Alice Duan
Dexin Qian

Restaurant Project

1.Introduction

This project is aimed to help UBC students planning out their diet, by examining the data records of the restaurant in BC. 373 restaurants are presented in the raw data where we will carefully filter out restaurants that are relatively close to UBC (Kumar, 2020).
How to eat well on a tight budget is always a problem for university students. Due to the outbreak of covid-19, many restaurants experienced high levels of pressure. According to news, up to 24 percent were breaking during this special time (Lazaruk, 2021). As the circumstance becomes better recently, many reopened restaurants start to increase their price in order to recover from the loss due to covid-19. According to Canada's Food Price Report 11th annual edition, there is a 5% overall food price increase in 2021(Canada's food price report, 2021). As meat and vegetables were predicted to have a growth of 4.5 to 6.5 percent (Canada's food price report, 2021). Since university students often tend to experience high volumes of coursework, cooking by themself is at a high time expense. Getting a food delivery could be a solution to this problem. However, delivery fees, as well as tips, are not cheap for students in general, especially most students don't have a steady job. What's more, reports have also shown that due to the increase in fuel costs, the shipping price is also growing (Lazaruk, 2021). Thus, it is important to construct a way to help UBC students to eat well at an affordable cost.
Consequently, we aimed to use this project to help students in UBC plan out their eating diet. Restaurant rating, prices, and locations were used to construct a linear programming problem that best design the recommended frequency to eat in each restaurant.

2. Developing the model

2.1Terminology and Definitions

x_n: The number of times should the nth restaurant in the filtered table be visited per week
r_n: The rating of the nth restaurant computed by R(x_n)
a_n: The transit time from the nth restaurant to UBC, estimated by their address according to the "landmark" column
b_n: The average cost of the nth restaurant, according to the "cost" column in the database
t: The upper bound of the sum of the time spending on transit to the restaurants per week (min)
c: The upper bound of the sum of the eating cost per week
m: The upper bound of sum of the meal a student wish to have in the restaurant per week
n: The upper bound of the student to visit the same restaurant per week (never visit one

restaurant for more than times per week)

2.2Assumptions

- Since the delivery fee can be expensive, we assume the student would reach the restaurant through transit, and not order the delivery. Therefore, we filter out the rows where both of the "Diner in ability" and "Take away ability" are false or missing.

- We trust the rating of each restaurant in the database. However, considering other factors such as the numbers of reviews are different and people might lower their expectations on cheap restaurant, we are not using the "rating" column in the database alone for the final rating. We developed a function to estimate the final rating when treating the raw data.

- This project assumes the travel time each day does not varies much at the same time and the traffic condition would stay the same at each mealtime. Thus, we used 6:00 pm on Monday as the leaving time for conducting the research.

2.3Linear Programing Model

Objective function: max $\sum_{i=1}^{n} r_i \cdot x_i$

The objective function computes the sum of scheduled visiting times at each restaurant multiplied by the corresponding rating. Because we want to maximize the eating experience under all the constraints, we would like to maximize the objective.

Subject to:

$\sum_{i=1}^{n} a_i \cdot x_i \leq t$

The first constraint ensures that the sum of time cost on transit to each restaurant per week does not succeed the upper bound t.

$\sum_{i=1}^{n} b_i \cdot x_i \leq c$

The second constrain ensures that the sum of cost on each restaurant does not succeed the upper bound on cost c.

$\sum_{i=1}^{n} x_i = m$

The third constrain ensures that the total meal eaten per week equals to m.

$$0 \leq x_1, x_2, \dots, x_n \leq n$$

Because the student would not want to eat at the same restaurant for a whole week, the last constraint ensures that the time eating at the same store does not succeed n. Since students can't eat negative times at a restaurant x_i should all be greater than zero.

Additional Information

We want to develop a program that allows users to schedule their dining individually. Therefore, for variables such as t, c, m, n, users are free to decide whatever they want and input them into the program.

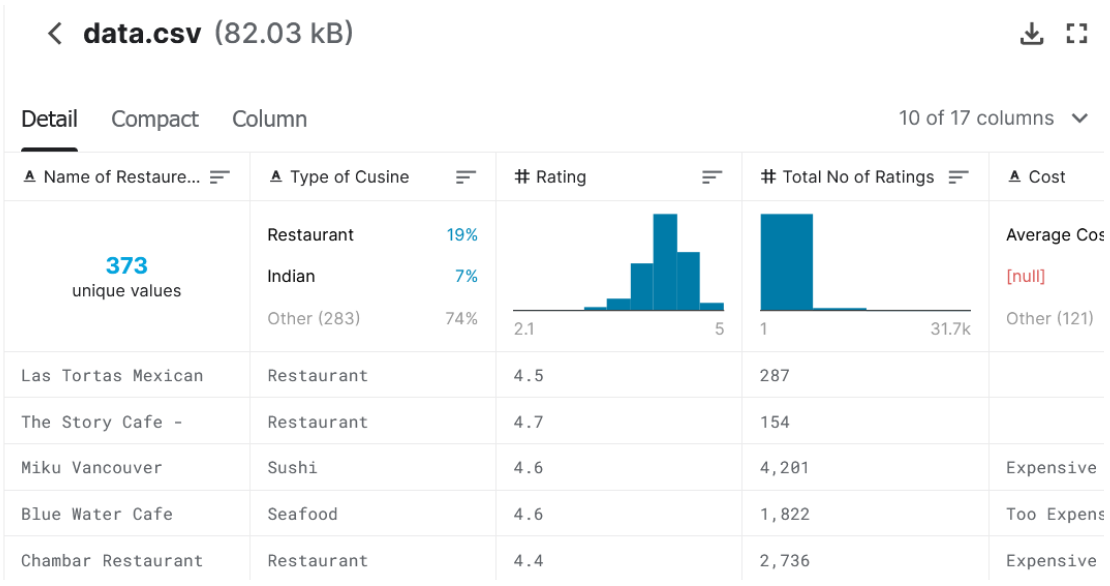
3.Procedures

3.1Pre-process of Data

We used python for examining and cleaning our data.

Choosing of Data

We utilize the database "Vancouver Restaurant Dataset" posted on Kaggle for our study. The data contains 373 rows of restaurants and 17 columns of their information such as "costs" and "ratings"(Kumar, 2020).

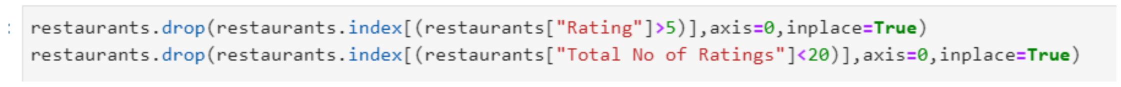


Name of Restaure...	Type of Cusine	Rating	Total No of Ratings	Cost
373 unique values	Restaurant 19%; Indian 7%; Other (283) 74%	2.1 – 5	1 – 31.7k	Average Cos; [null]; Other (121)
Las Tortas Mexican	Restaurant	4.5	287	
The Story Cafe -	Restaurant	4.7	154	
Miku Vancouver	Sushi	4.6	4,201	Expensive
Blue Water Cafe	Seafood	4.6	1,822	Too Expens
Chambar Restaurant	Restaurant	4.4	2,736	Expensive

Figure 1: The downloaded original database (The screenshot from the Kaggle)

Filtering

Columns containing missing data were filtered. Restaurants with low rating numbers (below 20 in our case) were also removed, which eliminates the data that isn't persuasive enough. As we are only looking at data that was rated out of 5, restaurants with a rating that is above 5 were also dropped.



```
restaurants.drop(restaurants.index[(restaurants["Rating"]>5)],axis=0,inplace=True)
restaurants.drop(restaurants.index[(restaurants["Total No of Ratings"]<20)],axis=0,inplace=True)
```

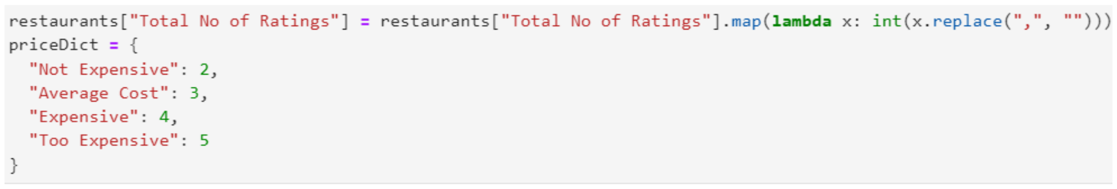
Figure 2: the python code to filter out “bad” data

Setting up Transit Time

The raw data does not contain the travel time nor the distance from UBC to each restaurant, so we computed the time ourselves. Since most university students commute using public transportation, we compute the travel time from UBC Bus Loop to each restaurant by using Google Map under the Transit section. The leaving time was set at 6:00 pm on Monday, a common time for dinner on weekdays. Travel time was recorded in minutes.

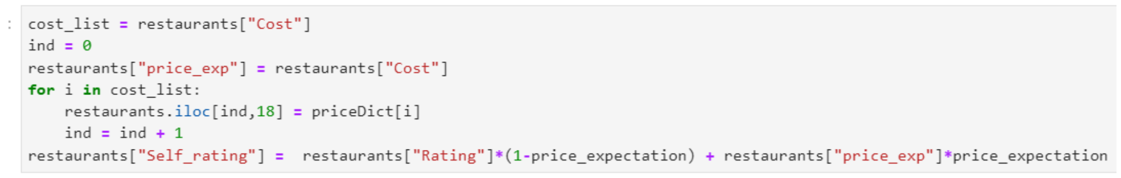
Re-evaluate the Rating

Since the more expensive restaurants might offer higher quality food, and when people rate more expensive restaurants they might have higher expectations, the price level of the restaurant should be considered as a factor in the evaluation. The raw data categorized the price of each restaurant into 4 levels: Not Expensive, Average Cost, Expensive, and Too Expensive. In order to evaluate, we assigned a corresponding number to each level: "Not Expensive": 2, "Average Cost": 3, "Expensive": 4, "Too Expensive": 5. However, the consideration of price is also the user's choice, for they can input the value of the “price_expectation” variable at the beginning of the program, which is a decimal between 0 and 1 that stands for the weight of the price on the final self-rating value. For example, if the user defines price expectation as 0.3, when calculating the self-rating value of the restaurant x with raw rating 4.6 and with price level as “average cost”, R(x) = 4.6*0.7 + 3*0.3 = 4.12; if the user inputs the price expectation as 0.2 when calculating the self-rating value of the restaurant y with raw rating 4.3 and with price level as “too expensive”, R(y) = 4.3*0.8 + 5*0.2 = 4.44.



```
restaurants["Total No of Ratings"] = restaurants["Total No of Ratings"].map(lambda x: int(x.replace(",", "")))
priceDict = {
  "Not Expensive": 2,
  "Average Cost": 3,
  "Expensive": 4,
  "Too Expensive": 5
}
```

Figure 3: the quantify of price standard



```
cost_list = restaurants["Cost"]
ind = 0
restaurants["price_exp"] = restaurants["Cost"]
for i in cost_list:
    restaurants.iloc[ind,18] = priceDict[i]
    ind = ind + 1
restaurants["Self_rating"] =  restaurants["Rating"]*(1-price_expectation) + restaurants["price_exp"]*price_expectation
```

Figure 4: the calculation of the “Self-rating” column (R(x))

Estimate the Price

As mentioned previously the price of each restaurant was recorded as categorical data. However, in our constraints, we are going to limit the total cost. Thus, the price of the restaurant per_meal under each category was estimated with approximate price: 10,20,40,60 respectively.

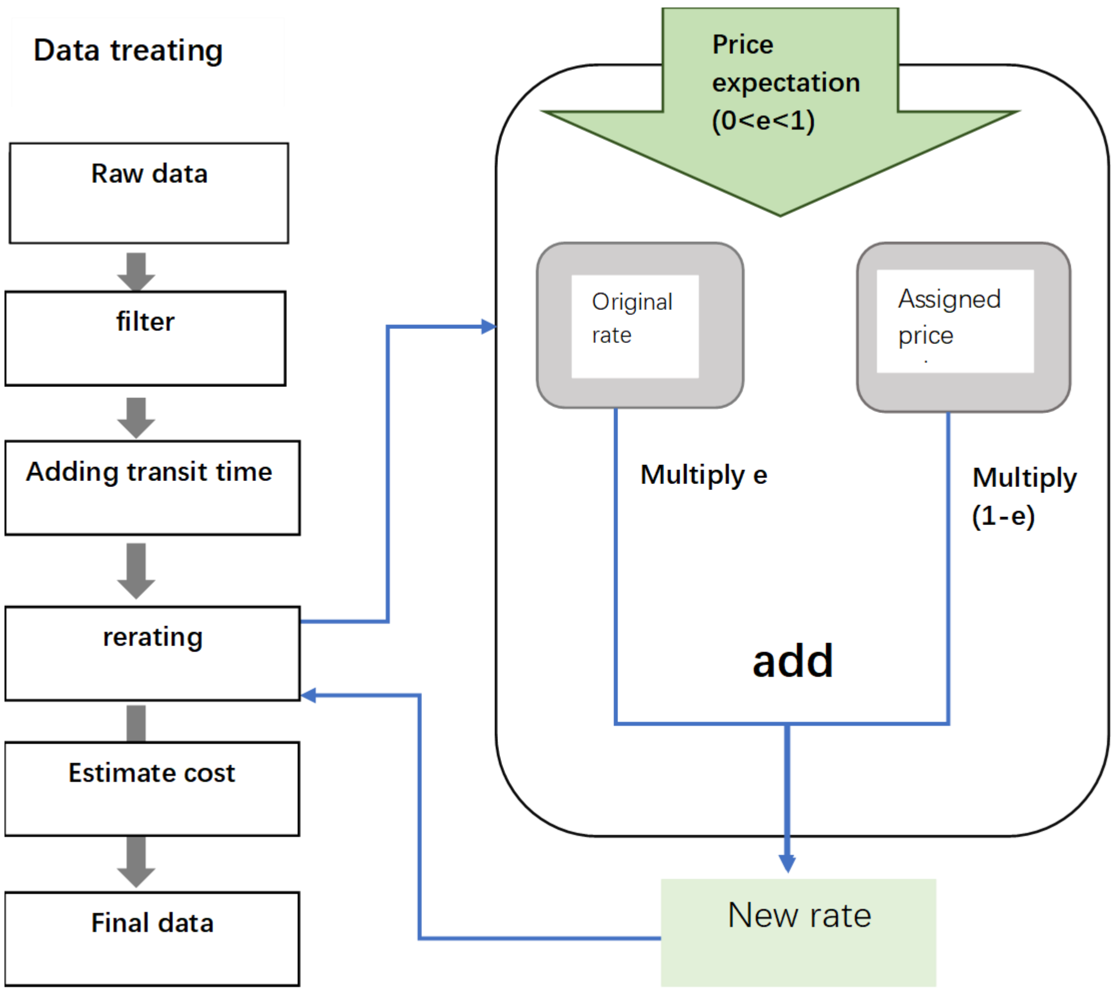


Figure 5: the flow chart of the pre-processing of data

3.2 Programing of the Model

Python PuLP pack was used to solve the Linear programming model.

LPMaximize was used to construct our LPproblem. A dictionary that contains the referenced variables was set for each restaurant. Since one is unable to go to a restaurant 0.5 times. The restaurant variables are constrained as integers with a lower bound of 0.

4.Result and Conclusion

Our program behaves differently when the input of the constants is different. We observed the following as we change the values of the user-defined variables.

The price expectation

The change of the price expectation influences the overall evaluation of all the restaurants, in a way that the smaller price expectation would make the final rating closer to the original rating and the larger expectation would make the more expensive restaurant have higher ratings.

If the upper bound of cost is high enough, a change in the result of linear programm ing can be observed. For example, when the maximum cost per meal is fixed at $50, when the price expectation index is 0.6, the final recommendation would be mostly th e restaurants that are rated as “too expensive” such as Bishop’s and St. Lawrence rest aurant; when the price expectation index is 0.3, the program will recommend more re staurants with average cost but higher rating such as Absinthe Bistro and Le Crocodil e.

The maximum cost per meal in a week

Since in our experimental data, the average price of all the restaurants are set to be at least $10, if the maximum cost per meal is set to be less than 10, the linear programming problem will be infeasible. The change of maximum cost within the feasible region will also change the linear programming result. When the maximum cost is exactly $10, only the “not expensive” rating restaurants such as the Hungry Guys Kitchen and the Northern Café would be recommended; when setting the maximum cost to $50, more expensive restaurants would be recommended, and the results will depend more on the price expectation index: the higher the index is, the more expensive restaurant would appear.

The maximum transit time per meal in a week

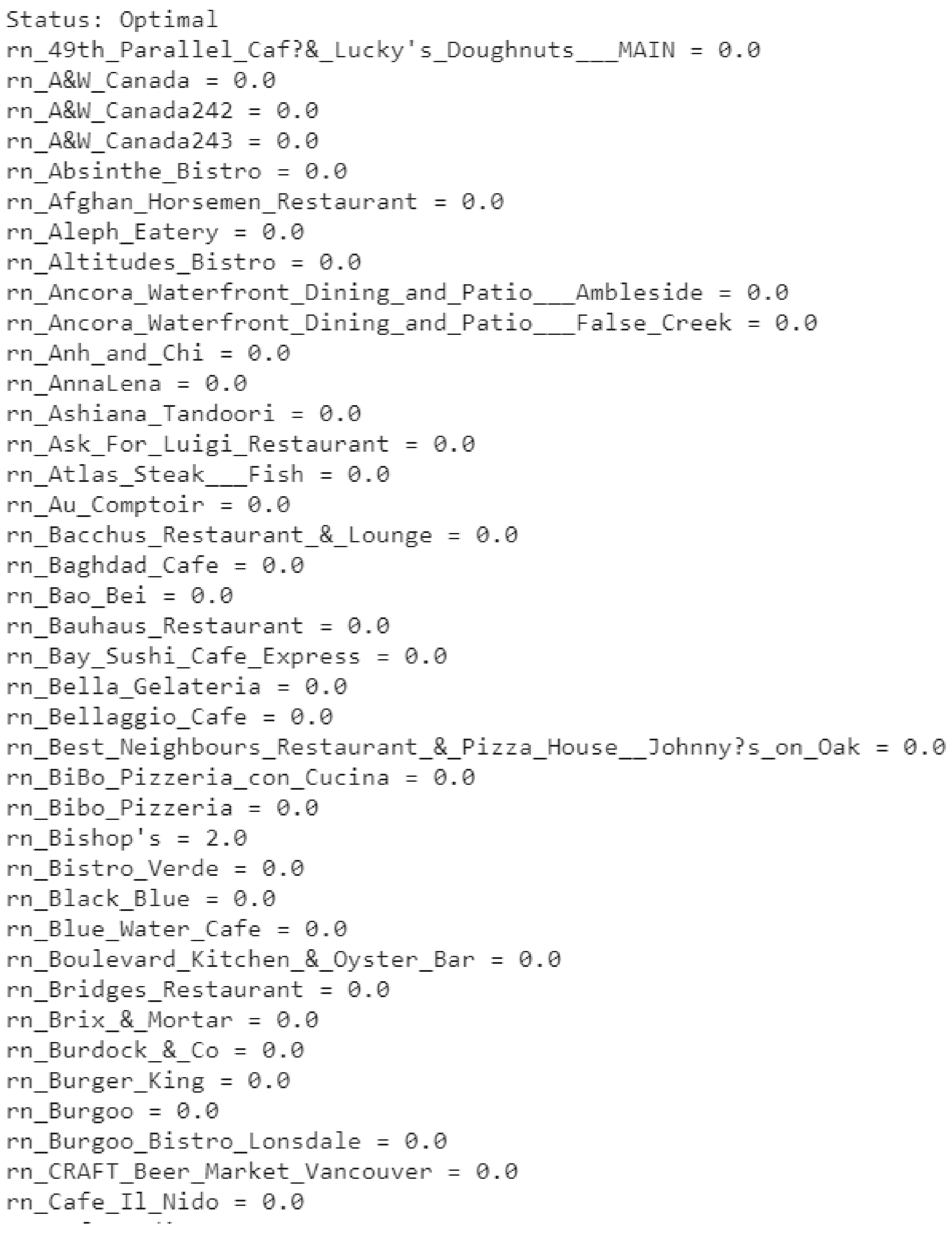
The change of maximum transit time per week will also affect the result. Since the minimum transit time to a restaurant is 5 minutes, any input that is less than 5 would cause the result to be infeasible. Other variables such as same store per week will also affect the LP problem’s feasibility when the maximum transit time is too small. For when the nearest restaurant used up all the visiting time this week, the program may seek for the second and third nearest, thus to satisfy the total number of meals in the week, the minimum average time spending on transit should be raised.

Overall, the change of the maximum transit time is similar to the change of maximum cost, setting other variables to be appropriate (price index = 0.3, max cost = 40, total meal = 4, same meal = 2), as the “t” is small, only near restaurants is recommended such as the Dark Table; but when “t” rises, other higher rating restaurants also appear in the result.

Total number meal per week and same store per week

The two number would affect the LP problem’s feasibility, especially when other factors are near the boundaries as we discussed in the change of maximum transit time. Also, these two variables could also be viewed as the conditions of potential sorting algorithm: the objective function and other constraints of the linear programing will sort the highest rated restaurant on the top and these two numbers will decide how many in the output and how many should be top one as well as how many should be top two respectively. For instance when total number of meals in a week is 4 and the same restaurant per week is 2, two top one and two top two restaurant under the other conditions would be in the result.

Additionally, we also observed some restaurants that are ideal in every aspect, for they can be seen in the result after modifying any of the variables, an example is the Bishop’s restaurant. Those should be restaurants highly recommended to UBC students for their excel overall quality.



```
Status: Optimal
rn_49th_Parallel_Caf?&_Lucky's_Doughnuts___MAIN = 0.0
rn_A&W_Canada = 0.0
rn_A&W_Canada242 = 0.0
rn_A&W_Canada243 = 0.0
rn_Absinthe_Bistro = 0.0
rn_Afghan_Horsemen_Restaurant = 0.0
rn_Aleph_Eatery = 0.0
rn_Altitudes_Bistro = 0.0
rn_Ancora_Waterfront_Dining_and_Patio___Ambleside = 0.0
rn_Ancora_Waterfront_Dining_and_Patio___False_Creek = 0.0
rn_Anh_and_Chi = 0.0
rn_AnnaLena = 0.0
rn_Ashiana_Tandoori = 0.0
rn_Ask_For_Luigi_Restaurant = 0.0
rn_Atlas_Steak___Fish = 0.0
rn_Au_Comptoir = 0.0
rn_Bacchus_Restaurant_&_Lounge = 0.0
rn_Baghdad_Cafe = 0.0
rn_Bao_Bei = 0.0
rn_Bauhaus_Restaurant = 0.0
rn_Bay_Sushi_Cafe_Express = 0.0
rn_Bella_Gelateria = 0.0
rn_Bellaggio_Cafe = 0.0
rn_Best_Neighbours_Restaurant_&_Pizza_House__Johnny?s_on_Oak = 0.0
rn_BiBo_Pizzeria_con_Cucina = 0.0
rn_Bibo_Pizzeria = 0.0
rn_Bishop's = 2.0
rn_Bistro_Verde = 0.0
rn_Black_Blue = 0.0
rn_Blue_Water_Cafe = 0.0
rn_Boulevard_Kitchen_&_Oyster_Bar = 0.0
rn_Bridges_Restaurant = 0.0
rn_Brix_&_Mortar = 0.0
rn_Burdock_&_Co = 0.0
rn_Burger_King = 0.0
rn_Burgoo = 0.0
rn_Burgoo_Bistro_Lonsdale = 0.0
rn_CRAFT_Beer_Market_Vancouver = 0.0
rn_Cafe_Il_Nido = 0.0
```

Figure 6: a screenshot of the LP result with the Bishop’s as the recommended restaurant

5.Possible drawbacks & Further improvement

There are no specific figures of cost in the original database and they only provided the level of costs such as “average”, “expensive”, and “very expensive”, we had given estimated values for each category according to research online. Thus, the actual price might be different from our taken value. Where we would have the optimal meal schedule to have either higher or lower cost than the given price limit. This problem could be improved by doing further research on the restaurant price data where we could find the pricing mean of each restaurant. Another possible drawback is that the time taken from UBC to the restaurant is computed using Google Maps. The time cost, in reality, might vary from the data we are taking due to weather and road conditions. Also, we manually searched on Google Maps for the transit time in our relatively small database (with 286 rows), to implement our program on bigger data, we should write functions or utilizes other contributors open source on the program and let it automatically generates the distance and calculate transit time given two addresses.

The program could also be improved by inputting more constraints to get a more specific optimal result. Such as given the choice of choosing different types of food.

References

Lazaruk, S. (2021, July 6). *Why prices at your favourite B.C. restaurants may be going up*. vancouversun. Retrieved October 9, 2021, from https://vancouversun.com/business/local-business/expect-higher-prices-on-menus-restaurateur-warns-as-operating-costs-mount.

Canada's food price report 2021. Dalhousie University. (n.d.). Retrieved October 9, 2021, from https://www.dal.ca/sites/agri-food/research/canada-s-food-price-report-2021.html.

Edwards, D. (2021, June 20). *Restaurateurs grapple with rising menu prices, food costs amid covid-19 pandemic - national*. Global News. Retrieved October 9, 2021, from https://globalnews.ca/news/7965723/restaurant-food-costs-covid/.

Naveen Kumar, (2020). Vancouver restaurant Dataset, Version1. Retriveved October 7, 2020 from https://www.kaggle.com/banaveenkumar/vancouver-restaurent-dataset