```
1. PROGRAM OUTPUT
================================================================================
=====
weather2_EWD.out...
b[1]= 0.33
b[2]= 0.67

    outlook,  $temperature,     $humidity,         windy,         =play,  #
      sunny,             85,             85,         FALSE,            no,  #
      sunny,             80,             90,          TRUE,            no,  #
   overcast,             83,             86,         FALSE,           yes,  #
      rainy,             70,             96,         FALSE,           yes,  #
      rainy,             68,             80,         FALSE,           yes,  #
      rainy,             65,             70,          TRUE,            no,  #
   overcast,             64,             65,          TRUE,           yes,  #
      sunny,             72,             95,         FALSE,            no,  #
      sunny,             69,             70,         FALSE,           yes,  #
      rainy,             75,             80,         FALSE,           yes,  #
      sunny,             75,             70,          TRUE,           yes,  #
   overcast,             72,             90,          TRUE,           yes,  #
   overcast,             81,             75,         FALSE,           yes,  #
      rainy,             71,             91,          TRUE,            no,  #

    outlook,   temperature,      humidity,         windy,         =play,  #
      sunny,              3,              2,         FALSE,            no,  #
      sunny,              3,              3,          TRUE,            no,  #
   overcast,              3,              3,         FALSE,           yes,  #
      rainy,              1,              3,         FALSE,           yes,  #
      rainy,              1,              2,         FALSE,           yes,  #
      rainy,              1,              1,          TRUE,            no,  #
   overcast,              1,              1,          TRUE,           yes,  #
      sunny,              2,              3,         FALSE,            no,  #
      sunny,              1,              1,         FALSE,           yes,  #
      rainy,              2,              2,         FALSE,           yes,  #
      sunny,              2,              1,          TRUE,           yes,  #
   overcast,              2,              3,          TRUE,           yes,  #
   overcast,              3,              1,         FALSE,           yes,  #
      rainy,              2,              3,          TRUE,            no,  #

================================================================================
=====
weather2_Gaussian.out
b[1]= -0.43
b[2]= 0.43

    outlook,  $temperature,     $humidity,         windy,         =play,  #
```



```
      sunny,             85,             85,         FALSE,            no,  #
      sunny,             80,             90,          TRUE,            no,  #
   overcast,             83,             86,         FALSE,           yes,  #
      rainy,             70,             96,         FALSE,           yes,  #
      rainy,             68,             80,         FALSE,           yes,  #
      rainy,             65,             70,          TRUE,            no,  #
   overcast,             64,             65,          TRUE,           yes,  #
      sunny,             72,             95,         FALSE,            no,  #
      sunny,             69,             70,         FALSE,           yes,  #
      rainy,             75,             80,         FALSE,           yes,  #
      sunny,             75,             70,          TRUE,           yes,  #
   overcast,             72,             90,          TRUE,           yes,  #
   overcast,             81,             75,         FALSE,           yes,  #
      rainy,             71,             91,          TRUE,            no,  #

    outlook,   temperature,      humidity,         windy,         =play,  #
      sunny,              3,              2,         FALSE,            no,  #
      sunny,              3,              3,          TRUE,            no,  #
   overcast,              3,              2,         FALSE,           yes,  #
      rainy,              1,              3,         FALSE,           yes,  #
      rainy,              1,              2,         FALSE,           yes,  #
      rainy,              1,              1,          TRUE,            no,  #
   overcast,              1,              1,          TRUE,           yes,  #
      sunny,              2,              3,         FALSE,            no,  #
      sunny,              1,              1,         FALSE,           yes,  #
      rainy,              2,              2,         FALSE,           yes,  #
      sunny,              2,              1,          TRUE,           yes,  #
   overcast,              2,              3,          TRUE,           yes,  #
   overcast,              3,              1,         FALSE,           yes,  #
      rainy,              2,              3,          TRUE,            no,  #

2. SOURCE CODES
================================================================================
< FILE discrete.py>
import labels
import reader
import tablestr

def discrete(table, t, bins):
    tables = {}
    #breaks = labels.ewdbreaks; label = labels.ewdlablef
    breaks = labels.gbreaks; label = labels.globalf
    b = {}
    breaks(b)
    newNames = labels.discreteNames(table.name, table.num)
```

```
        ntable = tablestr.Table()
90      reader.makeTable(newNames, ntable)
        discrete1(table, ntable, bins, b[bins], label)
        print 'b[1]=', b[3][0]
        print 'b[2]=', b[3][1]
        tables[0] = table
95      t1 = 'D_'+ str(t)
        tables[t1] = ntable
        return tables

    def discrete1(table, ntable, bins, b, label):
100     for d in range(len(table.data[0])):
            a = []
            for k in range(len(table.data)):
                val = table.data[k][d]
                if val != '?':
105                 if k in table.num:
                        k = table.num.index(k)
                        val = label(k, float(val), bins, b, table)
                a += [str(val)]
            reader.addRow(a, ntable)
110
    ================================================================================
    < FILE labels.py>
    import re
    def discreteNames(names, num):
115     newNames = []
        for k in range(len(names)):
            tmp = names[k]
            if k in num:
                tmp = re.sub(r'\$*\-*\+*','',tmp)
120         newNames += [tmp]
        return newNames

    ### Ewdbreaks
    def ewdbreaks(a):
125   breaks0(a,"                                    0.50
            ")
      breaks0(a,"                                0.33      0.67
            ")
      breaks0(a,                                "0.25 0.50 0.75
            ")
      breaks0(a,"                            0.20 0.40      0.60 0.80
            ")
      breaks0(a,"                            0.17 0.33 0.50 0.67 0.83
            ")
130   breaks0(a,"                        0.14 0.29 0.43      0.57 0.71 0.86
            ")
      breaks0(a,"                        0.12 0.25 0.38 0.50 0.62 0.75 0.88
            ")
      breaks0(a,"                  0.11 0.22 0.33 0.44      0.56 0.67 0.78 0.89
            ")
      breaks0(a,"                  0.10 0.20 0.30 0.40 0.50 0.60 0.70 0.80 0.90
            ")
      breaks0(a,"            0.09 0.18 0.27 0.36 0.45      0.55 0.64 0.73 0.82 0.91
            ")
135   breaks0(a,"            0.08 0.17 0.25 0.33 0.42 0.50 0.58 0.67 0.75 0.83 0.92
            ")
      breaks0(a,"       0.08 0.15 0.23 0.31 0.38 0.46      0.54 0.62 0.69 0.77 0.85
    0.92    ")
      breaks0(a,"       0.07 0.14 0.21 0.29 0.36 0.43 0.50 0.57 0.64 0.71 0.79 0.86
    0.93    ")
      breaks0(a,"  0.07 0.13 0.20 0.27 0.33 0.40 0.47      0.53 0.60 0.67 0.73 0.80
    0.87 0.93")

140 ### Gbreaks
    def gbreaks(a):
      breaks0(a,"                      0                   ")
      breaks0(a,"                -0.43   0.43              ")
      breaks0(a,"                -0.67 0 0.67              ")
145   breaks0(a,"            -0.84 -0.25   0.25 0.84       ")
      breaks0(a,"            -0.97 -0.43 0 0.43 0.97       ")
      breaks0(a,"      -1.07 -0.57 -0.18   0.18 0.57 1.07  ")
```



```
      breaks0(a,"      -1.15 -0.67 -0.32 0 0.32 0.67 1.15     ")
      breaks0(a,"-1.22 -0.76 -0.43 -0.14   0.14 0.43 0.76 1.22")
150   breaks0(a,"-1.28 -0.84 -0.52 -0.25 0 0.25 0.52 0.84 1.28")

    def breaks0(a, str):
        tmp = str.split()
        n = 1 + len(tmp)
155     t = []
        for i in tmp: t += [float(i)]
        a[n] = t

    def globalf(k, val, bins, b, table):
160     val = (val - table.mu[k]) / table.sd[k]
        for i in range(bins-1):
            if val <= b[i]: return i+1
        return bins
    def ewdlablef(k, val, bins, b, table):
165     val  = (val - float(table.lo[k])) / (float(table.hi[k]) - float(table.lo[k])
     + 0.00001)
        for i in range(bins-1):
            if val <= b[i]: return i+1
        return bins
    ================================================================================
170 < FILE tablestr.py>
    import lib
    class Table:
        def __init__(self):
            self.data = []     #data[[col1,...],[col2,...]]
175         self.name = []     #name of i-th column
            self.order = []    #order of the col
            self.nump = []     #is i-th column numeric?
            self.wordp = []    #is i-th column non-numeric?
            self.indep = []    #list of indep columns
180         self.dep = []      #list of dep columns
            self.less = []     #numeric goal to be minimized
            self.more = []     #numeric goal to be maximized
            self.klass = []    #non-numeric goal
            self.term = []     #non-numeric non-goal
185         self.num = []      #numeric non-goal
            # for all cols
            self.n = []        #count of things in this col
            # for wordp columns:
            self.count = []    #count of each word
190         self.mode = []     #most common word
            self.most = []     #count of most common word
            # for nump columns:
            self.hi = []       #upper bound
            self.lo = []       #lower bound
195         self.mu = []       #mean
            self.m2 = []       #sum of all nums
            self.sd = []       #standard deviation# -*- coding: utf-8 -*-
            # table printing format
            self.CONVFMT = '%4.2f'
200
    def centroid(table):
        "update the mode and most values for wordp type cols or update the mean and
    sd values for nump cols"
        rows = [[],[]]
        for c in range(len(table.name)):
205         s = table.mode[table.wordp.index(c)] if c in table.wordp else table.CONV
    FMT%table.mu[table.nump.index(c)]
            rows[0].append(str(s))
            if table.n[c] == '0':
                s = 0.0
            else:
210             s = float(table.most[table.wordp.index(c)])/table.n[c] if c in table
    .wordp else table.sd[table.nump.index(c)]
            rows[1].append(str(table.CONVFMT%s))
        return rows

    def tableprint(table, stats=''):
215     "print table on the console"
        print ' '
```

```
        if stats != '': table.CONVFMT = stats
        print(' ' + lib.rowprint(table.name)+ '  # notes'.ljust(10))
        print('#' + lib.rowprint(centroid(table)[0]) + '  # expected'.ljust(10))
        print('#' + lib.rowprint(centroid(table)[1]) + '  # certainty'.ljust(10))

        for j in range(len(table.data[0])):
            line = []
            for i in range(len(table.data)):
                line.append(table.data[i][j])
            print(' ' + lib.rowprint(line)+ '  #'.ljust(10))

    def tableprint_txt(table, f, stats=''):
        "print table on the indicated txt file with table name"
        f.write('\n')
        #f.write('\n' +tablename + '\n'*2)
        if stats != '': table.CONVFMT = stats
        f.write(' ' + lib.rowprint(table.name)+ '  # notes'.ljust(10) + '\n')
        f.write('#' + lib.rowprint(centroid(table)[0]) + '  # expected'.ljust(10) +
    '\n')
        f.write('#' + lib.rowprint(centroid(table)[1]) + '  # certainty'.ljust(10) +
     '\n')
        for j in range(len(table.data[0])):
            line = []
            for i in range(len(table.data)):
                line.append(table.data[i][j])
            f.write(' ' + lib.rowprint(line)+ '  #'.ljust(10) + '\n')
    ================================================================================
    < FILE reader.py>
    import re
    import tablestr
    def readcsv(filename, table):
        "read in data from csv and create a table"
        FS = ','                    #define field separator
        f = open(filename)
        seen  = 0
        while True:
            str = line(f)
            if str == -1:
                if seen == 0: print("WARNING: empty or missing file")
                return -1
            a = str.split(FS)        #compute the number of attributes in table
            if len(a) > 1:
                if seen: addRow(a, table)
                else: makeTable(a, table)
                seen += 1

    def line(f):
        "get one line data (without comments and whitespace)"
        str = f.readline()
        if not str: return -1             #readline finds nothing, output error
        else:
            str = "".join(str.split())    #kill whitespace
            str = re.sub(r'#.*','',str)   #kill comments
            if len(str) >= 1 and str[-1] == ',': return str + line(f)
            else: return str

    def makeTable(a, table):
        "read table titles and set all corresponding parameters"
        c = 0
        for ite in range(len(a)):
            if a[ite][0] == '?': continue  #the col with '?' is ignored
            table.order.append(ite)
            x = a[ite]
            table.name.append(x)
            isNum = 1
            if x.find('=') != -1:
                table.dep.append(c)
                table.klass.append(c)
                isNum = 0
            elif x.find('+') != -1:
                table.dep.append(c)
                table.more.append(c)
            elif x.find('-') != -1:
```



```
                table.dep.append(c)
                table.less.append(c)
            elif x.find('$') != -1:
                table.indep.append(c)
                table.num.append(c)
            else:
                table.indep.append(c)
                table.term.append(c)
                isNum = 0
            table.n.append('0')
            if isNum:
                table.nump.append(c)
                table.hi.append(-1.0*10**32)
                table.lo.append(10.0**32)
                table.mu.append(0)
                table.m2.append(0)
                table.sd.append(0)
            else:
                table.wordp.append(c)
                table.most.append(0)
                table.count.append({})
                table.mode.append('')
            c += 1
        for i in range(c): table.data.append([])

    def addRow(a, table):
        "add a row of data to the table"
        for c in range(len(table.name)):
            f = table.order[c]
            x = a[f]
            table.data[c].append(x)
            if x.find('?') == -1:
                table.n[c] = int(table.n[c]) + 1
                if c in table.wordp:
                    k = table.wordp.index(c)
                    if table.count[k].has_key(x): table.count[k][x] += 1
                    else: table.count[k][x] = 1
                    new = table.count[k][x]
                    if new > table.most[k]:
                        table.mode[k] = x
                        table.most[k] = new
                else:
                    k = table.nump.index(c)
                    if float(x) > float(table.hi[k]): table.hi[k] = x
                    if float(x) < float(table.lo[k]): table.lo[k] = x
                    delta = float(x) - table.mu[k]
                    table.mu[k] += delta/table.n[c]
                    table.m2[k] += delta*(float(x) - table.mu[k])
                    if table.n[c] > 1:
                        table.sd[k] = (table.m2[k]/(table.n[c] - 1))**0.5
                c += 1

    def klasses(table):
        "generate a set of tables based on different classes"
        if len(table.klass) == 0:
            print "No labeled classes in the given data set"
            return -1
        # assume there is only one class feature in the data set
        data = table.data[table.klass[0]]
        classnames = []
        for s in data:
            if s not in classnames:
                classnames.append(s)
        tables = klass1(table, classnames, data)
        tables['0'] = table
        tables['names'] = classnames
        return tables

    def klass1(table, classnames, data):
        tables = {}
        for s in classnames:
            tables[s] = tablestr.Table()
            makeTable(table.name, tables[s])
```

```
        for i in range(len(data)):
            if s == data[i]:
                a = []
                for j in range(len(table.order)):
                    a.append(table.data[j][i])
                addRow(a, tables[s])
    return tables
================================================================================
< FILE main.py>
import reader
import tablestr
import discrete

if __name__ == "__main__":
    filename = 'data/weather2.csv'
    table = tablestr.Table()             #create raw data structure
    reader.readcsv(filename,table )      #read the .csv data set
    f = '%4.2f'                          #set the formatting for the output
    #tables = reader.klasses(table)
    #tablestr.tableprint(tables['0'], f)
    bins = 3
    t = 0
    tables = discrete.discrete(table, t, bins)
    tablestr.tableprint(tables[0], f)
    tablestr.tableprint(tables['D_'+str(t)], f)
```