```python
#!/usr/bin/env python3
"""
bing1.py: stochastic landscape analysis for multi objective reasoning
(c) 2025 Tim Menseedzies, <timm@ieee.org>. MIT license

Options, with (defaults):

  -A   Acq    xploit or xplore or adapt (xploit)
  -F   Few    a few rows to explore (64)
  -G   Guess  division best and rest (0.5)
  -f   file   data name (../moot/optimize/misc/auto93.csv)
  -k   k      bayes hack for rare classes (1)
  -l   leaf   min leaf size (2)
  -m   m      bayes hack for rare attributes (2)
  -p   p      set mankowski coeffecient (2)
  -r   rseed  set random number rseed (123456781)
  -s   start  guesses, initial (4)
  -S   Stop   guesses, max (20)
 """
import traceback,random,math,sys,re
sys.dont_write_bytecode = True

### Sample data --------------------------------------------------------------------
EXAMPLE="""
Max_spout, hashing, Spliters, Counters, Throughput+, Latency-
1      , off   , 1     , 1     , 4255.3    , 2.5621
1      , on    , 1     , 3     , 4160.1    , 2.6057
1      , off   , 1     , 6     , 4089.5    , 2.55
1      , on    , 1     , 9     , 4156.9    , 2.5688
1      , on    , 1     , 12    , 4013.8    , 2.5474
1      , off   , 1     , 15    , 4194.1    , 2.5676
1      , on    , 1     , 18    , 3964.2    , 2.5503
1      , off   , 2     , 1     , 4294.7    , 4.7793
1      , on    , 2     , 3     , 4343.6    , 2.381
1      , off   , 2     , 6     , 4423.6    , 2.3538
1      , on    , 2     , 9     , 4369.6    , 2.4306
1      , off   , 2     , 12    , 4288.1    , 2.3965
1      , on    , 2     , 15    , 4291.2    , 2.4462
1      , off   , 2     , 18    , 4236.2    , 2.4647
1      , on    , 3     , 1     , 4980.3    , 2.1598
1      , off   , 3     , 3     , 5058.6    , 3.5506
1      , on    , 3     , 6     , 4836.7    , 2.1283
1      , off   , 3     , 9     , 4786.9    , 2.1468
1      , on    , 3     , 12    , 4528.8    , 3.0358
1      , off   , 3     , 15    , 4767.6    , 2.2173
1      , on    , 3     , 18    , 4949.1    , 2.1277
1      , off   , 6     , 1     , 4904.2    , 2.1626
1      , on    , 6     , 3     , 5151      , 2.0815
1      , off   , 6     , 6     , 4847.1    , 2.1376
1      , on    , 6     , 9     , 4891.9    , 2.1503
1      , off   , 6     , 12    , 4871      , 2.2277
1      , on    , 6     , 15    , 4645.8    , 2.1468
1      , off   , 6     , 18    , 4688.1    , 2.2277
10     , on    , 1     , 1     , 8226.1    , 13.733
10     , off   , 1     , 3     , 12697     , 9.2121
10     , on    , 1     , 6     , 14870     , 8.1247
10     , off   , 1     , 9     , 14807     , 7.5491
10     , on    , 1     , 12    , 15374     , 7.1335
10     , off   , 1     , 15    , 16019     , 7.3717
10     , on    , 1     , 18    , 15103     , 7.3965
10     , off   , 2     , 1     , 7006.2    , 15.859
10     , on    , 2     , 3     , 14169     , 8.1471
10     , off   , 2     , 6     , 18462     , 6.481
10     , on    , 2     , 9     , 18652     , 6.2867
10     , off   , 2     , 12    , 20233     , 5.7734
10     , on    , 2     , 15    , 19505     , 5.6023
10     , off   , 2     , 18    , 19335     , 5.641
10     , on    , 3     , 1     , 8219.4    , 13.865
10     , off   , 3     , 3     , 14591     , 7.6695
10     , on    , 3     , 6     , 15736     , 7.2908
10     , off   , 3     , 9     , 17161     , 6.5827
10     , on    , 3     , 12    , 17130     , 6.2694
10     , off   , 3     , 15    , 17209     , 6.2798
10     , on    , 3     , 18    , 16140     , 7.2948
10     , off   , 6     , 1     , 7524.2    , 13.959
10     , on    , 6     , 3     , 16238     , 7.0838
10     , off   , 6     , 6     , 20089     , 5.2988
10     , on    , 6     , 9     , 20066     , 5.0202
10     , off   , 6     , 12    , 19528     , 4.9185
10     , on    , 6     , 15    , 19157     , 5.0006
10     , off   , 6     , 18    , 18380     , 5.0711
100    , on    , 1     , 1     , 8511.2    , 135.2
100    , off   , 1     , 3     , 15515     , 75.825
100    , on    , 1     , 6     , 18264     , 61.409
100    , off   , 1     , 9     , 18652     , 62.08
100    , on    , 1     , 12    , 20872     , 55.886
100    , off   , 1     , 15    , 19875     , 53.539
100    , on    , 1     , 18    , 20121     , 56.687
100    , off   , 2     , 1     , 8746      , 117.57
100    , on    , 2     , 3     , 18568     , 65.437
100    , off   , 2     , 6     , 20814     , 53.103
100    , on    , 2     , 9     , 24962     , 43.247
100    , off   , 2     , 12    , 26373     , 40.169
100    , on    , 2     , 15    , 25948     , 46.001
100    , off   , 2     , 18    , 25565     , 39.447
100    , on    , 3     , 1     , 8465.1    , 132.78
100    , off   , 3     , 3     , 16941     , 65.185
100    , on    , 3     , 6     , 20045     , 58
100    , off   , 3     , 9     , 21448     , 54.396
100    , on    , 3     , 12    , 20821     , 56.731
100    , off   , 3     , 15    , 23240     , 51.463
100    , on    , 3     , 18    , 21234     , 53.927
100    , off   , 6     , 1     , 9214.4    , 116.13
100    , on    , 6     , 3     , 20359     , 55.501
100    , off   , 6     , 6     , 21587     , 48.702
100    , on    , 6     , 9     , 23142     , 37.915
100    , off   , 6     , 12    , 24892     , 41.478
100    , on    , 6     , 15    , 23675     , 32.286
100    , off   , 6     , 18    , 22884     , 33.092
1000   , on    , 1     , 1     , 10038     , 1063.6
1000   , off   , 1     , 3     , 20050     , 553.74
1000   , on    , 1     , 6     , 22015     , 511.62
1000   , off   , 1     , 9     , 24910     , 467.36
1000   , on    , 1     , 12    , 21808     , 470.82
1000   , off   , 1     , 15    , 23497     , 439.35
1000   , on    , 1     , 18    , 24392     , 419.91
1000   , off   , 2     , 1     , 8666.8    , 1239.5
1000   , on    , 2     , 3     , 22289     , 518.71
1000   , off   , 2     , 6     , 25805     , 463.33
1000   , on    , 2     , 9     , 28129     , 398.1
1000   , off   , 2     , 12    , 32399     , 332.68
1000   , on    , 2     , 15    , 33549     , 321.53
1000   , off   , 2     , 18    , 32815     , 341.28
1000   , on    , 3     , 1     , 9973.9    , 1105.8
1000   , off   , 3     , 3     , 19036     , 595.91
"""


### Create -------------------------------------------------------------------------
# Summary of numeric columns.
def Num(inits=[],at=0, txt=" "):
  return adds(o(it=Num,
                n=0,       ## items seen
                at=at,     ## column position
                txt=txt,   ## column name
                mu=0,      ## mean
                sd=0,      ## standard deviation
                m2=0,      ## second moment
                hi= -big,  ## biggest seen
                lo= big,   ## smallest seen
                heaven= (0 if txt[-1] == "-" else 1) ## 0,1 = minimze,maximize
                ), inits)

# Summary of symboloc columns.
def Sym(inits=[], at=0, txt=" "):
  return adds(o(it=Sym,
                n=0,      ## items see
                at=at,    ## column position
                txt=txt,  ## column name
                has={}    ## counts of symbols seen
                ), inits)

# Factory. <br> List[str] -> Dict[str, List[ Sym | Num ]]
def Cols(names):
  all,x,y = [],[],[]
  for c,s in enumerate(names):
    all += [(Num if s[0].isupper() else Sym)(at=c, txt=s)]
    if s[-1] != "X":
      (y if s[-1] in "+-" else x).append(all[-1])
  return o(it=Cols,
           names=names,   ## all the column names
           all=all,       ## all the columns
           x=x,           ## also, independent columns stored here
           y=y)           ## also, dependent columns stored here

# Data stores rows and columns.
def Data(inits):
  inits = iter(inits)
  return adds( o(it=Data,
                 n=0,                    ## items seen
                 _rows=[],               ## rows
                 cols=Cols(next(inits))  ## columns (which summarize the rows)
                 ), inits)

def clone(data, rows=[]):
  return Data([data.cols.names]+rows)

### Update -------------------------------------------------------------------------
# Subtraction means add, with a neative incre
def sub(i,v,purge=False):
  return add(i, v, inc= -1, purge=purge)

# Add `v` to `i`. Skip unknowns ("?"), return v.
def add(i,v, inc=1, purge=False): # -> v
  def _sym(sym,s): sym.has[s] = inc + sym.has.get(s,0)

  def _data(data,row):
    if inc < 0:
      if purge: data._rows.remove(v)
      [sub(col, row[col.at], inc) for col in data.cols.all]
    else:
      data._rows += [[add(col, row[col.at],inc) for col in data.cols.all]]

  def _num(num,n):
    num.lo = min(n, num.lo)
    num.hi = max(n, num.hi)
    if inc < 0 and num.n < 2:
      num.sd = num.m2 = num.mu = num.n = 0
    else:
      d      = n - num.mu
      num.mu += inc * (d / num.n)
      num.m2 += inc * (d * (n - num.mu))
      num.sd  = 0 if num.n <=2 else (num.m2/(num.n - 1)) ** .5

  if v != "?":
    i.n += inc
    (_num if i.it is Num else (_sym if i.it is Sym else _data))(i,v)
  return v

### Query --------------------------------------------------------------------------
# Middle tendancy.
def mid(i):
  _mode = lambda: max(i.has,key=i.has.get)
  return i.mu    if i.it is Num else (
         _mode() if i.it is Sym else ([mid(col) for col in i.cols.all]))

# Spread around middle tendancy.
def spread(i):
  _ent = lambda: -sum(p*math.log(p,2) for n in i.has.values() if (p:=n/i.n) > 0)
  return i.sd   if i.it is Num else (
         _ent() if i.it is Sym else ([spread(col) for col in i.cols.all]))

# Map v --> (0..1) for lo..hi.
def norm(num,v):
  return v if v=="?" else (v-num.lo) / (num.hi-num.lo + 1/big)


### Bayes --------------------------------------------------------------------------
# Return the `data` in `datas` that likes this `row` the most.
def likes(datas, row):
  n = sum(data.n for data in datas)
  return max(datas, key=lambda data: like(data, row, n, len(datas)))

# How much does this `data` like this `row`?
def like(data, row, nall=2, nh=100):
  prior = (data.n + the.k) / (nall + the.k*nh)
  tmp = [pdf(c,row[c.at],prior)
         for c in data.cols.x if row[c.at] != "?"]
  return sum(math.log(n) for n in tmp + [prior] if n>0)

# Pdf of `v` in Nums or Syms.
def pdf(col,v, prior=0):
  if col.it is Sym:
    return (col.has.get(s,0) + the.m*prior) / (col.n + the.m + 1/big)
  sd = col.sd or 1 / big
  var = 2 * sd * sd
  z = (v - col.mu) ** 2 / var
  return min(1, max(0, math.exp(-z) / (2 * math.pi * var) ** 0.5))

# Split rows to best,rest. Label row that's e.g. max best/rest."
def acquires(data):
  def _acquire(b, r, acq="xploit", p=1):
    b,r = math.e**b, math.e**r
    q   = 0 if acq=="xploit" else (1 if acq=="xplor" else 1-p)
    return (b + r*q) / abs(b*q - r + 1/big)
  def _guess(row):
    return _acquire(like(best,row,n,2), like(rest,row,n,2), the.Acq, n/the.Stop)

  random.shuffle(data._rows)
  n          = the.start
  todo       = data._rows[n:]
  bestrest   = clone(data, data._rows[:n])
  done       = ysort(bestrest)
  cut        = round(n**the.Guess)
  best       = clone(data, done[:cut])
  rest       = clone(data, done[cut:])
  while len(todo) > 2 and n < the.Stop:
    n       += 1
    hi, *lo = sorted(todo[:the.Few*2], key=_guess, reverse=True)
    todo    = lo[:the.Few] + todo[the.Few*2:] + lo[the.Few:]
    add(bestrest, add(best, hi))
    best._rows = ysort(bestrest)
    if len(best._rows) >= round(n**the.Guess):
       add(rest, sub(best,  best._rows.pop(-1)))
  return o(best=best, rest=rest, test=todo)

### Distance -----------------------------------------------------------------------
# Return pth root of the sum of the distances raises to p.
def minkowski(src):
  d, n = 0, 1/big
  for x in src:
    n += 1
    d += x**the.p
  return (d / n)**(1 / the.p)

# Distance to heaven.
def ydist(data, row):
  return minkowski(abs(norm(c, row[c.at]) - c.heaven) for c in data.cols.y)

def ysort(data,rows=None):
   return sorted(rows or data._rows, key=lambda row: ydist(data,row))

def xdist(data, row1, row2):
  def _aha(col,u,v):
    if u=="?" and v=="?": return 1
    if col.it is Sym: return u!=v
    u = norm(col,u)
    v = norm(col,v)
    u = u if u != "?" else (0 if v > .5 else 1)
    v = v if v != "?" else (0 if u > .5 else 1)
    return abs(u - v)

  return minkowski(_aha(c, row1[c.at], row2[c.at]) for c in data.cols.x)

# K-means plus plus: k points, usually D^2 distance from each other.
def kpp(data, k=None, rows=None):
  k = k or the.Stop
  row,  *rows = shuffle(rows or data._rows)
  some, rest  = rows[:the.Few], rows[the.Few:]
  centroids   = [row]
  for _ in range(1, k):
    dists = [min(xdist(data,x,y)**2 for y in centroids) for x in some]
    r     = random.random() * sum(dists)
    for j, d in enumerate(dists):
      r -= d
      if r <= 0:
        centroids.append(some.pop(j))
        break
  return centroids
```

```python
### Tree ------------------------------------------------------------------------
ops = {'<=' : lambda x,y: x <= y,
       "==" : lambda x,y: x == y,
       '>'  : lambda x,y: x >  y}

def selects(row, op, at, y): x=row[at]; return  x=="?" or ops[op](x,y)

def cuts(col,rows,Y,Klass):
  def _sym(sym):
    n,d = 0,{}
    for row in rows:
      if (x := row[sym.at]) != "?":
        n = n + 1
        d[x] = d[x] if x in d else Klass()
        add(d[x], Y(row))
    return o(div = sum(c.n/n * spread(c) for c in d.values()),
             hows = [("==",sym.at, k) for k,v in d.items()])

  def _num(num):
    out, b4, lhs, rhs = None, None, Klass(), Klass()
    xys = [(r[num.at], add(rhs, Y(r))) for r in rows if r[num.at] != "?"]
    xpect = rhs.sd
    for x, y in sorted(xys, key=lambda xy: xy[0]):
      if x != b4:
        if the.leaf <= lhs.n <= len(xys) - the.leaf:
          tmp = (lhs.n * lhs.sd + rhs.n * rhs.sd) / len(xys)
          if tmp < xpect:
            xpect, out = tmp, [("<=", num.at, b4), (">", num.at, b4)]
      add(lhs, sub(rhs,y))
      b4 = x
    if out:
      return o(div=xpect, hows=out)

  return (_sym if col.it is Sym else _num)(col)

def tree(data, Klass=Num, Y=None, how=None):
  Y          = Y or (lambda row: ydist(data,row))
  data.kids = []
  data.how  = how
  data.ys   = Num(Y(row) for row in data._rows)
  if data.n >= the.leaf:
    tmp = [x for c in data.cols.x if (x := cuts(c,data._rows,Y,Klass=Klass))]
    if tmp:
      for how1 in sorted(tmp, key=lambda cut: cut.div)[0].hows:
        rows1 = [row for row in data._rows if selects(row, *how1)]
        if the.leaf <= len(rows1) < data.n:
          data.kids += [tree(clone(data,rows1), Klass, Y, how1)]
  return data

def nodes(data1, lvl=0, key=None):
  yield lvl, data1
  for data2 in (sorted(data1.kids, key=key) if key else data1.kids):
    yield from nodes(data2, lvl + 1, key=key)

def leaf(data1,row):
  for data2 in data1.kids or []:
    if selects(row, *data2.how):
      return leaf(data2, row)
  return data1

def show(data, key=lambda z:z.ys.mu):
  stats = data.ys
  win = lambda x: 100-int(100*(x-stats.lo)/(stats.mu - stats.lo))
  print(f"{'d2h':>4} {'win':>4} {'n':>4}  ")
  print(f"{'----':>4} {'----':>4} {'----':>4}  ")
  for lvl, node in nodes(data, key=key):
    leafp = len(node.kids)==0
    post = ";" if leafp else ""
    xplain = ""
    if lvl > 0:
      op,at,y = node.how
      xplain = f"{data.cols.all[at].txt} {op} {y}"
    print(f"{node.ys.mu:4.2f} {win(node.ys.mu):4} {node.n:4}   {(lvl-1) * '| '}{xplain}" + post)


### Utils ------------------------------------------------------------------------
# Shortcuts
big = 1E32
pick = random.choice
picks = random.choices

# Shiffle
def shuffle(lst):
  random.shuffle(lst)
  return lst

# Bulk inits
def adds(i, src):
  [add(i,x) for x in src]; return i

# Read iterators.
def doc(file):
  with open(file, 'r', newline='', encoding='utf-8') as f:
    for line in f: yield line

def lines(s):
 for line in s.splitlines(): yield line.strip()

def csv(src):
  for line in src:
    if line: yield [atom(s) for s in line.strip().split(',')]

# String to thing
def atom(x):
  for what in (int, float):
    try: return what(x)
    except Exception: pass
  x = x.strip()
  y = x.lower()
  return (y == "true") if y in ("true", "false") else x

# Thing to strimg
def cat(v):
  it = type(v)
  inf = float('inf')
  if it is list:  return "{" + ", ".join(map(cat, v)) + "}"
  if it is float: return str(int(v)) if -inf<v<inf and v==int(v) else f"{v:.3g}"
  if it is dict:  return cat([f":{k} {cat(w)}" for k, w in v.items()])
  if it in [type(abs), type(cat)]: return v.__name__ + '()'
  return str(v)

# Simple class. Easy inits. Can print itself.
class o:
  __init__ = lambda i, **d: i.__dict__.update(**d)
  __repr__ = lambda i: cat(i.__dict__)


### Demos ------------------------------------------------------------------------
#### Utils
def eg__the(_):
  ":    show config"
  print(the)

def eg__str(_):
  ":    show string --> csv"
  s,n = 0,0
  for row in csv(lines(EXAMPLE)):
    assert len(row)==5
    if type(row[0]) is str: s += 1
    if type(row[0]) in [int,float]: n += 1
  assert s==1 and n==100

#### Create and Update
def eg__nums(_):
  ":    nums --> summary"
  num=Num([random.gauss(10,2) for _ in range(1000)])
  assert 10 < mid(num) < 10.2 and 2 < spread(num) < 2.1

def eg__sym(_):
  ":    chars --> summary"
  sym = Sym("aaaabbc")
  assert "a"==mid(sym) and 1.3 < spread(sym) < 1.4

def eg__cols(_):
  ":    List[str] --> columns"
  cols = Cols(["name","Age","Salary+"])
  for what,lst in (("x", cols.x), ("y",cols.y)):
    print("\n"+what)
    [print("\t"+cat(one)) for one in lst]

def eg__data(file):
  ":    csv data --> data"
  print(data.n)
  print("X"); [print(" ",col) for col in data.cols.x]
  print("Y"); [print(" ",col) for col in data.cols.y]

#### Query
def eg__addSub(file):
  ":   demo row addition / deletion"
  data1 = Data(csv(doc(file) if file else lines(EXAMPLE)))
  data2 = clone(data1)
  for row in data1._rows:
    add(data2,row)
    if len(data2._rows)==100:
      mids    = mid(data2)
      spreads = spread(data2)
  for row in data1._rows[::-1]:
    if len(data2._rows)==100:
      assert mids    == mid(data2)
      assert spreads == spread(data2)
      return
    sub(data2, row)

#### Distance
def eg__dist(file):
  ":    demo data distances"
  data = Data(csv(doc(file) if file else lines(EXAMPLE)))
  row1 = data._rows[0]
  assert all(0 <= xdist(data,row1,row2) <= 1 for row2 in data._rows)
  assert all(0 <= ydist(data,row2) <= 1      for row2 in data._rows)
  lst = ysort(data)
  [print(round(ydist(data,row),2), row) for row in lst[:3] + lst[-3:]]

def eg__line(file):
  ":    demo data distances"
  data = Data(csv(doc(file) if file else lines(EXAMPLE)))
  one = lambda: sorted([ydist(data,row) for row in kpp(data)])[0]
  print(cat(sorted([one() for _ in range(20)])))

#### Bayes
def eg__bayes(file):
  ":    demo bayes"
  data = Data(csv(doc(file) if file else lines(EXAMPLE)))
  print(cat(sorted([like(data,row,2,1000) for row in data._rows[::10]])))

def eg__lite(file):
  ":    demo active elarning"
  data = Data(csv(doc(file) if file else lines(EXAMPLE)))
  b4   = [ydist(data, row) for row in data._rows][::8]
  now  = [ydist(data, acquires(data).best._rows[0]) for _ in range(12)]
  print(o(b4=sorted(b4)))
  print(o(now=sorted(now)))

#### Tree
def eg__tree(file):
  ":    demo active elarning"
  data = Data(csv(doc(file) if file else lines(EXAMPLE)))
  show(tree(data))

#### Control
def eg__all(_):
  ":    run all demos"
  for s,fn in globals().items():
    if s.startswith("eg_") and s!="eg__all":
      print(f"\n# {'-'*78}\n# {s}\n")
      run(fn)

def eg_h(_):
  ":    show help"
  print("\n"+__doc__);
  for s,fn in globals().items():
    if s.startswith("eg_"):
      print(f" {s[2:].replace("_","-"):6s} {fn.__doc__[1:]}")
```

```
### Start-up -------------------------------------------------------------------
def cli(d):
  for k, v in d.items():
    for c, arg in enumerate(sys.argv):
      if arg == "-" + k[0]:
        d[k] = atom("False" if str(v) == "True" else (
                    "True" if str(v) == "False" else (
                    sys.argv[c + 1] if c < len(sys.argv) - 1 else str(v))))

def run(fn,x=None):
  try:
    random.seed(the.rseed)
    fn(x)
  except Exception as e:
    tb = traceback.format_exc().splitlines()[4:]
    return sys.stdout.write("\n".join(tb) + "\n")

the = o(**{m[1]: atom(m[2])
        for m in re.finditer(r"-\w+\s+(\w+)[^\(]*\(\s*([^)]+)\s*\)", __doc__)})

if __name__ == "__main__":
  cli(the.__dict__)
  for i,s in enumerate(sys.argv):
    if fn := globals().get("eg" + s.replace("-", "_")):
      run(fn, None if i == len(sys.argv) - 1 else atom(sys.argv[i+1]))
```