

```

1 local R = require
2 local _,the,CLS,BIN,NUM = R"lib", R"the", R"cls", R"bin", R"num"
3 local o,oo,downl,map,push,sort,powerset = _,o,_,oo,_.downl,_.map,_.push,_.sort,_.p
4 overset
5 local slice,merge,slots,fmt,rnds = _.slice, _.merge,_.slots,_.fmt,_.rnds
6 local class,OBJ = _.class, _.OBJ
7
8 local RULE = class("RULE",OBJ)
9
10 function RULE.best(bins,h)
11     local function score1(b1,b2) return RULE({b1},h).score > RULE({b2},h).score end
12     return slice(sort(bins, score1), 1, the.beam) end
13
14 function RULE.fromBins(bins,all,h,bests,rests, n,out,rule,sizes,scores)
15     out={}
16     sizes=NUM()
17     scores=NUM()
18     for _,some in pairs(powerset(RULE.best(bins,h))) do
19         if #some>0 then
20             rule = RULE(some,h)
21             sizes:add(#some)
22             scores:add(rule.score)
23             push(out, (size=#some,score=rule.score,rule=rule)) end end
24     local function order(one)
25         return ((0 - sizes:norm(one.size))^2 + (1 - scores:norm(one.score))^2)^.5 end
26     local n = 0
27     for _,three in pairs(sort(out, function(a,b) return order(a) < order(b) end)) do
28         local selected1= three.rule:selects(bests)
29         local cover1 = 100*#selected1/#bests//1
30         local selected2= three.rule:selects(rests)
31         local cover2 = 100*#selected2/#rests//1
32         local some = all:clone(three.rule:selects(all.rows))
33         if cover1 < 100 or cover2 < 100 then
34             print(o(rnds(some:mid()))),
35                 o(rnds(some:div()))),
36                 fmt("%.3f%.4u%.4u%s", three.score, cover1, cover2, three.rule)
37             n=n+1
38             if n > the.beam then return end end end
39     return out end
40
41 function RULE:new(bins,h, t)
42     self:seen={}
43     self.bins = {}
44     for _,bin in pairs(bins) do
45         self.bins[bin.at] = self.bins[bin.at] or {}
46         push(self.bins[bin.at], bin) end
47     for _,one in pairs(self.bins) do sort(one, function(a,b) return a.lo < b.lo end)
48     end
49     self.score = self:scored(h)
50     function RULE:___toString() return self:show(self.bins) end --return self:show(sel
51     f.bins) end
52
53 function RULE:like(klass,h) -- h=("true"=100, "false"=40) n=100+40
54     local n=0; for _,v in pairs(h) do n = n + v end
55     local fs = {}
56     for at,bins in pairs(self.bins) do
57         fs[at] = 0
58         for _,bin in pairs(bins) do
59             fs[at] = fs[at] + (bin.ys.has[klass] or 0) end end
60     self:seen[klass] = fs
61     local prior = ((h[klass] or 0) + the.K) / (n + the.K * 2)
62     local out = math.log(prior)
63     for at,v in pairs(fs) do
64         local inc = (v+the.M*prior)/(h[klass]+the.M)
65         out=out + math.log( inc)
66     end
67     return out end
68
69 RULE.bias = {}
70 local bias = RULE.bias
71 function bias.optimize(b,r) return b+r==0 and 0 or b^2/(b+r) end
72 function bias.monitor( b,r) return b+r==0 and 0 or r^2/(b+r) end
73 function bias.tabu( b,r) return b+r==0 and 0 or 1/(b+r) end
74
75 function RULE:scored(h)
76     return self.bias[the.rule](self:like("left",h), self:like("right",h)) end
77
78 function RULE:selects(rows)
79     return map(rows, function(row) if self:select(row) then return row end end) end
80
81 function RULE:select(row)
82     local function ors(bins)
83         for _,bin in pairs(bins) do if bin:select(row) then return true end end
84         return false end
85     for at,bins in pairs(self.bins) do if not ors(bins) then return false end end
86     return true end
87
88 function RULE:show(ands)
89     local cat, order, sortor, sortand
90     cat = function(t,sep) return table.concat(t,sep) end
91     sortand= function(t) return map(slots(t), function(k) return t[k] end) end
92     sortor = function(a,b) return a.lo < b.lo end
93     return cat(map(sortand(ands),
94         function(ands)
95             return ("..cat(map(sort(ands1,sortor),
96                 function(ors1) return tostring(ors1) end)," or ").."") end)," and ")
97     end
98
99 -- print has to wipe out fullranges and print selected items
100 --sort(bins,order)
101 -- ors= function(bins)
102 --     return cat(map(merge(sort(bins,order),BIN.mergeNext)), " or ") end
103 -- return cat(map(bins, ors), " and ") end
104 return RULE

```