Automated Detection of Restaurant Virality using Yelp Reviews

Christina Dominguez (ctd299), Chris Ick (ci411), Justin Mae (jm7519), Trevor Mitchell (tim225)
Center for Data Science, New York University

Table of Contents:

Business Understanding:

The time, labor, and capital required in the food writing industry is highly demanding. Readers frequently turn to their favorite publications for curated restaurant recommendations. A single city may have hundreds of new restaurant openings a year; sending a review team out to each location to sample and report back findings on each requires maintaining an expensive team of reviewers year-round.

With the rise in popularity of restaurant review sites such as Google Reviews, Yelp, Tripadvisor, and Facebook pages, diners increasingly self-report ratings, descriptions, and photos of their experience at new restaurants. We sought to leverage this information to automatically generate shortlists of promising new restaurants to visit to potentially feature in our publication. The benefit of this would be twofold; our publication could gain a competitive edge by being able to report more quickly on trending food establishments than our competitors, and we could reduce review team expenses as we would not have to do as much research on new restaurants or visit as many locations to find promising new restaurants to feature.

To do this, we used Yelp data. Yelp.com is the 43rd most visited site by visitors and page views in the US and has exploded in popularity since its founding in July 2004. There are 171 million lifetime reviews on Yelp and the site's monthly users continues to grow each year. It is among the most popular restaurant review websites in the US and thus could serve well as a crowdsourced source of diner feedback for our problem.

Dataset:

We leveraged data from the Yelp Dataset Challenge, offered by Yelp to students for the purpose of research and innovation. The dataset covered 5,996,996 reviews across 188,593 businesses and included 1,185,348 tips. It consisted of 5 json files of which we used the following 3:

- **Yelp_academic_dataset_review** - a single instance is a unique review with its corresponding rating, the business it concerns, the author, date, and user-voted reactions such as "funny"
- **Yelp_academic_dataset_business** - a single instance is a unique business with features such as location, star rating, and category/attribute tags
- **Yelp_academic_dataset_tip** - a single instance is the tip itself, along with the date it was made, the number of likes it received, which business it concerned, and the author

Our dataset was limited by the the data Yelp chose to make available for the Dataset Challenge, which was limited to 10 metropolitan areas only, thus some sampling bias was introduced. This is addressed by focusing on a single location as explained below.

Data Preparation, Modeling, and Evaluation:

The massive size of our dataset introduced significant challenges for the computational workload of standard personal computers. To remedy this we leveraged NYU's High Performance Computing resources (NYU HPC). To avoid the problem that regional variations would introduce in our model and limit the size of our dataset, we decided to focus solely on Arizona, the US state containing the highest review count.

Our dataset did not contain our target variable of interest. Deciding upon the criteria for defining our target variable involved some trial and error and in the process we ended up creating two entirely separate models; a yearly model to start, followed by an improved monthly model. We will elaborate upon the data preparation, modeling, and evaluation process for these two models separately.

Model Iteration 1 - Yearly

Data Preparation

As a proof of concept, we simplified our business problem by creating an intuitive model that predicts the virality of a restaurant in the second year of its opening based upon review counts received in its first year of operation. We defined a "viral" restaurant based upon the following criteria, which yielded 230 viral restaurants out of 3,010 or 7.6% of all restaurants.

1. Business contained a category of "Restaurants"
2. Second year of business operation from opening date has review count > 18 reviews (70th percentile)
3. Second year of business operation from opening date has average stars > 3.75 (50th percentile)
4. Change in review count from first to second year > 12.5% (60th percentile)

In addition to creating a viral restaurant target variable, we also created the following features:

- **Open Date** - opening date of a restaurant based on its earliest review date. This was used to split data into first and second year of business operation. Furthermore, we transformed daily reviews into monthly by binning all reviews into months and dropping the first month to ensure opening day of month is consistent across all restaurants.

- **Last 3, 6, 9, 12 month average stars** - average stars for different periods in the first year. These features along with their differences were used to determine quality growth.
- **Last 3, 6, 9, 12 month average review count** - average review count for different periods in the first year. These features along with their differences were used to determine review growth.

Model Creation and Evaluation

To start, we split our test and training sets in a stratified way because the percent of the dataset containing the target variable was small. For our evaluation metric, we chose AUC score because of its base rate invariant property.

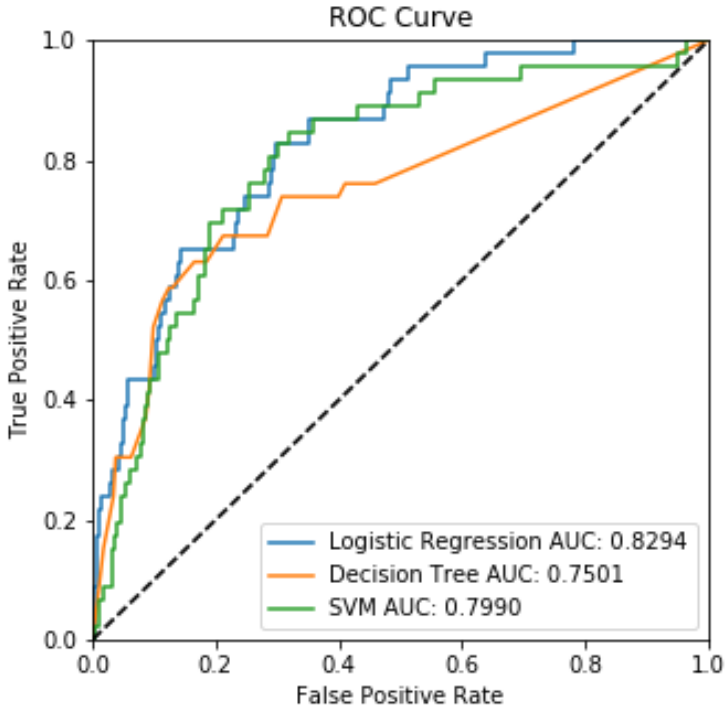
For our yearly model, we employed three major classifiers:

- **Decision Tree** - intuitive and handles variable interactions well, but can be biased to the training set
- **Logistic Regression** - efficient and robust to noise, but doesn't handle categorical features well
- **SVM** - can handle non-linear feature interactions, but is not very efficient with large observations

An added benefit of these classifiers is the outputted probability scores providing additional insight.

We started with decision tree as the baseline model and improved upon it using the other 2 classifiers and hyperparameter tuning. To tune our hyperparameters, we utilized grid search to test each model over a stratified 5-fold cross validation set. To avoid overfitting, we used the 1 standard deviation rule to choose the parameter that resulted in AUC scores 1 standard deviation below the best. These are the grid search results:

Classifier	**Best AUC Score**	**Best AUC Score - 1 std error**	**Optimal Parameters - 1 std error**
Decision Tree	0.7627	0.7553	Min Samples Leaf = 32
SVM	0.7792	0.7708	C = 1000, gamma = 0.0001, kernel = rbf
Logistic Regression	0.8286	0.8122	C = 0.1, penalty = L1



We settled upon logistic regression as our best model based on the fact that it had the highest AUC score and is not more complex than the other classifiers evaluated.

Our yearly model as a proof of concept reduced complexities of the problem such as seasonality and leakage and showed us that predicting restaurant virality is feasible. Realistically however, this model did not solve for our business problem. By design it required a year's worth of data and thus could not identify any restaurants as "viral" for a minimum of 1 year. This delay in the context of our business problem was unacceptable; a publication that does not cover a popular new restaurant for over a year would not be perceived as a valuable, up to date source of information by its readers. Thus in future iterations we pivoted to a monthly model which we hoped would successfully identify restaurant virality at a much quicker rate than our yearly model.

Model 2 - Monthly

Data Preparation

For our second iteration we modeled growth on a month-by-month basis by measuring how the number of reviews in a given month changed from one month to the next. We defined a "viral" restaurant as one with the top 7.41% of monthly growth in restaurants (the specific amount was due to the discrete nature of our target variable). Growth was assigned to the first month to indicate how the performance of the given month impacts growth into the next month.

During the creation of our yearly model we discovered quality issues with the Yelp "Restaurants" category we were using to segment relevant food-related businesses. So, for our second iteration we manually reviewed 1,312 unique business categories and extracted 245

relevant food-related tags. We then defined a "Restaurant" as any business containing at least one of these tags.

We then built a small but descriptive set of features we believed would be indicative of how the business was growing in any particular month. The features constructed were:

- **Percent Growth** - what percent of all reviews of a given business (up to the given date) were in the given month, indicating if the business is getting an unusually high amount of reviews given it's previous popularity
- **Star Growth** - how did the average star rating change from the previous month, indicating if the restaurant is suddenly improving in quality and is rated higher on Yelp
- **Weighted review growth** - the average number of reviews for a given restaurant weighted by "useful," "funny," and "cool" reactions measured from the previous month to the current month. We reasoned that review reactions could indicate higher quality reviews, which are more likely to reach and resonate with people, and that quantifying this would help us capture a truer sense of the increased "buzz" or "interest" a restaurant may be receiving.
- **Trendiness** - to leverage the Yelp-provided categories feature, we built a "trendiness" measure to predict how popular a certain restaurant would be in a given month based upon what other restaurants had high popularity in the same month. To do this we ranked the most popular categories for each month and selected the top 100 categories. From there, we compared the categories of a given business in a certain month to the top 100 of that month, and computed a score for how many categories overlapped.
- **Weighted Tips** - number of tips (records) for a given month and business ID weighted for every "like" the tip received, matched to businesses at specific time intervals. We hypothesized this could help us predict virality as the number of tips a business receives in a given month could be a good indicator of the amount of attention a restaurant is receiving.

To ensure no data leakage, these features used information from the month of interest as well as information from the previous month, whereas our target variable used the difference between the current month and the next month. In other words, we're using information from two adjacent months to predict the performance in the third month and never leveraged data from the future for our predictions.

Model Creation and Evaluation

As we did for our yearly model, we chose Decision Tree as our baseline for our monthly model. In addition to using the same classifiers as our yearly model (Decision Tree, SVM, Logistic Regression), we also tried the following classifiers to see what worked best for our problem:

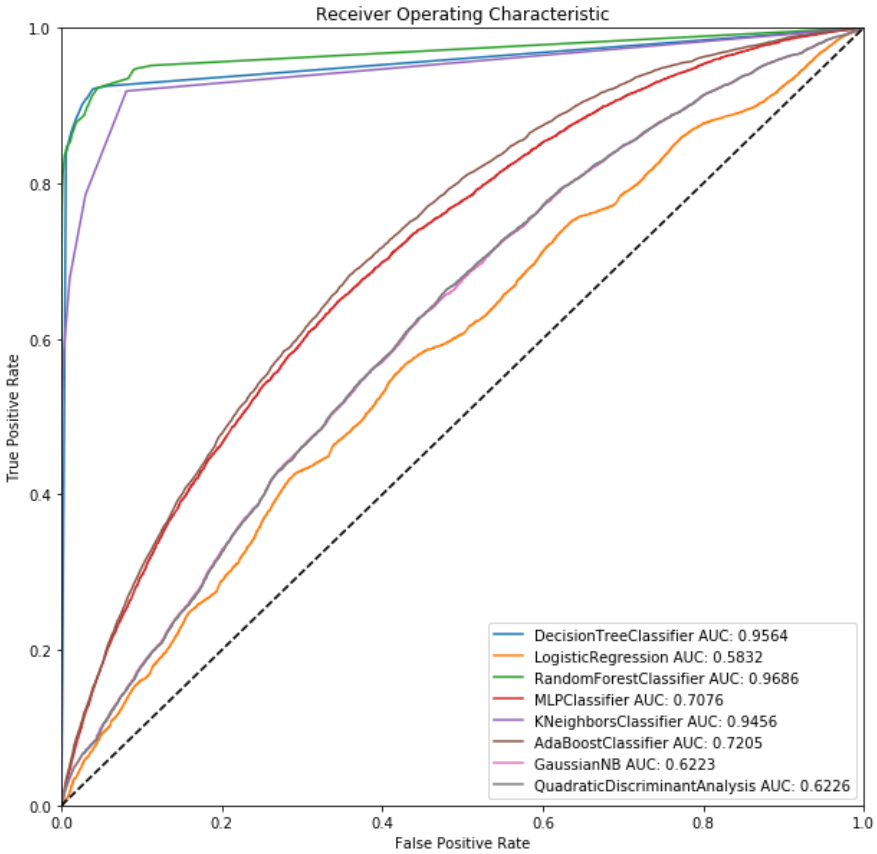
- **Random Forest** - an ensemble form of the decision tree classifier that computes the mode of a set of decision tree outputs trained over subsets of the training data to alleviate the decision tree classifier tendency to overfit. Hyperparameters of interest are the number of trees we build, as well as the same hyperparameters as the decision tree classifier (leaf size bounds, maximum tree depth).
- **Multilayer Perceptron Classifier (MLP) -** a single-hidden layer neural network using logistic regression that transforms input data into a linearly separable hidden layer where

we can easily compute weights for each feature vector corresponding to weight labels, which is passed to an output vector.

- **K-Nearest Neighbors Classifier -** simple lazy-learning classifier that groups data in parameter-space to assign labels based on grouping and distance to neighbors. Hyper parameters of interest are the leaf size of the group-generating trees.
- **Adaptive Boosting Classifier (AdaBoost) -** an adaptive boosting classifier takes the sum of several weak learning algorithms and weighs the result by the error of the weak learner. The hyperparameter of interest was the number of learners used in the computation of the output.
- **Gaussian Naive Bayes Classifier -** Naive Bayes uses the same classification method we've used before (estimating the individual probability of each vector as an independent assumption) and computing the probability of the class label. This type assumes a Gaussian distribution of each parameter (more appropriate than a Bernoulli Naive Bayes implementation).
- **Quadratic Classifier -** computes a quadratic surface that best separates our two classes. It conveniently has no hyperparameters to tune.

We again chose AUC score as our evaluation metric for our monthly model. To validate our results, we decided to compute these metrics over a stratified 5-fold cross validated set. We also utilized grid search to tune our hyperpara
meters. The results are as follows:

Classifier	**AUC (5-Fold)**	**AUC STD (5-Fold)**	**Optimal Parameters**
Random Forest	0.9686	0.0194	Min leaf samples = 10
Decision Tree	0.9564	0.0138	Min samples split = 2
K-Nearest Neighbors	0.9456	0.014	Number of neighbors = 10
AdaBoost	0.7205	0.0198	Learning rate = 1
Multilayer Perceptron	0.7076	0.0198	Hidden layer size = 100
Quadratic	0.6226	0.0189	Priors = None
Gaussian Naive Bayes	0.6223	0.0186	Priors = None
Logistic Regression	0.5832	0.0159	C = 0.001, penalty = L2



Our Random Forest Classifier had the best AUC, likely due to the flexibility of the ensemble sampling method over the dataset, both in the cross validated subsets and in the total training/testing set. Examining our results further, Random Forest outperformed our baseline Decision Tree, the second best algorithm, by an AUC of .012. Compared to Decision Tree, Random Forest adds complexity and is less interpretable for our stakeholders; because it does not add a significant benefit over Decision Tree and our business problem requires we run our model on a large dataset, our recommendation to our stakeholders would be to use Decision Tree as our final classifier.

As we hoped, our monthly model successfully predicted restaurant virality in a much shorter window of time than our yearly model. Considering that we set out to quickly and automatically identify "viral" restaurants, our monthly model solved our business problem much better.

Deployment

Although our model was trained on restaurants in Arizona, based upon our business problem we expect we will need to generalize to provide recommendations for other regions as well. Our model should be deployed as a user-friendly internal tool for food writers. We would recommend to our stakeholders that this tool:

- Automatically collect and store new review data daily to generate up to date restaurant recommendations
- Automatically separate data by location such as State or Zip Code when creating models and flagging viral restaurants as different locations will have different trends

- Contain filters such as locations of interest and cuisines the user can leverage based upon the type of article being written
- Return restaurant-level information such as name, location, and trendiness ranking

Future improvements

For our initial iterations we focused on ratings growth and scores, and ignored the review and tip content itself. In the future, we should try to incorporate this information to see if we could further improve our model's ability to predict viral new restaurants.

We used classification models to predict restaurant virality. In the future we might consider switching instead to regression models to rank predicted growth for each restaurants as classification thresholds are somewhat arbitrary.

Risks

Our final model requires a minimum of two months of data in which a restaurant is open and actively receiving reviews before it can be flagged for our writers. We set out to quickly and automatically identify emerging trends, but in practice solely using this model to identify restaurants and trends to write about could cause our publication to feature some subjects more slowly than we would have otherwise. For example prior to opening some restaurants such as an additional restaurant opening for a popular chef might generate buzz months before its opening. In our case however our model could not flag the restaurant until at least 60 days after it opens. This risk could be mitigated by using our model to supplement, rather than replace, the current efforts of our review and food writing team. We might also consider supplementing our review data with additional data sources such as local news websites that do report on upcoming restaurant openings.

We naively assumed the Yelp-provided features and categories were accurate. In practice however, we discovered quality issues with the provided tags. For example, we saw several CVS pharmacy locations tagged with a "Restaurants" category. In a production scenario, we would have to vet the categories more carefully and correct where appropriate to ensure business relevance.

Ethical considerations

Restaurant reviews are public and published with the intent to share with others, so there are no privacy concerns regarding our data or model. The legality of using Yelp review data on an ongoing basis outside of the academic challenge dataset, however, is questionable. Yelp terms of service specifically prohibit scraping and storing any site content as well as commercial use of the data. In a production scenario where we would need to continually pull and store new reviews across a broad region to then feed into our model, these terms would be problematic. To mitigate this we might consider alternate data sources such as other review sites or paid social listening tools.

Code:

https://github.com/ChrisIck/DS101_Project

References:

PyData. pandas: powerful Python data analysis toolkit, 8/5/18. Web, 11/1/18. http://pandas.pydata.org/pandas-docs/stable/
SciPy.org. Numpy Reference, 7/24/18. Web, 11/1/18. https://docs.scipy.org/doc/numpy-1.15.0/reference/index.html
Scikit Learn. sklearn.naive_bayes.BernoulliNB. Web, 11/1/18. https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html
Scikit Learn. sklearn.linear_model.LogisticRegression. Web, 11/1/18. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
Scikit Learn. 1.10. Decision Trees. Web, 11/1/18. https://scikit-learn.org/stable/modules/tree.html
Scikit Learn. 1.4. Support Vector Machines. Web, 11/1/18. https://scikit-learn.org/stable/modules/svm.html
Matplotlib. Matplotlib.pyplot, 11/11/18. Web, 12/2/18. https://matplotlib.org/api/_as_gen/matplotlib.pyplot.html
Foster Provost & Tom Fawcett. Data Science for Business, First Edition. O'Reilly Media, Inc., July 2013. Web, 9/1/18.