Milestone Report 2: Restaurant Collaborative Recommender System

Problem: Your client is a food delivery service or food review application that wants to predict what its users want to eat given a variety of factors. As a consumer, people often have a hard time choosing what restaurant to go to due to the many options to choose from. I am tasked with what kinds of foods people would prefer given their drinking habits, budget, ethnicity, etc. and recommend different restaurants or cuisines.

Audience: Consumers that have a hard time choosing what restaurants to go to. Potential food companies interested in this. Land developers looking to attract certain restaurateurs to open a restaurant in a particular area that has a certain demographic.

Dataset: https://data.world/uci/restaurant-consumer-data

Approach: Combine datasets, deal with any null values seen, use some kind of regression to predict what features impact people's decisions the most when choosing certain restaurants, and build a collaborative recommendation system that would recommend people certain restaurants.

Data Cleaning:

Loading Datasets: I loaded several datasets using pd.read_csv from the UCI Database. Upon inspecting, I noticed some ambiguous data as the Rcuisine column, which appeared both in user_cuisine and cuisine. Since there was no readme file or information for each of the datasets, I decided to omit the user_cuisine data, and use the cuisine dataset, user profile dataset, and ratings dataset.

Merging datasets: I decided to merge the datasets, first merging the user_profile and ratings datasets under the userID column. I then used the merged dataset and merged it with the cuisine dataset on placeID.

Dropping duplicates: Due to duplicate data in the dataset, I elected to drop them based on duplicates of userId and placeId, keeping the first instance of them.

Dealing with missing values: Checking through several of the columns, many of them have a ? value, which indicates an unanswered question the user had. In order to fill the question marks, I calculated the probabilities of each of the values and used

np.random.choice to assign a random value at the probability based on all the values. I wrote a function ‘fill_missing_values’ to do this for all of the columns that had missing values.

Feature engineering: To quantify the categorical variables, I changed the smoker, drink level, dress preference, etc. columns to numeric values under new columns. I then selected the newly converted numeric values as well as the other quantitative variables under dataframe df_cat. This was accomplished by creating the function cat_to_numeric, which created a new column in the DataFrame that had numeric values in place of the string values.

Exploratory Data Analysis and Inferential Statistics:

Data Visualizations: Using a heatmap, I checked for correlations between all of the variables. With the exception of food_ratings and service_ratings, none of the variables exhibited much correlation to rating. I also used a scatterplot with latitude and longitude differentiated by ratings and I found that many of the higher rated areas congregated in the same location. I drew histograms for the height and weight columns to see if they were normally distributed. While I

Checking collinearity: I checked collinearity between ratings with food ratings and ratings and service ratings to see which one contributed more to the overall rating. Food had a slight edge in this case.

Checking distributions: I wrote a function check_distributions, which would print a statement on whether or not a variable was normally distributed based on the p_value of the chi-square test. I found that weight and cuisine were not normally distributed, while height was.

Model Selection and Development:

RandomForestClassification: I tried both a RandomForestClassification and Logistic Regression to see how good the variables were at classifying the ratings. The Random Forest Classification model ended up with a R^2 of 0.558 and a RMSE of 0.826. I used GridSearchCV to tune the hyperparameters of the model and the test yielded a best score of about 0.581 with model parameters of a max depth of 8, 7 max_features and 11 n_estimators. The R^2 for the GridSearchCV after testing was 0.622 and a RMSE of 0.693, slightly better accuracy with less error..

RandomForestClassifier: Since the amount of variables I had were relatively high, the model was uninformative of which variables mattered. In order to find the most predictive ones I used SelectFromModel on a RandomForestClassifier to figure out which variables were most predictive. When running this model, it gave me the columns 'userID', 'latitude', 'longitude', 'weight', 'height', 'placeID', and 'Rcuisine_cat' as the most predictive. Using these 7 variables, I arrived at an R^2 score of 0.575 and a RMSE of 0.815. The tuned model had a best score of 0.595, with the parameters of a max depth of 8, 6 max_features, and 5 n_estimators. The tuned model predicted at an accuracy of 0.604 and a RMSE of 0.753, slightly better than the untuned model.

Recommendation System: In order to build the evaluate method, I used zip on userID and placeID and iterated through them to compare the error between them. The pearson method finds the pearson correlation coefficient between two series. The recommendation system is done with the class CollabPearsonReco which learns from a pivot table made from the ratings and takes the index as placeID and the columns as userID. The estimate function compares ratings from others by checking the Pearson correlation coefficient between other user profiles and evaluating the RMSE.

Findings: Based on the latitude, longitude graph, higher rated restaurants tend to congregate around the same location. This may be due to the clientele and what part of the city the restaurants are located in. Food ratings weighted very slightly better at predicting overall rating than service ratings. It was also possible to recommend certain restaurants, which the model had an RMSE of 0.719.

Possible biases: For the ratings, since they were only on a scale from 0-2, I assumed higher was better.

Future Developments: Building a Deep Learning model or possibly using more data to get better prediction. Also mapping out actual latitude and longitude on a map to see what geographic factors could have factored into a higher rating score.