Data Wrangling for Capstone 1

Data Collection

- We collected the data in csv from https://www.kaggle.com/shuyangli94/food-com-recipes-and-user-interactions
- We uploaded 'RAW_recipes.csv' and 'RAW_interactions.csv' files from into Pandas data frames

Irrelevant Data and Missing Values

- For the purposes of my Capstone 1 project we consider recipes with less than 10 reviews outliers. We explored the presence of recipes with less than 10 reviews by using the histogram plot from matplotlib and by filtering the interactions Pandas data frame by the number of recipes with interactions. The original interactions data frame had 210,244 recipes, but only 18,762 with more than 10 reviews. We updated the recipes and interactions data frames to only include recipes with over 10 reviews.
- Upon investigation of the recipes data frame it turned out that the description column had 406 empty values. We decided to keep the recipes with missing descriptions in the data for now.

Average rating per recipe

- We analyzed the average review rating and the number of users who reviewed by recipe in the resulting data frame. (The resulting code is in the Jupyter notebook attached)
- Most of the recipes have a review rating of 3.5 or higher

Recipe Tags

- We investigated the tags column of the recipes data frame. The tags column data in the recipes data frame was presented similar to the ingredients column: the individual values looked like a list, but were actually strings. There were 495 unique tags from all the recipes.
- We focused on creating a tags_matrix data frame with recipe ids in rows, all individual tags as columns, and values being 1 if the ingredient is present in the recipe and 0 if not. To do this, we performed the following steps:
    1) Added a column tags2 to the recipes data frame where the values from the tags column were converted to a list
    2) Added a column tags_check to the recipes data frame with a list of unique tags from all the recipes
    3) Created a function that run through all the values in the tags_check column and replaced the tag value with 1 if it is present in the recipe and with 0 if it is not present. Tested that the function is working on one recipe id.
    4) Applied the tags_check function to the column tags_check using list comprehension to get to the individual tags level
    5) Created the tags_matrix data frame with rows as recipe ids, columns as the individual recipe tags, and values as 1-s and 0-s for a particular recipe id-tag match from the tags_check column. We doublechecked that the tags_matrix has correct values by comparing the output of the tags_matrix for one recipe id to the sum of tag values from the tags_check column of the recipes dataframe for the same recipe id

Recipe Ingredients

- We investigated the ingredients column of the recipes dataframe. The data looked like a list, but the type of data was actually a string. We created a list with 8,091 unique ingredients from all the recipes. We found that this list has similar ingredients with different names. We decided not to modify the ingredients column itself and address the issue further by data wrangling the ingredients matrix.
- We narrowed down the list of the ingredients to only the ones that are present in more than a 100 recipes. There were 245 of such ingredients.
- The ingredients matrix was created similar to the tags_matrix. We then cleaned up the ingredients_matrix by getting rid of:
    1) Ingredients with names in multiple (e.g. 'eggs' vs 'egg')
    2) ingredient with the same first word in the name (e.g. 'garlic powder' vs 'garlic')
    3) ingredient with the same second word in the name (e.g. 'fresh cilantro' vs 'cilantro')
    4) ingredient with the same third word in the name (e.g. 'freshly ground pepper' vs 'pepper')
- We made sure that the values from the removed columns were correctly moved to the relevant ingredient column in the ingredients_matrix. E.g. if 'garlic powder' column had 1 and 'garlic' column had 0 for a particular recipe, then now the 'garlic' column would have a 1. The resulting ingredients_matrix had 149 columns.

Files for further use in Capstone 1

- As the result of the data wrangling we got several files for further analysis:
    1) Recipes_df
    2) Reviews_df
    3) Tags_matrix
    4) Ingredients_matrix
- We save these files in csv format to use in another Jupyter notebook for data story