Capstone Project Final Report: Predicting Restaurant Success with Demographic Data

Jeffrey Huang

September 20, 2020

Introduction

In my capstone project, I chose to examine the effects demographic data have on restaurant data. The motivation behind the project comes from the intuition that the type of people who live within a geographic region can have a large effect on a restaurant's success. For example, if a restaurant were to open near a university, it would intuitively make more financial sense to offer take-out and fast-food options because of a younger age demographic.

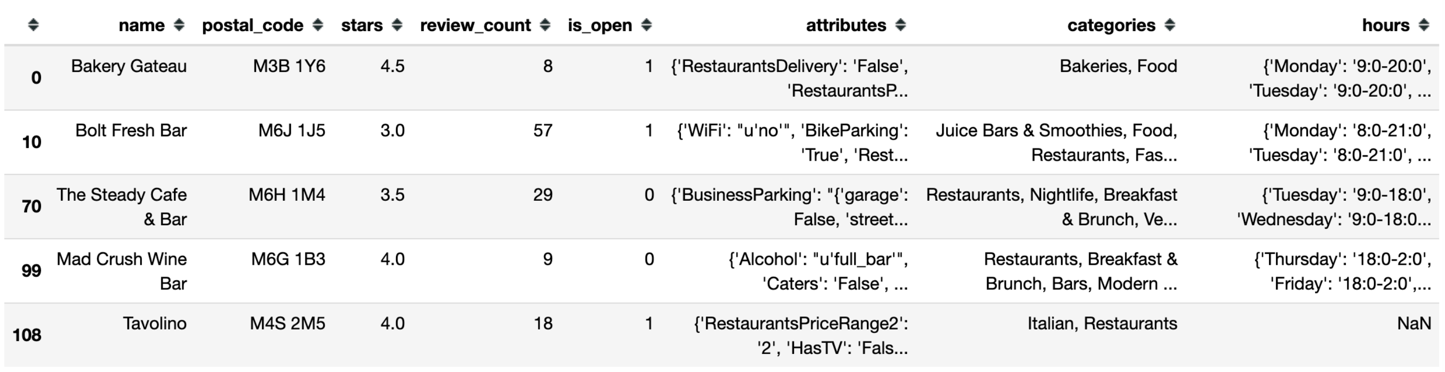
By the end of this project, my goal is to find a model that is adequate enough for predicting restaurant success. I do not expect the test accuracy of my classification model to be high, because there are more factors than just demographic data when determining a restaurant's success.

In this report, I will outline the methodology and findings of my capstone project. I first begin with explaining the data collection processes and the problems I faced during them. I then explain my data cleaning processes and some exploratory data analysis. Lastly, I will present the various models I created and potential future steps of the project.

Data Collection & Cleaning: Yelp Dataset

Data was collected from two sources: restaurant data and demographic data. There were other options in finding restaurant data (Zomato, OpenTable, Kaggle), but Yelp was chosen because it had the most data-points, and it also included postal codes (rather than just latitude and longitude). The inclusion of postal codes offered a pre-determined way to group restaurants together by geographic region, and demographic data can be filtered down to postal codes.

The Yelp dataset comes in a csv file that included not just restaurants, but other services (plumbing, laundromats, hardware stores, etc). Using MySQL, I filtered down the data to exclude review text and image data. From this, I had to decide what location I wanted the data to be filtered down to. I decided to choose Toronto, because I was familiar with the neigborhoods and I carried certain intuitions about the demographic makeup of these neighbourhoods. However, there were a limited number of data-points in a single Canadian city as opposed to that of the United States. If output proves to be successful, I will consider collecting data from American restaurants and modelling success with demographic data and zip codes.



	name	postal_code	stars	review_count	is_open	attributes	categories	hours
0	Bakery Gateau	M3B 1Y6	4.5	8	1	{'RestaurantsDelivery': 'False', 'RestaurantsP...	Bakeries, Food	{'Monday': '9:0-20:0', 'Tuesday': '9:0-20:0', ...
10	Bolt Fresh Bar	M6J 1J5	3.0	57	1	{'WiFi': "u'no'", 'BikeParking': 'True', 'Rest...	Juice Bars & Smoothies, Food, Restaurants, Fas...	{'Monday': '8:0-21:0', 'Tuesday': '8:0-21:0', ...
70	The Steady Cafe & Bar	M6H 1M4	3.5	29	0	{'BusinessParking': "{'garage': False, 'street...	Restaurants, Nightlife, Breakfast & Brunch, Ve...	{'Tuesday': '9:0-18:0', 'Wednesday': '9:0-18:0...
99	Mad Crush Wine Bar	M6G 1B3	4.0	9	0	{'Alcohol': "u'full_bar'", 'Caters': 'False', ...	Restaurants, Breakfast & Brunch, Bars, Modern ...	{'Thursday': '18:0-2:0', 'Friday': '18:0-2:0',...
108	Tavolino	M4S 2M5	4.0	18	1	{'RestaurantsPriceRange2': '2', 'HasTV': 'Fals...	Italian, Restaurants	NaN

Figure 1: The first five rows of the restaurant dataset after filtering the Yelp Dataset down to restaurants in Toronto, Ontario

For the rest of the capstone project (except for data visualizations), I used python in Jupyter Notebooks. The dataset (Figure 1) included 8 columns that represented 10,093 rows. The first step in my data cleaning process was to convert the "postal code" column into FSA values. FSA values are the first three characters of a postal code, a metric that Statistics Canada uses to filter down census data into smaller geographic regions. The next step was to convert the other columns into numeric values. "attributes" and "hours" were columns that contained strings of nested dictionaries. Within these dictionaries, some of the nested dictionaries were strings, while some were not. This proved to be difficult to flatten, but a function was created to expedite this process. Because the values of the nested dictionaries were all values of 'True' or 'False', they were turned into dummy variables with '1' and '0's. "hours" was also a column that is in a nested dictionary. This column was converted to 7 different columns, with each column denoting each day of the week and how many hours it was open that day. For example, if a restaurant was open from 9:00AM to 20:00PM on Mondays, then the "hours:Monday" column had a value of 11. After this transformation, the 7 columns were averaged to create a column that depicted the average number of hours a business is open a week. Lastly, the column "categories" proved to be the most difficult to clean. There were a total of 4,734 different categories, with many duplicated. For example, the categories "Mexican, Restaurant" and "Restaurant, Mexican" denoted the same type of restaurant. The method I chose to clean this column was to hardcode 15 different categories. These categories had an emphasis on demographic data - instead of labelling Thai restaurants as such, I labelled them as "South East Asian".

The possible metrics to measure a restaurant's success (response variable) was heavily skewed. There were 5,151 restaurants that were opened, as opposed to the 2,051 restaurants that were closed. The star ratings were also heavily skewed, with a majority of the data around 3.5 to 4.0 stars

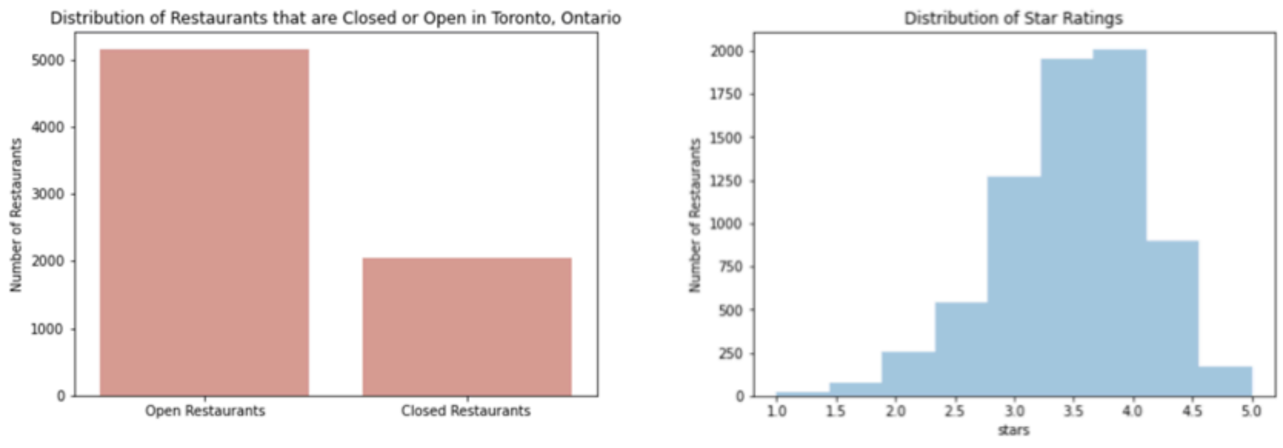


Figure 2: The skewed distributions of potential response variables

The solution to this was to create my own binary variable. To do this, I classified any restaurant as "successful" if it scored a star rating higher than 3.5 stars. Otherwise, if a restaurant was closed or scored less than 3.5, it was deemed unsuccessful. This classification yielded me 49% of the restaurants as successful, and 51% as unsuccessful.

Lastly, I had to deal with the missing values of the columns. There was quite a lot of missing data, because a lot of restaurants did not publish certain facts on their page. For example, the column "take out" was more often filled out, but columns such as "ambiance:Romantic" was not. As a result, columns that had a large amount of missing values were removed. The resulting dataset had 7,202 rows and 18 columns.

Data Collection & Cleaning: Statistics Canada Dataset

I gathered the census data of Toronto using Statistics Canada API. The API returned different datasets depending on the parameter inputted. The measures can come in counts, or percentages. However, this proved to be arbitrary because the data would be scaled when modelling. I chose to collect data from the following datasets: "Visible Minorities Data", "Population Data", "Income Data", "Commute Data", "Housing Data". The data comes in a JSON format, and when converted into a pandas data frame, comes with two columns: "Column Name" and "Data". I wanted the data to have a row represent a FSA value, but the API would only return a whole dataset for every FSA value. However, after transforming the data, I managed to combine all datasets to the appropriate FSA values.

Before modelling, I manipulated several features. I first turned the FSA values and the categories into dummy variables, which greatly increased the number of features in my modelling data. I also added a "Restaurant Density" column that represented the number of restaurants within a single FSA value. Lastly, after removing non-numeric data, I began implementing different models.

Modelling

Before splitting the data into training and testing data, I removed any features that had a higher correlation value of 0.8. This removed 12 features out of 148 total features. I considered using PCA to remove the number of features, but this proved to be ineffective and actually detrimental to my models. Even with 20 principle components, it only explained around 5% of the variance. I carried on with splitting my data with a 70/30 train/test split, and scaled the data using a standard scaler.

The first model I chose was a logistic regression. This yielded me a training score of 64.6% and a testing score of 63.6%. Decision trees (`max_depth = 6`) yielded a lower testing score and more overfitting, with 65.9% training accuracy and 61.27% testing accuracy. I believed that a neural network would perform better than my logistic regression and started off with 136 input neurons and one output layer. This was a very overfitting model, with a train accuracy of 70% and 61.2% test accuracy. Adding more layers lead to more overfitting, and using a L2 regularizer was ineffective in preventing overfitting. I also tried inputting drop-out nodes to stop this, but only managed a testing accuracy of 63.5% with signs of overfitting.

The set of models that proved to be most successful were boosting algorithms. Boosting algorithms such as gradient boosting or XGBoosting (Extreme Gradient Boosting) are balanced models that optimize both performance and time-efficiency. Because it creates a model based on weak classifiers, boosting algorithms can control over-fitting and handle missing values well. I optimized the hyper parameters using grid searching, and managed a testing accuracy of 64.5% for gradient boosting and a training accuracy of 72.7%.

Conclusions

Because I had a limited number of data points, the accuracy was low, but not low enough to not make any inferences from predictions. The next steps for my capstone project is to create a dashboard that would output the probability of opening a specific type of restaurant in every FSA value of Toronto.

To improve this project, perhaps using the rest of the Yelp dataset and collecting census from the United States would offer more datapoint and a more complex and accurate model.