```
    weather
    zeror
    WARNING: empty or missing file
    71.43 57.14 57.14 71.43
5   soybean
    zeror
    WARNING: empty or missing file
    11.73 11.44 11.44 12.32
```

```
    #Info for table
    #
    csvindex = -1 #initialized to -1 as lists start at zero
    colname = {k: [] for k in range(1)} #stores dict of names of columns
 5  data = {k: [] for k in range(1)} #stores dict of list of lists containing each r
    ow
    test = [] #stores test data
    #
    #metadata
    #
10  order = {k:dict.fromkeys(colname) for k in range(1)} #stores colnames and index
    of column in csv
    klass = {k: [] for k in range(1)} #dict of list of klass columns
    more = {k: [] for k in range(1)} #dict of list of more columns
    less = {k: [] for k in range(1)} #dict of list of less columns
    num = {k: [] for k in range(1)} #dict of list of num columns
15  term = {k: [] for k in range(1)} #dict of list of term columns
    dep = {k: [] for k in range(1)} #dict of list of dependent columns
    indep = {k: [] for k in range(1)} #dict of list of independent columns
    nump = {k: [] for k in range(1)} #dict of list containing nump column names
    wordp = {k: [] for k in range(1)} #dict of list containing wordp column names
20  #
    #for nump values
    #
    hi = {k:dict.fromkeys(nump) for k in range(1)} #dictionary containing highest va
    lues of nump columns
    lo = {k:dict.fromkeys(nump) for k in range(1)} #dictionary containing lowest val
    ues of nump columns
25  mu = {k:dict.fromkeys(nump) for k in range(1)} #dictionary containing mean value
    s of nump columns
    m2 = {k:dict.fromkeys(nump) for k in range(1)} #dictionary containing m2 values
    of nump columns
    sd = {k:dict.fromkeys(nump) for k in range(1)} #dictionary containing std dev of
     nump columns
    #
    #for wordp values
30  #
    mode = {k:dict.fromkeys(wordp) for k in range(1)} #dictionary containing mode of
     wordp columns
    most = {k:dict.fromkeys(wordp) for k in range(1)} #dictionary containing most oc
    cured item of wordp columns
    count = {k:dict(dict.fromkeys(wordp)) for k in range(1) }#dictionary of dictiona
    ries of count of each item in each wordp column
    #
35  #for all
    #
    n = {k:dict.fromkeys(colname) for k in range(1)} #stores number of elements in e
    ach column
    isnum = True
    #
40  #for the zeror
    #
    hypotheses = {}
    #
    #for naive bayes
45  l = {} #dictionary of likelyhood
    #
```

```
import re
from globfile import *
from random import *
from math import *
PI = 3.1415926535
EE = 2.7182818284

def line(csvfile): #returns formatted line from the csvfile
    l = csvfile.readline()
    endcommare = re.compile('.*,$')
    if l ≠ '':
        l = l.split('#')[0]
        l = l.replace('\t','')
        l = l.replace('\n','')
        l = l.replace(' ','')
        endcomma = endcommare.match(l)
        if endcomma:
            return l+line(csvfile)
        else:
            return l
    else:
        return -1

def rowprint(row): #returns neat rows
    columns = [ "%15s" % cell for cell in row]
    columns.append("%4s" % '#')
    return ' '.join(columns)

def expected(row,z): #returns expected outcome
    out = [c for c in colname[z]]
    for c in row:
        if c in wordp[z]:
            out[colname[z].index(c)] = str(mode[z][c])
        else:
            out[colname[z].index(c)] = str('%0.2f' % round(mu[z][c],2))
    return out

def indexes(lst):
    out = []*len(lst)
    for i in lst:
        out[i] = i
    return out

def newdlist(name, key):
    name[key] = []

def newddict(name,key):
    name[key] = {}

def newddictdict(name,key,c):
    name[key][c] = {}

def indexes(data,z):
    return data[z]

def shuffled(rows):
    shuffle(rows)

def norm(x,m,s):
    s += 1/10**23
    a = 1/sqrt(2*pi*s**2)
    b = (x-m)**2/(2*s**2)
    return a*e**(-1*b)
```

```
#! /usr/bin/env python
from lib import *
from reader import *
from xval import *
from math import *

def nb(test,data,hypotheses,z,k,m):
    total = 0.0
    acc = 0.0
    for h in hypotheses:
        total += len(data[h])
    where = klassAt(z)
    for t in test:
        want = t[where]
        got = likelyhood(t,data,total,hypotheses,l,z,k,m)
        if want ≡ got:
            acc+=1.0
    print '%0.2f' % round(100*acc/len(test),2),

def likelyhood(t,data,total,hypotheses,l,z,k,m):
    like = -0.1*10**23
    best = ''
    total += k*len(hypotheses)
    for h in hypotheses:
        nh = len(data[h])*0.1
        prior = (nh+k) / total
        tmp = log(prior)
        for c in term[h]:
            try:
                ind = colname[h].index(c)
                x = t[ind]
                if x ≡ '?':
                    continue
                y = count[h][c][x]
                tmp += log((y + m*prior)/(nh+m))
            except KeyError:
                continue
        for c in num[h]:
            ind = colname[h].index(c)
            x = t[ind]
            if x ≡ '?':
                continue
            y = norm(x, mu[h][c], sd[h][c])
            tmp += log(y)
        l[h] = tmp
        if tmp ≥ like:
            like = tmp;
            best = h
    return best
```

```
WARNING: empty or missing file
#got:  yes
71.43
==================================
        outlook          $temp       $humidity          windy          =play
#      notes
       overcast          72.50            79.00          FALSE            yes
#   expected
           0.50           7.23            12.94           0.75           1.00
# certainity
       overcast           81.0             75.0          FALSE            yes
#
          rainy           75.0             80.0          FALSE            yes
#
       overcast           64.0             65.0           TRUE            yes
#
          rainy           70.0             96.0          FALSE            yes
#
        outlook          $temp       $humidity          windy          =play
#      notes
          sunny          76.00            90.33          FALSE             no
#   expected
           0.67           7.81             5.03           0.67           1.00
# certainity
          sunny           72.0             95.0          FALSE             no
#
          rainy           71.0             91.0           TRUE             no
#
          sunny           85.0             85.0          FALSE             no
#
#got:  yes
57.14
==================================
        outlook          $temp       $humidity          windy          =play
#      notes
          sunny          73.40            79.20          FALSE            yes
#   expected
           0.40           6.02             9.12           0.60           1.00
# certainity
          sunny           75.0             70.0           TRUE            yes
#
          rainy           68.0             80.0          FALSE            yes
#
          sunny           69.0             70.0          FALSE            yes
#
       overcast           72.0             90.0           TRUE            yes
#
       overcast           83.0             86.0          FALSE            yes
#
        outlook          $temp       $humidity          windy          =play
#      notes
          sunny          72.50            80.00           TRUE             no
#   expected
           0.50          10.61            14.14           1.00           1.00
# certainity
          sunny           80.0             90.0           TRUE             no
#
          rainy           65.0             70.0           TRUE             no
#
#got:  yes
57.14
==================================
        outlook          $temp       $humidity          windy          =play
#      notes
          rainy          70.40            77.20          FALSE            yes
#   expected
           0.40           6.35            11.90           0.80           1.00
# certainity
          sunny           69.0             70.0          FALSE            yes
#
       overcast           64.0             65.0           TRUE            yes
#
          rainy           70.0             96.0          FALSE            yes
```



```
#
          rainy           68.0             80.0          FALSE            yes
#
       overcast           81.0             75.0          FALSE            yes
#
        outlook          $temp       $humidity          windy          =play
#      notes
          rainy          78.00            88.00           TRUE             no
#   expected
           0.50           9.90             4.24           0.50           1.00
# certainity
          rainy           71.0             91.0           TRUE             no
#
          sunny           85.0             85.0          FALSE             no
#
#got:  yes
71.43
==================================
        outlook          $temp       $humidity          windy          =play
#      notes
       overcast          76.25            81.50          FALSE            yes
#   expected
           0.50           4.72             8.70           0.50           1.00
# certainity
       overcast           83.0             86.0          FALSE            yes
#
          rainy           75.0             80.0          FALSE            yes
#
          sunny           75.0             70.0           TRUE            yes
#
       overcast           72.0             90.0           TRUE            yes
#
        outlook          $temp       $humidity          windy          =play
#      notes
          sunny          72.33            85.00           TRUE             no
#   expected
           0.67           7.51            13.23           0.67           1.00
# certainity
          sunny           80.0             90.0           TRUE             no
#
          sunny           72.0             95.0          FALSE             no
#
          rainy           65.0             70.0           TRUE             no
#
```

```
from reader import *
from table import *
from sys import argv
from xval import *

csvfile = open('../data/'+argv[1]+'.csv','r')
readCsv(csvfile,argv[2]) #takes predicted value as arguement
xvals(data,2,2,'nb',argv[2],1,2)

#tableprint(argv[1])
```

```
import re
from lib import *

def makeTable(lst,z):
    newdlist(klass,z)
    newddict(order,z)
    newdlist(less,z)
    newdlist(num,z)
    newdlist(term,z)
    newdlist(dep,z)
    newdlist(indep,z)
    newdlist(nump,z)
    newdlist(wordp,z)
    newdlist(colname,z)
    newdlist(data,z)
    newddict(count,z)
    newddict(n,z)
    newddict(mode,z)
    newddict(most,z)
    newddict(hi,z)
    newddict(lo,z)
    newddict(mu,z)
    newddict(m2,z)
    newddict(sd,z)
    newdlist(data,z)

    csvindex = -1
    for csvcol in lst:
        isnum=True
        csvindex+=1
        ignore = re.match('\?.*$',csvcol)
        if ignore:
            continue
        else:
            colname[z].append(csvcol)
            order[z][csvcol] = csvindex
            klasschk = re.match('=.*$',csvcol)
            morechk = re.match('\+.*$',csvcol)
            lesschk = re.match('-.*$',csvcol)
            numchk = re.match('\$.*$',csvcol)
            if klasschk:
                dep[z].append(csvcol)
                klass[z].append(csvcol)
                isnum = False
            elif morechk:
                dep[z].append(csvcol)
                more[z].append(csvcol)
            elif lesschk:
                dep[z].append(csvcol)
                less[z].append(csvcol)
            elif numchk:
                indep[z].append(csvcol)
                num[z].append(csvcol)
            else:
                indep[z].append(csvcol)
                term[z].append(csvcol)
                isnum = False
            n[z][csvcol] = 0
            if isnum:
                nump[z].append(csvcol)
                hi[z][csvcol] = 0.1 * (-10**13)
                lo[z][csvcol] = 0.1 * (10**13)
                mu[z][csvcol] = 0.0
                m2[z][csvcol] = 0.0
                sd[z][csvcol] = 0.0
            else:
                wordp[z].append(csvcol)
                count[z][csvcol] = {}
                mode[z][csvcol] = 0
                most[z][csvcol] = 0

def addRow(lst,z):
    temp = []
```



```
    skip = False
    for c in klass[z]:
        csvindex = order[z][c]
        item = lst[csvindex]
        if item ≠ z:
            skip = True
        if z ≡ "both":
            skip = False
    for c in colname[z]:
        csvindex = order[z][c]
        item = lst[csvindex]
        uncertain = re.match('\?',str(item))
        if skip:
            return
        if uncertain:
            temp.append(item)
        else:
            n[z][c] += 1
            if c in wordp[z]:
                temp.append(item)
                try:
                    new = count[z][c][item] = count[z][c][item] + 1
                    if new > most[z][c]:
                        most[z][c] = new
                        mode[z][c] = item
                except KeyError:
                    count[z][c][item] = 1
                    if count[z][c][item] > most[z][c]:
                        most[z][c] = 1
                        mode[z][c] = item
            else:
                item = float(item)
                temp.append(item)
                if item > hi[z][c]:
                    hi[z][c] = item
                if item < lo[z][c]:
                    lo[z][c] = item
                delta = item - mu[z][c]
                mu[z][c] += delta / n[z][c]
                m2[z][c] += delta * (item - mu[z][c])
                if n[z][c] > 1:
                    sd[z][c] = (m2[z][c] / (n[z][c] - 1))**0.5
    data[z].append(temp)

def readCsv(csvfile,z):
    seen = False
    FS = ','
    while True:
        lst = line(csvfile)
        if lst ≡ -1:
            print 'WARNING: empty or missing file'
            return -1
        lst = lst.split(FS)
        if lst ≠ ['']:
            if seen:
                addRow(lst,z)
            else:
                seen = True
                makeTable(lst,z)
```

```
    from globfile import *
    from lib import *
    def tableprint(z): #prints table with the summary
        print rowprint(colname[z]),'%10s' % 'notes'
        print rowprint(expected(colname[z],z)), '%10s' % 'expected'
        temp = [ c for c in range(len(colname[z]))]
        for c in colname[z]:
            if c in nump[z]:
                temp[colname[z].index(c)] = str('%0.2f' % round(sd[z][c],2))
            else:
                temp[colname[z].index(c)] = str('%0.2f' % round(float(most[z][c])/fl
    oat(n[z][c]),2))
        print rowprint(temp),'%10s' % 'certainity'
        for row in data[z]:
            print rowprint(row)

    def klass1(data, z):
        for k in klass[z]:
            return data[colname[z].index(k)]

    def klassAt(z):
        for k in klass[z]:
            return colname[z].index(k)
```

```
#! /bin/python
from lib import *
from reader import *
from table import *
from zeror import *
from nb import *

def xvals(data,x,b,f,z,k,m):
    rows = indexes(data,z)
    s = int(len(rows)/b)
    while x>0:
        shuffled(rows)
        for b1 in range(0,b):
            xval(b1*s, (b1+1)*s, data, rows, f, z, k, m)
        x=x-1

def  xval(start, stop, data, rows, f, z, k, m):
    rmax = len(rows)
    test = []
    hypotheses = {}
    temp = ""
    newddict(data,z)
    for r in range(0, rmax):
        d = rows[r]
        if r ≥ start ∧ r < stop:
            test.append(d)
        else:
            temp = klass1(d, z)
            try:
                hypotheses[temp] += 1
                if hypotheses[temp] ≡ 1:
                    makeTable(colname[z],temp)
                addRow(d,temp)
            except KeyError:
                hypotheses[temp] = 1
                if hypotheses[temp] ≡ 1:
                    makeTable(colname[z],temp)
                addRow(d,temp)
    zeror(test, data, hypotheses, z)
    #xvalTest1(test,data,hypotheses)
    #nb(test,data,hypotheses,z,k,m)

def xvalTest1(test,data,hypotheses):
    print "\n================================="
    for h in hypotheses:
        tableprint(h)
```

```
from reader import *
from  xval import *
from lib import *

def zeror(test,data,hypotheses,z):
    hmost = -10**23
    acc = 0
    got = ""
    for h in hypotheses:
        these = len(data[h])
        if these > hmost:
            hmost = these
            got = h
    #print "#got: ",got
    where = klassAt(z)
    for t in test:
        want = t[where]
        if want == got:
            acc+=1.0
    print '%0.2f' % round(100*acc/len(test),2),
```