Portfolio Presentation

MS in Analytics

Jae Yul Woo

Index

1. MnM : Meal Planning for the New Millennium
2. LEO-Wyndor : Advertising and Production
3. LEO-ELEQUIP : Time Series and Inventory
4. Feasibility Validation of Launching LA Metro Bike Relocation Service

Meal Planning
for the New Millennium

Introduction

- Build 5 weekdays meal plan for 25 year-old male, A and B.
- Utilize Internet resources, crowd-sourced recipes and the idea of optimization method to this completely revamped version of the diet problem from 1930's.
- This problem will focus on meal planning with nutritions, preferences and preparation fairness.

    a. Fairly divide the shopping, cooking, and meal selection responsibilities.
    b. Consider each others' schedule, budget, dietary restriction and preferences.

Data Crawling for Recipes

- Collected from epicurious.com
- Only consider protein, fat, sodium and calories
- Data crawling for 298 recipes with ingredients and personal ratings
    a. Recipes with NaN data and those with less than 9 reviews were deleted. (132 recipes)
    b. Reorganized and analyzed the recipes and users in a matrix to construct the final dataset, recipes-user-rating.
    c. 20 recipes were chosen as the candidates for the weekly meal plan.

ID	Title	Calories (kcal)	Fat(g)	Protein(g)	Sodium(mg)
2	Easy Green Curry with Chicken, Bell Pepper, and Sugar Snap Peas	549	36	42	760
10	Persian-Spiced Chicken with Spaghetti Squash, Pomegranate, and Pistachios	704	44	49	1002
13	Istanbul-Style Wet Burger (Islak Burger)	691	34	46	779
15	Pumpkin Spice Bundt Cake with Buttermilk Icing	372	14	5	282
18	Pumpkin Cheesecake with Bourbon sour Cream Topping	429	32	6	316
24	Kale Salad with Butternut Squash, Pomegranate, and Pumpkin Seeds	162	5	8	539
26	Stuffed Sweet Potatoes with Beans and Guacamole	733	34	23	1509
28	Slow Cooker Pork Shoulder with Zesty Basil Sauce	470	45	8	289
29	Easy General Tso's Chicken	699	35	50	1189
30	White Chicken Chili	534	14	45	968
38	One-Pot Curried Cauliflower with Couscous and Chickpeas	606	15	27	1365
48	Navy Bean and Escarole Stew with Feta and Olives	461	23	21	407
49	Butternut Squash, Kale, and Crunchy Pepitas Taco	233	17	7	363
50	Sheet-Pan Skirt Steak With Balsamic Vinaigrette, Broccolini, and White Beans	820	56	45	977
75	Summer Corn, Tomato, and Salmon Salad with Za'atar Dressing	810	53	40	1372
77	Curried Lentil, Tomato, and Coconut Soup	437	28	13	667
92	Sheet-Pan Cumin Chicken Thighs with Squash, Fennel, and Grapes	497	30	29	1177
104	Curried Pumpkin Soup	165	14	2	373
109	Persian Chicken with Turmeric and Lime	563	23	40	813
113	Baked Feta and Greens with Lemony Yogurt	577	38	23	839

Table 1-1. Nutrition Data for 20 popular recipes

Nutritional Restrictions and Additional Information

- Missing prices were filled its' the sum of all ingredients' price in the recipe at instacart.com
- Preparation time of each recipes are filled following our own experiences.
- Available hours, free times, are set as 2 hours (120 minutes) everyday for both A and B
- Nutritional restrictions were referred from mydailyintake.net

	Calories (Kcal)	Protein (g)	Fat (g)	Sodium (mg)
Lower	2700	140	150	2400
Upper	3600	200	250	3000

Table 1-3. Nutritional Restrictions

ID	Title	Price ($)	Preparation Time (min.)
2	Easy Green Curry with Chicken, Bell Pepper, and Sugar Snap Peas	7.2	30
10	Persian-Spiced Chicken with Spaghetti Squash, Pomegranate, and Pistachios	5.1	25
13	Istanbul-Style Wet Burger (Islak Burger)	4.3	25
15	Pumpkin Spice Bundt Cake with Buttermilk Icing	3.1	50
18	Pumpkin Cheesecake with Bourbon sour Cream Topping	4.5	120
24	Kale Salad with Butternut Squash, Pomegranate, and Pumpkin Seeds	4.3	10
26	Stuffed Sweet Potatoes with Beans and Guacamole	5.2	25
28	Slow Cooker Pork Shoulder with Zesty Basil Sauce	9.8	30
29	Easy General Tso's Chicken	6.1	30
30	White Chicken Chili	6.5	25
38	One-Pot Curried Cauliflower with Couscous and Chickpeas	4.5	35
48	Navy Bean and Escarole Stew with Feta and Olives	5.5	20
49	Butternut Squash, Kale, and Crunchy Pepitas Taco	3.5	30
50	Sheet-Pan Skirt Steak With Balsamic Vinaigrette, Broccolini, and White Beans	13	40
75	Summer Corn, Tomato, and Salmon Salad with Za'atar Dressing	12.5	20
77	Curried Lentil, Tomato, and Coconut Soup	4.8	45
92	Sheet-Pan Cumin Chicken Thighs with Squash, Fennel, and Grapes	7.5	25
104	Curried Pumpkin Soup	3.5	60
109	Persian Chicken with Turmeric and Lime	6.3	25
113	Baked Feta and Greens with Lemony Yogurt	5	30

Table 1-2. Price and Preparation Time

Rating matrix

- Randomly condense 40 user as 1 user. (not area like state)
- Choose 20 recipes with least Nan
- Add our own row(randomly choose some recipes to fill)

54 x 20

	Sheet-Pan Skirt Steak With Balsamic Vinaigrette, Broccolini, and White Beans	Navy Bean and Escarole Stew with Feta and Olives	Baked Feta and Greens with Lemony Yogurt	White Chicken Chili	Easy General Tso's Chicken	Easy Green Curry with Chicken, Bell Pepper, and Sugar Snap Peas	Persian-Spiced Chicken with Spaghetti Squash, Pomegranate, and Pistachios	Summer Corn, Tomato, and Salmon Salad with Za'atar Dressing	Pumpkin Cheesecake with Bourbon–Sour Cream Topping	Curried Lentil, Tomato, and Coconut Soup	Curried Pumpkin Soup	Stuffed Sweet Potatoes with Beans and Guacamole	One Cu Cauliflc Cous Chick
49	NaN	NaN	NaN	4.0	5.0	5.0	5.0	NaN	NaN	NaN	NaN	NaN	
50	NaN	2.0	5.0	NaN	NaN	5.0	3.0	NaN	NaN	5.0	5.0	3.0	
51	5	NaN	NaN	NaN	NaN	NaN	NaN	5.0	NaN	NaN	5.0	5.0	
52	5	5.0	5.0	NaN	NaN	NaN	NaN	5.0	5.0	5.0	NaN	NaN	
53	NaN	1.0	1.0	NaN	1.0	1.0	NaN	NaN	NaN	1.0	NaN	NaN	

SVD

$$\begin{pmatrix} & r_{ui} & \end{pmatrix} = \begin{pmatrix} & - & p_u & - & \end{pmatrix} \begin{pmatrix} | \\ q_i \\ | \end{pmatrix}$$

	Unnamed: 0	X	Sheet Pan Skirt Steak With Balsamic Vinaigrette Broccolini and White Beans	Navy Bean and Escarole Stew with Feta and Olives	Baked Feta and Greens with Lemony Yogurt	White Chicken Chili	Easy General Tso s Chicken	Easy Green Curry with Chicken Bell Pepper and Sugar Snap Peas	Persian Spiced Chicken with Spaghetti Squash Pomegranate and Pistachios	Summer Corn Tomato and Salmon Salad with Za atar Dressing	...	Curried Pumpkin Soup	Stuffed Sweet Potatoes with Beans and Guacamole	One P Curri Cauliflow wi Cousco a Chickpe
51	52	51	5.000000	4.770857	5.388502	4.862310	5.392882	5.272162	4.389102	5.000000	...	5.000000	5.000000	4.3045
52	53	52	5.000000	5.000000	5.000000	4.691488	5.237685	5.150414	4.235204	5.000000	...	4.044653	4.605018	4.0000
53	54	53	1.340146	1.000000	1.000000	0.466433	1.000000	1.000000	0.426214	1.100724	...	-0.340423	-0.059442	-0.0996
53	54	53	4.0	2.489010	1.000000	3.000000	3.398837	3.814655	5.000000	3.240246	...	2.000000	2.425446	2.18033

Mixed Integer Programming Mathematical Formulation

- The preferences on the recipes measured by predicted ratings are used to maximize the satisfaction for the derived meal plan.

- Nutrition, free time, budget and fair preparation schedule are used as the constraints for the meal plan.

- $x_i \in (0,1)$ is the decision variable of choosing a recipe among 20 candidate recipes, $y_{ij} \in (0,1)$ is decision variable for recipe i on day j for A, $z_{ij} \in (0,1)$ is decision variable for recipe i on day j for B.

$x_i \in (0,1)$	Decision variable for a recipe	c_i, p_i, f_i, s_i	Calories, protein, fat, sodium in recipe i
$y_{ij} \in (0,1)$	A's decision for recipe i on day j	$price_i$	Price($) in recipe i
$z_{ij} \in (0,1)$	B's decision for recipe i on day j	Pt_i	Preparation time for recipe i
R_i	Rating for recipe i	$At1_j, At2_j$	Available time for each person's day j
C_{lb}, C_{ub}	Calories lower bound and upper bound	c_{adj}, p_{adj}	Control parameter for calories, protein
P_{lb}, P_{ub}	Protein(g) lower bound and upper bound	f_{adj}, s_{adj}	Control parameter for fat and sodium
F_{lb}, F_{ub}	Fat(g) lower bound and upper bound	α	Control parameter for time balance, 0.01
S_{lb}, C_{ub}	Sodium(mg) lower bound, upper bound	w	Meal preparation time difference

Table 1-4. Notation for Mixed Integer Programming

Mixed Integer Programming Mathematical Formulation

$$Maximize \sum_{i=1}^{20} R_i \times x_i - \alpha * w$$

$$subject\ to \sum_{i=1}^{20} x_i = 5$$

Selection Constraints

$$\sum_{i=1}^{20}\sum_{j=1}^{5} y_{ij} \leq 5$$

$$\sum_{i=1}^{20}\sum_{j=1}^{5} z_{ij} \leq 5$$

$$\sum_{i=1}^{20} x_i = \sum_{i=1}^{20}\sum_{j=1}^{5} y_{ij} + \sum_{i=1}^{20}\sum_{j=1}^{5} z_{ij}$$

$$|\sum_{i=1}^{20}\sum_{j=1}^{5} y_{ij} - \sum_{i=1}^{20}\sum_{j=1}^{5} z_{ij}| \leq 1$$

$$\sum_{i=1}^{20} y_{ij} + \sum_{i=1}^{20} z_{ij} = 1,\ j \in (1,2,3,4,5)$$

One-one Constraint

$$C_{lb} \leq \sum_{i=1}^{20} x_i \times c_i \leq C_{ub} \times \boxed{c_{adj}}$$

Nutritional Constraints

$$P_{lb} \times \boxed{p_{adj}} \leq \sum_{i=1}^{20} x_i \times p_i \leq P_{ub}$$

$$F_{lb} \leq \sum_{i=1}^{20} x_i \times f_i \leq F_{ub} \times \boxed{f_{adj}}$$

$$S_{lb} \leq \sum_{i=1}^{20} x_i \times s_i \leq S_{ub} \times \boxed{s_{adj}}$$

$$\sum_{i=1}^{20} Pt_i \times y_{ij} \leq At1_j,\ j \in (1,2,3,4,5)$$

$$\sum_{i=1}^{20} Pt_i \times z_{ij} \leq At2_j,\ j \in (1,2,3,4,5)$$

Preparation Time Constraints

$$\sum_{i=1}^{20}\sum_{j=1}^{5} y_{ij} - \sum_{i=1}^{20}\sum_{j=1}^{5} z_{ij} \leq w$$

$$-\sum_{i=1}^{20}\sum_{j=1}^{5} y_{ij} + \sum_{i=1}^{20}\sum_{j=1}^{5} z_{ij} \leq w$$

Time Balance Constraints

$$\sum_{i=1}^{20} y_{ij} \leq 1,\ j \in (1,2,3,4,5)$$

$$\sum_{i=1}^{20} z_{ij} \leq 1,\ j \in (1,2,3,4,5)$$

Redundancy Constraints

$$\sum_{j=1}^{5} x_{ij} \leq 1,\ i \in (1,2,3,....19,20)$$

$$\sum_{i=1}^{20} price_i \times x_i \leq 100$$

Budget Constraints

MIP Formulation with Pyomo

- Combine the nutrition data we got from ‘Epicurious’ and the expected rating data from the SVD prediction.

	Unnamed: 0	title	calories	fat	protein	sodium	rating
5	2	Easy Green Curry with Chicken, Bell Pepper, an...	549.0	36.0	42.0	760.0	3.814655
11	10	Persian-Spiced Chicken with Spaghetti Squash, ...	704.0	44.0	49.0	1002.0	5.000000
6	13	Istanbul-Style Wet Burger (Islak Burger)	691.0	34.0	46.0	779.0	4.000000
13	15	Pumpkin Spice Bundt Cake with Buttermilk Icing	372.0	14.0	5.0	282.0	1.000000
12	18	Pumpkin Cheesecake with Bourbon–Sour Cream Top...	429.0	32.0	6.0	316.0	1.641707

- Create list of parameters for ratings and nutrition restrictions, etc…

```
# We have determined :
price = [7.2, 5.1,4.3,3.1,4.5,4.3,5.2,9.8,6.1,6.5,4.5,5.5,3.5,13,12.5,4.8,7.5,3.5,6.3,5]
p_times = [30,25,25,50,120,10,25,30,30,25,35,20,30,40,20,45,25,60,25,30]
available_time_y = [120,120,120,120,120]
available_time_z = [120,120,120,120,120]

# lower(1) and upper(2) bound of nutritions
c_bound=[2700, 3600]
p_bound= [140 ,200]
f_bound=[150 ,250]
s_bound= [2400 ,3000]
```

- Use Pyomo for MIP modeling with ‘Gurobi_ampl’ solver.
- Build the Pyomo model as a function so that it can be used repeatedly with different scenarios each week. (Such as more proteins, less sodium etc…)

``

```
def mnmplans(cal_rate=0,pro_rate=0,fat_rate=0,sod_rate=0):
    # Define parameters
    cal_ = 1-(cal_rate/100)
    pro_ = 1+(pro_rate/100)
    fat_ = 1-(fat_rate/100)
    sod_ = 1-(sod_rate/100)
```

Conclusion

- Scenario 1 : adjustment parameter value of 0 for $c_{adj}, p_{adj}, f_{adj}, s_{adj}$.
  - mnmplans(cal_rate=0, pro_rate=0, fat_rate=0, sod_rate=0)
  - x0 = 1, x18 = 1, x2 = 1, x4 = 1, x7 = 1, y00 = 1, y22 = 1, y71 = 1, z183 = 1, z44 = 1

	Day 1	Day 2	Day 3	Day 4	Day 5
A	Easy Green Curry with Chicken, Bell Pepper, and Sugar Snap Peas	Slow Cooker Pork Shoulder with Zesty Basil Sauce	Istanbul-Style Wet Burger (Islak Burger)	-	-
B	-	-	-	Curried Pumpkin Soup	Pumpkin Cheesecake with Bourbon Sour Cream Topping

Table 1-5. Meal Plan with no adjustments

- Scenario 2 : 20% restriction on calories, adjustment parameter as 20(%).
  - mnmplans(cal_rate=20)
  - x0 = 1, x18 = 1, x2 = 1, x4 = 1, x7 = 1, y184 = 1, y42 = 1, z00 = 1, z23 = 1, z71 = 1

	Day 1	Day 2	Day 3	Day 4	Day 5
A	-	-	Pumpkin Cheesecake with Bourbon Sour Cream Topping	-	Curried Pumpkin Soup
B	Easy Green Curry with Chicken, Bell Pepper, and Sugar Snap Peas	Slow Cooker Pork Shoulder with Zesty Basil Sauce	-	Istanbul-Style Wet Burger (Islak Burger)	-

Table 1-6. Meal Plan with 20% Calorie restriction

LEO-Wyndor : Advertising and Production

Introduction

- Maximization of the total revenue on the door production with respect to advertisement plans.
- Resource utilization for the production of high quality glass doors A and B in plants 1,2,3.

Plant	Prod.time for A	Prod.time for B	Total Hours
1	1	0	8
2	0	2	24
3	3	2	36
Profit per Batch	$3,000	$5,000	

Table 2-1. Data for the Wyndor Glass Problem

- The product mix to maximize the total revenue will be decided by the production plan, as well as the potential of future sales.
- Sales is assumed to be depended on the marketing strategy of “Wyndor Glass Co”.
  a. The advertisement on two different media, TV and radio in dollar amount.
  b. Sales predictions are based on the TV and radio advertising effect.
- Compare Statistical Learning prediction and Learning Enabled Optimization to get reliable and realizable predictions of the future revenue.

Predictive Model

- Multiple Linear Regression model is used to measure the total sales on each possible pair amounts of TV and Radio advertisement.
  a. Splitted data into training and validation data, 50% each with random seed 1.
  b. The trained model : $y_{Sales} = 3.74982951 + 0.05370186 \times x_{TV} + 0.22231025 \times x_{Radio}$
  c. Using the trained model, we find the difference between the actual sales data, W_i and predicted sales data, Z_i. (Empirical Additive Errors)
- The difference is denoted as $\xi_{i,T}$ and $\xi_{j,V}$ for train and validation data, respectively.
  a. $\xi_{i,T} = Z_{i,T} - W_{i,T}$ for all $i \in Training$ and $\xi_{j,V} = Z_{j,V} - W_{j,V}$ for all $j \in Validation$
- We plotted qq-plot to determine the outliers.
  a. Remove two outliers and refit the MLP.
  b. The final predictive model for the sales prediction is below.

$$y_{Sales} = 4.07459792 + 0.05119161 \times x_{TV} + 0.22734845 \times x_{Radio}$$

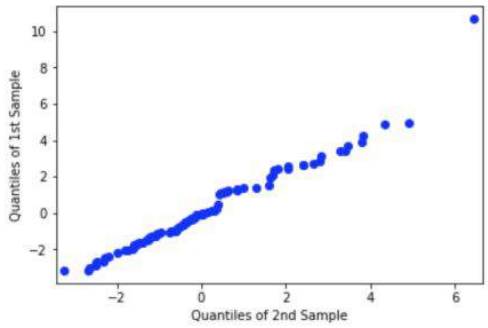


Figure 2-2. qq-Plot for $\xi_{i,T}$ and $\xi_{j,V}$

Statistical Learning

- The deterministic linear programming model for the statistical learning is formulated as model 2-1.1
  - Total sales constraint: Upper bound of predicted sales by the advertisement.
- = Linear program with 4 decision variables (x_1, x_2, y_a, y_b) and 9 constraints.

$$
\begin{aligned}
&Maximize: -0.1 \times x_1 - 0.5 \times x_2 + 3 \times y_a + 5 \times y_b \\
&subject\ to: y_a \leq 8 \\
&\qquad y_b \leq 24 \\
&\qquad 3 \times y_a + 2 \times y_b \leq 36 \\
&\qquad y_a + y_b \leq \beta_0 + \beta_1 \times x_1 + \beta_2 \times x_2 \\
&\qquad x_1 + x_2 \leq 200 \\
&\qquad x_1 - 0.5 \times x_2 \geq 0 \\
&\qquad x_1,\ x_2,\ y_a,\ y_b\ \geq 0 \\
&\qquad l_1 \leq x_1 \leq u_1 \\
&\qquad l_2 \leq x_2 \leq u_2
\end{aligned}
$$

Model 2-1.1. Deterministic MLP

	Optimal Solution
Objective	48.16914111658245
$x_{1,D}$	190.42285279145614
$x_{2,D}$	9.577147208543863
y_a	4.0
y_b	12.0

Table 2-2. Optimal Solution for Statistical Learning

Statistical Learning

- Validation of the statistical learning model with the possible errors derived from the validation dataset.
  a. Optimal solution $(x_{1,D}, x_{2,D})$ is applied to the model 2-1.2
  b. $\xi_{j,V}$ is introduced as the possible 100 errors during the validation procedure.

$$
\begin{aligned}
&Maximize: -0.1 \times x_{1,D} - 0.5 \times x_{2,D} + 3 \times y_a + 5 \times y_b \\
&subject\ to: y_a \leq 8 \\
&\qquad\qquad y_b \leq 24 \\
&\qquad\qquad 3 \times y_a + 2 \times y_b \leq 36 \\
&\qquad\qquad y_a + y_b - \beta_0 - \beta_1 \times x_{1,D} - \beta_2 \times x_{2,D} \leq \xi_{j,V} \\
&\qquad\qquad y_a,\ y_b\ \geq 0
\end{aligned}
$$

Model 2-1.2. Validation model for Deterministic MLP

- Derived 100 optimal objective values from the model 2-1.2. The 95% confidence interval for the optimal objective value is (45.56898, 46.57599).

Learning Enabled Optimization

- The stochastic linear programming model for the learning enabled optimization is formulated as model 2-2.1
  - Total sales constraint: The probabilistic right-hand-side with the empirical additive errors.
- Solve the Stochastic programming with the SD solver by PySP formulation with 4 decision variables (x_1, x_2, y_a, y_b) and 9 constraints.
- To prevent the objective becomes negative value, we added big M to the objective function and set M as 200.

$$
\begin{aligned}
&Minimize: 0.1 \times x_1 + 0.5 \times x_2 - 3 \times y_a - 5 \times y_b + M \\
&subject\ to: y_a \leq 8 \\
&\qquad y_b \leq 24 \\
&\qquad 3 \times y_a + 2 \times y_b \leq 36 \\
&\qquad y_a + y_b - \beta_1 \times x_1 - \beta_2 \times x_2 \leq \beta_0 + \xi_i \\
&\qquad x_1 + x_2 \leq 200 \\
&\qquad x_1 - 0.5 \times x_2 \geq 0 \\
&\qquad x_1,\ x_2,\ y_a,\ y_b\ \geq 0 \\
&\qquad l_1 \leq x_1 \leq u_1 \\
&\qquad l_2 \leq x_2 \leq u_2
\end{aligned}
$$

Model 2-2.1 Stochastic MLP

	Optimal Solution
Objective	45.9991
$x_{1,S}$	1.983564e+02
$x_{2,S}$	1.643590e+00

Table 2-3. Optimal Solution for Learning Enabled Optimization

Learning Enabled Optimization

- Validation of the learning enabled model with the EAE derived from the validation dataset.
  a. Optimal solution $(x_{1,S}, x_{2,S})$ is applied to the model 2-1.2
  b. $\xi_{i,T}, \xi_{j,V}$ is introduced as the possible 100 errors for the train and validation procedure, respectively.

$$
\begin{aligned}
&Maximize: -0.1 \times x_{1,D} - 0.5 \times x_{2,D} + 3 \times y_a + 5 \times y_b \\
&subject\ to: y_a \leq 8 \\
&\qquad\qquad y_b \leq 24 \\
&\qquad\qquad 3 \times y_a + 2 \times y_b \leq 36 \\
&\qquad\qquad y_a + y_b - \beta_0 - \beta_1 \times x_{1,D} - \beta_2 \times x_{2,D} \leq \xi_{i,T}/\xi_{j,V} \\
&\qquad\qquad y_a,\ y_b\ \geq 0
\end{aligned}
$$

Model 2-2.2. Training and Validation model for Deterministic MLP

- The stochastic model needs to confirm if the probabilistic distribution for the training objective values corresponds with the validation objective values.
  → Conducted the Kruskal Wallis H-test : H-statistic: 0.313, P-Value: 0.576
- We can conclude that there's no significant difference between groups.

Conclusion

- The 95% confidence interval for the predicted revenue of the validation dataset

  SL: ($45.56898, $46.57599) (in 1000s)

  LEO: (45.40, 46.90) (in 1000s)

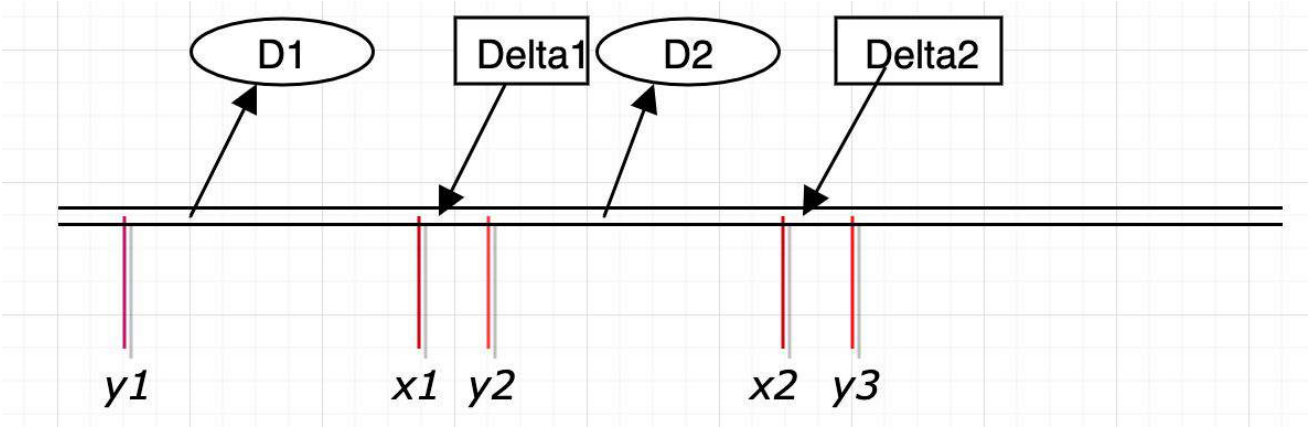
- The statistical model with the MLP overestimated the revenue as **$48.17 (in 1000s).**
  a. Reasonable outcome in that the regression model did not consider error terms which will depreciate the expected revenue.
- The learning enabled optimization model estimated the total revenue as **$46.00 (in 1000s).**
- The total revenue by LEO model is in the 95% confidence interval and the distribution of the training objective values corresponds with the validation objective values.
- Hence, we concluded that the LEO model gives more probable and reliable outcome for the prediction than the statistical learning model itself.

LEO-ELEQUIP :
Time Series and Inventory

- Introduction
- Time series model
- Deterministic linear programming
- Stochastic programming
- Comparison

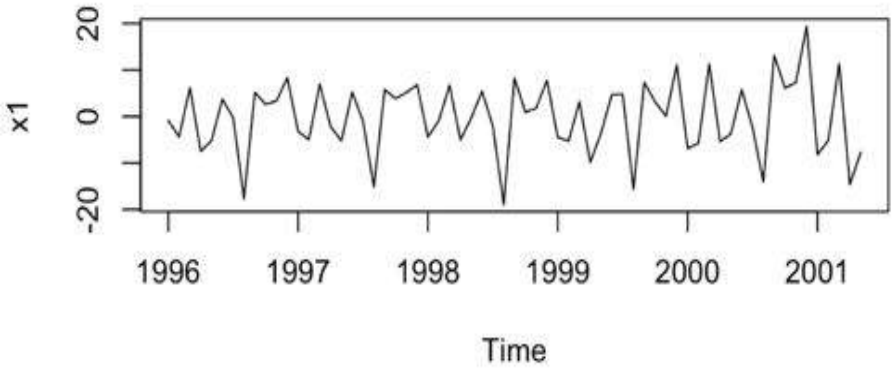
Introduction

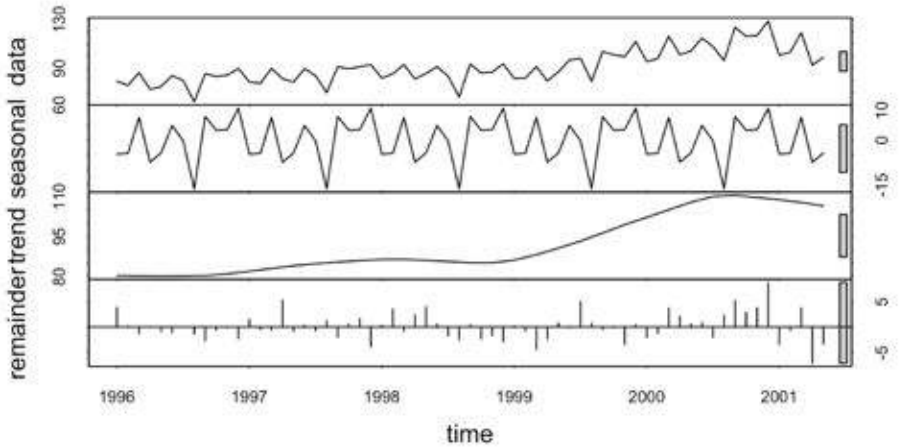
- Minimize expected cost of holding plus the expected cost of lost sales.
- Select training set from 1996 to 2001 to build ARIMA model and predict demand
- calculate 6 months holding cost and lost sales cost
- Comparing cost of DLP model and SP model



Time series model

- Processing outliers(tsclean)
- Remove trend part, check stationarity(kpss test ,p-value =0.1, do not reject H0)
- Use 60 months to build stationary ARIMA model(1996-2000)
- Forcast next 3 months demands
- Iteration: use 60+k months to build model and forcast next 3 months until we can get 6 months order
- Save all the forcasting mean demand and error term





Deterministic linear programing

$$Min \sum_{t=0}^{T-1} h_t x_t + b_t z_t$$

$subject.to.$

$$y_{t+1} - x_t - \Delta_t = 0$$

$$-y_t + x_t >= -D_t$$

$$y_t + z_t >= D_t$$

$$\Delta_t <= U_t$$

$$x_t, z_t, \Delta_t >= 0$$

$$Hoding\ cost = h_t * (y_t - D_{real}) = h_t x_t$$

$$Lost\ sale\ cost = b_t * (D_{real} - y_t) = b_t z_t$$

$$y_{start\ (t+1)} = x_t + \Delta_t or\ \Delta_t - z_t$$

AMPL code

```
var x1 >=0;
var x2 >=0;
var x3 >=0;
var y1 >=0;
var y2 >=0;
var y3 >=0;
var z1 >=0;
var z2 >=0;
var z3 >=0;
var delta1 >=0;
var delta2 >=0;
#var delta3 >=0;
param y_start := 120.83539999999999;
param d1 := 111.3754;
param d2 := 105.8107;
param d3 := 91.53898;
param Ut := 300;

minimize totalcost: x1+x2+x3 + 3*(z1+z2+z3);
#constraints
# U_t = 300
s.t. a1:  y1 = y_start;
s.t. a2:  x1  >=y1 -d1;
s.t. a3:  z1 +y1 >= d1;

s.t. a4:  y2 - x1 -delta1=0;
s.t. a5:  x2  >=y2 -d2;
s.t. a6:  z2 +y2 >= d2;
s.t. a7:  delta1 <= Ut;


s.t. a8:  y3- x2 -delta2 =0;
s.t. a9:  x3  >=y3 -d3;
s.t. a10: z3 +y3 >= d3;
s.t. a11: delta2 <= Ut;
```

d1, d2 ,d3
are mean values
from prediction

Stochastic programing

$$\text{Min } \mathbb{E}_{\tilde{\omega}}[\sum_{t=0}^{T-1} h_t x_t(\tilde{\omega}) + b_t z_t(\tilde{\omega})]$$

$$\begin{aligned}
\text{s.t.} \quad y_{t+1}(\omega) - x_t(\omega) - \Delta_t &= 0 \quad \text{for almost all } \omega \\
y_{t+1}(\omega) &\leq R_{t+1} \quad \text{for almost all } \omega \\
\Delta_t &\leq U_t \\
-y_t(\omega) + x_t(\omega) &\geq -D_t(\omega) \quad \text{for almost all } \omega \\
y_t(\omega) + z_t(\omega) &\geq D_t(\omega) \quad \text{for almost all } \omega \\
x_t(\omega), z_t(\omega), \Delta_t &\geq \mathbf{0}
\end{aligned}$$

- w is the error terms of demand
- assume scenarios are error terms of training data

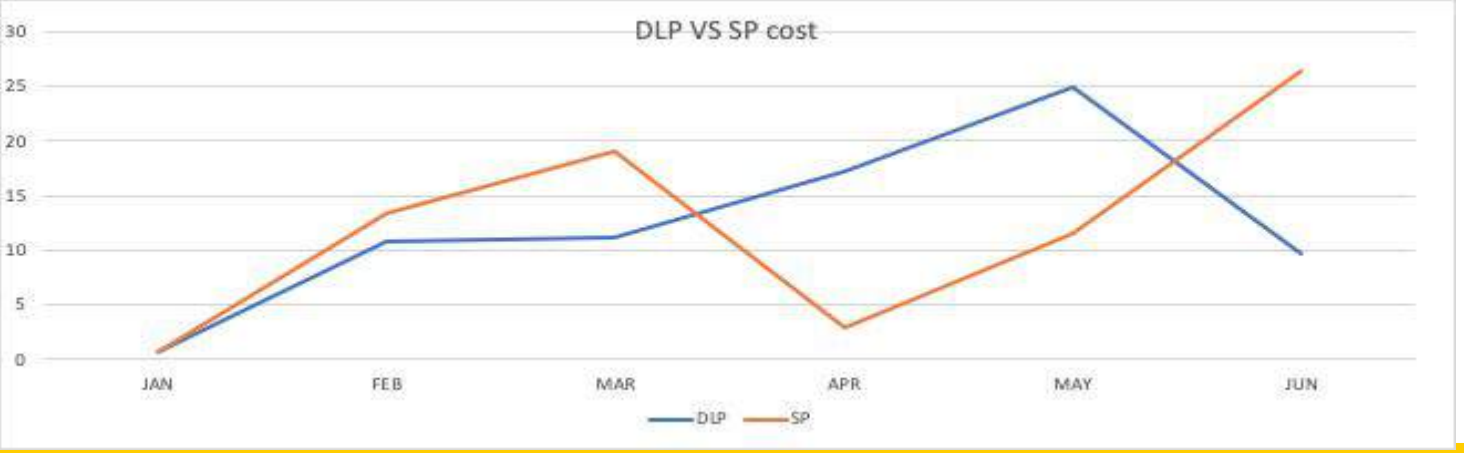
```
22 y_start=110.37
23 demand = pd.read_csv('1/out_mean5.csv')
24 demand = list(map(float, list(demand)))
25
26 error_term = pd.read_csv('1/out_error5.csv', header = None)
27 d1_rhs_table = []
28 d1_rhs_table = list(error_term[0])
```

```
98  model.s11 = Constraint(expr= model.y1 == y_start)
99  model.constraint_stage.declare(model.s11, 2)
100
101 model.s12 = Constraint(expr= model.x1 >= model.y1-demand[0]-model.d1_rhs)
102 model.constraint_stage.declare(model.s12, 2)
103 model.stoch_rhs.declare(model.s12)
104
105 model.s13 = Constraint(expr= model.z1 >= demand[0]+model.d1_rhs - model.y1)
106 model.constraint_stage.declare(model.s13, 2)
107 model.stoch_rhs.declare(model.s13)
108 ##########################################################################
109
110 model.s21 = Constraint(expr= model.y2 == model.x1+ model.delta1)
111 model.constraint_stage.declare(model.s21, 2)
112
113 model.s22 = Constraint(expr= model.x2 >= model.y2-demand[1]-model.d1_rhs)
114 model.constraint_stage.declare(model.s22, 2)
115 model.stoch_rhs.declare(model.s22)
116
117 model.s23 = Constraint(expr= model.z2 >= demand[1]+model.d1_rhs - model.y2)
118 model.constraint_stage.declare(model.s23, 2)
119 model.stoch_rhs.declare(model.s23)
120 ##########################################################################
121 model.s31 = Constraint(expr= model.y3 == model.x2+ model.delta2)
122 model.constraint_stage.declare(model.s31, 2)
123
124 model.s32 = Constraint(expr= model.x3 >= model.y3-demand[2]-model.d1_rhs)
125 model.constraint_stage.declare(model.s32, 2)
126 model.stoch_rhs.declare(model.s32)
127
128 model.s33 = Constraint(expr= model.z3 >= demand[2]+model.d1_rhs - model.y3)
129 model.constraint_stage.declare(model.s33, 2)
130 model.stoch_rhs.declare(model.s33)
```

Comparison of result (DLP VS SP)

DLP	JAN	FEB	MAR	APR	MAY	JUN
real_DEMAND	100.56	103.05	119.06	92.46	98.75	111.14
delta	113.127	119.344	98.5425	106.514	95.8079	
Y	101.38	113.947	130.241	109.7235	123.7775	120.8354
cost	0.82	10.897	11.181	17.2635	25.0275	9.6954
						74.8844

SP	JAN	FEB	MAR	APR	MAY	JUN
real_DEMAND	100.56	103.05	119.06	92.46	98.75	111.14
delta	115.58	99.34	101.83	107.37	90.71	
Y	101.38	116.4	112.69	95.46	110.37	102.33
cost	0.82	13.35	**19.11**	3	11.62	**26.43**
						74.33



Feasibility Validation of Launching LA Metro Bike Relocation Service

Jae Yul Woo

Data Configuration

- Metro Bike share open-data consists of 14 columns (with each data type):
  - Trip ID (int)
  - Duration (float)
  - Start Time (object)
  - End Time (object)
  - Starting Station ID (int)
  - Starting Station Latitude (float)
  - Starting Station Longitude (float)
  - Ending Station ID (int)
  - Ending Station Latitude (float)
  - Ending Station Longitude (float)
  - Bike ID (int)
  - Plan Duration (float)
  - Trip Route Category (string)
  - Passholder Type (string)

- Trip data from July 2016 to June 2017
- Trip Duration (in seconds)
- 64 Stations
  - Station Latitude and Longitude
- Bike ID
- Plan Duration :
  - 1 day, 30 days, 365 days
- Trip Route Category :
  - One Way, Round Trip
- Passholder Type :
  - Walk-up, Monthly Pass, Flex Pass, Staff Annual
- 112427 rows

trip_id	duration	start_time	end_time	start_station	start_lat	start_lon	end_station	end_lat	end_lon	bike_id	plan_duration	trip_route_cat	passholder_type
32815764	18	2017.6.25 19:53	2017.6.25 20:11	3047	34.039982	-118.2664	3005	34.0485	-118.25854	4727	30	One Way	Monthly Pass
32821341	25	2017.6.25 20:35	2017.6.25 21:00	3005	34.0485	-118.25854	3020	34.031052	-118.26709	4727	30	One Way	Monthly Pass
31652471	10	2017.6.16 8:41	2017.6.16 8:51	3023	34.050911	-118.24097	3005	34.0485	-118.25854	4727	30	One Way	Monthly Pass
31700167	4	2017.6.16 17:39	2017.6.16 17:43	3005	34.0485	-118.25854	3051	34.045422	-118.25352	4727	30	One Way	Monthly Pass
31717085	17	2017.6.16 20:17	2017.6.16 20:34	3051	34.045422	-118.25352	3005	34.0485	-118.25854	4727	0	One Way	Walk-up

Data Preprocessing

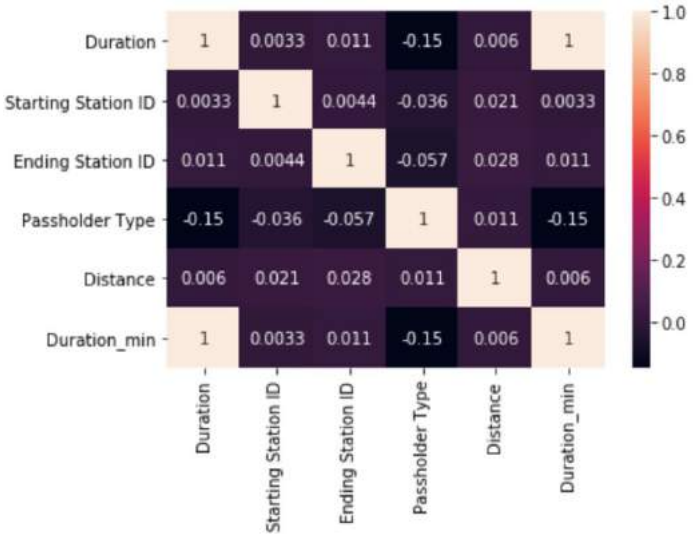
- Excluded... (112427 rows → 88266 rows)
  - 11177 rides with any NaN value
  - 3225 rides by LA Metro ‘Staff Annual’
  - 10459 rides of ‘Round-trips’
  - Station 4108 (Outlier)
- Category 0 (‘Walk-up’’) : 24241, Category 1 (‘Monthly Pass’ + ‘Flex Pass’) : 64025
- Remove non-relevant 9 columns
- Calculate the Manhattan distances in miles between starting stations and ending stations for each trip.
  - For cost estimation after the network optimization.

	Duration	Start Time	Starting Station ID	Ending Station ID	Passholder Type	Distance
5	780	2016-07-07T12:51:00	3021.0	3054.0	1	0.441488
6	600	2016-07-07T12:54:00	3022.0	3014.0	1	0.763951
7	600	2016-07-07T12:59:00	3076.0	3005.0	1	0.624177
9	960	2016-07-07T13:01:00	3031.0	3078.0	1	1.556036
10	960	2016-07-07T13:02:00	3031.0	3047.0	1	0.864494

Data Preprocessing

- Add Duration_min feature to the data for calculating the price for each trip.
- The current price policy is in the table below.
- The correlation between each factor are not significant.

Pass Type	Price		Category
1 Ride	$1.75/ 30min	every 30 min.	0
Monthly	$1.75/ 30min	first 30 min. free	1
Annual	$1.75/ 30min	first 30 min. free	1



EDA

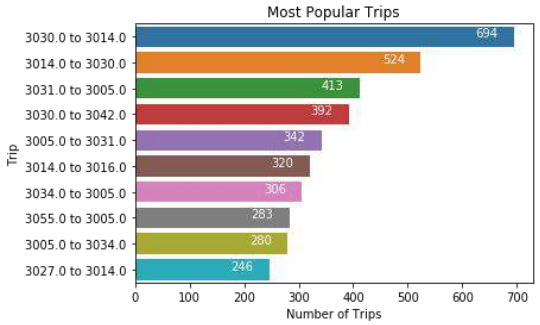
- Passholder Types according to each factor.

- Mean :

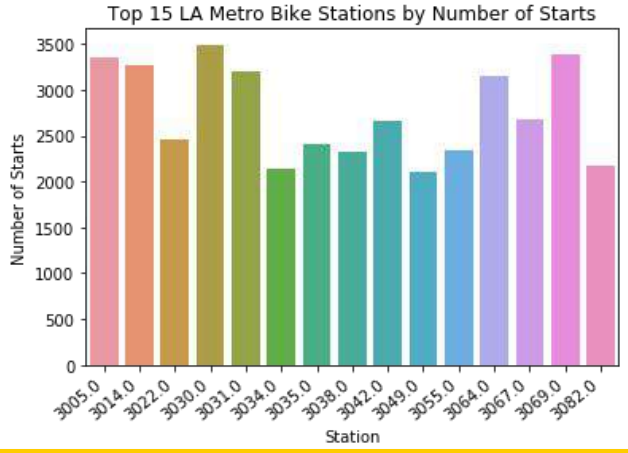
Passholder Type	Duration	Duration_min	Distance
0	2166.942783	36.115713	1.431571
1	771.725107	12.862085	5.347379

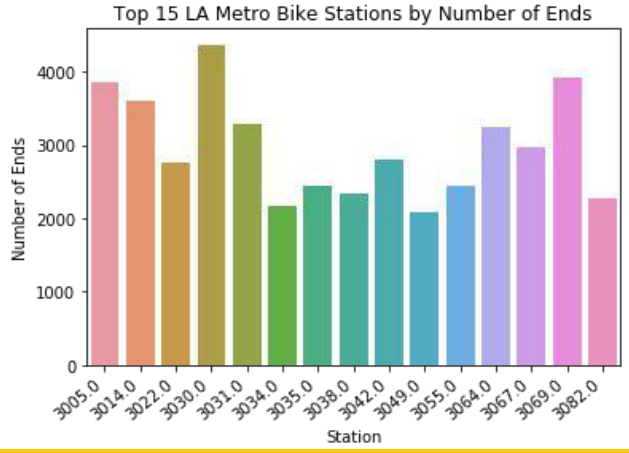
- Median :

Passholder Type	Duration	Duration_min	Distance
0	960	16.0	0.730822
1	480	8.0	0.612904



- The most frequent route : station 3030 to station 3014
- We can see there are imbalances in both starting stations and ending stations





Network Formulation - Supply and Demand

- For the relocation service, stations defined as
  - Supplying Stations : Incoming(Ending Trips) bikes > Outgoing(Starting Trips) bikes
  - Demanding Stations : Incoming bikes < Outgoing bikes

- Number of Supplying Stations and Demanding Stations

```
# Number of Supply stations and Demand stations
print(len(s_data), len(d_data))
```

```
36 27
```

- Ex) Supplying Stations (total 36) Demanding Stations (total 27)

Ending Station ID	Net
0	1012
1	143
3	107
5	191
6	587

Starting Station ID	Net
2	488
4	39
7	21
14	731
16	230

- Store the data into dictionaries. (key: Station ID, value: amount)
  - s_data_constraint = {0: 1012, 1: 143, 3: 107, …}
  - d_data_constraint = {2: 488, 4: 39, 7: 21, …}

Network Formulation - Route Price

- Assume we discount the full price of ‘Walk-up’ trips and reward the full price to ‘Passholders’
  - We only consider the mean duration time to get the ‘full Price’ for each path.

	Ending Station ID	Starting Station ID	Duration	Duration_min	Distance	Price
0	0	1	525.974026	8.766234	0.248361	1.75
1	0	2	1255.636364	20.927273	0.288265	1.75
2	0	3	1032.000000	17.200000	0.249927	1.75
3	0	4	1710.000000	28.500000	0.828233	1.75
4	0	5	2896.800000	48.280000	0.724004	3.50

- Store the full discount price into a dictionary. (key: route, value: price)
  - example : cost_constraint = {(0,1) : 1.75, (0,2) : 1.75, … }
- We need 972 (36*27) pairs but we only have 838 pairs ← 124 pairs with no trips in the original dataset.
  - For these pairs, we use big M method on the constraints to get 0 moves.
- Store the missing pairs into a dictionary. (key: route, value: big M=200)
  - missing_pairs = {(1,60) : 200, (3,7) : 200, (3,38) : 200, …}
- After deep-copy cost_constraint and update it with missing_pairs, we get 972 pairs.

Optimization Formulation

For each pair, parameters $Price[s, d]$ denote the `maximum discount dollar-amount` from the current price policy. $R_{discount} \in [0, 1]$ is defined as the `discount rate` for the 'Walk-ups' and $Price[s, d] \times R_{discount}$ is defined as the `reward dollar-amount` for 'Passholders.' Then, this becomes the company's cost for relocation service. The model solver will determine `amount of bikes to be relocated` over each pair, which will be represented as non-negative integer decision variables $x[s, d]$.

- The discount price for 'walk-ups' and the reward for 'passholders' are $Price[s, d] \times R_{discount}$
- By observing trade-off between discount rates and demands people will use Metro Bike, the realized demand rate is set as a function of the discount rate. Hence, the demand will become $Demand[d] \times f_{realized}(R_{discount})$
- The difference between realized demand and maximum demand (when discount rate is 1) is $(Demand[d] - x[s, d])$. This becomes cost as well.

Optimization Formulation

The problem objective is to minimize the total cost to all demanding stations from all supplying stations.

$$Minimize \sum_{s \in Supply} \sum_{d \in Demand} Price[s,d] \times R_{discount} \times x[s,d] + (price\_per\_mile) \times Distance[s,d] \times (Demand[d] - x[s,d])$$

$$subject\ to \sum_{d \in Demand} x[s,d] \leq \text{Supply}[s] \qquad \forall s \in Supply$$

$$\sum_{s \in Supply} x[s,d] = \text{Demand}[d] \times f_{realized}(R_{discount}) \qquad \forall d \in Demand$$

$$x[s,d]\ is\ integer.$$

> Relocation demands from all sources can not exceed the supplying capacity. Relocations to each station must satisfy their demand with respect to the discount rate.

Optimization Objective

- Find the optimal discount rate for the launch of relocation service.
- Find the routes (Starting Station → Ending Station) of relocation services and the number of discounted trips for each route.
- Find the range of outsourcing cost which makes the relocation service profitable.
    - After the launch of relocation service, the operation cost can be fluctuate.

Network Assumption

- The discount dollar amount for non-passholders and reward dollar amount for passholders are the same. (for modeling simplicity)
- The optimal moves will always be met when the price is free to users (when the discount rate is 1).
- The discount rate for prices and the relocation demand percentage for each trip is the same.
    - When the price is discounted by 10%, 10% of people on each route will use the relocation service.

Network Optimization - Basic Scenario

- There are 62 paths when relocation service is free.
  - Discount rate = 1, all the demands are met.

- Each 62 paths are…

	S	D	Optimal Moves
4	0	16	71.0
6	0	18	271.0
13	0	35	337.0
14	0	36	109.0
16	0	38	1.0
18	0	40	206.0
19	0	44	17.0
32	1	17	95.0
35	1	20	32.0
37	1	29	16.0
58	3	16	107.0
81	5	2	191.0
131	6	52	587.0

etc ...

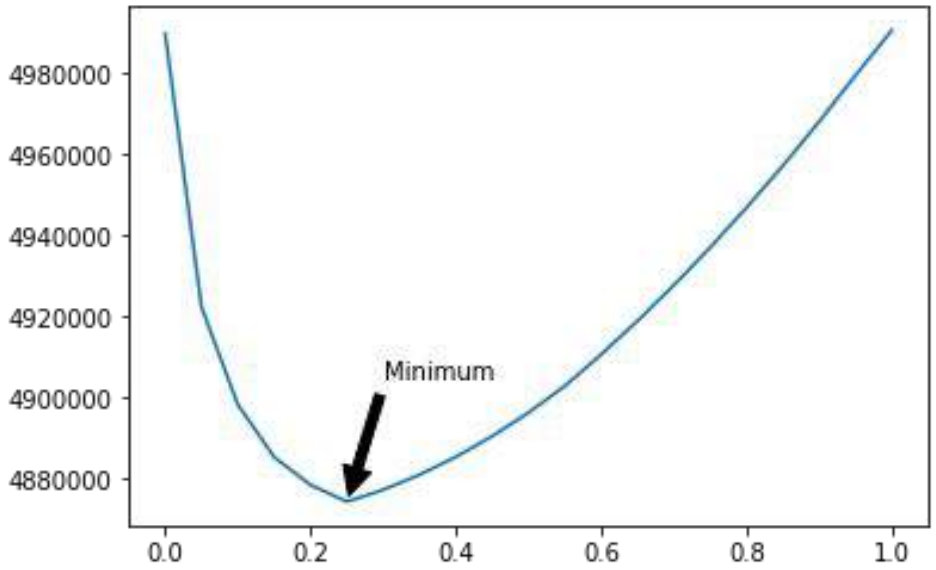
Network Optimization (1)

- **Optimal discount rate to minimize the total cost.**

➔ Used Pyomo for network optimization and solve the problem with gurobi solver.

➔ Found the realistic *price_per_mile* is $1/mile in the LA area.

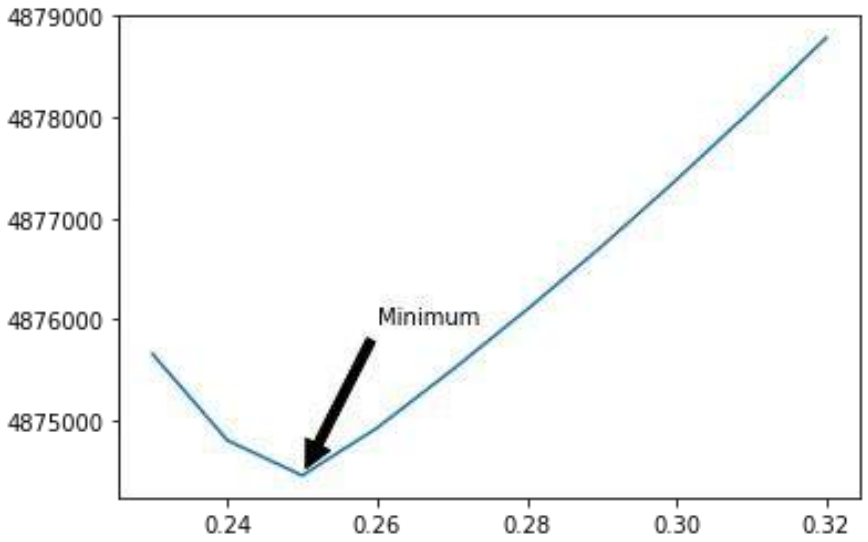
➔ Discount rate from 0% to 100%.



Network Optimization (1) - Detailed

- **Optimal discount rate to minimize the total cost.**

➔ Discount rate from 22% to 32%.



➔ We can conclude that the optimal discount price for minimizing the total cost is **25%**.

Network Optimization (1)

- **Optimal discount rate to minimize the total cost.**

➔ Set discount rate = 0.25

➔ We get the optimal solution of total cost, $4,061,567.47

➔ Total relocation movements are 7663.

➔ The optimal moves from supplying station to demanding station are stored in ‘optimal_sol.csv’

```
Solver:
- Status: ok
  Message: Gurobi 8.1.0\x3a optimal solution; objective 4061567.471694626
4; 93 simplex iterations
  Termination condition: optimal
  Id: 0
  Error rc: 0
  Time: 0.05458498001098633
```

```
# Store optimal solution into dataframe
optsol = []
for s in S_list:
    for d in D_list:
        optsol.append((s, d, model.x[s,d]()))
cols=['S','D','Optimal Moves']
optsol = pd.DataFrame(optsol,columns=cols)
optsol = optsol[optsol['Optimal Moves']!=0]
opts_list = optsol['S'].unique().tolist()
optd_list = optsol['D'].unique().tolist()
# optsol.to_csv('optimal_sol.csv')
optsol['Optimal Moves'].sum()
# Total movements for every route pairs.
```

```
7663.0
```

	S	D	Optimal Moves
3	0	14	103.0
5	0	17	117.0
7	0	19	15.0
8	0	20	131.0

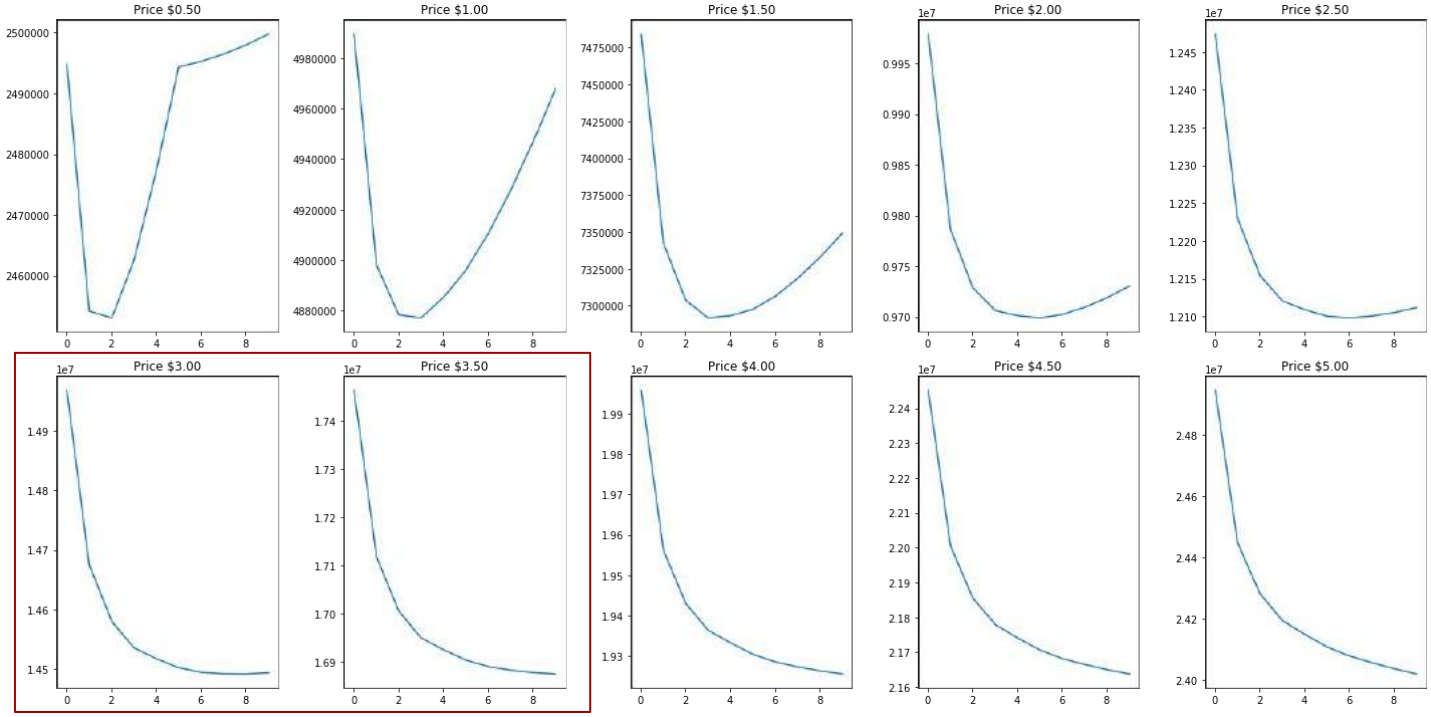
etc ...

Network Optimization (2)

- **Estimation of outsourcing cost to determine relocation service feasibility. <Criteria on the need of providing relocation service at free price.>**

➔ Used Pyomo for network optimization and solve the problem with gurobi solver.

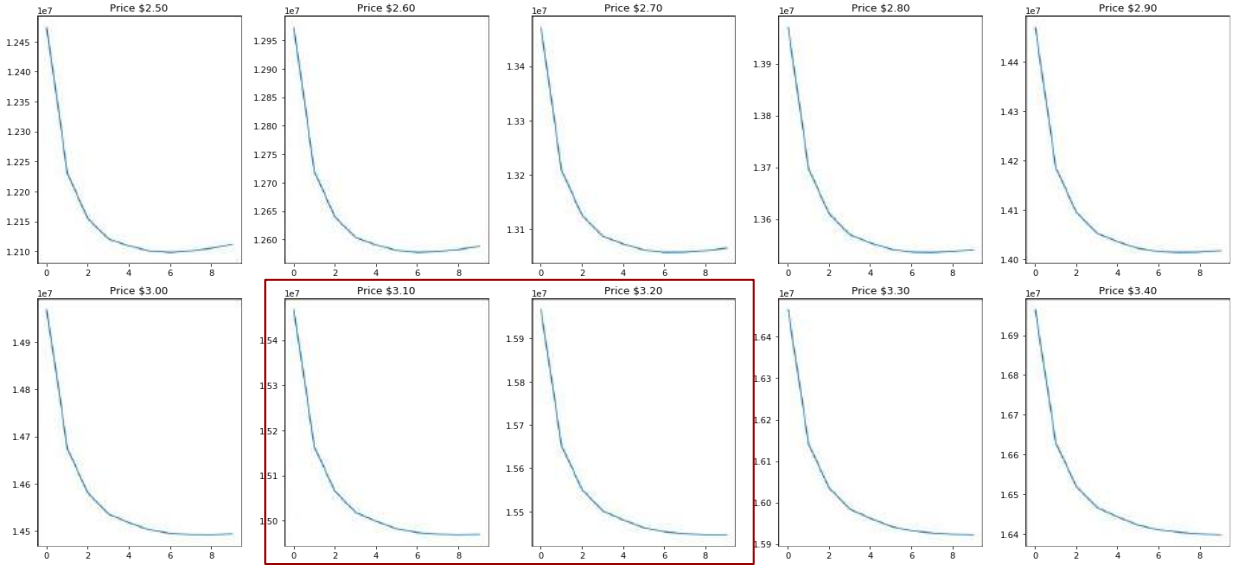
➔ Changed *price_per_mile* from $0.50/mile to $5.00/mile



Network Optimization (2) - Detailed

- **Estimation of outsourcing cost to determine relocation service feasibility. <Criteria on the need of providing relocation service at free price.>**

➔ Changed *price_per_mile* from $2.50/mile to $3.40/mile



➔ Total cost decreases monotonically when *price_per_mile* at $3.20/mile and increases at last when the price is $3.10./mile

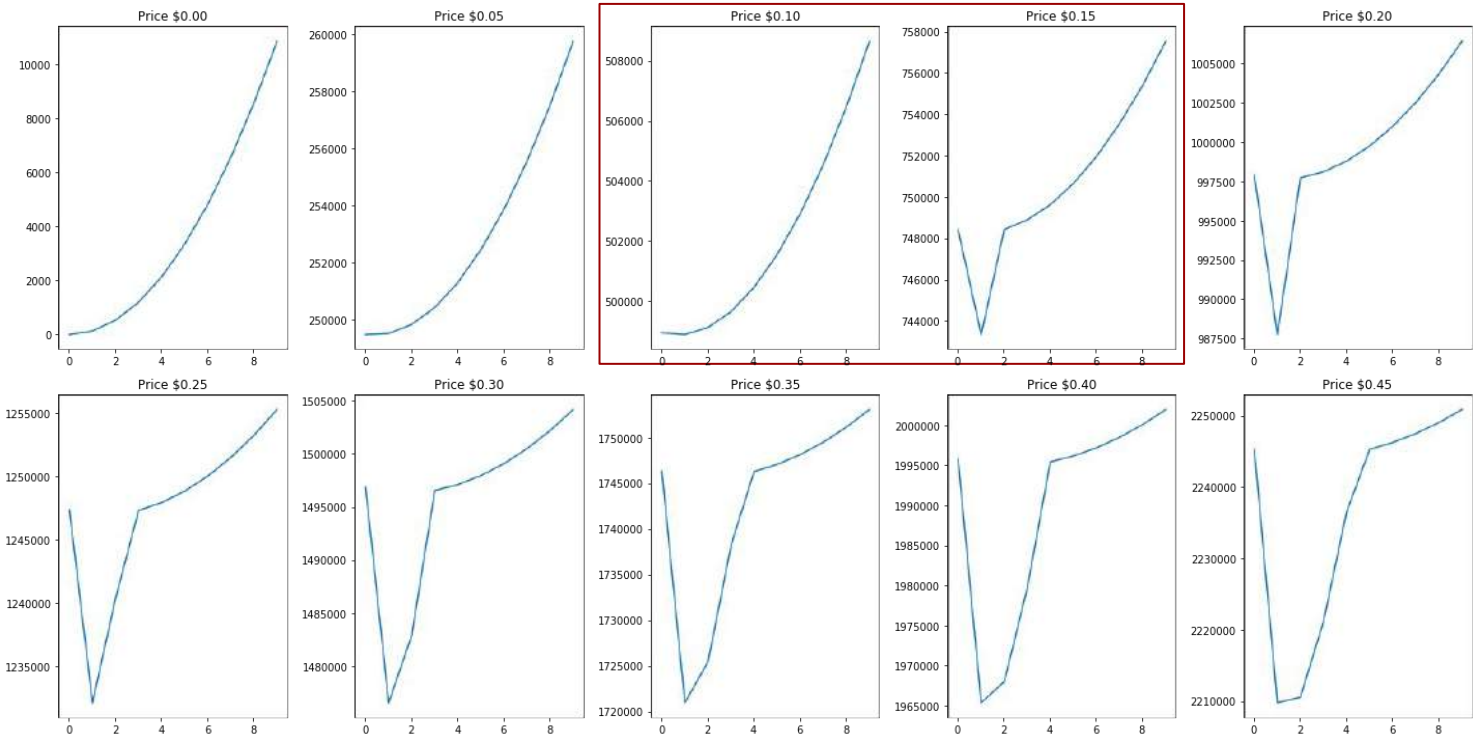
➔ We can observe : when *price_per_mile* is more than $3.10/mile, it is better to launch the relocation service at no price.

Network Optimization (2)

- **Estimation of outsourcing cost to determine relocation service feasibility. <Criteria on no need of relocation service.>**

➔ Used Pyomo for network optimization and solve the problem with gurobi solver.

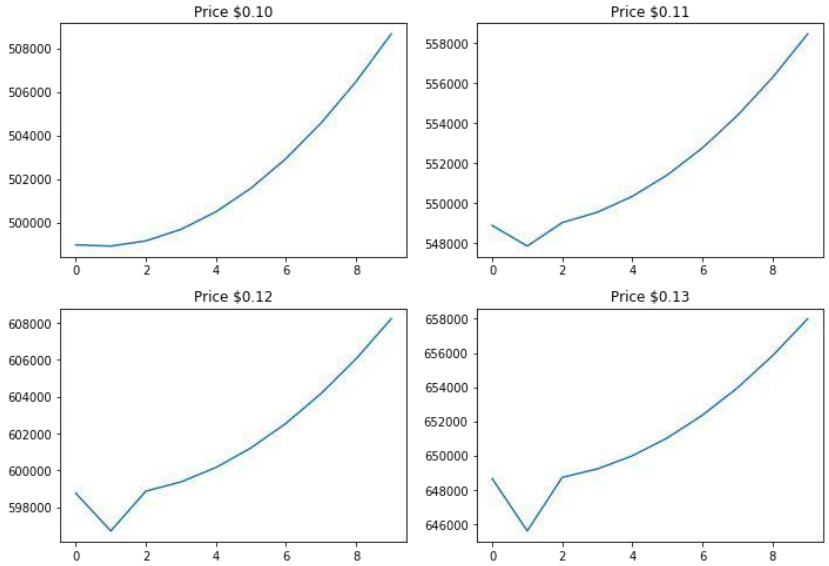
➔ Changed *price_per_mile* from $0.00/mile to $0.45/mile



Network Optimization (2) - Detailed

- **Estimation of outsourcing cost to determine relocation service feasibility. <Criteria on no need of relocation service.>**

➔ Changed *price_per_mile* from $0.10/mile to $0.15/mile



➔ We can observe : when *price_per_mile* is less than $0.11/mile, it is better not to launch the relocation service.

Conclusion

- The optimal discount rate for relocation service is **25%.**
- The optimal routes of relocation services and the number of discounted trips are stored in ‘optimal_sol.csv’.
- The relocation service is profitable when the outsourcing cost is in **between $0.11 and $3.10 per mile.**
  - If the outsourcing cost is less than $0.11/mile, it is better not to launch relocation service.
  - If the outsourcing cost is more than $3.10/mile, it is better to provide relocation service at free.

Detailed solution and codes :
https://github.com/yyul10/LA_metro_bike_relocation

Thank You