MGMT-590: Computing for Analytics – FINAL PROJECT REPORT

Sushree Patra | Ruthvik Arepalle | Deeksha Goyal | Vinitha Ravindran | Yuchen Li | Rukmini Sunil Nair

Zomato Bangalore dataset analysis

Introduction

Food is the main wellspring of imperative vitality in our body – it literally makes up every part of us. Eating makes us feel good; it is also a social urge nowadays. Advancements in technology have paved way for various innovations in the consumer services domain - food delivery apps being one among the important ones. *Zomato* (initially *Foodiebay*) is an Indian restaurant aggregator and food delivery startup, founded in 2008 by two IIT Delhi alums. Today, Zomato caters to 24 countries, has over 80 million active users and provides a variety of services such as restaurant search and discovery, online ordering, table reservations and management, POS systems and subscription services.

For our analysis, we decided to use the Zomato Bangalore dataset, which was scraped for educational purposes (https://www.kaggle.com/himanshupoddar/zomato-bangalore-restaurants). The copyrights are owned by Zomato Media Pvt. Ltd. The dataset contains information that one can find on the app – ratings, reviews and addresses of restaurants, most popular dishes and approximate cost of dining.

Bangalore is the IT capital of India – a significant portion of the population relies on restaurant food, due to lack of time to cook daily. There are over 12,000 restaurants in Bangalore, serving a multitude of cuisines. The main goal of our analysis is to gather latent insights from data that are easy for consumers and restaurant owners to understand and make better and informed choices. We built a variety of functionalities –

- Visual guide to gain insight about desirable restaurants based on constrains such as location, budget votes and rating
- An interactive guide which lets users select the location and restaurant of choice, and view a word cloud of reviews to gauge how good or bad the restaurant might be
- Estimating the probability of a customer rating a restaurant on the app after they visit, using Monte Carlo simulations
- Finding the optimal combination of restaurants in a desired location, aimed at maximizing dine outs in a month, based on ratings and cost of dining, given constraints such as budget (a variation of the Knapsack Problem). Also compare time taken by both algorithms
- Predicting rating using multiple linear regression – based on online ordering, table booking and cost of dining of two people
- Heatmap of number of restaurants by extracting latitudes and longitudes

- Predicting the rating of a restaurant based on attributes such as cost of dining and votes using Ordinary Least Squares method and regression using numpy

We also compared time complexities while implementing some of these functionalities -

- Comparison of time complexity while searching for a cuisine using various types of data structures
- Comparison of time taken for predicting the rating of a restaurant based on attributes such as cost of dining and votes using Ordinary Least Squares method and regression using numpy

Computational Steps

- Visual guide to gain insight about desirable restaurants based on constrains such as location, budget votes and rating
    - Data cleaning (remove NAs etc.), clean ratings column by fetch proper values
    - Create getLocationWiseActiveRestaurants function returns restaurants after grouping the data by location and sort restaurants on the basis of top vote count
    - Create getTopNRestsWithVotesCount to get average rating per restaurant
    - Save the data to plot in a dataframe and plot graph
    - Understand graph: Size of circle depends on vote (more trusted restaurant), color of circle depicts location it belongs, while x and y are average cost of meal and rating respectively

    Computational challenge: There are approximately 50k records in our dataset from which it's not possible to show all the points on a map. By using functions and sorting methods, only top choices will be mapped which is relevant to a person.

- An interactive guide which lets users select the location and restaurant of choice, and view a word cloud of reviews to gauge how good or bad the restaurant might be
    - Used the “wordcloud” module in python

- Data cleaning
- Provide users available locations to choose from
- After choosing desired location, provide available restaurants to choose from
- Display word cloud of reviews for the selected restaurant – using WordCloud function and matplotlib

- Estimating the probability of a customer rating a restaurant on the app after they visit, using Monte Carlo simulations
  - Data cleaning
  - Sub-setting our data to include only area "Banashankari"
  - Assuming that 20,000 customers have visited the restaurant (arbitrary value)
  - Sub-setting data to include only restaurants with a rating of more than 4.1/5
  - Performing simulations on calculation of number of times the customer has voted (yes/no)
  - Computing probability of voting based on the result of these simulations

  Computational challenge: We assumed an arbitrary value for number of customers who have visited a restaurant. If we had the actual value of this variable, our simulated probability value would have been more accurate.

```
The probability that a person votes for a restaurant in the selected area is 0.06349473684210526
```

- Finding the optimal combination of restaurants in a desired location, aimed at maximizing dine outs in a month, based on ratings and cost of dining, given constraints such as budget (a variation of the Knapsack Problem)
  - Defining a function to perform Greedy algorithm search on a dataset for the knapsack problem
  - Defining a function to perform optimal or Brute Force algorithm search on a dataset for the knapsack problem
  - Data cleaning
  - Define a function to measure time of performing a function on arguments
  - Run both algorithms on our problem (define constraints as budget – assuming a family of two), measure and output the time taken and solution returned by both algorithms

  Computational challenge: Optimal/Brute Force search took a long time to run (since its time complexity is $O(2^n)$).

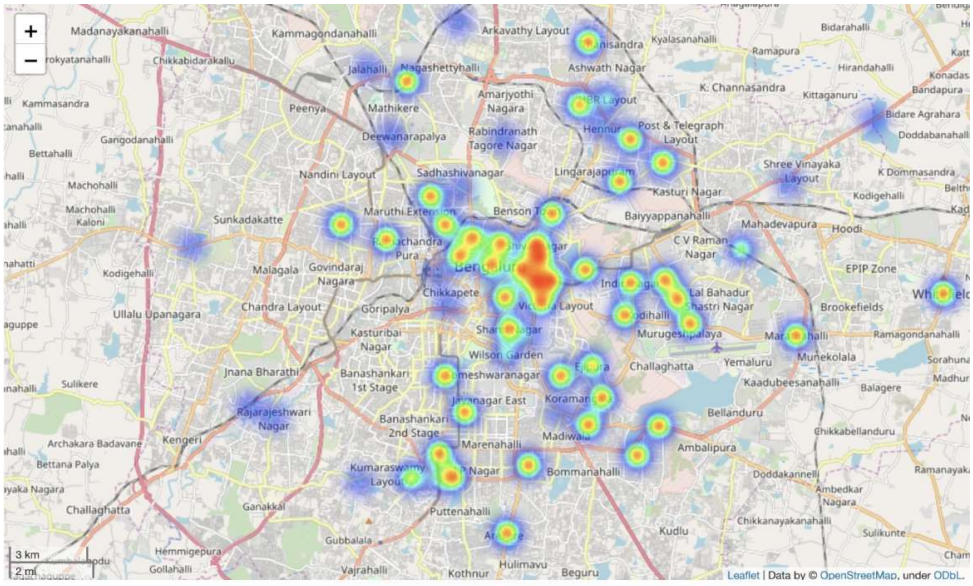
```
The list of restaurants for greedy solution is
 ['1131 Bar + Kitchen', 'The Fatty Bao - Asian Gastro Bar', 'Smoor', 'Smoke House Deli', 'Rim Naam - The Oberoi', 'Na
tural Ice Cream', 'Truffles', 'Apsara Ice Cream', 'Corner House Ice Cream'] ,
Total money spent is 9850.0

 The list of restaurant for optimum solution is
 ['1131 Bar + Kitchen', 'The Fatty Bao - Asian Gastro Bar', 'Smoor', 'Smoke House Deli', 'Rim Naam - The Oberoi', 'Na
tural Ice Cream', 'Truffles', 'Apsara Ice Cream', 'Corner House Ice Cream'] ,
 Total money spent is 9850.0

 The time taken to run greedy algorithm is  0.0032711029052734375

 The time taken to run brute force optimal solution is 99.21206998825073
```

- Predicting rating using multiple linear regression – based on online ordering, table booking and cost of dining of two people
    - Data cleaning, filter for restaurants with more than 10 votes
    - Split data into test and train
    - Convert the categorical variables to dummy variables
    - Use package sklearn to perform multiple linear regression with the mentioned predictors and response variable – fit training set
    - Predict results for the test set

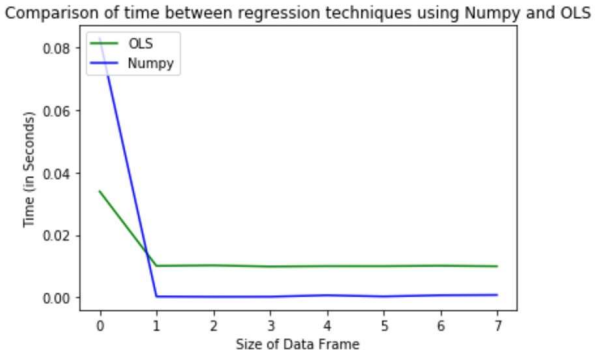
```
Value of R2 is  0.1572518395496979
Coefficient are  [[0.04045076 0.25920981 0.16862776]]
Intercept is  [3.09933592e-16]
```

Regression Equation: Rate = 3.0993e-16 + 0.04045076 * online_order + 0.25920981 * book_table + 0.16852776 * Cost_two_people

- Heatmap of number of restaurants by extracting latitudes and longitudes



- Comparison of time taken for predicting the rating of a restaurant based on attributes such as cost of dining and votes using Ordinary Least Squares method and regression using numpy
    - Data cleaning
    - Apply regression using numpy on the dataset to estimate rating of restaurants using number of votes and approximate cost of dining for two people
    - Apply regression using OLS on the dataset to estimate rating of restaurants using number of votes and approximate cost of dining for two people
    - Compare the times taken by both the regression techniques on different subsets of the dataset



- Comparison of time complexity while searching for a cuisine using various types of data structures
    - Data cleaning
    - Convert data into a dictionary and define a function to search for a given cuisine in the dataset and return index values of the restaurant
    - Define a function to search for a given cuisine in the dataset (data frame) and return index values of the restaurant
    - Define a function to measure time of performing a function on arguments and compare time taken

The most common challenge faced was data cleaning. Since this data was scraped off the web, all variables were character. There was a significant amount of cleaning and formatting required for every variable.

Improvements

The slowest parts of our code were the heatmap and the Brute Force algorithm. Also, we could have improved the simulation probability, had there been a measure of the number of customers who visited a certain restaurant.

If we had additional data on Zomato Gold membership scheme, we could identify how this would affect ratings of restaurants.