SI 206 Final Project Report

Carolyn Wu, Christina Li, Joseph Choi

Goals

Our goals were to compare nutrition facts among various food groups, find out which cuisines are the most popular in Ann Arbor, and find out which city in Michigan has the top average ratings of restaurant for a certain cuisine.

Goals achieved

We grabbed data from the Nutritionix API, US Restaurant Menus API, and Yelp API. Then, we uploaded the data to a database with a table for data from each API. From the Nutritionix API, we grabbed data for 20 fruits, 20 vegetables, 20 carbs, 20 proteins, 10 dairy products, and 10 desserts. The data included the name of the food, total calories and total grams of fat for about a serving of each food. Then, we uploaded the data into a table in our SQL database so that the table would have three columns - one column was for the name of the food, another column was for total calories, and the third column was for total grams of fat. Then, we grabbed data from the database and calculated the average calories and average grams of fat for each food group. Finally, we used these calculations to create a scatter plot with average calories on the x-axis and average grams of fat on the y-axis.

Using the US Restaurant Menus API, we used latitude, longitude, and radius values to find restaurants in the Ann Arbor area. We collected the names and cuisines of 15-20 restaurants at a time, and repeated this process 6 times to find data for 100 restaurants. The data was written to a table in our SQL database in two columns- restaurant name and cuisine of that restaurant. After data was collected we counted the number of times each cuisine appeared in the database table, and visualized these counts using a plotly bar graph. The graph had cuisine type on the x-axis and number of restaurants on the y-axis.

For the Yelp API, we grabbed the Top 6 Korean restaurants for each of the Top 20 cities in Michigan by population. We used Beautiful Soup to grab ta list of the Top 20 cities in Michigan by population. We then used the Yelp API to grab a list of the top 6 Korean restaurants from these cities. We uploaded the data into a table with three columns in our SQL database.

populated the first column with the row number, the second column with the rating of that restaurant, and the third column with the city where it was located. We then grabbed data from the database to calculate the average rating for each city by looping through every six cities using a modulo function. Lastly, we used these calculations to create a bar graph with the average ratings on the y axis and the city name on the x axis.

Problems we faced

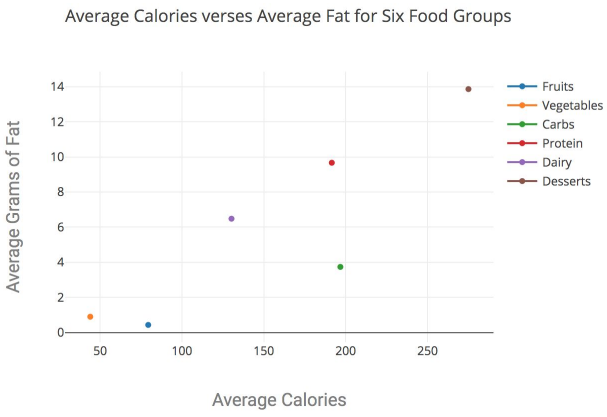
A major problem we faced was that some of our original ideas for the project did not include any calculations. For the Nutritionix API, we were originally going to graph every food on the scatterplot to see the full variation of calories and total fat that different foods have, but we decided to change the visualization to only have six points, representing the averages of calories and total fat for each food group.

The RapidAPI website (site where we found the Nutritionix and US Restaurant Menus APIs) recommended using the unirest python package to request data from the APIs, but we found that unirest only works with Python 2. We ended up using the requests library instead, and had to learn its new syntax.

Calculations file

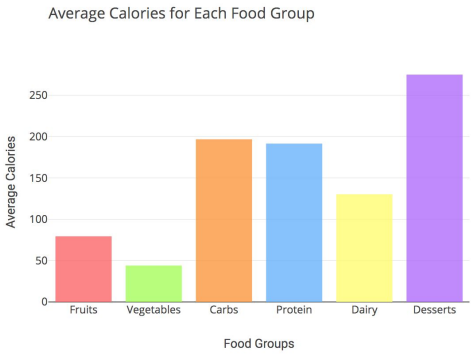
See files in folder: menus_calculations.json, NutritionData.txt, and yelp_calculations.json

Visualizations

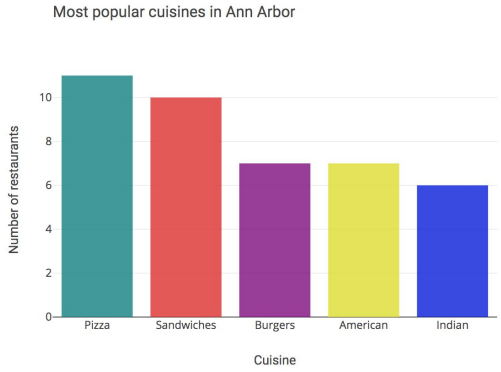


Nutritionix (scatterplot)

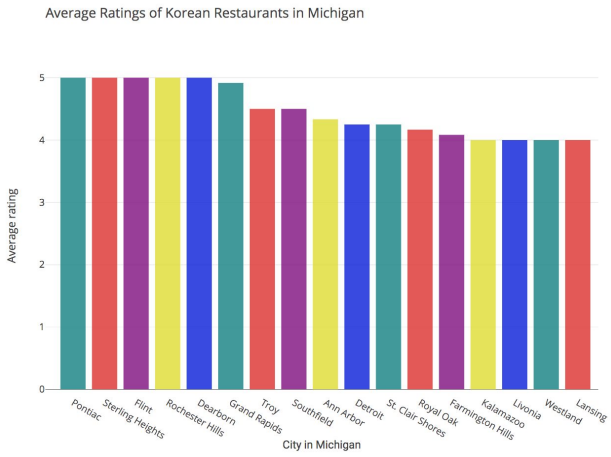
Nutritionix (bar chart)



US Restaurant Menus:



Yelp:



Instructions for running our code

1. Install python packages:
    a. requests, yelp, plotly, statistics, re, beautifulsoup4
2. Nutritionix: run nutritionix_data.py THEN nutritionix_visualization.py
3. US Restaurant Menus: run menus_main.py
4. Yelp: run main_yelp.py

5. If running these files fails create a .sqlite file, change the following lines of code to a valid path on your computer:
    a. menus_main.py line 11
    b. yelp.py line 42
    c. main_yelp.py line 7
    d. nutritionix_data.py line 45
    e. nutritionix_visualization.py line 137

Function documentation

Nutritionix:

def getData(list_items):

Input: A list of strings where each string is the name of a food

Output: A dictionary where the key is the name of the food from the input and the value is a list. The list contains data grabbed from the API, which is the name of the food as listed in the API, calories and total grams of fat.

Purpose: this function uses requests.get to insert the name of the food from list_items into the URL to grab data from the Nutritionix API.

def createTable(conn, cur):

Input: A connection to the SQL database and a cursor object

Output: No return value, but it creates a table titled Nutrition and when the program is rerun, it drops the table if it exists and creates a new table

Purpose: this function drops the Nutrition table if it already exists in the SQL database. Otherwise, it creates a table titled Nutrition in the SQL database so that the data can be uploaded into this table.

def setUpTable(dct, conn, cur):

Input: a dictionary that is the return value of getData function, a connection to the SQL database and a cursor object

Output: No return value

Purpose: It inserts data from the values in the dictionary into the database table so that the first column is the name of the food, the second column is the total calories, and the third column is the total grams of fat.

def calc_average_cals(list_tup, start_pos, end_pos)

Input: a list of tuples that contains data taken from the database for all 100 foods where each tuple represents a food and contains three strings (the name of the food, the total calories, and the total grams of fat), the start position of the list, and the end position of the list

Output: average calories for a food group

Purpose: loops through part of the list for a certain food group and calculates the average calories

def calc_average_fat(list_tup, start_pos, end_pos)

Input: a list of tuples that contains data taken from the database for all 100 foods where each tuple represents a food and contains three strings (the name of the food, the total calories, and the total grams of fat), the start position of the list, and the end position of the list

Output: average total grams of fat for a food group

Purpose: loops through part of the list for a certain food group and calculates the average total grams of fat

def write_file():

Input: none

Output: none

Purpose: write the calculations of average calories and average total grams of fat for each food group to a txt file

def create_scatter(data)

Input: takes in x and y values for six points to plot on a scatter plot using plotly where the x-value is the average calories for a food group and the y-value is the average grams of fat for a food group

Output: none

Purpose: uses plotly to create a scatter plot of average calories versus average grams of fat with six points where each point represents a food group

def create_bar(x, y)

Input: x is a list of the values for the x-axis and y is a list of average calories for each food group

Output: none

Purpose: uses plotly to create a bar graph where each food group appeared on the x-axis and the y-axis was the average calories for that food group

US Restaurant Menus:

def create_db(conn, cur):

Input: SQL connection object, SQL cursor object

Output: No return value

Purpose: Creates a table called Menus using the cursor object, with two columns: restaurant (text) and cuisine (text)

def collect_data(conn, cur):

Input: SQL connection object, SQL cursor object

Output: No return value

Purpose: Requests data from the US Restaurant Menus API. Collects data from 15-20 restaurants in Ann Arbor at a time, 6 times. Converts the result into a dictionary and inserts the values into the database.

def count_cuisines(cur):

Input: SQL cursor object

Output: Dictionary of each unique cuisine, and their counts

Purpose: Counts the number of times each unique cuisine appears in the database table. Also creates a json file and writes these counts to it.

def calculate_top_n(cuisines_dict, n):

Input: Dictionary of unique cuisines and their counts, desired number of top counts

Output: The top n cuisines by count, in decreasing order, in the form of a list of tuples

Purpose: Counts the number of times each unique cuisine appears in the database table. Also creates a json file and writes these counts to it.

def visualize(tuple_list):

Input: The top n cuisines by count, in decreasing order, in the form of a list of tuples

Output: No return value

Purpose: Uses plotly to create a bar graph of number of restaurants vs cuisine type, using the values of the list of tuples.

Yelp:

def getSoupObjFrom Urj(url):

Input: a website url in the form of a string

Output: soup object

Purpose: Takes a page of Michigan's Top 600 cities by population and creates a soup object

def get_cities(soup):

Input: soup object

Output: list of Top 600 Michigan cities

Purpose: web scrape data using div tags

def create_table(soup):

Input: soup

Output: Created SQL table named Yelp

Purpose: Create a list of 20 cities from the list of 600 cities returned by get_cities(soup). Create table with columns labeled rows, ratings, and location. Search Yelp for the Top 6 Korean restaurants in each location sorted by rating. Input data into table.

def calculate(conn, cur):

Input: conn, cur

Output: json file of dictionary of average rating for each city, and dictionary itself

Purpose: Create a dictionary of average ratings for each city by adding the sum and dividing by 6 every 6 cities

def create_tuple(dict_sum, n):

Input: dict_sum from the calculate function, number of cities we want to limit to

Output: sorted list of tuples of city, average rating

Purpose: Turn the dictionary of City: average rating to a tuple list of [(City, average rating)]

def visualize(tuple_list):

Input: tuple list of cities, average ratings that were returned by def create_tuple

Output: produce bar graph on plotly site

Purpose: Create a bar graph with 20 cities in the x axis and average ratings on y axis

Resource documentation

Date	**Issue Description**	**Location of Resource**	**Result (did it resolve the issue?)**
4/10	Looking for APIs	https://rapidapi.com/category/Food	Found food-themed APIs
4/15	Did not know how to use requests python library to authenticate	https://2.python-requests.org//en/master/user/authentication/	Used official documentation to authenticate request
4/17	Labeling plotly charts	https://plot.ly/python/figure-labels/	Learned how to set titles in plotly charts
4/19	Giving colors to plotly charts	https://www.december.com/html/spec/colorrgbadec.html	Found RGBA values