Hygene Prediction
(Analysis of Yelp dataset)

1. Feature Engineering – Review text

The dataset contains full, concatenated text of the reviews. To convert them into features, I combined 4 text representation techniques: unigrams, bigrams, trigrams and 4-grams. First, I used a tokenizer to split text into sentences (to avoid creating n-grams across sentences). Then I removed all punctuation, removed stopwords, and applied Porter stemmer to each word. After that, I generated the 4 text representations in the form of document-term matrix. For training data, this resulted in 822,800 features. After removing n-grams that were too frequent (in more than 80% of the cases), and not frequent enough (single occurrences), I ended up with 58,192 features (where each feature is the number of occurrences of corresponding n-gram in reviews of given restaurant).

2. Feature Engineering – Additional features

- Cuisines

Training data set contains 68 cuisines. I added to the feature matrix additional feature corresponding to each of these cuisines, with value 1 if the restaurant offered this cuisine and 0 otherwise

- Reviews and Rating

I also added two additional features – number of reviews and average review rating

- Zip codes

For zip codes I used a different approach. Test data contained zip codes not present in training data, but adjacent to zip codes in training data. So, instead of creating features containing 1 for matching zip code and 0 for non-matching, I encoded distances: I found public data with latitude and longitude of each zip code, calculated distance between each pair of zip codes, and created a feature for each zip code in the data. The feature either 0 (if the restaurant was in this zip code), or a distance to feature's zip code from restaurant's zip code.

If I had latitude and longitude of each restaurant (like we had in the data set we used for previous tasks), then instead of calculating geographical distance I would have used Google Maps API to calculate driving and walking distances between restaurants instead, since you can often have situations when two restaurants in the same zip code are actually far apart, two restaurants in adjacent zip codes are actually across the street from each other, and two restaurants with small geographical distance between them are actually separated by a river, ravine, etc. and should not be considered close to each other. Missing this data, I stuck with geographical distances between zip codes.

3. Predictive methods used, and their results

I tried the following classification methods:

- Logistic Regression
- Support Vector Machines
- K Nearest Neighbors
- Multinomial Naïve Bayes
- Random Forest
- Gradient Boosting (aka Gradient Boosting Regression Trees)

Additionally, I used PCA to reduce the number of features to 500 principal components, and repeated a subset of the set of methods above on this new dataset (Logistic Regression has shown such poor results that I decided not to repeat it, and the implementation of Multinomial Naïve Bayes that I used did not work with negative values in some of the principal components).

For SVM, KNN, and Multinomial Naïve Bayes I used grid search with parameter variation and 3-fold cross-validation to find the best set of parameters for the model.

For all methods, I used 10-fold cross-validation to make preliminary estimate F1 on the testing data, additional test on 10% of the training data I held out as test set, and actual F1 from submission of prediction on test data as the final grade for the method.

Method	without PCA	with PCA
Logistic Regression	0.540943369886	N/A
SVM	0.546466897674	0.549156176476
KNN (k=10)	0.541819757481	0.543406876048
Multinomial Naïve Bayes	0.546978898362	N/A
Random Forest	0.557377544221	0.529792996762
GB	0.554302238913	0.53728487075

4. Analysis of results

Logistic regression did the worst, and that's not a surprise, it's not a good method for detecting anomalies, with no clear boundary, which is what we're doing in this task.
KNN did a little better, but still not too well – the more dimensions (features) there are for distance calculation, the more data you need to train KNN well, and it has to be fairly dense data too, otherwise it will have a problem differentiating distances within same class from distances between classes. Not surprisingly, with dimension reduction from PCA, KNN showed an improvement.
SVM and Multinomial Naïve Bayes have shown very comparable results. For SVM, after a grid search to optimize parameters, I used rbf (radial basis function) kernel with c=100 and gamma=0.001. Using a non-linear kernel clearly helped. In MNB, to help the algorithm perform on unbalanced test set, I set a prior with probability 90% for passing an inspection and 10% for failing. SVM is somewhat sensitive to situations when number of features is greater than number of cases, so there is an expected improvement after applying PCA.
Random Forest and GB are my go-to methods for classification problems and I expected them to perform best – exactly what happened. Given that both methods have built-in identification of most important features (random forest does feature pruning for every tree it builds, based on subset of cases and features selected for that tree, and GB also prunes features as it builds a shallow tree at every iteration to optimize the loss function). For that reason, both did better with the original set of features than with PCA-reduced set of features.

I looked at actual cases that got misclassified in my held out test set, and compared them to cases that got classified correctly. What I noticed was that most correctly classified cases had either at least one review that mentioned something negative about food quality or cleanliness of the place, or had a combination of the restaurant's cuisine being among several cuisines that had high percentage of places that did not pass inspection, and the restaurant was located in the zipcode with high percentage of places that did not pass inspection (or in zipcode adjacent to it). The cases that got misclassified seemed to have no indication in the reviews or in cuisine and location that there was a risk of failing inspection.

Just to see importance of text features compared to cuisines, ratings and geographical location, I built additional Random Forest model using only the additional features (no text features), and got F1 of 0.531039068474; I also built additional Random forest model using only the text features, and got F1 of 0.547831189377.

I see some possibilities for improving the results further from using LDA on n-grams create clustering-based features, and also to do co-occurrence frequency calculation for cuisines (which cuisines occur together in the same restaurants) to create features that group together cuisines.

5. Tools Used

Data processing: Python with nltk and sklearn libraries. Specifically:

- Stemming: nltk.stem.porter.PorterStemmer
- Sentence tokenization: nltk English punkt tokenizer
- N-gram tokenization: sklearn.feature_extraction.text.CountVectorizer
- Zip code coordinates lookup and distance calculation: https://github.com/cmhulett/zipcode_distance
- Holdout test set: sklearn.cross_validation.StratifiedShuffleSplit
- Model parameter search: sklearn.grid_search.GridSearchCV
- Logistic regression: sklearn.linear_model.LogisticRegression
- SVM: sklearn.svm.SVC
- KNN: sklearn.neighbors.KNeighborsClassifier
- Multinomial Naïve Bayes: sklearn.naive_bayes.MultinomialNB
- Random Forest: sklearn.ensemble.RandomForestClassifier
- GB: sklearn.ensemble.GradientBoostingClassifier
- PCA: sklearn.decomposition.PCA