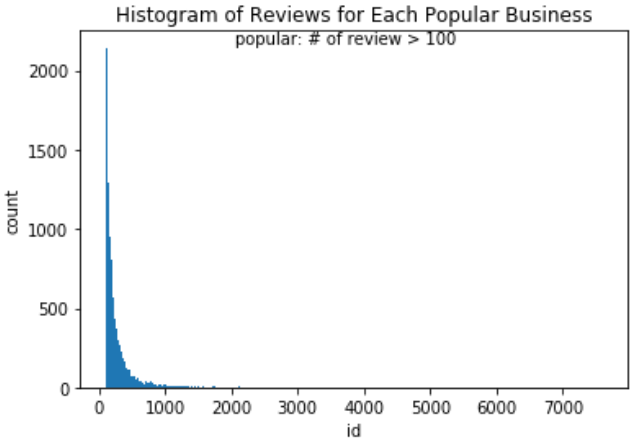
IEORE 4571 Personalization Project 2

Business Recommender System

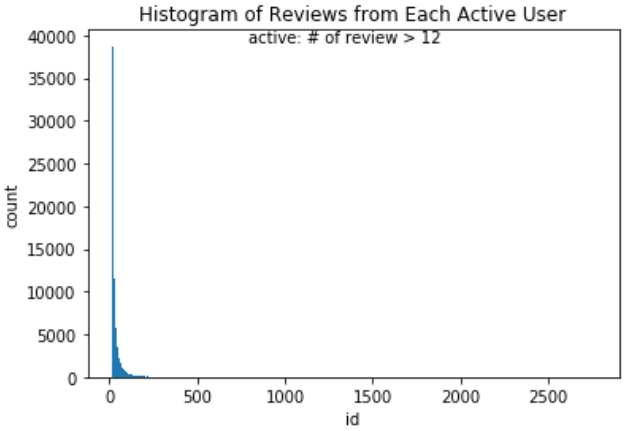
Group members

- Jing You, jy3035@columbia.edu, jy3035
- Yi Ping Tseng, yt2690@columbia.edu, yt2690
- Zihui Zhou, zz2694@columbia.edu, zz2694

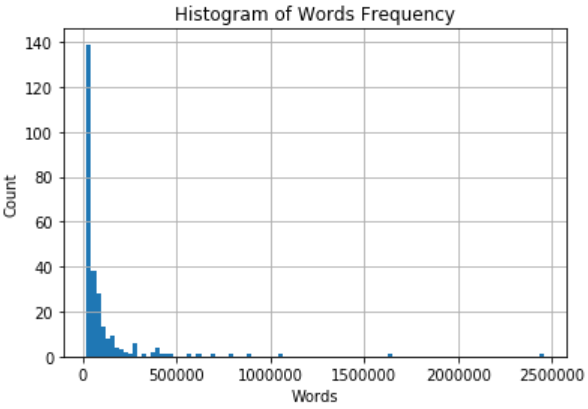
1. **Background and objectives**
    a. **Business objectives:** We want to predict the rating for users on business that they have never gone before, which can be useful in recommending new business to users. In order to build and evaluate our recommendation models, we hold on the users' last reviews in the test dataset and make a prediction for the scores.
    b. **Overview:** As a leader in business directory service and crowd-sourced review forum, Yelp has made a great profit with charging the companies based on CPC (cost-per-click). Developing personalized recommendations based on customers' behavior data can be very useful when pushing the desired business to the users, for which may retain loyal customer relationships and make profits in the long run. In this specific case, business recommendations are given to customers based on the information of the business, users, and also the potential review. In our project, we try to build a system to serve users by recommending new business they may like. We explore this topic through analyzing "yelp2019" dataset , in which we built four models using collaborative filtering, factorization machine, text-ming and further stacked into an ensemble to achieve the best performance.
2. **Data-analysis**
    a. **Distribution plot**
        i. Plot 1 (histogram of popular business)



ii. Plot 2 (histogram of active users)



iii. Plot 3 (histogram of words' frequencies)



b. **Popular business:** We defined the business with over 100 reviews as "popular business". It is clear from the Plot 1 that the percentage of such popular business reviews among all reviews is 55.20%, which indicates that some business is rated extremely often than the others.

c. **Active users:** We defined the users with over 12 reviews and at least 1 useful review as the "active users". Plot 2 shows that the percentage of active user

reviews among all reviews is 51.40%, which suggests that the rest of the users have very few reviews. Therefore, it is likely that we may run into a "cold start" problem for our recommendation system.

d. **Sparsity:** Both the business and the users has very skewed distribution, it is likely that the dataset would be very sparse. From our calculation, the sparsity level is 99.91%

e. **Top 100 frequency words:** We would also like to explore if we can predict the ratings based on the review contents, thus we collect all review texts and find the top 100 frequency words. Obviously words like 'loved, excellent, special..' can be indicators of positive ratings and words like 'ok, however, bad ..'may be indicators of lower ratings. There are also less descriptive words that may do not contribute much to ratings' prediction.

```
Top 100 Frequency Words in Reviews
array(['everyone', 'enjoy', 'once', 'decided', 'tasted', 'away', 'cooked',
       'need', 'ok', 'each', 'same', 'should', 'fun', 'home', 'probably',
       'things', 'loved', 'looking', 'huge', 'quite', 'prices', 'full',
       'feel', 'friend', 'nothing', 'enjoyed', 'however', 'special',
       'into', 'clean', 'large', 'friends', 'dishes', 'inside', 'options',
       'atmosphere', 'places', 'stars', 'excellent', 'spicy', 'enough',
       'top', 'coming', 'beef', 'awesome', 'wanted', 'find', 'visit',
       'check', '10', 'last', 'tasty', 'perfect', 'thing', 'where',
       'hour', 'dish', 'big', 'quality', 'asked', 'busy', 'cream', 'spot',
       'overall', 'ever', 'bad', 'am', 'vegas', 'being', 'different',
       'another', 'fried', 'long', 'something', 'every', 'drink', 'worth',
       'flavor', 'meat', 'took', 'sweet', 'hot', 'though', 'minutes',
       'give', 'now', 'her', 'most', 'fries', 'next', 'happy', 'price',
       'taste', 'rice', 'favorite', 'new', 'many', 'll', 'both', 'said'],
      dtype='<U10')
```

3. Data-preprocessing (following the previous business goals & rules)

a. Cleaning Business

i. **Closed business**: The project's final goal is to recommend new business to the users. Even though many closed businesses have a lot of reviews , it is not meaningful to recommend closed business to users. Also considering the fact that closed business may be closed long time ago and has relatively less important business values in comparison to the open business, we remove them from the business dataset and only keep the open business in the dataset. To avoid disappointing users by recommending them closed business, we decided to remove the closed ones from our dataset during data pre-process. If we allow the closed business remain in the dataset, the recommendation model would suggest them to the users and the recommendation quality drops significantly. In this way, we may even lose users rather than retaining and gaining users through making good recommendations.

ii. **Popular business:** Since more reviews provides more information on the business, which could improve the accuracy, we remove the business which have less than 100 reviews.

b. Cleaning users

i. **Active user:** Users with review_count >= 12 and useful >= 1

1. With fewer training data, the recommendation model commonly result in a higher bias and less accurate predictions. Comparatively, the recommendation model will perform worse for users with less than five reviews. Taking this factor into consideration, it is better to remove them from the dataset.
2. **Another business concern is that not all users write comments and give ratings cautiously:** for example, some users rate business arbitrarily and write poor-quality reviews. Since their ratings are relatively random, it is less accurate to give predictions on those users' last ratings based on their previous ratings. To resolve this concern, we only keep users with at least one useful vote in the dataset to ensure the quality of both ratings and reviews.
3. Above all, we set the following constraints on users: users should write at least 12 reviews and receive at least one useful vote.

c. Cleaning Reviews

i. **Recent review:** Since people's tastes and opinions change over time, older reviews are less useful and informational for predicting the latest rating. In order to keep the dataset consistent and make better predictions, we remove the reviews that were written before 2017-01-01.

ii. **Repeated review:** For those users who review the same business for more than once, we only keep the most recent ones (including ratings & texts). To ensure the ratings are unique and up-to-date, we only keep the most recent ratings from a specific customer given on a specific business.

d. After preprocessing the data

i. The number of reviews is reduced from 6685900 to 392471

ii. The number of business is reduced from 192609 to 9017

iii. The number of users is reduced from 1637138 to 48644

e. Open source tool for data format conversion

i. https://www.yelp.com/dataset/documentation/main
https://github.com/Yelp/dataset-examples

ii. The original dataset provided by Yelp is in Javascript Objective Notation (JSON) format. In order to flatten the data, we use the open source tool from the above link to convert it into Comma Separated Variables (CSV).

4. **Setup**
    a. **Rating for prediction:** Since we are aiming to predict users' last ratings, we do not want to include their last ratings when training the model to avoid influencing the accuracy. Then, we use their last ratings as the test set, and the remaining data as the train set. Throughout our project, we use the same train dataset and test dataset for every model.
    b. **Evaluation methods:** We implement the mean square error (MSE) evaluation metric across our modelling process to evaluate performance, which is calculated in the formula $MSE = \frac{1}{n}\Sigma(yi - \widehat{yi})^2$. We also use MSE as a reference for tuning the hyperparameters to optimize the models.
5. **Baseline Model -- Collaborative filtering with ALS**
    a. Model description
        i. Collaborative filtering makes use of the matrix factorization algorithm to represent users preferences with vectors in lower dimensions. Apache Spark ML implemented such model-based collaborative filtering with ALS model. Our dataset consists of users, business, and the ratings are still relatively sparse. Therefore, ALS model is suitable for our dataset as it is good at dealing with the sparsity of data.
    b. Evaluation methods
        i. Cross-validation setup
            1. **Number of latent dimensions:** The number of latent dimensions used in the matrix factorization impacts our result of prediction. Increasing the number of latent dimensions would improve the predictions until the number is too high that the model tends to overfit. We vary the number of latent dimensions from the following: {5, 30, 70}.
            2. **Parameter of regularization:** ALS uses L2 regularization (Ridge regularization). Increasing the regularization parameter would avoid overfitting but we also want to limit the parameter that it won't cause a great increase in bias in the model. As the regularization parameter used in the ALS model also impacts the prediction results, we tune this hyperparameter to figure out which L2 Ridge regressor would fit our dataset and the model better. We vary the parameter of regularization from the following: {0.1, 1, 10}.
    c. Model Performance
        i. After completing the cross validation with grid search among the parameters the number of latent dimensions and the parameter of

regularization, we find that MSE achieve the minimum 2.4158 when the number of latent dimensions is 5.

d. Conclusion of Collaborative Filtering model

i. Selecting the best hyper-parameters to construct the model, and use this model to predict the ratings of the final reviews for users, and save it to the

6. Factorization Machine model

a. Model description

i. Able to represent users and business in a lower dimensional latent space, matrix factorization is a highly efficient class of algorithms for rating predictions in recommender systems. The algorithms are based on matrix concatenations, which consist of one-hot encoding for selected features. In this way, the algorithms contain all cross-entity relationships, and achieve a higher accuracy in prediction.

b. **TensorFlow implementation** Factorization Machine based on the paper *Factorization Machines with libFM*

i. Online open source tffm:

1. https://github.com/geffy/tffm
2. Citation: Steffen Rendle (2012): *Factorization Machines with libFM*, in ACM Trans. Intell. Syst. Technol., 3(3), May.

ii. Matrix concatenation

1. Since most users and business are different from each other, the concatenation matrix will be really sparse and less efficient if we transform those IDs into one-hot-encodes. Therefore, we calculate the frequencies of each user and business, and categorize them into one of the following four frequency quantile intervals: 0~25%, 25~50%, 50~75%, 75~100%.
2. After processing with the IDs, we concatenate (funny/useful/cool) votes, stars, average stars, review counts, one-hot-encoded user IDs, one-hot-encoded business IDs, and one-hot-encoded states as our feature vector X. We include these features because they have greater impacts on the rating prediction and recommendation suggestions. For example, we include the state location as a feature because people live in a certain state tend to have similar preferences and, therefore, rate similarly. For example, Maine people's ratings are 0.35 higher than people who live in Ontario. Besides, since each person also has different preference, we also want to include these discrepancies' influences on our rating prediction. Therefore, we consider (funny/useful/cool) votes,

review counts and average stars as feature vectors in our matrix concatenation.

iii. **Performance**

1. Mean Square Error (MSE): we implement the mean square error measurements in our model to evaluate its performance. The achieved MSE is 1.4964, which is a significant improvement compared to the baseline model.
2. Since factorization machine tffm uses L2 (Ridge regularization), the regularization parameter used in the als model also impacts the prediction results. We we use 3-fold cross validation to tune the hyperparameters order and rank from the following sets {1,2,3,4}, {5,10,15} correspondingly. The paired (3,10) achieves the minimum MSE with the value of 1.4964 .

iv. **Conclusion for the Factorization Machine model**

1. After tuning the hyperparameters under each case, we conclude our models by choosing the best hyperparameters that minimize evaluation metrics. Then, we use the optimized model to predict the most recent ratings and save the results along with the predictions in the 'tffm_test_Predict.csv'file.

7. Text-mining -- Logistic Regression

a. Model description

i. Since the ratings range from 0 to 5, we treat them as 6 categories. Therefore, we use Logistic Regression and Ridge Classifier from the sklearn library to solve this multiclass case.

b. Important steps

i. We first perform text transformation on the training set's texts using TF-IDF to evaluate the importance of each word. There are words but frequently appeared words such as "a" and "we" which do not tell much about the positiveness of the scores. Similarly, some words seldom appear are also less informational texts. In order to remove those words, we set the max_df to 0.95 and min_df to 0.05 in the count vectorizer to remove the top and lower 5 percentile of overall frequency.

c. Conclusion

i. We fit the Linear Regression model to the training dataset, and give predictions on the ratings of the test dataset. We save the test predictions as well as train predictions to 'textLR_testPredict.csv' and 'textLR_trainPredict.csv' files for the final model combination use. The logistic regression model gives us MSE as 1.10, much better than the baseline collaborative filtering model.

8. **Text-mining -- Stack Ensemble (random forests + naive Bayes + ridge classifier)**
    a. **Model description**
        i. Since each recommendation algorithm has some drawbacks, we combine three models with low correlations in their errors into an ensemble. In our project, we stack random forests, naive Bayes, and ridge classifier into an ensemble. For the random forests model, we set the minimum samples leaf as 0.1% of the overall data points to refine the trees' deepness and avoid being too specific. In this ensemble model, we still use the train set text processed by TF-IDF as our training dataset. Again, we use the sklearn library to perform the analysis.
    b. **Conclusion**
        i. We fit the ensemble model to the training dataset, and then predict on the ratings of the test dataset. We save the predictions as as well as train predictions as 'testEnsemble_trainPredict.csv' file for the final model combination use. The logistic regression model gives us MSE as 1.2511
9. **Model combination**
    a. After achieving the predicted ratings from the above four models, we want to combine their results to have a better final prediction. We append the final predictions from the four models to the same dataframe, and then run linear regressions on the the four columns test set
    b. **Prediction**
        i. Firstly, we run the linear regression on the first model combination (Collaborative Filtering + Factorization Machine + Text:Logistic + Text: Ensemble) to achieve the weights of each model's rating prediction. Then we multiply models' weight and their prediction ratings to calculate the final prediction ratings. Then, we use MSE to calculate the deviations of the final prediction ratings to the true rating value, and achieved MSE value as 0.85. However, we notice that the coefficient of Collaborative Filtering (CF) is negative, which means that there is a negative correlation. Standing on the statistical point, we decide to remove CF from the model combination.
        ii. Then, we run the linear regression on the second model combination (Factorization Machine + Text:Logistic + Text: Ensemble) to achieve the coefficients of the model's predicted ratings and multiply that by the predicted ratings to achieve the final predicted value. Calculating the accuracy by MSE, we achieve the final MSE score as 0.86.
10. **Conclusion**

Collaborative Filtering	Factorization Machine	Text-mining (Logistic Regression)	Text-mining -- Stack Ensemble (Collaborative Filtering + Factorization Machine + Text:Logistic + Text: Ensemble)	Text-mining -- Stack Ensemble(Factorization Machine + Text:Logistic + Text: Ensemble)
2.4158	1.4964	1.2511	0.85	0.86

a. According to the attached table that shows the MSE of each model and the final combined model, we realize that text-mining (Stack Ensemble: Collaborative Filtering + Factorization Machine + Text:Logistic + Text: Ensemble) gives the lowest MSE. However, thinking about the statistics side, we conclude that text-mining (Stack Ensemble: Factorization Machine + Text:Logistic + Text: Ensemble) is the best model among the five models.

b. MSE table

i. Collaborative filtering: 2.4158
ii. Factorization Machine: 1.4964
iii. Text-mining -- Logistic Regression: 1.2511
iv. Text-mining -- Stack Ensemble
    1. (Collaborative Filtering + Factorization Machine + Text:Logistic + Text: Ensemble) : 0.85
    2. (Factorization Machine + Text:Logistic + Text: Ensemble): 0.86

c. Potential improvements on the model

i. Adding in side information: There are many information that we have not yet make use of in the dataset such as the categories of the business, the type of the users, and the detailed addresses of the business. Since the dataset is too sparse and have many features already, we have only select part of them for the concern of the complexity. If available, we may perform feature engineering on all features, rule out those with no variance and zero coefficient factor , thus may further improve our model performance.
ii. Parameter tuning as whole: For now we have tuned the hyperparameter of each model separately for the concern of complexity also. If possible, we can tune the ensemble as a whole model with cross validation, thus the result will be more accurate and the performance can be optimal.
iii. Content-based model: We think it is also reasonable to add a content-based model into our ensemble since we have many info on the business and it is possible to build a preference profile for every user if the complexity allows.

11. Reference

a. Steffen Rendle (2012): *Factorization Machines with libFM*, in ACM Trans. Intell. Syst. Technol., 3(3), May.
b. "TensorFlow implementation of an arbitrary order Factorization Machine."*GitHub,* https://github.com/geffy/tffm

c. *Yelp Dataset*, https://www.yelp.com/dataset/documentation/main.
d. Yelp. "Yelp/Dataset-Examples." *GitHub*, 7 Nov. 2014, https://github.com/Yelp/dataset-examples.