
37 local cli
38 local fmt,o,oo
39 local inc,inc2,inc3,has,has2,has3
40 local rogues
41 local classify,test,train,score,nb1,nb2,abcd
42 local bins,nb3
43 local eg,the,ako={},{},{}
44
45 ---
46 ---    _ _    _ _ __ __ __    _ _   __ __
47 ---   (_(_) | |_| | | | | |   | \/ |_) (_/__\
48 ---                            / |
49 local ako={}
50 ako.num    = function(x) return x:find"^[A-Z]" end
51 ako.goal   = function(x) return x:find"[-+!]"  end
52 ako.klass  = function(x) return x:find"!$"       end
53 ako.ignore = function(x) return x:find":$"       end
54 ako.less   = function(x) return x:find"-$"       end
55
56 --------------------------------------------------------------------------------
57 -- BSD 2-Clause License
58 -- Copyright (c) 2022, Tim Menzies
59 --
60 -- Redistribution and use in source and binary forms, with or without
61 -- modification, are permitted provided that the following conditions are met:
62 --
63 -- 1. Redistributions of source code must retain the above copyright notice,this
64 --    list of conditions and the following disclaimer.
65 --
66 -- 2. Redistributions in binary form must reproduce the above copyright notice,
67 --    this list of conditions and the following disclaimer in the documentation
68 --    and/or other materials provided with the distribution.
69 --
70 -- THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
71 -- AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
72 -- IMPLIED WARRANTIES OF MERCHANTABILITY & FITNESS FOR A PARTICULAR PURPOSE ARE
73 -- DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
74 -- FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
75 -- DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
76 -- SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
77 -- CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
78 -- OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
79 -- OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
80
```

```
80 --------------------------------------------------------------------------------
81 ---    __   __   __  __
82 ---   /  ` |  \ /  \ |  |
83 ---   \__, |__/ \__/ |__|
84
85 function classify(i,t)
86   local hi,out = -1
87   for h,_ in pairs(i.h) do
88     local prior = ((i.h[h] or 0) + the.K)/(i.n + the.K*i.nh)
89     local l = prior
90     for col,x in pairs(t) do
91       if x ~= "?" and col ~= #t then
92         l=l*(has3(i.e,col,x,h) + the.M*prior)/((i.h[h] or 0) + the.M) end end
93     if l>hi then hi,out=l,h end end
94   return out end
95
96 function test(i,t)
97   if i.n > i.wait then push(i.log,{want=t[#t], got=classify(i,t)}) end   end
98
99 function train(i,t)
100   local more, kl = false, t[#t]
101   for col,x in pairs(t) do
102     if x ~="?" then
103       more = true
104       inc3(i.e, col, x, kl)
105       if col ~= #t then
106         inc2(kl==the.goal and i.best or i.rest, col,x) end end end
107   if more then
108     i.n = i.n + 1
109     if not i.h[kl] then i.nh = i.nh + 1 end
110     inc(i.h, kl)
111     if kl==the.goal then i.bests=i.bests+1 else i.rests=i.rests+1 end end end
112
113 function score(i)
114   local acc,out=0,{}
115   for _,x in pairs(i.log) do if x.want==x.got then acc=acc+1/#i.log end end
116   for col,xns in pairs(i.best) do
117     for x,b in pairs(xns) do
118       local r1 = has2(i.rest,col,x)/i.rests
119       local b1 = b/i.bests
120       push(out, {100*(b1^2/(b1+r1))//1, col,x,b}) end end
121   return acc, sort(out,down1) end
122
123 function nb1(file, log)
124   local i = {h={}, nh=0,e={}, names=nil, n=0, wait=the.wait,
125             bests=0,rests=0,best={}, rest={},log=log or {}}
126   for row in lines(file) do
127     if not i.names then i.names=row else
128       test(i,row); train(i,row) end end
129   return i end
130
131 ---
132 ---    VV |`-|`-|`-|     (/_ VV (_|
133
134 function nb2(file,  log)
135   local tmp, i, create, update, discretize, discretize1 = {}
136   i = {h={}, nh=0,e={}, names=nil, n=0, wait=the.wait,
137        bests=0,rests=0,best={}, rest={},log=log or {},
138        hi={},lo={}, nums={}}
139
140   function create(t)
141     for j,txt in pairs(t) do
142       if ako.num(txt) then i.nums[j] = {lo=1E32, hi=-1E32} end end; return t end
143
144   function update(t,    x)
145     for j,n in pairs(i.nums) do
146       x=t[j]
147       if x~="?" then n.lo=min(x,n.lo); n.hi=max(x,n.hi) end end; return t end
148
149   function discretize(x,j)
150     if x~="?" then
151       n = i.nums[j]
152       x = n and (x - n.lo) // ((n.hi - n.lo+1E-32) / the.bins) or x end
153     return x end
154
155   tmp={}
156   for row in lines(file) do
157     if not i.names then i.names = create(row) else push(tmp,update(row)) end end
158   for _,row in pairs(tmp) do
159     row=collect(row,discretize)
160     test(i,row); train(i,row) end
161   return i end
162
163 ---
164 ---    |`-|`-|(/_-|`-|`(_\
165
166 function abcd(gotwants, show)
167   local i, exists, add, report, pretty = {
168     data=data or "data", rx= rx or "rx",known={},a={},b={},c={},d={},yes=0,no=0}
169
170   function exists(x,   new)
171     new = not i.known[x]
172     inc(i.known,x)
173     if new then
174       i.a[x]=i.yes + i.no; i.b[x]=0; i.c[x]=0; i.d[x]=0 end end
175
176   function report(    p,out,a,b,c,d,pd,pf,pn,f,acc,g,prec)
177     p = function (z) return math.floor(100*z + 0.5) end
178     out= {}
179     for x,_ in pairs( i.known ) do
180       pd,pf,pn,prec,g,f,acc = 0,0,0,0,0,0,0
181       a= (i.a[x] or 0); b= (i.b[x] or 0); c= (i.c[x] or 0); d= (i.d[x] or 0);
182       if b+d > 0    then pd   = d    / (b+d)         end
183       if a+c > 0    then pf   = c    / (a+c)         end
184       if a+c > 0    then pn   = (b+d) / (a+c)         end
185       if c+d > 0    then prec = d    / (c+d)         end
186       if 1-pf+pd > 0 then g=2*(1-pf) * pd / (1-pf+pd) end
187       if prec+pd > 0 then f=2*prec*pd / (prec + pd)   end
188       if i.yes + i.no > 0 then
189         acc= i.yes / (i.yes + i.no) end
190       out[x] = {data=i.data,rx=i.rx,num=i.yes+i.no,a=a,b=b,c=c,d=d,acc=p(acc),
191                 prec=p(prec), pd=p(pd), pf=p(pf),f=p(f), g=p(g), class=x} end
192     return out end
193
194   function pretty(t)
195     print""
196     local s1  = "%10s | %10s | %4s | %4s | %4s | %4s "
197     local s2  = "| %3s | %3s| %3s | %4s | %3s | %3s |"
198     local d,s = "----", (s1 .. s2)
199     print(fmt(s,"db","rx","a","b","c","d","acc","pd","pf","prec","f","g"))
200     print(fmt(s,d,d,d,d,d,d,d,d,d,d,d,d))
201     for _,x in pairs(slots(t)) do
202       local u = t[x]
203       print(fmt(s.." %s", u.data,u.rx,u.a, u.b, u.c, u.d,
204                 u.acc, u.pd, u.pf, u.prec, u.f, u.g, x)) end end
205
206   for _,one in pairs(gotwants) do
207     exists(one.want)
208     exists(one.got)
209     if one.want == one.got then i.yes=i.yes+1 else i.no=i.no+1 end
210     for x,_ in pairs(i.known) do
211       if   one.want == x
212       then inc(one.want == one.got and i.d or i.b, x)
213       else inc(one.got  == x        and i.c or i.a, x) end end end
214   return show and pretty(report()) or report() end
```

```
215  --------------------------------------------------------------------
216  ---
217  ---    SUPER RANGES
218  ---
219
220  function nb3(file,  log)
221    local tmp, i, create, update, discretize1, discretize = {}
222    i = {h={}, nh=0,e={}, names=nil, n=0, wait=the.wait,
223         bests=0,rests=0,best={}, rest={},log=log or {},
224         nums={}}
225
226    function create(t)
227      for j,txt in pairs(t) do
228        if ako.num(txt) then i.nums[j] = {} end end; return t end
229
230    function update(t,     x)
231      for j,n in pairs(i.nums) do
232        x=t[j]
233        if x~="?" then push(n, {x=x, y= t[#t]}) end end; return t end
234
235    function discretize(x,j,   bins)
236      if x ~= "?" then
237        bins = i.nums[j]
238        if bins then
239          for j,bin in pairs(bins) do
240            if bin.lo <= x and x < bin.hi then return j end end end end
241      return x end
242
243    tmp={}
244    for row in lines(file) do
245      if not i.names then i.names = create(row) else push(tmp,update(row)) end end
246    for j,xys in pairs(i.nums) do i.nums[j] = bins(xys) end
247    for _,row in pairs(tmp) do
248      row = collect(row, discretize);
249      test(i,row); train(i,row) end
250    return i end
251
252  ---     ~|~ | ⌐|⌐ |_||⌐|_⌐
253  ---     ~|~ |⌐|⌐ |_||⌐|_⌐
254
255  function bins(xys)
256    xys  = sort(xys, upx)
257    local cohen   = the.cohen * (per(xys,.9).x - per(xys, .1).x) / 2.54
258    local minItems = #xys / the.bins
259    local out, b4 = {}, -math.huge
260    local function add(f,z) f[z] = (f[z] or 0) + 1 end
261    local function sub(f,z) f[z] =  f[z] - 1       end
262    local function argmin(lo,hi)
263      local lhs, rhs, cut, div, xpect, xy = {},{}
264      for j=lo,hi do add(rhs, xys[j].y) end
265      div = ent(rhs)
266      if hi-lo+1 > 2*minItems
267      then
268        for j=lo,hi - minItems do
269          add(lhs, xys[j].y)
270          sub(rhs, xys[j].y)
271          local n1,n2 = j - lo +1, hi-j
272          if   n1        > minItems and           -- enough items (on left)
273               xys[j].x ~= xys[j+1].x and         -- there is a break here
274               xys[j].x  - xys[lo].x > cohen and -- not trivially small (on left)
275               xys[hi].x - xys[j].x  > cohen      -- not trivially small (on right)
276          then xpect = (n1*ent(lhs) + n2*ent(rhs)) / (n1+n2)
277               if xpect < div then                -- cutting here simplifies things
278                 cut, div = j, xpect end end end --end for
279      end -- end if
280      if  cut
281      then argmin(lo,    cut)
282           argmin(cut+1, hi )
283      else b4 = push(out, {lo=b4, hi=xys[hi].x, n=hi-lo+1, div=div}).hi end
284    end ----------------------------------------------
285    argmin(1,#xys)
286    out[#out].hi =  math.huge
287    return out end
```

```
288  --------------------------------------------------------------------
289  ---
290  ---    MISC
291  ---
292
293  ---
294  ---    |⌐⌐|⌐|⌐|_⌐
295
296  min = math.min
297  max = math.max
298
299  function per(t,p) return t[ (p or .5)*#t//1 ] end
300
301  function ent(t)
302    local n=0; for _,m in pairs(t) do n = n+m end
303    local e=0; for _,m in pairs(t) do if m>0 then e= e+m/n*math.log(m/n,2) end end
304    return -e end
305
306  ---
307  ---    |_ |⌐|⌐ _|_ |<
308
309  function rogues()
310    for k,v in pairs(_ENV) do if not b4[k] then print("??",k,type(v)) end end end
311
312  ---
313  ---    |_(|_)|_||⌐|⌐|⌐
314
315  function inc(f,a,n)       f=f or{};f[a]=(f[a] or 0) + (n or 1) return f end
316  function inc2(f,a,b,n)    f=f or{};f[a]=inc( f[a] or {},b,n);  return f end
317  function inc3(f,a,b,c,n)  f=f or{};f[a]=inc2(f[a] or {},b,c,n);return f end
318
319  function has(f,a)        return f[a]                     or 0 end
320  function has2(f,a,b)     return f[a] and has( f[a],b)    or 0 end
321  function has3(f,a,b,c)   return f[a] and has2(f[a],b,c)  or 0 end
322
323  ---
324  ---    |i_⌐|⌐|_⌐
325
326  function push(t,x) t[1 + #t] = x; return x end
327
328  function map(t, f, u) u={};for k,v in pairs(t) do u[1+#u]=f(v) end;return u end
329  function collect(t,f, u) u={};for k,v in pairs(t) do u[k]=f(v,k) end;return u end
330
331  function sort(t,f) table.sort(t,f); return t end
332
333  function upx(a,b)    return a.x < b.x end
334  function up1(a,b)    return a[1] < b[1] end
335  function down1(a,b)  return a[1] > b[1] end
336
337
338  function slots(t, u)
339    local function public(k) return tostring(k):sub(1,1) ~= "_" end
340    u={};for k,v in pairs(t) do if public(k) then u[1+#u]=k end end
341    return sort(u) end
342
343  ---     ⌐|⌐|⌐|⌐|⌐ '~)   |⌐ ⌐|⌐|⌐|⌐|⌐_⌐
344  ---     _⌐|⌐|⌐|⌐|⌐ '/_ ⌐|⌐|⌐|⌐|⌐_⌐
345  ---
346
347  function words(s,sep,   t)
348    sep="([^" .. (sep or ",") .. "]+)"
349    t={}; for y in s:gmatch(sep) do t[1+#t] = y end; return t end
350
351  function things(s) return map(words(s), thing) end
352
353  function thing(x)
354    x = x:match"^%s*(.-)%s*$"
355    if x=="true" then return true elseif x=="false" then return false end
356    return tonumber(x) or x end
357
358  function lines(file,f,     x)
359    file = io.input(file)
360    f    = f or things
361    return function() x=io.read(); if x then return f(x) else io.close(file) end end end
362
363  ---     ⌐|⌐|⌐|⌐|⌐_⌐ '~) _⌐|⌐|⌐|⌐|⌐
364  ---     ⌐|⌐|⌐|⌐|⌐_⌐ '/_ _⌐|⌐|⌐|⌐|⌐
365  ---
366
367  fmt = string.format
368
369  function oo(t) print(o(t)) end
370
371  function o(t,   seen, u)
372    if type(t)~="table" then return tostring(t) end
373    seen = seen or {}
374    if seen[t] then return "…" end
375    seen[t] = t
376    local function show1(x) return o(x, seen) end
377    local function show2(k) return fmt(":%s %s",k, o(t[k],seen)) end
378    u = #t>0 and map(t,show1) or map(slots(t),show2)
379    return (t.s or "").."{"..table.concat(u," ").."}" end
380
381  ---
382  ---    |_ |i
383
384  function cli(help)
385    local d,used = {},{}
386    help:gsub("\n ([-]([^%s]+))[%s]+(-[^%s]+)[^\n]*%s([^%s]+)",
387      function(long,key,short,x)
388        assert(not used[short], "repeated short flag ["..short.."]")
389        used[short]=short
390        for n,flag in ipairs(arg) do
391          if flag==short or flag==long then
392            x = x=="false" and true or x=="true" and "false" or arg[n+1] end end
393        d[key] = x=="true" and true or thing(x) end)
394    if d.help then os.exit(print(help)) end
395    return d end
396
```

page 4

```
396  --------------------------------------------------------------------------------
397  ---
398  ---   DEMOS
399  ---
```

```
400
401  function eg.ent()
402    print(ent{a=9,b=7}) end
403
404  function eg.nb1()
405    local i = nb1(the.file);
406    local acc, out = score(i); print(acc); map(out,oo) end
407
408  -- there is a "?"in the output. nope
409  function eg.nb2()
410    local i = nb2(the.file);
411    local acc, out = score(i); print(acc); map(out,oo) end
412
413  function eg.nb2a()
414    local i = nb2(the.file);
415    local acc, out = score(i)
416    abcd(i.log, true)
417    map(out,oo) end
418
419  function eg.bins(   t)
420    local t,n = {},30
421    for j=1,n do push(t, {x=j, y=j<.6*n and 1 or j<.8*n and 2 or 3}) end
422    map(bins(t),oo)
423  end
424
425  function eg.nb3(  i)
426    print(20)
427    i=nb3("../etc/data/diabetes.csv")
428    for n,bins in pairs(i.nums) do
429      print(n,#bins) end
430    local acc, out = score(i)  -- XXX
431    print(#out)
432    print(acc)
433    map(out,oo)
434    end
435
```

```
435  --------------------------------------------------------------------------------
436  ---
437  ---   [] START
438  ---
```

```
439
440  the=cli(help)
441  math.randomseed( the.seed or 10019 )
442  if eg[the.todo] then eg[the.todo]() end
443  rogues()
444
445  ---              .----------.
446  ---              |          |
447  ---         -=  |_____|  =-
448  ---
449  ---          |___) = (___|
450  ---          ---   ---
451  ---              ###
452  ---          #  =  #           "This ain't chemistry.
453  ---          #######            This is art."
```