

```

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 -S --small small leaf size = 5
10 -w --wait wait before classifying = .5
11
12 OPTIONS (other):
13 -f --file file = .././data/auto93.csv
14 -g --go start-up goal = nothing
15 -h --help show help = false
16 -s --seed seed = 10019]]
17
18
19
20 local lib = require"lib"
21 local argmax,big = lib.argmax, lib.big
22 local cil,csv,demos,i,s,normpdf = lib.cil, lib.csv, lib.demos, lib.is, lib.normpdf
23 local co,push,read,end = lib.co, lib.push, lib.read, lib.end
24 local same,str = lib.same,lib.str
25
26 local THE={}
27 help:gsub("[^-]|[^%[%$]|\\"%s|[%$%])+",function(key,x) THE[key] = read(x) end)
28
29 local NB,NUM,SYM,COLS,ROW,ROWS= "is"NB,"is"NUM,"is"SYM,"is"COLS,"is"ROW,"is"ROWS"
30 local RANGE, TREE = "is"RANGE,"is"TREE"
31
32
33
34 function NUM.new(i) i.n,i.mu,m2,i.mu,i.lo,i.hi = 0,0,0,0,big,-big end
35 function NUM.mid(i,p) return rnd(i.mu,p) end
36 function NUM.like(i,x,...) return normpdf(x,i.mu,i.sd) end
37 function NUM.bin(x) b=(i.hi-i.lo)/THE.bins; return i.lo==i.hi and 1 or math.floor(x/b+.5)*b end
38
39 function NUM.add(i,v, d)
40 if v=="?" then return v end
41 i.n = i.n + 1
42 d = v - i.mu
43 i.mu = i.mu + d/i.n
44 i.m2 = i.m2 + d*(v - i.mu)
45 i.sd = i.n<2 and 0 or (i.m2/(i.n-1))^.5
46 i.lo = math.min(x,i.lo)
47 i.hi = math.max(x,i.hi) end
48
49
50 function SYM.new(i) i.n,i.syms,i.most,i.mode = 0,{},0,nil end
51 function SYM.mid(i,...) return i.mode end
52 function SYM.like(i,x,prior) return ((i.syms[x] or 0)+THE.m*prior)/(i.n+THE.m) end
53 function SYM.bin(x) return x end
54
55 function SYM.sub(i,v) return add(v,-1) end
56 function SYM.add(i,v,inc)
57 if v=="?" then return v end
58 inc=inc or 1
59 i.n = i.n + inc
60 i.syms[v] = inc + i.syms[v] or 0
61 if i.syms[v] > i.most then i.most,i.mode = i.syms[v],v end end
62
63 function SYM.merge(i,j,k)
64 local k = SYM(i.at,i.txt)
65 for x,n in pairs(i.has) do k:add(x,n) end
66 for x,n in pairs(j.has) do k:add(x,n) end
67 return k end
68
69
70
71 local function usep(x) return not x:find"^S$" end
72 local function nump(x) return x:find"^[A-Z]" end
73 local function goalp(x) return x:find"^[a-z]" end
74 local function klasp(x,txt, i)
75 local function new(at,txt, i)
76 txt = txt or ""
77 i = (nump(txt) and NUM or SYM)()
78 i.txt,i.usep,i.at,i.w = txt,usep(txt),at or 0,txt:find"-S$" and -1 or 1
79 return i end
80
81 function COLS.new(i,t,col)
82 i.all,i.xs,i.ys,i.names={},{}},{},t
83 for at,x in pairs(t) do
84 col = push(i.all,new(at,x))
85 if col.usep then
86 if klasp(col.txt) then i.klass=col end
87 push(galp(col.txt) and i.ys or i.xs, col) end end end
88
89 function COLS.add(i,t)
90 for _ ,col in pairs(i.xs,i.ys) do
91 for _ ,col in pairs(cols) do col:add(t[col.at]) end end
92 return t end
93
94
95
96 function ROW.new(i,of,cells) i.of,i.cells,i.eval=of,cells,false end
97 function ROW.klass(i) return i.cells[i.of].klass.at end
98 function ROW.within(i,range,...) lo,hi,at,v=
99 lo, hi, at = range.xlo, range.xhi, range.yx.at
100 v = i.cells[at]
101 return v=="?" or (lo==hi and v==lo) or (lo<v and v<hi) end
102
103 function ROW.b4(i,j,at, x,y)
104 x,y = i.cells[at],j.cells[at]
105 x = x=="?" and -big or x
106 y = y=="?" and -big or y
107 return x < y end
108
109
110
111
112 local function load(src, fun)
113 if type(src)=="string" then for _,t in pairs(src) do fun(t) end
114 else for t in csv(src) do fun(t) end end end
115
116 function ROWS.new(i,t) i.cols=COLS(t); i.rows={} end
117 function ROWS.add(i,t)
118 t=t.cells and t or ROW(i,t)
119 i.cols=add(t.cells)
120 return push(i.rows, t) end
121
122 function ROWS.mid(i, cols, p, t)
123 t={};for _ ,col in pairs(cols or i.cols.ys) do t[col.txt]=col:mid(p) end;return t end
124
125 function ROWS.clone(i,t, j)
126 j= ROWS(i.cols.names);for _ ,row in pairs(t or {}) do j:add(row) end; return j end

```

```

127 function ROWS.like(i,t,nklasses,nrows, prior,like,inc,x)
128   prior = (#i.rows + THE.k) * (nrows + THE.k * nklasses)
129   like = math.log(prior)
130   for _col in pairs(i.cols.x) do
131     x = t.cells[col.at]
132     if x and x ~= "" then
133       inc = collike(x,prior)
134       like = like + math.log(inc) end end
135   return like end
136
137 -----
138
139 -- (0) Use row1 to initial our 'overall' knowledge of all rows.
140 -- (1) Add row1 to 'overall' and (2) ROWS about this row's class.
141 -- (3) After 'wait' rows, classify row BEFORE updating training knowledge
142 function NB.new(i,src,report, guess)
143   report = report or print
144   i.overall,i.dict,i.list = nil, {}, {}
145   load(src,function(row, k)
146     if not i.overall then i.overall = ROWS(row) else -- (0) eat row1
147       row = i.overall:add(row) -- (1) add to overall
148     if i.overall.rows == THE.wait then
149       report(row:klass(), i.guess) end -- (3) classify before updating
150     k = row:klass() -- what class is this?
151     i.dict[k] = i.dict[k] or push(i.list, i.overall:clone()) -- class is known
152     i.dict[k].txt = k -- each class knows its name
153     i.dict[k]:add(row) end end -- (2) add to this row's class
154
155 function NB.guess(i,row)
156   return argmax(i.dict,
157     function(klass) return klass:like(row,i.list,#i.overall.rows) end) end

```

```

150 -- function RANGE.new(i,xlo, xhi, ys) i.xlo, i.xhi, i.ys = xlo, xhi, ys end
151 function RANGE.add(i,x,y)
152   if x < i.xlo then i.xlo = x end -- works for string or num
153   if x > i.xhi then i.xhi = x end -- works for string or num
154   i.ysadd(y) end
155
156 function RANGE...tostring(i)
157   local x, lo, hi = i.ys.txt, i.xlo, i.xhi
158   if lo == hi then return fmt("(%s==%s",x, lo)
159   elseif hi == big then return fmt("(%s>%s",x, lo)
160   elseif lo == -big then return fmt("(%s<=%s", x, hi)
161   else return fmt("(%s<%s<=%s",lo,x,hi) end end
162
163 -- function TREE.new(i,listOfRows,gaud)
164 -- i.gaud, i.kids = gaud, {}
165 -- of = listOfRows[i][1].of
166 -- best = sort(map(of.col.s,
167 -- function(col) i:bins(col,listOfRows) end),lt"div")[]]
168 -- i.kids = map(best.rows, function(range)
169 --   listOfRows[] = {}
170 --   for label,rows in pairs(listOfRows) do
171 --     for row in pairs(rows) do
172 --       if row:selects(range) then push(listOfRows[label],
173 --         -- end
174 --
175 function TREE.bins(i,col,listOfRows)
176   local function merge(b4,min)
177     local t,j,a,b,c,A,B,C = {},1
178     while j <= #b4 do
179       a, b = b4[j], b4[j+1]
180       if b then
181         A,B = a.ys, b.ys
182         C = A:merge(B)
183         if A.<min or B.<min or C:div() <= (A.n*A:div() + B.n*B:div())/C.n then
184           j = j + 1
185           a = RANGE(a.xlo, b.xhi, C) end end
186           t[#t+1] = a
187           j = j + 1 end
188         if #t == #b4 then return merge(t,min) end
189         for j=2,#t do t[j].xlo = t[j-1].xhi end
190         t[1].xlo, t[#t].xhi = -big, big
191         return t
192       end
193       local out, ranges, n = {}, {}, 0
194       for label,rows in pairs(listOfRows) do
195         for row in pairs(rows) do
196           local v = row.cells[col.at]
197           if v ~= "?" then
198             n = n + 1
199             local r = col:bin(v)
200             ranges[r] = ranges[r] or push(out, RANGE(v,r, SYM(col.at, col.txt)))
201             ranges[r]:add(v,label) end end
202           out = sort(out,lt"xlo")
203           out = col:is=="NUM" and merge(out, n"THE.min) or out
204           return {col = col,
205             ranges = #out<2 and {} or out,
206             div = sum(out, function(ones) return one.n/n*one:div() end) end} end
207
208 -- (7 5 1 5
209
210 local no,go = {},{}
211 function go,the() return type(TH.the.p)=="number" and THE.p==2 end
212
213 function go.argmax(x,t,fun)
214   fun=function(x) return -x end
215   t={50,40,40,50,40}
216   return 3 == argmax(t,fun) end
217
218 function go.num(n) n=NUM(); for i=1,100 do n:add(i) end; return n.mu==50.5 end
219
220 function go.sym(s)
221   s=s:gsub("(.).x in pairs{"a","a","a","a","b","b","c"} do s:add(x) end
222   return s.mode=="a" end
223
224 function go.csv(n,s)
225   n,s,n,3; for row in Cols(TH.the.file) do n=n+1; if n>1 then s=s+row[1] end end
226   return rnd(s/n,3) == 5.441 end
227
228 function go.rows(rows)
229   TH.the.file,for row in Cols(TH.the.file) if rows then rows:add(t) else rows=ROWS(t) end end
230   return rnd(rows.cols.ys[1].sd,0)==847 end
231
232 function go.nb()
233   return 268 == #NB("J:/data/diabetes.csv").dict["positive"].rows end
234
235 local function _classify(file)
236   local abcd=require("abcd")
237   local abcd=Abcd()
238   NB(file, function(got,want) abcd:add(got,want) end)
239   abcd:pretty abcd:report()
240   return true end
241
242 function go.soybean() return _classify("J:/data/soybean.csv") end
243 function go.diabetes() return _classify("J:/data/diabetes.csv") end
244
245 -- 5 7 1 5 1 5
246
247 if pcall(debug, get_col, 4, 1)
248 then
249   RETURN (ROW=ROW, ROWS=ROWS, NUM=NUM, SYM=SYM, THE=THE,lib=lib)
250 else
251   THE = cli(TH,the,help)
252   demos(TH,the) end
253
254 --
255 --
256 --
257 --
258 --
259 --
260 --
261 --
262 --
263 --
264 --
265 --
266 --
267 --
268 --
269 --
270 --
271 --
272 --
273 --
274 --
275 --
276 --
277 --
278 --
279 --
280 --
281 --
282 --
283 --
284 --
285 --
286 --
287 --
288 --
289 --
290 --
291 --
292 --
293 --
294 --
295 --
296 --
297 --
298 --
299 --
300 --
301 --
302 --
303 --
304 --
305 --
306 --
307 --
308 --
309 --
310 --
311 --
312 --
313 --
314 --
315 --
316 --
317 --
318 --
319 --
320 --
321 --
322 --
323 --
324 --
325 --
326 --
327 --
328 --
329 --
330 --
331 --
332 --
333 --
334 --
335 --
336 --
337 --
338 --
339 --
340 --
341 --
342 --
343 --
344 --
345 --
346 --
347 --
348 --
349 --
350 --
351 --
352 --
353 --
354 --
355 --
356 --
357 --
358 --
359 --
360 --
361 --
362 --
363 --
364 --
365 --
366 --
367 --
368 --
369 --
370 --
371 --
372 --
373 --
374 --
375 --
376 --
377 --
378 --
379 --
380 --
381 --
382 --
383 --
384 --
385 --
386 --
387 --
388 --
389 --
390 --
391 --
392 --
393 --
394 --
395 --
396 --
397 --
398 --
399 --
400 --
401 --
402 --
403 --
404 --
405 --
406 --
407 --
408 --
409 --
410 --
411 --
412 --
413 --
414 --
415 --
416 --
417 --
418 --
419 --
420 --
421 --
422 --
423 --
424 --
425 --
426 --
427 --
428 --
429 --
430 --
431 --
432 --
433 --
434 --
435 --
436 --
437 --
438 --
439 --
440 --
441 --
442 --
443 --
444 --
445 --
446 --
447 --
448 --
449 --
450 --
451 --
452 --
453 --
454 --
455 --
456 --
457 --
458 --
459 --
460 --
461 --
462 --
463 --
464 --
465 --
466 --
467 --
468 --
469 --
470 --
471 --
472 --
473 --
474 --
475 --
476 --
477 --
478 --
479 --
480 --
481 --
482 --
483 --
484 --
485 --
486 --
487 --
488 --
489 --
490 --
491 --
492 --
493 --
494 --
495 --
496 --
497 --
498 --
499 --
500 --
501 --
502 --
503 --
504 --
505 --
506 --
507 --
508 --
509 --
510 --
511 --
512 --
513 --
514 --
515 --
516 --
517 --
518 --
519 --
520 --
521 --
522 --
523 --
524 --
525 --
526 --
527 --
528 --
529 --
530 --
531 --
532 --
533 --
534 --
535 --
536 --
537 --
538 --
539 --
540 --
541 --
542 --
543 --
544 --
545 --
546 --
547 --
548 --
549 --
550 --
551 --
552 --
553 --
554 --
555 --
556 --
557 --
558 --
559 --
560 --
561 --
562 --
563 --
564 --
565 --
566 --
567 --
568 --
569 --
570 --
571 --
572 --
573 --
574 --
575 --
576 --
577 --
578 --
579 --
580 --
581 --
582 --
583 --
584 --
585 --
586 --
587 --
588 --
589 --
590 --
591 --
592 --
593 --
594 --
595 --
596 --
597 --
598 --
599 --
600 --
601 --
602 --
603 --
604 --
605 --
606 --
607 --
608 --
609 --
610 --
611 --
612 --
613 --
614 --
615 --
616 --
617 --
618 --
619 --
620 --
621 --
622 --
623 --
624 --
625 --
626 --
627 --
628 --
629 --
630 --
631 --
632 --
633 --
634 --
635 --
636 --
637 --
638 --
639 --
640 --
641 --
642 --
643 --
644 --
645 --
646 --
647 --
648 --
649 --
650 --
651 --
652 --
653 --
654 --
655 --
656 --
657 --
658 --
659 --
660 --
661 --
662 --
663 --
664 --
665 --
666 --
667 --
668 --
669 --
670 --
671 --
672 --
673 --
674 --
675 --
676 --
677 --
678 --
679 --
680 --
681 --
682 --
683 --
684 --
685 --
686 --
687 --
688 --
689 --
690 --
691 --
692 --
693 --
694 --
695 --
696 --
697 --
698 --
699 --
700 --
701 --
702 --
703 --
704 --
705 --
706 --
707 --
708 --
709 --
710 --
711 --
712 --
713 --
714 --
715 --
716 --
717 --
718 --
719 --
720 --
721 --
722 --
723 --
724 --
725 --
726 --
727 --
728 --
729 --
730 --
731 --
732 --
733 --
734 --
735 --
736 --
737 --
738 --
739 --
740 --
741 --
742 --
743 --
744 --
745 --
746 --
747 --
748 --
749 --
750 --
75
```