Eat Easy: Final Report

By Arpit Vyas, Benjamin Tan, Rebecca Lysaght, and Yiqi Lin

Pitch:

Ever been spoilt for choice with food options? Hungry and want a quick recommendation? Eat Easy serves as a tool for you to seamlessly get a restaurant recommended to you. With the click of a few buttons, information is extracted from a database of users to suggest an ideal location for your next meal.

Underlying Problem:

Yelp, Tripadvisor and Zomato alike pull up a list of restaurants from their database to allow the user to decide which restaurants they would like to go to. The problem here is that the user is faced with the burden of choice even after keying in his or her preferences into these platforms. The problem that we have solved is to help create a seamless experience for the user to decide on a place to eat.

The Journey:

We spent the first few weeks researching previous projects involving the Yelp dataset in order to understand how to use this data. Originally we wanted to create a meal recommendation algorithm; however, we did not have the necessary menu data needed to do this. Because of this, we then decided to recommend restaurants to users instead. We wanted to create something new and worthwhile, so instead of recommending a list of restaurants, we modeled our design off of Tinder and recommended one restaurant that the user could then accept or decline. We relied heavily on the jupyter notebooks presented in class, as well as on online resources and academic articles.

We have made a conscious attempt to implement different parts into our final solution on our journey. Therefore, the solution that we have is an amalgamation of all of these individual parts. The description of our solution is detailed in the next section.

Solution:

After understanding our problem, we had to come up with a system where rankings for a specific user is established. This ranking will then allow us to recommend a specific restaurant to the user based on his or her preferences.

Get the data and preprocess it in order to understand how to use it:

First of all, a dataset had to be attained in order for any algorithm or tools to be tested. We did a search on the internet and decided that we will use the Yelp dataset to run our algorithms upon. We converted the .json files into .csv files and were able to parse the data in a Python notebook to be studied. And after analyzing the whole dataset, we decided to choose the most active top 1000 users and a specific state, say Arizona.

Try Matrix Factorization because it's the most popular recommendation method:

Initially, we explored using Matrix Factorization to output a recommendation for the specific user. After extensive research, Matrix Factorization is one of the most popular methods for recommendation models. In the Matrix Factorization method, the restaurant id versus user id matrix is decomposed into matrices that hold just information pertaining to the restaurants and the users. Using these decomposed matrices, a dot product will allow us to come up with a recommendation model for each user for each restaurant. However, we quickly found out there were other collaborative filtering methods and recommendation techniques that can better implement what we learned in class.

Add more value to our system by incorporating filters:

In order to make user-specific recommendations, we asked users to input certain filters such as mealtime and food category. We then wrote two functions, open_restaurants and categories, to remove any restaurants that did not conform to these filters. Open_restaurants does this by iterating through all restaurants, extracting their weekly hours, cleaning that string into a usable date/time format, comparing that date/time with the user inputted mealtime, and removing all restaurants that are not open during the given mealtime. Categories use the same process, except it extracts the categories of each restaurant in the dataset and only keeps those restaurants that belong to the user category.

Add more features to refine our recommendation model:

We add four features to get the predicted score for recommendation: the modified global star, the similarity between user and restaurants, the reciprocal of distance and average star rating. We do logistic regression based on the first two features and leave the last two features as the customized part.

The modified global star is calculated based on the idea of Collaborative filtering, the modified global star for user A should be the average star given by user A plus the stars given by the other users modified by the similarity between user A and other users. And the similarity between them is calculated by cosine distance from the vectors each user has.

In order to ensure that we attempt to implement what we have learned in class to deepen our understanding, we explored using Natural Language Processing(NLP) when calculating the similarity between the user A and the restaurants. We used NLP to run a sentiment analysis on the reviews given by user A and reviews received by the restaurants. We considered using Bag of Words to extract features of the user A and the restaurants, compare to get the similarity score.

After getting these first two features, we threw them into our logistic regression model and trained it. Actually, we also use other models like lasso regression, KNN, random forest, and logistic regression is the one fit most. After training, if the user gets to a new restaurant, we can just calculate the feature scores and get the prediction.

For the last two features, we let the user choose the weights, how importance the distance and the average star are to him because these two features really vary in different situations and for different users. In this way, we differentiate our system from other recommendation systems by involving the user into the design of the model himself.

Solve the Cold Start Problem:

The recommendation model we mentioned above is based on the information we have for the old users, in our solution, we also tried to deal with the Cold Start Problem. We introduced a similarity index which ranks how similar a new user is to an older user. The completely new user will enter preferred food type(s) and cuisine type(s) and he or she will be matched against the existing database of users. The similarity index will increase every time the existing users have visited a restaurant with the same type(s) or cuisine type(s). An existing user with the highest similarity index compared to the new user will help with the recommendation algorithm for a completely new user.

Use feedback to improve our model and make it more customized:

We take feedback from the user on "how comfortable were they traveling to the restaurant", "how useful did they find the global restaurant rating", and also we would ask them to insert their review of the place. Information on these shall help us populate the data matrix and also modify the weights given to various features in the model.

Further improvements:

We plan on obtaining the average price of the businesses to help further improve the algorithm by using price as a potential feature or filter.

Furthermore, the implementation of RNN will be able to store weights to different features from user feedback and make better predictions.

Lastly, clustering of the most relevant neighbours can be initially used before collaborative filtering is implemented for better accuracy.

Contribution of Each Team Member:

All the members of the team regularly partook in the weekly meeting held to discuss the project. We, as a team, brainstormed the approach to the problem and took time equally to visit the office hours and consult with the GSIs the progress of our work. The technical work related to data mining, data cleaning and machine learning algorithms was split across the members. While, Yiqi completed the data extraction, data conversion, NLP analysis, modified global star calculation (Cosine Distance), logistic regression, code for cold start problem, code for Matrix Factorization, code for category filter, all the function tests and code files for all the interfaces, Rebecca undertook implementing filters into the model and final presentation organization. Additionally, Ben was involved in Matrix Factorization and cold start problem, Arpit implemented the Geocode filter into the code and was also involved in the modified global star calculation.

Github link:

https://github.com/rebeccalysaght/DataXFinalProject/tree/master/final_all_codes/code%20files

Appendix

Description of all the code files

Python files for data preprocessing:

Python file 1: ‘convert_json_to_csv.py’
The data from yelp are huge json files, we use this python file to convert json files to csv files that we can process in python notebook or editors like pycharm.

Python file 2: ‘restaurants_AZ.py’
The ‘business.csv’ file contains all the merchants including non-restaurants like shopping mall or auto repair shops, so we use the file to choose restaurants from all the merchants. Since we want to make our recommendation more specific, we will let the user choose a specific state, say AZ, and make local recommendation system.

Python file 3: ‘user_top1000.py’
The data has millions of users, so it usually takes a long time for us to go through the whole user data file. Among them over 95% users only have less than 20 reviews, which will show much less accuracy when we do nlp analysis. Thus, we rank these users by their review accounts in ‘user.csv’ file, and choose the top 1000 users as sample to do further analysis. Actually, all the analysis and results can apply to other users, so we think it’s ok to narrow our user sample.

Python file 4: ‘review from user top1000.py’
According to the statement above, we choose top 1000 users as the sample, so correspondingly, we have to get their reviews. We just use this python file to get reviews of these ‘sample’ users.

Python file 5: ‘business distribution.py’
Use this file to visualize the business distribution in US based on business file. A map will be shown as the result:

Python files for Matrix Factorization:

Python file 6: ‘recommendation edition1.py’
Recommendation edition1 is based on the idea of Matrix Factorization. In this file, we will collect the stars each user give for each restaurant and get a very sparse matrix. Since stars can be identified as the preferences of users, by normalizing and decomposing this matrix, we can make recommendations for users.

Python files for filter building:

Python file 7: ‘categories_filter.py’
The user can choose the categories he/she likes to have for this meal, and our system will this file to filtrate restaurants meeting the category requirement.

Python file 8: ‘open_restaurants_filter.py’
The user can choose the time he/she likes to have for this meal, and our system will this file to filtrate restaurants meeting the time requirement.

Python file 9: ‘filter.py’
Combine the categories filter and open time filter. User can filtrate restaurants by choosing the categories and time he/she would like for this meal

Python files for recommendation model:

Recommendation model is divided into two parts: logistic regression part and customized part, for logistic regression part, we will include modified global star and NLP similarity score, for the customized part, we will include average star and reciprocal of distance.

NLP analysis:

Python file 10: ‘nlp_user.py’
In order to make recommendations, we want to get the features of users, more specifically, we want to get the features the user like. So for each user, we use this file to choose the restaurants he/she gives no less than 4 stars (hopefully have positive reviews), collect all the reviews the user gives for these restaurants and do NLP analysis.

Python file 11: ‘nlp_restaurant.py’
In order to make recommendations, we want to get the features of restaurants. We use this file to collect all the reviews the restaurant receives and do NLP analysis.

Python file 12: ‘nlp_similarity_calculation.py’
In order to make recommendations, we want to match the users and the restaurants, so we use this file to get the similarity between the features of each user and each restaurant based on the NLP analysis we have done.

Modified global star:

Python file 13:’modified_global_star.py’
The modified global star for user A to restaurants is calculated by the average of user A’s stars plus other users’ stars given to restaurants modified by the similarity between user A and other users. The similarity is calculated by cosine distance, and we use this file to get a matrix including modified global star from each user to each restaurant.

Logistic regression:

Python file 14:'regression including logistic.py'
After we get the data we need, we put the data into the model and train it. In the file, we tried logistic regression, perceptron, xgboost, SVM, KNN and random forest, finally, we found the best model is logistic regression model with an accuracy of about 50%. The accuracy is not very high because the regression model has 5 classes (star 1-5) but only 2 features, if we add more features like prices, the accuracy will improve. This will be our future work.

Distance calculation:

Python file 15: 'geo_feature_normalization.py'
We have the function that can get the distance of the user and the restaurants (Arpit has uploaded it). The distances we get are usually in a large number, in order to make it a feature for ranking, we use this file to normalize these distances.

Combination of recommendation

Python file 16:'recommendation_edition2.py'
In this file, we will go through the whole recommendation process including the regression part, and then for customized part, we will let the user decide the importance of the distance to them. Based on that, we will get the predicted scores of all the restaurants in AZ that the user hasn't been to, and make recommendations to him. And the weight for average score will be customized based on the feedback given by the user.

Cold Start Problem

For cold start problem, the idea is that, we will let the new user A to choose the categories he/she likes (including cuisine and food), and we will also get the preferred categories the each old user like. If the new user's preferred categories are very similar to one old user B, we will say A and B are similar users, and we will recommend A what we will recommend to B.

Python file 17: 'categories_AZ.py'
We use this file to let all the restaurants categories

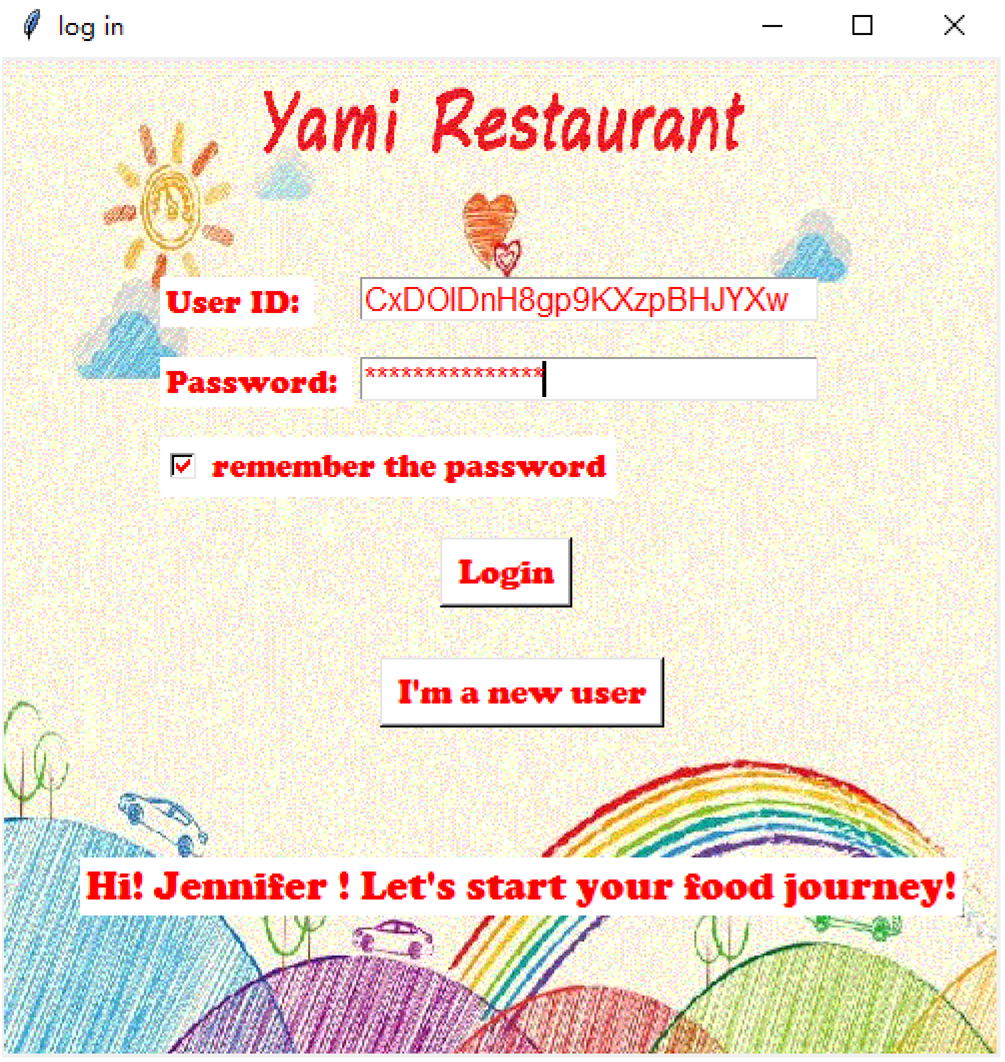
Python file 18: 'categories_user_like.py'
For each user, we will use this file to collect the categories of the restaurants receiving no less than 4 stars from him/her, and take them as his/her preferred categories

Python file 19: 'new_user.py'
In this file, we will let the new user input some of his/her information to sign up, then based on the information, find the similar user.

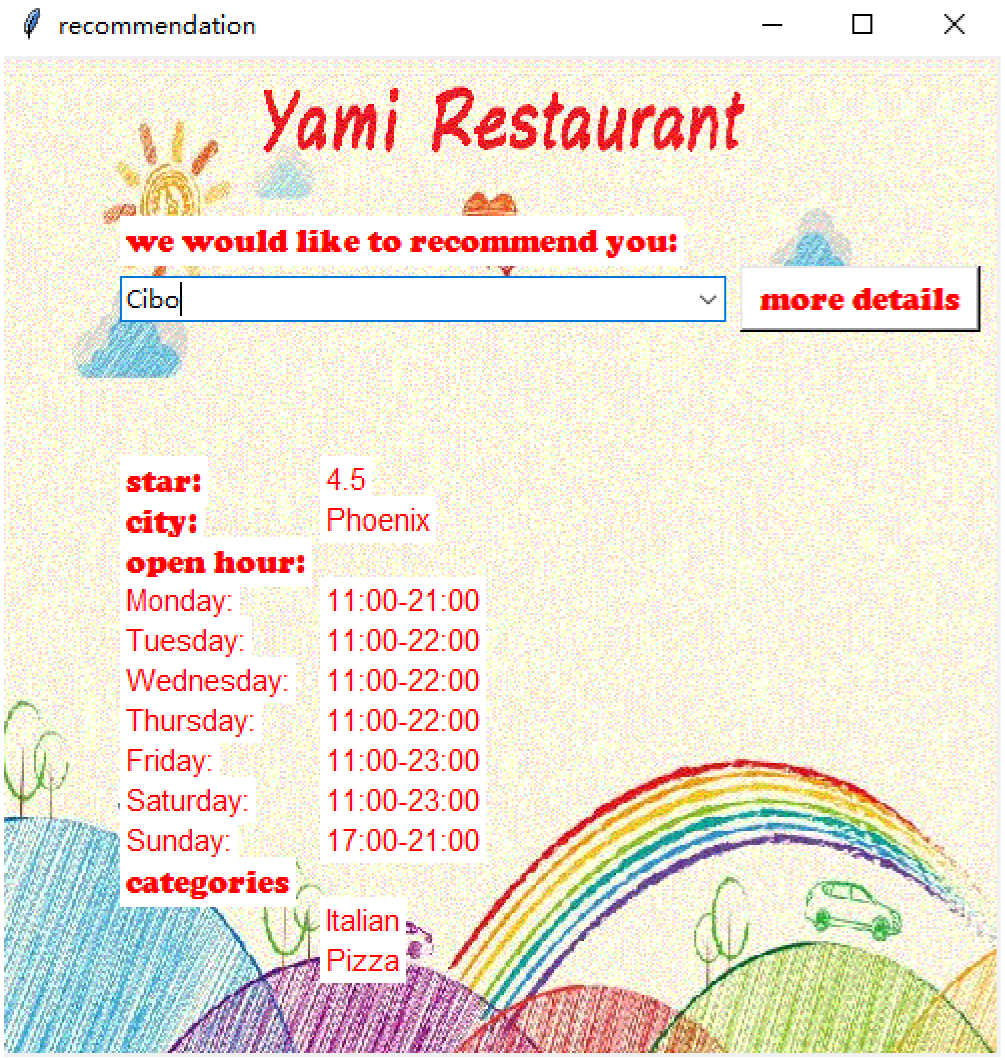
User Interface:

We use tkinter to build all the interfaces based on the code given above. (some codes are changed to fit the grammar of tkinter and some are optimized to get a shorter run time)

Python file 20: ‘login_interface_final.py’



Python file 21: ‘recommendation_edition1_interface_final.py’
The result shown in the interface is the result got from the initial Matrix Factorization



recommendation

Yami Restaurant

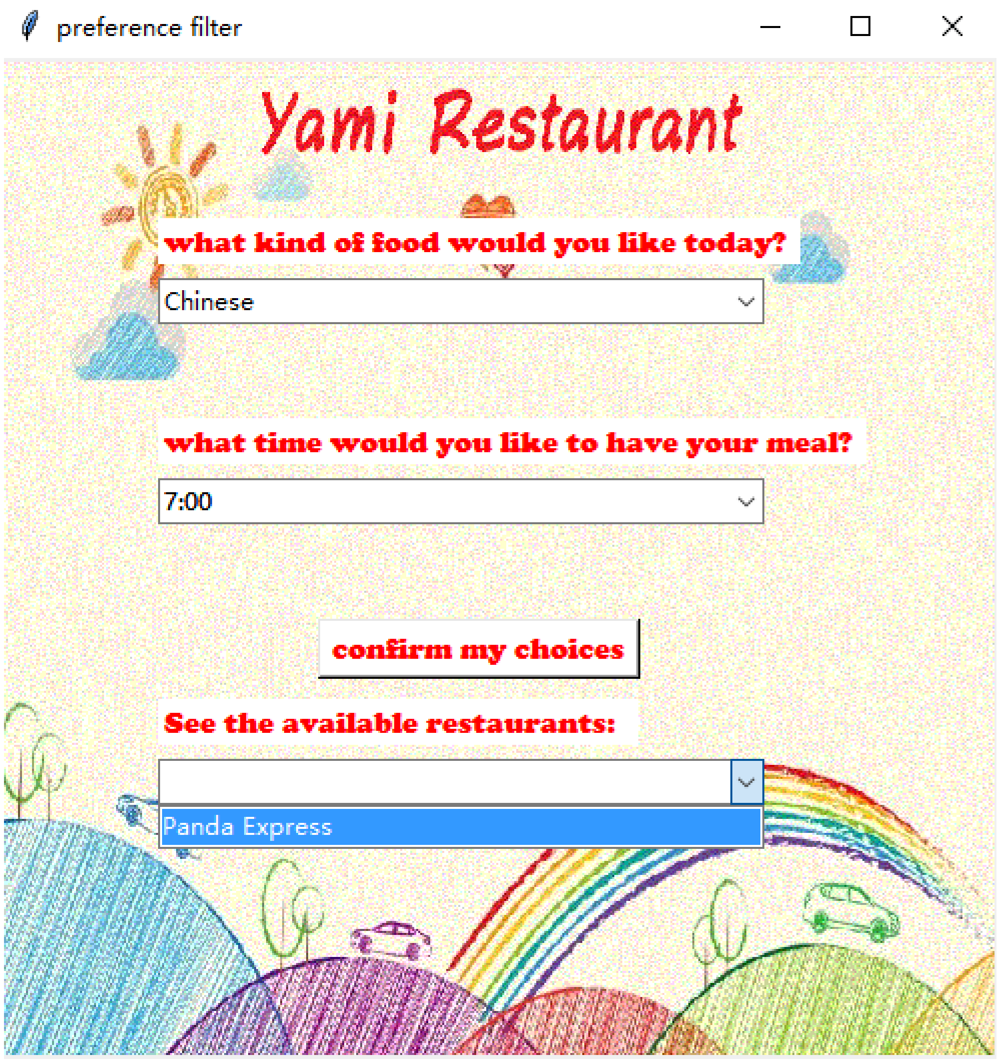
we would like to recommend you:

Cibo

more details

star:	4.5
city:	Phoenix
open hour:	
Monday:	11:00-21:00
Tuesday:	11:00-22:00
Wednesday:	11:00-22:00
Thursday:	11:00-22:00
Friday:	11:00-23:00
Saturday:	11:00-23:00
Sunday:	17:00-21:00
categories	
	Italian
	Pizza

Python file 22: ‘filter_interface_final.py’



preference filter

Yami Restaurant

what kind of food would you like today?

Chinese

what time would you like to have your meal?

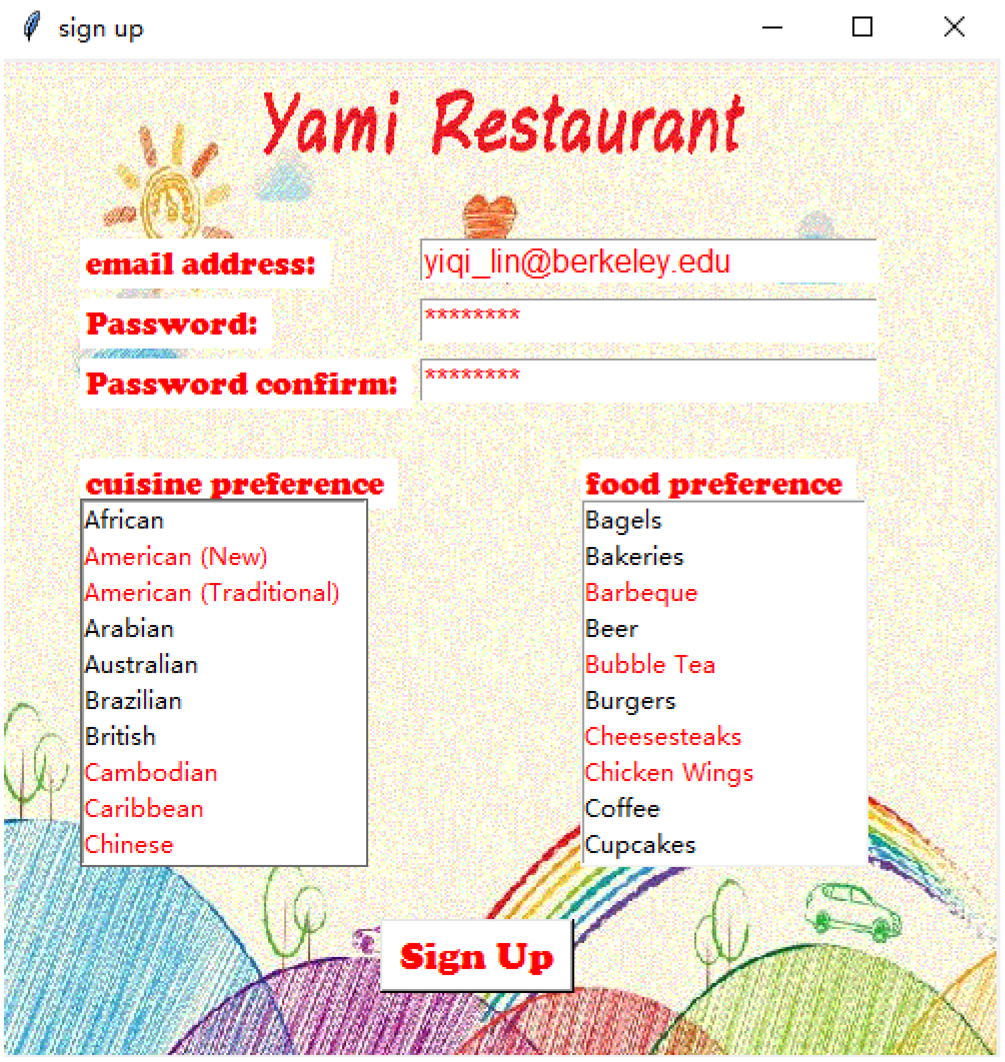
7:00

confirm my choices

See the available restaurants:

Panda Express

Python file 23: ‘new_user_interface.py’


sign up

Yami Restaurant

email address: yiqi_lin@berkeley.edu

Password: *********

Password confirm: *********

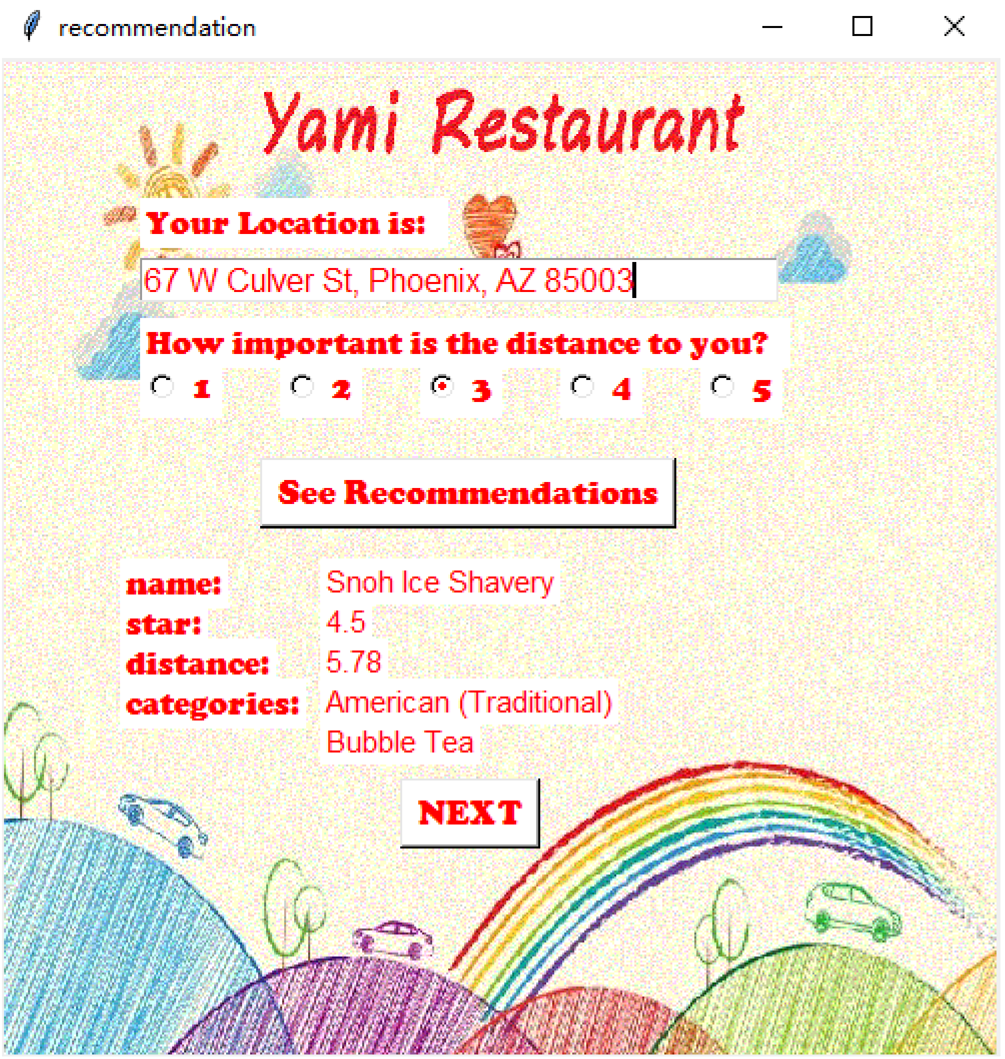
cuisine preference

African
American (New)
American (Traditional)
Arabian
Australian
Brazilian
British
Cambodian
Caribbean
Chinese

food preference

Bagels
Bakeries
Barbeque
Beer
Bubble Tea
Burgers
Cheesesteaks
Chicken Wings
Coffee
Cupcakes

Python file 24:'recommendation_edition2_interface_filter.py'
The result shown in the interface is the result got from the recommendation model including logistic regression and customized part

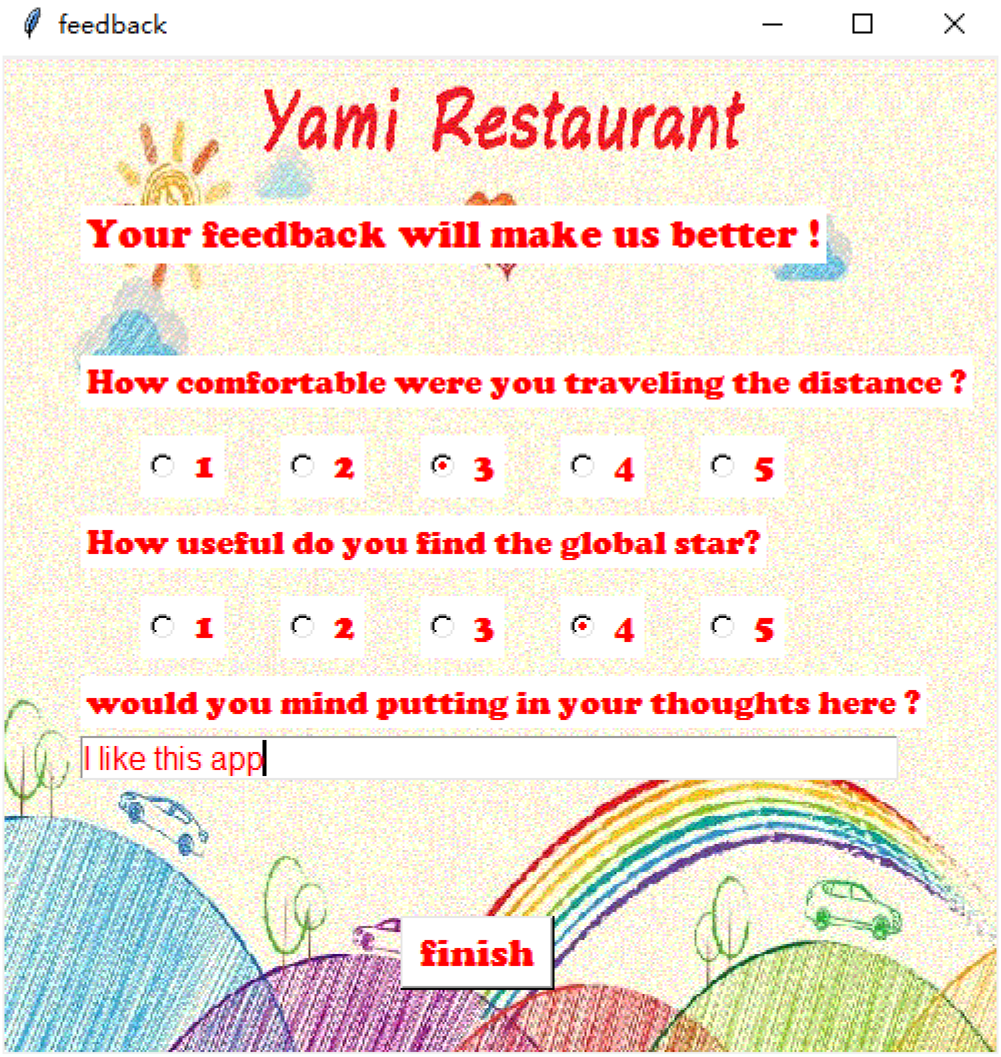


67 W Culver St, Phoenix, AZ 85003

○ 1 ○ 2 [✓ 3] ○ 4 ○ 5

name:	Snoh Ice Shavery
star:	4.5
distance:	5.78
categories:	American (Traditional)
	Bubble Tea

Python file 25: ‘feedback interface.py’



feedback

Yami Restaurant

Your feedback will make us better !

How comfortable were you traveling the distance ?

1 2 [✓ 3] 4 5

How useful do you find the global star?

1 2 3 [✓ 4] 5

would you mind putting in your thoughts here ?

I like this app