emmaredfoot / AHP

Branch: master ▾ AHP / AHP_Buckley.py Find file Copy path

emmaredfoot Small add 4b0ad67 on Mar 16

1 contributor

202 lines (160 sloc) | 8.35 KB

```
import numpy
import csv
import matplotlib.pyplot as plt

#Functions
def GeometricMean(LineList, Spot):
    values = [x[Spot] for x in LineList]
    GeoMean=1
    for i in range(len(LineList)):
        GeoMean=GeoMean*values[i]
    return(GeoMean**(1/len(LineList)))

def PerformanceScores(a, b, c, d):
    # The number of rows in each matrix will always be three, so I am hard coding it in
    PS=[0]*3
    for x in range(3):
        PS[x]=[a[x]/sum(d), b[x]/sum(c), c[x]/sum(b), d[x]/sum(a)]
    return(PS)

#Weight all of the performance score values
#Add all of the values in the individual locations together
#Return the utility function for safety, ability to fluctuate, and profitability
def WeightPS(r_ij, weight):
    weightedPS=[0]*len(r_ij)
    for i in range(len(r_ij)):
        weightedPS[i]=[a*b for a,b in zip(r_ij[i],weight[i])]
        print(i,r_ij[i])
        #L_1[i]=r_ij[i]
    Desal=weightedPS[0]
    HProd=weightedPS[1]
    SynFuel=weightedPS[2]
    #Find the right and left sides of each of the lines
    # for i in range(len(Desal)):
    #     L_1=(Desal[1]-weight[0])/(weight[1]-weight[0])
    #     R_1=(Desal[3]-Desal[2])/(Weight[3]-Weight[2])
    #     L_2=Weight[0]*(Desal[1]-Desal[0])+Desal[0]*(Weight[1]-Weight[1])
    #     R_2=-1*(Weight[3]*(Desal[3]-Desal[2])+Desal[0]*(Weight[3]-Weight[2]))
    #     return(L_1, L_2, R_1, R_2)
    return(Desal, HProd, SynFuel)

def Utility(A, B, C):
    utility=[0]*len(C)
    for x in range(len(C)):
        utility[x] = A[x]+B[x]+C[x]
    return(utility)



#CONSTANTS AND INPUT
#1 corresponds to equally important
#2-3 correspond to weakly more important
#4-5 corresponds to strongly more important
#6-7 correspond to very strongly more important
#8-9 correspond to absolutely more important
```

```
56
57  Equal = (1.0,1.0,1.0,1.0)
58  EqI = (1/2,3/4,5/4,3/2)
59  InvEqI=(2/3,4/5,4/3,2)
60  Weak = (1,3/2,5/2,3)
61  InvWeak=(1/3,2/5,2/3,1)
62  Strong = (2,5/2,7/2,4)
63  InvStrong=(1/4,2/7,2/5,1/2)
64  Very = (5,11/2,13/2,7)
65  InvVery=(1/7,2/11,2/13,1/5)
66  Abs=(7,15/2,17/2,9)
67  InvAbs=(1/9,2/17,2/15,1/7)
68
69  SafetyLine1=[Equal, Equal, Equal, Equal, Equal, Weak,Strong, EqI, Strong, Very, Strong, Strong, EqI, Strong, Strong]
70  SafetyLine2=[InvWeak, InvStrong, InvEqI, InvStrong, InvVery, Equal, Equal, Equal, Equal, Equal, EqI, EqI,EqI, EqI, InvWeak]
71  SafetyLine3=[InvStrong, InvStrong, InvEqI, InvStrong, InvStrong, InvEqI, InvEqI, InvEqI, InvEqI, Weak, Equal, Equal, Equal, Equal, Equal]
72
73  FluctuateLine1 = [Equal, Equal, Equal, Equal, Equal, EqI, EqI, EqI, Abs, Very, Strong, Abs, EqI, Strong, Strong]
74  FluctuateLine2= [InvEqI, InvEqI, InvEqI, InvAbs, InvVery, Equal, Equal, Equal, Equal, Equal, Abs, Abs, EqI, InvWeak, EqI]
75  FluctuateLine3= [InvStrong, InvAbs, InvEqI, InvStrong, InvStrong, InvAbs, InvAbs, InvEqI, Weak, Equal, Equal, Equal, Equal, Equal]
76
77  ProfitabilityLine1 = [Equal, Equal, Equal, Equal, Equal, InvVery, EqI, Very, InvStrong, InvVery, EqI, InvVery, InvAbs, InvStrong, InvWeak]
78  ProfitabilityLine2 = [Very, InvEqI, InvVery, Strong, Very, Equal, Equal, Equal, Equal, Equal, EqI, EqI, InvAbs, EqI, Strong]
79  ProfitabilityLine3 = [InvEqI, Very, Abs, Strong, Weak, InvEqI, InvEqI, Abs, InvEqI, InvStrong, Equal, Equal, Equal, Equal, Equal]
80
81  CharLine1=[Equal, Equal, Equal, Equal, Equal, Very, Abs, Abs, Very, Very, Abs, Abs, Abs, EqI, Very]
82  CharLine2=[InvVery, InvAbs, InvAbs, InvVery, InvVery, Equal, Equal, Equal, Equal, Equal, InvAbs, InvVery, EqI, InvAbs]
83  CharLine3=[InvAbs, InvAbs, InvAbs, InvEqI, InvVery, Abs, Very, InvEqI, Abs, Strong, Equal, Equal, Equal, Equal, Equal]
84
85  #Find the Geometric Mean for Each of the matrices and each location
86  SafeSpot0 = [GeometricMean(SafetyLine1, 0), GeometricMean(SafetyLine2, 0), GeometricMean(SafetyLine3, 0)]
87  SafeSpot1 = [GeometricMean(SafetyLine1, 1), GeometricMean(SafetyLine2, 1),GeometricMean(SafetyLine3, 1)]
88  SafeSpot2 = [GeometricMean(SafetyLine1, 2), GeometricMean(SafetyLine2, 2),GeometricMean(SafetyLine3, 2)]
89  SafeSpot3 = [GeometricMean(SafetyLine1, 3), GeometricMean(SafetyLine2, 3),GeometricMean(SafetyLine3, 3)]
90
91  FlucSpot0 =  [GeometricMean(FluctuateLine1, 0), GeometricMean(FluctuateLine2, 0), GeometricMean(FluctuateLine3, 0)]
92  FlucSpot1  = [GeometricMean(FluctuateLine1, 1), GeometricMean(FluctuateLine2, 1), GeometricMean(FluctuateLine3, 1)]
93  FlucSpot2  = [GeometricMean(FluctuateLine1, 2), GeometricMean(FluctuateLine2, 2), GeometricMean(FluctuateLine3, 2)]
94  FlucSpot3  = [GeometricMean(FluctuateLine1, 3), GeometricMean(FluctuateLine2, 3), GeometricMean(FluctuateLine3, 3)]
95
96  ProfitSpot0 =  [GeometricMean(ProfitabilityLine1, 0), GeometricMean(ProfitabilityLine2, 0), GeometricMean(ProfitabilityLine3, 0)]
97  ProfitSpot1  = [GeometricMean(ProfitabilityLine1, 1), GeometricMean(ProfitabilityLine2, 1), GeometricMean(ProfitabilityLine3, 1)]
98  ProfitSpot2  = [GeometricMean(ProfitabilityLine1, 2), GeometricMean(ProfitabilityLine2, 2), GeometricMean(ProfitabilityLine3, 2)]
99  ProfitSpot3  = [GeometricMean(ProfitabilityLine1, 3), GeometricMean(ProfitabilityLine2, 3), GeometricMean(ProfitabilityLine3, 3)]
100
101 CharSpot0 =  [GeometricMean(CharLine1, 0), GeometricMean(CharLine2, 0), GeometricMean(CharLine3, 0)]
102 CharSpot1  = [GeometricMean(CharLine1, 1), GeometricMean(CharLine2, 1), GeometricMean(CharLine3, 1)]
103 CharSpot2  = [GeometricMean(CharLine1, 2), GeometricMean(CharLine2, 2), GeometricMean(CharLine3, 2)]
104 CharSpot3  = [GeometricMean(CharLine1, 3), GeometricMean(CharLine2, 3), GeometricMean(CharLine3, 3)]
105
106 #Save the Performance Score values
107 SafetyPS = PerformanceScores(SafeSpot0, SafeSpot1, SafeSpot2, SafeSpot3)
108 FlucPS = PerformanceScores(FlucSpot0, FlucSpot1, FlucSpot2, FlucSpot3)
109 ProfitPS = PerformanceScores(ProfitSpot0, ProfitSpot1, ProfitSpot2, ProfitSpot3)
110 Weights = PerformanceScores(CharSpot0, CharSpot1, CharSpot2, CharSpot3)
111
112
113
114 #I am passing in the performance scores and returning the weighted performance scores for each of the options
115 DesalSafe, HProdSafe, SynFuelSafe = WeightPS(SafetyPS, Weights)
116 DesalFluc, HProdFluc, SynFuelFluc = WeightPS(FlucPS, Weights)
117 DesalProf, HProdProf, SynFuelProf = WeightPS(ProfitPS, Weights)
118
119
120
121 #Return the limits of the fuzzy numbers
122 #Finding the left and right points for the member Functions
```

```
def memberlimits(PerfScore, Weight):
    L_1=(PerfScore[1]-PerfScore[0])/(Weight[1]-Weight[0])
    R_1=(PerfScore[3]-PerfScore[2])/(Weight[3]-Weight[2])
    L_2=Weight[0]*(PerfScore[1]-PerfScore[0])+PerfScore[0]*(Weight[1]-Weight[1])
    R_2=-1*(Weight[3]*(PerfScore[3]-PerfScore[2])+PerfScore[0]*(Weight[3]-Weight[2]))
    return(L_1, L_2, R_1, R_2)

DSL1, DSL2, DSL3, DSL4=memberlimits(DesalSafe, Weights[0])
DFL1, DFL2, DFL3, DFL4=memberlimits(DesalFluc, Weights[1])
DPL1, DPL2, DPL3, DPL4=memberlimits(DesalFluc, Weights[2])
HSL1, HSL2, HSL3, HSL4=memberlimits(HProdSafe, Weights[0])
HFL1, HFL2, HFL3, HFL4=memberlimits(HProdFluc, Weights[1])
HPL1, HPL2, HPL3, HPL4=memberlimits(HProdProf, Weights[2])
SSL1, SSL2, SSL3, SSL4=memberlimits(SynFuelSafe, Weights[0])
SFL1, SFL2, SFL3, DFL4=memberlimits(SynFuelFluc, Weights[1])
DPL1, DPL2, DPL3, DPL4=memberlimits(SynFuelFluc, Weights[2])

DesalL1 = DSL1+DFL1+DPL1
DesalL2 = DSL2+DFL2+DPL2
DesalR1 =

def AddUp



#DesalL1, DesalL2, DesalR1, DesalR2 = memberlimits()

DesalU=Utility(DesalSafe, DesalFluc, DesalProf)
HProdU= Utility(HProdSafe, HProdFluc, HProdProf)
SynFuelU=Utility(SynFuelSafe, SynFuelFluc, SynFuelProf)

#Save the PerformanceScores to a file
with open('PerformanceScores_new.csv', 'w') as myfile:
    out=csv.writer(myfile)
    out.writerow('Safety')
    out.writerow(SafetyPS[0])
    out.writerow(SafetyPS[1])
    out.writerow(SafetyPS[2])
    out.writerow('\nFluctuate')
    out.writerow(FlucPS[0])
    out.writerow(FlucPS[1])
    out.writerow(FlucPS[2])
    out.writerow('\nProfitability')
    out.writerow(ProfitPS[0])
    out.writerow(ProfitPS[1])
    out.writerow(ProfitPS[2])
    out.writerow('\nFuzzy Weights')
    out.writerow(Weights[0])
    out.writerow(Weights[1])
    out.writerow(Weights[2])
myfile.close()

with open('Utility_2.csv', 'w') as ufile:
    output = csv.writer(ufile)
    output.writerow('Desalination')
    output.writerow(Utility(DesalSafe, DesalFluc, DesalProf))
    output.writerow("Hydrogen")
    output.writerow(Utility(HProdSafe, HProdFluc, HProdProf))
    output.writerow("Synthetic Fuels")
    output.writerow(Utility(SynFuelSafe, SynFuelFluc, SynFuelProf))

ufile.close()
```

```
190
191
192 #Graph Member Functions
193 # x=list(numpy.arange(0,2,0.01))
194 # alpha=[0]*len(x)
195 # for i in range(len(x)):
196 #     if x[i]<DesalU[0] or x[i]>DesalU[3]:
197 #         alpha[i]=0
198 #     else:
199 #         alpha[i]=1
200 # plt.plot(x,alpha)
201 # plt.show()
```