Examining Restaurant Closures using Yelp Data

Vishwa Bhuta, Emily Graves, Ryan Maas, Caroline Nelson, Jon Zeller

Agenda

Business Problem & Objectives

Data

Preliminary Analysis

Machine Learning

Insights & Conclusions

Business Problem & Objectives

Social Data Matters! Right?

- Word-of-mouth plays a major role in success
- Can we identify patterns in social data (Yelp reviews) of restaurants that have closed?
- Can we use these patterns to predict closure among open *and* closed restaurants?

Data

Sources and Collection

- Created Python scraper to pull restaurant reviews from Yelp
  - Per review: reviewer, rating, date, text of review
  - Per restaurant: cuisine, price, rating distribution
- 95 closed restaurants, 13k+ reviews
- 100 open restaurants, 36k+ reviews
- Geographic diversity: Austin, TX; Houston, TX; Charlottesville, VA; Chicago, IL

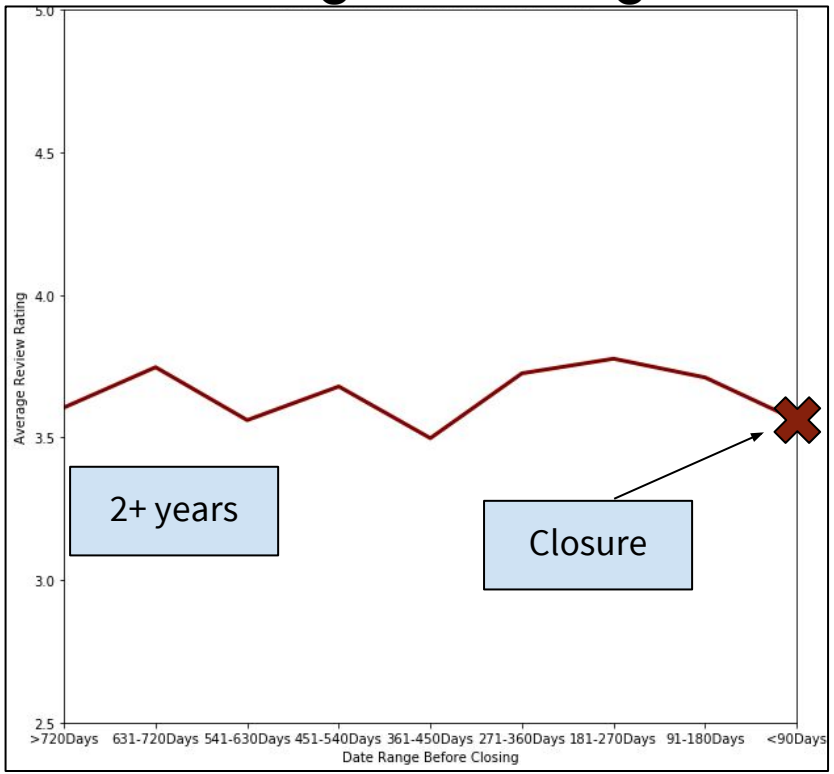
Data Challenges

- Restaurant opening and closing is “noisy”
  - Closure frequently occurs due to non-quality reasons (low profitability, retirement/sale, rent increase, etc.)
  - Not all closed reviews are negative, and some open restaurants may be on the verge of closing
- Reviews are inherently biased towards the extremes

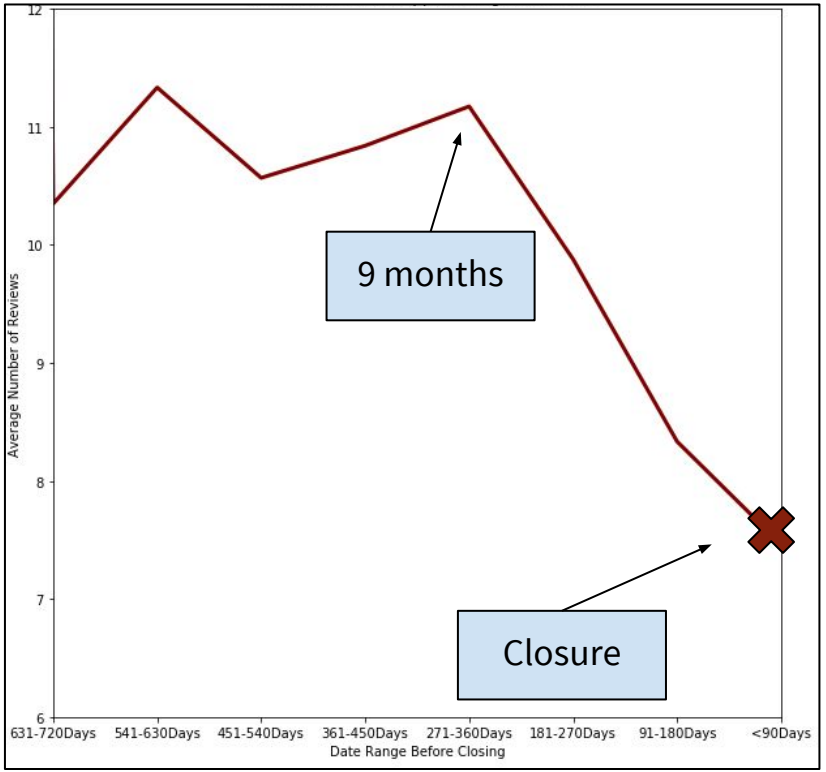
Preliminary Analysis

Do Review Score or Volume Decreases Indicate Closing?

Average Star Rating



Total Number of Reviews



Machine Learning

Word2Vec Overview

- NLP algorithm turns words into 100-dimension vectors
- Learns contexts and meanings

```
model.similar_by_word('good')
```

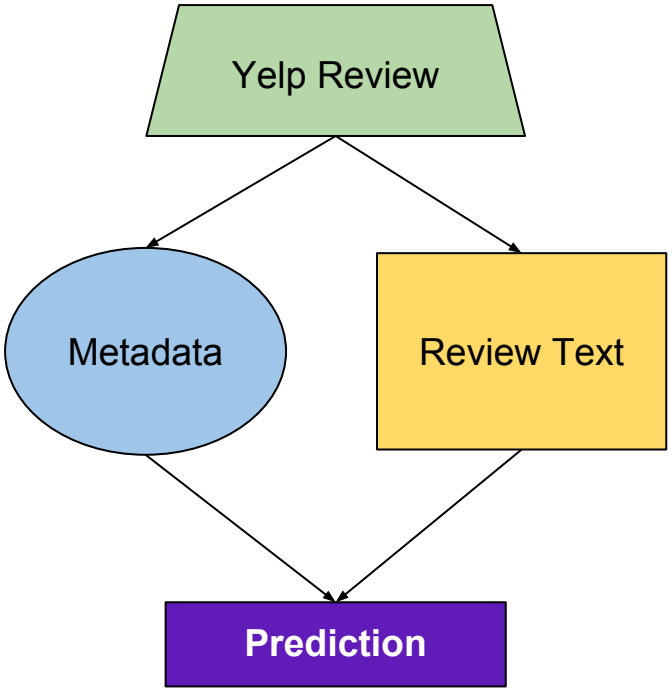
```
[('decent', 0.7752797603607178),
 ('solid', 0.7187694311141968),
 ('tasty', 0.7019286751747131),
 ('awesome', 0.6984829902648926),
 ('great', 0.6908780336380005),
 ('amazing', 0.675799548625946),
 ('fantastic', 0.6629873514175415),
 ('yummy', 0.6212614178657532),
 ('excellent', 0.6172307729721069),
 ('ok', 0.6032919883728027)]
```

```
model.similar_by_word('bad')
```

```
[('terrible', 0.7306864857673645),
 ('awful', 0.6803663969039917),
 ('complain', 0.6493364572525024),
 ('horrible', 0.6391400098800659),
 ('ok', 0.6282997131347656),
 ('okay', 0.626186728477478),
 ('negative', 0.5936901569366455),
 ('alright', 0.5831993818283081),
 ('disappointing', 0.5752206444740295),
 ('worse', 0.5739321708679199)]
```

Closed Restaurant Prediction

- Create word2vec review vectors by aggregating individual word vectors
- Incorporate metadata and word2vec representations of reviews to predict restaurant closure
- Do similar review patterns indicate restaurant closure (or not)?



Model Results - Accuracy

	Logistic Regression	Random Forest
Baseline	73.4%	
Review Vectors	75.5%	76.2%
Metadata	86.2%	93.8%
Review Vectors + Metadata	94.9%	97.3%

What Matters Overall?

Variable	Desirability	Importance
% 5-star reviews		
% 3-star reviews		
w2v dim. #91		
w2v dim. #92		
% 4-star reviews		
w2v dim. #22		
w2v dim. #89		

- Most important variables were high reviews and a few word2vec dimensions
- Higher proportions of 5-star reviews are good, but 3-star reviews are bad

What Matters in Review Content?

Dimension 92 **+**

- Desirable words (high):
  - ‘Perfection’, ‘Properly’, ‘Perfectly’, ‘Goodness’, ‘Octopus’, ‘Steak’, ‘Fresh’, ‘Juicy’…
- What could it mean?
  - Execution
  - Gourmet

Dimension 89 **-**

- Desirable words (low):
  - ‘Escargot’, ‘Shellfish’, ‘Calamari’, ‘Deviled’, ‘Oysters’, ‘Licorice’ …
- What could it mean?
  - Seafood
  - Style of food

Insights & Conclusions

Business Value

If you’re a restaurant-owner… pay attention to:

- Number of reviews across buckets of months
- Trends in review classification

Next Steps

- Collect more information regarding restaurant closures
  - Geographic/Location Data
  - Local Economic Data
- Increase sample size by collecting data from more restaurants and cities
- Compare restaurants by similar location and cuisine/price

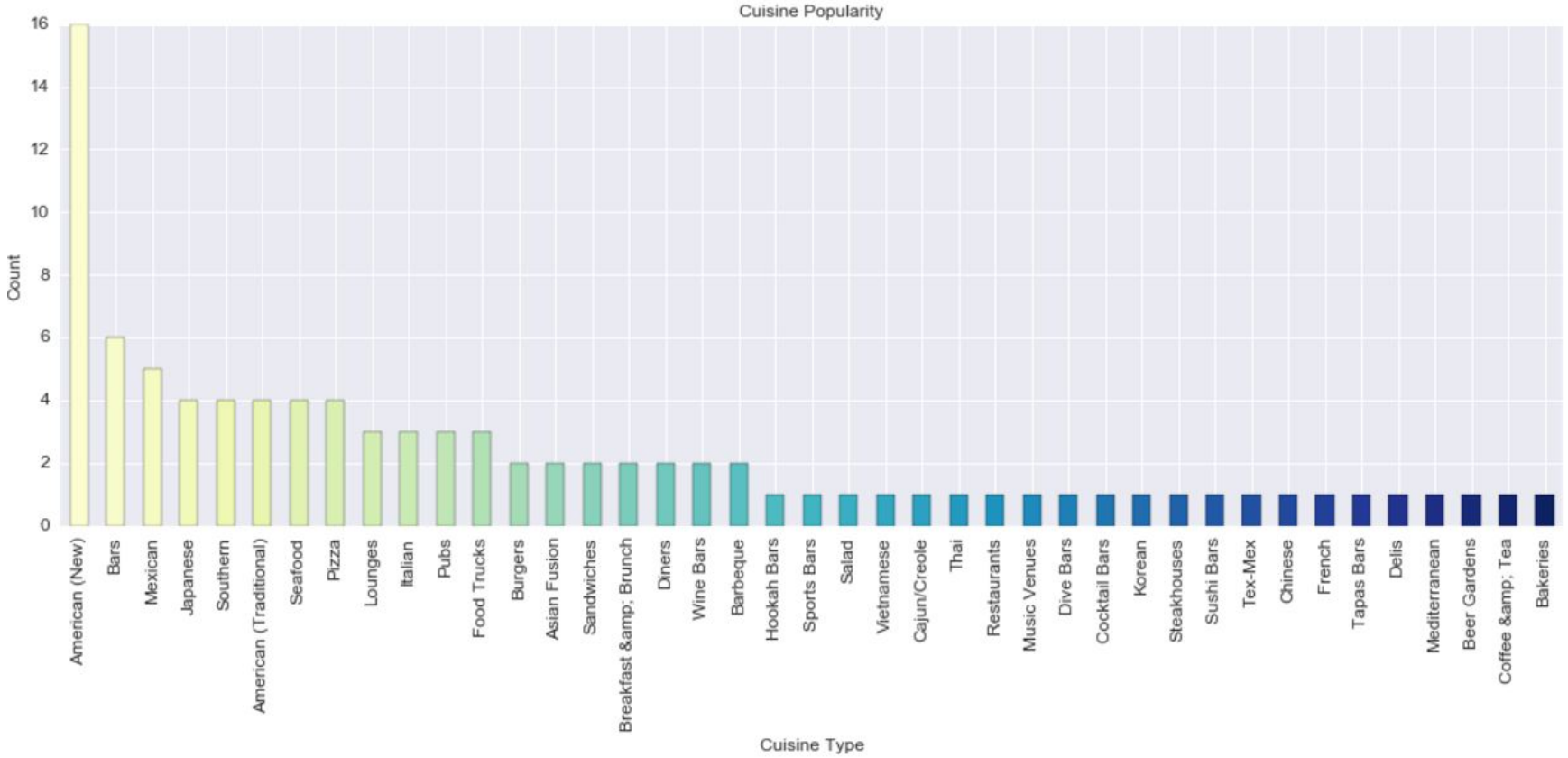
Questions?

Appendix

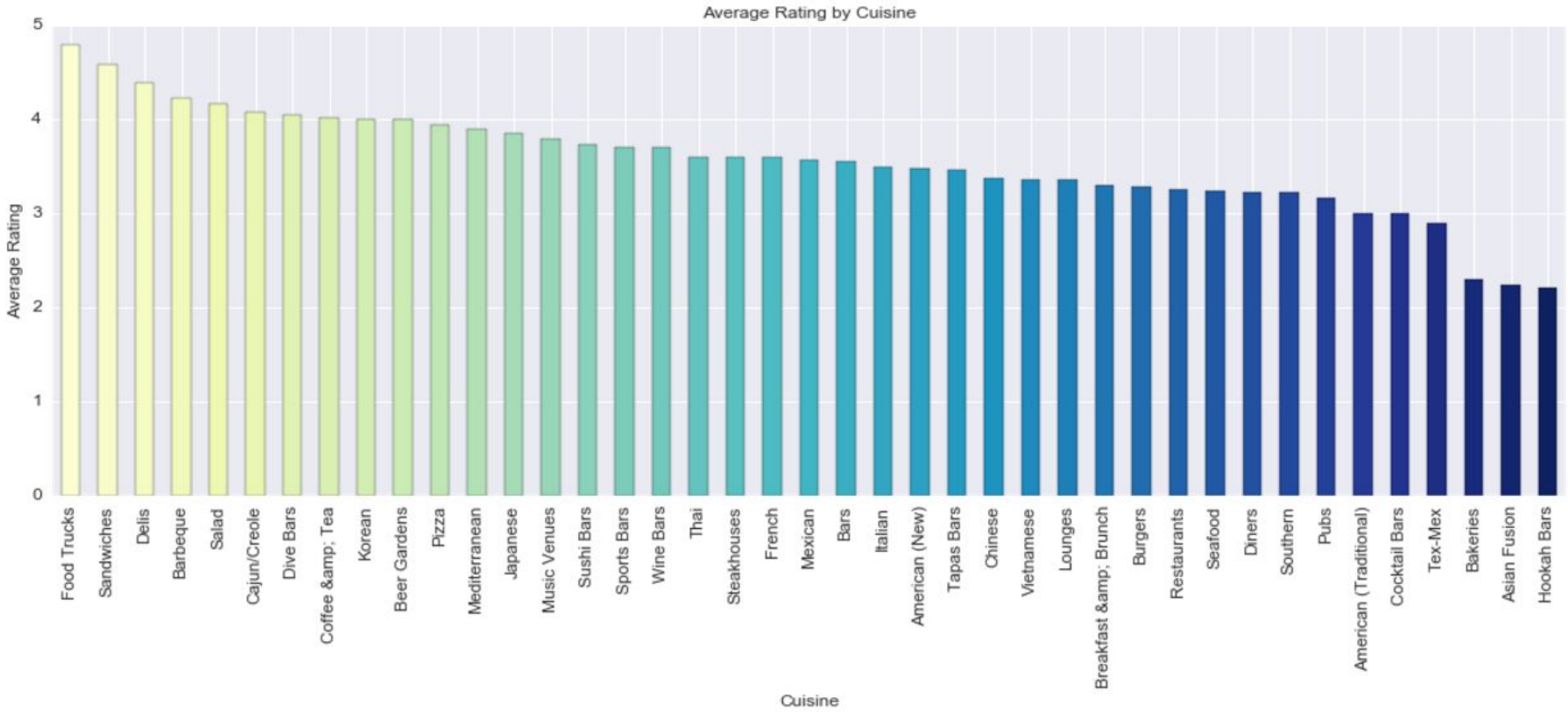
Restaurant Industry Background

- Long-tailed industry - many cuisines, price points, locations...
- Razor-thin margins
- Frequent turnover
- Lots of “Mom & Pop” establishments

Popularity of Cuisines - Closed Restaurants

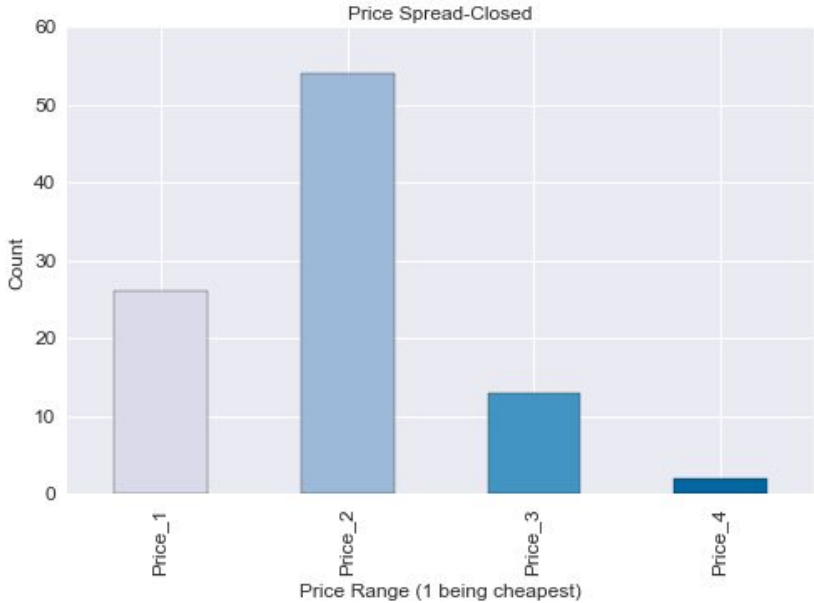


Average Rating by Cuisine - Closed Restaurants



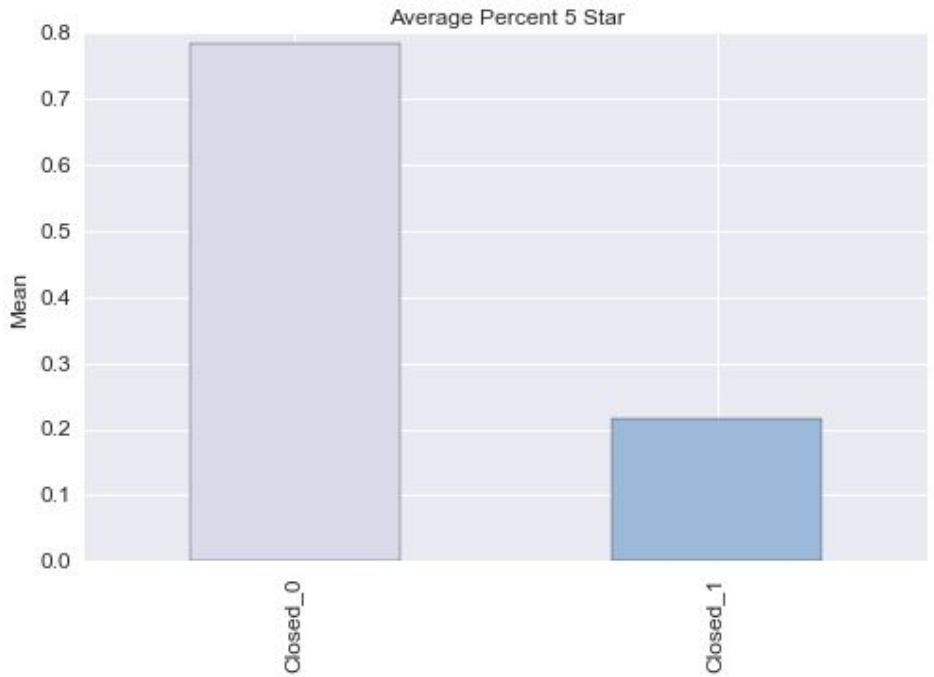
Price Distribution: Open vs. Closed

Are the bad reviews biased towards low quality/price? No!



Average Percent 5 Star - Open vs. Closed

Closed restaurants had some 5 star ratings, but overall, this proved to be a significant feature.



What Matters in Review Content?

Dimension 22

- What scored high:
  - ‘Everyone’, ‘Funny’, ‘Cared’, ‘Worked’, ‘Everything’, ‘Genuinely’, ‘Honest’…
- What could it mean?
  - Overall experience
  - Human interaction

Dimension 91

- What scored high:
  - ‘Favor’, ‘Wanting’, ‘Everytime’, ‘Organics’
  - ‘Interior’, ‘Intimate’, ‘Character’, ‘Lighting’, ‘Rustic’, ‘Chandeliers’, ‘Industrial’…
- What could it mean?
  - Ambiance
  - Decor

Python Screenshots 1 - Raw Scraped Data

In [3]:
```
# Read data
reviews = pd.read_csv('yelp_reviews_final.csv')
reviews['Review_Date'] = pd.to_datetime(reviews['Review_Date'])
reviews.head()
```

Out[3]:

	Rest_ID	Name	Review_Date	Name.1	Stars	Review
0	1	100 Pizzitas	2016-07-28	Eric G.	5	The owners of 100 Pizzitas were very receptive...
1	1	100 Pizzitas	2016-01-03	Dan K.	3	100 pizzitas is like tapas meets pizza. \n\nTh...
2	1	100 Pizzitas	2016-03-09	Lee Roy C.	4	This place has a great outdoor dining space wi...
3	1	100 Pizzitas	2016-05-28	Mj M.	5	This place is great! They had an EDM DJ today ...
4	1	100 Pizzitas	2016-04-02	Vicky N.	5	Let me start this w AMAZING SERVICE, simply am...

Python Screenshots 2 - Metadata

In [328]:
```
reviews_all_meta.head()
```

Out[328]:

	Rest_ID	Name	Cuisine	Price	Num_Reviews	Num_5_Star	Num_4_Star	Num_3_Star	Num_2_Star	Num_1_Star	%_5_Star	%_4_Star	%_3_S
0	1	100 Pizzitas	Pizza	$$	74	41	12	8	6	7	0.554054	0.162162	0.1081
1	2	40 North	Food Trucks	$$	82	59	18	5	0	0	0.719512	0.219512	0.0609
2	3	416 Bar & Grille	American (New)	$$	81	32	14	12	17	6	0.395062	0.172840	0.1481
3	4	Al Fico	Italian	$$$	36	17	10	7	2	0	0.472222	0.277778	0.1944
4	5	Bacon	American (New)	$$	545	185	172	87	59	42	0.339450	0.315596	0.1596

Python Screenshots 3 - Yelp Review Scraper

```
def run_yelp_crawler(url_inputs,name_inputs,output_file):
    # initialize output file
    review_file = open(output_file, 'wb')
    line = csv.writer(review_file)

    num = url_list.index[0]

    for url in url_inputs:

        num += 1

        # make some initial soup
        #soup = BeautifulSoup(urllib2.urlopen(url).read(), "lxml")

        try:
            page = urllib2.urlopen(url).read()
            soup = BeautifulSoup(page, "lxml")
        except httplib.IncompleteRead, e:  # handles the IncompleteRead exception
            page = e.partial
            soup = BeautifulSoup(page, "lxml")

        # get review counts
        reviewcounts = list(soup.findAll('span',{'itemprop':'reviewCount'}))

        # get reviews, ratings, and dates
        ratings = soup.findAll('meta',{'itemprop' : 'ratingValue'})
        ratings = list(ratings)[1:-1] # first and last ratings are aggregate
        reviews = soup.findAll('p',{'itemprop' : 'description'})
        dates = soup.findAll('meta',{'itemprop' : 'datePublished'})
        authors = soup.findAll('meta',{'itemprop' : 'author'})

        # get number of reviews
        reviewcount = int(max(map(lambda x: re.findall(r"[\w']+", str(x))[3], reviewcounts)))

        # get the first 20 reviews and write to file
        for x in range(0, len(ratings)):
            try:
                line_data = [num, name_inputs[num-1],
                             str(dates[x]).split('"')[1],
                             str(authors[x]).split('"')[1],
                             str(ratings[x]).split('"')[1], # get_text pulls just the text
                             reviews[x].get_text().lstrip().rstrip()] # pull just the text and clean it up

                #line.writerow([unicode(s).encode("utf-8") for s in line_data]) # does some transforming and writes it to file
                line.writerow([to_utf8(from_bytes(s)) for s in line_data])
            except IndexError:
                    continue
```

```
        # go through pages 2 - end and get reviews in increments of 20
        for i in range(1,(reviewcount/20)+1):
            new_url = url+'?start='+str(i*20)
            # make some new soup
            #soup = BeautifulSoup(urllib2.urlopen(new_url).read(), "lxml")

            try:
                page = urllib2.urlopen(new_url).read()
                soup = BeautifulSoup(page, "lxml")
            except httplib.IncompleteRead, e:  # handles the IncompleteRead exception
                page = e.partial
                soup = BeautifulSoup(page, "lxml")

            # get reviews, ratings, and dates
            ratings = soup.findAll('meta',{'itemprop' : 'ratingValue'})
            ratings = list(ratings)[1:21] # first and last ratings are aggregate
            reviews = soup.findAll('p',{'itemprop' : 'description'})
            dates = soup.findAll('meta',{'itemprop' : 'datePublished'})
            authors = soup.findAll('meta',{'itemprop' : 'author'})

            for x in range(0, len(ratings)):
                try:
                    line_data = [num, name_inputs[num-1],
                                 str(dates[x]).split('"')[1],
                                 str(authors[x]).split('"')[1],
                                 str(ratings[x]).split('"')[1], # get_text pulls just the text
                                 reviews[x].get_text().lstrip().rstrip()] # pull just the text and clean it up

                    #line.writerow([unicode(s).encode("utf-8") for s in line_data]) # does some transforming and writes it to fil
                    line.writerow([to_utf8(from_bytes(s)) for s in line_data])
                except IndexError:
                    continue

        print '#' + str(num) + ' ' + name_inputs[num-1] + ' Done!'

    review_file.close()
```

Python Screenshots 4-Yelp Metadata Scraper

```
def metadata_yelp_crawler(url_inputs,name_inputs,output_file):
    # initialize output file
    review_file = open(output_file, 'wb')
    line = csv.writer(review_file)

    num = url_list.index[0]


    for url in url_inputs:
        num += 1

        # make some initial soup
        soup = BeautifulSoup(urllib2.urlopen(url).read(), "lxml")

        # get review counts
        reviewcounts = list(soup.findAll('span',{'itemprop':'reviewCount'}))
        # get number of reviews
        reviewcount = int(max(map(lambda x: re.findall(r"[\w']+", str(x))[3], reviewcounts)))

        # get rating counts, price, cuisine
        counts = []
        ratingcounts = soup.findAll('td',{'class' : 'histogram_count'})
        for i in range(0,len(ratingcounts)):
            counts.append(str(ratingcounts[i]).split('>')[1].split('<')[0])

        prices = soup.findAll('span',{'class' : 'business-attribute price-range'})
        price = str(prices[0]).split('>')[1].split('<')[0]

        titles = soup.findAll('title')
        title = str(titles).split('-')[0].split('>')[1]

        cuisines = soup.findAll('span',{'class' : 'category-str-list'})
        cuisine = str(cuisines[0]).split('>')[2].split('<')[0]

        line_data = [num, name_inputs[num-1],title,cuisine,price,reviewcount,counts[0],counts[1],counts[2],counts[3],counts[4]]
        line.writerow([to_utf8(from_bytes(s)) for s in line_data]) # does some transforming and writes it to file


    review_file.close()
```