

Using Yelp to Study Ethnic Food Scene by U.S. Capital

Background:

Scope of Project:

In this project, I scraped Yelp restaurant data to investigate the food scene across different U.S capitals. This project helps us gain insight on geographic taste preferences through analyzing how different types of food are consumed across different places. This project compares the frequency of mexican, american, korean, japanese, italian, chinese, thai, mediterranean, indian, vietnamese, latin, french, filipino, greek, and hawaiian restaurants in the users home state. I created this code that ultimately prints a bar chart and pie chart to easily visualize this data. This code also recommends highly rated restaurant located in the user's home state.

More Information:

I chose Yelp, as opposed to other platforms such as - Foursquare, Zagat, or Google simply because is generally the go-to site for restaurant recommendations. Yelp's purpose is to connect people with great local businesses like restaurants, food, dentists, hair stylists and mechanics. Ever since Yelp was founded in 2004, it has received a monthly average of 33 million unique visitors who visited Yelp via the Yelp app and 69 million unique visitors who visited Yelp via mobile web (in Q4 2018). Yelp has over 1.2 million business attributes like hours, parking, availability, and ambiance. Furthermore, Yelp is free and contains the ability to filter by location which was vital for this project.

Technical Procedure:

Step 1: Installation and Importing The Necessary Libraries

Before getting started, you'll need to install the various libraries that I have used in this code. After you run all the installs, make sure the following commands work for you:

```
In [1]: # import necessary libraries
import requests as rq
import json
import pandas as pd # for data processing
import numpy as np
import urllib3 # for gathering url contents
urllib3.disable_warnings()
import seaborn as sns # provides the functionality to visualize Python histograms
import matplotlib.pyplot as plt
from pylab import savefig
```

Step 2: Choosing Our Target Capital to Analyze it's Ethnic Food Diversity

The only downside to using Yelp was that I could only scrape data on 2,000 businesses per endpoint. Due to this limitation, scrapping data of every restaurant located in a state would be impossible so this code finds the food scene within the capital of the user's home state. I chose to do this because more often than not the capital is the state's hub of restaurants and culture. Therefore, I created a dictionary linking US states and their capitals to ensure there would be no invalid inputs by the user. In addition, it allowed me to also select the capital corresponding to the state the user inputs.

```
In [2]: # create a dictionary linking US states and their capitals
# values should be all the U.S. capitals; capitals = state_capitals.values()
# keys should be all the U.S. states; state = state_capitals.keys()
state_capitals = {"Washington":"Olympia", "Oregon":"Salem", \
                  "California":"Sacramento", "Ohio":"Columbus", \
                  "Nebraska":"Lincoln", "Colorado":"Denver", \
                  "Michigan":"Lansing", "Massachusetts":"Boston", \
                  "Florida":"Tallahassee", "Texas":"Austin", \
                  "Oklahoma":"Oklahoma City", "Hawaii":"Honolulu", \
                  "Alaska":"Juneau", "Utah":"Salt Lake City", \
                  "New Mexico":"Santa Fe", "North Dakota":"Bismarck", \
                  "South Dakota":"Pierre", "West Virginia":"Charleston", \
                  "Virginia":"Richmond", "New Jersey":"Trenton", \
                  "Minnesota":"Saint Paul", "Illinois":"Springfield", \
                  "Indiana":"Indianapolis", "Kentucky":"Frankfort", \
                  "Tennessee":"Nashville", "Georgia":"Atlanta", \
                  "Alabama":"Montgomery", "Mississippi":"Jackson", \
                  "North Carolina":"Raleigh", "South Carolina":"Columbia", \
                  "Maine":"Augusta", "Vermont":"Montpelier", \
                  "New Hampshire":"Concord", "Connecticut":"Hartford", \
                  "Rhode Island":"Providence", "Wyoming":"Cheyenne", \
                  "Montana":"Helena", "Kansas":"Topeka", \
                  "Iowa":"Des Moines", "Pennsylvania":"Harrisburg", \
                  "Maryland":"Annapolis", "Missouri":"Jefferson City", \
                  "Arizona":"Phoenix", "Nevada":"Carson City", \
                  "New York":"Albany", "Wisconsin":"Madison", \
                  "Delaware":"Dover", "Idaho":"Boise", \
                  "Arkansas":"Little Rock", "Louisiana":"Baton Rouge"}

# Print the introductory statements:
print('Is your home state a diverse foodie mecca?')
print('I have set out to find the most diverse dining capitals in America. ')

# Ask for U.S. state and confirm that it's a valid location
# first: prompt the user for a state
# second: check that the user input is a valid state
# third: otherwise keep prompting until you get a valid input
while True:

    # prompt the user for a state
    state = input('Enter the name of your state: ').strip().title()

    # check that the user input is a valid string
    try:
        str(state)
        pass

    except ValueError:
        print("Sorry, I don't know that state, perhaps you mistyped?")
        break

    # pair the target state with its respective capital using the state_capitals dictionary
    if state in state_capitals.keys():
        target_location = (state_capitals[state] + ', ' + state)
        print('Nice! Lets find out if ' + target_location + ' is a diverse foodie mecca')
        break
    else:
        print("Sorry, I don't know that state, perhaps you mistyped?")
```

Is your home state a diverse foodie mecca?
I have set out to find the most diverse dining capitals in America.
Enter the name of your state: California
Nice! Lets find out if Sacramento, California is a diverse foodie mecca

Step 3: Giving the User Recommendations of Restaurants in our Target Capital

In order to scrape Yelp data, I used a Yelp Fusion API. An API (Application Programming Interface) is a collection of protocols and subroutines for building software. The Yelp fusion API gave me access to the best local content and user reviews from millions of businesses across 32 countries. The Yelp Fusion API uses private key authentication to authenticate all endpoints. All Yelp Fusion API endpoints are listed here: <https://api.yelp.com/v3>. I tried it out by using Postman, which you can also do yourself.

The following function recommends restaurants that have over 4 stars in the users specified home state. For this function, I am only interested in businesses that are categorized as "restaurants" and located in the users specified home state and rated over 4 stars. Therefore, I used the "term" parameter to filter out all businesses that are not restaurants. The "location" parameter, further filters the search results to encompass only restaurants located in the users home state. And lastly, the function checks that the business has not been (permanently) closed and is rated over 4 stars.

```
In [3]: # Grab restaurant information from Yelp
# The function by default has the search parameter: restaurants
def get_yelp_recgs(target_location, term = 'restaurant'):

    print('Here are some recommendations of highly rated restaurants in ' + target_lo
cation + ':')

    # Define API Key, Search Type, and header
    business_path = 'https://api.yelp.com/v3/businesses/search'

    headers = {'Authorization': 'Bearer anhkS-th_95DUa6hgmsSzpE1HSXjByUwtz9sk6onKNnaE
qMcKQvgx3VkBcoIEskqI3Dl_mxsCKPbzttnYJeuE1ULIYuwEmi2lPM70uSynsFd5j9Lo2EjjUVodI-XHYx'
}

    # Define the Parameters of the search
    params = {
        'term': 'restaurant',
        'location': target_location,
    }

    # Make a Request to the API, and return results
    # Convert response to a JSON String
    response = rq.get(business_path, params = params, headers = headers).json()

    for business in response['businesses']:
        rating = business['rating']
        restaurant_name = business['name']
        categories = business['categories']

        # Checks whether business has been (permanently) closed and over 4 stars
        is_closed = business['is_closed']
        if is_closed == False and rating > 4.0:
            print(restaurant_name + ' - ' + str(rating) + ' stars')
```

```
In [4]: # Run the function above to print restaurant recommendations
restaurant_scrapped_data = get_yelp_recgs(target_location)
```

Here are some recommendations of highly rated restaurants in Sacramento, Californi a:
Ramen House RyuJin - 4.5 stars
Localis - 4.5 stars
Don Chuy Birria Estilo Tijuana - 5.0 stars
The Kitchen Restaurant - 4.5 stars
Origami Asian Grill - 4.5 stars
Cacio - 4.5 stars
Cielito Lindo Mexican Gastronomy - 4.5 stars
Bacon & Butter - 4.5 stars
Woodlake Tavern - 4.5 stars
Hao Bao Dumpling House - 4.5 stars
Nash & Proper - 4.5 stars

Step 4: Finding the Number and Percentage of Restaurants in Each Ethnic Cuisine Type

There are a variety of available ethnic cuisines listed in the "categories" section of Yelp. The list of all Yelp's established categories can be found here: https://www.yelp.com/developers/documentation/v3/all_category_list. However, for this project, I'm only interested in comparing the amount of restaurants that are characterized as mexican, american, korean, japanese, italian, chinese, thai, mediterranean, indian, vietnamese, latin, french, filipino, greek, or hawaiian.

```
In [5]: # Function returns the maximum number of results that Yelp could return for a specif
ic search
def yelp_response_count(target_location, term = 'restaurant'):

    business_path = 'https://api.yelp.com/v3/businesses/search'

    headers = {'Authorization': 'Bearer anhkS-th_95DUa6hgmsSzpE1HSXjByUwtz9sk6onKNnaE
qMcKQvgx3VkBcoIEskqI3Dl_mxsCKPbzttnYJeuE1ULIYuwEmi2lPM70uSynsFd5j9Lo2EjjUVodI-XHYx'
}

    # Define the Parameters of the search
    params = {
        'term': term,
        'location': target_location,
    }

    # Make a Request to the API, and return results
    response = rq.get(business_path, params = params, headers = headers).json()

    for business in response['businesses']:

        # Checks whether business has been (permanently) closed
        is_closed = business['is_closed']
        if is_closed == False:

            # Total shows the maximum number of results that Yelp could return for t
hat specific search
            total_count = int(response['total'])
            return total_count
```

```
In [6]: # Define cuisines used to filter businesses
cuisines = ['mexican', 'american', 'korean', 'japanese', 'italian', 'chinese', 'thai', 'med
iterranean',
            'indian', 'vietnamese', 'latin', 'french', 'filipino', 'greek', 'hawaiian']

cuisine_count_list = []

for cuisine in cuisines:

    total_cat_count = yelp_response_count(target_location, term = 'restaurant')
    wanted_cat_count = yelp_response_count(target_location, cuisine)
    cuisine_count_list.append(wanted_cat_count)
    print('There is ' + str(wanted_cat_count) + ' ' + cuisine.title() + ' restaurant
s in ' + target_location + '.')
```

There is 1100 Mexican restaurants in Sacramento, California.
There is 1100 American restaurants in Sacramento, California.
There is 294 Korean restaurants in Sacramento, California.
There is 337 Japanese restaurants in Sacramento, California.
There is 766 Italian restaurants in Sacramento, California.
There is 477 Chinese restaurants in Sacramento, California.
There is 494 Thai restaurants in Sacramento, California.
There is 99 Mediterranean restaurants in Sacramento, California.
There is 132 Indian restaurants in Sacramento, California.
There is 297 Vietnamese restaurants in Sacramento, California.
There is 233 Latin restaurants in Sacramento, California.
There is 54 French restaurants in Sacramento, California.
There is 101 Filipino restaurants in Sacramento, California.
There is 144 Greek restaurants in Sacramento, California.
There is 221 Hawaiian restaurants in Sacramento, California.

```
In [7]: # Function calculates the sum of a list of numbers
# This function is used to calculate the sum of the count of restaurants of each cu
isine type
def list_sum(list):
    if len(list) == 1:
        return list[0]
    else:
        return list[0] + list_sum(list[1:])

# Function used to calculate percentages
# This function is used to calculate the percentage of each cuisine type for plottin
g as a pie chart
def percentage(part, total):
    percent = round(100 * (part / total), 2)
    return percent
```

```
In [8]: # Prints the total number of restaurants in the area
total = list_sum(cuisine_count_list)
print('There is over ' + str(total) + ' restaurants in ' + target_location + '.')
print()

# Calculate percentages by referencing the percentage function above
percentages = []
for count in cuisine_count_list:
    percent = percentage(part = count, total = total)
    percentages.append(percent)

# Get the list of tuples from two lists
# Merge them by using zip()
# Make dataframe to store data
data = list(zip(cuisines, cuisine_count_list, percentages))
df = pd.DataFrame(data, columns = ['Cuisine Type', 'Number of Restaurants', 'Percenta
ge'])
df.replace('None', 0)
print(df)
```

There is over 5849 restaurants in Sacramento, California.

	Cuisine Type	Number of Restaurants	Percentage
0	mexican	1100	18.81
1	american	1100	18.81
2	korean	294	5.03
3	japanese	337	5.76
4	italian	766	13.10
5	chinese	477	8.16
6	thai	494	8.45
7	mediterranean	99	1.69
8	indian	132	2.26
9	vietnamese	297	5.08
10	latin	233	3.98
11	french	54	0.92
12	filipino	101	1.73
13	greek	144	2.46
14	hawaiian	221	3.78

Step 5: Visualizing the Data With a Bar Plot and Pie Chart

Chart 1: Bar Plot of Cuisine Diversity

To visualize this set of data, this bar plot gives a the user a better sense for the popularity of each cuisine type in respect to one another.

```
In [9]: # Plot dataframe to bar plot
# Set the style
sns.set(style = "whitegrid")

# Create the plot
fig1, ax1 = plt.subplots(figsize=(10,10))
sns.barplot(y = cuisines, x = df['Number of Restaurants'])

# Labels and clean up on the plot
plt.title(('Cuisine Distribution in ' + target_location), fontsize = 24)
plt.ylabel('Cuisine Types', fontsize = 18)
plt.xlabel('Number of Restaurants', fontsize = 18)
plt.tick_params(labelsize = 12)

# Saving your Figures
plt.savefig('cuisine_diversity_barplot.png', format = 'png', dpi = 400, auto_open =
True)
```

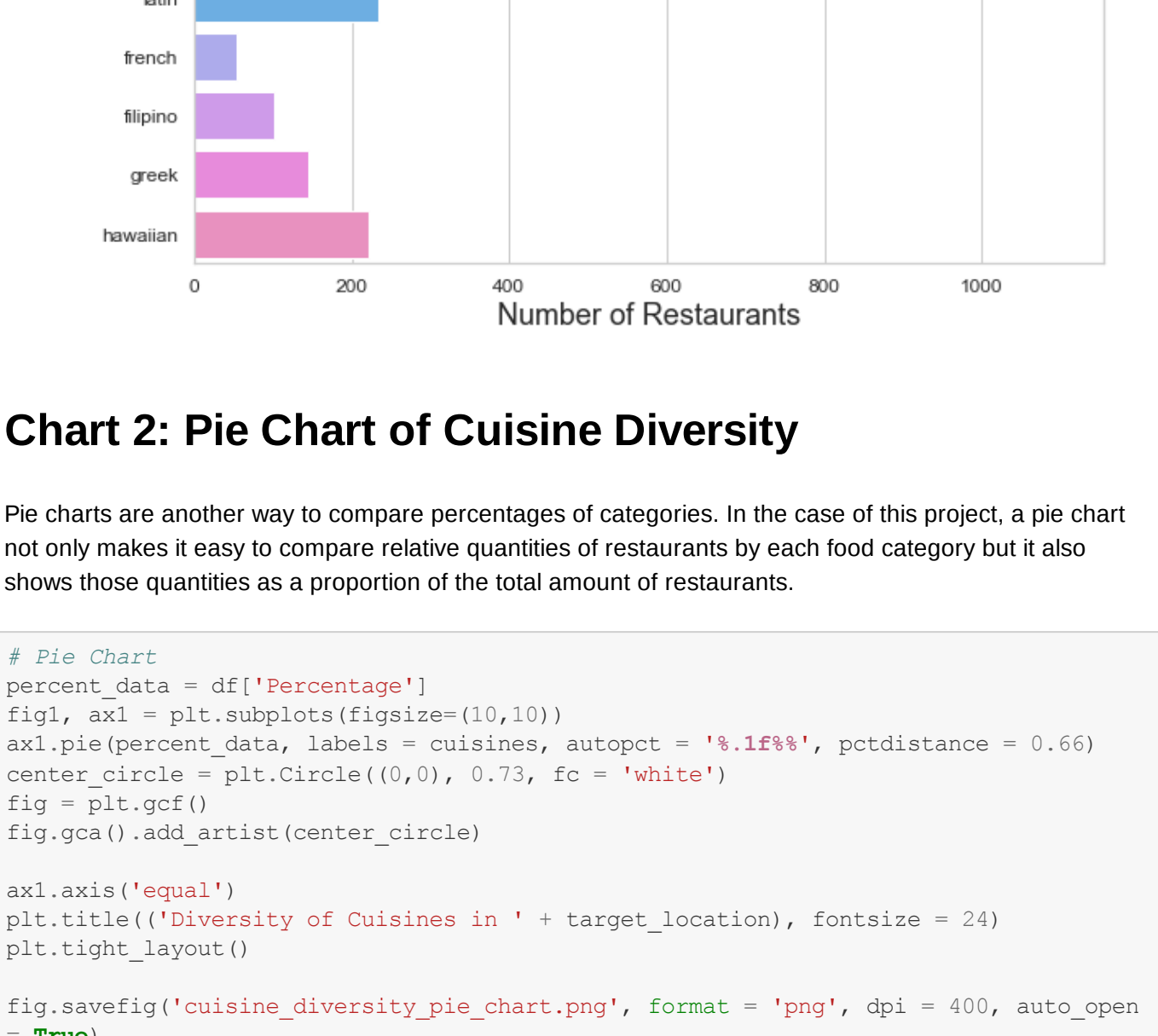


Chart 2: Pie Chart of Cuisine Diversity

Pie charts are another way to compare percentages of categories. In the case of this project, a pie chart not only makes it easy to compare relative quantities of restaurants by each food category but it also shows those quantities as a proportion of the total amount of restaurants.

```
In [10]: # Pie Chart
percent_data = df['Percentage']
fig1, ax1 = plt.subplots(figsize=(10,10))
ax1.pie(percent_data, labels = cuisines, autopct = '%.1f%%', pctdistance = 0.66)
center_circle = plt.Circle((0,0), 0.73, fc = 'white')
fig = plt.gcf()
fig.gca().add_artist(center_circle)

ax1.axis('equal')
plt.title(('Diversity of Cuisines in ' + target_location), fontsize = 24)
plt.tight_layout()

fig.savefig('cuisine_diversity_pie_chart.png', format = 'png', dpi = 400, auto_open =
True)
```



Learning Something New:

In my code I incorporated the Yelp Fusion API, which I explained above. Furthermore, I ran the API in Postman.