MP Optimization Model

January 18, 2022

[1]:
```
import pandas as pd
from pyomo.environ import *
from pyomo.opt import SolverFactory
```

[2]:
```
#load data
df = pd.read_excel(open('MP_scenarios.xlsx', 'rb'), sheet_name='Sheet1')
df.head()
```

[2]:
```
   No  Price  Demand (0% discount)  Demand (15% discount)  \
0   1  55.48                   120                    160
1   2  53.68                   115                    149
2   3  61.56                   140                    191
3   4  65.72                   115                    151
4   5  64.98                   120                    173

   Demand (30% discount)  Demand (50% discount)  Weeks Left  \
0                    194                    223          15
1                    171                    197          15
2                    207                    459          15
3                    242                    278          15
4                    221                    335          15

   Inventory Remaining  Salvage Value
0                 1421         13.870
1                 2396         13.420
2                 2544         15.390
3                 1316         16.430
4                 1377         16.245
```

[3]:
```
# define variables in first row of markdown pricing problem
num_weeks_each_disc = 4
demand = [120,160,194,223]
discount = [0, 0.15, 0.3, 0.5]
weeks = 15
price = 55.48
starting_inventory = 1421
discount_price = [price*(1-discount[i]) for i in range(num_weeks_each_disc)]
discount_price
```

[3]:
```
[55.48, 47.157999999999994, 38.836, 27.74]
```

[4]:
```
# solve one instance of markdown pricing problem

#declare model
model = ConcreteModel()

#declare DVs
model.x = Var(range(num_weeks_each_disc), domain = NonNegativeReals)

#specify the objective function
model.Objective = Objective(expr = sum(discount_price[i]*(demand[i]*model.x[i])␣
 ↪for i in range(num_weeks_each_disc)), sense = maximize)

#specify the constraints
model.Constraint_weeks = Constraint(expr = sum(model.x[i] for i in␣
 ↪range(num_weeks_each_disc)) <= weeks)
model.Constraint_inventory = Constraint(expr = sum(demand[i]*model.x[i] for i␣
 ↪in range(num_weeks_each_disc)) <= starting_inventory)

opt = SolverFactory('glpk')
opt.solve(model)

print('Max Total Revenue =', model.Objective())
for i in range(num_weeks_each_disc):
    print(model.x[i],":",value(model.x[i]))
```

```
Max Total Revenue = 78837.08000000022
x[0] : 11.8416666666667
x[1] : 0.0
x[2] : 0.0
x[3] : 0.0
```

[5]:
```
# put model into a function

def solve(price, demand, discount, weeks, starting_inventory):
    discount_price = [price*(1-discount[i]) for i in range(num_weeks_each_disc)]

    #declare model
    model = ConcreteModel()

    #declare DVs
    model.x = Var(range(num_weeks_each_disc), domain = NonNegativeReals)

    #specify the objective function
    model.Objective = Objective(expr = sum(discount_price[i]*(demand[i]*model.
 ↪x[i]) for i in range(num_weeks_each_disc)), sense = maximize)
```

```
    #specify the constraints
    model.Constraint_weeks = Constraint(expr = sum(model.x[i] for i in␣
↪range(num_weeks_each_disc)) <= weeks)
    model.Constraint_inventory = Constraint(expr = sum(demand[i]*model.x[i] for␣
↪i in range(num_weeks_each_disc)) <= starting_inventory)

    opt = SolverFactory('glpk')
    opt.solve(model)
    solution = []
    solution.append('Max Total Revenue = ' + str(model.Objective()))
    for i in range(num_weeks_each_disc):
        solution.append(model.x[i].value)
    return solution
```

[6]:

```
# test function

num_weeks_each_disc = 4
demand = [120,160,194,223]
discount = [0, 0.15, 0.3, 0.5]
weeks = 15
price = 55.48
starting_inventory = 1421
solve(price, demand, discount, weeks, starting_inventory)
```

[6]:

```
['Max Total Revenue = 78837.08000000022', 11.8416666666667, 0.0, 0.0, 0.0]
```

[7]:

```
# use markdown pricing function with data read from the file

k = 1
rowdata = df.iloc[k].values.tolist()
price = rowdata[1]
demand = rowdata[2:6]
discount = [0, 0.15, 0.3, 0.5]
weeks = rowdata[6]
starting_inventory = rowdata[7]
solve(price, demand, discount, weeks, starting_inventory)
```

[7]:

```
['Max Total Revenue = 101978.58', 0.0, 15.0, 0.0, 0.0]
```

[8]:

```
# solve each of the markdown pricing file and append the solution in a new␣
↪column

outputs = []
for k in range(len(df)):
    rowdata = df.iloc[k].values.tolist()
    price = rowdata[1]
    demand = rowdata[2:6]
    discount = [0, 0.15, 0.3, 0.5]
    weeks = rowdata[6]
    starting_inventory = rowdata[7]
    outputs.append(solve(price, demand, discount, weeks, starting_inventory))
```

[10]:

```
# add solutions into dataset

df['solution'] = outputs
df
```

[10]:

```
      No  Price  Demand (0% discount)  Demand (15% discount)  \
0      1  55.48                   120                    160
1      2  53.68                   115                    149
2      3  61.56                   140                    191
3      4  65.72                   115                    151
4      5  64.98                   120                    173
..   …    …                   …                    …
195  196  58.18                   105                    152
196  197  58.58                   125                    169
197  198  50.18                   145                    218
198  199  59.58                   140                    185
199  200  64.56                   135                    182

     Demand (30% discount)  Demand (50% discount)  Weeks Left  \
0                      194                    223          15
1                      171                    197          15
2                      207                    459          15
3                      242                    278          15
4                      221                    335          15
..                     …                    …         …
195                    160                    454          15
196                    177                    298          15
197                    263                    362          15
198                    224                    292          15
199                    191                    315          15

     Inventory Remaining  Salvage Value  \
0                   1421         13.870
1                   2396         13.420
2                   2544         15.390
3                   1316         16.430
4                   1377         16.245
..                   …            …
195                 2320         14.545
196                 1465         14.645
197                 1433         12.545
```

```
198                  1355         14.895
199                  2595         16.140

                                             solution
0    [Max Total Revenue = 78837.08000000022, 11.841…
1    [Max Total Revenue = 101978.58, 0.0, 15.0, 0.0…
2    [Max Total Revenue = 141254.12752941175, 6.294…
3    [Max Total Revenue = 86487.52000000027, 11.443…
4    [Max Total Revenue = 89477.46, 11.475, 0.0, 0…
..                                                …
195  [Max Total Revenue = 113506.48291390749, 0.0, …
196  [Max Total Revenue = 85819.70000000001, 11.72,…
197  [Max Total Revenue = 71907.94000000005, 9.8827…
198  [Max Total Revenue = 80730.90000000001, 9.6785…
199  [Max Total Revenue = 146158.34553191497, 2.872…

[200 rows x 10 columns]
```

[22]:

```python
#export dataset with solutions to csv

df.to_csv('P2_Keith_Hines.csv', index = 'No')
```

[ ]:

```python
```