Project Portfolio

University of Southern California
Spring 2022
ISE-533 31533

Group 7
Yiwen Chen (1678961165)
Jiachen Tu (2260088714)
Jiaqi Ma (3028009001)

Table of Contents

Introduction

This portfolio contains reports for 3 projects. We addressed three real-world problems using multiple integrative analytics techniques.

The first project is The Transshipment Problem. In this project, we solved a transshipment problem using the All-In-One Stochastic Linear Programming and the Stochastic Quasi-Gradient method.

The second project is The Meal Planning Problem. This project generates a suggested weekly meal plan using Matrix Factorization for prediction and Mixed Integer Programming for optimization.

The third project is Integrative Analytics with Cross-Sectional Data. This project combines multiple linear regression and stochastic optimization together to solve a two-stage predictive stochastic programming problem.

The Transshipment Problem

1. Introduction

In this project, we solved the transshipment problem using the All-In-One Stochastic Linear Program (SLP) and the Stochastic Quasi-Gradient (SQG) methods. The transshipment problem is to minimize the long-run average costs between retailers and a supplier. The specific problem description and the steps of algorithms are illustrated in the following report.

2. The Transshipment Problem

2.1 Problem Description & Assumptions

The transshipment problem includes one supplier and N retailers at distinct locations satisfying orders from customers. It allows transshipment between any pair of retailers to meet their demands. The goal of the problem is to find an optimal order-up-to stock level quantity minimizing the long-run average costs including replenishment, transshipment, holding (for inventory) and penalty (for backorders) costs. We divide the long-run business into infinite periods which can be treated as identical. The whole process within one period is described as follows.

S: Order-up-to inventory (the same in every period)
B: Beginning inventory after replenishments and backorders met)
M: Demand at each retailer
R: Replenishment before next period
E: Ending inventory including units on order from supplier

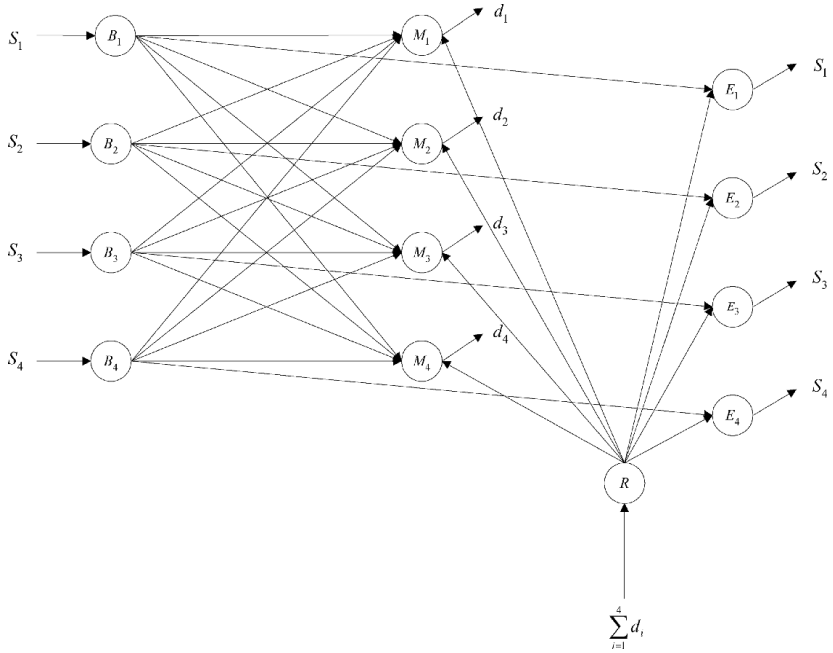

Figure 1: Network flow representation (Herer, 2006)

At the beginning of a period, every retailer has its beginning inventory according to order-up-to stock level. Then, each of the retailers has distinct demands from customers. Every retailer has three sources to satisfy the demands: (i) its own beginning; (ii) transshipment from any other retailers; (iii) replenishment from supplier to satisfy backorders. After all the demands are met in this period, all the retailers will replenish their inventory to order-up-to stock level again and start next period.

In our model, we assume that retailers are non-identical, and each retailer faces uncertain customer demand which distribution is stationary. Also, we assume transshipment policies are stationary and no-buildup, replenishment policies are non-shortage inducing. To be specific, stationary means the order placed by each retailer only depends on pre-transshipment inventory and the observed demand. No-buildup means transshipments are only made to satisfy actual observed demand. Non-shortage inducing means the beginning inventory (after replenishments and backorders met) cannot be negative.

2.2 Notations

The following notations are used to set up the transshipment problem.

N = number of retailers
S_i = order-up-to quantities at retailer i
D_i = Demand realization at retailer i
e_i = Ending inventory held at retailer i
f_i = Stock used to satisfy demand at retailer i
q_i = Inventory increased through replenishment at retailer i
r_i = Amount of shortage met after replenishment at retailer i
t_{ij} = Stock at retailer i used to meet demand at retailer j by transshipment
h_i = Unit cost of holding inventory at retailer i
c_{ij} = Unit cost of transshipment from retailer i to retailer j
p_i = Penalty cost for shortage at retailer i

2.3 Primal Problem

The goal of this transshipment problem is to find an order-up-to S policy that minimizes the cost of the system while satisfying the relationships in the network flow:

$$h(S,D) = \min \sum_i h_i e_i + \sum_{i \neq j} c_{ij} t_{ij} + \sum_i p_i r_i \qquad (1)$$

$$f_i + \sum_{j \neq i} t_{ij} + e_i = s_i, \quad \forall i \qquad (1a)$$

$$f_i + \sum_{j \neq i} t_{ji} + r_i = d_i, \quad \forall i \tag{1b}$$

$$\sum_i r_i + \sum_i q_i = \sum_i d_i \tag{1c}$$

$$e_i + q_i = s_i, \quad \forall i \tag{1d}$$

$$e_i, f_i, q_i, r_i, s_i, t_{ij} \geq 0, \quad \forall i, j.$$

2.4 Data

The following are the data to be used in solving the program.

Number of Retailers, $N = 7$

The transshipment system is assumed to contain 7 different retailers.

Holding Cost, $h = 1$

The unit cost of holding inventory is assumed to be the same across all retailers at \$1.

Transshipment Cost, $c = 0.1$

The unit cost of transshipment across all retailers is uniform at \$0.1.

Backorder Cost, $p = 4$

The unit cost of backorder when shortage exists at retailers is uniform at \$4.

Demand, D_i

The demand of each retailer is assumed to be normally distributed random variables with mean and standard deviation shown in Table 1.

Retailer	1	2	3	4	5	6	7
Mean	100	200	150	170	180	170	170
SD	20	50	30	50	40	30	50

Table 1: Parameters of random variable D_i

Using quantiles of distributions to represent the demand scenarios is a simple way. The 25th, 50th, and 75th percentile are then generated to simulate the low, medium, and high level of demand. This would give a total of $M = 3^7 = 2187$ different combinations of demand scenarios.

Retailer	1	2	3	4	5	6	7
Low	86.51	166.28	129.77	136.28	153.02	149.77	136.28
Medium	100.00	200.00	150.00	170.00	180.00	170.00	170.00
High	113.49	233.72	170.23	203.72	206.98	190.23	203.74

Table 2: Levels of demand

Another way of generating scenarios is to sample from normal distributions with parameters in Table 1. We use truncated normal distributions in Table 3 to generate sample points which are within limit of $\mu \pm 3\sigma$.

Retailer	1	2	3	4	5	6	7
Lower Bound	40	50	60	20	60	80	20
Mean, SD	100, 20	200, 50	150, 30	170,50	180,40	170,30	170,50
Upper Bound	160	350	240	320	300	260	320

Table 3: Truncated normal distributions

3. Models

3.1 All-in-One Stochastic Linear Program (SLP)

With 2187 different demand scenarios simulated, we can determine the order-up-to policy S for each scenario using the basic program (1). Assuming that all the demand scenarios provide a well simulation of the real-world long run demand situations, we can modify the model to determine one order-up-to policy S that takes all 2187 scenarios into consideration and minimizes the average long-run cost of the transshipment system.

The order-up-to quantities that minimize the average long-run costs can be obtained by solving the following objective

$$\min E\left[h\left(S,\widetilde{D}\right)\right] = \min \frac{1}{M}\sum_{k}\left(\sum_{i} he_{ki} + \sum_{i\neq j} ct_{kij} + \sum_{i} pr_{ki}\right) \qquad (2)$$

subject to

$$f_{ki} + \sum_{j\neq i} t_{kij} + e_{ki} = s_{ki}, \quad \forall i,k \qquad (2a)$$

$$f_{ki} + \sum_{j\neq i} t_{kji} + r_{ki} = d_{ki}, \quad \forall i,k \qquad (2b)$$

$$\sum_i r_{ki} + \sum_i q_{ki} = \sum_i d_{ki}, \quad \forall k \qquad (2c)$$

$$e_{ki} + q_{ki} = s_{ki}, \quad \forall i, k \qquad (2d)$$

$$e_{ki}, f_{ki}, q_{ki}, r_{ki}, s_{ki}, t_{kij} \geq 0, \quad \forall i, j, k.$$

where k denotes the different demand scenarios.

3.2 Stochastic Quasi-Gradient (SQG)

The SQG method estimates the sub-gradient of $E[h(S, \widetilde{D})]$ by solving the dual formulation.

The dual program of (1) is given by

$$\max \sum_i s_i B_i + \sum_i d_i M_i + \sum_i d_i R + \sum_i s_i E_i \qquad (3)$$

subject to

$$B_i + E_i \leq h_i, \quad \forall i \qquad (3a)$$

$$B_i + M_i \leq 0, \quad \forall i \qquad (3b)$$

$$B_i + M_j \leq c_{ij}, \quad \forall i, j, i \neq j \qquad (3c)$$

$$M_i + R \leq p_i, \quad \forall i \qquad (3d)$$

$$R + E_i \leq 0, \quad \forall i \qquad (3e)$$

where B_i , M_i , R_i and E_i denotes the dual multipliers of (1a) - (1d).

Steps of SQG method:

1. Initial S
2. M i.i.d. scenarios are selected, denoted as $D^1, \ldots, D^M$
3. Solve the dual problem and get dual multipliers $B^{k,m}$ and $E^{k,m}$
4. Calculate the unbiased estimate of sub-gradient: $\hat{\xi}^k = \frac{1}{M}\sum_{m=1}^{M}(B^{k,m} + E^{k,m})$
5. Update S: $S^{k+1} = P_+(S^k - \alpha_k \hat{\xi}^k)$
6. Repeat Steps 2 - 5 until stop condition (predetermined maximum number of iterations N) is met

Note that $P_+(\cdot)$ denotes the positive part of the argument, and α_k are nonnegative scalars that satisfy $\alpha_k \to 0$, $\sum_k \alpha_k \to \infty$, and $\sum_k \alpha_k^2 < \infty$. Learning rate $\alpha_k = \frac{N/a}{i}$ where a is a preset coefficient and i is the iteration number. Therefore, the learning rate is decreasing along with the iterations. As default, the initial order-up-to level S is set as the mean of the normal distribution, $M = 100$, $N = 100$, and $a = 10$. Figure 2 shows the values of learning rate α_k along with iteration i in our model.

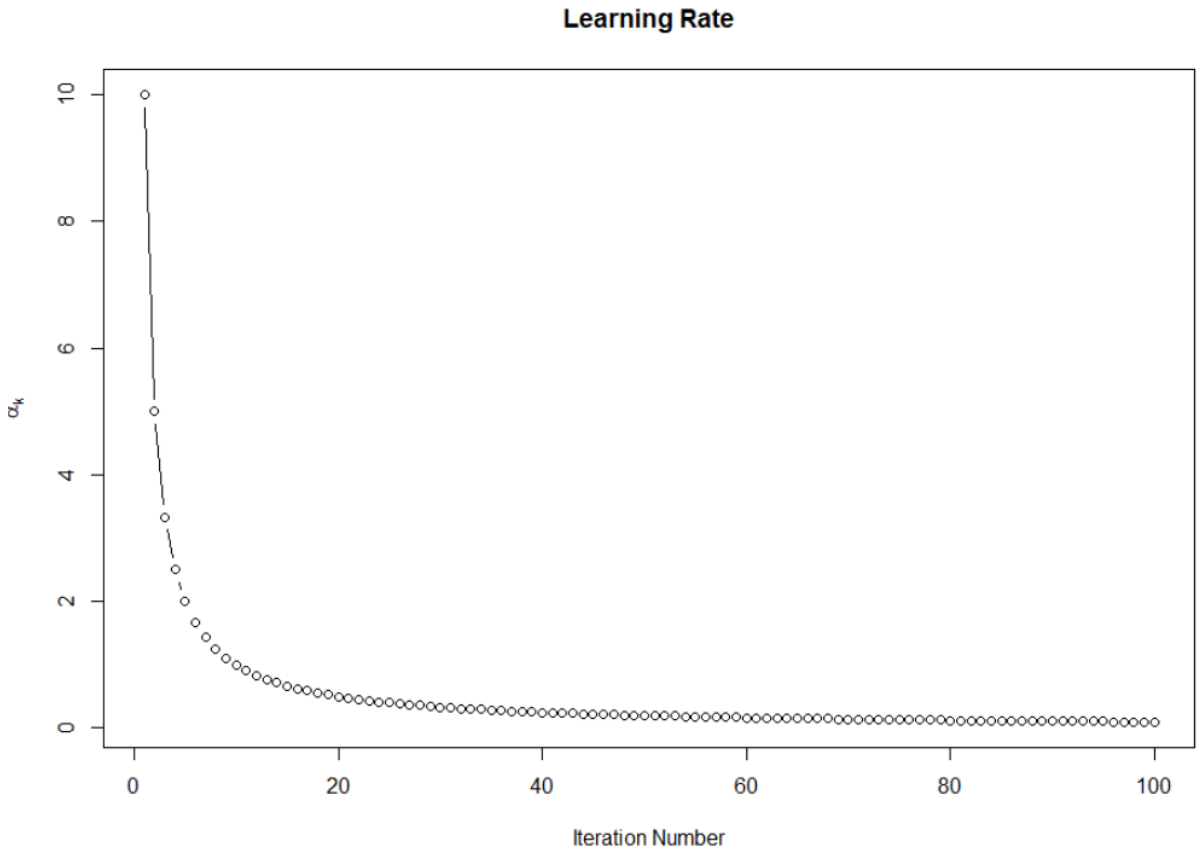

Figure 2: Learning Rate vs. Iteration Number

4. Model Outputs & Analysis

4.1 Result using quantile samples

	SLP model	SQG model	95% CI (SQG model)
S (Retailer 1)	104.50	107.22	[107.18, 107.27]
S (Retailer 2)	208.99	207.41	[207.36, 207.46]
S (Retailer 3)	156.74	157.45	[157.40, 157.50]
S (Retailer 4)	178.99	177.37	[177.32, 177.42]
S (Retailer 5)	188.99	187.35	[187.30, 187.40]
S (Retailer 6)	176.74	177.44	[177.40, 177.49]
S (Retailer 7)	178.99	177.52	[177.47, 177.57]
Total Cost	$85.99	$86.01	NA

Table 3: Outputs of SLP and SQG model

4.2 Result using truncated normal distribution samples

	SLP model	SQG model	95% CI (SQG model)
S (Retailer 1)	105.85	111.76	[111.72, 111.80]
S (Retailer 2)	215.59	212.35	[212.31, 212.39]
S (Retailer 3)	160.19	162.10	[162.06, 162.14]
S (Retailer 4)	187.30	182.41	[182.37, 182.45]
S (Retailer 5)	191.79	192.23	[192.19, 192.26]
S (Retailer 6)	179.33	182.05	[182.01, 182.09]
S (Retailer 7)	184.77	182.32	[182.28, 182.35]
Total Cost	$149.96	$150.02	NA

Table 4: Output of SLP and SQG models with truncated normal distribution samples

4.3 Convergence of order-up-to level

The convergence of S is shown in Figure 3. All the variables converge very quickly.

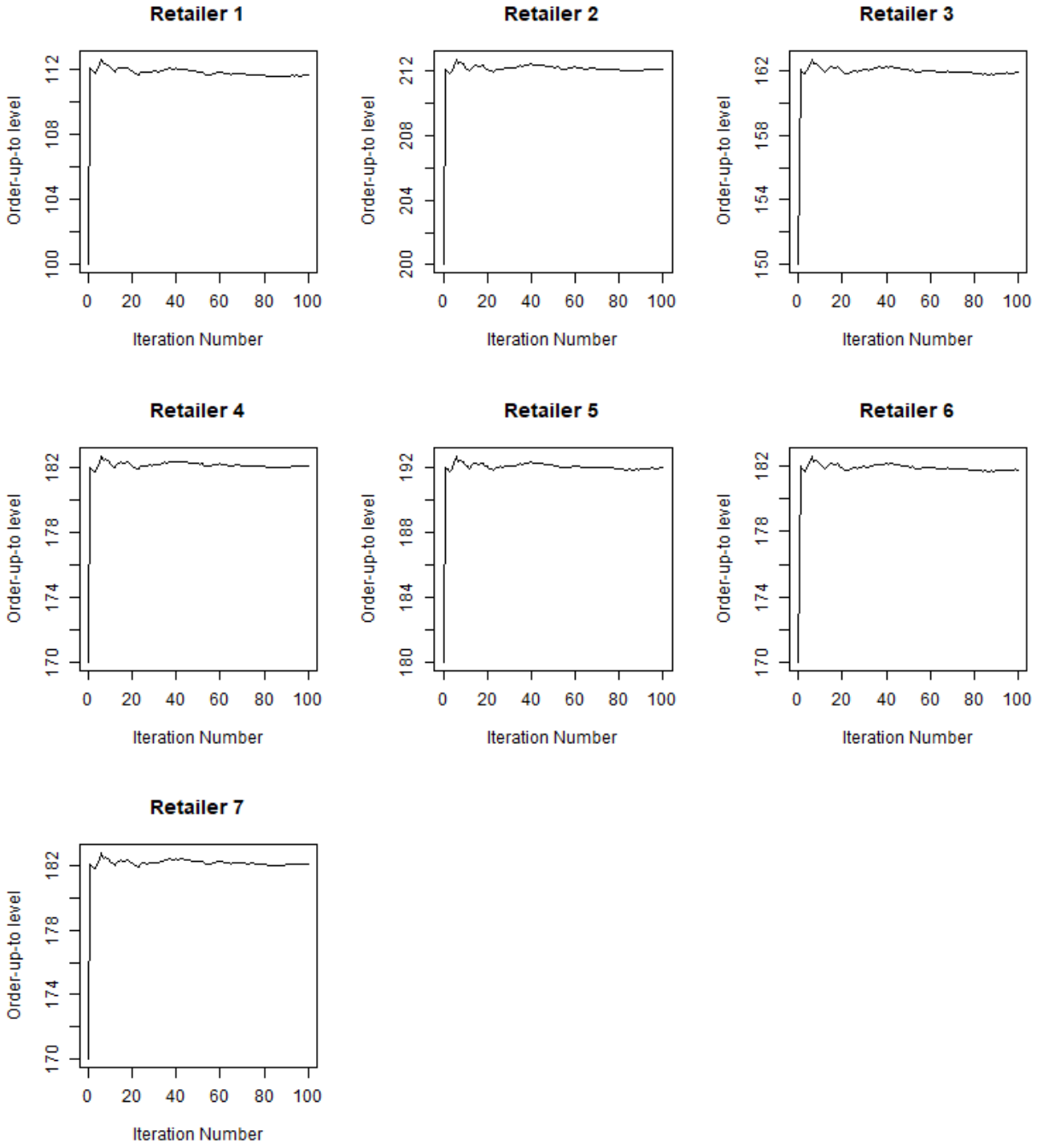

Figure 3: Convergence of order-up-to level in SQG model

5. Sensitivity Analysis on Convergence in SQG

When using the SQG method, slight changes in parameters can sometimes lead to very different results. This is common in gradient descent methods. To have a brief understanding of effects of different parameters on convergence, we conduct some trials by tuning them. All the trials are run using SQG method for 100 replications with the same random seed making the results comparable. Here, we try different values of initial stock level and learning rate. All these changes are compared with SQG method results with default values and the same random seed.

5.1 Initial value of stock level

The original initial values of stock level are set as the mean of the normal distributions. But in real world situations, we may not be able to estimate the distributions of demands exactly. We could start with zeros or other arbitrary numbers which may be larger. Here we try zero and three hundred as initial values of stock levels while keeping all other parameters not changed. The results are shown in Table 5 and 6.

	SQG model (Initial stock level = mean)	SQG model (Initial stock level = 0)
Retailer 1	111.76 95%CI: [111.72, 111.80]	168.64 95%CI: [168.60, 168.67]
Retailer 2	212.35 95%CI: [212.31, 212.39]	170.83 95%CI: [170.79, 170.86]
Retailer 3	162.10 95%CI: [162.06, 162.14]	170.04 95%CI: [170.00, 170.07]
Retailer 4	182.41 95%CI: [182.37, 182.45]	170.37 95%CI: [170.33, 170.40]
Retailer 5	192.23 95%CI: [192.19, 192.26]	170.63 95%CI: [170.60, 170.67]
Retailer 6	182.05 95%CI: [182.01, 182.09]	170.56 95%CI: [170.52, 170.59]
Retailer 7	182.32 95%CI: [182.28, 182.35]	170.36 95%CI: [170.32, 170.39]
Total Cost	$150.02	$161.71

Table 5: Output of SQG model with initial stock level equal to zero comparing with former result

	SQG model (Initial stock level = mean)	SQG model (Initial stock level = zero)
Retailer 1	111.76 95%CI: [111.72, 111.80]	248.13 95%CI: [248.13, 248.13]
Retailer 2	212.35 95%CI: [212.31, 212.39]	248.47 95%CI: [248.47, 248.48]
Retailer 3	162.10 95%CI: [162.06, 162.14]	248.13 95%CI: [248.13, 248.13]
Retailer 4	182.41 95%CI: [182.37, 182.45]	248.24 95%CI: [248.24, 248.24]
Retailer 5	192.23 95%CI: [192.19, 192.26]	248.19 95%CI: [248.19, 248.19]
Retailer 6	182.05 95%CI: [182.01, 182.09]	248.13 95%CI: [248.13, 248.13]
Retailer 7	182.32 95%CI: [182.28, 182.35]	248.22 95%CI: [248.21, 248.22]
Total Cost	$150.02	$600.95

Table 6: Output of SQG model with initial stock level equal to three hundred comparing with former result

The result of SQG method with initial stock level equal to zero is close to optimal since its error rate of objective value is less than 10%. However, the result of SQG method with initial stock level as three hundred does not perform well. It seems not converged. We will discuss this situation later.

5.2 Learning rate

Learning rate is also a very important parameter in SQG method. It usually cooperates with the initial value set in the model. In the process of gradient descent (quasi-gradient descent in SQG), the step sizes in each iteration are usually set as the product of learning rate and gradient. In our SQG model, learning rate is

$$\alpha_k = \frac{N/a}{i} = \frac{N}{a \cdot i}$$

Recall that N is the maximum number of iterations, a is a preset coefficient and i is the iteration number. When N is fixed, we could control the learning rate by tuning coefficient a. As default, we set $a = 10$ in our SQG model. It determines that the step size is equal to $\frac{1}{a}$ in the last iteration. A smaller coefficient a makes the step size bigger in all iterations. But with the change of initial

stock level, the learning rate should also be tuned and may lead to a better result. Here, following the two trials above in Section 5.1, we further tune the models by setting $a = 2, 5$.

The result of models with initial stock level equal to zero shown in Table 7 does not change much while tuning the learning rate. It performs not bad with default value of coefficient a. When we reduce the coefficient a to 5, the objective value gets closer to the optimal. But when we continue to reduce the coefficient a to 2, making the learning rate bigger, the objective value only gets a little bit smaller. It may tell us that in this case, the learning rate is appropriate. Keeping on increasing the learning rate is not a good choice now since a learning rate that is too large can cause fluctuations in the gradient descent process, making the target higher.

To prove our conjecture, the plot showing the convergence of variable S in models with different values of coefficient a is shown in Figure 4. In the models with initial stock level equal to zero, the models with a equal to 2 and 5 converge quickly. While the model with $a = 10$ may need more steps to get close enough to optimal.

	Stock level = 0 a = 10	Stock level = 0 a = 5	Stock level = 0 a = 2
Retailer 1	168.64 [168.60, 168.67]	171.62 [171.57, 171.66]	167.21 [167.14, 167.27]
Retailer 2	170.83 [170.79, 170.86]	177.09 [177.04, 177.13]	180.86 [180.80, 180.93]
Retailer 3	170.04 [170.00, 170.07]	173.87 [173.83, 173.92]	171.33 [171.26, 171.40]
Retailer 4	170.37 [170.33, 170.40]	175.50 [175.46, 175.54]	176.36 [176.29, 176.43]
Retailer 5	170.63 [170.60, 170.67]	176.15 [176.11, 176.20]	177.81 [177.74, 177.88]
Retailer 6	170.56 [170.52, 170.59]	175.46 [175.42, 175.50]	175.39 [175.32, 175.46]
Retailer 7	170.36 [170.32, 170.39]	175.47 [175.43, 175.51]	176.21 [176.15, 176.28]
Total Cost	$161.71	$152.86	$152.50

Table 7: Output of SQG model with initial stock level equal to zero with different coefficient a

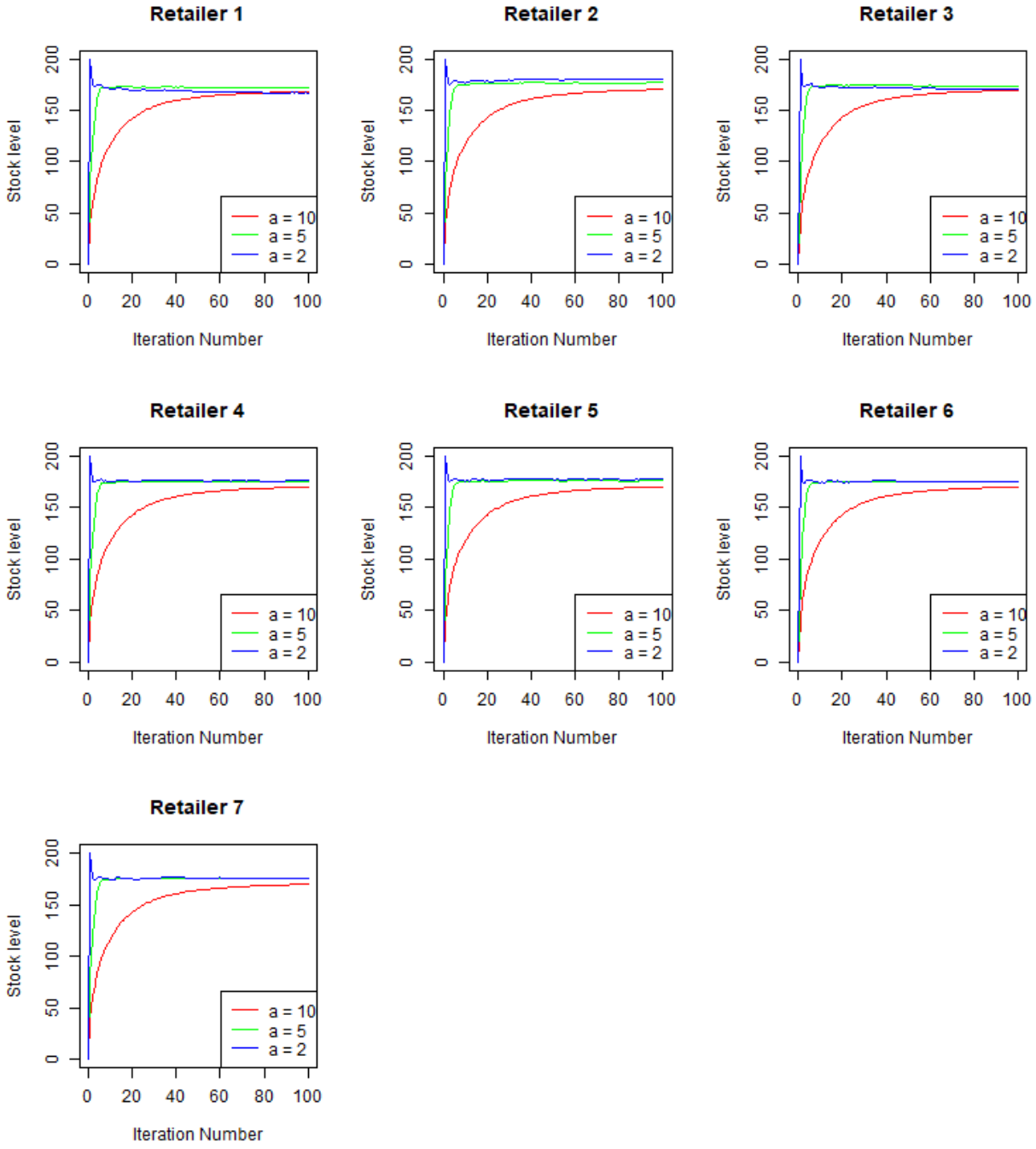

Figure 4. Convergence Process of Stock level S in SQG model (first replication) with initial stock level equal to zero

	Stock level = 300 a = 10	Stock level = 300 a = 5	Stock level = 300 a = 2
Retailer 1	248.13 [248.13, 248.13]	196.48 [196.47, 196.48]	168.66 [168.60, 168.72]
Retailer 2	248.47 [248.47, 248.48]	198.43 [198.42, 198.44]	180.32 [180.24, 180.39]
Retailer 3	248.13 [248.13, 248.13]	196.55 [196.54, 196.56]	171.77 [171.70, 171.83]
Retailer 4	248.24 [248.24, 248.24]	197.38 [197.37, 197.39]	176.15 [176.08, 176.21]
Retailer 5	248.19 [248.19, 248.19]	197.32 [197.31, 197.33]	177.22 [177.16, 177.29]
Retailer 6	248.13 [248.13, 248.13]	196.76 [196.75, 196.77]	175.05 [174.98, 175.11]
Retailer 7	248.22 [248.21, 248.22]	197.35 [197.34, 197.36]	176.03 [175.96, 176.09]
Total Cost	$600.95	$249.52	$152.59

Table 8. Output of SQG model with initial stock level equal to three hundred with different coefficient a

However, from the result of SQG models with initial stock level equal to three hundred shown in Table 8, the change of learning rate performs well. At the beginning, the model with coefficient $a = 10$ does not converge. When we reduce the coefficient a to 5, though it still does not converge, its objective value gets much smaller than before. When we continue to reduce a to 2, it seems to converge, and its objective value becomes very close to optimal. From Figure 5 showing the convergence of variable S, the model with a equal to 2 performs the best. The other models with $a = 5$ and 10 do not converge.

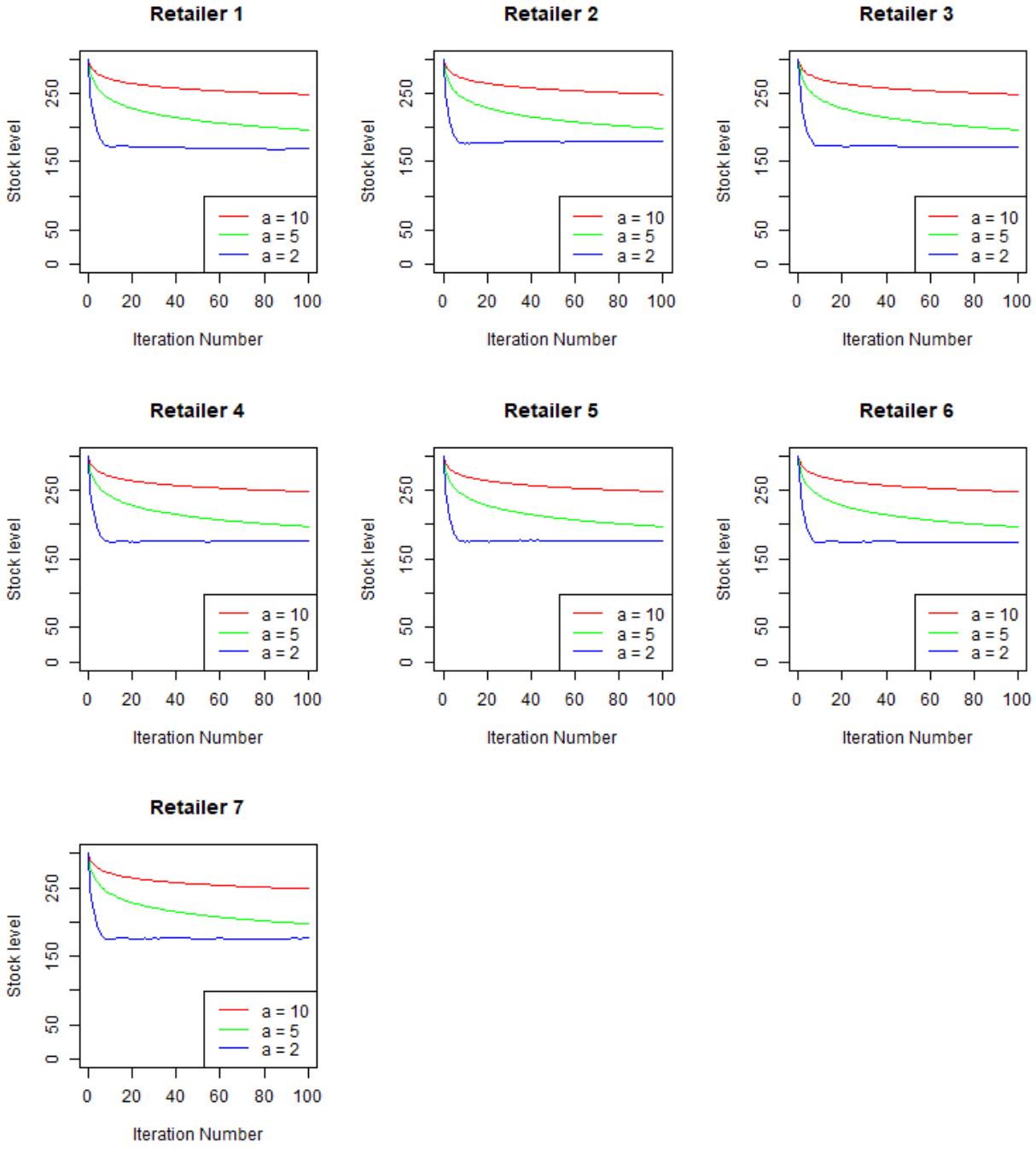

Figure 5. Convergence Process of Stock level S in SQG model (first replication) with initial stock level equal to three hundred

6. Conclusion

By using the all-in-one stochastic linear program method, the exact solution could be solved under pre-assumed variable distribution. Therefore, a variable distribution which is close enough to reality is required. But it also brings challenges to the solver for solving such a huge LP program. While using the stochastic quasi-gradient method, an approximation solution could be solved. It only needs to solve many much smaller LP programs consecutively. Similar to the all-in-one method, it also requires a variable distribution which is close to reality to ensure the solution is closer to the true value.

According to the result in Section 5, it should pay more attention when using the SQG method. Different starting points and learning rates may lead to very different results. It is recommended to try many different combinations of these parameters before giving conclusions. In the transshipment problem discussed here, a smaller starting point converges faster than a larger one. Besides, increasing the number of iterations is also a good direction after finding out a set of suitable combinations of starting point and learning rate.

The Meal Planning Problem

1. Introduction

In this project, we will focus on a meal planning problem that generates a suggested weekly meal plan. To do so, we will first explore and predict the ratings of different dishes by each team member and other users, using Matrix Factorization. Then we will use the predicted ratings from part one to generate a weekly meal plan that maximizes team's satisfaction while meeting budget, scheduling, nutritional constraints by formulating and solving a Mixed Integer Problem (MIP).

2. Problem Description

This problem can be divided into two steps as illustrated in Introduction.

First step is to solve a matrix completion problem to predict each team member's ratings to a lot of dishes which he/she never tasted before. We have a raw dataset containing 2884 users with their ratings to some of 289 dishes. In this step, we will build a rating matrix M with M_{ij} being the rating of user i to dish j using data from the raw dataset and enlarge the matrix to include each team member's ratings to several dishes. Note that the matrix M is very sparse because each user cannot rate every dish and each dish cannot be rated by every user. Then we will complete this matrix M with predictions of each user's rating to each dish via the matrix completion technique.

Second step is to make a weekly meal plan by solving a Mixed Integer Programming problem to maximize the sum of all team member's ratings of some selected dishes and ensuring some additional requirements at the same time. The additional requirements can be proposed by each team member like their nutrition requirements, the budget constraint, food allergies, time scheduling constraint, dish category diversity requirement, vegetarianism constraint and so on.

3. Rating Prediction (Matrix Completion)

3.1 The Raw Dataset

The original dataset provided contains a total of 289 dishes and 2884 users. After filtering out dishes that have never been rated, dishes that have missing nutritional data, the dataset contains 240 dishes and 2541 users.

	user	title	rating
0	rhaeredekop from Winnipeg, MB	Curried Lentil, Tomato, and Coconut Soup	5
1	lizgoldsmith1960 from Lincoln, MA	Curried Lentil, Tomato, and Coconut Soup	5
2	bmaybeegeorge from Canada	Curried Lentil, Tomato, and Coconut Soup	5
3	adventurousz from Boulder, CO	Curried Lentil, Tomato, and Coconut Soup	3
4	epaul77 from Mississippi	Curried Lentil, Tomato, and Coconut Soup	5

Table 1. Sample of raw data

Each of our team members then rated some of the dishes that they have tasted before. Some ratings are shown below.

	user	title	rating
2792	Jiachen	Quick Chicken Tikka Masala	5
2793	Jiachen	Double-Pork Carnitas	1
2794	Jiachen	Chicken and Dumplings with Mushrooms	2
2795	Jiachen	Sheet-Pan Curry Pork Chops and Sweet Potatoes	3
2796	Jiachen	Grilled Cheese Tacos	5
2817	Yiwen	"Antipasto" Pasta with Sausage, Artichoke Hear...	3
2818	Yiwen	"Nextover" Chicken Tacos with Quick Refried Be...	4
2819	Yiwen	Tuna and Artichoke Cooler-Pressed Sandwiches	4
2820	Yiwen	Grain Bowls with Chicken, Spiced Chickpeas, an...	1
2821	Yiwen	Broiled Cod with Fennel and Orange	3
2848	Jiaqi	Slow-Cooker Shredded Chicken	4
2849	Jiaqi	Slow-Cooker Chipotle-Orange Pork Tacos	1
2850	Jiaqi	Hot Honey Pork Chops with Escarole and White B...	1
2851	Jiaqi	Kabocha Squash Cake with Brown Sugar Cream	5
2852	Jiaqi	Roasted Acorn and Delicata Squash Salad	3

Table 2. Sample of ratings from the team

With additional 3 team members' rating included, the dataset contains 3101 ratings in total, with 2544 unique users and 240 unique dishes. The median number of dishes rated per user is 1. The data is then converted into a 2544×240 matrix M, with M_{ij} represents user i's rating of dish j ($M_{ij} = 0$ if there exists no rating from user i to dish j).

	Curried Lentil, Tomato, and Coconut Soup	Roasted Butternut Squash with Herb Oil and Goat Cheese	Pumpkin Muffins	Chopped Salad with Shallot Vinaigrette, Feta, and Dill	Grain Salad with Olives and Whole-Lemon Vinaigrette	Chilled Coconut Corn Soup	Vietnamese-Style Spaghetti Squash "Noodle" Bowls with Skirt Steak
rhaeredekop from Winnipeg, MB	5.0	0.0	0.0	0.0	0.0	0.0	0.0
lizgoldsmith1960 from Lincoln, MA	5.0	0.0	0.0	0.0	0.0	0.0	0.0
bmaybeegeorge from Canada	5.0	0.0	0.0	0.0	0.0	0.0	0.0
adventurousz from Boulder, CO	3.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 3. Sample of incomplete rating matrix M

3.2 Rating Prediction Algorithm

Rating prediction can be formalized into a matrix completion problem where we use the knowledge of known ratings to replace the missing ratings. One approach is matrix factorization, where we factorize the matrix M into user profile U and dish profile V. Matrix U has the shape of $2544 \times d$ and matrix V has the shape $d \times 240$, with column vectors u_i and v_j representing user-specific and dish-specific latent feature vectors respectively. The problem basically turns into estimating U and V.

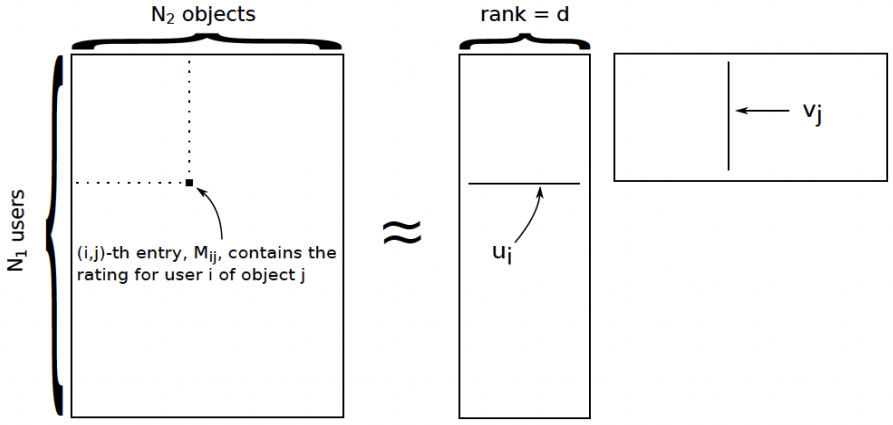

Figure 1. Matrix Factorization

Probabilistic Matrix Factorization can be used to solve this problem, which was introduced by Salakhutdinov and Mnih in 2006. It is assumed that the entries of M are normally distributed around ${u_i}^T v_j$ with a common variance σ^2. The conditional distribution of observed rating is:

$$p(M_o|U,V) = \int p(M_o, M_m|U,V)dM_m$$

where M_0 represents the part of matrix M that has observed rating and M_m represents the missing part of M.

The joint likelihood:

$$p(M_0,U,V) = [\prod\prod p(M_{ij}|u_i,v_j)I_{ij}] \times [\prod_{i=1}^{2544} p(u_i)][\prod_{j=1}^{240} p(v_j)]$$

where $I_{ij} = 1$ if M_{ij} has a observed rating, and $I_{ij} = 0$ if M_{ij} has a missing rating.

The Maximum A Posteriori (MAP) solution for U and V is the maximum of the log joint likelihood

$$U_{MAP}, V_{MAP} = \arg\max_{U,V} \sum I_{ij} \ln p(M_{ij}|u_i,v_j) + \sum \ln p(u_i) + \sum \ln p(v_j)$$

which is equivalent to maximize

$$\mathcal{L} = -\sum \frac{1}{2\sigma^2} I_{ij} \|M_{ij} - u_i^T v_j\|^2 - \sum \frac{\lambda}{2} \|u_i\|^2 + \sum \frac{\lambda}{2} \|v_j\|^2 + C$$

To obtain estimates for U and V, we can consider a coordinate ascent algorithm. First we initialize values of v_j , for example generate random values from a normal distribution with mean 0 and variance $\lambda^{-1}I$. Then we can start the iterations in which we update the values for u_i and v_j .

To update u_i :

1. take the derivative of L with respect to u_i and set to 0

$$\nabla_{u_i}\mathcal{L} = \sum_j \frac{1}{\sigma^2} I_{ij}(M_{ij} - u_i^T v_j)v_j - \lambda u_i = 0$$

2. Solve and update u_i

$$u_i = (\lambda\sigma^2 \mathrm{I} + \sum_j I_{ij} v_j v_j^T)^{-1} (\sum_j M_{ij} v_j I_{ij})$$

To update v_j:

3. take the derivative of L with respect to v_j and set to 0

$$\nabla_{v_j}\mathcal{L} = \sum_i \frac{1}{\sigma^2} I_{ij}(M_{ij} - v_j^T u_i)u_i - \lambda v_j = 0$$

4. Solve and update u_i

$$u_i = (\lambda\sigma^2 \mathrm{I} + \sum_i I_{ij} u_i u_i^T)^{-1} (\sum_i M_{ij} u_i I_{ij})$$

The predicted ratings (completed matrix) can be obtained by calculating

$$u_i^T v_j$$

and rounding to the rating scale of 1-5 accordingly.

3.3 Rating Prediction Result

By assuming $d = 10, \sigma = 1, l = 1$ and iterating 500 times, the following predicted ratings are obtained.

	Curried Lentil, Tomato, and Coconut Soup	Roasted Butternut Squash with Herb Oil and Goat Cheese	Pumpkin Muffins	Chopped Salad with Shallot Vinaigrette, Feta, and Dill	Grain Salad with Olives and Whole-Lemon Vinaigrette	Chilled Coconut Corn Soup	Vietnamese-Style Spaghetti Squash "Noodle" Bowls with Skirt Steak	Roasted Squash with Mint and Toasted Pumpkin Seeds	Butternut Squash Steaks with Brown Butter–Sage Sauce	Freeform Chicken Meatballs with Carrots and Yogurt Sauce
Jiachen	5.0	3.0	2.0	5.0	1.0	4.0	2.0	3.0	4.0	1.0
Yiwen	1.0	5.0	2.0	1.0	2.0	4.0	5.0	2.0	4.0	1.0
Jiaqi	1.0	5.0	2.0	1.0	2.0	4.0	2.0	4.0	1.0	1.0

Table 4. Sample of completed rating matrix for the team

4. Meal Planning

By completing the matrix above, we get every member's rating for all recipes. Together with nutrition information, price, and cooking time of each recipe, we can build a mixed integer program (MIP). The objective is to maximize the group members' satisfaction in terms of the total score for the weekly meal plan, while meeting other constraints of budget, nutritional needs, timing, etc.

4.1 Notations

1. Indices:

$Person: p \in \{1,2,3\}$ $\quad$ $Day: d \in \{1,2,3,4,5\}$
$Dish: r \in \{1,2,\ldots,240\}$ $\quad$ $Nutrition: i \in \{1,2,3,4\}$

2. Parameters:

$C_L = Lower\ limit\ of\ daily\ calories\ intake\ per\ person = 1600$

$C_U = Upper\ limit\ of\ daily\ calories\ intake\ per\ person = 3000$

$S_L = Lower\ limit\ of\ daily\ sodium\ intake\ per\ person = 0$

$S_U = Upper\ limit\ of\ daily\ sodium\ intake\ per\ person = 2300$

$F_L = Lower\ limit\ of\ daily\ fat\ intake\ per\ person = 44$

$F_U = Upper\ limit\ of\ daily\ fat\ intake\ per\ person = 77$

$P_L = Lower\ limit\ of\ daily\ protein\ intake\ per\ person = 40$

$P_U = Upper\ limit\ of\ daily\ protein\ intake\ per\ person = 70$

$Rating_{p,r} = rating\ from\ person\ p\ to\ dish\ r$

$Price_r = price\ of\ dish\ r$

$Calories_r = calories\ of\ dish\ r$

$Sodium_r = sodium\ of\ dish\ r$

$Fat_r = fat\ of\ dish\ r$

$Protein_r = protein\ of\ dish\ r$

$Budget = daily\ budget = 100$

$\beta_i = coefficient\ for\ overflow\ penalty\ of\ nutrition\ i = 0.1$

$Time_r = cooking\ time\ of\ dish\ r$

3. Variable:

$$x_{r,d,p} = \begin{cases} 1\ if\ dish\ r\ is\ cooked\ by\ person\ p\ on\ day\ d \\ 0\ if\ not \end{cases}$$

4.2 Data

The nutritional data for each dish, including protein(g), fat(g), sodium(mg) and calories are obtained from the original dataset provided.

	Protein	Fat	Sodium	Calories
Curried Lentil, Tomato, and Coconut Soup	13.0	28.0	667.0	437.0
Roasted Butternut Squash with Herb Oil and Goat Cheese	4.0	9.0	576.0	175.0
Pumpkin Muffins	6.0	11.0	183.0	364.0
Chopped Salad with Shallot Vinaigrette, Feta, and Dill	6.0	13.0	413.0	170.0
Grain Salad with Olives and Whole-Lemon Vinaigrette	8.0	19.0	483.0	330.0

Table 5. Sample of nutrition data

The prices are approximated by its main ingredients and the cooking times are obtained from Epicurious (from TA's notes). For those dishes that have missing price or cooking time, a random value would be generated to replace the missing value.

	Price	Cooking_time
Curried Lentil, Tomato, and Coconut Soup	9.19	25
Roasted Butternut Squash with Herb Oil and Goat Cheese	4.13	54
Pumpkin Muffins	1.20	24
Chopped Salad with Shallot Vinaigrette, Feta, and Dill	5.33	56
Grain Salad with Olives and Whole-Lemon Vinaigrette	6.42	43

Table 6. Sample of price and cooking time data

4.3 Model

The objective for the MIP model is to maximize the total scores for all members of the team for the weekly meal plan, which can be formulated as below:

$$max \sum_{d} \sum_{r} \sum_{p} rating_{p,r} x_{r,d,p} - \sum_{d} \sum_{i} \beta_i f_{d,i}^{+}$$

where the first term calculates the total ratings of all dishes in the menu and the second term represents the overflow penalty when certain constraints are exceeded.

The objective should be bounded by the following constraints:

1. Nutritional constraints

$$\sum_{r,p} c_r x_{r,d,p} \geq C_L \quad \forall d \qquad \sum_{r,p} f_r x_{r,d,p} \geq F_L \quad \forall d$$
$$\sum_{r,p} c_r x_{r,d,p} - C_U = f_c^{+} - f_c^{-} \quad \forall d \qquad \sum_{r,p} f_r x_{r,d,p} - F_U = f_f^{+} - f_f^{-} \quad \forall d$$
$$\sum_{r,p} s_r x_{r,d,p} \geq S_L \quad \forall d \qquad \sum_{r,p} p_r x_{r,d,p} \geq P_L \quad \forall d$$
$$\sum_{r,p} s_r x_{r,d,p} - S_U = f_s^{+} - f_s^{-} \quad \forall d \qquad \sum_{r,p} p_r x_{r,d,p} - P_U = f_p^{+} - f_p^{-} \quad \forall d$$

Although people may require different levels of nutrition intake as the basal metabolic rate varies for individuals, on average it is suggested that each person should intake at least 1600 calories, 44 grams fat and 40 grams protein per day to maintain the human body's basic operations. On the other hand, it is important to keep in mind that each person should not ingest more than 3000 calories, 2300mg sodium, 77g fat and 70g protein per day on average due to health concerns. If such a level is exceeded, penalties will be applied to the objective.

2. Budget constraints:

$$\sum_{r,p} price_r x_{r,d,p} \leq B \qquad \forall d$$

As all team members are university students, it is assumed that we would have a limited budget at $25 per day for meals, giving a total of $75 per day for the entire team.

3. Menu variations:

$$\sum_{d,p} x_{r,d,p} \leq 2 \qquad \forall r$$

Definitely no one would like to have the same dishes every day, even if they rate the dish the highest score of 5. Therefore, to ensure menu variation on a weekly basis, this constraint is included so that each dish cannot be planned for more than twice a week.

4. Scheduling constraints:

$$\sum_{r} time_r x_{r,d,p} \leq 90 \qquad \forall d, p$$

The total time each team member spends on cooking per day should not exceed 90 minutes since we do not want one person to be responsible for all the cooking while others remain idle.

4.4 Result

As the model in 4.3 is too big to use CPLEX or GLPK to solve, we modified the model into a glide-path approach as below:

1. By setting D=1, we first solve for Day 1's menu.
2. Add additional constraint x=0 for all dishes included in Day 1's menu, solve the problem again to obtain Day 2's menu.
3. Repeat step 2 for Day 3-5's menu.

Such a process can greatly reduce the processing time while keeping weekly plans with enough variability. The generated 5-Day meal plan is:

COOK BY	DAY 1	DAY 2
JIACHEN	1. Pumpkin Cheesecake with Bourbon–Sour Cream Topping 2. Cinnamon White Hot Chocolate 3. Kabocha Squash Pilaf with Coconut	1. The Winterized Penicillin 2. Thai-Spiced Turkey Burgers
YIWEN	1. Bourbon Pumpkin Pie 2. Party Posole Rojo 3. Cinnamon White Hot Chocolate 4. Winter Squash Agrodolce 5. 10-Minute Chicken Flatbreads with Hummus and Yogurt	1. Butternut Squash Steaks with Brown Butter, ÄìSage Sauce 2. 3-Ingredient Creamy Pumpkin Pasta 3. White Bean Salad with Lemon and Cumin 4. Dandelion Salad with Pomegranate Seeds, Pine Nuts, and Roasted Delicata Squash
JIAQI	1. Quick Sesame Chicken with Broccoli 2. Green Goddess Tuna Salad Sandwich 3. Simplest Kale Salad 4. Cold Soba Noodles with Miso and Smoked Tofu	1. 10-Minute Sausage Skillet with Cherry Tomatoes and Broccolini 2. Navy Bean and Escarole Stew with Feta and Olives 3. Black-Eyed Peas with Chard and Green Herb Smash 4. Garlic-Chile Ground Pork 5. Quick Sweet Potato, Mushroom, and Black Bean Burrito
SCORE	**58.6**	**48.0**
BUDEGT	**$74.96**	**$74.45**

COOK BY	DAY 3	DAY 4
JIACHEN	1. Curried Lentil, Tomato, and Coconut Soup 2. Basil-Cashew-Lime Vermicelli Bowls with Pork and Green Beans 3. Broiled Cod with Fennel and Orange	1. Pumpkin Muffins 2. Sheet-Pan Steak Fajitas 3. Cheesy Chicken Enchilada Skillet
YIWEN	1. Shrimp with Herby White Beans and Tomatoes 2. Butternut Squash, Kale, and Crunchy Pepitas Taco 3. Twice-Roasted Squash with Parmesan Butter and Grains	1. Pumpkin Muffins 2. Garlic Mojo Sauce 3. Stuffed Sweet Potatoes with Beans and Guacamole
JIAQI	1. Chilled Coconut Corn Soup 2. Pantry Pasta Puttanesca 3. Bucatini with Lemony Carbonara	1. Easy Fried Rice with Chicken and Broccolini 2. Niçoise Toast 3. Use-It-Up Frittata
SCORE	**42.1**	**36.9**
BUDEGT	**$72.47**	**$71.92**

COOK BY	DAY 5
JIACHEN	1. Pumpkin Cheesecake with Bourbon–Sour Cream Topping 2. Cinnamon White Hot Chocolate 3. Kabocha Squash Pilaf with Coconut
YIWEN	1. Bourbon Pumpkin Pie 2. Party Posole Rojo 3. Cinnamon White Hot Chocolate 4. Winter Squash Agrodolce 5. 10-Minute Chicken Flatbreads with Hummus and Yogurt
JIAQI	1. Quick Sesame Chicken with Broccoli 2. Green Goddess Tuna Salad Sandwich 3. Simplest Kale Salad 4. Cold Soba Noodles with Miso and Smoked Tofu
SCORE	**35.0**
BUDEGT	**$71.65**

It is noticeable that across Day 1 to Day 5, the total score of the menu (total ratings) and the budget keeps decreasing, which is a downside of applying the glide-path approach that we have used up the best dished in the first few days and the quality of meals is not the same across the week.

Integrative Analytics with Cross-Sectional Data

1. Introduction

Is it possible to combine predictive models like linear regression and prescriptive models like stochastic optimization together to make a better decision? In this project, we are trying to bring data and decisions on a common footing and combine statistical and optimization models in one problem to make full use of cross-sectional data to make our decision. The workflow of this problem is shown below:

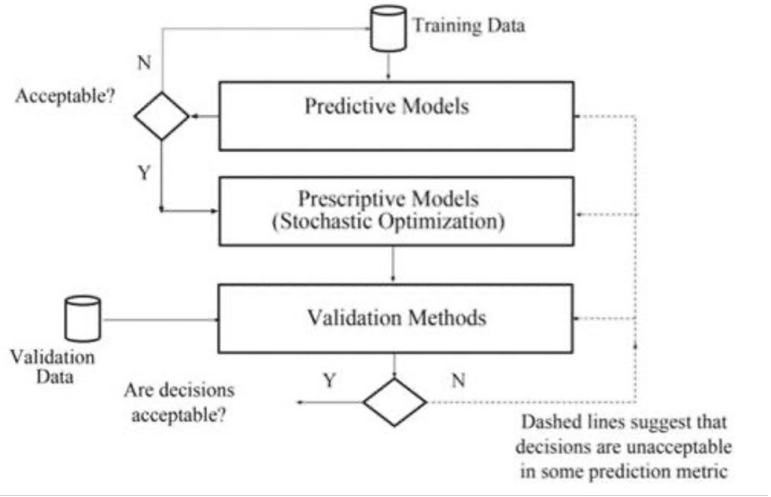

Figure 1: Learning enabled optimization
(Predictive Stochastic Programming, Yunxiao Deng, Suvrajeet Sen, 2019)

According to this workflow, we will have an integrative model that first builds a predictive model like linear regression using training data, and we will use the predictive result to draw a conclusion using stochastic optimization, finally we will do a validation using validation data to determine whether our models or our decisions are acceptable. Actually, validations will be conducted after every model to make sure the following step is using the validated result from the earlier step.

2. Problem Description

The need for such a combined model is raised by a realistic example. Let's consider the Wyndor Glass Example. Below are two data sets for Wyndor Glass company.

The company sells two products product A and product B produced by plant 1, 2 and 3 as shown in *Table 1*. The profit for product A is $3,000/batch, and the profit for product B is $5,000/batch. Each plant has its own producing constraints.

Table 2 is about the advertising results. It shows the relationships between the advertising budget allocation (for TV and Radio) and the sales result.

To bring these two data sets into one problem, let us suppose that the company can sell up to the number of products demanded by its customers — the Sales in *Table 2*, determined by the budget allocated into TV and Radio.

Plant	Prod. time A (Hours/Batch)	Prod. time B (Hours/Batch)	Total Hours
1	1	0	4 *2
2	0	2	12*2
3	3	2	18*2
Profit	$3,000	$5,000	

Table 1: Data for the Wyndor Glass Problem
(Hillier and Lieberman, 2012)

	TV	Radio	Sales
1	230.1	37.8	22.1
2	44.5	39.3	10.4
3	17.2	45.9	9.3
...	...	...	...
200	232.1	8.6	13.4

Table 2: The advertising data set
(James et al., 2013)

Suppose the management is considering an advertising budget of $200,000, the advertising cost for TV is $100, the advertising cost for Radio is $500, and the management also has a policy requiring the Radio advertising expenditures be at least some positive fraction of expenditures in TV advertising (say this fraction is $\alpha \in (0, 1)$, we choose $\alpha=0.5$ for example). Finally, we have lower and upper limits for TV advertising expenditures and Radio advertising expenditures to allow the model to only include predictions within the range of data in *Table 2*.
Now, the problem becomes how much advertising budget should be devoted to TV and how much should be devoted to Radio to maximize the expected profit. We can break the problem into two stages (the units are all thousands of dollars).

2.1 First stage

The first stage is *mainly* about the advertising budget allocation.

$$\begin{aligned} \text{Max} \quad & -0.1x_1 - 0.5x_2 + \mathbb{E}[\text{Profit}(\tilde{\omega})] \\ \text{s.t.} \quad & x_1 + x_2 \leq 200 \\ & x_1 - 0.5x_2 \geq 0 \\ & L_1 \leq x_1 \leq U_1, L_2 \leq x_2 \leq U_2 \end{aligned}$$

x_1: *advertising expenditures in TV*
x_2: *advertising expenditures in Radio*
L_1, U_1: *Lower bound and Upper bound of advertising expenditures in TV*
L_2, U_2: *Lower bound and Upper bound of advertising expenditures in Radio*

0.1 and 0.5 denote costs for TV advertising and Radio advertising separately.
The first constraint is about the whole advertising budget. The second constraint is about the policy constraint. We can clearly notice that, to solve this subproblem, we need to know the expectation of the profit, and that leads us to the second stage of the whole problem.

2.2 Second Stage

The aim of the second stage is to maximize the expectation of the profit. So, we can have such an optimization problem:

$$\begin{aligned} \text{Profit}(\omega) \quad = \quad \text{Max} \quad & 3y_A + 5y_B \\ \text{s.t.} \quad & y_A \leq 8 \\ & 2y_B \leq 24 \\ & 3y_A + 2y_B \leq 36 \\ & y_A + y_B \leq \omega \\ & y_A, y_B \geq 0 \end{aligned}$$

y_A: *number of product A*
y_B: *number of product B*
ω: *total sales*

3 and 5 denotes profits for product A and product B separately. The first constraint, the second constraint and the third constraint are corresponding to the producing constraint of plant 1, plant 2 and plant 3. ω in the second stage subproblem can only be calculated and rewritten using data from *Table 2*, that is to say we need to find the relationship between x_1, x_2 and ω first, which leads us to descriptive analytics and predictive models.

3. Descriptive Analytics

We know that we need to find the relationship between x_1, x_2 and ω first from the analysis above. According to the workflow, we will use some training data to build the predictive model first. We can use some methods like scatter plot to find that the relationship between x_1, x_2 and ω is multiple linear regression, that is to say we can rewrite ω using x_1, x_2. But considering that multiple linear regression model involves some uncertainty mainly in the error (difference between the least-squares estimate and the observed data), which should be normally distributed and independent on the choice of x_1 and x_2, it is more accurate to write the relationship as below:

$$\omega = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon$$

β_0, β_1 and β_2 are the linear regression coefficients and ε is the error.

Besides, we need to validate the error to make sure it is normally distributed.

3.1 Predictive Model: Multiple Linear Regression

Our raw data set has 200 rows and 5 columns. Notice we will only use the TV column, the Radio column and the Sales column when building the multiple linear regression model.

	Unnamed: 0	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9
...	...	...	...	...	...
195	196	38.2	3.7	13.8	7.6
196	197	94.2	4.9	8.1	9.7
197	198	177.0	9.3	6.4	12.8
198	199	283.6	42.0	66.2	25.5
199	200	232.1	8.6	8.7	13.4

200 rows × 5 columns

Table 3: The row data set

We split data into training and validation sets. A 50-50 split is recommended since we need enough data to estimate the validation object later. So, in the first try, we just use the first 100 rows as training set and the remaining 100 rows as validation set.

```
train = data[:100]
```

```
train
```

	Unnamed: 0	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9
...	...	...	...	...	...
95	96	163.3	31.6	52.9	16.9
96	97	197.6	3.5	5.9	11.7
97	98	184.9	21.0	22.0	15.5
98	99	289.7	42.3	51.2	25.4
99	100	135.2	41.7	45.9	17.2

100 rows × 5 columns

Table 4: The training data set

```
validation = data[100:]
```

```
validation
```

	Unnamed: 0	TV	Radio	Newspaper	Sales
100	101	222.4	4.3	49.8	11.7
101	102	296.4	36.3	100.9	23.8
102	103	280.2	10.1	21.4	14.8
103	104	187.9	17.2	17.9	14.7
104	105	238.2	34.3	5.3	20.7
...	...	...	...	...	...
195	196	38.2	3.7	13.8	7.6
196	197	94.2	4.9	8.1	9.7
197	198	177.0	9.3	6.4	12.8
198	199	283.6	42.0	66.2	25.5
199	200	232.1	8.6	8.7	13.4

100 rows × 5 columns

Table 5: The validation data set

After splitting, we can obtain the lower bound L_1 and the upper bound U_1 of advertising expenditures in TV is 5.4 and 293.6 separately, the lower bound L_2 and the upper bound U_2 of advertising expenditures in Radio is 1.4 and 49.6 separately from the training data set.

Then on the training set, we use TV (x_1) and Radio (x_2) as regressors and Sales (ω) as the target to build a linear regression model.

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	3.1470	0.398	7.900	0.000	2.356	3.938
TV	0.0459	0.002	24.395	0.000	0.042	0.050
Radio	0.1841	0.011	16.918	0.000	0.163	0.206

Table 6: The linear regression model

The result shows that $\beta_0 = 3.1470$, $\beta_1 = 0.0459$ and $\beta_2 = 0.1841$. Then we calculate the error of each observation in both training set and validation set using

$$\varepsilon_{ti} := \omega_i - \beta_0 - \beta_1 x_{1i} - \beta_2 x_{2i}$$

$$\varepsilon_{vi} := \omega_i - \beta_0 - \beta_1 x_{1i} - \beta_2 x_{2i}$$

Now we have two error sets ε_t and ε_v, ε_t for the training set and ε_v for the validation set.

3.2 Validation for Predictive Model

Before moving to prescriptive analytics, we need to validate that the predictive analytics is right. The validation is about the error. First, we do F-test to check whether ε_t and ε_v are drawn from the same distribution. The p-value is 0.20451, greater than 0.05, meaning that we cannot reject the null hypothesis, they are drawn from the same distribution. We also plot the histograms of ε_t and ε_v to show their same distribution.

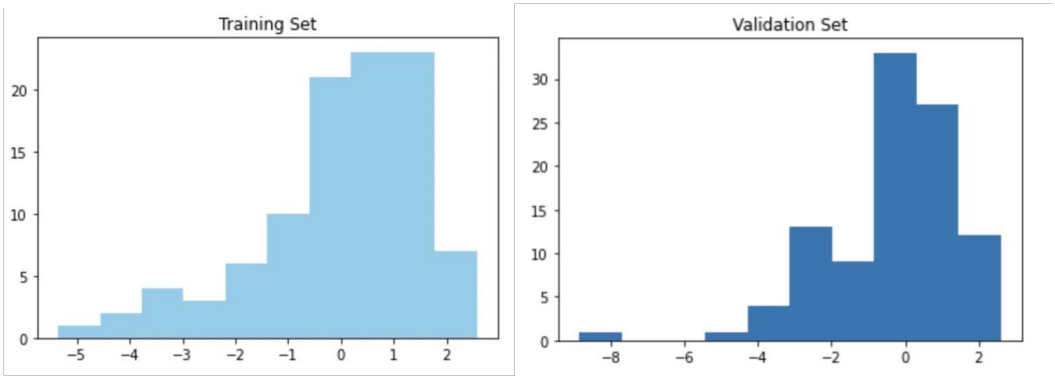

Figure 2: histograms of ε_t and ε_v

To illustrate ε_t and ε_v are normally distributed, we plot the Q-Q graphs, and the results are as below.

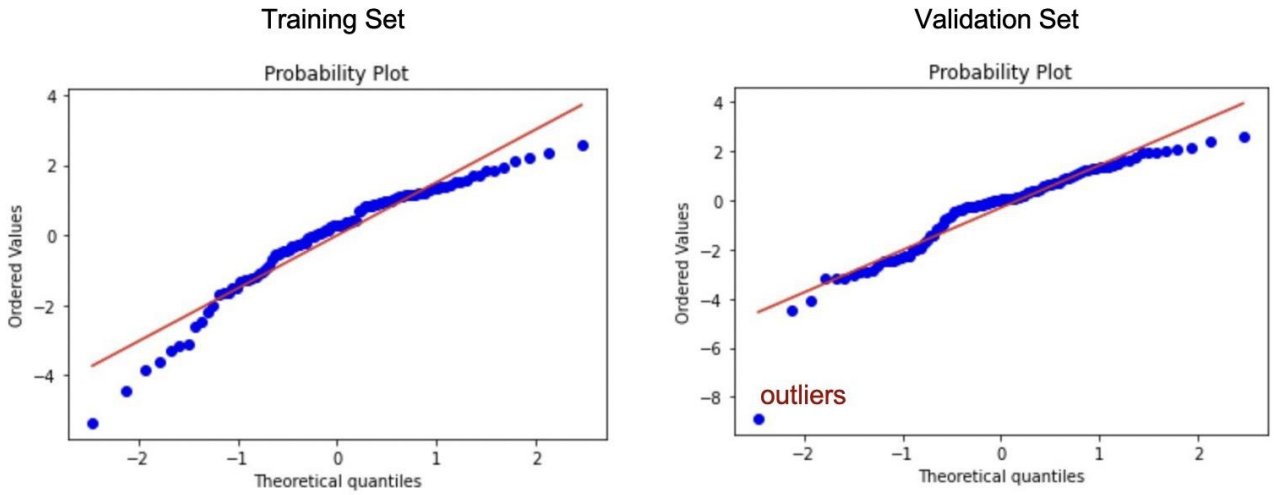

Figure 3: Q-Q graphs of ε_t and ε_v

We can see most observations fall on the 45-degree line, meaning ε_t and ε_v are normally distributed. We can use the predictive model in the following prescriptive analytics. However, we notice that in F-test, the p-value is not large enough, the histograms are both left skewed and there are several outliers on the left and the right of the Q-Q graphs, so we decide to move these outliers later in our creative work section to improve the accuracy of our models.

4. Prescriptive Analytics

After finding the relationship between x_1, x_2 and ω, we can substitute ω in the second stage subproblem with $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon$, ε could introduce the domain uncertainty for this problem.

4.1 Deterministic Linear Programming and Validation

Let us consider the simplest situation, that is $\varepsilon = 0$, without any uncertainty. So we can write the linear programming which is deterministic as below:

$$
\begin{aligned}
\text{Max} \quad & -0.1x_1 - 0.5x_2 + \quad 3y_A + 5y_B & \\
\text{s.t.} \quad & x_1 + x_2 & \le 200 \\
& x_1 - 0.5x_2 & \ge 0 \\
& y_A & \le 8 \\
& 2y_B & \le 24 \\
& 3y_A + 2y_B & \le 36 \\
\text{deterministic} \quad & -\beta_1 x_1 - \beta_2 x_2 + y_A + y_B & \le \beta_0 \\
& y_A, y_B \ge 0 & \\
& L_1 \le x_1 \le U_1, \quad L_2 \le x_2 \le U_2 &
\end{aligned}
$$

Solving this deterministic linear programming problem gives us the answer $x_1 = 173.42$, $x_2 = 26.58$, $MPO = 41369.03$ (MPO: the objective function estimate provided by the optimization model).

To validate the result of the optimization program, we first calculate ω_i using

$$\omega_i := \beta_0 + \beta_1 \hat{x}_1 + \beta_2 \hat{x}_2 + \varepsilon_{vi}$$

with fixed $\hat{x}_1 = 173.42$, $\hat{x}_2 = 26.58$ and each ε_{vi} from ε_v. Since we have 100 observations in the validation set, meaning we have 100 ε_{vi} in ε_v, and we will have 100 ω_i denoting 100 scenarios. Then we calculate profit for each scenario $Profit(\omega_i)$ by solving the second stage subproblem:

$$\begin{aligned} \text{Profit}(\omega) \quad = \quad \text{Max} \quad & 3y_A + 5y_B \\ \text{s.t.} \quad & y_A \leq 8 \\ & 2y_B \leq 24 \\ & 3y_A + 2y_B \leq 36 \\ & y_A + y_B \leq \omega \\ & y_A, y_B \geq 0 \end{aligned}$$

Now we have 100 profits. Adding -0.1$\hat{x}_1$-0.5$\hat{x}_2$ to each profit brings us back to the first stage — we have 100 object values of the whole problem. We can calculate the mean μ and standard deviation σ of the 100 scenarios (object values), then we can calculate the confidence interval using

$$[\mu - 1.96\frac{\sigma}{\sqrt{n}}, \mu + 1.96\frac{\sigma}{\sqrt{n}}]$$

which gives us MVSAE (Model Validation Sample Average Estimate). Now, we can conclude the result of the deterministic linear programming is:

x_1	x_2	$MPO\ (in\ \$)$	$MVSAE\ (in\ \$)$
173.42	26.58	41369.03	[37930.37, 39822.96]

Table 7: The result of deterministic linear programming

We notice that MPO is inconsistent with MVSAE, that means not considering the uncertainty in the first stage can lead to a wrong result.

4.2 Stochastic Linear Programming: Sample Average Approximation and Validation

From deterministic linear programming, we should be aware that we need an optimization approach accommodating the assumptions of the multiple linear regression — Stochastic Linear Programming (SLP), which allows a representation of data uncertainty using discrete random variables in an optimization model.

We replace the expectation operator E in the first stage with the average of finite profits because every scenario has the same empirical probability $\frac{1}{N}$. Now we can write all 100 scenarios in an All-In-One problem as below:

$$
\begin{aligned}
\text{Max } & -0.1x_1 - 0.5x_2 + \frac{1}{N}\sum_{i=1}^{N}(3y_{Ai} + 5y_{Bi}) \\
\text{s.t.} \quad & x_1 + x_2 \le 200 \\
& x_1 - 0.5x_2 \ge 0 \\
& y_{Ai} \le 8 \quad i = 1, \ldots, N \\
& 2y_{Bi} \le 24 \quad i = 1, \ldots, N \\
& 3y_{Ai} + 2y_{Bi} \le 36 \quad i = 1, \ldots, N \\
& -\beta_1 x_1 - \beta_2 x_2 + y_{Ai} + y_{Bi} \le \beta_0 + \varepsilon_{ti} \quad i = 1, \ldots, N \\
& L_1 \le x_1 \le U_1, \quad L_2 \le x_2 \le U_2, y_{Ai}, y_{Bi} \ge 0.
\end{aligned}
$$

Where ε_{ti} introduces the domain uncertainty to this problem. Here we consider the uncertainty when solving the *stochastic linear programming (SLP)* and replace the expectation of profit with a *sample average approximation (SAA)*.

To validate, we can calculate ω_i , $Profit(\omega_i)$ and $MVSAE$ via the same approach as described in section 4.1, the only difference is that we use the fixed $\hat{x}_1 = 184.86$, $\hat{x}_2 = 15.14$ from the result of solving this SLP problem. The concluded result and the comparison with deterministic linear programming are as below:

Methodology	x_1	x_2	MPO $(in\ \$)$	$MVSAE$ $(in\ \$)$
Deterministic LP	173.42	26.58	41369.03	[37930.37, 39822.96]
SLP with SAA	184.86	15.14	40860.69	[38624.25, 41066.54]

Table 8: The result of deterministic LP and SLP with SAA

We can see in SLP with SAA, the MPO is consistent with MVSAE perfectly. Comparing with Deterministic LP, the decisions (x_1, x_2) in SLP with SAA are quite different, the validated expected profit is much higher.

4.3 Stochastic Decomposition: The Case for Replications and Validation

One way to reduce variance is by using replications. Stochastic Decomposition (SD) is one method which automatically undertakes replications in SLP. Stochastic Decomposition is a sequential sampling algorithm based on sampling from the training set, drawing data points by sampling from the empirical distribution. It repeats the re-sampling process as many times as desired to make better decisions. Using replications, SLP with SD can reduce variance and be able to generate decisions which are more likely to produce lower variability in the validation phase, too. From analysis above, the randomness appears only through the multiple linear regression model, hence the problem obeys the assumptions for stochastic decomposition. We still use the formula below to run our SD algorithm.

$$
\begin{aligned}
\text{Max} \quad & -0.1x_1 - 0.5x_2 + \quad 3y_A + 5y_B & & \\
\text{s.t.} \quad & x_1 + x_2 & & \leq 200 \\
& x_1 - 0.5x_2 & & \geq 0 \\
& & y_A & \leq 8 \\
& & 2y_B & \leq 24 \\
& & 3y_A + 2y_B & \leq 36 \\
& -\beta_1 x_1 - \beta_2 x_2 + y_A + y_B & & \leq \boxed{\beta_0} \\
& & y_A, y_B \geq 0 & \quad \text{change} \\
L_1 \leq x_1 \leq U_1, & \quad L_2 \leq x_2 \leq U_2 & &
\end{aligned}
$$

However, this time, β_0 will change each run. We break the problem into two stages and change β_0 in the second stage in each run. Finally, we obtain the decision result (x_1, x_2) after many repeats.

```
@variable(model, x1 >= 0)
@variable(model, x2 >= 0)
@variable(model, yA >= 0)
@variable(model, yB >= 0)

@constraint(model, con1, x1 + x2 <= 200)
@constraint(model, con2, x1 - 0.5*x2 >= 0)
@constraint(model, con3, 5.4 <= x1)
@constraint(model, con4, x1 <= 293.6)
@constraint(model, con5, 1.4 <= x2)
@constraint(model, con6, x2 <= 49.6)
@constraint(model, con7, yA <= 8)
@constraint(model, con8, 2*yB <= 24)
@constraint(model, con9, 3*yA + 2*yB <= 36)
@constraint(model, con10, -0.0459*x1 - 0.1841*x2 + yA + yB <= 3.1470)

@objective(model, Min, 0.1*x1 + 0.5*x2 - 3*yA - 5*yB)

split_position = Position(con7, yA)
```

```
function mystoc()::OneRealization
    error = sample(error_train)
    binding = [Position(con10, "RHS") => 3.1470 + error]
    return OneRealization(binding)
end

user_mean = [3.1470]

result = solve_sd(model, split_position, user_mean, mystoc)

decision(x1, result, CompromiseSolution)
```

To validate the result of SD, we calculate ω_i , $Profit(\omega_i)$ and $MVSAE$ via the same approach as described in section 4.1, the only difference is that we use the fixed $\hat{x}_1 = 184.97, \hat{x}_2 = 15.04$ from the result of solving this SD problem. The concluded result and the comparison with deterministic linear programming and SLP with SAA are as below:

Methodology	x_1	x_2	$MPO\ (in\ \$)$	$MVSAE\ (in\ \$)$
Deterministic LP	173.42	26.58	41369.03	[37930.37, 39822.96]
SLP with SAA	184.86	15.14	40860.69	[38624.25, 41066.54]
SLP with SD	184.97	15.03	40953.13 [40672.17, 41234.09]	[38624.25, 41098.54]

Table 9: The result of deterministic LP, SLP with SAA and SLP with SD

5. Results

From *Table 7*, we find both SLP with SAA and SLP with SD show an MPO fall close to the median of their own MVSAE while SLP with SD is much closer. The most likely reason is that replications from SD reduce variance and SD provides us the compromise decision which has better variance reduction compared with the optimal solution from any other run.

6. Creative Work

To improve the accuracy of such an integrative model, we can try several methods like dropping outliers, using different train-test splits, and using different regression models in the descriptive analytics phase.

6.1 Drop Outliers

From Section 3.2, we noticed that in F-test, the p-value is not large enough, the histograms are both left skewed and there are several outliers on the left and the right of the Q-Q graphs, so we decide to drop these outliers and retrain the linear regression model.

We only choose observations within one standard deviation from the mean for both training set and validation set. Then we have 76 observations in the training set and 71 observations in the validation set.

`train`

	Unnamed: 0	TV	Radio	Newspaper	Sales	error
0	1	230.1	37.8	69.2	22.1	1.43243
3	4	151.5	41.3	58.5	18.5	0.79582
4	5	180.8	10.8	58.4	12.9	-0.53400
6	7	57.5	32.8	23.5	11.8	-0.02473
7	8	120.2	19.6	11.6	13.2	0.92746
...	...	...	...	...	...	...
95	96	163.3	31.6	52.9	16.9	0.43997
96	97	197.6	3.5	5.9	11.7	-1.16119
97	98	184.9	21.0	22.0	15.5	-0.00001
98	99	289.7	42.3	51.2	25.4	1.16834
99	100	135.2	41.7	45.9	17.2	0.17035

76 rows × 6 columns

`validation`

	Unnamed: 0	TV	Radio	Newspaper	Sales	error
101	102	296.4	36.3	100.9	23.8	0.36541
103	104	187.9	17.2	17.9	14.7	-0.23813
104	105	238.2	34.3	5.3	20.7	0.30499
105	106	137.9	46.4	59.0	19.2	1.18115
106	107	25.0	11.0	29.7	7.2	0.88040
...	...	...	...	...	...	...
193	194	166.8	42.0	3.6	19.6	1.06468
194	195	149.7	35.6	6.0	17.3	0.72781
196	197	94.2	4.9	8.1	9.7	1.32713
197	198	177.0	9.3	6.4	12.8	-0.18343
199	200	232.1	8.6	8.7	13.4	-1.98365

71 rows × 6 columns

Table 10: The new training data set (left) and validation data set (right)
(After dropping outliers)

Using the data from new training set, we have the linear regression model with $\beta_0 = 3.5250$, $\beta_1 = 0.0441$ and $\beta_2 = 0.1907$. For the new ε_t and ε_v, p-value of F-test is $0.1734 > 0.05$, meaning they are drawn from the same distribution. Histograms and Q-Q graphs are also shown as below. They are normally distributed.

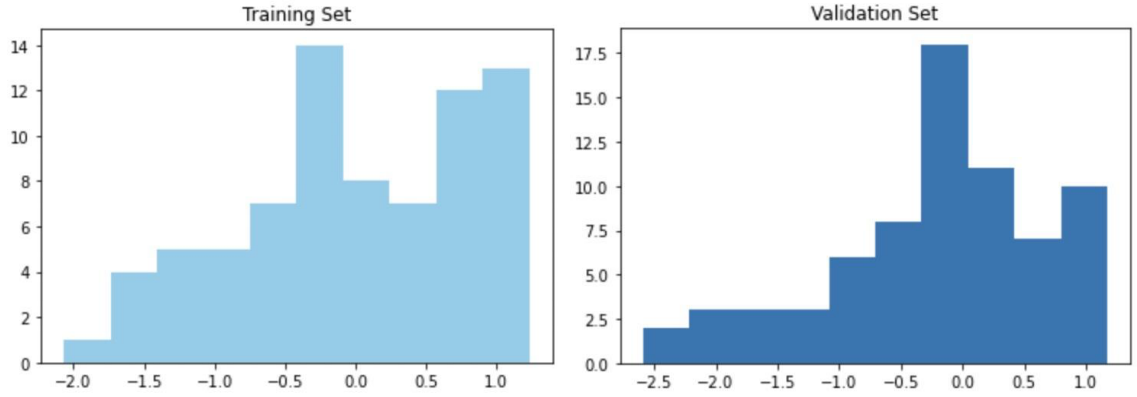

Figure 4: histograms of ε_t and ε_v(after dropping outliers)

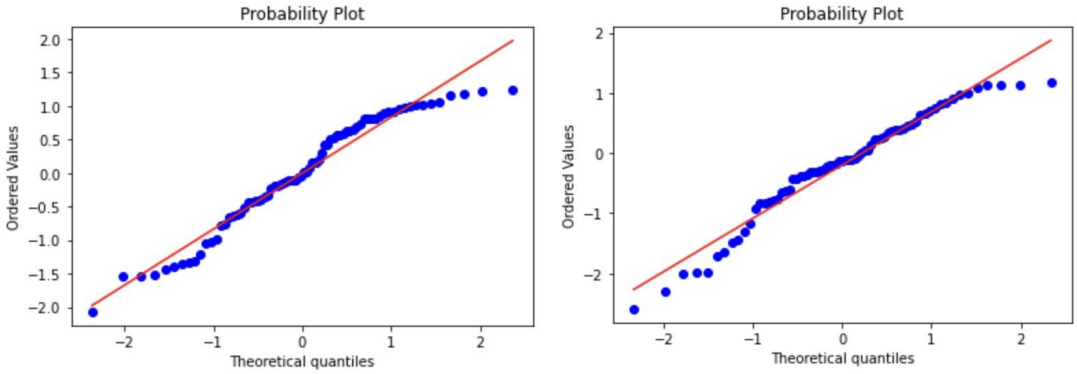

Figure 5: Q-Q graphs of ε_t (left) and ε_v (right)
(After dropping outliers)

We do Deterministic LP, SLP with SAA and SLP with SD again using the same steps as illustrated before to calculate the new x_1, x_2, MPO and MVSAE for each methodology. *Table 9* here is pasted from section 4.3 just to make a comparison.

Methodology	x_1	x_2	$MPO\ (in\ \$)$	$MVSAE\ (in\ \$)$
Deterministic LP	173.42	26.58	41369.03	[37930.37, 39822.96]
SLP with SAA	184.86	15.14	40860.69	[38624.25, 41066.54]
SLP with SD	184.97	15.03	40953.13 [40672.17, 41234.09]	[38624.25, 41098.54]

Table 9: The result of deterministic LP, SLP with SAA and SLP with SD
(Without dropping outliers)

Methodology	x_1	x_2	$MPO\ (in\ \$)$	$MVSAE\ (in\ \$)$
Deterministic LP	175.07	24.93	42027.29	[40314.67, 41081.58]
SLP with SAA	182.06	17.94	41711.36	[40619.69, 41649.02]
SLP with SD	182.02	17.98	41956.89 [41687.19, 42226.58]	[40603.69, 41633.02]

Table 11: The result of deterministic LP, SLP with SAA and SLP with SD
(After dropping outliers)

Surprisingly, we find after dropping outliers, the MPO does not fall into the MVSAE in both SLP with SAA and SLP with SD. The reason may be that we do not have enough information from the training set. So, we will not drop such "outliers" in the following tries.

6.2 A Different Train-Test Split

Since we just use the first 100 observations as the training set and the remaining 100 observations as the validation set in the first try, we use a different random train-test split in this section.

	TV	Radio	Sales
166	17.9	37.6	8.0
36	266.9	43.8	25.4
153	171.3	39.7	19.0
138	43.0	25.9	9.6
137	273.7	28.9	20.8
...	...	...	...
43	206.9	8.4	12.9
180	156.6	2.6	10.5
101	296.4	36.3	23.8
195	38.2	3.7	7.6
53	182.6	46.2	21.2

100 rows × 3 columns

	TV	Radio	Sales
68	237.4	27.5	18.9
45	175.1	22.5	14.9
152	197.6	23.3	16.6
176	248.4	30.2	20.2
21	237.4	5.1	12.5
...	...	...	...
124	229.5	32.3	19.7
64	131.1	42.8	18.0
74	213.4	24.6	17.0
196	94.2	4.9	9.7
156	93.9	43.5	15.3

100 rows × 3 columns

Table 12: The new training data set (left) and validation data set (left)
(New train-test split)

The new linear regression model has coefficients $\beta_0 = 2.5635$, $\beta_1 = 0.0454$ and $\beta_2 = 0.2000$. For the new ε_t and ε_v, p-value of F-test is 0.2815 > 0.05, meaning they are drawn from the same distribution. Histograms and Q-Q graphs are also shown as below. They are normally distributed.

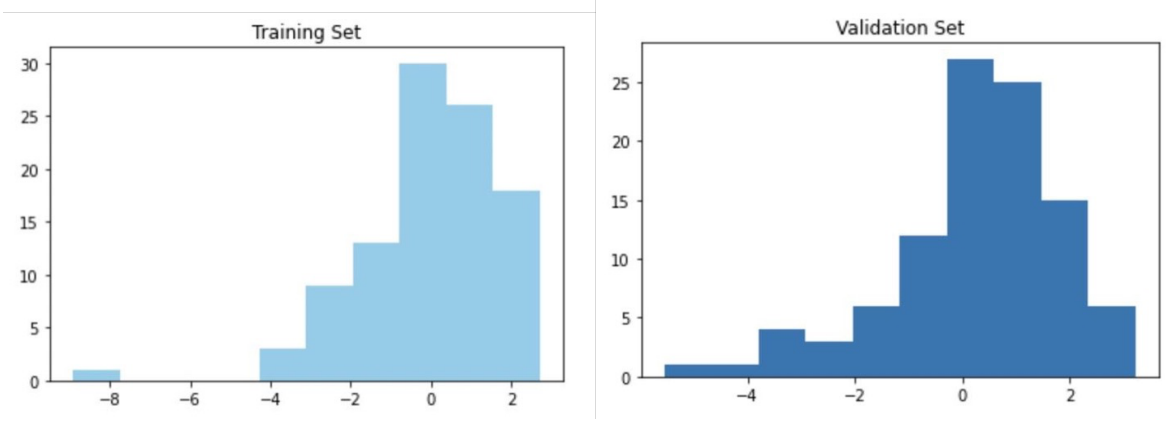

Figure 6: histograms of ε_t and ε_v
(New train-test split)

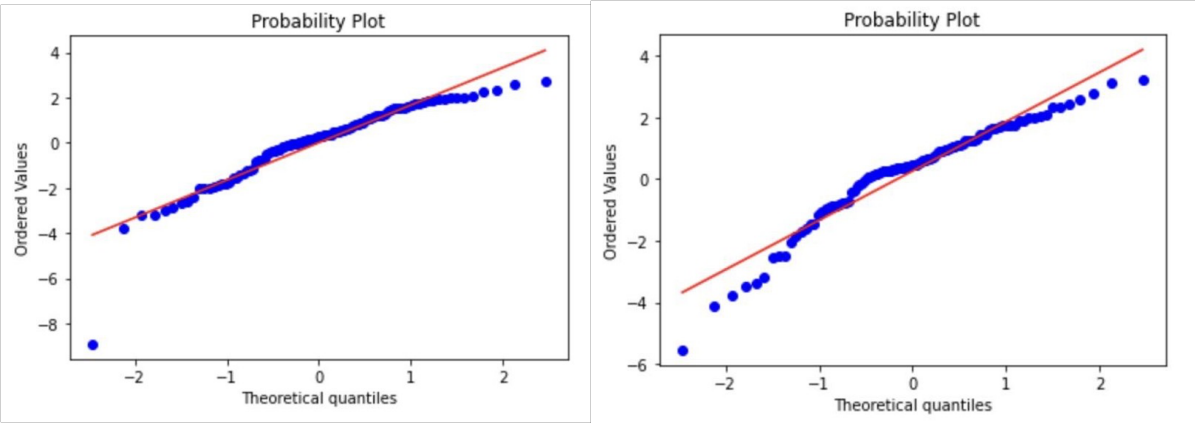

Figure 7: Q-Q graphs of ε_t (left) and ε_v (right)
(New train-test split)

The results for three methodologies are as below. Pretty much the same as *Table 9*.

Methodology	x_1	x_2	$MPO\ (in\ \$)$	$MVSAE\ (in\ \$)$
Deterministic LP	171.82	28.18	40728.33	[38493.55, 39818.43]
SLP with SAA	181.72	18.28	39651.15	[39351.80, 41319.18]
SLP with SD	181.76	18.24	39716.67 [39435.49, 39997.86]	[39367.80, 41335.18]

Table 13: The result of deterministic LP, SLP with SAA and SLP with SD
(New train-test split)

6.3 L1-Regularized (LASSO) Regression

Using the same train-test split, we train a L1-Regularized (LASSO) regression model. The newly trained L1-Regularized (LASSO) regression model has coefficients $\beta_0 = 3.4633$, $\beta_1 = 0.0448$ and $\beta_2 = 0.1778$. For the new ε_t and ε_v, p-value of F-test is 0.1784 > 0.05, meaning they are drawn from the same distribution. Histograms and Q-Q graphs are also shown as below. They are normally distributed.

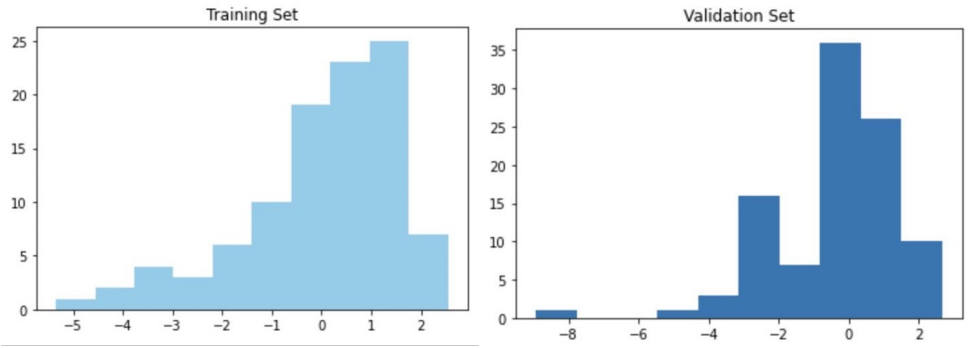

Figure 8: histograms of ε_t and ε_v
(LASSO Regression)

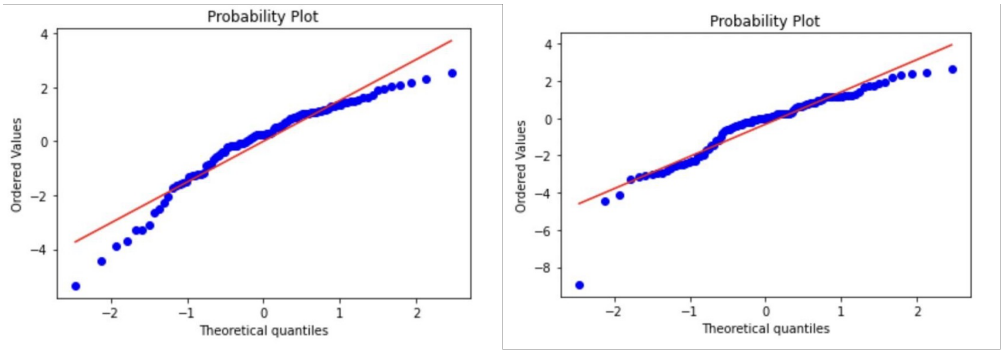

Figure 9: Q-Q graphs of ε_t (left) and ε_v (right)
(LASSO Regression)

The results for three methodologies are as below. SLP with SD looks strange because its MPO does not fall in its MVSAE.

Methodology	x_1	x_2	$MPO\ (in\ \$)$	$MVSAE\ (in\ \$)$
Deterministic LP	198.60	1.40	41268.50	[37652.87, 40641.99]
SLP with SAA	187.28	12.72	40926.39	[38521.39, 41070.47]
SLP with SD	187.02	12.98	41047.15 [40769.18, 41325.11]	[38417.39, 40966.47]

Table 14: The result of deterministic LP, SLP with SAA and SLP with SD
(LASSO Regression)

6.4 L2-Regularized (Ridge) Regression

Using the same train-test split, we train a L2-Regularized (Ridge) regression model. The L2-Regularized (Ridge) regression model has coefficients $\beta_0 = 3.2489$, $\beta_1 = 0.0454$ and $\beta_2 = 0.1825$. For the new ε_t and ε_v, p-value of F-test is 0.1973 > 0.05, meaning they are drawn from

the same distribution. Histograms and Q-Q graphs are also shown as below. They are normally distributed.

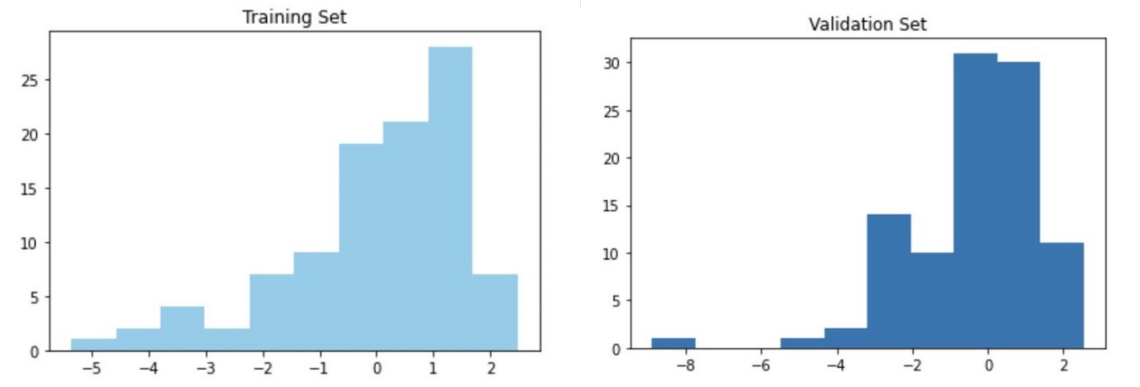

Figure 10: histograms of ε_t and ε_v
(Ridge Regression)

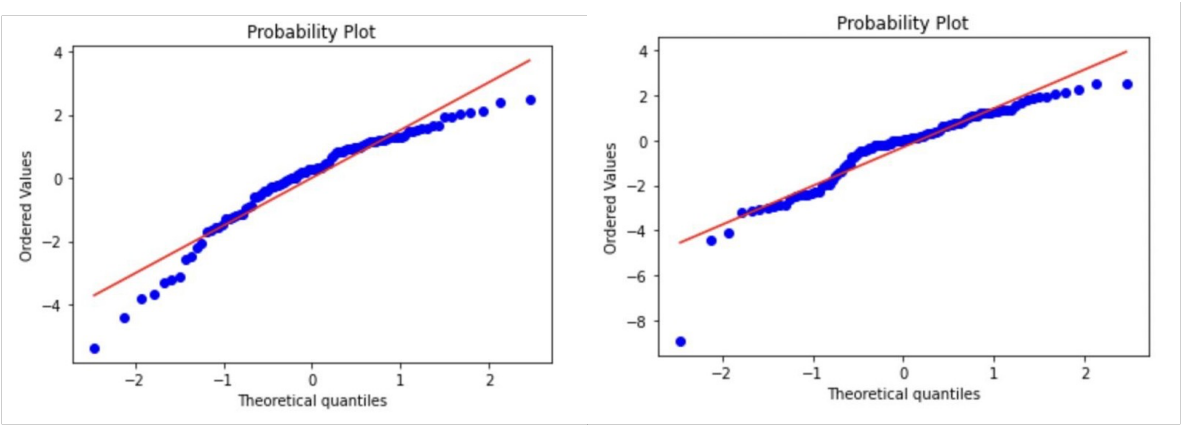

Figure 11: Q-Q graphs of ε_t (left) and ε_v (right)
(Ridge Regression)

The results for three methodologies are as below.

Methodology	x_1	x_2	$MPO\ (in\ \$)$	$MVSAE\ (in\ \$)$
Deterministic LP	173.22	26.78	41289.28	[37853.91, 39749.17]
SLP with SAA	185.26	14.74	40851.99	[38578.40, 41041.39]
SLP with SD	185.08	14.92	40962.95 [40682.23, 41243.67]	[38506.40, 40969.39]

Table 15: The result of deterministic LP, SLP with SAA and SLP with SD
(Ridge Regression)

Creative Contribution

1. Jiachen Tu

Explored convergence of SQG method in project 1, proposed glide path method for solving MIP problem. Wrote code and report in project 1, code in project 2.

2. Jiaqi Ma

Project 1: Wrote the Introduction part and the Problem Description & Assumptions part of the report.
Project 2: Wrote the Problem Description part of the report.
Project 3: Tried different methods including dropping outliers, using a different train-test split, using L1-Regularized (LASSO) Regression, using L2-Regularized(Ridge) Regression; wrote the report.

3. Yiwen Chen

For project 1 I focused on formulating the All-in-One method including algebraic formulation and Julia formulation, as well as the presentation preparation

For project 2 I have completed most parts for the matrix completion problem as well as the mix integer problem. I have been looking at multiple approaches to solve the matrix completion, including MAP coordinate ascent algorithm, Stochastic Gradient Descent, etc. I also looked at how to use grid search to find the optimal value for hyperparameters such as d, learning rate, etc. For the MIP modeling, I suggested to add constraint to make menu more variable in a weekly menu. I also had an idea trying to maximize the average total rating per day, rather than maximize the total rating. (However, since the basic model took too long to execute, I did not have a chance to test out this idea). Jiachen and I suggested to apply a glide path strategy in generating the 5-day plan so that the processing time is significantly speed up while ensuring menu variability. However, we noticed that using glide path would resulting a decreasing total rating and budget across day 1 to day 5, which is not desired in real meal planning situation. This issue might be resolved or mitigated if we aim to maximize the average score per day, but this has not been figured out yet and would need further exploration.

Reference

L. Zhao and S. Sen, "A Comparison of Sample-Path-Based Simulation-Optimization and Stochastic Decomposition for Multi-Location Transshipment Problems," Proceedings of the 2006 Winter Simulation Conference, 2006, pp. 238-245, doi: 10.1109/WSC.2006.323079.

Predictive stochastic programming, Yunxiao Deng, Suvrajeet Sen, 2019

R. Salakhutdino and A. Mnihv. Probabilistic Matrix Factorization. (2008), NIPS Proceedings. https://proceedings.neurips.cc/paper/2007/file/d7322ed717dedf1eb4e6e52a37ea7bcd-Paper.pdf

Yale T. Herer, Michal Tzur & Enver Yücesan (2006) The multilocation transshipment problem, IIE Transactions, 38:3, 185-200, DOI: 10.1080/07408170500434539