School of Information, University of Michigan
Final Project, SI 601 - Data Manipulation
Nishan Bose, nishan@umich.edu

Comparing NYC Restaurants Inspection Results with Yelp Ratings & Reviews

(NYC Open Data and Yelp API)

Motivation

This project aims to explore the correlation between NYC Restaurants Health scores with Yelp Ratings and Reviews. According to The New York City Department of Health and Mental Hygiene, the inspection grades given to restaurants significantly improved the general health conditions of these businesses overtime. The public may or may not be aware of such ratings and violations by these restaurants. Moreover, the Yelp ratings and reviews may or may not have a direct affect on inspection results. The trend in information can be found out by comparing the available data points and how they change overtime.

Is there an improvement in health conditions of a restaurant overtime depending on the reviews and/or ratings on Yelp? Do reviews and ratings drive the upkeep decisions taken by owners of businesses? Answering such questions could also allow us to understand the factors which contribute towards growth of business - two of them in particular inspection scores and yelp ratings.

Data Sources

The project utilises two datasets which are of different formats owned by separate organisations. **One**, NYC Open Data (https://data.cityofnewyork.us/) has published inspection results for restaurants in the New York City region provided by Department of Health and Mental Hygiene (DOHMH). **Two,** Yelp has an API which allows access to ratings and reviews for restaurants across the world. Following is the description in detail:

1. NYC Open Data is updated daily and is available in csv format for restaurants in the NYC region like Manhattan, Brooklyn and Queens.
    - Location:
      https://data.cityofnewyork.us/Health/DOHMH-New-York-City-Restaurant-Inspection-Results/xx67-kt59
    - Format Used:
      CSV file containing information about all the restaurants in NYC region.
    - Important variables:
        - Category
        - Area Location
        - Name
        - Grade Date
        - Grade Score
        - Zipcode
    - Records Used:
      For Manhattan alone there are about 8500 restaurants containing multiple entries for each restaurant. Including Brooklyn and Queens, there are about 18,000 restaurants in total. (Total Lines Processed: 516,727)
    - Time Period:
      NYC data dates back to January, 2013 but is updated on a regular basis

2. Yelp API allows access to all possible restaurants in the NYC regions containing information like location, reviews, ratings(stars), times open, categories etc. The data size cannot be accounted for all the data format is in JSON.
    - Location:
      http://api.yelp.com/v2, https://www.yelp.com/developers/documentation/v2/
    - Format Used:
      JSON through API requests. The request is carried out using a custom python library which allows restaurant name based search
    - Important variables:
        - ID
        - Name
        - Rating
        - Review Counts
        - Neighbourhood
        - Zipcode
    - Records Used:
      About 18,000 API calls were made to Yelp to get the latest information about restaurants
    - Time Period:
      The data used from Yelp is the latest upto-the-minute information for best trend analysis

Data Manipulation Methods

1. NYC Data
    - Data Manipulation:
      The entire 200 mb file is processed to extract key variables. These variables are filtered according to the selected zones. There are multiple data entries for each zone which results in multiple grades for each restaurant. These grades are standardized to match the yelp ratings

    - How did you handle missing, incomplete, or incorrect data?
      The missing data was either replaced or that particular entry ignored. Especially because some of the entries did not have restaurant names (these entries were ignored). Some of the scores for a restaurant was missing and these entries were replaced with ‘zero’ since there are multiple entries for the same restaurant. Even though the inspection was carried out, the grades were not given.

    - How did you perform conversion or processing steps?
      The NYC Data file is a CSV file which was read as a dictionary to extract all the matching areas (Manhattan, Brooklyn and Queens). The average score is then calculated after taking into account the missing data.

2. Yelp Data
   - Data Manipulation:
     The restaurant names was extracted from the main csv file, duplicates removed and API calls were made to YELP. The response was then tuned to only the required pieces of information to match the NYC data.

   - How did you handle missing, incomplete, or incorrect data?
     The challenge in case of YELP came up when the data of all the restaurants was stored locally because it did not match with the NYC restaurant data. Some of the entries were ignored as they were not present on YELP. This is close to 3% of the total restaurants extracted form the Manhattan, Queens and Brooklyn area.

   - How did you perform conversion or processing steps?
     As the request used additional libraries, the response was in dictionary format. This format allowed easy access to information which is part of the response. The associated values were then extracted and stored in a list format.

General

- What variables and steps did you use to join the two data resources to perform your data analysis?
  Used following code to extract information from the NYC Data files(200 MB) and store it into a list which is accessed later below:

```
with open('inspect.csv') as csvfile:
  reader = csv.DictReader(csvfile)
  for row in reader:
      if row['BORO'] == 'MANHATTAN' or row['BORO'] == 'BROOKLYN' or
row['BORO'] == 'QUEENS':
          if row['DBA'].strip():
              name = row['DBA']
          else:
              continue
          #grade = row['GRADE']
          score = row['SCORE']
          zipcode = row['ZIPCODE']
          area = row['BORO']
          gradedate = row['GRADE DATE']
          newlist.append([name, score, gradedate, zipcode, area])

data = defaultdict(list)

for row in newlist:
```

```
    n = row[0]
    s = row[1].strip()
    if len(s) == 0:
        s1 = 0
    else:
        s1 = int(s)
    data[n].append(s1)

avglist = defaultdict(list)

for k, v in data.iteritems():
    avglist[k] = (sum(v) / float(len(v)))
```

The following code is used to combine both the results and create a text file which can store the information as a list. This include the nyc inspection scores.

```
for k, v in avglist.items():
    try:
        search_results = yelp_api.search_query(limit=1, term=k, location='NYC')
        a = search_results['businesses']
        id = a[0]['id']
        name = a[0]['name']
        rating = a[0]['rating']
        rev_count = a[0]['review_count']
        zipcode = a[0]['location']['postal_code']
        nyc_rating = avglist[k]
        test = [id, name, rating, rev_count, zipcode, nyc_rating]
        OUT.write("%s\n" % test)
    except:
        continue
```

The primary variables used were: search_results, test, newlist and data(default dict)

- Briefly describe the workflow of your source code and what the main parts do.
First python file - The NYC data file is read, the restaurant names, scores, locations and grade date is extracted. This data is then stored in a dictionary where the average score is calulated for all the grades for the same restaurant. This is stored in a dictionary. For all keys in this dictionary an API call is made. The value for each restaurant which is the average of all available grades is then stored into a list. This list is stored into a text file which combines NYC and Yelp data.

Second python file - This file is used to convert the newly created text file into a csv file which can then be used for plotting a scatter chart.

- What challenges did you encounter and how did you solve them?
  There were multiple challenges encountered especially with missing or invalid data. This was handled by using if conditional statements throughout the application. Another major issue encountered was that CSV reader/writer is not unicode compliant by default so all the input and output had to be sanitized to get the right results. The restaurant names in the NYC data file and Yelp API was different by the slightest of margins which was handled using an external library. To be able to get the latest data, most recent API data had to be created. This data took the longest amount of time to retrieve. In short following challenges were faced:
    1. Missing grade scores
    2. Missing grading dates
    3. Unicode characters
    4. Missing names

Analysis and Visualisation

The ratings from the NYC open data platform was averaged across the multiple duplicate entries to account for a similar single number approach for yelp. Yelp has only the current rating number and does not contain historical data. This allowed the affordance of using the same number of data points between the two. The Yelp response for each of these restaurants contained the review counts which was left out of this analysis since normalisation required diving deeper into the behavior. To find the correlation between the inspection score and ratings, a scatter plot is created for each restaurant. This allows the plotting of a coefficient line. This line shows how these two data points are related to each other.

The following piece of code helped look for missing names:

```
if row['DBA'].strip():
            name = row['DBA']
        else:
            continue
```

The following piece of code helped look for missing values:

```
 n = row[0]
   s = row[1].strip()
   if len(s) == 0:
      s1 = 0
   else:
      s1 = int(s)
```

The following piece of code helped in combining the averages.

```
for k, v in data.iteritems():
        avglist[k] = (sum(v) / float(len(v)))
```

The following code was used for creating the input for the plot.

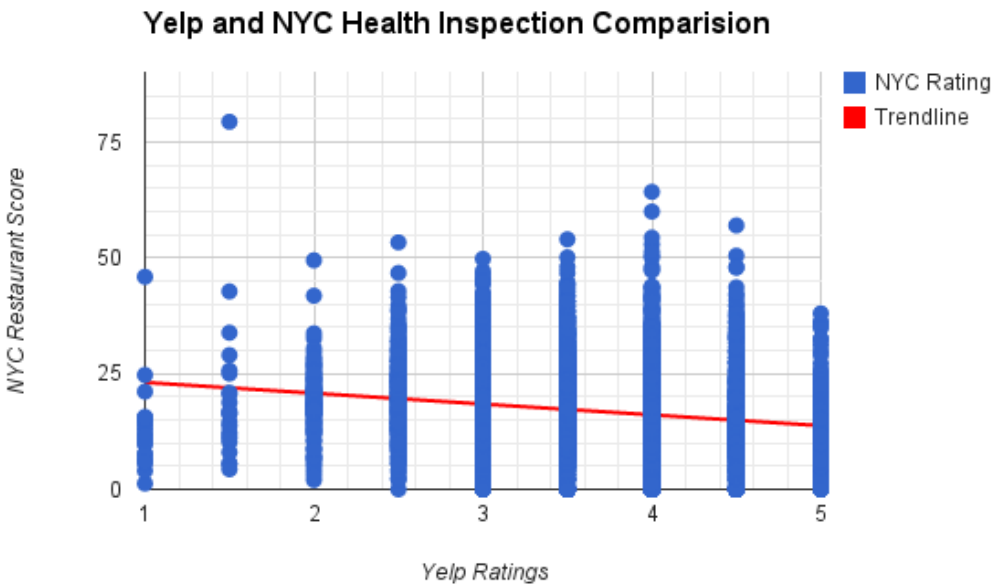
```
nycfile = open(r'combined_results.txt', 'rU')
unpack_list = (literal_eval(s) for s in nycfile)
for item in unpack_list:
  newlist.append((item[0],item[2],item[5]))
```

The key finding of the analysis was that the restaurants with higher yelp ratings had considerably lower inspection scores (lower score is better) compared to the restaurants which had higher inspection scores (higher means health violation) and lower yelp ratings.

The trendline shows how the score decreases depending on the increasing rating of the restaurants. But the correlation is not very strong. This could be because of certain reasons like cost of a place or the fact that reviewers/raters on Yelp are incentivised. Also, one can notice the outliers which do not stick to the trend meaning they are rated high but still suffer health hazards.

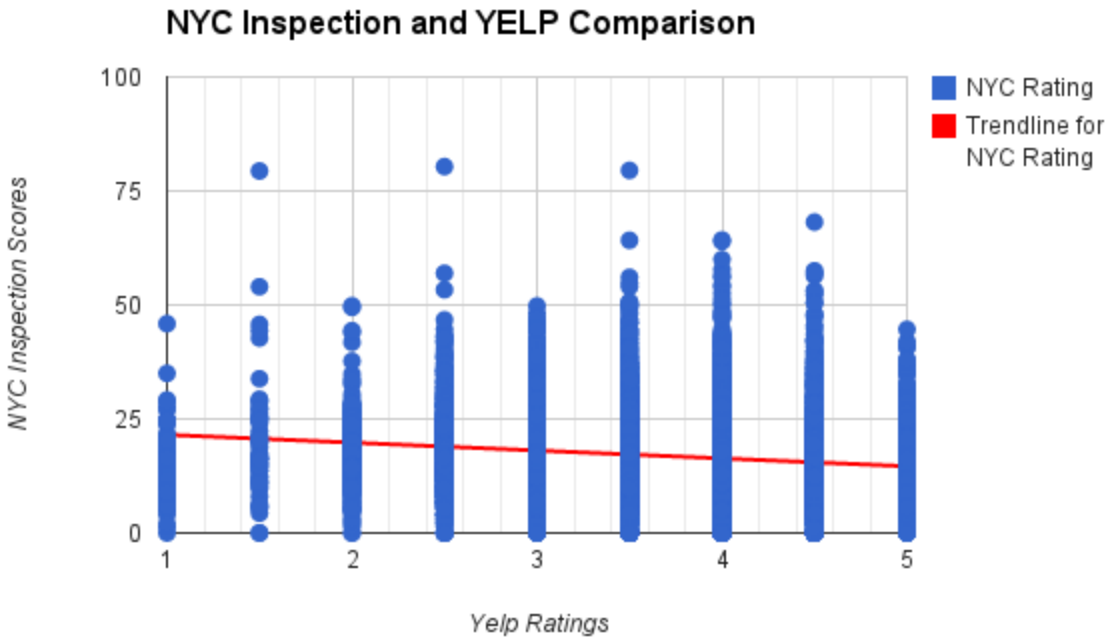
One of the possibilities I was exploring was to use the yelp review count and factor it with the yelp ratings but this resulted in very different results. I also wanted to categorize the results according to the zipcode which proved to be difficult as the time taken to execute the program every time was time consuming. This could then be plotted on a map to generate greater insight as to where the health problems are more common.

NYC - Manhattan Only Plot Data:



School of Information, University of Michigan
Final Project, SI 601 - Data Manipulation
Nishan Bose, nishan@umich.edu

NYC - Manhattan, Queens, Brooklyn Plot Data:



Miscellaneous Details:
Total Restaurants in final output: 17035
Number of Error Restaurants: 1131

Install YelpApi by Geoffrey Fairchild
Github Link: https://github.com/gfairchild/yelpapi
To install: pip install yelpapi