# The Insight from customer's reviews(Yelp.com)

**Final Report – 2018 Spring - BIA660D**

**Instructor: Zac Wentzell**

**Submission Date: 09/05/2018**

**Abstract:** This project is targeting to discover the business insight from customer's reviews data to support decision making and create values for business owners. To reach this main target, we finished 3 sub objectives to predict the customer's rating given their reviews text, to find out the popular topics customers cared about, and to build a recommendation system to create business value.

Team Member:

ALEC KULAKOWSKI - akulakow@stevens.edu

HE LI - hli81@stevens.edu

XIN LIN - xlin11@stevens.edu

NI MAN – nman@stevens.edu

RICHARD ROMA - rroma@stevens.edu

# I. Introduction (Ni)

## Part 1. Background

Companies care about the insights they could obtain from billions of customer's reviews text. These insights could help them develop their business and improve revenue by elevating the service quality, modifying the product or service, delivering suitable product or service to targeted customers.

This project is targeting to discover the business insight from customer's reviews data to support decision making and create values for business owners. Yelp.com and the Yelp mobile app publish crowd-sourced reviews about local businesses (F&B service, Home service, Dentist etc.), as well as the online reservation service Yelp Reservations. In this case, we focus on the reviews of restaurants.

The insight from these reviews could benefit the social community and support the development of dozens of functional applications like the customer's rating prediction system, the popular topics report as well as the precise recommendation system

## Part 2. Data

We have two different data sources, the website of Yelp.com and the Yelp open source API. We will build a web scraper in Python to collect the reviews data from the website and we will interact with Yelp open source API to collect the big data and metadata.

We will understand and manipulate data for specific objectives incorporating data cleaning, data preprocessing and data preparing.

## Part 3. Sub Objectives

To reach our main target, we will map 3 sub objectives. The first one is predicting the customer's rating given their reviews text. We will use techniques including natural language processing, sentiment analysis, and machine learning algorithms to explore the possibility to predict customer's rating.

The second objective is to find out the popular topics customers cared about. This insight could support the restaurants owner to elevate their services quality and product. We will include natural language processing, LDA and Machine learning algorithms to reach this objective.

Last but not least, we will build a recommendation system to create business value for business owners as well as yelp.com platform. We will include popular recommendation system methods like co-occurrence matrix, collaborative filtering and matrix factorization and deep learning methods like neural network.

# II. Objective 1: Future Stars Prediction(Xin Lin)

First goal of the project is to predict how many stars a user will give to a restaurant he/she have not previously visited. With this model we can recommend a user restaurant he/she may be interested based on prediction.

## Part 1. Data

There are two data sources that can be used for the project: Hoboken data crawled from yelp.com and full dataset yelp provided on its website. Because we want our model can be applied across the country which means better generalization ability. Bigger training data can get model with better generalization ability, so the full dataset yelp provided with about 3 million rows will be used to train the data.

| business_id | name | is_open | review_count | address | city | Restaurant_Star | date | user_id | Review_Star | funny | useful | cool |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 72956 | Brick House Tavern + Tap | 1 | 116 | 581 Howe Ave | Cuyahoga Falls | 3.5 | 2016-05-05 | 823046.0 | 4.0 | 0.0 | 0.0 | 0.0 |
| 72956 | Brick House Tavern + Tap | 1 | 116 | 581 Howe Ave | Cuyahoga Falls | 3.5 | 2017-03-19 | 1070670.0 | 2.0 | 0.0 | 0.0 | 0. |

Above is preview of the data that will be used.

Business_id: Restaurant ID

name: Name of the restaurant

is_open: Whether the restaurant is still open

review_count: Number of review the restaurant received

address: Address of the restaurant

city: City of the Restaurant

Restaurant_Star: The Star of the Restaurant

user_id: User ID

Review_Star: The star user gave to the restaurant

Date: Time Review was made

Funny: Number of people thought the review is funny

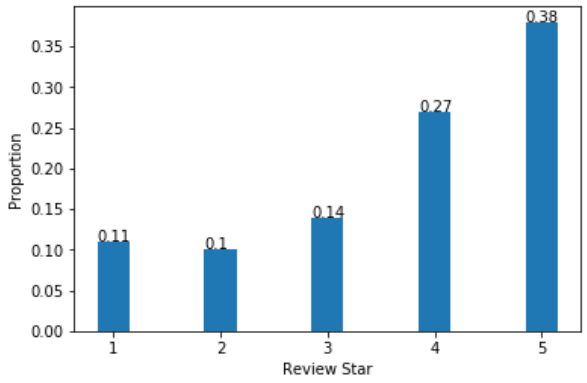
Useful: Number of people thought the review is useful

Cool: Number of people thought the review is cool

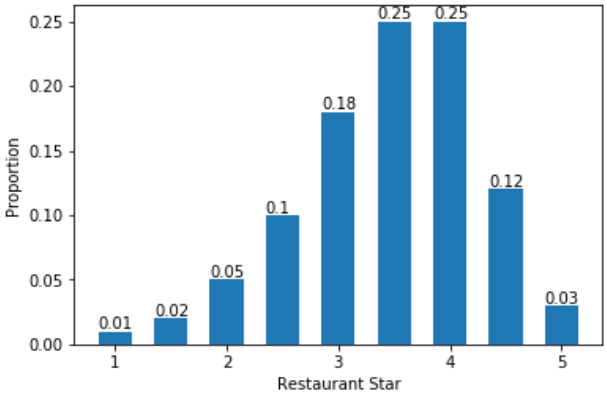
## Part 2. Exploratory Data Analysis

After get the data, an exploratory data analysis helps to get some hidden patterns behind data which may help us build the model.

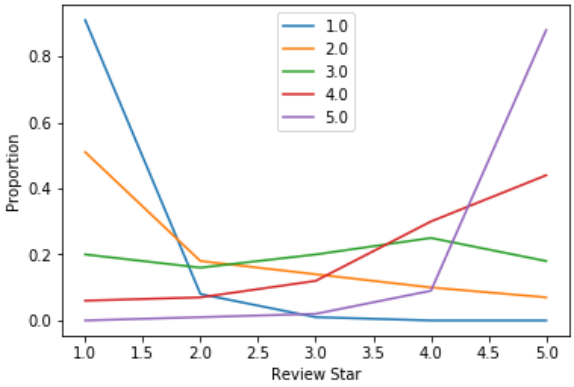
Because our model's target feature is the review star, so let's first check the distribution of the review star in the dataset.




This plot shows that 38% of reviews are given 5 stars, and only about 35% of reviews are stared below 3 stars.

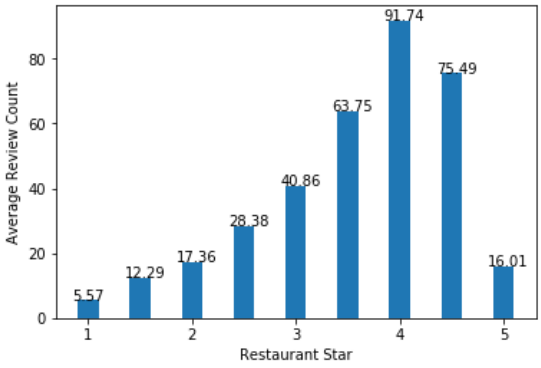


But when we check the Restaurant Star distribution plot, only 3% of the restaurant get 5 stars, and 80% of restaurants get stars between 3 to 4.5 although people prefer to give 5 stars. Let's have a look the review star distribution at different restaurant star level.



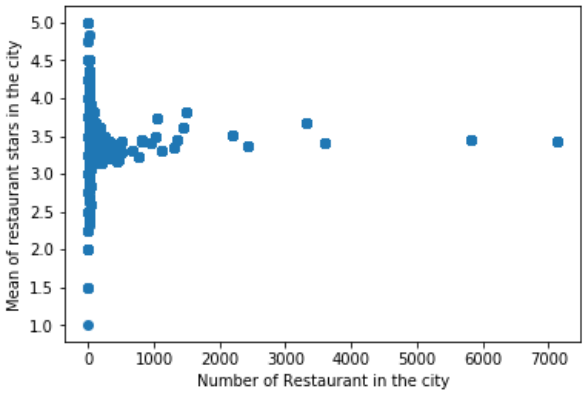

This plot clearly shows that with the star of the restaurant goes up, the proportion of 5 star reviews increases, a restaurant need about 80% of 5 star reviews to be a 5 star restaurant but only about 40% of 5 star reviews to be a 4 star restaurant. So it seems when we are building our models, restaurant star would be a important feature.

Our data provide the number of reviews a restaurant reveiced which may reflect the popularity of the restaurant.



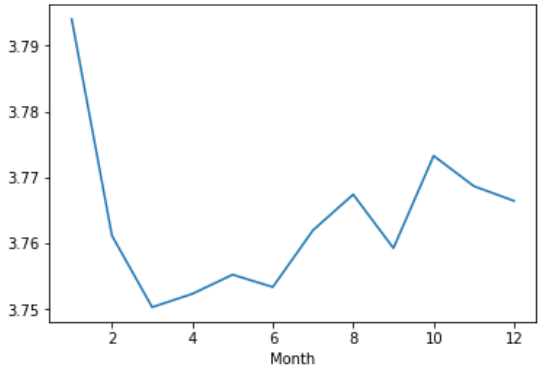

This plot shows that restaurants with 4 stars have most review counts while restaurants with 5 stars have review counts as low as 2-star restaurants. Maybe the 5-star restaurant received 5 star because of lack of reviews.

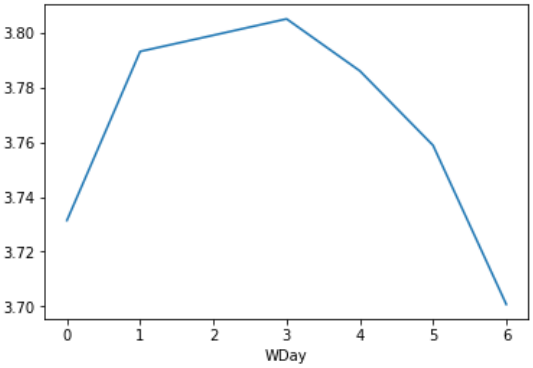
Also, we want to see if the number of the restaurant in the same city will affect users' review star because of competitiveness or changing standard.



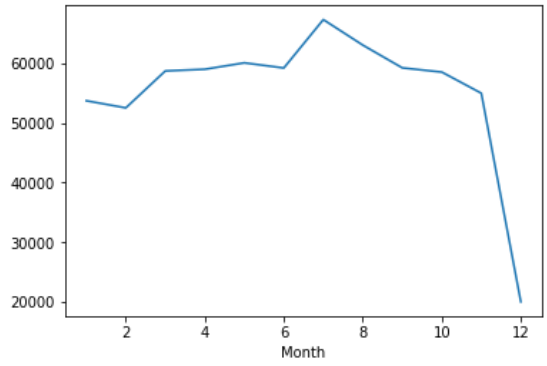

We can see because of the size limit of our data, most cities has restaurants less than 100, there isn't obvious patterns behind number of restaurants in the city.

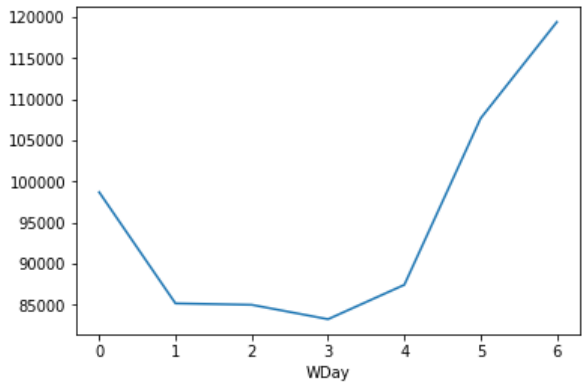
Last, let's check if there is time patterns behind data.




These two plots show that user tend to give higher stars at the beginning of the year and middle of the week. So, these two features may be helpful for our model.




These two plots show the relation between time and review counts, users tend to make fewer reviews at the end of the year and middle of the week.

## Part 3. Model and Metrics:

Before building our models, we have to first decide that our task should be modeled by classifier or a regressor. Since Star rating can only be integer numbers within [1,2,3,4,5] or a regressor. Consider for object 1, our model will be used to predict star the user will give to a new restaurant, which can be used for recommending. So we don't have to set integer constrain for our target star and star we predict has linear trend so in order to take linear trend into account, we decided to use regressor models.

For regressor models, there are root-mean-square error(RMSE) and root-mean-logarithmic-square error (RMSLE), because our predictions are remain in small scales, we don't need to penalize huge over estimates, so RMSE will be choose to evaluate the performance of the model.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(p_i - a_i)^2}{n}}$$

Where pi means predicted value and ai means raw value and n means the number of samples.

## Part 4. Feature Selection:

In order to predict the star a user will give to a restaurant, we need both collect user and restaurants' aggregated historical data.

There are several statistic indicators can be used when we aggregate data：

count, mean, variance, sum

There are two feature selection strategies in our model. First is by business understanding and conclusion from Exploratory Data Analysis. Second is using random forest model's feature importance function to check redundant features and see if cross-validation RMSE is reduced after the redundant features are removed.

After feature importance showed by random forest model, we choose to pick most importance ones to simplify our model on the basis of maintaining RMSE low. Because our data has so many single instances, the mean and variance can't be used efficiently so they will be removed from model.

Here are features we selected for our model:

Restaurant Number(Number of restaurants in the city),

City Mean Star(Average restaurant star in the city),

Month (The month of the predicted visit),

Day of Week(The day of the week of the predicted vist),

Num of Reviews User Made(Number of reviews the user has made),

Num of Useful User Get(Number of Useful the user has get for his/her review),

Num of Cool User Get,

Num of Funny User Get,

User Mean Star(Average star the user gave).

## Part 5. Models Used:

There are several Choices for regression tasks including Epsilon-Support Vector Regression, random forest regressor, and simple linear models like linear regression, lasso regression and ridge regression.

Consider our dataset needs days to be trained on SVR and we have enough data to prevent overfitting, we decided not to use SVR and lasso regression and ridge regression and used GridSearchCV to tuning our hyper parameters.

After Tuning our random forest regressor model gains 0.820 and 0.8577 for simple linear regressor RMSE with 80% training data and 20% validation data on full 3 million rows data.

# II. Objective 1 (cont.): Prediction Model (Richard)

## Part 1. Training

The main objective of my model was to predict whether the user rating was positive or negative based on the user text of a given review. In doing so, I began by adding an additional variable "Positive" in the data frame that was a "1" for a review greater than 3 and a "0" for a review of 3 or less. This allowed me to use logistic regression in the standard fashion, with Tf-Idf, and with N-grams to determine whether each of these prediction methods could most accurately determine the user's rating based on the user's text. 75% of the data was used for feature selection and 25% of the data was used for the test set.

## Part 2. Methods

### Part 2 -1 Logistic Regression

For the prediction method, I simply used Logistic regression and compared the three types to determine the most accurate predictor. The logistic regression was used because although the restaurant ratings were numerical, they were categorical and therefore a better method than other approaches.

### Part 2 - 2 Results

The Logistic regression with the highest accuracy based on an AUC was N-grams which makes sense given the greater number of features as compared to the standard and Tf-Idf type.

| Logistic Regression | |
|---|---|
| **Type** | **Accuracy** |
| **Standard** | 71.80% |
| **Tf-Idf** | 61.53% |
| **N-grams** | 72.78% |
| **Average** | 68.70% |

# II. Objective 1 (cont.): Predicting Yelp Ratings by Review Text and Metadata（Alec）

## Part 1. Introduction

Review websites such as Yelp are great sources of data for applications of natural language processing and sentiment analysis (Yelp). Reviews are by their very nature, expressions of consumer sentiment. The crowd-sourced nature of web reviews yields rich and varied data, which has the characteristic of being raw and unfiltered. By being able to scrape data from the Yelp website, we gain access to a highly targeted source of both review text, but also metadata relating to that review.

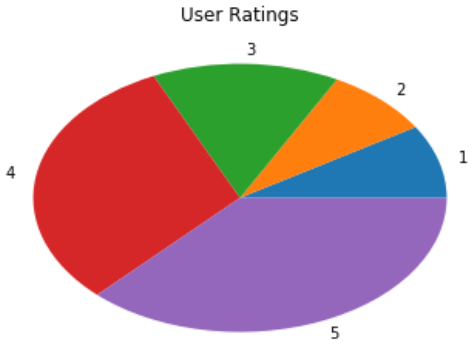
In this project we attempt to scrape data from Yelp, specifically targeting reviews of restaurants within the city of Hoboken, New Jersey. The text and metadata from each review was then investigated in our exploratory data analysis. We attempted some naïve modeling to explore the predictive power of the review text in terms of the customer's rating of the restaurant. Building on that modeling, we conducted additional EDA and factored what we had learned into the creation of additional features for our dataset. We ultimately developed models that proved highly successful at predicting what a Yelp user's rating of a restaurant would be.

## Part 2. Scraping and Preliminary Cleaning

Scraping of the Yelp website was conducted by Ni Man using Selenium and BeautifulSoup4. The scraped data in its original form was raw and unusable: it featured data columns comprised solely of logograms, lists of categorical features had to be extracted from complex strings, and tags which were supposed to be of a uniform format had been seriously altered during the scraping. The data cleaning involved domain knowledge specific to Yelp reviews (i.e. 'Beer Bar' restaurants were recorded as two separate categories: 'Beer' and 'Barawsuvqvreabrdvevr').

## Part 3. EDA and Feature Formatting1

Our initial EDA focused on the characteristics of the metadata that was provided (post-cleaning). This included attributes such as restaurant price, category, and average star rating. We examined the distribution of user ratings and found that it was not symmetrical (leading to our subsequent decisions to use oversampling). We also examined the average customer rating of various 'restaurant types' (Cuban, Sports Bars, Street Vendors, etc.) and discovered that it was not uniform: certain categories of restaurants scored higher than their peers.




The 'Restaurant Star Rating' displayed on Yelp is based on all customer ratings (except for those flagged by Yelp's algorithms, but such processes are industry secrets and not within the scope of this project), using it to predict a user's rating would be an instance of look-ahead bias. Since our scraping had managed to obtain every single review for every single restaurant in Hoboken, we were able to calculate what the restaurant's star rating would be *without* any influence from the rating linked to the review in question (with a few caveats, as documented on Github).

After concluding our exploration of the metadata of each review, we moved on to the true body of the problem: the review text. Raw/unclean text data necessitates an extensive data cleaning process, requiring thought to be paid to language, spelling, unique proper nouns, regional vernacular and content-specific features. Using Python

[1] Though in actuality EDA/data manipulation and modeling were each performed one after the other in separate batches, I summarize what was done within each category.

libraries such as nltk and autocorrect (for language/spelling checks as well as for stemming, tokenization, and vectorization), we were able to not only ensure the text data was in a uniform format, but also measure properties of the text that wouldn't be picked up in traditional text-based machine learning workflows.

From the text we were able to extract several additional features for each review, including review length and number of spelling errors in a review. The features developed during the metadata exploration (restaurant price, restaurant rating, restaurant type) were converted into numerical values and examined. All these variables displayed machine-exploitable relationships to our predicted variable (not necessarily linear, but patterns were visible that should be able to be exploited by certain machine learning systems). We combined these features with the vectorized-text features (described below) and normalized them before using them for predictive models.

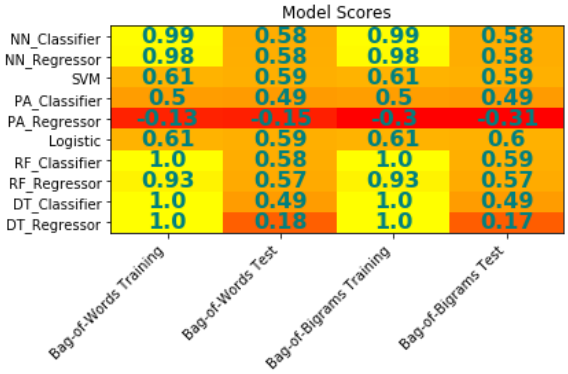
## Part 4. Text Vectorization and Machine Learning

For our first attempts at modeling user reviews, a naïve approach was taken, to demonstrate the predictive properties of the text. We converted the user's review into a vector representation known as 'bag-of-words,' recording whether or not a word was present in a review, for a certain number of words (which we capped, to prevent overfitting at this stage). We also explored the addition of 2-grams, pairs of two words that appear subsequently in a body of text. These are of course much less frequent than the individual words (or 1-grams) that make them up, yet they typically contain much greater predictive power within NLP classification problems than 1-grams alone.

We used these bag-of-words and bag-of-bigrams representations as inputs for a variety of models, to compare the performance of the models on the problem at hand. We reviewed 10 different predictive models ranging from simple models such as the logistic regression or passive-aggressive to more complex machine learning

models such as random forests and neural networks (scikit-learn). Both classifying models and regressing models were used in these tests because the data that was being predicted, while technically categorical (integers 1, 2, 3, 4, and 5), also could be modeled using a regression, since the output was numeric and there existed a relationship from lower numbers to higher numbers that could be learnt by the algorithms.

Overall, the algorithms, even with highly simplified data to work with, managed to reach testing accuracies as high as 60% (for classification problems). Furthermore, when comparing between the accuracy of our classification models and the performance metrics from our regression models (i.e. $R^2$), we found that both classification and regression models showed similar performance. In fact, if we were to apply a sort of "classification filter" to regression model predictions (mapping continuous predictions into the discrete range of possible outcomes), the regression models would likely have outperformed the classification models while using the same hyperparameters and underlying data[2].




After this preliminary testing of different model architectures on the data, we were ready to begin our final model. We started with additional review cleaning, review tokenization, word stemming, and then finally we vectorized the text using TF-IDF instead of bag-of-words (still incorporating n-grams) and normalized. TF-IDF is more informative than bag-of-words because where bag-of-words gives a binary output: yes or no (the word is

[2] This possibility was not able to be explored due to time constraints resulting from waiting for scraped data to be made available and spending time cleaning and altering poorly-formatted data.

present in the document), term-frequencies allow additional insight that would otherwise be simplified by bag-of-words (Liu, 2015). This text vector data as well as the constructed and scraped metadata were now collectively used as inputs to our predictor models.

In our naïve testing, the random forest classifier had achieved the highest training accuracy and was tied second for highest test accuracy (across both the bag-of-words and bag-of-bigrams inputs). We thus started our testing this time using the random forest classifier. Our first tests of this classifier using the new data achieved astounding results. Even when we tuned the hyperparameters to limit the maximum features uses and maximum depth of each random tree, we were still able to achieve a ROC AUC of 94%, a Cohen's Kappa of 92%, and a test accuracy of 95%. We then tried to see if the closely related decision tree algorithm would have similar results. As expected it did, and with generous hyperparameters it was able to achieve 99% testing accuracy (the same as random forest did, when it was unconstrained). When constraints were applied to the decision tree it's ROC AUC was 95%, Cohen's Kappa 93%, and test accuracy 95%. Though the decision tree outperformed the random forest, we'd be tempted to lean towards the random forest in this instance since the difference was so slight, since random forests are more robust in terms of avoiding overfitting the data (Donges, 2018). Some basic testing showed that with additional random trees, the random forest algorithm's robustness and performance metrics would continue to improve. We could end up with a random forest spanning hundreds of trees with very good performance metrics that would likely be highly robust against overfitting.

When we tested multilayer perceptron classifiers on the data they were unable to perform to the same standards, with their AUC hovering around the low 70s, a Cohens Kappa in the mid 40s, and a testing accuracy of roughly 60%, barely better than it performed with the simple bag-of-words inputs. We ran the neural network through sklearn's grid search functionality to try and explore different combinations of hyperparameters (specifically the architecture of the hidden layers) and were unable to improve the model much from it's default

hyperparameters, aside from enabling early stopping for the training. We attempted to use the same hyperparameters as were used for the classifier in the bag-of-words models, but this did not improve performance. Likely this was because the size of our input layer differed greatly between the two implementations of the models. Our TF-IDF data represented more words than our simpler vector models did (part of what allowed the tree-based algorithms to perform so well) and all the additional features that were added based on text and review data. The neural network simply couldn't learn efficiently from the data given the hyperparameter formats that we tested.

We tested several other models on this new dataset, but the overall performance of the tree-based algorithms was the most impressive. Overall, we conclude that the random forest-based algorithm was the most appropriate for the data, it mapped feature importance correctly and was able to transition to achieving stellar performance scores on testing data.

# III. Objective 2: Popular Topics(He)

## Part 1: Data

### Part 1 -1 Data collection

We scraped the Hoboken data from the yelp.com, and saved data to csv format. We have 8 columns including: user_name, usering_rating, restaurant_name, restaurant_rating, restaurant_price, restaurant_type.

### Part 1 - 2 Clean data

Hoboken data was collected in csv format, totaling over 74000 messages. Before creating "bag of words", I needed to clean reviews, including:

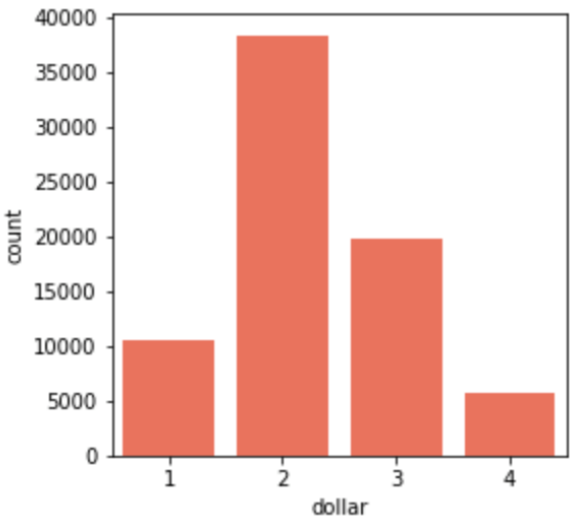
- Removing stop words
- Use re removing numbers, punctuation only keep letters(re.sub('[^a-zA-Z]')
- Use PoterStemmer stemming words
- Split words and change all words to lower words

### Part 1 - 3 Saved the cleaned reviews in dataset

## Part 2: Exploratory Data Analysis

### Part 2 -1 Check the Dataset

The first step I wanted to check whether all data are valid, but I found that in the restaurant price columns had 132 missing values. Missing values will affect building model, so first I will analyze price information to find the best way to handle these missing values.



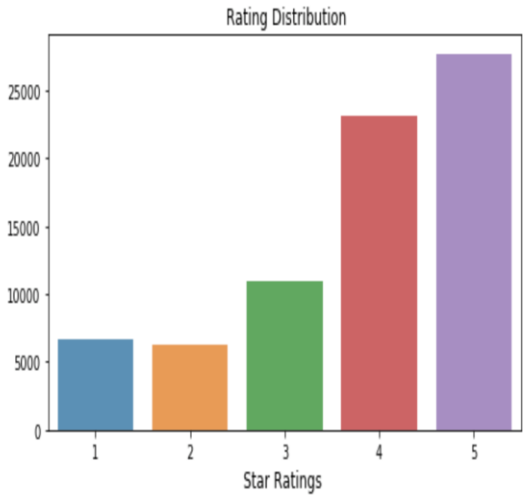

As picture shows: the most restaurant price is 2 dollar signs. In these cases I chose use mode to replace nan values. Cause the mean will be affected by 1 dollar sign restaurant and 4 dollar signs restaurant, this not a fair method. We also can drop the rows which contains nan values but is less accuracy. When I replace the non values, let us check dataset again:

```
dataset.isnull().values.any()
```

```
False
```

### Part 2 - 2 Rating distribution

Cause we will predict rating form reviews, so check the rating distribution form customers.

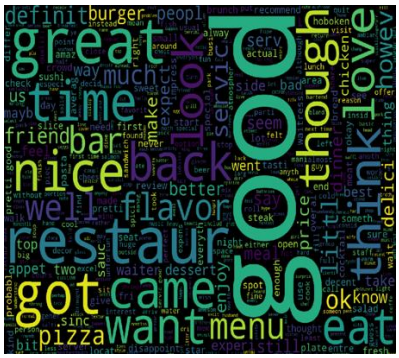


AS this picture shows: there were 27702 reviews about 5 stars, only around 6000 reviews about 1 or 2 stars.
When I checked real restaurant stars, only 484 reviews about 5 stars restaurants, so I thought this not the important feature during prediction.

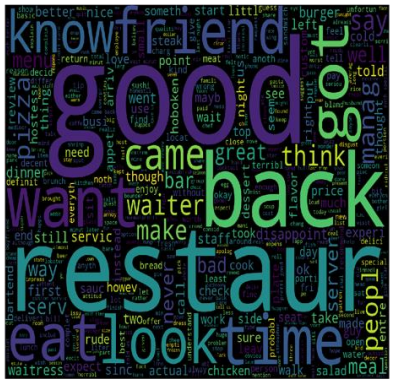
### Part 2 – 3 Use WordCloud

Used generate method to find words frequency in reviews. In order to see clearly, I separated reviews in 3 level based on user rating. Put over 3 stars in high level, 3 stars in mid level, lower than 3 stars in low level.


high level




mid level




low level

## Part 3: LDA

In this part, I wanted to use cv and tf-idf to get features from reviews. And used LDA to get topics.

**CV(CountVectorizer):**
This model has many parameters, however the default values are reasonable, so I use the default value. Each term found by the analyzer during the fit is assigned a unique integer index corresponding to a column in resulting matrix.

**Tf-idf**:
In a large text corpus, some words will be very present (e.g. "the", "a", "is" in English) hence carrying very little meaningful information about the actual contents of the document. If we were to feed the direct count data directly to a classifier those very frequent terms would shadow the frequencies of rarer yet more interesting terms. After processing, there are over 40000 features.

TF(t) = of times t occurs in the document total / of terms in the document
IDF(t) = log total (of documents /of docs with the term t in it)

When I got features, I wanted to use LDA to select topics. I used perplexity values to make sure the number of topics, when the perplexity values decreased, the results would be good. However, when I tried to do many times iteration, my laptop can not handle, so I tested 10 topics, iteration 10 times and 50 topics iteration 4 times. As compared, the perplexity of 20 topics had best score.

```
iteration: 1 of max_iter: 10, perplexity: 2017.056
iteration: 2 of max_iter: 10, perplexity: 1868.057
iteration: 3 of max_iter: 10, perplexity: 1817.046
iteration: 4 of max_iter: 10, perplexity: 1795.085
iteration: 5 of max_iter: 10, perplexity: 1784.334
iteration: 6 of max_iter: 10, perplexity: 1778.160
iteration: 7 of max_iter: 10, perplexity: 1774.189
iteration: 8 of max_iter: 10, perplexity: 1771.440
iteration: 9 of max_iter: 10, perplexity: 1769.455
iteration: 10 of max_iter: 10, perplexity: 1767.91
```

10 topics

```
iteration: 1 of max_iter: 4, perplexity: 18425.1222
iteration: 2 of max_iter: 4, perplexity: 17677.0376
iteration: 3 of max_iter: 4, perplexity: 17486.4388
iteration: 4 of max_iter: 4, perplexity: 17401.7384
```

50 topics

After using LDA fitted, I could get features name and printed top words to analysis popular topics.

As compared, I thought cv is better than tf-idf, cause after printing top words, the words according tfidf processing were unreadable, but the words according cv processing were easily understanding. I could summary some information by these top words.

```
Topic #0: stl redonk flashforward leeki around liitl revelatori fare defi
nitali siriu litttttl chettinad redbean hap dessertveri cerveza panella m
emoribiila coach eden
Topic #1: houston msr coursesfor corton persuas mannnnn candac palenta tr
iolobst ofsalad staffcon kc carrozza ballerina selction keep pralin woop
payback dutchess
Topic #2: musselschicken bobba decreas pleni rosselini chelsi operandi pe
peroncino nomenom esquina fleeki pastasa minder fishcod supposecthat tigh
cronut categori myfirst trish
Topic #3: accckkk basiclali walru isabel lafreida tataki nosier plaster v
inagarett younger bet wyett samich iwth shutthefrontdoor simpledeliveri q
uickcon undiscov thedowntownfox lunchtim
Topic #4: wiener cheesefood trist freezer taquer atmostph burboun preconc
ept stagioni suav unimagin horni stabl discript ooooouuu noisey mill cappo
n decend steller
Topic #5: tartuffo vlog antipasta wayyyyy agadash flourless while sleev e
spinica damien saxophonist tricki famoud bock proced snowmaggedon resrvat
easi chikarashi perhapsso
Topic #6: galleri delv pupkin longan jason emmett beefspinach coerc orich
etta norwegian aki ovin deutschland servci vareiti begrudgedli close taco
sgril capish cheesehellfir
Topic #7: thesesid fingerscrispi ead cocinar truthhhhh nostalgia packedha
rd model reinvigor smooch dismay daft dishth viscos bassmi tacna trough s
tll cappachio impercept
Topic #8: shakshouka thsn blond dulsot penthouse nine chickenwa multisens
ori riddl itowtr nemet cardiac muffl freshli malic hyperbol wineo anndd t
ab cajan
Topic #9: seductress personsdirti flavorsand udang yeungl kinneson pestl
excellentdinn interview itfood clearwat marmelad progeni beginn teasti ye
srecommend quatro inhospit bobbbi prais
Topic #10: empenada umeshisu soupman calimari sendnoodz scnackenberg loub
otin vn lumpia hudon linguettin eoe maclaren organis though loosen terayk
i secondari vbbvvb mfeliciano
```

cv

```
Topic #0: matt tingli mock korta malai delhi gobhi daal potion abysm saze
rac justin teh grinder protect homesick makhni ganouj mimi etiquett
Topic #1: kati biryani bagatel puri marco tikka biriyani stein masala mof
ongo gauc dahi sarcast ft dilli joey meera bhel spa huancaina
Topic #2: dan branzino saki bier sichuan cavatelli sorellina pio marisco
lotu poop gordo overdu cotton alcapurria brasa comeback spendi dank patat
Topic #3: bday british germani zeppelin tendon bibim foi bap numb buckwhe
at broccolini flush shiitak tattoo ali monkey peppercorn diavolo aussi sa
lumi
Topic #4: barbuto gyoza pilsen mixologist catalina coolest houseman godzi
lla mingl whiz optimist spain tarot unattent rage mozarella luca kolo wv
sarah
Topic #5: good great servic taco time like love back best delici pizza wo
uld also nice definit amaz wait chicken us price
Topic #6: paneer naan dosa tandoori meera chaat jamun gulab washugyu dal
shepard pepe smh cacio evid kali aloo naancho causal cosmo
Topic #7: soba tamarindo thanh lomo cc saltado hoai trang nha rockin hang
ri lac briefli kardashian ea miscommun fundido rehears bleecker luc
Topic #8: chilean fettuccin shatter agedashi stephani tenjun tripe rutger
rapidli lonni rico disput volunt ricki tiffani limoncello herm brit petal
coctail
Topic #9: uni malaysian roti niku canai chirashi tijuana sri viet nori ma
ssaman blanc kabab yook sauvignon thier rendang rosu testicargot nan
```

Hoboken EDA and LDA He LI

Tf-idf

Topic summary: fresh food, bathroom, food looks like, menu and price, good service, events in restaurants, restaurant's website, good for health, food sides, wait time, parking.

## Part 4: Prediction

### Part 4 - 1 Set training set and test set

Because in this part, I wanted to predict restaurant stars by user reviews, so I set reviews as independent variable, rating level as dependent variable. Put 80% data in taring set, 20% data in test set.

### Part 4 - 2 Method

There are several options for prediction process, including: Naïve Bayes, Logistic Regression, Random Forest, SVM. In this part I wanted to predict restaurant level include high level, mid level, low level

**Linear Regression**: Linear regression is generally used to solve continuous-valued variable prediction problems. There is supervised learning. This is called linear regression because we assume that there is a linear correlation between the independent variables and the dependent variables. In this part, I thought that variables were not continuous-valued variable, so I did not use liner regression.

**Logistic Regression**: compared with linear regression, logistic regression is a non-linear regression model that studies the relationship between the independent variables and

dependent variables. I chose this method cause this method is simple and faster than others.

**SVM**:

This method has high accuracy, especially, popular in text classification problems, but it is memory-intensive, hard to run and tune, so I did not use this method.

**Random Forest**： This method also does not need continues variable, and it can use in classification and regression area, compared with SVM, this method can save a lot of time, especially in the non-linear area, so I chose this method.

**Naïve Bayes**: This method is suitable for less data, so I did not use this method.

### Part 4 - 3 Sentiment analysis

I want to increase the prediction accuracy, so I use sentiment analysis in reviews.

I chose use confusion matrix to calculate accuracy. In this part I want to calculate the.

The first step I need create a new bag of words plus sentiment score, then set independent variables and dependent variables, then use logistic regression and random forest to calculate accuracy.

| 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 38203 | 38204 | 38205 | 38206 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | |

### Part 4 - 4 results:

In this. Prpject, I need to know the ability of the classifier to judge the overall, ie, the proportion of correctly predicted, so I calculated the each model accuracy by confusion. The digital representation on the diagonal is the number of the forecasted correct.

After adding sentiment analysis, logistic regression accuracy increased 0.01. random forest increased 0.01. The logistic regression had best score (84%).

Hide Code

# IV. Objective 3: Recommendation System (Ni)

In this part, I will build recommendation system with trying three popular recommendation methods and one deep learning approach(Artificial Nerual Network). The target of this part is to recommend 3 more different restaurants to each customer in the Hoboken Restaurants Reviews dataset. The recommendation system will help business owner as well as yelp to create profit. The methods I will try showed as follows:

1. Co-occurrence Matrix
2. Collaborative Filtering
3. Matrix Factorization
4. Deep learning (Neural Network)

## Part 1. Data Understanding and Cleaning

Before building the model, I will introduce the data I will use. The raw data in this part is the Hoboken Restaurants Reviews dataset, which includes 74,611 rows and 8 columns. First, I will clean the data and prepare required data for following recommendation system methods.

### Part 1 - 1 Basic Understanding of data

Definition of each variable:

1. user_id: Unique id for each customer
2. user_name: customer's name
3. user_raing: original rating for one restaurant per review
4. user_text: text of customer's review for one restaurant per review
5. restaurant_name: unique name for each restaurant
6. restaurant_rating: the integrated rating of each restaurant
7. restaurant_price: degree of cheap or expensive of one restaurant
8. restaurant_type: the style and theme of one restaurant

**Table 1** show a sample dataset and basic information of raw data. **Chart 1** shows the datatype of each column. There are some columns like user_rating, restaurant_rating and restaurant_price should be convert into numerical data. In addition, there are some repeated comma in restaurant_type column, so I will do data cleaning first and try to use this variable in Neural Network part. In addition, it use '$$' symbol represent the degree of price of each restaurant and the symbol should be converted to integer, which could be recognized by the computer.

. . .

| | user_id | user_name | user_rating | user_text | restaurant_name | restaurant_rating | restaurant_price | restaurant_type |
|---|---|---|---|---|---|---|---|---|
| 0 | dRuCO4NYO7zyAF8-CeJmZg | Jason L. | 5.0 star rating | We booked Grand Vin as our brunch location to ... | Grand Vin | 4.0 star rating | $$ | Wine, Bars,, Italian,, Cocktail, Bars |
| 1 | f36YZ1cA291bNtMHXWtu1Q | Danyale W. | 4.0 star rating | Sooooo for date night it was his turn to pick ... | Grand Vin | 4.0 star rating | $$ | Wine, Bars,, Italian,, Cocktail, Bars |
| 2 | -xYUKfWQTaB-7BeizsQA3w | Robin G. | 5.0 star rating | Adorable little wine bar with outdoor seating ... | Grand Vin | 4.0 star rating | $$ | Wine, Bars,, Italian,, Cocktail, Bars |
| 3 | tt1vLgAP5UpRXAKJLT2KWg | Alec K. | 4.0 star rating | One of the top restaurants in Hoboken. Well ma... | Grand Vin | 4.0 star rating | $$ | Wine, Bars,, Italian,, Cocktail, Bars |
| 4 | -K79Xep4IElqIChsJYWuiQ | Robbie O. | 5.0 star rating | Great space-service is on point - short rib ... | Grand Vin | 4.0 star rating | $$ | Wine, Bars,, Italian,, Cocktail, Bars |

**Table 1: Sample of raw data**

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 74611 entries, 0 to 74610
Data columns (total 8 columns):
user_id              74611 non-null object
user_name            74611 non-null object
user_rating          74611 non-null object
user_text            74611 non-null object
restaurant_name      74611 non-null object
restaurant_rating    74611 non-null object
restaurant_price     74479 non-null object
restaurant_type      74611 non-null object
dtypes: object(8)
memory usage: 4.6+ MB
```

**Chart 1: Datatype**

We could find some basic information of the raw data from **Table 2**. It shows that there are 74,611 reviews in total, while only 44,949 unique users leave the reviews, which means multiple users attended to more than one restaurants and it is valuable to build the recommendation system. Also, it shows that there are only 24,781 unique user's name, so in order to reduce the ambigurity, I will use user_id instead of user name in the recommendation system.

| | user_id | user_name | user_rating | user_text | restaurant_name | restaurant_rating | restaurant_price | restaurant_type |
|---|---|---|---|---|---|---|---|---|
| **count** | 74611 | 74611 | 74611 | 74611 | 74611 | 74611 | 74479 | 74611 |
| **unique** | 44949 | 24781 | 5 | 69542 | 302 | 7 | 4 | 230 |
| **top** | QxTNaHoxTL8f7jAK5nwQ1g | Nicholas G. | 5.0 star rating | I've been coming to STK downtown since they ha... | Morimoto | 4.0 star rating | $$ | Japanese,, Sushi, Bars |
| **freq** | 101 | 103 | 27702 | 3 | 2740 | 31568 | 38323 | 3620 |

**Table 2: Basic Information**

After checing the null value, we could find that there are null value in restaurant_price column. **Table 3** shows that there are some missing restaurant_price for some restaurants, since yelp have not updated the price for these restaurants. In following part, I will refill these missing value accordindly.

```
user_id              False
user_name            False
user_rating          False
user_text            False
restaurant_name      False
restaurant_rating    False
restaurant_price      True
restaurant_type      False
dtype: bool
```

| | user_id | user_name | user_rating | user_text | restaurant_name | restaurant_rating | restaurant_price | restaurant_type |
|---|---|---|---|---|---|---|---|---|
| **5273** | i_WxJpoxfsddmBne6I8cKQ | Durva L. | 4.0 star rating | This is such a small place, but some great rea... | Funjabi | 2.5 star rating | NaN | Indian,, Chinese |
| **5274** | zr0jkmEJLnaZxWZi7cp09Q | Wasbir R. | 1.0 star rating | Giving 1 star because theres no 0 star. This i... | Funjabi | 2.5 star rating | NaN | Indian,, Chinese |
| **6782** | cnNddUd4fn7h-Pb_Cma-9Q | mako y. | 5.0 star rating | Wonderful!!! It was traditional with unique tw... | 奥田 | 4.5 star rating | NaN | Japanese |
| **6783** | q1P19WvpTXFTmfU7oXqkaq | Chester G. | 2.0 star | When I heard Chef Toru Okuda | 奥山 | 4.5 star rating | NaN | Japanese |

```
number of missing value: 132
```

**Table 3: Missing value**

## Part1 - 2 Data Cleaning and Preprocessing

In this part, I will check each column one by one and prepare the dataset for recommendation system part.

**- user_id**

**Table 4** list the top 10 users who left most reviews to restaurants in Hoboken. Multiple customers left reviews for different restaurants more than one time which is valuable for the recommendation system.

| | user_id | count |
|---|---|---|
| 0 | QxTNaHoxTL8f7jAK5nwQ1g | 101 |
| 1 | 5aZX8bTiD0k9vR60SG588Q | 76 |
| 2 | GRY2acZtl5q4P1KdCWhcUQ | 66 |
| 3 | LcWOq7p7Mhtv9hIlDrhy9A | 62 |
| 4 | Y59HQNazSLR1EUMUmMOaFg | 61 |
| 5 | NWLqOKl0Vxi7qK-6EoBLbg | 60 |
| 6 | A5dqSwriUs8cV4DEzS_V9A | 59 |
| 7 | svFycHjXZYpNVutZ_0_gDQ | 59 |
| 8 | dl4ENy4Bk6-lCu59A8vxbg | 57 |
| 9 | 94V-snVUg2Gd-OPd7rcZEg | 54 |

**Table 4: Top 10 users left most reviews**

| | rating | count |
|---|---|---|
| 0 | 5 | 27702 |
| 1 | 4 | 23140 |
| 2 | 3 | 10916 |
| 3 | 1 | 6608 |
| 4 | 2 | 6245 |

**Table 5: Count of rating**

| | restaurant_name | count |
|---|---|---|
| 177 | Morimoto | 2740 |
| 261 | The Spotted Pig | 2640 |
| 153 | Los Tacos No.1 | 2100 |
| 258 | The Park | 1640 |
| 287 | Wafels & Dinges | 1500 |
| 46 | Catch | 1480 |
| 90 | Employees Only | 1300 |
| 15 | Artichoke Basille's Pizza & Bar | 1300 |
| 71 | Del Posto | 1260 |
| 93 | Fig & Olive | 1220 |

**Table 6: Restaurants received most reviews**

**- user_name**

Since different customers might have same name, in order to reduce the ambigurity, I will use user_id instead of user name in the recommendation system. There is nothing to do with the user_name column.

**- user_rating**

I will convert the string into numerical format as I mentioned and include user_rating variable to calculate the weights or scores the given customer rated. **Table 5** shows the count of number of each rating star in the Hoboken_restaurants_reviews dataset.

**- Restaurant_name**

The restaurant's names are unique in this dataset. So I will use the restaurant name represent each restaurant in the following part. There are 302 different restaurants in the dataset in total. **Table 6 and Table 6.1** shows that some restaurants received more than 2 thousands revies in Yelp and some only received 1 or 2. This information might influence the accuracy of the recommendation system, since some types of recommendation methods will be limited by data sparsity. A restaurant with too seperated reviews might not be selected as a recommended result.

| | restaurant_name | count |
|---|---|---|
| 165 | Manhattan Bar Grill & Lounge | 1 |
| 18 | Azteca Taqueria Restaurant | 1 |
| 159 | MONDO - DOGS | 1 |
| 81 | El Cantante | 1 |
| 289 | Wah Yoan | 1 |
| 92 | Field House Grill | 1 |
| 1 | 52 Restaurant | 1 |
| 87 | El Salvador Restaurant | 1 |
| 240 | Taquitos Mexicanos | 2 |
| 99 | Funjabi | 2 |

**Table 6.1: Restaurants received least reviews**

| | restaurant_rating | count |
|---|---|---|
| 4 | 4.0 | 31568 |
| 3 | 3.5 | 26130 |
| 5 | 4.5 | 8866 |
| 2 | 3.0 | 7263 |
| 6 | 5.0 | 484 |
| 1 | 2.5 | 267 |
| 0 | 1.5 | 33 |

**Table 7: Rating Distribution**

| | restaurant_price | count |
|---|---|---|
| 1 | 2.0 | 38323 |
| 2 | 3.0 | 19809 |
| 0 | 1.0 | 10567 |
| 3 | 4.0 | 5780 |

**Table 8: Price Distribution**

**- restaurant_rating**

I will convert the restaurant_rating column into numerical data. **Table 7** shows the distribution of different rating scores. The restaurant_rating variable might influence the customer's decision making, so I will take this variable into account in the Neural Network method.

**- Restaurant_price**

I will convert the restaurant_price column into numerical data. **Table 8** shows the distribution of degree of restaurant's price. We could find that the level 2 degree is the most popular degree of restaurant's price.

The restaurant_price variable might influence the customer's decision making. There are some missing restaurant_price for some restaurants, since yelp have not updated the price for these restaurants. I will use 2 to refill these cells according to the statistical result that there are most restaurants with restaurant price in level 2.

. . .

**- Restaurant_type**

I will do data cleaning for the restaurant_type column. The restaurant_type variable is significant to influence customer select a restaurant. So I will focus on this variable in the Neural Network method. **Table 9** shows the most popular restaurant type in this dataset, 4000 users have left reviews to a 'sushi japanese bars' restaurant.

. . .

| | restaurant_type | count |
|---|---|---|
| **204** | sushi bars japanese | 4080 |
| **169** | pizza italian | 3348 |
| **133** | italian | 3070 |
| **151** | mexican | 2988 |
| **122** | gastropubs burgers | 2640 |
| **98** | cocktail new spaces venues american bars event | 1640 |
| **116** | food belgian trucks waffles | 1500 |
| **186** | seafood bars fusion sushi asian | 1480 |
| **10** | american new bars | 1451 |
| **103** | cuban | 1320 |

**Table 9: Restaurant Type with most reviews**

Now, I finish data cleaning and preprocessing part. I will store the cleaned dataset into 'Hoboken_restaurants_reviews_cleaned.csv'. This dataset will also be used in the method 4 - Neural Network methods.

| | user_id | user_name | user_rating | user_text | restaurant_name | restaurant_rating | restaurant_price | restaurant_type |
|---|---|---|---|---|---|---|---|---|
| **0** | dRuCO4NYO7zyAF8-CeJmZg | Jason L. | 5 | We booked Grand Vin as our brunch location to ... | Grand Vin | 4.0 | 2.0 | cocktail bars wine italian |
| **1** | f36YZ1cA291bNtMHXWtu1Q | Danyale W. | 4 | Sooooo for date night it was his turn to pick ... | Grand Vin | 4.0 | 2.0 | cocktail bars wine italian |
| **2** | -xYUKfWQTaB-7BeizsQA3w | Robin G. | 5 | Adorable little wine bar with outdoor seating ... | Grand Vin | 4.0 | 2.0 | cocktail bars wine italian |
| **3** | tt1vLgAP5UpRXAKJLT2KWg | Alec K. | 4 | One of the top restaurants in Hoboken. Well ma... | Grand Vin | 4.0 | 2.0 | cocktail bars wine italian |
| **4** | -K79Xep4IElqIChsJYWuiQ | Robbie O. | 5 | Great space-service is on point - short rib ... | Grand Vin | 4.0 | 2.0 | cocktail bars wine italian |

**Table 10: Clean dataset and Dataset for Method 4**

## Part 2. Data Preparation

In this part I will prepare the data for co-occurence, collaborative filtering and matrix factorization methods. I will only keep the user_id, restaurant_name and user_rating column to build 3 basic recommendation system. **Table 11** shows a sample of this dataset.

| | user_id | restaurant_name | user_rating |
|---|---|---|---|
| 0 | dRuCO4NYO7zyAF8-CeJmZg | Grand Vin | 5 |
| 1 | f36YZ1cA291bNtMHXWtu1Q | Grand Vin | 4 |
| 2 | -xYUKfWQTaB-7BeizsQA3w | Grand Vin | 5 |
| 3 | tt1vLgAP5UpRXAKJLT2KWg | Grand Vin | 4 |
| 4 | -K79Xep4lElqlChsJYWuiQ | Grand Vin | 5 |

**Table 11: Sample of dataset**

| | user_id | restaurant_name | user_rating |
|---|---|---|---|
| 0 | ---xAZNw9fFPBoy7jmkA2A | Chef Of India | 4 |
| 1 | --68ZwhCrUJUmCXXkMTMKw | Prime Food Market | 5 |
| 2 | --8M2DZ9JkDwTveuRhLPTQ | Del Posto | 5 |
| 3 | --ARr3m5JsxaX3DTUVQW7w | Morimoto | 2 |
| 4 | --CZJeSlpxwQ0VULjnM57w | The Brass Rail | 1 |

**Table 12: A group-by Dataset(Method 2)**

Customers might rate more than one time for one restaurant. I will use the average rating scores for the cases that customer rated more than one time. This group-by table, **Table 12**, will also be used in collaborative filtering recommendation system.

Based on the group-by dataset, I use pivot() function to convert it into a matrix like dataset. Each row represent each user and each column represent each restaurant. **Table 13** shows the sample of the matrix like dataset, which will be used for method 3, the matrix factorization recommendation system.

| restaurant_name | 10th & Willow Bar & Grill | 52 Restaurant | 8th Street Tavern | Adoro Lei | Aether Game Cafe | Ahri's Kitchen | Ainsworth Hoboken | Aldys Restaurant | Ali Baba | Amanda's Restaurant | ... | White Star Bar | Wicked Wolf Tavern |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| user_id | | | | | | | | | | | | | |
| ---xAZNw9fFPBoy7jmkA2A | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 |
| --68ZwhCrUJUmCXXkMTMKw | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 |
| --8M2DZ9JkDwTveuRhLPTQ | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 |
| --ARr3m5JsxaX3DTUVQW7w | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 |
| --CZJeSlpxwQ0VULjnM57w | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 |

5 rows × 302 columns

**Table 13: A Matrix Like dataset with rating(Method 3)**

The **Table 14** is similar with the **Table 13**, the only difference is that I convert all of the rating scores into 1, which represent the customers attended to the

| restaurant_name | 10th & Willow Bar & Grill | 52 Restaurant | 8th Street Tavern | Adoro Lei | Aether Game Cafe | Ahri's Kitchen | Ainsworth Hoboken | Aldys Restaurant | Ali Baba | Amanda's Restaurant | ... | White Star Bar | Wicked Wolf Tavern |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| user_id | | | | | | | | | | | | | |
| ---xAZNw9fFPBoy7jmkA2A | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 |
| --68ZwhCrUJUmCXXkMTMKw | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 |
| --8M2DZ9JkDwTveuRhLPTQ | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 |
| --ARr3m5JsxaX3DTUVQW7w | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 |
| --CZJeSlpxwQ0VULjnM57w | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 |

5 rows × 302 columns

**Table 14: A matrix Like dataset without rating(Method 1)**

After preparing the dataset for following part, I will store these dataset as 'Method_1_dataset.csv', 'Method_2_dataset.csv' and 'Method_3_dataset.csv'.

## Part 3 Recommendation System Methods:

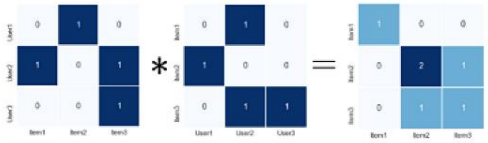
**In this part I will use following methods to build recommendation system model:**

- co-occurrence matrices
- collaborative filtering
- matrix decomposition
- Neural Network

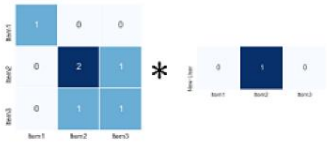
### Part3 - 1 co-occurrence Matrices

- Background: This is the simpliest method I use to build the recommendation system. The underlying assumption is that the customers will be interested in the restaurants that the other customers who have similar interest.
- Algorithms/equations:
  - Co-occurrence Matrix = A.T * A
  - Recommendation = Co-occurrence Matrix * u
    Following chart will show a simple sample to apply co-occurrence matrix for recommendation system.

$$Co\text{-}occurrence\ Matrix = A * A^T =$$



$$Recommendation = (A * A^T) * u =$$



$$=$$

***Recommend the Item 3***

'A' represent the user-item matrix, which summarizes the interactions between users and items. The matrix is just from the raw data, with one row for each user, and one column for each item. 'A.T' is the transpose of this matrix. In 'A.T', rows means item and the column means user. In the first step, we should multiple 'A' and transposed 'A' and will get the co-occurrence Matrix, which is also the recommender to recommend the item to user. Assume that we have a new user, this new user have bought the item 2. We multiple the co-occurrence matrix and the array represent the new user. Finally, we will obtain the recommendation result which shows we should recommend the Item 3 to this new user. The reason why we ignore the item 2 is that this user have already bought it.

- Tools:
  - Specifically, I will use numpy, pandas in python to build the recommendation model.

**- Step 1: Data Loading**

The data I input is the **Table 14**, a matrix like dataset without rating scores.

**- Step 2: build co-occurrence matrix**

I use numpy to convert the matrix like dataset into numpy matrix and then obtain the transpose of this numpy matrix. Now I have two matrices, one original matrix and one transposed matrix. I multiple these two matrices and then eliminate the value in the diagonal cell of the result matrix to avoid repeadedly take attended restaurants into account. And then, I obtain the co-occurrence matrix like **chart 2**.

```
matrix([[ 0.,  0.,  9., ..., 11.,  3.,  0.],
        [ 0.,  0.,  0., ...,  0.,  0.,  0.],
        [ 9.,  0.,  0., ...,  6.,  0.,  0.],
        ...,
        [11.,  0.,  6., ...,  0., 13.,  0.],
        [ 3.,  0.,  0., ..., 13.,  0.,  0.],
        [ 0.,  0.,  0., ...,  0.,  0.,  0.]])
```

**Chart 2: co-occurrence matrix**

**- Step 3: Recommend restaurant for each user**

In order to reach our target, to provide 3 more restaurants to each customer, I build a function could use co-occurrence matrix to provide recommended restaurants for each user. The underlying method is multiple the numpy array represented each user and the co-occurrence matrix we obtained in step 2.

I designed two options to select the user, user's index in the dataset or user's id. Following list show a sample result to recommend 3 more restaurants for a user with index 2. For this user, Morimoto is the first restaurant we will recommend with highest scores in 118.0 and so on.

```
[['1', 'Morimoto', 118.0],
 ['2', 'The Spotted Pig', 106.0],
 ['3', 'EN Japanese Brasserie', 66.0]]
```

In addition, I build a function with using for loop to recommend the restaurant for each user in the dataset. So the co-occurrence matrix could provide recommend for each customer in the Hoboken_restaurants_reviews_cleaned.csv dataset.

**- Step 4: Check the result**

**Table 15** shows a sample of the recommendation result dataset for co-occurrence matrix method. Each user will receive recommendation of restaurants specifically.

| | user_id | recommendation |
|---|---|---|
| 0 | ---xAZNw9fFPBoy7jmkA2A | [['1', 'Gino's Pizzeria', 9.0], ['2', 'Noodlef... |
| 1 | --68ZwhCrUJUmCXXkMTMKw | [['1', 'The Cheesecake Factory', 3.0], ['2', '... |
| 2 | --8M2DZ9JkDwTveuRhLPTQ | [['1', 'Morimoto', 118.0], ['2', 'The Spotted ... |
| 3 | --ARr3m5JsxaX3DTUVQW7w | [['1', 'The Spotted Pig', 143.0], ['2', 'EN Ja... |
| 4 | --CZJeSIpxwQ0VULjnM57w | [['1', 'La Isla Restaurant', 48.0], ['2', 'Ama... |

**Table 15: co-occurrence matrix result**

These are two samples of recommendation result for the user with id '---xAZNw9fFPBoy7jmkA2A' and '--ARr3m5JsxaX3DTUVQW7w'. The model recommended three more restaurants for each user which stored in the recommendation column. Each cell is a list include three results. The first item (1, 2, 3) in the list means the ranking, the second item represent the name of restaurant, and the third item(9.0, 8.0, 6.0) means the weights/scores for each restaurant.

The weights/scores for different users are in different range according to the shape and values of co-occurrence matrix. For example, for the user with id '---xAZNw9fFPBoy7jmkA2A', the weights are 9.0, 8.0 and 6.0, while the weights are 143.0, 127.0 and 118.0 for the user with id '--ARr3m5JsxaX3DTUVQW7w'.

```
[['1', 'Gino's Pizzeria', 9.0], ['2', 'Noodlefan', 8.0], ['3', 'Fox and Crow', 6.0]]
```

```
[['1', 'The Spotted Pig', 143.0], ['2', 'EN Japanese Brasserie', 127.0], ['3', 'Del Posto', 118.0]]
```

## co-occurrence matrix summary:

In general, the co-occurrence matrix is easier to build than the other methods and it could build with dataset in any size, so it could be used for sparse dataset. However, the limitations for co-occurrence matrix is obvious. For example, It can only be built based on the user's past behaviors and it will ignore the user's rating for restaurants.

## Part 3 - 2 Collaborative Filtering

- My second recommendation system is collaborative filtering recommednation system. This method is tring to find similar users or items, evaluate the scores of those users/items, and recommend given user the items they will probably rate high. There are two different approaches in Collaborative Filtering: user-based collaborative filtering and item-based collaborative filtering. Basically, all of those two methods contains two steps:
  - First Step: Find out how many users/items in the database are similar to the given user/item.
  - Second Step: Assess other users/items to predict what grade you would give the user of this product, given the total weight of the users/items that are more similar to this one. I will include the user-based collaborative filtering in this project.
- Algorithms/Equations:
  - Similarity Calculation: The first step for collaborative filtering is calculate the similarity between the given user and the users in the dataset. There are multiple different methods to calculate the similarity and I will compare three popular methods, cosine similarity, jaccard similarity and pearson similarity.

    1. cosine similarity:

$$\text{Cosine similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

    2. Jaccard Similarity:

$$\text{Jaccard similarity} = J(\mathbf{x}, \mathbf{y}) = \frac{\sum_i \min(x_i, y_i)}{\sum_i \max(x_i, y_i)}$$

    3. Pearson Similarity:

$$\text{Pearson similarity} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

  - Recommend_items: I will use weighted arithmetic mean according to the degree of similarity to fill empty cells in the recommendation-result table. The equation will be showed as follows:

$$\bar{x} = \frac{w_1 x_1 + w_2 x_2 + \cdots + w_n x_n}{w_1 + w_2 + \cdots + w_n}.$$

- Tools:
  I will use graphlab libaray in Python to build the recommendation system.

**- Step 1: Data Loading**

The data I input is the **Table 12**, a groub-by dataset for collaborative filtering.

Since I use the graphlab library to build the collaborative filtering recommendation, I convert the pandas dataframe into graphlab dataframe, which is similar with the pandas dataframe but is suitable for function in graphlab library. In order to evaluate the performance of different methods to calculate similarity, I split the dataset into training set and test set with a ratio in 80/20.

```
This non-commercial license of GraphLab Create for academic use is assigned to nman@stevens.edu and will expire on
April 25, 2019.
```

**- Step 2: calculate similarity**

In this part I will test three different methods to calculate similarity and compare the precision and recall for each method. The three methods will be showed as follows:

1. cosine similarity;
2. Jaccard Similarity;
3. Pearson Similarity.

The definition of precision and recall will be showed as follows:

- Recall: It represent the ratio of items that a user likes were actually recommended.
- Precision: It means the user actually liked how many items out of all the recommended items.

**- Cosine similarity**

Similarity is the cosine of the angle between the 2 vectors of the item vectors of A and B.

$$\text{Cosine similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Precision and recall summary statistics by cutoff

| cutoff | mean_precision | mean_recall |
|---|---|---|
| 1 | 0.029708085766 | 0.0227225509452 |
| 2 | 0.0267372771894 | 0.0412972804151 |
| 3 | 0.0233646201096 | 0.0540945097843 |
| 4 | 0.0222164815293 | 0.0675257182038 |
| 5 | 0.0208731593903 | 0.0792679885142 |
| 6 | 0.0196475214558 | 0.089786578108 |
| 7 | 0.0189442865754 | 0.10113746566 |
| 8 | 0.0180293636442 | 0.109458753029 |
| 9 | 0.0172699178124 | 0.117481097992 |
| 10 | 0.0170240247998 | 0.128617790437 |

[10 rows x 3 columns]

**Table 16: Cosine precision and recall**

. . .

In **Table 16**, along with the increase of training epoches, the scores of precision and recall are enhance accordingly. In the cutoff 10, the precision is 0.0170 and recall is 0.1286.

**- Jaccard Similarity**

Jaccard similarity is based on the number of users which have rated item A and B divided by the number of users who have rated either A or B.clicked.

$$\text{Jaccard similarity} = J(\mathbf{x}, \mathbf{y}) = \frac{\sum_i \min(x_i, y_i)}{\sum_i \max(x_i, y_i)}$$

Precision and recall summary statistics by cutoff

| cutoff | mean_precision | mean_recall |
|---|---|---|
| 1 | 0.0296219753724 | 0.0229579193542 |
| 2 | 0.0257039524671 | 0.0395347799249 |
| 3 | 0.0231349923936 | 0.0534933772276 |
| 4 | 0.0220227331439 | 0.0681769636154 |
| 5 | 0.0205631619737 | 0.0788656554071 |
| 6 | 0.0193604868108 | 0.08881618977 |
| 7 | 0.0190180954841 | 0.101965200101 |
| 8 | 0.0182338758288 | 0.1114334695 |
| 9 | 0.0173560282059 | 0.118891347166 |
| 10 | 0.0167743046586 | 0.127352106486 |

[10 rows x 3 columns]

**Table 17: Jaccard precision and recall**

. . .

**Table 17** shows that along with the increase of training epoches, the scores of precision and recall are enhance accordingly. In the cutoff 10, the precision is 0.0167 and recall is 0.1274.

**- Pearson Similarity**

Pearson Similarity is the pearson coefficient between the two vectors.

$$\text{Pearson similarity} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Precision and recall summary statistics by cutoff

| cutoff | mean_precision | mean_recall |
|---|---|---|
| 1 | 0.000172220787049 | 0.000129165590287 |
| 2 | 8.61103935245e-05 | 0.000129165590287 |
| 3 | 8.61103935245e-05 | 0.000215275983811 |
| 4 | 8.61103935245e-05 | 0.000301386377336 |
| 5 | 8.61103935245e-05 | 0.00038749677086 |
| 6 | 8.61103935245e-05 | 0.000416200235368 |
| 7 | 8.61103935245e-05 | 0.000502310628893 |
| 8 | 0.000118401791096 | 0.000710410746577 |
| 9 | 0.000124381679535 | 0.000882631533626 |
| 10 | 0.000129165590287 | 0.00105485232068 |

[10 rows x 3 columns]

**Table 18: Pearson precision and recall**

. . .

. . .

In **table 18**, along with the increase of training epoches, the scores of precision and recall are enhance accordingly. In the cutoff 10, the precision is 0.0001 and recall is 0.0011.

So, with comparison of these three methods to calculate similarity, the cosine similarity perform better than the others, while the Jaccard similarity only slightly lower than cosine similarity. So either cosine similarity or jaccard similarity would be good for the collaborative filtering.

Now I will use cosine similarity and jaccard similarity to recommend restaurants for each customer in the dataset.

**- Step 4: Recommend top 3 rating restaurants to each customer**

Cosine Similarity Result looks like **Table 19**. Jaccard Similarity result looks like **Table 20**. **Table 21** is the Pearson similarity result. Each customer receive three recommended restaurants ranking by the score.

| user_id | restaurant_name | score | rank |
|---|---|---|---|
| fsq7a4Mog2pJ00er6DuHvw | Los Tacos No.1 | 0.160737276077 | 1 |
| fsq7a4Mog2pJ00er6DuHvw | Decoy | 0.0817471146584 | 2 |
| fsq7a4Mog2pJ00er6DuHvw | Artichoke Basille's Pizza ... | 0.0772345662117 | 3 |
| J_nwnpiV4ZqohauRp7XgEQ | Los Tacos No.1 | 0.267895460129 | 1 |
| J_nwnpiV4ZqohauRp7XgEQ | Decoy | 0.136245191097 | 2 |
| J_nwnpiV4ZqohauRp7XgEQ | Artichoke Basille's Pizza ... | 0.12872427702 | 3 |

[113163 rows x 4 columns]

**Table 19: Cosine Result**

| user_id | restaurant_name | score | rank |
|---|---|---|---|
| fsq7a4Mog2pJ00er6DuHvw | Los Tacos No.1 | 0.0233147740364 | 1 |
| fsq7a4Mog2pJ00er6DuHvw | Decoy | 0.0152284502983 | 2 |
| fsq7a4Mog2pJ00er6DuHvw | Artichoke Basille's Pizza ... | 0.0133729577065 | 3 |
| J_nwnpiV4ZqohauRp7XgEQ | Los Tacos No.1 | 0.0233147740364 | 1 |
| J_nwnpiV4ZqohauRp7XgEQ | Decoy | 0.0152284502983 | 2 |
| J_nwnpiV4ZqohauRp7XgEQ | Artichoke Basille's Pizza ... | 0.0133729577065 | 3 |

[113163 rows x 4 columns]

**Table 20：Jaccard Result**

| user_id | restaurant_name | score | rank |
|---|---|---|---|
| fsq7a4Mog2pJ00er6DuHvw | Gold Roast Cafe | 5.0 | 1 |
| fsq7a4Mog2pJ00er6DuHvw | 52 Restaurant | 5.0 | 2 |
| fsq7a4Mog2pJ00er6DuHvw | Luna's Pizzeria & Restaurant | 5.0 | 3 |
| J_nwnpiV4ZqohauRp7XgEQ | Gold Roast Cafe | 5.0 | 1 |
| J_nwnpiV4ZqohauRp7XgEQ | 52 Restaurant | 5.0 | 2 |
| J_nwnpiV4ZqohauRp7XgEQ | Luna's Pizzeria & Restaurant | 5.0 | 3 |

[113163 rows x 4 columns]

**Table 21: Pearson Result**

I am curious about why the Pearson similarity result seems similar for different users.

```
restaurant_name
Manà Restaurant                    5.000000
Wah Yoan                           5.000000
Taquitos Mexicanos                 5.000000
Bocata's Deli                      5.000000
Azteca Taqueria Restaurant         5.000000
Ujala Kabab                        5.000000
Manhattan Bar Grill & Lounge       5.000000
MONDO - DOGS                       5.000000
Ventura Restaurant                 5.000000
Terranova Deli                     5.000000
El Salvador Restaurant             5.000000
52 Restaurant                      5.000000
Gold Roast Cafe                    5.000000
Griot Cafe                         5.000000
Taste of Greece                    4.857143
Andrea Salumeria                   4.840000
Toscana                            4.789474
Royal Grill Halal Food             4.785714
Salerno Salumeria                  4.700000
Losurdo's Italian Bakery & Deli    4.700000
Name: user_rating, dtype: float64
```

We could find that all the results are almost same when using pearson model. Since all the recommended restaurant have an average rating of 5. So the pearson recommender is not accurate enough.

**Collaborative Filtering Summary:**

The collaborative filtering method seems better than the co-occurrence matrix, since it take the user's rating into account, which means it would be more relative to customers' real demand and it is still easy to understand. In addition, the collaborative filtering system will perform better and better with more and more users in the dataset. However, it also only focus on the privious user's behaviors like the co-occurrence matrix method, so it doesn't have the ability to update accordingly. Moreover, it will be limited by data sparsity. If we could not collect enough data and the data sparsity is too low, the collaborative filtering will not perform well.

### Part 3 - 3 Matrix Factorization

- Background: The Matrix Factorization is the third method I used to build recommendation system. The assumption of matrix factorization is that each user have tendency to like different features of item. For example, in Hoboken restaurant dataset, the features might be theme and style of the restaurant, environment, service quality, food quality of restaurants and so on. Different user have different expectation and preferences for different features. In addition, each item, restaurant in this case, have features in different degree. We assume that the user will rate higher to a new restaurant with more features this user like and will rate lower to a new restaurant with less features this user like.
- Algorithms/equation:
  - Basic Algorithm:

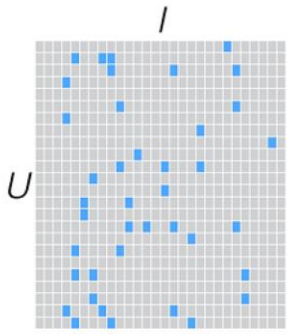
$$\hat{r}_{ui} = \sum_{k=1}^{K} U_{uk} V_{ki}$$

```
- R_ui: user u's rating for item i
- U_uk:item i's similarity with a feature k
- V_ki:user u's preference for a feature k
```

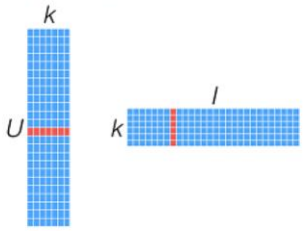
  - Algorithm with bias:

$$\hat{r}_{ui} = \sum_{k=1}^{K} U_{uk} V_{ki} + \mu + b_u + b_i$$

  - This algorithm could also include more parameters like customer's feedback, taste changes according to time and so on.
- Tools:
  - I will use sklearn and numpy in python to build the matrix factorization recommendation system.
  - The input data looks like the example as follows:




  - And then I will seperate this user and item matrix into two marix look like follows:




  - K means features in this case so each row in the users and features matrix will represent the customer's preference for different features and each column in the features and items matrix will represent the degree of a restaurant have such features.

**- Step 1: Data Preparation**

In this step, I will convert the input data, **Table 13: A matrix like table with rating scores** into a numpy matrix.

**- Step 2: Extract features k**

I could use sklearn library to extract features k and seperate the matrix into two matrices, the user and features matrix as well as the features and items matrix. In this part, we reduce the dimension of original matrices and got two new matrices.

```
- new matrix 1: represent user i's preference of different features k
- new matrix 2: represent the item j's similarity to features k
```

After building the function, I randomly use features_no= 10 at the first try to create two new matrices. The shape of first matrix is (44949, 10) and second matrix is (10, 302).

**- Step 3: Recommender**

In this part, I will get the recommender matrix by dot product of these two matrices with using numpy.dot()function. The recommender matrix will look like **table 22**.

| | user_id | 10th & Willow Bar & Grill | 52 Restaurant | 8th Street Tavern | Adoro Lei | Aether Game Cafe | Ahri's Kitchen | Ainsworth Hoboken | Aldys Restaurant | Ali Baba | ... | White Star Bar |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | ---xAZNw9fFPBoy7jmkA2A | 0.003127 | 8.309046e-07 | 0.000713 | 0.000139 | 0.000200 | 0.000706 | 0.001725 | 0.000039 | 0.001400 | ... | 0.002331 |
| 1 | -68ZwhCrUJUmCXXkMTMKw | 0.000378 | 9.981413e-08 | 0.000086 | 0.000032 | 0.000024 | 0.000096 | 0.000209 | 0.000005 | 0.000168 | ... | 0.000285 |
| 2 | --8M2DZ9JkDwTveuRhLPTQ | 0.000000 | 0.000000e+00 | 0.000000 | 0.005092 | 0.000000 | 0.000156 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 |
| 3 | --ARr3m5JsxaX3DTUVQW7w | 0.000000 | 0.000000e+00 | 0.000000 | 0.000000 | 0.000000 | 0.001011 | 0.000159 | 0.000000 | 0.000000 | ... | 0.000000 |
| 4 | --CZJeSIpxwQ0VULjnM57w | 0.007670 | 2.037205e-06 | 0.001752 | 0.000336 | 0.000491 | 0.001732 | 0.004230 | 0.000096 | 0.003434 | ... | 0.005721 |

5 rows × 303 columns

**Table 22: recommender matrix**

**- Step 4: Export the Result**

In order to recommend new restaurants to each customer, I will remove the score for the restaurant each customer have already attended. And then each customer will receive three recommended restaurants. A sample result will look like following list:

```
. . .
```

```
[['1', 'The Cuban Restaurant and Bar', 0.0017525621561418668],
 ['2', 'La Isla Restaurant', 0.0015292799793646238],
 ['3', 'Morimoto', 0.0014210628091436324]]
```

I build a function with using for loop to recommend the restaurants for each customer in the dataset and save the result as 'matrix_factorization_result.csv'.

**- Step 5 Check the result**

The **Table 23** shows a sample result of matrix factorization recommendation. The first item in the result list represent the ranking, the second item is the restaurant's name and the third item is the scores of each restaurant.

| | Unnamed: 0 | user_id | Recommendation |
|---|---|---|---|
| 0 | 0 | ---xAZNw9fFPBoy7jmkA2A | [['1', 'The Cuban Restaurant and Bar', 0.01458... |
| 1 | 1 | --68ZwhCrUJUmCXXkMTMKw | [['1', 'The Cuban Restaurant and Bar', 0.00175... |
| 2 | 2 | --8M2DZ9JkDwTveuRhLPTQ | [['1', 'Perry St', 0.23603528958327633], ['2',... |
| 3 | 3 | --ARr3m5JsxaX3DTUVQW7w | [['1', 'STK Downtown', 0.037760289982801606], ... |
| 4 | 4 | --CZJeSIpxwQ0VULjnM57w | [['1', 'The Cuban Restaurant and Bar', 0.03576... |

```
"[['1', 'The Cuban Restaurant and Bar', 0.01458922419511693], ['2', 'La Isla Restaurant', 0.012734421654190975],
['3', 'Pilsener Haus & Biergarten', 0.00954516675514815]]"
```

**Table 23: A sample of matrix factorization result**

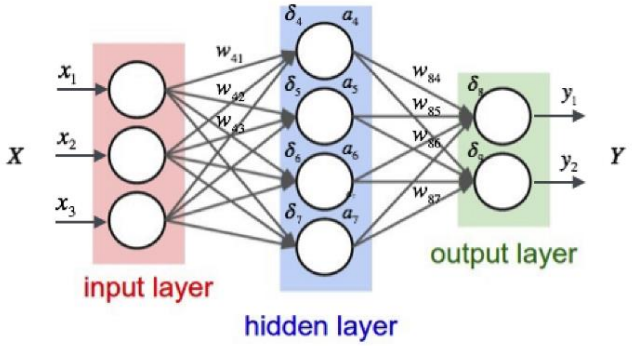
**Matrix Factorization Summary：**

In general, matrix factorization method is more elegant than previous two methods. It could easily apply to masive dataset and we could add more new parameter(bias, historical feedback, time changes and so on) to the algorithms accordindly. However, the data sparisity will also limit its performance.

### Part 3 - 4. Deep Learning - neural network:

I have already build three recommendation systems with popular recommendation algorithms. Now, I will try to use deep learning method - Neural Networks to build a recommendation system. The objective is same as previous methods. I will recommend 3 resturants each customer most likely to be interested to them.

Neural Network method simulate the processes that human or animal recognized the object. An artificial neural network includes one input layer, at least one hidden layer with nodes(kernel) and bias, one output layer including defined target classes. Each layers and nodes will be randomly generated when launching the training process. Or if we apply transfer learning, the layers and nodes could be like a pre-trained model. Input variables will be normalized firstly, multiple with each neural weights, sum together, plus an bias and connect with the next layer. The target of each training process is trying to reduce the differences between real labels and the predicted results. We could use sigmoid, adam or other methods to optimize the reducing process. The revised information will transfer back to each layer and nodes as well as modify the weights and bias. Finally, we will get a neural network with relatively accurate weights and bias for each node on each layer.




Before fiting the variable into neural network, we should normalize the input variables for each row first, since the Neural Network Algorithm assume that the input variable is an number between 0 to 1.

Tools:

- I will use sklearn in python to build the NN recommendation system.

Model design:

- Label:
  - Restaurants Name (302 classes in total)
- According to the EDA part, I will include some variables in the model.
- Input variables:
  - user_rating
  - restaurant rating
  - restaurant price
  - restaurant type
- The first three variables are numerical data and the restaurant type is data in text format. So I will also apply NLP to convert the restaurant_variables into the data type that NN could recognize.
- I design this model because the Neural Network model will return a restaurant that most likely match with the input variables. In a situation that accuracy eaqual to 100%, NN model will return the label (restaurant) we inputed. At the same time, it will also give a probability for each class in the dataset, which will show how does one restaurant is likely to be the accurate one. In another word, in order to build a recommendation system based on NN, I could rank both of these classes(302 restaurants) by the probability rate. And then, I could remove the accurate label in the ranking list and return other classes from higer probability to lower. So, the top n restaurants in this ranking will be the restaurants that are most similar to the accurate one.

### Part 3 - 4 - 1: Data Preparation

**Table 24** is the dataset I use to build the Neural Network recommendation system. I will use the restaurant_name as the label and use user_rating, restaurant_rating, restaurant_price and restaurant_type as the independent variables.

| | user_id | restaurant_name | user_rating | restaurant_rating | restaurant_price | restaurant_type |
|---|---|---|---|---|---|---|
| 0 | dRuCO4NYO7zyAF8-CeJmZg | Grand Vin | 5 | 4.0 | 2.0 | cocktail bars wine italian |
| 1 | f36YZ1cA291bNtMHXWtu1Q | Grand Vin | 4 | 4.0 | 2.0 | cocktail bars wine italian |
| 2 | -xYUKfWQTaB-7BeizsQA3w | Grand Vin | 5 | 4.0 | 2.0 | cocktail bars wine italian |
| 3 | tt1vLgAP5UpRXAKJLT2KWg | Grand Vin | 4 | 4.0 | 2.0 | cocktail bars wine italian |
| 4 | -K79Xep4lElqlChsJYWuiQ | Grand Vin | 5 | 4.0 | 2.0 | cocktail bars wine italian |

**Table 24: Neural Network input data**

**Step 1: Normalized numerical data**

In this step, I use 'max and min' normalized method to encode the user_rating, restaurant_rating and restaurant_price column into normalized value.

**Step 2: Normalized text data by applying NLP- tf-idf**

In this step, I use 'tf-idf' to convert the restaurant_type data into weights.

**Step 3: Combine step 1 and step 2 and store the new matrix like dataframe**

**Table 25** shows the new matrix like dataframe. Each row represent each customer. The 0 to 128 columns(129 in total) represent 129 features of restaurant type and the weight of preferences that each customer tend to like each feature. User_rating, restaurant_rating and restaurant_type columns are the encoded value according to original dataframe. Output column is the restaurant_name and will also be the label for each observation.

| user_id | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 123 | 124 | 125 | 126 | 127 | 128 | user_rating | restaurant |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| dRuCO4NYO7zyAF8-CeJmZg | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.309521 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.670605 | 0.0 | 0.0 | 0.004578 | 0. |
| f36YZ1cA291bNtMHXWtu1Q | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.309521 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.670605 | 0.0 | 0.0 | 0.003663 | 0. |
| -xYUKfWQTaB-7BeizsQA3w | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.309521 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.670605 | 0.0 | 0.0 | 0.004578 | 0. |
| tt1vLgAP5UpRXAKJLT2KWg | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.309521 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.670605 | 0.0 | 0.0 | 0.003663 | 0. |
| -K79Xep4lElqIChsJYWuiQ | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.309521 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.670605 | 0.0 | 0.0 | 0.004578 | 0. |

5 rows × 133 columns

**Table 25: Neural Network dataset**

## Part 3 - 4 - 2: Build Neural Network model and tune to find a best model

**Step 1: Data loading and spliting**

The first target in this step is to figure out a best model, so I split the input data into two set with ratio, 80/20.

**Step 2: Build the model**

At the first try, I randomly generate a neural network model with following parameters:

- hidden layer siz e= (16,6)
- solver = adam
- learning rate = 0.001

After training the model with training set and predicting the model with test set, the accuracy rate is 86.28% for the test set.

```
Accuracy: 0.8628962004958789
Wall time: 3min 10s
```

**Step 3:Tuning**

In order to get a better model, I tune some parameters and obtain a better model with 87.77% accuracy rate and parameters as follows:

- hidden layer siz e= (25,6)
- solver = adam
- learning rate = 0.005

```
{'hidden_layer_sizes': (25, 6), 'learning_rate_init': 0.005}
Wall time: 2h 56min 34s

Accuracy: 0.8777055551832741
```

**Step 4: Get recommendation result with best estimator**

Following table shows a sample result of the neural network model.

| | probability | restaurant_name |
|---|---|---|
| 0 | 1.504943e-37 | 10th & Willow Bar & Grill |
| 1 | 4.619613e-58 | 52 Restaurant |
| 2 | 5.929570e-134 | 8th Street Tavern |
| 3 | 7.670625e-54 | Adoro Lei |
| 4 | 2.107623e-145 | Aether Game Cafe |

```
[['1', 'Augustino's', 0.022578243355864183],
 ['2', 'Hudson Clearwater', 0.0003665265326626242],
 ['3', 'The Lobster Place', 0.00035943076450474433]]
```

**Step 5: Check the result**

Following table shows the final result that recommend 3 restaurants to each customers. The shape of the result is similar with the co-occurrence result and matrix factarization result. The only different is that the third item means the probability that the recommended restaurant match with the restaurants the user attended.

| | user_id | Recommendation |
|---|---|---|
| 0 | dRuCO4NYO7zyAF8-CeJmZg | [['1', 'Barbuto', 0.0034586612971036304], ['2'... |
| 1 | f36YZ1cA291bNtMHXWtu1Q | [['1', 'Barbuto', 0.0032257489325294257], ['2'... |
| 2 | -xYUKfWQTaB-7BeizsQA3w | [['1', 'Barbuto', 0.0034586612971036304], ['2'... |
| 3 | tt1vLgAP5UpRXAKJLT2KWg | [['1', 'Barbuto', 0.0032257489325294257], ['2'... |
| 4 | -K79Xep4lElqlChsJYWuiQ | [['1', 'Barbuto', 0.0034586612971036304], ['2'... |

```
"[['1', 'Barbuto', 0.0034586612971036304], ['2', 'Zack's Oak Bar & Restaurant', 0.00043441651480210933], ['3', 'The
Ear Inn', 0.00029910823588837521]]"
```

**Neural Network Summary:**

Neural Network recommendation system is different from previous three methods. The underlying concept is to predict the probability a restaurant matched with a customer's interests. This method could include more and take into account more parameters like customer's gender, their consumer behaviors, their social network relationship and so on. Also, along with more and more observations are using to feed the model, the accuracy of the model will be better and better. In addition, we could tune the model to find better parameters to build the neural network model. There are still space to tune and find a better model. However, the neural network recommendation system also have limitation that it could not perform well for a sparse dataset.

## Part 4: Comparison of the Recommendation System methods

In real world, there are multiple parameters to evaluate the performance of recommendation system, such as customer's satisfaction, diversity of recommendation, creativity, innovation, surprice degree, real time updated and cold start, profit of project. However, this project was built in offline sandbox, it is hard to use the online evaluation to detect the performance of these methods.

So I will compare these four recommendation system methods through describing their limitaions, advantages and application scenarios.

- Co-occurrencce Matrix
  - Advantage： Easy to understand; Start with any size dataset
  - Limitation： Only focus on previous behaviors; ignore customer's rating
- Collaborative Filtering
  - Advantage：Perform well in masive dataset; Take customer's rating into account
  - Limitation: Only focus on previous behaviors; rely on customer's rating ; Limited by data sparsity
- Matrix Factorization
  - Advantage：Take more parameters like bias, customer's feed back, changes of taste into account
  - Limitation: Limited by data sparsity
- Neural Network |
  - Advantage：Easy to add more independent variables to build the NN model; Performance could be elevated with more data; Performance could be enhanced by tuning
  - Limitation: Limited by data sparsity

# V. Project Conclusion (Ni)

Our main target of this project is to explore the insight from customer reviews from Yelp.com. We have two data sources, 1. Customer Reviews of Restaurants at Hoboken (74, 611 rows). We build a Web Scraper to scrape this dataset. 2. Full dataset yelp provided (321 million rows). we use yelp open source API to collect this dataset.

In addition, we have tree sub objectives. The first one is predicting stars based on customer's reviews and historical data. We use Random Forest algorithm in to predict the result. Our second sub objective is figuring out popular topics customer cared about, which will benefit owners of restaurants to improve their service quality. In this part, we applied NLP & Sentiment Analysis with using CV, tf-idf, LDA methods and machine Learning algotithms like Logistic Regression and Random Forest. The thrid sub target is recommending top 3 restaurants to each customer in the Hoboken dataset. In this part, we use four different recommendation methods including Co-occurrence Matrix, Collaborative Filtering, Matrix Factorization and a deep learning method Neural Network.

In general, we reach the main target we setup and figure out the insight from reviews data. We provide scores prediction, help owners of restaurants modify their services and products and create profit for both restaurateurs and yelp.com.

# VI. Bibliography:


- Predicting Yelp Ratings by Review Text and Metadata part:

  1. Bhandary, V. (2018, Febuary). Exploring yelp reviews dataset. Retrieved from Kaggle: https://www.kaggle.com/vksbhandary/exploring-yelp-reviews-dataset (https://www.kaggle.com/vksbhandary/exploring-yelp-reviews-dataset)
  2. Donges, N. (2018, Febuary 22). The Random Forest Algorithm. Retrieved from Towards Data Science: https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd (https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd)
  3. How to choose the number of hidden layers and nodes in a feedforward neural network? (2015, Febuary 6). Retrieved from CrossValidated (StackExchange): https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw (https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw)
  4. Liu, E. (2015, November 17). TF-IDF, Term Frequency-Inverse Document Frequency. Retrieved from http://ethen8181.github.io/machine-learning/clustering_old/tf_idf/tf_idf.html (http://ethen8181.github.io/machine-learning/clustering_old/tf_idf/tf_idf.html)
  5. Ng, A. (n.d.). Machine Learning. Retrieved from coursera: https://www.coursera.org/learn/machine-learning (https://www.coursera.org/learn/machine-learning)
  6. sikit-learn. (2017). Classifier comparison. Retrieved from scikit learn: http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html (http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html)
  7. Yelp Inc. (2018). Retrieved from Yelp: https://www.yelp.com/ (https://www.yelp.com/)
- Popular Topic part：

  1. http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html#sklearn.decomposition.LatentDirichletAllocation (http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html#sklearn.decomposition.LatentDirichletAllocation)
  2. http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)
  3. https://stackoverflow.com/questions/30102973/how-to-get-best-estimator-on-gridsearchcv-random-forest-classifier-scikit (https://stackoverflow.com/questions/30102973/how-to-get-best-estimator-on-gridsearchcv-random-forest-classifier-scikit)
  4. http://scikit-learn.org/stable/modules/naive_bayes.html (http://scikit-learn.org/stable/modules/naive_bayes.html)
  5. https://www.thoughtco.com/difference-between-accuracy-and-precision-609328 (https://www.thoughtco.com/difference-between-accuracy-and-precision-609328)
  6. https://www.itl.nist.gov/div898/handbook/eda/section1/eda11.htm (https://www.itl.nist.gov/div898/handbook/eda/section1/eda11.htm)

- Recommendation System part:
  1. Collaborative Filtering Recommender Based on Co-occurrence Matrix - Marjan Sterjev
  2. Moviex.ai - Movie recommendations with Deep Learning
  3. Deep Neural Networks for YouTube Recommendations - Paul Covington, Jay Adams, Emre Sargin
  4. Matrix Factorization Techniques for recommender system - Yehuda Koren, Robert Bell, Chris Volinsky
  5. https://learnforeverlearn.com/cooccurrence/ (https://learnforeverlearn.com/cooccurrence/)
  6. https://www.analyticsvidhya.com/blog/2016/06/quick-guide-build-recommendation-engine-python/ (https://www.analyticsvidhya.com/blog/2016/06/quick-guide-build-recommendation-engine-python/)
  7. https://hub.packtpub.com/building-recommendation-engine-spark/ (https://hub.packtpub.com/building-recommendation-engine-spark/)