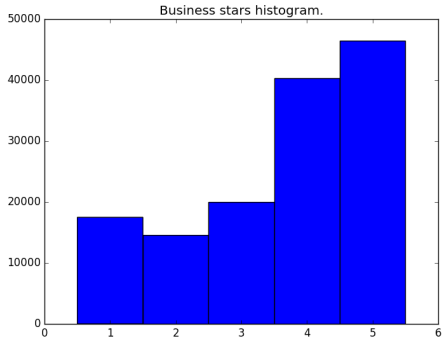
ORIE 4741 Project Midterm Report

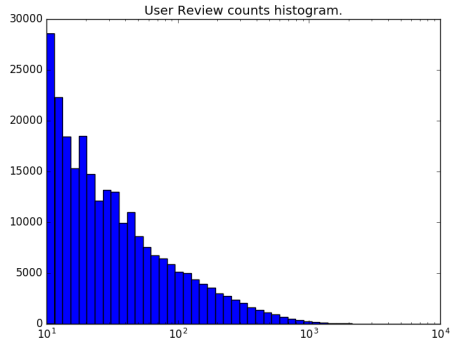
October 28 , 2016

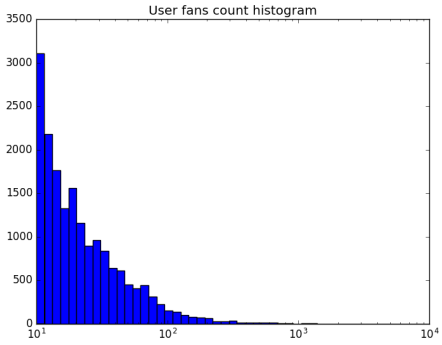
Siyuan Wang (sw884) Mingchen Zhang (mz466) Xuan Zhao (xz544)

1 Data Sets Description

In this project we plan to develop a recommendation system of restaurants for Yelp users based on information from both users and business. From Yelp Challenge website we acquired 5 data sets related to users and business. Two of them, User and Reviews, contain information about the users (User ID, historical compliments, and reviews). A third data set, Business, contains information of the business, such as Business ID, opening hours, categories, etc. The last two two data sets, Check-in and Tip, include the visiting records of customers to a certain business and the comments they left behind. A detailed structure of the data in each data set is listed in Appendix. To visualize the data we have three histograms: the one on the left shows the average score the customers gives to a restaurant, the second one shows the distribution of number of reviews given by customers, and the last one shows the distribution of number of fans the Yelp users have.







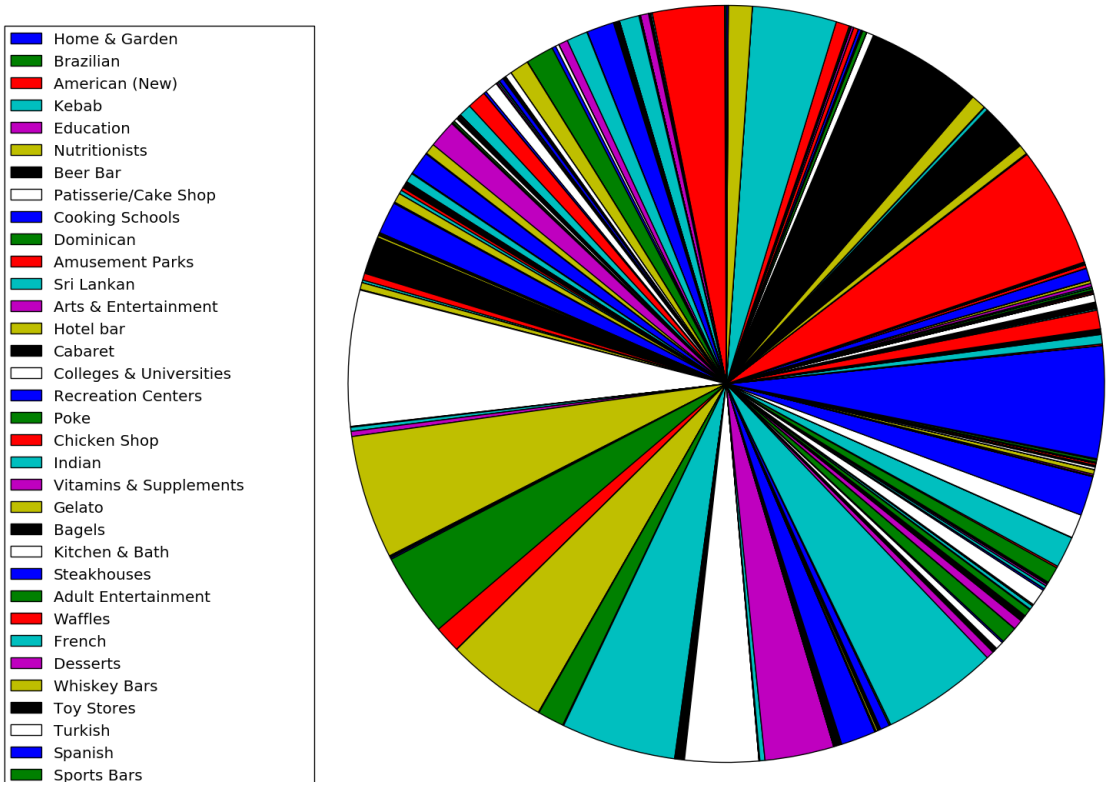
Since our project scope is to generate a customized recommendation list of restaurants to users, we first need to clean the data sets so that they only contain information about customers' visits and comments to restaurants. The raw data initially have 85901 business on Yelp, and we kept 26729 of them that belong to dining business.

After data clearning we start to build out feature matrix X and output matrix Y. The matrix Y is rather straightforward, which would be the review score a customer gives to a certain restaurant he visited before (an integer between 1 to 5), so that our prediction model could predict a score the customer would likely have for a restaurant he never visited and therefore generate a recommendation list to them. The building of feature matrix X is more time-consuming since 1) we have quite a lot of features available in the data sets and 2) we need to select those significant features from them. The selection of features were conducted as follows:

1) We first remove the features that intuitively won't contribute to our model, such as whether the restaurant accept bitcoins, longitude and latitude, or the address.

2) The next step was to fit a preliminary testing model with features left and we found the coefficients for restaurant categories were pretty low since a restaurant normally only belongs to

one specific category (Japanese, Indian, American, etc). To fix the issue we look at the historical visits of the customer and calculate the proportion of each category he visited and when it comes to a new restaurant, the probability that he will like it will be the sum of proportions of categories of this new restaurant, which would be a real number between 0 and 1. In this way we removed all the category features and only this preference estimation feature was kept.



We used a total of 45 features. Nine of them comes from the user: Average star of a user, number of "cool"/"hot"/"more"/"writer" compliment, number of "hot" compliment, number of "more" compliment, number of "writer" compliment, user's fan count, user's review count, user's number of "cool"/"useful" votes. The rest of the features comes from the business information: What type of alcohol do they serve, the ambiance of the restaurant, the attire of the restaurant, etc.

2 How we built our model

- Linear Regression

  Initially, we tried to combine linear regression and Lasso regularization to fit our model.we split our data set into three parts: training set, validation set and test set. In the training set, we construct our linear model by minimizing the error of least square regression. At the same time, to our model to prevent over-fitting. We derived a set of linear models by applying lasso regularization with different λ on them. By comparing the error from those models in the validation set, we picked up the model which satisfied our minimization goal with $\lambda \approx 0.6$. Then we applied this model to test set. However, we found that the output y, which is the predictive stars a customer may evaluate given a certain business, sometimes beyond the scope of stars$[1, 5]$. This is possible since linear regression doesn't enforce the range of output. When training with 40K training data points, we were getting around 15.38635463 in-sample error and 15.83295704 out-of-sample error. This is

way too large for us to make reasonable rating prediction.

- Logistic Regression
  To increase our model's accuracy, we decided to tackle it from a multinomial classification perspective and applied logistic regression to our model to specify which star group a customer may contribute to a given business. We split our data set into just two parts: training set and test set. In the training stage, we used K-fold generator model to split our training set into five subset. That is to say, if we train our model based on one subset, then we need the rest subset to complete cross validation to choose the best regularization parameter. By doing this, we fit the model for logistic regression and then calculate the coefficient matrix W_{46X5} and regularization parameter λ. We used a Python library called "scikit-learn" to parametrize our logistic regression algorithm. We used l2 regularization as part of our model. For cross-validation phase, we made the training algorithm to try out 9 values of λ between $1e-4$ to $1e4$. This python library also produces the output directly instead of the probabilities of each output, which simplified our code as well.

- Validating our model coefficients
  Our first step was to decide the size of our input data matrix. By trying out small input dataset size first, we found out that certain features, such as the "Ambience" of a business being "divey" are incredibly rare to be true. In order for each feature of our input matrix to have at least one data point contains a non-zero value, our dataset has to have at least 40K data points. We have to make sure all features have at least one point that has a non-zero value, since otherwise the standard deviation of that feature will be zero, which will cause issues during normalization(divide by 0). This is something we have to improve in later part of the semester - to either make our matrix less sparse, which means that we have to design our features better. We ended up using 40K reviews to gather our training data set, and then use another 40K reviews to gather our test data set. When testing our model against our training data, we were getting in-sample error of 1.74225 and mean accuracy of 0.4379. When testing our model against the test data set, we were getting out-of sample error of 1.8029 and mean accuracy of 0.46355. It is acceptable to have a lower out of sample error but a higher accuracy in the test set, since accuracy just account for whether or not the predicted value matches the exact value, but not the amount of deviation from the actual value(only the error value account for that). Our next step is to find more ways to improve the accuracy of our model and reduce out of sample error.

3 What we are going to do

Even though we have reached a prediction with acceptable rating prediction. There are still a few steps we need to do to further increase our prediction accuracy.

- Our feature matrix X is still a little more sparse than we expected, this happened for some features do not happens frequently. In the future, we hope to take measures to better represent these 'sparse' feature columns to make a more compact X.

- We currently used logistic regression on our project and we got round 44.6% accuracy for out sample data set. We hope to dig out more useful and practical algorithms to fit our model to enhance our accuracy rate in the future.

Appendix

4 Business

```
    {
        'type': 'business',
        'business_id': (encrypted business id),
        'name': (business name),
        'neighborhoods': [(hood names)],
        'full_address': (localized address),
        'city': (city),
        'state': (state),
        'latitude': latitude,
        'longitude': longitude,
        'stars': (star rating, rounded to half-stars),
        'review_count': review count,
        'categories': [(localized category names)]
        'open': True / False (corresponds to closed, not business hours),
        'hours': {
            (day_of_week): {
                'open': (HH:MM),
                'close': (HH:MM)
            },
            ...
        },
        'attributes': {
            (attribute_name): (attribute_value),
            ...
        },
    }
}
```

5 Review

```
{
    'type': 'review',
    'business_id': (encrypted business id),
    'user_id': (encrypted user id),
    'stars': (star rating, rounded to half-stars),
    'text': (review text),
    'date': (date, formatted like '2012-03-14'),
    'votes': {(vote type): (count)},
}
```

6 User

```
{
    'type': 'user',
    'user_id': (encrypted user id),
    'name': (first name),
    'review_count': (review count),
    'average_stars': (floating point average, like 4.31),
    'votes': {(vote type): (count)},
    'friends': [(friend user_ids)],
    'elite': [(years_elite)],
    'yelping_since': (date, formatted like '2012-03'),
    'compliments': {
        (compliment_type): (num_compliments_of_this_type),
        ...
    },
    'fans': (num_fans),
}
```

7 Check-in

```
{
    'type': 'checkin',
    'business_id': (encrypted business id),
    'checkin_info': {
        '0-0': (number of checkins from 00:00 to 01:00 on all Sundays),
        '1-0': (number of checkins from 01:00 to 02:00 on all Sundays),
        ...
        '14-4': (number of checkins from 14:00 to 15:00 on all Thursdays),
        ...
        '23-6': (number of checkins from 23:00 to 00:00 on all Saturdays)
    }, # if there was no checkin for a hour-day block it will not be in the
       dict
}
```