[Highly Classified]

The Manhattan Project
Presents

Yelp Rating Prediction

Rong Feng (rf1316)
Yunan Hu (yh1844)
Josh Meisel (jm7955)
Bing Zou (bz1031)

Table of Contents

1. Business Problem

Restaurant owners and customers would both significantly benefit from a model that makes tailor-made predictions for how well a given diner would like a particular establishment. Such a model will have two immediate use cases. The first would be to offer restaurants a ranked list of users who will likely enjoy their restaurant, which creates value by allowing restaurants to better target their advertising efforts. The second would be to offer users recommendations of restaurants for which they have a high probability of enjoying, leading to improved user experience and value for Yelp in their own market penetration.

For restaurants, recommendation products have significant business value. The restaurant industry is extremely competitive and unforgiving, especially in large metropolitan areas, which happens to coincide with where our model would be most effectively implemented (these markets have more data, and a higher need for a data-driven model as consumers have more choices – including ones they aren't familiar with — and restaurants have to navigate a more complex consumer landscape). One New York restaurant investor describes[1] a world in which only 2.8% of the pre-tip bill goes to operating costs, amounting to less than credit card fees. With these razor-thin profit margins, optimizing marketing effectiveness while keeping seats filled with satisfied customers who are willing to spend money on food they like, tip well, spread reputation via word-of-mouth, and write positive reviews is imperative to keeping the business alive. Employing our model would help Yelp cater to this demand and increase subscription sales.

The demand for food recommendations from diners is clear as well. Food aggregators and reviewers such as Thrillist, Zagat and Yelp are increasingly popular, with many specializing in the overwhelming

---

1 https://www.newyorker.com/business/currency/the-thrill-of-losing-money-by-investing-in-a-manhattan-restaurant

food scene of large cities such as Toronto, from where we trained our models. However, these curated lists all reflect the tastes of the food critic or the herd. Any improvement made by customizing recommendations to users could set apart a business in the space, increasing usership and engagement and therefore increasing its value to the businesses that pay to be listed. If users are shown restaurants that they are more likely to patronize and enjoy, then more restaurants will be willing to pay and to pay more to be shown on Yelp.

2. Data

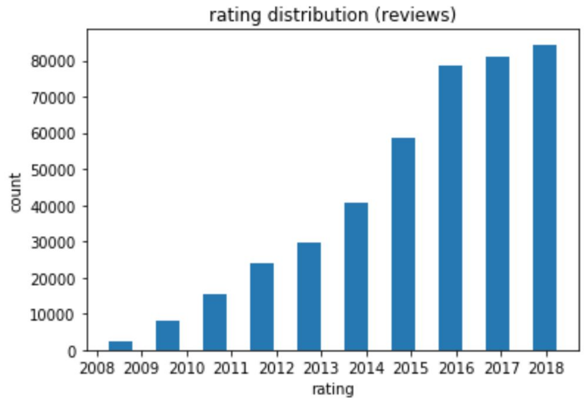
2.1 Yelp Dataset

We use the Yelp dataset for this project. The Yelp dataset[2] is publicly published by Yelp for personal, educational, and academic use. From this dataset we selected 3 subsets corresponding to businesses, users and reviews.

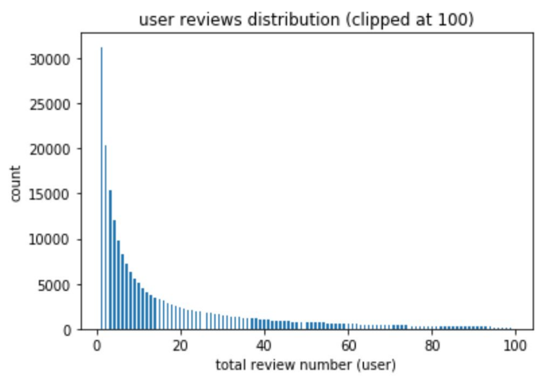
We restricted to restaurants in a single city as a proof-of-concept that can be extended to more cities with additional training on the same model or specific city-level models. We chose Toronto since it was the most represented in the dataset. The business dataset contains 24,453 businesses labeled with cities belonging to the Greater Toronto Area (GTA). We first narrowed this down to 23,482 businesses, removing outliers outside of the expected longitude/latitude range of GTA. We found that restaurants in the database corresponded exactly with businesses tagged with the category of 'restaurant', ‘bars’, or ‘bakeries.’ Filtering on this gave us a final total of 10,914 restaurants, 93,075 users, and 422,790 reviews.

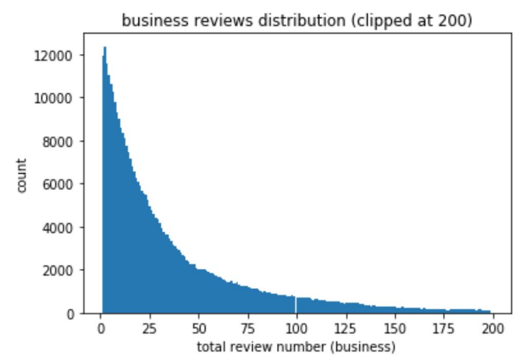
Reviews range from Jan 3, 2018 to July 2, 2018. Most reviews are from recent years.

[2] https://www.yelp.com/dataset/



Most users and business only have a few reviews, which might harm our prediction performance.





2.2 Dinesafe Dataset

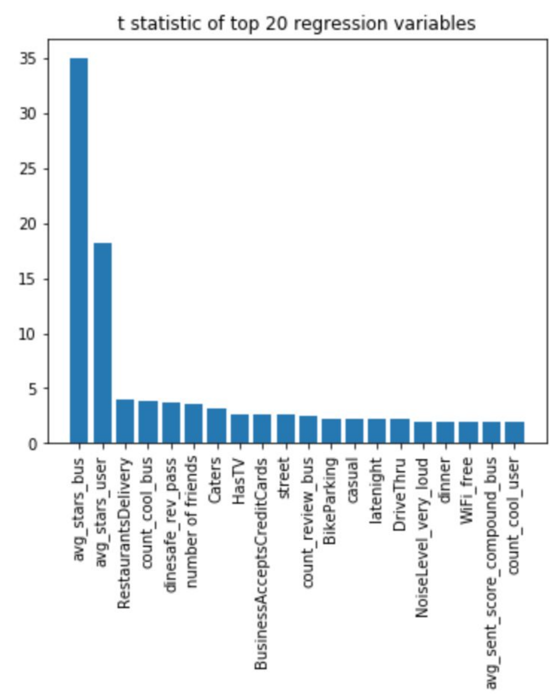
We theorized that users' preferences for perceived cleanliness of restaurants would widely vary (include a possible predilection towards more authentic-seeming hole-in-the-wall establishments), and thus searched for a complementary dataset with these potentially highly predictive features. The city of Toronto has an office that enforces sanitary conditions in the restaurants that fall within the city bounds, and openly publishes a dataset on their website[3]. Since the datasets do not share a common key, we needed to explore other ways of joining the datasets. We found that fuzzy string matching using levenshtein distance on both the business name and the address achieves the best mapping results, with roughly 50% of our Yelp dataset mapping to a dinesafe record, as opposed to ~10% for exact

[3] https://www.toronto.ca/city-government/data-research-maps/open-data/open-data-catalogue/health/

matches. From the dinesafe dataset, we extracted several dummy variables based on health score and severity of infractions. For the full list of features, refer to appendix B.

2.3 Feature Engineering

Preparing our data involved a number of engineered features. The most obvious features are average star ratings of the business and user, and we correctly expected these to be the most important (see the figure below). As a result we chose for our baseline model simply predicting the average stars of the business (which unsurprisingly performs better than the user's average stars). We compute the average rating based only on previous reviews in order to avoid data leakage, and to have the distribution of training data match production data, which of course does not have access to future reviews. We also only used the previous two years, controlling for possible shifts in restaurant quality and user taste.



We include other numerical aggregations, such as review counts and the total number of times reviews are labeled by other users as cool, funny, or useful, for both each user and each restaurant. We include restaurant-level binary variables for each category and attribute (excluding the rare ones), and categorical information such as restaurant attire.

We use NLP techniques to extract numerical features from the raw text of the reviews. Based on the assumption that users will give a higher score to those restaurant that similar users rated highly, we generated different features measuring the similarity of the language between reviews of a specific user and reviews of a specific restaurant. In order to do this, all non-stopword words in the reviews are first mapped to their corresponding FastText 300 dimension word vectors[4]. Then for each review, we compute the review vector embedding by downweighting the more frequent words[5] and computing the weighted average word vector. We do this methodology on the set of all words, all nouns, verbs and adjectives and arrive at 4 different review embeddings per review. For each review, we compute the average business review embedding for the set of reviews on that business excluding the current review, and the same for the user associated with the review. Finally, we compute the cosine similarity between the 4 business vector representations with the 4 user vector representations to get 4 numerical features. Note that the business vectors and user vectors are unique to each review as we are excluding just that review from the representation.

We also extract NLP features on sentiment, counting positive word count minus negative word count normalized by dividing the total word count of the review. In addition to the word-based net sentiment score, we calculate the compound sentiment score for each review by the polarity_scores from the SentimentIntensityAnalyzer[6] in NLTK library. We also directly measure the intensity of emotions conveyed by the review using punctuation counts, excluding commas and periods. Finally, we extract

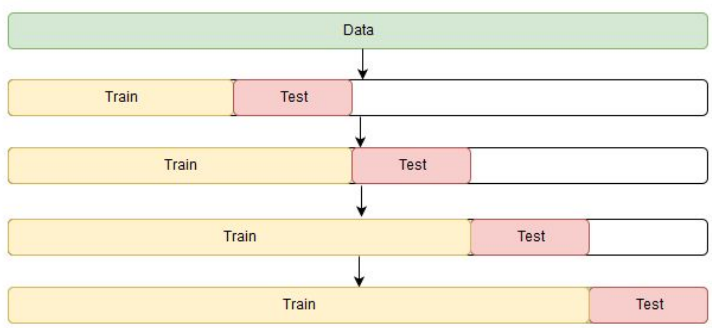
[4] https://fasttext.cc/docs/en/english-vectors.html
[5] A Simple but Tough to Beat Baseline for Sentence Embeddings, https://openreview.net/pdf?id=SyK00v5xx
[6] Gilbert, CJ Hutto Eric. "Vader: A parsimonious rule-based model for sentiment analysis of social media text." In Eighth International Conference on Weblogs and Social Media (ICWSM-14). Available at (20/04/16) http://comp. social. gatech. edu/papers/icwsm14. vader. hutto. pdf. 2014.

readability features by measuring review length and average word lengths. Again, we only use available data from before the review.

2.4 Train-Test Split and Cross Validation

Given our time series data, we need to be aware of look-ahead bias when splitting the data into training test sets. To do this, we first sort the data date chronologically, then set the 90 percentile date as the splitting point, which is July 05, 2017.

For cross-validation, we split the training set at three points in time, training on the data before the split and validating on the fold after the split (see the image below).



[Image from StackOverflow.com]

3. Models

After formulating our problem as a regression problem, we cross-validated a handful of algorithms from the linear space, tree space and neural network space. We chose models that could be trained efficiently given the deadline, and opted out of slower algorithms like KNN and SVM that require distance calculations over the whole dataset. We included neural networks despite their substantial training time given their effectiveness, which has been demonstrated recently across a number of applications.

3.1 Feature Selection

In total, we have 114 features. To reduce the complexity and increase the generalizability of the model, we first select the top relevant features based on random forest regression. Among the most important features, we notice several important features which match our intuition, for example, average business ratings, average business sentiment scores (compound), cosine similarity (adjectives), cosine similarity(noun). In the end, we decided to keep only the top half of features ranked by importance in our final model that was tested against the test set.

3.2 Evaluation Metric

The models are scored using Mean Squared Error (MSE) in order to heavily penalize highly inaccurate predictions, which would be very detrimental to Yelp's business goals.

3.3 Baseline Model

We use the trailing average review of the business as the baseline model, which is how current Yelp users can glean predictions from the app. The baseline achieves a MSE of 1.47 on the training data.

3.4 Linear Models

We trained Logistic Regression, Ridge, Lasso and Elastic Net models on the data and ran time series cross validation with 3 periods. For logistic regression, since the resulting predictions are discrete classes rather than a continuous variable, we use it to compute class probabilities and take the expectation over that probability density as the predicted continuous score.

3.4.1 Logistic Regression Results

We search over the C parameter for values of (0.001, 0.01, 0.1, 1, 10). C = 0.001, achieved the best MSE, at 2.11. Since this is no better than our baseline, we don't think this class of models will perform well.

3.4.2 Ridge Regression Results

For the Ridge alpha hyperparameter (which penalizes the L2 size of the weights), we use the same parameter set as the C parameter of logistic regression. The best result is achieved at an alpha value of 10, with the MSE loss 1.44, beating the baseline model.

3.4.3 Lasso Regression Results

Empirically, Lasso has a hard time converging at small values of alpha (which controls the L1 regularization), so we search over a slightly different set of alpha parameters: (0.01, 0.05, 0.1, 1, 10). Lasso also achieves the best cross validation results of MSE loss 1.44, but at an alpha value of 0.01.

3.4.4 Elastic Net Results

Elastic Nets attempts to solve the limitations of Ridge and Lasso regression by combining both L1 and L2 regularization. For simplicity, we search over cases were the 2 alphas are equal, and sums to the alpha_sum set: (0.01, 0.05, 0.1, 1, 10). The best MSE loss achieved was 1.43 using alpha = 0.01.

3.5 Ensemble methods

We tested ensemble-based regression methods, as they often are highly successful non-deep learning methods. By combining different predictions, they improve underlying predictions, boost stability, and control overfitting. We cross-validated Adaboost, Bagging, Extra-tree, Gradient tree boosting, and random forest Regressors. We use random grid search[7] with 3-fold time series cross validation. Our best model is Gradient tree boosting with an average MSE of 1.40 on validation set.

[7] http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf

3.5.1 Bagging Regression Results

We searched over n_estimators for values of (5, 10, 20), max_samples for values of (0.5, 0.75, 1) and max_features for values of (0.5, 0.75, 1); the best result is achieved at n_estimators = 20, max_samples = 0.5, and max_features = 0.5, which gives us MSE = 1.51.

3.5.2 Random Forest Regression Results

We used the same hyperparameter set as in bagging; the best result is achieved at n_estimators = 10, min_samples_split = 4, and min_samples_split = 1, which gives us a MSE of 1.55.

3.5.3 Extra-trees Regression Results

Extra trees is similar to random forest, using bootstrap subsampling to fit many decision trees. Extra-trees model differ in that they do not try to select the optimal cut-off point for each split.

With the same hyperparameter set again, the best result is achieved at n_estimators = 20, min_samples_split = 4, and min_samples_split = 2, yielding a MSE of 1.49.

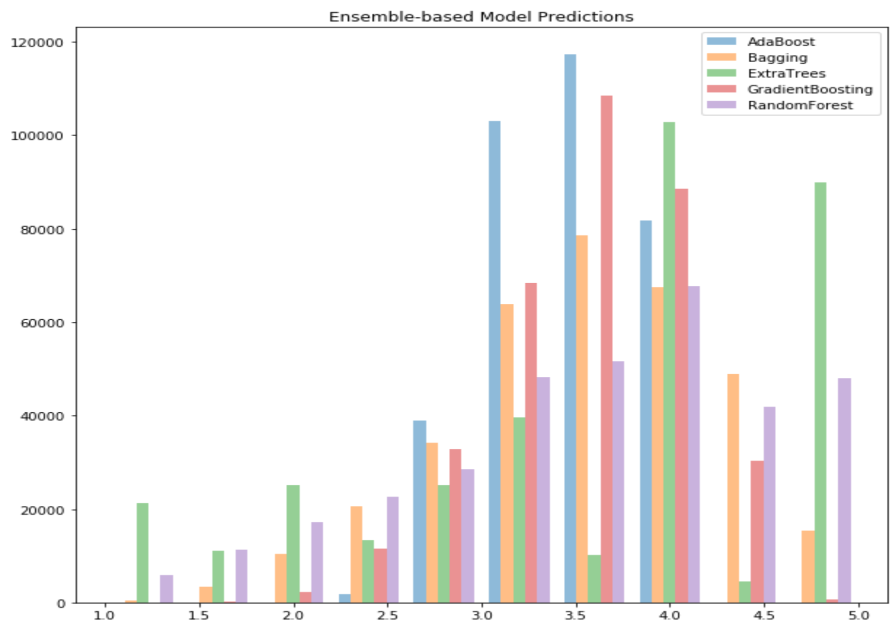
3.5.4 Gradient Tree Boosting Regression Results

We modified the hyperparameter set for n_estimators to (50, 100, 150). The best result is achieved at n_estimators = 100, min_samples_split = 6, and min_samples_split = 1, which gives us MSE = 1.40.

3.5.5 Adaboost Regression Results

Adaboost regressor is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction. As such, subsequent regressors focus more on difficult cases.

We searched over n_estimator values of (25, 50, 75) and learning_rate values of (0.1, 0.5, 1); the best result is achieved at n_estimators = 50 and learning_rate = 1, which gives us a MSE of 1.45.



Summary of all the traditional Machine Learning models

Model	Cross Validation MSE
Logistic Regression	2.11
Ridge Regression	1.44
Lasso Regression	1.44
Elastic Net Regression	1.43
Adaboost Regression	1.45
Bagging Regression	1.51
Extra-trees Regression	1.49
Gradient tree boosting regression	**1.40**
Random forest regression	1.55

3.6 Neural Networks

We trained neural net models as well, hoping to find complex non-linear relations in the data. We used as hyperparameters:

1) Activation (relu, tanh, and logistic sigmoid) 2) Solver (stochastic gradient descent, Adam[8] -- a generalization of sgd which adds a momentum term based on a running average of recent gradient magnitudes and their squares in order to speed up convergence, and lbfgs[9], or Limited-memory BFGS, which uses approximations of the second-order derivative) 3) learning rate schedule (constant, inverse scaling -- where the learning rate decreases by a constant ratio, and adaptive -- where the learning rate decreases 5-fold whenever the training loss does not decrease enough) 4) batch size (irrelevant for lbfgs) (100, 200, and 300) 5) size of the hidden layers -- we used the rule of thumb of decreasing the size of deeper hidden layers, allowing for between 1 and 10 hidden layers

As these models took longer to train, we did not use time series cross validation during hyperparameter tuning, but rather split the last 10% of the training data into a validation set (the baseline achieved 1.57 MSE on this set). We randomly selected hyperparameters to test 61 different models. Then we did time series cross validation on the best-performing model on the validation set.

MSE ranged from 1.48 to 1.723 (excluding models without any hidden layers), with the best model coming from activation=relu, solver=lbfgs, alpha=1e-5, batch_size=200, learning_rate=invscaling, hidden_layer_sizes= (81, 61, 41, 21). The time series cross validation loss was 1.68.

[8] Diederik, Kingma; Ba, Jimmy (2014). "Adam: A method for stochastic optimization". arXiv:1412.6980.
[9] Liu, D. C.; Nocedal, J. (1989). "On the Limited Memory Method for Large Scale Optimization". Mathematical Programming B. 45 (3): 503–528. doi:10.1007/BF01589116.

3.7 Summary

Gradient Boosting regression performs the best on the time series validation. It achieves a MSE of 1.42 on the test set, compared to 1.55 for our baseline. We also tested our model's accuracy in predicting whether a user will rate a restaurant higher or lower than that restaurant's overall rating. We get an accuracy of 58%.

4. Deployment and Monitoring

Circling back to the two used cases indicated in the opening paragraph, we can deploy our model in the same two ways: recommending users to restaurants and vice versa. For user recommendations, we can generate user lists for each restaurant in our universe and sell the lists to the restaurants for marketing purposes. We can even create marketing programs directly in our platform to create further value for business to be on our platform. For restaurant recommendations, this would be used to improve our bread and butter: restaurant searches. The model can be integrated as part of the restaurant search engine to move better matched restaurants higher in the user search results, resulting in more relevant results and a happier user experience.

In addition, there are some other issues that Yelp should keep in mind:

4.1 Legal and ethical issues

All dataset usage agrees to Yelp Dataset Terms Of Use. Yelp should not deceptively manipulate its service to favor companies based on payments made to Yelp, which will result in a credibility issue. Also, Yelp should monitor the predictions to see whether the recommendations are unfairly assigned to certain subpopulations, for example, people who live in ethnic neighborhoods. However we don't think this is a big issue for food.

4.2 Data availability

Since the Toronto government publishes the health inspection score every two years, it will prohibit us from making real time predictions using health score, but luckily the health scores shouldn't vary drastically day-to-day.

4.3 Scalability

Considering that the Yelp open dataset only contains a subset of reviews and features, a production-level model would need to obtain more complete data, which should aid in the prediction task.

4.4 Selection bias

Our model only looks at users who have at least one review on Yelp, and are more heavily weighted to users that review more. This could introduce bias if the behaviour of frequent reviewer are fundamentally different from the general population.

4.5 Covariate shift

Since the business and user distributions might change over time, it is recommended to retrain the model every time when there is a major drop in performance or whenever a new helpful feature has been available on Yelp.

4.6 Look-ahead bias

Also, although we try to exclude the look ahead bias by using time series split and generating features by using only data available before the review date, however, we do not have such historical business-level data, such as if the business at one point was assigned different categories and attributes. In deployment, Yelp should keep track of these changes.

Code and Data

All the code and data is available on github:

https://github.com/nyumanhattanproject/ds1001_proj.git

Appendix A - Contribution

Rong Feng	NLP feature extraction - word vectors and sentence vectors, joining dinesafe dataset using fuzzy string matching, linear models, report writing and editing
Yunan Hu	NLP feature extraction - numerical features; transform review feature into business level and user level features; train-test splitting; ensemble-based models running (with Bing); report writing and editing
Josh Meisel	Business feature extraction, user feature extraction and attributes feature extraction (with Bing); joining the data; neural net models; report writing and editing
Bing Zou	Business feature extraction, user feature extraction and attributes feature extraction (with Josh); ensemble-based models running (with Yunan); report writing and editing

Appendix B - Feature Dictionary

Feature Column	**Feature Description**
cos_sim_all	cos similarity between vectors of all words between business and user
cos_sim_noun	cos similarity between vectors of all nouns between business and user
cos_sim_adj	cos similarity between vectors of all adjectives between business and user
cos_sim_verb	cos similarity between vectors of all verbs between business and user
dinesafe_rev_closed	dummy var for inspection result category - closed
dinesafe_rev_condpass	dummy var for inspection result category - conditional pass
dinesafe_rev_pass	dummy var for inspection result category - pass
dinesafe_status_crucial	dummy var for inspection severity - crucial
dinesafe_status_minor	dummy var for inspection severity - minor
dinesafe_status_na	dummy var for inspection severity - not applicable
dinesafe_status_significant	dummy var for inspection severity - significant
business_id	The ID of business
date	Date of reviews

stars	The target variable
text	Review text
user_id	ID of user
count_review_bus	total number of reviews of business
avg_stars_bus	Average rating of business
count_funny_bus	total number of reviews marked as funny of business
count_cool_bus	total number of reviews marked as cool of business
count_useful_bus	total number of reviews marked as useful of business
avg_sent_score_compound_bus	average compound sentiment score of business
avg_sent_score_net_bus	average net sentiment score of business
avg_review_length_bus	average review length of business
'avg_punc_count_bus	average punctuation counts of business
avg_word_len_bus	Average word length of business
count_review_user	total number of reviews of user
avg_stars_user	Average rating of user
count_funny_user	total number of reviews marked as funny of user
count_cool_user	total number of reviews marked as cool of user
count_useful_user	total number of reviews marked as useful of user
avg_sent_score_compound_user	average compound sentiment score of user
avg_sent_score_net_user	average net sentiment score of user
avg_review_length_user	average review length of user
avg_punc_count_user	average punctuation count of user
avg_word_len_user	average word length of user
RestaurantsPriceRange	price range of restaurants
Alcohol	whether alcohol is provided
BusinessAcceptsCreditCards	whether credit cards are accepted
GoodForKids	whether good for kids
OutdoorSeating	whether there is outdoor seating
RestaurantsAttire	whether there is a dress code

RestaurantsGoodForGroups	whether good for adults
RestaurantsTableService	whether has table service or not
RestaurantsTakeOut	whether has take out
BikeParking	whether has bike parking
Caters	cater or not
DogsAllowed	whether dogs are allowed
HasTV	whether has TV
NoiseLevel	level of noise
RestaurantsDelivery	Whether it delivers
RestaurantsReservations	Whether take reservations
WheelchairAccessible	Whether wheelchair is accessible
WiFi	Has wifi or not
CoatCheck	Whether check coat
GoodForDancing	Good for dancing or not
HappyHour	Whether has happy hour
Smoking	Whether smoking is allowed
DriveThru	Whether it has drive through
ByAppointmentOnly	By appointment only or not
BestNights	Time of the best nights
Open24Hours	Open 24 hours or not
AgesAllowed	Ages allowed
HairSpecializesIn	Hair specialization
DietaryRestrictions	Dietary restrictions
RestaurantsCounterService	Whether has counter service
BusinessAcceptsBitcoin	Whether accept bitcoin
AcceptsInsurance	Whether accept insurance
casual	Ambience
classy	Ambience
hipster	Ambience

intimate	Ambience
romantic	Ambience
touristy	Ambience
trendy	Ambience
upscale	Ambience
garage	Parking
lot	Parking
street	Parking
valet	Parking
validated	Parking
breakfast	Whether good for breakfast
brunch	Whether good for brunch
dessert	Whether good for dessert
dinner	Whether good for dinner
latenight	Whether good for late night
lunch	Whether good for lunch
background_music	Whether has background_music
dj	Whether has dj
jukebox	Whether has jukebox
karaoke	Whether has karaoke
live	Whether has live music
no_music	Whether has music
video	Whether has video
compliment_cool	number of cool compliments received by the user
compliment_cute	number of cute compliments received by the user
compliment_funny	number of funny compliments received by the user
compliment_hot	number of hot compliments received by the user
compliment_list	number of list compliments received by the user
compliment_more	number of more compliments received by the user
compliment_note	number of note compliments received by the user

compliment_photos	number of photos compliments received by the user
compliment_plain	number of plain compliments received by the user
compliment_profile	number of profile compliments received by the user
compliment_writer	number of writer compliments received by the user
cool	number of times the user is marked as ‘coo’
fans	number of fans a user has
funny	number of times the user is marked as cool
review_count	number of reviews
useful	whether review content contains ‘useful’
number of years of elite	number of years being yelp elite status
number of friends	number of friends
membershiptime	time being a yelp member