```
#!/usr/bin/env python3 -B
"""xai.py: explainable multi–objective optimzation
(c) 2025 Tim Menzies, MIT license"""
import ast,sys,random
from math import sqrt,exp,floor
from types import SimpleNamespace as obj

BIG=1e32

the=obj(bins=7, budget=30, seed=1)

### Constructors ---------------------------------------------------------------
def Sym(): return obj(it=Sym, n=0, has={})
def Num(): return obj(it=Num, n=0, mu=0, m2=0)

def Col(at=0, txt=" "):
  col = (Num if txt[0].isupper() else Sym)()
  col.at, col.txt, col.best = at, txt, 0 if txt[-1]=="–" else 1
  return col

def Cols(names):
  cols = [Col(i,s) for i,s in enumerate(names)]
  return obj(it=Cols, names=names, all=cols,
             x = [col for col in cols if col.txt[-1] not in "+–X"],
             y = [col for col in cols if col.txt[-1] in "+–"])

def Data(rows=None):
   return adds(rows, obj(it=Data, rows=[], n=0, cols=None, _mid=None))

def clone(data, rows=None): return Data([data.cols.names] + (rows or []))

### Functions ------------------------------------------------------------------
def adds(src, it=None):
  it = it or Num()
  [add(it,v) for v in src]
  return it

def sub(it,v): return add(it,v,-1)

def add(it, v, inc=1):
  if v=="?": return v
  it.n += inc
  if   Sym is it.it: it.has[v] = inc + it.has.get(v,0)
  elif Num is it.it:
    if inc < 0 and it.n < 2: it.n = it.mu = it.sd = it.m2 = 0
    else:
      d     = v - it.mu
      it.mu += inc * d / it.n
      it.m2 += inc * d * (v - it.mu)
  elif Data is it.it:
    if it.cols:
      it._mid = None
      v = [add(c, v[c.at], inc) for c in it.cols.all]
      (it.rows.append if inc > 0 else it.rows.remove)(v)
    else: it.cols = Cols(v)
  return v

def norm(num,n):
  z = (n - num.mu) / sd(num)
  return 1 / (1 + exp(-1.7 * max(-3, min(3, z))))

def sd(num):
  return 1e-32 + (0 if num.n < 2 else sqrt(num.m2/(num.n - 1 )))

def mids(data):
  if not data._mid: data._mid = [mid(col) for col in data.cols.all]
  return data._mid

def mid(col): return col.mu if Num is col.it else max(col.has,key=col.has.get)

def disty(data,row):
  ys = data.cols.y
  return sqrt(sum(abs(norm(y,row[y.at]) - y.best)**2 for y in ys) / len(ys))

def distx(data,row1,row2):
  xs = data.cols.x
  return sqrt(sum(_aha(x, row1[x.at], row2[x.at]_*2) for x in xs) / len(xs))

def _aha(col,a,b):
  if a==b=="?": return 1
  if Sym is col.it : return a != b
  a,b = norm(col,a), norm(col,b)
  a = a if a != "?" else (0 if b>0.5 else 1)
  b = b if b != "?" else (0 if a>0.5 else 1)
  return abs(a - b)

def near(data):
  x = lambda d, r: distx(data, mid(d), r)
  y = disty
  rows = shuffle(data.rows[:])
  train, test = rows[:len(rows)//2], rows[len(rows)//2:]
  labeled = clone(data, train[:the.warm])
  pool = sorted(labeled.rows, key=lambda r: y(labeled, r))
  best, rest = clone(data, pool[:len(pool)//2]), clone(data, pool[len(pool)//2
:])

  for r in train[the.warm:the.budget]:
    add(labeled, r)
    if x(best, r) < x(rest, r) and y(labeled, r) < y(labeled, best.rows[-1]):
      add(best, r)
      best.rows.sort(key=lambda r: y(labeled, r))
      if best.n > labeled.n**0.5:
        add(rest, sub(best, best.rows.pop()))

  test.sort(key=lambda r: x(best, r) - x(rest, r))
  out = min(test[:the.test], key=lambda r: y(data, r))
  return out, y(data, out)


```

```
## Lib ------------------------------------------------------------------------
def o(v=None, dec=2,**d):
  isa = isinstance
  if d: v=d
  if isa(v, (int, float)): return f"{round(v, dec):,}"
  if isa(v, list):  return f"[{', '.join(o(k,dec) for k in v)}]"
  if isa(v, tuple): return f"({', '.join(o(k,dec) for k in v)})"
  if callable(v):   return v.__name__
  if hasattr(v, "__dict__"): v = vars(v)
  if isa(v, dict): return "{"+ " ".join(f":{k} {o(v[k],dec)}" for k in v) +"}"
  return str(v)

def coerce(s):
  try: return ast.literal_eval(s)
  except: return s

def csv(fileName):
  with open(fileName,encoding="utf–8") as f:
    for l in f:
      if (l:=l.split("%")[0].strip()):
        yield [coerce(x.strip()) for x in l.split(",")]

def shuffle(lst): random.shuffle(lst); return lst

#------------------------------------------------------------------------------
def go_h():
  "–h          show help"
  print(__doc__,"\n\nOptions:\n")
  for k,fun in globals().items():
    if k.startswith("go_"): print(" "+fun.__doc__)

def go_s(s):
  "–s [1]        set random SEED "
  the.seed = coerce(s); random.seed(the.seed)

def go_b(s):
  "–b [5]        set number of BINS used on discretization"
  the.bins = coerce(s)

def go_B(s):
  "–B [30]       set BUDGET for rows labelled each round"
  the.budget = coerce(s)

def go__all(file):
  "––all FILE     run all actions that use a FILE"
  for k,fun in globals().items():
    if k.startswith("go__") and k != "go__all":
      print("\n#",k,"––––––––––––"); fun(file)

def go__csv(file):
  "––csv FILE     test csv loading"
  for i,row in enumerate(csv(file)):
    if i % 40 ==0: print(i,row)

def go__data(file):
  "––data FILE    test ading columns from file"
  data =  Data(csv(file))
  print(*data.cols.names)
  for col in data.cols.x: print(o(col))

def go__clone(file):
  "––clone FILE    test echoing structure of a table to a new table"
  data1 =  Data(csv(file))
  data2 = clone(data1,data1.rows)
  assert data1.cols.x[1].mu == data2.cols.x[1].mu

def go__disty(file):
  "––disty FILE    can we sort rows by their distance to heaven?"
  data=Data(csv(file))
  print(*data.cols.names)
  for row in sorted(data.rows, key=lambda r: disty(data,r))[::40]:
    print(*row)

def go__xai(file):
  "––xai FILE     can we succinctly list main effects in a table?"
  print("\n"+file)
  xai(Data(csv(file)))

def go__six(file):
  "––six FILE     redo xai, but in each loop, just read BUDGET rows"
  xai(Data(csv(file))); print(" ")
  go_s(the.seed)
  for b in [5,10,20,30]:
    go_B(the.budget)
    print(b,sorted(six(Data(csv(file))) for _ in range(20)))

if __name__ == "__main__":
  for n, s in enumerate(sys.argv):
    if fn := vars().get(f"go{s.replace('–', '_')}"):
      fn(sys.argv[n+1]) if n < len(sys.argv) - 1 else fn()
```