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

BIG=1e32
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):
   data = obj(it=Data, rows=[], n=0, cols=None)
   [add(data,row) for row in rows or []]
   return data

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

### Functions ----------------------------------------------------------------
def add(it,v):
  if v=="?": return v
  it.n += 1
  if   Sym  is it.it: it.has[v] = 1 + it.has.get(v,0)
  elif Num  is it.it: d = v - it.mu; it.mu += d/it.n; it.m2 += d*(v - it.mu)
  elif Data is it.it:
    if    it.cols: it.rows.append([add(col,v[col.at]) for col in it.cols.all])
    else: it.cols = Cols(v); it.n=0
  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 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))

## Cutting -------------------------------------------------------------------
def score(num): return num.mu + sd(num) / (sqrt(num.n) + 1e-32)

def cut(data, rows):
  all_bins = (b for col in data.cols.x for b in cuts(col, rows, data))
  return min(all_bins, key=lambda b: score(b.y), default=None)

def cuts(col, rows, data):
  d, xys = {}, [(r[col.at], disty(data, r)) for r in rows if r[col.at]!="?"]
  for x, y in sorted(xys):
    k = x if Sym is col.it else floor(BINS * norm(col, x))
    if k not in d: d[k] = obj(at=col.at, txt=col.txt, xlo=x, xhi=x, y=Num())
    add(d[k].y, y)
    d[k].xhi = x
  return _complete(col, sorted(d.values(), key=lambda b: b.xlo))

def _complete(col, lst):
  if Num is col.it:
    for i, b in enumerate(lst):
      b.xlo = lst[i-1].xhi if i > 0 else -BIG
      b.xhi = lst[i+1].xlo if i < len(lst)-1 else BIG
  return lst

### Main ---------------------------------------------------------------------
def select(rule, row):
  if (x:=row[rule.at]) == "?" or rule.xlo == rule.xhi == x: return True
  return rule.xlo <= x < rule.xhi

def xai(data):
  print(o(bins=BINS))
  print(*data.cols.names)
  def go(rows, lvl=0, prefix=""):
    ys = Num(); rows.sort(key=lambda row: add(ys, disty(data, row)))
    print(f"{o(rows[len(rows)//2])}: {o(mu=ys.mu, n=ys.n, sd=sd(ys)):25s} {prefix}")
    if rule := cut(data, rows):
      now = [row for row in rows if select(rule, row)]
      if 4 < len(now) < len(rows):
        go(now, lvl + 1, f"{"|..  " * lvl}{rule.txt} {o(rule.xlo)}..{o(rule.xhi)} ")
  go(data.rows, 0)

def six(data):
  seen = clone(data)
  unique=set()
  def go(rows, lvl=0, prefix=""):
    ys = Num(); rows.sort(key=lambda row: add(ys, disty(data, row)))
    some = shuffle(rows)[:BUDGET]
    for row in some:
      add(seen,row)
      unique.add(tuple(row))
    if rule := cut(seen, some):
      now = [row for row in rows if select(rule, row)]
      if 4 < len(now) < len(rows):
        return go(now, lvl + 1, f"{"|..  " * lvl}{rule.txt} {o(rule.xlo)}..{o(rule.xhi)} ")
    return int(100*ys.mu)
  return go(data.rows, 0)


```

```python
## 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 "
  global SEED; SEED=coerce(s); random.seed(SEED)

def go_b(s):
  "-b [5]       set number of BINS used on discretization"
  global BINS; BINS=coerce(s)

def go_B(s):
  "-B [30]      set BUDGET for rows labelled each round"
  global BUDGET;  BUDGET=coerce(s)

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"
  for col in Data(csv(file)).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 = cline(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))
  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(SEED)
  for b in [5,10,20,30]:
    go_B(b)
    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()
```