Predicting Restaurant Health Scores from Yelp User Reviews

Big D Project Report

Sean Mozarmi
Jason Mach
Jonathan Nakhla

Problem Statement and Background

Problem Statement

The question we seek to answer: can we predict San Francisco food establishment health scores based on Yelp user reviews? Because San Francisco has such a large number of food establishments compared to other cities in the U.S., there are not enough health inspectors to regularly schedule health inspections for all of these food establishments. We attempt to develop an early warning system in order to help health inspectors determine which food establishments are in need of inspection.

Background

The San Francisco Department of Public Health determines the frequency of health inspections for food establishments based on the category of food establishment and the health score of the establishment. The category of the food establishment is based on government health codes. Category 1 food establishments are classified as restaurants and establishments that serve cooked food. Category 2 establishments may serve food as a secondary function—a bar, for example. Category 3 establishments, such as gas stations, sell only packaged and non-perishable goods. Our research is meant to assist health inspectors with inspections of 'Category 1' and 'Category 2' food establishments, although our research can be used to define more specific thresholds. Health scores are of the range (0-100), inclusive:

Category 1 (health score > 80 points)	2 routine inspections per year
Category 1 (health score <= 80 points)	3 routine inspections per year
Category 2	1 routine inspection per year
Category 3	Routine inspections are not required

Impact and Importance

By using our information, health inspectors can examine predicted health scores for restaurants based on information *after* the restaurant's most recent health inspection. Based on these health scores, health inspectors can set thresholds to determine which restaurants they would prioritize for health inspections. Ultimately, this prioritization would give restaurant-goers a more accurate and up-to-date status of a restaurant's health inspection score.

Data Sources

We used Yelp restaurant websites to obtain all of our initial data. Data on the Yelp food establishment websites includes general information about the establishment such as name, price category, and location, as well as user ratings and text reviews. Each restaurant also contains a separate Yelp link that includes all information for each health inspection, taken directly from the San Francisco Department of Health's food establishment health score data set. Yelp updates this data pull from the San Francisco Department of Health periodically, so some restaurants may not have had the most recent inspection listed. Some restaurants did not even contain associated health information. We found that these cases were minor in number and did not affect our sampling, as we treated each health inspection as a separate data point (i.e. two health inspections for the same restaurant would be treated independently).

Ground Truth and Base Accuracy Metrics

Because we had ground truth for our research in the form of food establishment health scores from the San Francisco Department of Public Health, we could easily form informal success measures. We could measure the deviation of our predicted health scores from actual health scores given by the Department of Health for previous inspection periods. For example, we could form a metric for accuracy, where a health score prediction would be considered accurate if it was within +-5 points of the actual health score. Because SF health inspectors increase the frequency of inspections based on thresholds, our predictions do not need to be accurate to the health point, but rather only need to be accurate within a threshold range.

Cross Validation

Of the data that we scraped, we randomly assigned inspection periods to our training and validation sets, in a ratio of 80% for training and 20% for validation. We had a natural test set – our test set was comprised of all Yelp user reviews in time periods per restaurant that did not receive a health inspection yet. For example, if today's date was December 1, 2014, and a restaurant's most recent inspection was February 1, 2014, we would include Yelp user reviews from after February 1, 2014 to the current date in our test set. This is a natural test set because there are no "labels" in the form of health scores associated with the most recent time period.

Related Work

We could not find any related work. We suspect this is because of the difficulty with obtaining Yelp user review data, as full user reviews of food establishments are not queryable under Yelp's public API.

Methods

Data Extraction

We resorted to scraping relevant data from Yelp food establishment sites because Yelp's public API was very limited and did not allow querying of full user review text. We wrote a tailored scraper to extract all of the data from each food establishment that we thought could be used as a variable in our regression later on. The data that we performed scraping operations on was of HTML format. Yelp includes San Francisco Health Department health inspections and associated health scores, so we obtained both the inspection data and restaurant data simultaneously.

Another potential source of data was the San Francisco Department of Health's public data set on food establishment inspections. This data set included all food establishment names and corresponding health scores, with specific health violations. Yelp pulled this data directly from the SF Department of Health, so it was easier for us to stick to one data set: Yelp. We found that the SF Department of Health's data set would have had major problems in referencing Yelp's list of restaurants as well, as it had different spellings of food establishments than Yelp's spellings. This would have made joining tables troublesome, as our primary key would have been food establishment names.

Our final data set consisted of 1,021 data points and 200,942 Yelp user reviews. It is important to keep in mind that each data point is an inspection period. We used around 100 SF restaurants of varying inspection period scores for our data.

There are many potential biases that could have arisen from our sample selection. Our sample set may not be representative of the overall San Francisco restaurant distribution, in regards to restaurant price category and restaurant category. This could have resulted in categories of restaurants being used as strong features, even though they were not representative of the overall restaurant category health implications. For example, in our data set, "german" was one of our top features. We only had one German restaurant in our data set, and it happened to have low health scores, resulting in overfitting. We also are obviously biased towards less popular restaurants, as we do not look at restaurants with over 500 Yelp user reviews. Thus, if restaurant popularity is correlated with restaurant health scores, we could be underestimating or overestimating our predictions based on the correlation sign. However, we are confident that our Yelp user review data is representative and covers all inspection periods listed per restaurant.

Data Wrangling

After we had scraped data from Yelp, the next step was to wrangle our data to meet our problem statement. We needed a way to label our data points if we were going to perform supervised learning in the form of regressions. To do this, we transformed our data in order to factor out time. Because health inspections are inherently bound to a certain time frame, we could associate Yelp user reviews with their corresponding health inspections in order to form data points devoid of time. As a result, each data point in our data set is not a food establishment and associated score, but rather, an inspection period and associated score. In this way, we could treat data points more independently, as even data points of the same restaurant would have different health scores and different associated Yelp user reviews. In order to fit this data set to the popular *csv* format, we had each row contain 3 major columns:

1. *Food Establishment Inspection Period:* The food establishment name and inspection period (i.e. inspection period of '0' would refer to the oldest inspection on record for the food establishment)
2. *Yelp User Review Text Corpus:* An accumulation of all Yelp user reviews for the associated inspection period of that food establishment. Each user review was treated as a string and concatenated to other user review strings (reviews were space delimited). We did not delimit user reviews further, as there was no reason for us to physically separate Yelp user reviews. We treated each word as unit in our bag of words model, each pair of words as a unit in our bigram model, and later each sentence as a unit in our NLP model.
3. *Health Score:* The health score for the corresponding inspection period given by the San Francisco Department of Health, on a scale of 0-100, inclusive.

Although those were our main three columns, we included many more columns in order to perform periphery analyses to get better context around our main problem, which we will explain further on. Our data set is formatted as such, where each '|' delimited phrase is a column of our data set:

name|total_restaurant_rating|restaurant_category|restaurant_price_category|number_of_restaurant_reviews|inspection_period|inspection_period_rating|inspection_period_review_text|number_of_inspections|inspection_period_health_score|number_of_violations|inspection_period_type|inspection_period_violations

The columns 'name', 'total_restaurant_rating', 'restaurant_category', 'restaurant_price_category', and 'number_of_restaurant_reviews' are simply meta information about the restaurant the inspection period data point belongs to. The last 3 columns are extra health information about the inspection period that we did not use in our research. The other columns are inspection period specific, which is what our research analysis revolved around. 'Inspection_period_review_text' is an aggregation of all Yelp user reviews for that corresponding inspection time period.

Although this was our final data set format, it is relevant to explore a previous format that we used and later scrapped in order to understand why we wanted to factor out time from our data set. Originally we had our data set formatted by Yelp user review, so that each data point was one Yelp user review. Although this gave us more slightly more accurate information (each Yelp user review would be separated by data point and not aggregated), the duplication of data and preponderance of rows greatly outweighed any benefits from separation by Yelp review. Because each data point was separated by Yelp user review, all other columns were repeated needlessly. With this method, we also had to include date columns for the date of the user review, as well as the dates of the San Francisco health inspection periods. The idea was to later use these dates via Pandas or SQL operations to group together data according to dates and label Yelp user reviews. We found it much easier to perform these groupings at an earlier level via our Python scraping rather than perform needlessly complicated post processing.

Feature Extraction

Once we had obtained all of our data in a labeled format, our next step was to extract a feature set from our data. We will discuss our featurization procedures for Yelp user reviews first, and after discuss our featurization methods for periphery regressors that we explored, such as Yelp user period ratings, number of Yelp reviews per period, restaurant price category, and restaurant category.

Our first featurization method for Yelp user reviews was a bag of words model. For each Yelp user review text corpus (one corpus per inspection period, which is the data point), we formed a list of words. We then used the count of each word as our weight, so that words that occurred in higher frequencies were ranked as more important features. While this method was a good starting point, simple word frequency rankings did not take into account the total number of words in each review corpus; for example, one review corpus could have had 2,000 words, while another could have had 10,000. Because we were not taking into account the denominator, our rankings could have been biased towards text corpuses with more words. This method also did not take into account the fact that some words are generally more common than others. To solve these problems, we utilized term frequency-inverse document frequency, or tf-idf, weights for our words, so that we offset common words and took into account the denominator of total words. We attempted 3 variations of our bag of words tf-idf method: only single words, only bigrams (word pairs), and a mixture of single words and bigrams. We found that a mixture of single words and bigrams performed slightly better than only single words or only bigrams, although the extra time necessary to compute the mixture might not be worth the improvement when testing with larger data sets. Furthermore, we excluded any words that were very common and meaningless by using a set of Standard English stop words to compare against. We also unified all words in our feature set that were small variations of each other into one feature (e.g. treat “greeeeeat” as “great”).

After trying a bag of words model, we explored natural language processing featurization methods, seeing as our text corpus consisted of user-defined entries in English. We utilized a dependency parser in order to get parts of speech (POS) labels for each word of a sentence in our Yelp user review text corpus. In this case, we treated each sentence as a unit rather than each word or pairings of words as in the bag of words method. This gave us more information to work with. At first, we tried incorporating all POS tags for each word in each sentence of our corpus, but found that this process, although information heavy, was not worth the hit to performance. Many POS tags were of little use to us, and including all POS tags would result in a level of performance too slow even for our modest data set. In order to get the most out of our POS tags, we chose to focus on combinations of all adjectives, adverbs, and nouns in each sentence. We used these noun phrase combinations per sentence as our final feature vector.

Before our POS tagging NLP method, we explored the NLP method of sentiment analysis. However, this method was quickly dropped when we realized that Yelp user sentiments were not correlated with restaurant health scores, a conclusion we reached after performing a regression with said variables and finding an enormously high residual sum of squares.

We evaluated each of these feature extraction methods by using regularized regressions on our feature set. We essentially depended on regularized regression for feature selection and did not manually choose a subset of our total feature set.

Supervised Learning: Classification or Regression

Once we had extracted and selected useful features from our data set, our next step was to choose a supervised learning algorithm. At this point, it is important to explore why we chose to pursue regression as our supervised learning method instead of classification, and how we reframed our original problem statement to do so. Originally, the problem we were trying to solve was: can we predict whether food establishments in San Francisco are in need of inspection based on yelp user reviews? This problem is very different than the one we actually solved: predicting food establishment health scores based on yelp user reviews. The original problem was a binary classification problem, in which we would determine whether a food establishment was in need of inspection, or not in need of inspection, based on some self-defined threshold. For example, if we were solving the original problem, we would have chosen health scores above 80 points as "not in need of inspection", while health scores below 80 points were "in need of inspection", in accordance with Category 1 San Francisco health codes.

Instead, we chose to reframe the problem to be one of regression, in which we seek to predict a health score per inspection period rather than classify a food establishment. Although this was a more difficult problem, it gave us a lot more flexibility with creating useful thresholds, and our hope is that this will allow health inspectors to develop more fine thresholds that will allow them to prioritize health inspections. In addition, framing the problem in terms of regression allowed us to treat each inspection period as a data point instead of classifying food establishments directly, in which we would have to aggregate all health inspections per restaurant in some way. Aggregating in this way would lead to information loss.

Regression

We used a variety of regression methods, but found that regularized regression algorithms performed better than their non-regularized counterparts. Regularized regression allowed us to essentially perform feature selection by tuning the regularization parameter of the regression. By changing the measurement of the regression loss function via a feature penalty, the regularized regression mitigated results from overfitting of our training data.

Tools

Data Extraction & Wrangling

For our scraping efforts, we chose to use the python library Beautiful Soup. We purely relied on this tool for scraping because all of the data we scraped from Yelp was in HTML format. As such, we did not need to perform scraping on JavaScript, which would have required other libraries such as PhantomJS. We used the Python requests module to handle all of our HTTP requests instead of urllib2 because of the ease of using proxies with the requests module.

We primarily used Pandas for our data manipulation efforts. We chose Pandas because our data set was too large to even view in traditional csv viewers such as Csv Viewer for Linux. We also used

Microsoft SQL Server (MSSQL) to view early versions of our data set. We found that working with Pandas was much easier than MSSQL because Pandas was Python based, and all of our scraping and featurization/regression was implemented in Python.

Featurization

We primarily relied on Python's sklearn library for our featurization methods. For our bag of words model, we used sklearn's CountVectorizer for our very first model, where we rated features by frequency. We then used sklearn's TfidfVectorizer to use tf-idf weights for our features. We chose these methods instead of a DictVectorizer (which would allow for different feature categories, such as restaurant category and price category) because we wanted to focus solely on the relationship between Yelp user reviews and restaurant health scores in our primary regression. For our periphery regressions, where we regressed restaurant health score on restaurant category, restaurant price, and inspection period rating, we used sklearn's DictVectorizer. Each inspection period had a dictionary of period Yelp user rating, a continuous variable, and the categorical variables: restaurant price range and restaurant category. We also used the Python nltk module for its stop words list. We filtered out stop words from our feature set, except for negative words such as "not". We utilized Python' regular expression functionally to create a single feature for words that were very similar.

When we ventured into the field of NLP, we first tried using the Stanford Parser, but found that it was too slow for even our modest amount of data. We then looked into TextBlob, a very nicely integrated Python library for NLP. We were very attracted to TextBlob because of the multitude of NLP options it had available via its API, including noun phrase extraction. We quickly found that TextBlob was not scalable for our data needs, as even passing data into the TextBlob constructor took too long. We would have liked to see TextBlob accept Pandas data structures such as numpy sparse arrays into a TextBlob constructor instead of just text. Eventually, we used the fast dependency parser MaltParser. Although MaltParser was a written in Java and therefore would require extra labor to integrate into our feature extraction (implemented in Python), its performance was very fast without losing too much on accuracy. We used MaltParser's pre-trained English model in order to perform our POS tagging. In order to use MaltParser, we first extracted our text corpus into a csv file. We then converted the text csv into CoNLL format, which we fed into MaltParser. We had to perform some configuration in order to output a file in CoNLL format from a text csv file. This involved changing some default configurations in MaltParser as well as using the Stanford CoreNLP tool suite. Feeding the CoNLL file into MaltParser resulted in a file with POS tagging. We then loaded this POS file into Pandas and performed dataframe manipulation in order to form our own combinations of noun phrases in our feature set.

We used Amazon's EC2 service in order to perform NLP using MaltParser, as our local machines did not have enough memory. We used g2.2xlarge instances, with 8 virtual computers and 15 GB of memory.

Regression

We primarily used Python's sklearn library for our main regressions. For our initial exploratory regressions, we used Python numpy's simple regression module, polyfit. These exploratory regressions including regressing inspection period health score on the number of Yelp user reviews in the inspection period, as well as regressing inspection period health score on the inspection period Yelp user rating.

We explored many different regressions in sklearn's library, including ridge regression, lasso regression, forest regression, linear regression, and SGD regression. We narrowed down our

regressions to regularized regressions in order to also perform feature selection by adjusting the regularization parameter. We settled on ridge regression because of its slightly better results.

Visualization

We developed word cloud visualizations for our top negative and positive coefficient features of our regression. We made this word cloud based on the corresponding tf-idf weights of our features. We excluded restaurant category features like “chinese” that could have been a result of overfitting or sampling selection bias. The tool used to create the visual was http://www.wordle.net. Moreover, we used Python to display regression plots for our exploratory regressions. We used Python’s matplotlib library, specifically pyplot, to plot our regressions.

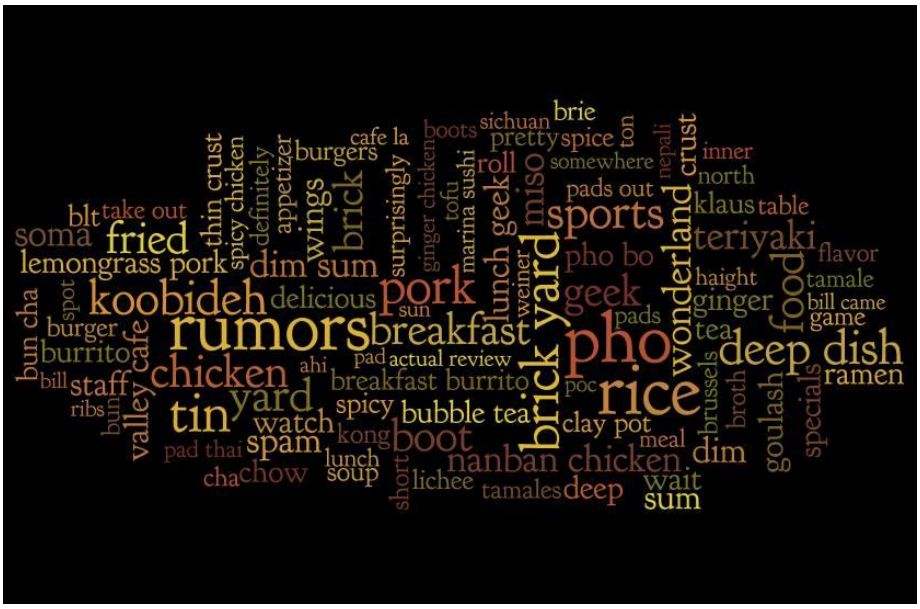
Results

We measured the performance of our predictions with three main metrics: residual sum of squares, variance, and our self-defined metric: accuracy within += n deviations from the actual score. The residual sum of squares is another metric we used to evaluate our model. It is a measure of the discrepancy between the true data and our estimates. The lower the metric, the less the discrepancy, and better our model accuracy. We also used variance. The variance score tells us how accurate our prediction was, on a scale of (0-1) inclusive, where 1 is a perfect prediction. Specifically, this variance metric is explained variance, which explains how well our model accounts for the dispersion of our data set.

Below are the top features of our NLP model. We evaluated features by ranking them according to their tf-idf weights. We included the top hundred features for each word cloud here, excluding potentially biased features.

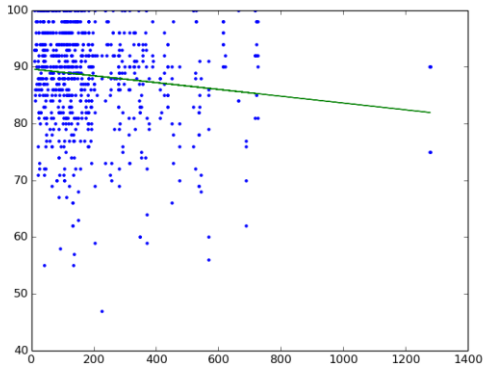
Our top **positively** correlated features were:

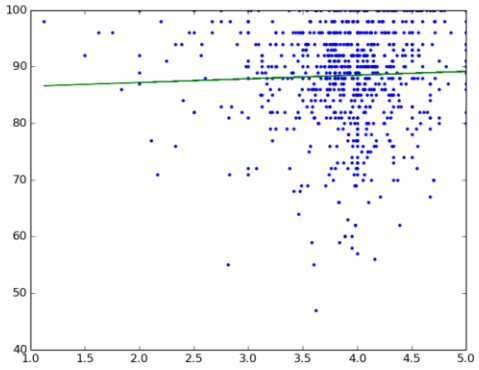
Our top **negatively** correlated features were:



Our most negative feature had a coefficient of: -18.84
Our most positive feature had a coefficient of: 13.91

The residual sum of squares for our regression of number of Yelp reviews on health score: 62934.96 (Regression depicted on left graph)
The residual sum of squares for our regression of health scores on Yelp user restaurant ratings: 63848.91 (Regression depicted on right graph)





As you can see, the above RSS scores were extremely high. We performed a chi-squared test on each regression above, with 1 degree of freedom on each regression. Our resulting p-value at a 0.05 significance level was less than 0.00001 for both regressions, which means our result was significant. We thus concluded that there was no correlation between a restaurant's health score and its number of Yelp reviews. We concluded that there was no correlation between a restaurant's health score and Yelp user ratings for the same reason. This means that a restaurant's average Yelp user rating has little to do with its health scores, i.e. a positively rated restaurant can either have a low or high health score rating.

We found that ridge regression performed the best with our manually tuned regularization parameter (alpha = .001). Our final results are below:

```
Residual mean squared error (lower=better): 67.66
Variance score (1 is perfect): 0.32
Accuracy WITHIN +- 0 range, measured on integer scores: 9.49%
Accuracy WITHIN +- 3 range, measured on integer scores: 41.13%
Accuracy WITHIN +- 5 range, measured on integer scores: 52.53%
```

We found a correlation between Yelp user reviews and restaurant health scores. It is unclear whether this correlation was statistically significant under a p-value test, as each of our features had an associated p-value by our calculations. We suspect that the correlation was not statistically significant at a 0.05 significance level, as many of our p-values were close to 1. This is not a unexpected given our small data set.

We also found a correlation when regressing Yelp restaurant health scores on Yelp restaurant category and price category. From our results, it seems that restaurants in the "below $10" price range were associated with lower health scores. In terms of restaurant category, we saw some restaurant categories that had lower associated health scores, such as those restaurants classified by Yelp as "Ramen in San Francisco". Our residual sum of squares metric for this regression was ~50, with a variance of ~0.35. We did not include more detailed information about this regression in this paper, as it is only corollary to our main research goal.

Lessons Learned

The primary lesson we learned in our research was not to underestimate the time required in the data extraction phase, i.e. scraping Yelp for our raw data. We spent about 75% of our overall time on this project in the data extraction phrase, both in scraping the raw data from Yelp and wrangling the data to match our needs. This was an unexpected time commitment for us, as we did not realize before starting our research that Yelp did not have an open API where you could query all of Yelp user reviews per restaurant. It is advisable to scout all potential data sources very early in the project timeline to spot potential data extraction problems and plan accordingly.

We learned an important lesson about performance versus accuracy tradeoff as well. This lesson was learned when we were choosing our NLP tool. Stanford Parser was very accurate, but very slow compared to the acceptable but much faster MaltParser. We were willing to give up some accuracy in exchange for substantial increases in performance, as our corpus had so many Yelp user review sentences that some inaccurate reports would not be of consequence to us. In fact, the inaccurate POS markings would almost certainly be excluded in our feature selection process, so the only part that would be affected slightly if at all would be regression feature coefficients. We also learned that often there is a tradeoff between compatibility and performance. The easiest method for us to integrate NLP into our Python learning modules would have been TextBlob (it even had noun phrases built in as a function), which was a Python library. However, TextBlob was too slow for our needs, so we resorted to MaltParser. Even though MaltParser required a lot of configuration and implementation of our own noun phrase logic, the performance increase was well worth the trouble.

Another crucial concept we learned was not to manually create features and form biased expectations for features. For example, before performing regression and feature selection in our project, we assumed that the top features that would be associated with low health scores would be directly related adjectives to cleanliness such as "messy" and "unclean". We were surprised to learn that this was not the case at all; Yelp users usually did not directly take cleanliness into account in

their restaurant evaluations. Rather, our top features contained words and phrases that were correlated with health scores across all restaurants, such as “take out”, “pork”, and “fried” (features associated with top *negative* coefficients of the regression). We learned to let the regressions perform initial feature extraction for us rather than prematurely insert our own biased ideas of important features into the feature matrix.

Team Contributions

Initial responsibility guidelines:

Sean Mozarmi: Responsible for feature extraction and regression.
Jason Mach: Responsible for data visualization and data scraping.
Jonathan Nakhla: Responsible for data visualization and data scraping.

Each member had defined responsibilities at the start of the project, but every member helped out with other members’ responsibilities. For example, every member had to contribute to the data scraping effort because it was the most time consuming part of the project. We estimate an even estimation of time put into the project by each member (33.3%, 33.3%, 33.3%).