

```

1  -- It is vain to do with more what can be done with less.
2  -- William Of Occam<>
3  -- The more you have, the more you are occupied.
4  -- The less you have, the more free you are.<br>-- Mother Teresa<>
5  -- Simplicity is the ultimate sophistication.<br>-- Leonardo da Vinci<>
6  -- Simplicity is prerequisite for reliability.<br>AM~*BM~*T Edsger W. Dijkstra<>
7  -- Less is more.<br>-- Dieter Rams<>
8  -- Less, plz<br>-- tim<br><br>
9  local help= []
10 NB:
11 (c)2022 Tim Menzies, tim@ieee.org
12
13 OPTIONS:
14 -f --bins      max number of bins      = 16
15 -k --k         handle rare classes     = 1
16 -m --m         handle rare attributes  = 2
17 -p --p         distance coefficient    = .5
18 -S --small     small leaf size        = 2
19 -W --wait      wait before classifying =
20
21 OPTIONS (other):
22 -f --file      file                    = .././data/auto93.csv
23 -g --go       start-up goal           = nothing
24 -h --help     show help                = false
25 -s --seed     seed                    = 10019]]
26
27 -----
28 -- ## Names
29 local _ = require"lib"
30 local argmax,big = _argmax, _big
31 local cli, csv, demos, klass, normpdf = _cli, _csv, _demos, _klass, _normpdf
32 local oo, push, read, rnd, same, str = _oo, _push, _read, _rnd, _same, _str
33
34 local THE={}
35 help:gsub("[^-][^%s+][^\\n]*%s{(%s)+}", function(key,x) THE[key] = read(x) end)
36
37 local COLS, NB, NUM = klass"COLS", klass"NB", klass"NUM"
38 local RANGE, ROW, ROWS = klass"RANGE", klass"ROW", klass"ROWS"
39 local SOME, SYM TREE = klass"SOME", klass"SYM", klass"TREE"
40
41 -----
42 -- ## class RANGE
43 function RANGE.new(i, xlo, xhi, ys) i.xlo,i.xhi,i.ys,i.rows = xlo,xhi,ys,{} end
44 function RANGE.add(i,x,y)
45 if x < i.xlo then i.xlo = x end -- works for string or num
46 if x > i.xhi then i.xhi = x end -- works for string or num
47 i.ys:add(y) end
48
49 function RANGE._tostring(i)
50 local x, lo, hi = i.ys.txt, i.xlo, i.xhi
51 if lo == hi then return fmt("%s=%s",x, lo)
52 elseif hi == big then return fmt("%s<%s",x, lo)
53 elseif lo == -big then return fmt("%s<=%s", x, hi)
54 else return fmt("%s<%s<=%s", lo,x,hi) end end
55
56 -----
57 -- ## class SOME
58 function SOME.new(i) i.n,i.t,i.ok=0,{},true end
59 function SOME.has(i) i.t=i.ok and i.t or sort(i.t); i.ok=true; return i.t end
60 function SOME.add(i,x)
61 if x=="?" then return x end
62 i.n,i.n+1
63 if #i.t < THE.some then i.ok=false; push(i.t,x)
64 elseif rand() < THE.some/i.n then i.ok=false; i.t[rand(#i.t)]=x end end
65
66 -----
67 -- ## class NUM
68 function NUM.new(i) i.n,i.mu,i.m2,i.w,i.lo,i.hi,i.some=0,0,0,1,big,-big,SOME() end
69 function NUM.mid(i,p) return rnd(i.mu,p) end
70 function NUM.like(i,x,...) return normpdf(x, i.mu, i.sd) end
71 function NUM.bin(x)
72 b=(i.hi - i.lo)/THE.bins; return i.lo==i.hi and 1 or math.floor(x/b+.5)*b end
73
74 function NUM.add(i, NUM, v, number)
75 if v=="?" then return v end
76 i.some:add(v)
77 i.n = i.n + 1
78 local d = v - i.mu
79 i.mu = i.mu + d/i.n
80 i.m2 = i.m2 + d*(v - i.mu)
81 i.sd = i.n<2 and 0 or (i.m2/(i.n-1))^0.5
82 i.lo = math.min(v, i.lo)
83 i.hi = math.max(v, i.hi) end
84
85 function NUM.merge(i,j, k)
86 local k = NUM(i.at, i.txt)
87 for _,n in pairs(i.some.t) do k:add(x) end
88 for _,n in pairs(j.some.t) do k:add(x,n) end
89 return k end
90
91 -----
92 -- ## class SYM
93 function SYM.new(i) i.n,i.syms,i.most,i.mode = 0,{},0,nil end
94 function SYM.mid(i,...) return i.mode end
95 function SYM.like(i,x,prior) return ((i.syms[x] or 0)+THE.m*prior)/(i.n+THE.m) end
96 function SYM.bin(x) return x end
97 function SYM.add(i,v,inc)
98 if v=="?" then return v end
99 inc=inc or 1
100 i.n = i.n + inc
101 i.syms[v] = inc + i.syms[v] or 0
102 if i.syms[v] > i.most then i.most,i.mode = i.syms[v],v end end
103
104 function SYM.merge(i,j, k)
105 local k = SYM(i.at, i.txt)
106 for x,n in pairs(i.has) do k:add(x,n) end
107 for x,n in pairs(j.has) do k:add(x,n) end
108 return k end
109
110 -----
111 -- ## class COLS
112 function COLS.new(i,t, new,is)
113 is={}
114 function is.use(x) return not x:find"$.S" end
115 function is.num(x) return x:find"^[A-Z]" end
116 function is.goal(x) return x:find"^[=]" end
117 function is.klass(x) return x:find"$" end
118 function is.weight(x) return x:find"$-" and -1 or 1 end
119 i.xs, i.ys, i.names = {}, {}, {}
120 for at,txt in pairs(t) do
121 new = (is.num(txt) and NUM or SYM) (at,txt)
122 new.usep, new.w = is.use(txt), is.weight(txt)
123 if new.usep then
124 if is.klass(new.txt) then i.klass=new end
125 push(is.goal(new.txt) and i.ys or i.xs, new) end end end
126
127 function COLS.add(i,t)
128 for _,cols in pairs(i.xs,i.ys) do
129 for _,col in pairs(cols) do col:add(t.cells[col.at]) end end
130 return t end
131
132 -----

```

```

133 -- ## class ROW
134 function ROW.new(i,of,t) i.of,i.cells,i.evald = of,t,false end
135 function ROW.cell(i) return i.cells[i.of.cols.klass.at] end
136 function ROW.within(i,range)
137 local lo, hi, at = range.xlo, range.xhi, range.ys.at
138 local v = i.cells[at]
139 return v=="?" or (lo==hi and v==lo) or (lo<v and v<=hi) end
140
141 -----
142 -- ## class ROWS
143 local function doRows(src, fun)
144 if type(src)=="string" then for _,t in pairs(src) do fun(t) end
145 else for t in csv(src) do fun(t) end end end
146
147 function ROWS.new(i,t) i.cols=COLS(t); i.rows={} end
148 function ROWS.add(i,t) return push(i.rows, i.cols:add(t.cells and t or ROW(i,t))) end
149
150 function ROWS.mid(i, cols, p, t)
151 t={}for _,col in pairs(cols or i.cols.ys) do t[col.txt]=col:mid(p) end;return t end
152
153 function ROWS.clone(i,t, j)
154 j= ROWS(i.cols.names);for _,row in pairs(t or {}) do j:add(row) end; return j end
155
156 function ROWS.like(i,t, nklasses, nrows, prior,like,inc,x)
157 prior = (#i.rows + THE.k) / (nrows + THE.k * nklasses)
158 like = math.log(prior)
159 for _,col in pairs(i.cols.xs) do
160 x = t.cells[col.at]
161 if x and x ~= "?" then
162 like = col:like(x,prior)
163 like = like + math.log(inc) end end
164 return like end
165
166 -----
167 -- ## class NB
168 -- (0) Use row1 to initial our 'overall' knowledge of all rows.
169 -- After that (1) add row to 'overall' and (2) ROWS about this row's klass.
170 -- (3) After 'wait' rows, classify row BEFORE updating training knowledge
171 function NB.new(i,src,report, row)
172 report = report or print
173 i.overall, i.dict, i.list = nil, {}, {}
174 doRows(src, function(row, k)
175 if not i.overall then i.overall = ROWS(row) else -- (0) eat row1
176 row = i.overall:add(row) -- add to overall
177 if #i.overall.rows > THE.wait then report(row:klass(), i:guess(row)) end
178 i:train(row) end end) end -- add tp rows's klass
179
180 function NB.train(i, row) i:known(row:klass()) :add(row) end
181 function NB._known(i,k)
182 i:dict[k] = i:dict[k] or push(i.list, i.overall:clone()) -- klass is known
183 i:dict[k].txt = k -- each klass knows its name
184 return i:dict[k] end
185
186 function NB.guess(i,row)
187 return argmax(i:dict,
188 function(klass) return klass:like(row,#i.list,#i.overall.rows) end) end
189
190 function TREE.new(i,listOfRows, gaurd)
191 i.gaurd,i.kids = gaurd, {}
192 of = listOfRows[1][1].of
193 best = sort(map(of.cols.x,
194 function(col) i:bins(col,listOfRows) end),lt"div")[1]
195 i.kids = map(best.ranges, function(range)
196 listOfRow=1)
197 listOfRow=1)
198 local function within(row) return row:within(best) end
199 local function within(rows) return map(rows, within) end
200 map(listOfRows, function(rows) return within(rows) end) end
201 tmp= map(rows,within)
202 if #tmp > stop then
203 end
204 i.kids = map(ranges,
205 function(range) return TREE(range.rows,xcols,yklass,y,range) end)
206 end
207
208 labels , all, xcols = {},{}
209 for label,rows in pairs(listOfRows) do
210 for _,row in pairs(rows) do
211 xcols = row.of.cols.xs
212 labels[ push(all,row).id ] = label end end
213 return TREE(all, xcols, SYM, function(row) return labels[row.id] end) end
214
215 local _ranges, _merge
216 function _ranges(i,rows,xcol,yklass,y)
217 local n,list, dict = 0, {}, {}
218 for _,row in pairs(rows) do
219 local v = row.cells[xcol.at]
220 if v ~= "?" then
221 n = n + 1
222 local pos = xcol:bin(v)
223 dict[pos] = dict[pos] or push(list, RANGE(v,v, yklass(xcol.at, xcol.txt)))
224 dict[pos]:add(v, y(row)) end end
225 list = sort(list, lt"do")
226 list = xcol.is=="NUM" and _merge(list, n*THE.min) or list
227 return _ranges = list,
228 div = sum(list,function(z) return z.ys:div() * z.ys.n/n end) end
229
230 -----

```

```

231 function _merge(b4,min)
232 local j,t a,b,c,ay,by,cy = 1,{}
233 while j <= #b4 do
234 a, b = b4[j], b4[j+1]
235 if b then
236 ay,by,cy = a.ys, b.ys, a.ys:merge(b.ys)
237 if ay.n<min or by.n<min or cy:div() <= (ay.n*ay:div()+by.n*by:div())/cy.n
238 then a = rANGE(a.xlo, b.xhi, cy)
239 j = j + 1 end end -- skip one, since it has just been merged
240 t[#t+1] = a
241 j = j + 1 end
242 if #t < #b4 then return _merge(t,min) end
243 for j=2,#t do t[j].xlo = t[j-1].xhi end
244 t[1].xlo, t[#t].xhi = -big, big
245 return t end
246
247 -----
248 -- ## TESTS
249 local no,go = {},{}
250 function go.the() print(1); print(THE); return type(THE.p)=="number" and THE.p==2 end
251
252 function go.argmax( t,fun)
253 fun=function(x) return -x end
254 t={50,40,0,40,50}
255 return 3 == argmax(t,fun) end
256
257 function go.num(n) n=NUM(); for x=1,100 do n:add(x) end; return n.mu==50.5 end
258
259 function go.sym(s)
260 s=SYM(); for _,x in pairs{"a","a","a","a","b","b","c"} do s:add(x) end
261 return s.mode=="a" end
262
263 function go.csv( n,s)
264 n,s=0,0; for row in csv(THE.file) do n=n+1; if n>1 then s=s+row[1] end end
265 return rnd(s/n,3) == 5.441 end
266
267 function go.rows( rows)
268 doRows(THE.file,function(t) if rows then rows:add(t) else rows=ROWS(t) end end)
269 return rnd(rows.cols.ys[1].sd,0)==847 end
270
271 function go.nb()
272 return 268 == #NB("../data/diabetes.csv"):dict["positive"].rows end
273
274 local function _classify(file)
275 local abcd=require"abcd"
276 local abcd=Abcd()
277 NB(file, function(got,want) abcd:add(got,want) end)
278 abcd:pretty(abcd:report())
279 return true end
280
281 function go.soybean() return _classify("../data/soybean.csv") end
282 function go.diabetes() return _classify("../data/diabetes.csv") end
283
284 -----
285 -- ## START
286 if poall(debug.getlocal, 4, 1)
287 then return (ROW=ROW, ROWS=ROWS, NUM=NUM, SYM=SYM, THE=THE, lib=lib)
288 else THE = cli(THE,help)
289 demos(THE,go) end
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
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```