min.py


```python
import ast,math,sys,random
from math import sqrt,exp
from types import SimpleNamespace as obj

### 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(at,name) for at,name 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))

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

### Functions ------------------------------------------------------------
def adds(src, it):
  it = it or Num(); [add(it, x) for x in src]; return it

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

def bestCut(best,rest):
  cuts= [(c1,c2,cut(c1,c2)) for c1,c2 in zip(best.cols.x, rest.cols.x)]
  c1,c2,where = max(cuts,key=lambda x: -win(*x))
  return c1.at,c1.txt,where, win(c1,c2,where)

def cut(col1,col2):
  if Sym is col1.it:
    return max(col1.has, key=lambda k: win(col1,col2,k))
  w1, w2 = 1/sd(col1), 1/sd(col2)
  return (w1 * col1.mu + w2 * col2.mu) /(w1 + w2 + 1e-32)

def win(best:Col, rest:Col,v): # (best,rest:Col) n frosplitNum(best,mrest)
  if Sym is best.it:
    b,r = best.has.get(v,0)/(best.n + 1e-32), rest.has.get(v,0)/(rest.n + 1e-32)
  else:
    b,r = norm(best,v), norm(rest,v)
    if best.mu > rest.mu: b,r= 1-b, 1-r
  return b*b/(r + 1e-32)

def norm(num,n): return 1 / (1 + exp(-1.7 * (n - num.mu)/(sd(num) + 1e-32)))

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))

## Lib -------------------------------------------------------------------
def o(x):
  if type(x) is float: return str(int(x)) if x == int(x) else f"{x:.2f}"
  if type(x) is dict: return "{"+' '.join(f":{k} {o(x[k])}" for k in x)+"}"
  if type(x) is tuple: return "("+','.join(o(y) for y in x)+")"
  if type(x) is list: return "["+'.'.join(o(y) for y in x)+"]"
  if type(x) is obj: return o(x.__dict__)
  if type(x) == type(o): return x.__name__
  return str(x)

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) for x in l.split(",")]

def buffer(src, k=100):
  cache = []
  for n,x in enumerate(src):
    if n==0: yield x
    else:
      cache += [x]
      if len(cache) > k:
        random.shuffle(cache); yield from cache
        cache=[]
  if cache: random.shuffle(cache); yield from cache

def main(data)
  ys = adds([disty(data,row) for row in data.rows])
  if data.n < 4:
    print(ys.mu)
  else:
    best, rest = clone(data), clone(data)
    for row in data.rows:
      add(best if disty(data,row) <= ys.mu else rest, row)
    print(o(bestCut(best,rest)))
    main(best)

### Main ----------------------------------------------------------------
file = sys.argv[1] if len(sys.argv)>1 else "data.csv"
random.seed( float( sys.argv[2] if len(sys.argv)>2 else 1) )
all, best, rest= Data(), Data(), Data()
ys = Num()
for n,row in enumerate(buffer(csv(file),k=20)):
  if n==0:
    add(all, add(rest, add(best, row)))
  else:
    add(all, row)
    y = add(ys, disty(all, row))
    add(best if y <= ys.mu else rest, row)
    if not (n % 20):
      print(o(bestCut(best,rest)))
```