

```

1 local help= {}
2 NB:
3 (c)2022 Tim Menzies, timm@ieee.org
4
5 OPTIONS:
6 -k --k handle rare classes = 1
7 -m --m handle rare attributes = 2
8 -p --p distance coefficient = 2
9 -W --wait wait before classifying = 5
10
11 OPTIONS (other):
12 -h --help show help = false
13 -g --go start-up goal = nothing
14 -s --seed seed = 10019
15 -f --file file = ../data/auto93.csv]]
16
17 -- i n c i n i n g o s -----
18
19 local lib = require"lib"
20 local argmax = lib.argmax
21 local cli, csv, demos, is, normpdf = lib.cli, lib.csv, lib.demos, lib.is, lib.normpdf
22 local oo, push, read, rnd, str = lib.oo, lib.push, lib.read, lib.rnd, lib.str
23
24 local THE={}
25 help:gsub("[^-][^%s+][^m]%(^%s+)",function(key,x) THE[key] = read(x) end)
26
27 local NB, NUM, SYM, COLS, ROW, ROWS= is"NB", is"NUM", is"SYM", is"COLS", is"ROW", is"ROWS"

```

```

29 -- c o l u m n i n -----
30
31 function NUM.new(i) i.n,i.mu,i.m2,i.mu = 0,0,0,0 end
32 function NUM.mid(i,p) return rnd(i.mu,p) end
33 function NUM.like(i,x,...) return normpdf(x, i.mu, i.sd) end
34 function NUM.add(i,v, d)
35 if v=="?" then return v end
36 i.n = i.n + 1
37 d = v - i.mu
38 i.mu = i.mu + d/i.n
39 i.m2 = i.m2 + d*(v - i.mu)
40 i.sd = i.n<2 and 0 or (i.m2/(i.n-1))^0.5 end
41
42
43 function SYM.new(i) i.n,i.syms,i.most,i.mode = 0, {}, 0, nil end
44 function SYM.mid(i,...) return i.mode end
45 function SYM.like(i,x,prior) return ((i.syms[x] or 0)+THE.m*prior)/(i.n+THE.m) end
46 function SYM.add(i,v,inc) function i:add(v, -1) end
47 if v=="?" then return v end
48 inc=inc or 1
49 i.n = i.n + inc
50 i.syms[v] = inc + (i.syms[v] or 0)
51 if i.syms[v] > i.most then i.most,i.mode = i.syms[v],v end end
52
53 -- c o l s -----
54
55 local function usep(x) return not x:find"$" end
56 local function nump(x) return x:find"[A-Z]" end
57 local function goalp(x) return x:find"[0-9]" end
58 local function klassp(x) return x:find"[S]" end
59 local function new(at,txt, i)
60 txt = txt or ""
61 i = (nump(txt) and NUM or SYM)()
62 i.txt, i.usep, i.at, i.w = txt, usep(txt), at or 0, txt:find"--$" and -1 or 1
63 return i end
64
65 function COLS.new(i,t, col)
66 i.all, i.xs, i.ys, i.names = {}, {}, {}, t
67 for at,x in pairs(t) do
68 col = push(i.all, new(at,x))
69 if col.usep then
70 if klassp(col.txt) then i.klass=col end
71 push(goalp(col.txt) and i.ys or i.xs, col) end end end
72
73 function COLS.add(i,t)
74 for _,cols in pairs(i.xs,i.ys) do
75 for _,col in pairs(cols) do col:add(t[col.at]) end end
76 return t end
77
78 -- i n v -----
79
80 function ROW.new(i,of,cells) i.of,i.cells,i.evald=of,cells,false end
81 function ROW.klass(i) return i.cells[i.of.cols.klass.at] end
82
83 -- i n v s -----
84
85 local function load(src, fun)
86 if type(src)=="string" then for _,t in pairs(src) do fun(t) end
87 else for t in csv(src) do fun(t) end end end
88
89 function ROWS.new(i,t) i.cols=COLS(t); i.rows={} end
90 function ROWS.add(i,t)
91 t=t.cells and t or ROW(i,t)
92 i.cols:add(t.cells)
93 return push(i.rows, t) end
94
95 function ROWS.mid(i, cols, p, t)
96 t={};for _,col in pairs(cols or i.cols.ys) do t[col.txt]=col:mid(p) end;return t end
97
98 function ROWS.clone(i,t, j)
99 j= ROWS(i.cols.names);for _,row in pairs(t or {}) do j:add(row) end; return j end
100
101 function ROWS.like(i,t, nklasses, nrows, prior,like,inc,x)
102 prior = (#i.rows + THE.k) / (nrows + THE.k * nklasses)
103 like = math.log(prior)
104 for _,col in pairs(i.cols.xs) do
105 if x and x ~= "?" then
106 inc = col:like(x,prior)
107 like = like + math.log(inc) end end
108 return like end
109
110 -- i n s -----
111
112 --
113
114 -- (0) Use row1 to initial our 'overall' knowledge of all rows.
115 -- After that (1) add row to 'overall' and (2) ROWS about this row's klass.
116 -- (3) After 'wait' rows, classify row BEFORE updating training knowledge
117 function NB.new(i,src, guess)
118 i.overall, i.dict, i.list = nil, {}, {}
119 load(src, function(row, k)
120 if not i.overall then i.overall = ROWS(row) else -- (0) eat row1
121 row = i.overall:add(row) -- (1) add to overall
122 if #i.overall.rows > THE.wait then
123 print(row:klass(), i:guess(row)) -- (3) classify before updating
124 k = row:klass() -- what klass is this?
125 i.dict[k] = i.dict[k] or push(i.list, i.overall:clone()) -- klass is known
126 i.dict[k].txt = k -- each klass knows its name
127 i.dict[k]:add(row) end end end -- (2) add to this row's klass
128
129 function NB.guess(i,row)
130 return argmax(i.dict,
131 function(klass) return klass:like(row,#i.list,#i.overall.rows) end) end

```

```

132 -- t e s t s -----
133
134 local no,go = {},{}
135 function go.the() return type(THE.p)=="number" and THE.p==2 end
136
137 function go.argmax( t,fun)
138 fun=function(x) return -x end
139 t={50,40,0,40,50}
140 return s = argmax(t,fun) end
141
142 function go.num(n) n=NUM(); for i=1,100 do n:add(i) end; return n.mu==50.5 end
143
144 function go.sym(s)
145 s=SYM(); for _,x in pairs("a","a","a","a","b","b","c") do s:add(x) end
146 return s.mode=="a" end
147
148 function go.csv( n,s)
149 n,s=0,0; for row in csv(THE.file) do n=n+1; if n>1 then s=s+row[1] end end
150 return rnd(s/n,3) == 5.441 end
151
152 function go.rows( rows)
153 load(THE.file,function(t) if rows then rows:add(t) else rows=ROWS(t) end end)
154 return rnd(rows.cols.ys[1].sd,0)==847 end
155
156 function go.nb()
157 return Z68 == #NB("../data/diabetes.csv").dict["positive"].rows end
158
159 -- s t e s t -----
160
161 if pcall(debug.getlocal, 4, 1)
162 then return (ROW=ROW, ROWS=ROWS, NUM=NUM, SYM=SYM, THE=THE, lib=lib)
163 else THE = cli(THE,help)
164 demos(THE,go) end

```