アライアンス・加算モデル・相乗モデルの Python 言語のプログラミング

```
# -*- coding: utf-8 -*-
import xlrd, xlwt
import numpy as np
from math import sqrt



INPUT_FILE = './cdata.xls'            # 入力ファイル
OUTPUT_FILE = './comp2.xls'           # 出力ファイル

xl_Data_Start = 11            # Excel の何行目からデータが始まるか
xl_Status_Start = 29      # 強み弱みの開始列
xl_Status_Items = 8           # 強み弱みの項目数
xl_Area_Start = 42            # 地域の開始列
xl_Area_Items = 7           # 地域の項目数
xl_Motive_Col = 49            # やる気の列

vector = 8                    # 相互補完のベクトル数
Comp_Max = sqrt(2*((4*vector/2)**2))

Area_Weight = (8, 4, 3, 1, 9, 12, 5)      # 地域の重み付け(首都圏,関西,東海,北陸,その他,ネット,
海外)
Area_Max = np.sum(Area_Weight) * 2            # 地域を加重加算したときの最大値
```

Motive_Min = (0.8, 0.6, 0.4)　　# やる気の換算範囲

Motive_Max = (1.2, 1.4, 1.6)

出力項目の列番号

brank1,¥

Code1_Col, Code2_Col, Name1_Col, Name2_Col, Comp_Col, Areas_Col, Motive1_Col, Motive2_Col,¥

ResultA_Col, RatioA_Col, ResultM1_Col, ResultM2_Col, ResultM3_Col, RatioM1_Col, RatioM2_Col, RatioM3_Col,¥

brank2,¥

S_Code1_Col, S_Code2_Col, S_Name1_Col, S_Name2_Col, S_Comp_Col, S_Areas_Col, S_Motive1_Col, S_Motive2_Col,¥

S_ResultA_Col, S_RatioA_Col, S_ResultM1_Col, S_ResultM2_Col, S_ResultM3_Col, S_RatioM1_Col, S_RatioM2_Col, S_RatioM3_Col,¥

brank3,¥

F_Code1_Col, F_Code2_Col, F_Name1_Col, F_Name2_Col, F_Comp_Col, F_Areas_Col, F_Motive1_Col, F_Motive2_Col,¥

F_ResultA_Col, F_RatioA_Col, F_ResultM1_Col, F_ResultM2_Col, F_ResultM3_Col, F_RatioM1_Col, F_RatioM2_Col, F_RatioM3_Col¥

= (0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,¥

31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50)

class Analyze:

```
def __init__(self):
        self.__code = ()            # 企業番号
        self.__name = ()            # 企業名

        self.__alliance = ()      #  2社間の成立,不成立

        self.__comps = ()           # 強み弱み 152×152 社

        self.__areas = ()           # 地域 152×152 社

        self.__motive = ()          # やる気
        self.__motive_comb = ()           # やる気×やる気

        self.__result_add = ()          # 加算値
        self.__ratio_add = ()           # 係数

        self.__result_mul = ()          # 加算相乗値
        self.__ratio_mul = ()           # 係数


def get_lastrow(self, sheet):
        lastrow = 0
        for row in range(xl_Data_Start, sheet.nrows):
                if sheet.cell(row,0).value == '': break
                lastrow+=1
```

```
            self.__lastrow = xl_Data_Start + lastrow
            print "the last row is " + str(self.__lastrow) + " (" + str(self.__lastrow-xl_Data_Start) +
" entries)"



        def get_codename(self, sheet):
            print "append code, name"


            for row in range(xl_Data_Start, self.__lastrow):
                self.__code.append(int(sheet.cell(row,0).value))
                self.__name.append(sheet.cell(row,1).value)



        def calc(self, sheet):
            print "calculating"
            self.calc_comp(sheet)
            self.calc_area(sheet)
            self.calc_motive(sheet)


            result_add, ratio_add = (), ()
            result_max_add = Comp_Max + Area_Max
            for i in range(len(self.__comps)):
                result_add = self.__comps(i) + self.__areas(i)
                ratio_add = result_add / result_max_add
                self.__result_add.append(result_add)
```

```
            self.__ratio_add.append(ratio_add)
        #print len(self.__result_add)
        #print len(self.__ratio_add)

        for i in range(len(Motive_Max)): # 0,1,2
            result_mul, ratio_mul = (), ()
            result_max_mul = (Comp_Max + Area_Max) * Motive_Max(i) * Motive_Max(i)
            #print result_max_add
            #print result_max_mul
            for j in range(len(self.__comps)):
                r_mul = (self.__comps(j) + self.__areas(j)) * self.__motive_comb(i+1)(j)
                result_mul.append(r_mul)
                ratio_mul.append(r_mul/result_max_mul)
                r_mul = 0
            self.__result_mul.append(result_mul)
            self.__ratio_mul.append(ratio_mul)
        #print self.__ratio_mul
        #print len(self.__ratio_mul)


def calc_comp(self, sheet):
        print "append comp"

        status_array = np.zeros((0))
        for row in range(xl_Data_Start, self.__lastrow):
```

```
            status_record = np.zeros((0))
            for col in range(xl_Status_Start, xl_Status_Start+xl_Status_Items):
                status_record = np.append(status_record, int(sheet.cell(row,col).value))
            status_array = np.concatenate((status_array, status_record), axis=0)
        status_array.resize((len(self.__code), xl_Status_Items))

        comp = ()
        for i in range(len(self.__code)):
            for j in range(len(self.__code)):
                vector = status_array(i,:) - status_array(j,:)
                plus = 0
                minus = 0
                for k in range(len(vector)):
                    if vector(k)>0 : plus+=vector(k)
                    else : minus+=vector(k)
                comp = sqrt( 2*((4*len(vector)/2)**2) ) - sqrt( (4*len(vector)/2-plus)**2 +
(-4*len(vector)/2-minus)**2 )
                self.__comps.append(comp)
         #print self.__comps
         #print len(self.__comps)


    def calc_area(self, sheet):
        print "append area"
```

```
        areas, area_sum = (), ()
        area_array = np.zeros((0))
        for row in range(xl_Data_Start, self.__lastrow):
            area_record = np.zeros((0))
            for col in range(xl_Area_Start, xl_Area_Start+xl_Area_Items):
                if sheet.cell(row,col).value == '': # regard empty cell as 0
                    area_record = np.append(area_record, 0)
                else: # if 1, multiply area weights
                    area_record = np.append(area_record, int(sheet.cell(row,col).value) *
Area_Weight(col-xl_Area_Start))
            area_array = np.concatenate((area_array, area_record), axis=0)
            #print area_record
        area_array.resize((len(self.__code), xl_Area_Items))

        for i in range(len(self.__code)):       # add each area scores
            for j in range(len(self.__code)):
                area = area_array(i,:) + area_array(j,:)
                areas.append(area)

        area_sum = np.sum(areas, axis=1) # total of each 7 areas

        for i in range(len(area_sum)):
            self.__areas.append(area_sum(i))

        #print self.__areas
```

```
            #print len(self.__areas)



    def calc_motive(self, sheet):
        print "append motive"

        motive_input = ()
        for row in range(xl_Data_Start, self.__lastrow):
            motive_input.append(int(sheet.cell(row,xl_Motive_Col).value))

        self.__motive.append(motive_input)

        for i in range(len(Motive_Max)):
            motive_weight = ()
            for j in range(len(motive_input)):
                motive_weight.append(((Motive_Max(i)-Motive_Min(i))/100.0)          *
motive_input(j) + Motive_Min(i))
            self.__motive.append(motive_weight)

        for i in range(len(self.__motive)):
            motive_comb = ()
            for j in range(len(motive_input)):
                for k in range(len(motive_input)):
                    motive_comb.append(self.__motive(i)(j)*self.__motive(i)(k))
            self.__motive_comb.append(motive_comb)
```

```
        #print len(self.__motive)
        #print len(motive_input)



def check_alliance(self): # 152*(x-1) + y - 1
        print "check alliances"

        relbook=xlrd.open_workbook(INPUT_FILE)
        relsheet=relbook.sheet_by_index(1)
        for i in range(len(self.__code)):
            for j in range(len(self.__code)):
                if relsheet.cell(self.__code(i)+1, self.__code(j)+1).value == 1 :
                    self.__alliance.append(1)
                elif relsheet.cell(self.__code(i)+1, self.__code(j)+1).value == 0 :
                    self.__alliance.append(0)
                else:
                    self.__alliance.append(2)

        #print self.__alliance



def write_title(self, sheet):
        print "write output titles"
```

```
sheet.write(0, brank1, u'全社 >>')
sheet.write(0, Code1_Col, u'企業番号 1')
sheet.write(0, Code2_Col, u'企業番号 2')
sheet.write(0, Name1_Col, u'企業名 1')
sheet.write(0, Name2_Col, u'企業名 2')
sheet.write(0, Comp_Col, u'相互補完')
sheet.write(0, Areas_Col, u'地域合計')
sheet.write(0, Motive1_Col, u'やる気 1')
sheet.write(0, Motive2_Col, u'やる気 2')
sheet.write(0, ResultA_Col, u'加算値')
sheet.write(0, RatioA_Col, u'加算係数')
sheet.write(0, ResultM1_Col, str(Motive_Min(0))+u'-'+str(Motive_Max(0)))
sheet.write(0, ResultM2_Col, str(Motive_Min(1))+u'-'+str(Motive_Max(1)))
sheet.write(0, ResultM3_Col, str(Motive_Min(2))+u'-'+str(Motive_Max(2)))
sheet.write(0, RatioM1_Col, str(Motive_Min(0))+u'-'+str(Motive_Max(0)))
sheet.write(0, RatioM2_Col, str(Motive_Min(1))+u'-'+str(Motive_Max(1)))
sheet.write(0, RatioM3_Col, str(Motive_Min(2))+u'-'+str(Motive_Max(2)))
sheet.write(0, brank2, u'成立 >>')
sheet.write(0, S_Code1_Col, u'企業番号 1')
sheet.write(0, S_Code2_Col, u'企業番号 2')
sheet.write(0, S_Name1_Col, u'企業名 1')
sheet.write(0, S_Name2_Col, u'企業名 2')
sheet.write(0, S_Comp_Col, u'相互補完')
sheet.write(0, S_Areas_Col, u'地域合計')
sheet.write(0, S_Motive1_Col, u'やる気 1')
```

```
sheet.write(0, S_Motive2_Col, u'やる気 2')
sheet.write(0, S_ResultA_Col, u'加算値')
sheet.write(0, S_RatioA_Col, u'加算係数')
sheet.write(0, S_ResultM1_Col, str(Motive_Min(0))+u'-'+str(Motive_Max(0)))
sheet.write(0, S_ResultM2_Col, str(Motive_Min(1))+u'-'+str(Motive_Max(1)))
sheet.write(0, S_ResultM3_Col, str(Motive_Min(2))+u'-'+str(Motive_Max(2)))
sheet.write(0, S_RatioM1_Col, str(Motive_Min(0))+u'-'+str(Motive_Max(0)))
sheet.write(0, S_RatioM2_Col, str(Motive_Min(1))+u'-'+str(Motive_Max(1)))
sheet.write(0, S_RatioM3_Col, str(Motive_Min(2))+u'-'+str(Motive_Max(2)))
sheet.write(0, brank3, u'不成立 >>')
sheet.write(0, F_Code1_Col, u'企業番号 1')
sheet.write(0, F_Code2_Col, u'企業番号 2')
sheet.write(0, F_Name1_Col, u'企業名 1')
sheet.write(0, F_Name2_Col, u'企業名 2')
sheet.write(0, F_Comp_Col, u'相互補完')
sheet.write(0, F_Areas_Col, u'地域合計')
sheet.write(0, F_Motive1_Col, u'やる気 1')
sheet.write(0, F_Motive2_Col, u'やる気 2')
sheet.write(0, F_ResultA_Col, u'加算値')
sheet.write(0, F_RatioA_Col, u'加算係数')
sheet.write(0, F_ResultM1_Col, str(Motive_Min(0))+u'-'+str(Motive_Max(0)))
sheet.write(0, F_ResultM2_Col, str(Motive_Min(1))+u'-'+str(Motive_Max(1)))
sheet.write(0, F_ResultM3_Col, str(Motive_Min(2))+u'-'+str(Motive_Max(2)))
sheet.write(0, F_RatioM1_Col, str(Motive_Min(0))+u'-'+str(Motive_Max(0)))
sheet.write(0, F_RatioM2_Col, str(Motive_Min(1))+u'-'+str(Motive_Max(1)))
```

```
        sheet.write(0, F_RatioM3_Col, str(Motive_Min(2))+u'-'+str(Motive_Max(2)))



def write(self, sheet):
        print "write output value"


        row = 1
        value = 0
        for i in range(0, len(self.__code)):
            for j in range(0, len(self.__code)):
                if i != j:
                    sheet.write(row, Code1_Col, self.__code(i))
                    sheet.write(row, Code2_Col, self.__code(j))
                    sheet.write(row, Name1_Col, self.__name(i))
                    sheet.write(row, Name2_Col, self.__name(j))
                    sheet.write(row, Comp_Col, self.__comps(value))
                    sheet.write(row, Areas_Col, self.__areas(value))
                    sheet.write(row, Motive1_Col, self.__motive(0)(i))
                    sheet.write(row, Motive2_Col, self.__motive(0)(j))
                    sheet.write(row, ResultA_Col, self.__result_add(value))
                    sheet.write(row, RatioA_Col, self.__ratio_add(value))
                    for k in range(len(self.__result_mul)): # 0,1,2
                        sheet.write(row, ResultM1_Col+k, self.__result_mul(k)(value))
                        sheet.write(row, RatioM1_Col+k, self.__ratio_mul(k)(value))
                    row+=1
```

```
            value+=1


row = 1
value = 0
for i in range(0, len(self.__code)):
    for j in range(0, len(self.__code)):
        if i != j and self.__alliance(value) == 1:
            sheet.write(row, S_Code1_Col, self.__code(i))
            sheet.write(row, S_Code2_Col, self.__code(j))
            sheet.write(row, S_Name1_Col, self.__name(i))
            sheet.write(row, S_Name2_Col, self.__name(j))
            sheet.write(row, S_Comp_Col, self.__comps(value))
            sheet.write(row, S_Areas_Col, self.__areas(value))
            sheet.write(row, S_Motive1_Col, self.__motive(0)(i))
            sheet.write(row, S_Motive2_Col, self.__motive(0)(j))
            sheet.write(row, S_ResultA_Col, self.__result_add(value))
            sheet.write(row, S_RatioA_Col, self.__ratio_add(value))
            for k in range(len(self.__result_mul)): # 0,1,2
                sheet.write(row, S_ResultM1_Col+k, self.__result_mul(k)(value))
                sheet.write(row, S_RatioM1_Col+k, self.__ratio_mul(k)(value))
            row+=1
        value+=1


row = 1
value = 0
```

```
            for i in range(0, len(self.__code)):
                for j in range(0, len(self.__code)):
                    if i != j and self.__alliance(value) == 0:
                        sheet.write(row, F_Code1_Col, self.__code(i))
                        sheet.write(row, F_Code2_Col, self.__code(j))
                        sheet.write(row, F_Name1_Col, self.__name(i))
                        sheet.write(row, F_Name2_Col, self.__name(j))
                        sheet.write(row, F_Comp_Col, self.__comps(value))
                        sheet.write(row, F_Areas_Col, self.__areas(value))
                        sheet.write(row, F_Motive1_Col, self.__motive(0)(i))
                        sheet.write(row, F_Motive2_Col, self.__motive(0)(j))
                        sheet.write(row, F_ResultA_Col, self.__result_add(value))
                        sheet.write(row, F_RatioA_Col, self.__ratio_add(value))
                        for k in range(len(self.__result_mul)): # 0,1,2
                            sheet.write(row, F_ResultM1_Col+k, self.__result_mul(k)(value))
                            sheet.write(row, F_RatioM1_Col+k, self.__ratio_mul(k)(value))
                        row+=1
                    value+=1


if __name__ == '__main__':
    analyze = Analyze()

    inputbook=xlrd.open_workbook(INPUT_FILE)
    inputsheet=inputbook.sheet_by_index(0)
```

```
analyze.get_lastrow(inputsheet)

analyze.get_codename(inputsheet)
analyze.calc(inputsheet)

analyze.check_alliance()

outputbook=xlwt.Workbook()
outputsheet=outputbook.add_sheet('sheet 1')

analyze.write_title(outputsheet)
analyze.write(outputsheet)

try:
    outputbook.save(OUTPUT_FILE)
except Exception,e:
    print e
    pass
```