The Restaurant Bubble

Utilizing Yelp API to understand restaurant closings in Los Angeles

***Restaurant Industry Overview***

In the US, the industry employees 14.7M people (10% of the workforce). There are over 1M restaurant locations in the US, with 7% just in California.

Margins for restaurants are slim. With rising labor costs and rent as well as market saturation, restauranteurs need to take a deeper dive into analytics to make smarter decisions regarding new ventures. The entrance of ‘disruptors’ in the industry (like DoorDash, UberEATs) will continue to threaten the industry.

***Executive Summary***

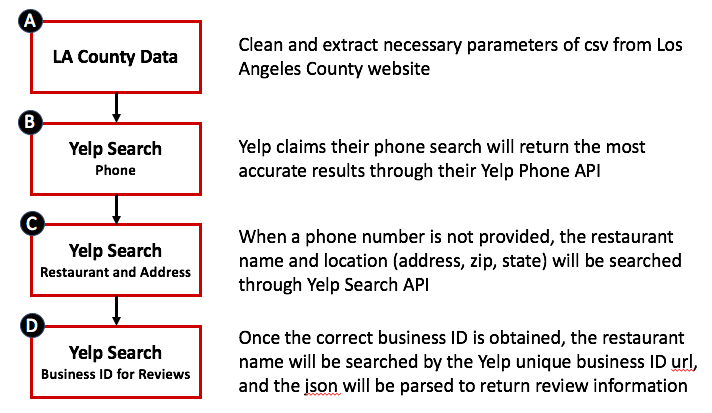
Given the changing landscape, it is important to remove as many barriers to profit as possible for a restaurant. In the prediction of restaurant closings, the data gathered from LA County and Yelp will also allow for advanced analytics in the LA restaurant scene for a deeper understanding of the industry today. Yelp API presented unique roadblocks that did not allow for sufficient modeling and model tuning time, but the processes set forth are valuable and repeatable if additional data is acquired.

***Data Acquisition***

*Reference notebooks 1.01, 1.02, 2.01, 2.02*

There are two datasets that I will be working with:

1. LA County Dataset (A)
2. Yelp Dataset (B, C, D)



**LA County**

*Issues*

The most challenging aspect of this problem was data acquisition. Yelp considers data to be clean in a different manner than someone would expect.

* Suite or Store Number: A pound sign will throw off our ‘best match’ Yelp results when we use their API. Regex will help solve some of these issues.
* Multiple Inspections: One restaurant can be inspected multiple times, or have multiple inspections for more than one kitchen. Groupbys and dictionaries will help solve some of these issues.
* Misspellings: A restaurant may have multiple inspections where the name is misspelled. There are some solutions we can implement, but this will be part of the error.
* Non-restaurant inspections: entertainment/sporting venues, airports, grocery stores, are all included in the LA county data. Comprehensive dictionaries can help solve these issues, if they can be identified.

The cleaner the data is, the more successful our Yelp API calls will be.

**Yelp**

*Process*

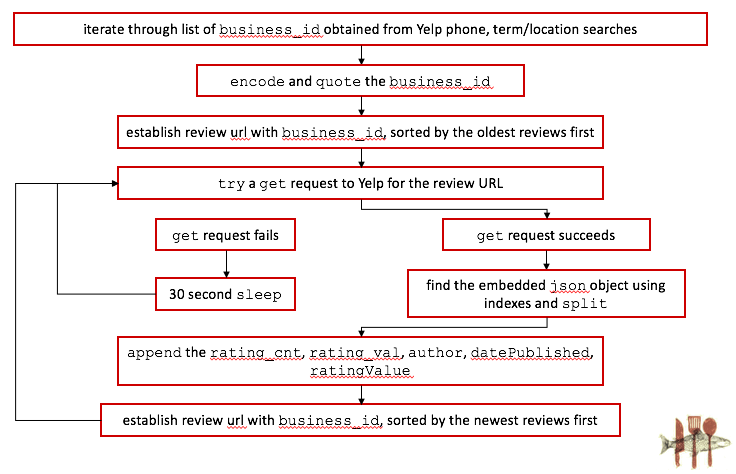
1. Create searches: yp.create\_searches
2. Make get requests: yp.yelp\_api\_calls
3. Track searches that failed: yp.find\_failed\_searches
4. Format the get requests to maintain relevant information: yp.format\_srch\_dict
5. Make sure our search matches what Yelp returned: yp.pick\_correct\_matches

*Issues*

Two main issues with Yelp API:

1. It’s limited—you can only gather so much information. If you find a new way to use the Yelp API (possibly, in an unintended manner), they will attempt to kick you out.
2. Yelp operates on a ‘best match’ model, where the reasons for the output when using the API requests being the ‘best’ are not clear.

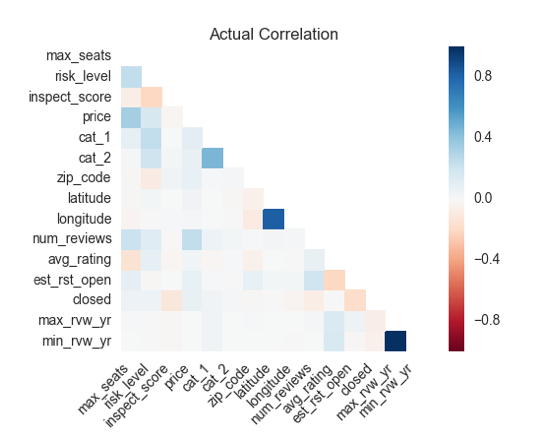
To solve our first problem, I created a function that has sleep time built in:



The second problem is not readily solved, and will be worked on at a later date.

***Data Mining***

After we have gathered all data and view the correlations, it is clear that there is no unexpected correlation between our parameters, indicating we won’t need to consolidate any of our dimensions. Also for that reason, it might be a good idea to use a Random Forest.

******

***Data Modeling***

Our random forest performs quite well, but considering 80% of the restaurants are open, our benchmark needs to perform better than 80%. Thankfully, it does over a maximum tree depth of 4. However, given the scope of the data, we aren’t quite confident in the model to represent a subset of the population when Yelp returned those ‘best matches’.



***Additional Work***

1. The first thing I want to do is gather the entire scope of Los Angeles from Yelp. This will be an arduous task.
2. Some ‘easier’ things I can do:
   1. Divide the data into franchised and unfranchised
   2. Separate cultural neighborhoods
   3. Bring rent data into play