```
1. PROGRAM OUTPUT (txt file)

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                    weather1.csv/Table['0']

        outlook,    -$humidity,         windy,         =play,  # notes
#          sunny,         81.83,         FALSE,           yes,  # expected
#          0.36,         10.01,          0.57,          0.64,  # certainty
          sunny,            90,         FALSE,            no,  #
          sunny,            90,          TRUE,            no,  #
       overcast,            86,         FALSE,           yes,  #
          rainy,            96,         FALSE,           yes,  #
          rainy,            80,         FALSE,           yes,  #
          rainy,             ?,          TRUE,            no,  #
       overcast,            65,          TRUE,           yes,  #
          sunny,             ?,         FALSE,            no,  #
          sunny,            70,         FALSE,           yes,  #
          rainy,            80,         FALSE,           yes,  #
          sunny,            70,          TRUE,           yes,  #
       overcast,            90,          TRUE,           yes,  #
       overcast,            75,         FALSE,           yes,  #
          rainy,            90,          TRUE,            no,  #

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                    weather1.csv/Table['no']

        outlook,    -$humidity,         windy,         =play,  # notes
#         sunny,         90.00,          TRUE,            no,  # expected
#          0.60,          0.00,          0.60,          1.00,  # certainty
          sunny,            90,         FALSE,            no,  #
          sunny,            90,          TRUE,            no,  #
          rainy,             ?,          TRUE,            no,  #
          sunny,             ?,         FALSE,            no,  #
          rainy,            90,          TRUE,            no,  #

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                    weather1.csv/Table['yes']

        outlook,    -$humidity,         windy,         =play,  # notes
#      overcast,         79.11,         FALSE,           yes,  # expected
#          0.44,         10.22,          0.67,          1.00,  # certainty
       overcast,            86,         FALSE,           yes,  #
          rainy,            96,         FALSE,           yes,  #
          rainy,            80,         FALSE,           yes,  #
       overcast,            65,          TRUE,           yes,  #
          sunny,            70,         FALSE,           yes,  #
          rainy,            80,         FALSE,           yes,  #
          sunny,            70,          TRUE,           yes,  #
       overcast,            90,          TRUE,           yes,  #

2. SOURCE CODES
================================================================================
File : <tablestr.py>

import lib
class Table:
    def __init__(self):
        self.data = []     #data[[col1,...],[col2,...]]
        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
        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
        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
```

```
        self.mode = []     #most common word
        self.most = []     #count of most common word
        # for nump columns:
        self.hi = []       #upper bound
        self.lo = []       #lower bound
        self.mu = []       #mean
        self.m2 = []       #sum of all nums
        self.sd = []       #standard deviation# -*- coding: utf-8 -*-
        # table printing format
        self.CONVFMT = '%06d'

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)):
        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))
        s = float(table.most[table.wordp.index(c)])/table.n[c] if c in table.wor
dp else table.sd[table.nump.index(c)]
        rows[1].append(str(table.CONVFMT%s))
    return rows

def tableprint(table, stats=''):
    "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, tablename, stats=''):
    "print table on the indicated txt file with table name"
    f.write('\n' +'* '*40+'\n'+' '*20+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*10**32)
            table.lo.append(10**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 : <lib.py >

def rowprint(a):
    "get a row with some format"
    max = len(a)
    line = ''
    for j in range(max):
        line += (a[j] + ',').rjust(15)
    return line
===============================================================================

File : <main.py>

import reader
import tablestr
if __name__ == "__main__":
    filename = 'data/weather1.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
    filename = 'output/table1.txt'       #define output txt file
    out = file(filename, 'w')
    tables = reader.klasses(table)
    tablestr.tableprint(tables['0'], f)
    tablestr.tableprint_txt(tables['0'], out, "weather1.csv/Table['0']", f)
    for h in tables['names']:
        tablestr.tableprint(tables[h], f)
        tablestr.tableprint_txt(tables[h], out, "weather1.csv/Table['"+h+"']", f
)
===============================================================================

3. CODE ILLUSTRATE
a. Tables are stored in a dictionary structure:
   - tables = {'0':table1, 'yes':table2, 'no':table3, 'names':[...]}
   - keys are the classes'names
   - data are the objects of the predefined Table class
   - tables['0'] is the original read-in table
   − tables[’yes’] and tables[’no’] are the splited tables based on different cl
asses
   − tables[’names’] is the list structure that indicates all the types for the
dependent variable

b. Generated tables can be printed on both screen or indicated txt file
```