

```

1 #!/usr/bin/env python3 -B
2 # vim: ts=2:sw=2:sts=2:et
3 """
4 binr.py : build rules via stochastic incremental XAI
5 (c) 2025, Tim Menzies, timm@ieee.org, mit-license.org
6
7 Options:
8   -h      Show help.
9   -b bins=? Number of bins for discretization (int).
10  -B Budget=? Max rows to eval (int).
11  -c eras10 Number of rows in an (int)
12  -p p2=? Distance coefficient.
13  -r repeats=? Number of experimental repeats (int).
14  -s seed=? Random number seed (int).
15  -f file=? /data/auto93.csv File to load (str).
16 """
17 from math import floor,sqrt,cos,log,exp,pi
18 from typing import Any,Iterable
19 import fileinput,random,sys,re
20 rand = random.random
21
22 class o(dict):
23     "Structs with slots accessible via x.slot. And pretty print."
24     __getattribute__ = dict.__getitem__
25     __setattr__ = dict.__setitem__
26     def __repr__(i): return show(i)
27
28 the = o(bins=7, Budget=30, eras=10, p=2, repeats=20, seed=42,
29         file="/data/auto93.csv")
30
31 Qty = float | int
32 Atom = Qty | str | bool
33 Row = list[Atom]
34 Rows = list[Row]
35 # Num,Sym,Cols = o,o,o # defined below
36 # Col = Num | Sym # defined below
37 # Data = tuple[Rows, Cols] # defined below
38 #
39 def Sym() -> o:
40     "Summarize symbol."
41     return o(it=Sym, n=0, has={}, bins={})
42
43 def Num() -> o:
44     "Summarize numbers."
45     return o(it=Num, n=0, mu=0, sd=0, m2=0, bins={})
46
47 def Col(at=0, of="") -> o:
48     "Column name or data."
49     if (Num if of[0].isupper() else Sym)():
50         at = at
51     of = of
52     if of[-1] == "-":
53         return it
54
55 def Cols(names:list[str]) -> o:
56     "Factory. Turn column names into columns."
57     cols = [Col(at=i, of=s) for i,s in enumerate(names)]
58     return o(it=Cols, names=names,
59             all_=all, x=x, y=y)
60     x = [col for col in cols if str(col.of)[-1] not in "+-X"]
61     y = [col for col in cols if str(col.of)[-1] in "+-"])
62
63 def Data(rows = None) -> o:
64     "Summarize rows into columns."
65     return adds(rows, o(it=Data, n=0, rows=[], cols=None))
66
67 #
68 def add(i: o, # o = Col | Data,
69         item: Any, # item[i] -> Any: # returns item
70         ) -> Any: # returns item
71     "+Add or subtract items from columns or data."
72     if item=="?": return item
73     i.n += inc
74     if i.it is Sym: i.has[item] = inc + i.has.get(item, 0)
75     elif i.it is Num:
76         item = float(item)
77         if inc < 0 and i.h < 2:
78             i.n = i.mu + i.sd = i.m2 = 0
79         else:
80             d = item - i.mu
81             i.mu += inc * d / i.n
82             i.m2 += inc * d * (item - i.mu)
83             i.sd = 0 if i.n < 2 else sqrt(max(0,i.m2)/(i.n - 1))
84     elif i.it is Data:
85         if i.cols:
86             row = [add(c, item[c.of], inc) for c in i.cols.all]
87             i.rows.append(row) if inc > 0 else i.rows.remove(row)
88         else: i.cols = Cols(item)
89     return item
90
91 def sub(i,item):
92     "Subtract items."
93     return add(i,item,-1)
94
95 def adds(items:Iterable = None, it=None ) -> o: # returns it
96     "+Load many items into 'it' (default is 'Num')."
97     it = it or Num()
98     if str(items)[-4]==".csv":
99         with open(items, encoding="utf-8") as f:
100            for line in f:
101                if line: add(it, [s.strip() for s in line.split(",")])
102            else: [add(it, item) for item in (items or [])]
103    return it

```

```

105     def norm(num:Num, v:Dty) -> float:
106         "Returns 0..1."
107         return 1 / (1 + exp(-1.702 * (v - num.mu) / (num.sd + le-32)))
108
109     def bin(col:Col, v:Atom) -> int | Atom:
110         "Returns bins-1."
111         if v==None: return None
112         return floor( the.bins * norm(col,v) ) if v!="?" and col.it is Num else v
113
114     def dist(src:Iterable) -> float:
115         "Minkoski distance."
116         dn = 0,0
117         for d1 in src:
118             d = d1 ** the.p
119             dn += d
120         return (dn/n) ** (1/the.p)
121
122     def disty(data:Data, row:Row) -> float:
123         "Distance of 'row' to best values in each goal column."
124         return dist(abs(norm(col, row[col.of]) - col.best)) for col in data.cols.y)
125
126     def distx(data:Data, row1:Row, row2:Row) -> float:
127         "Distance between 'x' attributes of two rows."
128         return dist(_abs([col, row1[col.of], row2[col.of]]) for col in data.cols.x)
129
130     def _abs(a:Col, a:Atom, b:Atom) -> float:
131         "If any unknowns, assume max distance."
132         if a==b=="?": return 1
133         if col.it is Sym: return a != b
134         a,b = norm(col,a), norm(col,b)
135         if a <= b: return 0 if a>b-0.5 else 1
136         b = b if b != "?" else (0 if a>0.5 else 1)
137         return abs(a - b)
138
139     def scoreGet(data:Data, row:Row) -> Row:
140         "Sum the score of the bins used by 'row'."
141         return sum(x.bins[b].mu for x in data.cols.x
142                     if (b := bin(x.row[x.at])) in x.bins)
143
144     def scorePut(data:Data, row:Row, score:Qty):
145         "Increment the bins used by 'row'."
146         for x in data.cols.X:
147             if (b := bin(x, row[x.at])) != "?":
148                 add(x.bins[b], x.bins.get(b) or Num(x.at, b))
149                 add(x.bins[b], score)
150
151     def score(data:Data, eps=0):
152         "Guess next few scores using scores seen to date."
153         best_score, best_row = 1e32, None
154         random.shuffle(data.rows)
155         seen, rows, model = set(), data.rows, Data([data.cols.names])
156         for r in sorted(rows):
157             if len(seen) >= the.Budget: break
158             if len(seen) >= the.Budget: break
159             add(model, row)
160             scorePut(model, row, disty(model, rows))
161             seen.add(id(row))
162             j = len(rows) - 1
163             candidate = min(rows[j+1 : j+20], key=lambda r: scoreGet(model, r))
164             seen.add(id(candidate))
165             if (score := disty(model, candidate)) < best_score - eps:
166                 best_score, best_row = score, candidate
167
168     return best_row
169
170     def show(x):
171         "Pretty print."
172         t = type(x)
173         if t is o: return "[" + ", ".join(f"\"{k}\": {show(v[k])}" for k in x) + "]"
174         if t is float: return str(int(x)) if x == int(x) else f"\"{x:.3f}\""
175         if t is type(show): return x._name_ + "(" + ")"
176         return str(x)
177
178     #
179     def test_h(_): -> None:
180         print(__doc__)
181
182     def test_the(_): -> None:
183         print(the)
184
185     def test_s(n: str) -> None:
186         the.seed = float(n); random.seed(the.seed)
187
188     def test_sym_() -> None:
189         print(adds("aaabbc",Sym()))
190
191     def test_num_() -> None:
192         def box_muller(mu, sd):
193             return mu + sd * sqrt(-2 * log(rand())) * cos(2 * pi * rand())
194         print(adds(box_muller(10,2) for _ in range(10^4)))
195
196     def test_data(f = None):
197         ys, data = Num(), Data(f or the.file)
198         Y=lambda row: floor(100*disty(data, row))
199         for r in sorted(data.rows, key=Y)[:10]:
200             print(Y(r), r)
201
202     def test_distx(f = None):
203         xs, data = Num(), Data(f or the.file)
204         X=lambda row: floor(100*distx(data, row, data.rows[0]))
205         for r in sorted(data.rows, key=Y)[:10]:
206             print(X(r), r)
207
208     def test__tests_(k):
209         xs, data = Num(), Data(f or the.file)
210         X=lambda row: floor(100*disty(data, row))
211         for r in sorted(data.rows, key=Y)[:10]:
212             print(X(r), r)
213
214     _tests= {k:fun for k,fun in vars().items() if "test_" in k}
215
216     def test_all():
217         for k,fun in _tests.items(): print("\n----- "+k); fun_()
218
219     #
220     if __name__ == "__main__":
221         for n, s in enumerate(sys.argv):
222             if n == 1: vars().get("test").replace("-", "_"))
223             random.seed(int(s))
224             fn(sys.argv[n+1] if n < len(sys.argv)-1 else None)

```