Capstone Project Final Report: Restaurant Health

Sarah Khederian, Matthew Steele, Mark Muldoon, David Barnhart and Mary Mallampalli

Georgetown University – School of Continuing Studies

Data Science Certificate Program

Fall 2016

I. Abstract

According to the CDC, one in six Americans contracts a foodborne illness each year.[1] Given that half of all money spent on food is spent in restaurants, according to one 2008 study, many Americans are becoming sick from food they consume in restaurants.[2] Although the health of patrons is becoming increasingly important to state and local governments, health department inspectors are frequently overwhelmed by the large number of restaurants they are responsible for assessing each year. While many health inspections occur at random or whenever agencies have the capacity to assess them, our team proposed to create a data product that could use Yelp's written reviews and star ratings to determine a restaurant's health inspection score. This tool would enhance the efficiency of health inspection departments by providing predictions of restaurants that are likely to have a high number of health code violations. This will help health departments streamline their processes for targeting high-risk establishments, enabling them to better allocate time and resources.

II. Hypothesis

Our group's initial hypothesis was that a restaurant's Yelp reviews (including text reviews, ratings, and features) will be an accurate predictor of a restaurant's health code violations at a given inspection date.

III. Motivation to Prove Hypothesis

If our group were to prove through machine learning that Yelp's star ratings and written reviews were strong predictors of health code violations, there would be many potential uses for this data product. First, our tool could be utilized to help health inspectors more efficiently allocate resources by targeting restaurants that have a strong likelihood of having a high number of health code violations. This could save state and local health departments time and money. Right now, there is no standard schedule for inspecting restaurants. As we saw first hand in the data, some restaurants may be inspected twice in two months, or twice in two years. While this may be fine for restaurants that are up to code, there can be serious consequences for those that aren't. Second, this tool could be made available to the public as an app. Restaurant patrons could use the app to better educate themselves about the health code compliance of establishments they frequent. Third, this tool could be added to the Yelp app/website so that on every restaurant's page, the likelihood of health code compliance is available to viewers. At the most basic level, our group was motivated to prove this hypothesis because doing so would promote public health and transparency.

[1] CDC, "Foodborne Germs and Illnesses," http://www.cdc.gov/foodsafety/foodborne-germs.html
[2] Jones TF. and Grimm K., "Public knowledge and attitudes regarding public health inspections of restaurants," June 2008, http://www.ncbi.nlm.nih.gov/pubmed/18471587

IV. Steps of Analysis

Data Ingestion

We collected data from two sources. The first was Yelp's Academic Dataset,[3] which provided us with restaurant review and star rating data from six U.S. cities: Charlotte, North Carolina; Las Vegas, Nevada; Boston, Massachusetts; Madison, Wisconsin; Pittsburgh, Pennsylvania; and Phoenix, Arizona. Yelp provided this dataset in a deeply nested JSON format. Next, each member of the team contacted health departments from these cities. We were only able to obtain quality data from Charlotte, Las Vegas, and Boston, in CSV format (some of which was in third normal form, already).

One of the earliest challenges we faced was determining what the target variable would be. When we first analyzed the rating scales of different cities we realized that this would be a crucial decision. Since health inspections are carried out by local governments, the rules and regulations surrounding inspections and ratings are unique to each city. We eventually determined that the best target variable available would be a continuous count of the number of violations received at a given inspection. Though more detailed infromation was available from some cities, such as the criticality of the violation, the variance in what was deemed critical across cities differed so widely we realized this would not be a reliable measure. This process established our instance as a single restaurant at the date of a health inspection.

We used two different databases to house the Yelp and health inspection datasets. We kept our original, uncleaned version of both Yelp reviews and business attributes MongoDB, which acted as a WORM store. The raw CSV files that we received either via email or physical mail (usb in a box!) were kept clean on disk and copies used for wrangling were stored in PostgreSQL.

Wrangling

The first step in wrangling our data was standardizing each of the health inspection datasets and creating the normalized target variable. We used Pandas to wrangle each dataset so that there were standard columns in each: Restaurant Name, Full Address, Date of Inspection, Latitude and Longitude, and our dependent variable Number of Violations.

The second step in wrangling was converting the JSON dataset from Yelp into CSV format. Using Mongo we were able to export the documents as a CSV, however the more nested information remained in large dictionaries stored in a single column. First, we tried to write a custom script to grab each of the key:value pairs in the order they appeared. The problem with this approach is that the JSON data did not contain a standardized document format. For each restaurant there was a unique number of nested dictionaries and lists based on what information

[3] "Yelp Dataset Challenge," https://www.yelp.com/dataset_challenge

had been tagged to the restaurant. Our first solution to this problem was to grab each unique key and rebuild the script, but we explored our data further and found over 1000 attributes which would have needed hand coding or a recursive script. This was due to the fact that the bulk data download contained information on all businesses in each city- not just restaurants. Once we realized that we were only interested in a relatively small subset of the attributes, we sat down and created a list of features we wanted and decided to parse them out after we had completed the joins between the Yelp and inspection data.

Joining the data from these disparate sources also proved very challenging. First, we tried to join the health inspection and the Yelp data by finding perfect matches for both restaurant name and address in each dataset. We then tried to join the data based on latitude and longitude matches, but we were unsuccessful due to inconsistent latitude and longitude lengths. Finally, we were able to join data by iteratively matching on substrings of street addresses and restaurant names.

For example, we first joined restaurants in Las Vegas by matching the first 10 characters of the address where the restaurant names were a perfect match. After spot checking this proved to be highly reliable. Second, we reduced the parameters to only the first 4 characters of address, while still keeping the restaurant names equal. This was necessary after we determined that the user inputted data in Yelp was simply too messy. For example, a restaurant on "Massachusetts Avenue" was labeled as "Mass ave" in Yelp. We did this for each city separately. Once we had created the merged datasets for each city, we went through the less restricted joins manually to spot check for and delete imperfect matches. After this, we created one master dataset by unioning the datasets of each of the three cities.

Now we had all our attribute data in one place, it was time to parse out the nested features that we would use to test our hypothesis. To accomplish this task we wrote a script that searched each of the nested dictionaries and returned the value if it was found in the instance. The parsing yielded features such as "Alcohol: True", "Dogs Allowed", "Classy:True", and "Price Range". From there, we used dummy encoding to indicate its occurrence for each feature (see AttributeParsing and CategoryParsing notebooks in Github). For encoding the categorical neighborhood names we decided to use binary encoding for the 68 unique neighborhoods in order to only add 7 columns - Tony provided us with code for this! (see Neighborhoods and Encoding notebooks in Github).

Our last wrangling step dealt with the separate text review data provided by Yelp. In order to use the text in a way consistent with our instance we had to include text only written up to an inspection date, but not before the previous inspection. We did this by creating "from" and "thru" dates for each instance, where "from" was the prior inspection date, and "thru" was the current inspection date. This allowed us to aggregate and create a single corpus of all text reviews for each instance, giving us the most accurate snapshot in time. Once this was accomplished the text could be joined with other features in PostgreSQL.

Analysis

Our attributes dataset included a total of 34,991 instances. Once again we defined our instance as a single restaurant during a specific point in time (the date of a health inspection). The features contained within each instance included the total number of Yelp reviews (a proxy for popularity), Yelp average star rating, the inspection date, total number of violations at the inspection date, neighborhood, city, and 50 categorical restaurant attributes and categories (i.e. “Italian Food”, “Pub”, “Delivery”, “Ambiance”, etc.) We also included two calculated features, previous violation count, and the difference in violations between the previous two inspections. We also derived change in count from the previous inspection date, but this feature was almost perfectly predictive, indicating leakage.

We attempted to identify relationships between our continuous variables and the number of violations a restaurant received using visualizations and general descriptives. We also explored the effect of geography by mapping restaurant locations and violation counts to city maps.

Modeling

Before modeling, we pre-processed the data, trying out three different kinds of data scaling to see which was best performing: scaled with mean 0; scaled between 0 and 1; and normalized. Ultimately the data scaled with mean 0 was best performing. After running our first model we also dropped instances with violation count outliers (defined as more than four standard deviation from the mean) to prevent model skewing. We also performed two different feature transformations in order to achieve better results: Polynomial Feature transformation and PCA dimensionality reduction.

Since we were attempting to predict a continuous variable, number of violations, we knew we would be looking for models in the regression family. We ended up using normal Linear Regression, Ridge, Lasso, ElasticNet, and Random Forest.

With our text review dataset we also wanted to attempt to perform natural language processing using classification. Due to time constraints, we focused on one text vectorization process (count vectorization) and one classifier’s performance (Naive Bayes Multinomial). We then pickled the pipeline including the vectorizer and best fitting classifier,[4] and used it to classify the Yelp review corpus into a new categorical feature, which was encoded as the maximum violation count per bin. This feature was then added to the existing dataset to rerun through our regression models to identify the best fitting one.

[4] Zac Stewart, “Using scikit-learn Pipelines and FeatureUnions,” Accessed January 10, 2017, http://zacstewart.com/2014/08/05/pipelines-of-featureunions-of-pipelines.html

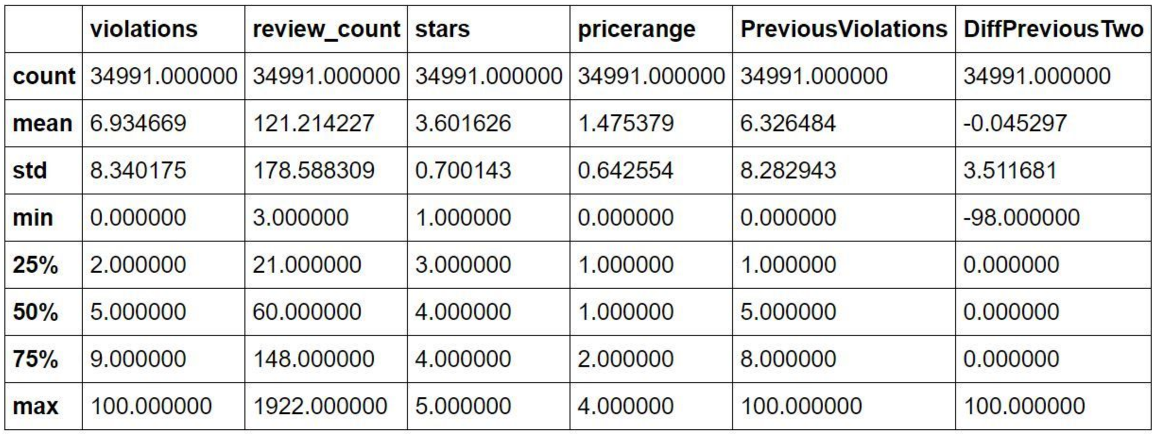
V. Findings Report*

*After re-evaluating our code and re-running some of our notebooks, some of the numbers in this report have changed from our initial presentation, though our overall results remain the same.

Data Analysis

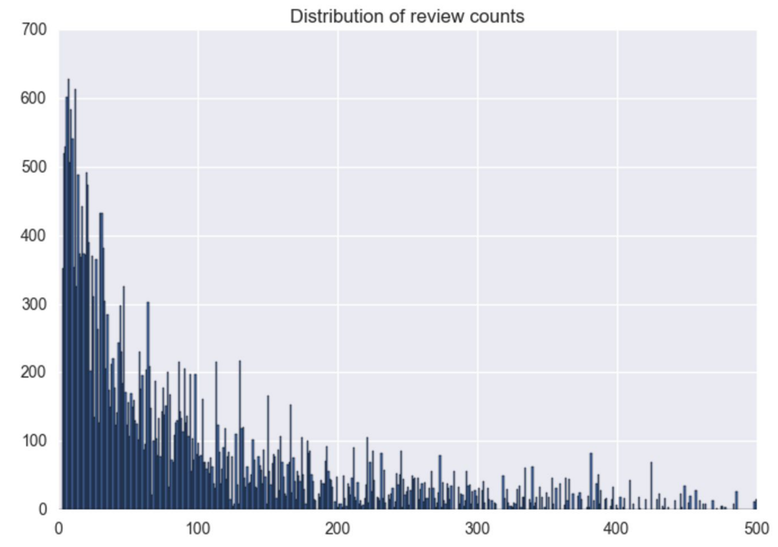
Health code violations ranged from 0 to 100, and the number of reviews ranged from 3 to 1922. However, as mentioned previously we removed instances with violation counts 4 standard deviations above the mean or greater, which reduced the violation range to 0-40. The health inspections dated back to October 4, 2006, whereas the most recent inspections occurred on October 25, 2016. There were a total of 45 different attributes within our dataset, with the most popular being American Food (6291 instances), Pizza (6090 instances), Sandwich Shops (5650 instances), Mexican Food (4403 instances), and Fast Food (2748 instances).

Each restaurant instance had an average of 121 Yelp reviews, 3.6 Yelp stars and 6.93 violations, with a change in violations from the previous inspection data of 1.5 violations.

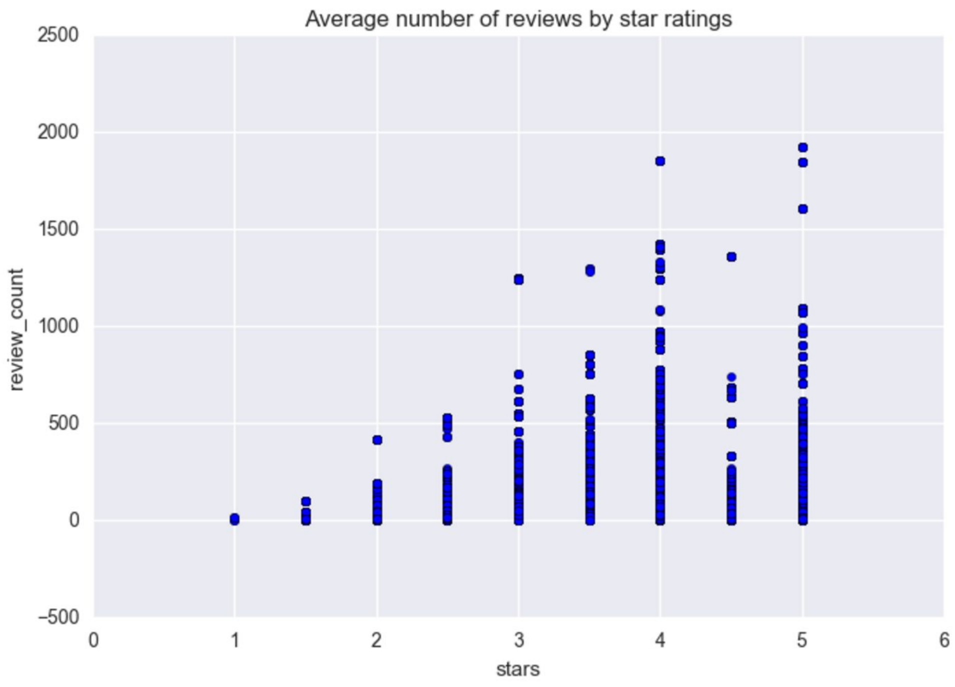


	violations	review_count	stars	pricerange	PreviousViolations	DiffPreviousTwo
count	34991.000000	34991.000000	34991.000000	34991.000000	34991.000000	34991.000000
mean	6.934669	121.214227	3.601626	1.475379	6.326484	-0.045297
std	8.340175	178.588309	0.700143	0.642554	8.282943	3.511681
min	0.000000	3.000000	1.000000	0.000000	0.000000	-98.000000
25%	2.000000	21.000000	3.000000	1.000000	1.000000	0.000000
50%	5.000000	60.000000	4.000000	1.000000	5.000000	0.000000
75%	9.000000	148.000000	4.000000	2.000000	8.000000	0.000000
max	100.000000	1922.000000	5.000000	4.000000	100.000000	100.000000

```
Mean number of reviews: 121.21422651538967
```

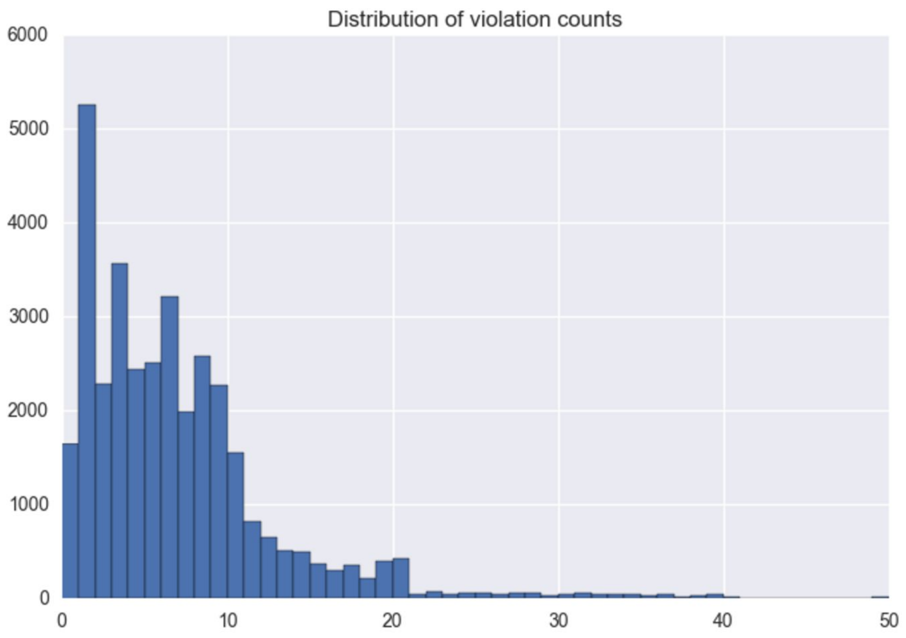


```
The average number of stars is: 3.6016261324340544 reviews: 121.21422651538967
```

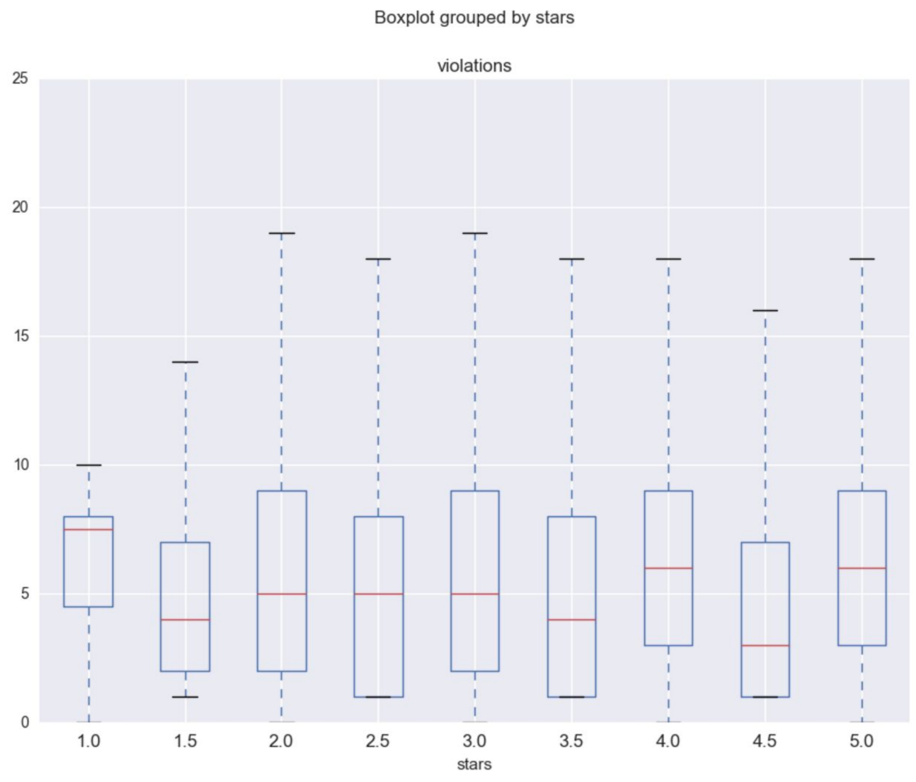


Almost all restaurant instances had between 0 and 21 violations.

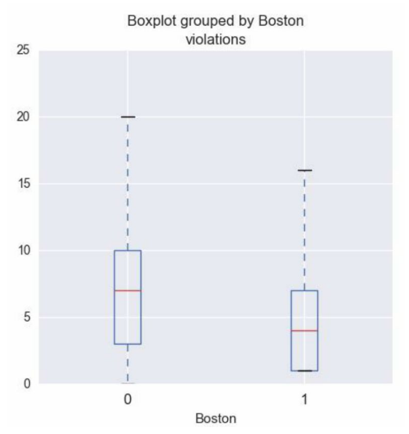
```
The mean number of violations is: 6.934668914863822 Standard deviation 8.340175123804492
```

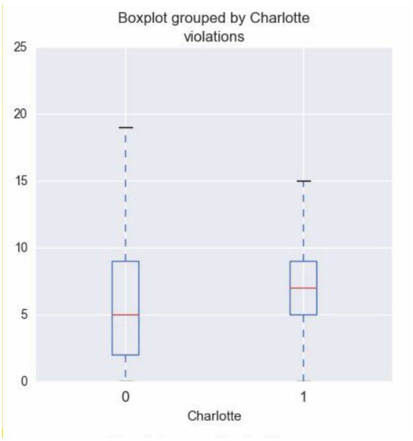


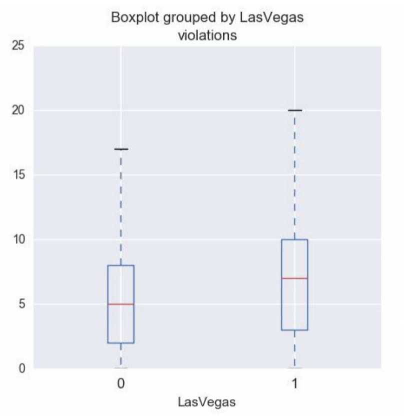
Unexpectedly, Yelp star count did not appear to correlate with violation count, except potentially the lowest star rating (1), having a slightly higher average of violations.



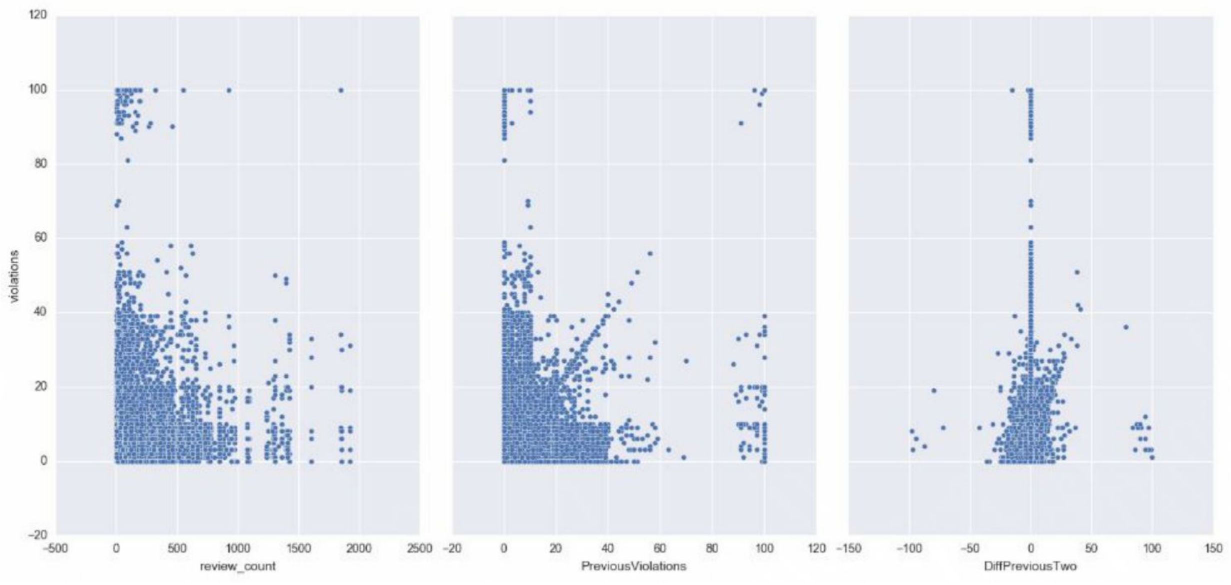
Restaurant instances in Boston appeared to have lower violation count averages than Charlotte and Las Vegas.



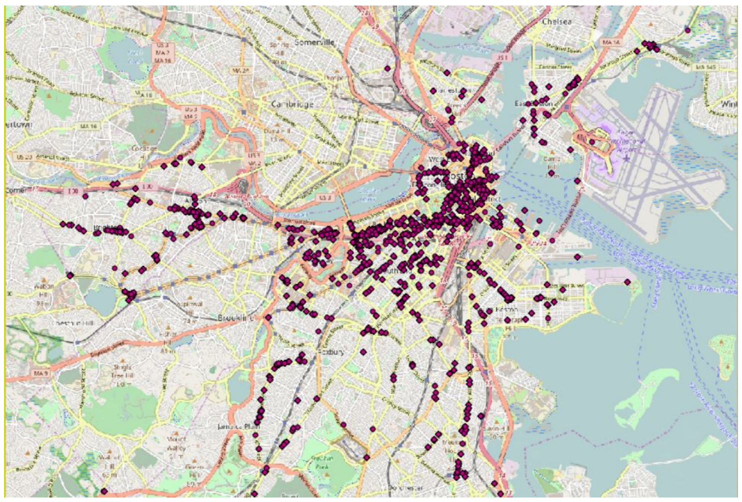
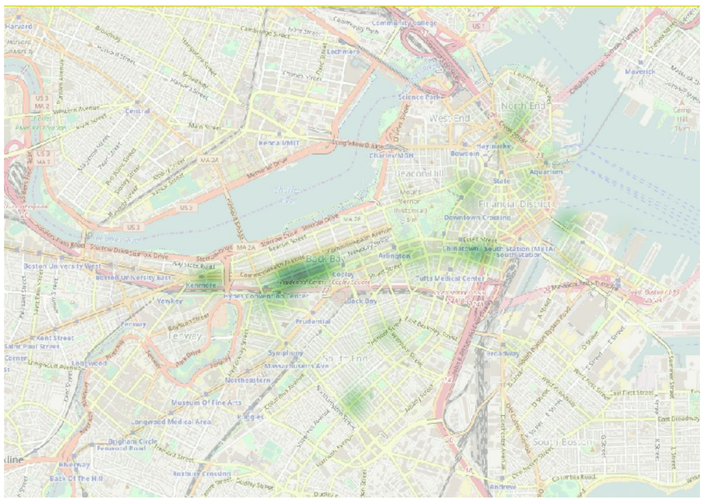




Predictably, the fewer reviews a restaurant received from Yelp users, the more violations it appeared the restaurant received. Previous violations roughly had a similar inverse relationship with violation count, however a 1:1 line does emerge somewhat as violation counts get higher, indicating that if a restaurant received a high number of violations previously, it will likely receive a similar number again. There may be a correlation between the difference in the previous two violation counts and current violations, as there is some direction indicating the more previous violations increased, the higher violations the restaurant will receive now.



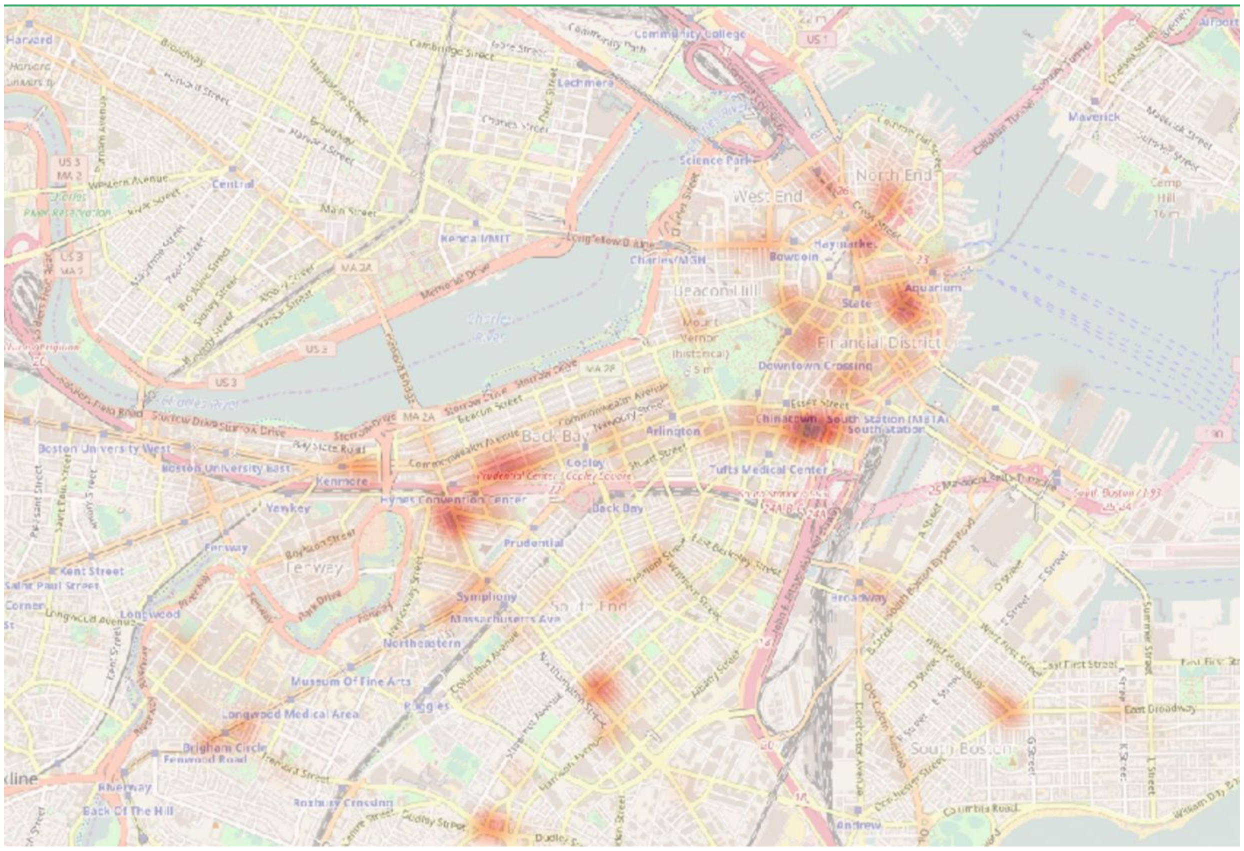
Geographic analysis, where latitude and longitude data was available (Boston), demonstrated that some neighborhoods have higher violation counts than others.

Restaurants in Boston

Restaurants in Boston with Yelp Reviews

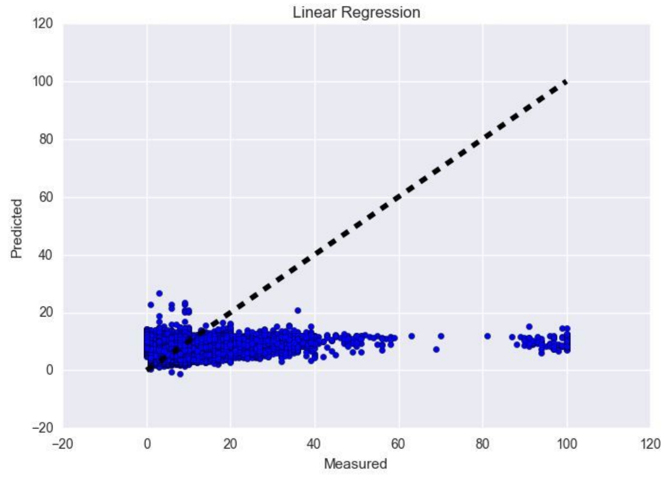
Restaurants in Boston by Violation Count



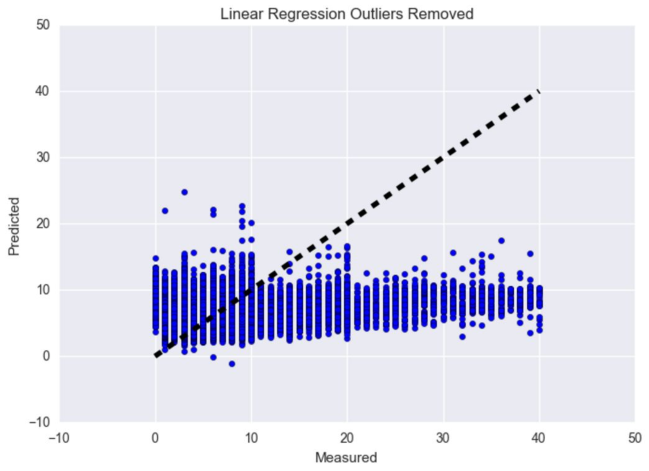
Data Modeling

Most of the regression models we fitted performed poorly, with R^2 scores of less than .2.

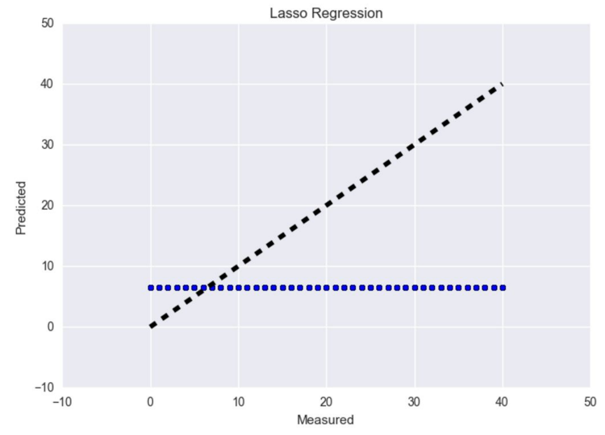
```
Linear Regression model
Mean Squared Error: 67.173
Coefficient of Determination: 0.081
```



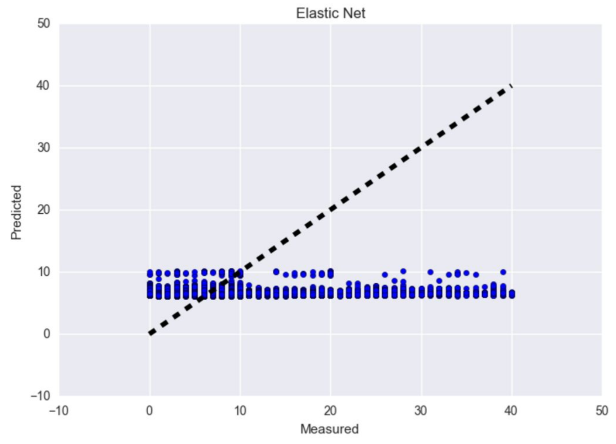
```
Linear Regression model
Mean Squared Error: 32.579
Coefficient of Determination: 0.092
```



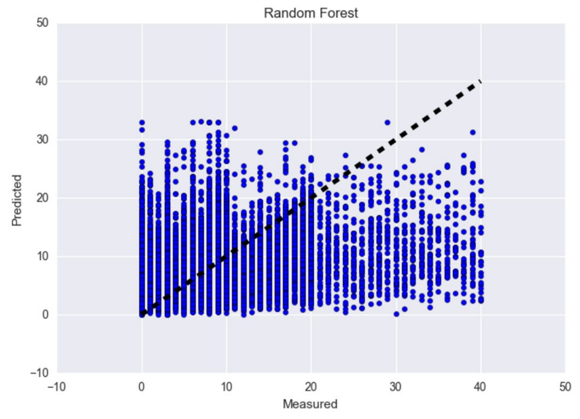
```
Lasso Regression model
Mean Squared Error: 37.902
Coefficient of Determination: -0.001
```



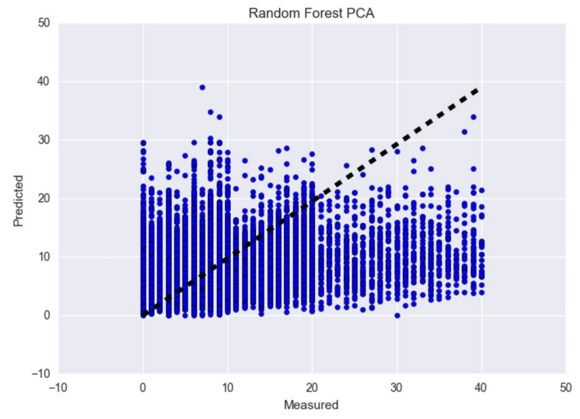
```
Random Forest model
Mean squared error = 37.288
R2 score = 0.015
```



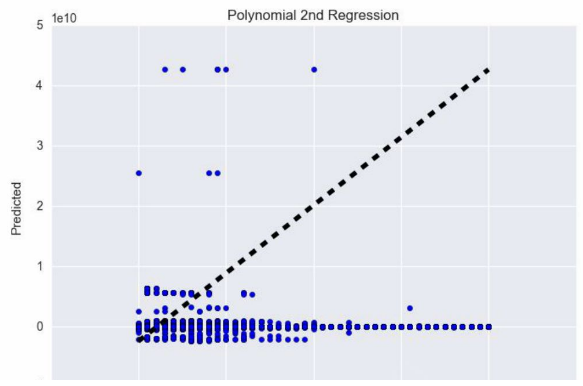
```
Random Forest model
Mean squared error = 35.195
R2 score = 0.035
```



```
Random Forest model
Mean Squared Error: 34.224
Coefficient of Determination: 0.037
```



```
Mean Squared Error: 30.654
Coefficient of Determination: 0.179
```



Ultimately, the second order polynomial model performed the best, with a mean square error of 30.6 and an R^2 of .18. Our R^2 score ranged from 0 (Lasso) to .18 (Second order polynomial). The mean squared error averaged close to 30, which is very high considering the average number of violations was 6.9, with a standard deviation of 8.3.

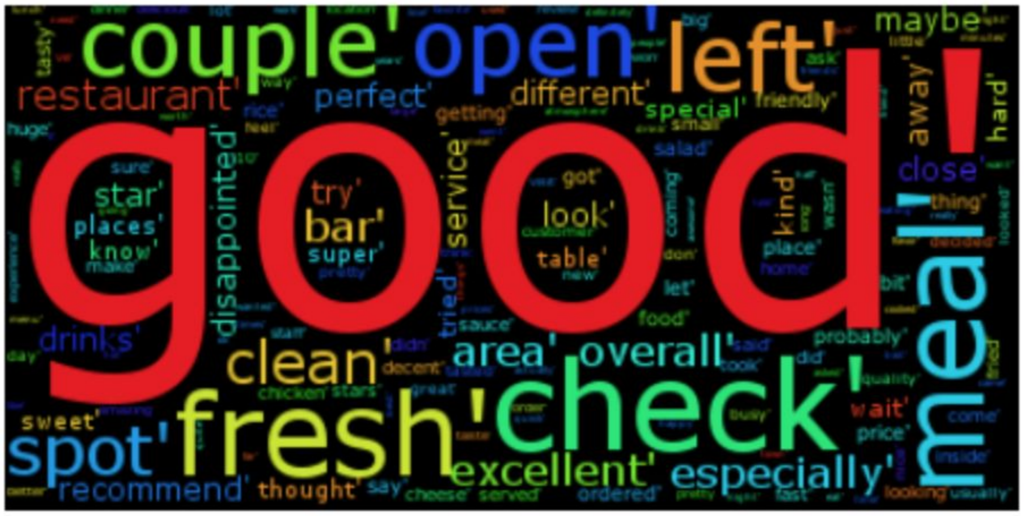
There are a number of possible explanations for such a poor model. First and foremost- lack of a relationship! In addition to the features in the dataset not being predictive of restaurant violations, it is possible that our our normalization of the violation counts between cities further damaged this relationship. Second, the sparsity of our Yelp user-created categorical attributes may have been a problem. Though we did have a few categorical variables with a significant amount of observations, many of our attributes only registered a few hundred, and this sparsity may have damaged the model performance.

Natural Language Processing

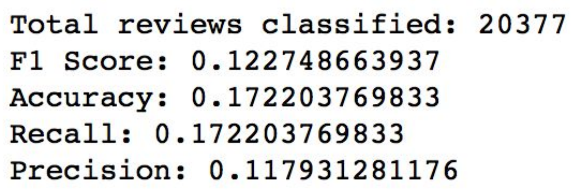
For the NLP portion of our modeling, we employed both count vectorization and text vectorization to explore the data. Ultimately, due to the very small nature of the word feature set after removing stop words and setting a min_df to .25, we chose to use the higher performing 20248x114008 sparse matrix with CountVectorizer's default parameters.

TF-IDF	Count Vectorization
63x136973 sparse matrix (min_df=0) 63x23388 sparse matrix (min_df=.25) (combined reviews from like violation counts)	20248x114008 sparse matrix (default parameters) 20248x163 sparse matrix (n_gram = (1,3), min_df=.25, stop_words = "english")

Those informative features from Count Vectorization are represented in the word cloud below.

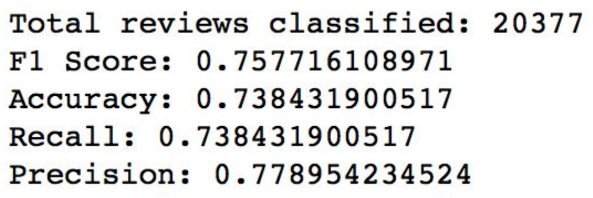


After running the Naive Bayes Multinomial classifier on this matrix, K-Fold Cross-Validation produced an F1 score of .12 (see NLP-NoBinning notebook on Github). This poor performing classifier is the result of F1 scores being set to zero during cross validation when the test data did not have a predicted value due to some violation count labels being unique.



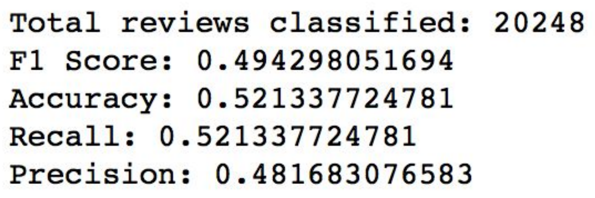
```
Total reviews classified: 20377
F1 Score: 0.122748663937
Accuracy: 0.172203769833
Recall: 0.172203769833
Precision: 0.117931281176
```

Therefore, we binned the data, with bins of 10 violations each. Binning is actually a more realistic depiction of how our end user might determine the need to re-inspect a restaurant, as a specific violation count prediction is no more useful than a dangerous violation count range. With binning, the classifier's F1 score, and all other measures of effectiveness, increased markedly. However, we predicted that this was likely due to the fact that most restaurants' violation counts fell between 0 and 40, roughly, making those four bins (of 10) disproportionately easy to predict upon.



```
Total reviews classified: 20377
F1 Score: 0.757716108971
Accuracy: 0.738431900517
Recall: 0.738431900517
Precision: 0.778954234524
```

Therefore, we removed outliers and created 8 bins with increments of 5 violations each. As might be expected, the accuracy of F1 score and accuracy fell as the dispersion of the data increased.



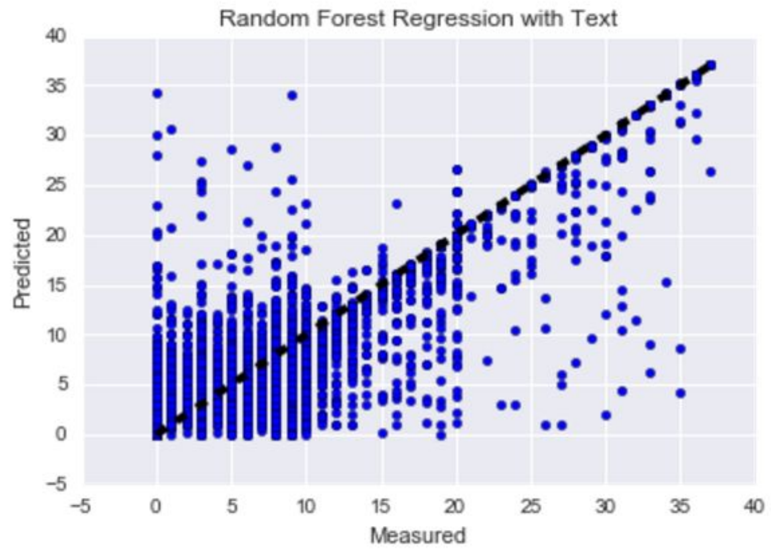
```
Total reviews classified: 20248
F1 Score: 0.494298051694
Accuracy: 0.521337724781
Recall: 0.521337724781
Precision: 0.481683076583
```

After running this classifier on our entire corpus, we merged the predicted values with our initial dataset. The predicted values were encoded with the maximum value of each bin:

```
group_names = {'Perfect': 5,
               'Excellent': 10,
               'Great':15,
               'Good': 20,
               'Bad': 25,
               'Very Bad': 30,
               'rats': 35,
               'Shutdown': 40
               }
```

The new text review dataset included 18,879 instances and 58 usable features. When this new dataset was tested with multiple regression models, the best fit was RandomForest with an R2 score of .79.

```
Random Forest model
Mean squared error = 5.543
R2 score = 0.792
```



This model was very well performing; however, this may be due to the fact that the new text-predicted variable was predicted on the data it was trained, which may have produced some leakage in the model.

VI. Lessons Learned and Future Research

By far the most difficult challenges we faced during this project occurred during the wrangling phase. The Yelp data was practically all user-generated, which made it extremely difficult to merge with the restaurant health inspection data. In the future, we'd like to attempt to use better entity resolution techniques, like fuzzy matching, to both speed up this process and improve the fidelity of the data. Additionally, there were several features from the inspection datasets that were dropped due to the time it took to merge the most basic features. Additionally, the nested dictionaries of restaurant categories and attributes only allowed us to extract these features in a sparse dataset, which likely diminished the predictive capacity of the model. We also lost quite a bit of Yelp and health inspection data that we were not able to merge, due to a small number of exact address and name matches. Incorporating these lost features might help improve the performance of a future model. These include: health inspection report text, name of inspector, correct latitudes/longitudes, violation severity, etc.

We also learned that being more flexible in our modeling approach may have yielded better results. For example, since we wanted to predict violation count, we assumed we would have to use a regression; however, binning violation counts and using a classifier, or even clustering may have had uncovered more insights into the data. We also would have liked to have had more time to determine the generalizability of our model, by testing its accuracy for each city alone. In doing so, we might also be able to determine which cities were skewing our data (and potentially scaling or munging the data to prevent this). Additionally, subject matter expertise from local health inspectors would help us to better understand how each cities qualifies "one violation" so that we could better scale across cities to normalize our data - for example, if one violation in Boston was equivalent to two in Las Vegas, this would have been valuable knowledge.

In the future, we'd like to turn this model into a tool for health inspection agencies, to provide updated violation predictions as more Yelp users review restaurants. At the very least, we would like to develop an application where you could manually input different variables and receive a prediction on the violation count- this could be very useful for conscious consumers. We'd also like to explore the natural language processing section of our report more. Given the vastness of this field, there is significantly more we could do to improve the accuracy of our model, such as lemmatization, other vectorization techniques, and finally other classifiers (outside NLP).

Finally, there has been work by other data scientists on this problem set in several other cities. Reaching out to other researchers to better understand their methods could provide us with additional methods, relief of pain points, and potentially greater generalizability across these cities.

VII. Conclusion

We found that the many features derived from Yelp reviews, which we originally thought would be predictive, did not appear to have very much predictive power in reality. Though adding in the natural language processing significantly increased the accuracy of the model, it is difficult to say if leakage played too much of a role in its high score.

Ultimately, ensuring more time to wrangle our data, incorporate subject matter expertise, and focus on natural language processing will hopefully provide greater accuracy to our model. However, it is possible that yelp features, aside from text, simply do not predict a restaurant's violation count.