NewDisneyLoc

December 16, 2021

[1]:
```
#This is the project to find the 3 new locations for new Disneylands!
#Import all the packages we need.
import pandas as pd
import numpy as np
from pyomo.environ import *
from pyomo.opt import SolverFactory
```

[2]:
```
#Read in data
data = pd.read_excel('Disney_Loc_Final.xlsx', sheet_name = 'country')
data.head()
```

[2]:

	rank	country	area (km2)	region	subregion	pop2021	Exist Disney park number	GDP_score	GDP (billions)	temp_score	avg_temp	disney+_score	Disney+ consumer base	set-up cost(billion)	develped country score
0	1	China	9706961	Asia	Eastern Asia	1.444216e+09	1	5	10216.630334	4	6.325664	2	0	4	10
1	2	India	3287590	Asia	Southern Asia	1.393409e+09	0	5	2100.751461	1	23.945434	2	0	2	10
2	3	United States	9372610	Americas	Northern America	3.329151e+08	1	1	65279.529026	4	10.000000	1	1	4	1
3	4	Indonesia	1904569	Asia	South-Eastern Asia	2.763618e+08	0	5	4135.201531	1	25.718970	2	0	2	10
4	5	Pakistan	881912	Asia	Southern Asia	2.251999e+08	0	5	1284.702047	2	19.981015	2	0	4	10

[3]:
```
#Drop data with Null Value, small space, small GDP
data = data.dropna()
data = data.drop(data[data['area (km2)'] < 1000].index)
data = data.drop(data[data['GDP (billions)'] < 2500].index)
```

[4]:
```
#Create two dataframes with dummy variables.
region_dummy = pd.get_dummies(data['region'])
subregion_dummy = pd.get_dummies(data['subregion'])
```

Factors

[5]:
```
#Location
country = data['country'].to_list()
region = data['region'].to_list()
subregion = data['subregion'].to_list()
exist = data['Exist Disney park number'].to_list()

#Attendance
population = data['pop2021'].to_list()
expark = data['Exist Disney park number'].to_list()
GDP_sc = data['GDP_score'].to_list()
temp_sc = data['temp_score'].to_list()
DNplus_sc = data['disney+_score'].to_list()
dv_sc = data['develped country score'].to_list()
growth_att = 0.01 #growth rate for number of attendance in each year

#Cost
setup_cost = data['set-up cost(billion)'].to_list() #charge only in the first␣
 ↪year
fixed_cost = 1 #billion dollars charge every year
growth_fc = 0.01 #growth rate for fixed cost in each year
var_cost = 50 #per attendance

#Revenue
ticket_price = 200 #for first year
growth_tkprice = 0.05 #growth rate for ticket price in each year
SF_price = 200 #souvenir and food price average charge per attendance
dis_rate = 0.05

#Constraint
budget = 10 #billion dollars
```

[6]:
```
#Attendance calculation (scale:billion)
attendance = []
for i in range(len(country)):
    attendance.append((population[i]/
 ↪(GDP_sc[i]*10)+(temp_sc[i])+(DNplus_sc[i])+(dv_sc[i]*5))/1000000000)
```

[7]:
```
#Declare decision variables
model = ConcreteModel()
model.x = Var(range(len(country)), domain = Binary) #location exist
```

[8]:
```
#Constraint1: Total set-up costs should be less or equal than our budget.
model.Bud = ConstraintList()
model.Bud.add(expr = sum(setup_cost[i] * model.x[i] for i in␣
 ↪range(len(country))) <= budget)

#Constraint2: We want 3 new Disneylands.
model.DisneyNumb = ConstraintList()
model.DisneyNumb.add(expr = sum(model.x[i] for i in range(len(country))) == 3)

#Constraint3: We don't want another Disneyland in the country that already has␣
 ↪one.
model.NotSameLoc = ConstraintList()
for i in range(len(country)):
    model.NotSameLoc.add(expr = model.x[i] + exist[i] <= 1 )

#Constraint4: For each region, we want less or equal than 2 new Disneylands.
model.Region = ConstraintList()
for j in range(len(region_dummy.columns)):
    model.Region.add(expr = sum(model.x[i]*region_dummy.iloc[i,j] for i in␣
 ↪range(len(country))) <= 2)

#Constraint5: For each subregion, we want less or equal than 1 new Disneylands.
model.SubRegion = ConstraintList()
for k in range(len(subregion_dummy.columns)):
    model.SubRegion.add(expr = sum(model.x[i]*subregion_dummy.iloc[i,j] for i␣
 ↪in range(len(country))) <= 1)
```

[10]:
```
#Calculation for revenues (for npv in 20 years).
revenue = {}
npv_revenue = []
for j in range(len(country)):
    for i in range(20):
        revenue[j,i] = (ticket_price * (1 + growth_tkprice) ** i + SF_price) *␣
 ↪attendance[j] * (1 + growth_att) ** i


for j in range(len(country)):
    npv_revenue.append(sum(revenue[j,i]/(1 + dis_rate)**i for i in range(20)))
```

[11]:
```
#Calculation for costs (for npv in 20 years).
cost = {}
npv_cost = []
for j in range(len(country)):
```

```
    for i in range(20):
        cost[j,i] = fixed_cost*(1 + growth_fc)**i + (var_cost * attendance[j] *␣
 ↪(1 + growth_att) ** i)

for j in range(len(country)):
    npv_cost.append(setup_cost[j] + sum(cost[j,i]/(1 + dis_rate)**i for i in␣
 ↪range(20)))
```

[12]:
```
#Calculation for NPV
npv = []
for j in range(len(country)):
    npv.append(npv_revenue[j] - npv_cost[j])
```

[13]:
```
#Objective
model.Objective = Objective(expr = sum(model.x[i]*npv[i] for i in␣
 ↪range(len(country))), sense = maximize)
```

[14]:
```
#Solve and results
opt = SolverFactory('glpk')
opt.solve(model)
results = opt.solve(model, tee=True)
```

```
GLPSOL--GLPK LP/MIP Solver 5.0
Parameter(s) specified in the command line:
 --write /var/folders/dl/2xvw5bbd5_382j15_z3hc52m0000gn/T/tmp4pr5_3ik.glpk.raw
 --wglp /var/folders/dl/2xvw5bbd5_382j15_z3hc52m0000gn/T/tmpqq6_968p.glpk.glp
 --cpxlp /var/folders/dl/2xvw5bbd5_382j15_z3hc52m0000gn/T/tmpemp75ahw.pyomo.lp
Reading problem data from
'/var/folders/dl/2xvw5bbd5_382j15_z3hc52m0000gn/T/tmpemp75ahw.pyomo.lp'…
/var/folders/dl/2xvw5bbd5_382j15_z3hc52m0000gn/T/tmpemp75ahw.pyomo.lp:1169:
warning: lower bound of variable 'x1' redefined
/var/folders/dl/2xvw5bbd5_382j15_z3hc52m0000gn/T/tmpemp75ahw.pyomo.lp:1169:
warning: upper bound of variable 'x1' redefined
143 rows, 116 columns, 501 non-zeros
115 integer variables, all of which are binary
1284 lines were read
Writing problem data to
'/var/folders/dl/2xvw5bbd5_382j15_z3hc52m0000gn/T/tmpqq6_968p.glpk.glp'…
1022 lines were written
GLPK Integer Optimizer 5.0
143 rows, 116 columns, 501 non-zeros
115 integer variables, all of which are binary
Preprocessing…
27 rows, 110 columns, 370 non-zeros
110 integer variables, all of which are binary
Scaling…
 A: min|aij| =  1.000e+00  max|aij| =  4.000e+00  ratio =  4.000e+00
Problem data seem to be well scaled
```

```
Constructing initial basis…
Size of triangular part is 27
Solving LP relaxation…
GLPK Simplex Optimizer 5.0
27 rows, 110 columns, 370 non-zeros
      0: obj =  -5.388454279e+01 inf =   5.000e+00 (3)
      2: obj =  -3.932906457e+01 inf =   0.000e+00 (0)
*     8: obj =   3.990744874e+01 inf =   0.000e+00 (0)
OPTIMAL LP SOLUTION FOUND
Integer optimization begins…
Long-step dual simplex will be used
+     8: mip =     not found yet <=              +inf        (1; 0)
+     8: >>>>>   3.990744874e+01 <=   3.990744874e+01   0.0% (1; 0)
+     8: mip =   3.990744874e+01 <=     tree is empty   0.0% (0; 1)
INTEGER OPTIMAL SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.2 Mb (196854 bytes)
Writing MIP solution to
'/var/folders/dl/2xvw5bbd5_382j15_z3hc52m0000gn/T/tmp4pr5_3ik.glpk.raw'…
268 lines were written
```

[15]:
```
#Print solution and our selection
print ("The top 3 locations for our new Disneyland would be: ")
for i in range(len(country)):
    if model.x[i] != 0:
        print(country[i], "with total revenue at:", round(npv[i],3), "billion␣
 ↪dollars.")
print("Our total revenue from 3 new Disneylands would be: ", round(model.
 ↪Objective(),3),"billion dollars.")
```

```
The top 3 locations for our new Disneyland would be:
Indonesia with total revenue at: 19.918 billion dollars.
Brazil with total revenue at: 10.772 billion dollars.
Germany with total revenue at: 9.218 billion dollars.
Our total revenue from 3 new Disneylands would be:  39.907 billion dollars.
```

[ ]: