M.Eng in Operations Research and Information Engineering

Homework 2

Subject: Pricing Analytics and Revenue Management

Professor: Omar El Housni **oe46**

Authors: Federico Tiberga **ft232**

Due date: March 3rd

Problem 1

Networking Revenue Management

An airline company offers 8 different itineraries presented in the table below. There are only four flight legs: Bos-Chi, NYC-Chi, Chi-SF, and Chi-LA and each itinerary uses one or two of these flight legs. There is one aircraft per leg and all planes have 200 seats. The table presents the demand for each itinerary for two fare classes: the low fare (Q-class) and the high fare (Y-class).

			Q-Class		Y-Class	
Departure	Destination	Connection	Dem	Fare	Dem	Fare
Bos	Chi	direct	25	\$200	20	\$230
Bos	SF	stop at Chi	55	\$320	40	\$420
Bos	LA	stop at Chi	65	\$400	25	\$490
NYC	Chi	direct	24	\$250	16	\$290
NYC	SF	stop at Chi	65	\$410	50	\$550
NYC	LA	stop at Chi	40	\$450	35	\$550
Chi	SF	direct	21	\$200	20	\$230
Chi	LA	direct	25	\$250	14	\$300

1. We would like to determine the optimal way to allocate the seats for each leg. Write an optimization problem that maximizes the total revenue. Solve the optimization problem using a solver (e.g. Gurobi, OR tools, etc) and report the optimal revenue.

   Solution:
   To solve the problem, we simply create an optimization model of the form

   $$\begin{aligned} \max \quad & \text{profits} \\ \text{s.t.} \quad & \text{total ticket sales of flight } j \leq 200 \\ & \text{total ticket sales of type } i \leq \text{demand for tickets of type } i \end{aligned}$$

   We can use Gurobi to optimize this model. First we define all the data, the decision variables, the objective function and the constraints.

In [1]:
```
# demand of each itinerary
demand = [[25, 20],
          [55, 40],
          [65, 25],
          [24, 16],
          [65, 50],
          [40, 35],
          [21, 20],
          [25, 14]]
# cost of each itinerary
fares = [[200, 230],
         [320, 420],
         [400, 490],
         [250, 290],
         [410, 550],
         [450, 550],
         [200, 230],
         [250, 300]]
```

In [2]:
```
# number of seats occupied by each itinerary
seats = [[1, 0, 0, 0],
         [1, 0, 1, 0],
         [1, 0, 0, 1],
         [0, 1, 0, 0],
         [0, 1, 1, 0],
         [0, 1, 0, 1],
         [0, 0, 1, 0],
         [0, 0, 0, 1]]
N = len(demand) # number of itineraries
F = len(seats[0]) # number of flights
```

In [3]:
```
# create a new model
m = gp.Model("Network Revenue Management")

# create variables = number of tickets sold on each itinerary (Q and Y
    classes)
q = m.addVars(N, vtype=gp.GRB.CONTINUOUS, lb = 0, name="Q")
y = m.addVars(N, vtype=gp.GRB.CONTINUOUS, lb = 0, name="Y")

# objective function
objExpr = gp.LinExpr()
for i in range(N):
    objExpr += fares[i][0]*q[i] + fares[i][1]*y[i]
m.setObjective(objExpr, gp.GRB.MAXIMIZE)
```

In [4]:
```
# tickets sold are lower than the demand
for i in range(N):
    m.addConstr(lhs = q[i], sense = gp.GRB.LESS_EQUAL, rhs = demand[i][0],
     name = "demand q class " + str(i))
    m.addConstr(lhs = y[i], sense = gp.GRB.LESS_EQUAL, rhs = demand[i][1],
     name = "demand y class " + str(i))

# number of seats occupied by each itinerary
for j in range(F):
    constrExpr = gp.LinExpr()
    for i in range(N):
        constrExpr += seats[i][j]*q[i] + seats[i][j]*y[i]
    m.addConstr(lhs = constrExpr, sense = gp.GRB.LESS_EQUAL, rhs = 200,
    name = "seats occupied " + str(j))
```

We can then optimize the model.

In [5]:
```
m.write(filename = "HW2_PARM_ft232.lp")

# optimize the model
m.optimize()

# print optimal objective and optimal solution
print("\nOptimal Objective: " + str(m.ObjVal))
print("\nOptimal Solution:")
allVars = m.getVars()
for var in allVars:
    print(var.varName + " " + str(var.x))

# print dual variables
print("\nDual Variables:")
duals = m.getAttr("Pi", m.getConstrs())
print(duals)
```

Out[5]:
```
Optimal Objective: 182280.0

Optimal Solution:
Q[0] 25.0
Q[1] 25.0
Q[2] 65.0
Q[3] 19.0
Q[4] 44.0
Q[5] 36.0
Q[6] 21.0
Q[7] 25.0
Y[0] 20.0
Y[1] 40.0
Y[2] 25.0
Y[3] 16.0
Y[4] 50.0
Y[5] 35.0
Y[6] 20.0
Y[7] 14.0

Dual Variables:
[40.0, 70.0, 0.0, 100.0, 40.0, 130.0, 0.0, 40.0, 0.0, 140.0, 0.0, 100.0, 40.0,
    70.0, 50.0, 100.0, 160.0, 250.0, 160.0, 200.0]
```

We note that although we chose that the decision variables vary in the continuum (so that we could then do a study of shadow prices), the optimal solution has integer values. So the optimal solution of the MIP will be the same.

2. If you can add one seat on one leg, which leg would it be?

Solution:
When we optimized the model, we also printed the dual variables for all the constraints. The shadow prices linked to the flight capacity constraint are the last 4 values in our dual variables vector, and the highest shadow price is associated with the flight from NYC to Chicago. Therefore, if we could add one seat on one leg, we should add it on the leg **NYC-Chi**, and our profit would increase by $**250**.

Problem 2

Pricing using a linear demand function

For this question, we use the data set in `Demand.xlsx`. Suppose the demand function is linear: $d(p) = a - bp,\ a, b > 0$

1. Use the data in `Demand.xlsx` to estimate a, b using linear regression

Solution:

We can simply use the `LinearRegression()` function from the `sklearn.linear_model` package.

In [6]:
```
df2 = pd.read_excel("Demand.xlsx", sheet_name='Sheet1')
x = df2['price'].values.reshape(-1, 1)
y = df2['Demand'].values.reshape(-1, 1)
m = LinearRegression().fit(x, y)
a = m.intercept_[0]
b = m.coef_[0][0]
print('the line is: ' +  str(a) + str(b) + 'x')
```

Out[6]:
```
the line is: 395.3217512993542-1.9797083268982867x
```

Therefore, $\mathbf{a = 395.32}$ and $\mathbf{b = 1.98}$

This result is highlighted in Figure 1

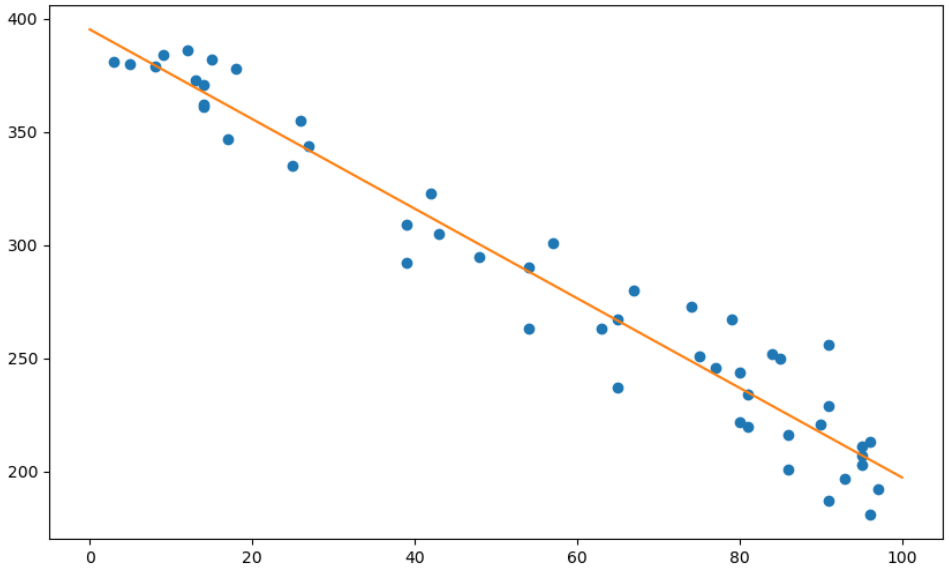

Figure 1: Linear demand estimation through linear regression

Problem 2 continued on next page...

2. Find the optimal price p^* that maximizes the expected revenue.

Solution:
Now, we can simply solve the optimization problem using the first-order conditions

$$\begin{aligned} &\max_{p \geq 0} \quad p \cdot d(p) \\ &\max_{p \geq 0} \quad p \cdot (a - bp) \\ &\max_{p \geq 0} \quad ap - bp^2 \end{aligned}$$

Then, we set the derivative equal to 0

$$\begin{aligned} a - 2bp^* &= 0 \\ p^* &= \frac{a}{2b} = 99.84 \end{aligned}$$

The expected revenue is maximized when the price is equal to **99.84**

Problem 3

Empirical Willingness-to-pay distribution

In the file `InSample.xlsx`, you are given data $(p_1, y_1), \ldots, (p_N, y_N)$, where p_i is the price offered for a product and y_i is the indicator function of whether the product was sold at price p_i. Suppose the willingness of paying this product is a random variable $W \sim \text{cdf}F(\cdot)$, here cdf refers to the cumulative density function of W.

1. Use the data in `InSample.xlsx` to estimate the cdf $F(\cdot)$. Note F is a stair-wise function in this case. Draw a rough sketch to depict the function.

   Solution:
   First, we define a function `wtp(df, p)` that takes in input the data *df* and a price value p and computes the probability

$$P(W \geq p) = \frac{\text{buys at price greater than p}}{\text{buys at price greater than p} + \text{no-buys at price lower than p}}$$

In [7]:
```
def wtp(df, p):
    no_buy = 0
    buy = 0
    for i in range(len(df)):
        if df['p_i'][i] >= p and df['y_i'][i] == 1:
            buy += 1
        elif df['p_i'][i] <= p and df['y_i'][i] == 0:
            no_buy += 1
    return buy / (no_buy + buy)
```

   Now we can use the function to estimate the cdf $F(\cdot)$

In [8]:
```
df3 = pd.read_excel("InSample.xlsx", sheet_name='Sheet1')
xw = []
Fw = []
for i in range(max(df3['p_i'] + 1)):
    xw.append(i)
    Fw.append(1 - wtp(df3, i))
```

In [9]:
```
 # plot the willingness-to-pay curve
plt.figure(figsize=(10, 6))
plt.step(xw, Fw, where = 'post')
plt.savefig('willingness_to_pay_F.png')
plt.show()
```

   The results are shown in Figure 2

2. Find the optimal price p^* that maximizes the expected revenue. Suppose $W_1, \ldots, W_N$ are the willingness-to-pay data points in `InSample.xlsx`. Then expected revenue for any price p can be expressed as

$$\frac{1}{N} \cdot \sum_{i=1}^{N} p \cdot \mathbf{1}(W_i \geq p)$$

   where $\mathbf{1}(\cdot)$ is an indicator function that evaluates to 1 if the condition is true and 0 otherwise.

   Solution:
   We define a function `expected_revenue(df, p)` that takes in input the willingness-to-pay data points and a price p and returns the expected profit for that price

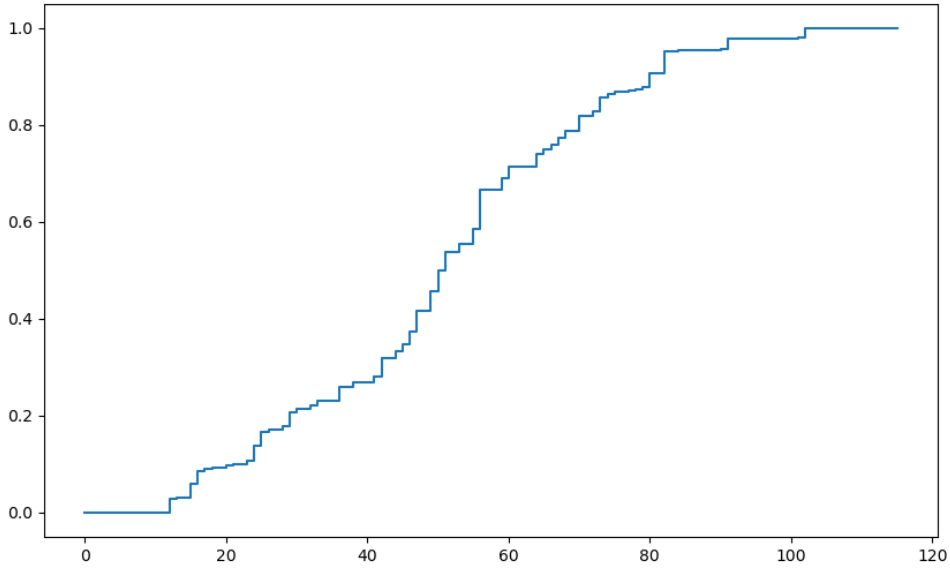

Figure 2: cdf Estimation using the data

In [10]:
```
def expected_revenue(df, p):
    revenue = 0
    for i in range(len(df)):
        if df['p_i'][i] >= p:
            revenue += p
    return revenue/len(df)
```

Now we can compute the optimal price p^*, and plot the revenues curve

In [11]:
```
revenues = [expected_revenue(df3, p) for p in range(max(df3['p_i'] + 1))]
pstar = xw[np.argmax(revenues)]
print('The optimal price is ' + str(pstar))
plt.figure(figsize=(10, 6))
plt.step(xw, revenues, '-')
plt.plot(pstar, max(revenues), 'o')
plt.savefig('revenue.png')
plt.show()
```

Out[11]:
```
The optimal price is 55
```

Therefore, the optimal price that maximizes the expected revenue is $\mathbf{p^* = 55}$. The expected revenues curve is shown in Figure 3.

3. The file `OutOfSample.xlsx` contains willingness-to-pay data points. Each data point represents the willingness-to-pay of a certain customer. Suppose p^* is the optimal price computed in the previous question. Compute the revenue obtained by p^* if the actual demand is given by data in `OutOfSample.xlsx`.

Solution:

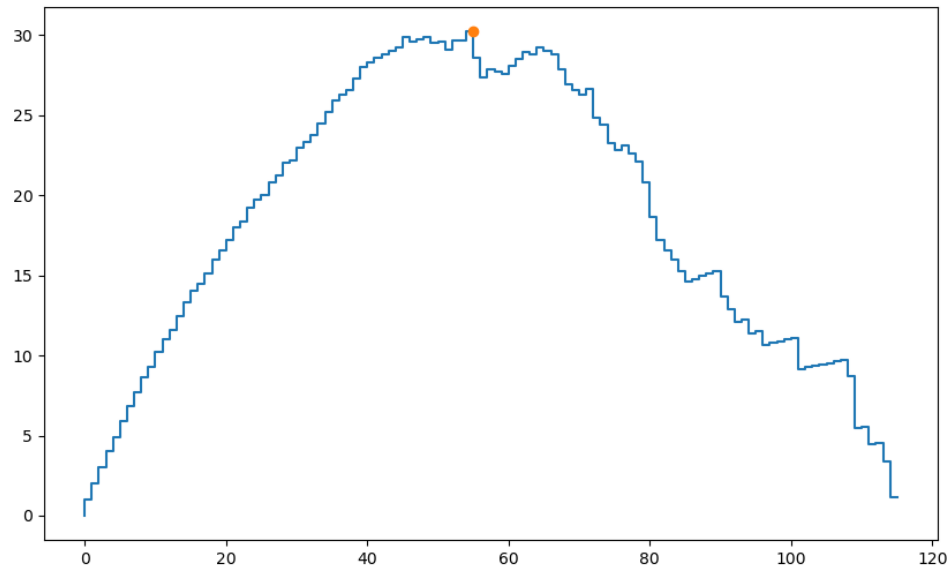

Figure 3: Expected revenues curve

Given a customer with willingness-to-pay equal to W, the revenue obtained by price p is equal to $p \cdot \mathbf{1}(W \geq p)$, indeed the revenue is equal to p if and only if the willingness-to-pay is greater than p. To compute the total revenue we define a function `compute_revenue(W, p)` that takes in input the willingness-to-pay vector and a price p and returns the total revenue

In [12]:
```
def compute_revenue(W, p):
    revenue = 0
    for i in range(len(W)):
        if W[i] >= p:
            revenue += p
    return revenue
```

We can now compute the total revenue with the actual demand

In [13]:
```
df3b = pd.read_excel("OutOfSample.xlsx", sheet_name='Sheet1')
wtp_vector = list(df3b['WTP'])
print('The revenue for the optimal price is ' + str(compute_revenue(
    wtp_vector, pstar)))
```

Out[13]:
```
The revenue for the optimal price is 1045
```

The revenue obtained with the actual data is, therefore, $\$\mathbf{1045}$